HAL
open science

Predicting the Performance and the Power Consumption of MPI Applications With SimGrid

Franz C. Heinrich, Alexandra Carpen-Amarie, Augustin Degomme, Sascha Hunold, Arnaud Legrand, Anne-Cécile Orgerie, Martin Quinson

To cite this version:

Franz C. Heinrich, Alexandra Carpen-Amarie, Augustin Degomme, Sascha Hunold, Arnaud Legrand, et al.. Predicting the Performance and the Power Consumption of MPI Applications With SimGrid. 2017. hal-01446134

HAL Id: hal-01446134

https://inria.hal.science/hal-01446134

Preprint submitted on 25 Jan 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Predicting the Performance and the Power Consumption of MPI Applications With SimGrid

Franz C. Heinrich[1], Alexandra Carpen-Amarie[2], Augustin Degomme[1], Sascha Hunold[2], Arnaud Legrand[1], Anne-Cécile Orgerie[3], Martin Quinson[3]

1: CNRS/Inria/Univ. Grenoble Alpes, France
Email: firstname.lastname@imag.fr

2: Univ. of Technology Wien, Austria
Email: {carpenamarie,hunold}@par.tuwien.ac.at

3: CNRS/Inria/ENS Rennes, France
Email: firstname.lastname@irisa.fr

ABSTRACT

The past decade witnessed a rapid development of powerful but energy-hungry parallel and distributed systems, making energy efficiency of large data centers an important optimization goal. Simulation is a popular approach for studying the behavior of HPC applications in a variety of scenarios. However, simulators are infrequently able to provide faithful performance predictions of applications and typically lack the capability of providing details about the energy consumption of the simulated platforms, especially when comprised of multi-core architectures. Furthermore, studying the impact of different application configurations on energy consumption is a difficult task as only few platforms are equipped with proper power measurement devices. In this paper, we present an extension of the SimGrid simulation toolkit that addresses these challenges. We firstly introduce a model for application energy consumption that supports dynamic voltage/frequency scaling (DVFS) of simulated processors. Secondly, we discuss means to account for coarse-grain memory effects in multi-core architectures. The advantages of our approach, compared to cycle-level simulators, are faster simulation run times and enhanced scalability with, provided the target platform is correctly modeled, a retained excellent accuracy. We discuss our model in detail and demonstrate how it can be instantiated by profiling different applications during the calibration phase. Finally, the proposed simulator is validated through an extensive set of experiments with common HPC benchmarks.

1. INTRODUCTION

The exhaustion of Moore's Law leads to construction of high-performance computing (HPC) platforms with a rapidly growing number of processors to ensure further performance increases. For instance, the system ranked as number one by the November 2016 issue of the Top500 list contains 10,649,600 cores (Sunway TaihuLight), 80 times more than the number one 10 years ago (BlueGene/L with 131,072 cores in November 2006) [56].

Alas, this proliferation of multi-core processors, HPC infrastructures is insufficient to mitigate the unprecedented energy consumption levels. Indeed, the energy efficiency of devices does not scale with integration capacity [18]. This level of energy consumption is not only concerning from a social and environmental point of view but is also jeopardizing the exponential performance scaling that we have observed over the last three decades. In fact, the transistor power reduction does not reach the same rate as the transistor area reduction, resulting in *dark silicon*, i.e., underutilization of the device integration capacity [18].

Consequently, energy-efficient techniques will have to play a more prominent role in future HPC infrastructures. An example of such techniques is Dynamic Frequency and Voltage Scaling (DVFS) that reduces processor frequency and voltage during low workload periods [54]. Yet, properly studying the impact of changing an application's configuration on energy is a difficult task, as only few platforms are currently equipped with proper power measurement devices.

Simulation is already a common approach for analyzing HPC applications in a variety of scenarios as well as for determining the quality of proposed improvements. However, such HPC simulators currently lack the capability to provide details about the energy consumption of simulated platforms and applications. In this paper, we explain how we extended the SimGrid open-source simulation toolkit by introducing a model of application energy consumption that allows SimGrid to account for dynamic voltage/frequency scaling (DVFS) of the simulated multi-core processors.

Since the relevance of energy consumption is particularly conditioned by the quality of time prediction, we first focus on explaining how to get accurate simulations in terms of performance for multi-core architectures with HPC applications. After a careful analysis of the experimental factors influencing both energy and performance, we provide a model and a calibration methodology to achieve meaningful simulations. Our experimental results using typical HPC benchmarks show a difference of only few percents in power consumption with our simulator compared to measurements on a real platform.

The contributions of this work include:

- We formalize a model of server performance and power consumption that supports dynamic voltage/frequency scaling (DVFS) of the simulated multi-core processors;
- We identify several bias incurred by the emulation of applications and propose a way to account for coarse-grain memory effects in multi-core architectures;
- We extend the SimGrid simulation toolkit by implementing the proposed model of application energy consumption with DVFS;
- We present a list of the key factors influencing the energy consumption and that should be carefully controled or monitored. From our experience, it is essential to rely on such methodology to accurately calibrate the models of the nodes of the platform;

- We present experimental results on several nodes using HPC benchmarks, demonstrating the accuracy of our models under the provided methodology.

The remainder of this article is organized as follows. Section 2 discusses related work. An overview of the SimGrid toolkit is provided in Section 3. Section 4 presents the proposed multicore and energy models and their implementation within SimGrid. Section 5 exposes our experimental setup. The experimental model verification and calibration is shown in Section 6. Experimental validation results are provided in Section 7. A discussion about the lessons learnt is provided in Section 8 and in particular, we present the methodology we used to identify and control the key factors influencing the energy consumption and the performance of the applications. Section 9 concludes this article and discusses limitations and future work.

2. RELATED WORK

2.1 Energy models for computing servers

In contemporary HPC nodes, processors are responsible for the lion's share of the energy consumption [41] as modern processors consist of several billion transistors [14]. The frequency of a CPU greatly influences its power consumption [19]: the lower the frequency, the less energy consuming but also the less productive the CPU becomes.

Efforts to design generic power models for processors have resulted in the realization that power consumption is not a linear function of the utilization in the general case, due to the intricacy of prevalent processor architectures and the heterogeneity of their utilization [42]. It means that one linear power model cannot fit all kinds of applications. However, the relation between a CPU's power consumption and load (CPU utilization) is linear for a given application at a given frequency as explained in [41] and shown in Section 6.

Power models traditionally break the power consumption of nodes into two parts: a static part that represents the consumption when the node is powered-on but idle; and the dynamic part that relates to the server's utilization [14]. The static part can represent a significant percentage of the maximum power consumption: approximately 50% for the servers used in Section 6. For this reason, turning off servers during idle periods can save significant amounts of energy [34].

For HPC servers with few idle periods, Dynamic Voltage Frequency Scaling (DVFS) constitutes a favorable alternative to help save energy. DVFS adapts the processor frequency according to the application workload and, for instance, decreases the frequency during communication phases [33]. Such frequency scaling strategies usually assume that performance loss is linear in the decrease of the frequency [31] and that power consumption is a quadratic function of frequency [57] although Han *et al.* showed that this relation is not perfectly quadratic [26].

The network is considered negligible for energy consumption of HPC servers as it typically reaches only 2% of the overall server's consumption [14] and it does not exhibit large variations related to traffic [41]. Memory is an important factor, accountable for 20-30% of the consumption of HPC nodes [14]. Alas, its power usage displays little variability and is hard to measure [23]. This could change in the future with an increase of memory and secondary storage utilization. These aforementioned factors (network, memory and storage), are typically accounted for in the static part of a server's power consumption [41]. Last, although a general model of power consumption of HPC servers may be designed, applying it to predict would require a careful instantiation as even seemingly homogeneous clusters may exhibit inter-node variability that can have significant impact on the power consumption [13, 36].

2.2 Techniques to save energy

Several approaches have been studied to tackle energy issues in HPC systems. The Mont-Blanc project explores an HPC system architecture based on application-specific system-on-chip [46]. They envision 40% of energy savings with such an architecture compared to a classical one.

Lin *et al.* investigate the potential energy savings by dynamically turning off servers during periods of low load [34]. To validate their approach, they perform case studies using two workload traces coming from Hotmail, a large email service, and from a Microsoft Research storage system.

Schöne *et al.* examine low-power states implemented in processors (C-States), which are power saving states entered by idling processors [49]. Their validation, conducted through real measurements on three different processors, shows that the wake-up latencies for re-establishing full performance can be significant ($100\mu s$ in the deeper state).

Savoie *et al.* propose to shift unused power from applications in I/O phases to applications in computation phases through explicit staggering techniques[47]. To evaluate their proposal, they design a simulator (PowerShifter) that simulates applications executing on a power-constrained HPC system. This simulator has been validated against a cluster of 1,296 Intel Xeons running three different HPC applications but it does not seem to be publicly available.

Peraza *et al.* consider adapting the CPU frequency depending on the application phase using DVFS [44]. They detect application phases through static loop analysis and execution traces. The evaluation of their proposed solution is conducted on 1024 cores of a supercomputer. Tsafack *et al.* introduce an energy-efficient, application-agnostic framework that performs on-line analysis of an HPC system in order to identify application execution patterns without a priori information of their workload [12]. This solution uses DVFS techniques with a dynamic governor switching between frequencies. The validation involves a 34-node cluster, equipped with per-node powermeters, running the NAS Parallel Benchmarks and other real-life HPC applications.

Recently, a new power saving technique has started to draw attention: software controlled clock modulation mechanism [48]. The latency for enabling and disabling clock modulation is comparable to DVFS latency (even better for some architectures) but its main advantage lies in its per-processor-core granularity, thus making it potentially more opportunistic than DVFS itself.

All these energy-efficient techniques remain difficult to study, compare and extrapolate at large-scale. This is why many studies (e.g., Freeh *et al.* [20] for classical MPI applications) rely on analytical models or on simulation tools, which makes sense provided such tools have been validated.

2.3 Cloud and HPC simulators

Energy optimization is a primary concern when operating a data center, which is why many simulators have been designed with a cloud context in mind and embed (or have been

extended with) a power consumption model [43, 55]. For example Guérout *et al.* [24] extend cloudsim [9] with DVFS models to study cloud management strategies while GreenCloud [32] is an extension of the NS2 simulator for energy-aware networking in cloud infrastructure. DCSim [55] is a simulation tool specifically designed to evaluate dynamic virtualized resource management strategies. Núñez *et al.* develop IcanCloud, which relies on cycle-based low-level architectural models of the CPUs and of the memory, and use it to predict the performance of MPI applications by [40].

However, as explained in [58], some of these tools have not been validated or are known to suffer from severe flaws in their communication models, rendering them unfit for an HPC application-centric context. Although those using packet-level and cycle-level models are arguably realistic (provided they are correctly instantiated and used, which can be intricate [39]), they suffer from severe scalability issues that also make them unsuited for our context.

Several simulators have been proposed to study the performance of MPI applications on complex platforms (e.g., [5, 59, 10, 27, 30, 17, 37]). Most of these tools are designed to study or to extrapolate the performance of such applications at scale when changing network parameters (e.g., bandwidth, topology, noise) but surprisingly, few of them embed a sound model of multi-core architecture. A notable exception is Dimemas [5] which implements a network model that allows to clearly discriminate between communications within a node (going through shared memory) and communications that go through the network. The PMAC framework [51] also proposes a rather elaborate model of cache hierarchy and can be combined with Dimemas to provide predictions of complex applications at scale. However, both tools rely on application traces, which limits their use to relatively static applications whereas it is more and more common to have MPI applications that dynamically adapt to the platform and to the load by heavily using non blocking and opportunistic communications. Finally, to the best of our knowledge, all of these tools neither embed a power model nor allow researchers to study energy related policies.

3. SIMGRID OVERVIEW

SimGrid, an open-source simulation toolkit initially designed for distributed systems simulation [10], has been extended with the SMPI module to study the performance of MPI applications [7]. SMPI implements the MPI-2 standard (and a subset of the MPI-3 standard) and allows users to execute unmodified MPI applications directly on top of SimGrid. Such an approach is key when studying applications whose control flow depends on the platform characteristics, a property that is becoming more and more common. This may not be required when investigating simple and mostly regular applications, in which case, a classical trace replay mechanism can be used. Both approaches are implemented within the same framework, which allows to use a mixture of both approaches whenever needed (e.g., emulating some parts of the application and injecting simulated delays in regular parts of the application to speed up the simulation).

The emulated MPI runtime provided by SMPI implements all the specific collective communication algorithms from several real MPI implementations (OpenMPI, MPICH, ...) and their selection logic. SMPI can hence account for performance variation based on the algorithm used for collective communications, allowing researchers to investigate a multitude of environments and configurations.

Finally, the network models are implemented using a flow-level approach that scales better than packet-level simulations, but still faithfully accounts for both topology and contention as well as for several non-trivial phenomena (e.g., RTT-unfairness of TCP or cross-traffic interferences [58]).

Most efforts over the last years have been devoted to compare simulation predictions with real experiments to validate the approach and to improve the quality of network and application models. Since the relevance of power consumption is particularly conditioned by the quality of time prediction, it is only recently, after the SMPI framework has been validated with many different use cases, that we have been able to invest in power models and an API to control them.

4. MODELING MULTI-CORE ARCHITECTURE AND ENERGY CONSUMPTION

4.1 Modeling Computation

In this Section, we explain two flaws of the SMPI approach that were particularly problematic when handling multi-core architectures and which we had to overcome to obtain accurate predictions.

1. SimGrid relies on a sequential (but fast) discrete-event simulation kernel that controls when each process should be executed and ensures they all run in mutual exclusion between two MPI calls. When MPI applications are emulated with SMPI, each MPI rank is mapped onto a thread and folded within a single UNIX process, which raises semantic issues and requires to privatize global variables. This is done by making a copy of the `data` segment for each rank and by leveraging the virtual memory mechanism of the operating system to `mmap` this `data` segment every time we context-switch from one rank to another. Since ranks run in mutual exclusion, the time elapsed between two MPI calls can be measured and dynamically injected in the simulator. If the architecture on which the simulation is run is similar to the target architecture, we generally expect that such time is a good approximation of what would be obtained when running in a real environment.

2. In SimGrid, computing resources are modeled by a capacity (in FLOP/s) and are fairly shared between the processes at any point in time.[1] Hence, when p processes run on a CPU comprising n cores of capacity C, if $p \leq n$, each process progresses at rate C while if $p > n$, each process progresses at rate Cn/p. Although such a model is a reasonable approximation for identical CPU bound processes, it can be wildly inadequate for more complex processes. In particular, when several processes run on different cores of the same node, they often contend on the cache hierarchy or on the memory bus even without explicitly communicating. It is thus essential to account for the potential slowdown that the computations of the MPI ranks may inflict on each others.

This combination of dynamic computation time measurement and of a simplistic computation model can lead to particularly inaccurate estimations. Let us consider on the one

[1] Note that the same property also holds for network links, which are fairly shared between flows.

hand a target application consisting of many small computation blocks heavily exploiting the L1 cache and interspersed with frequent calls to MPI (for example to ensure communication progress). Each MPI call would result in injecting the duration of the preceding computation in the simulator and immediately yielding to another rank. Despite all the care we took in implementing efficient and lightweight context switches, the content of the L1 cache will be cold for the new rank and its performance will therefore be much lower than the ones he would have had if it was running on its dedicated core (i.e., with dedicated L1 cache). Our emulation may therefore be biased and result in a significant apparent slow down for such applications.

On the other hand, let us consider a target application consisting of relatively coarse grain computation blocks which shall be considered to be memory-bound, i.e., that contend on L3 or on the memory bus when using all the cores of the machine. Since we measure each computation in mutual exclusion, during the simulation each rank benefits from an exclusive access to the L3 cache and the computation times injected in the simulation will thus be very optimistic compared to what they would have been in a normal execution.

Evidently, real HPC codes comprise both kinds of situations and knowing beforehand whether a given code region will be sped up or slowed down during the emulation compared to the real execution is very difficult as it is dependent on both the memory access pattern and the memory hierarchy. As we will explain in Section 6.2, we simply determine such a speed ratio per code region on small workloads and apply the correction dynamically when emulating the application.

4.2 Modeling Energy Consumption

As it is commonly accepted [41], power consumption breaks into two parts: a static part, which represents the consumption when the server is on but idle; and the dynamic part, which is linear with the server utilization and depends on the nature of computational workload (e.g., computation vs. memory intensive, provided such characterization can be done). Therefore, for a given machine i, frequency f, computational workload w, and a given usage u (in percentage), the instantaneous power consumption is:

$$P_{i,f,w}(u) = P_{i,f}^{\text{static}} + P_{i,f,w}^{\text{dynamic}} \times u$$

As we will see in further experiments, it appears that we can generally assume that $P_{i,f}^{\text{static}} = P_i^{\text{static}}$, which corresponds to the power consumption of the machine when idle.

As exposed in Section 2, many previous works indicate that power consumption is roughly quadratic (in the frequency) but we decided not to build on this assumption and instead to allow users to specify arbitrary linear (in resource usage) relations for each possible frequency (see Figure 1) since technology is likely to significantly evolve in the near future. This permits users to easily encode different specific power consumption states to account for booting, shutdown or a deep sleep mode (where the idle power consumption is significantly different from what can be obtained when simply changing the frequency). As SimGrid was originally designed for heterogeneous platforms, each machine can have its own performance (in flop per second) and power model (in Watts depending on the load), which allows us to account for possible heterogeneity in a cluster.

In simulation, we keep track at any time of which processes

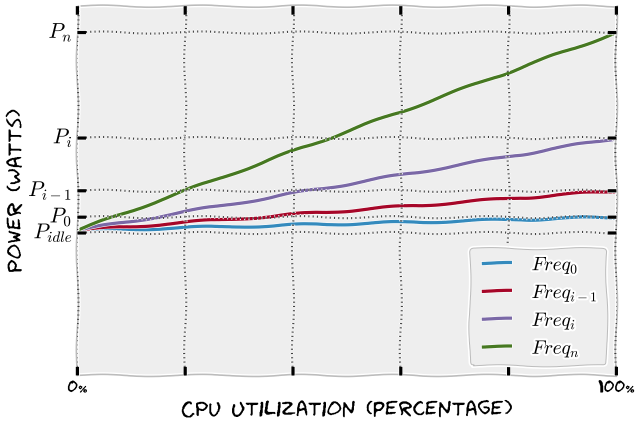


Figure 1: Power model implemented in SimGrid.

are actively computing and which ones are idle or engaged into communications. We can therefore compute on the fly the CPU usage and update the power consumption accordingly. Such power consumption is exposed per host through the `sg_get_host_consumed_energy` function, which allows an application to gather statistics on its execution and dynamically decide whether or not it should change the current frequency (thereby mimicking a `userspace` governor) of a computing node through the `sg_host_set_pstate` function. All such events (frequency modifications, load variations, power consumption queries) are integrated in the SimGrid tracing infrastructure that enables users to perform a *post-mortem* analysis and to specifically relate energy consumption with the application behavior.

5. EXPERIMENTAL SETUP

In this work, we relied on the Grid'5000 [6] infrastructure, in particular on the taurus cluster[2] due to the availability of accurate hardware wattmeters. The measurements of these wattmeters are accessed through the Grid'5000 API and the monitoring ensures a sampling rate for each machine of 1Hz with an accuracy of 0.125 Watts.

The taurus cluster is composed of 16 homogeneous nodes; each node consists of 2 `Intel Xeon E5-2630` CPUs with 6 physical cores per CPU and 32 GB of RAM. Each CPU has 3 cache levels of the following sizes: 32 KByte for L1, 256 KByte for L2 and 15 MB for L3. These computing nodes are interconnected via 10 Gb Ethernet links to the same switch as two other small 4-node clusters, `hercule` and `orion`, as well as a service network. In order to rule out any performance issue incurred by other users, in particular regarding network usage, we took care of reserving these clusters during our experiments as well, although they are not part of this study. We also took care of deploying our own custom Debian GNU/Linux images before any experiment to ensure that we are in full control of the software stack used.

To evaluate the relevance of our approach, we use three MPI applications. The first two originate from the MPI NAS Parallel Benchmark suite (v3.3). The NAS EP benchmark performs independent computations with three `MPI_Allreduce()` operations at the end to check the correctness of the re-

[2]Technical specification at https://www.grid5000.fr/mediawiki/index.php/Lyon:Hardware#Taurus.

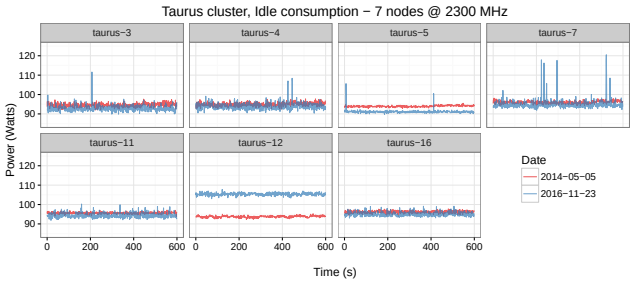


Figure 2: Idle power consumption along time when the frequency is set to 2300 MHz for each machine in 2014 and in 2016. The Y-axis does not start at 0 to provide a better appreciation of the variations.

sults. The NAS LU benchmark performs a Lower-Upper decomposition using the Gauss-Seidel method and moderately relies on the `MPI_Allreduce()` and `MPI_Bcast()` operations. Most of its communication patterns are implemented through blocking and non-blocking point-to-point communications. Finally, we selected the HPL benchmark (v2.2) as it is commonly used to rank supercomputers both in the top500 and in the green500 [56].

As we will explain in Section 8, many parameters can impact both performance and power consumption of such a cluster, which is why one should be rigorous when proposing a model and validating it. This work has been conducted with reproducible research in mind by using the principles and the Git/Org-mode workflow presented in [52]. We used literate programming [50] and the `R` statistical software [45] to automate data processing and figure generation and we tracked as much information as we could on our experiments. All the results that are given in this document are available online[3] for further inspection.

6. MODEL VERIFICATION AND CALIBRATION

We now describe our calibration procedure, i.e., the measurements we perform to obtain a model of the target platform. Our aim is to keep this series of measurements as brief as possible and to rely on as few nodes as possible.

6.1 Power Model

The previously described power model is essentially linear in the load of the whole machine and is dependent on the machine as well as on the application. We have checked such properties on the taurus cluster and now illustrate these aspects.

Figure 2 depicts the power consumption along time at two different dates (May 2014 and October 2016) for various nodes. Not only can significant differences be observed between nodes, but the power consumption of taurus-12 has for example increased by 11W while the one of taurus-5 has decreased by 3W. In 2014, the cluster could be considered as homogeneous but in 2016, this is evidently not the case anymore. These measurements are however quite stable. On two-hours time scales, a few outliers (around 0W or 50W) per node can be easily detected and removed as they can be attributed to powermeter unreliability. The sample mean is thus a very good approximation of the distribution and we calibrated our models accordingly.

[3] https://gitlab.inria.fr/fheinric/paper-simgrid-energy

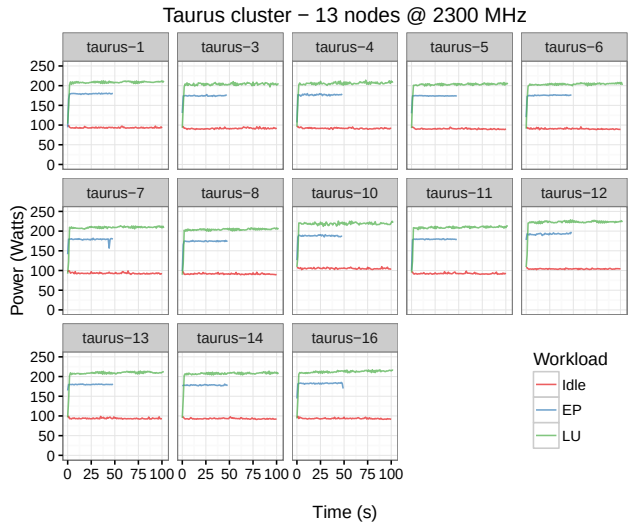


Figure 3: Power consumption along time when running EP, LU or idling (12 active cores and the frequency set to 2300 MHz).

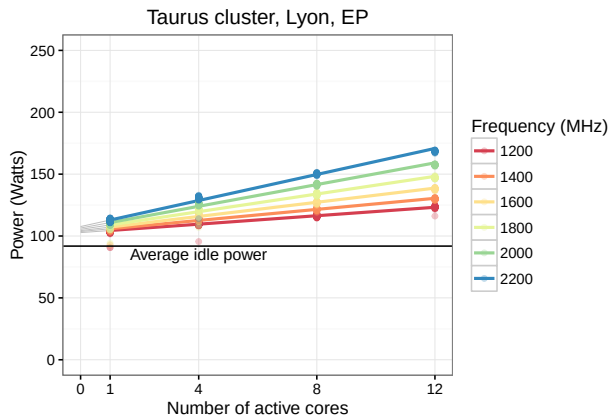


Figure 4: Power consumption on taurus-8 when running EP, class C, varying the frequency and the number of active cores.

The heterogeneity of computing nodes with respect to power consumption, even once we ensured its homogeneity in terms of performance, can also be observed in Figure 3 that depicts how the workload (independent executions of the EP or of the LU NAS PB with all cores used of each node) influences power consumption at a macroscopic scale. Indeed, making micro-estimations of such operations requires extremely fine tracing and power measurement tools that are rarely available.

Figure 4 illustrates the linearity in load of power consumption and finally, Figure 5 illustrates the quadratic influence on frequency even if we decided to not build on this property.

When calibrating the power model, we fix the application workload and the compilation chain. Then for every machine/frequency, we measure the idle consumption and the full load consumption for a few dozens of seconds. The first/last two values should be dropped as there is often some clock differences between the power meters clock and the CPU clock (see for example the first and last measurements

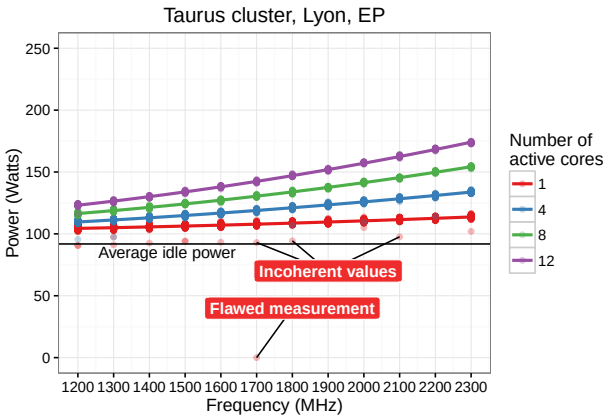


Figure 5: Power consumption on taurus-8 when running EP (class C) and varying the frequency and the number of active cores. A quadratic model in the frequency provides a perfect fit.

in Figure 3). Then, a linear regression on all measurements is performed to obtain the parameters of the model. From our experience, there is no need to perform a more complex analysis (e.g., median or quantile regression). The few outlier values (as seen in Figure 5) are gracefully handled by the standard linear model.

If the cluster can be assumed homogeneous (as it used to be the case for our cluster in 2014), this calibration process can be sped up by selecting a few nodes at random for the calibration. Likewise, since the quadratic model is a reasonable assumption, it is possible to speed up the calibration by simply measuring for low, medium and high frequency and fitting accordingly.

Figure 6 is an excerpt from the XML SimGrid platform description. For a given host (here, `taurus-8`) a power consumption model can be provided for each possible frequency (12 here, starting from the highest frequency). Being able to specify such a model per host allows researchers to account for machines that behave in a significantly different way (e.g., the `taurus-12` machine as illustrated in Figure 2) and thus to free our model from any homogeneity assumption.

6.2 Multicore Computation Model

As we explained earlier, whether a given piece of code is sped up or slowed down is quite difficult to foresee since it is very dependent on its nature and on the memory hierarchy. To characterize the true performance of the application, we first run the target application with a small workload **using all the cores of a single node**. This execution is traced and the duration of every computation is recorded ($\texttt{Calibration}^{RL}$). We then re-execute the application with the exact same workload but on top of the simulator (hence **using a single core**) and trace accordingly the duration of every computation as well as the portion of code it corresponds to ($\texttt{Calibration}^{SG}$). The origin of the code (each computation is simply identified by the filename and the line number of the surrounding two MPI calls) is obtained during compilation to incur a minimal overhead during the simulation.

Since the application code is emulated by SimGrid, the duration of computations in $\texttt{Calibration}^{RL}$ may be quite different from the ones in $\texttt{Calibration}^{SG}$. We automatically align the $\texttt{Calibration}^{RL}$ and $\texttt{Calibration}^{SG}$ traces in an `R` script to identify for each code region a speed up or slow down factor that should be applied when emulating the target application. For a given code region c, this factor is defined as the ratio of the total time over all ranks spent in c in $\texttt{Calibration}^{SG}$ to the total time over all ranks spent in c in $\texttt{Calibration}^{RL}$, which enables SimGrid to scale dynamically measured elapsed times accordingly.

```
<host id="taurus-8.lyon.grid5000.fr" speed="20000" core="12">
  <prop id="watt-per-state" value="92.75:114.62:174.38, 92.75:113
      .25:168.62, 92.88:112.25:162.88, 92.88:110.75:157.12, 92.88
      :110.38:151.75, 92.88:109.38:147.25, 92.88:108.62:142.75, 93
      :107.38:138.25, 93.12:106.75:134, 93:106.5:130.62, 93:105.12
      :127, 93.25:104.62:123.62" />
  <prop id="watt-off" value="10" />
</host>
```

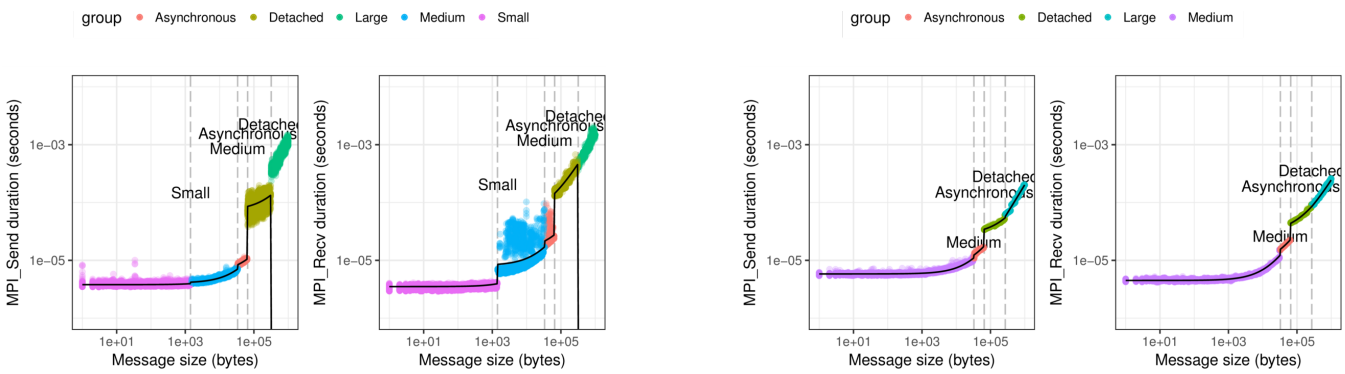
Figure 6: Power model (derived from the linear regressions of Figure 4) for the the `taurus-8` node with the EP benchmark. Each frequency is characterized by three values: the idle power, the power consumption when 1 core is being used and the power consumption when all 12 cores are being used. The "watt_off" value indicates the power consumption when the machine is turned off.

```
"start-stop","ratio"
"bcast-inputs.f:37:exchange-3.f:42",0.165580862005622
"exchange-1.f:30:exchange-1.f:48",14.6704533221381
"exchange-1.f:30:exchange-1.f:113",1.29679941120005
"exchange-1.f:30:exchange-1.f:130",1.29943496285544
. . .
"exchange-3.f:288:exchange-1.f:48",0.893744914907817
"exchange-3.f:288:exchange-1.f:113",0.900446944415142
"exchange-3.f:288:l2norm.f:57",0.904388406443553
"init-comm.f:31:bcast-inputs.f:28",9.84073354384261
"ssor.f:210:exchange-3.f:42",0.149455207049195
```

Figure 7: Computation factors for the LU benchmark.

Figure 7 depicts an excerpt of the file resulting from the calibration of computations. This file serves as input to SMPI and is used to correctly account for the duration of the computations on the target architecture. Some code regions have a speedup factor of around 1.29, which means that the duration of the corresponding code is actually faster when emulating than when running in a normal environment, while some other code regions have a speedup around 0.9, which means they are slower when emulating. For some applications, like EP or for HPL, most factors are very close to 1 and such correction has therefore almost no impact on the overall makespan prediction. However, for a code like LU, not accounting for such slowdowns and speed ups leads to an overall runtime estimation error of the magnitude of 20 to 30%. It is interesting to note that some code regions (e.g., the first and the last ones) can have very low speedup factors while others (e.g., the second one) can have very important speedup factors. In our experience, the ones with low speedup factors are seldom called (e.g., only once per rank) while the ones with large speedup factors have very frequent calls (possibly hundreds of thousands) and a very short duration.

6.3 Network Model



(a) Remote communications (Ethernet)

(b) Local communications (shared memory)

Figure 8: Communication time (either between two nodes or within the same node) of Taurus. The duration of both `MPI_Send` and `MPI_Recv` are piece-wise linear functions of message size but with different regimes and performance depending whether communication take place over Ethernet or over shared memory.

The complexity of the networking stack often makes modeling of communications rather complicated. To faithfully account for communication times in a wide range of settings, it is essential to correctly characterize the behavior of the MPI layer. As explained in [7], the communication model implemented in SMPI is a hybrid model between the LogP family and a fluid model that takes into account whether messages are sent asynchronously in eager mode (e.g., for relatively small messages), in a detached way (i.e., without necessarily blocking the sender but requiring the receiver to post the reception before the communication can actually take place) or in a synchronized way (i.e., using a *rendez-vous* protocol). Switches from one mode to another depend on message size and the resulting performance can be modeled through a piece-wise linear model with as many pieces as needed. The calibration procedure we use[4] consists of running series of `MPI_Send` and `MPI_Recv` of carefully randomized sizes between two nodes and to fit piece-wise linear models with the `R` statistical language.

Figure 8(a) illustrates the time spent in the `MPI_Send` (resp. `MPI_Recv`) function by ensuring that the receiver (resp. sender) is always ready to communicate. As illustrated on this figure, at least 5 modes can be distinguished depending on message size and correspond not only to different synchronization modes but also to different kinds of performances. Although the protocol switches from one mode to another could clearly be optimized, such kind of behavior is common and more than 5 modes are commonly found for TCP Ethernet networks. From such data, the breaks and the corresponding linear regressions can then be provided to SimGrid (see Figure 9 for an excerpt of the platform description).

Note that such model only makes sense for remote communications and that communications that remain internal to a node use shared memory rather than the network card. A similar series of measurements was run between two cores of the same node to calibrate accordingly the model for local communications. Since such communications use shared memory, it is common to observe not only very different performances, but also slightly different behaviors (protocol changes are not done for the same messages and the regressions are different). Figure 8(b) illustrates the calibration of local communications. Although they are much more stable, simpler and more efficient than remote communications, using a completely different model to distinguish between local and remote communications turned to be of little importance because the applications we considered do not exploit locality and communication is dominated by remote communications. Communications over the memory where therefore simply modeled by a 5GiB shared link (the maximal bandwidth we measured), which allows to account for contention (in particular for the HPL benchmark that heavily communicates locally), and using the same model (breaks and bandwidth correction factors) as for remote communications.

```
<prop id="smpi/os" value="0:3.79946267082783e-06:1.09809596167633e
      -10; 1420:4.06752467953734e-06:8.98782555257323e-11; 33500:6
      .01204627458251e-06:7.10122202865045e-11; 65536:7
      .28270630967833e-05:1.9683266729216e-10; 320000:0:0"/>
<prop id="smpi/ois" value="0:3.65848336553355e-06:1.33280621516301
      e-10; 1420:3.83673729379869e-06:7.84867337035856e-11; 33500:5
      .57232433176236e-06:6.5668893954931e-11; 65536:4
      .17803219267394e-06:2.37460347640595e-12; 320000:4
      .70677307448713e-06:3.38065421824938e-13"/>
<prop id="smpi/or" value="0:3.51809764924934e-06:3.01847204118237e
      -10; 1420:8.16124874852713e-06:2.66840481979518e-10; 33500:1
      .49347740713389e-05:1.97645004617501e-10; 65536:5
      .88893263987424e-05:1.29160163208845e-09; 320000:0:0"/>
<prop id="smpi/bw-factor" value="0:0.0489825651012801; 1420:0
      .824385608826111; 33500:0.600278012183156; 65536:1; 320000:0
      .536759617074721"/>
<prop id="smpi/lat-factor" value="0:1; 1420:2.16408517748122;
      33500:1.76905573216394; 65536:2.9114462429055; 320000:2
      .5981998109037"/>
<prop id="smpi/async-small-thresh" value="65536"/>
<prop id="smpi/send-is-detached-thresh" value="320000"/>
```

Figure 9: Parameters of the MPI communication model built from the multiple linear regressions.

It should be noted that if the network topology is known or expected to be decisive (as it was the case in [7], where a much more contended network topology was considered), then some saturation experiments should be done to properly evaluate where bottlenecks may occur. We ran such experiments for consistency but as our cluster is of limited size with a well provisioned router, a flat topology was sufficient.

Finally, one should make sure that the same collective

[4] See https://gitlab.inria.fr/simgrid/platform-calibration for more details.

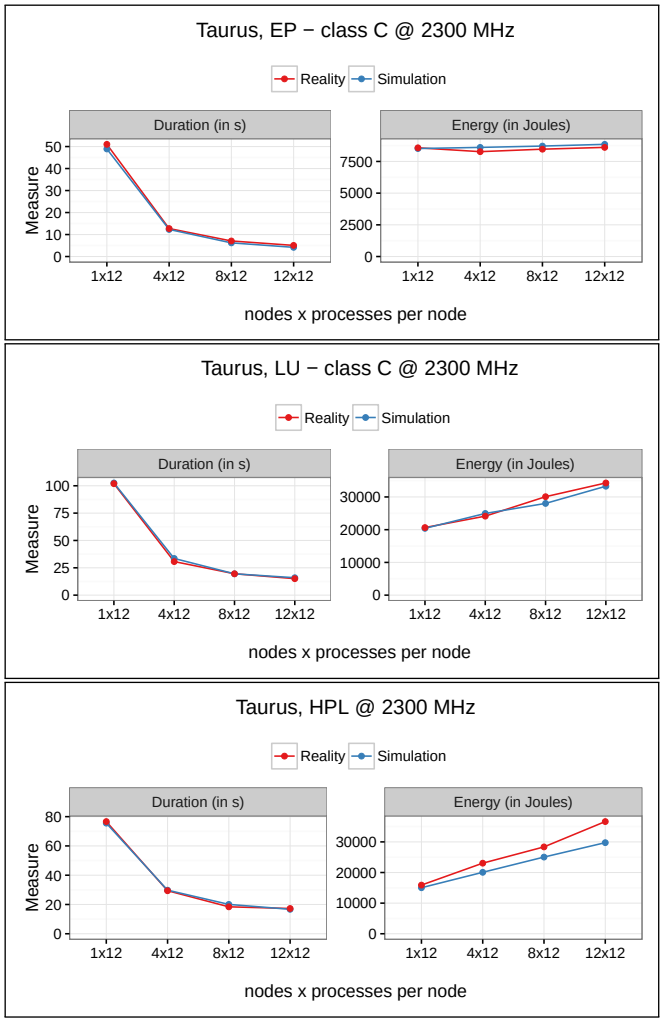


Figure 10: Validating simulation results with EP, LU, and HPL for up to 144 processes.

communication selector is used for both real life experiments and the simulation. Yet, the applications we used to evaluate our new models barely rely on collective communications, so the impact of this particular configuration is limited as well.

7. VALIDATION

To validate that the previously detailed approach is effective in predicting both the performance and the power consumption of HPC applications from a calibration at small scale, we compare in Figure 10 the results of simulations with real executions for the three applications presented earlier: EP, LU and HPL. In all cases, the performance prediction is almost undistinguishable from the outcome of the real experiments.

Although the previous general behavior is somehow expected (perfect speed up for EP, and sublinear for LU, hence an increasing energy consumption), we manage to predict performance systematically within a few percents (within 2%), except for the HPL benchmark. This mismatch for larger configurations can be explained by the fact that HPL busy waits on communications through `MPI_Probe` and that we do not currently model such power consumption overhead.

8. CONTROLLING THE EXPERIMENTAL ENVIRONMENT

Computers have become increasingly complex and even minor modifications to the setup can have major impact on performance [38]. In this discussion, we make an inventory of all parameters that may influence the behavior of the system, in terms of both speed and power consumption (see Figure 11). We identified these parameters as the principal ones: every experimenter should track them so that a faithful decision can be made whether or not the system requires to be re-calibrated.

The first category is related to the hardware at hand. In our case, the cluster is made of the same types of CPUs. Yet, it is common to encounter performance and power variability even in high-end systems [29]. Even in a small homogeneous cluster like the one we used for our study, machines do not all exhibit the same behavior. These variations can be explained by the fact that, although for instance all CPUs have the same type, they may actually come from different batches/factories. The position in the rack and the network port to which they are connected may also have an impact, which is why one should make sure to carefully record which nodes are used and measure them individually if one has any doubt on the homogeneity of the cluster [13].

The second category of factors is actually related to **when** the system is measured. Computers are indeed quite sensitive to temperature and so can the temperature of the machine room affect the speed of processors, their power consumption and even sometimes their clock drift [2]. Likewise, vibrations can negatively affect the performance of mechanical hard drives. In this work, we focus on rather macroscopic measurements and never encountered situations where these factors seemed responsible for major behavior modifications from an experiment to another. However, in the two year time frame of this work, the hardware experienced several BIOS and firmware updates. Although the Grid'5000 team is extremely careful about how such operations are conducted and documented, unexpected side-effects can always happen and go unnoticed. For example, during our experiments, we realized that a few nodes were significantly slower (about 18%) than the others on specific benchmark (NASPB LU). It is only after several days of careful investigations that we were able to narrow this down (and fix it in the BIOS settings) to the fact that these machines were using a different memory operating mode (AdvECCMode instead of OptimizerMode) that provides a better reliability at the cost of slower performance. During our two year time frame, the hardware also somehow wore out and after ensuring that all nodes were behaving identically in terms of speed, significant differences in terms of power consumption can be observed even when idle (as depicted in Figure 2).

The third category is related to the operating system. On Grid'5000, even when deploying our own custom images, some systemd scripts of the platform often timed-out shortly after the deployment and boot procedures and influenced performance in a seemingly random way, which is why a delay of a few minutes should be observed before running measurements. The software stack and how applications are compiled also has a major influence on both speed and instantaneous power consumption (e.g., depending whether the compiler manages to exploit vector units or not).

The fourth category of parameters is related to the ker-

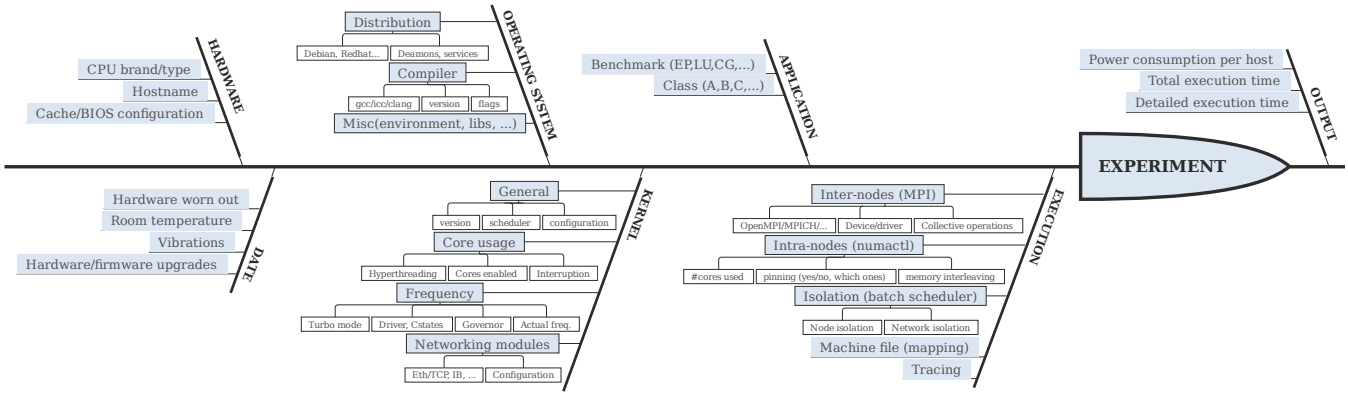


Figure 11: Causal diagram associated with the performance of an HPC system.

nel configuration. The versions of the kernel and of the scheduling algorithm greatly impact core usage, sometimes in a dramatic way [35]. When speed and power consumption are at stake, the question of whether hyperthreading and turbomode should be activated or not, whether the frequency should be fixed or even if cores should be dynamically enabled and disabled, is often raised. As a matter of fact, frequency directly impacts performance of HPC codes, but on the hardware we had, neither changing the frequency nor deactivating cores influences idle power consumption.

Note that activating and deactivating cores is not harmless as it often influences how interrupts are handled. When these interrupts are not re-balanced over all cores, a significant performance degradation when communicating may be the result. Finally, before even setting a frequency, one should decide which driver (`intel_pstate` vs. `ACPI/cpu_freq`) is used, which energy saving C-states are allowed and which governor (`performance`, `userspace`, ...) is used. For this study, we used the generic ACPI driver, the userspace governor and allowed the CPU to enter all C-states (i.e., no C-states were disabled). Finally, the networking module has a great influence on usable bandwidth and latency and therefore on parallel application performance, even to some extent on CPU power consumption (network cards have a fixed limited power consumption that is relatively independent of the load).

The fifth category is related to the application itself. Parallel applications alternate between communication and computation bursts. How much energy is consumed depends on the application because of the instruction mix (integer, floating point instructions, memory access, cache misses, ...) and is depicted in Figure 3. We saw in this experiment with several applications that the behavior is homogeneous along time and to some extent across machines (the fixed cost associated to having the node turned on even when idle depends on each node). Also, the variability is minimal and characterizing power consumption by estimating its expectation at full load is thus perfectly sound for a given application.

Depending on the application workload class, variations may appear as it modifies the instruction mix and how data fits into the caches, but in our experiments this was generally hardly noticeable.

The sixth category is related to runtime. Which implementation of MPI is used and how it is configured heavily influences the overall runtime of an application. Since all our nodes are multi-core, one should decide how many cores are actually used and pin every MPI process to a particular core. Then, the machine file indicates which MPI rank is mapped to each node/core. Lastly, whenever tracing is activated, one should ensure that files are written on local disks or in ram disks and not on network file systems.

Finally, once the application is executed, we capture all output, the total execution time and sometimes even detailed traces of the application. Although the tracing overhead may slightly bias measurements compared to a normal execution, it turns out to be very useful when comparing with simulation outputs as it allows to either understand where some mismatch may come from or to compare both at a much finer level of detail than just the total computation time. Up to now, we did not observe other key parameters influencing the behavior of the system. However, future architectures may bring new ones.

9. CONCLUSION AND FUTURE WORK

9.1 Contribution

SimGrid is recognized for the quality of its support for modeling network operations (both regarding the complexity of point-to-point operations and for contention). Alas, proper support to correctly account for the complexity of computations on multi-core architecture was still missing. Several options, including the exploitation of performance counters (as done for example in [51]), had been considered but, so far and from our experience, they revealed less effective to deal with the emulation bias (e.g., frequent context switching or the suppression of contention on the L3 cache) than the coarse-grain calibration/correction procedure we propose in this article. Although this approach may be regarded as over-fitting, it is important to understand that only a single node is used when performing this calibration and that the rest of the behavior at scale does not require any particular adjustment. It is also important to understand that neglecting such bias quickly incurs 20 to 30% of errors both at small and large scales.

We additionally implemented a simple but effective way to account for power consumption in SimGrid as well as the ability for the simulated code to switch from one frequency (including turning off the machines) to another and to query for current and past power consumption.

Finally, we explained how such modeling and calibration of the platform should be done and recalled how important it was to carefully control all parameters that can influence the application performance and power consumption. Al-

though the models we propose are relatively simple models, we proved that, provided they are correctly instantiated, they could provide very accurate predictions (within a few percents) for classical HPC benchmarks like HPL and some of the NAS parallel benchmarks. Note that our proposal is generic and we strongly believe that the same methodology can be applied to more complex real-life applications like BigDFT [15] or Ondes3D [21] which have already been successfully emulated with SMPI.

All these features have been integrated in the latest version of SimGrid and can readily be used. We also took care of making all our calibration code and scripts[5] available to encourage and ease the reproduction of our work.

9.2 Discussion and Limitations

Currently, only a single power model can be specified per host while we have illustrated how important the application workload was regarding power consumption (at full speed, there are about 30 Watts of difference depending on whether LU or EP is being run). The approach we propose requires the application to be somehow regular in time. This is also somehow one of the source of inaccuracies the HPL workload that alternates between computations and intensive busy waiting. If the application exhibits very different computation patterns, it will be necessary to modify SimGrid so that the power consumption also depends on the code section being run. Although such an implementation would not be difficult, it would require a very careful instantiation and the ability to perform very precise measurements, which is not possible with the hardware we currently have at our disposal as it has a low sampling-rate (one second). Recent works [25, 28] propose solutions that address these limitations and could be used in our context.

The other potential weakness we can identify is our current inability to account in SimGrid for a significantly different network performance model for local and remote communications. Although arbitrarily complex hierarchies or combinations of topologies [8] can be modeled in SimGrid, only one network model can be used at a time for the whole simulation. Support for each network to have its own network model and set of parameters requires some refactoring of SimGrid and is currently underway.

9.3 Future Work

We believe the power model we propose is sufficiently flexible and faithful to allow complex scheduling studies involving new resource allocation policies, workload consolidation and the use of DVFS capabilities (e.g., to select the more energy-efficient frequency depending on the application phase, or to mitigate energy waste incurred by load imbalance). There exists several current efforts [16, 22] to allow real batch schedulers like SLURM or OAR to be emulated on top of SimGrid. Such projects will clearly benefit from the power models presented in this article.

At a different scale, there are recent efforts to incorporate such strategies at the application level, directly in the runtime [11]. Indeed, the growing complexity and heterogeneity of hardware (big.Little, accelerators, etc.) provides many optimization opportunities that are likely to be addressed only at the runtime level. Our previous work [53] on performance prediction of dynamic task-based runtimes (StarPU [4]) for hybrid (multi-core and multi-GPU) architectures should directly benefit from our new power models. Carefully modeling the energy consumption of GPUs and possibly of transfers between CPUs and GPUs is also likely to require much finer energy tracing tools than the ones currently at our disposal. Such runtimes also work at a larger scale and efficiently leverage MPI [3, 1] and one of our current efforts is related to solving all technical problems allowing us to emulate such complex and dynamic applications.

Finally, we are working towards making our calibration procedure more robust and more automatic. Such a tool will allow to more easily populate a platform repository[6] and to investigate whether we can reproduce the Top500 and the Green500 ranking by simulation.

Acknowledgments

The authors would like to thank the SimGrid team members and collaborators who contributed to SMPI. This work is partially supported by the Hac Specis Inria Project Lab, the ANR SONGS (11-ANR-INFR-13), and the European Mont-Blanc (EC grant 288777) projects. Experiments were carried out on the Grid'5000 experimental testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and other Universities and organizations (see https://www.grid5000.fr).

10. REFERENCES

[1] E. Agullo, O. Aumage, M. Faverge, N. Furmento, F. Pruvost, M. Sergent, and S. Thibault. Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model. Research Report RR-8927, Inria Bordeaux Sud-Ouest ; Bordeaux INP ; CNRS ; Université de Bordeaux ; CEA, June 2016.

[2] D. W. Allan. Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, 34(6), Nov. 1987.

[3] C. Augonnet, O. Aumage, N. Furmento, R. Namyst, and S. Thibault. StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators. In S. B. Jesper Larsson Träff and J. Dongarra, editors, *The 19th European MPI Users' Group Meeting (EuroMPI 2012)*, volume 7490, Vienna, Austria, Sept. 2012. Springer.

[4] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience*, 23(2), 2011.

[5] R. M. Badia, J. Labarta, J. Giménez, and F. Escalé. Dimemas: Predicting MPI Applications Behaviour in Grid Environments. In *Proc. of the Workshop on Grid Applications and Programming Tools*, June 2003.

[6] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In I. Ivanov, M. Sinderen, F. Leymann, and T. Shan, editors, *Cloud Computing and Services Science*, volume

[5] See https://gitlab.inria.fr/fheinric/paper-simgrid-energy and https://gitlab.inria.fr/simgrid/platform-calibration

[6] https://gitlab.inria.fr/simgrid/platform-calibration

367 of *Communications in Computer and Information Science*. Springer International Publishing, 2013.

[7] P. Bedaride, A. Degomme, S. Genaud, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell, Lee, F. Suter, and B. Videau. Toward Better Simulation of MPI Applications on Ethernet/TCP Networks. In *Proc. of the 4th Intl. Workshop on Performance Modeling, Benchmarking and Simulation*, volume 8551 of *LNCS*, Denver, CO, Nov. 2013. Springer.

[8] L. Bobelin, A. Legrand, D. A. G. Márquez, P. Navarro, M. Quinson, F. Suter, and C. Thiery. Scalable Multi-Purpose Network Representation for Large Scale Distributed System Simulation. In *Proc. of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Ottawa, Canada, 2012.

[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience*, 41(1), Jan. 2011.

[10] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *Journal of Parallel and Distributed Computing*, 74(10), 2014.

[11] J. Chen, L. Tan, P. Wu, D. Tao, H. Li, X. Liang, S. Li, R. Ge, L. Bhuyan, and Z. Chen. Greenla: Green linear algebra software for gpu-accelerated heterogeneous comp. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016.

[12] G. L. T. Chetsa, L. Lefevre, J. M. Pierson, P. Stolf, and G. D. Costa. Application-Agnostic Framework for Improving the Energy Efficiency of Multiple HPC Subsystems. In *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2015.

[13] J. D. Davis, S. Rivoire, M. Goldszmidt, and E. K. Ardestani. Including Variability in Large-Scale Cluster Power Models. *IEEE Computer Architecture Letters*, 11(2), 2012.

[14] M. Dayarathna, Y. Wen, and R. Fan. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys Tutorials*, 18(1), 2016.

[15] F. Dupros, F. D. Martin, E. Foerster, D. Komatitsch, and J. Roman. High-performance finite-element simulations of seismic wave propagation in three-dimensional nonlinear inelastic geological media. *Parallel Computing*, 36(5–6), 2010.

[16] P.-F. Dutot, M. Mercier, M. Poquet, and O. Richard. Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. In *20th Workshop on Job Scheduling Strategies for Parallel Processing*, Chicago, United States, May 2016.

[17] C. Engelmann. Scaling To A Million Cores And Beyond: Using Light-Weight Simulation to Understand The Challenges Ahead On The Road To Exascale. *FGCS*, 30, Jan. 2014.

[18] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Power Limitations and Dark Silicon Challenge the Future of Multicore. *ACM Transactions on Computer Systems (TOCS)*, 30(3), Aug. 2012.

[19] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. Understanding the future of energy-performance tradeoff via DVFS in HPC environments. *Journal of Parallel and Distributed Computing (JPDC)*, 72(4), Apr. 2012.

[20] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer. Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[21] L. Genovese, A. Neelov, S. Goedecker, T. Deutsch, S. A. Ghasemi, A. Willand, D. Caliste, O. Zilberberg, M. Rayson, A. Bergman, and R. Schneider. Daubechies Wavelets as a Basis Set for Density Functional Pseudopotential Calculations. *Journal of Chemical Physics*, 129(014109), 2008.

[22] Y. Georgiou, D. Glesser, K. Rzadca, and D. Trystram. A Scheduler-Level Incentive Mechanism for Energy Efficiency in HPC. In *CCGrid 2015 - 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Shenzhen, China, May 2015.

[23] B. Giridhar, M. Cieslak, D. Duggal, R. Dreslinski, H. M. Chen, R. Patti, B. Hold, C. Chakrabarti, T. Mudge, and D. Blaauw. Exploring DRAM Organizations for Energy-efficient and Resilient Exascale Memories. In *International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2013.

[24] T. Guérout, T. Monteil, G. D. Costa, R. N. Calheiros, R. Buyya, and M. Alexandru. Energy-aware simulation with dvfs. *Simulation Modelling Practice and Theory*, 39, dec 2013.

[25] D. Hackenberg, T. Ilsche, J. Schuchart, R. Schöne, W. E. Nagel, M. Simon, and Y. Georgiou. Hdeem: High definition energy efficiency monitoring. In *Proceedings of the 2Nd International Workshop on Energy Efficient Supercomputing*, E2SC '14, Piscataway, NJ, USA, 2014. IEEE Press.

[26] D. Han and T. Shu. Thermal-aware energy-efficient task scheduling for DVFS-enabled data centers. In *International Conference on Computing, Networking and Communications (ICNC)*, Feb 2015.

[27] T. Hoefler, C. Siebert, and A. Lumsdaine. LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model. In *Proc. of the LSAP Workshop*, June 2010.

[28] T. Ilsche, D. Hackenberg, S. Graul, R. Schöne, and J. Schuchart. Power measurements for compute nodes: Improving sampling rates, granularity and accuracy. In *Green Computing Conference and Sustainable Computing Conference (IGSC), 2015 Sixth International*. IEEE, 2015.

[29] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda, M. Kondo, and I. Miyoshi. Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, New York, NY, USA, 2015. ACM.

[30] C. L. Janssen, H. Adalsteinsson, S. Cranford, J. P. Kenny, A. Pinar, D. A. Evensky, and J. Mayo. A simulator for large-scale parallel architectures. *International*

Journal of Parallel and Distributed Systems, 1(2), 2010.

[31] H. Kimura, M. Sato, Y. Hotta, T. Boku, and D. Takahashi. Emprical study on Reducing Energy of Parallel Programs using Slack Reclamation by DVFS in a Power-scalable High Performance Cluster. In *IEEE International Conference on Cluster Computing*, 2006.

[32] D. Kliazovich, P. Bouvry, and S. U. Khan. A packet-level simulator of energy-aware cloud comuting data centers. *Journal of Supercomputing*, 62(3), 2012.

[33] E. Le Sueur and G. Heiser. Dynamic voltage and frequency scaling: the laws of diminishing returns. In *USENIX International conference on Power aware computing and systems (HotPower)*, 2010.

[34] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic Right-sizing for Power-proportional Data Centers. *IEEE/ACM Transactions on Networking (TON)*, 21(5), Oct. 2013.

[35] J.-P. Lozi, B. Lepers, J. Funston, F. Gaud, V. Quéma, and A. Fedorova. The Linux Scheduler: a Decade of Wasted Cores. In *EuroSys 2016*, Proceedings of the Eleventh European Conference on Computer Systems, London, United Kingdom, Apr. 2016. ACM.

[36] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta. Evaluating the Effectiveness of Model-based Power Characterization. In *USENIX Annual Technical Conference (ATC)*, 2011.

[37] M. Mubarak, C. D. Carothers, R. B. Ross, and P. H. Carns. Enabling parallel simulation of large-scale hpc network systems. *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[38] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney. Producing wrong data without doing anything obviously wrong! In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV. ACM, 2009.

[39] T. Nowatzki, J. Menon, C. H. Ho, and K. Sankaralingam. Architectural simulators considered harmful. *IEEE Micro*, 35(6), Nov 2015.

[40] A. Núñez, J. Fernández, J.-D. Garcia, F. Garcia, and J. Carretero. New Techniques for Simulating High Performance MPI Applications on Large Storage Networks. *Journal of Supercomputing*, 51(1), 2010.

[41] A.-C. Orgerie, M. Dias de Assunção, and L. Lefèvre. A Survey on Techniques for Improving the Energy Efficiency of Large-Scale Distributed Systems. *ACM Computing Surveys (CSUR)*, 46(4), 2014.

[42] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Demystifying Energy Consumption in Grids and Clouds. In *Work in Progress in Green Computing (WIPGC) Workshop, in conjunction with IEEE International Green Computing Conference (IGCC)*, Aug. 2010.

[43] S. Ostermann, R. Prodan, and T. Fahringer. Dynamic Cloud Provisioning for Scientific Grid Workflows. In *Proc. of the 11th ACM/IEEE Intl. Conf. on Grid Computing (Grid)*, Oct. 2010.

[44] J. Peraza, A. Tiwari, M. Laurenzano, L. Carrington, and A. Snavely. PMaC's Green Queue: A Framework for Selecting Energy Optimal DVFS Configurations in Large Scale MPI Applications. *Concurrency and Computation: Practice & Experience*, 28(2), Feb. 2013.

[45] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[46] N. Rajovic et al. The mont-blanc prototype: An alternative approach for hpc systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.

[47] L. Savoie, D. K. Lowenthal, B. R. d. Supinski, T. Islam, K. Mohror, B. Rountree, and M. Schulz. I/O Aware Power Shifting. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016.

[48] R. Schöne, T. Ilsche, M. Bielert, D. Molka, and D. Hackenberg. Software Controlled Clock Modulation for Energy Efficiency Optimization on Intel Processors. In *International Workshop on Energy Efficient Supercomputing (E2SC)*, 2016.

[49] R. Schöne, D. Molka, and M. Werner. Wake-up latencies for processor idle states on current x86 processors. *Computer Science - Research and Development*, 30(2), 2015.

[50] E. Schulte, D. Davison, T. Dye, and C. Dominik. A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software*, 46(3), 1 2012.

[51] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia, and A. Purkayastha. A Framework for Performance Modeling and Prediction. In *Proc. of the ACM/IEEE Conference on Supercomputing*, Baltimore, MA, Nov. 2002.

[52] L. Stanisic, A. Legrand, and V. Danjean. An effective git and org-mode based workflow for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1), Jan. 2015.

[53] L. Stanisic, S. Thibault, A. Legrand, B. Videau, and J.-F. Méhaut. Faithful Performance Prediction of a Dynamic Task-Based Runtime System for Heterogeneous Multi-Core Architectures. *Concurrency and Computation: Practice and Experience*, May 2015.

[54] L. Tan, Z. Chen, and S. L. Song. Scalable Energy Efficiency with Resilience for High Performance Computing Systems: A Quantitative Methodology. *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(4), Nov. 2015.

[55] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya. DCSim: a data centre simulation tool for evaluating dynamic virtualized resource management. In *Int. Conf. on Network and Service Management*, 2012.

[56] Top500. https://www.top500.org, 2016.

[57] E. Valentin, M. Salvatierra, R. de Freitas, and R. Barreto. Response time schedulability analysis for hard real-time systems accounting DVFS latency on heterogeneous cluster-based platform. In *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2015.

[58] P. Velho, L. M. Schnorr, H. Casanova, and A. Legrand. On the validity of flow-level tcp network models for grid and cloud simulations. *ACM Trans. Model. Comput. Simul.*, 23(4), Dec. 2013.

[59] G. Zheng, G. Kakulapati, and L. Kale. BigSim: A Parallel Simulator for Performance Prediction of Extremely Large Parallel Machines. In *Proc. of the 18th IPDPS*, 2004.