Hardware and application aware performance, power and energy models for modern HPC servers with DVFS

Georges da Costa

▶ **To cite this version:**

Georges da Costa. Hardware and application aware performance, power and energy models for modern HPC servers with DVFS. Sustainable Computing : Informatics and Systems, 2025, 46, pp.101106. ⟨10.1016/j.suscom.2025.101106⟩. ⟨hal-04983485⟩

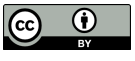
HAL Id: hal-04983485

https://hal.science/hal-04983485v1

Submitted on 16 Mar 2025

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.



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

Sustainable Computing: Informatics and Systems 46 (2025) 101106

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Hardware and application aware performance, power and energy models for modern HPC servers with DVFS

Georges Da Costa

IRIT, Université de Toulouse, CNRS, Toulouse, France

ARTICLE INFO

Keywords:
Performance
Power
Energy
Model
HPC

ABSTRACT

Energy usage and its ecological impact is now a major concern in High Performance Computing (HPC). To optimize supercomputers efficiency, researchers rely on models, as accessing actual platform is complex and costly. Changing DVFS (Dynamic Voltage and Frequency Scaling) is the most studied method, but it impacts power, performance and energy in a complex way.

We propose to bridge the gap between the theoretical and the practical approaches. We propose a multi cluster, multi application model accurately describing from a theoretical point of view the power and performance of applications subject to DVFS. We show how to use it on a runtime system with a minimal overhead, using only a few hardware performance counters and RAPL (Running Average Power Limit).

We validate our models using an extensive dataset, obtained using 18 different clusters and running 9 benchmarks. We also show how such model can be used to optimize the energy-to-solution for HPC workload.

1. Introduction

Energy usage is now a major concern in High Performance Computing (HPC) due to the ecological impact of electricity production, but also to its cost. Still, HPC usage is increasing due to its importance for addressing always increasing large scientific problems. The efficiency of its usage becomes a major concern.

Multiple strategies exist to increase the energy efficiency of supercomputers. Servers can be switched on and off, applications can be run on the optimal hardware on heterogeneous context, and they can be assigned the optimal number of resources when they are moldable. Finally one of the most efficient leverage is the capacity to change the DVFS (Dynamic Voltage and Frequency Scaling) of the processors, as it impacts the server power. Such leverages allows to optimize energy efficiency but can also be used in case of need of power-capping.

Reducing the processor frequency reduces the power, but at the cost of slowing down the applications. As both effects are contrary, this leverage is complex to use. Most research are either addressing this problem from a theoretical point of view, or are proposing runtime framework to optimize one server.

In this article, we propose to bridge the gap between the theoretical and the practical approaches. To do so we propose a multi cluster, multi application model accurately describing from a theoretical point of view the power and performance of applications subject to DVFS. We also show how to use it on a runtime system with a minimal overhead.

The contributions are the following:

- Power, performance and energy theoretical models taking into account the heterogeneity of servers and applications, for theoretical and experimental usage;
- Optimal frequency based on these models to minimize energy consumption for diverse servers and applications;
- Low overhead online practical models for power, performance and energy.

In Section 2 we will overview the existing literature. Then Section 3 will describe the proposed theoretical model, along with other existing models. Section 4 will explain the evaluation methodology as one important part of this work concerns the evaluation of multiple applications on multiple servers. The models will be compared using multiple metrics in Section 5. Finally, the low overhead usage of the models will be presented and evaluated in Section 6 before the conclusion.

2. State of the art

Several either practical [1] or theoretical [2] studies rely on power, performance and energy model of HPC servers under DVFS. They use these models either to overall minimize task energy, achieve slack reclamation, and many other uses.

Most studies trying to model HPC applications performance under Dynamic Voltage Frequency Scaling evaluate that the increase in time is not proportional to the change in frequency and link this

E-mail address: georges.da-costa@irit.fr.

https://doi.org/10.1016/j.suscom.2025.101106
Received 28 October 2024; Received in revised form 21 January 2025; Accepted 21 February 2025
Available online 8 March 2025
2210-5379/© 2025 The Author. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

to non-CPU (processor) activities (memory and network ones). For example, [3] introduces the β metric, later investigated in [4] that investigates the application slowdown compared to the CPU slowdown: $T(f)/T(fmax) = \beta \times fmax/f + c_0$. Their approach works as a source-to-source method, with a profile-driven approach. It is validated using several benchmarks on a single computer. We generalize a similar idea (the `Fluid` metric) with also proposing an actual online system based on system measures of the platform.

In the survey [5], authors describe several other researches addressing the problem of finding these models. Most of the existing work is focusing on single-server DVFS due to the problem of coordination between the servers. One of the existing work on multiple servers [6] focuses on NAS Parallel Benchmarks and uses DVFS to reduce the processors frequency when the network becomes a bottleneck. More recently, in [7], authors evaluate the β metric for several application, linking these values to the bottleneck of the particular application (cpu-bound, memory-bound, ...). Most of these research are limited to a single type of hardware, contrary to our approach where we use multiple type of servers to obtain more generic results.

Another approach is to use neural network architecture to learn the optimal frequency in function of the several performance counters values. In [8] authors uses their proposed online neural network on several application on a single server. Such type of approach leads to good results, but with a high overhead, and with a very limited theoretical usage. Similarly, in [9] authors use a Particle Swarm Optimization (PSO) system to estimate the power consumption in function of the frequency on three different applications on one server. This reduced context allows to have only between 4 to 6 percent of average error on power estimation, but needs a large amount of data. In the following we will show how our proposed method reacts when the amount of data is limited.

Finally very specific work can be done when focusing on a particular architecture (including a CPU and a GPU — Graphical Processor). In [10], authors propose a very specific model for their particular application with multiple phases, leading to a very precise model, taking into account the arithmetic pressure specific to the application. The change for the optimal frequency for each of the phases is added at the beginning of the said phase in the source code. This method is invasive and needs an access to the source code, while our approach consider applications as black-boxes.

Concerning the choice of hardware performance counters to monitor, they usually are studied in the context of a particular hardware. For example, in [11], authors explore an 8-core AMD FX-8320. Instead in our study we evaluate 18 clusters built over 10 years.

Similarly, in [12] authors focus on the online execution linked with a fine-grained model of the micro-architecture. While precise, the model is mostly tailored for online runtime, not theoretical work. It allows to find the frequency reaching the lowest energy state.

Most studies zoom on a particular hardware and obtain the associated constants after measurements. These values are linked to the particular environment but allows to reach a very high accuracy. In [13], authors use a single benchmark on a single type of server. The obtained constants are linked to this particular system. With such a specialized model, the accuracy on predicted energy is high, at 94%.

3. Approach

At any point in time, any application is usually either idle or using one of the server resource, such as the CPU, the memory, the network, ... In the following, we will only study processor-related DVFS. Similar work could be done on the other elements supporting DVFS such as GPU or memory.

The usage of DVFS impacts the application only when it uses the CPU.

We want to verify the following properties:

Table 1
Definition of the variables.

$Power$	Power consumption of a server in W
$Duration$	Duration of an application in s
$Energy$	Energy consumption of an application in J
$Fmax$	Maximum frequency of a server in Hz
$Pcoef$	Power coefficient of the frequency in formula (1)
$Pstatic$	Static part of the power consumption of a server in W
$Pdyn$	Dynamic part of the power consumption of a server in W
c	Cluster id
a	Application id
f	Frequency in Hz

- All applications have different behaviors power-and performance-wise in function of the hardware because of their resource usage;
- Variation of the classical models can include this diversity;
- Such modified model allows to precisely predict power and performance;
- This model can be used to find the optimal frequency for reducing energy-to-solution;
- The constants in the model can be measured with a low overhead at run-time.

In the following we will describe different models, starting with the classical one, up to the model we propose.

3.1. Classical model

Existing classical models [14] express the link between processor frequency and duration as linear, and between power and frequency derived from the underlying CMOS technology. They also assume that these models are only related to the hardware, not depending on the particular application running. They usually model using equations like (1) for the power model, (2) for the duration and (3) for the energy (which is the product of duration and power).

$$Power_f^c = Pstatic^c + Pdyn^c \left(\frac{f}{Fmax^c}\right)^{Pcoef} \tag{1}$$

$$Duration_f^{c,a} = Duration_{Fmax^c}^{c,a} \frac{Fmax^c}{f} \tag{2}$$

$$Energy_f^{c,a} = Duration_f^{c,a} * Power_f^c \tag{3}$$

The variables used in these equations are described in Table 1.

We always assume the frequency is one of the available frequency on the targeted cluster. The input data are in this case : $Duration_{fmax}^{c,a}$, $Fmax^c$, $Pcoef^c$, $Pstatic^c$, $Pdyn^c$. Out of these characteristics, all except one depends only on the hardware. If the performance is expressed as a speed-up (using $Duration_f^{c,a}/Duration_{Fmax^c}^{c,a}$), then the model depends only on the hardware. To obtain these constants, the usual method consist in running an application at maximum frequency and at minimum frequency on each cluster, measuring the associated Power consumption and solving the linear equation system to obtain $Pstatic^c$ and $Pdyn^c$. Concerning $Pcoef^c$, it is usually measured with at least one more measure (for a frequency between the maximum and minimum). Most publications use a value between 2 and 3. In the following we will use a constant of 2 in order to reduce the number of changing elements between the models.

Fig. 1 shows that the main characteristic impacting the energy behavior of this model is the ratio between $Pstatic$ and $Pdyn$. The optimal frequency to reduce the total energy depends on this ratio.

We will show in the following that this classical model has three limits:

- Different applications consume different power even at maximum frequency;

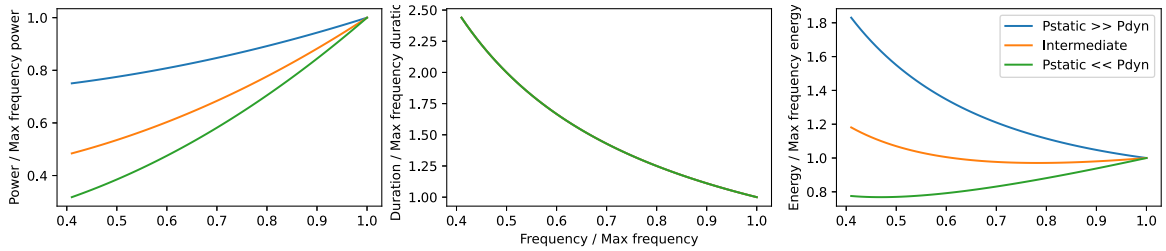


Fig. 1. Power, duration and energy for the classical model. Different ratio $Pstatic/Pdyn$ are displayed for a constant power consumption and duration when frequency is at its maximum. Minimum frequency is $0.4 \times Fmax$ in this example as most hardware architecture cannot go below this frequency.

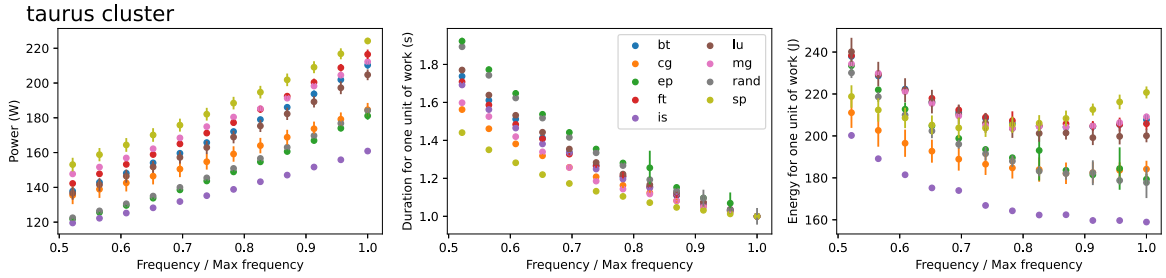


Fig. 2. Measure (power, performance, energy) on the `taurus` cluster of Grid'5000 of several benchmarks (one color for each different benchmark). The x-axis represents the ratio $f/Fmax$. Duration for each benchmarks is normalized relatively to the value at maximum frequency, hence the name *Unit of work*. Energy is the product of power and the normalized time. Each point represent the mean value of 10 executions, along with the standard error bar. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

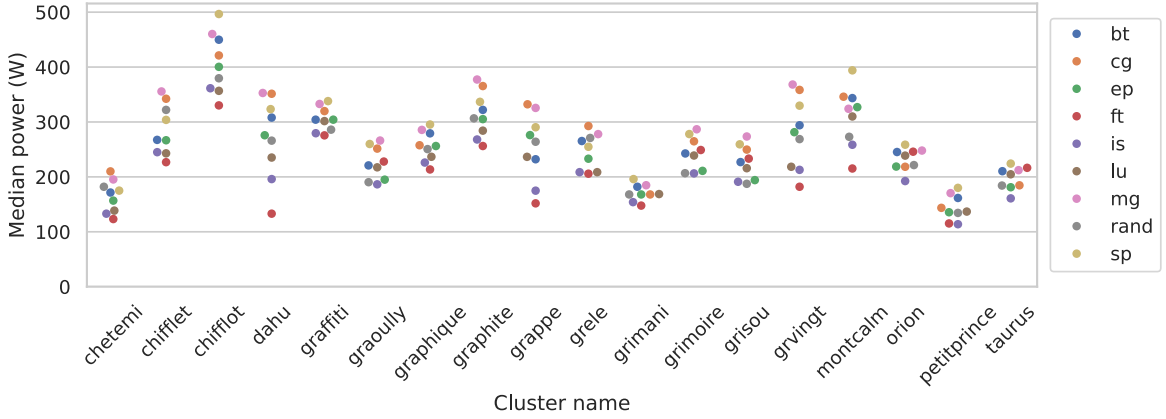


Fig. 3. Median power during the execution of each bench at maximum frequency of the servers. Slight jitter on the x axis has been added to visually separate the points.

- Applications do not react totally to DVFS from a power and duration point of view as they use other resources such as memory or I/Os;
- The impact of DVFS on duration and power depends on the application and on the underlying hardware.

3.2. Application related power model

Fig. 2 shows that contrary to the classical model depending only on the server characteristics, power and performance also depend on the particular application. The applications presented here are from the NAS Parallel Benchmark (`bt`, `cg`, `ep`, `ft`, `is`, `lu`, `mg`, `sp`) and an application (`rand`) that only makes a loop of calls to `rand()`, one thread per core.

Fig. 3 shows that the power behavior depends on both the application and the cluster. Depending on the cluster, the maximum power consuming benchmark changes. For example, `cg` consumes the most on `chetemi` cluster while being in the middle on `petitprince`.

To account for this, a possible model would be to associate one specific model for each combination of applications and clusters. Eq. (1) becomes:

$$Power_f^{c,a} = Pstatic^{c,a} + Pdyn^{c,a}\left(\frac{f}{Fmax^c}\right)^{Pcoef} \tag{4}$$

3.3. Application related performance model

Similarly as for the power model, Fig. 2 shows that the performance model cannot be simplified as a slow down proportional to the ratio of frequency. Our hypothesis is that for a HPC code, a percentage of the executed code will be impacted by the DVFS while another will not. We note $\alpha^{c,a}$ the ratio of the code execution that is impacted by the DVFS relative to the execution at maximum frequency. We call $Fluid$ the duration that is impacted by DVFS and $Rigid$ the part that is not. With $Duration^{c,a}_{Fmax^c} = Rigid^{c,a} + Fluid^{c,a}$ and with $\alpha^{c,a} = Fluid^{c,a}/Duration^{c,a}_{Fmax^c}$, we have the following behavior:

$$Duration_f^{c,a} = Rigid^{c,a} + Fluid^{c,a} \times \frac{Fmax^c}{f}$$

So Eq. (2) becomes:

$$Duration_f^{c,a} = \left((1-\alpha^{c,a}) + \alpha^{c,a}\frac{Fmax^c}{f}\right) \times Duration^{c,a}_{fmax} \tag{5}$$

4. Evaluation methodology

We compare the following models

- `Naive rand`: For reference purpose we use a classical model that aggregate the data-centers as an average one (even the maximum frequency is averaged) and uses a single application (`rand`, described below).
- `Naive all app`: Similarly to the `Naive rand` but with using all the applications (NAS Parallel Benchmark, `rand`) data aggregated as an average one.
- `Per Server rand`: One classical model per server, using data from `rand` for determining the constants.
- `Per Server all app`: One classical model per server, using averaged values for the applications.
- `Per App`: One classical model for each combination of servers and applications.
- `Fluid`: One model (using Eq. (5)) for each combination of servers and applications. This performance model uses the fluid ratio α.

The servers are from Grid'5000 [15] experimental platform: a large-scale and flexible testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing including Cloud, HPC and Big Data and AI. Each cluster is homogeneous. Different clusters are using different hardware and were built between 2013 and 2021. The servers are all being monitored by wattmeters with an accuracy between .1 to 10 W and a time from 1 s up to 7 s between measures depending on the cluster.

The benchmarks used are the NAS parallel benchmark (NPB) with size tuned running with a duration of around one minute : `is-D`, `cg-D`, `lu-C`, `ep-D`, `mg-D`, `ft-C`, `sp-C`, `bt-C`. There is also a random one (`rand` benchmark) which runs a constant number of calls to the `rand()` function on each core. It is used to evaluate the quality of a model obtained using measures on a simple benchmark compared to one based on more precise measures. The current models are only evaluated for full single servers applications. For NPB it means using the maximum possible number of cores for each benchmark. For `rand` it means one thread running series of rand for each core. The network is thus not used by any of the applications.

Each benchmark was run on each cluster with each available frequency 10 times. In the following a *run* will refer to running one benchmark on one cluster at one of the available frequencies. Also, an *experiment* will refer to a series of consecutive runs of one benchmark on a single server of a cluster at all frequencies. Each of these runs will be monitored independently, but it guarantees that this experiment is done on the same computer for all these frequencies in the same environment. Different experiments on a single cluster might have run on different servers. Each *run* is started after an identical pattern is executed on the server for two purposes: first it allows to match the external power measure with the system measures at a higher precision; second it guaranties that all run will start in similar temperature conditions.

Data are obtained using Expetator[1] version 0.3.20. It is a tool for running HPC benchmarks using several type of leverages (DVFS) and low-level monitoring (hardware performance counters, RAPL — Running Average Power Limit) mostly on Grid5000. It packages benchmarks including the NAS Parallel Benchmarks and the random benchmark, a monitoring system including external power consumption and hardware performance counters, and a large number of leverages including DVFS. Depending on the cluster, the external power measure is done either every second or every five seconds. Using the methodology from [16], the applications are started with peak of CPU usage just before and after their execution to match the system measures and the power measures which are obtained on different servers with different latency. These peaks are removed before the analysis of the data.

This tool relies on external monitoring system for the system-level measures. For the measures on hardware performance counters, the tool used is MojitO/S [17]. It is an Open Source System, Energy and Network Monitoring Tools at the O/S level running on GNU/Linux. It allows to monitor classical system metrics such as load, network, but also memory layers, RAPL values and hardware performance counters. These counters inside the processor can be used to count the number of cache misses, instructions, arithmetic operations, among others.

For each model, the constants relative to the model have been extracted from the measures. For example, $Duration_{Fmax}$ is obtained using the mean duration for all the applications on all the servers for the *Naive* model and thus is constant for all clusters and applications. $Duration^{c,a}_{Fmax^c}$ for the Per-app model is obtained using the ten runs for each configuration cluster (c)/benchmark (a), resulting on one value for each of these couple.

The main metric for comparison is the MAPE (Mean Absolute Percentage Error) computed between the measured values and the one predicted from the model. Another metric will be the amount of data needed to compute the constants associated with each model along with the number of these constants. Eq. (6) shows the calcul of MAPE between n measured (M_i) and estimated values (E_i). This value is a percentage with a lower value, the better.

$$MAPE = \frac{1}{n}\sum_{i=0}^{i=n-1}\left|\frac{M_i - E_i}{M_i}\right| \times 100 \tag{6}$$

Table 2
MAPE for the different models. Mean measured *Power* over all servers is 229 W, with a first quartile of 176 W to a third quartile of 280 W. For *Duration* these values are 1.2, 1.1 and 1.4 while for *Energy* they are 274J, 213 and 317.

Name	Power	Duration	Energy
Naive rand	24.30	17.35	28.82
Naive all app	26.83	17.35	35.13
Per Server rand	13.62	14.53	15.31
Per Server all app	15.40	14.53	23.06
Per App	8.23	14.53	21.48
Fluid	8.23	2.05	8.55

5. Comparison of the models

5.1. Reproducibility

The whole dataset of power and system measures (approx 7Go) is publicly available [18]. All the script to redo the analysis, and to generate the models are available[2]

5.2. Evaluation of the quality of the models

The comparison of the models are obtained using the overall MAPE value (Table 2 which uses the duration relatively to the duration at maximum frequency), and more precisely the difference between their precision (Fig. 4). The normalization using the duration at maximum frequency is always computed after the model's prediction, only for the sake of simplification of the figures. With this, the mean duration at maximum frequency of all benchmark is 1. It can be considered as a unit of work, a speed-up to ease the understanding.

Table 2 shows the error between the `Naives` and the `Per Server` power models are divided by two. Intuitively taking into account the heterogeneity of the hardware already allows to go from a MAPE of more than 24% to one just over 13%. Actually, taking into account the

[1] https://gitlab.irit.fr/sepia-pub/expetator.

[2] https://gitlab.irit.fr/sepia-pub/open-science/ppe-models-for-hpc-servers-dvfs.

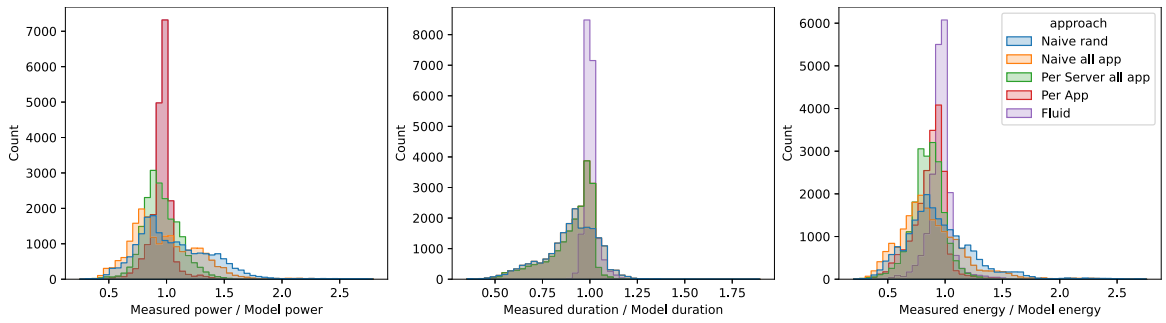


Fig. 4. Comparison of the models for estimated power, duration, and energy relatively to the measured values. The histograms use 50 bins of constant width. `Per App` and `Fluid` use the same power model, so their power histograms overlap perfectly.

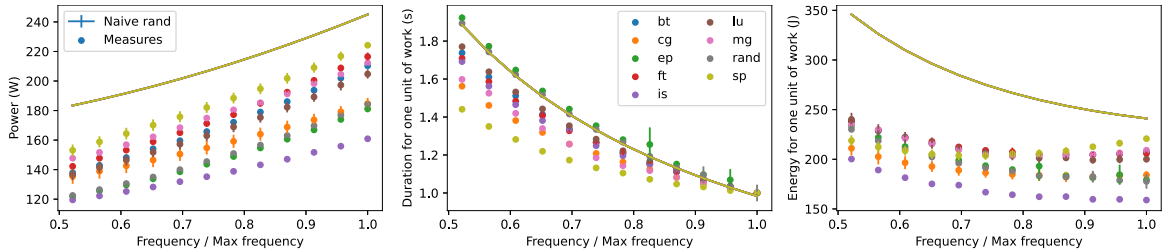


Fig. 5. `Naive rand` model (lines) using Eqs. (1) and (2) with measures (dots) on `taurus` cluster using only values from the `rand` benchmark to obtain the constants of the model. Idle power consumption of `taurus` servers is 92 W.

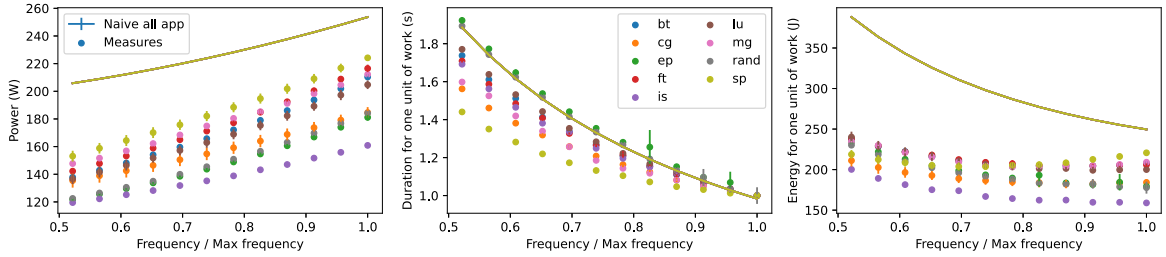


Fig. 6. Same as 5 but using data from all benchmarks to compute the constants of the model.

difference of power profile from the different applications (the `Per App` model) allows to increase the precision with a MAPE on power below 10%. It validates our hypotheses on the difference between the power consumption of different applications. Even if each of these applications fully use the servers, their maximum power consumption and the impact of DVFS on their power consumption is different and can be modeled. Finally, using a precise model for the duration (the `Fluid` model) allows to reduce the MAPE error on duration from more than 14% down to 2%.

Fig. 4 shows an histogram of the ratio between the measured values compared to the different models. As the `Per App` and the `Fluid` model only differ on the duration model, their power histograms overlap. On this figure a perfect model would only have values counted for a ratio of one. This figure shows the same increase in quality as the previous table, but also show that the deviation of the results follows the same improvement. We can also see that all the models are slightly skewed toward an overestimation of the values.

Figs. 5, 6, 7, 8, 9 and 10 shows the comparison between the values of the different models and the actual measures on the `taurus` cluster.

Figs. 5 and 6 show that using an identical model for all clusters leads to a power model which is not relevant. This case is comparable to use a single model in a highly heterogeneous system. In this particular case, as the server is consuming relatively less energy than the other, this model overshoot the power. Also as the model does not depends on the application, it only fits well cpu-intensive applications such as `rand` and `ep`.

Figs. 7 and 8 show that using one power model per cluster is more relevant. Using only the `rand` benchmark (Fig. 7) leads to a final model relevant for cpu-intensive applications, while using all applications (Fig. 8) leads to a more average model. It is coherent with the error shown in Table 2.

Fig. 9 shows that using one power model per application allows to precisely model their power. As shown in Table 2, the error on the power estimation is reduced by 30%. For example, on the power part, the differences between the dynamic of `cg` and `ep` are clearly visible.

The `Fluid` model extends to the duration the concept of one model per application. Fig. 10 shows the same quality on the power model as the previous one (they are using the same equations). It also shows that using the proposed α is a relevant way to model the differences between the application behaviors. On this precise example, the dynamic of the benchmarks is well followed. Similarly the impact of DVFS on the performance of tasks is accurately modeled. Nevertheless, as the energy is the product of power and time, the slight differences on both have a larger impact on energy. Still, the overall dynamic such as respective amount of energy, optimal frequency ratio to minimize energy, are well modeled.

5.3. Heterogeneity of the error

As the measure quality depends on the particular cluster, the resulting models precision is also linked to this quality. Fig. 11 shows the error (MAPE) for power, duration, and energy for each cluster.

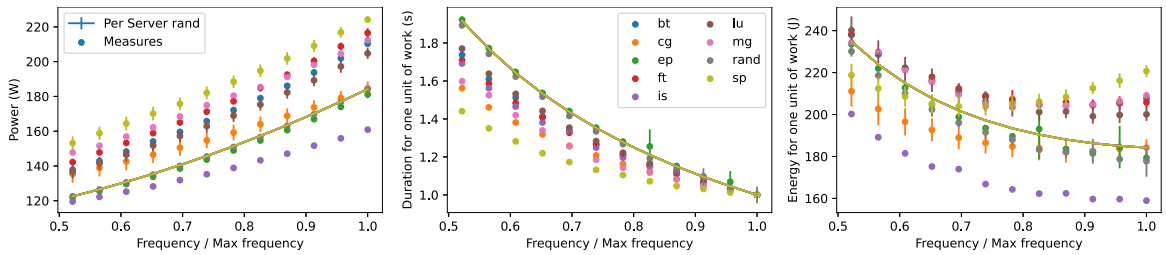


Fig. 7. Same as 5 but with one model per cluster and the data used to obtain the constants of the model come from monitoring only the `rand` benchmark.

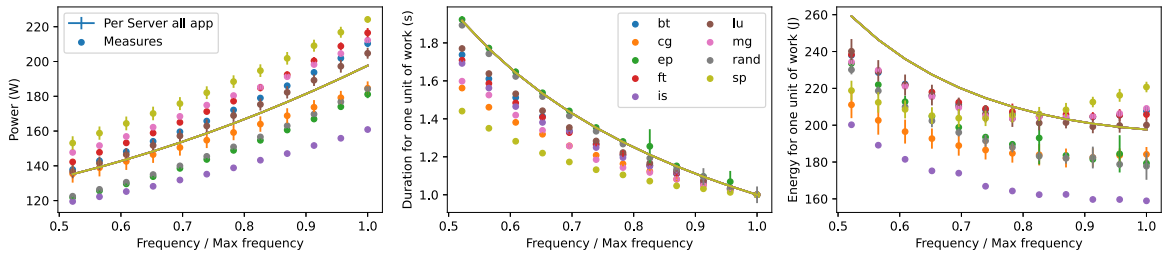


Fig. 8. Same as 5 but with one model per cluster and the data used to obtain the constants of the model come from monitoring all benchmarks.

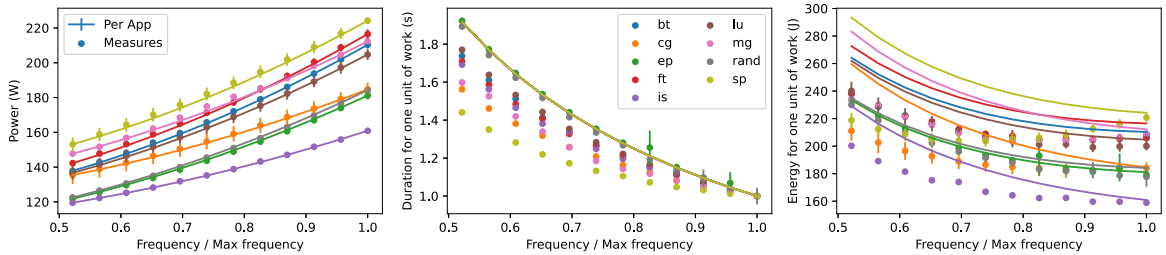


Fig. 9. Same as 5 but with one model per combination of clusters and benchmarks. The monitored data of each benchmark on all clusters are individually used for each of these models following Eq. (4).

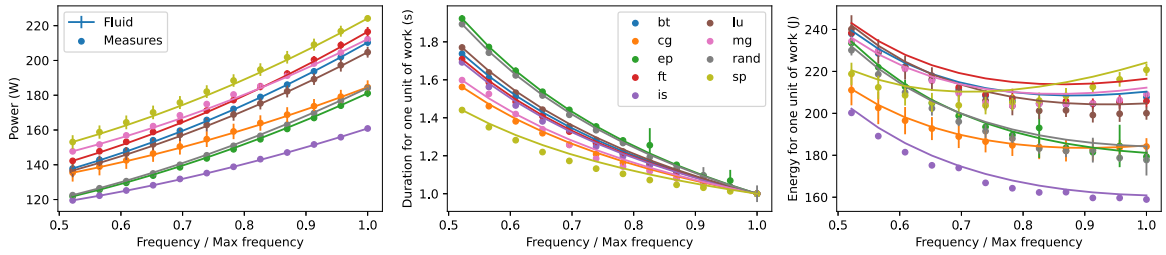


Fig. 10. Evaluation of the `Fluid` model (lines) using Eqs. (4) and (5) with measures (dots) on `taurus` cluster.

They have been sorted depending on the precision on the measures. The precision has been estimated by using the standard deviation of the measures on each cluster. It shows that for all clusters the `Fluid` model is the most precise. It also shows that on the clusters with an higher precision of the measures, the model is more accurate.

5.4. Measurement needed for each model

`Naive rand` model: The values needed to compute the `Naive rand` model constants are the following:

- $Fmax$: the mean value of the maximum frequency across all clusters;
- $Fmin$: same methodology as $Fmax$ but for the minimum frequency of each cluster;
- $Power_{Fmax}$: the mean value of power consumption of the `rand` benchmark at maximum frequency for all clusters;
- $Power_{Fmin}$: similarly for the power consumption of the `rand` benchmark at minimal frequency.

`Naive all app` model: The constants are computed in the same way as in the `Naive rand` model, except that the data used for $Power_{Fmax}$ and $Power_{Fmin}$ comes from the average of the monitored values from all benchmarks instead of coming only from the one monitored only from the `rand` benchmark.

`Per Server rand` model: The values needed to compute the `Per Server rand` model constants are only coming from the monitoring values from the `rand` benchmark but are dependent on the cluster:

- $Fmax^c$, $Fmin^c$, $Power^c_{Fmin}$, $Power^c_{Fmax}$: These values are computed in the same way as for the `Naive rand` model, but independently for each cluster. As an example, $Fmax^c$ is the maximum frequency of servers in the cluster c. As a reminder, we assume that servers in the same cluster are homogeneous.

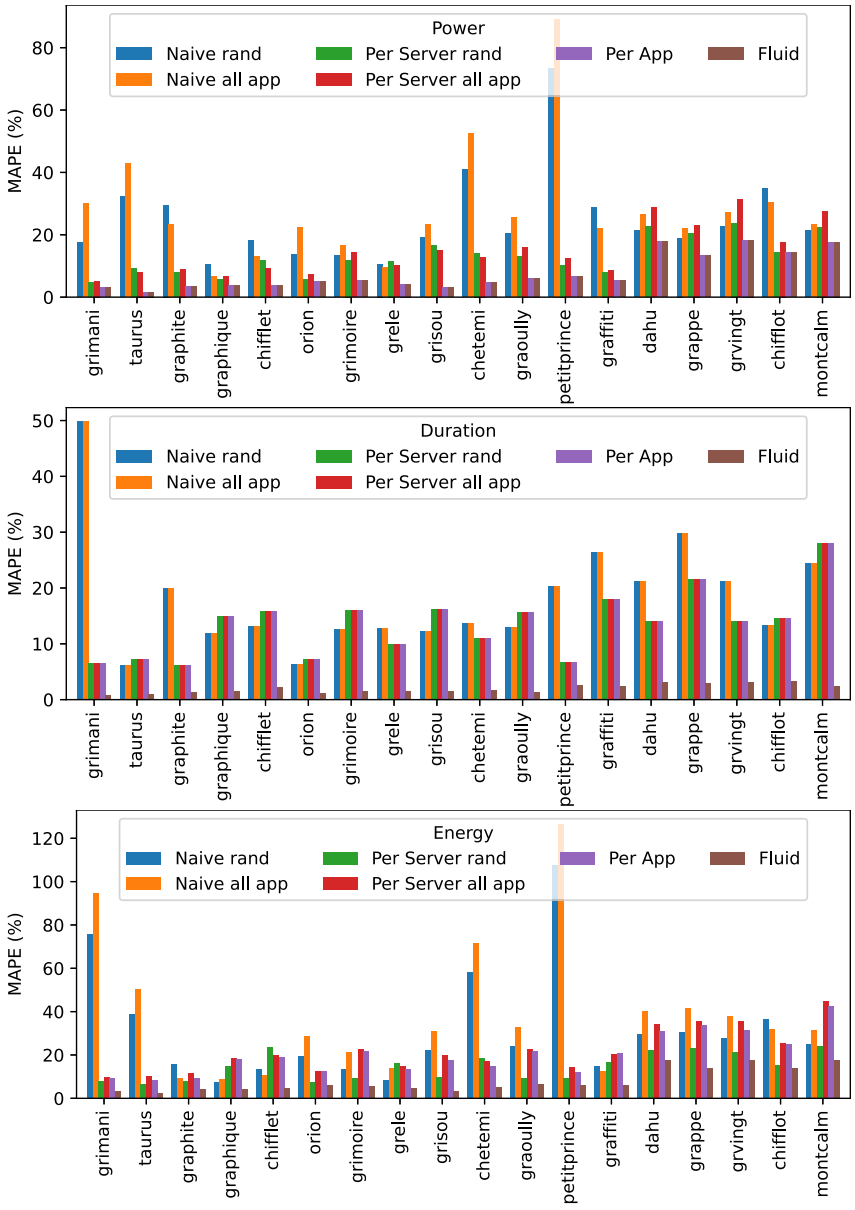


Fig. 11. Error (MAPE) for the different models on each cluster. The clusters have been sorted from the one with the most precise measures (`grimani`) to the one with the lower quality measures.

`Per Server all app` model: Similarly as the difference between `Naive rand` and `Naive all app`, the values $Power^{c}_{Fmin}$, $Power^{c}_{Fmax}$ are computed by averaging monitored values from all benchmarks.

`Per App` model: There is one model instance for each combination of clusters and benchmarks. Thus, the monitored values come from all benchmarks:

- $Fmax^{c}$, $Fmin^{c}$: are the same as in the previous model;
- $Power^{c,a}_{Fmin}$, $Power^{c,a}_{Fmax}$: are the same as in the previous model except they are specific for each combination of clusters (c) and applications (a).

`Fluid` model: The main difference between the `Fluid` model and the `Per App` one concerns a more precise model of the performance impact of DVFS. To compute the rigid and fluid part of the duration, and thus the α linked to a combination (cluster, application), the speed-up between minimum and maximum frequency is needed. Thus the monitored values are:

- $Fmax^{c}$, $Fmin^{c}$, $Power^{c,a}_{Fmin}$, $Power^{c,a}_{Fmax}$: are the same as in the previous model
- $Duration^{c,a}_{Fmax}$: is the duration at maximum frequency of executing application a on cluster c.
- $Duration^{c,a}_{Fmin}$: is the duration at minimum frequency of executing application a on cluster c.

5.5. Usage of the `Fluid` model

One way to use an energy model is to find the best frequency for a particular benchmark on a particular cluster to minimize energy.

By reusing Eqs. (4) and (5), we can use the fact that energy is the product of duration and power, and that the optimal energy is a local minimum. Thus the optimal energy is obtained when the derivative of energy is zero. This is equivalent to:

$$Coef\frac{1-\alpha}{\alpha}\left(\frac{f}{Fmax}\right)^{Coef+1}+(Coef-1)\left(\frac{f}{Fmax}\right)^{Coef}-\frac{Pstatic}{Pdyn}=0 \tag{7}$$

Table 3
Difference (in percentage) of the optimal energy for each model compared to the energy consumed at maximum frequency. Each experiment (a run of one benchmark on a cluster for all frequencies) provides one value.

Cluster	Optimal	Per Server rand	Per App	Fluid
All clusters	−6.16	−0.40	−0.73	−1.79
chetemi	−2.53	0.00	0.00	−0.27
chifflet	−3.00	0.00	0.00	−1.75
chifflot	−8.64	0.00	0.00	−0.59
dahu	−19.78	1.68	−4.63	−7.38
graffiti	−2.40	0.00	0.00	−0.03
graoully	−2.61	0.00	0.00	−2.04
graphique	−0.98	0.00	0.00	−0.30
graphite	−0.31	0.00	0.00	0.00
grappe	−19.08	0.00	0.00	−0.53
grele	−1.97	0.00	0.00	0.01
grimani	−0.55	0.00	0.00	−0.15
grimoire	−2.20	0.00	0.00	−1.39
grisou	−3.76	0.00	0.00	−1.65
grvingt	−22.40	0.05	−4.02	−3.20
montcalm	−16.70	−8.87	−4.40	−10.18
orion	−1.11	0.00	0.00	−0.83
petitprince	−1.03	0.00	0.00	−0.55
taurus	−1.90	0.00	0.00	−1.41

Table 4
Values of α for all clusters and benchmarks combinations.

Bench cluster	bt	cg	ep	ft	is	lu	mg	rand	sp
chetemi	0.73	0.26	1.02	0.68	0.64	0.75	0.28	1.05	0.25
chifflet	0.59	0.28	1.00	0.52	0.58	0.56	0.15	1.00	0.14
chifflot	0.75	0.22	1.00	0.65	0.44	0.83	0.42	0.97	0.49
dahu	0.81	0.28	0.99	0.66	0.40	0.81	0.12	1.00	0.36
graffiti	0.73	0.22	0.96	0.63	0.41	0.82	0.28	1.01	0.30
graoully	0.66	0.18	1.02	0.62	0.50	0.77	0.09	1.11	0.11
graphique	0.65	0.34	0.99	0.58	0.55	0.67	0.18	1.00	0.10
graphite	0.83	0.67	0.98	0.79	0.75	0.89	0.62	0.96	0.47
grappe	0.74	0.31	0.94	0.51	0.26	0.69	0.12	1.00	0.19
grele	0.76	0.32	0.99	0.66	0.68	0.71	0.34	1.04	0.36
grimani	0.67	0.29	1.02	0.61	0.44	0.72	0.20	1.00	0.13
grimoire	0.66	0.18	0.99	0.62	0.50	0.76	0.09	0.95	0.11
grisou	0.67	0.20	0.91	0.64	0.50	0.76	0.09	0.96	0.10
grvingt	0.81	0.27	1.00	0.65	0.40	0.81	0.13	1.00	0.36
montcalm	0.67	0.24	1.13	0.43	0.24	0.59	0.03	1.00	0.16
orion	0.81	0.61	1.04	0.77	0.76	0.84	0.65	1.01	0.47
petitprince	0.77	0.59	0.96	0.81	0.76	0.77	0.70	0.93	0.45
taurus	0.80	0.61	1.01	0.77	0.75	0.84	0.65	0.97	0.48

If $Coef$ is equal to 2 like in the experiments below, it becomes a degree three equation in function of f which can be solved analytically. Otherwise, due to the low number of available frequencies, it is possible to test all of them and to choose the one optimizing the energy.

In Table 3, we compare the energy using the frequency minimizing the frequency (`Optimal`), or the one using the models. `Naive rand`, `Naive all app`, and `Per Server all app` are always 0 as they always returns the maximum frequency when optimizing the energy. The highest gain is for the cluster `montcalm` with a gain of 10% out of the maximum reachable of 16%.

5.6. Analysis of the properties of alpha

Table 4 shows the measured values of $\alpha^{c,a}$ for each clusters and applications using the duration measured at maximum and minimum frequencies.

Concerning the benchmarks, except for `rand` and `ep` which are almost identical, the values are quite different for the different benchmarks. First there is no global order between them. For example, depending on the cluster, `mg` and `sp` respective rankings change. Still there are tendencies for the α related to the benchmarks. CPU intensive benchmarks such as `ep` and `rand` are fully responsive to DVFS (value of alpha equal to 1). Other memory intensive benchmarks [19] such as `mg`, `sp` or `cg` have overall lower α values, consistent with their lower usage of the CPU.

Concerning the clusters, the one with similar processors have similar values. For example, `dahu` and `grvingt` are both Intel Xeon Gold 6130, or `orion` and `taurus` are both Intel Xeon E5-2630. Still, different hardware have different reaction to DVFS. For example `petitprince` (using Intel Xeon E5-2630L from Q1'12) is quite different from `grvingt` (using Intel Xeon Gold 6130 from Q3'17) or `montcalm` (using Intel Xeon Silver 4314 from Q2'21).

6. Online detection of the constants

While the `Fluid` model allows to correctly predict Power, Duration and Energy, it still needs to have a-priori information on the application, particularly its behavior at minimum and maximum frequencies. If used in a production system, it also actually needs to be able to know which is the current running benchmark, which might be difficult as usually the system runs applications as black boxes. It also prevent managing an application which has never been registered and monitored before.

For online algorithms, finding the values for Eqs. (7), should be simpler. The values needed are either solely dependent on the hardware ($Fmax$, f), or of the combination of hardware and applications ($Pstatic^{c,a}$, $Pdyn^{c,a}$, $\alpha^{c,a}$).

As the link with the application is actually a link to the resource consumption of the application, it is possible to model these values using online resource usage. As an example, we use several low level Hardware Performance Counters available in most recent Linux systems. In the current dataset, the following counters are available: `cache_misses`, `cache_references`, `instructions` and `branch_instructions` with a frequency of 10 Hz. These counters were selected to provide the most generic set. As hardware span from 2013 to 2021, the counters are the generic one of the Linux Kernel which are available on all type of hardware, without the need of any specific library, ensuring reproducible results. The number of counters (four) comes from the classical number of hardware registers used to store these performance counters inside the processors. With more counters, the operating system starts multiplexing these counters, leading to a decreased precision and difficulties on older types of hardware. These four counters span the different categories of the standards Linux counters. At the same frequency, we have access to RAPL [20] counters that give access to the energy consumption of the processor and memory. In the following the `rapl` counters will be the sum of the power consumption of the processor and memory obtained through their regular energy monitoring. As the power, duration and energy evaluation of the applications are done for a run of the application, we associate for each run one value for each hardware performance counter. To reduce the impact of the application starting and finishing phases, we associate one *run* with the median values of the performance counters. For example, in the following, the use of `cache_references` is used for the median value of this hardware performance counter during the whole execution of the application for one particular *run* in a cluster at a certain frequency.

To minimize the impact of the online model detection, we only used linear regression using these 5 hardware performance counters for each $Pstatic$, $Pdyn$ and α. We harness the hypothesis that these values depend on the resource usage, and that the hardware performance counters provide a relevant evaluation of this resource usage.

For each of the three values ($Pstatic$, $Pdyn$, and α) there are 6 associated constants, five for the weight for each counter, and one for the intercept.

To obtain these values, following Eqs. (4) and (5), we only need monitoring information from two out of all the frequencies. All the data used to obtain the constants will come from monitoring the benchmarks at the highest and lowest frequency on each cluster.

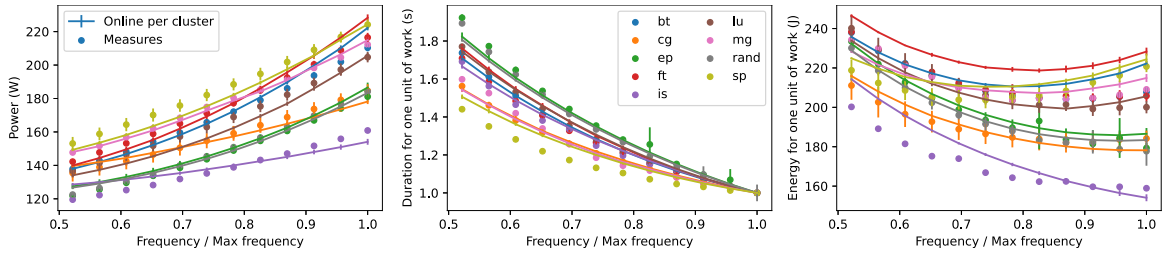


Fig. 12. Power, duration and energy for the `Online per cluster` model compared to measures on the `taurus`cluster.

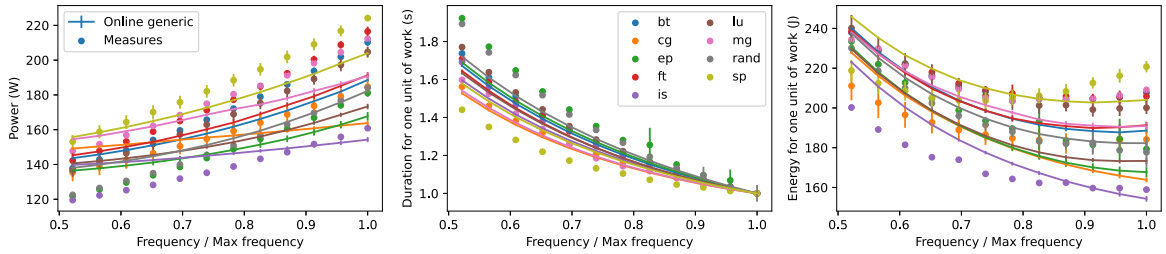


Fig. 13. Power, duration and energy for the `Online generic` (i.e., the same constants for all clusters) compared to measures on the `taurus`cluster.

Table 5
MAPE for the online methods compared to the references (`Native Rand`, `Fluid`) using all the data for learning. The error is evaluated on all runs (i.e. including all frequencies). For `Online generic BT CG`, only data from these two benchmarks are used for learning.

Name	Power	Duration	Energy
Naive rand	24.30	17.35	28.82
Fluid	8.23	2.05	8.55
Online per cluster	8.99	3.32	9.70
Online generic	11.34	4.85	12.91

Each of the *experiments* contains an execution of a benchmark for each frequency on a single computer of a cluster. In the following, for the training, we select some of these *experiment*, and keep only the data related to the maximum and minimum frequency. So selecting 10% of the runs would lead to use less than 2% of the total number of *runs*, on average, the clusters have 12 frequencies.

We will evaluate two models:

- `Online per cluster`: The linear models for each values ($Pstatic$, $Pdyn$, and α) will be dependent on the cluster. As we have 18 clusters, we will have $18 \times 6 \times 3$ constants as there are 3 linear model, each with 6 constants, for each cluster. The constants are obtained using a linear regression using for each *experiments* the data from the *runs* limited to maximum and minimum frequency.
- `Online generic`: The methodology is the same as previously, but instead of having one equation set per cluster, there is only one for all the clusters. The only difference is that the power model is computed after removing the mean idle power consumption of each cluster. In this case, there are only 6×3 constants.

6.1. Accuracy of the online models

A direct evaluation of the error is shown in Table 5. The evaluation of the error is made with all data, even the ones with frequency not used for learning (i.e. $Fmax$ and $Fmin$). So only a part (one twelves as there are on average 12 frequencies per clusters) of the data has been used during the linear regression.

While being with a lower precision than the `Fluid` model, these models have error twice smaller than of the classical models such as `Native Rand`.

Figs. 12 and 13 show the prediction of these two models compared to the actual monitored values. As shown in Table 5, the cluster specific model is more precise than the generic one. Nevertheless, for a way smaller number of constants, both models are quite similar to the `Fluid` model with only hardware performance counter measures. There is also no need to know beforehand which benchmark is running.

Fig. 14 shows the histogram of values for these performance counter-based models.

6.2. Heterogeneity of the error

With the same method as on Fig. 11, Fig. 15 shows the error (MAPE) for power, duration, and energy for each cluster. Models `Naive rand` and `Fluid` are present for comparison purpose. The two online models show comparable models without a-priori knowledge on the application. The error `Online per cluster` is mostly comparable with the `Fluid` model for Power and Duration. The `Online generic` model has a slightly lower quality but with a simpler model and while being largely more precise than the `Naive rand` model. The `Online generic BT CG` is the online generic model where all the learning data come from monitoring only BT and CG. It shows that the results are also relevant for applications that have never have been used for the learning.

6.3. Obtaining the constants

Table 5 shows the MAPE for a best case, where all the data are available to compute the constants with linear regression, before evaluating the error on the same dataset. Thus it uses data from all experiments. In the remaining of this section, we will evaluate the impact of the training dataset size.

Fig. 16(a) presents the MAPE of Power, Duration and Energy compared to the `Fluid` model. The whole dataset is split between a training set of experiments (which ratio is in the x axis) and a testing set of experiments. The data presented are the comparison between the `Fluid` model using all data to compute its constants, and the `Online per cluster` model using only the training set. The comparison of both models is only done on the remaining data, ie the testing set. As only data from runs at maximum and minimum frequencies are used for the linear regression of the online model, the training set keeps only data related to runs at minimum or maximum frequency. The testing set contains both the runs of the training experiments with frequencies

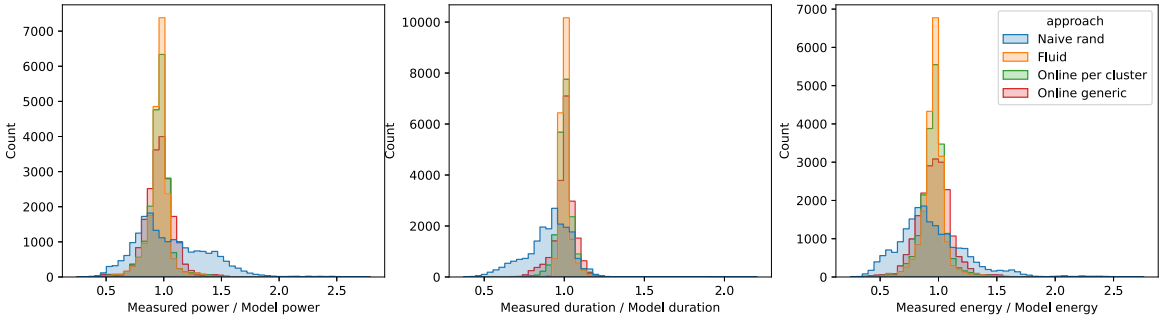


Fig. 14. Comparison for estimated power, duration, and energy relatively to the measured values for the references (`Naive rand`, `Fluid`) and online methods using no application-related information.

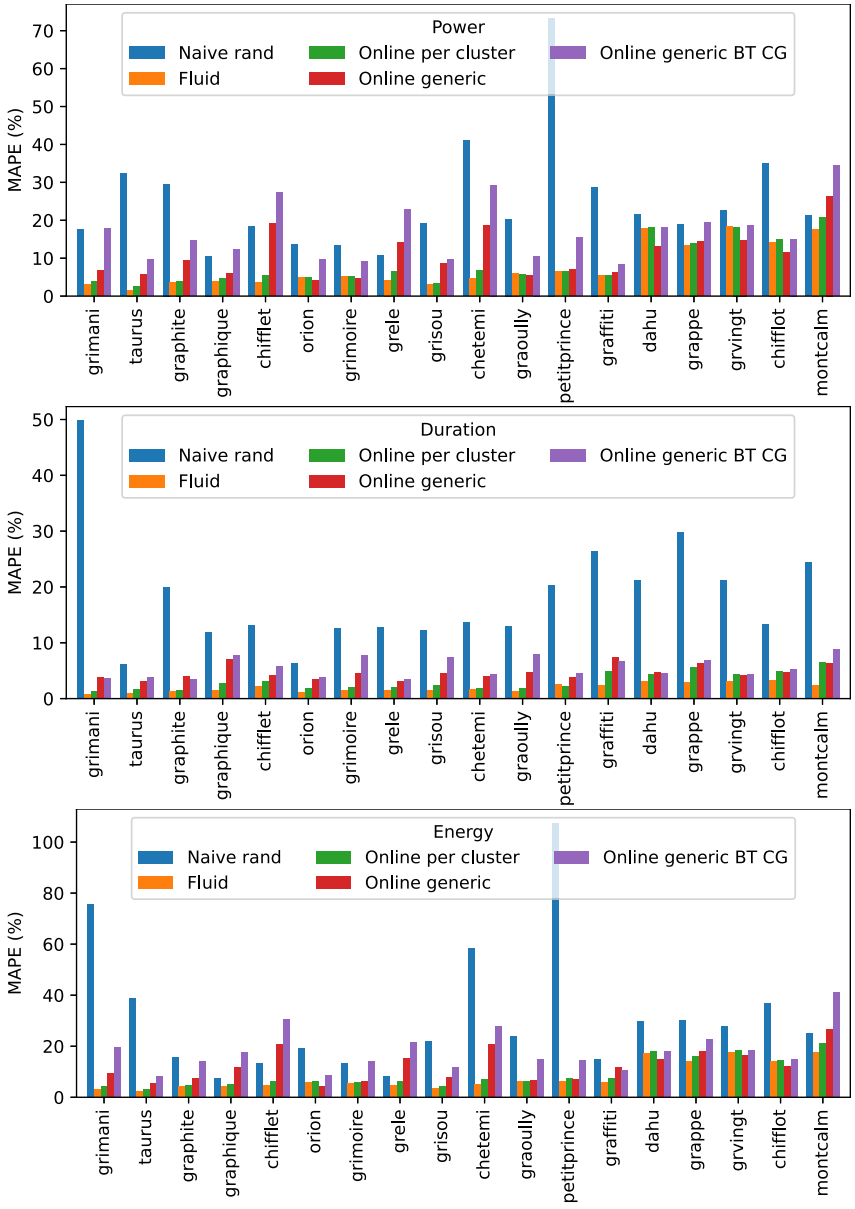
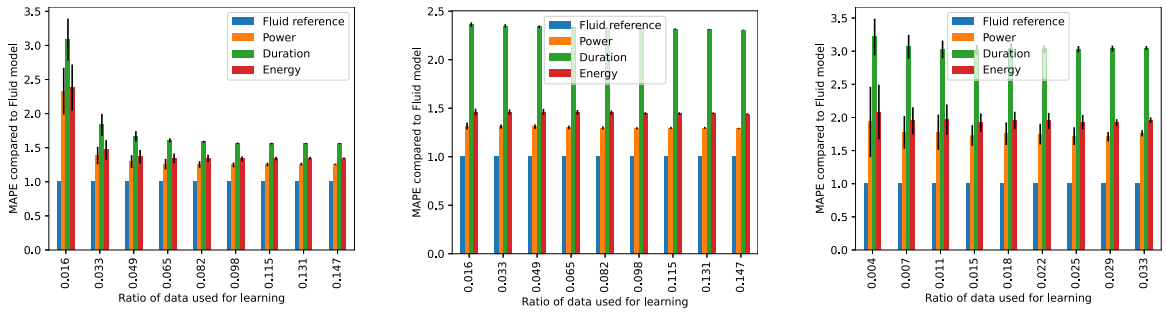


Fig. 15. Error for the different online models on each cluster. The clusters have been sorted from the one with the most precise measures (`grimani`) to the one with the lower quality measures.

different from $Fmax$ and $Fmin$, along with all the experiments which are not selected for the training set. As only 2 frequencies out of the average 12 are used, 10% of the experiments represent only 10%/6 = 1.6% of all the runs when keeping only the minimum and maximum frequencies. Similarly, 90% of the experiments represents 14.7% of the total number of runs.



(a) Using only the data measured at the highest and lowest frequency for the learning of `Online per cluster` model.

(b) Using only the data measured at the highest and lowest frequency for the learning of `Online generic` model.

(c) Same as b with restricting the training set to only two benchmarks: `bt` and `cg`.

Fig. 16. MAPE compared to the `Fluid` model (Table 5). The ratio represent the amount of data used of the learning part compared to the total available data. It corresponds to 10% up to 90% of experiments. For example in a, 10% of experiments means $2 \times 10\%/12 = 1.6\%$ of runs as for each experiment we use only the runs with minimum and maximum frequency out of the 12 availables frequencies in average.

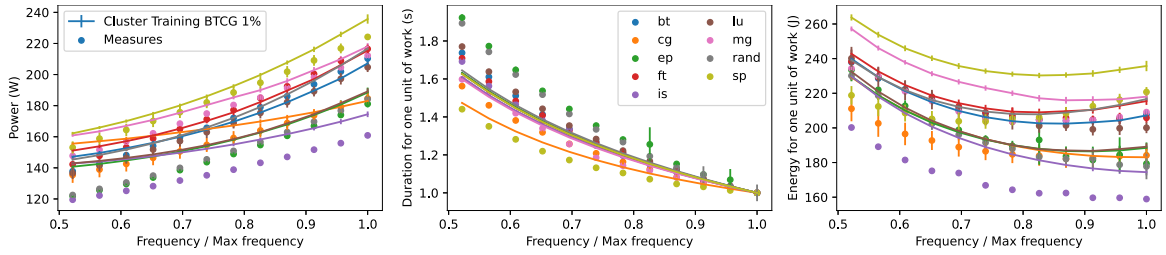


Fig. 17. Power, duration and energy for the `Online generic` model using only data from `bt` and `cg` for learning compared to measures on the `taurus` cluster. The training dataset represent 1% of the available data.

Fig. 16(a) shows that even with 10% of the data, the accuracy is stable, representing an MAPE 25% (resp. 56% and 34%) higher than the `Fluid` model for the power estimation (resp. duration and energy).

Fig. 16(b) shows the same method but using the `Online generic` model. The main difference is that as the number of constant to compute of 18 times lower, the model accommodates better with a smaller dataset. With 10% of the data, its MAPE is 30% (resp. 32% and 45%) higher than the Fluid model. Results on the power and energy models are quite similar to the `Online per server` model while having a way lower complexity. The duration model still has higher difference, but the `Fluid` model is actually very precise and Table 2 shows that the second best model has a duration MAPE 7 times higher than the `Fluid` model.

Fig. 16(c) shows the same method, for the `Online generic` model, but with restricting the benchmarks to only `bt` and `cg`. In actual data centers, all applications are not necessarily available for training the models. With 1% of the data, the MAPE is 78% (resp. 203% and 97%) higher than the `Fluid` model. Once again, with a very limited training dataset, using only two of the available frequencies, and two of the available benchmarks, the resulting model is quite precise. Fig. 17 shows that even if the values are not exactly the same, the dynamic and the diversity of the behaviors is still present.

6.4. Overhead of the online method

To evaluate the overhead of the online methods execution, we evaluated the cost of accessing the four performance counters along with RAPL. On montcalm cluster, using mojitO/S, mean time cost for these hardware performance counters including RAPL is 16 ms. This tool uses only one thread, so the cost is for a single core. Using the online method with these hardware performance counters and RAPL could be used during the execution of HPC applications with a negligible overhead as the remaining of the computation is the linear product of these values with the constants obtained during the learning phase.

7. Conclusion

We proposed a simple yet precise `Fluid` model to account for DVFS impact in heterogeneous HPC systems. We validated this model using an extensive amount of experiments using several benchmarks and diverse computing hardware. We also showed that taking into account the application for the power and performance models is important. We also showed how to use this model to obtain the optimal frequency. Finally we extended our model to the online case with a low overhead approach and with only hardware performance counters monitoring.

Still, the findings are currently limited to a single-server approach. The overall approach can be extended to multi-servers applications, but the online detection method should be adapted to include the network monitoring values.

Also more studies are needed to deduce the values used in the models for one server using information related to their micro-architecture and memory structure.

The next step would be to make use of these models in a production platform to change the frequency of the processor in real-time depending on the monitored values of the hardware performance counters, and to evaluate the impact of the approach for applications having several phases.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Georges Da Costa reports financial support was provided by French National Research Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work presented in this paper has been funded by the ANR in the context of the project ENERGUMEN, ANR-18-CE25-0008 and the France 2030 NumPEx Exa-SofT (ANR-22-EXNU-0003) project managed by the French National Research Agency (ANR) Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

Data availability

A zenodo link is in the article. All data are available under an open license.

References

[1] M. Etinski, J. Corbalan, J. Labarta, M. Valero, Understanding the future of energy-performance trade-off via DVFS in HPC environments, J. Parallel Distrib. Comput. 72 (4) (2012) 579–590, URL https://www.sciencedirect.com/science/article/pii/S0743731512000172.

[2] Anne Benoit, Louis-Claude Canon, Redouane Elghazi, Pierre-Cyrille Heam, List and shelf schedules for independent parallel tasks to minimize the energy consumption with discrete or continuous speeds, J. Parallel Distrib. Comput. 174 (2023) 100–117.

[3] Chung-Hsing Hsu, Ulrich Kremer, The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction, in: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, 2003, pp. 38–48.

[4] Vincent W Freeh, David K Lowenthal, Feng Pan, Nandini Kappiah, Rob Springer, Barry L Rountree, Mark E Femal, Analyzing the energy-time trade-off in high-performance computing applications, IEEE Trans. Parallel Distrib. Syst. 18 (6) (2007) 835–848.

[5] Mario Bambagini, Mauro Marinoni, Hakan Aydin, Giorgio Buttazzo, Energy-aware scheduling for real-time systems: A survey, ACM Trans. Embed. Comput. Syst. (TECS) 15 (1) (2016) 1–34.

[6] Georges Da Costa, Jean-Marc Pierson, DVFS governor for HPC: Higher, faster, greener, in: 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, IEEE, 2015, pp. 533–540.

[7] Srinivasan Ramesh, Swann Perarnau, Sridutt Bhalachandra, Allen D Malony, Pete Beckman, Understanding the impact of dynamic power capping on application progress, in: 2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS, IEEE, 2019, pp. 793–804.

[8] Mohak Chadha, Michael Gerndt, Modelling dvfs and ufs for region-based energy aware tuning of hpc applications, in: 2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS, IEEE, 2019, pp. 805–814.

[9] Patricia Arroba, José L Risco-Martín, Marina Zapater, José M Moya, José L Ayala, Katzalin Olcoz, Server power modeling for run-time energy optimization of cloud computing facilities, Energy Procedia 62 (2014) 401–410.

[10] Enrico Calore, Alessandro Gabbana, Sebastiano Fabio Schifano, Raffaele Tripiccione, Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications, Concurr. Comput.: Pr. Exp. 29 (12) (2017) e4143.

[11] Bo Su, Junli Gu, Li Shen, Wei Huang, Joseph L Greathouse, Zhiying Wang, PPEP: Online performance, power, and energy prediction framework and DVFS space exploration, in: 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, IEEE, 2014, pp. 445–457.

[12] Georgios Keramidas, Vasileios Spiliopoulos, Stefanos Kaxiras, Interval-based models for run-time DVFS orchestration in superscalar processors, in: Proceedings of the 7th ACM International Conference on Computing Frontiers, 2010, pp. 287–296.

[13] Fabio Diniz Rossi, Mauro Storch, Israel de Oliveira, César AF De Rose, Modeling power consumption for DVFS policies, in: 2015 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2015, pp. 1879–1882.

[14] Jason Mair, Zhiyi Huang, David Eyers, Leandro Cupertino, Georges Da Costa, Jean-Marc Pierson, Helmut Hlavacs, Power modeling, Large- Scale Distrib. Syst. Energy Effic.: Holist. View (2015) 131–158.

[15] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, Luc Sarzyniec, Adding virtualization capabilities to the Grid'5000 testbed, in: Cloud Computing and Services Science, Vol. 367, Springer International Publishing, 2013, pp. 3–20.

[16] Georges Da Costa, Jean-Marc Pierson, Leandro Fontoura-Cupertino, Mastering system and power measures for servers in datacenter, Sustain. Comput.: Inform. Syst. 15 (2017) 28–38, URL https://www.sciencedirect.com/science/article/pii/S2210537917300318.

[17] Georges Da Costa, MojitO/S, 2021, URL https://hal.archives-ouvertes.fr/hal-03453537, version 1.0, https://gitlab.irit.fr/sepia-pub/mojitos.

[18] Georges Da Costa, Power, performance and system measures of HPC benchmarks on multiple hardware, 2024, http://dx.doi.org/10.5281/zenodo.10982238, Dataset on Zenodo.

[19] Subhash Saini, Johnny Chang, Robert Hood, Haoqiang Jin, A scalability study of columbia using the nas parallel benchmarks, J. Comput. Methods Sci. Eng. (2006).

[20] Intel, Intel-64 and ia-32 architectures software developer's manual, Vol. 3: Syst. Program. Guid. 3 (2015).