Light-weight prediction for improving energy consumption in HPC platforms

Danilo Carastan-Santos, Georges da Costa, Millian Poquet, Patricia Stolf, Denis Trystram

Light-weight prediction for improving energy consumption in HPC platforms

Danilo Carastan-Santos[1][0000−0002−1878−8137], Georges Da Costa[2][0000−0002−3365−7709], Millian Poquet[2][0000−0002−1368−5016], Patricia Stolf[2][0000−0001−5169−1831], and Denis Trystram[1][0000−0002−2623−6922]

[1] Univ. Grenoble Alpes, CNRS, INRIA, Grenoble INP**, LIG, Grenoble, France
{danilo.carastan-dos-santos,denis.trystram}@univ-grenoble-alpes.fr
[2] Université de Toulouse, IRIT, CNRS
{georges.da-costa,millian.poquet,patricia.stolf}@irit.fr

Abstract. With the increase of demand for computing resources and the struggle to provide the necessary energy, power-aware resource management is becoming a major issue for the High-performance computing (HPC) community. Including reliable energy management to a supercomputer's resource and job management system (RJMS) is not an easy task. The energy consumption of jobs is rarely known in advance and the workload of every machine is unique and different from the others.

We argue that the first step toward properly managing energy is to deeply understand the energy consumption of the workload, which involves predicting the workload's power consumption and exploiting it by using smart power-aware scheduling algorithms. Crucial questions are (i) how sophisticated a prediction method needs to be to provide accurate workload power predictions, and (ii) to what point an accurate workload's power prediction translates into efficient energy management.

In this work, we propose a method to predict and exploit HPC workloads' power consumption, with the objective of reducing the supercomputer's power consumption while maintaining the management (scheduling) performance of the RJMS. Our method exploits workload submission logs with power monitoring data, and relies on a mix of light-weight power prediction methods and a classical EASY Backfillling inspired heuristic.

We base this study on logs of Marconi 100, a 980 servers supercomputer. We show using simulation that a light-weight history-based prediction method can provide accurate enough power prediction to improve the energy management of a large scale supercomputer compared to energy-unaware scheduling algorithms. These improvements have no significant negative impact on performance.

Keywords: Machine learning · HPC · Resource management · Power capping · Simulation.

** Institute of engineering Univ. Grenoble Alpes

1 Introduction

High-Performance Computing (HPC) technology is becoming more accessible and less expensive to build, which opens the door to new fields to capitalize on the large computational capabilities afforded only by such large systems. However, as opposed to the production cost, the power consumption of HPC platforms only increases, reaching levels [16] in the order of the power consumption of a small city. Besides the carbon footprint issue [2] raised by this increase in the power consumption, current climate events may heavily strain the electricity grids [22] that power HPC platforms. To avoid outages, it has become crucial for HPC platform maintainers to deploy measures to ease the strain in the electricity grid, which is typically achieved by enforcing a power capping over time in the platform. The platform's resource manager must therefore adapt to the available power during this power constrained period.

Most existing works propose methods to predict the power consumption of the workload, coupled with a speed scaling (DVFS) method, to adapt to the available power. The drawback is the risk of unforeseeable effects on Quality of Service (QoS). Only few works in the literature propose a full framework, including a workload power prediction method feeding energy data at the submission time to a resource manager. These few works often result in complex and/or heavyweight optimization schemes that are perceived to be either too risky that might disrupt regular functioning, or too expensive in terms of computational resources, thus reducing the (constrained) power available for the applications.

In contrast with related works, this work aims to adapt to the available power and deal with the power constraints *as lightweight and simple as possible.* We exploit power consumption data to develop models to predict the power consumption of an HPC application in advance. These models feed power consumption predictions of arriving applications to a scheduler, and the scheduler uses these predictions to comply with the power constraints while keeping the supercomputer operational. Our experimental results highlight that a lightweight, history-based predictor of the mean power consumption – which is arguably one of the simplest descriptors of an applications' power consumption – coupled with an EASY Backfilling [12] inspired scheduler can make a close to optimal use of the available power in constrained periods.

We organize the rest of this paper as follows: Section 2 presents related works in the literature, Section 3 presents preliminary concepts needed to understand our work's context, and Sections 4 and 5 present our methods to predict HPC applications power consumption and to schedule them in an HPC platform, respectively. We present and discuss our experimental results in Section 6, and we present our concluding remarks and future perspectives in Section 7.

2 Related Work

This section provides an overview of the related works regarding supercomputer's power monitoring/predictions and prediction-aided HPC resource management.

The reader can consult Kocot *et al.* [15] work for a more comprehensive survey on energy-aware resource management in HPC platforms.

Many works propose to exploit predictions to improve the performance of HPC resource management. For instance, Zrigui *et al.* [23] used a coarse grain prediction of jobs into long and short to design a scheduling algorithm taking this information into account for the minimization of the maximum completion time of a set of jobs. In [13], the authors propose a new scheduling algorithm that outperforms the popular EASY backfilling algorithm by 28% considering the average bounded slowdown objective taking into account predictions on the job running times. In the context of predicting the power consumption, Storlie *et al.* [21] developed a framework that predicts energy consumption of arriving jobs. They build a statistical model to approximate the power used by HPC jobs using hierarchical Bayesian modeling with hidden Markov and Dirichlet process models. The goal of their model is to enable the use of an individual node-capping power strategy shown to be more effective for limiting energy consumption than a uniform one.It is the most wholesome model in comparison to the others, though it comes with a high level of complexity. Bugbee *et al.* [5] proposed another model by combining *a priori* (resource manager's meta-data) and *in situ* data (collected during jobs execution). They focus on the specific applications that exhibit a periodic behavior, which accounts for only 45% of the total workload. Another limitation is that developing fine-tuned models for each possible application may be impractical and too resource demanding. Borghesi *et al.* [3] and more recently Saillant *et al.* [19] and Antici *et al.* [1] proposed Machine Learning (ML) and Rote-Learning approaches that rely on resource manager meta-data in order to predict the power consumption of a HPC workload. They combine this information measurements using the RAPL [14] interface. Borghesi *et al.* [3] introduced the idea that the mean value is a good descriptor of the HPC applications' power consumption. We distinguish from these works in two aspects: (i) we explore and compare a prediction method that do not rely on ML to predict the power consumption, and (ii) we further investigate on how the predicted mean power value can be useful to a resource manager to adapt to the constrained power. In [20], the authors focused on large-scale parallel jobs for predicting the energy of a parallel application just by observing a few of its active nodes (as opposed to monitor all the deployed nodes). Chapsis *et al.* [7] propose a power prediction and a resource management framework that includes the power variability due to hardware manufacturing. Their work involves a fine-grained monitoring of the applications and using hardware specific features (hardware counters) to predict the power consumption. Such an approach can be computationally expensive, especially due to the overhead introduced in the computing nodes by the fine-grained monitoring of numerous counters. The approach we propose in this paper intents to reach a balance in the granularity of the used information: providing coarse grained information on the power profile while using only resource manager related information and past, coarse-grained, monitored executions on the platform.

3 Preliminary Concepts

Modern HPC platforms contain large number of nodes. They usually are homogeneous[3]. Many users submit parallel applications (hereafter referred to as jobs) to be executed in the HPC platform, and these jobs can arrive at any point in time (i.e., online job submission). The jobs' submissions are managed by the Resources and Jobs Management System (RJMS). It decides when and where to process each job. Typical meta-data available to the RJMS for a given job j is its arrival time (r_j), requested number of processors (q_j) and an estimation of its processing time ($\tilde{p}_j$) which is provided by the user who submitted j. Resources allocation and execution is usually represented in a Gantt chart (Figure 1).

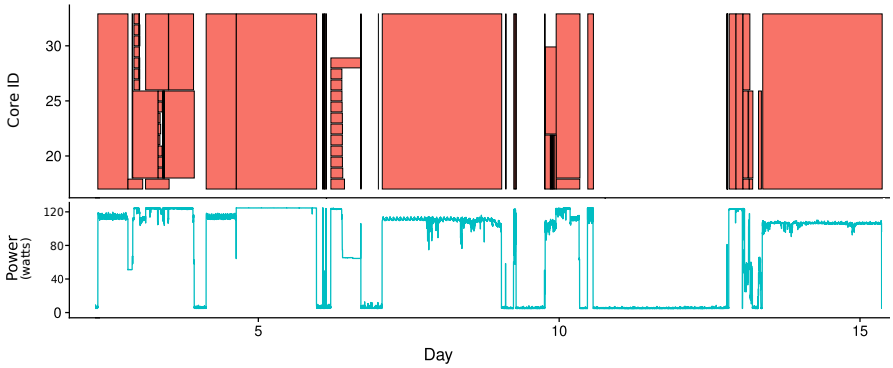


Fig. 1: The two sources of data used in a single, 32-core processor example: (top) data coming from the RJMS regarding the jobs' execution and allocation (processing time *versus* cores allocated), and (bottom) data coming from a power monitoring tool of the computing node (power *versus* time).

RJMS needs even more information when scheduling jobs under power capping. It needs at least a power profile which will serve as a power constraint over a certain time window. Recent HPC platforms are deployed with energy consumption monitoring tools such as IPMI [8], wattmeters, or software modules (often using RAPL [14]). Such an energy monitoring tool can provide power consumption data at the computing node level, as a time series of the power consumption of the computing node in function of time (bottom graph in Figure 1).

We can cross the RJMS's jobs data with the computing nodes' power monitoring data to get an idea about the power consumption of the jobs. In the case of jobs that share a same computing node (stacked rectangles in Figure 1), the resulting power consumption is in function of each of the jobs' energy consumption plus potential interference between the jobs, which makes it hard to

[3] The term homogeneous in this paper means that all computing nodes have the same CPU/accelerators configuration.

distinguish each of the jobs' contribution to the power consumption of the node. When taking into account jobs that had exclusive access to computing nodes, however, we can identify a power consumption profile of the jobs. The exact jobs' power profile can only be known after the jobs' execution.

In this work we focus on exploiting the jobs' power consumption profiles to perform better scheduling under power constraints *in the simplest way possible*. We choose simplicity because (i) sophisticated methods often leads to heavy-weight computations whose power consumption can reduce or at some point nullify the power savings from better scheduling, and (ii) sophisticated methods are often hard to explain/justify which hinders their deployment in practice [11].

More specifically, we focus on the following research questions:

1. Which **simple piece of information** regarding the jobs' power profile contribute to better scheduling under power constraints?
2. How to predict this simple piece of information **before the jobs' execution**? How much prediction accuracy can we achieve?

We consider as simple piece of information (hereafter referred to as jobs' power consumption for simplicity) the **mean** and **maximum** metrics of the jobs' power consumption profile. We decide to study the mean and the maximum because these two metrics can have different impacts in the scheduling with power constraints. We uncover these impacts in Section 6.5.

4 Predicting the power consumption of HPC jobs

For each job, the RJMS follows the same algorithm before the job's execution: (i) read all the meta-data of the job (including the user's id); (ii) predict the mean and max power consumption of the job; (iii) provide the job to the scheduler along with power information. The RJMS also has access to past measures, including meta-data and power consumption of past jobs.

We propose two job power prediction methods with an increasing level of complexity to predict the jobs power consumption: the first method uses the power consumption of previous users' jobs (users' job history). The second method uses previous jobs' power consumption data and jobs metadata (Table 1) with Machine Learning (ML) regression methods.

4.1 The first method: predicting power consumption with users' jobs history.

For a job j of a given user and a sliding window size s in seconds, we predict the power consumption of the job $\hat{P_j}$ as follows,

$$\hat{P_j} = \frac{\sum_{j' \in W} \theta_{j'} P_{j'}}{\sum_{j' \in W} \theta_{j'}} \tag{1}$$

$$W = \{j' \mid r_j \geq C_{j'} \geq (r_j - s)\} \tag{2}$$

$$\theta_{j'} = \left(1 - \left(\frac{r_j - C_{j'}}{s}\right)\right)^{\alpha} \tag{3}$$

where $C_{j'}$ is the completion time of a previously executed job j' of same user who submitted j, and $P_{j'}$ is the measured mean power of j'. The power consumption of a previous job $P_{j'}$ can be known at the time we predict $\hat{P}_j$ since $C_{j'} \leq r_j$.

Equation 1 is a weighted average of previous jobs j' of the user that finished within a sliding time window with size s (Equation 2). We assign the weight $\theta_{j'}$ (Equation 3) in function of how long in the past j' finished compared to the arrival time of j. A value of $\theta_{j'} = 0$ means that j' finished at the oldest allowed date, and $\theta_{j'} = 1$ means that the j' finished exactly at the arrival time of j. The parameter α changes the way we penalize older jobs by changing the $\theta_{j'}$ behavior between 0 and 1, from a linear $\alpha = 1$, to a super-linear $\alpha = 2$ or sub-linear $\alpha = 0.5$ fashion.

In this paper, we considered the *max* and the *mean* power consumption as we assume that both metrics have different effects when used for performing scheduling decisions. These effects are explored in more detail in Section 6.5.

4.2 The second method: predicting using supervised regression.

The former prediction method can not harness the jobs' metadata (e.g., requested resources, submission time, expected processing time, etc.) to potentially provide better predictions. We can circumvent this problem by using supervised regression with the hypothesis of increasing the prediction accuracy, using the jobs' metadata and the power consumption history as input features.

We propose an online learning method to predict the jobs' power consumption. The method retrains the prediction models at periodic time intervals (e.g., at the end of each week) in order to adapt, as the jobs' history increases and changes. In this context, online still refers to the behavior of the RJMS using all past data, and only meta-data of the arriving jobs. For an *already passed* week with index w (i.e., week 0, 1, 2, ...), let $t(w)$ be the timestamp of when the models will be retrained at the end of week w. Then, we define our job dataset $\mathbf{J}_{train}$ as follows.

$$\mathbf{J}_{train} = \{j \mid C_j < t(w)\} \tag{4}$$

In other words, $\mathbf{J}_{train}$ contains the jobs' history. Then, we train a predictor $\hat{f}(j)$ of the jobs' power consumption that minimizes the Mean Squared Error (MSE, Equation 5).

$$MSE = \frac{1}{|\mathbf{J}_{train}|} \sum_{j \in \mathbf{J}_{train}} \left(P_j - \hat{f}(j)\right)^2 \tag{5}$$

After training a predictor $\hat{f}(j)$ we use it to predict the power consumption $\hat{P}_{j'}$ for all jobs $j' \in \mathbf{J}_{inference}$, where $\mathbf{J}_{inference}$ is defined as follows.

$$\mathbf{J}_{inference} = \{j' \mid t(w) \leq r_{j'} \leq t(w+1)\} \tag{6}$$

In other words, we use the jobs history to train a model at week w that already passed, and use this model to perform predictions of the power consumption of the jobs that will arrive online at week $w+1$. At the end of week $w+1$ (i.e., at timestamp $t(w+1)$) the training procedure repeats, generating a new predictor $\hat{f}(j)$, which will be used for week $w+2$. A particular situation is for week $w=0$, where there is no such dataset to train a regression model. In this case we can use $\hat{f}(j) = \hat{P}_j$ (Equation 1). We present the choice of regression methods to train $\hat{f}(j)$, and how we exploit the jobs' data (i.e, the features of j) in Section 6.2.

5 Scheduling with jobs' power profile prediction

As the focus of this article is on the impact of power prediction, we will use a simple and classical EASY backfilling algorithm [12]. This algorithm is used in production in a large number of HPC centers. We assume that the platform is static: no failure of nodes nor new nodes during the experiment. We also assume that the servers will always be switched on. We finally assume there is no constraints on the applications related to the host they can run on. All these assumptions enable us to focus on the impact of the prediction framework, further studies on these hypotheses are kept as perspectives.

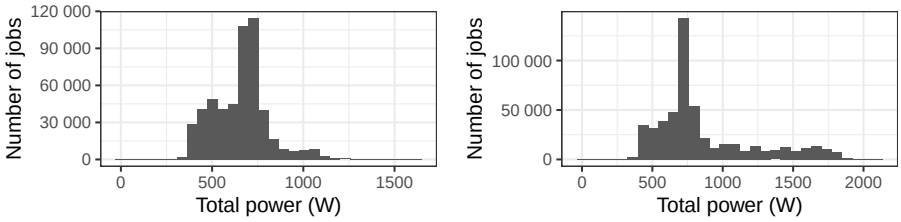
The implemented EASY uses a first-come first serve (FCFS) policy and works as follows. EASY is executed when these events occur: a new task arrives, a task finishes. Tasks are stored in a queue in their order of arrival — the order of this queue will never be changed. EASY starts the oldest jobs in the queue until either finishing the queue or arriving on a job (called high-priority job) that cannot start due to lack of resources. In the later case, EASY will try to *backfill* jobs, that is to say start remaining task right now if they do not impact the high-priority job estimated starting time.

The only modification we have made to the classical EASY is how to check whether resources are available. Classical EASY only checks whether there are enough available nodes/cores. Here, our implementation named EASY+PC also checks whether there is enough power w.r.t. a given power cap. We assume that an estimator (such as the ones we propose) can estimate the power needed for a job. Based on this estimator, EASY+PC estimates the currently used power and the requested power for each job in the queue. All of EASY+PC decisions are based solely on these estimations.

6 Results

6.1 Jobs' power prediction: dataset description

This work uses the trace collected from the Marconi100 supercomputer [4]. Marconi100 consisted of 980 computing nodes, each of which consisted of a two-socket IBM POWER9 AC922 (32 cores in total), and four NVIDIA Volta V100 GPUs. The trace contains jobs' meta-data such as jobs submission times, processing times, anonymized user ids, and node ids allocated to the jobs. It also contains



(a) Mean total power consumption (b) Max total power consumption

Fig. 2: Distributions of the actual mean and maximum power consumption of the filtered jobs in the Marconi100 trace.

data about the nodes' total power consumption, measured at each 20-second periods, using an IPMI module installed in the nodes.

From this trace we used the data regarding the operation of the Marconi100 from January 2022 to September 2022. To circumvent the limitation mentioned in Section 3, we filtered out the jobs that shared a node from the original trace. We also filtered out jobs that run in less than a minute because they have a too small number of power measurements. After this filtering we end up having a job dataset submissions from 576 users, accounting for 523,204 jobs.

Figure 2 shows the mean and max power consumption of the computing nodes for each of the jobs in the dataset. We can observe a peak density of jobs with mean and max consumption of around 700 watts. This insight in itself could serve as a prediction if the whole distribution was clear and concentrated at around 700 watts. However, the jobs distribution is not so clear for other power values. which justifies the need of more sophisticated prediction methods.

6.2 Jobs' power prediction: experimental setting

For the history based power prediction method (Section 4), we set the parameter α (Equation 3) with a value of 2. This choice is based on the hypothesis that recent jobs have more importance than older jobs when predicting the power consumption. An $\alpha = 2$ puts more weight into recent jobs, and the weight decreases fast for older jobs. We set the sliding window size s to account for the whole user's job history, which translates to s being completion time of the first finished job of a user. We decided on this design to avoid eventual empty sliding windows during the prediction process. For the regression based power prediction method, we use the features presented in Table 1. Features 1 to 4 are standard job data that can be obtained at job submission, and features 5 to 7 are lagged features (i.e., a standard feature engineering technique), which can be obtained by using the user's job submission history.

We trained predictors $\hat{f}(j)$ using a selection of regression methods in the `scikit-learn` [17] library: (i) Linear Regression (`LinearRegression`), (ii) Random Forest (`RandomForestRegressor`), (iii) Support Vector Regression with

Table 1: Features used in the mean power consumption regression methods

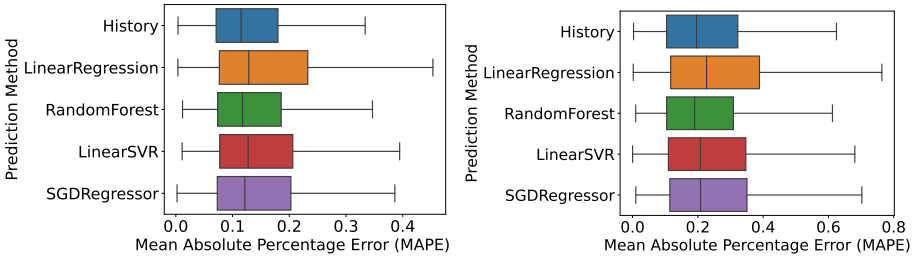
Feature	Description
1	Submission time (hour of the day)
2	Number of processors (q_j)
3	Number of nodes
4	Requested processing time $\tilde{p}_j$
5	The sliding window history prediction $\hat{P}_j$ (Equation 1)
6	Standard deviation $\sigma(\{P_{j'} \mid j' \in W\})$
7	The power consumption P_{j^*} where j^* is the last finished user job

Linear Kernel (`LinearSVR`), and (iv) Stochastic Gradient Descent Regression (`SGDRegressor`). For each training week and method, we applied `scikit-learn`'s recursive feature elimination technique to choose the appropriate subset of features from Table 1 to use according to the training data. We perform this feature elimination and we set each of the regression methods hyper-parameters with a 5-fold cross validation scheme.

6.3 How to predict jobs' power information before the jobs' execution? How much prediction accuracy can we achieve?

Figure 3 shows the mean absolute percentage error (MAPE) of the methods used to predict the mean and the max jobs' power consumption. The boxplots represent the distribution of the prediction performance for each of the 576 users present in our job dataset. For the mean power consumption, we achieve a prediction MAPE from 0.115 (using jobs' history method, Section 4) to 0.128 (using linear regression, Section 4). This translates into a prediction accuracy of 88.5% and 87.2%, respectively. For predicting the maximum power consumption, we achieve a prediction MAPE from 0.195 (jobs' history method) to 0.215 (linear regression), which translates to a prediction accuracy of 80.5% and 78.5%, respectively. For both predicting the mean and the maximum power consumption, we can not clearly distinguish which prediction method is better. We can observe, however, that our jobs' history prediction method achieves equivalent prediction performance than the more sophisticated ML methods.

This is an important finding. Because we develop these prediction methods to reduce the energy consumption of operating supercomputers, we must reduce the energy consumption overhead induced by introducing these methods as much as possible. Achieving a high level of power prediction performance with the lowest level of energy consumption is therefore a priority. Although all of the regression methods used can be seen as lightweight when compared to neural network methods, a simple jobs' history based method is clearly much less energy demanding than a Machine Learning regression method, which requires much more processing steps (i.e., normalizing data, selecting the best features and finding the best hyper-parameters with cross-validation, etc.).



(a) Mean total power consumption (b) Max total power consumption

Fig. 3: Mean Absolute Percentage Error of the mean and maximum power consumption prediction methods for the jobs in the Marconi100 trace.

6.4 Jobs scheduling with power prediction: experimental setup

We simulate 30 different workloads using EASY+PC (Section 5) with various powercap values and with various job power predictors. Each workload consists of jobs taken from the filtered Marconi100 supercomputer dataset (Section 6.1). Jobs are selected following workload trace replay guidelines [10].

EASY+PC applies a power constraint solely during the first 3 hours of each simulation. The powercap values we use range from 10% to 70% of the highest dynamic power consumption of the Marconi100 dataset in 2022 (955080 W).

We study EASY+PC's behavior depending on the information it uses to determine the power consumption of a job: (i) predicted `mean` and (ii) predicted `max` use the users' job history prediction method (Section 4.1), (iii) real `mean` and (iv) real `max` are the mean and max power consumption obtained from the workload dataset, and (v) `naive` uses the maximum reachable power consumption of a job (i.e., 2100W * q_j, where 2100W is the maximum single-node power value present in the Marconi100 dataset in 2022). `naive` is used as a baseline predictor to evaluate the accuracy of the other predictors proposed.

We use Batsim [9] and SimGrid [6] to perform the scheduling simulations, using a SimGrid representation of the whole 980-node of Marconi100. We use the schedule produced by the simulators plus the time-series data about the jobs' power consumption from the Marconi100 dataset to calculate the total power consumption. Taking into account the 30 workloads and the job power predictors, our experimental campaign consists of 1950 simulation instances. Additionally, each workload is executed on EASY (without powercap) to serve as baseline and evaluate both impacts on total power consumption and QoS.

6.5 Which jobs' power information contribute to better scheduling under power constraints?

Figures 4 and 5 summarizes results of our simulation campaign (Section 6.4) by aggregating data from all workloads together. 1 workload has been excluded from Figure 5 as powercapping greatly increases scheduling performance on it. Values

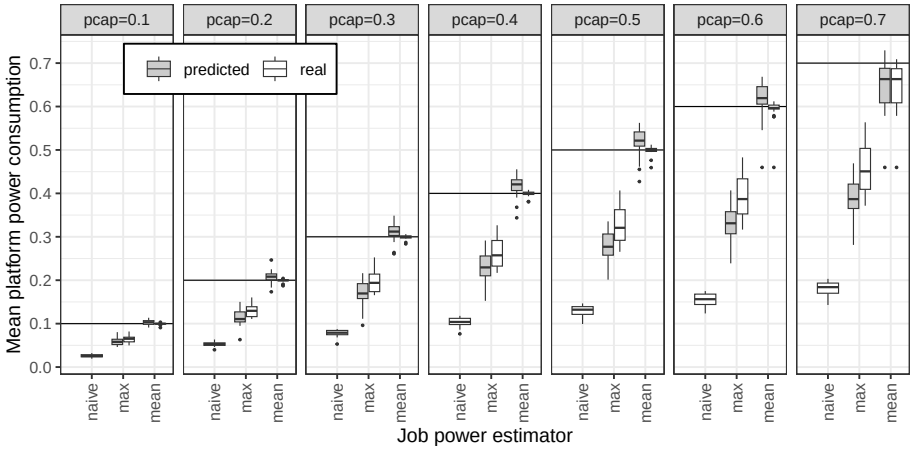


Fig. 4: Distribution of the platform power consumption during the powercap-constrained 3-hour time window. Powercap is the horizontal black line. All power values (y axis, facet powercap `pcap`) are expressed as a proportion of the maximum dynamic power range. Standard boxplots (Q_{1-3}, 1.5 IQR).

given in the remaining of this section are computed by (workload, powercap) group, and then averaged on all groups. They analyze (I) the extent to which each predictor is able to use the power at its disposal while powercap is active, and (II) how they degrade QoS performance (through turnaround time) compared to baseline EASY's.

Since `naive` considers the maximum achievable node power for the whole jobs' duration, EASY+PC is incapable of harnessing the power consumption fluctuations that occur during job execution to better use the available power. This incapability leads to a severe power under-utilization (74%), and a significant increase in the turnaround time (15%).

`max` provides better information about the maximum power consumption than `naive`, thus better harnessing the fluctuations. `max` still remains, however, as a "conservative" method which hypothesizes that the maximum value occurs all the time during the jobs' execution. Such hypothesis helps assuring that EASY+PC does not trespass the power cap (it never did, even when using our `max` prediction method), though this results in power under-utilization and increase in the turnaround time (respectively around 44% and 11%).

More "aggressive", `mean` hypothesizes that the mean power is the value that occurs most of the time during the jobs' execution. This hypothesis gives more flexibility to EASY+PC to harness the jobs' power fluctuations. The drawback is the increased risk of trespassing the powercap. In our experiments, the `mean` method is the one that makes the best use of the available power (3% power over-utilization) and the one that increases the turnaround time the least (6%). Using `mean` trespasses the powercap in most instances (95%) but the trespassing

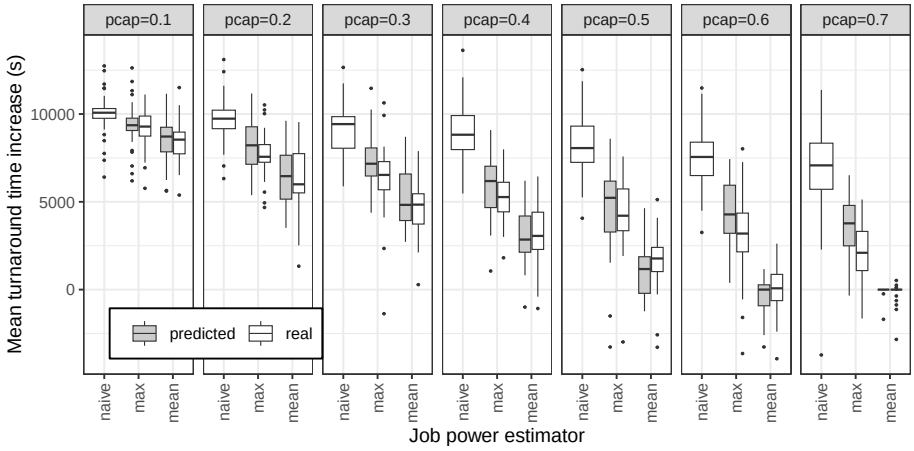


Fig. 5: Distribution of the performance degradation (compared to EASY) for 29/30 workloads. The turnaround time of a job is the amount of time the job spends in the system (from submission to finish). The workload mean turnaround time is the arithmetic mean all the jobs' turnaround time. Standard boxplots.

is small: the maximum instantaneous powercap break observed is in average and median 14% above the powercap.

Lastly, the prediction accuracy of the `mean` method (Section 6.3) results in satisfactory performances when compared to using real mean values. Please note that real values are baselines and cannot be obtained before the jobs' execution.

7 Conclusion

We presented in this paper two main contributions: a complete integrated environment from monitored data to the jobs execution and an evaluation of several jobs' power consumption prediction methods. In particular, we showed that "lightweight" (frugal) predictions used in the scheduling module lead to similar performance improvements, when compared to more costly and sophisticated predictions or compared to the optimal value. Simple history prediction method (such as mean) is sufficiently good to express the jobs' power profiles during scheduling, which fosters "lower-tech" scheduling algorithms.

The proposed approach focused on the capability to take into account these lightweight predictors for a classical and widely used EASY scheduling policy. From this positive experience on EASY, the next step is to investigate new scheduling policies harnessing the new information from the predictors, but also adding actual monitoring values to improve the quality of its decision. It would also be interesting to refine the jobs' model, to take into account phases of long duration applications. Using such information would help to improve the management of short duration jobs.

Acknowledgements and Artifact Availability

Experiments presented in this paper were carried out using the Grid'5000 testbed. This work was supported by the research program on Edge Intelligence of the Multi-disciplinary Institute on Artificial Intelligence MIAI at Grenoble Alpes (ANR-19-P3IA-0003), ENERGUMEN (ANR-18-CE25-0008), the France 2030 NumPEx Exa-SofT (ANR-22-EXNU-0003) and Cloud CareCloud (ANR-23-PECL-0003) projects managed by the French National Research Agency (ANR), REGALE (H2020-JTI-EuroHPC-2019-1 agreement n. 956560), and LIGHTAIDGE (HORIZON-MSCA-2022-PF-01 agreement n. 101107953). We thank Salah Zrigui for starting the study on the job energy profiles. We also thank Francesco Antici for curating and sharing the Marconi100 dataset.

The experiments described in this article have been made with open science and reproducibility concerns in mind. Code, data and documentation to reproduce our work is available on Zenodo [18].

References

1. Antici, F., Yamamoto, K., Domke, J., Kiziltan, Z.: Augmenting ml-based predictive modelling with nlp to forecast a job's power consumption. In: Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis. pp. 1820–1830 (2023)
2. Bates, N., Ghatikar, G., Abdulla, G., Koenig, G.A., Bhalachandra, S., Sheikhalishahi, M., Patki, T., Rountree, B., Poole, S.: Electrical grid and supercomputing centers: An investigative analysis of emerging opportunities and challenges. Informatik-Spektrum **38**(2), 111–127 (2015)
3. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: Predictive modeling for job power consumption in hpc systems. In: Kunkel, J.M., Balaji, P., Dongarra, J. (eds.) High Performance Computing. pp. 181–199. Springer International Publishing, Cham (2016)
4. Borghesi, A., Di Santi, C., Molan, M., Ardebili, M.S., Mauri, A., Guarrasi, M., Galetti, D., Cestari, M., Barchi, F., Benini, L., et al.: M100 exadata: a data collection campaign on the cineca's marconi100 tier-0 supercomputer. Scientific Data **10**(1), 288 (2023)
5. Bugbee, B., Phillips, C., Egan, H., Elmore, R., Gruchalla, K., Purkayastha, A.: Prediction and characterization of application power use in a high-performance computing environment. Statistical Analysis and Data Mining: The ASA Data Science Journal **10**(3), 155–165 (2017)
6. Casanova, H., Giersch, A., Legrand, A., Quinson, M., Suter, F.: Versatile, scalable, and accurate simulation of distributed applications and platforms. Journal of Parallel and Distributed Computing **74**(10), 2899–2917 (Jun 2014)
7. Chasapis, D., Moretó, M., Schulz, M., Rountree, B., Valero, M., Casas, M.: Power efficient job scheduling by predicting the impact of processor manufacturing variability. In: Proceedings of the ACM International Conference on Supercomputing. pp. 296–307 (2019)

8. Da Costa, G., Pierson, J.M., Fontoura-Cupertino, L.: Mastering system and power measures for servers in datacenter. Sustainable Computing: Informatics and Systems **15**, 28–38 (2017). https://doi.org/10.1016/j.suscom.2017.05.003
9. Dutot, P.F., Mercier, M., Poquet, M., Richard, O.: 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), https://hal.science/hal-01333471
10. Emeras, J.: Workload Traces Analysis and Replay in Large Scale Distributed Systems. Theses, Université de Grenoble (Oct 2013)
11. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U., Sevcik, K.C., Wong, P.: Theory and practice in parallel job scheduling. In: JSSPP: IPPS'97 Processing Workshop Geneva, Switzerland, April 5, 1997 Proceedings 3. pp. 1–34. Springer (1997)
12. Feitelson, D.G., Weil, A.M.: Utilization and predictability in scheduling the ibm sp2 with backfilling. In: Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing. pp. 542–546. IEEE (1998)
13. Gaussier, E., Glesser, D., Reis, V., Trystram, D.: Improving backfilling by using machine learning to predict running times. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '15, Association for Computing Machinery, New York, NY, USA (2015)
14. Khan, K.N., Hirki, M., Niemi, T., Nurminen, J.K., Ou, Z.: Rapl in action: Experiences in using rapl for power measurements. ACM Trans. Model. Perform. Eval. Comput. Syst. **3**(2) (mar 2018). https://doi.org/10.1145/3177754
15. Kocot, B., Czarnul, P., Proficz, J.: Energy-aware scheduling for high-performance computing systems: A survey. Energies **16**(2), 890 (2023)
16. Laboratory, O.R.N.: Frontier's architecture. https://olcf.ornl.gov/wp-content/uploads/Frontiers-Architecture-Frontier-Training-Series-final.pdf (2023), online; last access 29 november 2023
17. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
18. Poquet, M., Carastan-Santos, D., Da Costa, G., Stolf, P., Trystram, D.: Artifact data of article "Light-weight prediction for improving energy consumption in HPC platforms", Euro-Par 2024 (May 2024). https://doi.org/10.5281/zenodo.11173631
19. Saillant, T., Weill, J.C., Mougeot, M.: Predicting job power consumption based on rjms submission data in hpc systems. In: Sadayappan, P., Chamberlain, B.L., Juckeland, G., Ltaief, H. (eds.) High Performance Computing. pp. 63–82. Springer International Publishing, Cham (2020)
20. Shoukourian, H., Wilde, T., Auweter, A., Bode, A.: Predicting the energy and power consumption of strong and weak scaling hpc applications. Supercomputing Frontiers and Innovations **1**(2) (2014)
21. Storlie, C., Sexton, J., Pakin, S., Lang, M., Reich, B., Rust, W.: Modeling and predicting power consumption of high performance computing jobs (2015)
22. Wikipedia: 2021 Texas power crisis, Online; last access 29 november 2023. https://en.wikipedia.org/wiki/2021_Texas_power_crisis (2023)
23. Zrigui, S., de Camargo, R.Y., Legrand, A., Trystram, D.: Improving the performance of batch schedulers using online job runtime classification. Journal of Parallel and Distributed Computing **164**, 83–95 (2022)