Scheduling with lightweight predictions in power-constrained HPC platforms

Danilo Carastan-Santos*, Georges Da Costa†, Igor Fontana de Nardin†, Millian Poquet†, Krzysztof Rzadca‡, Patricia Stolf† and Denis Trystram*

*University Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG
Grenoble, France
Email: danilo.carastan-dos-santos@inria.fr, Denis.Trystram@univ-grenoble-alpes.fr

†University of Toulouse, IRIT, CNRS
Toulouse, France
Email: georges.da-costa@irit.fr, igor.fontana@irit.fr, millian.poquet@univ-tlse3.fr, patricia.stolf@irit.fr

‡University of Warsaw, Google
Warsaw, Poland
Email: krzadca@mimuw.edu.pl

***Index Terms*—Machine learning, HPC, Resource management, Power capping, Simulation**

I. 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 [1] 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 [3] 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. Only few works in the literature have proposed a full framework, including a workload power prediction method feeding energy data at the submission time to a resource manager [4]. These few works often result in complex and/or heavyweight optimization schemes.

In contrast with related works, the purpose of our work is to propose methods that 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 contributions are: **(i)** we predicted the job's power consumption *mean*, *max*, and *standard deviation* using a lightweight history-based predictor. We show that the lightweight predictor is close to the Machine Learning regressors; **(ii)** we compare different ways to verify if a set of jobs is under power capping. They are *Naive*, *Max*, *Mean*, and *Gaussian*; and **(iii)** we coupled the power prediction and power verification in two scheduling algorithms: *EASY Backfilling* [5] inspired and *Greedy Knapsack*. While *EASY Backfilling* is a well-known scheduling algorithm, the *Greedy Knapsack* improves the Quality of Service (QoS) and avoids starvation. This abstract compiles a previous work [6] and an article in the revision process for the IEEE Transactions on Parallel and Distributed Systems journal [7].

II. Methods

A. Power prediction

We propose two job power prediction methods. The first method uses the power consumption of previous users jobs (users job history). Therefore, for each job j of a given user and a sliding window size s, we estimate some statistics $\hat{P}_j^{\bullet}$ of its power consumption by taking a weighted average of historical power consumption over previous runs. The statistics we estimate can be the mean ($\hat{P}_j^{avg}$), maximum ($\hat{P}_j^{max}$), or standard deviation ($\hat{P}_j^{std}$) of the job's power consumption. The general method to predict $\hat{P}_j^{\bullet}$ is $\hat{P}_j^{\bullet} = \frac{\sum_{j' \in W} \theta_{j'} P_{j'}^{\bullet}}{\sum_{j' \in W} \theta_{j'}}$, where $W = \{j' \mid (r_j - s) \leq C_{j'} \leq r_j\}$, $\theta_{j'} = \left(1 - \left(\frac{r_j - C_{j'}}{s}\right)\right)^{\alpha}$, r_j is the release time of job j, $C_{j'}$ is the completion time of a previously executed job j' of same user who submitted j, $P_{j'}^{\bullet}$ is the measured metric of power consumption of j', $\theta_{j'}$ is aging adjustment for jobs, and α controls the way we penalize older jobs (i.e., $\alpha = 1$ is linear, $\alpha = 2$ is superlinear, and $\alpha = 0.5$ is sublinear). The metric of power consumption of a previous job $P_{j'}^{\bullet}$ can be known at the time we predict $\hat{P}_j$ since $C_{j'} \leq r_j$.

The second method uses previous jobs power consumption data and jobs metadata with Machine Learning (ML) regression methods. 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 $\hat{f}(j) = \hat{P}_j^{\bullet}$ from the first method.

B. Power verification

After predicting the power consumption, the scheduler must verify if the power usage of a set of jobs $\mathcal{J}[t]$ would violate the platform power cap $\overline{P}$. We use the following methods for verification: **Naive** uses the maximum reachable server's power consumption of a job. **Mean** uses the predicted mean of each job. Therefore, $\sum_{i \in \mathcal{J}[t]} \hat{P}_i^{avg} \leq \overline{P}$. **Max** uses the predicted max of each job. Therefore, $\sum_{i \in \mathcal{J}[t]} \hat{P}_i^{max} \leq \overline{P}$. **Gaussian** uses a probabilistic model for jobs energy consumption. This method tries to quantitatively formalize the intuition that, when many jobs execute concurrently, it is unlikely that all jobs simultaneously consume their peak power usage. When testing if $\mathcal{J}[t]$ fits the power budget, we verify $\sum_{i \in \mathcal{J}[t]}(\hat{P}_i^{avg}) + (\sigma \times \sqrt{\sum_{i \in \mathcal{J}[t]}(\hat{P}_i^{std})^2}) \leq \overline{P}$. σ controls how certain we are that the power capping will not be violated (e.g., $\sigma = 1$ has probability of 0.68 of not passing the power capping, $\sigma = 2$ has probability of 0.95, and $\sigma = 3$ has 0.99).

C. Scheduling

Regarding the scheduling, we first implemented an EASY Backfilling inspired algorithm. Usually, this algorithm only verifies if there are enough resources (e.g., cores, servers, memory, etc.). Besides this default verification, we included the tests from Section II-B. We proposed two EASY algorithms, sorting the waiting queue by FCFS (EASY FCFS) and Smallest Area First (EASY SAF). The area is the number of resources multiplied by the expected execution time. Then, we modeled the problem as a 0/1 knapsack problem. This optimization problem considers a knapsack with capacity Kc and m items denoted by index j. Each item j has a profit v_j and a weight we_j. It needs to determine the subset of items that fits into the knapsack ($\sum_{j=0}^{j=m-1} x_j \times we_j \leq Kc$) and that maximizes the total profit ($maximize \sum_{j=0}^{j=m-1} x_j \times v_j$). The classic formulation is to define a boolean x_j that is set to 1 if the item is selected and 0 if it is not [8]. In our case, the items are the jobs in the queue. The capacity Kc is the power capping and the weight we_j is the predicted job power consumption. We implemented two profits related to the Quality of Service: waiting time ($v_j = wait_j$, where $wait_j$ is the waiting time) and stretch ($v_j = \frac{wait_j + \tilde{p}_j}{\tilde{p}_j}$ where $\tilde{p}_j$ is the expected execution time). We solve this problem using a greedy solution that takes the jobs in descending order of $\frac{v_j}{we_j}$.

III. Results

This work uses the trace collected from the Marconi100 supercomputer [9]. 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 [10]. Figure 1 compares all predictors regarding the Mean Absolute Error of the mean power consumption prediction. They have similar accuracy, highlighting that our jobs' history prediction method achieves equivalent prediction performance to the more sophisticated ML methods.

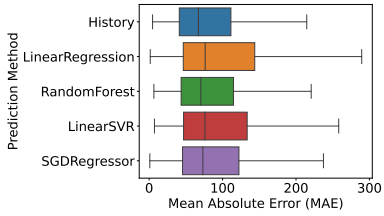

Fig. 1. Mean Absolute Error of the mean power consumption prediction methods for the jobs in the Marconi100 trace. Max and standard deviation have similar results.

Figure 2 demonstrates the results of the different power verifications and scheduling policies using the history based predictor. First, Figure 2a compares the different power verifications for a power capping of 0.5 of the total platform power and uses the same scheduling policy (EASY FCFS). The best results are the values close to the power capping but not surpassing it. Therefore, Gaussian 99 presents the best approach, with the majority of the results closer but under the power capping. Regarding Figures 2b and 2c, we compare the same power verification (Gaussian 99) with different scheduling algorithms. The results are compared to a baseline algorithm (EASY FCFS without power capping). EASY SAF has the best mean turnaround improvement (2b where the higher is the better) but the worst max turnaround increase (2c where the lower is the better). On the other hand, both knapsack algorithms present a good trade-off between mean and max turnaround.

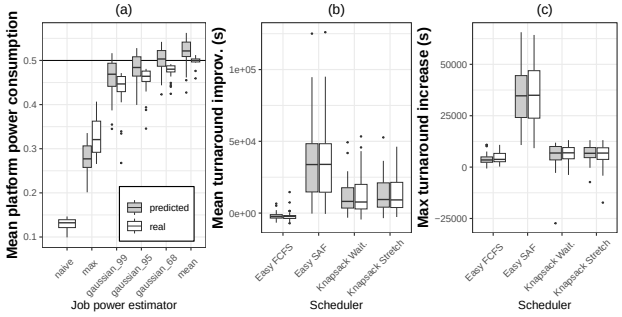

Fig. 2. The results of different power verifications and scheduling policies.

IV. Conclusion

This article presents a simple and lightweight way to predict and schedule jobs in an HPC platform under the power capping. From this positive experience on EASY and knapsack, the next step is to investigate more complex scheduling policies harnessing the new information from the predictors, but also adding actual monitoring values to improve the quality of its decision.

ACKNOWLEDGEMENTS

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). 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 ExaSofT (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), Polish National Science Center grant Opus (UMO-2017/25/B/ST6/00116), and LIGHTAIDGE (HORIZON-MSCA-2022-PF-01 agreement n. 101107953). A CC-BY public copyright licence has been applied by the authors to the present document and will be applied to all subsequent versions up to the Author Accepted Manuscript arising from this submission, in accordance with the grants' open access conditions. 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.

CONFLICTS OF INTEREST

Krzysztof Rzadca is also affiliated with Google.

REFERENCES

[1] "Frontier's architecture," https://www.olcf.ornl.gov/wp-content/uploads/Frontier-Architecture-Overview_Abraham.pdf, 2024, last access 18 October 2024.

[2] N. Bates, G. Ghatikar, G. Abdulla, G. A. Koenig, S. Bhalachandra, M. Sheikhalishahi, T. Patki, B. Rountree, and S. Poole, "Electrical grid and supercomputing centers: An investigative analysis of emerging opportunities and challenges," *Informatik-Spektrum*, vol. 38, no. 2, pp. 111–127, 2015.

[3] Wikipedia, "2021 Texas power crisis, Online; last access 29 november 2023," https://en.wikipedia.org/wiki/2021_Texas_power_crisis, 2023.

[4] B. Kocot, P. Czarnul, and J. Proficz, "Energy-aware scheduling for high-performance computing systems: A survey," *Energies*, vol. 16, no. 2, p. 890, 2023.

[5] D. G. Feitelson and A. M. Weil, "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*. IEEE, 1998, pp. 542–546.

[6] D. Carastan-Santos, G. Da Costa, M. Poquet, P. Stolf, and D. Trystram, "Light-weight prediction for improving energy consumption in hpc platforms," in *Euro-Par 2024: Parallel Processing*, J. Carretero, S. Shende, J. Garcia-Blas, I. Brandic, K. Olcoz, and M. Schreiber, Eds. Cham: Springer Nature Switzerland, 2024, pp. 152–165.

[7] D. Carastan-Santos, G. da Costa, I. Fontana de Nardin, M. Poquet, K. Rzadca, P. Stolf, and D. Trystram, "Scheduling with lightweight predictions in power-constrained HPC platforms," Oct. 2024, working paper or preprint. [Online]. Available: https://hal.science/hal-04747713

[8] S. Martello and P. Toth, "Algorithms for knapsack problems," *North-Holland Mathematics Studies*, vol. 132, pp. 213–257, 1987.

[9] A. Borghesi, C. Di Santi, M. Molan, M. S. Ardebili, A. Mauri, M. Guarrasi, D. Galetti, M. Cestari, F. Barchi, L. Benini *et al.*, "M100 exadata: a data collection campaign on the cineca's marconi100 tier-0 supercomputer," *Scientific Data*, vol. 10, no. 1, p. 288, 2023.

[10] M. Poquet, D. Carastan-Santos, I. Fontana de Nardin, G. Da Costa, K. Rzadca, P. Stolf, and D. Trystram, "Artifact data of article "Scheduling with lightweight predictions in power-constrained HPC platforms", TPDS 2024," 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13961003