PM100: A Job Power Consumption Dataset of a Large-scale Production HPC System

Francesco Antici
francesco.antici@unibo.it
DISI, University of Bologna
Italy

Mohsen Seyedkazemi Ardebili
mohsen.seyedkazemi@unibo.it
DEI, University of Bologna
Italy

Andrea Bartolini
a.bartolini@unibo.it
DEI, University of Bologna
Italy

Zeynep Kiziltan
zeynep.kiziltan@unibo.it
DISI, University of Bologna
Italy

ABSTRACT

The power requirements of modern High-Performance Computing (HPC) systems pose environmental and financial challenges, as they contribute to carbon emissions and strain power grids. Optimizing power consumption together with system performance has thus become a crucial goal for HPC centers. As jobs running on a system contribute to the whole system's power usage, predicting their power requirements before execution on the system would allow to forecast the overall power consumption and perform techniques like power capping at the workload manager level. Such predictive studies need quality data, which is limited due to the inherent complexity of collecting structured data for job power characterization in a production system. This paper aims to fill the lack of resources for job power prediction and provide the HPC community with (i) a methodology to create a job power consumption dataset from workload manager data and node power metrics logs, and (ii) a novel and large dataset comprising around 230K jobs and their corresponding power consumption values. The dataset is derived from M100, a holistic dataset extracted from a production supercomputer hosted at the HPC centre CINECA in Italy.

KEYWORDS

High-performance computing, M100, job power consumption, workload analysis, dataset, machine learning.

ACM Reference Format:
Francesco Antici, Mohsen Seyedkazemi Ardebili, Andrea Bartolini, and Zeynep Kiziltan. 2023. PM100: A Job Power Consumption Dataset of a Large-scale Production HPC System. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3624062.3624263

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624263

1 INTRODUCTION

In the rapidly advancing landscape of High-Performance Computing (HPC), the demand for computational resources continues to surge, driven by the ever-increasing complexity of scientific simulations, data analysis, and machine learning applications. The power requirements of HPC systems are significant, and they not only pose financial challenges but also raise environmental concerns. Therefore, optimizing power consumption together with system performance has become a central objective for HPC facilities, academic institutions, and research laboratories worldwide[9].

A job running on an HPC system contributes to the whole system's power consumption. Job power consumption refers to the amount of electrical power consumed by the job while executing its computational tasks on the system resources. Hence, the total power consumption of a job is computed by aggregating the power consumption of all the resources allocated to the job during its execution. Predicting the power consumption of a job before its execution would allow to forecast the whole system's power consumption. Past work proved the feasibility of predicting job power consumption by leveraging on workload manager information [5, 13–15]. The prediction can then be exploited by the workload manager to perform techniques like power capping [6, 16].

All the referenced work addressed the prediction task by using data-driven techniques exploiting a structured dataset. Due to the steady development of such techniques, the availability of quality data extracted from a production HPC system has thus become a leading priority. Such resources are, however, often limited due to the inherent complexity of collecting structured data for job power characterization in a production system. Power consumption measurement relies on, among others, hardware sensors and different software interface. To give an idea, most of the modern systems have in-band software interface (available within the operating system of the compute node) for the power measurement of compute elements, whereas for node-level and component-level measurement, they rely on out-of-band interface (observable in the management network) and smart power switches which monitor cluster power consumption (observable in the facility management network). Job information instead requires monitoring the workload manager, which is possible from the login and master node. Accessing simultaneously to all these data resources requires different privilege levels and monitoring software validation procedures. While this is relatively easy in a test environment, it is not the case

in a production machine. On the other hand, job characteristics of a production machine are more relevant for predictive studies as they reflect the behaviour of a multitude of real user. Test clusters tend to have synthetic and small-scale workload submitted from a smaller set of users with larger idle time, which limits generality.

In this paper, we fill the lack of resources for job power prediction. We propose an approach to extract job power consumption data from workload manager data and node power metrics logs, which can easily be obtained through system plugins. Moreover, we present a large job dataset named PM100, with fine-grained job power consumption information. The PM100 dataset is derived from the M100 workload [8], a holistic dataset extracted from a large-scale production HPC, using the approach that we propose and is accessible through Zenodo [3]. Finally, we provide an overview of PM100, by analysing the data and discussing how it can be exploited for different prediction tasks.

To the best of our knowledge, the only publicly available dataset for job power consumption is presented in [13]. The dataset contains 80K jobs using CPU cores and is extracted from two production HPC clusters, with 560 and 728 nodes, respectively. Our work differs from [13] in a number of ways. First, our dataset is based on the first holistic dataset M100 of a more powerful tier-0 production supercomputer (Marconi100) and contains many more jobs (80k vs 230k). Second, as the nodes of Marconi100 are equipped with GPUs, the collected power consumption data refers to two types of jobs (those using only cores and also GPUs). Third, the data presented in [13] lack job information present in PM100, such as the job exit state, making the dataset not suitable for the job failure analysis and prediction tasks presented in Section 4. Finally, our approach can be applied to any public workload data with node power metrics logs so as to create new job power consumption datasets. With our work, we strive to empower the HPC community with tools to drive research in optimizing system performance and power consumption.

In the rest of the paper, we first briefly describe the M100 workload dataset (Section 2), then explain our methodology to create the PM100 dataset starting from M100 (Section 3), finally give an overview of PM100 (Section 4) and conclude (Section 5).

2 M100 DATASET

Marconi100 is a tier-0 production supercomputer hosted at CINECA[1] in Italy, and at the time of writing (July 2023), is ranked 26^{th} in the top500 list.[2] The cluster is composed of 980 computing nodes, and all the components are connected by a Mellanox Infiniband EDR DragonFly+ 100 Gb/s network infrastructure. Computing resources are allocated through job submission to the workload manager Slurm installed in the system. Technical details of the system are summarized in Table 1.

The M100 workload data [8] is collected during two and a half years of operation of Marcon1100. It is the first holistic dataset of a tier-0 supercomputer, and it is the largest (49.9 TB in size before compression) publicly available. It contains data ranging from the computing nodes' internal information such as core load, temperature, power consumption, to the system-wide information, including the liquid cooling infrastructure, the air-conditioning system, the power supply units, workload manager statistics, and job-related information. For our purposes, we are interested in the job data and node power metrics.

System characteristic	Description
#Nodes	980
#Processors (per node)	2x16 cores IBM POWER9 AC922, 3.1 GHz
#Accelerators (per node)	4 x NVIDIA Volta V100 GPUs, 16 GB
#CPU cores (per node)	32
Amount of RAM (per node)	256 GB
Peak performance	32 PFlop/s

Table 1: Marconi100 system characteristics.

Job data. The job data is collected in the job table plugin. We focus on the data that describes the jobs present in the workload by features related to their submit-time, run-time and end-time. The first category contains the information available when a job is submitted, such as submission time, number of requested resources, user information and system state. The second category comprises the information about the job launch, such as waiting time, execution start time, and the actual number of allocated resources. At job termination, the end-time features are collected, e.g., ending time, duration and outcome of the execution. The full list of job features is available at the dataset repository.[3] The job termination features do not contain job power consumption, so we need to extract this information from the power metrics logs of the nodes present in M100.

For the purposes of our work, we consider only a part of the dataset[4] and use only the data collected between May 2020 and October 2020. The reason is that this is the only period where the dataset contains information on the requested resources, which is useful to give a more in-depth description of each job and could be exploited for prediction tasks. The considered period contains around 1 million jobs.

Node power metrics. The power consumption data of the system components is contained in the IPMI plugin, which collects several metrics on cluster nodes, such as *ambient temperature, node temperature, fan speed, node power, CPU power, memory power.* The full list of metrics present in the IPMI data, and their relative sampling time, is reported in the original documentation of the dataset.[5] The values of all the metrics are collected every 20 seconds on all the system nodes. The final data is saved in a table, divided by node and collection time.

For our purposes, we consider only the metrics $ps0\_input\_power$ and $ps1\_input\_power$, which contain the power consumption values recorded at the input of the two power supplies of the nodes. Thus, the power consumption of a node n at time t_i can be obtained by summing $ps0\_input\_power_{n,t_i}$ and $ps1\_input\_power_{n,t_i}$.

[1] https://www.cineca.it
[2] https://www.top500.org

[3] https://gitlab.com/ecs-lab/exadata/-/blob/main/documentation/plugins/job_table.md
[4] https://doi.org/10.5281/zenodo.7588815
[5] https://gitlab.com/ecs-lab/exadata/-/blob/main/documentation/plugins/ipmi.md

3 PM100 DATASET CREATION

In this section, we first discuss the information that needs to be included in the data to apply our methodology. Then, we describe how we selected the jobs in M100 to include in the dataset PM100; this step is crucial to guarantee data soundness. We then explain how to extract job power consumption data starting from the power metrics logs of the nodes present in M100. At the end, we discuss the job features present in PM100.

Data requirements. The methodology we propose in this work can be applied to any workload manager data and node power metrics logs, providing the following information.

The workload manager data need to contain, for each job j, its start time, end time, and the nodes $nodes_j$ allocated to the job j. This information will be used to filter out the exclusive jobs and to extract the job j power consumption p_j.

The node power metrics logs must keep record of the power consumption values of the single nodes of the systems, at different timestamps t_i. Such values are used to compute the job j power consumption p_j, at the different time t_i.

In this work, we consider the node n power consumption at time t_i as the sum of $ps0\_input\_power_{n,t_i}$ and $ps1\_input\_power_{n,t_i}$, as discussed in Section 2. However, if the $ps0\_input\_power$ and $ps1\_input\_power$ values are missing, the node power consumption can be defined differently, in accordance with the data present in the node power logs.

Job filtering. In Marconi100, multiple jobs can run on the same node at the same time. Therefore, node power consumption depends on the power consumption of the execution of multiple jobs on the node's resources. We are not aware of a methodology to evaluate accurately the contribution of each job execution to the node power consumption. Thus, we consider only the jobs that run alone on all the allocated nodes throughout their execution, to provide accurate power consumption information.

In order to filter out the exclusive jobs, we implement a pipeline defined as follows. First, we create a hash-table for each node n, where the keys are the timestamps t_i of a fixed period of time Δ, sampled every θ seconds. The value related to the key t_i is a list containing the IDs of the jobs running on n during that particular timestamp t_i. For each job j, we round the start time (ceiling rounding) and the end time (floor rounding) to the closest t_i (referred to as $start\_time_j$ and end_time_j). We then add the job ID to the lists of all the t_i that fall between $start\_time_j$ and end_time_j, for all the nodes $n \in nodes_j$ allocated to j. Finally, from the resulting tables, we filter out all the jobs j that are the only members of the lists hashed by t_i falling between $start\_time_j$ and end_time_j for all $n \in nodes_j$.

Despite considering all the t_i of all the jobs is a costly operation, it can be very useful for future work. For instance, it can be used to study the power consumption of concurrent jobs and how their power consumption intertwine during their execution. Moreover, one can augment the data by considering the power profiles of the concurrent jobs just in the timespans where they ran alone.

Job power consumption extraction. In order to extract job power consumption, we need to perform a data post-processing pipeline to correlate each job to the power consumption caused by its execution. We define the power consumption of an exclusive job j as a list p_j, where each element is the job power consumption computed at t_i, for all the t_i intersecting the job execution. The job power consumption at time t_i is obtained as the sum of the power consumption of the nodes $n \in nodes_j$ allocated to the job j (p_nodes_{j,t_i}). As mentioned in Section 2, the power consumption of a single node n at time t_i is the sum of $ps0\_input\_power_{n,t_i}$ and $ps1\_input\_power_{n,t_i}$. Thus, p_nodes_{j,t_i} can be computed as shown in Equation 1. Combining the definitions, we can create the job power consumption values list as shown in Equation 2.

Input : $job\ j, ps0\_input\_power, ps1\_input\_power$
Output : p_j
$p_j = []$
$t_i = start\_time_j$
$\theta = 20$
while $t_i \leq end\_time_j$ **do**
  $p\_nodes_{j,t_i} = 0$
  for $n \in nodes_j$ **do**
    $p\_nodes_{j,t_i} = p\_nodes_{j,t_i} + ps0\_input\_power_{n,t_i} + ps1\_input\_power_{n,t_i}$
  end
  $p_j = p_j + [p\_nodes_{j,t_i}]$
  $t_i = t_i + \theta$
end

Algorithm 1: Job power consumption extraction

$$p\_nodes_{j,t_i} = \sum_{n \in nodes_j} ps0\_input\_power_{n,t_i} + ps1\_input\_power_{n,t_i} \tag{1}$$

$$p_j = [p\_nodes_{j,start\_time_j}, \ldots, p\_nodes_{j,end\_time_j}] \tag{2}$$

Algorithm 1 lists the steps performed to extract the power consumption of each exclusive job filtered in the previous step. For each job j that has $end\_time_j - start\_time_j > 0$, we initialize p_j as an empty list. The length of p_j will be $end\_time_j - start\_time_j/\theta$, where θ is the sampling time of the power values in the power data (20 seconds in our case), namely the distance in time between two consecutive power measurements in the data. We iterate over all the t_i intersecting the execution of j, (i.e. $start\_time_j \leq t_i \leq end\_time_j$), and we compute the p_nodes_{j,t_i}. At the end of each iteration, p_nodes_{j,t_i} is added to p_j and t_i is updated. The algorithm returns p_j containing the power consumption values of the job j during its execution.

The code to perform the full pipeline is available on the GitHub repository[6] of the dataset. We run the scripts on a machine endowed with 32 cores and 256 GB of RAM, and the runtime of the whole process was around 8 weeks, with only the job power consumption extraction requiring more than 7 weeks. The lengthy computation is mainly due to the size of the initial dataset, which contains around 1 million jobs. The filtering process removed all the jobs that do not run exclusively on the nodes and with which we encountered problems due to missing values in the power data. The final dataset (~ 100 MB) contains 231,238 jobs and is accessible through Zenodo [3].

[6] https://github.com/francescoantici/PM100-data

Job features. Each job in PM100 is represented with a set of features, most of which are imported from the original dataset M100. After inspecting the original feature values, we notice that some information is duplicated or not meaningful enough. For instance, multiple fields are related to the names of the allocated nodes, so we aggregate them into a single feature called *nodes*. We also observe that some features contain the same information in different formats, e.g., *time_limit_str* and *time_limit*, thus we keep only one of such features. We further remove the features that have the same values across all the jobs, such as *power_flags*.

After this initial pre-processing, we modify some features to make the underlying information more explicit and easy to access. In particular, we engineer the features related to the resources. In the original data, each job record contains two fields listing the amount of requested and allocated resources, namely *tres_req_str* and *tres_alloc_str*. This data is structured as a comma-divided list of features (#nodes, #cores, #GPUs, and amount of RAM) and their values, e.g. *"nodes=1,cpus=32,gpu=1,mem=256G"*. We unpack the list of features and their values to individual features f_k, where $f \in \{num\_nodes, num\_cores, num\_gpus, mem\}$ and $k \in \{req, alloc\}$. Finally, we add the $power\_consumption$ feature, which is the list p_j of job power consumption values recorded during its execution, as computed in the previous paragraph.

Overall, each job has 32 features, which are documented in the GitHub repository of the dataset.

4 PM100 DATASET OVERVIEW

In this section, we offer an overview of the PM100 dataset. First, we conduct an Exploratory Data Analysis (EDA) to understand the dataset and its underlying patterns, such as distributions, correlations, and relationships between certain features. As the description of the original M100 dataset [7] did not provide any such analysis, we first start with the jobs and then continue with their power consumption. This process provides insights into the nature of the data, which could be useful for prediction purposes and choosing the appropriate modeling techniques. We conclude the section with some examples of prediction tasks that can be performed with the dataset.

4.1 Job analysis

With job analysis, we investigate the classical workload characteristics, such as job submission date, exit state, duration, and allocated resources.

Submission date. In Figure 1, we plot the distribution of the number of jobs (y-axis) submitted per day (x-axis), and its Kernel Density Estimate (KDE). The KDE (blue solid line) provides a probability density function estimation of the job submission per day, and it is plotted to spot patterns or seasonality more easily. Although we may expect a uniform distribution of job submission throughout the days, the figure shows that neither seasonality nor uniformity is present in the data. This is not unexpected. The PM100 dataset does not include all the jobs submitted to the system. Also, in a real HPC environment, jobs maybe submitted non-uniformly due to several reasons, like scheduled maintenance, holidays, and node failures and outages. Moreover, occasionally, it may be necessary to dedicate (a part of) the cluster to certain computations, as it happened in our case with COVID-19-related tasks.[7]

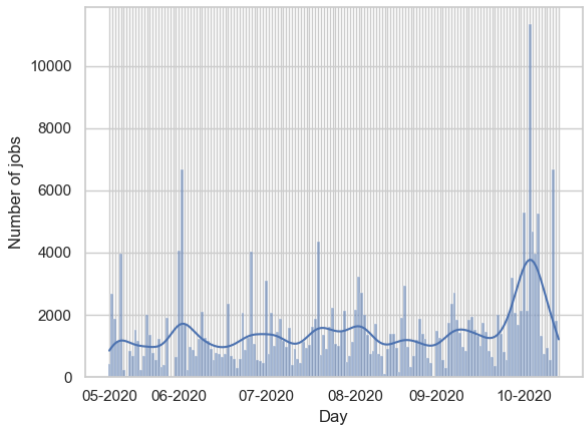

Figure 1: The distribution of the jobs throughout the days.

Exit state. The exit state of a job represents its execution outcome. Since the jobs in our dataset are executed on a production machine, we expect to have a workload composed of mainly successfully completed jobs. This is because production machines are not used for tests, but only to execute stable jobs. We present in Figure 2 the distribution of the possible outcomes. As expected, the vast majority of the jobs (77%) are successfully *completed*. The second most frequent category is *failed*, with the 14%. These are the jobs that fail due to generic errors encountered during their execution, such as bugs in the code or errors in the job script. Some jobs are *cancelled* by their user during their execution due to reasons like discovery of bugs or errors in the code by checking intermediate results or of misconfigurations in batch scripts (e.g. run-time duration configuration is not enough). They represent 5% of the jobs in the dataset. With similar percentage (4%), there is the *timeout* category, referring to the jobs that exceeded the time limit set by the user or the system. Less than 1% of jobs run *out of memory* (*oom*) due to misconfiguration and underestimation of memory requirements. Similarly, less than 1% of the jobs fail due to *node fail*. This phenomenon is very rare in production machines, indeed, *oom* and *node fail* jobs combined account for around 1%. Even though the dataset does not reflect the whole system load, the expected unbalancing towards the successfully completed jobs is still present and yet there exist a significant amount of jobs that did not successfully complete (~ 50K).

Duration. In Figure 3, we present the job duration distribution. In an HPC system, job duration may range from a few seconds to several days, depending on the allocated resources, the operations performed, and the system requirements (some systems may allow

[7] Quoting an e-mail from the HPC system support to the users: "Due to a COVID-related urgent computing activity, most of M100 nodes will not be available for the standard production from Friday, November 6, 4 pm, to Monday, November 9, 4 pm. This will cause a significant increase in the waiting times and the impossibility to run jobs with more than 60 nodes".

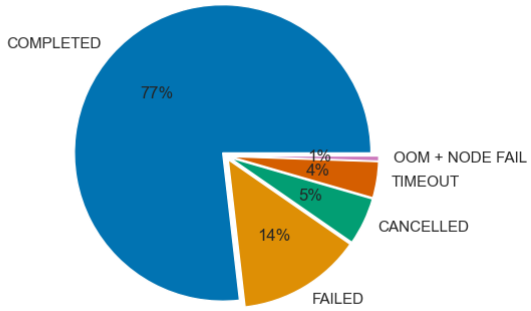

Figure 2: Distribution of job exit state.

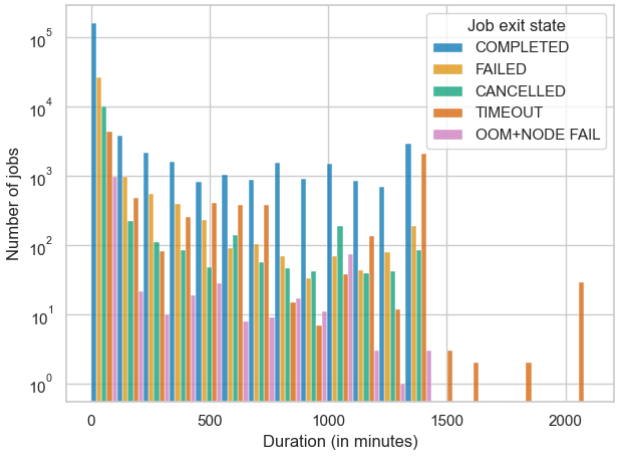

Figure 3: Distribution of job duration, divided by exit state.

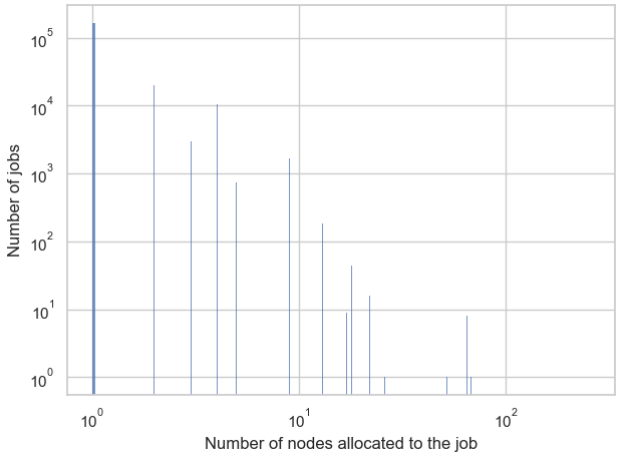

Figure 4: Distribution of the number of allocated nodes.

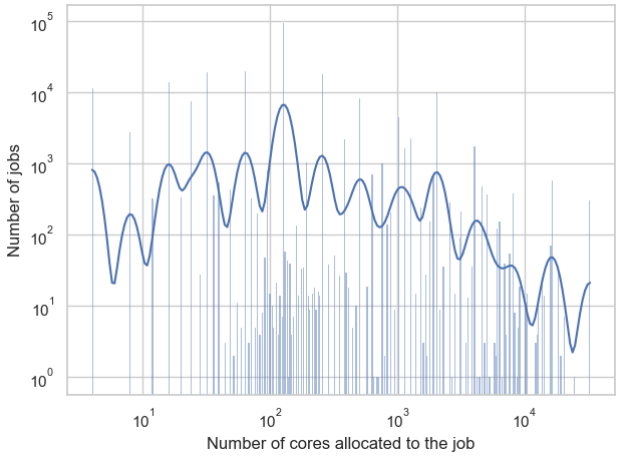

Figure 5: Distribution of the number of allocated cores.

long executions while others not). The figure reveals that the majority of jobs has a duration of less than 100 minutes. Very few jobs run for more than a day, meaning that the jobs in our dataset are mainly short to medium length. This can be a characteristic of the users (not submitting jobs that require excessive computation) or due to a system setting (lengthy executions are not allowed). In the figure we also investigate how job duration is related to job exit state. We observe all possible outcomes with jobs running up to 1300 minutes. We observe further that the jobs that run for more than 1300 minutes all timeout. This is consistent with the information provided in the official documentation of Marconi100[8], stating that the system limits the job duration to 24 hours (1440 minutes) except some particular cases.

Allocated resources. The dataset is dominated by the jobs using both cores and GPUs (91%). In Figures 4, 5, 6, and 7, we show the distribution of the amount of allocated nodes, cores, GPUs and RAM, respectively. From Figure 4, we can conclude that the majority of the jobs uses just one node for their execution. No job uses a significant portion ($\geq$ 20%) of the system nodes (980 in total), meaning that the jobs in our data mainly represent small-scale executions. This is also reflected in Figures 5, 6, and 7. Indeed, we notice an evident spike in these figures in correspondence to the amount of cores (128), GPUs (4), and RAM of a node (256 GB).

4.2 Job power consumption analysis

With job power consumption analysis, we investigate whether jobs differ in their power consumption values and trends, as well as the influence of GPU usage.

Power consumption. As discussed in Section 3, *power_consumption* is the feature containing the list of power consumption values of the job throughout its execution. We start our analysis by plotting in

[8]https://wiki.u-gov.it/confluence/pages/viewpage.action?pageId=336727645

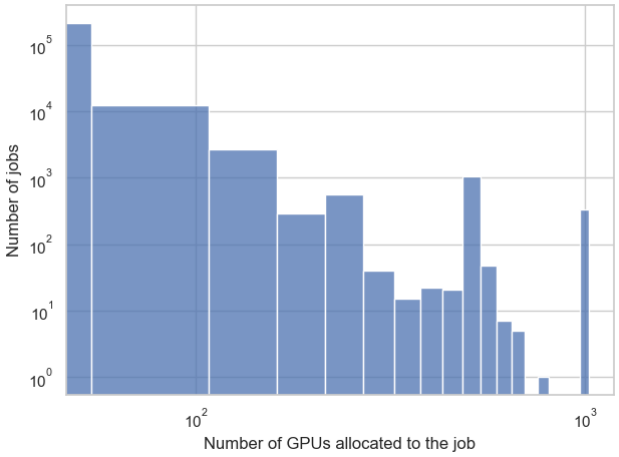

Figure 6: Distribution of the number of allocated GPUs.

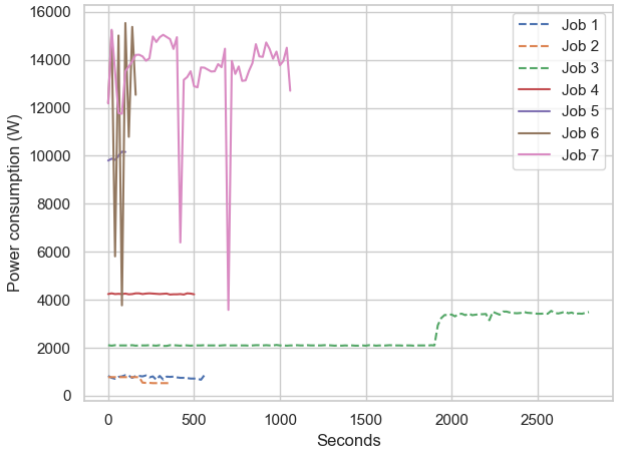

Figure 8: Job power consumption during execution.

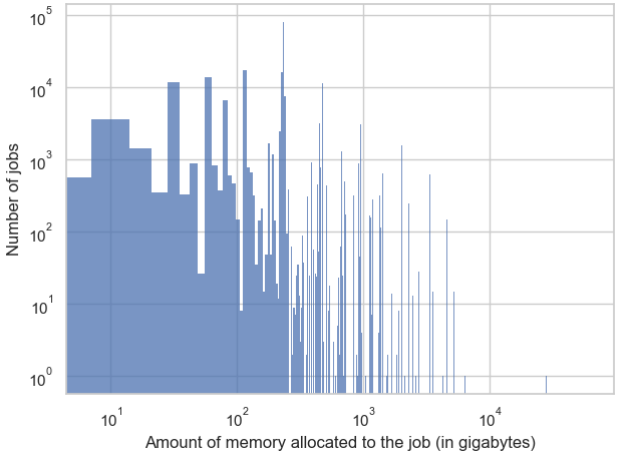

Figure 7: Distribution of the amount of allocated RAM.

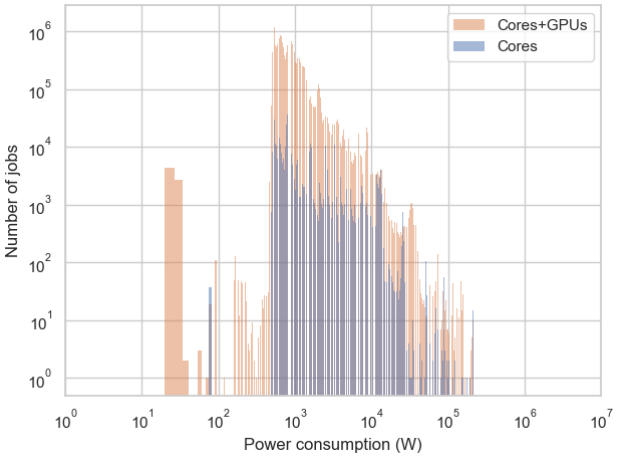

Figure 9: Distribution of job power consumption.

Figure 8 the *power_consumption* values of 7 randomly chosen jobs whose resource usage is reported in Table 2. We observe that the jobs with similar features (single-node jobs 1 and 2) tend to behave similarly in their power consumption. As the number of allocated nodes, cores and GPUs increase, so does the power consumption values. We also observe that the consumption trend of the jobs vary. For instance, while jobs 6 and 7 fluctuate between higher and lower values, the others show a smoother behaviour. We can conclude that our dataset contains a diverse set of jobs in terms of power usage.

GPU influence on power consumption. Since the jobs are extracted from a heterogeneous machine, we want to explore how the GPU usage impacts power consumption. In this analysis, we consider the individual values of the jobs' instantaneous power consumption at each time point rather than their time distribution. Thus, we merge all the values of all the power consumption lists into a single set, independently of the job. Then, we use this data to plot Figures 9 and 10. In Figure 9, we show the distribution of the set of power consumption values by distinguishing between the jobs using only cores (9%) from those using also GPUs (91%). The figure shows that jobs using only cores tend to reach lower power values compared to the ones using also GPUs. This behavior is normal in heterogeneous systems, since GPUs have higher functional unit density and are usually characterized by a larger Thermal Design Power (TDP) than cores. In Figure 10, we focus on the single-node jobs and plot the distribution of the instantaneous power consumption values, again by separating the jobs using only cores from those using also GPUs. We see that independently of the number of allocated nodes, the range of power consumption values of jobs using also GPUs are higher than those using only cores, validating the findings of the analysis of the previous plot.

Job #	#Nodes	#Cores	#GPUs	RAM
1	1	40	4	74
2	1	128	4	237
3	4	64	16	512
4	8	1024	32	1900
5	12	1536	48	2695
6	16	2048	64	3800
7	20	1280	80	4804

Table 2: Amount of allocated resources of jobs in Figure 8.

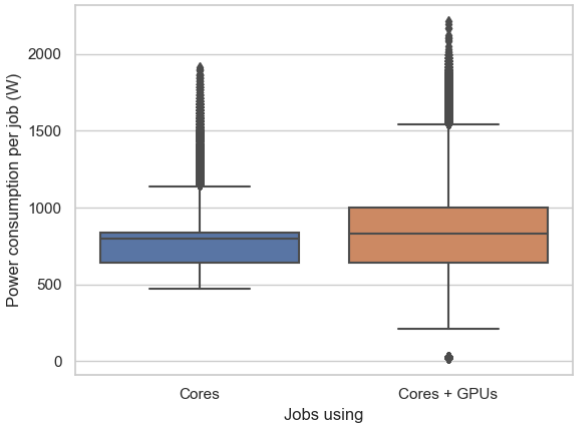

Figure 10: Power consumption of single-node jobs.

4.3 Prediction Tasks

Our dataset enables performing various prediction tasks in HPC systems such as job duration prediction, job failure prediction, and job power consumption prediction.

Job duration prediction. This concerns forecasting the execution duration of a job before its allocation in the system. This information can be useful to develop dedicated workload management strategies, as shown in [10], aiming at improving system performance and achieving high quality of service (QoS) levels. Past work, such as [1, 17], explored the task by relying on machine learning techniques, while others like [10] tackled the problem by employing a data-driven heuristic algorithm in the scope of an online job dispatching problem. Such approaches can easily be performed on our dataset by exploiting the *run_time* feature of the jobs as target.

Job failure prediction. This is one of the hottest topics in the area of workload prediction, and it concerns forecasting possible failures during the execution of a job before its allocation in the system. Failing jobs unnecessarily occupy resources which could delay other jobs, adversely affecting the system performance, QoS and power consumption. Similar to the job duration prediction, forecasting failures a priori would allow to adopt ad-hoc workload management strategies, as shown in [11]. Several past work addressed the job failure prediction task. For instance, [2, 4, 11, 12] relied on data-driven techniques aimed at predicting job failure by analysing workload features. As discussed in Section 4.1, our dataset contains the exit state of each job, which can be found in the *job_state* feature and used as a target in a classification task.

Job power consumption prediction. This is about predicting the power consumption caused by the execution of a job on the system. It can assist the development of power-aware workload management techniques to optimize the system performance and power consumption, as shown in [6]. The prediction can be performed in different ways; for instance, by predicting the power consumption values of a job throughout its execution time, as done in [15]. Alternatively, it can be performed by predicting the average, or the maximum power consumption value, as done in [5, 13, 14]. The data in PM100 can be used for both purposes. As discussed in Section 3, each job has the *power_consumption* feature, which is a list of power consumption values computed at each timestamp intersecting the job execution. This feature can be used as it is for the first purpose, and its average and maximum values can easily be computed for the second.

5 CONCLUSIONS

In this paper, we discussed the importance and yet the scarcity of publicly available job datasets including power consumption information extracted from a production HPC system. To fill the lack of resources, we proposed an approach to extract job power consumption data, which is designed to create datasets starting from any workload manager data and node power metrics logs. We then presented PM100, a large dataset containing around 230K jobs with fine-grained job power consumption information. PM100 is derived from the M100 workload [8], a holistic dataset extracted from a large-scale production HPC, using the approach that we proposed and is accessible through Zenodo [3]. To provide insights into the nature of the PM100 data, we conducted an EDA, showing the characteristics of the jobs and their power consumption. We finally presented some of the possible prediction tasks that can be performed using PM100.

In future work, we plan to add in the dataset the power consumption of the compute resources (cores, GPU, and RAM). In M100, there are specific metrics for this information at the node level. Thus, knowing the resources allocated to each job and that the dataset contains only jobs running alone in the nodes, we can extract the job power consumption at compute resources level. Moreover, we plan to study techniques to isolate the power consumption of concurrent jobs, so not to remove them from the original data. As another future work, we want to find new workload data sources so as to apply our methodology and create new power datasets and increase the number of public data for job power prediction tasks.

ACKNOWLEDGMENTS

This study has been partially funded by the EU H2020-JTI-EuroHPC-2019-1 project REGALE (g.n. 956560).

REFERENCES

[1] Nasim Ahmed, Andre LC Barczak, Mohammad A Rashid, and Teo Susnjak. 2022. Runtime prediction of big data jobs: performance comparison of machine learning algorithms and analytical models. *Journal of Big Data* 9, 1 (2022), 67.

[2] Francesco Antici, Andrea Borghesi, and Zeynep Kiziltan. 2023. Online Job Failure Prediction in an HPC System. In *European Conference on Parallel Processing Workshops*. Springer.
[3] Francesco Antici, Mohsen Seyedkazemi Ardebili, Andrea Bartolini, and Zeynep Kiziltan. 2023. *PM100: A Job Power Consumption Dataset of a Large- Scale HPC System*. https://doi.org/10.5281/zenodo.8129258
[4] Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop. 2021. A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms. *Journal of Advances in Information Technology* 12, 2 (2021).
[5] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. 2016. Predictive modeling for job power consumption in HPC systems. In *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings*. Springer, 181–199.
[6] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. 2018. Scheduling-based power capping in high performance computing systems. *Sustainable Computing: Informatics and Systems* 19 (2018), 1–13.
[7] Andrea Borghesi, Carmine Di Santi, Martin Molan, Mohsen Seyedkazemi Ardebili, Alessio Mauri, Massimiliano Guarrasi, Daniela Galetti, Mirko Cestari, Francesco Barchi, Luca Benini, et al. 2023. M100 ExaData: a data collection campaign on the CINECA's Marconi100 Tier-0 supercomputer. *Scientific Data* 10, 1 (2023), 288.
[8] Andrea Borghesi, Carmine Di Santi, Martin Molan, Mohsen Seyedkazemi Ardebili, Alessio Mauri, Massimiliano Guarrasi, Daniela Galetti, Mirko Cestari, Francesco Barchi, Luca Benini, Francesco Beneventi, and Andrea Bartolini. 2023. M100 dataset. https://gitlab.com/ecs-lab/exadata
[9] Tomasz Ciesielczyk, Alberto Cabrera, Ariel Oleksiak, Wojciech Piątek, Grzegorz Waligóra, Francisco Almeida, and Vicente Blanco. 2021. An approach to reduce energy consumption and performance losses on heterogeneous servers using power capping. *Journal of Scheduling* 24 (2021), 489–505.
[10] Cristian Galleguillos, Zeynep Kiziltan, Alina Sîrbu, and Ozalp Babaoglu. 2019. Constraint programming-based job dispatching for modern HPC applications. In *Principles and Practice of Constraint Programming: 25th International Conference, CP 2019, Stamford, CT, USA, September 30–October 4, 2019, Proceedings 25*. Springer, 438–455.
[11] Mohammad S Jassas and Qusay H Mahmoud. 2022. Analysis of job failure and prediction model for cloud computing using machine learning. *Sensors* 22, 5 (2022), 2035.
[12] Jie Li, Rui Wang, Ghazanfar Ali, Tommy Dang, Alan Sill, and Yong Chen. 2023. Workload Failure Prediction for Data Centers. *arXiv preprint arXiv:2301.05176* (2023).
[13] Tirthak Patel, Adam Wagenhäuser, Christopher Eibel, Timo Hönig, Thomas Zeiser, and Devesh Tiwari. 2020. What does power consumption behavior of hpc jobs reveal?: Demystifying, quantifying, and predicting power consumption characteristics. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 799–809.
[14] Théo Saillant, Jean-Christophe Weill, and Mathilde Mougeot. 2020. Predicting job power consumption based on rjms submission data in hpc systems. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35*. Springer, 63–82.
[15] Shigeto Suzuki, Michiko Hiraoka, Takashi Shiraishi, Enxhi Kreshpa, Takuji Yamamoto, Hiroyuki Fukuda, Shuji Matsui, Masahide Fujisaki, and Atsuya Uno. 2021. Power Prediction for Sustainable HPC. *Journal of Information Processing* 29 (2021), 283–294.
[16] Sean Wallace, Xu Yang, Venkatram Vishwanath, William E Allcock, Susan Coghlan, Michael E Papka, and Zhiling Lan. 2016. A data driven scheduling approach for power management on hpc systems. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 656–666.
[17] Qiqi Wang, Hongjie Zhang, Jing Li, Yu Shen, and Xiaohui Liu. 2022. Predicting job finish time based on parameter features and running logs in supercomputing system. *The Journal of Supercomputing* 78, 17 (2022), 18551–18577.