This is the post peer-review accepted manuscript of:

F. Antici, A. Borghesi, J. Domke, and Z. Kiziltan, "UoPC: A User-Based Online Framework to Predict Job Power Consumption in HPC Systems" in ISC High Performance 2025 Research Paper Proceedings (40th International Conference), pp. 1-12, June 2025.

The published version is available online at:

https://doi.org/10.23919/ISC.2025.11017729

UoPC: A User-based Online Framework to Predict Job Power Consumption in HPC Systems

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

Andrea Borghesi
University of Bologna
Bologna, Italy
andrea.borghesi3@unibo.it

Jens Domke
Riken Center for Computational Science
Kobe, Japan
jens.domke@riken.jp

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

***Abstract*—It is fundamental to design accurate workload power prediction techniques to address environmental sustainability challenges in modern high-performance computing (HPC) systems. While existing Machine Learning (ML) approaches are effective, they retain some limitations in production environments. To address these, we introduce UoPC, a user-based online framework for predicting job power consumption in HPC systems. UoPC leverages ML-based predictive models tailored for individual users, eliminating the need for voluminous data and training. It offers a user-friendly Python implementation suitable for both end-user usage and integration into workload management systems. We evaluate UoPC on more than 1.3 million jobs executed on Fugaku,[1] a supercomputer hosted at RIKEN, demonstrating its effectiveness. It achieves only a 10% prediction error, with minimal overhead on the system operations. By employing a k-nearest neighbours (KNN) prediction model augmented with Natural Language Processing (NLP), UoPC streamlines prediction processes for newly submitted jobs. It requires only limited historical data, making it practical for diverse high-performance computing environments and workloads.**

I. Introduction

High-Performance Computing (HPC) systems are fundamental to process vast amounts of information, as they provide powerful architectures capable of addressing the ever-expanding computing needs of today's society. On the other hand, their increasing power consumption poses a significant threat to their environmental sustainability [29]. Thus, optimizing the energy efficiency of such systems is pivotal to guarantee scientific progress, while reducing the operational costs and environmental footprint [17]. This can be achieved, for instance, by developing software-based techniques operating at the workload management level [9], [27], [31]. For this approach, it is critical to accurately predict the power requirements of HPC jobs to devise proactive power-aware workload management strategies [1], [21], [26], [42].

Recent work has shown that Machine Learning (ML)-based methods can be effective in a context where users continuously submit their jobs to a system [5], [39]. The proposed solutions predict the power consumption of a job before its execution, leveraging only the information available at job submission time and the historical data gathered on past job executions. To adapt to the changes in the workload submission and provide accurate prediction, the predictive models are employed *online* on live-streaming data by periodically retraining them on recently submitted job data.

While these approaches are valid, they present some limitations when employed in production environments. Model training requires voluminous historical execution data of jobs submitted by diverse users, as otherwise prediction effectiveness can be compromised [28]. In a real system, however, it is non-trivial to collect large amounts of data from multiple users. This can be due to privacy concerns [22] or difficulties in the data collection phase (caused by for instance different privilege levels requirements and monitoring software validation procedures) [4], [7]. An observation is that in production environments, users tend to submit jobs performing similar operations (due to for instance running similar experiments) and consequently with similar power consumption [40]. Hence, job power consumption prediction using only the historical data of its own user is likely to improve accuracy, while eliminating the need for large amounts of data from various users. However, there can be significant variances in the job power consumption values of even a single user (due to for instance change of experiments), which renders the prediction task non-trivial, requiring a prediction model tailored to a user's behaviour and job execution characteristics.

Another limitation of the existing approaches is that they are usually designed to be employed at the system level, without taking into account the end-users. From a user perspective, knowing the energy cost of their jobs is useful in systems where power-based pricing strategies are in place [23]. It has been argued that to encourage the adoption of greener systems, the energy cost of jobs needs to be made explicit and accounted for in the pricing scheme [10]. In this context, users also need tools to estimate the power consumption of their jobs before submission, for accounting reasons and to better plan their workload.

To address the aforementioned limitations and challenges, we introduce a new online framework UoPC to predict the power consumption of jobs submitted to production HPC environments.[2] Our contributions can be summarized as follows:

- We design UoPC, which exploits ML-based predictive models. UoPC eliminates the need for model training and large amounts of data. It builds a separate and simpler

[1]https://www.fujitsu.com/global/about/innovation/fugaku/

[2]https://github.com/francescoantici/UoPC/settings

model for each user by relying solely on the user's data, differing from the existing approaches.
- We provide an easy-to-use Python implementation of UoPC, which can be either used by the end-users as a stand-alone tool or integrated into the workload management system to enable power/energy-aware job scheduling strategies.
- We deploy UoPC for the Supercomputer Fugaku, a production HPC system hosted at the RIKEN Center for Computational Science, in Japan, and evaluate it on a very large dataset of job runs on Fugaku, obtaining a prediction error of only 10%, while incurring a small overhead on the standard workload submission workflow.

UoPC exploits the k-nearest neighbours (KNN) [20] prediction algorithm, which is non-parametric and thus does not require model training. The KNN is augmented with Natural Language Processing (NLP) tools to encode the textual features present in the job data. The framework streamlines the standard prediction process for a newly submitted job before its execution, requiring only the information available at job submission time and the data of k (which can be as low as 5) previous job executions by the same user. This makes our approach more practical for production environments than the existing solutions. Moreover, the implementation requirements of UoPC transcend architectural boundaries, offering a solution applicable to different systems and workloads.

We tested UoPC on the job execution data of more than 1.3 million jobs submitted to Fugaku, by more than 700 different users between February and May 2024. Our experimental results show that UoPC outperforms recent ML-based solution in predicting the *average* and *maximum* job power consumption while significantly reducing the overhead on the system operation and the amount of historical data needed. Our approach has also proven effective in predicting the system power consumption, obtaining an error of only 4%.

The rest of the paper is organized as follows. After we give the necessary background in Section II, we introduce in Section III the UoPC framework. Then we discuss its deployment for the Fugaku Supercomputer in Section IV and present in Section V our experimental study. We finally conclude in Section VI.

II. Background

In this section, we first present the ML and NLP models employed to predict job power consumption and then discuss the related work.

A. ML Models

The Random Forest (RF) [11] is a well-known ensemble ML algorithm leveraging several independent Decision Tree (DT) model instances for regression tasks. The RF model is trained on historical data to compute statistical correlations between input feature values and a given prediction target. Each DT is trained individually by tuning the internal parameters on a random subset of the training data and a random subset of the input features. At inference time, a majority voting among the trained DTs is carried out. This is done to make up for the tendency of individual DTs to overfit on the training data and thus obtain less error-prone prediction performance, as explained in [11]. In general, the RF benefits from having a voluminous amount of past data, since the computation of statistical correlation becomes more accurate for the given sample.

The k-nearest neighbours (KNN) [20] is a widely used instance-based algorithm for regression tasks. Contrarily to the typical ML-based methods, it does not require a training set to build a prediction model, since KNN does not rely on internal parameters. During model building, it stores a fixed amount of historical data points in a given feature space. Then, at inference time, it computes the k-nearest neighbours as the most similar elements among the stored ones, according to a distance metric, such as Minkowski. Because of this, the minimum amount of data needed to perform prediction is k.

Sentence Bert (SBert) is a state-of-the-art sentence embedding model, obtained by fine-tuning pre-trained BERT (Bidirectional Encoder Representations from Transformers) in sentence similarity tasks [19]. BERT is trained on millions of textual documents to understand language patterns. The model can generate word-level embeddings, namely a semantically meaningful floating-point array representation. However, when working with pieces of text or strings in regression tasks, this representation is impractical [15], [16]. Conversely, SBert generates meaningful sentence-level embeddings, i.e. a 384-dimensional floating-point array.

B. Related Work

Several past works tackled the job power consumption prediction task through the analysis of workload data, but they differ from our approach in various aspects.

In [12], [43], [48], the authors propose models using job information not limited to submission-time features. This makes the prediction infeasible prior to job execution, and not suited for our work. Instead, in [5], [8], [39], [40], the prediction is performed leveraging only submission-time information, and we limit the discussion of related work to such approaches.

In [8], [14], job average power consumption per node is predicted via an RF model trained on historical HPC jobs data. Also in [40] the RF is used to perform the task, and this time in a user-based fashion, similar to ours. Both approaches train the model only once, without updating the model with the most recent data. This has been demonstrated to be less effective, in terms of prediction accuracy, w.r.t. a prediction algorithm like ours which is updated periodically on recent data [2], [3], [5]. Moreover, they train the model with a large amount of data which may not be always available, as discussed previously, especially per user. If we were to retrain the model, the approach of [40] would be even more impractical in a real system, which are used by hundreds of users. Hence, training a model per user with a large amount of data would result in excessive overhead on the system operations.

Similarly to our approach, in [5], [14], [39] the models are updated periodically on more recent data, and in [5]

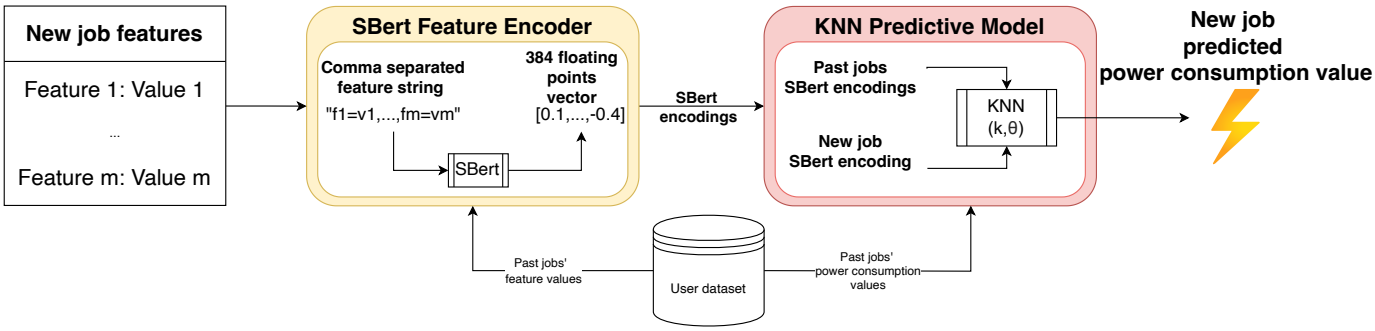

Fig. 1. High-level functioning and components of UoPC.

SBert is exploited to encode job data. However, the predictive algorithm of [5] uses the RF model while [39] relies on exponential smoothing of similar past jobs' power consumption. This approach relies only on categorical job features (*user id*, *group id*, *#tasks per node*) and is shown to be outperformed by an RF model trained only once on all the data (as in [8]). In [5], [8] the authors evaluate the performance at the system level as well. Our approach is more accurate than that of [8], as they obtain an error around 9% against our 4%. Concerning [5], they obtain a similar error on the average system power consumption (around 4%), but our approach is better for the maximum one (5% against our 4%). Despite being more accurate than the previous works, both approaches require a large amount of data from different users for training.

All the cited approaches are designed to work at the system level, and they do not provide a tool for end-users. We instead enable end-users to estimate the power consumption of their jobs, while improving the prediction accuracy at the same time. Finally, they all build their supervised models by merging the data of all the users, while our approach keeps separate the data of different users, thus preventing possible privacy concerns related to data sharing.

III. UoPC Framework

In this section, we describe UoPC's functioning. We first provide an overview of the framework, and then detail its components and implementation.

A. UoPC Overview

UoPC is designed to predict job power consumption in a real HPC system, where jobs are submitted and executed continuously, and various information regarding job submission, execution and completion (referred to as job data) is streaming in time. In this context, the prediction for a job before its execution can be performed leveraging exclusively features available at job submission, together with the historical data of the jobs that are completed by that time.

The only implementation requirement for UoPC is the presence of a data collection infrastructure gathering the job execution data per user. Such an infrastructure should update the *user dataset* with the record of completed jobs, including all the features regarding job submission (such as *requested resources*, *user information*, *job name*), job execution and completion time (such as *duration*, *#nodes allocated*, and *power consumption*). The user should be able to interact with their dataset at any time. This is not a strict requirement, since modern systems are typically endowed with monitoring software that permits the collection of job data [24], [30], [41], including power consumption [18], [32]. Moreover, systems usually provide a user-friendly interface to this data [33]. For users who are not system admins (usually the great majority), such tools allow them to retrieve only their data, due to privacy and security concerns. The data we require are restricted to a single user, therefore both privacy concerns and technical difficulties in the data collection do not represent a problem.

Practically speaking, our framework works as an inference engine without having to train large statistical models. UoPC takes as input the data of a new job submission. The high-level functioning of its prediction algorithm for a new job is presented in Figure 1. The main components of UoPC are:

- The SBert Feature Encoder, which takes as input a series of job feature values and returns the encoded data to be fed into the KNN Prediction Model.
- The *KNN Prediction Model*, which predicts the power consumption of a new job based on the encoded job data and the data of the past job executions of the same user.

We provide an easy-to-use Python implementation for UoPC, which can be used as a stand-alone tool by the end-user, or deployed in a workload management system. The components are software components implemented as Python classes, with a method for each functionality they provide.

B. SBert Feature Encoder

As job feature values are in a textual format, this component converts them into a numeric format suitable for the *KNN Prediction Model*. The conversion is performed by the *encode* method of the class, which takes as input a list of feature values describing the job at submission time (e.g. *"user_1","job_1","48","1","env_1,2000"*). Internally, the feature values are concatenated into a comma-separated string (e.g. *"user_1,job_1,48,1,env_1,2000"*) and encoded with an instance of a pre-trained SBert model. The final output of the *encode* method is a 384-dimensional floating-point vector, normalized in the range [-1, 1] (e.g. [-0.2,...,0.3]).

Past work [3], [5] demonstrated that an NLP encoding allows to extract more meaningful information from the job data, in the scope of predicting HPC job characteristics. We here identify additional advantages in terms of generality and data protection. Jobs submitted to different systems, or to the same system at different times, are likely to be described by different feature sets. SBert does not require a fixed feature set contrarily to typical ML models , thus can be used in a variety of settings. Additionally, we note that users tend to consider their job data sensitive and avoid its easy access and storage. Ideally, they prefer their data to be obfuscated and not available in a non-encoded format. SBert addresses this concern as it projects the information on a latent space. Moreover, this step complicates the association between the original user-sensitive information and the encoded data used for training [47].

C. KNN Predictive Model

The prediction model we opt for is the KNN, and a specific model instance is built for each user. Predicting a job power consumption based on its user allows to maximize the accuracy of the KNN algorithm, while minimizing the amount of data needed, as explained in Section I. Moreover, it also makes the framework suitable for systems where multiple user data may not be available (e.g. cloud or edge).

At initialization, the class requires the user dataset[3]. Then, it implements a *predict* method, which takes as input the SBert-encoded job submission data and generates a power consumption prediction for the job execution as follows:

1) The user dataset is queried to look for past job execution data.
   a) If the dataset has at least k data points, the algorithm moves on to the next step. Otherwise, the prediction cannot be performed, and an error is returned; this is due to the constraint on the minimum number of data points required by KNN.
   b) Conversely, if the dataset contains more than θ data points, we set a cap to the number of points that will be passed to the KNN module, to keep the inference time low (as applying KNN to a smaller set of points is computationally faster). To do that, we sort the data points by their completion time (w.r.t. the job arrival time) and keep the first θ points from the sorted list.
2) A KNN instance is built on the resulting past data, and it is used to generate power consumption prediction for the job execution.

The parameters k and θ are set by the UoPC user. Users' model building and prediction are done in parallel.

In this paper, we focus on the job power consumption per node. Nevertheless, the component can easily be configured to predict power consumption at the CPU, GPU or memory level. Moreover, the energy consumption of a job can be computed as its average power consumption multiplied by its duration. Therefore, UoPC can also be used to estimate the job energy consumption per node as the predicted average power consumption multiplied by the duration predicted by the user.

UoPC can also be used to estimate the power consumption of the whole system at a given time t. The total power consumption of a job can be computed as the predicted job power consumption multiplied by the number of nodes allocated. By summing the total power consumption of all the jobs running concurrently at a given t, we obtain an estimate for the system. We note however that this estimation concerns only the job executions, as the total system power consumption depends also on other factors, such as cooling system or idle nodes power consumption.

D. UoPC Implementation

We provide an `install` script to install all the required dependencies at the time of the first deployment. Then, a `predict` script can be used to analyze job submission data and perform the prediction; the script takes as input either a series of comma-divided named parameters or a file from which to read the feature values. The frequency of the prediction depends on the use case. It can be called periodically (e.g. after a certain number of jobs are submitted) or when needed without any periodicity.

We provide a Docker [37] configuration, to distribute the framework as a container and make it scalable through container orchestration techniques, such as Kubernetes [13].

IV. UoPC Deployment for Fugaku

In this section, we present the deployment of UoPC for Fugaku to experimentally evaluate the prediction algorithm.

A. Fugaku Dataset

To evaluate our prediction algorithm on real HPC jobs, we extract the data of almost 3 million jobs executed on Fugaku between December 2023 and May 2024 through a specialized management software[4] installed on the system.[5] The software features job management operations (e.g. job manager and scheduler), and enables the recording and storage of information on the job executions.

Figure 2 shows the distribution of the job submission in time. The number of submitted jobs is steadily higher than 10k per day, with a mean value of 20k jobs submitted per day. The only exceptions are in the first days of February and April, where scheduled system maintenance caused a system shutdown. In Figure 3, we show the distributions of the number of users, divided by their number of job data in the dataset. Out of more than 700 users, the great majority has less than 500 job traces. This justifies and favors our approach, since it can work well with a small amount of historical data.

Each job entry contains data concerning the job submission (such as *job name*, *submission time*, *requested resources*,

[3]We consider the user dataset to be accessible in a tabular data frame format, e.g. CSV, TSV, Parquet, JSON, etc.

[4]https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article10.html#cap-03

[5]To respect double-blind anonymization, the data repository will be disclosed upon acceptance.

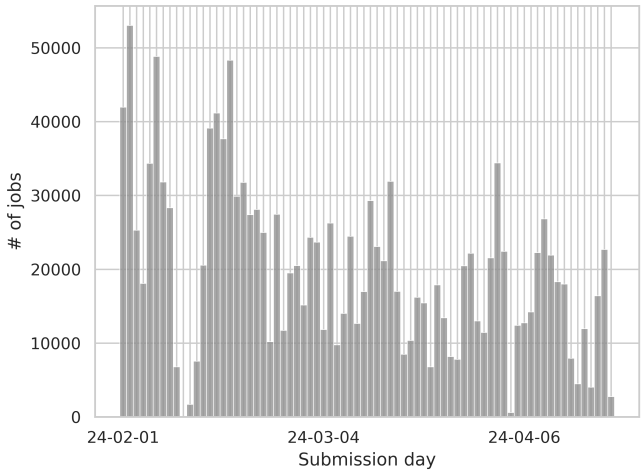

Fig. 2. Distribution of jobs submitted to Fugaku between Dec.'23 and May'24.

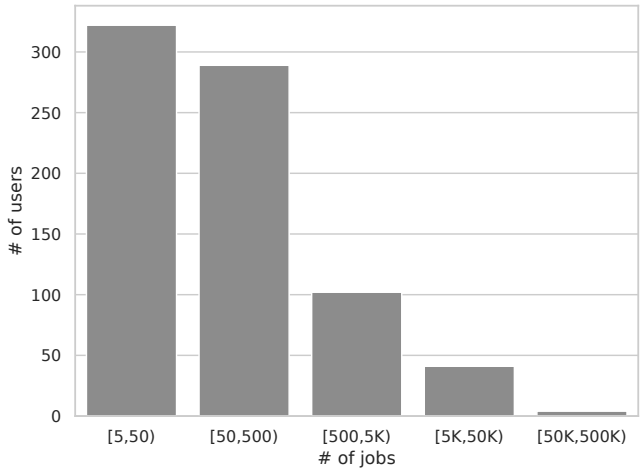

Fig. 3. Distribution of users, divided by the number of jobs in the dataset.

user information, and *system state*), job execution (such as *allocated resources* and *start time*) and outcome (such as *waiting time*, *duration*, *outcome*, and *power consumption*). The job power consumption is recorded at the node level. Since Fugaku's job manager prevents node sharing among jobs, the power consumption value recorded on a node depends only on a single job execution and there is no interference caused by other job executions (besides the typically shared resources like storage and network). The final value of the job power consumption is computed as the sum of the power consumption values recorded on all the nodes allocated to the job during its execution.

B. Data Preparation for Prediction

For our prediction task, we need to associate each job with a set of features representative of its characteristics. In general, the job power consumption depends on the computational operations performed and the amount of hardware used.

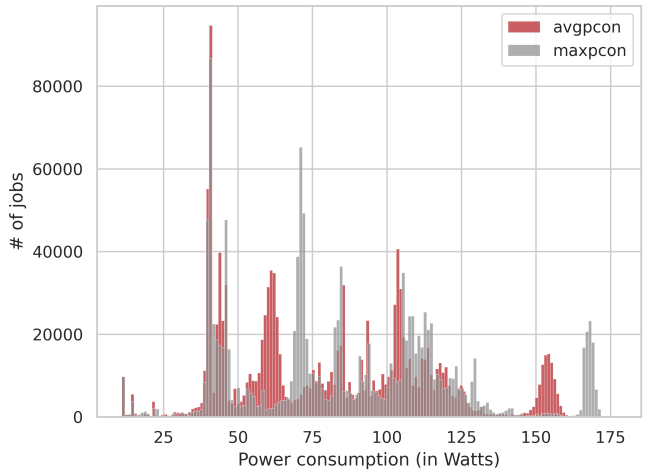

Fig. 4. Distribution of average and maximum power consumption per node of jobs.

Consequently, jobs performing similar operations and having a similar type and amount of hardware allocated, are likely to have similar power consumption. As argued in [5], [34], features like the *job name* and *requested resources*, are key in identifying similar jobs, and consequently performing accurate job power consumption prediction. Since we work with the data of a single user, we exclude the *username* feature which is common to all the job data. The feature set we use to represent the job data is composed of *job name*, *# cores requested*, *# nodes requested*, *requested node frequency*, and *environment*. We chose this set based on past work [5] and upon empirical evaluation of which feature set yields the best prediction.

As for the prediction target, we focus on the average and maximum job power consumption, similarly to [5]. The power consumption of a job is measured as the sum of the average ($avgpcon$) or the maximum ($maxpcon$) power consumption values recorded on each of the nodes allocated to the job during its execution. Such values are normalized on the number of nodes, allowing us to predict the power consumption of each job as if it were running on a single node and makes the prediction task less error-prone. This strategy was adopted in past work [8], [35] and proven useful, especially in the context of workload scheduling [25], [36].

In Figure 4, we show the distribution of the $avgpcon$ and $maxpcon$ per job. The majority of the values lies in the range of 40W-110W, with a maximum value located around 40W. For values greater than 140W, the $avgpcon$ is shifted to the left of $maxpcon$. This is explainable by the fact that for a single job, the $maxpcon$ is always greater or equal to the $avgpcon$. Thus, jobs may reach a high peak of power consumption (maximum), while maintaining a lower average power consumption throughout their execution.

In Section I, we argued that while users tend to submit similar jobs with similar power consumption values, a significant variance may appear across these values. Figures 5 and 6 reveal that this is the case for the Fugaku users and

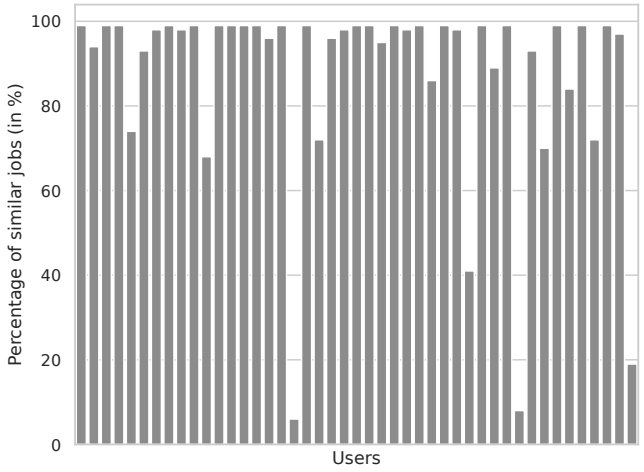

Fig. 5. Percentage of similar jobs for the top 45 users.

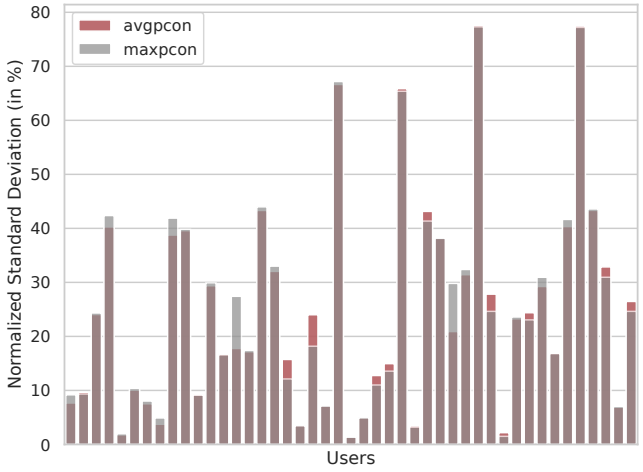

Fig. 6. Normalized tandard deviation (on the average value per user) of the power consumption values for the top 45 users. The two bars of a user are placed one in front of the other.

their jobs. In both figures, we show only the top 45 users (out of more than 700), i.e. the users with the largest amount of job data in the dataset. This is done to ease the readability of the plot, and also because such users submitted $\sim$ 85% of all the jobs executed on the system in the studied period, meaning that they are a relevant sample for analysis. Figure 5 shows the percentage of similar jobs per user. We say that two jobs of a user are similar if they have the same *job name*, *# cores requested*, *# nodes requested*, *requested node frequency*, *environment* (following the features decided earlier to represent similarity), and a difference of less than 5 Watts in the $avgpcon$ and $maxpcon$ values. We observe that the majority of the users submits more than 80% of similar jobs, which supports the idea of user-based prediction.

Figure 6 instead shows the normalized (on the average value per user) standard deviation of the $avgpcon$ and $maxpcon$ values per user. We notice that for some of the users with many similar jobs, the job power consumption values vary a lot (up to more than 70% of the normalized standard deviation). Conversely, some users with few similar jobs have indeed low values of normalized standard deviation. We note that exogenous factors (e.g. room temperature) do not significantly affect power consumption of Fugaku jobs [44]. Hence, the normalized standard deviation shown in Figure 6 depends solely on endogenous factors, such as the resources allocated to the job or the operations it performs.

These observations confirm that the prediction task is non-trivial and that a proper prediction model tailored to a specific user's behavior and job execution characteristics is necessary.

C. Online Prediction Algorithm Implementation

We rely on the *scikit-learn*[6] v1.2.1 Python library for the ML models and use their default implementation (e.g. Minkowski distance for the KNN). The SBert model is provided by the *sentence transformers* library,[7] while the weights are pulled from Huggingface.[8] We use the pre-trained model *all-MiniLM-L6-v2*[9] since it provides the best trade-off between prediction quality and speed [38].

To decide the k and θ parameters of the prediction algorithm, we experimented with different values of θ ($50, 100, 200, 500, 1000, 2000$, and 5000) and k ($5, 10, 20$, and 50). We observed that, for any value of θ, $k = 5$ always yields the best prediction performance. Conversely, the prediction accuracy stops increasing for $\theta > 500$, while the inference time grows up to several seconds. Hence, we set k = 5 and θ = 500. UoPC is implemented and executed with Python 3.12, and the experiments are run on a machine with similar characteristics to one of the supercomputer's nodes (two AMD EPYC 7302 CPUs with 64 cores and 512 GB of RAM).

V. Experimental Study

In this section, we experimentally evaluate the UoPC online prediction algorithm and discuss the results.

A. Online Prediction Algorithm Evaluation

a) Metrics: To evaluate the prediction quality, we compute the Mean Absolute Percentage Error (MAPE), and the R^2 score, between the actual value (prediction target) and the predicted job power consumption value. The MAPE is the mean of all absolute percentage errors on the predictions. Conversely, the R^2 score represents how much of the variation in the target values is predictable from the model. R^2 is not suitable to evaluate the numerical error, but it is meaningful to evaluate the prediction model's quality and adaptability. Generally speaking, an accurate regression model for a job

[6]https://scikit-learn.org/stable/

[7]https://www.sbert.net

[8]https://huggingface.co

[9]https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2

Approach	$avgpcon$				$maxpcon$			
	MAPE (%)	R^2	Avg Train Time (s)	Avg Inf Time (s)	MAPE (%)	R^2	Avg Train Time (s)	Avg Inf Time (s)
User based (U-)								
UoPC (U-KNN)	**10**	**0.80**	**0**	**0.08**	**10**	**0.79**	**0**	**0.08**
U-RF	-	-	-	-	-	-	-	-
All data (A-)								
A-KNN	16	0.60	0	9	17	0.58	0	9
A-RF	14	0.76	2080	0.13	15	0.76	2080	0.13

TABLE I

RESULTS FOR THE $avgpcon$ AND $maxpcon$ PREDICTION TASKS. FOR MAPE LOWER VALUES INDICATE BETTER RESULTS, WHILE FOR R^2 HIGHER VALUES ARE PREFERRED. FOR TRAINING AND INFERENCE TIME, THE LOWER THE BETTER. A "-" MEANS THAT NO RESULT WAS OBTAINED IN A WEEK OF COMPUTATION. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

should obtain a MAPE lower than 20% [6], [46], and an R^2 of at least 0.50 [40].

We also evaluate the prediction overhead to see how the algorithm impacts the system workload submission workflow. We define overhead as the computational time incurred by the prediction algorithm. The overhead of any ML-based prediction algorithm is composed of training and inference time. We define training time as the average time needed to train the model on the past data, and inference time as the time required to perform a prediction for a single job. When training is repeated periodically, it contributes to the overhead. A practical algorithm should incur a small overhead on the system operations, to be seamlessly integrated in the default workload submission workflow. A way to evaluate an algorithm's overhead is to compare its inference time to the average job waiting time (the time spent between the submission and the insertion in the job execution queue). If the first is negligible w.r.t. to the second, the algorithm would not delay the job execution, hence it can be seamlessly integrated in the system workload submission workflow.

b) Baselines: From the related work discussed in Section II-B, we can conclude that RF is the the most used and effective model to predict job execution characteristics. The best results for power consumption prediction have been obtained with the online algorithm presented in [5]. The algorithm involves a daily rebuilding of the model, on the data of the jobs executed in the last 60 days, from all the users. This method is used as a baseline to show how our approach performs with respect to the state-of-the-art. We employ both RF and KNN (the predictive model of our algorithm) in this algorithmic setting and refer to these baselines as A-RF and A-KNN, with "A-" meaning all users' data. To support that our approach would incur significant runtime overhead if the user-based models needed training, we also employ RF as predictive model in our algorithm, which we refer to as U-RF, with "U" meaning user-based. The algorithm of UoPC thus corresponds to U-KNN. Finally, we consider a simple baseline which predicts a job's power consumption by fitting a Linear Regression (LR) model on the power consumption of the last k (i.e. 5) jobs executed by the same user.

c) Testing on historical data: To test on historical data, we use the time information provided by the *submit_time, start_time* and *end_time* features to simulate the actual timeline of job submission and execution on a machine. We use this temporal information also to guarantee that the data used for model building always comes before in chronological order the data of the test set. For a fair comparison, we evaluate both approaches on the same test set. Since the baseline requires 60 days of historical data also during the first training, the test set starts with the jobs submitted as of February $1st$.

For the UoPC algorithm, we iterate over all the job data and for every job j we perform prediction. To this end, we create the user-specific dataset by selecting the jobs belonging to the same user that were completed before the submission of j. This guarantees a realistic set-up where the prediction for a given job cannot be made with the data of future jobs.

Since both approaches require the SBert data encoding, we create them initially for all the jobs executed before February $1st$. Then, when we test a job j, we save its encoding to be retrieved for future training and prediction. This is done not to increase the overhead, as the encoding time for a single job is negligible ($2 * 10^{-3}$ seconds), while it can be significant for a set of training data. A similar time-saving mechanism can be enacted when UoPC is deployed in a real system.

We employ both approaches to predict $avgpcon$ and $maxpcon$ for every job, keeping them as two distinct learning tasks. Finally, we compute the evaluation metrics on all the predictions performed on the test set data.

d) System power consumption: We evaluate the prediction quality also at the system level, as in [5], to understand how accurately the prediction algorithm can reconstruct the system power consumption, and whether it can be an effective tool to predict the whole system power consumption by analyzing only the submitted jobs. As discussed in [5], the prediction error at the job level can be either an overestimation or an underestimation of the actual power consumption. Hence, such errors might cancel out each other at the system level, given the large number of jobs running concurrently.

We consider all the jobs for which we computed the predictions, and we group them based on the hour of the day when they were running. For all the hours of all the days $d \in D$, we compute the system power consumption as the sum of the actual power consumption of all the running jobs. Then, we compute the daily system power consumption $psys_d$, as the average of all the hourly ones. By replacing the actual job power consumption with the generated prediction, multiplied by the number of nodes allocated to the jobs, we can also compute the predicted system power consumption $\overline{p}sys_d$.

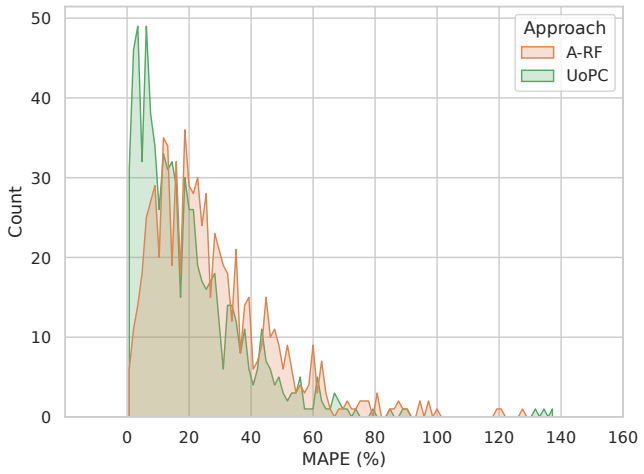

Fig. 7. Distribution of the MAPE values per user for the *avgpcon* prediction.

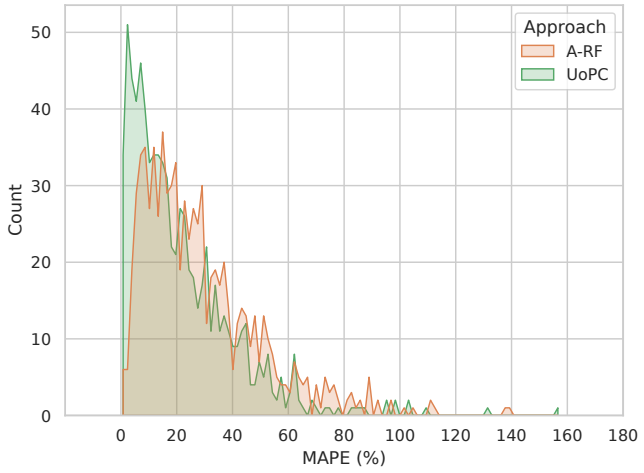

Fig. 8. Distribution of the MAPE values per user for the *maxpcon* prediction.

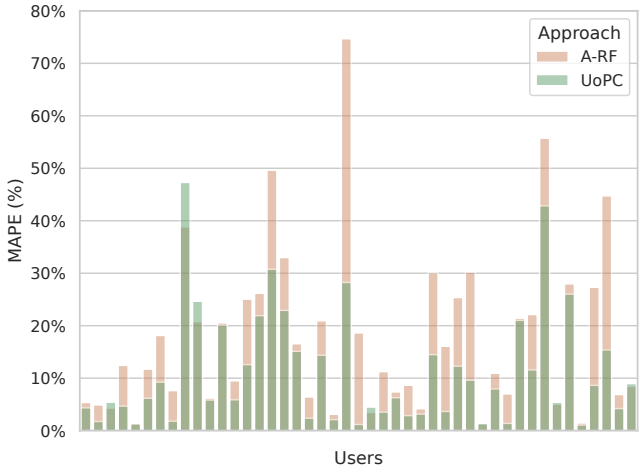

Fig. 9. MAPE per user of the *avgpcon* prediction for the top 45 users. The bars are placed one in front of the other.

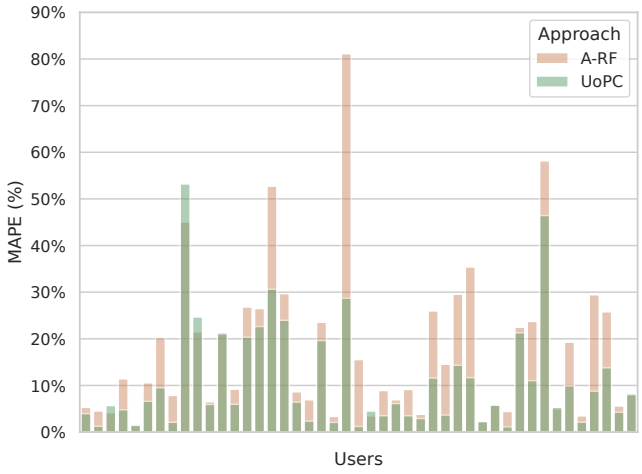

Fig. 10. MAPE per user of the *maxpcon* prediction for the top 45 users. The bars are placed one in front of the other.

Differently from [5], we evaluate the numerical error of the predictions by computing the $mean\ error$ score in Equation 1, as defined in [8]. In addition, we also calculate the R^2 score.

$$mean\ error = \frac{1}{|D|} \sum_{d \in D} \frac{|psys_d - \overline{p}sys_d|}{psys_d} * 100 \qquad (1)$$

B. Experimental Results

a) Job power consumption prediction: In Table I, we show the prediction performance of UoPC (U-KNN) and the baselines defined in Section V-A. We exclude LR from the table as it obtains a very high MAPE ($> 100\%$). As expected, U-RF turned out to be computationally expensive and did not terminate within a week, even when experimenting with less DTs (50 and 75) than the default (100). We conclude that the method is not suited to a production environment. As A-RF outperforms A-KNN both in prediction performance and overhead, from now on we consider A-RF as the only baseline. While both UoPC and A-RF lead to accurate power consumption prediction, our algorithm outperforms the baseline for both the $avgpcon$ and $maxpcon$ prediction tasks. In the $avgpcon$ task, our algorithm improves the MAPE score of the A-RF by 40%, going from 14% to 10%. It also improves the R^2, from 0.76 to 0.80. For the $maxpcon$, the MAPE of our approach is stable at 10%, while it slightly increases to 15% for the A-RF baseline. The R^2 of our approach is 0.01 lower in the $avgpcon$, however, it is steadily higher than the baseline value of 0.76. These results show that our approach not only obtains a very low prediction error in general terms but is also more accurate than A-RF.

In Figures 7-10, we compare UoPC against A-RF across single users. Figures 7 and 8 show the distribution of the MAPE values of the approaches on all the users. We observe that the majority of the MAPE values obtained by UoPC is less than 10%, while A-RF obtains generally higher MAPE values. This means that overall, A-RF obtains a higher error w.r.t.

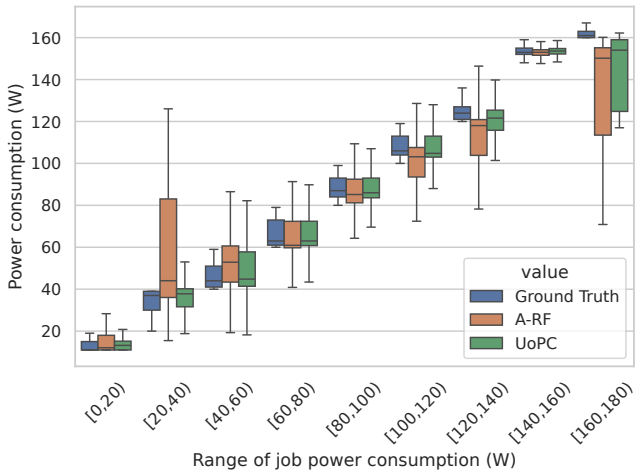

Fig. 11. Distribution of actual and predicted $avgpcon$ in ranges of 20W.

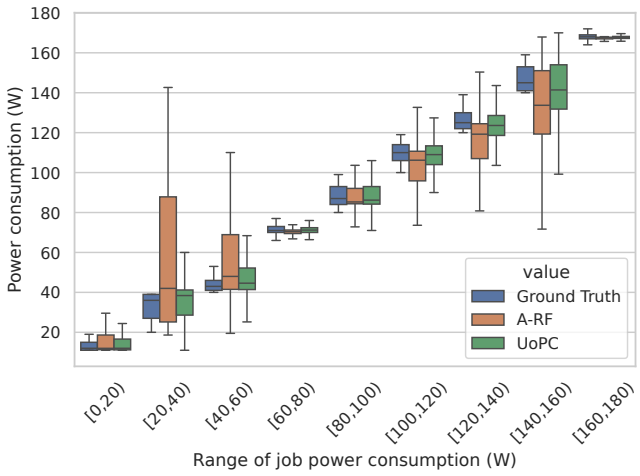

Fig. 12. Distribution of actual and predicted $maxpcon$ in ranges of 20W.

UoPC. This is highlighted also in Figures 9 and 10, where we show the MAPE obtained by the two approaches on the top 45 users (as in Figure 6). Here we notice that except for a couple of users (where the A-RF error is anyway similar), UoPC obtains a significantly lower error on the single users.

The high R^2 score obtained by our approach outlines its capability to accurately predict the variability in the power values. To observe this, we split the jobs based on their actual power consumption into power ranges of 20W. We then plot in Figures 11 and 12 the distribution of the ground truth and the predicted values in each range using both approaches. We observe that the UoPC distributions are more aligned with the ground truth w.r.t. to the A-RF ones. This is particularly noticeable in the ranges [100, 140) and [20, 40), which is also the most populated range, as shown in Figure 4. Both A-RF and UoPC struggle in the ranges [160, 180) for the $avgpcon$ task, and [140,160) for the $maxpcon$ prediction tasks. This can be explained by the low density of data in such areas, as shown in Figure 4. Nevertheless, UoPC still approximates the distribution better than A-RF.

b) Amount of data required: In Figure 13, we show the average amount of job execution data per month used by the prediction algorithms. The amount used by the UoPC algorithm is computed by summing the dimensions of all the user datasets used during prediction. We observe that the amount needed by the A-RF is always over 1.1 million job traces, while UoPC requires around 800k data points. Even with fewer data, UoPC outperforms A-RF.

c) Overhead: In Table I, we also compare the overhead of the two approaches. Since the A-RF baseline requires daily retraining, we report the average daily training time. Our approach does not require a training phase, and the only overhead is the inference time. We observe that the training overhead of the A-RF is non-negligible, as it is more than half an hour per day. Thus, our approach saves time and resources, while resulting in a more accurate prediction.

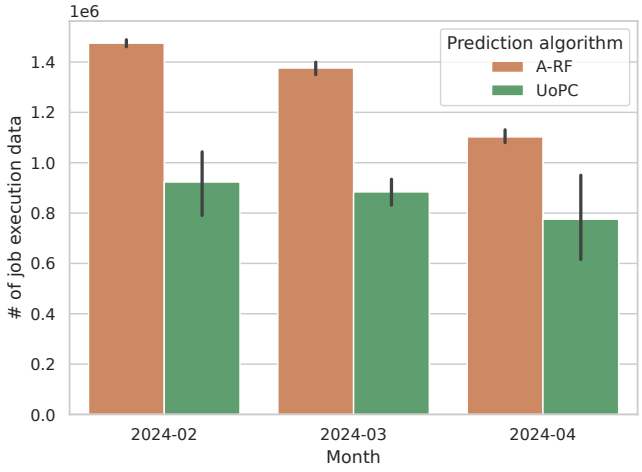

Fig. 13. Average amount of data per month used by the prediction models (aggregated value and confidence intervals)

The inference time of the two approaches is comparable and negligible compared to the average job waiting time in the system (up to 3 minutes). However, our algorithm still manages to predict in almost half of the A-RF time, 0.08 seconds against 0.13. Considering an average of 20k job submissions per day, if both approaches were to be deployed in a real system to analyze all the jobs systematically, the average daily inference time would be around 45 minutes for the A-RF and around 25 for UoPC. We note that these numbers reflect the worst-case scenario where 20k jobs arrive in the system simultaneously and need to be processed right away.

d) System power consumption prediction: Figures 14 and 15 compare the estimated system power consumption values with UoPC and A-RF to the actual values for both the $avgpcon$ and $maxpcon$ tasks. The drops in values are due to the system shutdown, as can be seen also in Figure 2. We observe that UoPC outperforms A-RF both in terms of $mean\ error$ and R^2

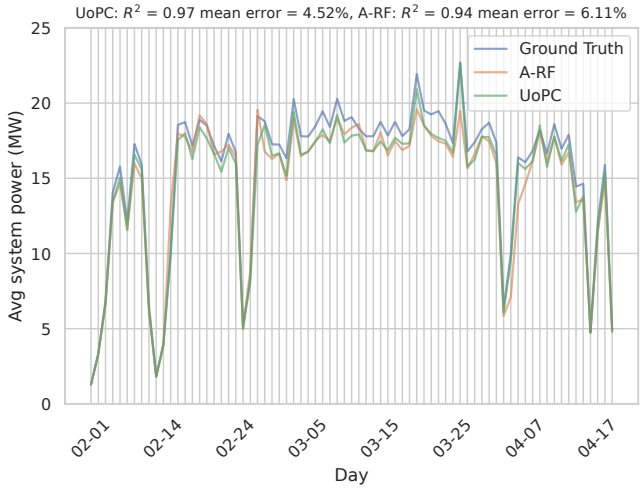

Fig. 14. System power consumption prediction for the $avgpcon$ task.

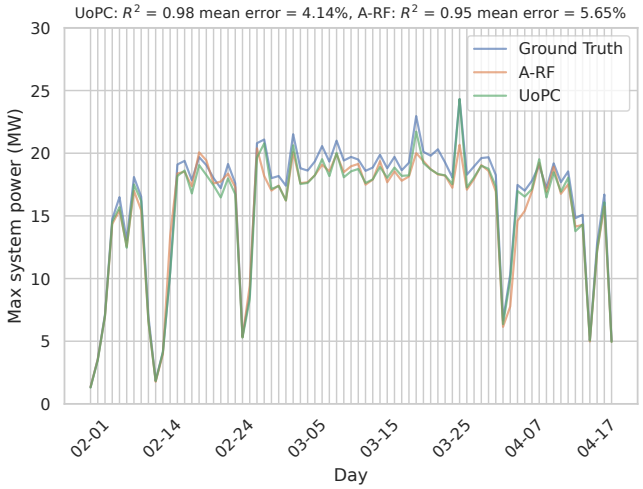

Fig. 15. System power consumption prediction for the $maxpcon$ task.

in both tasks. It obtains an R^2 greater than 0.97 and an error smaller than 5% for both tasks, meaning that our approach is accurate enough to be used to predict the system-level power consumption. We note that UoPC's purpose is not to predict system-level power consumption; however, given its accuracy, such a prediction can be instrumental in guiding power-aware scheduling strategies in power-constrained systems.

C. Discussions

Our experimental results show that, with respect to the state-of-the-art ML methods using all users' data, UoPC obtains better predictions (lower MAPE), while incurring a smaller overhead and requiring less amount of data. An important consideration is that even if the improvement in the MAPE value is low, it can have a big impact when the prediction is used, for instance, to estimate the energy consumption of a job. In the Fugaku data, the average job duration is 10k seconds, while the average job power consumption is 5576W. If we consider job energy consumption as job power consumption multiplied by its duration, a prediction more accurate by just 1% would lead to a more precise job energy consumption estimation of around 156Wh. Considering this at the system scale, where on average 20k jobs are submitted per day, the improvement reaches around 3,000kWh. To put this number in perspective, the average US household energy consumption is 30kWh per day [45]. Hence, an improvement of 4% and 5%, respectively, in job power consumption prediction tasks, can be a highly significant achievement. For instance, when using a more accurate model in a scheduling algorithm, the scheduling decisions would allow for more energy savings and better performance for each job execution.

We validated our approach on another dataset (PM100) [4] extracted from Marconi100, a heterogeneous system with GPUs mounted on the nodes. The job power consumption data includes the GPU power contribution, and we tested UoPC on predicting the total power consumption, not just the CPU's power usage. Such values span from a few watts to more than 1kW, making the prediction task significantly more challenging. A summary of the obtained results is shown in Table II, similarly to those presented in Table I. Again, our algorithm improves A-RF (which is again better than A-KNN). In the $avgpcon$ task, UoPC obtains a MAPE of 18% against a MAPE of 20% in A-RF, while not requiring model training and just using a fraction of the data. UoPC also obtains a better R^2 score (0.11 against 0.07 of A-RF). For the $maxpcon$, while the MAPE of our approach increases to 22%, it is still lower than 24% of A-RF, with the R^2 value being 0.13 in both cases. These results confirm that UoPC is a valid approach for other systems as well, including those with GPUs. UoPC does not require any structural changes to be adapted to other systems; only a preprocessing step is needed to account for potential differences in the job feature sets.

One limitation of our approach is that a prediction cannot be performed until the kth job execution of a user. However, k can be set to lower values to minimize this inconvenience. Alternatively, UoPC can be coupled with a simple linear regression model or with A-RF (when it is possible to obtain all the user's data), until the user executes the kth job. Another limitation could be that our approach has been tested on jobs executed in exclusive mode. However, as described in [4], no current method can accurately determine jobs' node power consumption when jobs run concurrently, hence no public dataset offers such data. If future monitoring of resources enables precise job attribution, UoPC could be easily modified for this goal.

Finally, we note that UoPC can be seamlessly modified to predict other job features, such as *job failure, duration*, and *performance metrics*. These predictions can be used in conjunction with power consumption prediction to make informed decisions on job scheduling.

VI. Conclusions and Future Work

We presented UoPC, a user-based online framework to predict job power consumption in HPC systems. UoPC offers a

Approach	$avgpcon$				$maxpcon$			
	MAPE (%)	R^2	Avg Train Time (s)	Avg Inf Time (s)	MAPE (%)	R^2	Avg Train Time (s)	Avg Inf Time (s)
User based (U-)								
UoPC (U-KNN)	**18**	**0.11**	**0**	**0.07**	**22**	**0.13**	**0**	**0.07**
U-RF	-	-	-	-	-	-	-	-
All data (A-)								
A-KNN	22	-0.1	0	0.18	27	-0-03	0	0.18
A-RF	20	0.07	400	0.12	24	**0.13**	400	0.12

TABLE II

RESULTS FOR THE $avgpcon$ AND $maxpcon$ PREDICTION TASKS ON PM100 DATA. FOR MAPE LOWER VALUES INDICATE BETTER RESULTS, WHILE FOR R^2 HIGHER VALUES ARE PREFERRED. FOR TRAINING AND INFERENCE TIME, THE LOWER THE BETTER. A "-" MEANS THAT NO RESULT WAS OBTAINED IN A WEEK OF COMPUTATION. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

practical solution to address the limitations of alternative, ML-based predictive approaches, which render them impractical in production systems. These limitations are i) the need for voluminous historical data from diverse users to train the models, and ii) the lack of support for usage by end-users. Our approach is based on the KNN, which is a non-parametric supervised ML algorithm. The prediction is performed before the execution of a job, and it requires only the job features available at submission time and the historical job execution data, which can be as low as 5 as a quantity. We provide an easy-to-use Python implementation, which allows UoPC to be used either by end-users as a stand-alone tool to evaluate the power requirements of their job executions, or integrated into the workload management system to enable power/energy-aware job scheduling strategies. As the implementation requirements transcend architectural boundaries, UoPC can be easily adapted to other environments (e.g., cloud, edge, fog).

We deployed UoPC for the Supercomputer Fugaku and tested the online prediction algorithm on the execution data of more than 1.3 million jobs submitted by more than 700 different users. The experimental results demonstrate that our solution consistently outperforms the best representative of the existing approaches in terms of prediction accuracy and overhead on the system operations. We obtain an error of 10% when working at the level of single jobs, and 4% when working at system level. This means that UoPC is suitable for usage by end-users and for enabling the adoption of energy-efficient solutions for cluster systems. Even though we did not explicitly address the privacy concerns, we highlight another positive aspect of our contribution: our approach can make predictions for a specific user by using *exclusively that user's data*, hence it naturally preserves the privacy of each user. This is an important feature where data is very sensitive and cannot be shared among different users.

In future work, we will look into obtaining more fine-grained job data (e.g., the MPI-version, the loaded libraries, the values of the environment variables) to improve the prediction performance. We also plan to explore the integration of UoPC into a workload submission pipeline, aiming to leverage the predictions to optimize the energy efficiency of HPC systems.

ACKNOWLEDGMENTS

The research has been partially funded by the HORIZON-RIA DECICE project (g.a. 101092582), and the HE EU Graph-Massivizer project (g.a. 101093202).

REFERENCES

[1] M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis, and E. Agiatzidou, "Energy-aware cost prediction and pricing of virtual machines in cloud computing environments," *Future Generation Computer Systems*, vol. 93, pp. 442–459, 2019.

[2] F. Antici, A. Bartolini, Z. Kiziltan, O. Babaoglu, and Y. Kodama, "Mcbound: An online framework to characterize and classify memory/compute-bound hpc jobs," in *To appear in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024.

[3] F. Antici, A. Borghesi, and Z. Kiziltan, "Online job failure prediction in an hpc system," in *Euro-Par 2023: Parallel Processing Workshops: Euro-Par 2023 International Workshops, Limassol, Cyprus, August 28–September 1, 2023, Revised Selected Papers*. Springer Nature, 2023.

[4] F. Antici, M. Seyedkazemi Ardebili, A. Bartolini, and Z. Kiziltan, "Pm100: A job power consumption dataset of a large-scale production hpc system," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 1812–1819.

[5] F. Antici, K. Yamamoto, J. Domke, and Z. Kiziltan, "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*, 2023, pp. 1820–1830.

[6] K. Assogba, E. Lima, M. M. Rafique, and M. Kwon, "Predictddl: Reusable workload performance prediction for distributed deep learning," in *2023 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2023, pp. 13–24.

[7] L. Baier, F. Jöhren, and S. Seebacher, "Challenges in the deployment and operation of machine learning in practice." in *ECIS*, vol. 1, 2019.

[8] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "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, 2016, pp. 181–199.

[9] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Scheduling-based power capping in high performance computing systems," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 1–13, 2018.

[10] A. Borghesi, A. Bartolini, M. Milano, and L. Benini, "Pricing schemes for energy-efficient hpc systems: Design and exploration," *The International Journal of High Performance Computing Applications*, vol. 33, no. 4, pp. 716–734, 2019.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[12] B. Bugbee, C. Phillips, H. Egan, R. Elmore, K. Gruchalla, and A. Purkayastha, "Prediction and characterization of application power use in a high-performance computing environment," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 10, no. 3, pp. 155–165, 2017.

[13] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.

[14] 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.

[15] H. Choi, J. Kim, S. Joe, and Y. Gwon, "Evaluation of bert and albert sentence embedding performance on downstream nlp tasks," in *2020 25th International conference on pattern recognition (ICPR)*. IEEE, 2021, pp. 5482–5487.

[16] Y. Chu, H. Cao, Y. Diao, and H. Lin, "Refined sbert: Representing sentence bert in manifold space," *Neurocomputing*, vol. 555, p. 126453, 2023.

[17] T. Ciesielczyk, A. Cabrera, A. Oleksiak, W. Piątek, G. Waligóra, F. Almeida, and V. Blanco, "An approach to reduce energy consumption and performance losses on heterogeneous servers using power capping," *Journal of Scheduling*, vol. 24, pp. 489–505, 2021.

[18] M. Colmant, P. Felber, R. Rouvoy, and L. Seinturier, "Wattskit: Software-defined power monitoring of distributed systems," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2017, pp. 514–523.

[19] J. Devlin, M.-W. Chang, K. Lee, and et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[20] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[21] P. Hamandawana, R. Mativenga, S. J. Kwon, and T.-S. Chung, "Towards an energy efficient computing with coordinated performance-aware scheduling in large scale data clusters," *IEEE Access*, vol. 7, pp. 140 261–140 277, 2019.

[22] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, pp. 751–774, 2016.

[23] M. S. Hasan, F. A. de Oliveira, T. Ledoux, and J.-L. Pazat, "Enabling green energy awareness in interactive cloud application," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016, pp. 414–422.

[24] G. Juve, B. Tovar, R. F. Da Silva, D. Król, D. Thain, E. Deelman, W. Allcock, and M. Livny, "Practical resource monitoring for robust high throughput computing," in *2015 IEEE International Conference on Cluster Computing*. IEEE, 2015, pp. 650–657.

[25] H. Khaleghzadeh, R. Reddy Manumachu, and A. Lastovetsky, "Efficient exact algorithms for continuous bi-objective performance-energy optimization of applications with linear energy and monotonically increasing performance profiles on heterogeneous high performance computing platforms," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 20, p. e7285, 2023.

[26] M. I. K. Khalil, I. Ahmad, S. A. A. Shah, S. Jan, and F. Q. Khan, "Energy cost minimization for sustainable cloud computing using option pricing," *Sustainable Cities and Society*, vol. 63, p. 102440, 2020.

[27] 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.

[28] V. Kulkarni, M. Gawali, A. Kharat *et al.*, "Key technology considerations in developing and deploying machine learning models in clinical radiology practice," *JMIR Medical Informatics*, vol. 9, no. 9, p. e28776, 2021.

[29] B. Li, R. Basu Roy, D. Wang, S. Samsi, V. Gadepally, and D. Tiwari, "Toward sustainable hpc: Carbon footprint estimation and environmental implications of hpc systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–15.

[30] J. Li, G. Ali, N. Nguyen, J. Hass, A. Sill, T. Dang, and Y. Chen, "Monster: an out-of-the-box monitoring tool for high performance computing systems," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2020, pp. 119–129.

[31] S. Loganathan, R. D. Saravanan, and S. Mukherjee, "Energy aware resource management and job scheduling in cloud datacenter." *International Journal of Intelligent Engineering & Systems*, vol. 10, no. 4, 2017.

[32] J. Muraña, S. Nesmachnow, F. Armenta, and A. Tchernykh, "Characterization, modeling and scheduling of power consumption of scientific computing applications in multicores," *Cluster Computing*, vol. 22, pp. 839–859, 2019.

[33] M. Nakao, H. Kaneyama, M. Nagaku, I. Fujiwara, A. Takefusa, S. Miura, and K. Yamamoto, "Introducing open ondemand to supercomputer fugaku," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 720–727.

[34] A. Okanlawon, H. Yang, A. Bose, W. Hsu, D. Andresen, and M. Tanash, "Feature selection for learning to predict outcomes of compute cluster jobs with application to decision support," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2020, pp. 1231–1236.

[35] T. Patel, A. Wagenhäuser, C. Eibel, T. Hönig, T. Zeiser, and D. Tiwari, "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, 2020, pp. 799–809.

[36] B. Qureshi, "Profile-based power-aware workflow scheduling framework for energy-efficient data centers," *Future Generation Computer Systems*, vol. 94, pp. 453–467, 2019.

[37] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to docker and analysis of its performance," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

[38] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[39] T. Saillant, J.-C. Weill, and M. Mougeot, "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, 2020, pp. 63–82.

[40] A. Sîrbu and O. Babaoglu, "Power consumption modeling and prediction in a hybrid cpu-gpu-mic supercomputer," in *Euro-Par 2016: Parallel Processing: 22nd International Conference on Parallel and Distributed Computing, Grenoble, France, August 24-26, 2016, Proceedings 22*. Springer, 2016, pp. 117–130.

[41] N. Sukhija and E. Bautista, "Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*. IEEE, 2019, pp. 257–262.

[42] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Computing*, vol. 21, no. 3, pp. 1581–1593, 2018.

[43] X. Tian, X. Li, J. Zhang, Z. Zhao, C. Wang, X. Wang, and J. Wang, "An online incremental learning framework for hpc job power consumption prediction," in *Proceedings of the 2023 7th International Conference on High Performance Compilation, Computing and Communications*, 2023, pp. 176–183.

[44] Y. Tsujita, A. Uno, R. Sekizawa, K. Yamamoto, and F. Sueyasu, "Job classification through long-term log analysis towards power-aware hpc system operation," in *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2021, pp. 26–34.

[45] U.S. Energy Information Administration, "How much electricity does an american home use?" 2024. [Online]. Available: https://www.eia.gov/tools/faqs/faq.php?id=97&t=3

[46] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for {Large-Scale} advanced analytics," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 363–378.

[47] Z. Xu, Z. Guo, and N. Cristianini, "On compositionality in data embedding," in *International Symposium on Intelligent Data Analysis*. Springer, 2023, pp. 484–496.

[48] K. Yamamoto, Y. Tsujita, and A. Uno, "Classifying jobs and predicting applications in hpc systems," in *High Performance Computing: 33rd International Conference, ISC High Performance 2018, Frankfurt, Germany, June 24-28, 2018, Proceedings 33*. Springer, 2018, pp. 81–99.