

Trustworthy AI for Energy-Efficient HPC Under Prediction Errors and Uncertainty

Micaella Coelho[1], Alexandre H. L. Porto[1], Isabella Muniz[1], Kary Ocaña[1], Arthur Lorenzon[2], Philippe O. A. Navaux[2], Carla Osthoff[1], Hiago Mayk G. de A. Rocha[1(✉)], and Douglas O. Cardoso[3]

[1] National Laboratory for Scientific Computing (LNCC), Petrópolis, RJ, Brazil
mayk@lncc.br
[2] Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil
[3] CeBER, Faculty of Economics, Univ. Coimbra, Av. Dias da Silva 165, 3004-512 Coimbra, Portugal

Abstract. As High-Performance Computing (HPC) systems become faster and more complex, their energy consumption continues to increase, raising both economic and environmental concerns. Therefore, it is essential to manage energy usage efficiently, balancing energy consumption with application performance. In this work, we extend our prior research on predictive modeling for energy-efficient resource allocation in HPC systems, which uses the Energy-Delay Product (EDP) as a measure of energy efficiency. Furthermore, instead of focusing solely on predictive accuracy, we aim to provide a deeper understanding of model behavior and its impact on configuration decisions. While previous results demonstrated that an Extra Trees Regressor (ETR) can approximate Oracle configurations with high accuracy, little is known about when and why prediction errors occur. In this work, we present an analysis of prediction errors, an initial uncertainty-aware perspective, and model interpretability in EDP prediction. Our results show that although exact-match accuracy is 81.25%, over 90% of predictions fall within a small regret margin, indicating robust practical performance. Furthermore, uncertainty-aware filtering reduces high-error recommendations, improving reliability for real-world deployment. These findings position the proposed model as not only an accurate but also trustworthy tool for sustainable HPC resource allocation. This work contributes toward the development of explainable and reliable AI systems for energy-efficient computing, aligning with emerging goals in sustainable digital infrastructures.

Keywords: High-Performance Computing · Machine Learning · Resource Allocation · Model Interpretability · Uncertainty Quantification

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16762, pp. 260–276, 2027.
https://doi.org/10.1007/978-3-032-30524-4_17

1 Introduction

Energy consumption in High-Performance Computing (HPC) systems has become one of the main challenges in the field of parallel computing, driven by the increasing demand for computational capacity and the execution of large-scale scientific applications. In this context, it has become increasingly important to develop strategies that balance performance and energy efficiency in HPC environments [11]. This challenge is not only associated with low-level aspects, such as operating frequency and memory behavior, but also with high-level decisions made at application submission time, such as the choice of the number of computational nodes (#N) and the number of threads per node (#T).

The choice of these execution parameters can substantially affect both execution time and energy consumption, directly influencing the overall efficiency of the application in terms of both response time to obtain results and energy consumption during execution. A widely adopted way to capture this trade-off is through the Energy-Delay Product (EDP), a metric that combines time and energy into a single optimization criterion. From this perspective, the optimal configuration is not necessarily the one that minimizes execution time or energy consumption, but rather the one that achieves the best trade-off between these two metrics.

In recent years, Machine Learning (ML)-based techniques have gained prominence as an alternative for supporting efficient resource allocation in HPC, since they can capture complex relationships between input parameters, application characteristics, and performance metrics [4,5,14]. In particular, predictive models can be employed to estimate the EDP for different execution configurations and thus recommend promising allocations without the need for exhaustive searches across multiple executions. Despite this potential, many approaches still focus their evaluation only on predictive accuracy, or on a single performance dimension, such as execution time [15], while others depend on complex instrumentation or additional monitoring mechanisms [12].

In previous work, we proposed a methodology based on Extra Trees Regressor (ETR) to indicate the (#N,#T) configuration with the best EDP for the execution of parallel applications in HPC systems [5]. Using the RAxML [21] application as a case study, we showed that the model can approximate the Oracle configuration with good accuracy, defined as the configuration that yields the lowest real EDP for each input combination. However, although these results highlight the potential of the approach, they do not explain the model behavior in depth. In particular, a recommendation that differs from the optimal configuration does not necessarily constitute a severe failure from a practical standpoint, and the observed deviations may result from different trade-offs between execution time and energy consumption.

Motivated by this limitation, this work extends our previous analysis by shifting the focus from an evaluation centered only on accuracy to a more in-depth investigation of the predictability and interpretability of efficient resource allocation in HPC. Rather than introducing a new predictive method, our contribution lies in analyzing how an already validated ETR-based strategy behaves under

non-exact recommendations. More specifically, we analyze: *(i)* the structure of the experimental data from the RAxML application considering HPC system, Santos Dumont Supercomputer (SDumont)[1]; *(ii)* the variability across repeated executions; *(iii)* the impact of aggregating these executions on the definition of representative configurations, and, finally, (*iv*) the nature of prediction errors from complementary EDP, time, and energy perspectives. In this way, we seek to understand not only whether the model matches the Oracle configuration, but also how and why its deviations occur.

For this analysis, we again use the RAxML application [21], a representative bioinformatics application in the context of HPC, widely employed in large-scale phylogenetic inference. The application combines distributed-memory parallelism via the Message Passing Interface (MPI) with intra-node parallelism via *PThreads*, making it suitable for studying the impact of different resource allocation choices on execution time, energy consumption, and EDP.

In summary, the main contributions of this work are:

- an exploratory analysis of RAxML execution data, including aspects of variability, aggregation, and the interpretation of competitive configurations on the HPC system;
- a deeper analysis of prediction errors in EDP-oriented resource allocation, showing how deviations from the Oracle can be interpreted in terms of execution time and energy consumption.

The results show that, although the exact accuracy of the methodology remains high, at around 81.25% correct, the proposed ETR-based strategy remains consistently close to the Oracle across EDP, execution time, and energy consumption. Furthermore, the analyses reveal that the compared strategies, MaxRes and MinRes, never match the Oracle in the evaluated scenarios, and that their deviations correspond to distinct error profiles: excessive energy overhead in MaxRes and excessive runtime in MinRes. These findings indicate that the practical value of the proposed methodology lies not only in its exact-match accuracy but also in its ability to avoid the extreme inefficiencies observed in fixed-resource allocation strategies.

The remainder of this article is organized as follows. Section 2 reviews related work. Section 3 presents the theoretical background. Section 4 briefly describes the proposed methodology. Section 5 provides an exploratory study of the RAxML execution dataset. Section 6 reports and analyzes the prediction errors in our proposal. Section 7 concludes the paper and highlights future research directions.

2 Related Work

Several studies have investigated the use of ML techniques to support energy-aware decisions in HPC systems [1,12,14,19]. These studies differ in terms of

[1] http://sdumont.lncc.br.

optimization objective (e.g., execution time, energy, power, or EDP), required input information (e.g., application's inputs, number of nodes, and requested execution time), and the type of execution parameter being recommended (e.g., CPU or GPU frequencies, number of threads, or thread and data mappings). In this section, we focus on the works that are most closely related to our proposal, particularly those that employ predictive models to improve energy efficiency or to support resource allocation decisions.

Lorenzon et al. [14] employed a Random Forest model at runtime to select GPU frequency and execution node, achieving EDP improvements in the Frontier Supercomputer exascale system. Although effective, their approach is restricted to GPU-based environments and relies on Dynamic Voltage and Frequency Scaling (DVFS), whereas our work focuses on CPU-based systems and investigates resource allocation through the number of computational nodes and threads. Kunas et al. [12] proposed NeurOPar, a neural-network-based strategy that recommends the number of threads and CPU frequency to optimize EDP. Despite its favorable results, that approach depends on detailed profiling and standardized executions, increasing its deployment complexity.

Carastan-Santos et al. [4] also explored predictive support for energy-aware management in HPC environments. Their work focuses on lightweight prediction of power-related behavior to support scheduling decisions under power constraints. Similarly, Springborg et al. [2] investigate predictive support for energy-aware resource management at job submission time. In contrast, our approach is not designed as a scheduler-integrated mechanism but rather as an application-oriented predictive strategy that estimates EDP from historical execution data to recommend efficient resource configurations.

Scravaglier et al. [20] employ decision-tree models to identify efficient execution configurations involving NUMA placement and hardware prefetching. Their approach considers parameters such as thread count, thread and data affinity, number of NUMA nodes, and prefetcher configurations, while also accounting for different problem sizes. However, it relies on synthetic benchmarks and requires extensive profiling to gather detailed execution traces. In contrast, the approach proposed here is designed to be simpler and more practical: predictions are made early, using only information available at job submission time, without additional instrumentation. Furthermore, it is grounded in real execution data collected via RAPL and focuses on predicting EDP as the key metric for guiding configuration decisions.

The study by Imes et al. [9] applies classification models to identify efficient combinations of task sets and CPU frequencies, reporting an average energy reduction of 19.3% in bioinformatics workloads. These results demonstrate the effectiveness of machine learning in supporting energy-aware configuration. However, their approach does not employ a composite metric such as EDP as the primary optimization objective, nor does it rely on power measurements obtained via RAPL. In contrast, the present work leverages RAPL-based measurements and adopts EDP as the central metric to drive configuration selection.

Another relevant direction is the use of submission-time information to anticipate the energy behavior of HPC jobs. Patel et al. [18], for example, showed that a small set of submission attributes – user, number of nodes, and requested runtime – is sufficient to accurately predict per-node power consumption in academic clusters. This result reinforces the feasibility of early prediction based on job-level information, an idea that is also central to our work. However, their study focuses on power prediction rather than on recommending a resource allocation configuration that minimizes a composite metric such as EDP.

The work of Coelho et al. [5] (our previous work) introduced an ETR-based methodology to recommend (#N,#T) configurations that minimize EDP for parallel applications in HPC systems, without requiring application execution or code instrumentation during the recommendation phase. Using RAxML as a case study, the work showed that the methodology could accurately identify configurations close to the Oracle. The present work builds directly on that initi al proposal.

Our main contribution in this paper consists of, instead of introducing a new predictive methodology, we extend the evaluation beyond exact-match accuracy and investigate the behavior of the model from a broader perspective, including exploratory analysis of the execution data, variability across repeated runs, and the interpretation of prediction errors in terms of time and energy.

3 Theoretical Background

This section presents the main concepts that support the analyses developed in this work. In particular, we briefly discuss the ETR, the EDP, the coefficient of variation, and the Pareto frontier.

The ETR is an *ensemble* learning algorithm for regression that combines multiple decision trees built with a high degree of randomization [6]. Its main characteristic is that randomness is introduced both in the choice of candidate attributes and in the selection of split points during tree construction. This design increases diversity among the trees and helps reduce the variance of the final model. For regression tasks, the predictions of the individual trees are combined by averaging, producing a model with good generalization ability and reduced sensitivity to overfitting.

ETR is particularly suitable for tabular datasets, especially in scenarios where nonlinear relationships may exist between the input variables and the target metric [8]. In the present study, the execution data are organized exactly in this form, with each observation describing an application input and a resource configuration. For this reason, ETR is adopted as the predictive model used to estimate the EDP of candidate configurations and to support the recommendation of efficient resource allocations.

The EDP is a metric that expresses, in a single value, the trade-off between execution time and energy consumption [17]. It is defined as the product of these two variables, that is,

$$EDP(app) = Energy(app) \times Time(app), \tag{1}$$

where $Energy(app)$ and $Time(app)$ correspond to the energy consumption and execution time measured during the execution of application *app*. Because it jointly captures performance and energy cost, EDP is widely used in studies concerned with energy-efficient computing [7,14,17].

In this work, EDP plays a central role in two senses. First, it is the target variable predicted by the model. Second, it is the criterion used to define the Oracle configuration, that is, the configuration with the lowest real EDP for a given input. The analyses presented later in the paper are therefore organized around EDP, while also examining how deviations from the Oracle can be decomposed into time and energy components.

The Coefficient of Variation (CV) [3] is a normalized measure of dispersion, commonly defined as the ratio between the standard deviation and the mean of a set of observations. Unlike absolute measures of spread, the CV expresses variability relative to the scale of the data, which makes it useful for comparing dispersion across different groups or scenarios. Lower CV values indicate more stable observations, whereas higher values indicate greater relative variability.

In the context of this work, the CV is used to analyze the repeated executions of the same configuration. Since each configuration was executed multiple times, the CV helps quantify how much variability exists before aggregation. This is particularly important because the study relies on representative values derived from repeated runs, and the CV provides evidence about how much information may be lost when these repetitions are summarized into a single value, which could represent the whole poorly.

The Pareto frontier [13] is a fundamental concept in multi-objective optimization. A solution is said to be Pareto-optimal if there is no other solution that improves one objective without worsening at least one other objective. In other words, the Pareto frontier is formed by the set of non-dominated solutions, representing the best trade-offs available among the objectives under consideration.

In this study, the Pareto frontier is used to analyze the joint behavior of execution time and energy consumption. Although the Oracle is defined according to the minimum EDP, the Pareto frontier provides an additional perspective on the configuration space by identifying which solutions remain competitive in the time–energy trade-off. This perspective is useful for understanding not only the structure of the experimental data, but also the cases in which a predicted configuration differs from the Oracle while still remaining close to the set of competitive solutions.

4 Methodology

This work builds directly on the initial proposal presented in our previous work [5], and the methodological setup is as follows.

We used the RAxML application (version 8.2.12), which is widely used in bioinformatics for phylogenetic inference. The application is implemented in a hybrid manner, combining MPI for inter-node parallelism and *PThreads* for

intra-node parallelism. The input parameters considered in this study include the *bootstrap* value, which defines the number of statistical replicates performed to increase the reliability of the phylogenetic trees generated by the application, and the input file size. We consider two *bootstrap* values (100 and 1000) and eight input file sizes (182.6, 193.9, 225.7, 237.0, 266.2, 280.3, 368.2, and 473.2 Kilobytes) corresponding to multiple sequence alignments of Dengue virus genomes, obtained from Bacterial and Viral Bioinformatics Resource Center (BV-BRC) [16].

The experiments were conducted on the Santos Dumont supercomputer[2] (SDumont), operated by the National Laboratory for Scientific Computing (LNCC). The executions used up to 5 compute nodes, each equipped with two *Intel Xeon Cascade Lake Gold 6252* processors (24 physical cores per *socket*), totaling 48 physical cores per node, with *hyperthreading* disabled. Each node has 384 GB of RAM, runs the *Red Hat Enterprise* Linux 8.8 operating system with *kernel* version 4.18, and is interconnected via an *InfiniBand* network. Job management was performed using the *SLURM Workload Manager* (version 17.02).

The experiments consider the application of the proposed methodology over a subset of the possible resource allocations on SDumont, including #N equal to 1 or 5 and executions with #T equal to 2, 24, or 48. Based on these combinations, four resource allocation solutions in HPC were defined for comparison, as described below:

- **Oracle:** represents the best possible configuration of (#N, #T) for each combination of *bootstrap* values and input files, according to the possible values defined for each of these parameters;
- **MaxRes:** a strategy that prioritizes maximum performance, in which the user allocates #N = 5 compute nodes and #T = 48 *threads*;
- **MinRes:** a strategy focused on reducing computational costs, in which the user allocates #N = 1 compute node and #T = 2 *threads*, since the RAxML version used requires at least two *threads* for execution.
- **ComRes:** a strategy based on the most frequent configuration in the Oracle (observed in 60% of the cases), considering all *bootstrap* values and input files, in which #N = 5 compute nodes and #T = 2 *threads* are allocated.

The EDP values were computed according to Eq. (1), using execution time and energy consumption data collected as follows:

- **Execution Time.** Execution times were obtained using the `sacct`[3] command from the *Simple Linux Utility for Resource Management* (SLURM) [22], specifically the *ElapsedRaw* field, which provides the total duration of a *job* in seconds;
- **Energy Consumption.** Energy values were collected from counters provided by the *Running Average Power Limit* (RAPL) [10], with proper handling of possible *overflows* – energy readings were performed at intervals of up to 10 s.

[2] http://sdumont.lncc.br.
[3] https://slurm.schedmd.com/sacct.html.

For data collection, 5 executions of the RAxML application were performed for each of the 96 possible combinations of 2 *bootstraps* × 8 input files × 2 numbers of compute nodes × 3 numbers of *threads* per node, totaling 480 executions. Based on this data, we generated a dataset that enables us to move beyond exact-match accuracy and investigate model behavior from a broader perspective, including: *(i)* exploratory analysis of execution data and the variability across repeated runs, in Sect. 5; and *(ii)* the interpretation of prediction errors in terms of time and energy, in Sect. 6.

For error analysis *(ii)*, we compare all strategies with the Oracle using the normalized difference:

$$\text{Error} = (\text{Metric}_{\text{model}} - \text{Metric}_{\text{Oracle}}) / \text{Metric}_{\text{Oracle}},$$

where $\text{Metric}_{\text{model}}$ denotes the value of the evaluated metric for the suggested configuration and $\text{Metric}_{\text{Oracle}}$ denotes the corresponding value for the Oracle configuration.

5 Exploratory Data Analysis

In this section, we analyze the experimental data collected from RAxML executions. As described in the Sect. 2, the experimental dataset consists of repeated executions of RAxML collected under different combinations of input parameters and resource configurations, allowing us to observe the variability associated with execution time, energy consumption, and EDP. This analysis provides a broader understanding of how execution time, energy consumption, and EDP vary across different input sizes, bootstrap values, and resource configurations, while also supporting the preprocessing choices adopted later in the study.

5.1 Overview of the Raw Experimental Data

Figure 1 presents the raw execution data in the time (X-axis) and energy (Y-axis) space, organized by input size (columns) and bootstrap value (rows). Each point represents one observed execution configuration; marker style indicates the number of nodes, marker size indicates the number of threads, and color represents the corresponding EDP value: e.g., in the plot corresponding to Bootstrap=100 and RAxML input file size = 368.2 KB, the observation which stands out from the rest regards an execution with 5 nodes (for its marker style), 48 threads (for its marker size), and EDP $\simeq 10^7$ (for its "sea green" color) which corresponds to how much it spent regarding time $\simeq 10^2$ (for its position regarding the X-axis) and energy $\simeq 10^5$ (for its position regarding the Y-axis).

We can observe two patterns: *(i)* as the input size increases, the points tend to move toward regions of higher execution time and higher energy consumption; and *(ii)* for the same input size, the executions with bootstrap equal to 1000 generally appear in a more expensive region of the space than those with bootstrap equal to 100, reflecting the additional computational cost associated

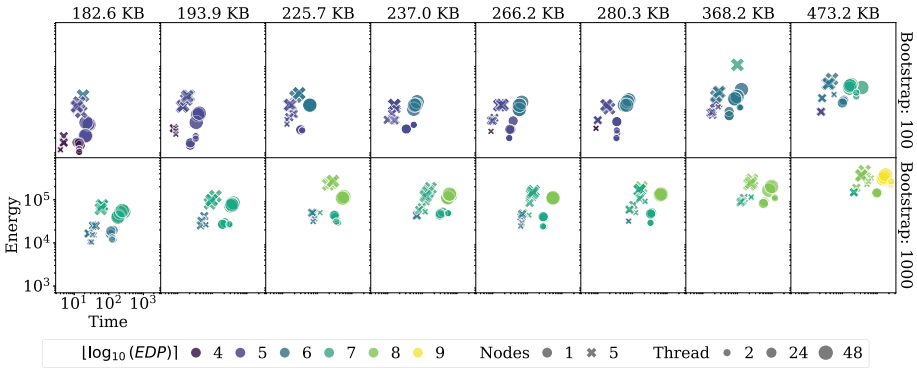

Fig. 1. Scatter plots of raw data. The size of the respective RAxML input files is indicated on top of each grid column.

with larger bootstrap values. The figure suggests that the relationship between runtime and energy consumption is not simple. In several panels, configurations that achieve shorter runtimes do not necessarily have lower energy consumption, and vice versa. This indicates the presence of a trade-off between the two objectives and justifies the use of EDP as a composite criterion.

5.2 Aggregated Configurations

Since each resource configuration was executed multiple times, the next step consists of summarizing repeated executions into a single representative point per configuration. In this work, we adopt the median of the observed values for each complete configuration. Figure 2 shows the resulting aggregated representation in the same time and energy space as Fig. 1, now enriched with the corresponding Pareto frontier (red line) and the Oracle solution (circle), defined as the configuration with minimum EDP, for each combination of bootstrap and input size. It can be observed that in each graph, each marker is unique, representing a singular combination of the number of nodes (1 or 5) and the number of threads (2, 24 or 48), so that there are $2 \times 3 = 6$ points in each graph.

Compared to the raw data, the aggregated view provides a clearer picture of the effective trade-offs between configurations. In each panel, only a small subset of the available configurations lies on the Pareto frontier (red line), indicating that only a few alternatives are truly competitive when execution time and energy are considered together. Figure 2 also helps connect the Pareto frontier and the Oracle construction. The Oracle generally lies on the Pareto frontier, which is expected since the EDP combines time and energy. At the same time, the frontier highlights that the Oracle is only one of the non-dominated solutions and that other configurations may remain attractive depending on the desired balance between runtime and energy consumption. This perspective is particularly useful for the later analysis of prediction errors, since a recommendation different from the Oracle may still be on or near the Pareto frontier.

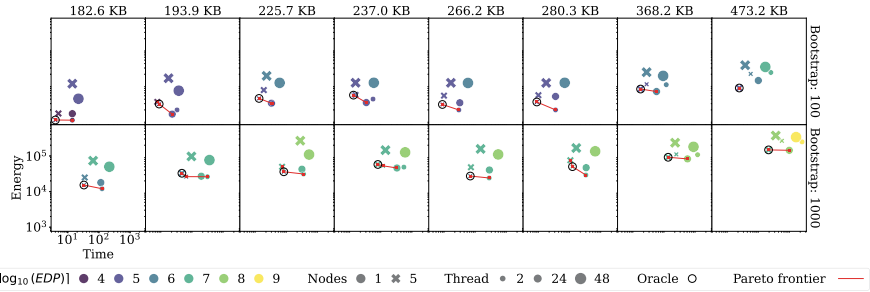

Fig. 2. Scatter plots of summarized data. The size of the respective RAxML input files is indicated on top of each grid column.

5.3 Execution Variability

Aggregating repeated executions inevitably discards part of the variability present in the raw data: the difference between Fig. 1 and Fig. 2 highlights this. To assess this loss, we analyze the coefficient of variation of EDP, denoted by cv(EDP), for repeated runs of the same complete configuration (i.e., a combination of a number of nodes, threads, bootstraps, and input size). Figure 3 presents histograms of these values for each combination of bootstrap and input size, so that each graph is based on the same 6 elements represented in each panel in Fig. 2, each of which in their turn is the aggregation of 5 executions of the same complete configuration, which were individually depicted in Fig. 1. In each panel of Fig. 1, the X-axis represents the observed cv(EDP) values and the Y-axis (Count) represents the number of configurations in each interval: the height of each bar regards the number of configurations whose repeated executions have a similar level of relative variability of EDP, and the sum of these heights is always 6.

In general, the distributions are similar across panels, with most values concentrated on the left side of the X-axis, indicating relatively low variability in the majority of cases. Only one clearly more extreme case appears in the dataset, corresponding to input size 368.2 KB and bootstrap 100, where the CV exceeds 1. Although this case remains isolated, it also suggests that the outlier filtering strategy could still be made more restrictive. Even so, the general pattern indicates that the variability among repeated executions is limited enough to support the use of a representative statistic per configuration.

Figure 4 shows the coincidence frequency between different summarization functions with respect to determining the optimal RAxML parameters. More specifically, for each pair of summarization functions, the value in the heatmap indicates how often they select the same resource configuration as optimal when repeated executions are summarized using median, mean, minimum, or maximum EDP values. Thus, a value of 1.00 means that two functions always lead to the same parameter choice, whereas smaller values indicate less agreement. The color scale reinforces this interpretation, with lighter tones corresponding

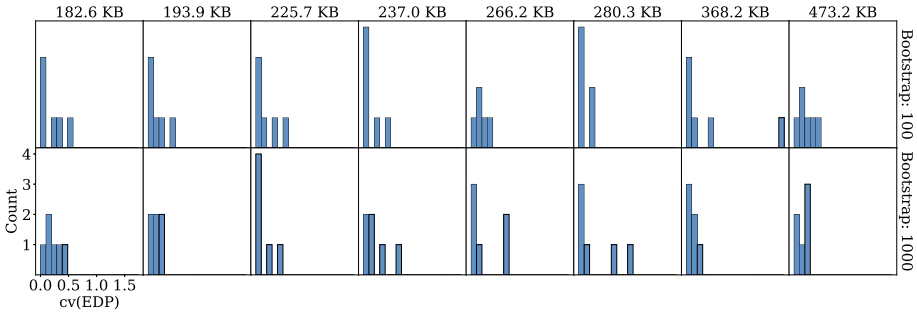

Fig. 3. Histograms of the coefficient of variation of EDP, grouped by input size and bootstrap value.

to higher coincidence frequencies and darker tones corresponding to lower ones. This comparison is relevant because each summarization function reflects a different view of configuration quality: the mean captures the average case, whereas the minimum and maximum represent more optimistic and more conservative perspectives, respectively.

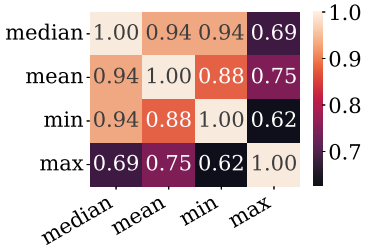

Fig. 4. Coincidence frequency of summarization functions with respect to determining optimal RAxML parameters.

The results indicate a high coincidence frequency between the median-based selection and those based on the mean and the minimum, with values of 0.94 in both cases. This suggests that, after outlier filtering, using the median or the mean rarely changes which configuration is selected as optimal. It also indicates that the configurations achieving the lowest observed EDP are often the same as those selected when more representative summaries are used. In contrast, the coincidence values involving the maximum are substantially lower, ranging from 0.62 to 0.75. This indicates that selecting configurations according to the worst observed EDP may lead to different parameter choices, since this criterion emphasizes unfavorable cases and therefore provides a more conservative view of configuration quality.

6 Prediction Errors Analysis

While our previous study [5] focused primarily on how often the proposed Extra Tree Regression (ETR) model matches the Oracle configuration and how close its recommendations are in terms of EDP, the results presented here allow a more detailed interpretation of prediction errors. Instead of evaluating the recommendations only through normalized EDP, we also examine normalized execution time and normalized energy consumption, always with respect to the Oracle configuration defined by the minimum real EDP.

This perspective is important because a recommendation that does not coincide with the Oracle may still fail in different ways. On the one hand, the model may suggest a more aggressive configuration, reducing runtime but increasing energy consumption excessively. On the other hand, it may recommend a more conservative configuration, saving energy but increasing runtime so much that the overall EDP becomes worse.

6.1 Normalized EDP

Figure 5 presents the normalized EDP (Y-axis) obtained by the evaluated strategies (represented by groups of bars in distinct colors) for each combination of bootstrap and input size. Values closer to 1 indicate closer agreement with the Oracle, whereas larger values indicate worse recommendations.

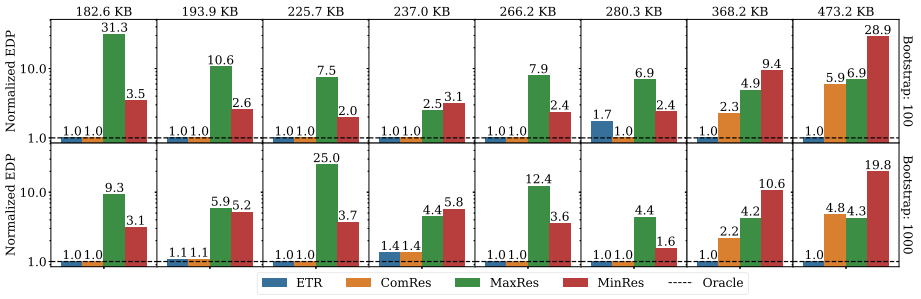

Fig. 5. EDP across all bootstrap samples for RAxML parameters.

The results show that MaxRes and MinRes do not match the Oracle in any of the evaluated cases. This result can be directly related to the exploratory analysis presented in Sect. 5.2. The Oracle solutions are consistently associated with configurations with 5 nodes, which explains why MinRes, fixed at 1 node and 2 threads, never becomes optimal. Furthermore, among the 5 node configurations, the Oracle is not associated with the 48 thread configuration, which explains why MaxRes, fixed at 5 nodes and 48 threads, also fails to match the Oracle in all cases.

On the other hand, ComRes is generally more competitive than these two extreme baselines, as it matches the most frequent Oracle configuration in the dataset. Even so, it still exhibits noticeable degradation in several cases, especially for larger inputs. These results reinforce the idea that fixed allocation policies are often inadequate for balancing execution time and energy consumption. Allocating the maximum amount of resources does not guarantee the best energy and performance, and minimizing resource use can be equally inefficient.

However, the strategy based on the ETR shows consistency. In both bootstrap configurations, its normalized EDP remains the same or very close to that of the Oracle in most cases, with only a few visible deviations. This confirms that the predictive approach is robust.

6.2 Normalized Execution Time

Figure 6 shows the same comparison from the perspective of normalized execution time. Again, the ETR recommendations remain close to the Oracle for most inputs, although some deviations become more visible.

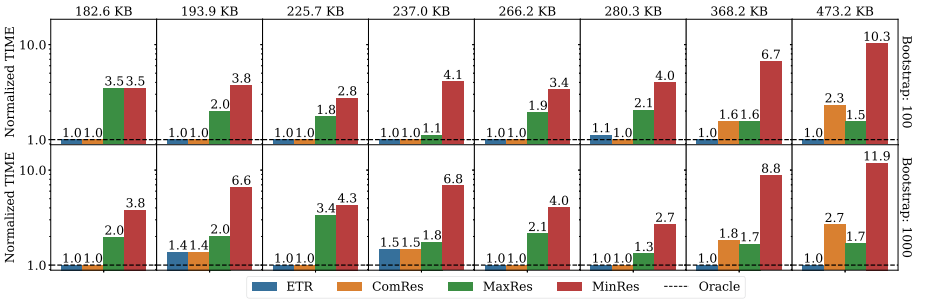

Fig. 6. Execution time across all bootstrap samples for RAxML parameters.

The results demonstrate that allocating more resources does not systematically lead to shorter execution times. Although an average user of HPC systems might expect MaxRes to dominate in terms of execution time, the figure shows that this is not always the case. For several inputs, its execution time remains above Oracle's, indicating that the additional resources do not translate into proportional performance gains. One hypothesis for why this happens is the overhead associated with coordination, synchronization, and parallel execution when more resources than necessary are employed. These results suggest that, under the workload and execution scales considered in this study, the benefits of additional resources may be partially offset by communication and coordination costs.

MinRes, in turn, performs poorly in this dimension, especially for larger inputs. Its normalized runtime grows sharply in several cases, showing that

reducing the resource allocation too aggressively may significantly increase execution time. This pattern characterizes a type of error in which the configuration is economical in terms of resource usage, but too slow to remain competitive. ComRes generally occupies an intermediate position. Although it performs better than MinRes from an execution time perspective, it still does not match the consistency of the recommendation based on ETR. Thus, this analysis indicates that both over-allocation and very conservative resource allocation strategies can fail in terms of execution time.

6.3 Normalized Energy Consumption

Figure 7 presents the results in terms of normalized energy consumption. Here, the contrast among the evaluated strategies becomes even clearer.

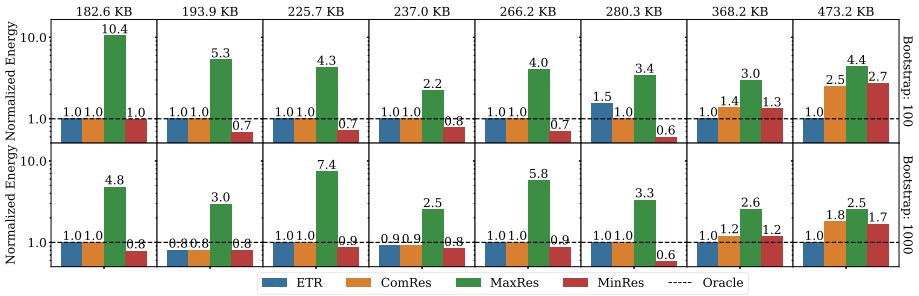

Fig. 7. Energy across all bootstrap samples for RAxML parameters.

MaxRes consistently exhibits the highest normalized energy consumption, often by a large margin, suggesting it is an energy-costly policy. Even in situations where additional resources can reduce execution time, the energy cost often outweighs this benefit. This confirms that systematically using the largest available resource configuration is not a suitable strategy for energy-efficient execution.

MinRes shows more varied behavior. For smaller input sizes, its normalized energy remains below or close to the Oracle, indicating that reducing resource usage can indeed help control energy consumption. However, this pattern does not hold in all cases; for larger input sizes, MinRes can also exceed Oracle's value in energy consumption. This is a particularly interesting result, as it shows that even a conservative configuration can become energy inefficient when the execution time penalty becomes too large (Fig. 6).

ETR recommendations also remain close to the Oracle, with only a few moderate deviations. These cases are informative because they show that when the model does not exactly reproduce the Oracle, the deviation is usually associated with a small change in the balance between execution time and energy, rather than a clearly inadequate recommendation.

7 Conclusion

This work extended our previous ETR-based approach for energy-efficient resource allocation in HPC by moving from an accuracy-only evaluation to a broader analysis of reliability and error behavior. Using RAxML as a case study, we investigated the structure of the execution data, the variability across repeated runs, and the practical meaning of prediction deviations with respect to the Oracle configuration.

The exploratory analysis showed that summarizing repeated executions with representative statistics is appropriate for this dataset, since most configurations present low relative variability. In addition, the time–energy space confirmed that only a subset of configurations is truly competitive, as indicated by the Pareto frontier, and that the Oracle is typically located among these non-dominated solutions. This Pareto-based perspective shows that non-dominated configurations other than the Oracle may also be relevant, depending on the desired trade-off between execution time and energy consumption.

From the prediction-error perspective, the results indicate that the ETR recommendations remain consistently close to the Oracle across EDP, execution time, and energy consumption. Although the exact-match accuracy is around 81.25%, most non-exact predictions still incur small practical regret. In contrast, fixed strategies based on extreme allocations (MaxRes and MinRes) systematically fail to reproduce Oracle decisions and exhibit distinct inefficiency profiles: MaxRes tends to incur high energy overhead, while MinRes frequently suffers from excessive runtime. These findings reinforce that trustworthy support for HPC resource allocation should be evaluated not only by exact matches but also by the quality and interpretability of non-exact recommendations.

Overall, the study provides additional evidence that the proposed predictive strategy is a robust and practical alternative for sustainable HPC operation. A limitation of this study is that it considers only one application and one HPC environment. Broader validation across other workloads and platforms remains future work. In addition, we intend to incorporate explicit uncertainty estimates into the recommendation process and investigate confidence-aware policies for low-confidence predictions.

Acknowledgments. The authors acknowledge the National Laboratory for Scientific Computing (LNCC/MCTI) for providing access to HPC resources on the SDumont supercomputer. This research was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grant no.440360/2022-6, and by Portuguese national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., through project UIDB/05037/2025 with DOI 10.54499/UIDB/05037/2025 and BULL Ltda.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. de A. Rocha, H.M.G., Schwarzrock, J., Lorenzon, A.F., Beck, A.C.S.: Using machine learning to optimize graph execution on NUMA machines. In: Proceedings of the 59th ACM/IEEE Design Automation Conference, pp. 1027–1032 (2022)
2. Aaen Springborg, A., Albano, M., Xavier-de Souza, S.: Automatic energy-efficient job scheduling in HPC: a novel slurm plugin approach. In: Proceedings of the SC 2023 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, pp. 1831–1838. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3624062.3624265
3. Arachchige, C.N.P.G., Prendergast, L.A., Staudte, R.G.: Robust analogs to the coefficient of variation. J. Appl. Stat. **49**(2), 268–290 (2020). https://doi.org/10.1080/02664763.2020.1808599, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9196089/
4. Carastan-Santos, D., Da Costa, G., Poquet, M., Stolf, P., Trystram, D.: Lightweight prediction for improving energy consumption in HPC platforms. In: Carretero, J., Shende, S., Garcia-Blas, J., Brandic, I., Olcoz, K., Schreiber, M. (eds) Euro-Par 2024. LNCS, vol. 14801, pp. 152–165. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-69577-3_11
5. Coelho, M., et al.: Modelagem preditiva de energy-delay product para otimização de submissões em supercomputadores. In: Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD), pp. 458–469. SBC (2025). https://doi.org/10.5753/sscad.2025.16760
6. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006). https://doi.org/10.1007/s10994-006-6226-1
7. Gonzalez, R., Horowitz, M.: Energy dissipation in general purpose microprocessors. IEEE J. Solid-State Circuits **31**(9), 1277–1284 (1996)
8. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS 2022, pp. 507–520. ACM, Curran Associates Inc., Red Hook (2024)
9. Imes, C., Hofmeyr, S., Hoffmann, H.: Energy-efficient application resource scheduling using machine learning classifiers. In: Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3225058.3225088
10. Khan, K.N., Hirki, M., Niemi, T., Nurminen, J.K., Ou, Z.: RAPL in action: experiences in using RAPL for power measurements. ACM Trans. Model. Perform. Eval. Comput. Syst. **3**(2) (2018). https://doi.org/10.1145/3177754
11. Khosravi, A., Sandoval, O.R., Taslimi, M.S., Sahrakorpi, T., Amorim, G., Garcia Pabon, J.J.: Review of energy efficiency and technological advancements in data center power systems. Energy Build. **323**, 114834 (2024). https://doi.org/10.1016/j.enbuild.2024.114834, https://www.sciencedirect.com/science/article/pii/S0378778824009502
12. Kunas, C.A., Rossi, F.D., Luizelli, M.C., Calheiros, R.N., Navaux, P.O.A., Lorenzon, A.F.: NeurOPar, a neural network-driven EDP optimization strategy for parallel workloads. In: 2023 IEEE 35th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 170–180 (2023). https://doi.org/10.1109/SBAC-PAD59825.2023.00026

13. Legriel, J., Le Guernic, C., Cotton, S., Maler, O.: Approximating the pareto front of multi-criteria optimization problems. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 69–83. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12002-2_6
14. Lorenzon, A.F., Beck, A.C.S., Navaux, P.O., Messer, B.: Energy-efficient GPU allocation and frequency management in exascale computing systems. In: ISC High Performance 2025 Research Paper Proceedings (40th International Conference), pp. 1–11. Prometeus GmbH (2025). https://doi.org/10.23919/ISC.2025.11018306
15. Maros, A., Almeida, J., Murai, F., da Silva, A.P., Ardagna, D., Lattuada, M.: Aprendizado de máquina para previsão do tempo de execução de aplicações spark. In: Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, pp. 197–210. SBC, Porto Alegre (2019). https://doi.org/10.5753/sbrc.2019.7360, https://sol.sbc.org.br/index.php/sbrc/article/view/7360
16. Olson, R.D., et al.: Introducing the bacterial and viral bioinformatics resource center (BV-BRC): a resource combining PATRIC, IRD and ViPR. Nucleic Acids Res. **51**(D1), D678–D689 (2023). https://doi.org/10.1093/nar/gkac1003
17. Papadimitriou, G., Chatzidimitriou, A., Gizopoulos, D.: Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore CPUs. In: 2019 IEEE International Symposium on High Performance Computer Architecture, pp. 133–146. IEEE (2019)
18. Patel, T., Wagenhäuser, A., Eibel, C., Hönig, T., Zeiser, T., Tiwari, D.: 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), pp. 799–809 (2020). https://doi.org/10.1109/IPDPS47924.2020.00087
19. Porto, A.H., Coelho, M., Rocha, H.M., Osthoff, C., Ocana, K., Cardoso, D.O.: Assuming the best: towards a reliable protocol for resource usage prediction for high-performance computing based on machine learning. Future Gener. Comput. Syst. 108070 (2025)
20. Scravaglieri, L., Popov, M., Lima Pilla, L., Guermouche, A., Aumage, O., Saillard, E.: Optimizing performance and energy across problem sizes through a search space exploration and machine learning. J. Parallel Distrib. Comput. **180**, 104720 (2023). https://doi.org/10.1016/j.jpdc.2023.104720
21. Stamatakis, A.: RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics **30**(9), 1312–1313 (2014). https://doi.org/10.1093/bioinformatics/btu033
22. Yoo, A.B., Jette, M.A., Grondona, M.: SLURM: simple Linux utility for resource management. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2003. LNCS, vol. 2862, pp. 44–60. Springer, Heidelberg (2003). https://doi.org/10.1007/10968987_3