Modelagem Preditiva de *Energy-Delay Product* para Otimização de Submissões em Supercomputadores

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

[1]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ, Brasil

{micaella,xandao,mayk,isamuniz,karyann,osthoff}@lncc.br

[2]Universidade de Coimbra, CeBER, Faculdade de Economia
Coimbra, Portugal

docardoso@uc.pt

[3]Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS, Brasil

{navaux,arthur.lorenzon}@inf.ufrgs.br

Resumo. *O consumo de energia em sistemas de Processamento de Alto Desempenho (PAD) tem se tornado um desafio cada vez mais relevante diante do aumento da demanda computacional e da execução de aplicações científicas em larga escala. Fatores relacionados à alocação de recursos, como a quantidade de nós (#N) e o número de threads por nó (#T), impactam diretamente tanto o tempo de execução quanto a eficiência energética, frequentemente avaliada por meio do Energy-Delay Product (EDP). Neste trabalho, propomos uma metodologia baseada no Extra Trees Regressor para recomendar configurações de #N e #T que minimizem o EDP em aplicações paralelas, sem a necessidade de executar a aplicação ou instrumentar seu código. Como estudo de caso, utilizamos a aplicação RAxML e comparamos nossa abordagem com outras quatro estratégias de alocação de recursos em um sistema PAD. Os resultados mostram que a metodologia alcança 81,25% de precisão em relação a melhor estratégia comparada (nosso Oráculo), gerando soluções que diferem, em média, apenas 6,02% do Oráculo e proporcionam uma melhoria de 33,35% no EDP em comparação com a segunda melhor estratégia.*

1. Introdução

O consumo de energia em sistemas de Processamento de Alto Desempenho (PAD) tem se tornado um dos principais desafios da área, resultado do aumento da demanda por maior capacidade computacional e da intensificação de aplicações científicas de larga escala. Por exemplo, El Capitan, líder do *ranking* do TOP500 de junho de 2025, consome aproximadamente 29,6 megawatts (MW)[1], evidenciando o impacto energético dos sistemas de PAD. Em paralelo, *data centers* já respondem por mais de 1% do consumo global de eletricidade, o que pressiona a adoção de estratégias que equilibrem desempenho e eficiência

[1]https://www.top500.org/lists/top500/2025/06/

energética em supercomputadores[2] [Khosravi et al. 2024] .

Entre os fatores que impactam o consumo energético de um sistema de PAD, além de aspectos de baixo nível (e.g., frequência de unidades computacionais e *cache misses*), destacam-se também fatores de alto nível, como os parâmetros definidos pelo usuário durante a submissão de um *job*, incluindo o número de nós computacionais (#N) e o número de *threads* por nó (#T). Assim, torna-se essencial desenvolver mecanismos que auxiliem os usuários na escolha de configurações de #N e #T – também denotado pela tupla (#N,#T) – mais adequadas, de forma a equilibrar o tempo de execução e o consumo de energia, métrica tratada neste trabalho por meio do *Energy-Delay Product* (EDP) (ver Seção 5.3).

Existem diversas estratégias na literatura para otimizar o consumo de energia em sistemas de PAD ajustando configurações como *power capping*, *Dynamic Voltage and Frequency Scaling* (DVFS) e *Dynamic Concurrency Throttling* [Schwarzrock et al. 2025, Krzywaniak et al. 2018]. Contudo, técnicas baseadas em Aprendizado de Máquina (AM) têm ganhado destaque nos últimos anos devido à sua capacidade de capturar padrões complexos a partir dos parâmetros da aplicação (e.g., tamanho da carga de trabalho e número de iterações) ou de de dados coletados da execução das aplicações (e.g., tempo de execução, consumo de energia e uso de memória). Apesar do emprego dessas técnicas para apoiar a alocação de recursos [Lorenzon et al. 2025, Carastan-Santos et al. 2024], muitas dessas abordagens ainda apresentam limitações: algumas se concentram exclusivamente na previsão de tempo de execução [Maros et al. 2019] e outras dependem de instrumentação complexa [Kunas et al. 2023].

Com o objetivo de contornar as limitações descritas anteriormente, este trabalho propõe uma metodologia que usa *Extra Trees Regressor* (ETR) para indicar a configuração (#N,#T) de melhor EDP na execução de aplicações paralelas em sistemas de PAD. A metodologia consiste em duas fases principais: **(*i*)** construção de um modelo de ETR (*Preditor*) utilizando dados de execução anteriores sob um subconjunto de entradas da aplicação, variando diferentes combinações de #N e #T; **(*ii*)** dada uma nova entrada da mesma aplicação, o *Preditor* é utilizado para encontrar a configuração (#N,#T) que entrega o menor EDP – sem a necessidade de executar a aplicação ou instrumentar seu código.

Para validar a proposta, utilizamos como estudo de caso a aplicação RAxML [Stamatakis 2014] (ver Seção 5), uma aplicação representativa da área de bioinformática no contexto de sistemas de PAD, sendo amplamente empregada na inferência filogenética em larga escala. Ela combina processamento distribuído via *Message Passing Interface* (MPI) com paralelismo intra-nó por *PThreads*. Por se tratar de uma aplicação tanto *CPU-intensive* quanto *memory-intensive*, ela se mostra adequada para avaliar o impacto de diferentes configurações de alocação de recursos sobre a eficiência energética.

Em resumo, as principais contribuições deste trabalho são:

- Uma **metodologia** baseada no uso do ETR para apoiar a seleção de configurações de (#N,#T) na execução de aplicações paralelas em sistemas de PAD, visando melhorar a eficiência energética.

[2] `https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks`

- Emprego de uma **aplicação real** de bioinformática (RAxML) como estudo de caso representativo para validar a metodologia proposta em ambiente de PAD.

Nossos resultados experimentais, considerando 1 e 5 nós computacionais com 48 núcleos cada, indicam que a metodologia proposta atinge 81,25% de acurácia e gera soluções que, em média, diferem apenas 6,02% do Oráculo, além de apresentar um EDP 33,35% superior à segunda melhor entre as estratégias comparadas.

2. Fundamentação Teórica

Esta seção apresenta o *Extra Trees Regressor* (ETR), modelo utilizado na metodologia proposta, e a métrica *Energy-Delay Product* (EDP), nosso objetivo de otimização, fornecendo os conceitos necessários para a compreensão da proposta.

2.1. Modelagem do Problema com *Extra Trees Regressor (ETR)*

O ETR é um algoritmo de *ensemble* que combina múltiplas árvores de decisão construídas de forma totalmente expandida (*top-down*, sem poda). Sua principal característica é a introdução de aleatoriedade tanto na seleção dos atributos candidatos quanto na definição dos pontos de corte em cada nó. Essa estratégia reduz a variância do modelo, melhora a eficiência computacional e favorece a diversidade entre as árvores geradas. Diferentemente de métodos baseados em *bagging*, como o *Random Forest*, nos quais cada árvore é treinada a partir de uma amostra *bootstrap* do conjunto de treinamento, o ETR utiliza, por padrão, todo o conjunto de dados disponível, o que contribui para a redução do viés e para a obtenção de predições mais consistentes – embora também ofereça suporte à amostragem *bootstrap*. Além disso, ETR reduz a necessidade de ajuste de hiperparâmetros, o que simplifica seu uso prático. Para tarefas de regressão, as predições individuais das árvores são combinadas por meio da média aritmética, produzindo modelos robustos, com boa capacidade de generalização e menor propensão ao *overfitting* [Geurts et al. 2006].

O ETR foi selecionado neste estudo devido à sua adequação para dados tabulares, característica compartilhada com outros modelos *ensemble* baseados em árvores de decisão [Grinsztajn et al. 2024]. Além disso, o ETR é computacionalmente eficiente, como evidenciado em experimentos recentes [Porto et al. 2026], especialmente quando os conjuntos de dados disponíveis para treinamento são relativamente pequenos. No presente trabalho, cada conjunto de dados utilizado contém poucas amostras, reforçando a escolha do ETR por sua capacidade de gerar predições consistentes mesmo em cenários com quantidade limitada de dados.

2.2. *Energy-Delay Product* (EDP) como Métrica de Desempenho

O EDP é uma métrica que quantifica, em um único valor, o *trade-off* entre o tempo de execução e o consumo de energia de uma aplicação, sendo amplamente utilizada em contextos de otimização de sistemas de PAD sob restrição energética [Papadimitriou et al. 2019]. O cálculo é definido como: $EDP(app) = Energia(app) \times Tempo(app)$, onde $Energia(app)$ e $Tempo(app)$ correspondem ao consumo de energia e ao tempo de execução medidos durante a execução real da aplicação *app*.

3. Trabalhos Relacionados

Diversos estudos propõem o uso de técnicas de AM para prever configurações eficientes em múltiplas dimensões, como tempos de execução e consumo de energia, apoi-

ando decisões sobre alocação de recursos em ambientes PAD [Lorenzon et al. 2025, Kunas et al. 2023, Muralidhar et al. 2022].

Em [Lorenzon et al. 2025] aplica *Random Forest* para selecionar, em tempo de execução, a frequência da GPU e o nó de execução, otimizando o EDP em até 41% em 15 aplicações no supercomputador Frontier. Apesar da eficácia, restringe-se a GPUs e uso de DVFS, enquanto a proposta deste trabalho foca em CPUs e utiliza dados históricos de execuções como base preditiva. Já o NeurOPar [Kunas et al. 2023] utiliza redes neurais para prever número de *threads* e frequência de CPU, com bons resultados em EDP, mas demanda instrumentação detalhada e uma execução padrão para coleta de dados.

Os trabalhos [Carastan-Santos et al. 2024, Aaen Springborg et al. 2023] integram modelos preditivos ao escalonador para alocação de recursos sob restrições de potência ou metas de eficiência, mostrando a viabilidade de predições na submissão. No entanto, a proposta atual opera de forma independente, antecipando predições do EDP a partir de dados históricos de execuções, sem necessidade de acoplamento direto ao escalonador.

Outra pesquisa relevante foca no uso de contadores de desempenho e sensores de *hardware* para modelar o comportamento energético em tempo de execução. Em [Krzywaniak et al. 2018], são investigados os *trade-offs* entre tempo e energia aplicando limites de potência, via RAPL em arquiteturas distintas. Embora sem AM, apontam como futuro o uso de predições, um objetivo já incorporado neste trabalho pelo modelo ETR.

Estudos em *clusters* e *data centers* reais reforçam a viabilidade de abordagens preditivas. Em [Patel et al. 2020], a análise de 80 mil *jobs* em dois *clusters* acadêmicos mostrou que apenas três atributos (usuário, número de nós e tempo solicitado) permitem prever o consumo por nó com alta acurácia via árvore de decisão, alinhando-se à proposta deste estudo, que também usa atributos de submissão para prever EDP e recomendar configurações eficientes. Já [Kumar et al. 2022] comparou oito modelos de regressão para prever o *Power Usage Effectiveness* (PUE) em *data centers*, alcançando alta acurácia com medições instrumentadas, o que evidencia o potencial do AM em padrões energéticos complexos e reforça sua aplicabilidade na proposta atual.

Por fim, a revisão sistemática de [Muralidhar et al. 2022] destaca a fragmentação entre soluções existentes e defende abordagens integradas que combinem *hardware*, *software* e AM para decisões energéticas. A proposta aqui apresentada exemplifica essa integração ao utilizar dados reais de produção, sensores confiáveis (RAPL) e modelos preditivos, visando apoiar decisões energéticas de forma prática e automatizada.

3.1. Nossas Contribuições

Com base na análise dos trabalhos relacionados, esta proposta se distingue por três aspectos principais: (i) usa apenas parâmetros disponíveis na submissão, como tamanho da entrada e outros parâmetros da aplicação, prática observada apenas em [Patel et al. 2020]; (*ii*) dispensa instrumentação detalhada com *profiling*, prática observada apenas em [Carastan-Santos et al. 2024], [Aaen Springborg et al. 2023], [Patel et al. 2020], [Kumar et al. 2022]; (*iii*) adota o EDP como métrica composta para orientar decisões energéticas, critério observado apenas em [Lorenzon et al. 2025] e [Kunas et al. 2023]. Além disso, apenas três trabalhos ([Carastan-Santos et al. 2024], [Patel et al. 2020], [Kumar et al. 2022]) utilizam árvores de decisão com foco em CPUs. Portanto, este trabalho contempla simultaneamente os critérios (*i*)–(*iii*) acima. Além

disso, nossa metodologia é flexível e expansível, podendo ser aplicada a outras aplicações paralelas, inclusive aquelas avaliadas nos trabalhos relacionados aqui discutidos.

4. Proposta

Este trabalho propõe uma metodologia baseada no ETR para apoiar a seleção da configuração ótima de número de nós computacionais e de *threads* por nó (#N,#T) na execução de aplicações paralelas em sistemas de PAD. A metodologia é estruturada em duas fases principais: ***Construção do Preditor*** e ***Uso do Preditor***, conforme ilustrado na Fig. 1 e detalhado nas seções seguintes.

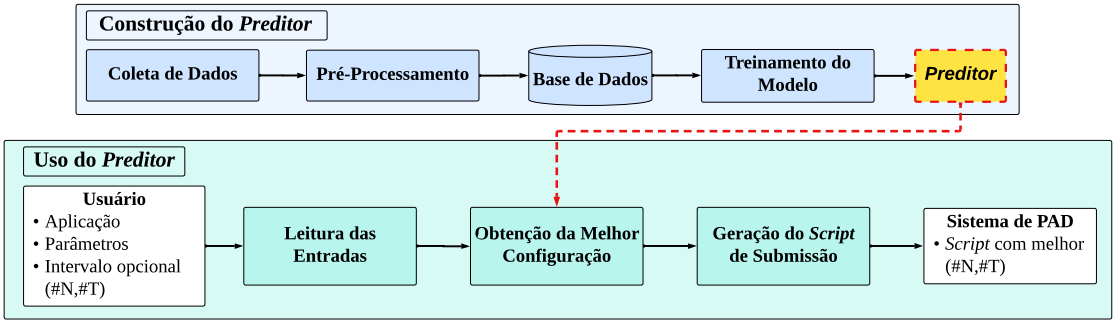


Figura 1. Fluxo geral da metodologia proposta.

4.1. Construção do *Preditor*

Esta fase tem como objetivo construir um modelo de ETR, denominado *Preditor*, capaz de estimar o EDP da execução de uma aplicação com base em suas execuções anteriores. O modelo considera variações de parâmetros comuns na submissão de *jobs* em sistemas de PAD (combinações de #N e #T), bem como parâmetros específicos da aplicação, que podem diferir conforme o domínio e o tipo de aplicação (e.g., tamanho das entradas, precisão numérica, número de iterações). A seguir, descrevemos cada etapa necessária para a construção do *Preditor*, as quais são também ilustradas na parte superior da Fig. 1:

Coleta de Dados. Para gerar o conjunto de dados necessário à construção do modelo preditivo, nossa metodologia realiza execuções da aplicação no ambiente de PAD, variando de forma controlada os parâmetros críticos de execução – as combinações de (#N,#T), bem como parâmetros específicos da aplicação, como já mencionado.

Pré-Processamento. Os dados coletados são pré-processados para assegurar a consistência dos resultados e sua adequação ao processo de construção do modelo *Preditor*. Nessa etapa, nossa metodologia realiza uma filtragem de ruídos decorrentes da coleta dos valores de energia – como eventuais *overflows* nos contadores. A detecção de *outliers* foi realizada por meio de um critério estatístico robusto, baseado na mediana (M) e no desvio absoluto mediano (D): calcula-se M e D dos valores de EDP e um ponto é classificado como *outlier* se estiver fora do intervalo $[M - k \cdot D,\ M + k \cdot D]$, onde k é um fator de sensibilidade escolhido que ajusta a tolerância à variabilidade (no nosso caso $k = 10^2$). Os dados resultantes desta etapa são armazenados em uma **Base de Dados** com formato tabular, em que cada linha representa uma execução distinta da aplicação, contendo as configurações dos parâmetros de execução seguidas dos respectivos valores de EDP, enquanto cada coluna corresponde a uma variável relevante para o processo preditivo.

Treinamento do *Preditor*. Essa etapa consiste no treinamento do modelo de ETR a partir dos dados coletados na etapa anterior. Para a implementação, empregamos a biblioteca *scikit-learn* em *Python*. Na construção do modelo, realizamos uma avaliação empírica do hiperparâmetro de profundidade máxima das árvores, variando valores de 5 a 10 com incrementos unitários. Essa análise tem como objetivo equilibrar capacidade preditiva, generalização e mitigação de sobreajuste, ao mesmo tempo em que permite capturar relações não lineares entre os parâmetros de entrada (como #N, #T e parâmetros específicos da aplicação) e os valores de EDP. Após a definição da profundidade máxima mais adequada, o modelo é treinado e resulta em um *Preditor*, que será utilizado na próxima fase para estimar o EDP de novas combinações de parâmetros, sem a necessidade de reexecutar a aplicação.

4.2. Uso do *Preditor*

Após o treinamento na fase anterior, o *Preditor* é disponibilizado como um executável em *shell script*, possibilitando sua utilização direta pelo usuário. A execução segue o formato: `./preditor app par [-n min:max:inc] [-t min:max:inc]`, em que `app` representa o binário da aplicação alvo e `par` os parâmetros de entrada necessários para sua execução, e as opções -n e -t são parâmetros opcionais que permitem definir intervalos com valores mínimo, máximo e incremento de #N e #T, respectivamente. Como saída, o *Preditor* gera automaticamente um *script* de submissão contendo a configuração predita de recursos computacionais (#N,#T) e demais configurações (e.g., partição/*queue*, carregamento de módulos e variáveis de ambiente) pronta para ser submetido ao gerenciador de filas SLURM. A seguir, detalhamos as etapas que viabilizam esse processo, também ilustradas na parte inferior da Fig. 1.

Leitura das Entradas. O *Preditor* recebe os parâmetros de entrada da aplicação, como o tamanho do arquivo e parâmetros específicos da aplicação. Opcionalmente, o usuário pode fornecer um intervalo customizado de valores de #N e #T a serem considerados na predição (detalhado adiante). Contudo, caso nenhum intervalo seja especificado, a metodologia adota automaticamente um conjunto de valores padrão, previamente ajustados às características do sistema de alto desempenho utilizado para a avaliação (ver Seção 5).

Obtenção da Melhor Configuração. Essa etapa consiste na execução de um procedimento que estima a melhor configuração (#N,#T), visando obter o menor valor de EDP, sem a necessidade de executar a aplicação. Conforme detalhado no Algoritmo 1, o procedimento considera um conjunto C de pares (#N,#T). Em cada iteração (linhas 3–7), o *Preditor* é utilizado para estimar o EDP correspondente à configuração avaliada, sendo selecionada, ao final do procedimento, aquela que apresentar o menor valor estimado de EDP – (n^*,t^*) na linha 8.

Geração do *Script* de Submissão. Por fim, a configuração (n^*, t^*) selecionada é automaticamente incorporada a um *script* de submissão, no qual são especificados os parâmetros da aplicação, bem como as diretivas necessárias para execução no gerenciador de filas (e.g., SLURM). Esse *script* resulta em uma descrição completa e pronta para submissão, assegurando a execução da aplicação com a configuração recomendada pelo *Preditor*.

Algoritmo 1: Obtém a melhor configuração (#N,#T) base *Preditor*.

Input: $C \rightarrow$ Conjunto de pares (#N,#T)
$e \rightarrow$ Parâmetros da aplicação
Output: Configuração (n^*, t^*) com menor EDP estimado

```
1 EDP_min ← ∞
2 (n*, t*) ← (0, 0)
3 foreach (n, t) ∈ C do
4     edp ← Preditor.estimativa(n, t, e)
5     if edp < EDP_min then
6         EDP_min ← edp
7         (n*, t*) ← (n, t)
8 return (n*, t*)
```

5. Metodologia

Esta seção descreve a metodologia adotada para obter os resultados, abrangendo o ambiente de execução, a aplicação utilizada, a preparação dos dados usados para treinar e avaliar o *Preditor* e as configurações dos experimentos.

5.1. Ambiente Experimental

Os experimentos foram conduzidos no supercomputador Santos Dumont[3] (SDumont), operado pelo Laboratório Nacional de Computação Científica (LNCC). As execuções utilizaram até 5 nós computacionais, sendo que cada nó é equipado com dois processadores *Intel Xeon Cascade Lake Gold 6252* (24 núcleos físicos por *socket*), totalizando 48 núcleos físicos por nó, com *hyperthreading* desabilitado. Cada nó possui 384 GB de memória RAM, sistema operacional *Red Hat Enterprise* Linux 8.8 com *kernel* versão 4.18 e está interconectado por rede *InfiniBand*. O gerenciamento dos *jobs* foi realizado por meio do *SLURM Workload Manager* (versão 17.02).

5.2. Aplicação Avaliada

Para os experimentos, consideramos como estudo de caso a aplicação **RAxML** (versão 8.2.12), amplamente utilizada na bioinformática para inferência filogenética. A aplicação está implementada de forma híbrida, combinando MPI para paralelismo entre nós e *PThreads* para paralelismo intra-nó. Os parâmetros de entrada considerados nesse estudo foram o valor de *bootstrap*, que define o número de replicações estatísticas realizadas para aumentar a confiabilidade das árvores filogenéticas geradas pela aplicação, e o tamanho do arquivo de entrada: foram considerados dois valores de *bootstrap* (100 e 1000) e oito tamanhos de arquivos de entrada (187.026, 198.574, 231.143, 242.689, 272.549, 287.037, 377.019, 484.520 *bytes*) de alinhamentos múltiplos de genomas do vírus da Dengue [Olson et al. 2023].

5.3. Medida de Desempenho

Os valores de EDP foram calculados conforme a equação apresentada na Seção 2.2, utilizando os dados de tempo de execução e consumo de energia, coletados da seguinte maneira:

[3] `http://sdumont.lncc.br`

Tempo de Execução. Os tempos de execução foram obtidos por meio do comando `sacct`, do *Simple Linux Utility for Resource Management* (SLURM), utilizando o campo *ElapsedRaw*, que fornece a duração total de um *job* em segundos.

Consumo de Energia. Os valores de energia foram coletados a partir dos contadores fornecidos pelo *Running Average Power Limit* (RAPL), com tratamento adequado de possíveis *overflows*, conforme descrito na etapa de pré-processamento da Seção 4.1. Para reduzir a ocorrência de *overflows*, as leituras de energia foram realizadas em intervalos de até 10 segundos.

Para a coleta de dados, foram realizadas cinco execuções da aplicação RAxML em cada uma das 96 combinações possíveis de 2 *bootstraps* $\times$ 8 arquivos de entrada $\times$ 2 número de nós de computação $\times$ 3 número de *threads* por nó, totalizando 480 execuções. Os resultados apresentados na Seção 6 correspondem à mediana das cinco repetições de cada configuração.

5.3.1. Configurações Avaliadas

Os experimentos consideram a aplicação da metodologia proposta em um subconjunto das possíveis alocações de recursos no SDumont, contemplando #N igual a 1 ou 5 e execuções com #T igual a 2, 24 ou 48. A partir dessas combinações, foram definidas quatro soluções de alocação de recursos em PAD para comparação, que são descritas a seguir:

- **Oráculo**: representa a melhor configuração possível de (#N,#T) para cada combinação de *bootstrap* e arquivos de entrada – de acordo com os possíveis valores definidos para cada um desses parâmetros.
- **MaxRec**: estratégia que prioriza o máximo desempenho, na qual o usuário aloca #N = 5 nós de computação e #T = 48 *threads*.
- **MinRec**: estratégia voltada à redução dos custos computacionais, na qual o usuário aloca #N = 1 nó de computação e #T = 2 *threads*, já que a versão do RAxML utilizada exige, no mínimo, duas *threads* para execução.
- **ComRec**: estratégia baseada na configuração mais frequente no Oráculo (obtido em 60% das vezes), considerando todos os valores de *bootstrap* e arquivos de entrada, na qual #N = 5 nós de computação e #T = 2 *threads* são alocados.

6. Resultados

Esta seção apresenta os resultados obtidos com a metodologia proposta, em comparação com as demais estratégias avaliadas (Oráculo, MaxRec, MinRec e ComRec), considerando a validação do modelo e o desempenho em termos de EDP.

6.1. Avaliação do Modelo

Para a validação do modelo proposto, adotamos a estratégia *Leave-One-Group-Out* (LOGO), a fim de comprovar sua aplicabilidade efetiva no estudo de caso. LOGO é uma técnica de validação cruzada em que o conjunto de dados é particionado em grupos mutuamente exclusivos. No presente trabalho, cada grupo corresponde a uma combinação de *Bootstrap* e *Tamanho* (ver Seção 5.2). Em cada iteração, um grupo é completamente removido do treinamento e utilizado como conjunto de teste, enquanto os demais são empregados para treinar o modelo. No conjunto de teste, o modelo gera uma única previsão

de configuração (#N,#T) para cada combinação, selecionando o par que minimiza o EDP estimado, conforme descrito na Seção 4.2. Esse procedimento é repetido até que todos os grupos tenham sido usados como conjunto de teste ao menos uma vez.

6.2. Cenários de Avaliação

Esta seção descreve os cenários experimentais utilizados para avaliar a capacidade dos modelos de recomendar configurações de *hardware* que minimizem o EDP para a aplicação RAxML em sistemas PAD. O objetivo é simular o uso real do modelo, no qual ele deve prever a configuração ótima (#N,#T) para um novo par de parâmetros (*bootstrap* e tamanho do arquivo) não visto durante o treinamento.

A Tabela 1 apresenta as soluções de (#N,#T) geradas pela nossa metodologia, validadas com a estratégia LOGO (ETR), em comparação com o Oráculo. A tabela também inclui a diferença relativa entre as estratégias, expressa como o erro relativo adimensional, calculado pela razão entre o EDP mediano da configuração sugerida pelo ETR e o EDP mediano do Oráculo, com valores menores indicando maior proximidade ao Oráculo. Os resultados estão organizados por valor de *bootstrap* (100 e 1000) e abrangem os oito tamanhos de arquivos avaliados. Esses resultados destacam que a nossa metodologia consegue encontrar a mesma configuração do Oráculo em 13 dos 16 casos (acurácia de 81,25%).

Tabela 1. Comparação das soluções da metodologia proposta (ETR) com relação ao Oráculo.

Bootstrap	**Métrica**	**Tamanho**							
		187026	198574	231143	242689	272549	287037	377019	484520
100	ETR (#N,#T)	5,2	5,2	5,2	5,2	5,2	5,24	5,24	5,24
	Oráculo (#N,#T)	5,2	5,2	5,2	5,2	5,2	5,2	5,24	5,24
	Diferença	0,00	0,00	0,00	0,00	0,00	0,74	0,00	0,00
1000	ETR (#N,#T)	5,2	5,2	5,2	5,2	5,2	5,2	5,24	5,24
	Oráculo (#N,#T)	5,2	5,24	5,2	5,24	5,2	5,2	5,24	5,24
	Diferença	0,00	0,08	0,00	0,35	0,00	0,00	0,00	0,00

A Tabela 2 apresenta uma análise detalhada da qualidade das soluções das diferentes estratégias avaliadas, considerando a diferença relativa entre o EDP de cada estratégia (MinRec, MaxRec, ComRec, ETR) e o EDP do Oráculo. A tabela mostra a **Média** aritmética (e respectivo **Desvio** padrão), a menor diferença observada (**Min.**), os quartis (**25%-Q**, **50%-Q** e **75%-Q**) e a maior diferença (**Max.**). Os resultados demonstram que a proposta supera as demais estratégias, uma vez que apresenta menor diferença relativa média (0,07 com desvio padrão de 0,20), além de **25%-Q**, mediana (50%-Q) e **75%-Q** iguais a zero. Isso indica que, em pelo menos 75% dos casos, o modelo foi capaz de identificar exatamente a configuração de menor EDP, isto é, a mesma selecionada pelo Oráculo. Além disso, o baixo desvio padrão (0,20) evidencia a precisão e a consistência do modelo nas recomendações geradas.

Considerando as outras estratégias avaliadas, a ComRec apresentou desempenho satisfatório, com mediana da diferença relativa igual a zero, o que indica que, em pelo menos 50% dos casos, a configuração coincidiu com a do Oráculo. Contudo, o desvio padrão mais elevado (1,49) revela que, nos casos em que a ComRec não corresponde à solução ótima, as diferenças relativas tendem a ser mais acentuadas. Em contraste, a

MinRec apresentou resultados significativamente inferiores, com média de 5,72 e desvio padrão de 7,51, evidenciando que essa configuração raramente corresponde à escolha ideal em termos de eficiência energética. A diferença relativa máxima observada (27,87) mostra o impacto negativo dessa estratégia em determinadas combinações de entrada. Por sua vez, a MaxRec obteve os piores resultados, com diferença relativa média de 8,28 e máxima de 30,34. Esses achados demonstram que o superdimensionamento de recursos não garante melhor eficiência energética, destacando a necessidade de estratégias que promovam a alocação otimizada dos recursos em sistemas de PAD, assim como a proposta no presente trabalho.

Tabela 2. Comparação entre as diferenças relativas das estratégias comparadas com relação ao Oráculo.

Métrica	Média	Desvio	Min.	25%-Q	50%-Q	75%-Q	Max.
MinRec	5,72	7,51	0,56	1,52	2,51	5,67	27,87
MaxRec	8,28	7,90	1,48	3,42	5,90	8,62	30,34
ComRec	0,72	1,49	0,00	0,00	0,00	0,56	4,92
ETR	0,07	0,20	0,00	0,00	0,00	0,00	0,74

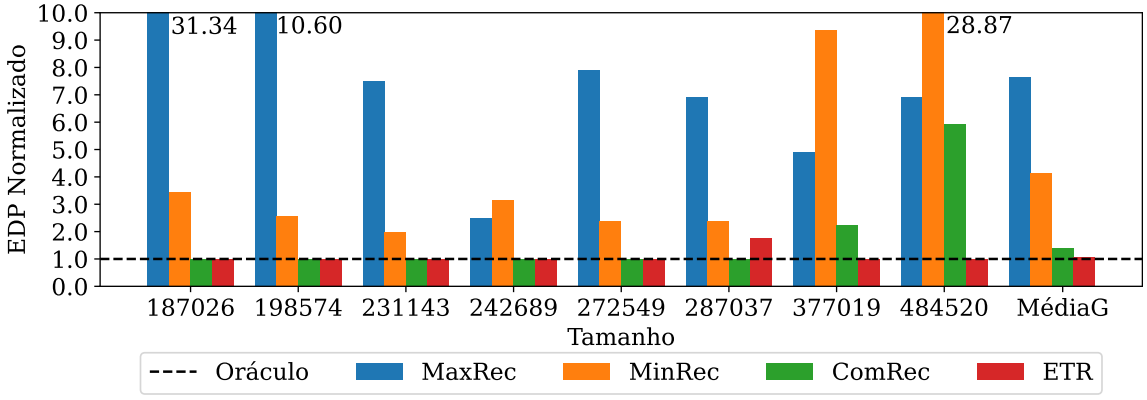
6.3. Desempenho

Esta seção compara o desempenho em termos de EDP da metodologia proposta em relação às demais estratégias avaliadas (MaxRec, MinRec e ComRec). A Figura 2 apresenta o EDP (eixo y) obtido pela execução da aplicação RAxML sobre diferentes entradas (eixo x), comparando a solução predita pela nossa metodologia (ETR) com as demais estratégias, bem como a média geométrica dos resultados (representada por grupos de barras em cores distintas). Os resultados são apresentados para os valores de *bootstrap* (a) 100 e (b) 1000, e estão normalizados em relação ao Oráculo, de modo que valores menores indicam melhor eficiência energética.

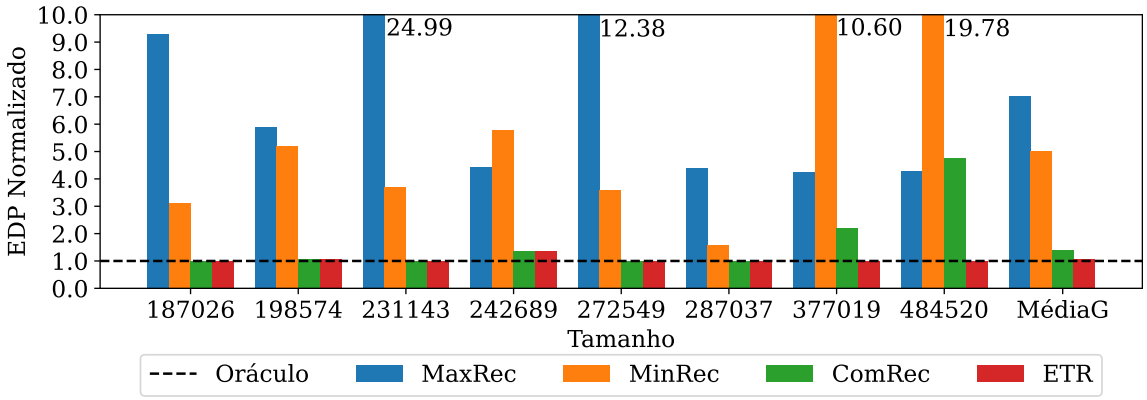
Os resultados indicam que a metodologia proposta alcança os melhores valores de EDP em comparação com todas as demais estratégias, apresentando apenas 7,20% e 4,85% de diferença em relação ao Oráculo para os *bootstrap* 100 e 1000, respectivamente. A única estratégia competitiva foi a ComRec, que coincidiu com o Oráculo em 10 das 16 entradas. No entanto, como ela nem sempre entrega a melhor solução, apresenta diferenças de 38,22% e 40,53% em relação ao Oráculo para os *bootstrap* 100 e 1000. Esses resultados evidenciam que a metodologia proposta combina alta acurácia e consistência, sendo capaz de fornecer soluções próximas ao Oráculo, e superando estratégias alternativas que podem ocasionalmente apresentar desempenho competitivo, mas com maior variabilidade.

7. Conclusão

Neste trabalho, propomos uma metodologia baseada em ETR para recomendar a configuração de (#N,#T) que maximize a eficiência energética em sistemas de PAD. Os resultados experimentais mostram que a abordagem alcança 81,25% de acurácia, fornecendo configurações com EDP apenas 6,02% distante do Oráculo e 33,35% melhor EDP com relação a segunda melhor estratégia considerada. Como trabalhos futuros, planeja-se ampliar os experimentos incluindo um maior número de configurações de (#N, #T) e



(a) *Bootstrap*=100



(b) *Bootstrap*=1000

Figura 2. Comparação dos valores de EDP obtidos na execução das diferentes estratégias, em relação ao Oráculo, considerando (a) *bootstrap*=100 e (b) *Bootstrap*=1000.

outros *benchmarks* representativos, aplicar testes estatísticos formais e evoluir a proposta para um *framework* de suporte à decisão energética, integrado a escalonadores.

Agradecimentos

Agradecemos ao Laboratório Nacional de Computação Científica (LNCC/MCTI) pelo acesso a recursos de PAD do supercomputador SDumont. Esta pesquisa contou com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (processo nº 440360/2022-6) e fundos nacionais portugueses por meio da FCT – Fundação para a Ciência e a Tecnologia, projeto UIDB/05037/2020, DOI 10.54499/UIDB/05037/2020.

Referências

Aaen Springborg, A., Albano, M., and Xavier-de Souza, S. (2023). Automatic energy-efficient job scheduling in HPC: A novel SLURM plugin approach. SC-W '23, page 1831–1838, New York, NY, USA. ACM.

Carastan-Santos, D., Da Costa, G., Poquet, M., Stolf, P., and Trystram, D. (2024). Light-weight prediction for improving energy consumption in HPC platforms. In *Euro-Par*, pages 152–165. Springer.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2024). Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS*, NIPS '22, pages 507–520, Red Hook, NY, USA. ACM, Curran Associates Inc.

Khosravi, A., Sandoval, O. R., Taslimi, M. S., Sahrakorpi, T., Amorim, G., and Garcia Pabon, J. J. (2024). Review of energy efficiency and technological advancements in data center power systems. *Energy and Buildings*, 323:114834.

Krzywaniak, A., Proficz, J., and Czarnul, P. (2018). Analyzing energy/performance trade-offs with power capping for parallel applications on modern multi and many core processors. In *FedCSIS*, pages 339–346. IEEE.

Kumar, R., Khatri, S. K., and Diván, M. J. (2022). Performance analysis of machine learning regression techniques to predict data center power usage efficiency. *SSRG IJETT*, 70(5):328–338.

Kunas, C. A., Rossi, F. D., Luizelli, M. C., Calheiros, R. N., Navaux, P. O. A., and Lorenzon, A. F. (2023). NeurOPar, a neural network-driven edp optimization strategy for parallel workloads. In *SBAC-PAD*, pages 170–180.

Lorenzon, A. F., Beck, A. C. S., Navaux, P. O., and Messer, B. (2025). Energy-efficient gpu allocation and frequency management in exascale computing systems. In *ISC*, pages 1–11. Prometeus GmbH.

Maros, A., Almeida, J., Murai, F., da Silva, A. P., Ardagna, D., and Lattuada, M. (2019). Aprendizado de máquina para previsão do tempo de execução de aplicações Spark. In *SBRC*, pages 197–210, Porto Alegre, RS, Brasil. SBC.

Muralidhar, R., Borovica-Gajic, R., and Buyya, R. (2022). Energy efficient computing systems: Architectures, abstractions and modeling to techniques and standards. *ACM Comput. Surv.*, 54(11s).

Olson, R. D., Assaf, R., Brettin, T., Conrad, N., Cucinell, C., Davis, J. J., Dempsey, D. M., Dickerman, A., Dietrich, E. M., Kenyon, R. W., et al. (2023). Introducing the bacterial and viral bioinformatics resource center (BV-BRC): a resource combining PATRIC, IRD and ViPR. *Nucleic acids research*, 51(D1):D678–D689.

Papadimitriou, G., Chatzidimitriou, A., and Gizopoulos, D. (2019). Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore CPUs. In *HPCA*, pages 133–146. IEEE.

Patel, T., Wagenhäuser, A., Eibel, C., Hönig, T., Zeiser, T., and Tiwari, D. (2020). What does power consumption behavior of hpc jobs reveal? : Demystifying, quantifying, and predicting power consumption characteristics. In *IPDPS*, pages 799–809.

Porto, A. H., Coelho, M., Rocha, H. M., Osthoff, C., Ocaña, K., and Cardoso, D. O. (2026). Assuming the best: Towards a reliable protocol for resource usage prediction for high-performance computing based on machine learning. *FGCS*, 175:108070.

Schwarzrock, J., Rocha, H. M. G. d. A., Lorenzon, A. F., de Souza, S. X., and Beck, A. C. S. (2025). Integration framework for online thread throttling with thread and page mapping on NUMA systems. *JPDC*, page 105145.

Stamatakis, A. (2014). RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313.