Performance and Energy Prediction of OpenMP and CUDA Applications using Machine Learning and Pre-execution Features

Fellipe Queiroz[1], **Luan Siqueira**[1], **Erick Damasceno**[1], **Thiago Rodrigues**[1], **Marcos Amaris** [1]

[1]Universidade Federal do Pará
Faculdade de Engenharia de Computação, Tucuruí - Pará

`{fellipe.queiroz, luan.siqueira, erick.silva, thiago.rodrigues}@tucurui.ufpa.br, amaris@ufpa.br`

Abstract. *This study investigates how machine learning techniques can predict the performance of CUDA applications on GPUs and the energy consumption of OpenMP applications on multi-core machines. Models such as ridge regression and random forest are applied using pre-execution data. The analysis, based on the Rodinia (GPU) and PolyBench (CPU) benchmarks, aims to understand how these ML techniques can predict the performance and energy efficiency of these hardware systems. The results indicate that task optimization can reduce energy consumption without affecting performance.*

Resumo. *Este estudo investiga como técnicas de aprendizado de máquina podem prever o desempenho de aplicações CUDA em GPUs e o consumo de energia de aplicações OpenMP em máquinas multi-core. Modelos como regressão ridge e floresta aleatória são aplicados usando dados de pré-execução. A análise, baseada nos benchmarks Rodinia (GPU) e PolyBench (CPU), busca entender como essas técnicas de ML podem prever o desempenho e a eficiência energética desses hardwares. Os resultados indicam que a otimização de tarefas pode reduzir o consumo de energia sem afetar o desempenho.*

1. Introduction

The rapid expansion of high-performance applications in areas such as artificial intelligence, scientific simulations, and big data analysis has driven the development and optimization of GPU (Graphics Processing Unit) architectures. GPUs are recognized for their massive parallelism capabilities, offering an efficient alternative for executing complex tasks[NVIDIA 2024]. To harness this computational power, Nvidia introduced CUDA in 2006, a platform that enables developers to leverage the GPUs' parallel processing capabilities. However, predicting the performance of these applications remains a challenge due to the variability in factors influencing execution, such as GPU architecture and application-specific characteristics [NVIDIA and CUDA 2015]

Simultaneously, the exponential growth of the digital world, marked by a doubling in the number of internet users and a 25-fold increase in global data traffic since 2010, has made data centers crucial for the efficiency of the global network. In 2022, data centers consumed between 240 to 340 TWh of electricity, with projections reaching 752 TWh by 2030, accounting for 1.3% of global electricity consumption and 0.3% of global CO2 emissions [IEA 2023]. This alarming rise in energy consumption raises significant concerns about the sustainability of these infrastructures. In response, Green Computing has emerged as a vital research area, aiming to make computing processes more sustainable and eco-friendly [Cordeiro et al. 2023]. Innovations such as multi-core processors and OpenMP have shown promise in improving energy efficiency but present optimization challenges.

In this context, understanding current technologies and developing strategies to optimize them is essential. Predicting an application's performance or energy consumption is complex due to the often incomplete and non-deterministic nature of applications. Machine Learning, with its potential to identify complex and nonlinear relationships in data, offers a promising solution.

This study investigates the use of Machine Learning techniques to predict both performance and energy efficiency in multi-core machines. The research operates on two fronts: for CPUs, the focus is on predicting energy consumption during the execution of OpenMP applications, while for GPUs, the study aims to predict the execution time of CUDA applications. Models such as ridge regression, decision tree, and random forest are employed, leveraging pre-execution characteristics and GPU architecture information for prediction. The analysis focuses on data from Rodinia/GPU and PolyBench/OpenMP benchmarks to develop predictive models that optimize performance and energy efficiency, contributing to decision-making in high-performance computing task scheduling.

This document is divided into 6 sections, as follows: Section 2 demonstrates the essential concepts of this research. Section 3 presents some relevant papers for understanding this work. Section 4 presents the methodology. Section 5 explains the results and finally Section 6 presents the conclusions and future work.

2. Theoretical Background

2.1. CUDA for Nvidia GPUs and OpenMP for CPUs Multicore

NVIDIA's GPU architecture consists of processing cores organized into streaming multiprocessors, enabling the parallel execution of thousands of threads and the efficient handling of complex tasks. Components such as registers, load/store units, cache, and clock speed directly influence application performance. CUDA, a parallel computing API developed by NVIDIA, allows developers to harness the parallel processing capabilities of GPUs. It organizes threads into blocks and grids, facilitating a high level of parallelism and computational efficiency [NVIDIA and CUDA 2015].

OpenMP is an API for multi-threaded programming on Unix and Windows NT systems, introduced in 1997. It facilitates the parallelization of applications through simple compiler directives [Penha et al. 2002]. OpenMP employs the fork-join model, where master threads split to execute tasks simultaneously and then synchronize afterward. Despite its standardization and portability across platforms, OpenMP faces challenges in scalability across many cores and complexity in large-scale systems.

2.2. Machine Learning for Performance and Energy Consumption Predictions

Machine learning presents a promising approach for predicting the performance of complex applications. Techniques such as Ridge Regression, Decision Tree, and Random

Forest are employed to create predictive models that estimate outcomes based on specific inputs. These models are trained on pre-execution data, capturing the relationships between code characteristics and expected performance [Susto et al. 2014]. Ridge Regression extends linear regression by incorporating regularization to address multicollinearity and enhance the robustness of estimated coefficients [Hastie et al. 2009]. Decision Tree is a non-parametric technique that constructs a predictive model by dividing the population into branches based on simple decision rules [yan Song 2015]. Random Forest combines multiple decision trees to reduce model variance and improve prediction accuracy [Breiman 2001].

2.3. Linear and Polynomial Regression

The simple Linear Regression model is a basic predictive method that relates a dependent variable Y to an independent variable X through a straight-line representation [Chein 2019]. Although it can suggest a cause-and-effect relationship, it is crucial to note that simple Linear Regression alone does not prove causality. Polynomial Regression is an extension of Linear Regression where the relationship between the independent variable X and the dependent variable Y is modeled as a polynomial of degree n. Unlike the simple linear model, which can only represent linear relationships, Polynomial Regression can capture more complex and curvilinear relationships between variables.

2.4. Performance Metric: Mean Absolute Percentage Error (MAPE)

The Mean Absolute Percentage Error (MAPE) is a widely used metric for evaluating the accuracy of predictions due to its ease of interpretation and scale independence. MAPE calculates the average of the absolute differences between actual and predicted values, expressed as a percentage of the actual values. This allows for an intuitive understanding of the absolute percentage error [Kim and Kim 2020]. The MAPE is defined in Equation 1,

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{1}$$

where n is the number of observations, y_i is the actual value of the i-th observation, and $\hat{y}_i$ is the predicted value of the i-th observation. The interpretation of MAPE values generally follows a scale where smaller numbers indicate more accurate predictions [Lewis 1982].

MAPE	Interpretation
-10%	Highly accurate prediction
10 to 20%	Good prediction
20 to 50%	Reasonable prediction
50% +	Inaccurate prediction

Table 1. Interpretation of MAPE values.

2.5. Green Computing

With the proliferation of computational resources, there has been a growing demand for computational power, leading to increased energy consumption and CO_2 emissions. Green Computing aims to use comethodmputational resources efficiently and in an environmentally responsible manner, reducing energy consumption and CO_2 emissions. Its practices include energy-efficient hardware and software design, sustainable design, waste management, virtualization, green data centers, and cloud computing. Environmental issues arising from increased CO_2 emissions and the financial costs associated with energy consumption have driven studies focused on developing mechanisms and technologies for energy-efficient computing, known as Green Computing [Williams and Curtis 2008].

3. Related Works

Previous studies have demonstrated the effectiveness of machine learning in predicting application performance. For instance, [Amaris et al. 2023] proposed a model based on the Bulk Synchronous Parallel (BSP) paradigm to estimate the execution times of CUDA applications, highlighting its simplicity and ability to handle various GPU configurations. The authors discussed various approaches for predicting CUDA kernel execution times on general-purpose GPUs, noting that while analytical models work well for regular applications, machine learning techniques yield better predictions for irregular ones.

Similarly, [Wang et al. 2019] developed a hybrid framework that combines supervised and unsupervised learning, emphasizing the critical role of feature selection in building precise models. [Mittal and Vetter 2016] further reinforced the importance of integrating application-specific features with machine learning techniques to enhance prediction accuracy, especially in heterogeneous computing environments.

More recent research by [Zhang et al. 2020] showed that deep neural networks could capture complex relationships between variables, significantly improving performance prediction accuracy. Their work suggests that the complexity and non-linearity of HPC applications demand more sophisticated models to achieve reliable predictions.

Saurav et al.[Saurav and Benedict 2021] provided a taxonomy of energy-aware schedulers for scientific workflows in large-scale heterogeneous infrastructures, offering a foundation for designing energy-efficient scheduling techniques. Similarly, Caripán et al.[Caripán Uribe 2022] introduced the 'PowerTester' platform, which utilizes Intel RAPL and Linux 'Perf' to measure and analyze energy consumption and performance in C++ programs. The platform organizes and presents data in graphical formats and CSV files, with its usability validated through experiments and evaluated using the SUS scale, highlighting areas for improvement.

[Benedict et al. 2015] introduced a Random Forest Modeling (RFM) method for predicting the energy consumption of OpenMP applications within compilers. Tested on various applications like NAS benchmarks and matrix multiplication, the RFM approach achieved a prediction accuracy with a Mean Square Error (MSE) of less than 0.699 and an R2 value of 0.998. The study also explores the impacts of energy variation, the number of independent variables, and the proportion of testing data on the performance of the model.

[Shahid et al. 2021] addressed the challenge of energy efficiency in ICT, focusing on energy predictive modeling based on performance monitoring counters (PMCs). It introduces a new theoretical framework that clarifies the role of PMCs in energy consumption and identifies causes of model inaccuracies. The theory provides guidelines for selecting model variables and improving prediction accuracy. Experiments on Intel multicore servers show that using this framework enhances the accuracy of linear regression models from 31.2% to 18% and can achieve up to 80% energy savings compared to existing measurement tools.

In this article, we implement a machine learning model to predict kernel execution time on NVIDIA GPUs and the energy consumption of OpenMP applications. Unlike other works that rely on runtime profiling, our approach extracts data during the compilation phase, providing early-stage predictions for performance and energy efficiency.

4. Methodology

4.1. Test Algorithms

Nine CUDA kernels were utilized, including four for matrix multiplication, two for matrix addition, and one each for dot product, vector addition, and the maximum submatrix problem. The matrix multiplications covered strategies involving both uncoalesced and coalesced accesses to global memory, as well as uncoalesced and coalesced accesses to shared memory. The matrix additions included algorithms with uncoalesced and coalesced accesses to global memory. The vector addition performed parallel computations on GPUs, while the dot product calculated the multiplication of elements from two vectors. The maximum submatrix problem optimized the search for the contiguous submatrix with the largest combined element count.

4.2. Hardware and Software Components

The hardware setup utilized in this study consists of an Intel Core i5-7200U processor with 2 physical cores and 4 logical cores, operating at a base frequency of 2.50 GHz, coupled with 12 GB of DDR4 memory at 2133 MHz. The operating system used was Linux Ubuntu version 22.04, and the GCC compiler version 13.2.0 was employed for all compilation tasks.

NVIDIA GPUs from various architectures, such as Kepler (Compute Capability 3.X) and Maxwell (Compute Capability 5.X), were used. Detailed specifications of each GPU are listed in Table 2, including compute version, memory size, memory bandwidth, L2 cache size, and the number of cores per Streaming Multiprocessor (SM). These specifications are crucial for understanding the impact of hardware characteristics on application performance.

Table 2. GPU hardware specifications

Model	CC/Memory	Bandwidth	L2	Cores/SM
GTX-680	3.0/2 GB	192.2 GB/s	0.5 MB	1536/8
Tesla-K40	3.5/12 GB	276.5 GB/s	1.5 MB	2880/15
Tesla K-20	3.5/4 GB	200 GB/s	1 MB	2496/13
Titan	3.5/6 GB	288.4 GB/s	1.5 MB	2688/14
Q k5200	3.5/8 GB	192.2 GB/s	1 MB	2304/12
Titan X	5.2/12 GB	336.5 GB/s	3 MB	3072/24
GTX-970	5.2/4 GB	224.3 GB/s	1.75 MB	1664/13
GTX-980	5.2/4 GB	224.3 GB/s	2 MB	2048/16

4.3. Dataset

For experiments predicting execution times of CUDA applications with pre-executions characteristics, data collection involved compiling heterogeneous applications across various GPU architectures (3.0, 3.5, and 5.2) using a Python script. Different CUDA thread block sizes (8^2, 16^2, 32^2) were considered to evaluate parallelization. The CUDA Occupancy Calculator was used to analyze GPU occupancy, capturing data such as registers and memory utilization, stored in a CSV file, Figure 3 shows the features used for performance prediction of GPU kernels. The analysis was further enriched with additional data from [Amaris et al. 2016], resulting in 8,459 samples and twelve features, ensuring consistency in data merging.

Table 3. Features used as input for model learning

Feature	Description
compute_version	Compute capability of the architecture
registers	Number of registers
smem	Amount of shared memory
cmem	Amount of constant memory
num_of_cores	Number of cores per GPU
l2	Cache
bandwidth	Memory bandwidth
theoretical_flops	Floating-point operations per second
occupancy	Kernel multiprocessor occupancy
input_size	Input size
duration	Execution time
block_x	Number of threads per block

The applications selected for energy consumption predictions in this research were sourced from PolyBench/ACC, chosen due to their parallelized implementations. PolyBench-ACC is derived from the PolyBench/C benchmark suite and provides implementations using OpenMP, OpenACC, CUDA, OpenCL, and HMPP. The benchmarks used in this study are listed below in Table 4

Benchmark	Benchmark	Benchmark
2mm	3mm	Trmm
Symm	Gemm	Syr2k
Syrk	Gramschmidt	Correlation
Covariance	LU	Cholesky

Table 4. Selected Benchmarks

The benchmarks cover a range of computational tasks: 2mm and 3mm perform matrix multiplications, Trmm tests triangular matrix multiplications, Symm and Gemm assess symmetric and general matrix multiplications, Syr2k and Syrk conduct symmetric rank updates, Gramschmidt implements orthogonalization, Correlation and Covariance calculate correlation and covariance, and LU and Cholesky execute matrix decompositions. The methodology involved using a script to automate the execution of benchmarks, each run 101 times with input sizes ranging from 800 to 4000, incremented by 32, ensuring repeatability and the analysis of different problem sizes.

The script compiled and executed the benchmarks, gradually adjusting the input values, and simultaneously used the JouleIt tool to monitor and collect energy consumption data in CSV files. The collected data were categorized by attributes such asCORE (Processing core), CPU (Central Processing Unit), DRAM (Dynamic Random Access Memory), DURATION (Execution time), UNCORE (Part of the processor that is not the processing core), and INPUT (Values provided to a system for processing)..

Energy metrics were recorded in microjoules and duration in microseconds to capture fine variations. A static code evaluation provided additional data, including INPUTS (number of inputs), 1D (use of one-dimensional arrays), ARRAYS (number of arrays), LOOPS (loops), and mathematical operations (MULT, SOM, SUB, DIV), which help in understanding the complexity and energy efficiency of the applications.

4.4. Data Preprocessing

Initially, we predicted performance in GPU kernels with a correlation analysis among the variables to identify significant relationships. The correlation matrix, presented in Figure 1, shows the presence of multicollinearity, indicated by high correlations between several variables, such as L2 and theoretical_flops (0.91), num_of_cores and bandwidth (0.79), and bandwidth and theoretical_flops (0.85), complicating the determination of individual effects in a regression model.

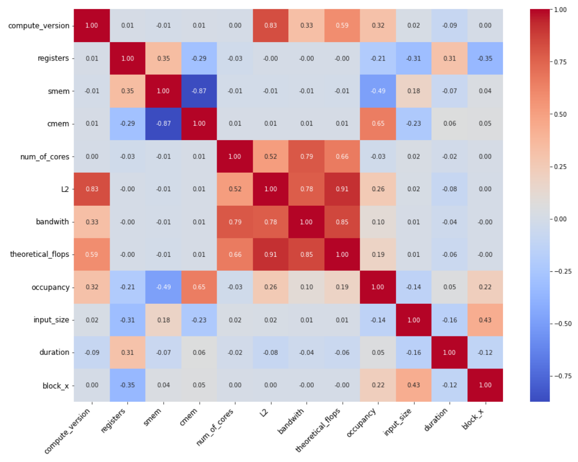

Figure 1. Correlation Matrix

Next, the frequency distribution of the “duration” variable was analyzed to better understand how the execution times of applications were distributed. Figure 2.Left shows that most durations were concentrated at lower values, with a long tail to the right, indicating left-skewness or negative skewness.

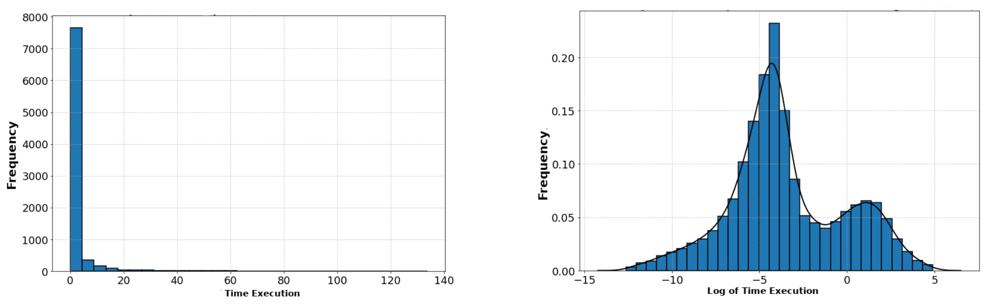

Figure 2. Left: Frequency distribution of the ’Duration’ variable. Right: Frequency distribution after logarithmic transformation of the ’Duration’ variable.

To facilitate machine learning model training and improve performance, a logarithmic transformation was applied to the ”duration” variable. The logarithmic transformation helps reduce skewness in the data, making the distribution more symmetric and closer to a normal distribution, which is desirable for many machine learning techniques. Figure 2.Right shows the distribution of the “duration” variable after logarithmic transformation. We followed the same steps with energy consumption predictions for OpenMP Applications.

When analyzing the energy consumption of the benchmarks OpenMP, Syr2k and Syrk, which perform Linear Algebra, were the highest consumers. Trmm, Symm, 3mm, 2mm, and Gemm, which perform matrix multiplications with some differences in operations, showed similar consumption, reflecting variations in specific operations. Cholesky and LU had comparable consumption as they perform matrix decomposition, while Covariance, Gramschmidt, and Correlation had the lowest consumption, with no direct relationship between them.

The analysis of execution time revealed that energy consumption is not directly linked to execution duration, as shown in Table 5. Syrk and Syr2k had high energy consumption despite short execution times, while Gramschmidt had lower energy consumption but a longer execution time. This suggests that optimizing task distribution in terms of execution time could reduce energy consumption without compromising efficiency.

	3mm	GramSchmidt	Symm	2mm	Covariance	Gemm	Correlation	Syr2k	Syrk	Tmm	LU	Cholesky
Duration (h:m:s)	5:25:08	4:51:12	3:38:18	3:36:41	1:55:22	1:48:15	1:31:21	0:27:38	0:14:22	0:12:25	0:10:06	0:03:20
Consumption (watts)	14.50	13.00	15.25	14.00	12.30	13.50	12.85	17.40	17.00	15.55	14.25	15.25

Table 5. Execution duration and energy consumption of applications

Finally, for the OpenMP applications, we performed a correlation analysis with hierarchical clustering, which highlighted patterns among the applications, as shown in Figure 4.4. Energy consumption relationships were observed between Covariance, Gemm, and Gramschmidt, as well as between Correlation and Symm. This suggests that specific adjustments can be implemented to improve efficiency in similar applications, while isolated ones like Cholesky and LU may require distinct optimization approaches.

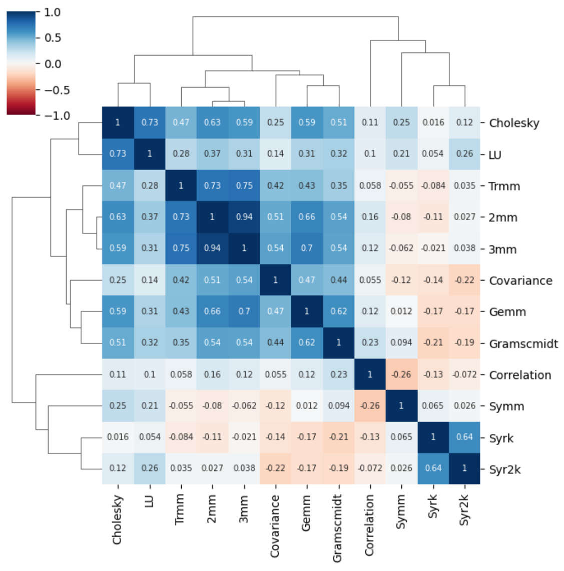

Figure 3. Heatmaps of correlation with hierarchical clustering.

4.5. Model Training

For the development of each model, eight applications were selected as the training set while one application was reserved for the test set. This process was repeated for each of the nine applications. Machine learning models, including *Ridge Regression*, *Decision Tree* and *Random Forest* were used to evaluated the performance predictions in GPU Kernels. All scripts developed during this research are publicly available on the GitHub repository to ensure that the results of this study are clear and replicable. It can be accessed via the following link: GitHub - Project Repository.

Linear and polynomial regression were applied to predict energy consumption, with the goal of understanding current behavior and forecasting future trends. With polynomial regression, we proposed an approach of combining multiple benchmark datasets for training and using a separate dataset for testing, with the polynomial degree varying from 1 to 3. MAPE was used to evaluate the prediction accuracy in this approach. The scripts developed in this research are publicly available on the GitHub repository. The repository can be accessed through the following link: GitHub - Project Repository.

5. Results and Discussions

For experiments about performance prediction of CUDA Kernels using Pre-execution features, we split the dataset into 90% for training and 10% for testing. The effectiveness of each model was assessed using the Mean Absolute Percentage Error (MAPE) metric.

Application	**Random Forest**	**Ridge Regression**	**Decision Tree**
MMGU	5.96%	**0.2%**	3.66%
MMGC	**0.45%**	1.98%	0.94%
MMSU	0.88%	0.86%	**0.51%**
MMSC	4.66%	**1.53%**	5.0%
MAC	**0.4%**	20.58%	0.03%
MAU	**0.03%**	13.07%	0.04%
dotP	**0.17%**	5.48%	0.04%
vAdd	**0.07%**	25.78%	0.03%
MSA	**1.03%**	13.6%	0.09%

Table 6. Comparison of MAPE for Models Across Different Applications

For the experiments predicting energy consumption of OpenMP applications, 11 out of the 12 datasets were used for training, with the remaining dataset reserved for testing. The performance of each model was evaluated using the Mean Absolute Percentage Error (MAPE) metric.

The data presented in the Tables 6 and7 demonstrate the performance of different regression models (Random Forest, Ridge Regression, and Decision Tree for CUDA; and Linear Regression, Polynomial Degree 2, and Polynomial Degree 3 for OpenMP) based on MAPE values. For performance predictions of CUDA kernels, Ridge Regression showed the best performance in most applications, indicating that it is more suitable for this type of data. In the case of energy consumption predictions for OpenMP, Polynomial Regression Degree 2 presented the lowest MAPE values across several benchmarks,

Table 7. Comparison of MAPE values for different regression models

Benchmark	Linear Regression	Polynomial Degree 2	Polynomial Degree 3
2mm	3.04%	**1.86%**	7.55%
3mm	6.37%	**5.71%**	7.06%
Correlation	**8.96%**	46.01%	33.42%
Covariance	62.15%	**15.41%**	15.90%
Cholesky	20.13%	30.26%	**14.92%**
Gemm	8.22%	**2.81%**	8.39%
Syrk	20.30%	**7.61%**	8.33%
Syr2k	16.23%	6.00%	**4.12%**
Symm	46.82%	**39.33%**	30.75%
Trmm	**4.12%**	5.87%	7.26%
Gramschmidt	41.96%	**29.41%**	59.52%
LU	8.11%	9.09%	**4.26%**

suggesting that the relationship between energy consumption and input variables may be better captured by a moderately nonlinear model.

These choices indicate that for execution time predictions, a regularized linear model like Ridge can handle multicollinearity well, while for energy consumption predictions, a second-degree polynomial model balances simplicity and accuracy in capturing the nonlinear relationships in the data.

6. Conclusions and Future Works

This study presented a comprehensive analysis of machine learning models to predict both the performance of CUDA applications and the energy consumption of OpenMP applications on multi-core machines. The results demonstrated that Ridge Regression consistently outperformed other models in predicting execution times for CUDA kernels, showcasing its effectiveness in handling the complex relationships between GPU architecture and performance. On the other hand, Polynomial Regression of degree 2 was the most effective for predicting energy consumption in OpenMP applications, indicating that moderately nonlinear models can capture the inherent complexities of energy behavior more accurately.

The results highlight that energy consumption in applications is not directly correlated with execution time. This suggests that optimizing task distribution, particularly for applications where quick execution is not critical, can enhance energy efficiency without compromising computational performance. Furthermore, the study emphasizes the importance of selecting suitable machine learning models tailored to the specific application type and hardware characteristics to achieve optimal results in high-performance computing environments.

Future work should focus on expanding the dataset to include a wider variety of benchmarks and applications, both for CUDA and OpenMP, to ensure more generalized model performance. Additionally, exploring more advanced machine learning techniques such as deep learning or ensemble models may lead to further improvements in prediction accuracy. Integrating additional features from source code analysis, exploring hardware-specific optimizations, and applying Natural Language Processing (NLP) to low-level code analysis, such as PTX, could also provide new insights into performance and energy efficiency optimization.

Lastly, future research should investigate the impact of operating system configurations and programming practices on energy consumption and performance, potentially uncovering additional avenues for improving the sustainability of high-performance computing systems.

References

[Amaris et al. 2023] Amaris, M., Camargo, R., Cordeiro, D., Goldman, A., and Trystram, D. (2023). Evaluating execution time predictions on gpu kernels using an analytical model and machine learning techniques. *JPDC*, 171:66–78.

[Amaris et al. 2016] Amaris, M., de Camargo, R. Y., Dyab, M., Goldman, A., and Trystram, D. (2016). A comparison of gpu execution time prediction using machine learning and analytical modeling. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 326–333. IEEE.

[Benedict et al. 2015] Benedict, S., Rejitha, R., Gschwandtner, P., Prodan, R., and Fahringer, T. (2015). Energy prediction of openmp applications using random forest modeling approach. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 1251–1260.

[Breiman 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Caripán Uribe 2022] Caripán Uribe, D. (2022). Plataforma para medir el consumo energético de algoritmos. Universidad de Concepción.

[Chein 2019] Chein, F. (2019). *Introdução aos modelos de regressão linear: um passo inicial para compreensão da econometria como uma ferramenta de avaliação de políticas públicas*. ENAP, Brasília.

[Cordeiro et al. 2023] Cordeiro, D., Francesquini, E., Amarís, M., Castro, M., Baldassin, A., and Lima, J. (2023). Green cloud computing: Challenges and opportunities. In *Anais do XIX SBSI*, pages 129–131, Maceió/AL. SBC.

[Hastie et al. 2009] Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd edition.

[IEA 2023] IEA (2023). Tracking clean energy progress. `https://www.iea.org/reports/tracking-clean-energy-progress-2023`. Licença: CC BY 4.0.

[Kim and Kim 2020] Kim, S. and Kim, H. (2020). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 36(3):1115–1127.

[Lewis 1982] Lewis, C. D. (1982). *Industrial and Business Forecasting Methods*. Butterworths.

[Mittal and Vetter 2016] Mittal, S. and Vetter, J. S. (2016). A survey of methods for analyzing and improving gpu energy efficiency. *ACM Computing Surveys (CSUR)*, 49(3):41.

[NVIDIA 2024] NVIDIA (2024). Gpu accelerated solutions for data science. *NVIDIA Newsroom*.

[NVIDIA and CUDA 2015] NVIDIA, C. C. and CUDA, C. (2015). Programming guide, version 7.

[Penha et al. 2002] Penha, D., Corrêa, J., and Martins, C. (2002). Análise comparativa do uso de multi-thread e openmp aplicados a operações de convolução de imagem. In *Anais do III Workshop em Sistemas Computacionais de Alto Desempenho*, pages 118–125, Porto Alegre, RS, Brasil. SBC.

[Saurav and Benedict 2021] Saurav, S. K. and Benedict, S. (2021). A taxonomy and survey on energy-aware scientific workflows scheduling in large-scale heterogeneous architecture. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, pages 820–826.

[Shahid et al. 2021] Shahid, A., Fahad, M., Manumachu, R. R., and Lastovetsky, A. (2021). Energy predictive models of computing: Theory, practical implications and experimental analysis on multicore processors. *IEEE Access*, 9:63149–63172.

[Susto et al. 2014] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. (2014). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820.

[Wang et al. 2019] Wang, X., Huang, K., Knoll, A., and Qian, X. (2019). A hybrid framework for fast and accurate gpu performance estimation through source-level analysis and trace-based simulation. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 201–214. IEEE.

[Williams and Curtis 2008] Williams, J. and Curtis, L. (2008). Green: The new computing coat of arms? *IT Professional Magazine*, 10(1):12.

[yan Song 2015] yan Song, Y. (2015). Decision tree methods: applications for classification and prediction. *International Journal of Data Analysis Techniques and Strategies*, 7(2):228–243.

[Zhang et al. 2020] Zhang, Y., Wang, S., and Chen, G. (2020). Deep learning-based performance prediction for gpu-accelerated applications. *Journal of Parallel and Distributed Computing*, 144(3):12–23.