A Framework and Methodology for Performance Prediction of HPC Workloads

Júlia Orteu
Barcelona Supercomputing Center
Barcelona, Spain
julia.orteu@bsc.es

Marc Clascà
Barcelona Supercomputing Center
Barcelona, Spain
marc.clasca@bsc.es

Elise Jennings
ParTec AG
Germany
elise.jennings@par-tec.com

Jesús Labarta
Universitat Politecnica de Catalunya
& Barcelona Supercomputing Center
Barcelona, Spain
jesus.labarta@bsc.es

Marta Garcia-Gasulla
Barcelona Supercomputing Center
Barcelona, Spain
marta.garcia@bsc.es

ABSTRACT

The presented poster outlines an approach for predicting the performance of High-Performance Computing workloads (HPC). By utilizing data gathered from runtime hardware counters across a range of HPC applications and benchmarks, we develop an artificial intelligence model based on ensemble tree algorithms. Through this approach, we prove that a prediction of the instructions per cycle (IPC) metric of unseen applications is possible based on architectural performance counters that can be obtained easily with already used and convenient performance tools.

CCS CONCEPTS

• **Computing methodologies** → **Parallel computing methodologies**; **Artificial intelligence**; *Classification and regression trees.*

KEYWORDS

HPC Workloads, Performance Prediction, Runtime Hardware Counters, Instructions per Cycle (IPC), Performance Tools, Parallel Applications, Regression trees, ML & AI

ACM Reference Format:
Júlia Orteu, Marc Clascà, Elise Jennings, Jesús Labarta, and Marta Garcia-Gasulla. 2024. A Framework and Methodology for Performance Prediction of HPC Workloads. In *Proceedings of 11th ACM Celebration of Women in Computing (womENcourage™ 2024).* ACM, New York, NY, USA, 2 pages.

1 EXTENDED POSTER ABSTRACT

The research line focuses on exploring the possibility of predicting the performance, and ultimately the execution time, of known workloads in HPC machines prior to their execution.

womENcourage™ 2024, June 26-2, 2024, Madrid, Spain

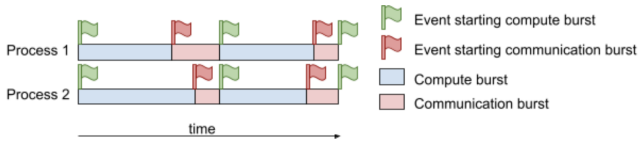

Figure 1: Extrae and Paraver generalization of burst

Our approach distinguishes itself from existing research by its focus on the granularity of training and prediction. Using performance analysis tools Extrae and Paraver [3] [4], developed in Barcelona Supercomputing Center (BSC), we extract data at the computational burst level and target the IPC metric of every burst as a performance measure. A burst is defined as the time interval of active operation between two successive events in a process.

In Figure 1, we provide a visual example of a timeline from a execution representation of an application running two processes. The activities within these processes have been categorized into two types: communication bursts, indicated in red, and computation bursts, represented in blue. These categorizations are derived from MPI event markers, with green flags initiating computation and red flags marking the start of communication.

In our study, we concentrate on computation bursts, which we refer to as *useful bursts*. These bursts are periods where the application is actively engaged in data processing and executing instructions. These individual useful bursts serve as the input samples to the model, as one entry of our dataset, highlighting that this approach provides a very precise performance prediction of a very specific application's part.

1.1 The data

1.1.1 Features and workload characterization. To facilitate our study, we consider a set of Performance Application Programming Interface (PAPI) counters [1] as the foundational data. These counters include the total number of instructions completed (N), the total cycles seen by thread (Cyc), memory load instructions (N_{LD}), memory store instructions (N_{SR}), branch instructions (N_{BR}), and total cache miss events at both L1 and L3 levels (miss_{L1}, miss_{L3}). We normalize the data in each burst by using the ratio of instructions and cache misses instead of absolute counters values. This allows us to compare any individual burst from any part of a trace and from

any application. From the counters, we characterize the variables using the following computations:

- Instruction mixes, which are ratios of specific instruction types to the total number of instructions, such as $r_{LD} = \frac{N_{LD}}{N}$ for loads, $r_{SR} = \frac{N_{SR}}{N}$ for stores, and $r_{BR} = \frac{N_{BR}}{N}$ for branches.
- Cache miss rates, calculated as $r_{L1} = \frac{\text{miss}_{L1}}{N_{LD}+N_{SR}}$ for the L1 cache and $r_{L3} = \frac{\text{miss}_{L3}}{N_{LD}+N_{SR}}$ for the L3 cache, which are indicators of memory access efficiency.
- The average node concurrency during core execution, through the integral of the Parallelism function over the time from T_{begin} to T_{end}: $\int_{T_{\text{begin}}}^{T_{\text{end}}} Par(t)dt$.
- IPC, which is the target performance metric, given by the equation $IPC = \frac{N}{Cyc}$.

1.1.2 Data sources. The process involves the selection of a specific set of benchmarks and kernels to extract data and adapt it for the purpose of training the models. Additionally, a separate set of applications has been chosen for evaluating the performance of the trained models (testing). The selection of kernels and benchmarks for the training set is a pivotal decision that facilitate the representation of the burst space, enabling the generalization to new applications. For each application selected for the training set, we've varied the problem sizes to capture a comprehensive dataset. The nature of the variation depends on the application's characteristics—it could be the size of an array, the granularity of a mesh, or the complexity of inputs.

Additionally, we've scaled the computational workload by altering the number of processes within a single node for each variant of problem size. This methodical approach allows us to construct a training dataset that covers a wide range of scenarios.

1.1.3 Data extraction Framework. The data extraction process for each application, both for training and testing datasets,begins with the execution of a known application on a known machine. In this case, we have used the *MareNostrum 4* supercomputer's architecture for reference. We obtain the useful bursts from a program execution using the BSC tools, which are then processed into a features format.

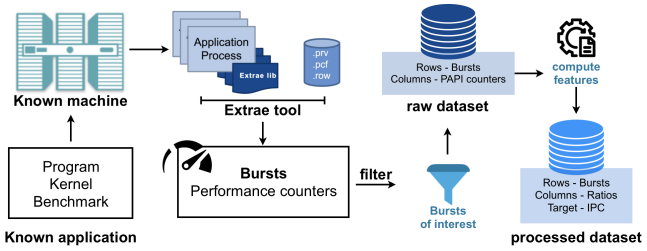

Figure 2: Flowchart of the data extraction part of the training framework

1.2 AI Model

The study investigates a range of diverse machine learning algorithms with the aim of training an effective predictive model. In this exploration, we have employed a 10-fold cross-validation method on our training dataset to evaluate the performance and compatibility of these algorithms with our type of data.

The empirical results highlight a clear advantage of using Boosting Ensembles that rely on decision trees, leading to a focus on *XGBoost method* [2] . This also matches with previous research conclusions [5]. To ensure a fair assessment of the models, we've developed a method to inject these predictions into *Paraver* traces and we've devised specific error metrics into the trace tailored to evaluate the performance outcomes of each model. This allows for a more precise analysis of how well the models predict application performance.

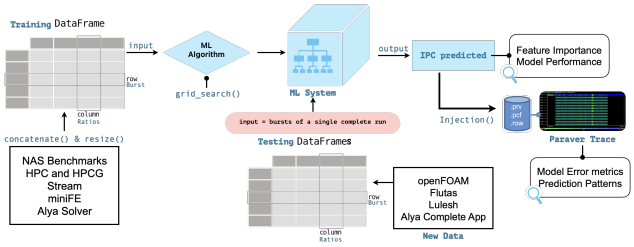

Figure 3: Flow chart of the training part of the framework.

The flow chart diagram in Figure 3 outlines the systematic process for developing predictive models for IPC estimation.

1.3 Summary

The poster shows the main findings and discusses our exploration of this topic. We present a method of data collection and preprocessing based on low-effort program instrumentation and automatic tools, as it is well known in the literature that being able to collect data automatically and building the model effortlessly is critical to end up with valuable and convenient training and prediction workflow. We also studied how changing the feature set, the training data size or the machine learning algorithm affects the accuracy.

2 ACKNOWLEDGMENT

This work bas been published in proceedings of the 11th ACM Celebration of Women in Computing (womENcourage™ 2024).

REFERENCES

[1] Shirley Browne, Christine Deane, George Ho, and Philip Mucci. 1999. PAPI: A Portable Interface to Hardware Performance Counters.
[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785
[3] Vincent Pillet, Jesús Labarta, Toni Cortes, and Sergi Girona. 1995. Paraver: A tool to visualize and analyze parallel code. In *Proceedings of WoTUG-18: transputer and occam developments*, Vol. 44. 17–31.
[4] Harald Servat, Germán Llort, Kevin Huck, Judit Giménez, and Jesús Labarta. 2013. Framework for a productive performance optimization. *Parallel Comput.* 39, 8 (2013), 336–353.
[5] Jingwei Sun, Guangzhong Sun, Shiyan Zhan, Jiepeng Zhang, and Yong Chen. 2020. Automated Performance Modeling of HPC Applications Using Machine Learning. *IEEE Trans. Comput.* 69, 5 (2020), 749–763. https://doi.org/10.1109/TC.2020.2964767