Unified Performance Modeling Stack for Distributed GPU Applications: Complementing Analytical Insights with Machine Learning

Urvij Saroliya
urvij.saroliya@tum.de
Technical University of Munich
Munich, Germany

Eishi Arima (Advisor)
eishi.arima@tum.de
Technical University of Munich
Munich, Germany

Martin Kronbichler (Advisor)
martin.kronbichler@rub.de
Ruhr University Bochum
Bochum, Germany

ABSTRACT

Modern HPC applications increasingly use GPUs to solve larger problems with higher accuracy and speed. However, committing resources to these large-scale systems is often costly and time-consuming. Hence, performance modeling enables developers to estimate runtime, analyze scalability, and identify resource bottlenecks in advance. In this work, we propose a unified software ecosystem for end-to-end performance modeling of distributed GPU applications. To this end, we propose a combination of analytical and machine learning based modeling methodology, and design a comprehensive software stack to combine the various components for implementing such an approach. We validate the proposed framework using two real-life applications and provide performance estimations for the GPU kernel and inter-GPU MPI communications.

CCS CONCEPTS

• **General and reference** → **Performance**; • **Networks** → **Network performance modeling**.

KEYWORDS

Performance Modeling, GPU Computing, MPI Communication, Machine Learning

1 INTRODUCTION

Performance modeling is a structured mathematical way to estimate the performance of an application on a given computing platform. It can enable application developers to get an estimation for application runtime, identify scalability issues, and perform guided application tuning, without committing compute resources on a large scale. To this end, a performance model must address the following concerns: (i) *prediction accuracy:* model should guide the performance optimizations in the correct direction, (ii) *interpretability of results:* it should help developers identify resource bottlenecks and help in planning application tuning, and (iii) *portability:* in an increasingly heterogeneous and diverse computing scenario, it should be possible to implement and extend the modeling approach to various target platforms. Portability becomes even more crucial for GPUs, where GPUs from different architectures across vendors are often employed in modern HPC systems. In this work, we aim to address these three challenges and propose a comprehensive methodology to implement performance modeling as a unified software ecosystem.

Various performance models have been proposed using both analytical and empirical methods. Analytical performance models

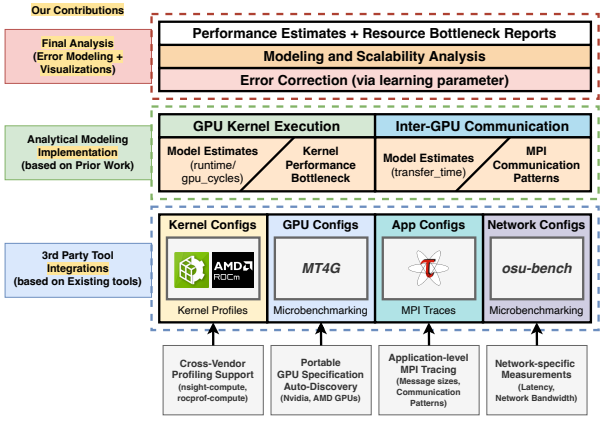

Figure 1: Unified Software Stack Design for Performance Modeling of Distributed GPU Applications

are often based on an abstract machine model of the underlying architecture and propose simplified equations to estimate the performance. Such approaches lead to interpretable results where application developers can easily identify key resource bottlenecks. However, for building an accurate analytical model, in-depth architectural details have to be considered, and hence, these approaches often struggle with prediction accuracy. In contrast, machine learning uses empirical data to learn from the performance data and provide accurate estimates. Nevertheless, ML-based approaches fail to deliver clear application insights necessary for targeted application tuning. Combining the benefits of analytical and ML-based models, we propose a hybrid approach, where we get initial estimates and application analysis using analytical methods and improve estimation accuracy using an ML-based error correction model.

2 UNIFIED SOFTWARE STACK DESIGN

As shown in Figure 1, we propose a software stack where we aim to integrate various components to perform performance modeling of a distributed GPU application. A performance model requires application and system parameters; hence, our stack's lowest layer consists of 3rd-party tool integrations. For gathering application parameters, we use GPU kernel profiling tools such as *Nvidia Nsight Compute* [2] or *AMD ROCProfiler* [6], and *Tau Performance Tool* [7] for MPI communications. On the other hand, we perform microbenchmarking to collect system parameters. We integrate

MT4G [9] to gather GPU specifications for Nvidia/AMD GPUs. Lastly, we use *OSU Micro-Benchmarks 7.5.1* [8] for network measurements.

We implement an analytical performance modeling layer on top of the tool integration layer. For GPUs, we extend and implement the GPU warp parallelism model [5], where the model specifically targets warp parallelism characteristics of GPU kernel execution. The original model [5] does not consider caches; hence, we extend the model to filter cache hits to use the appropriate model parameter values across the memory hierarchy. This model has been particularly selected for its deeper architectural considerations for GPUs. Next, for the MPI communication modeling, we utilize K-model [1], a variant of the well-known postal model for MPI communications. This model explicitly differentiates between intra/inter-node transfers, which is crucial for a multi-GPU scenario.

Lastly, we build our top-most layer for final analysis. Here, we perform necessary error corrections to the performance estimations from the underlying analytical models. To this end, we learn separate ML-based models for kernel and MPI performance corrections. We use a Gradient Boosting Regression technique to learn the error correction terms for both analytical models.

3 RESULTS

We analyzed two real-life applications – (1) Helmholtz equation solver with matrix-free implementation of high-order finite element discretizations using deal.II Library [3] of CFD models (using up to 2 nodes of 8x Nvidia A100 GPUs and 1 AMD Instinct MI210 GPU), and (2) Gromacs application [4] from Molecular Dynamics (using 1 NVIDIA A100 GPU). The proposed workflow has achieved estimation errors below 5% for GPU kernels (for adequate problem sizes) and below 9% for inter-GPU communications. We aim to conduct extensive evaluations by (1) evaluating other HPC and AI workloads, and (2) comparing the methodology against other SOTA modeling methods.

ACKNOWLEDGMENTS

This work was conducted under the doctoral supervision of Professor Martin Schulz, TU Munich. It has been supported by the German Federal Ministry of Research, Technology, and Space (BMFTR) through the SCALEXA initiative and the PDExa project (16ME0641). Special thanks extended to the Erlangen National High Performance Computing Center (NHR@FAU, Germany) and Leibniz Supercomputing Centre (LRZ, Germany), for providing access to the compute resources.

REFERENCES

[1] Jaemin Choi, David F Richards, Laxmikant V Kale, and Abhinav Bhatele. 2020. End-to-end performance modeling of distributed GPU applications. In *Proceedings of the 34th ACM International Conference on Supercomputing*. 1–12.

[2] NVIDIA Corporation. 2025. *Nsight Compute Documentation - NsightCompute 12.9 documentation 2025*. NVIDIA Corporation. https://docs.nvidia.com/nsight-compute/ Accessed: 2025-08-01.

[3] deal.II. 2025. *The deal.II Finite Element Library*. https://dealii.org/ Accessed: 2025-09-28.

[4] Gromacs. 2025. *Welcome to GROMACS*. https://www.gromacs.org/ Accessed: 2025-09-28.

[5] Sunpyo Hong and Hyesoon Kim. 2009. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. In *Proceedings of the 36th annual international symposium on Computer architecture*. 152–163.

[6] Xiaomin Lu, Cole Ramos, Fei Zheng, Karl W. Schulz, Jose Santos, Keith Lowery, Nicholas Curtis, and Cristian Di Pietrantonio. 2025. *ROCm/rocprofiler-compute: v3.1.0 (12 February 2025)*. https://doi.org/10.5281/zenodo.7314631

[7] Sameer S Shende et al. 2006. The TAU parallel performance system. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 287–311.

[8] Ohio State University. 2025. MVAPICH :: Benchmarks – OSU Micro-Benchmarks 7.5.1. https://mvapich.cse.ohio-state.edu/benchmarks/. Online; accessed September 30, 2025.

[9] Stepan Vanecek, Manuel Walter Mußbacher, Dominik Größler, Urvij Saroliya, and Martin Schulz. 2025. MT4G: A Tool for Reliable Auto-Discovery of NVIDIA and AMD GPU Compute and Memory Topologies. In *Proceedings of the the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25)*.