



Unmasking Performance Variability in GPU Codes on Supercomputers



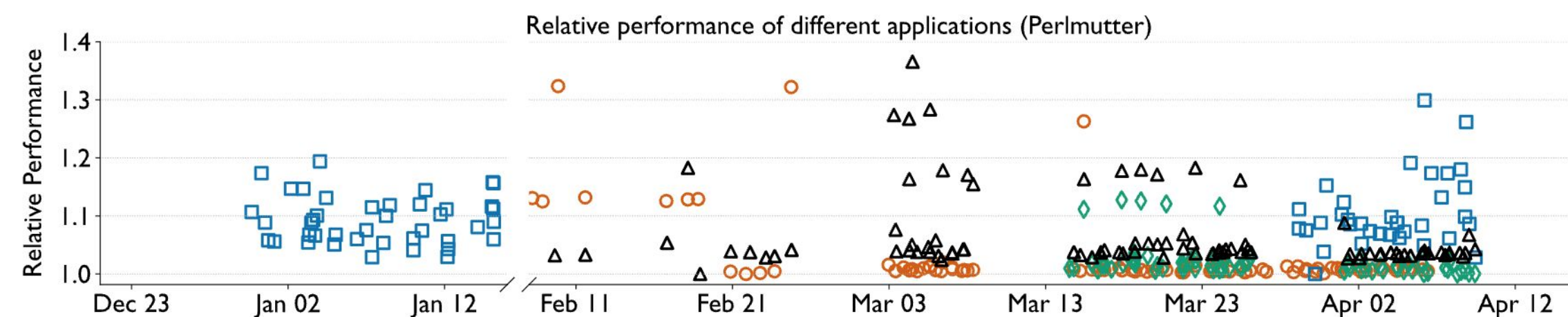
Cunyang Wei, Keshav Pradeep, Abhinav Bhatele
Department of Computer Science, University of Maryland

Abstract

Performance variability is often a critical issue on GPU-accelerated systems, undermining efficiency and reproducibility. Since large-scale investigations of performance variability on GPU clusters are lacking, we set up a longitudinal experiment on Perlmutter and Frontier. We benchmark representative HPC and AI applications and collect detailed performance data to assess the impact of compute variability, allocated node topology, and network conditions on overall runtime. We also use a ML-based approach to identify potential correlations between these factors and to forecast the execution time. Our analysis identifies network performance as the dominant source of runtime variability. These findings provide crucial insights that can inform the development of future mitigation strategies.

What is Performance Variability?

- **Performance Variability:** fluctuations in application runtime across repeated executions under seemingly identical inputs, environments, and system conditions.
- Sources of variability: HW defects, network contention, job placement, OS jitter.
- Prior studies focused on CPU-based systems, leaving the impact of new communication patterns on variability in GPU-based supercomputers unexplored.

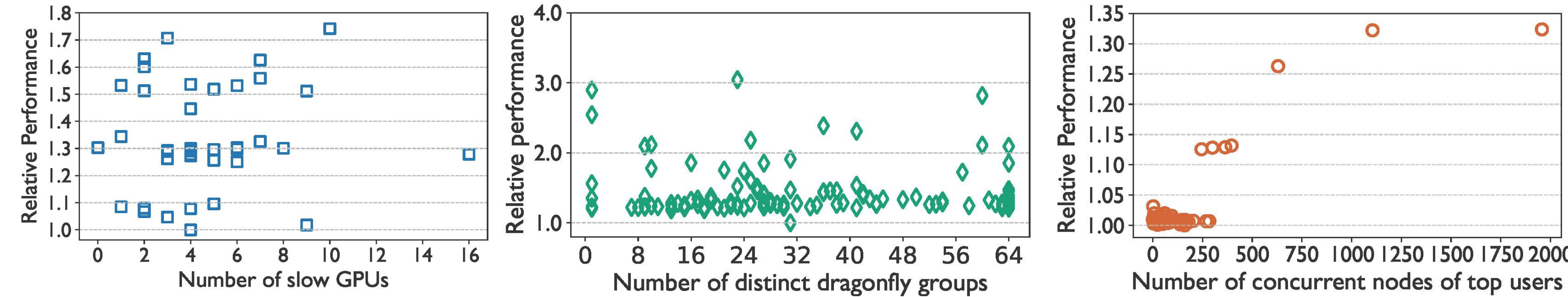


Methodology for Measuring Variability

- We measure performance of two HPC apps (AMG and MILC) and two AI apps (DeepCAM and nanoGPT).
- We collect the following features to analyze performance variability: **mpiP** for profiling HPC apps, **PyTorch profiler** for AI workloads, **SLURM's sacct logs**, **Cassini NIC hardware counters**, and performance data from **GEMM** and **All-Reduce microbenchmarks**.
- Platforms used (network – HPE Slingshot 11):

	NERSC Perlmutter	OLCF Frontier
GPU model	NVIDIA A100 GPU	AMD MI250X GPU
CPU model	AMD EPYC 7763 Milan CPU	AMD EPYC 7713 Trento CPU
GPUs/GCDs per node	4	8

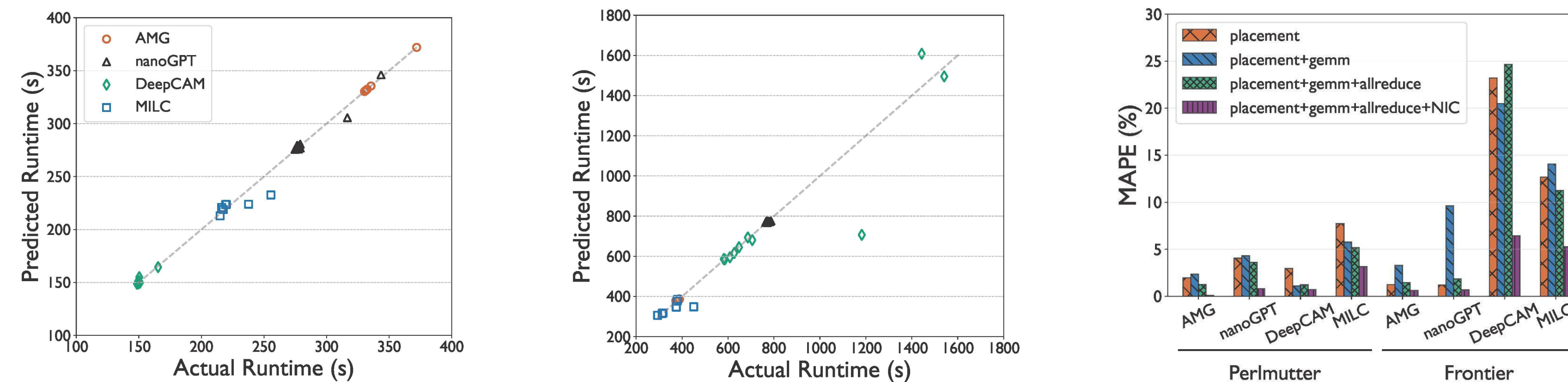
Analysis of the Gathered Data



- **Left:** X-axis = Number of GPUs in job allocation which fall in slowest 1% of system's GPUs. Our analysis reveals the quantity of “slow” GPUs has no impact on performance. **Middle:** On both systems, job allocation across more Dragonfly groups does not degrade app performance. **Right:** Top Users: Users whose allocated number of nodes correlates with our application's performance variation and who concurrently request more than 32 nodes. On both systems, when concurrent node allocation by *Top Users* crosses a threshold, app runtime consistently increases.

ML-based Analysis and Performance Prediction

- Methodology: We train an **XGBoost** model to predict runtime. We use 90% of our runs for training data, reserving 10% of testing.
 - Summary of performance dataset features [name (feature count per run)]: Application Name (1), Placement (1), GEMM (3), MPI Allreduce (11), NCCL Allreduce (8), NIC Counters (29*3).
- Evaluation: We use Mean Absolute Percentage Error (MAPE)

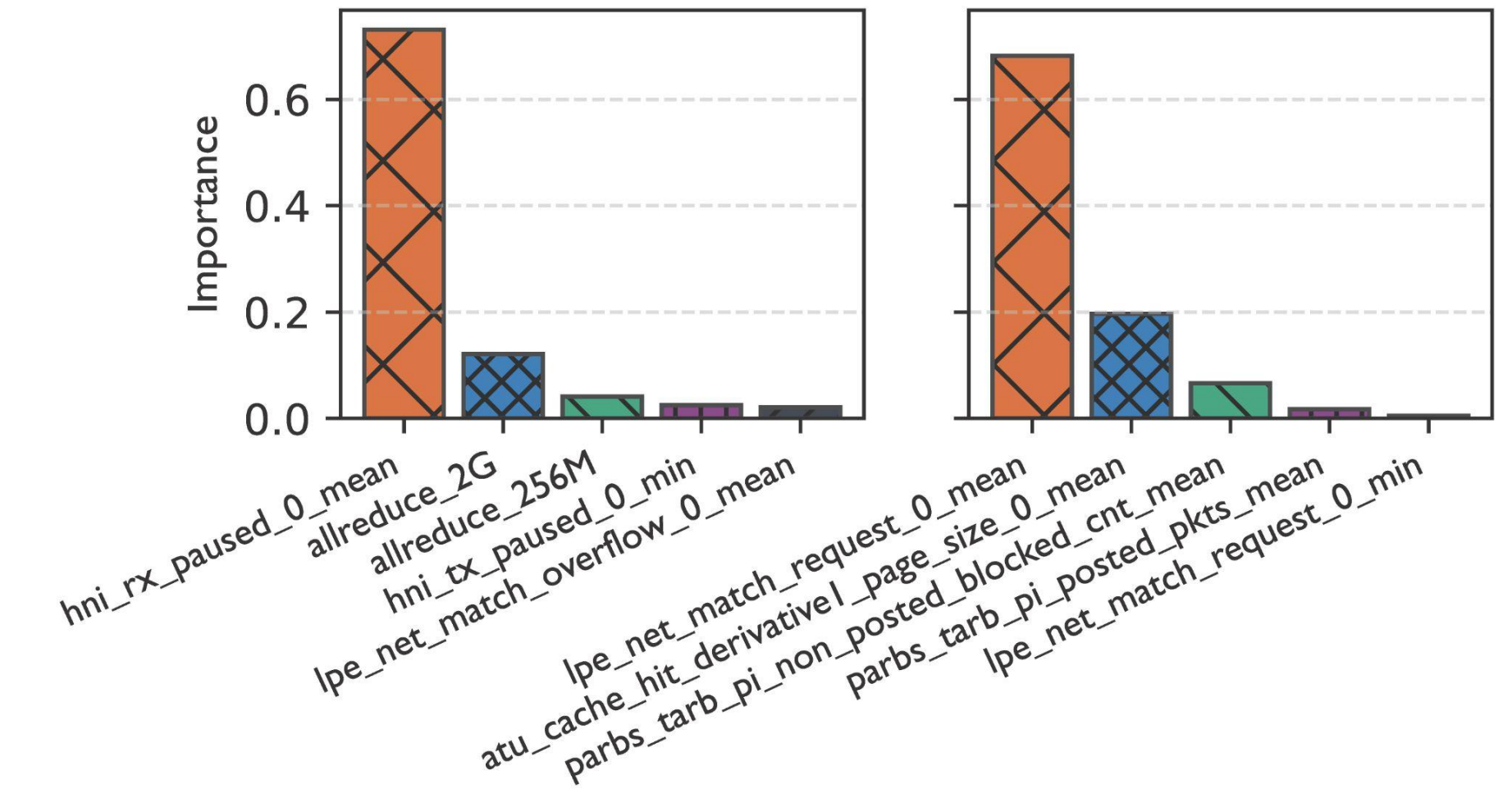


- **Left & middle:** Actual vs predicted runtime using placement, GEMM, Allreduce, and NIC counters features (Perlmutter & Frontier).
- **Right:** When NIC counter features are included in input, MAPE decreases significantly, especially for apps with more variation (e.g. DeepCAM on Frontier). This highlights the importance of NIC counters for explaining variability.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2047120. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy (DOE) Office of Science User Facility, operated under Contract No. DE-AC02-05CH11231 using NERSC award DDR-ERCAP0034262, and that of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. DOE under Contract No. DE-AC05-00OR22725.

Top 5 feature importances: Perlmutter (left) vs. Frontier (right)



- **Feature importances based on XGBoost models. Left:** On Perlmutter, most important feature is hni_rx_paused_0_mean (num cycles where recv path is paused, suggesting network pushing data quicker than NIC can read) **Right:** On Frontier, most important feature is lpe_net_match_request_0_mean (num requests matched on software endpoints).

Conclusion

- Network conditions - not GPU variability or job placement - are the primary drivers of runtime variability in large-scale GPU workloads.
- A small subset of users running communication-heavy jobs account for most of the observed performance degradation.
- Our ML model accurately predicts runtime variability, even with limited training data per application.

Recommendations for Future Efforts

- “Top User” jobs should run on isolated nodes to prevent their communication patterns from impacting network performance for others.
- Future designs need not *over-engineer* the topology for increased network hops. The dragonfly topology is very robust, essentially neutralizing the incurred penalties from node allocations being spread out.
- The application of ML models (such as XGBoost) for system-wide monitoring can predict runtime variability with high accuracy using features such as NIC and network counters. System administrators can use these predictions to detect early signs of congestion and allow users to delay or cancel their workloads.

References

- [1] J. Kim, V. J. Dally, S. Scott, and D. Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In 2008 International Symposium on Computer Architecture.
- [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).
- [3] Abhinav Bhatele, Kathryn Mohror, Steven H. Langer, and Katherine E. Isaacs. 2013. There goes the neighborhood: performance degradation due to nearby jobs. In Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '13).
- [4] Daniel Nichols, Alexander Movsesyan, Jae-Seung Yeom, Daniel Milroy, Tapasya Padki, Abhik Sarkar, and Abhinav Bhatele. 2024. Predicting Cross-Architecture Performance of Parallel Programs. In Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS '24).