Scalable Performance Diagnostics on Leadership Computing Platforms

Dragana Grbic
Department of Computer Science
Rice University
Houston, TX, USA
dg76@rice.edu

Summary

Exascale supercomputers – Frontier [18, 19] at Oak Ridge National Laboratory, Aurora [4, 5] at Argonne National Laboratory, and El Capitan [17] at Lawrence Livermore National Laboratory – provide cutting-edge infrastructure for executing diverse heterogeneous workloads. Users can execute applications across thousands of compute nodes, each spawning numerous MPI ranks, CPU threads, and/or GPU streams. Analyzing performance of these applications remains a critical challenge, as traditional tools struggle with the immense scale and heterogeneity of exascale workloads.

HPCToolkit [1, 22] performance suite was developed at Rice University to enable efficient performance measurement at scale. It employs statistical sampling of hardware counters with call stack unwinding, enabling fine-grained, instruction-level measurement of CPU and GPU code across thousands of compute nodes. The suite includes a GUI, `hpcviewer` [2], for performance presentation; however, visual inspection of exascale measurements is tedious and time-consuming, with no support for automation or integration with external tools. The recently introduced `hpcanalysis` [13] framework was designed to enable scalable, efficient, and interactive analysis of large-scale performance measurements.

In this poster, we refine `hpcanalysis` for more efficient analysis of large-scale performance measurements. We restructured it as a multi-layered infrastructure: a low-level `C++` engine for high-throughput ingestion; a middle-level API for extracting measurements via specialized queries and processing them on specialized hardware; and a high-level Python API for automated and interactive analysis. This decouples low-level data access, middle-level query expressions, and high-level analytical tasks, optimizing each for maximum scalability and efficiency. We also integrated a novel algorithm that scans the summary profile to detect critical code regions and bottlenecks, then propagates the extracted indices across all parallel profiles via indexing, binary search, and internal parallelism – transforming `hpcanalysis` into a tool capable of processing entire exascale executions with high fidelity.

Many widely adopted tools, including Hatchet [11] and Thicket [12], use Pandas [10] to analyze performance profiles. While Pandas provides vectorized operations via `numpy` [14], it is limited to single-threaded CPU execution, hindering scalable processing of exascale measurements. We integrated a novel `HpcDataFrame` abstraction into `hpcanalysis` for vendor-neutral, GPU-accelerated analysis across NVIDIA, AMD, and Intel GPUs. It dynamically selects a backend based on detected hardware: `cuDF` [20] on NVIDIA GPUs, `hipDF` [3] on AMD GPUs, and a custom SYCL wrapper built atop `dpnp` [15] on Intel GPUs. We benchmarked `HpcDataFrame` on Polaris [6, 7] (NVIDIA A100 40GB GPUs), Frontier (AMD Instinct MI250X GPUs), and Aurora (Intel Max 1550 GPUs), demonstrating speedups of up to 314× over a CPU-based Pandas backend on execution traces across 100K MPI ranks.

We evaluated our infrastructure on a massive-scale execution of the AMG [16] benchmark on Aurora across 1,000 compute nodes with 100 MPI ranks per node, producing 100K parallel profiles each with up to 97,833 distinct calling contexts. `hpcanalysis` pruned the summary profile to 110 critical bottlenecks, propagated their indices across all 100K MPI ranks, and ingested the full dataset in 9.69 seconds. Using queries `"rank"`, `"function(MPI_*)"`, and `"cputime:prop (i)"`, we examined load balance for major MPI bottlenecks across 100K ranks. Clustering ranks by performance similarity revealed a critical instance of `MPI_Allreduce` within the `hypre_GMRESSetup` → `hypre_BoomerAMGSetup` → `hypre_ParCSRMatrixSetNumNonzeros_core` call chain, exhibiting significant imbalance: two exclusive groups of 798 and 202 compute nodes, averaging 3.12s and 5.19s respectively. Extending `hpcanalysis` with a workflow that maps logical node IDs to their physical location within Aurora's Slingshot [8] interconnect, we determined that the affected compute nodes spanned 22 distinct racks, with entire chassis affected in some and only a few in others. This demonstrates `hpcanalysis`' ability to detect critical bottlenecks and localize them to physical hardware at massive scale.

Finally, we integrated `hpcanalysis` with Thicket to enable advanced trace analysis. Thicket provides a two-dimensional model for cross-experiment analysis of performance profiles; we extended it with a third dimension, producing a model (events × traces × iterations) that examines iterative patterns within and across execution traces. This model re-materializes iterative behavior typically lost in aggregate profiles and lets users examine iterative patterns using a few lines of Python – by fixing the trace or iteration index. With this model, we predicted a 32.28% speedup under ideal GPU load balancing for a GAMESS [9] workload on Frontier.

Collectively, our contributions provide a scalable and extensible substrate for performance diagnostics at the absolute limit of modern supercomputing. `hpcanalysis` is an open-source project available at [21].

Acknowledgments

This research was supported in part by UT-Battelle, LLC subcontract CW54422 (DOE Prime DE-AC05-00OR22275), LLNL subcontracts B665301 and B670681 (DOE Prime DE-AC52-07NA27344), ANL subcontract 4F-60094 (DOE Prime DE-AC02-06CHII357), and a contract from TotalEnergies E&P Research & Technology USA, LLC.

This research used resources of the Oak Ridge Leadership Computing Facility, a DOE Office of Science User Facility at Oak

Ridge National Laboratory, supported under Contract DE-AC05-00OR22725. This research also used resources of the Argonne Leadership Computing Facility, a DOE Office of Science User Facility at Argonne National Laboratory, based on research supported by the U.S. DOE Office of Science–Advanced Scientific Computing Research Program, supported under Contract DE-AC02-06CH11357.

References

[1] Laksono Adhianto, Jonathon Anderson, Robert Matthew Barnett, Dragana Grbic, Vladimir Indic, Mark Krentel, Yumeng Liu, Srđan Milaković, Wileam Phan, and John Mellor-Crummey. 2024. Refining HPCToolkit for application performance analysis at exascale. *The International Journal of High Performance Computing Applications* 38, 6 (2024), 612–632. https://www.doi.org/10.1177/10943420241277839

[2] Laksono Adhianto, Sinchan Banerjee, Mike Fagan, Mark Krentel, Gabriel Marin, John Mellor-Crummey, and Nathan R Tallent. 2010. HPCToolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience* 22, 6 (2010), 685–701. https://www.doi.org/10.1002/cpe.1553

[3] AMD ROCm Development Team. 2026. hipDF: HIP-based DataFrames for AMD GPUs. Accessed: August 2026. https://rocm.docs.amd.com/projects/hipDF/en/latest

[4] Argonne Leadership Computing Facility. 2026. Aurora. Accessed: August 2026. https://www.alcf.anl.gov/aurora

[5] Argonne Leadership Computing Facility. 2026. Aurora User Guide. Accessed: August 2026. https://docs.alcf.anl.gov/aurora/getting-started-on-aurora

[6] Argonne Leadership Computing Facility. 2026. Polaris. Accessed: August 2026. https://www.alcf.anl.gov/polaris

[7] Argonne Leadership Computing Facility. 2026. Polaris User Guide. Accessed: August 2026. https://docs.alcf.anl.gov/polaris/getting-started

[8] Argonne Leadership Computing Facility. 2026. Running Jobs on Aurora. Accessed: August 2026. https://docs.alcf.anl.gov/aurora/running-jobs-aurora

[9] Giuseppe MJ Barca, Colleen Bertoni, Laura Carrington, Dipayan Datta, Nuwan De Silva, J Emiliano Deustua, Dmitri G Fedorov, Jeffrey R Gour, Anastasia O Gunina, Emilie Guidez, et al. 2020. Recent developments in the general atomic and molecular electronic structure system. *The Journal of chemical physics* 152, 15 (2020). https://www.doi.org/10.1063/5.0005188

[10] Joey Bernard. 2016. Python data analysis with pandas. In *Python recipes handbook: A problem-solution approach*. Springer, 37–48. https://www.doi.org/10.1007/978-1-4842-0241-8_5

[11] Abhinav Bhatele, Stephanie Brink, and Todd Gamblin. 2019. Hatchet: Pruning the overgrowth in parallel profiles. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–21. https://www.doi.org/10.1145/3295500.3356219

[12] Stephanie Brink, Michael McKinsey, David Boehme, Connor Scully-Allison, Ian Lumsden, Daryl Hawkins, Treece Burgess, Vanessa Lama, Jakob Lüttgau, Katherine E Isaacs, et al. 2023. Thicket: seeing the performance experiment forest for the individual run trees. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*. 281–293. https://www.doi.org/10.1145/3588195.3592989

[13] Dragana Grbic and John Mellor-Crummey. 2025. Analyzing the Performance of Applications at Exascale. In *Proceedings of the 39th ACM International Conference on Supercomputing*. 792–806. https://www.doi.org/10.1145/3721145.3730417

[14] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with NumPy. *nature* 585, 7825 (2020), 357–362. https://www.doi.org/10.1038/s41586-020-2649-2

[15] Intel oneAPI Development Team. 2026. dpnp: NumPy-compliant Interface for Data Parallel C++ (DPC++). Accessed: August 2026. https://intelpython.github.io/dpnp

[16] Lawrence Livermore National Laboratory. 2026. AMG: Algebraic Multi-Grid Parallel Iterative Solver Benchmark. Accessed: August 2026. https://www.osti.gov/biblio/1389816

[17] Lawrence Livermore National Laboratory. 2026. El Capitan: Preparing for NNSA's first exascale machine. Accessed: August 2026. https://asc.llnl.gov/exascale/el-capitan

[18] Oak Ridge Leadership Computing Facility. 2026. Frontier. Accessed: August 2026. https://www.olcf.ornl.gov/frontier

[19] Oak Ridge Leadership Computing Facility. 2026. Frontier User Guide. Accessed: August 2026. https://docs.olcf.ornl.gov/systems/frontier_user_guide.html

[20] RAPIDS Development Team. 2026. cuDF: GPU-Accelerated Pandas-like DataFrames. Accessed: August 2026. https://docs.rapids.ai/api/cudf/stable

[21] The HPCToolkit Project. 2026. hpcanalysis. Accessed: August 2026. https://gitlab.com/hpctoolkit/hpcanalysis

[22] Keren Zhou, Laksono Adhianto, Jonathon Anderson, Aaron Cherian, Dejan Grubisic, Mark Krentel, Yumeng Liu, Xiaozhu Meng, and John Mellor-Crummey. 2021. Measurement and analysis of GPU-accelerated applications with HPCToolkit. *Parallel Comput.* 108 (2021), 102837. https://www.doi.org/10.1016/j.parco.2021.102837