Performance Comparison of MPIs using Hatchet

Ian Lumsden[1], Michael Wyatt[1], Stephanie Brink[2], Todd Gamblin[2] , Michela Taufer[1]

[1] Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, USA
[2] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA

Introduction

- Profiling is a way to measure the performance of code and how code runs on systems. In high-performance computing (HPC), it is commonly used to:
  - Identify and mitigate memory, I/O bottlenecks in programs
  - Improve program performance by analyzing parallelism
  - Improve program performance through scalability analysis
- MPI is a standard with many implementations that defines a powerful, high performance form of message passing-based parallelism and interprocess communication.
  - Most HPC programs use MPI (sometimes alongside another tool) to provide parallelism and best utilize system resources.
- Numerous tools for HPC profiling (e.g. TAU, Caliper, HPCToolkit) use their own custom data format and analysis tools.
  - Locks users into only being able to analyze their data with the tools the profiler provides.
- Hatchet allows hierarchical profiling data *from various profilers* to be loaded and analyzed with traditional Python-based data analysis tools [1].

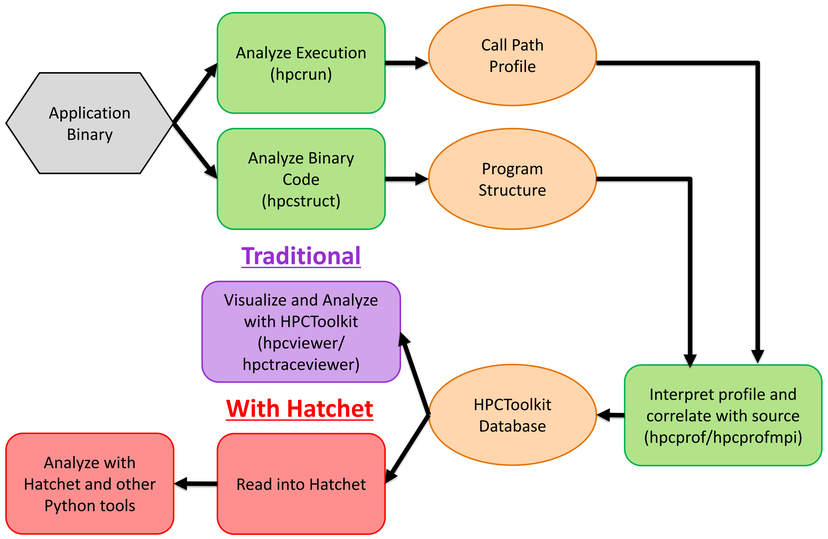

Figure 1: The HPCToolkit Workflow with both traditional data analysis with HPCToolkit's tools and Hatchet-based data analysis.

Research Goals

- Leverage Hatchet to analyze performance of different MPIs using various benchmark applications.
- Identify general performance differences across MPIs that can be studied more in-depth in the future.
- Identify ways in which to extend Hatchet for future analysis.

Methods

- Performed 20 runs of the following benchmarks (10 with MVAPICH 2.3.2 and 10 with Spectrum-MPI 10.2) while profiling with HPCToolkit:
  - Kripke
  - Lammps (with various subpackages enabled)
  - AMG2013
  - Ior (running with MPI I/O support)
- Loaded the HPCToolkit databases into Hatchet and filtered out all non-MPI function calls.
- Ran on UTK's Tellico system (2 compute nodes with 2 16-core IBM Power9 CPUs per node)

Conclusions

- MVAPICH tends to be faster than Spectrum-MPI, but the degree to which it is faster is dependent on the application.
- Spectrum-MPI produces a smaller variance in runtime than MVAPICH for three-quarters of the tested applications.
- Currently, Hatchet is very good at enabling and performing broad and general data analysis, but it is difficult to examine results at a deeper level with the current tools Hatchet makes available.

Future Work

- Add a query language to Hatchet to enable more detailed and expressive data extraction and analysis.
- Expand on this work with more benchmarks and MPIs.
- Use the query language and other new Hatchet features to perform more novel analysis of MPI profiles.
  - Example: determine the reason why MVAPICH outperformed Spectrum-MPI with Kripke.
- Run future benchmarking on larger systems (i.e. Lassen) to examine the scalability of different MPIs.

Results

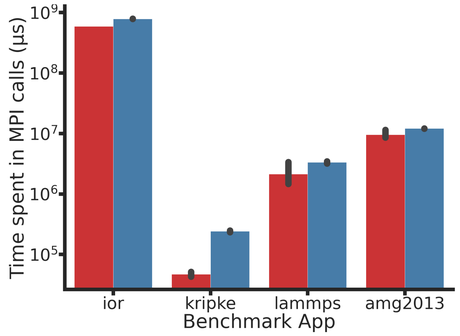

Figure 2: Total time spent in MPI for each benchmark

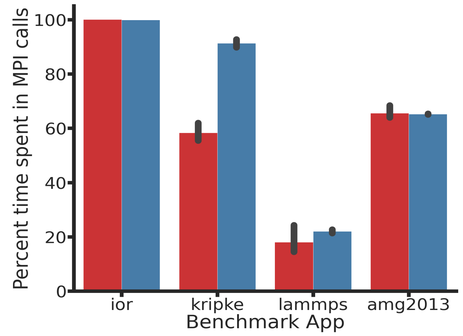

Figure 3: Percent of runtime spent in MPI for each benchmark

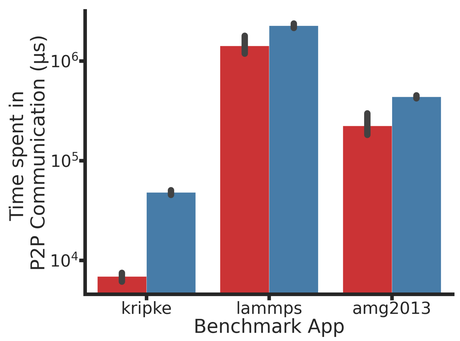

Figure 4: Time spent in Point-to-Point Communications for each benchmark*

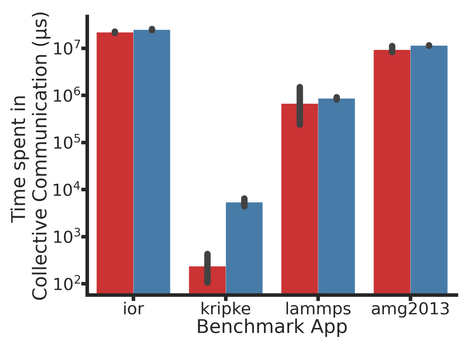

Figure 5: Time spent in Collective Communications for each benchmark*

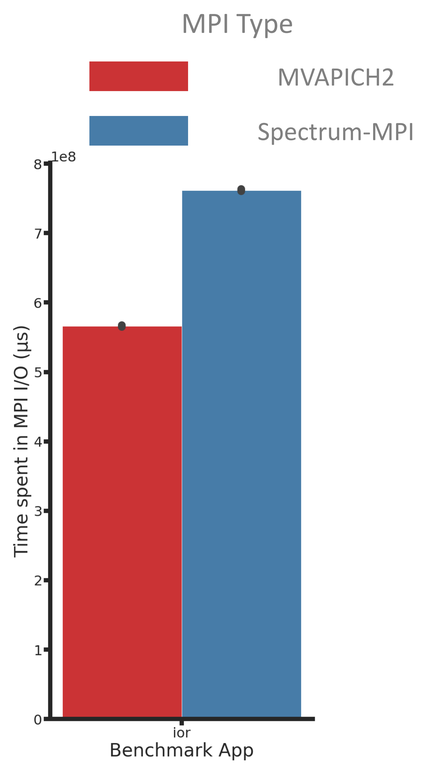

Figure 6: Time spent in MPI I/O for each MPI

Sponsors

THE UNIVERSITY OF TENNESSEE KNOXVILLE

IBM

NSF

* Point-to-Point and Collective Communications functions were chosen based on the MPI 1.0 Standard [2].

References

[1] A. Bhatele, S. Brink, and T. Gamblin, "Hatchet," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019.
[2] I. Laguna, R. Marshall, K. Mohror, M. Ruefenacht, A. Skjellum, and N. Sultana, "A large-scale study of MPI usage in open-source HPC applications," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019.