Justin Bahr
NUID: 001592707

High Performance Computing
Homework 4 - Question 3

Scalasca is a German performance tuning tool that measures runtime behavior for MPI, OpenMP, and hybrid (combination of MPI and OpenMP) parallel programming interfaces, with a focus on science and engineering applications, such as simulations. Also, third-party tools like Vampir and Paraver can be integrated to further visualize performance. Additionally, Scalasca was designed for large, highly scalable systems, but can also be useful for smaller high performance computing systems. After runtime, the tool identifies performance issues mainly targeting synchronization and communication bottlenecks. It can be used in profiling mode, which looks at individual function call paths to determine which aspects of the program are resource intensive and report local process metrics, or tracing mode, which dives deeper to identify when wait states and uneven distributed workloads appear. However, this intense post-mortem tracing introduces a large memory and computation overhead, so it is good practice to carefully select which parts of an application to trace. Finally, one can download this software for free under the BSD Open Source license.

MPIProf is an API that uses PMPI (the built in MPI standard profiling interface) to analyze many MPI implementations, making it a lightweight tool. It reports information about how much memory each process is using and on which node as well as call path information for MPI calls such as latency, message size, and number of occurrences. MPIProf can report information about both collective MPI calls (ie MPI_Scatter()) and point-to-point MPI calls (ie MPI_Send()). It can also track when MPI I/O functions use underlying POSIX I/O functions by using the -mio flag. The statistics output file produced reports on execution time, I/O size and rate. It marks an application when I/O contributes to more than 10% of total execution time.

Both of these performance tools have the benefit of benign open source. Additionally, they each provide useful information about execution time, communication latency, and individual function calls. However, they each have certain perks and drawbacks. The main benefit of MPIProf is that it is not resource intensive, leaving more RAM and compute power for the actual workload being investigated. One drawback I anticipate with MPIProf is that it may be difficult for a beginner to use. There are different report styles chosen when executing, and the reports are broad, meaning a programmer will need to draw conclusions about the data and be familiar with the application they are testing. In contrast, Scalasca provides in-depth reports and a graphical user interface with system visuals. Furthermore, the identification of wait state provides a clear identification of bottlenecks, pointing the programmer directly to areas for optimization. However, Scalasca, especially when used for deep tracing, has a significant memory and computation cost.

In summary, both of these frameworks are powerful, open source tools that can help programmers better understand and tune MPI performance. Scalasca is a better option if a programmer wants to generate visuals, is new to MPI, needs assistance identifying specific communication bottlenecks, wants to perform thorough tracing, wants to integrate with other MPI performance tools, and/or wants to also analyze OpenMP performance. MPIProf is a better option if a programmer prefers a tool that requires less memory and compute resources, is interested in file system I/O performance, and/or wants a simple overview of whether an application is I/O intensive.

Sources:

Scalasca Home Page:
https://www.scalasca.org/

The Scalasca Performance Toolset Architecture - Research Paper:
https://apps.fz-juelich.de/jsc-pubsystem/aigaion/attachments/sthec08-special.pdf-c565f219e0f9e1814e48c543dfe71039.pdf

Using MPIProf for Performance Analysis - NASA:
https://www.nas.nasa.gov/hecc/support/kb/using-mpiprof-for-performance-analysis_525.html

Using MPIProf or I/O Profiling - NASA:
https://www.nas.nasa.gov/hecc/support/kb/using-mpiprof-for-io-profiling_692.html