Parallel Performance Wizard: A Performance Analysis Tool for Partitioned Global-Address-Space Programming Models

Hung-Hsun Su[1], Adam Leko[1], Dan Bonachea[2], Bryan Golden[1], Hans Sherburne[1], Max Billingsley III[1], Alan George[1]
University of Florida[1] / UC Berkeley[2]

Scientific programmers must optimize the total time-to-solution, the combination of software development/refinement time and actual execution time. The increasing complexity at all levels of supercomputing architectures, coupled with advancements in sequential performance and a growing degree of hardware parallelism, often leads to programs that do not yield an expected performance level. As a result, programmers must perform an iterative analysis and optimization process, which places the bulk of the time-to-solution cost into the software development and tuning phase. Performance analysis tools facilitate this process by alleviating the work required to determine the root cause of performance degradation. While several tools are available to support parallel programs written using the well-known message-passing model, there is insufficient tool support for programs developed using Global-Address-Space (GAS) programming models that are gaining popularity. Examples of GAS models include the Unified Parallel C (UPC), Co-array Fortran, and Titanium languages and the SHMEM library. This poster introduces the Parallel Performance Wizard (PPW) tool for analysis of GAS applications, along with the Global Address Space Performance (GASP) interface that enables tool support for programs using GAS models on a variety of machines and implementations.

A major factor in the success of a performance analysis tool is the performance data it gathers and the techniques used to gather this data in the instrumentation and measurement process. Techniques such as source instrumentation, binary instrumentation, and the use of wrapper libraries have been successfully deployed for programs based on the message-passing programming model. Unfortunately, these techniques are not sufficient for programs based on the GAS model, due to the wide range of GAS implementation techniques and difficulties associated with one-sided memory operations, aggressive parallel compiler optimizations, and other aspects of the global address memory model. The Global Address Space Performance (GASP) interface resolves this issue by specifying the interaction between the user program, compiler and the performance analysis tool, and thereby allows GAS model implementers to avoid optimization interference stemming from instrumentation. GASP permits tool developers to support GAS languages on all platforms and languages supporting the interface, which employs a simple callback mechanism applicable to virtually any language. These calls are then handled by the performance tool to record the desired performance data.

GASP support has been successfully implemented in the Berkeley UPC compiler. Figures 1 and 2, respectively, show the tracing and profiling overhead with Berkeley UPC and the GASP PPW instrumentation module.

PPW exists to maximize user productivity while assisting users in analyzing and correcting performance problems in their parallel programs. The tool is geared towards programs written in GAS languages, employing support for the GASP interface to provide functionality for all implementations supporting GASP.

Using data gathered from the GASP interface, PPW provides visualizations and semi-automatic analyses that facilitate the optimization process. With full support for source-line correlation, the visualizations include: profiling table and call-tree diagram (Figure 3), which provide a high-level overview of program performance; a timeline view provided by export of data to Jumpshot (Figure 4) and other popular trace viewers for a detailed view of program performance; and viewers for shared-memory distribution and communication volume (Figure 5) that aid in understanding the one-sided communication patterns that typify GAS applications. Furthermore, PPW includes mechanisms for automatic detection of possible performance bottlenecks and in some cases can provide practical hints on how to remove these bottlenecks.

With the Parallel Performance Wizard tool and the Global Address Space Performance interface, users can effectively and productively analyze the performance of their GAS language programs, ultimately improving the total time-to-solution for their computing problem.

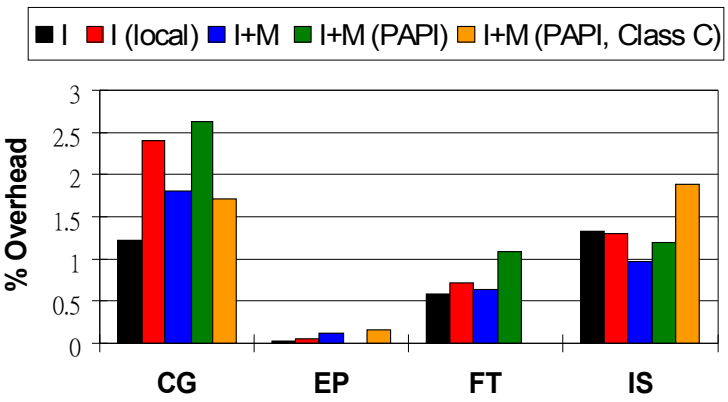

Figure 1: Berkeley UPC GASP Profiling Overhead. Results are for NAS benchmark 2.4 class A unless otherwise noted.

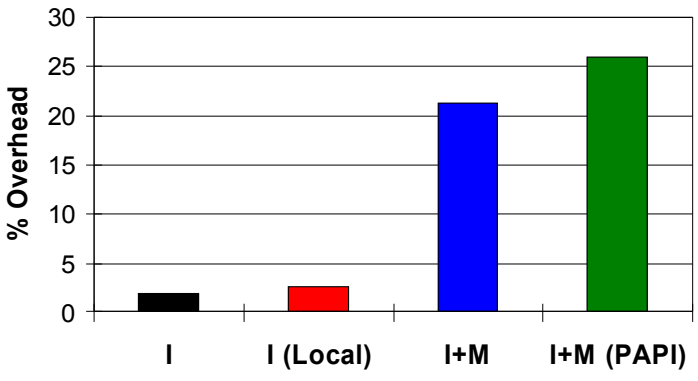

Figure 2: Berkeley UPC GASP Tracing Overhead. I – instrumentation only (empty calls). I+M – actual events recorded through preliminary measurement layer. PAPI – measurement layer records PAPI hardware counter events.

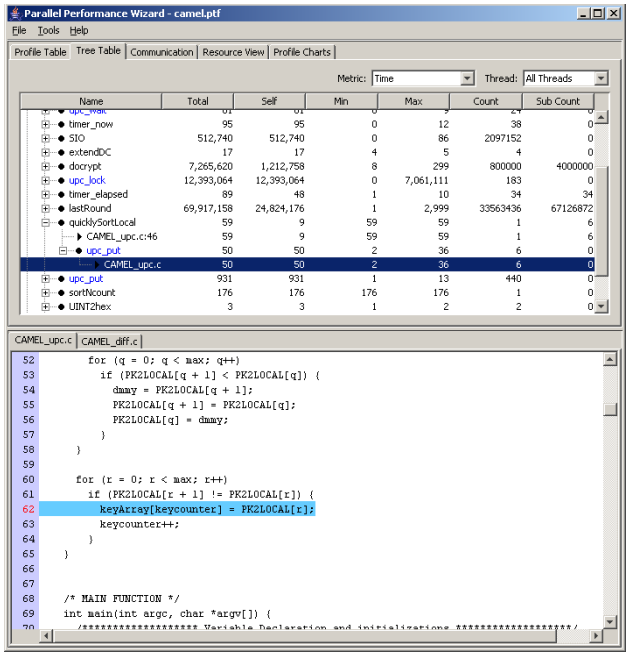

Figure 3: PPW call-tree diagram. This display shows profile data alongside the original source code for a UPC cryptanalysis application.

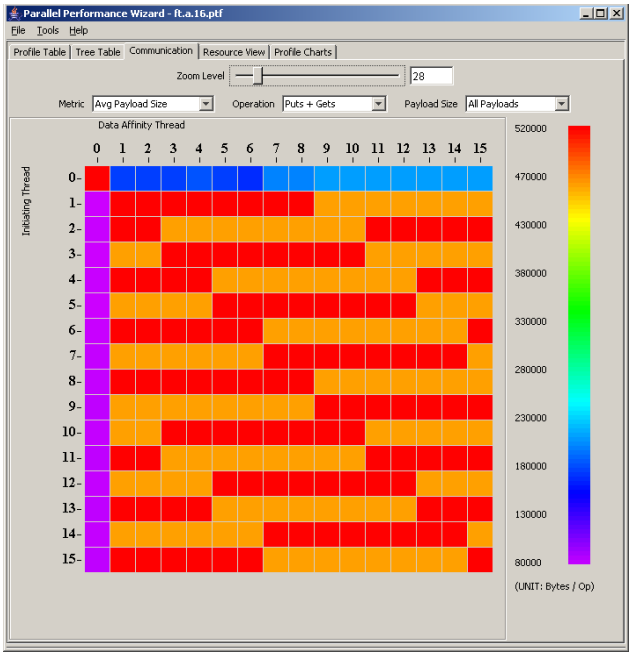

Figure 5: PPW communication volume viewer. This visualization graphically illustrates the communication pattern from the UPC implementation of the NAS FT benchmark (v2.4).

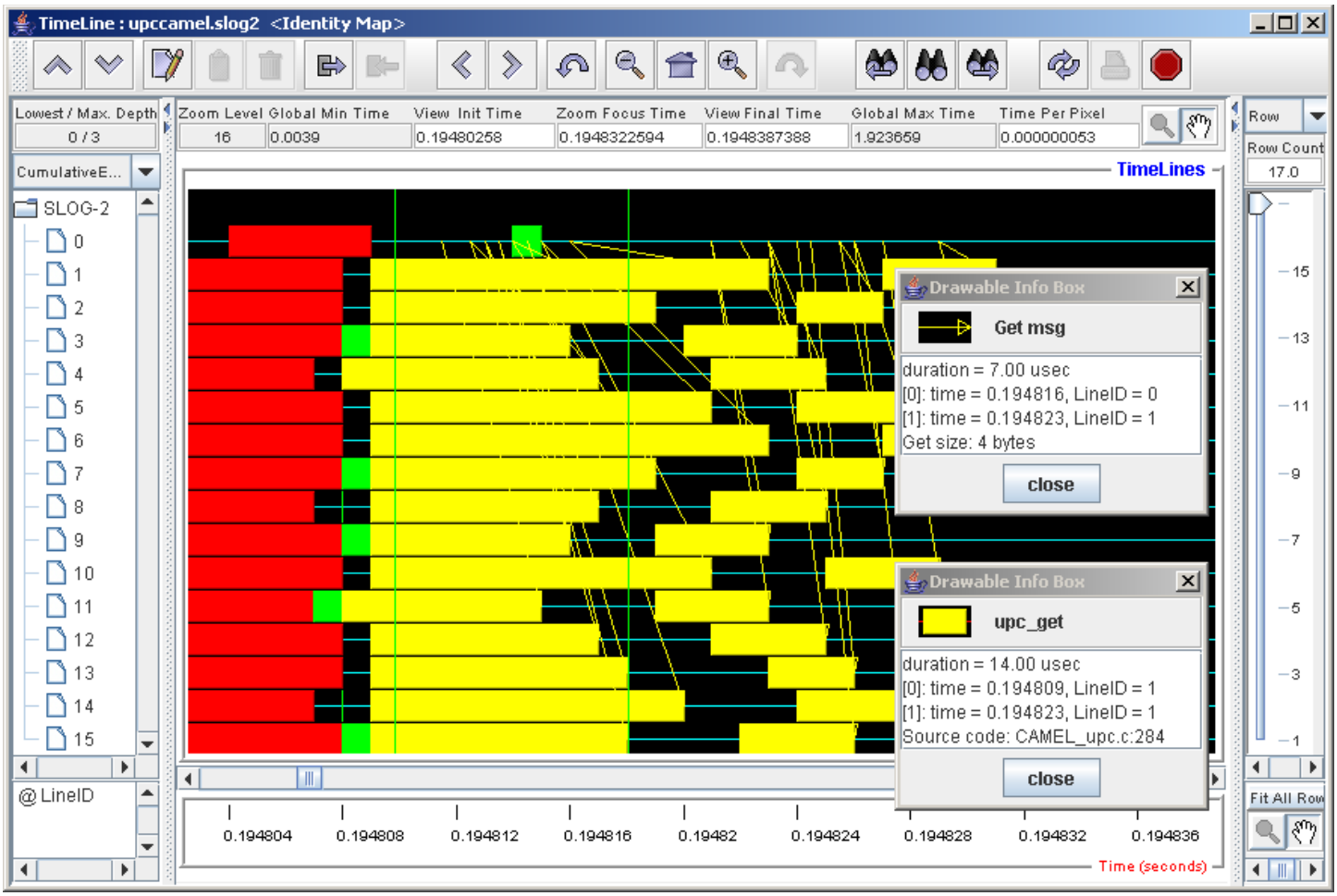

Figure 4: SLOG-2 export viewed in the Jumpshot viewer. Right-clicking on events in the timeline brings up detailed information about that event, including source-code information.