ParaTools

Improving Support for OpenSHMEM in TAU

Milestone 1 and Milestone 2

Subcontract Number: 4000146681

Contractor: ParaTools, Inc.
2836 Kincaid St.
Eugene, OR 97405
(541) 913-8797
info@paratools.com

Overview

This report documents the completion of Milestone 1 and Milestone 2 for Oak Ridge National Laboratory (ORNL). Completion of these milestones requires compliance with the following,

"Milestone 1: TAU with support for tracking callsites"

and

"Milestone 2: TAU with support for writing a merged XML profile files for OpenSHMEM."

The software was delivered to the Company on 10 March 2017 and can be downloaded from **http://tau.uoregon.edu/tau.tgz**.

Introduction

Users of the OpenSHMEM library need simple, yet powerful performance evaluation tools that capture high-utility information and present it in meaningful ways, such as how much time is spent in OpenSHMEM routines, when and where these routines are called in the source code, and on which processing elements (PEs). They need tools that minimize manual steps needed to generated performance data (e.g., work with unmodified binaries to reduce source instrumentation) and perform automatic analysis of key metrics (e.g., the extent and volume of communication). To address these concerns, we have extended the TAU Performance System® to better support performance evaluation of OpenSHMEM applications and to simplify the usage of TAU for OpenSHMEM applications.

OpenSHMEM Callsite Support

We have implemented support for tracking OpenSHMEM callsites in TAU. This allows a user to observe how much time is being spent in OpenSHMEM calls and where the call was invoked in the source code. TAU utilizes debugging information (i.e. executables must be compiled with -g) to resolve callsite addresses to source code file names and line numbers. Binutils 2.27 or later and libunwind 1.1 or later are required. TAU will download and install both requirements if the flags "-bfd=download -unwind=download" are passed to the TAU configuration script.

We tested callsite support with various SHMEM implementations on multiple computing systems available to ParaTools including Linux workstations and clusters, Cray XC30 and XC40 systems, Titan (ORNL), and Godzilla (University of Oregon). SHMEM implementations included OpenSHMEM Reference 1.3, Sandia OpenSHMEM 1.3.1, and Cray SHMEM. We tested with the ISx integer sort application (https://github.com/ParRes/ISx), NAS Parallel Benchmarks, and small matrix multiplication kernels.

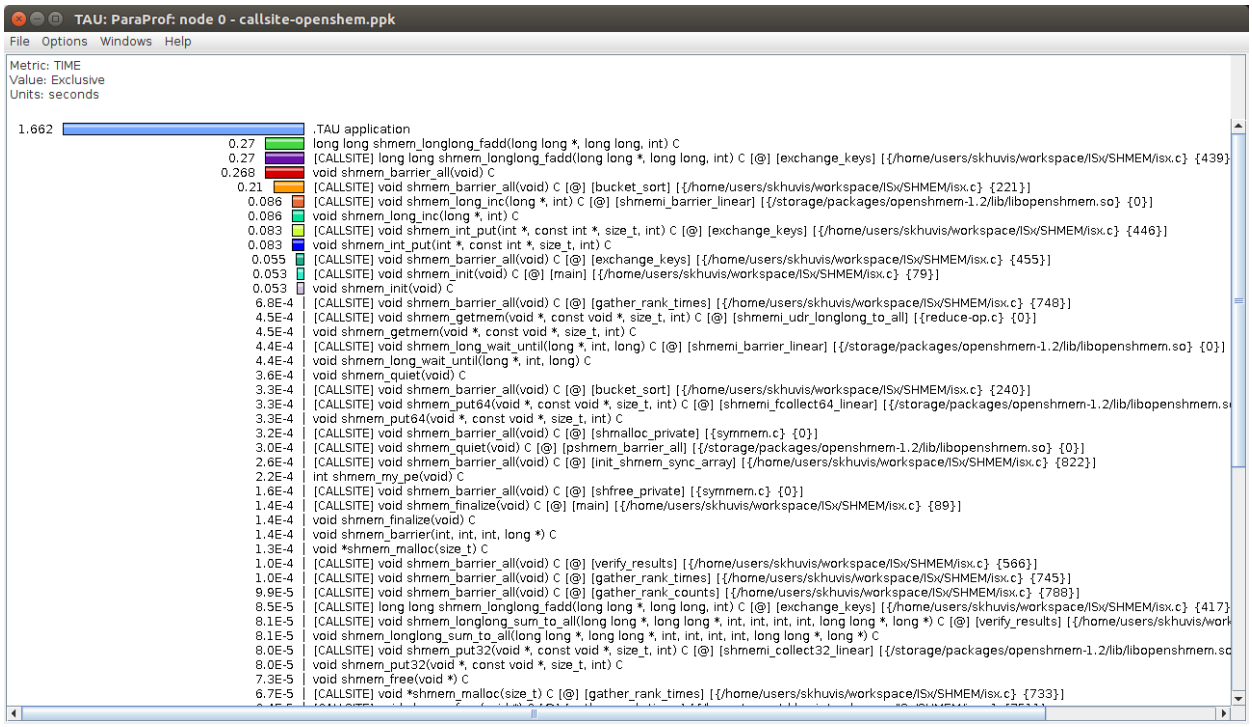

Figure 1: Callsite profile of ISx executing on Godzilla with OpenSHMEM Reference 1.3.

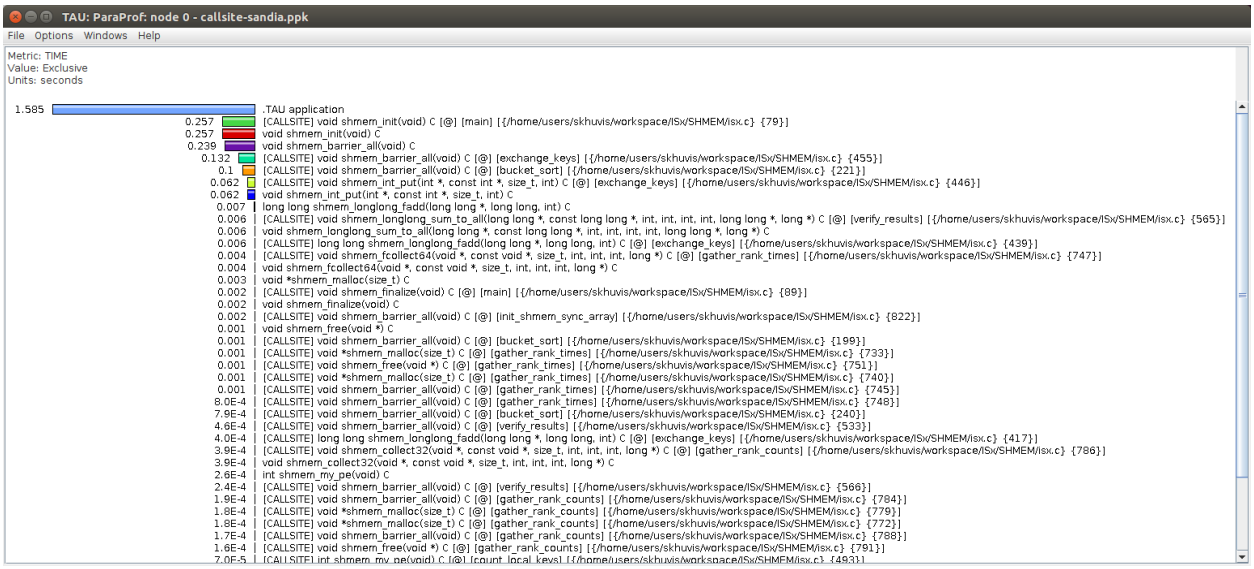

Figure 2: Callsite profile of ISx executing on Godzilla with Sandia OpenSHMEM 1.3.1.

Figures 1 and 2 show callsite profiles of ISx executing on Godzilla with OpenSHMEM 1.3 reference implementation and Sandia OpenSHMEM 1.3.1. The callsite events are indicated by the "[CALLSITE]" tag and show the name of the SHMEM function, the name of the function which invoked that SHMEM function, and the source location where the SHMEM function was invoked. For example, Figure 2 shows that shmem_init was invoked by the "main" function at line 79 of isx.c.

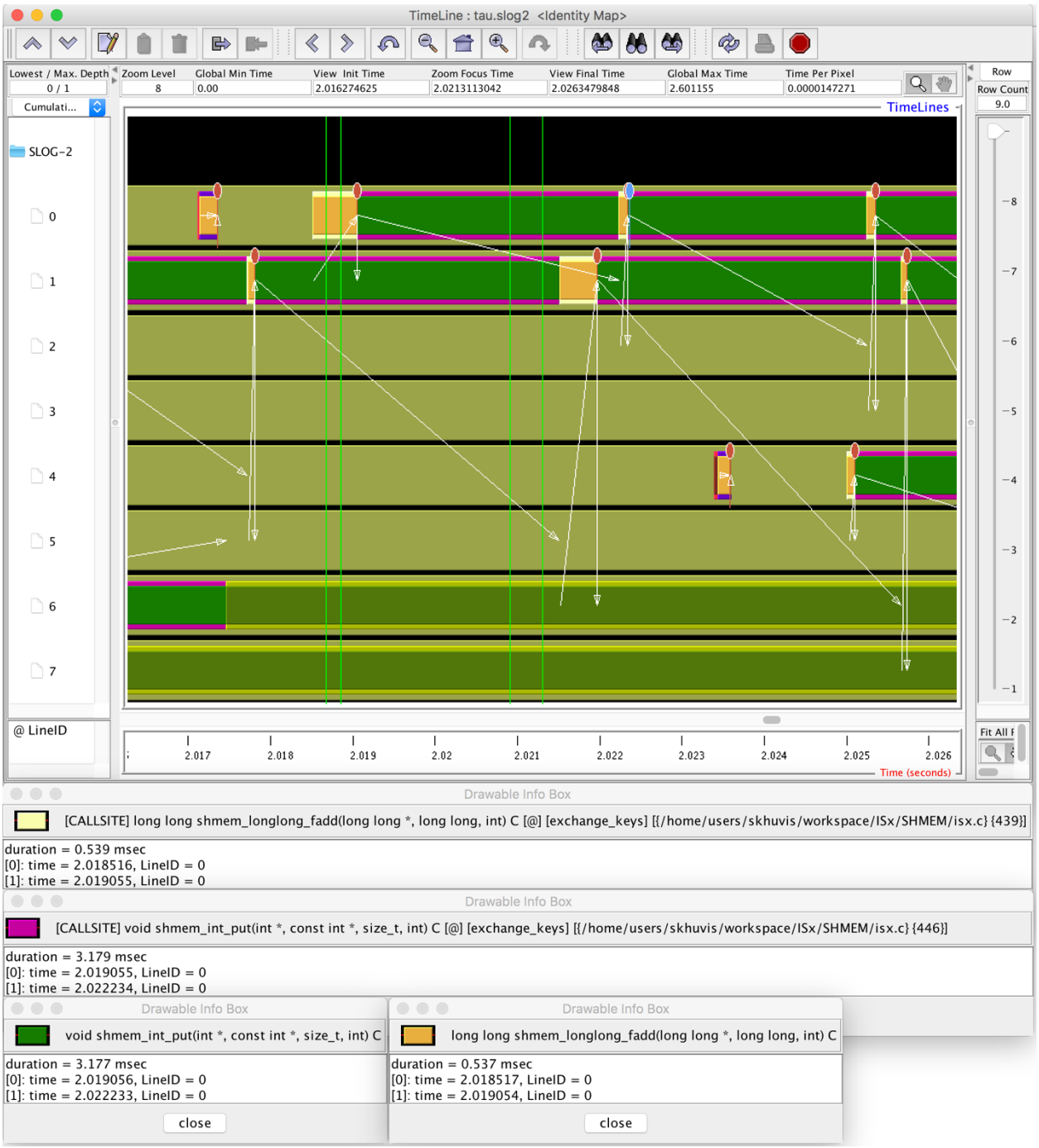

Figure 3: Callsites in a trace of ISx executing on Godzilla with Sandia OpenSHMEM 1.3.1.

Callsites are also supported in TAU's trace format. Figure 3 shows a trace of ISx executing on Godzilla with Sandia OpenSHMEM 1.3.1. The callsites of shmem_int_put and shmem_longlong_fadd are visible as enclosing boxes around the SHMEM function. Messages in flight are shown as arrows between processes. Figure 4 shows SHMEM callsites in the trace function legend.

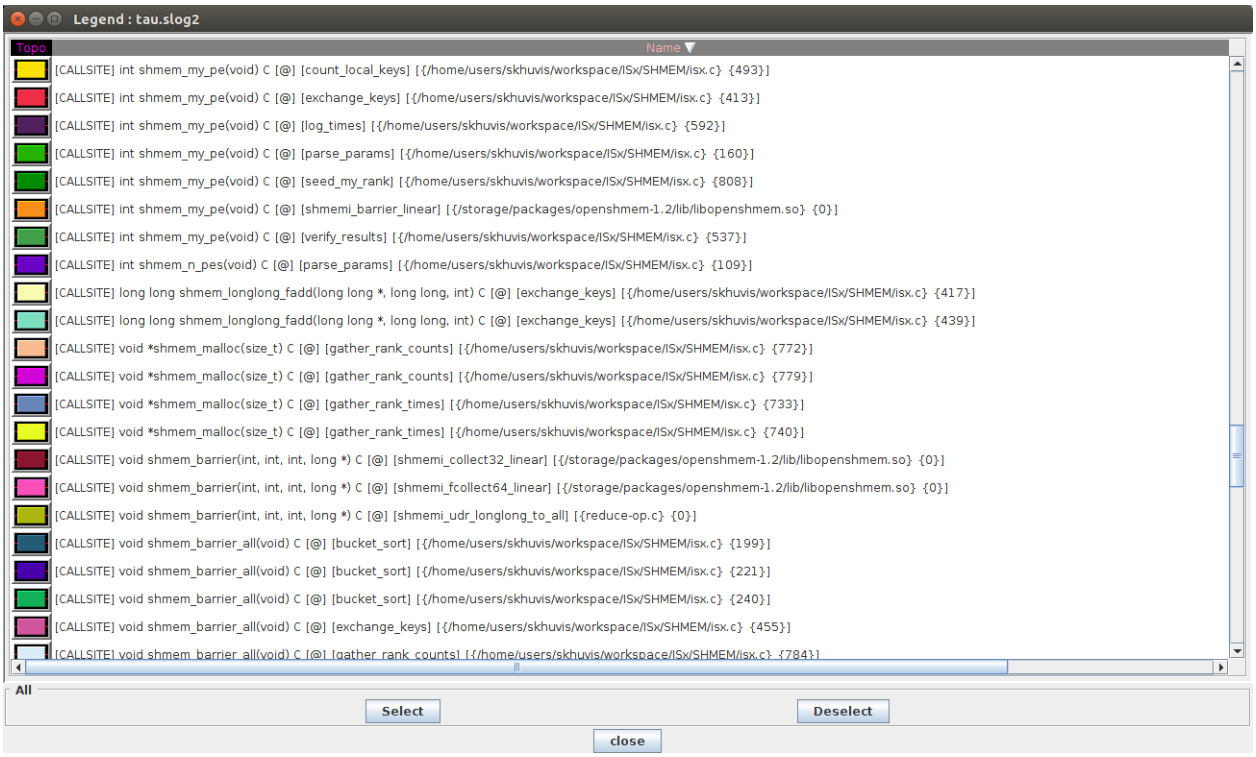

Figure 4: The Jumpshot function legend window showing callsites for SHMEM functions in ISx.

OpenSHMEM Merged Profile Support

We have implemented support for merged profile files in TAU for OpenSHMEM. The merged profile coalesces profile output from each processing element to generate a single XML profile file named “tauprofile.xml”. This is done when the application exits, so events occurring after a call to shmem_finalize are also included in the XML profile file. The profile data contained in tauprofile.xml is exactly the same data as in a typical distributed profile (profile.N.C.T format). Set the TAU_PROFILE_FORMAT environment variable to “merged” to generate merged profile files as shown in Figures 5 and 6.

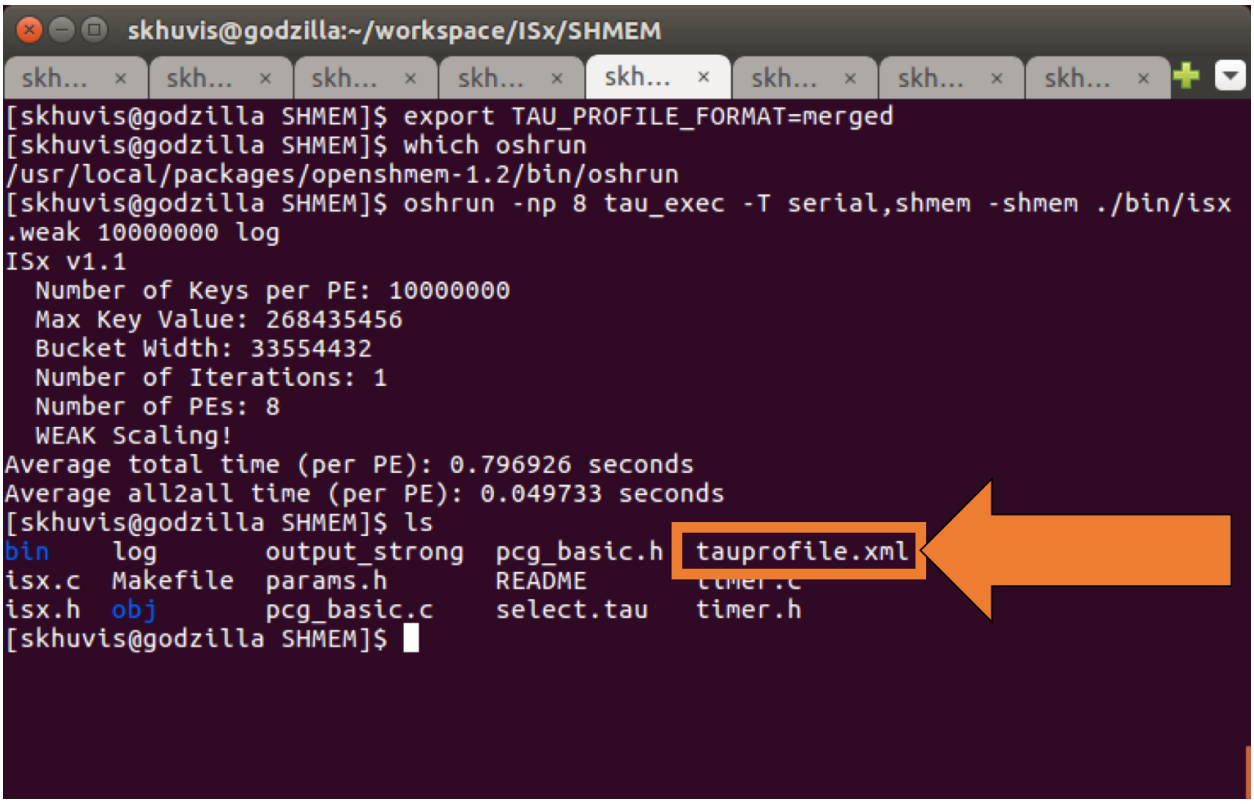

Figure 5: Demonstration of merged profile support with ISx and OpenSHMEM Reference 1.3 on Godzilla.

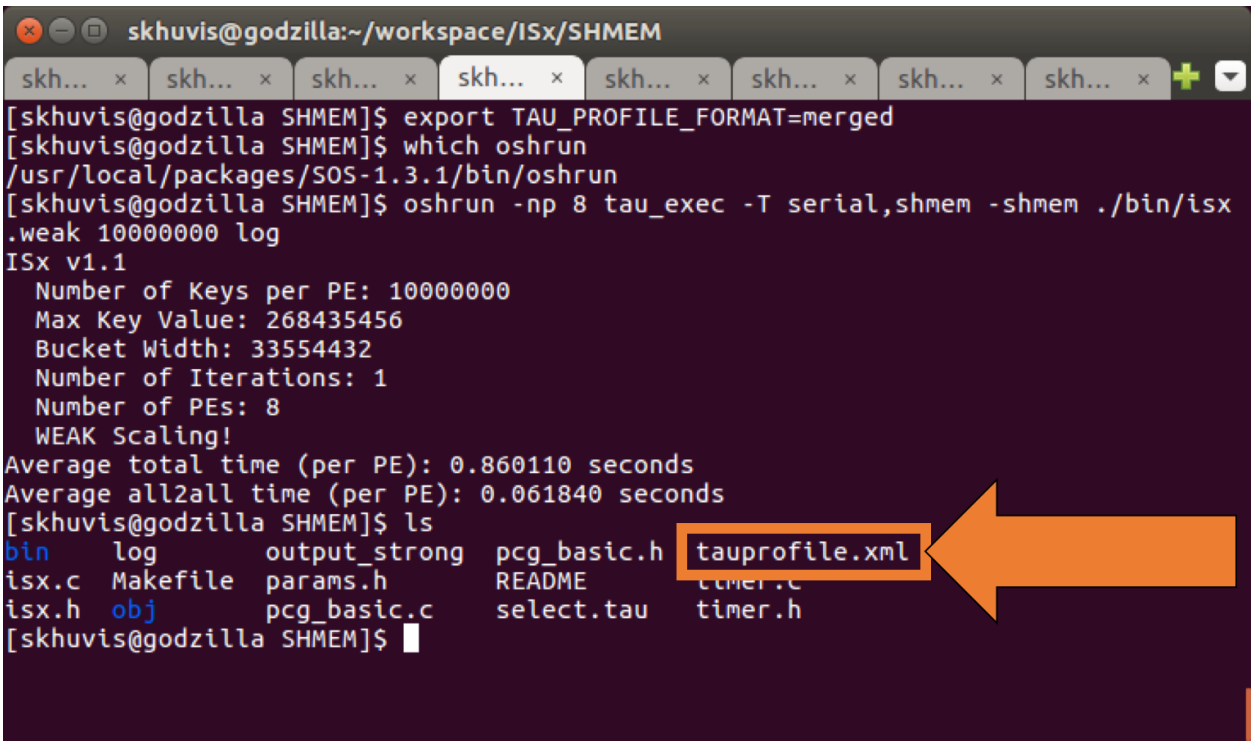

Figure 6: Demonstration of merged profile support with ISx and Sandia OpenSHMEM 1.3.1 on Godzilla.

Summary

We have implemented support for tracking OpenSHMEM callsites in TAU, and for generating merged profile files in TAU for OpenSHMEM. We have tested callsite support with various SHMEM implementations on multiple computing systems available to ParaTools including Linux workstations and clusters, Cray XC30 and XC40 systems, Titan (ORNL), and Godzilla (University of Oregon). SHMEM implementations included OpenSHMEM 1.3, Sandia OpenSHMEM 1.3.1, and Cray SHMEM. We tested with the ISx integer sort application, NAS Parallel Benchmarks, and small matrix multiplication kernels. Our next deliverable "TAU with support for generating OTF2 traces for OpenSHMEM" is on track and will be delivered as scheduled on 13 October 2017.