Status of TAU

Presentation by:

Sameer Shende

ParaTools, Inc,

BoF: OpenSHMEM in the era of Exascale

#501-502, Wednesday, Nov 15, 2017, 5:15pm, SC17, Denver, CO

FY17 Accomplishments

- Improvements the TAU Performance System®:
  - Support for callsites in profiles and traces.
  - Support for TAU merged profiles: TAU_PROFILE_FORMAT=merged
  - Native OTF2 trace support: TAU_TRACE_FORMAT=otf2
- Improvements in TAU Commander:
  - Support for all of the above features added to the user interface.
  - Support for SOS, Cray, OpenMPI, and reference OpenSHMEM compiler wrappers.
- One-hour invited talk presented at OpenSHMEM'17.
- Paper accepted to OpenSHMEM'17: *Performance Analysis of OpenSHMEM Application with TAU Commander.*
- Poster accepted to SC'17 PGAS booth.
- SHMEM support tested on Titan, Cori, Cray XC40, Cray XC30, SGI ICE X, IBM iDataPlex and others.

OpenSHMEM'17 Paper

Performance Analysis of OpenSHMEM Applications with TAU Commander

John C. Linford[1], Samuel Khuvis[1], Sameer Shende[1], Allen Malony[1], Neena Imam[2], and Manjunath Gorentla Venkata[2]

[1]ParaTools, Inc.
2836 Kincaid St., Eugene, OR 97405, USA
{jlinford,skhuvis,sameer,malony}@paratools.com
http://www.paratools.com/

[2]Oak Ridge National Laboratory
1 Bethel Valley Rd, Oak Ridge, TN 37831, USA
{imamn,manjugv}@ornl.gov
http://ut-battelle.org/

Abstract. The TAU Performance System® (TAU) is a powerful and highly versatile profiling and tracing tool ecosystem for performance engineering of parallel programs. Developed over the last twenty years, TAU has evolved with each new generation of HPC systems and scales efficiently to hundreds of thousands of cores. TAU's organic growth has resulted in a loosely coupled software toolbox such that novice users first encountering TAU's complexity and vast array of features are often intimidated and easily frustrated. To lower the barrier to entry for novice TAU users, ParaTools and the US Department of Energy have developed "TAU Commander," a performance engineering workflow manager that facilitates a systematic approach to performance engineering, guides users through common profiling and tracing workflows, and offers constructive feedback in case of error. This work compares TAU and TAU Commander workflows for common performance engineering tasks in OpenSHMEM applications and demonstrates workflows targeting two different SHMEM implementations, Intel Xeon "Haswell" and "Knights Landing" processors, direct and indirect measurement methods, callsite, profiles, and traces.

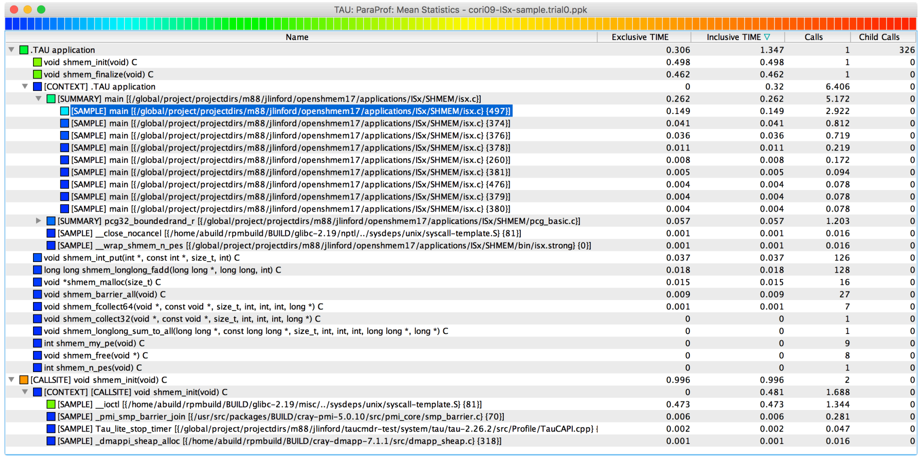
Keywords: TAU Commander; Tau Performance System; Performance Engineering; Profiling; Tracing; Callsite.

1 Introduction

The TAU Performance System® (TAU) is a powerful and highly versatile profiling and tracing tool ecosystem for performance analysis of parallel programs developed in part by Department of Energy (DOE) and National Science Foundation (NSF) research funding granted to the University of Oregon [18, 13, 12]. Developed over the last twenty years, TAU has evolved with each new generation of HPC systems and presently scales efficiently to hundreds of thousands

Abstract. The TAU Performance System® (TAU) is a powerful and highly versatile profiling and tracing tool ecosystem for performance engineering of parallel programs. Developed over the last twenty years, TAU has evolved with each new generation of HPC systems and scales efficiently to hundreds of thousands of cores. TAU's organic growth has resulted in a loosely coupled software toolbox such that novice users first encountering TAU's complexity and vast array of features are of- ten intimidated and easily frustrated. To lower the barrier to entry for novice TAU users, ParaTools and the US Department of Energy have developed ***TAU Commander*****, a performance engineering workflow manager that facilitates a systematic approach to performance engineering, guides users through common profiling and tracing workflows, and offers constructive feedback in case of error.** This work compares TAU and TAU Commander workflows for common performance engineering tasks in **OpenSHMEM** applications and demonstrates workflows targeting two different SHMEM implementations, **Intel Xeon "Haswell" and "Knights Landing" processors, direct and indirect measurement methods, callsite, profiles, and traces.**

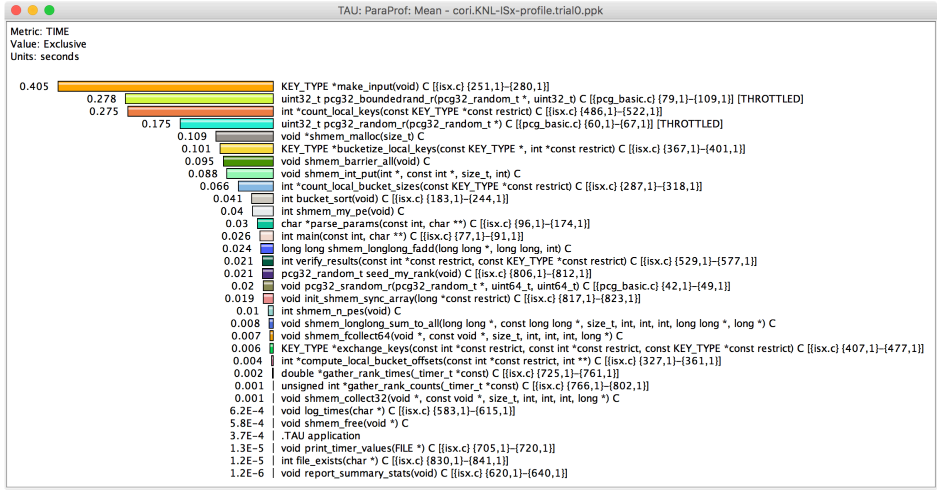
OpenSHMEM Performance Data



TAU: ParaProf: Mean Statistics - cori09-ISx-sample.trial0.ppk

Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
▼ 🟩 .TAU application	0.306	1.347	1	326
🟩 void shmem_init(void) C	0.498	0.498	1	0
🟩 void shmem_finalize(void) C	0.462	0.462	1	0
▼ 🟦 [CONTEXT] .TAU application	0	0.32	6.406	0
▼ 🟩 [SUMMARY] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c}]	0.262	0.262	5.172	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {497}]	0.149	0.149	2.922	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {374}]	0.041	0.041	0.812	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {376}]	0.036	0.036	0.719	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {378}]	0.011	0.011	0.219	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {260}]	0.008	0.008	0.172	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {381}]	0.005	0.005	0.094	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {476}]	0.004	0.004	0.078	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {379}]	0.004	0.004	0.078	0
🟦 [SAMPLE] main [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/isx.c} {380}]	0.004	0.004	0.078	0
▶ 🟦 [SUMMARY] pcg32_boundedrand_r [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/pcg_basic.c}]	0.057	0.057	1.203	0
🟦 [SAMPLE] __close_nocancel [{/home/abuild/rpmbuild/BUILD/glibc-2.19/nptl/../sysdeps/unix/syscall-template.S} {81}]	0.001	0.001	0.016	0
🟦 [SAMPLE] __wrap_shmem_n_pes [{/global/project/projectdirs/m88/jlinford/openshmem17/applications/ISx/SHMEM/bin/isx.strong} {0}]	0.001	0.001	0.016	0
🟦 void shmem_int_put(int *, const int *, size_t, int) C	0.037	0.037	126	0
🟦 long long shmem_longlong_fadd(long long *, long long, int) C	0.018	0.018	128	0
🟦 void *shmem_malloc(size_t) C	0.015	0.015	16	0
🟦 void shmem_barrier_all(void) C	0.009	0.009	27	0
🟦 void shmem_fcollect64(void *, const void *, size_t, int, int, int, long *) C	0.001	0.001	7	0
🟦 void shmem_collect32(void *, const void *, size_t, int, int, int, long *) C	0	0	1	0
🟦 void shmem_longlong_sum_to_all(long long *, const long long *, size_t, int, int, int, long long *, long *) C	0	0	1	0
🟦 int shmem_my_pe(void) C	0	0	9	0
🟦 void shmem_free(void *) C	0	0	8	0
🟦 int shmem_n_pes(void) C	0	0	1	0
▼ 🟧 [CALLSITE] void shmem_init(void) C	0.996	0.996	2	0
▼ 🟦 [CONTEXT] [CALLSITE] void shmem_init(void) C	0	0.481	1.688	0
🟩 [SAMPLE] __ioctl [{/home/abuild/rpmbuild/BUILD/glibc-2.19/misc/../sysdeps/unix/syscall-template.S} {81}]	0.473	0.473	1.344	0
🟦 [SAMPLE] _pmi_smp_barrier_join [{/usr/src/packages/BUILD/cray-pmi-5.0.10/src/pmi_core/smp_barrier.c} {70}]	0.006	0.006	0.281	0
🟦 [SAMPLE] Tau_lite_stop_timer [{/global/project/projectdirs/m88/jlinford/taucmdr-test/system/tau/tau-2.26.2/src/Profile/TauCAPI.cpp}	0.002	0.002	0.047	0
🟦 [SAMPLE] _dmappi_sheap_alloc [{/home/abuild/rpmbuild/BUILD/cray-dmapp-7.1.1/src/dmapp_sheap.c} {318}]	0.001	0.001	0.016	0

Cray SHMEM, Event-based Sampling with Callsites

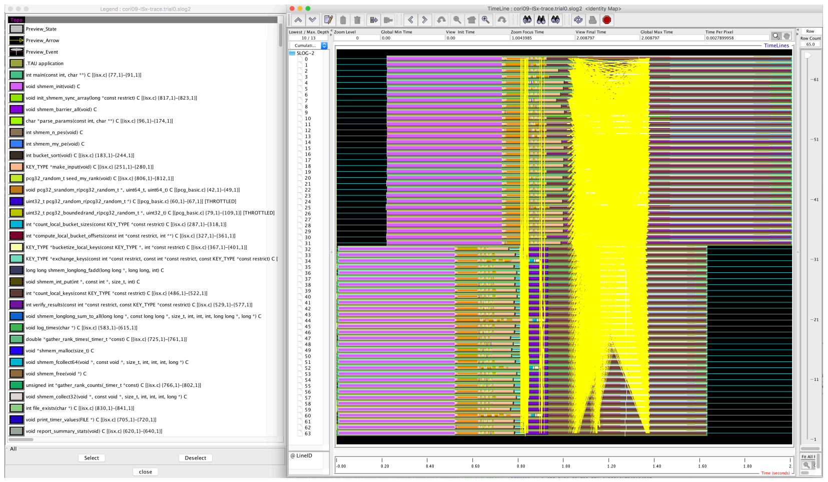


TAU: ParaProf: Mean - cori.KNL-ISx-profile.trial0.ppk

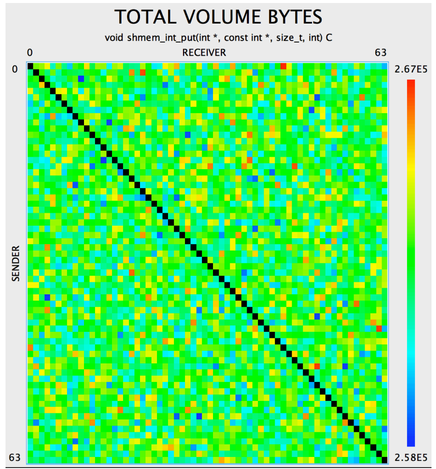
```
Metric: TIME
Value: Exclusive
Units: seconds

0.405   KEY_TYPE *make_input(void) C [{isx.c} {251,1}-{280,1}]
0.278   uint32_t pcg32_boundedrand_r(pcg32_random_t *, uint32_t) C [{pcg_basic.c} {79,1}-{109,1}] [THROTTLED]
0.275   int *count_local_keys(const KEY_TYPE *const restrict) C [{isx.c} {486,1}-{522,1}]
0.175   uint32_t pcg32_random_r(pcg32_random_t *) C [{pcg_basic.c} {60,1}-{67,1}] [THROTTLED]
0.109   void *shmem_malloc(size_t) C
0.101   KEY_TYPE *bucketize_local_keys(const KEY_TYPE *, int *const restrict) C [{isx.c} {367,1}-{401,1}]
0.095   void shmem_barrier_all(void) C
0.088   void shmem_int_put(int *, const int *, size_t, int) C
0.066   int *count_local_bucket_sizes(const KEY_TYPE *const restrict) C [{isx.c} {287,1}-{318,1}]
0.041   int bucket_sort(void) C [{isx.c} {183,1}-{244,1}]
0.04    int shmem_my_pe(void) C
0.03    char *parse_params(const int, char **) C [{isx.c} {96,1}-{174,1}]
0.026   int main(const int, char **) C [{isx.c} {77,1}-{91,1}]
0.024   long long shmem_longlong_fadd(long long *, long long, int) C
0.021   int verify_results(const int *const restrict, const KEY_TYPE *const restrict) C [{isx.c} {529,1}-{577,1}]
0.021   pcg32_random_t seed_my_rank(void) C [{isx.c} {806,1}-{812,1}]
0.02    void pcg32_srandom_r(pcg32_random_t *, uint64_t, uint64_t) C [{pcg_basic.c} {42,1}-{49,1}]
0.019   void init_shmem_sync_array(long *const restrict) C [{isx.c} {817,1}-{823,1}]
0.01    int shmem_n_pes(void) C
0.008   void shmem_longlong_sum_to_all(long long *, const long long *, size_t, int, int, int, long long *, long *) C
0.007   void shmem_fcollect64(void *, const void *, size_t, int, int, int, long *) C
0.006   KEY_TYPE *exchange_keys(const int *const restrict, const int *const restrict, const KEY_TYPE *const restrict) C [{isx.c} {407,1}-{477,1}]
0.004   int *compute_local_bucket_offsets(const int *const restrict, int **) C [{isx.c} {327,1}-{361,1}]
0.002   double *gather_rank_times(_timer_t *const) C [{isx.c} {725,1}-{761,1}]
0.001   unsigned int *gather_rank_counts(_timer_t *const) C [{isx.c} {766,1}-{802,1}]
0.001   void shmem_collect32(void *, const void *, size_t, int, int, int, long *) C
6.2E-4  void log_times(char *) C [{isx.c} {583,1}-{615,1}]
5.8E-4  void shmem_free(void *) C
3.7E-4  .TAU application
1.3E-5  void print_timer_values(FILE *) C [{isx.c} {705,1}-{720,1}]
1.2E-5  int file_exists(char *) C [{isx.c} {830,1}-{841,1}]
1.2E-6  void report_summary_stats(void) C [{isx.c} {620,1}-{640,1}]
```

Sandia OpenSHMEM on KNL (Cori)

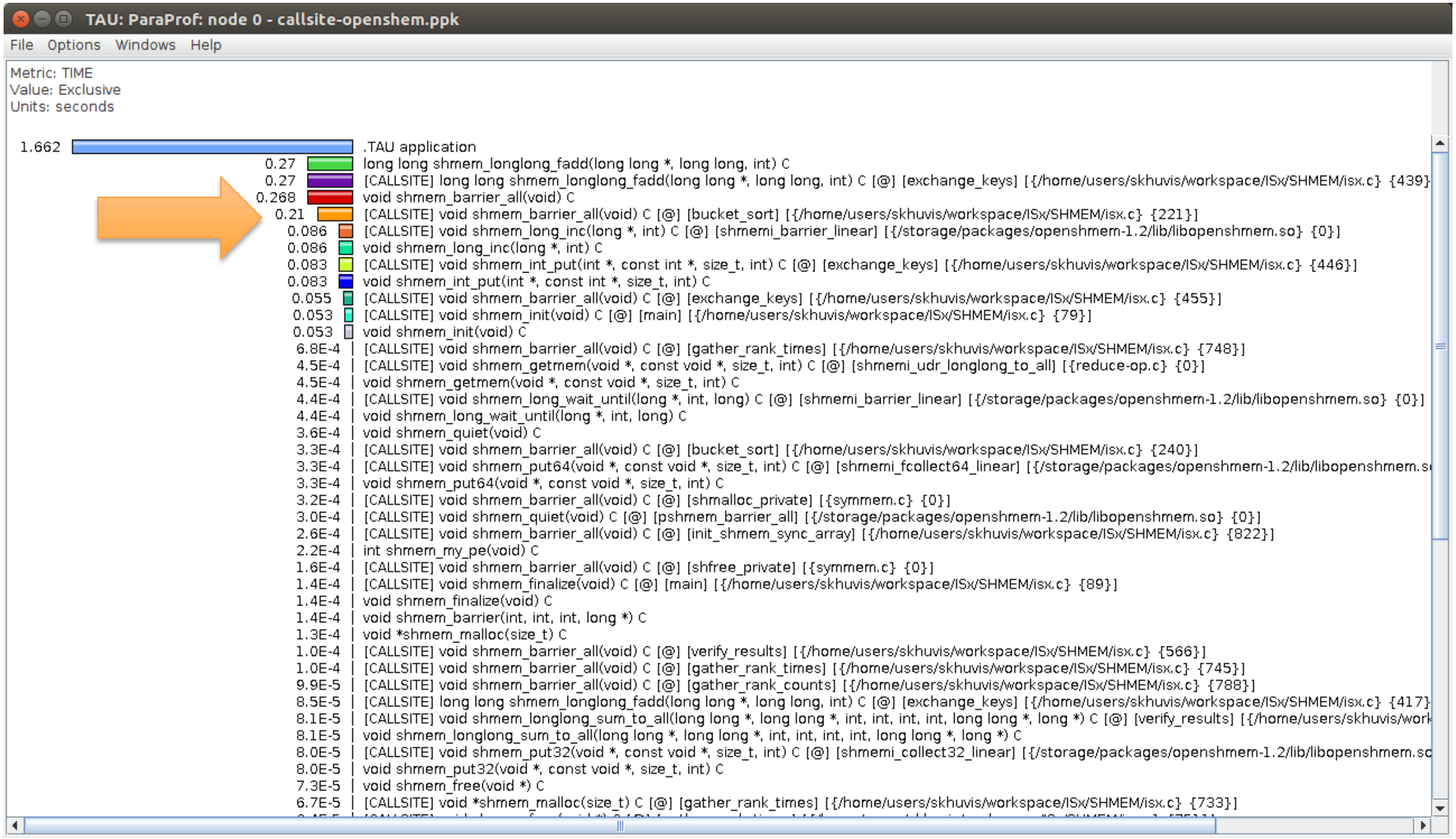


Sandia OpenSHMEM, Source-based Instrumentation with Callsites



shmem_int_put() communication matrix

OpenSHMEM Callsites in Profiles


TAU: ParaProf: node 0 - callsite-openshem.ppk

File Options Windows Help

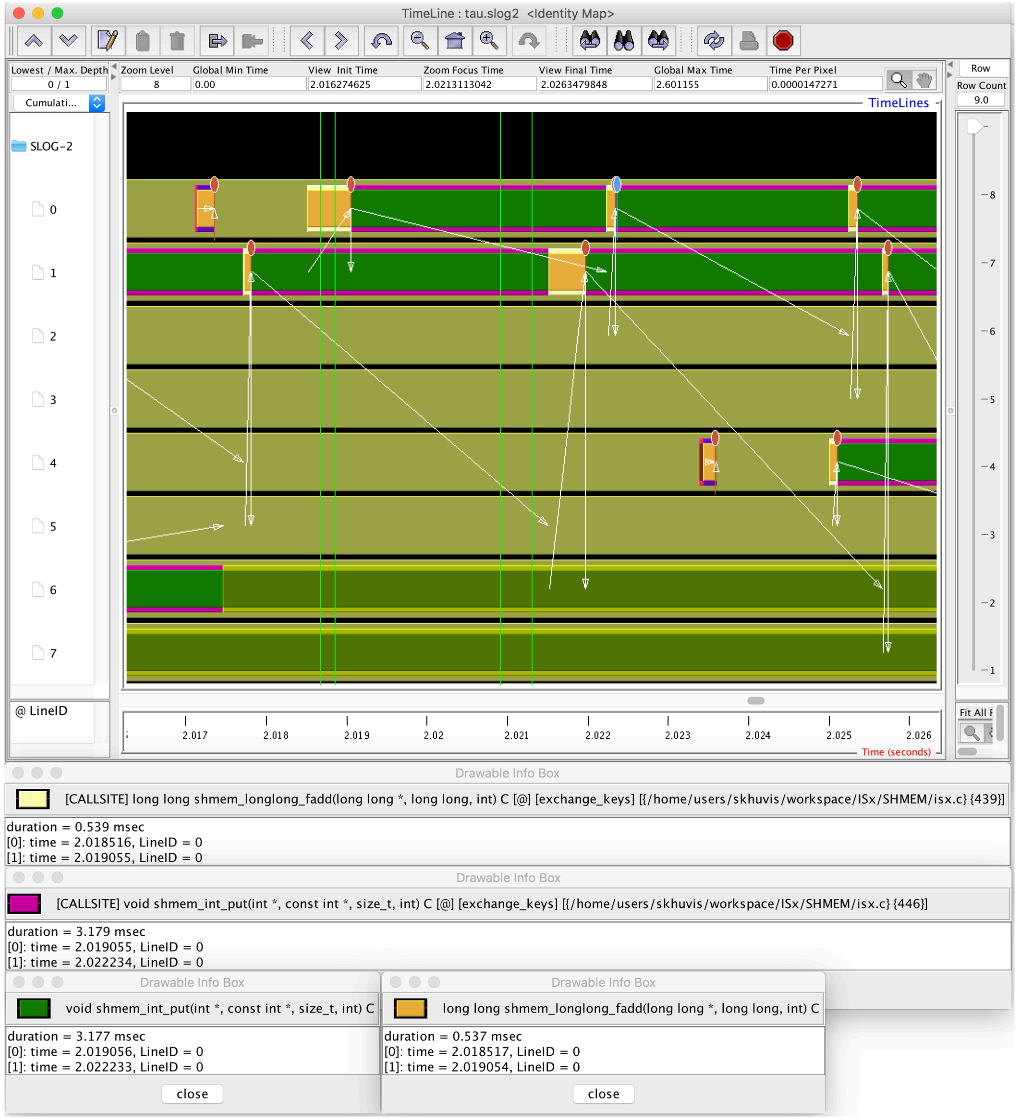
```
Metric: TIME
Value: Exclusive
Units: seconds

1.662   .TAU application
0.27    long long shmem_longlong_fadd(long long *, long long, int) C
0.27    [CALLSITE] long long shmem_longlong_fadd(long long *, long long, int) C [@] [exchange_keys] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {439}
0.268   void shmem_barrier_all(void) C
0.21    [CALLSITE] void shmem_barrier_all(void) C [@] [bucket_sort] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {221}]
0.086   [CALLSITE] void shmem_long_inc(long *, int) C [@] [shmemi_barrier_linear] [{/storage/packages/openshmem-1.2/lib/libopenshmem.so} {0}]
0.086   void shmem_long_inc(long *, int) C
0.083   [CALLSITE] void shmem_int_put(int *, const int *, size_t, int) C [@] [exchange_keys] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {446}]
0.083   void shmem_int_put(int *, const int *, size_t, int) C
0.055   [CALLSITE] void shmem_barrier_all(void) C [@] [exchange_keys] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {455}]
0.053   [CALLSITE] void shmem_init(void) C [@] [main] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {79}]
0.053   void shmem_init(void) C
6.8E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [gather_rank_times] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {748}]
4.5E-4  [CALLSITE] void shmem_getmem(void *, const void *, size_t, int) C [@] [shmemi_udr_longlong_to_all] [{reduce-op.c} {0}]
4.5E-4  void shmem_getmem(void *, const void *, size_t, int) C
4.4E-4  [CALLSITE] void shmem_long_wait_until(long *, int, long) C [@] [shmemi_barrier_linear] [{/storage/packages/openshmem-1.2/lib/libopenshmem.so} {0}]
4.4E-4  void shmem_long_wait_until(long *, int, long) C
3.6E-4  void shmem_quiet(void) C
3.3E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [bucket_sort] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {240}]
3.3E-4  [CALLSITE] void shmem_put64(void *, const void *, size_t, int) C [@] [shmemi_fcollect64_linear] [{/storage/packages/openshmem-1.2/lib/libopenshmem.s
3.3E-4  void shmem_put64(void *, const void *, size_t, int) C
3.2E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [shmalloc_private] [{symmem.c} {0}]
3.0E-4  [CALLSITE] void shmem_quiet(void) C [@] [pshmem_barrier_all] [{/storage/packages/openshmem-1.2/lib/libopenshmem.so} {0}]
2.6E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [init_shmem_sync_array] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {822}]
2.2E-4  int shmem_my_pe(void) C
1.6E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [shfree_private] [{symmem.c} {0}]
1.4E-4  [CALLSITE] void shmem_finalize(void) C [@] [main] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {89}]
1.4E-4  void shmem_finalize(void) C
1.4E-4  void shmem_barrier(int, int, int, long *) C
1.3E-4  void *shmem_malloc(size_t) C
1.0E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [verify_results] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {566}]
1.0E-4  [CALLSITE] void shmem_barrier_all(void) C [@] [gather_rank_times] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {745}]
9.9E-5  [CALLSITE] void shmem_barrier_all(void) C [@] [gather_rank_counts] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {788}]
8.5E-5  [CALLSITE] long long shmem_longlong_fadd(long long *, long long, int) C [@] [exchange_keys] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {417}
8.1E-5  [CALLSITE] void shmem_longlong_sum_to_all(long long *, long long *, int, int, int, int, long long *, long *) C [@] [verify_results] [{/home/users/skhuvis/work
8.1E-5  void shmem_longlong_sum_to_all(long long *, long long *, int, int, int, int, long long *, long *) C
8.0E-5  [CALLSITE] void shmem_put32(void *, const void *, size_t, int) C [@] [shmemi_collect32_linear] [{/storage/packages/openshmem-1.2/lib/libopenshmem.sc
8.0E-5  void shmem_put32(void *, const void *, size_t, int) C
7.3E-5  void shmem_free(void *) C
6.7E-5  [CALLSITE] void *shmem_malloc(size_t) C [@] [gather_rank_times] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {733}]
```

TAU Status

- Callsites:
  - TAU fully supports callsites in profiles and traces of C, C++, and Fortran SHMEM applications.
- Merged Profiles:
  - TAU fully supports merged profiles of C, C++ and Fortran SHMEM applications.
  - Hybrid MPI+SHMEM also supported.
  - Local events occurring before shmem_init() and after shmem_finalize() are shown in the merged profile.
- SHMEM support in TAU Commander:
  - Full support for C, C++, and Fortran.
  - Tested on Titan, Cori, Cray XC40, Cray XC30, SGI ICE X, IBM iDataPlex.
- OTF2 native support for SHMEM:
  - Successfully generated OTF2 profiles of simple codes.
  - Now taking a new approach to remove Score-P dependency.
- OpenSHMEM Fortran Bindings Generator
  - Modified the SHMEM library wrapper code generator to automatically generate Fortran ISO C bindings.

OpenSHMEM Callsites in Traces



TimeLine : tau.slog2 <Identity Map>

Lowest / Max. Depth: 0 / 1
Zoom Level: 8
Global Min Time: 0.00
View Init Time: 2.016274625
Zoom Focus Time: 2.0213113042
View Final Time: 2.0263479848
Global Max Time: 2.601155
Time Per Pixel: 0.0000147271
Cumulati...
Row Count: 9.0

Drawable Info Box

[CALLSITE] long long shmem_longlong_fadd(long long *, long long, int) C [@] [exchange_keys] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {439}]

```
duration = 0.539 msec
[0]: time = 2.018516, LineID = 0
[1]: time = 2.019055, LineID = 0
```

Drawable Info Box

[CALLSITE] void shmem_int_put(int *, const int *, size_t, int) C [@] [exchange_keys] [{/home/users/skhuvis/workspace/ISx/SHMEM/isx.c} {446}]

```
duration = 3.179 msec
[0]: time = 2.019055, LineID = 0
[1]: time = 2.022234, LineID = 0
```

Drawable Info Box

void shmem_int_put(int *, const int *, size_t, int) C

```
duration = 3.177 msec
[0]: time = 2.019056, LineID = 0
[1]: time = 2.022233, LineID = 0
```

close

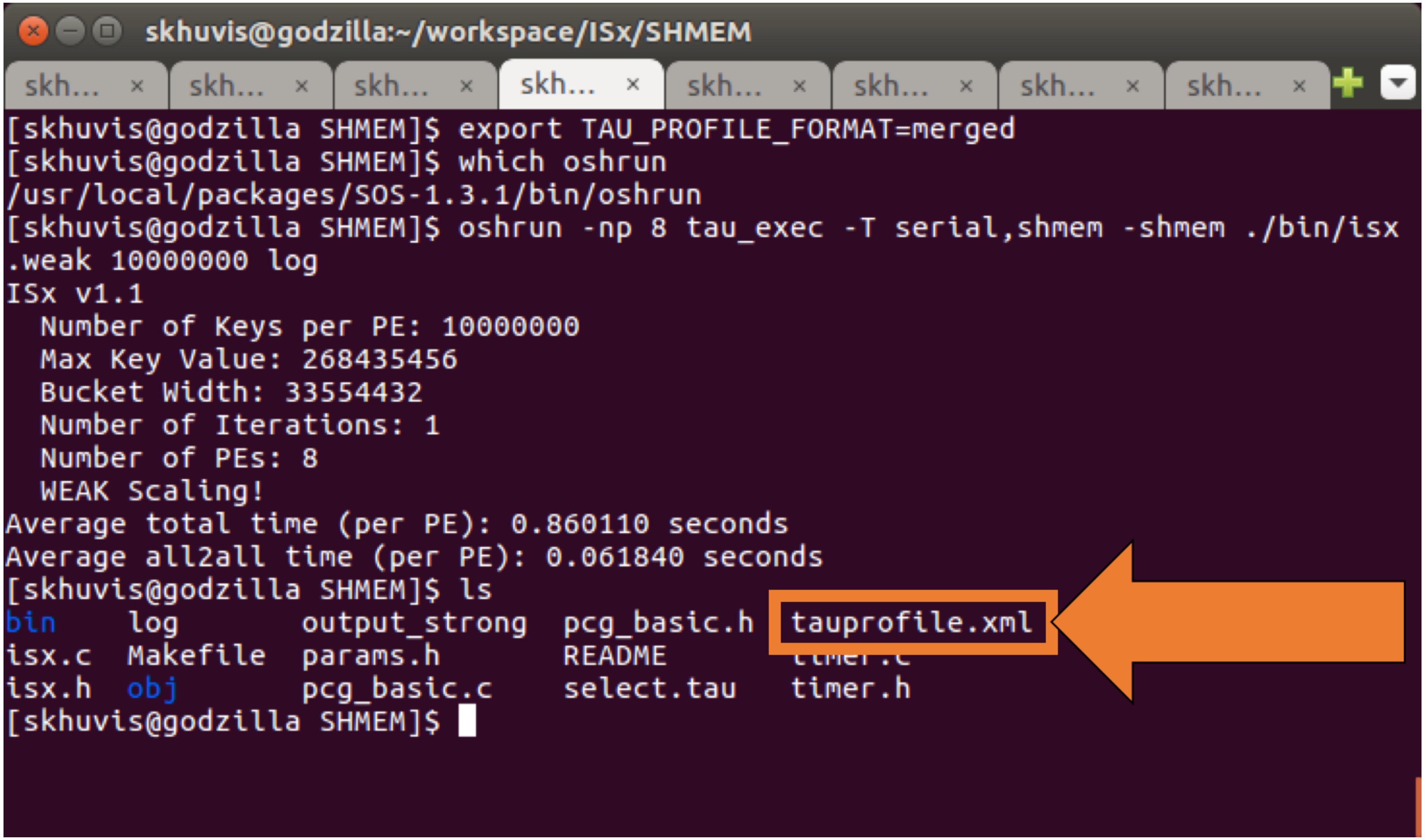
Drawable Info Box

long long shmem_longlong_fadd(long long *, long long, int) C

```
duration = 0.537 msec
[0]: time = 2.018517, LineID = 0
[1]: time = 2.019054, LineID = 0
```

close

OpenSHMEM Merged Profiles



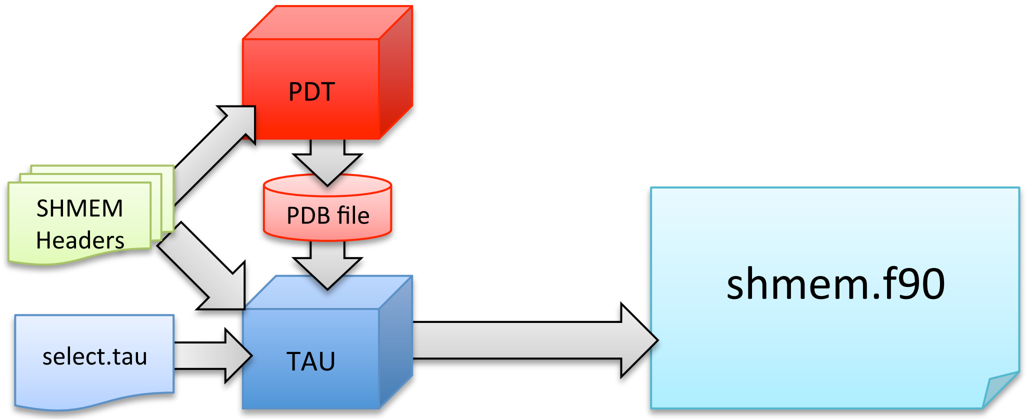
```
[skhuvis@godzilla SHMEM]$ export TAU_PROFILE_FORMAT=merged
[skhuvis@godzilla SHMEM]$ which oshrun
/usr/local/packages/SOS-1.3.1/bin/oshrun
[skhuvis@godzilla SHMEM]$ oshrun -np 8 tau_exec -T serial,shmem -shmem ./bin/isx
.weak 10000000 log
ISx v1.1
  Number of Keys per PE: 10000000
  Max Key Value: 268435456
  Bucket Width: 33554432
  Number of Iterations: 1
  Number of PEs: 8
  WEAK Scaling!
Average total time (per PE): 0.860110 seconds
Average all2all time (per PE): 0.061840 seconds
[skhuvis@godzilla SHMEM]$ ls
bin     log        output_strong  pcg_basic.h   tauprofile.xml
isx.c   Makefile   params.h       README        timer.c
isx.h   obj        pcg_basic.c    select.tau    timer.h
[skhuvis@godzilla SHMEM]$
```

SHMEM in TAU Commander

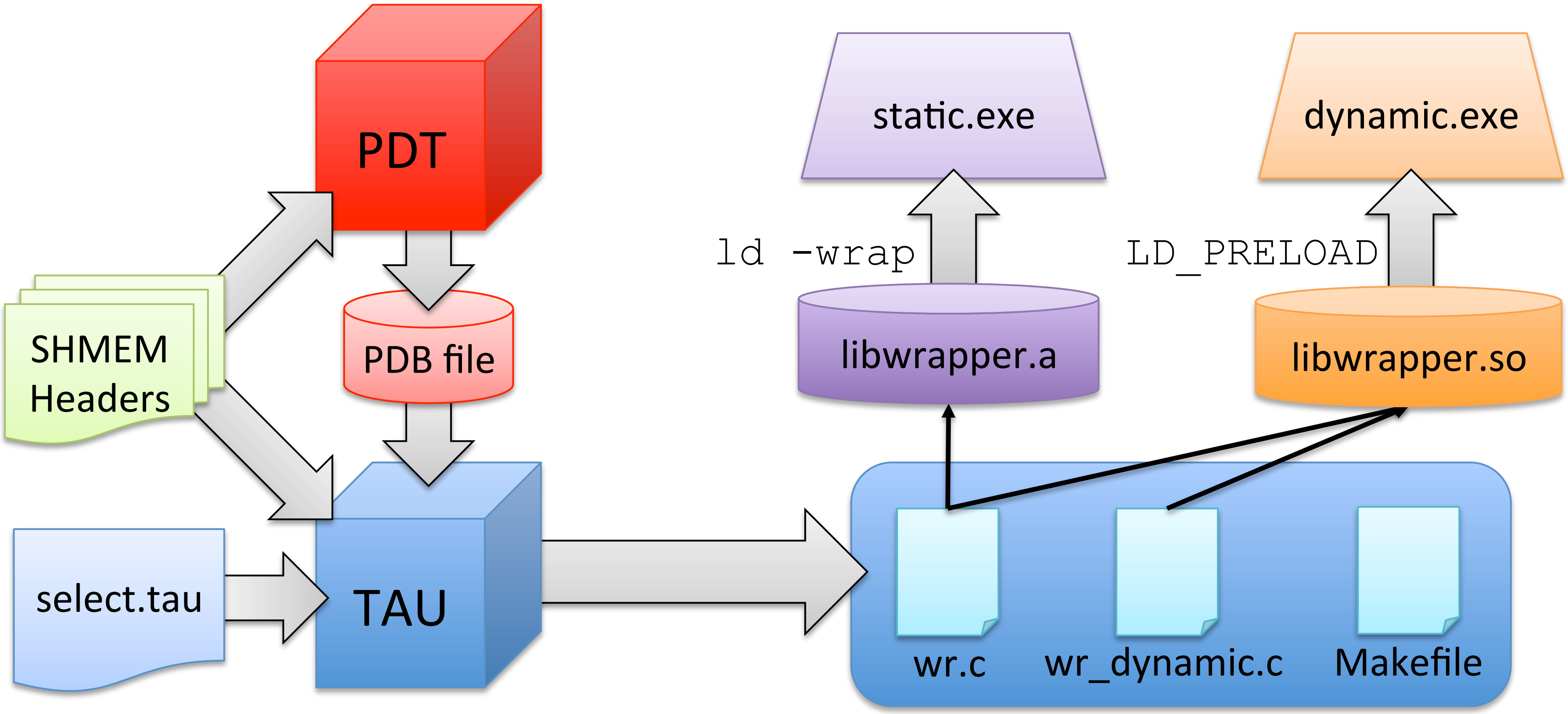
- Added automatic SHMEM version detection for improved support for pre-OpenSHMEM 1.3 implementations.
- Added support for "oshCC" as SHMEM C++ compiler.
- Tested with ISx and Fortran NPB.
  - DOE: Titan and Cori
  - DOD: Armstrong (XC30), Conrad (XC40), Copper (XE6m), Excalibur (XC40), Thunder (ICE X), Gordon (XC40), Haise (iDataPlex), Shepard (XC30), and Topaz (ICE X).
  - 23 issues logged on Github, 19 resolved.
- TAU Commander workflow is the same on all systems:
  - $ tau initialize --shmem
  - $ tau oshfort *.f90
  - $ tau ./a.out
  - $ tau show

OpenSHMEM Fortran ISO C Bindings

- Created a custom version of the tau_wrap tool that automatically generates Fortran bindings for OpenSHMEM that use ISO_C_BINDING.
  - https://github.com/jlinford/shmem_iso_c_binding
- Support Fortran users after Fortran deprecation.



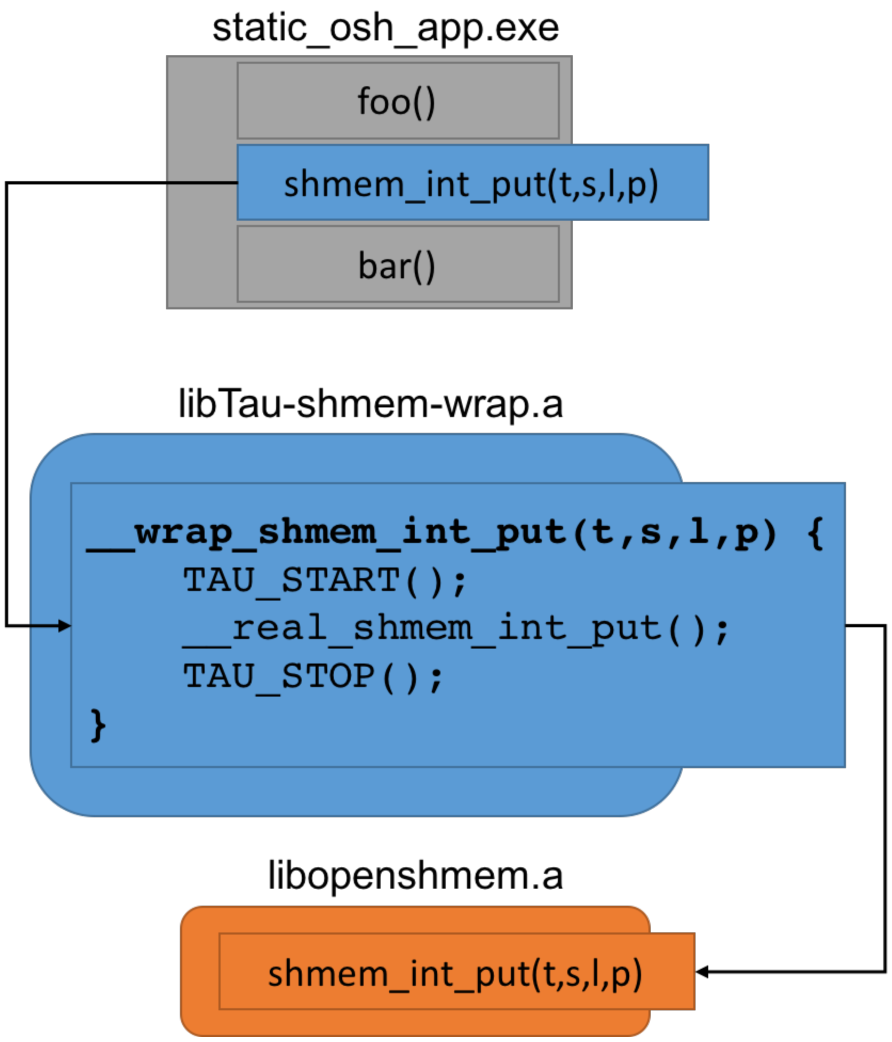
SHMEM Wrapper Library Generation



Symbol Wrapping

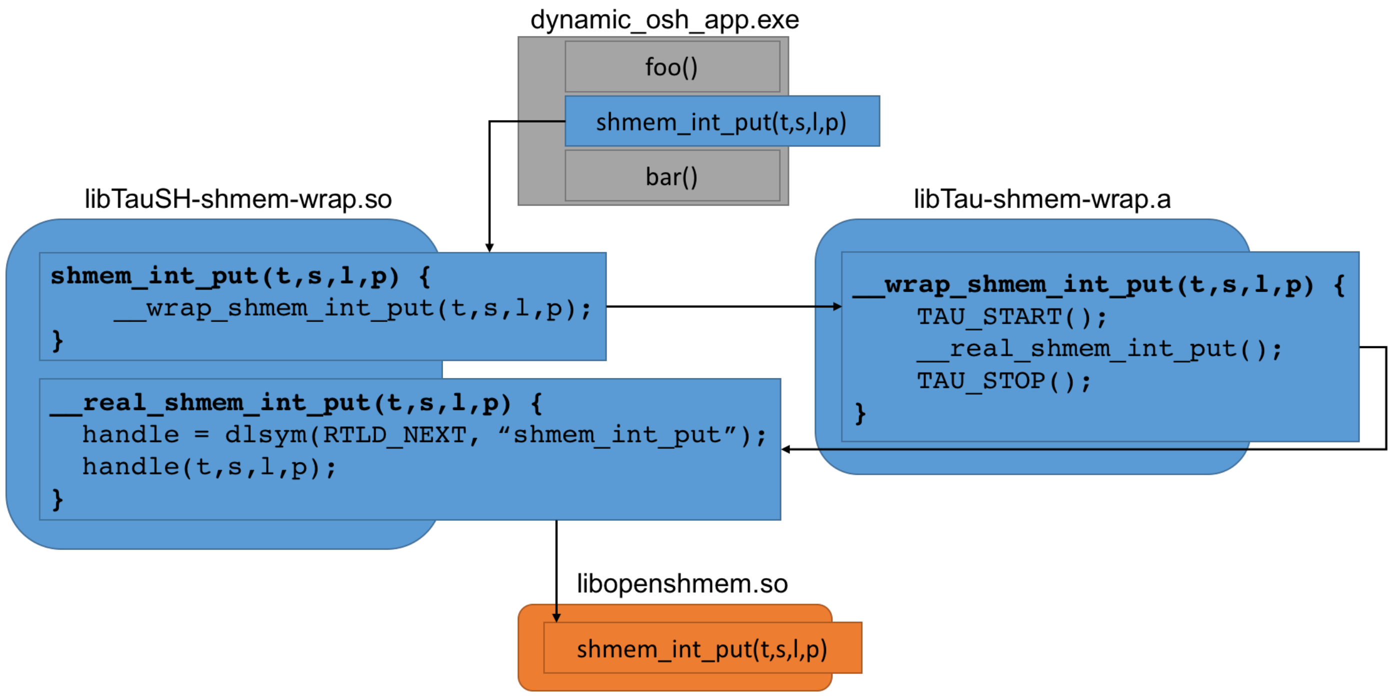
- Use the `–wrap` linker option
- Undefined reference to “`sym`” resolves to “`__wrap_sym`”
- Undefined reference to “`__real_sym`” resolves to “`sym`”
- Use TAU compiler wrapper scripts with -optLinkOnly, e.g.

```
tau_ftn.sh \
    -optLinkOnly \
    -o static.exe *.o
```



```
__wrap_shmem_int_put(t,s,l,p) {
    TAU_START();
    __real_shmem_int_put();
    TAU_STOP();
}
```

Library Preloading



```
shmem_int_put(t,s,l,p) {
     __wrap_shmem_int_put(t,s,l,p);
}
```

```
__real_shmem_int_put(t,s,l,p) {
   handle = dlsym(RTLD_NEXT, “shmem_int_put”);
   handle(t,s,l,p);
}
```

```
__wrap_shmem_int_put(t,s,l,p) {
     TAU_START();
     __real_shmem_int_put();
     TAU_STOP();
}
```

- Unresolved reference to “`sym`” binds to “`sym`” in wrapper library, which calls “`__wrap_sym`” implemented in the static wrapper.
- Unresolved reference to “`__real_sym`” in static wrapper binds to “`__real_sym`” defined in dynamic wrapper.
- “`__real_sym`” uses dynamic linker API to locate “`sym`” and call it.
- Use tau_exec:
  - **`oshrun -np 256 tau_exec –T serial -shmem ./foo.exe`**

Acknowledgments

- John Linford, Nicholas Chainov, Izaak Beekman, Samuel Khuvis, Srinath Vadlamani, Allen Malony, ParaTools, Inc.
- Neena Imam, Manjunath Gorentla Venkata, ORNL
- This work was supported by the United States Department of Defense (DoD) and used resources of the Computational Research and Development Programs and the Oak Ridge Leadership Computing Facility (OLCF) at the Oak Ridge National Laboratory.