OpenSHMEM Profiling Using the TAU Performance System®

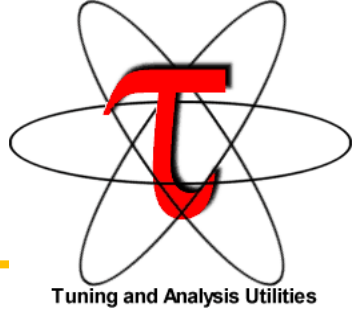
OpenSHMEM BOF, SC'11, TCC 303, Nov. 16, 2011, 5:30pm

Sameer Shende

ParaTools, Inc.

http://tau.uoregon.edu/

Introduction: TAU Performance System



- http://tau.uoregon.edu/
- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-ported parallel performance profiling system
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, mixed-mode, hybrid
- Integration in complex software, systems, applications

What is TAU?

- TAU is a performance evaluation tool
- It supports parallel profiling and tracing
- Profiling shows you how much (total) time was spent in each routine
- Tracing shows you *when* the events take place in each process along a timeline
- TAU uses a package called PDT for automatic instrumentation of the source code
- Profiling and tracing can measure time as well as hardware performance counters from your CPU
- TAU can automatically instrument your source code (routines, loops, I/O, memory, phases, etc.)
- TAU runs on all HPC platforms and it is free (BSD style license)
- TAU has instrumentation, measurement and analysis tools
  - paraprof is TAU's 3D profile browser
- To use TAU's automatic source instrumentation, you need to set a couple of environment variables and substitute the name of your compiler with a TAU shell script

Using TAU with source instrumentation

- TAU supports several measurement options (profiling, tracing, profiling with hardware counters, etc.)
- Each measurement configuration of TAU corresponds to a unique stub makefile and library that is generated when you configure it
- To instrument source code using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:

  % export TAU_MAKEFILE=$TAU/Makefile.tau-shmem-pdt
  % export TAU_OPTIONS='-optVerbose …' (see tau_compiler.sh -help)

  And use tau_f90.sh, tau_cxx.sh or tau_cc.sh as Fortran, C++ or C compilers:

  % mpicc foo.c

  changes to

  % tau_cc.sh foo.c
- Execute application and analyze performance data:

  % pprof (for text based profile display)
  % paraprof (for GUI)

TAU Measurement Configuration

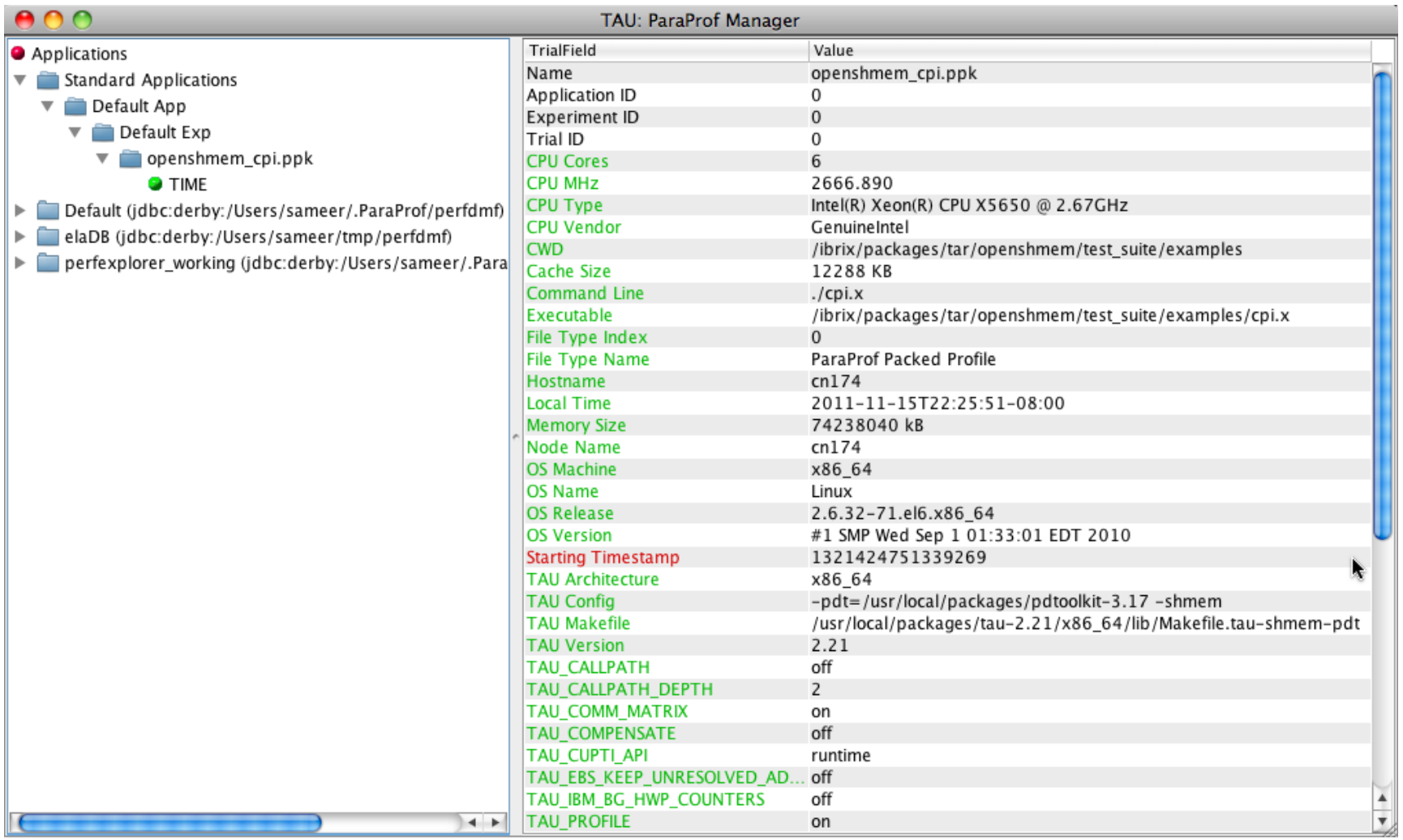
```
% cd $TAU; ls Makefile.*
Makefile.tau-pdt
Makefile.tau-mpi-pdt
Makefile.tau-shmem-pdt
Makefile.tau-mpi-openmp-pdt
Makefile.tau-papi-mpi-pdt
Makefile.tau-papi-pthread-pdt
```

- For a SHMEM application, you may want to start with:

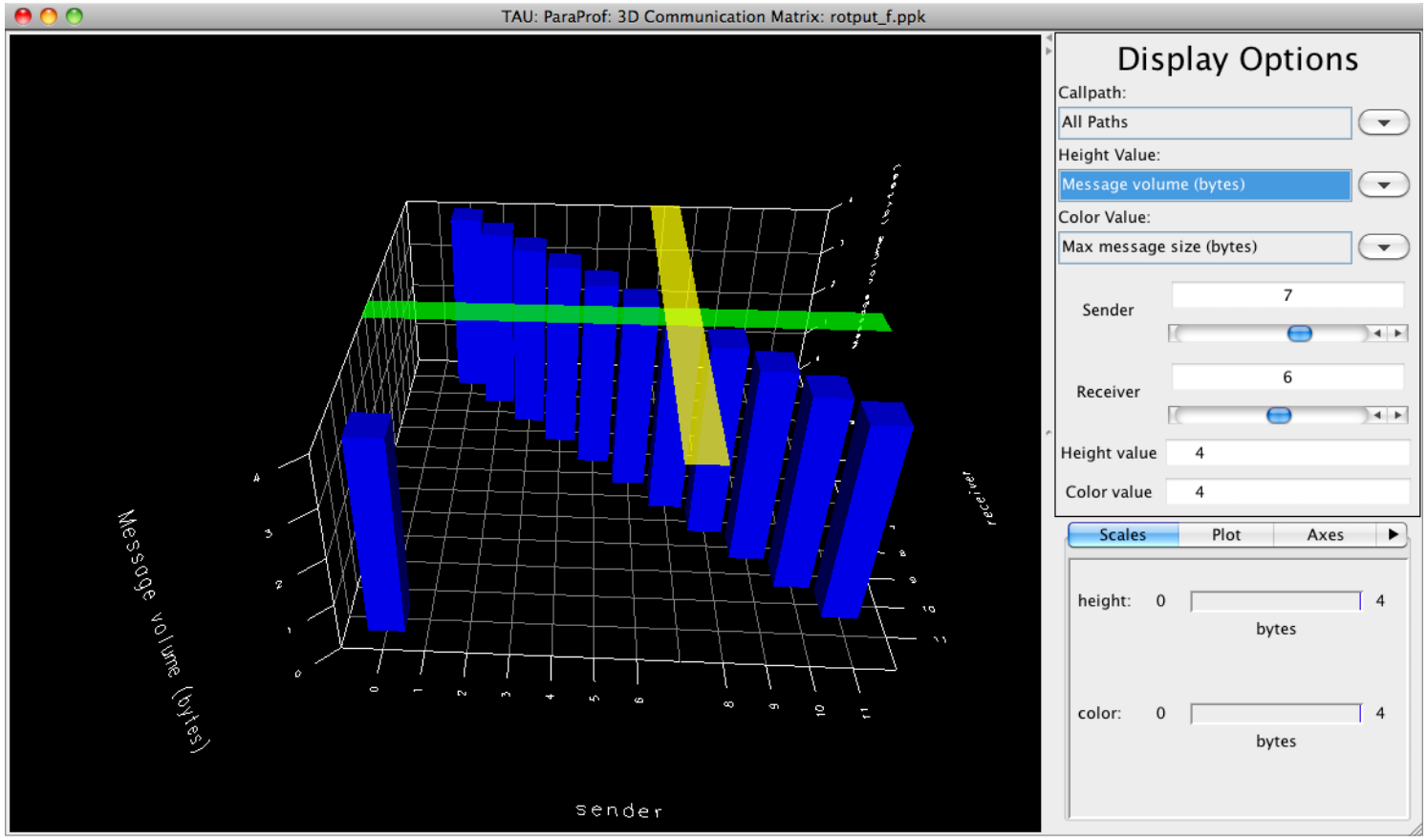
Makefile.tau-shmem-pdt

- Supports MPI instrumentation & PDT for automatic source instrumentation
- `% export TAU_MAKEFILE=$TAU/Makefile.tau-shmem-pdt`
- `% tau_cc.sh cpi.c -o cpi.x`
- `% oshrun -np 256 ./cpi.x`
- `% paraprof`

TAU's ParaProf Profile Browser



TAU's Communication Matrix Display: OpenSHMEM rotput.f

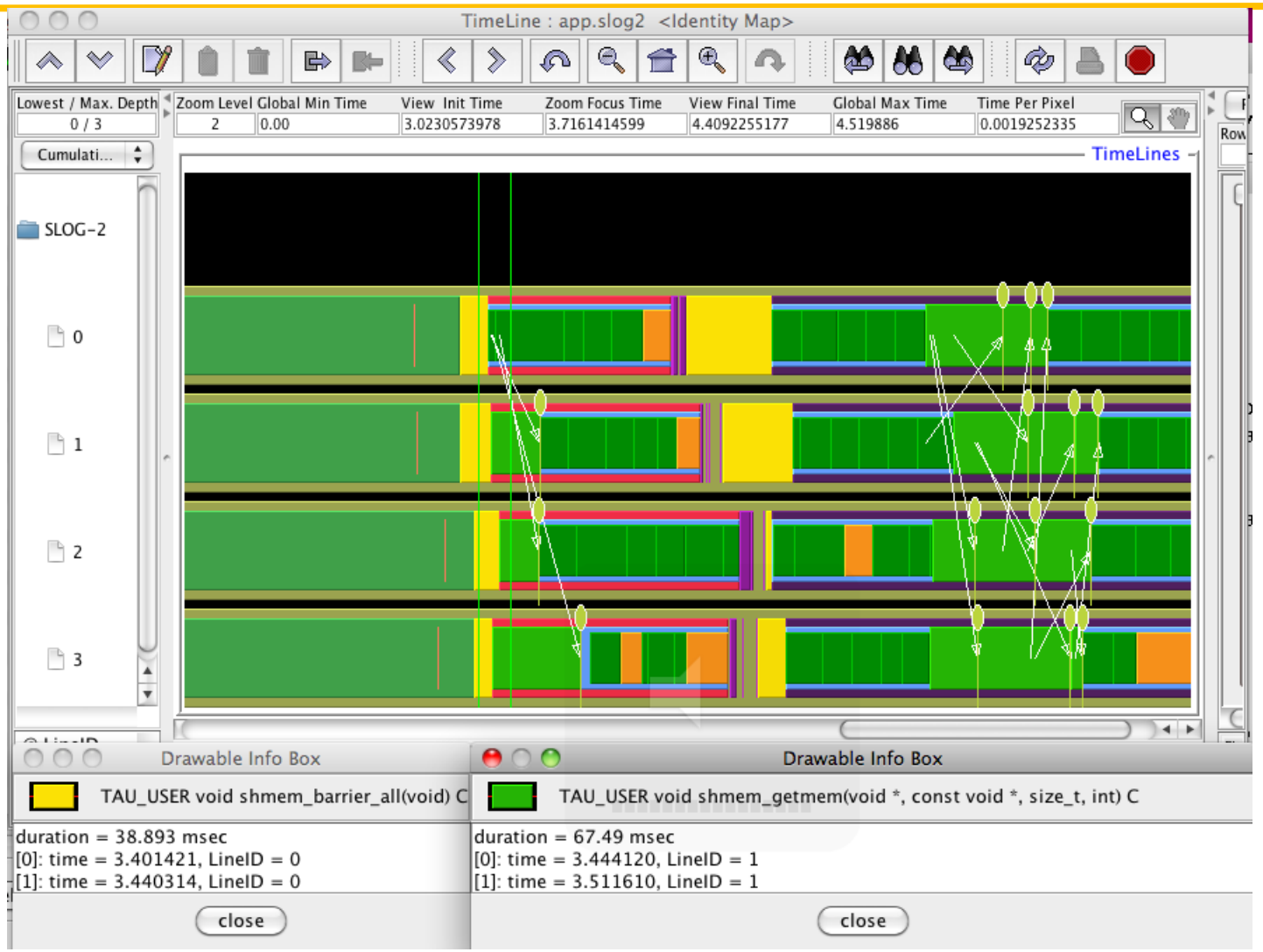


OpenSHMEM Profile: CPI Testcase, PE 0

TAU: ParaProf: Thread Statistics: n,c,t, 0,0,0 – openshmem_cpi.ppk

Name	Exclusive TIME △	Inclusive TIME	Calls	Child Calls
int shmem_version(int *, int *) C	2	2	1	0
void shmem_broadcast32(void *, const void *, size_t, int, int, int, int, long *) C	5	32,947	1	1
void shmem_barrier(int, int, int, long *) C	23	34,638	3	72
void shmem_double_sum_to_all(double *, double *, int, int, int, int, double *, long *) C	24	7,507	1	13
double f(double) C [{cpi.c} {56,1}-{60,1}]	73	73	834	0
void shmem_barrier_all(void) C	161	161	2	0
void shmem_long_wait_until(long *, int, long) C	163	163	6	0
int main(int, char **) C [{cpi.c} {76,1}-{136,1}]	179	7,378,225	1	839
void shmem_getmem(void *, const void *, size_t, int) C	5,787	5,787	11	0
void shmem_long_inc(long *, int) C	34,452	34,452	66	0
void start_pes(int) C	7,337,356	7,337,358	1	1

Jumpshot Trace Visualizer: CPI with OpenSHMEM



Building and Using TAU on Cray XE6 with Cray SHMEM

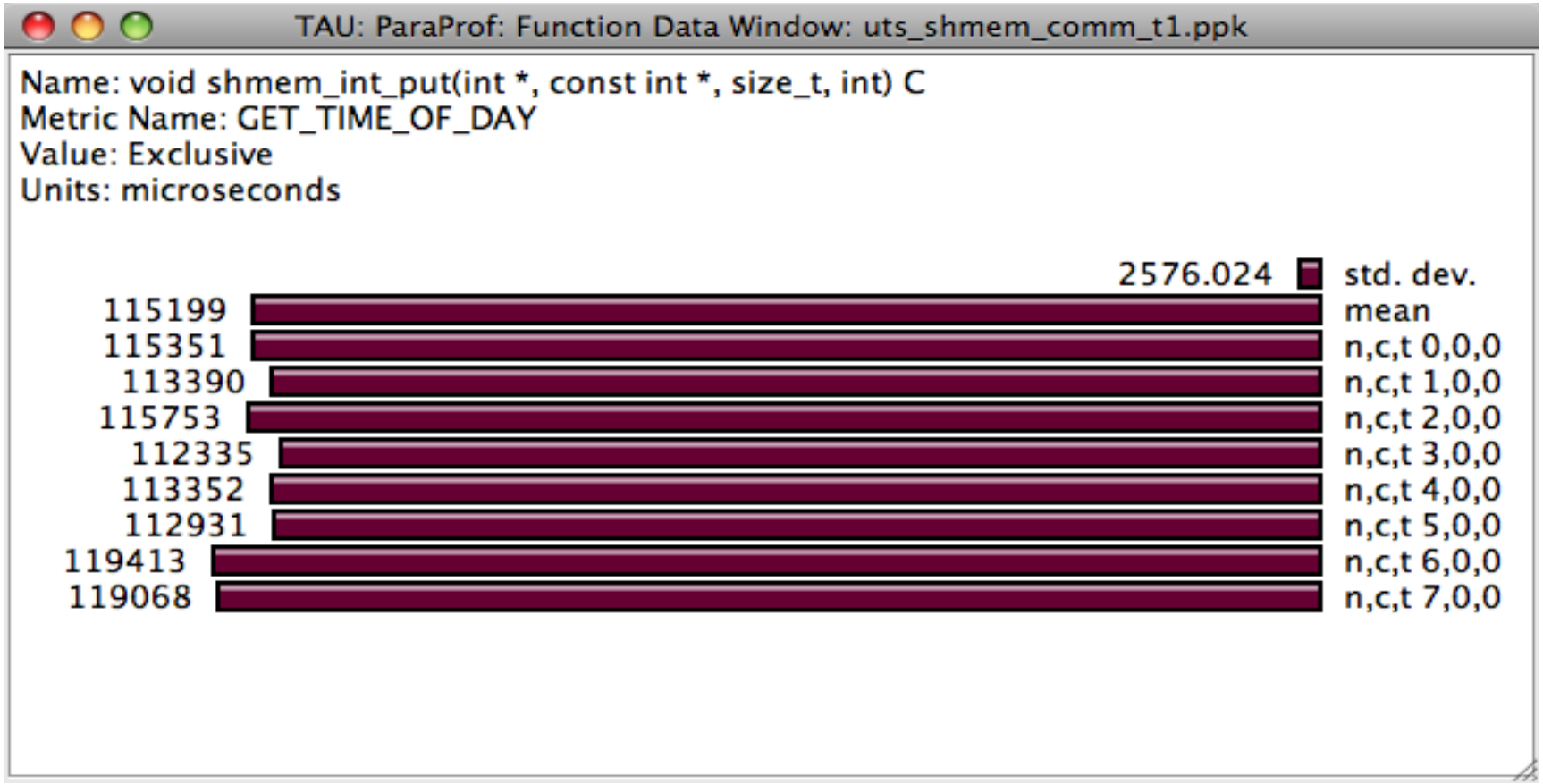
- Configure TAU:
  - ./configure –bfd=download –pdt=<dir> -shmem –arch=craycnl –pdt_c++=/usr/bin/g++
  - make install
- Compiling:
  - setenv TAU_MAKEFILE $TAUDIR/craycnl/lib/Makefile.tau-shmem-pdt-pgi
  - set path=($TAUDIR/craycnl/bin $path)
  - make CC=tau_cc.sh CXX=tau_cxx.sh
  - aprun –n 4 ./a.out

SHMEM Profiling in UTS-1.1

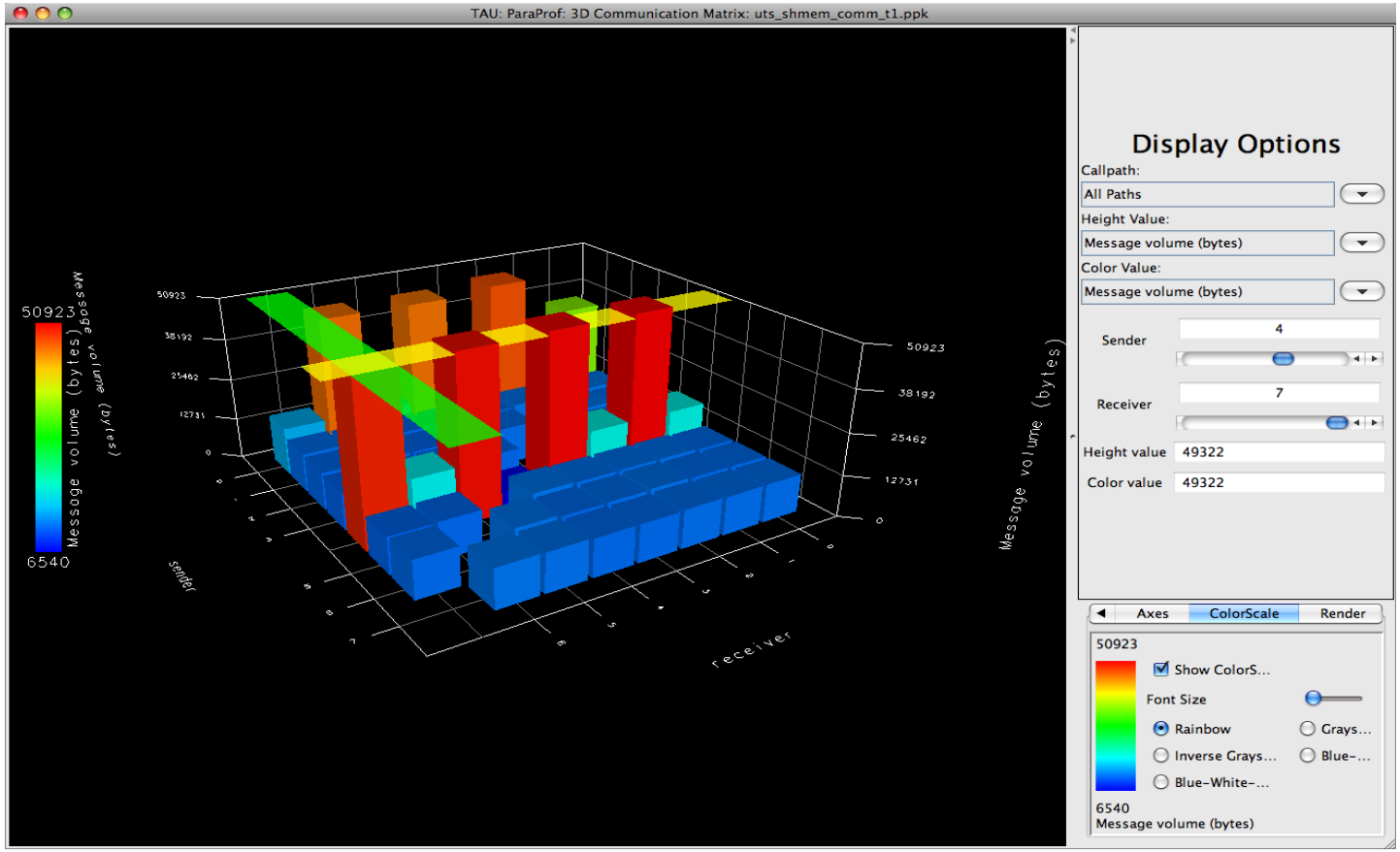
TAU: ParaProf: Mean Statistics – uts_shmem_comm_t1.ppk

Name	Exclusive GET_TIME_O...	Inclusive GET_TIME_OF...	Calls	Child Calls
void uts_showStats(int, int, double, counter_t, counter_t, counter_t) C [{uts.c} {442,1}-{456,1}]	52.375	52.375	0.125	0
void uts_printParams() C [{uts.c} {343,1}-{352,1}]	206	209.625	0.125	0.25
void uts_parseParams(int, char **) C [{uts.c} {354,1}-{415,1}]	22	23.625	1	6
void uts_initRoot(Node *, int) C [{uts.c} {151,1}-{159,1}]	0.75	4.5	0.125	0.125
void start_pes(int) C	668,097.125	668,097.125	1	0
void ss_setState(StealStack *, int) C [{uts_shm.c} {726,1}-{759,1}] [THROTTLED]	19,987.375	20,260.375	100,001	958.5
void ss_release(StealStack *, int) C [{uts_shm.c} {572,1}-{582,1}]	1,307.875	29,691.5	1,862	3,724
void ss_push(StealStack *, Node *) C [{uts_shm.c} {519,1}-{530,1}] [THROTTLED]	21,139.25	21,139.25	100,001	0
void ss_pop(StealStack *) C [{uts_shm.c} {546,1}-{556,1}] [THROTTLED]	20,481	20,481	100,001	0
void ss_mkEmpty(StealStack *) C [{uts_shm.c} {462,1}-{469,1}]	5.25	1,154.5	8	16
void ss_initState(StealStack *) C [{uts_shm.c} {713,1}-{724,1}]	17.125	17.25	1	1
void ss_init(StealStack *, int) C [{uts_shm.c} {478,1}-{515,1}]	99,443.75	100,819.875	8	16
void showStats(double) C [{uts_shm.c} {1282,1}-{1403,1}]	68.375	125.375	0.125	5.625
void shmem_set_lock(long *) C	88,867.625	144,531.25	5,516.875	6,246.5
void shmem_putmem(void *, const void *, size_t, int) C	10,957.25	10,957.25	1,459.25	0
void shmem_int_put(int *, const int *, size_t, int) C	115,199.125	115,199.125	19,669.875	0
void shmem_int_get(int *, const int *, size_t, int) C	25,410.25	25,410.25	4,666.75	0
void shmem_getmem(void *, const void *, size_t, int) C	27,133.875	27,133.875	1,027.25	0
void shmem_clear_lock(long *) C	4,172.5	39,560.25	5,516.875	5,587.25
void shmem_barrier_all() C	299.875	299.875	2	0
void sha1_hash(const unsigned char *, unsigned long, sha1_ctx *) C [{brg_sha1.c} {264,1}-{281,1}] [THROTTLED]	21,002.5	21,002.5	100,001	0
void sha1_end(unsigned char *, sha1_ctx *) C [{brg_sha1.c} {285,1}-{327,1}] [THROTTLED]	50,766.375	99,133.875	100,001	100,001
void sha1_compile(sha1_ctx *) C [{brg_sha1.c} {195,1}-{249,1}] [THROTTLED]	48,367.5	48,367.5	100,001	0
void sha1_begin(sha1_ctx *) C [{brg_sha1.c} {251,1}-{259,1}] [THROTTLED]	20,316.125	20,316.125	100,001	0
void rng_spawn(RNG_state *, RNG_state *, int) C [{brg_sha1.c} {67,1}-{81,1}] [THROTTLED]	94,972.875	235,423.875	100,001	300,002.625
void rng_init(RNG_state *, int) C [{brg_sha1.c} {49,1}-{65,1}]	2.25	3.75	0.125	0.375
void releaseNodes(StealStack *) C [{uts_shm.c} {1137,1}-{1158,1}] [THROTTLED]	36,437.5	56,426.875	100,001	60,529.75
void parTreeSearch(StealStack *) C [{uts_shm.c} {1166,1}-{1237,1}]	1,041,502.5	2,161,392.5	1	519,294.5
void initRootNode(Node *, int) C [{uts_shm.c} {951,1}-{984,1}]	1.375	5.875	0.125	0.125
void initNode(Node *) C [{uts_shm.c} {929,1}-{948,1}]	0.5	0.5	1	0

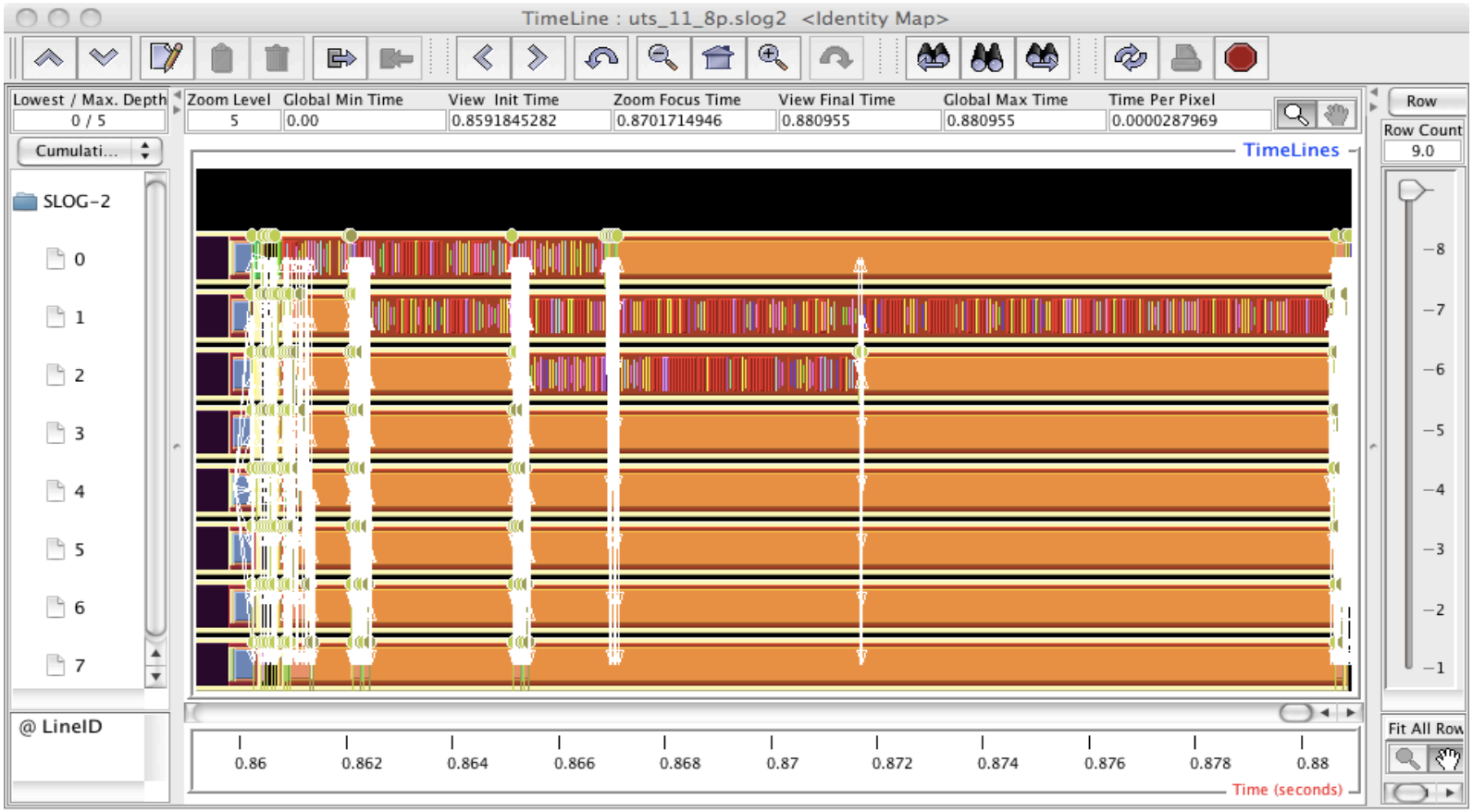
SHMEM Wrapper Instrumentation in UTS-1.1



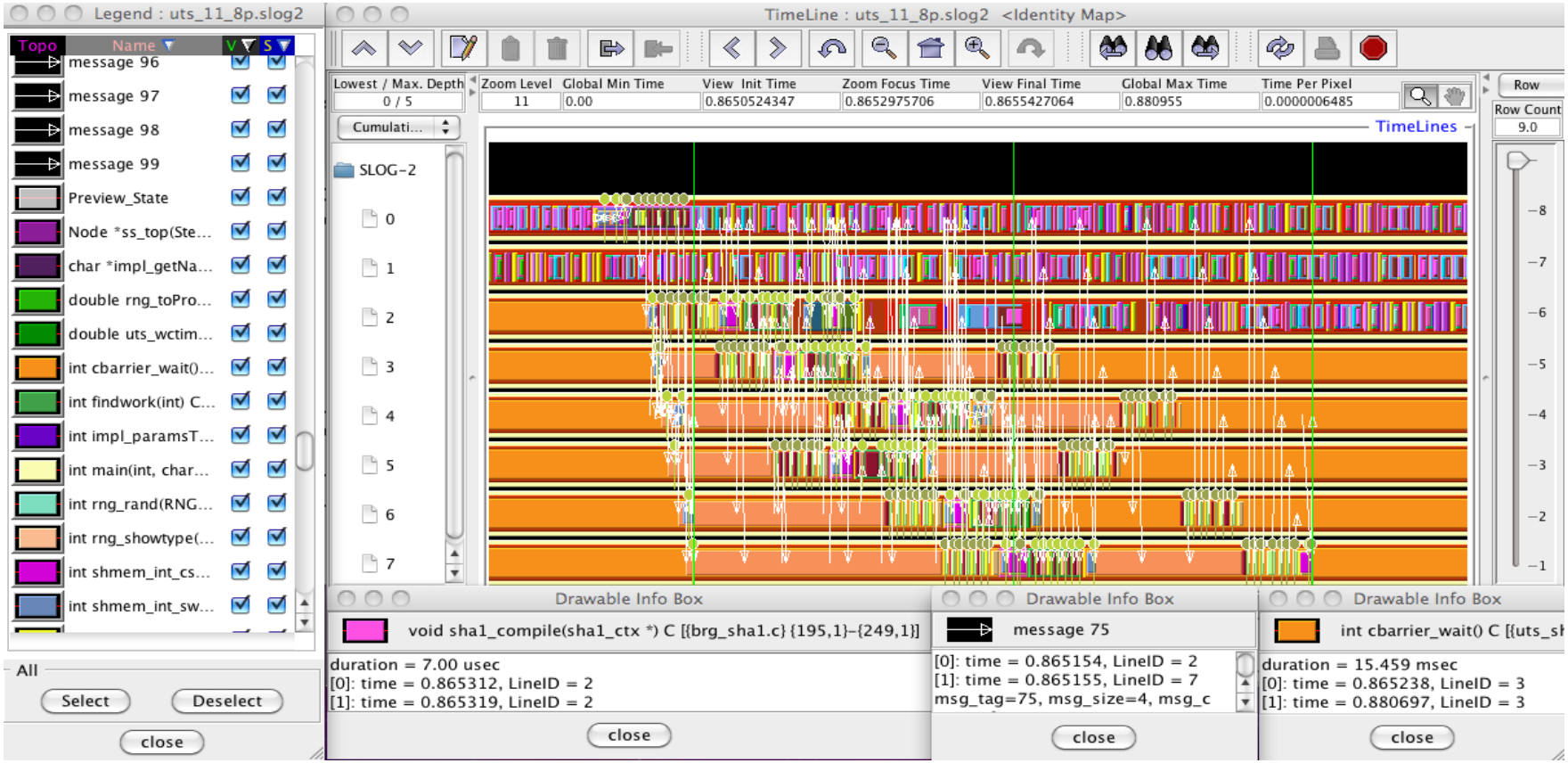
Communication Matrix (TAU_COMM_MATRIX=1)



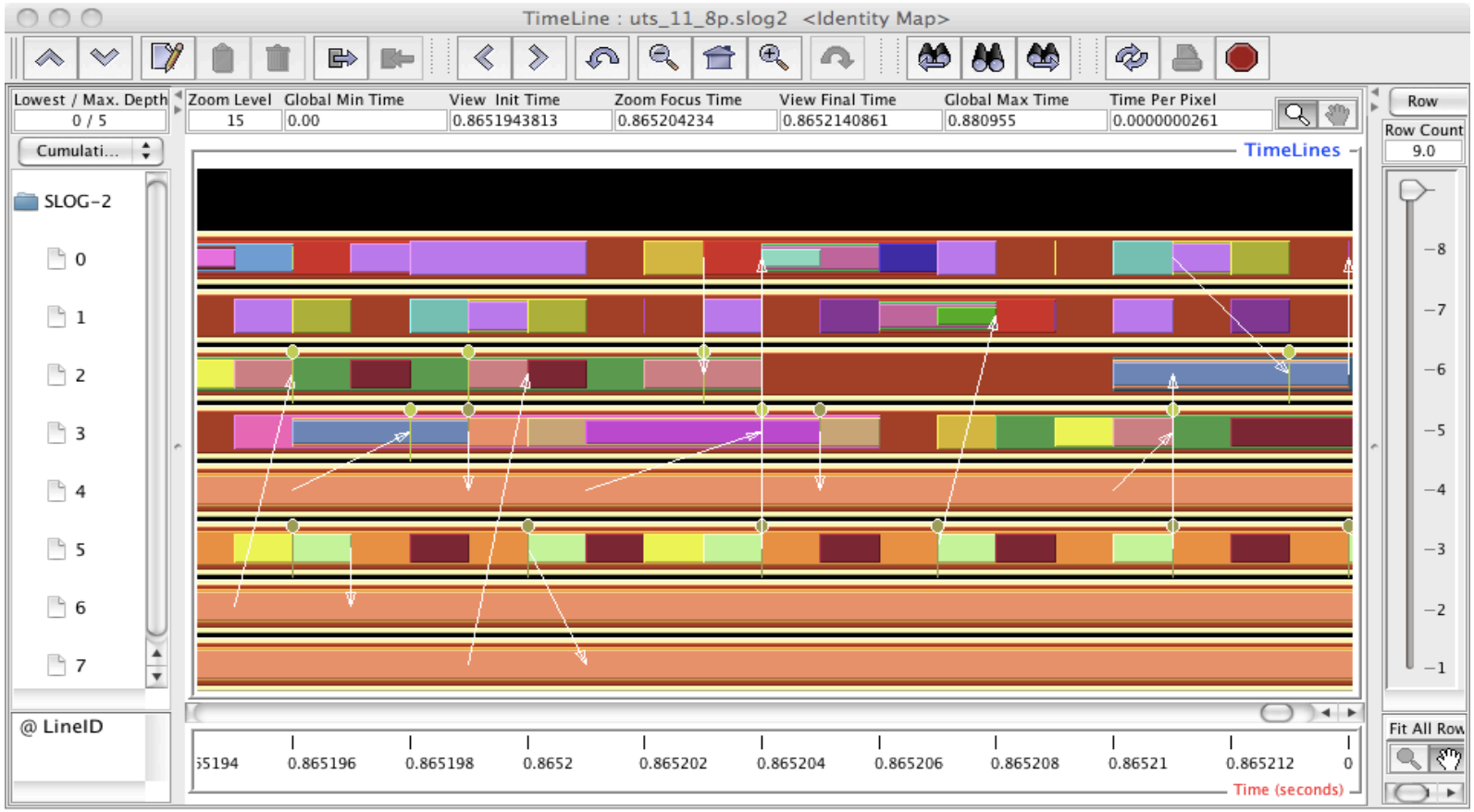
Tracing UTS 1.1: Jumpshot



Tracing UTS 1.1: Jumpshot



Tracing: Zooming in



LiveDVD [http://www.hpclinux.com]

To profile a code using TAU:

1. Choose TAU stub makefile
   % export TAU_MAKEFILE=$TAU/Makefile.tau-[options]
2. Change the compiler name to tau_cxx.sh, tau_f90.sh, tau_cc.sh:
   % make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
3. If stub makefile has -papi in its name, set the TAU_METRICS environment variable:
   % export TAU_METRICS=TIME:PAPI_L2_DCM:PAPI_TOT_CYC...
4. Execute the application:
   % oshrun -np 4 ./a.out
5. Build and run workshop examples, then run pprof/paraprof

TAU Support Acknowledgements

- Department of Energy (DOE)
  - Office of Science contracts
  - SciDAC contracts, LBL
  - LLNL-LANL-SNL ASC/NNSA contract
  - Battelle, PNNL contract
- Department of Defense (DoD)
  - PETTT, HPTi
- National Science Foundation (NSF)
  - POINT, SI-2