Parallel Performance Optimization Using TAU Training and Workshop

Sameer Shende

ParaTools, Inc. and University of Oregon.

sameer@paratools.com

http://www.paratools.com/TAU

Download slides and workshop tarball from:

http://tau.uoregon.edu/tau_llnl18.pdf

quartz.llnl.gov: /usr/global/tools/tau/training/workshop.tgz

Serrano.sandia.gov: /projects/tau/workshop.tgz

LANL: /turquoise/usr/projects/packages/tau/workshop.tgz

Outline

Day 1:

Morning:

- Introduction to TAU
- Instrumentation: PDT, MPI, OpenMP OMPT, tau_exec
- I/O, and memory evaluation
- Hands-on using paraprof
- PAPI
- Hands-on using loop level instrumentation, PAPI

Afternoon

- Demonstration of analysis tools: Paraprof, TAUdb and PerfExplorer
- Vampir and Jumpshot
- Hands-on

Breaks

- 10:30am – 10:45am, noon – 1:30pm, 3pm – 3:15pm

Day 2:

Individual Sessions:.

- Applying performance evaluation tools to user codes

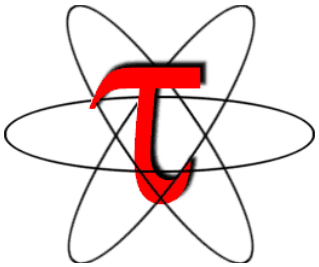
Tutorial Goals

This tutorial is an introduction to portable performance evaluation tools. You should leave here with a better understanding of…

- Concepts and steps involved in performance evaluation
- Understanding key concepts in understanding code performance
- How to collect and analyze data from hardware performance counters (PAPI)
- How to instrument your programs with TAU
- Measurement options provided by TAU
- Environment variables used for choosing metrics, generating performance data
- How to use ParaProf, TAU's profile browser
- General familiarity with TAU use for Fortran, C++, C, and mixed language
- How to generate trace data in different formats

TAU Performance System®

http://tau.uoregon.edu



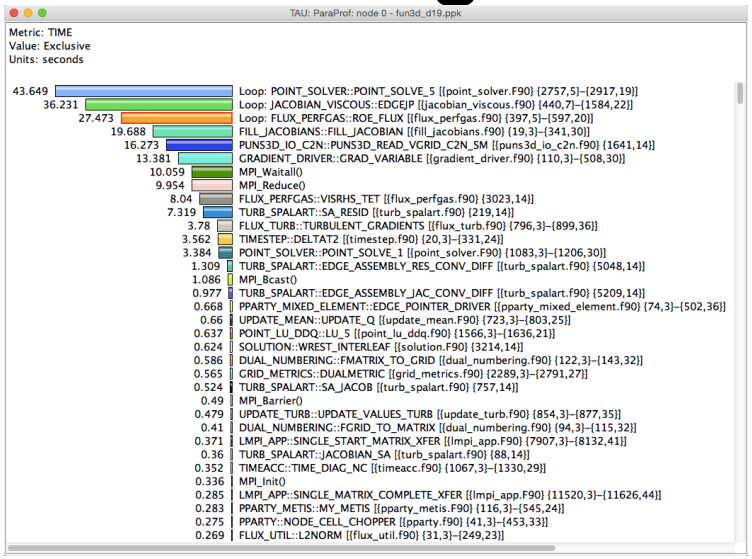
- **Tuning and Analysis Utilities (20+ year project)**
- **Comprehensive performance profiling and tracing**
  - Integrated, scalable, flexible, portable
  - Targets all parallel programming/execution paradigms
- **Integrated performance toolkit**
  - Instrumentation, measurement, analysis, visualization
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
  - Open source (BSD-style license)
- **Integrates with application frameworks**

Understanding Application Performance using TAU

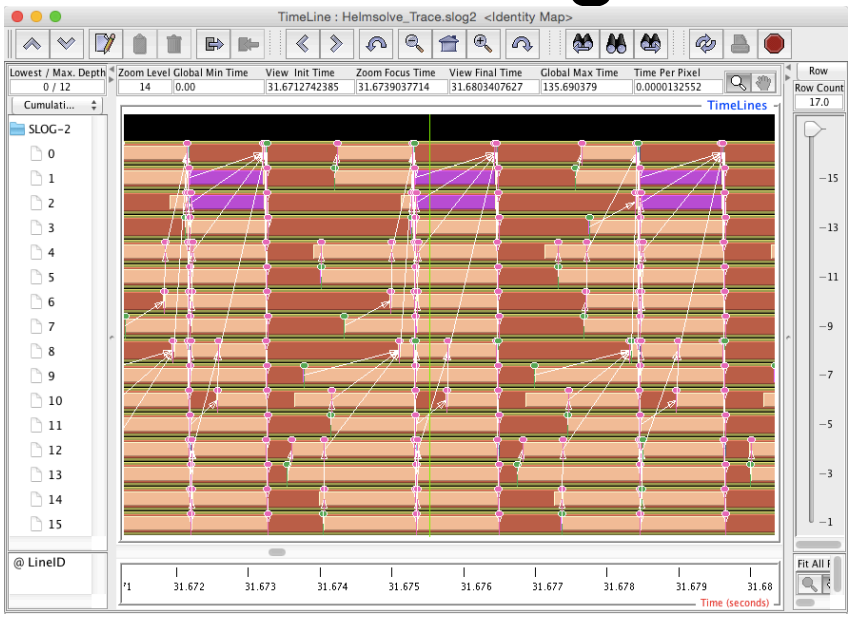
- **How much time** is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*?
- **How many instructions** are executed in these code regions? Floating point, Level 1 and 2 *data cache misses*, hits, branches taken?
- **What is the memory usage** of the code? When and where is memory allocated/de-allocated? Are there any memory leaks?
- **What are the I/O characteristics** of the code? What is the peak read and write *bandwidth* of individual calls, total volume?
- **What is the contribution of each *phase*** of the program? What is the time wasted/spent waiting for collectives, and I/O operations in Initialization, Computation, I/O phases?
- **How does the application *scale***? What is the efficiency, runtime breakdown of performance across different core counts?

Profiling and Tracing

Profiling



Tracing



- Profiling and tracing

  Profiling shows you **how much** (total) time was spent in each routine

  Tracing shows you **when** the events take place on a timeline

Instrumentation

Direct and indirect performance observation

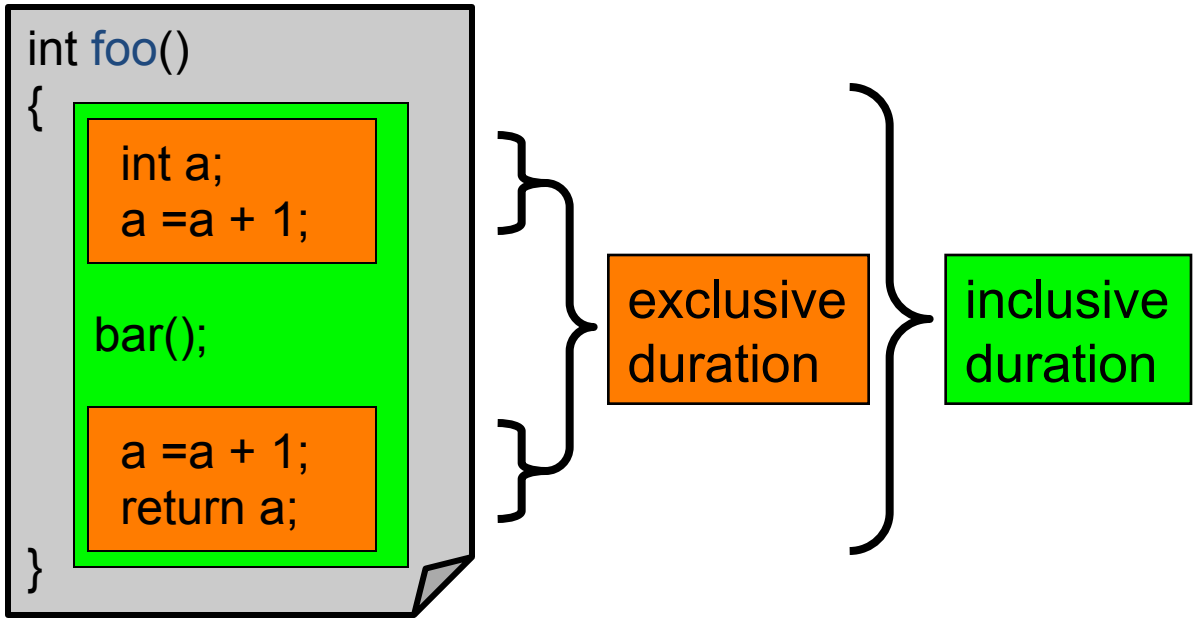
- Instrumentation invokes performance measurement
- Direct measurement with *probes*
- Indirect measurement with periodic sampling or hardware performance counter overflow interrupts
- Events measure performance data, metadata, context, etc.

User-defined events

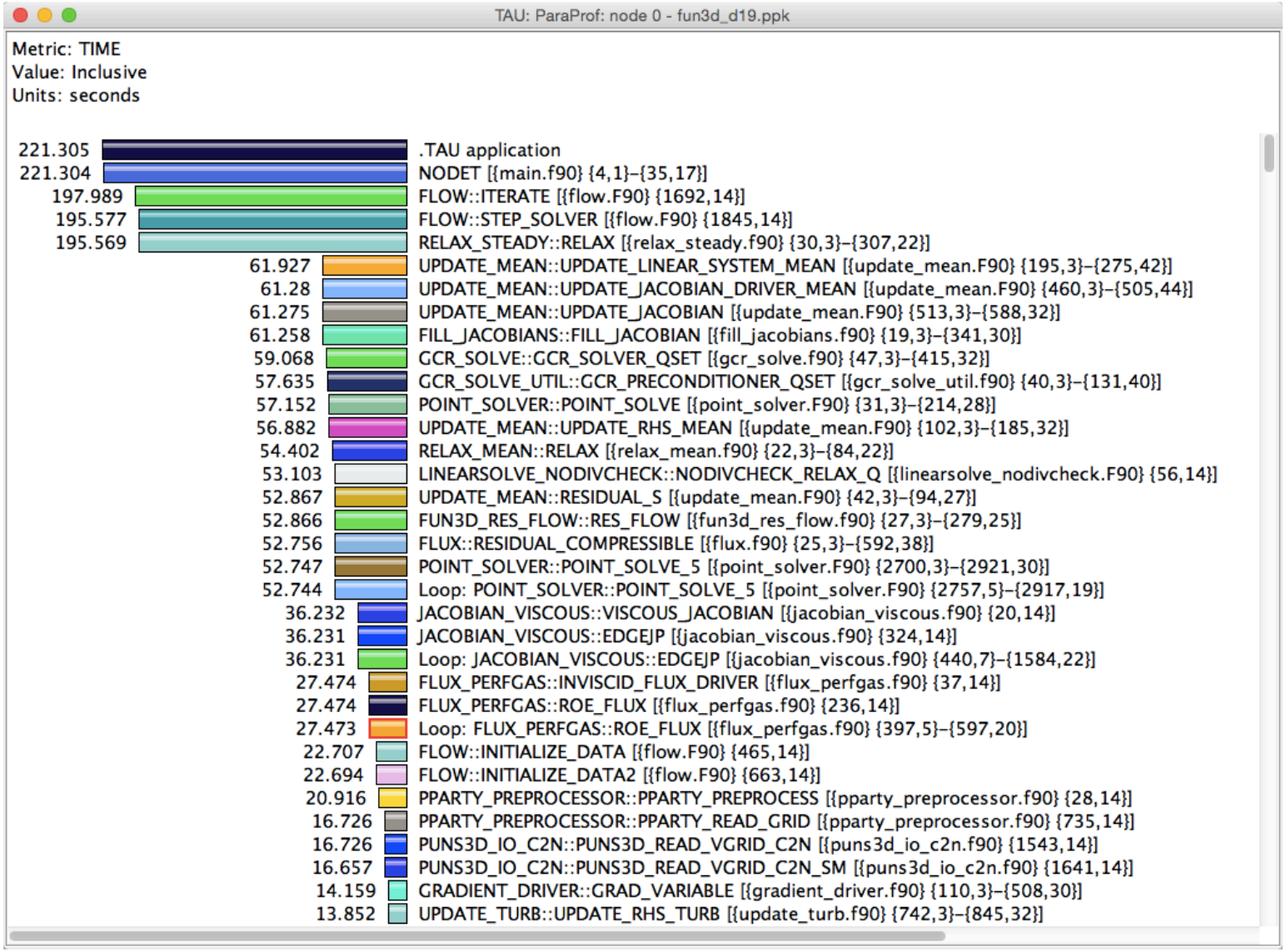
- ***Interval*** (start/stop) events to measure exclusive & inclusive duration
- ***Atomic events*** take measurements at a single point
  - Measures total, samples, min/max/mean/std. deviation statistics
- ***Context events*** are atomic events with executing context
  - Measures above statistics for a given calling path

Inclusive vs. Exclusive Measurements

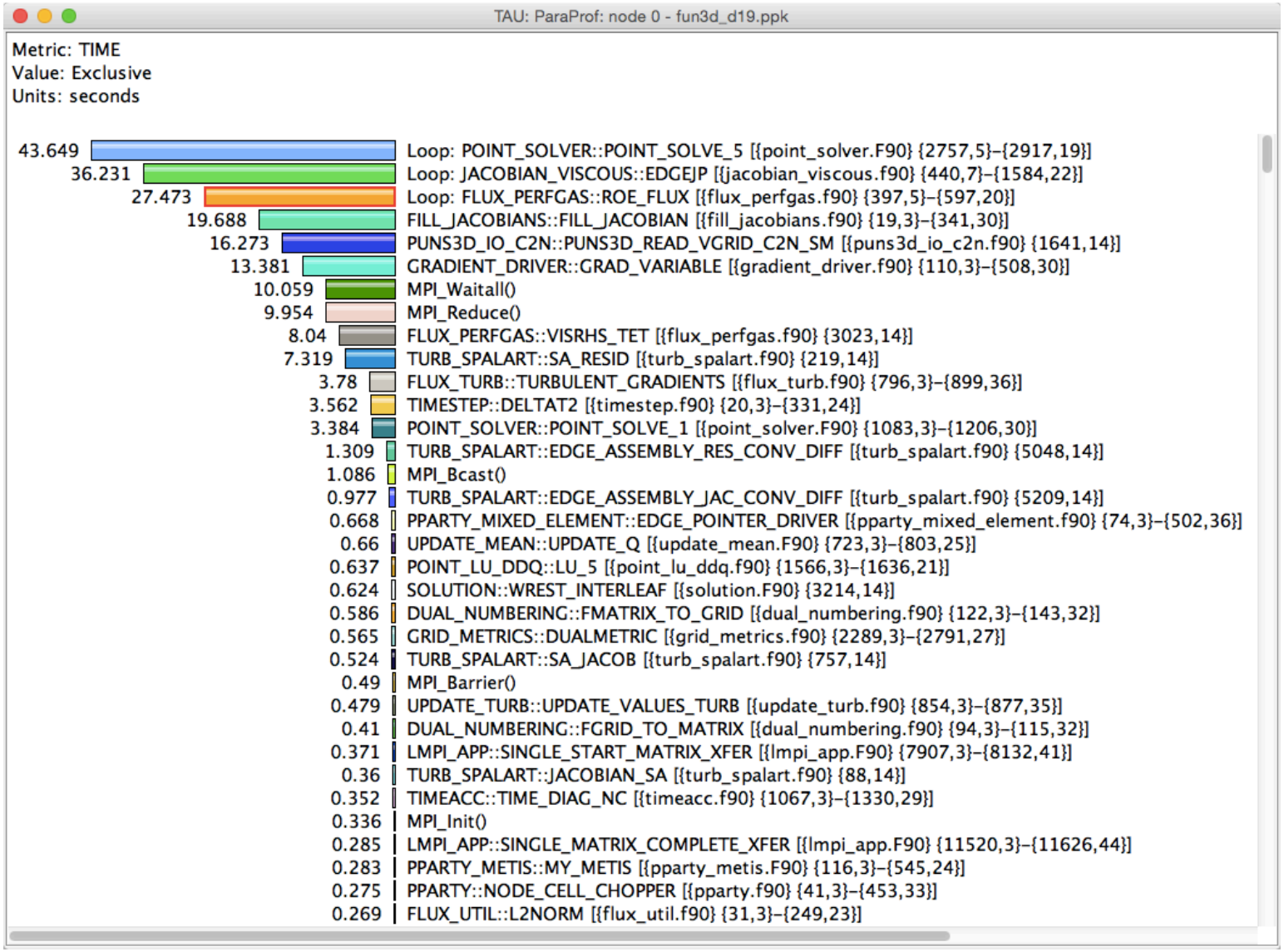
- Performance with respect to code regions
- Exclusive measurements for region only
- Inclusive measurements includes child regions



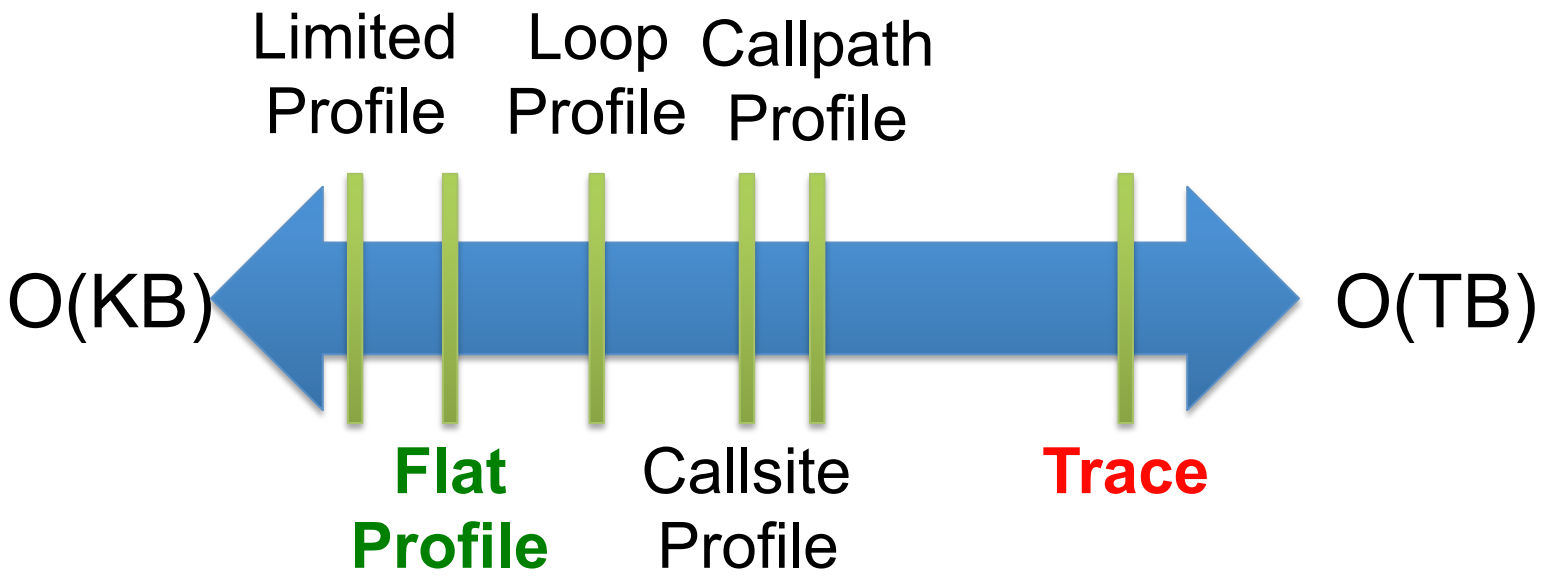
Inclusive Measurements



Exclusive Time



How much data do you want?



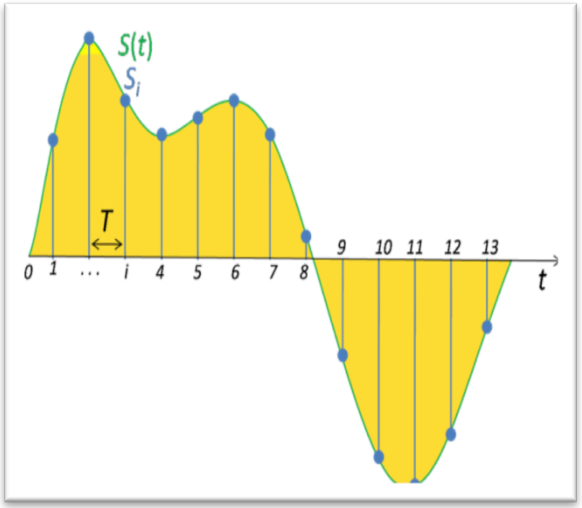
Performance Data Measurement

Direct via Probes

```
Call
START('potential')
// code
Call
STOP('potential')
```

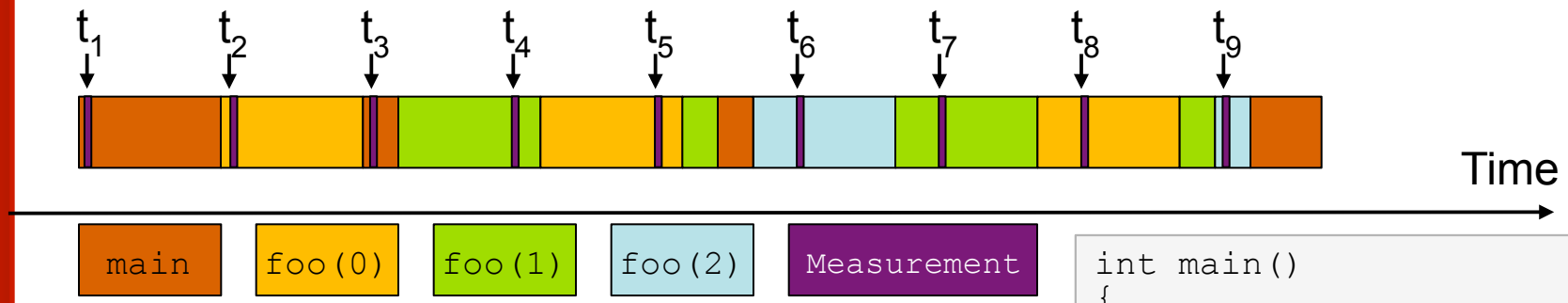
- Exact measurement
- Fine-grain control
- Calls inserted into code

Indirect via Sampling



- No code modification
- Minimal effort
- Relies on debug symbols (**-g**)

Sampling



Running program is periodically interrupted to take measurement

- Timer interrupt, OS signal, or HWC overflow
- Service routine examines return-address stack
- Addresses are mapped to routines using symbol table information

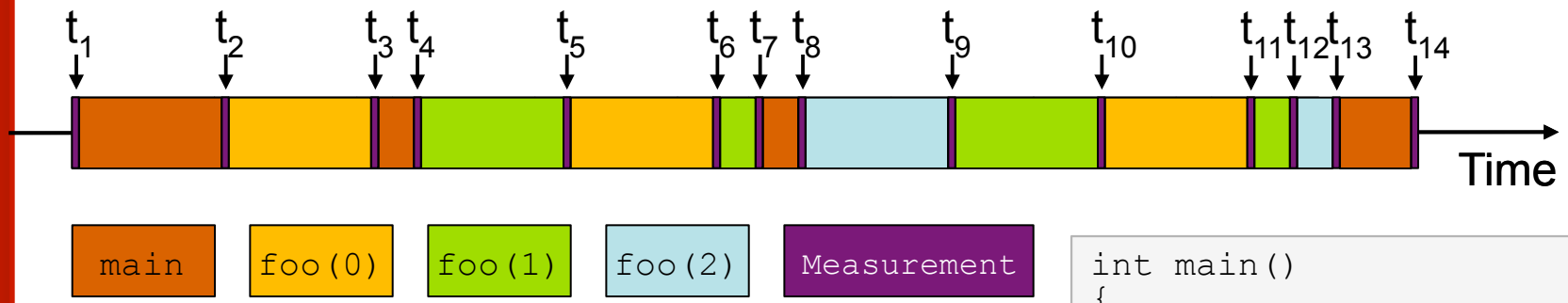
Statistical inference of program behavior

- Not very detailed information on highly volatile metrics
- Requires long-running applications

Works with unmodified executables

```
int main()
{
  int i;

  for (i=0; i < 3; i++)
    foo(i);

  return 0;
}

void foo(int i)
{

  if (i > 0)
    foo(i - 1);

}
```

Instrumentation



Measurement code is inserted such that every event of interest is captured directly

- Can be done in various ways

Advantage:

- Much more detailed information

Disadvantage:

- Processing of source-code / executable necessary
- Large relative overheads for small functions

```
int main()
{
  int i;
  Start("main");
  for (i=0; i < 3; i++)
    foo(i);
  Stop("main");
  return 0;
}

void foo(int i)
{
  Start("foo");
  if (i > 0)
    foo(i - 1);
  Stop("foo");
}
```

Types of Performance Profiles

Flat **profiles**

- Metric (e.g., time) spent in an event
- Exclusive/inclusive, # of calls, child calls, …

Callpath **profiles**

- Time spent along a calling path (edges in callgraph)
- "*main=> f1 => f2 => MPI_Send*"
- Set the TAU_CALLPATH and TAU_CALLPATH_DEPTH environment variables

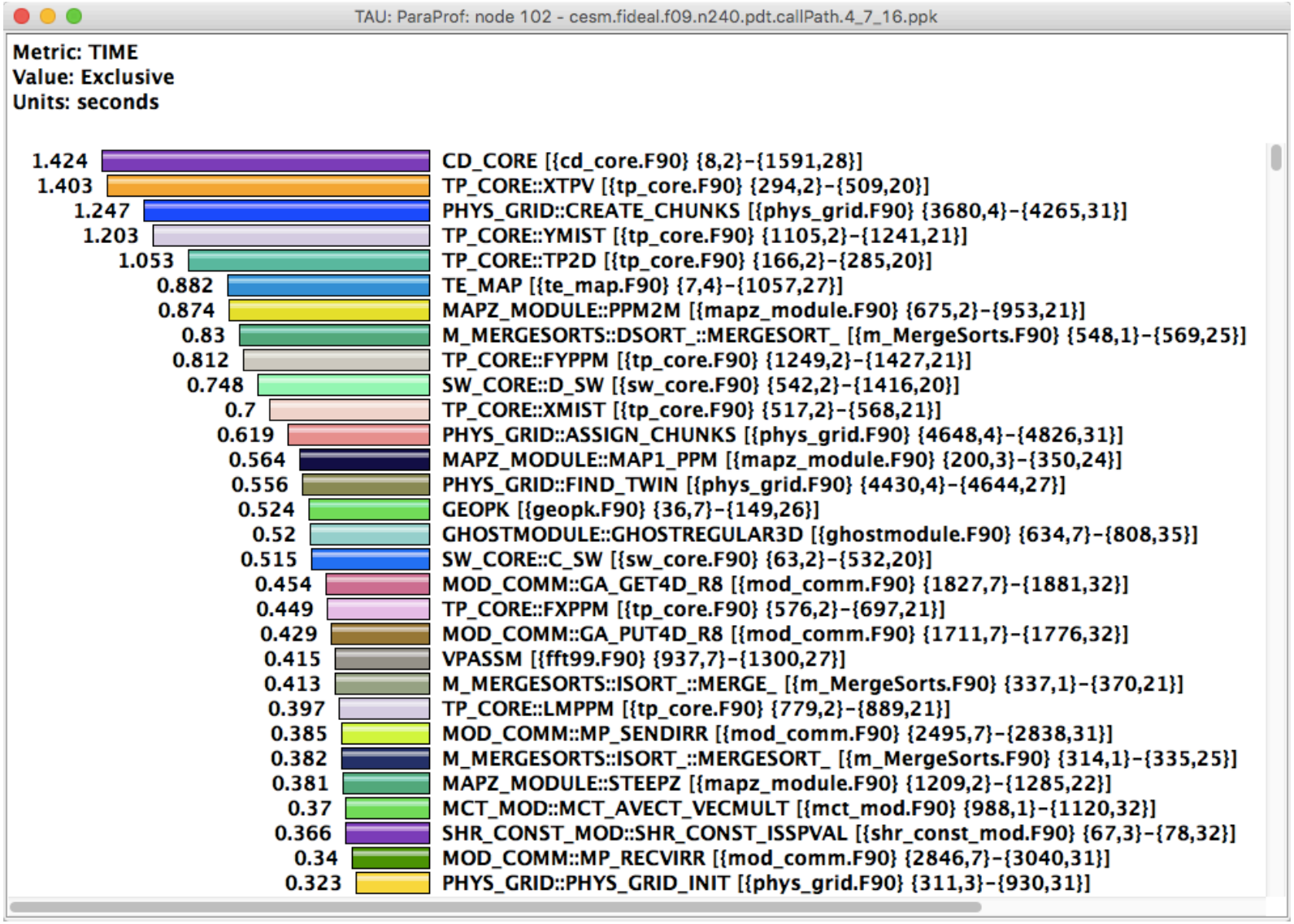
Callsite **profiles**

- Time spent along in an event at a given source location
- Set the TAU_CALLSITE environment variable

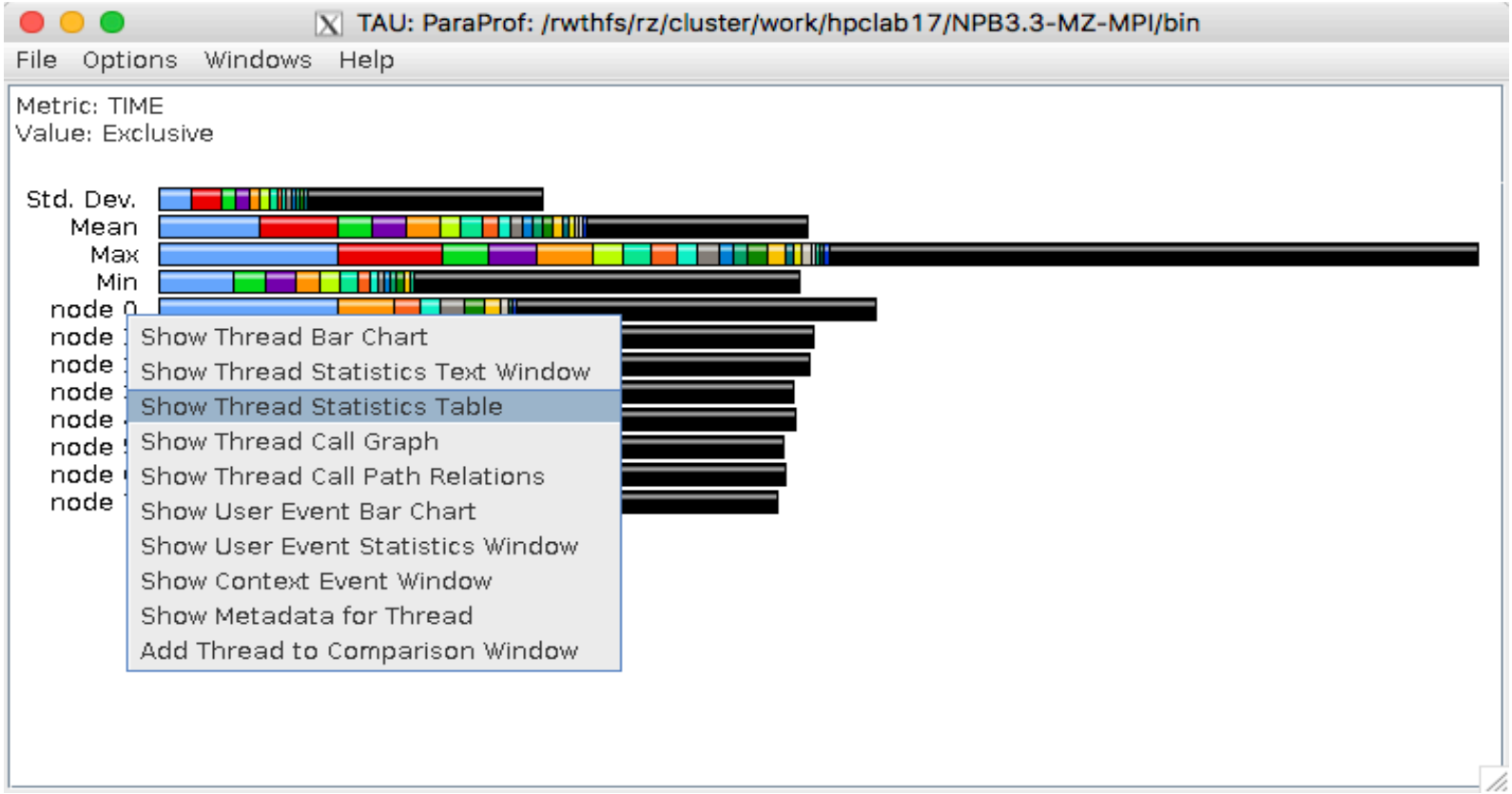
Phase **profiles**

- Flat profiles under a phase (nested phases allowed)
- Default "main" phase
- Supports static or dynamic (e.g. per-iteration) phases

TAU – Flat Profile

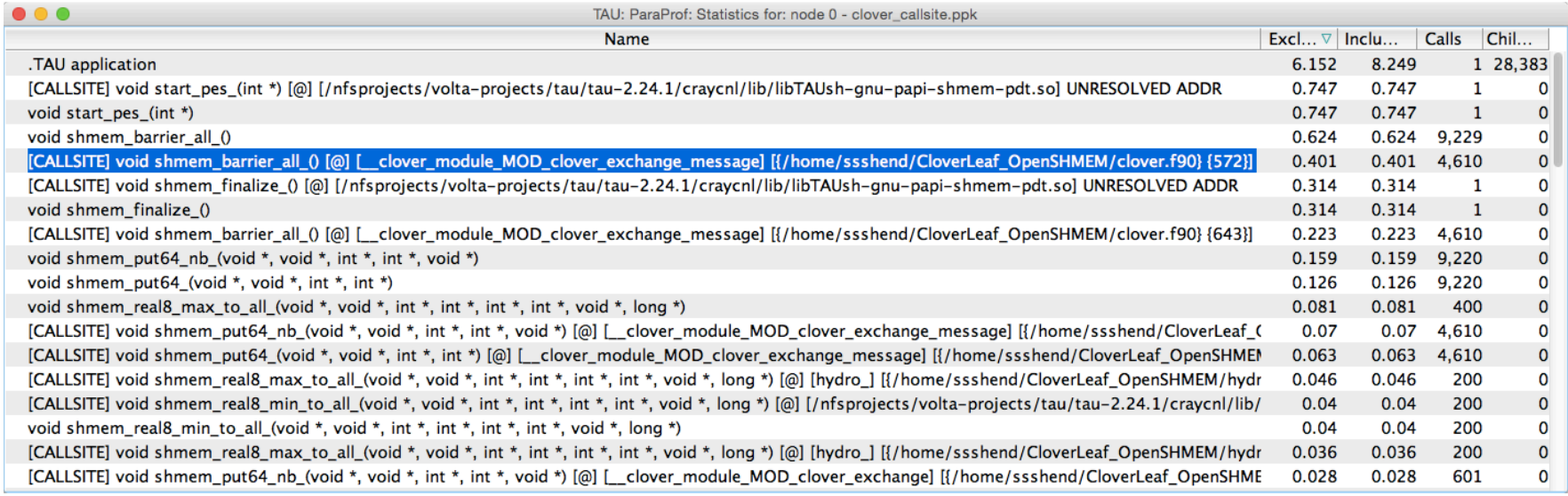


ParaProf Thread Statistics Table



Right click over “node X” and choose
Show Thread Statistics Table

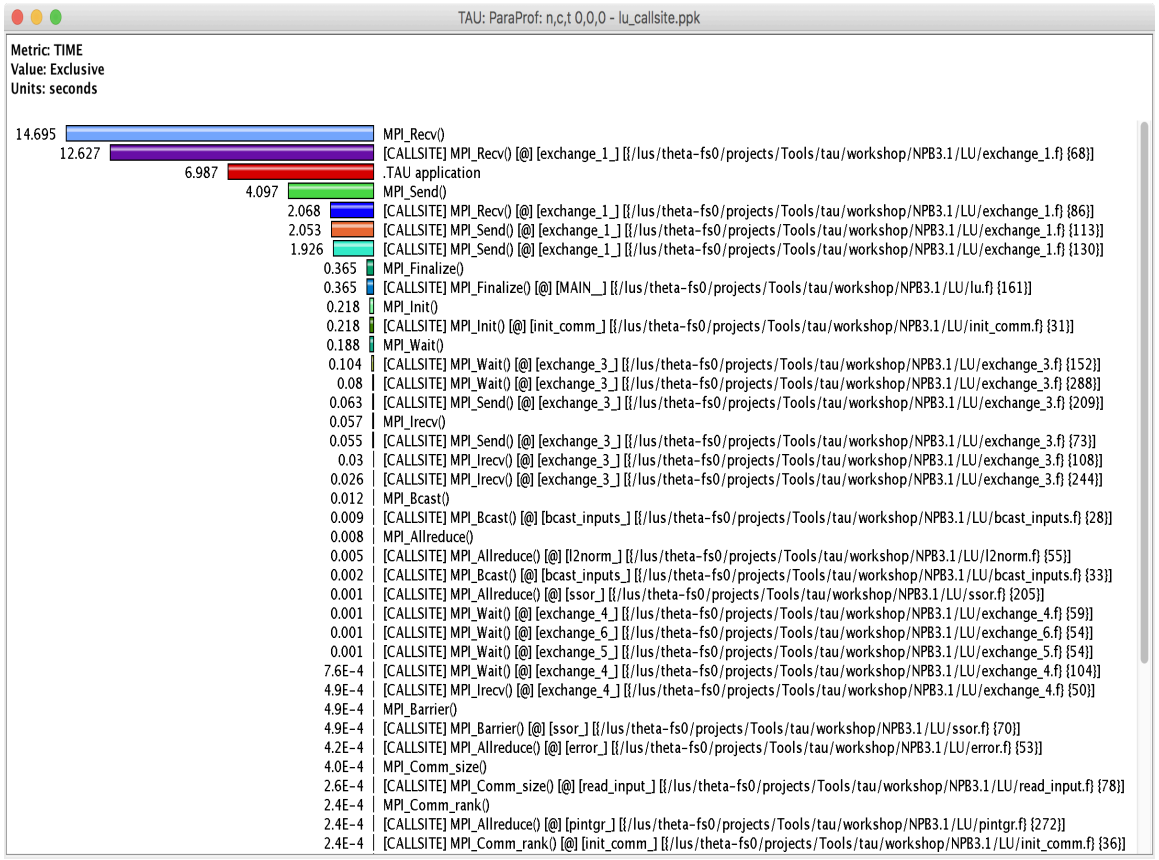
TAU – Callsite Profiling


TAU: ParaProf: Statistics for: node 0 - clover_callsite.ppk

Name	Excl...	Inclu...	Calls	Chil...
.TAU application	6.152	8.249	1	28,383
[CALLSITE] void start_pes_(int *) [@] [/nfsprojects/volta-projects/tau/tau-2.24.1/craycnl/lib/libTAUsh-gnu-papi-shmem-pdt.so] UNRESOLVED ADDR	0.747	0.747	1	0
void start_pes_(int *)	0.747	0.747	1	0
void shmem_barrier_all_()	0.624	0.624	9,229	0
[CALLSITE] void shmem_barrier_all_() [@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90} {572}]	0.401	0.401	4,610	0
[CALLSITE] void shmem_finalize_() [@] [/nfsprojects/volta-projects/tau/tau-2.24.1/craycnl/lib/libTAUsh-gnu-papi-shmem-pdt.so] UNRESOLVED ADDR	0.314	0.314	1	0
void shmem_finalize_()	0.314	0.314	1	0
[CALLSITE] void shmem_barrier_all_() [@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90} {643}]	0.223	0.223	4,610	0
void shmem_put64_nb_(void *, void *, int *, int *, void *)	0.159	0.159	9,220	0
void shmem_put64_(void *, void *, int *, int *)	0.126	0.126	9,220	0
void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.081	0.081	400	0
[CALLSITE] void shmem_put64_nb_(void *, void *, int *, int *, void *) [@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_(	0.07	0.07	4,610	0
[CALLSITE] void shmem_put64_(void *, void *, int *, int *) [@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_OpenSHMEM	0.063	0.063	4,610	0
[CALLSITE] void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *) [@] [hydro_] [{/home/ssshend/CloverLeaf_OpenSHMEM/hydr	0.046	0.046	200	0
[CALLSITE] void shmem_real8_min_to_all_(void *, void *, int *, int *, int *, int *, void *, long *) [@] [/nfsprojects/volta-projects/tau/tau-2.24.1/craycnl/lib/	0.04	0.04	200	0
void shmem_real8_min_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.04	0.04	200	0
[CALLSITE] void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *) [@] [hydro_] [{/home/ssshend/CloverLeaf_OpenSHMEM/hydr	0.036	0.036	200	0
[CALLSITE] void shmem_put64_nb_(void *, void *, int *, int *, void *) [@] [__clover_module_MOD_clover_exchange] [{/home/ssshend/CloverLeaf_OpenSHME	0.028	0.028	601	0

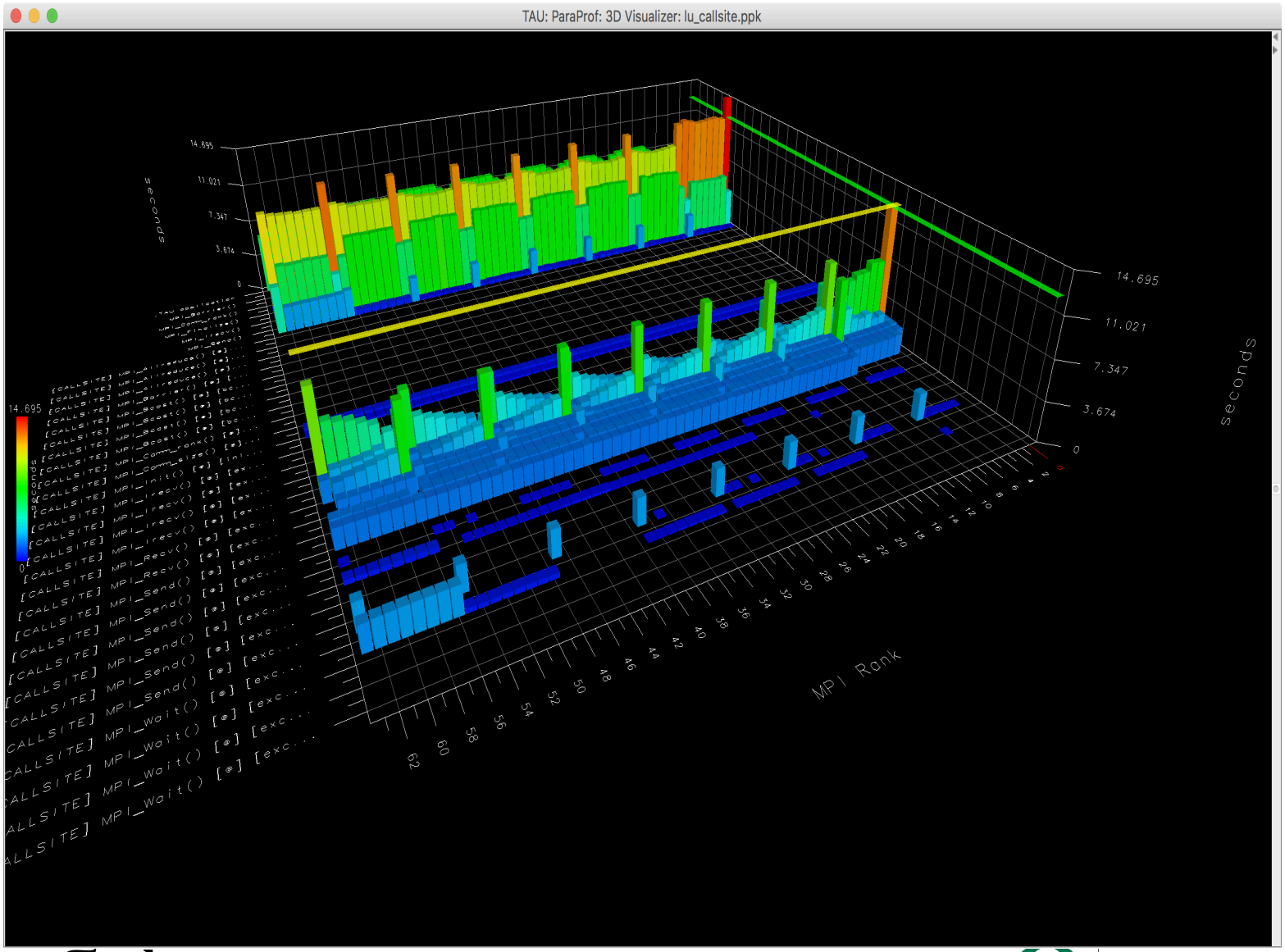
% export TAU_CALLSITE=1

Callsite Profiling and Tracing

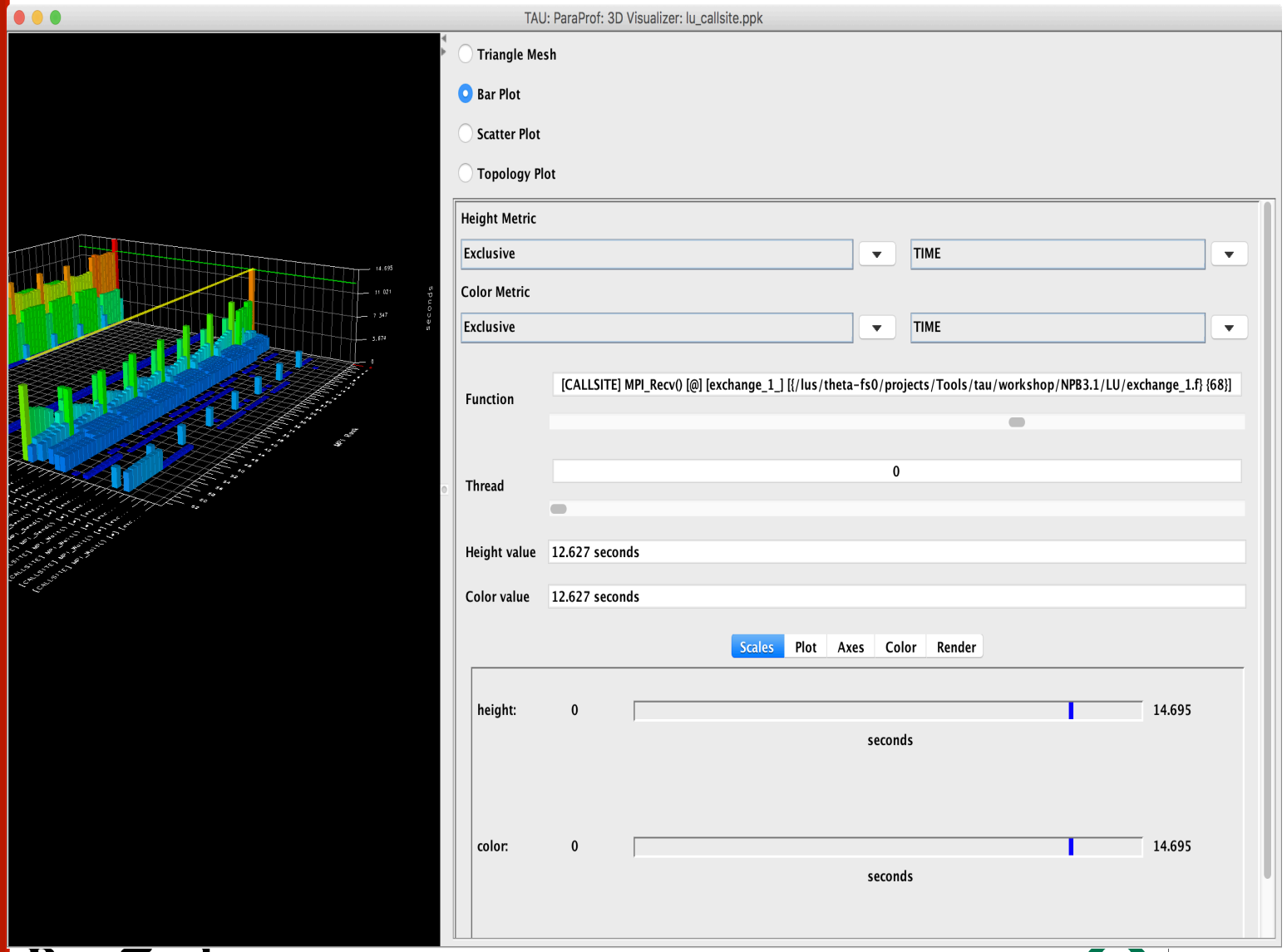


% export TAU_CALLSITE=1

Callsite Profiling and Tracing



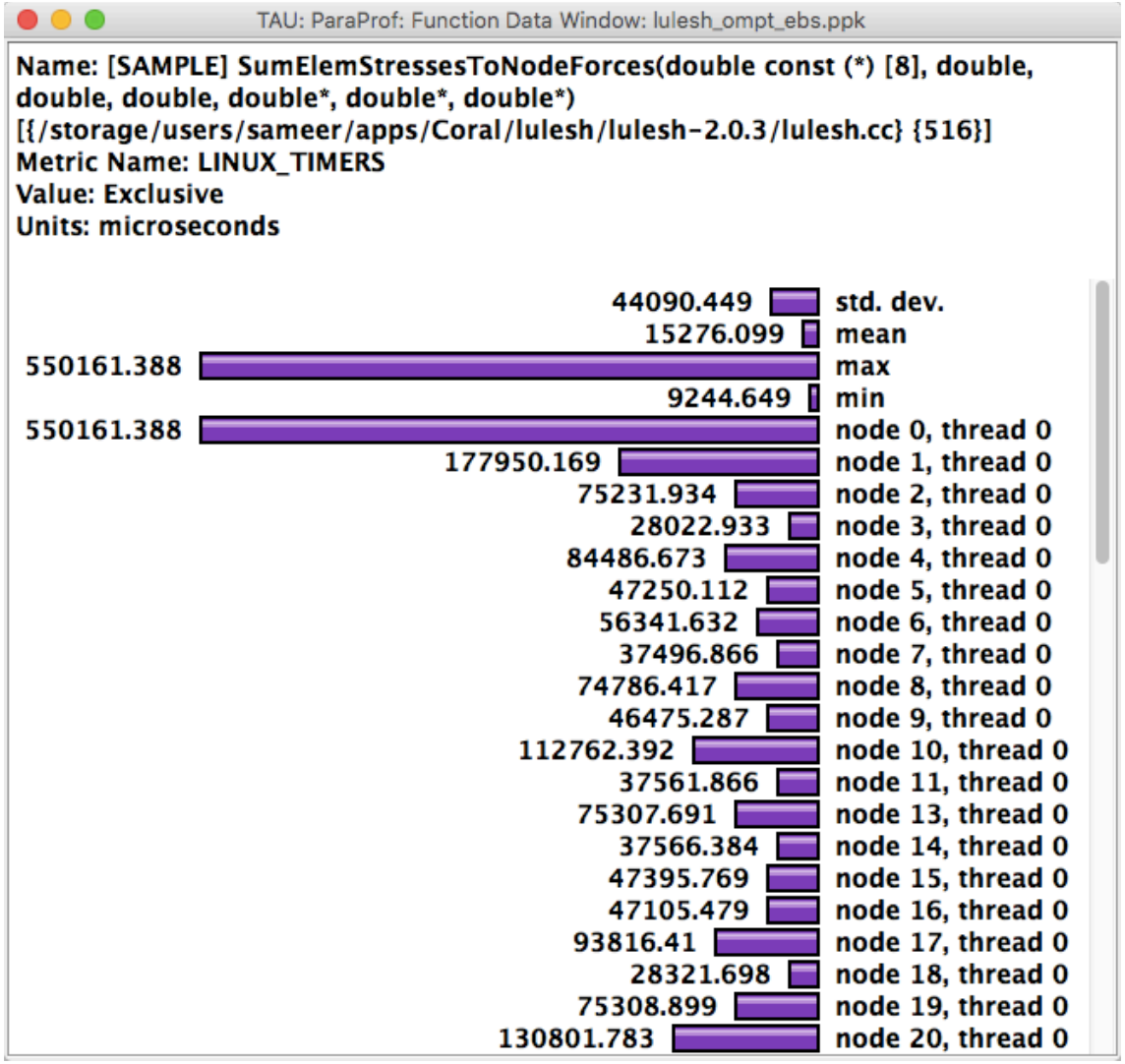
Callsite Profiling and Tracing



TAU – Callstack Sampling

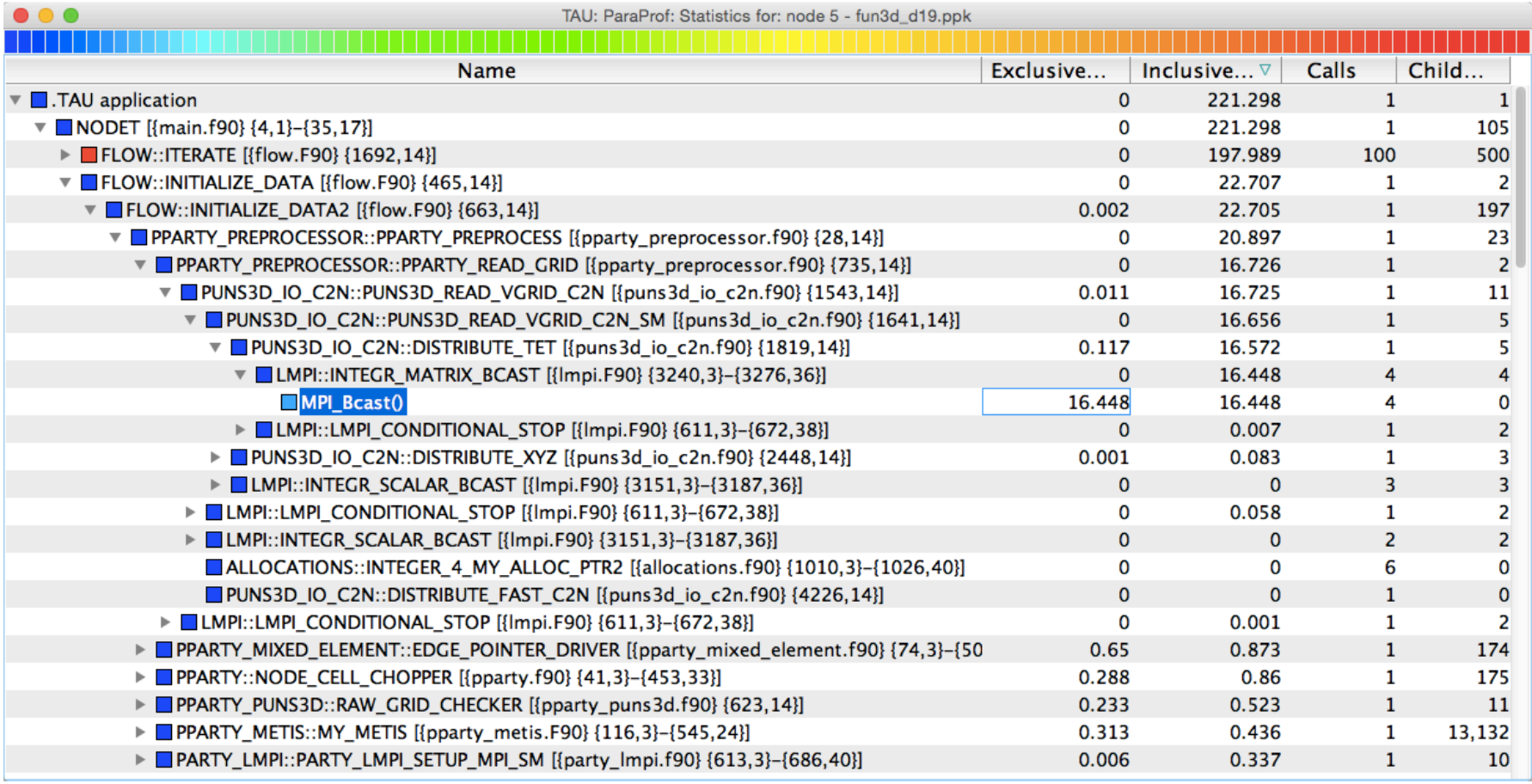
% export TAU_SAMPLING=1; export TAU_EBS_UNWIND=1

TAU – Event Based Sampling (EBS)



% export TAU_SAMPLING=1

TAU – Callpath Profiling



Name	Exclusive...	Inclusive... ▽	Calls	Child...
.TAU application	0	221.298	1	1
NODET [{main.f90} {4,1}-{35,17}]	0	221.298	1	105
FLOW::ITERATE [{flow.F90} {1692,14}]	0	197.989	100	500
FLOW::INITIALIZE_DATA [{flow.F90} {465,14}]	0	22.707	1	2
FLOW::INITIALIZE_DATA2 [{flow.F90} {663,14}]	0.002	22.705	1	197
PPARTY_PREPROCESSOR::PPARTY_PREPROCESS [{pparty_preprocessor.f90} {28,14}]	0	20.897	1	23
PPARTY_PREPROCESSOR::PPARTY_READ_GRID [{pparty_preprocessor.f90} {735,14}]	0	16.726	1	2
PUNS3D_IO_C2N::PUNS3D_READ_VGRID_C2N [{puns3d_io_c2n.f90} {1543,14}]	0.011	16.725	1	11
PUNS3D_IO_C2N::PUNS3D_READ_VGRID_C2N_SM [{puns3d_io_c2n.f90} {1641,14}]	0	16.656	1	5
PUNS3D_IO_C2N::DISTRIBUTE_TET [{puns3d_io_c2n.f90} {1819,14}]	0.117	16.572	1	5
LMPI::INTEGR_MATRIX_BCAST [{lmpi.F90} {3240,3}-{3276,36}]	0	16.448	4	4
MPI_Bcast()	16.448	16.448	4	0
LMPI::LMPI_CONDITIONAL_STOP [{lmpi.F90} {611,3}-{672,38}]	0	0.007	1	2
PUNS3D_IO_C2N::DISTRIBUTE_XYZ [{puns3d_io_c2n.f90} {2448,14}]	0.001	0.083	1	3
LMPI::INTEGR_SCALAR_BCAST [{lmpi.F90} {3151,3}-{3187,36}]	0	0	3	3
LMPI::LMPI_CONDITIONAL_STOP [{lmpi.F90} {611,3}-{672,38}]	0	0.058	1	2
LMPI::INTEGR_SCALAR_BCAST [{lmpi.F90} {3151,3}-{3187,36}]	0	0	2	2
ALLOCATIONS::INTEGER_4_MY_ALLOC_PTR2 [{allocations.f90} {1010,3}-{1026,40}]	0	0	6	0
PUNS3D_IO_C2N::DISTRIBUTE_FAST_C2N [{puns3d_io_c2n.f90} {4226,14}]	0	0	1	0
LMPI::LMPI_CONDITIONAL_STOP [{lmpi.F90} {611,3}-{672,38}]	0	0.001	1	2
PPARTY_MIXED_ELEMENT::EDGE_POINTER_DRIVER [{pparty_mixed_element.f90} {74,3}-{50	0.65	0.873	1	174
PPARTY::NODE_CELL_CHOPPER [{pparty.f90} {41,3}-{453,33}]	0.288	0.86	1	175
PPARTY_PUNS3D::RAW_GRID_CHECKER [{pparty_puns3d.f90} {623,14}]	0.233	0.523	1	11
PPARTY_METIS::MY_METIS [{pparty_metis.F90} {116,3}-{545,24}]	0.313	0.436	1	13,132
PARTY_LMPI::PARTY_LMPI_SETUP_MPI_SM [{party_lmpi.f90} {613,3}-{686,40}]	0.006	0.337	1	10

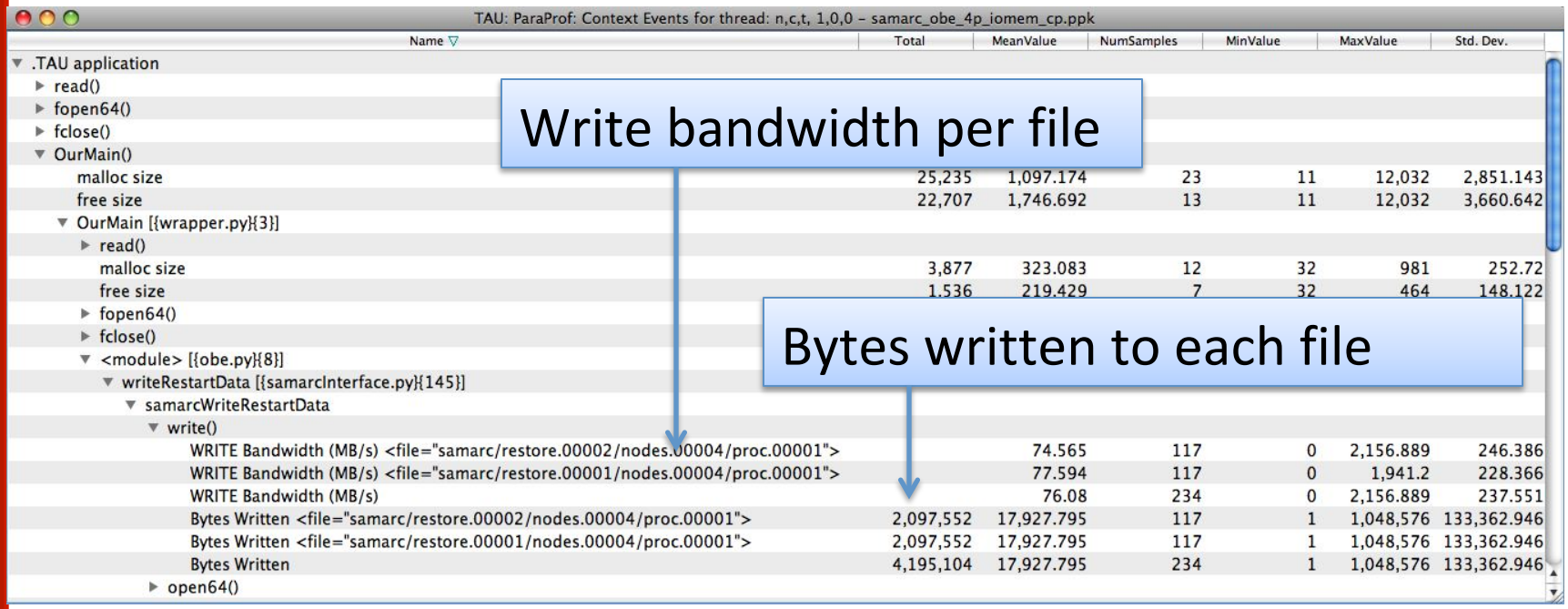
% export TAU_CALLPATH=1; export TAU_CALLPATH_DEPTH=100

TAU Atomic Events

TAU: ParaProf: Context Events for: node 0 - /Users/sameer/tmp

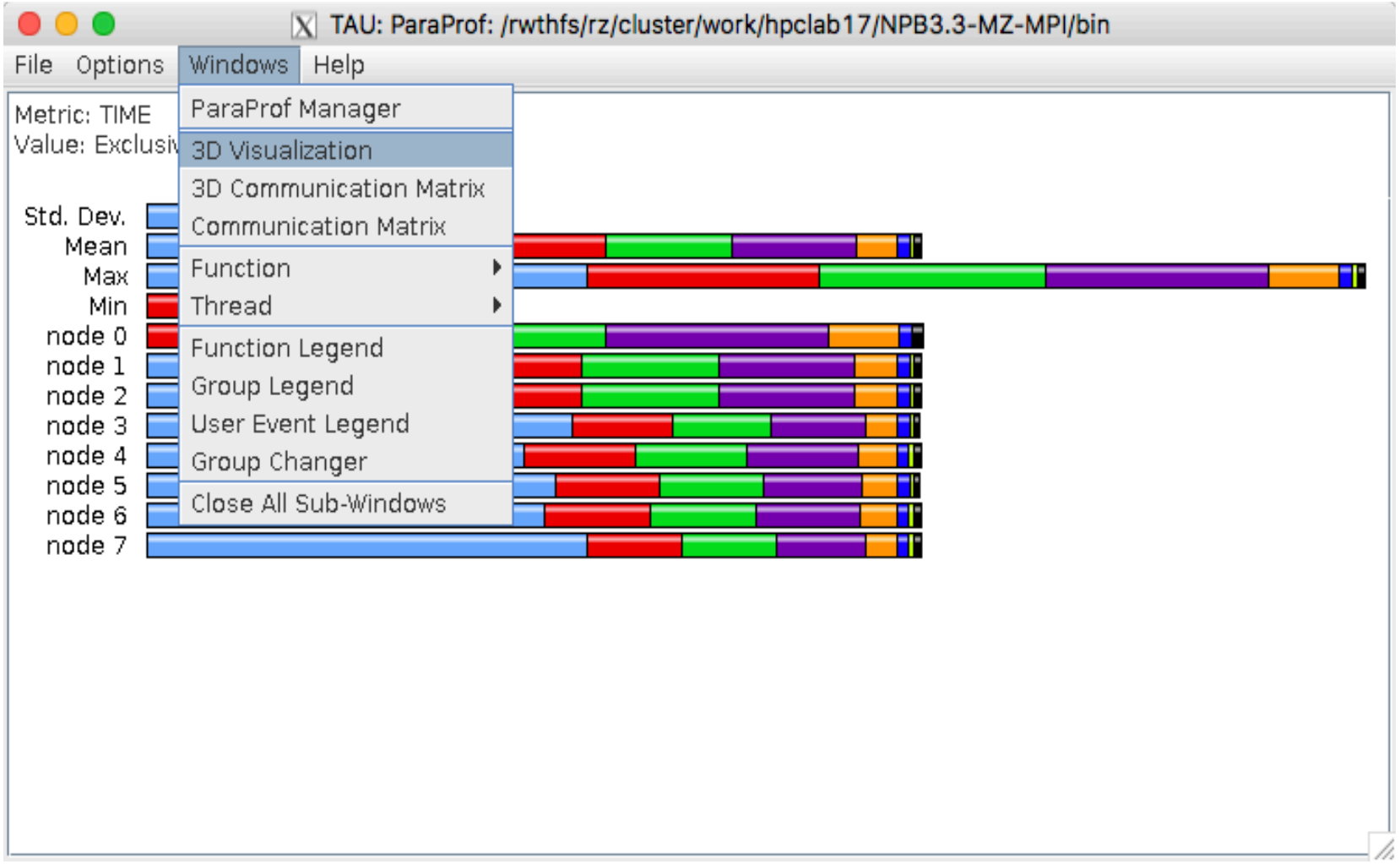
Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
Bytes Written <file=stdout>	911	62	21	1	14.694	7.441
Bytes Written <file=pipe>	22	22	1	1	1	0
Bytes Written <file=Process_Output/VelRsdl.dat>	7,826	100	302	76	78.26	22.487
Bytes Written <file=Process_Output/MomRsdl.dat>	7,826	100	302	76	78.26	22.487
Bytes Written <file=Process_Output/MassRsdl.dat>	11,325	100	435	110	113.25	32.337
Bytes Written <file=Grid_Output/bodyBndry.dat>	9,724	5	8,192	4	1,944.8	3,174.201
Bytes Written <file=/home/sameer/apps/sukra/RotCFD_Regression/case_catalog/UNS2D/N/	45	1	45	45	45	0
Bytes Written <file=./Restarts/Restart_History//NACA0012_LargeGrid_00010.Rst>	44,619,720	5,484	8,192	4	8,136.346	640.325
Bytes Written <file=./Restarts/Restart_History//NACA0012_LargeGrid_00005.Rst>	44,619,720	5,484	8,192	4	8,136.346	640.325
Bytes Written <file=./Restarts//NACA0012_LargeGrid.Rst>	44,619,720	5,484	8,192	4	8,136.346	640.325
Bytes Written <file=./Process_Output/TurbRsdl.dat>	4,271	72	224	57	59.319	19.544
Bytes Written <file=./Process_Output/Solver.out>	2,039	13	797	43	156.846	191.359
Bytes Written <file=./Field_Solutions/Solution_History/NACA0012_LargeGrid_00010.Sln>	4,356,976	534	8,192	4	8,159.131	501.319
Bytes Written <file=./Field_Solutions/Solution_History/NACA0012_LargeGrid_00005.Sln>	4,356,976	534	8,192	4	8,159.131	501.319
Bytes Written <file=./Field_Solutions/NACA0012_LargeGrid.Sln>	4,356,976	534	8,192	4	8,159.131	501.319
Bytes Written <file=./Body_Pressure/NACA0012_LargeGrid_00010_body.Prs>	65,986	9	8,190	1,300	7,331.778	2,133.204
Bytes Written <file=./Body_Pressure/NACA0012_LargeGrid_00005_body.Prs>	65,986	9	8,190	1,300	7,331.778	2,133.204
Bytes Written <file=./Body_Pressure/FrcMnt.out>	1,497	3	1,185	108	499	486.656
Bytes Written	147,107,546	18,550	8,192	1	7,930.326	1,420.552

TAU – Context Events

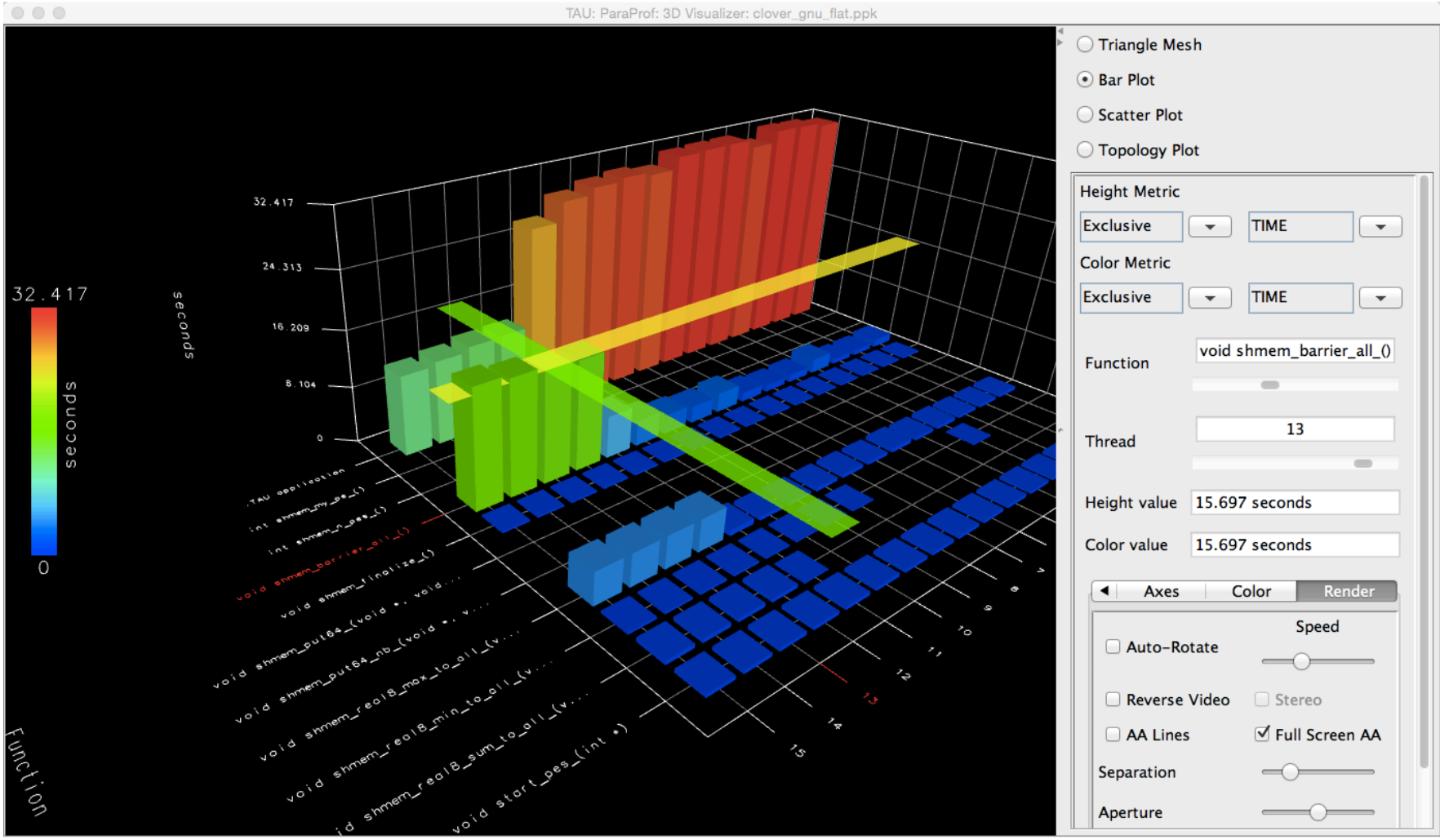


Name	Total	MeanValue	NumSamples	MinValue	MaxValue	Std. Dev.
.TAU application						
read()						
fopen64()						
fclose()						
OurMain()						
malloc size	25,235	1,097.174	23	11	12,032	2,851.143
free size	22,707	1,746.692	13	11	12,032	3,660.642
OurMain [{wrapper.py}{3}]						
read()						
malloc size	3,877	323.083	12	32	981	252.72
free size	1,536	219.429	7	32	464	148.122
fopen64()						
fclose()						
<module> [{obe.py}{8}]						
writeRestartData [{samarcInterface.py}{145}]						
samarcWriteRestartData						
write()						
WRITE Bandwidth (MB/s) <file="samarc/restore.00002/nodes.00004/proc.00001">		74.565	117	0	2,156.889	246.386
WRITE Bandwidth (MB/s) <file="samarc/restore.00001/nodes.00004/proc.00001">		77.594	117	0	1,941.2	228.366
WRITE Bandwidth (MB/s)		76.08	234	0	2,156.889	237.551
Bytes Written <file="samarc/restore.00002/nodes.00004/proc.00001">	2,097,552	17,927.795	117	1	1,048,576	133,362.946
Bytes Written <file="samarc/restore.00001/nodes.00004/proc.00001">	2,097,552	17,927.795	117	1	1,048,576	133,362.946
Bytes Written	4,195,104	17,927.795	234	1	1,048,576	133,362.946
open64()						

3D Visualization in ParaProf

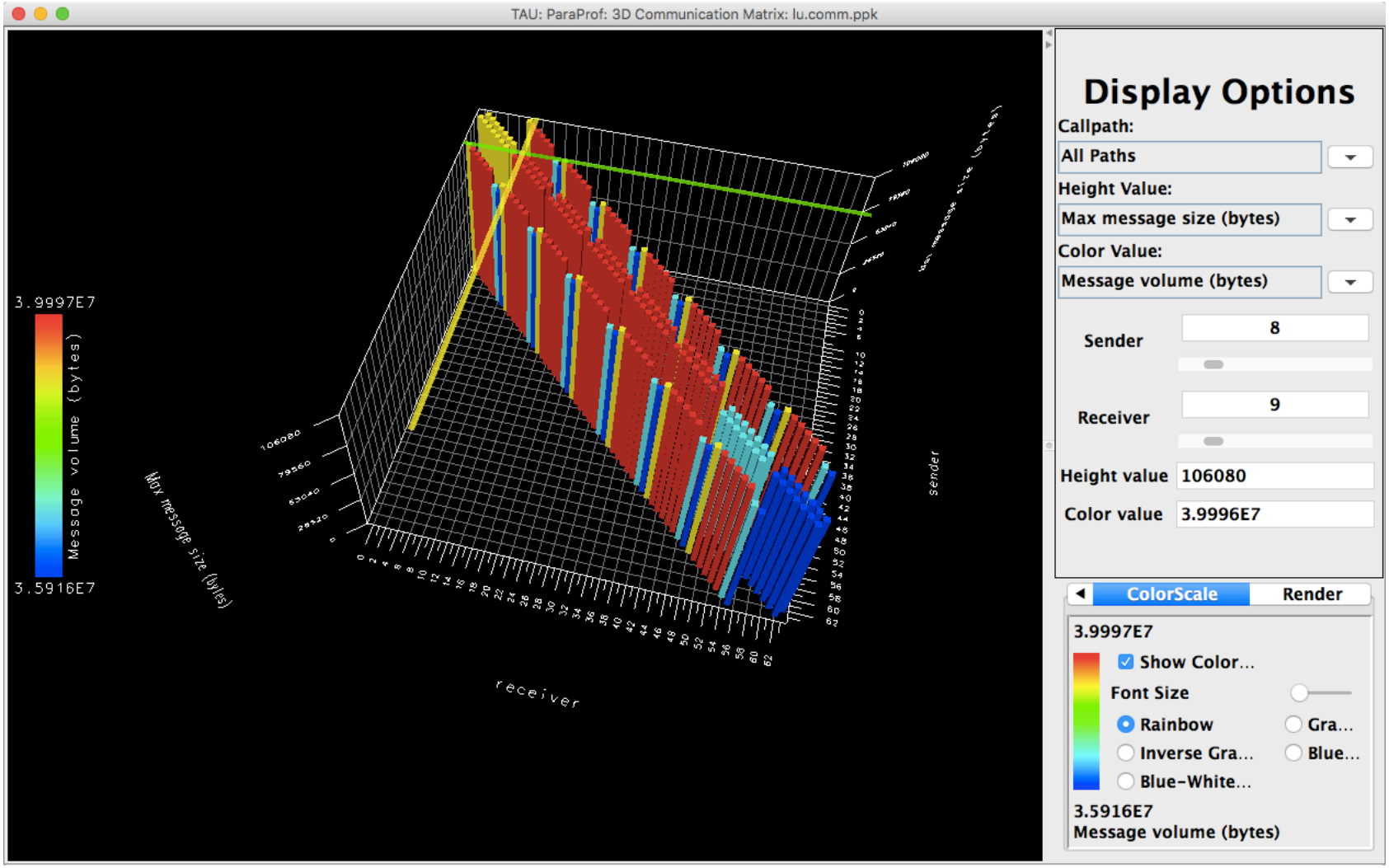


TAU – ParaProf 3D Visualization



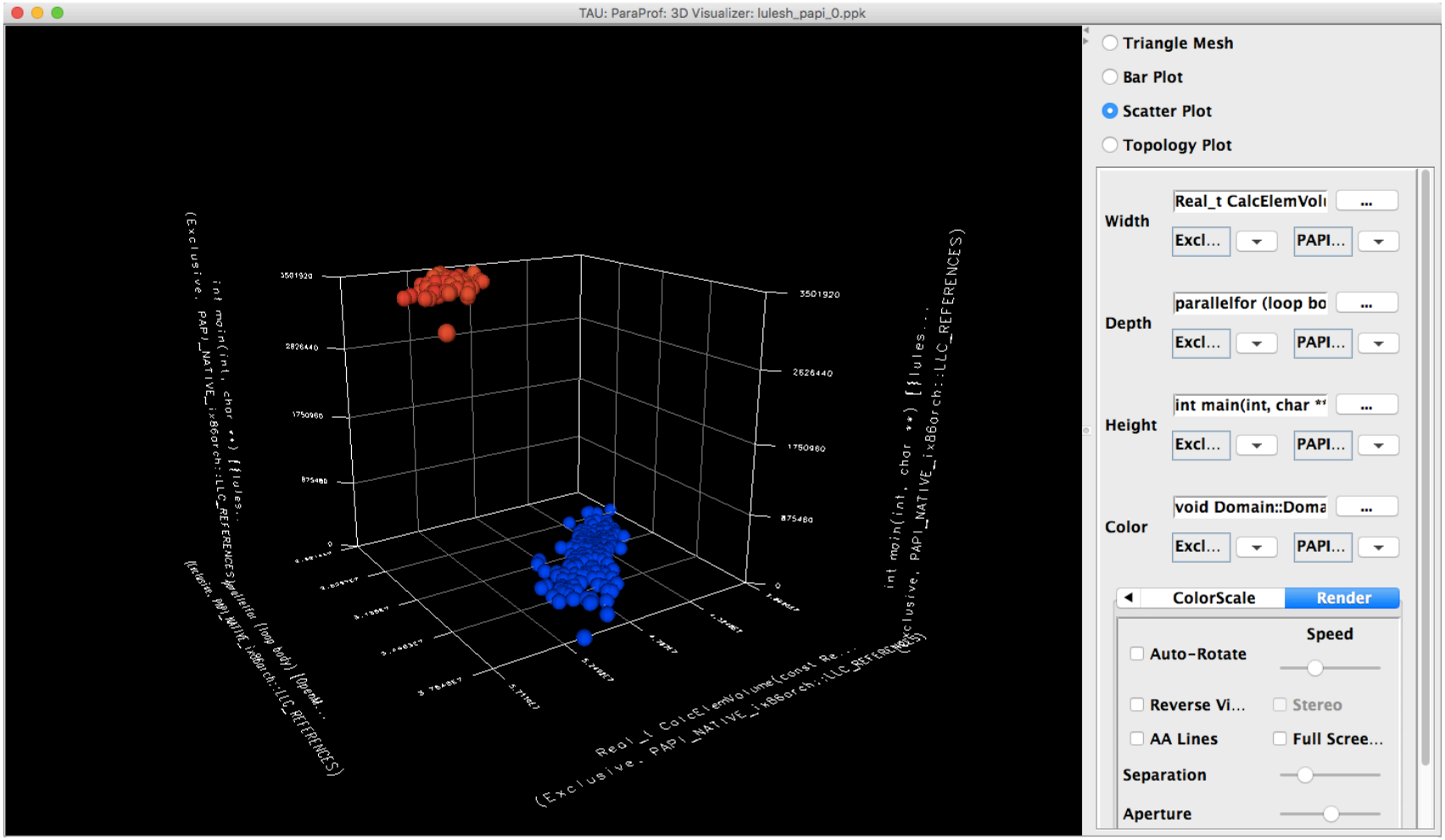
% paraprof app.ppk
Windows -> 3D Visualization -> Bar Plot (right pane)

TAU – 3D Communication Window



% export TAU_COMM_MATRIX=1; mpirun ⋯ tau_exec ./a.out
% paraprof app.ppk; Windows -> 3D Communication Matrix

TAU – 3D Scatter Plot



% paraprof app.ppk; Windows -> 3D Visualization -> Scatter Plot

Examples

Setting up Accounts, Examples at LLNL

```
On quartz.llnl.gov:
% tar zxf /usr/global/tools/tau/training/workshop.tgz
% cat workshop/README

We have a pReserved queue on Quartz.llnl.gov
% salloc -N 1 -p pReserved
After the workshop:
% salloc -N 1 -p pdebug
% srun -n <procs> ./a.out

% source /usr/global/tools/tau/training/tau.bashrc (or cshrc)
```

See http://tau.uoregon.edu/workshop_summer18.tgz (cat README)

Setting up Accounts, Examples at Sandia

```
On Serrano.sandia.gov
% tar zxf /projects/tau/workshop.tgz
% cd workshop
% source /projects/tau/tau.bashrc  (or tau.cshrc) for OpenMPI
% echo $TAU/Makefile.tau-
% cat README
And try the examples.

OR:
```

See http://tau.uoregon.edu/workshop_summer18.tgz (cat README)

Setting up Accounts, Examples at LANL

```
On Wolf wf-fe.lanl.gov:
% tar zxf /usr/projects/packages/tau/workshop.tgz
% cat README
% module load friendly-testing
% module load intel
% module load openmpi
% module load tau/2.27.1
% echo $TAU/Makefile.tau-
/usr/projects/hpcsoft/toss2/wolf/tau/2.27.1/x86_64/lib/
Makefile.tau-intel-18.0.2-openmpi-2.1.2-icpc-mpi
% echo $TAU_LTAG
intel-18.0.2-openmpi-2.1.2
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-mpi-pdt
OR:
```

See http://tau.uoregon.edu/workshop_summer18.tgz (cat README)

Simplifying the use of TAU!

Uninstrumented code:

- % cd workshop/mm; make
- % srun –n 8 ./matmult

With TAU:

- % srun –n 8 tau_exec ./matmult
- % paraprof

For more information at the statement level:

- % srun –n 8 tau_exec –ebs ./matmult
- % paraprof

To rewrite the binary to instrument individual functions (using MAQAO):

- % tau_rewrite matmult –o matmult.inst; srun –n 8 ./matmult.inst (beta)
- % paraprof

TAU Demo

```
On quartz.llnl.gov:
% tar zxf /usr/global/tools/tau/training/workshop.tgz
% cat workshop/README

We have a pReserved queue on Quartz.llnl.gov
% salloc -N 1 -p pReserved
After the workshop:
    % salloc -N 1 -p pdebug

% source /usr/global/tools/tau/training/tau.bashrc (or cshrc)
% cd workshop/mm
% make
% srun -n 8 ./matmult
% srun -n 8 tau_exec -ebs ./matmult
% paraprof &
% pprof -a | more
```

TAU Execution Command (tau_exec)

Uninstrumented execution

- % srun –n 8 ./a.out

Track GPU operations

- % srun –n 8 tau_exec –cupti ./a.out
- % srun –n 8 tau_exec –cupti -um ./a.out (for Unified Memory)
- % srun –n 8 tau_exec –opencl ./a.out
- % srun –n 8 tau_exec –openacc ./a.out

Track MPI performance

- % srun –n 8 tau_exec ./a.out

Track OpenMP, and MPI performance (MPI enabled by default)

- % export TAU_OMPT_SUPPORT_LEVEL=full;
  % export TAU_OMPT_RESOLVE_ADDRESS_EAGERLY=1
- % srun –n 8 tau_exec -T ompt,tr6,mpi –ompt ./a.out

Track I/O operations

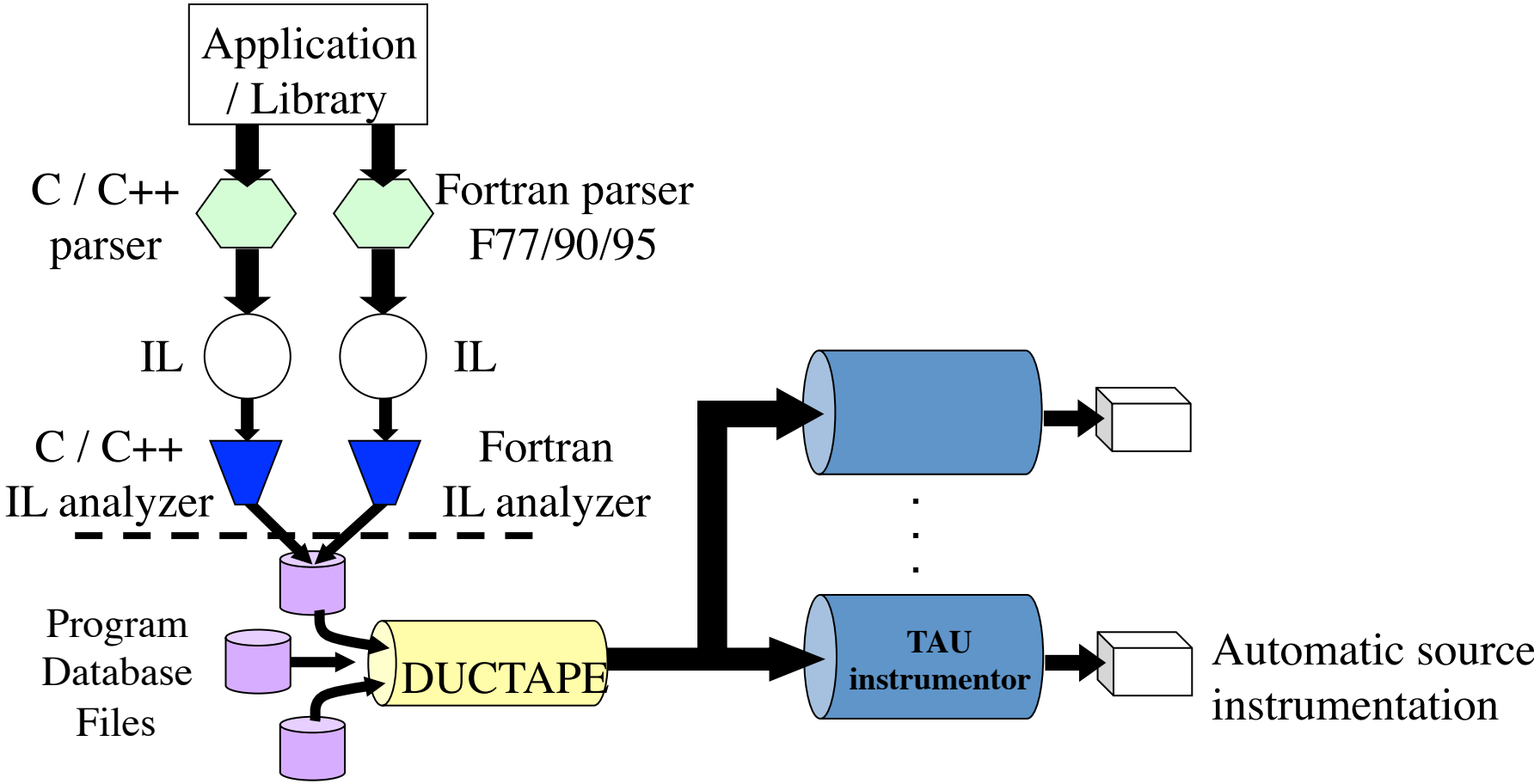
- % srun –n 8 tau_exec –io -T pthread ./a.out

Use event based sampling (compile with –g)

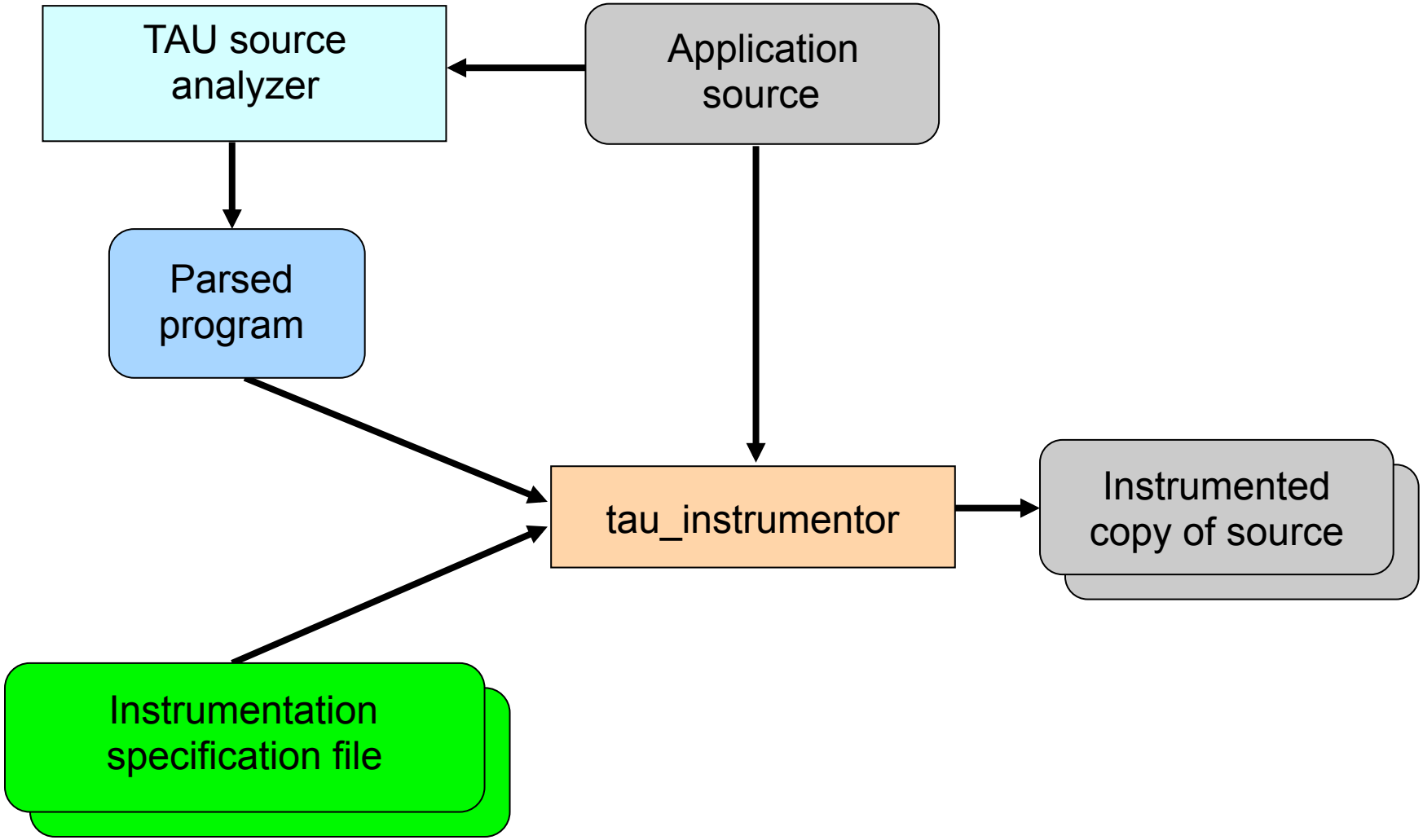
- % srun –n 8 tau_exec –ebs ./a.out
- Also –ebs_source=<PAPI_COUNTER> -ebs_period=<overflow_count>

Automatic Source Instrumentation

TAU's Static Analysis System: Program Database Toolkit (PDT)



Automatic Source Instrumentation using PDT



Automatic Source Instrumentation

- **Use TAU's compiler wrappers**
- **% make MPIF77=tau_f77.sh**
  - Simply replace `CXX` with `tau_cxx.sh`
  - Automatically instruments source code, links with TAU libraries.

Before

```
CXX = mpicxx
F90 = mpif90
CXXFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
     $(CXX) $(LDFLAGS) $(OBJS) -o $@
     $(LIBS)
.cpp.o:
     $(CXX) $(CXXFLAGS) -c $<
```

After

```
CXX = tau_cxx.sh
F90 = tau_f90.sh
CXXFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
     $(CXX) $(LDFLAGS) $(OBJS) -o $@
     $(LIBS)
.cpp.o:
     $(CXX) $(CXXFLAGS) -c $<
```

Using Source Instrumentation

TAU supports several compilers, measurement, and thread options

Intel compilers, profiling with hardware counters using PAPI, MPI library, OpenMP…

Each measurement configuration of TAU corresponds to a unique stub makefile (configuration file) and library that is generated when you configure it

To instrument source code automatically using PDT

Choose an appropriate TAU stub makefile in <arch>/lib:

% source /home/tauteam/pkgs/tau.bashrc

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

Use tau_f90.sh, tau_cxx.sh, tau_upc.sh, or tau_cc.sh as F90, C++, UPC, or C compilers respectively:

% mpif90 foo.f90 changes to

% tau_f90.sh foo.f90

Set runtime environment variables, execute application and analyze performance data:

% pprof (for text based profile display)

% paraprof (for GUI)

INSTALLING TAU on Laptops

Installing TAU under Mac OS X:

- http://tau.uoregon.edu/tau.dmg

Installing TAU under Windows

- http://tau.uoregon.edu/tau.exe

Installing TAU under Linux

- http://tau.uoregon.edu/tau.tgz
- ./configure; make install
- export PATH=<taudir>/x86_64/bin:$PATH

Different Makefiles for TAU Compiler

```
%  source /usr/global/tools/tau/training/tau.bashrc (or .cshrc)
% ls $TAU/Makefile.*
```

/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-mpi-pdt
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-mpi-pthread-pdt
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-ompt-mpi-pdt-openmp
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-ompt-tr6-mpi-pdt-openmp
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-ompt-tr6-openmp
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-openmp-opari
/usr/global/tools/tau/training//tau_latest/x86_64/lib/Makefile.tau-icpc-papi-pthread

For an MPI+C application with Intel compilers, you may choose

Makefile.tau-icpc-papi-mpi-pdt-openmp

- Supports MPI instrumentation & PDT for automatic source instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% tau_cc.sh app.c -o app;

% mpirun -np 8 ./app
% paraprof
% pprof -a | more
```

Configuration tags for tau_exec

```
% ./configure -pdt=<dir> -mpi -papi=<dir>; make install
Creates in $TAU:
Makefile.tau-papi-mpi-pdt(Configuration parameters in stub makefile)
shared-papi-mpi-pdt/libTAU.so

% ./configure -pdt=<dir> -mpi; make install  creates
Makefile.tau-mpi-pdt
shared-mpi-pdt/libTAU.so

To explicitly choose preloading of shared-<options>/libTAU.so change:
% mpirun -np 256 ./a.out     to
% mpirun -np 256  tau_exec -T <comma_separated_options> ./a.out

% mpirun -np 256  tau_exec -T papi,mpi,pdt ./a.out
Preloads $TAU/shared-papi-mpi-pdt/libTAU.so
% mpirun -np 256  tau_exec -T papi ./a.out
Preloads $TAU/shared-papi-mpi-pdt/libTAU.so by matching.
% mpirun -np 256 tau_exec -T papi,mpi,pdt -s ./a.out
Does not execute the program. Just displays the library that it will
preload if executed without the -s option.
NOTE: -mpi configuration is selected by default. Use -T serial for
Sequential programs.
```

Binary Rewriting Instrumentation

- Support for both **static and dynamic** executables
- Specify a list of routines to instrument
- Specify the TAU measurement library to be injected
- **Dyninst [U. Wisconsin, U. Maryland]**:

```
% tau_run -T [tags] a.out -o a.inst
```

- **MAQAO [Intel Exascale Labs, UVSQ]**:

```
% tau_rewrite -T [tags] a.out -o a.inst
```

- **Pebil [SDSC]**:

```
% tau_pebil_rewrite -T [tags] a.out \
  -o a.inst
```

- Execute the application to get measurement data:

```
% srun -n 8   ./a.inst
```

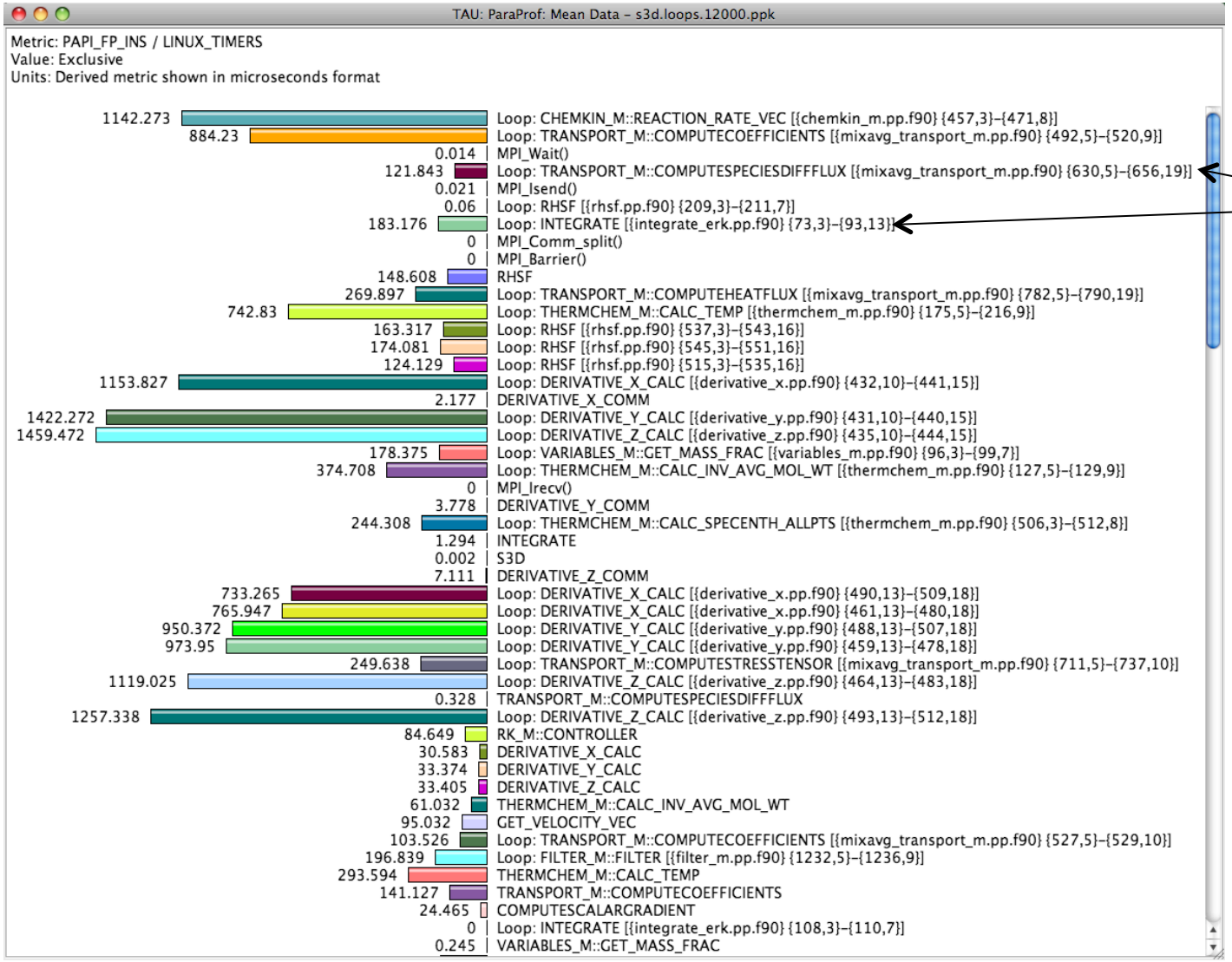
Binary Rewriting Instrumentation

```
% mpicc -g matmult*.c -o matmult
% tau_rewrite matmult -o matmult.i

Or use a selective instrumentation file (include/exclude lists)
% tau_rewrite -f select.tau -T icpc,mpi,papi \
      ./matmult -o matmult.i

% srun -n 8 ./matmult.i
% paraprof
```

Selective Instrumentation File

```
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
BEGIN_INCLUDE_LIST
int main#
int dgemm#
END_INCLUDE_LIST
BEGIN_FILE_INCLUDE_LIST
Main.c
Blas/*.f77
END_FILE_INCLUDE_LIST
# replace include with exclude list (BEGIN_EXCLUDE_LIST/END…)

BEGIN_INSTRUMENT_SECTION
loops routine="foo"
loops routine="int main#"
END_INSTRUMENT_SECTION
% export TAU_SELECT_FILE=select.tau    (to use at runtime)
```

Identifying Potential Bottlenecks



low mflops in loops?

OpenACC with PGI compilers

TAU: ParaProf: Statistics for: node 0, thread 0 - benchADM_acc2.ppk

Name	Exclusive...	Inclusive... ▽	Calls	Child...
.TAU application	4.982	9.443	1	5,168
openacc_enqueue_upload bench_staggeredleapfrog2 [{/storage	0.694	3.35	3,700	29,867
cuMemcpyHtoDAsync_v2	2.47	2.47	3,700	0
cuEventRecord	0.06	0.06	7,400	0
cuDeviceGetCount	0.032	0.032	7,401	0
cuEventElapsedTime	0.031	0.031	3,700	0
cuCtxSynchronize	0.031	0.031	3,700	0
cuEventSynchronize	0.028	0.028	3,700	0
cuDeviceGetAttribute	0.002	0.002	249	0
cuDeviceGetName	0	0	3	0
cuDeviceTotalMem_v2	0	0	3	0
cuCtxGetDevice	0	0	1	0
cuEventCreate	0	0	2	0
cuDeviceGet	0	0	3	0
cuDriverGetVersion	0	0	1	0
cuCtxGetCurrent	0	0	2	0
cuCtxGetApiVersion	0	0	1	0
cuInit	0	0	1	0
openacc_enqueue_download bench_staggeredleapfrog2 [{/stora	0.116	0.556	600	5,407
cuMemcpyDtoHAsync_v2	0.405	0.405	600	0
cuEventRecord	0.013	0.013	1,800	0
cuDeviceGetCount	0.005	0.005	1,200	0

% configure –c++=pgCC –cc=pgcc –fortran=pgi …
% tau_exec –T pgi -openacc -cupti ./a.out

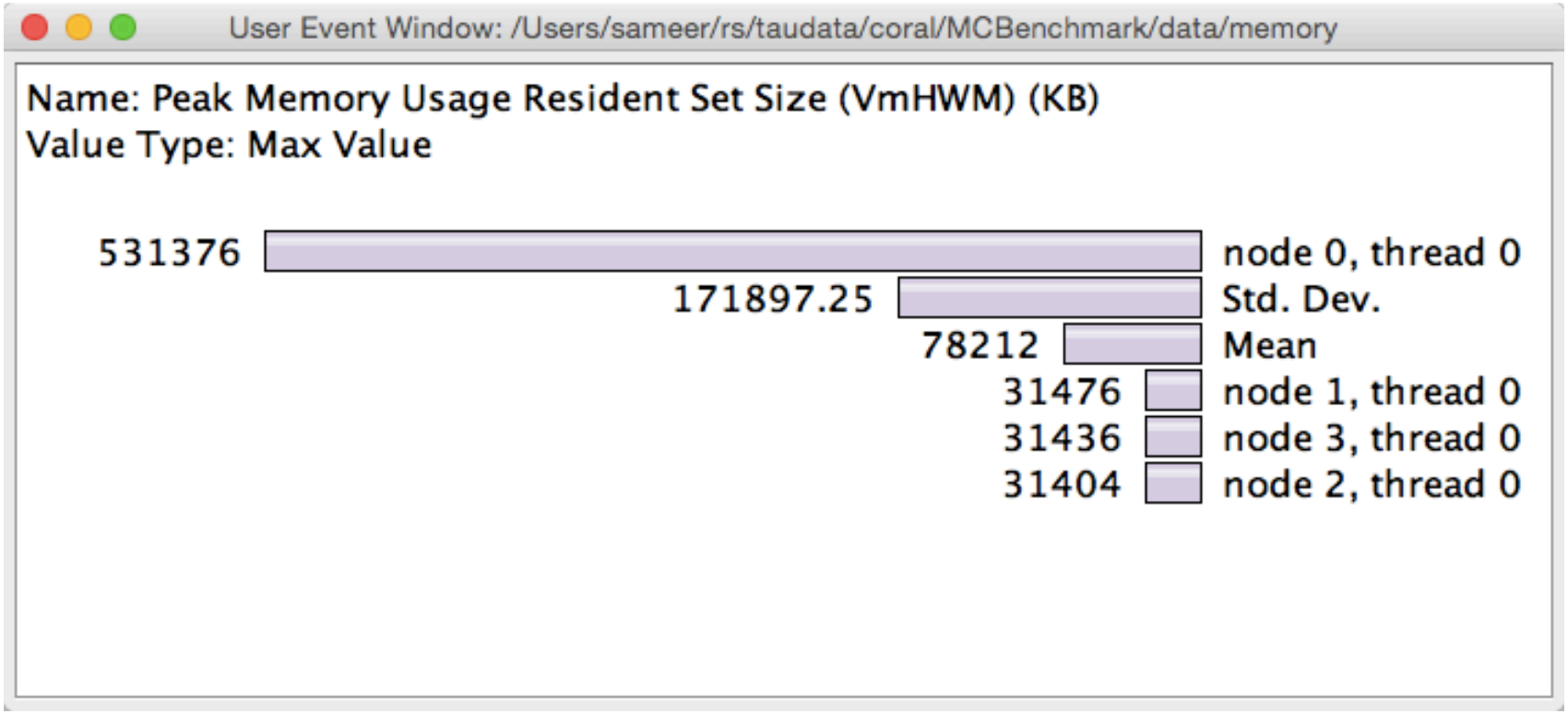
Tracking OpenACC Data Transfers

TAU: ParaProf: Context Events for: node 0, thread 0 - benchADM_acc2.ppk

Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
.TAU application						
openacc_enqueue_upload bench_staggeredleapfrog2 [{/st						
cuMemcpyHtoDAsync_v2						
[GROUP=MAX_MARKER] Bytes copied from Host to De	512,000	1	512,000	512,000	512,000	0
Bytes copied from Host to Device	973,016,000	3,700	512,000	120	262,977.297	255,846.506
openacc_enqueue_download bench_staggeredleapfrog2 [{						
cuMemcpyDtoHAsync_v2						
Bytes copied from Device to Host	307,200,000	600	512,000	512,000	512,000	0
Bytes copied from Device to Host	307,200,000	600	512,000	512,000	512,000	0
Bytes copied from Host to Device	973,016,000	3,700	512,000	120	262,977.297	255,846.506
[GROUP=MAX_MARKER] Bytes copied from Host to Device	512,000	1	512,000	512,000	512,000	0

% configure –c++=pgCC –cc=pgcc –fortran=pgi …
% tau_exec –T pgi -openacc -cupti ./a.out
Context events show extent of variation

Measuring Memory Footprint



% export TAU_TRACK_MEMORY_FOOTPRINT=1

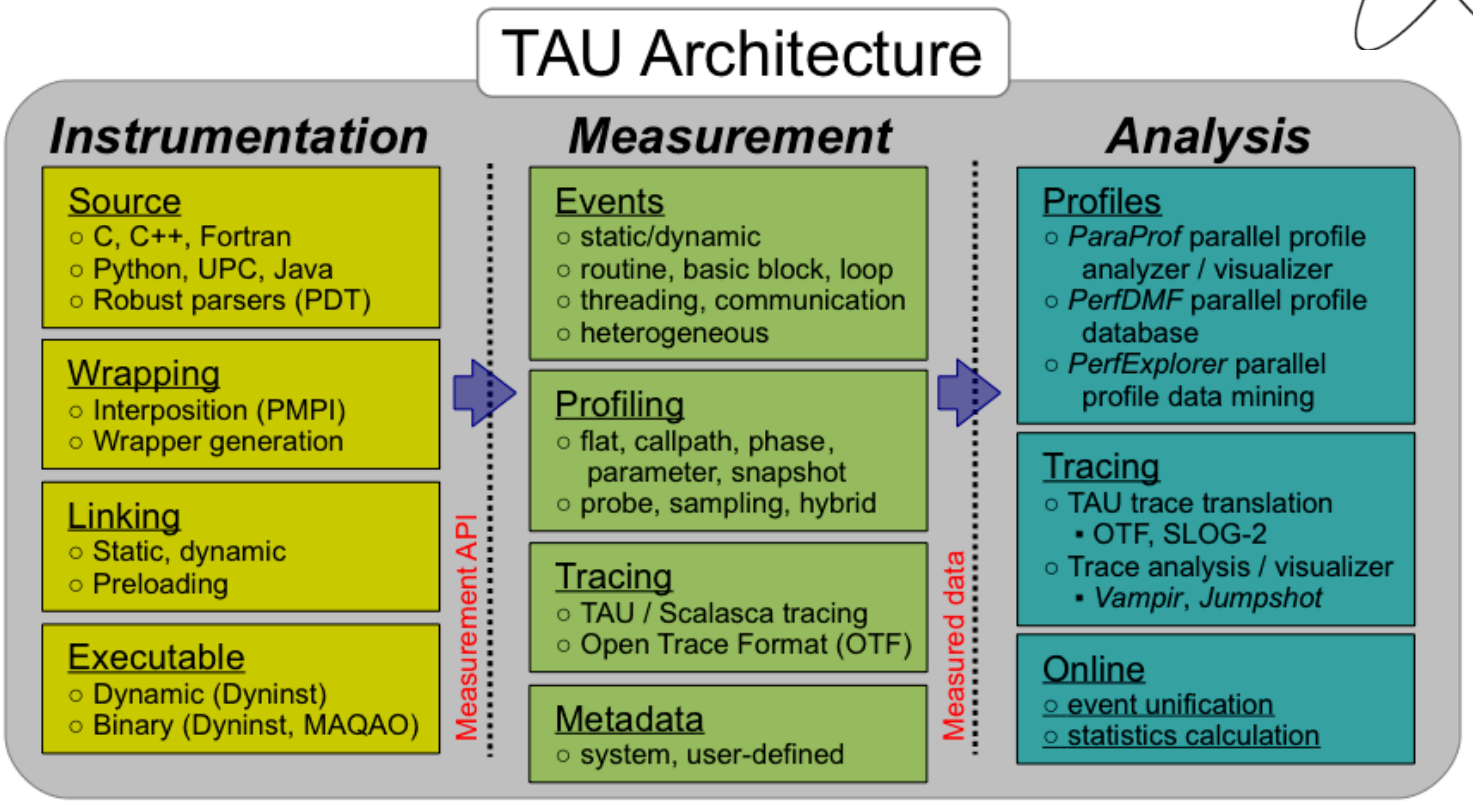
Paraprof:

Right click on a node -> Show Context Event Window -> see memory events

What does TAU support?

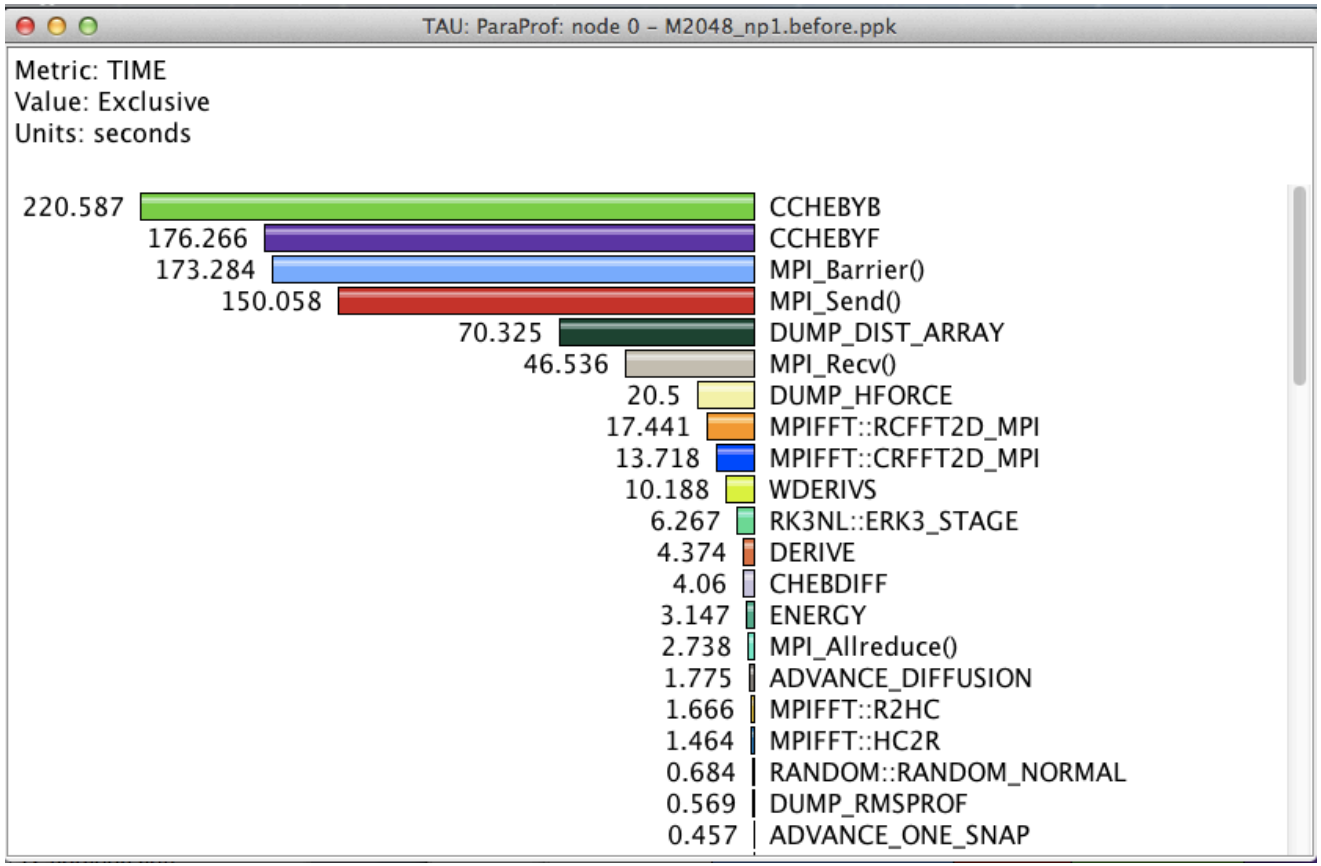
C/C++
CUDA
UPC
OpenCL
Python
GPI
Fortran
OpenACC
Java
MPI
pthreads
Intel MIC
OpenMP
Intel
GNU
PGI
Cray
Sun
LLVM
MPC
NEC
AIX
Linux
Windows
Insert yours here
BlueGene
Fujitsu
ARM64
NVIDIA
Power 8
OS X

TAU Architecture and Workflow



Routine Level Profile

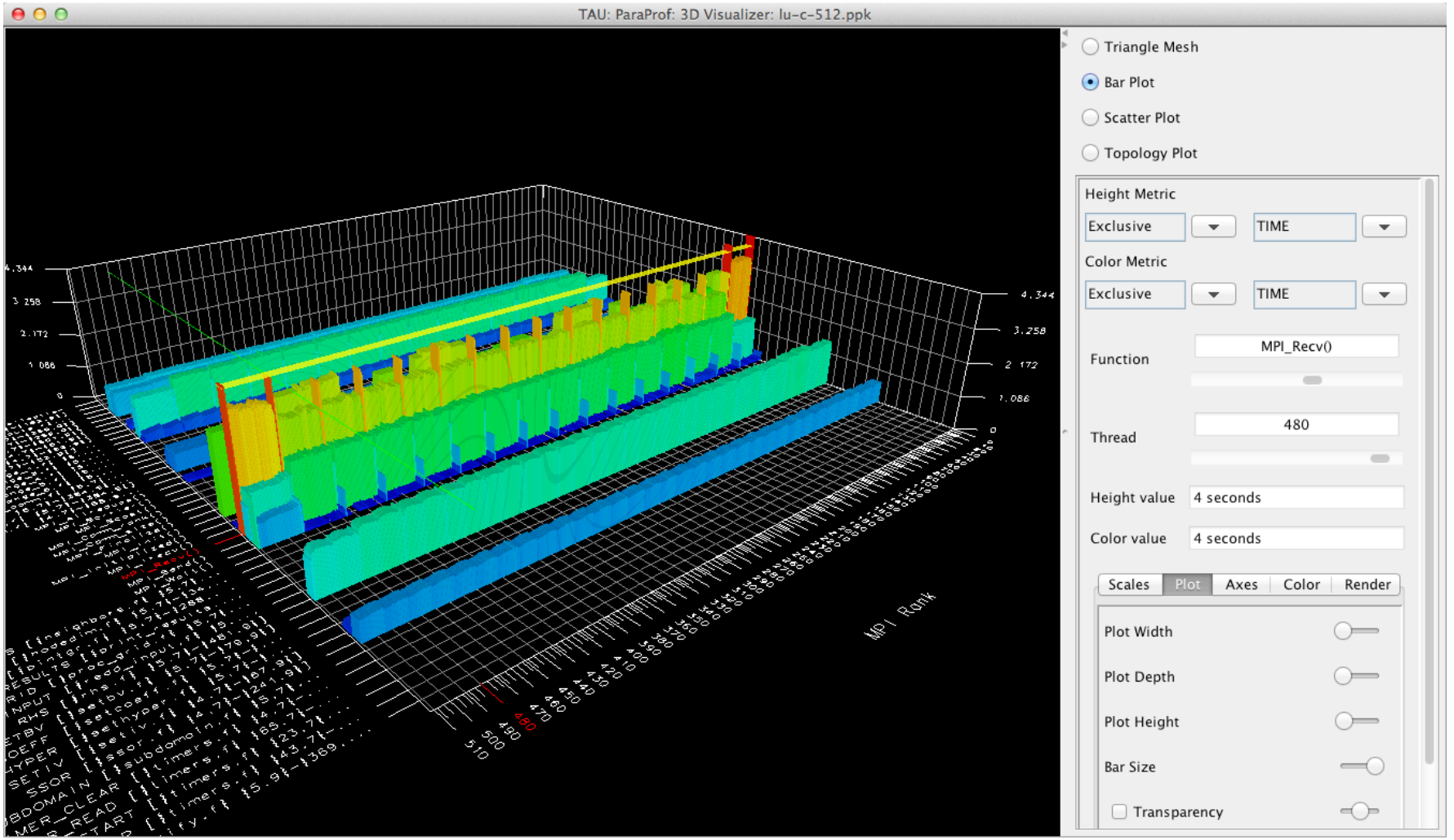
How much time is spent in each application routine?



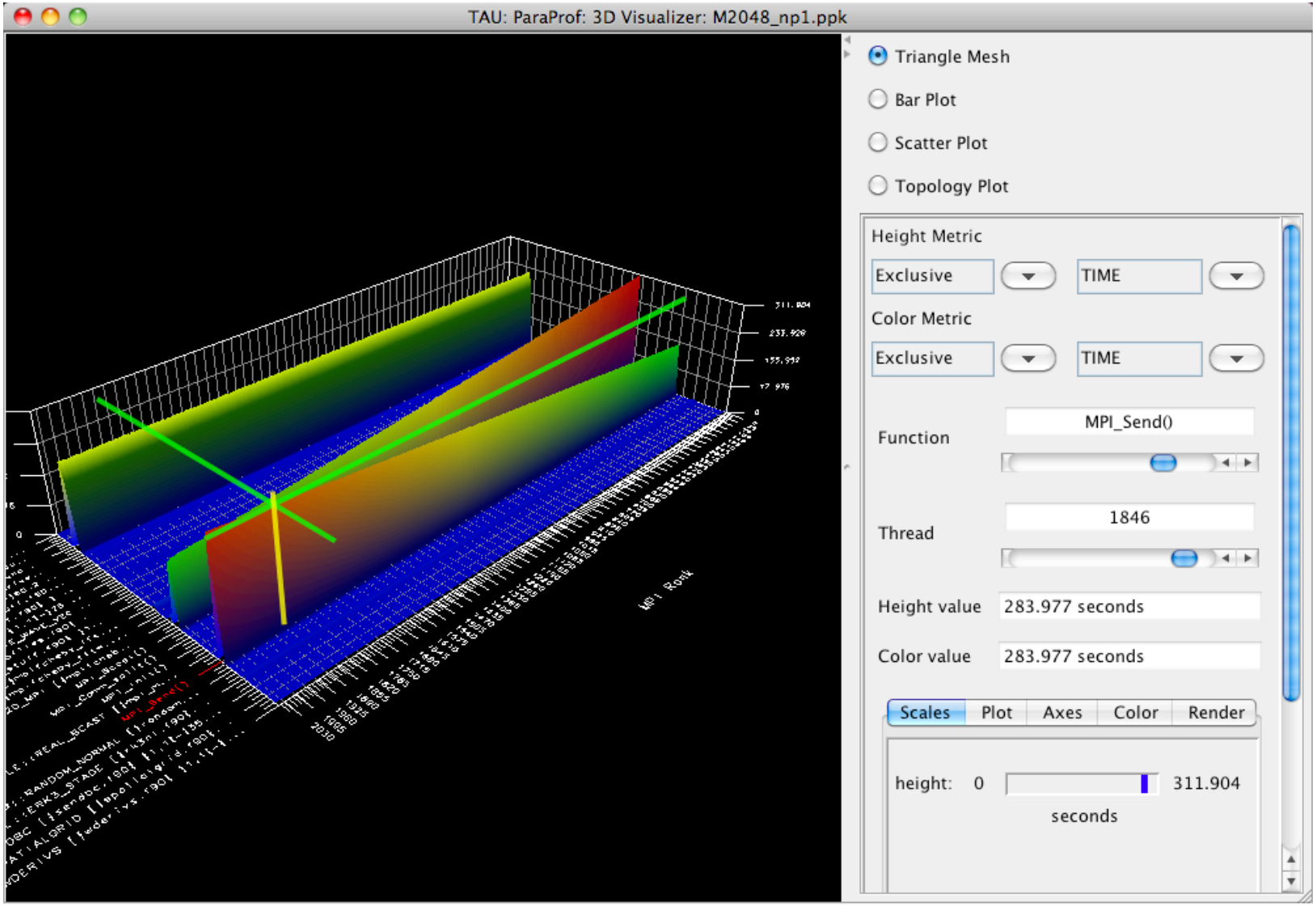
Generating a flat profile with MPI

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh
Or
% tau_f90.sh matmult.f90; srun -n 16  ./a.out
% paraprof
To view. To view the data locally on the workstation,
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk

Click on the "node 0" label to see profile for that node. Right
click to see other options. Windows -> 3D Visualization for 3D
window.
```

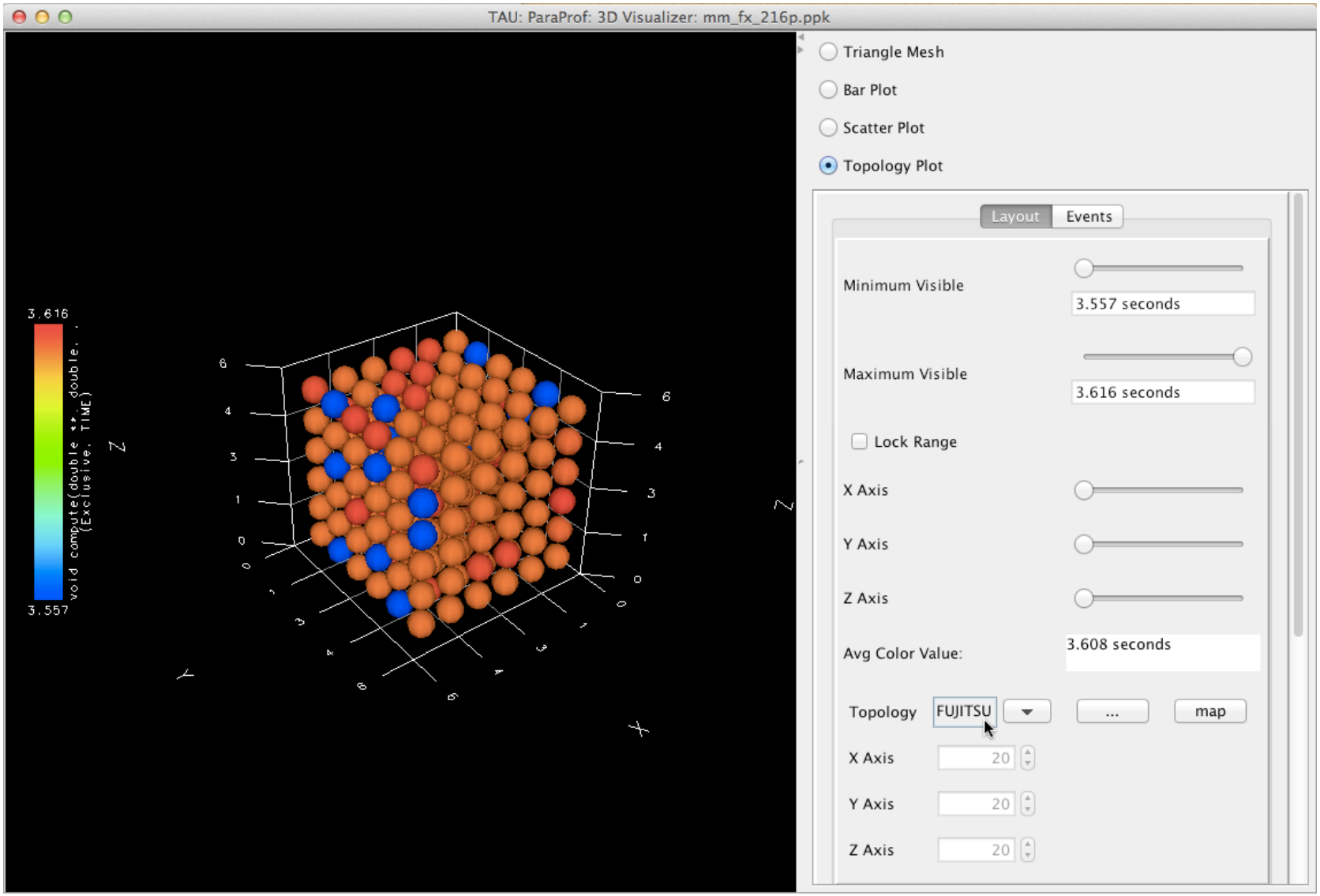
ParaProf 3D Profile Browser



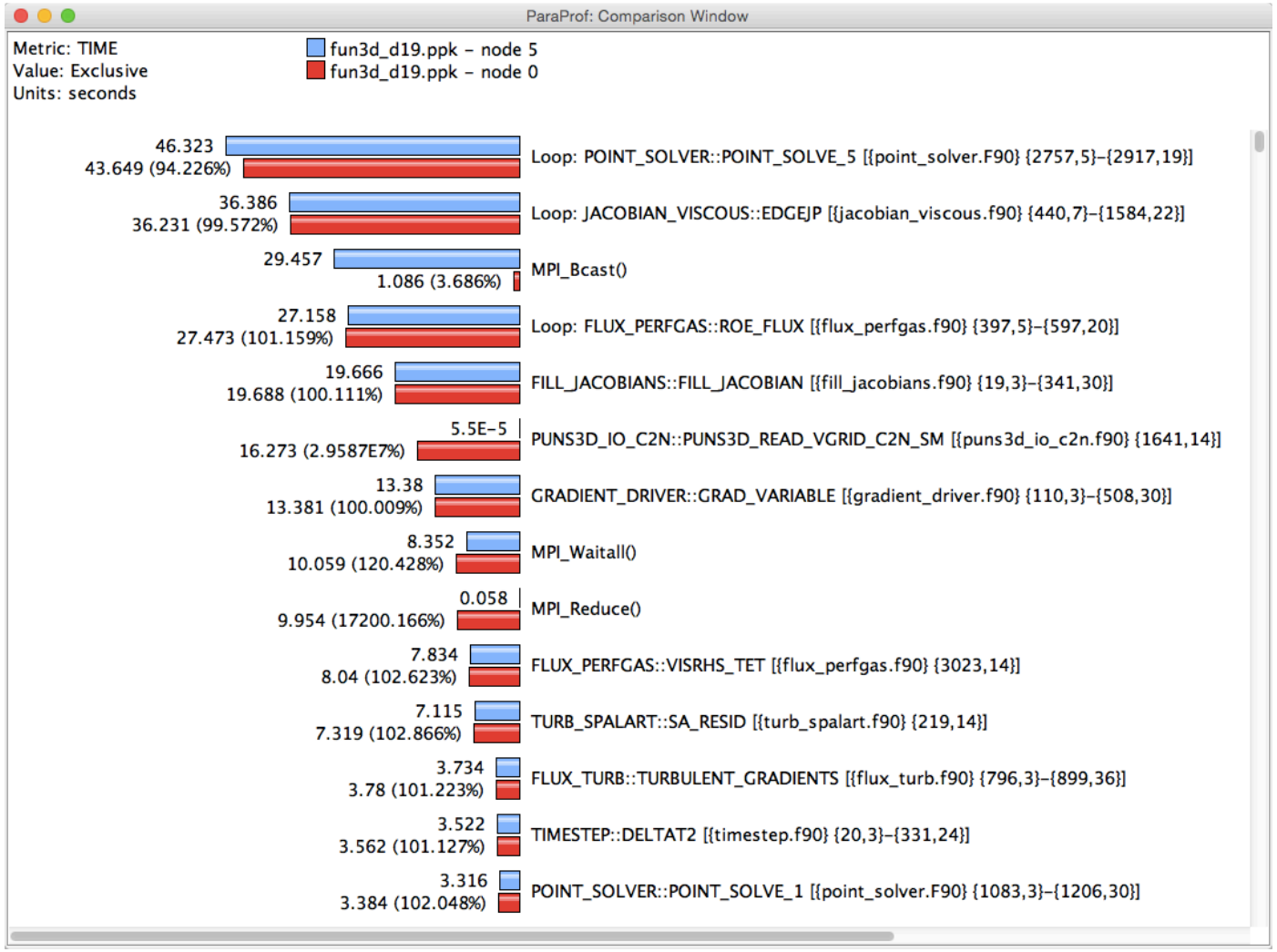
ParaProf


TAU: ParaProf: 3D Visualizer: M2048_np1.ppk
Triangle Mesh
Bar Plot
Scatter Plot
Topology Plot
Height Metric
Exclusive
TIME
Color Metric
Exclusive
TIME
Function
MPI_Send()
Thread
1846
Height value
283.977 seconds
Color value
283.977 seconds
Scales
Plot
Axes
Color
Render
height: 0
311.904
seconds
MPI Rank

ParaProf 3D Topology Display



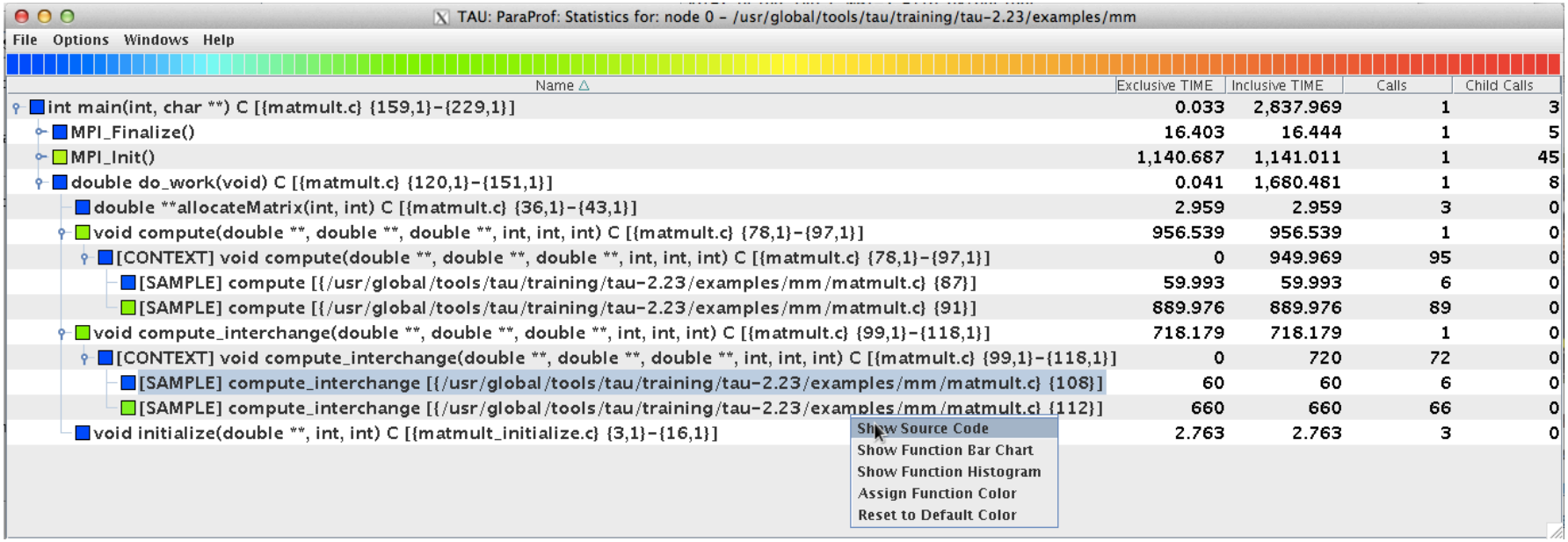
ParaProf Comparison Window



Comparing Rank 0 with 5.

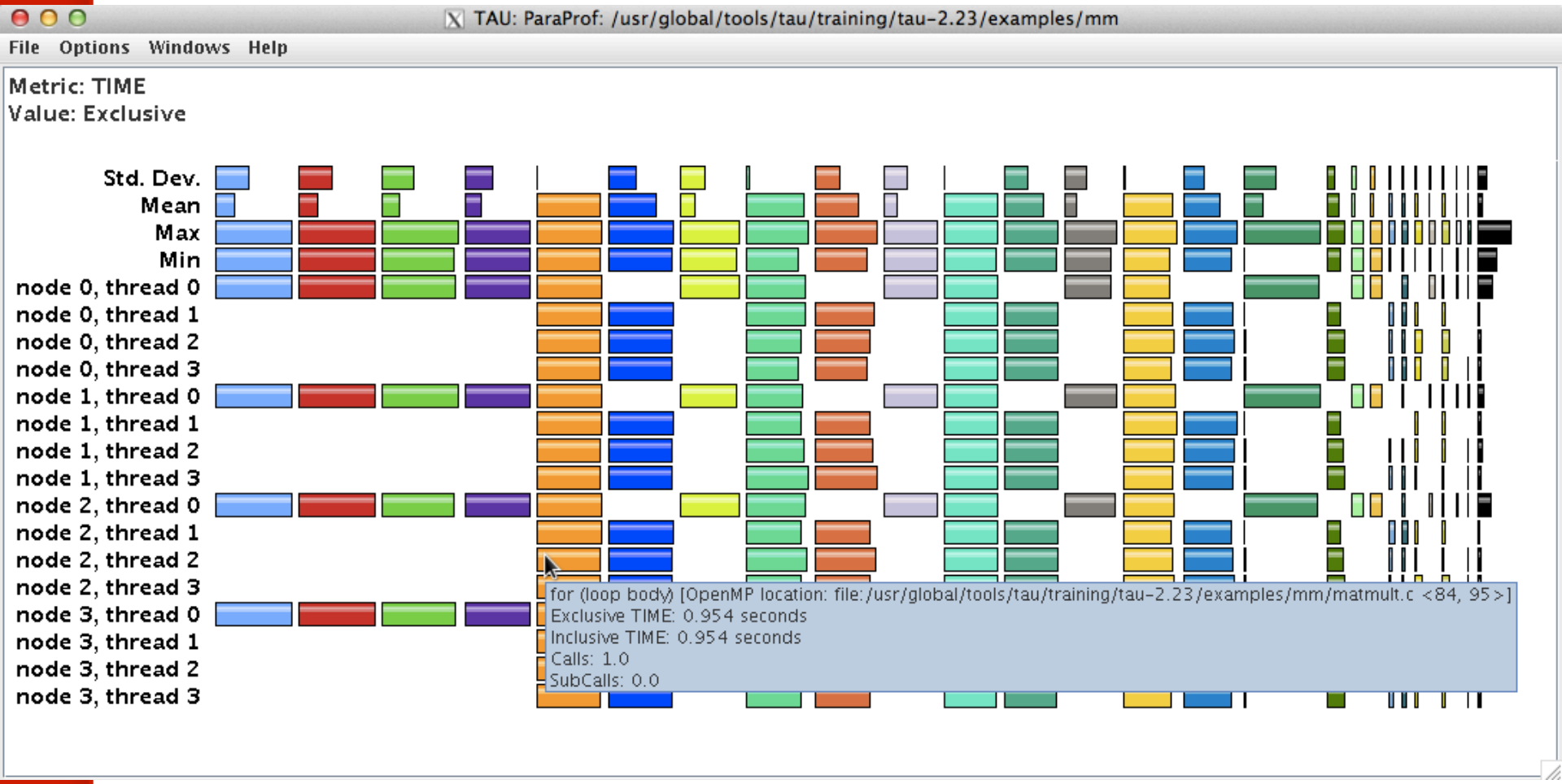
Right click on “node 5” -> Add node to comparison window

Event Based Sampling in TAU



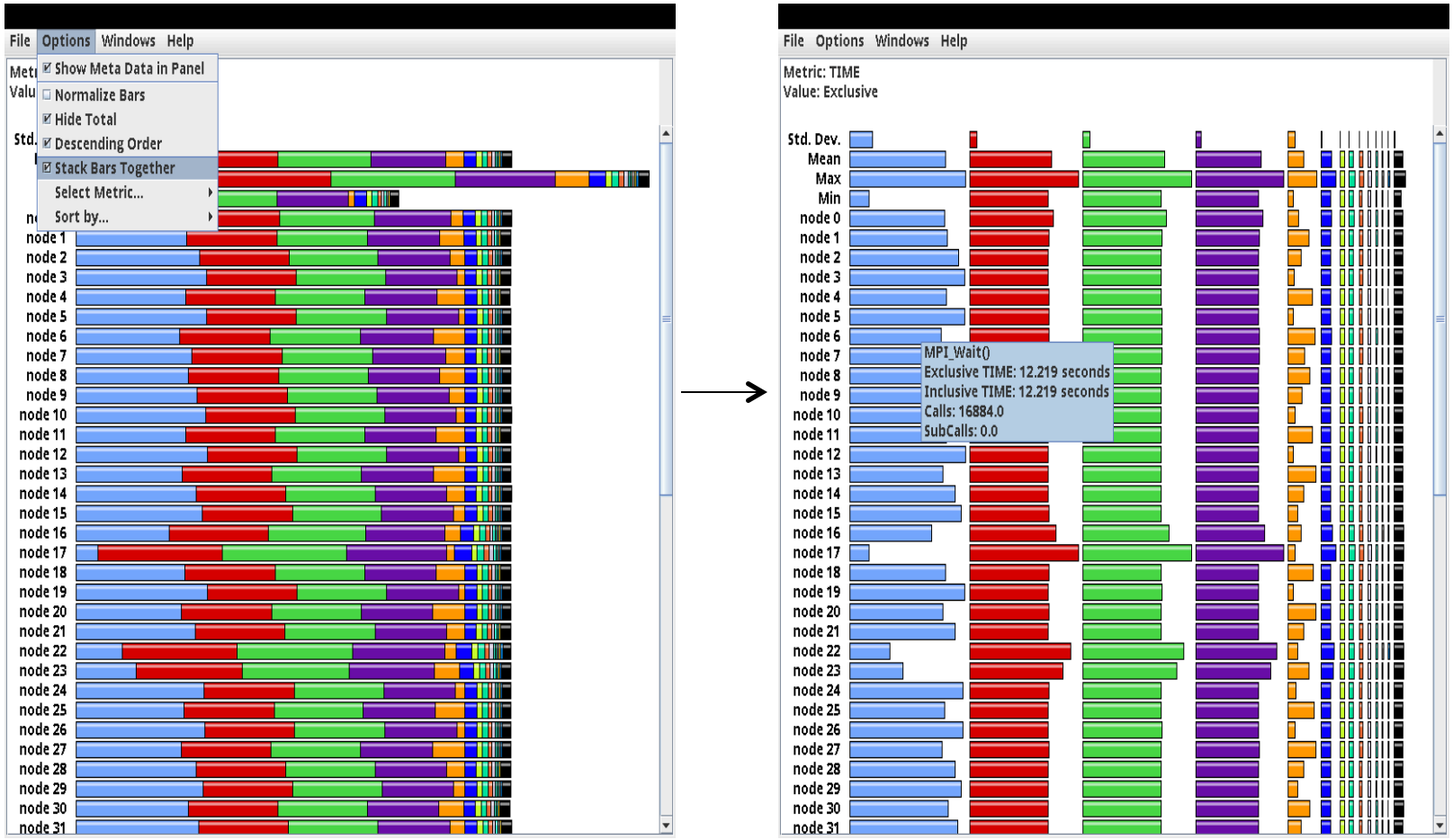

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh CXX=tau_cxx.sh
% export TAU_SAMPLING=1
% srun -n 8 ./a.out
% paraprof
```

Mixed MPI and OpenMP Instrumentation



Options -> Uncheck “Stack Bars Together”

ParaProf Profile Browser



Click “node X” next to see details

Opari OpenMP Instrumentation, Sampling

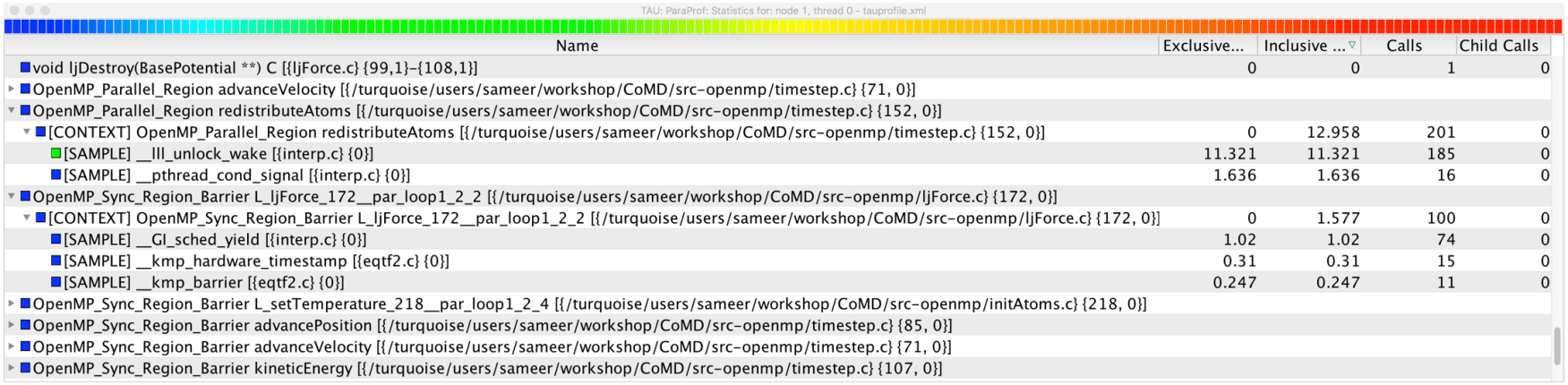
TAU: ParaProf: Statistics for: node 0, thread 0 - /usr/global/tools/tau/training/tau-2.23/examples/mm

File Options Windows Help

Name	Exclu...	Inclu...	C...	C...
int main(int, char **) C [{matmult.pomp.c} {224,1}-{294,1}]	0	3.134	1	3
MPI_Finalize()	0.191	0.191	1	5
MPI_Init_thread()	1.148	1.151	1	45
double do_work(void) C [{matmult.pomp.c} {185,1}-{216,1}]	0.001	1.792	1	8
double **allocateMatrix(int, int) C [{matmult.pomp.c} {38,1}-{45,1}]	0.003	0.003	3	0
void compute(double **, double **, double **, int, int, int) C [{matmult.pomp.c} {111,1}-{146,1}]	0	0.97	1	1
parallel fork/join [OpenMP]	0	0.97	1	1
parallel (parallel fork/join) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <80, 96>]	0	0.97	1	1
parallel begin/end [OpenMP]	0	0.97	1	1
parallel (parallel begin/end) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <80, 96>]	0	0.97	1	2
barrier enter/exit [OpenMP]	0	0.003	1	1
parallel (barrier enter/exit) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <80, 96>]	0.003	0.003	1	0
for enter/exit [OpenMP]	0	0.967	1	1
for (loop body) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <84, 95>]	0.967	0.967	1	0
[CONTEXT] for (loop body) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <84, 95>]	0	0.93	58	0
[SAMPLE] L_compute_118__par_region0_2_204 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.pomp.c} {127}]	0.039	0.039	3	0
[SAMPLE] L_compute_118__par_region0_2_204 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.pomp.c} {131}]	0.891	0.891	55	0
void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.pomp.c} {148,1}-{183,1}]	0	0.812	1	1
parallel fork/join [OpenMP]	0	0.812	1	1
parallel (parallel fork/join) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <101, 117>]	0	0.811	1	1
parallel begin/end [OpenMP]	0	0.811	1	1
parallel (parallel begin/end) [OpenMP location: file:/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c <101, 117>]	0	0.811	1	2
void initialize(double **, int, int) C [{matmult_initialize.pomp.c} {5,1}-{33,1}]	0	0.007	3	3

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt-openmp-opari
% make CC=tau_cc.sh CXX=tau_cxx.sh
% export TAU_SAMPLING=1; export OMP_NUM_THREADS=16
% srun −n 8 ./a.out
% paraprof
```

TAU's support for OMPT TR6



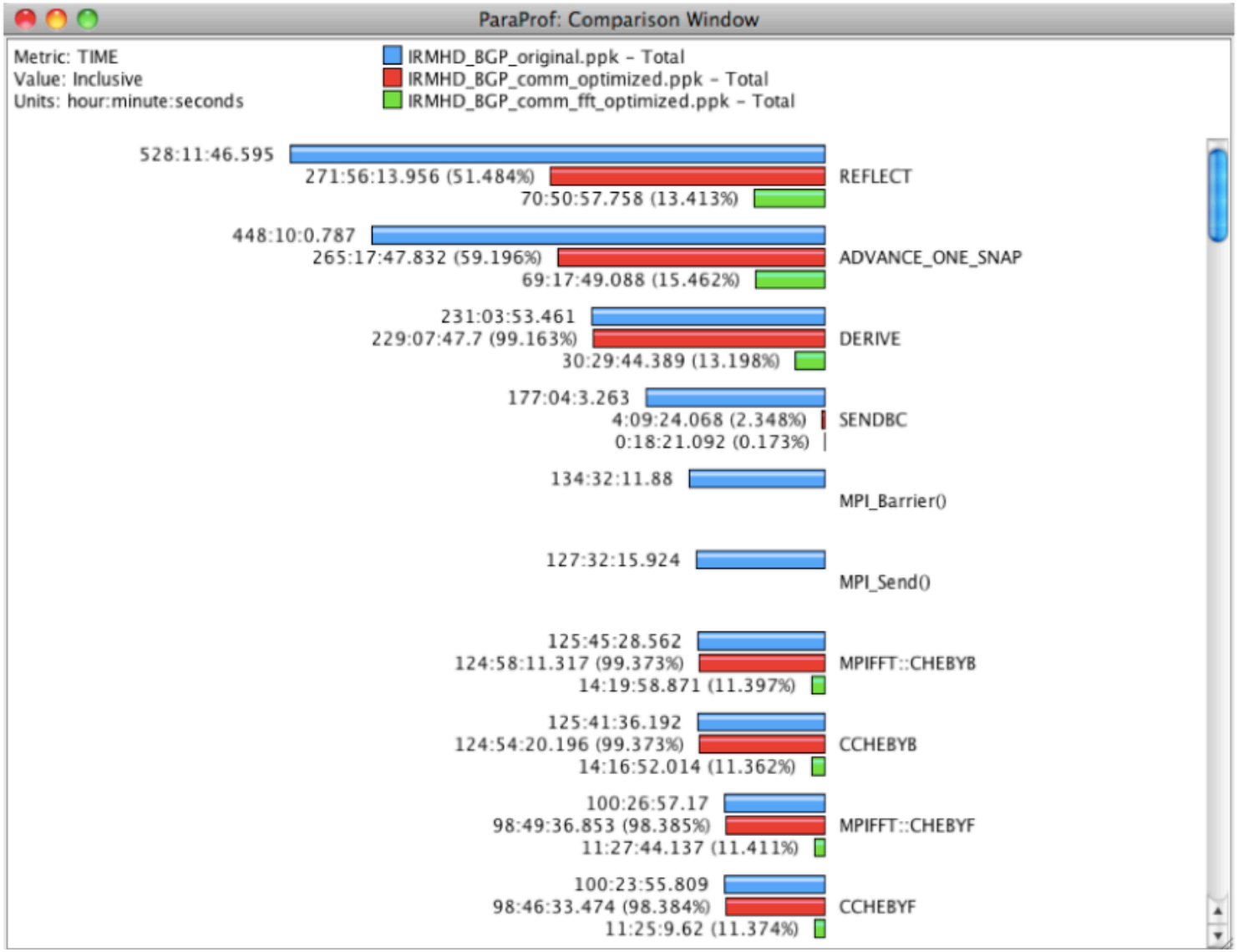
Name	Exclusive...	Inclusive ...	Calls	Child Calls
void ljDestroy(BasePotential **) C [{ljForce.c} {99,1}-{108,1}]	0	0	1	0
OpenMP_Parallel_Region advanceVelocity [{/turquoise/users/sameer/workshop/CoMD/src-openmp/timestep.c} {71, 0}]				
OpenMP_Parallel_Region redistributeAtoms [{/turquoise/users/sameer/workshop/CoMD/src-openmp/timestep.c} {152, 0}]				
[CONTEXT] OpenMP_Parallel_Region redistributeAtoms [{/turquoise/users/sameer/workshop/CoMD/src-openmp/timestep.c} {152, 0}]	0	12.958	201	0
[SAMPLE] __lll_unlock_wake [{interp.c} {0}]	11.321	11.321	185	0
[SAMPLE] __pthread_cond_signal [{interp.c} {0}]	1.636	1.636	16	0
OpenMP_Sync_Region_Barrier L_ljForce_172__par_loop1_2_2 [{/turquoise/users/sameer/workshop/CoMD/src-openmp/ljForce.c} {172, 0}]				
[CONTEXT] OpenMP_Sync_Region_Barrier L_ljForce_172__par_loop1_2_2 [{/turquoise/users/sameer/workshop/CoMD/src-openmp/ljForce.c} {172, 0}]	0	1.577	100	0
[SAMPLE] __GI_sched_yield [{interp.c} {0}]	1.02	1.02	74	0
[SAMPLE] __kmp_hardware_timestamp [{eqtf2.c} {0}]	0.31	0.31	15	0
[SAMPLE] __kmp_barrier [{eqtf2.c} {0}]	0.247	0.247	11	0
OpenMP_Sync_Region_Barrier L_setTemperature_218__par_loop1_2_4 [{/turquoise/users/sameer/workshop/CoMD/src-openmp/initAtoms.c} {218, 0}]				
OpenMP_Sync_Region_Barrier advancePosition [{/turquoise/users/sameer/workshop/CoMD/src-openmp/timestep.c} {85, 0}]				
OpenMP_Sync_Region_Barrier advanceVelocity [{/turquoise/users/sameer/workshop/CoMD/src-openmp/timestep.c} {71, 0}]				
OpenMP_Sync_Region_Barrier kineticEnergy [{/turquoise/users/sameer/workshop/CoMD/src-openmp/timestep.c} {107, 0}]				

Configure TAU with –ompt=download (without –opari)

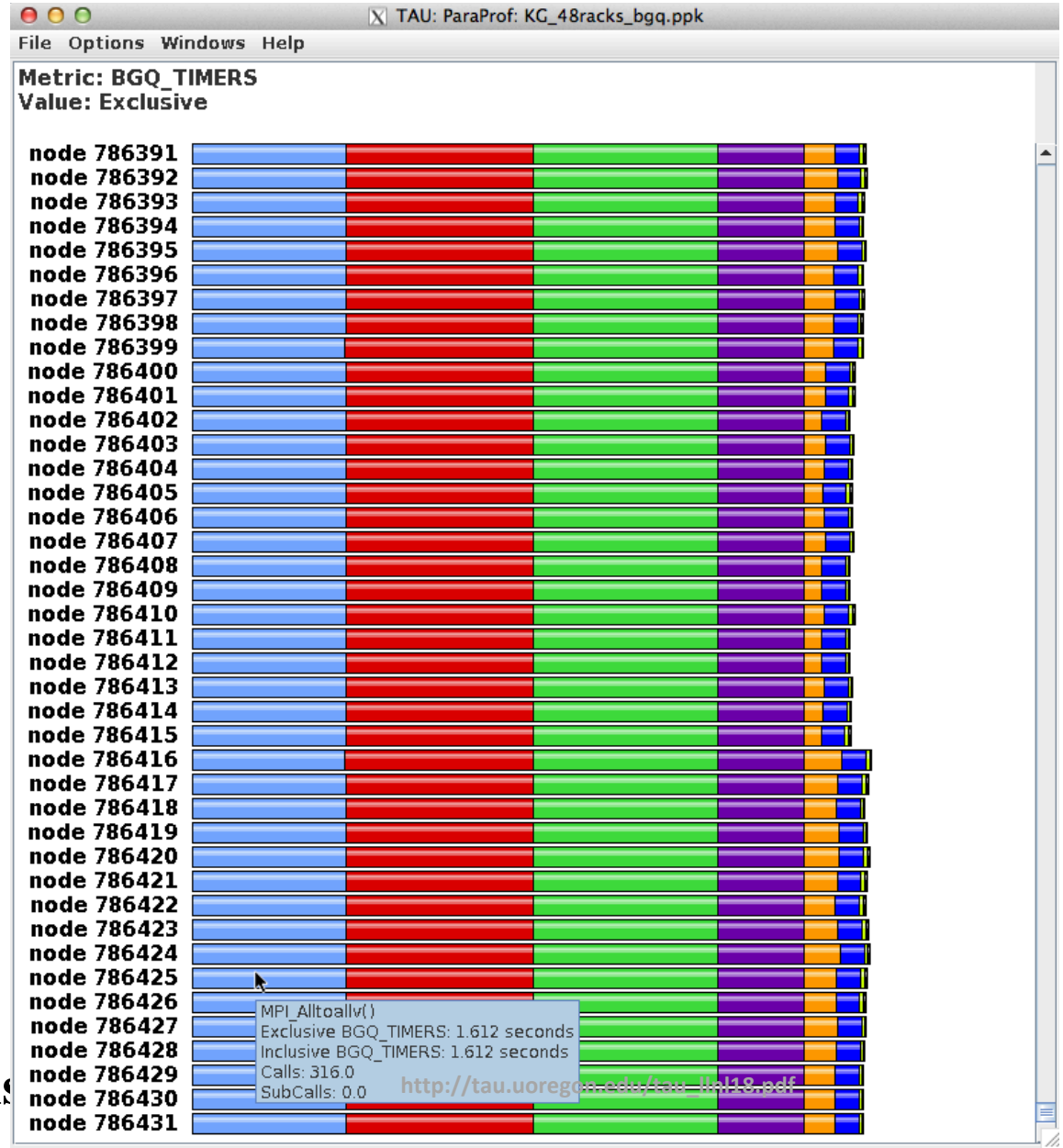
```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-ompt-tr6-mpi-pdt-openmp
% export TAU_OMPT_SUPPORT_LEVEL=full
% export TAU_OMPT_RESOLVE_ADDRESS_EAGERLY=1
% make CC=tau_cc.sh CXX=tau_cxx.sh
% export OMP_NUM_THREADS=16
% srun -n 8 tau_exec –T ompt,tr6,papi,pdt  –ompt -ebs ./a.out
% paraprof
```

NOTE: Instrumentation is at the source, MPI, and OpenMP levels with sampling

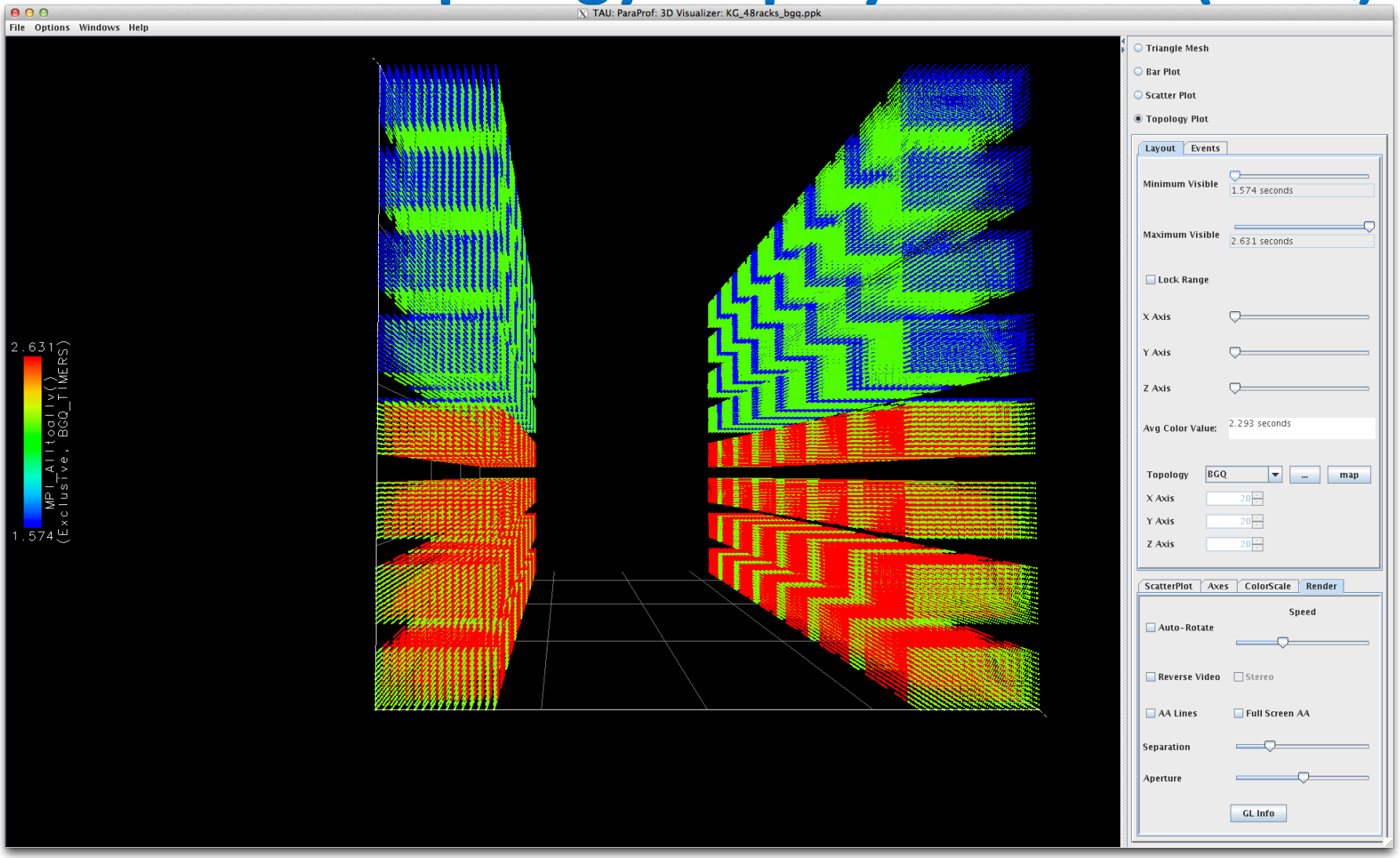
ParaProf Comparison Window



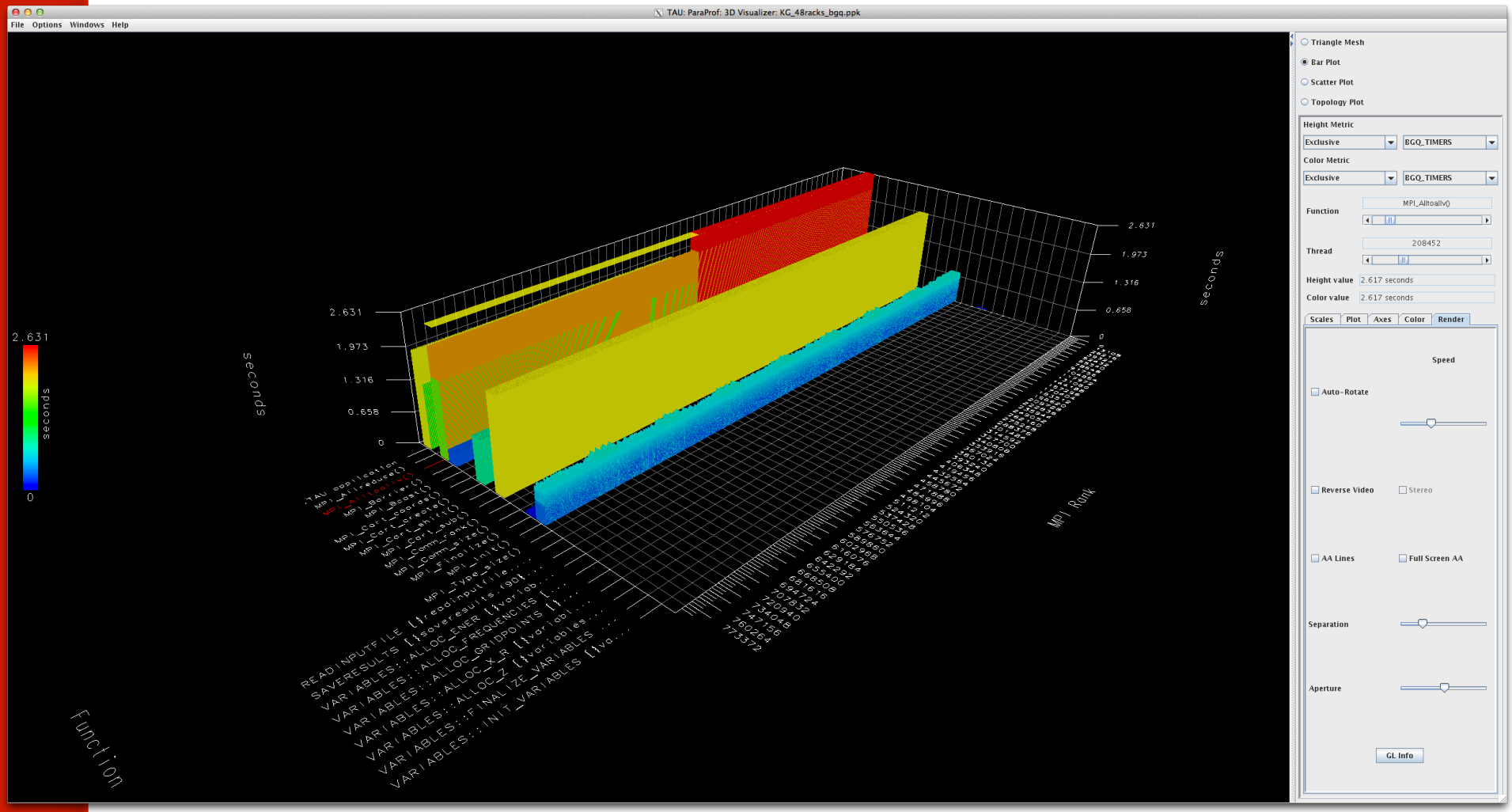
TAU's ParaProf Profile Browser



ParaProf's Topology Display Window (BGQ)



ParaProf's Scalable 3D Visualization (BGQ)



786,432 ranks

TAU's Runtime Merging of Profile Data

TAU: ParaProf Manager

File Options Help

- Applications
  - Standard Applications
    - Default App
      - Default Exp
        - KG_48racks_bgq.ppk
          - BGQ_TIMERS
  - Default (jdbc:h2:/home3/sameer/
  - perfexplorer_working (jdbc:h2:/h

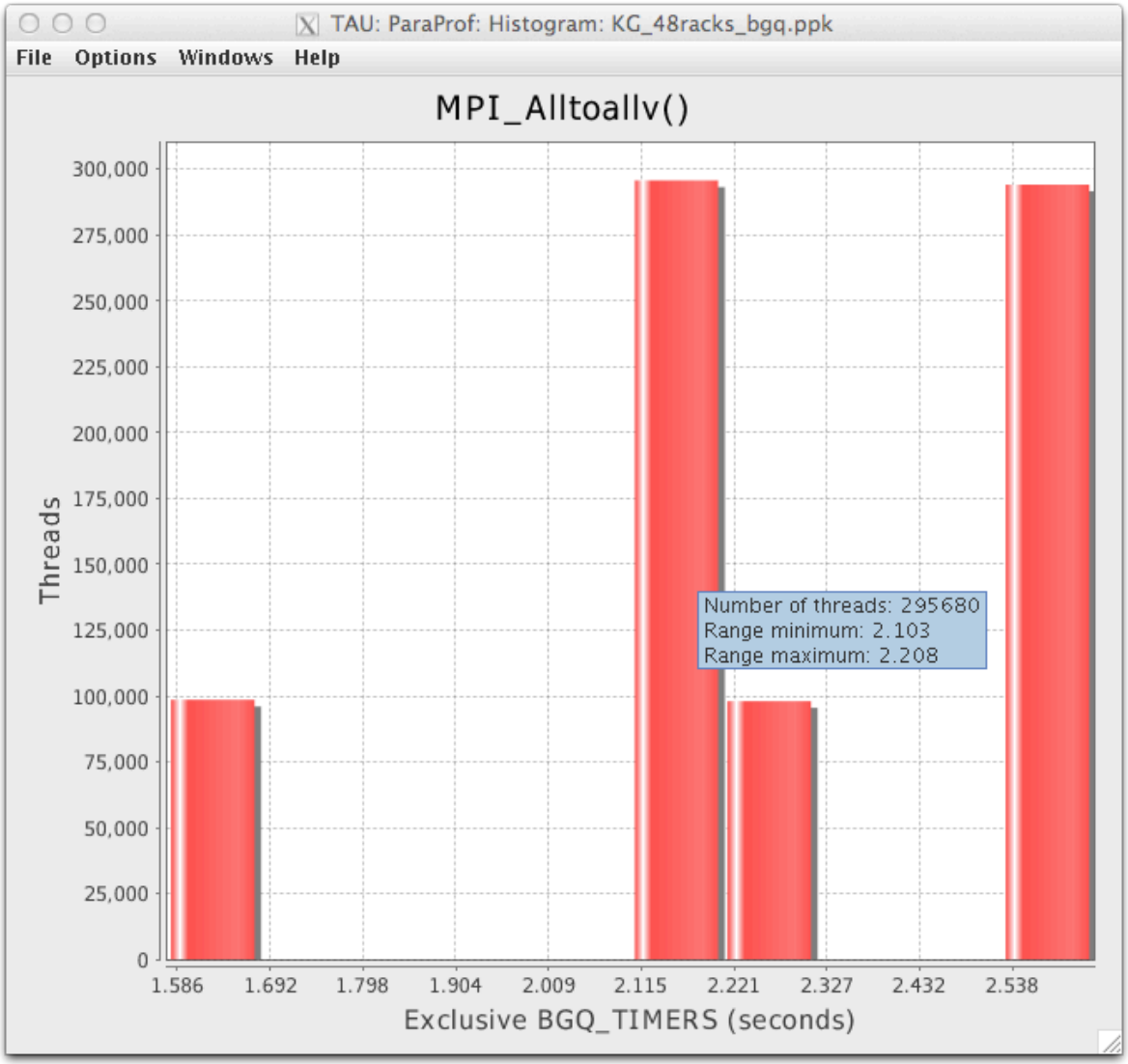
TrialField	Value
Name	KG_48racks_bgq.ppk
Application ID	0
Experiment ID	0
Trial ID	0
BGQ Block Thread ID	3342328
BGQ Coords	(188,108,272)
BGQ DDR Size (MB)	16384
BGQ Job ID	221109
BGQ Node ID	77280
BGQ Node Name	R2b-M0-N0f-J00 <8,12,16,16,2>
BGQ Period	(0,0,0)
BGQ Physical HW Thread ID	0
BGQ Physical Processor ID	15
BGQ Process Count	16
BGQ Processor Core ID	15
BGQ Processor Count	4
BGQ Processor ID	60
BGQ Processor Thread ID	0
BGQ Rank	786431
BGQ Size	(8,12,16,16,2,64)
BGQ tCoord	15
CPU MHz	1600.000000MHz
CPU Type	A2 (Blue Gene/Q)
CWD	/gpfs/mira-fs0/projects/MiraBootCamp2013/KG/tau/49152
Command Line	/gpfs/mira-fs0/projects/MiraBootCamp2013/KG/tau/49152/./Kg
Executable	/gpfs/mira-fs0/projects/MiraBootCamp2013/KG/tau/49152/./Kg
File Type Index	0
File Type Name	ParaProf Packed Profile
Hostname	Q2H-I3-J03.mira.i2b
Local Time	2013-05-24T19:20:06+00:00
MPI Processor Name	Task 786431 of 786432 (7,11,15,15,1,15) R2B-M0-N15-J00
Memory Size	16718464 kB
Node Name	Q2H-I3-J03.mira.i2b
OS Machine	BGQ
OS Name	CNK
OS Release	2.6.32-279.14.1.bgq.el6_V1R2M0_26.ppc64
OS Version	1
Starting Timestamp	1369423205614897
TAU Architecture	bgq
TAU Config	-BGQTIMERS -arch=bgq -pdt=/home/projects/tau/pdt_latest -pdt_c++=xlC -mpi -papi=/soft/perftools/tau/papi_latest -bfd=/home/projects/tau/tau2/bgq/binutils-2.20 -iowrapper
TAU Makefile	/soft/perftools/tau/tau-2.22.2p1/bgq/lib/Makefile.tau-bgqtimers-papi-mpi-pdt
TAU MetaData Merge Time	0.000545 seconds
TAU Profile Merge Time	47.34 seconds
TAU Unification Time	0.01323 seconds
TAU Version	2.22.2
TAU_CALLPATH	off
TAU_CALLPATH_DEPTH	2
TAU_CALLSITE_LIMIT	1

47.34 seconds

% export TAU_PROFILE_FORMAT=merged

It took ~48 seconds to merge and write profiles from 786,432 ranks

ParaProf Histogram Display



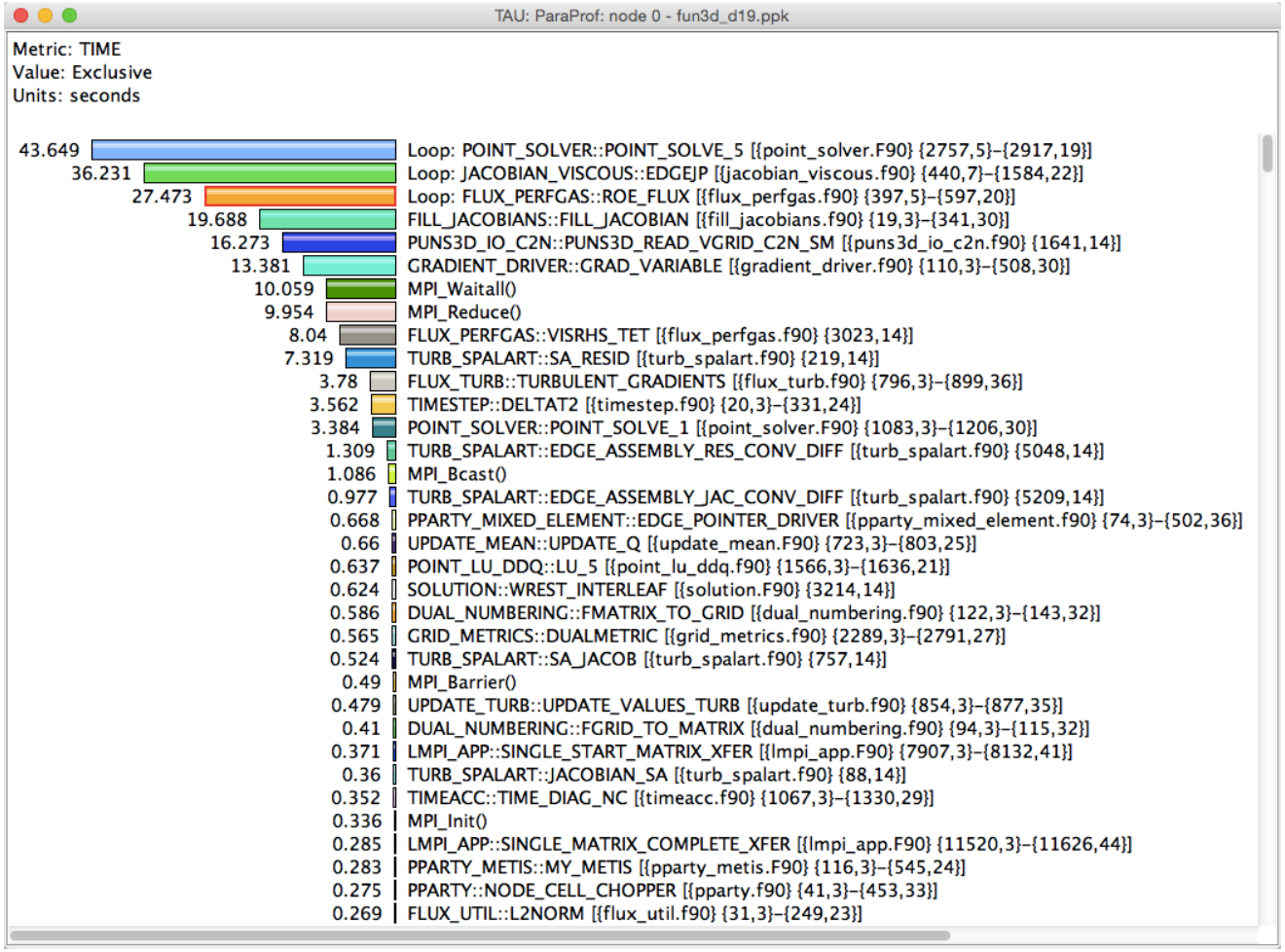
Generating a loop level profile

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION

% module load tau
% make CC=tau_cc.sh
(Or edit Makefile and change CC=tau_cc.sh )

% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.

% paraprof app.ppk
Also, you may use export TAU_SELECT_FILE=select.tau to
filter events at runtime!
```

Loop Level Instrumentation

Goal: What loops account for the most time? How much?

Flat profile with wallclock time with loop instrumentation:



Tools: PAPI

PAPI

- Consistent interface to hardware performance counters
- Preset and native events
- Higher level tools use PAPI
  - TAU, Score-P, HPCToolkit, VampirTrace, Open|SpeedShop
- API, Library, Tools
  - papi_avail - shows the list of preset counters
  - papi_native_avail – shows the list of native processor specific counters
  - papi_event_chooser – allows you to find a list of compatible events
  - papi_decode – shows how a preset event is comprised of native events
  - Components – perf, RAPL for energy, network counters
- ***From University of Tenessee, Knoxville***
  - http://icl.cs.utk.edu/papi

PAPI's Preset Cache Events

tg457572@c455-073.stampede2:~/tmp/mm — ssh stampede — 84×47

```
c455-073[knl](24)$ papi_avail --ca
Available PAPI preset and user defined events plus hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 5.5.1.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz (87)
CPU Revision             : 1.000000
CPUID Info               : Family: 6  Model: 87  Stepping: 1
CPU Max Megahertz        : 1600
CPU Min Megahertz        : 1000
Hdw Threads per core     : 4
Cores per Socket         : 68
Sockets                  : 1
NUMA Nodes               : 1
CPUs per Node            : 272
Total CPUs               : 272
Running in a VM          : no
Number Hardware Counters : 5
Max Multiplex Counters   : 384
--------------------------------------------------------------------------------


================================================================================
  PAPI Preset Events
================================================================================
    Name        Code    Deriv Description (Note)
PAPI_L1_DCM  0x80000000  No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  No   Level 1 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes  Level 1 cache misses
PAPI_L2_TCM  0x80000007  No   Level 2 cache misses
PAPI_TLB_DM  0x80000014  No   Data translation lookaside buffer misses
PAPI_L1_LDM  0x80000017  No   Level 1 load misses
PAPI_L2_LDM  0x80000019  No   Level 2 load misses
PAPI_STL_ICY 0x80000025  No   Cycles with no instruction issue
PAPI_BR_UCN  0x8000002a  Yes  Unconditional branch instructions
PAPI_BR_CN   0x8000002b  No   Conditional branch instructions
PAPI_BR_TKN  0x8000002c  No   Conditional branch instructions taken
PAPI_BR_NTK  0x8000002d  Yes  Conditional branch instructions not taken
PAPI_BR_MSP  0x8000002e  No   Conditional branch instructions mispredicted
PAPI_TOT_INS 0x80000032  No   Instructions completed
PAPI_LD_INS  0x80000035  No   Load instructions
PAPI_SR_INS  0x80000036  No   Store instructions
PAPI_BR_INS  0x80000037  No   Branch instructions
PAPI_RES_STL 0x80000039  No   Cycles stalled on any resource
PAPI_TOT_CYC 0x8000003b  No   Total cycles
PAPI_LST_INS 0x8000003c  Yes  Load/store instructions completed
PAPI_L1_DCA  0x80000040  Yes  Level 1 data cache accesses
PAPI_L1_ICH  0x80000049  No   Level 1 instruction cache hits
```

% papi_avail –ca

PAPI's Native Events for powercap

sameer@grover:lulesh-2.0.3 — ssh zorak — 96×50

```
Native Events in Component: powercap
| powercap:::ENERGY_UJ:ZONE0
|            package-0
| powercap:::MAX_ENERGY_RANGE_UJ:ZONE0
|            package-0
| powercap:::MAX_POWER_A_UW:ZONE0
|            package-0
| powercap:::POWER_LIMIT_A_UW:ZONE0
|            package-0
| powercap:::TIME_WINDOW_A_US:ZONE0
|            package-0
| powercap:::MAX_POWER_B_UW:ZONE0
|            package-0
| powercap:::POWER_LIMIT_B_UW:ZONE0
|            package-0
| powercap:::TIME_WINDOW_B_US:ZONE0
|            package-0
| powercap:::ENABLED:ZONE0
|            package-0
| powercap:::ENERGY_UJ:ZONE0_SUBZONE1
|            dram-package-0
| powercap:::MAX_ENERGY_RANGE_UJ:ZONE0_SUBZONE1
|            dram-package-0
| powercap:::MAX_POWER_A_UW:ZONE0_SUBZONE1
|            dram-package-0
| powercap:::POWER_LIMIT_A_UW:ZONE0_SUBZONE1
|            dram-package-0
| powercap:::TIME_WINDOW_A_US:ZONE0_SUBZONE1
|            dram-package-0
| powercap:::ENABLED:ZONE0_SUBZONE1
|            dram-package-0
                                                    3871,79       99%
```

% papi_native_avail

Profiling with multiple counters

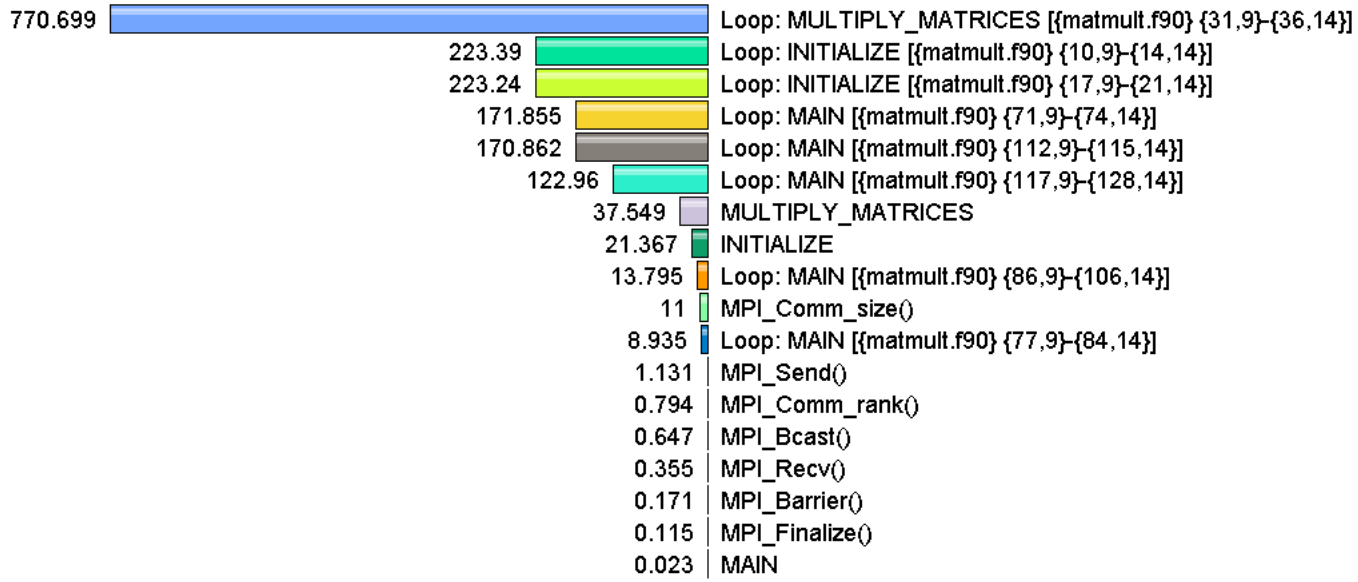
```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION
% make CC=tau_cc.sh

% export TAU_METRICS=TIME,PAPI_TOT_CYC,PAPI_L1_DCM
% srun -n 8  ./matmult
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
  Choose Options -> Show Derived Panel -> Click PAPI_TOT_CYC,
 Click "/",  Click TIME, Apply, Choose new metric by double
clicking.
```

Computing FLOPS per loop

Goal: What is the execution rate of my loops in MFLOPS?

Flat profile with PAPI_FP_INS and time with loop instrumentation:

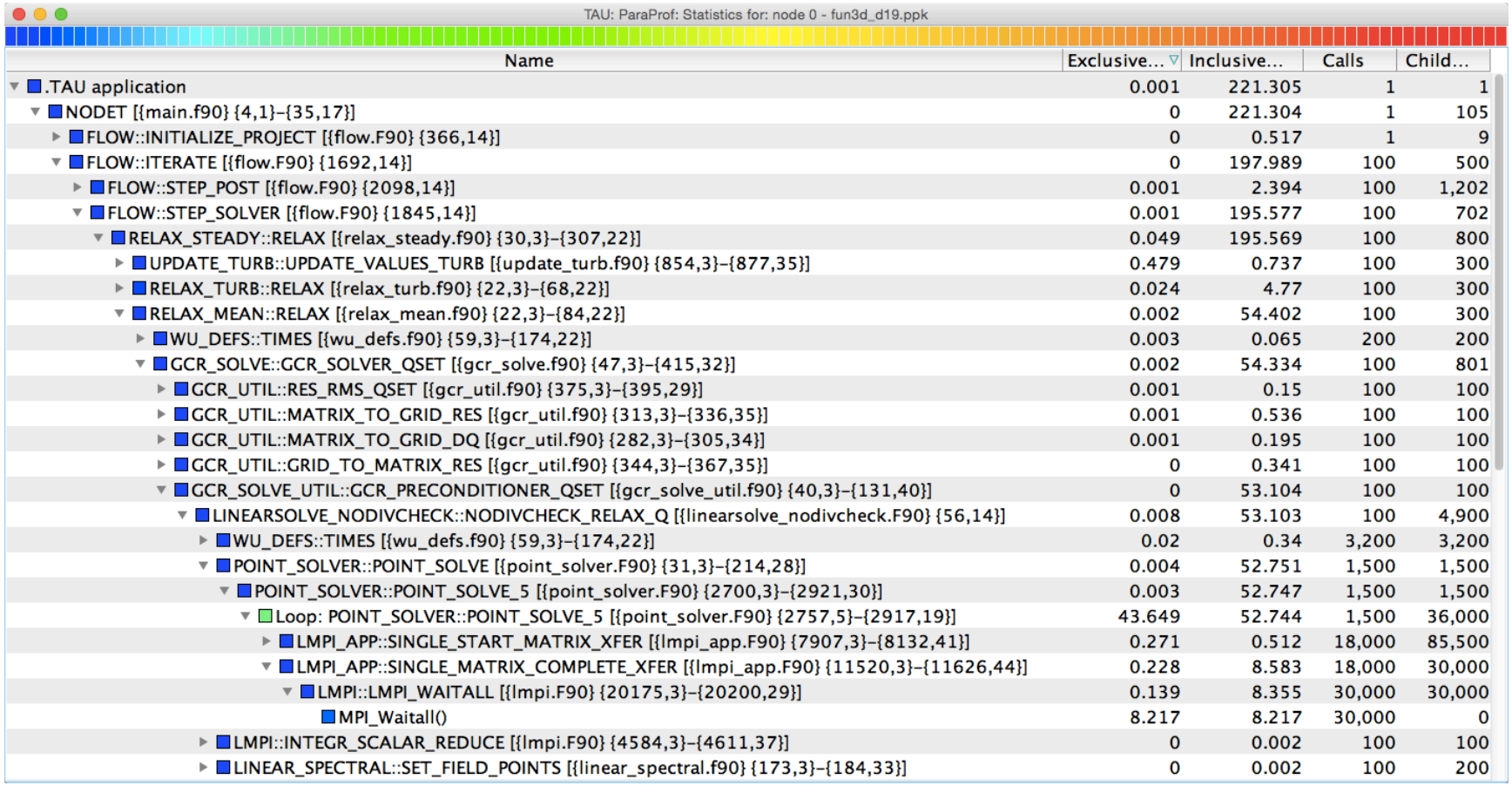
Metric: PAPI_FP_INS / GET_TIME_OF_DAY
Value: Exclusive
Units: Derived metric shown in microseconds format



Generate a Callpath Profile

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh
(Or edit Makefile and change CC=tau_cc.sh )


% export TAU_CALLPATH=1
% export TAU_CALLPATH_DEPTH=100
(truncates all calling paths to a specified depth)
% srun -n 8  ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
(Windows -> Thread -> Call Graph)
```

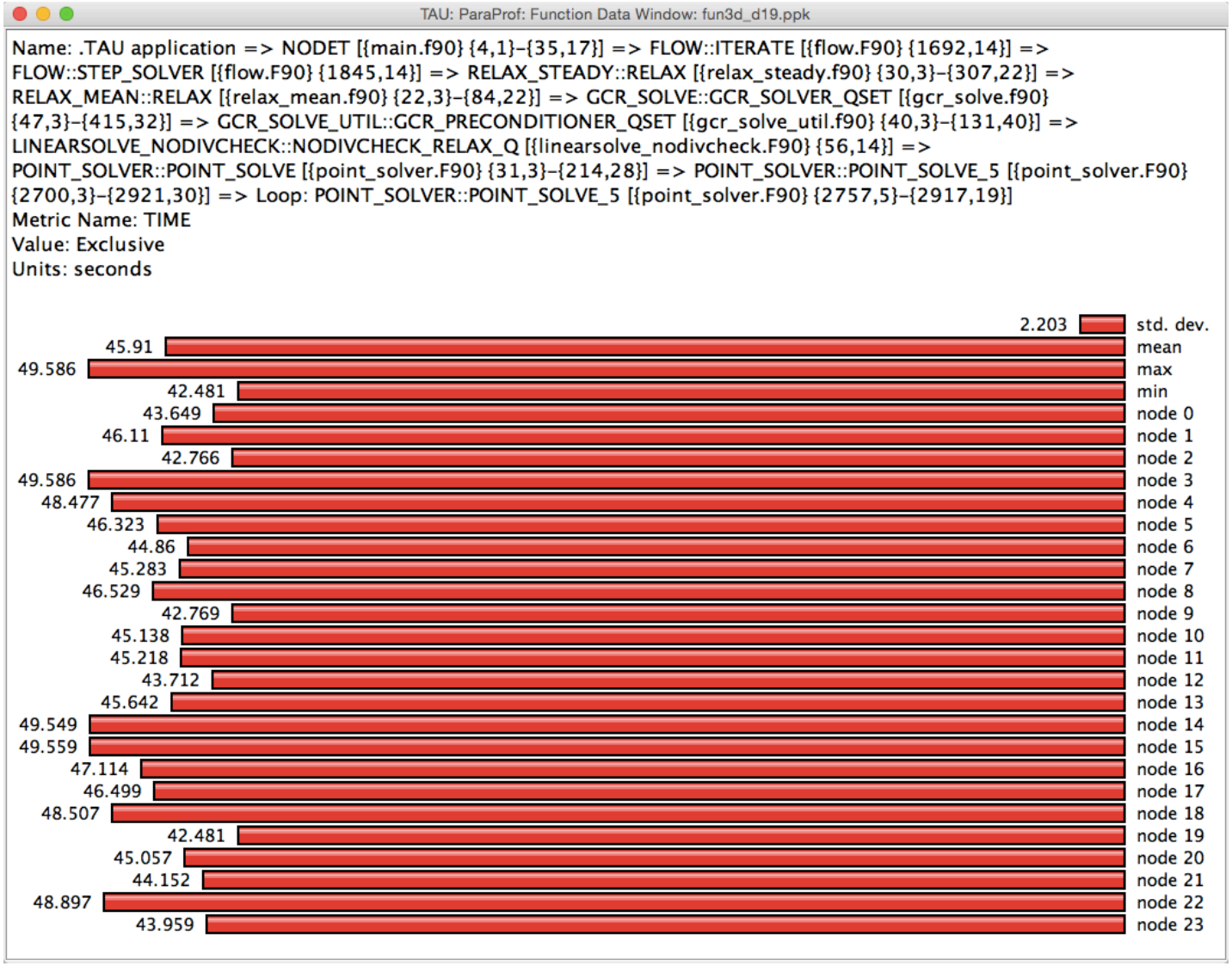
Callpath Profiling: FUN3D



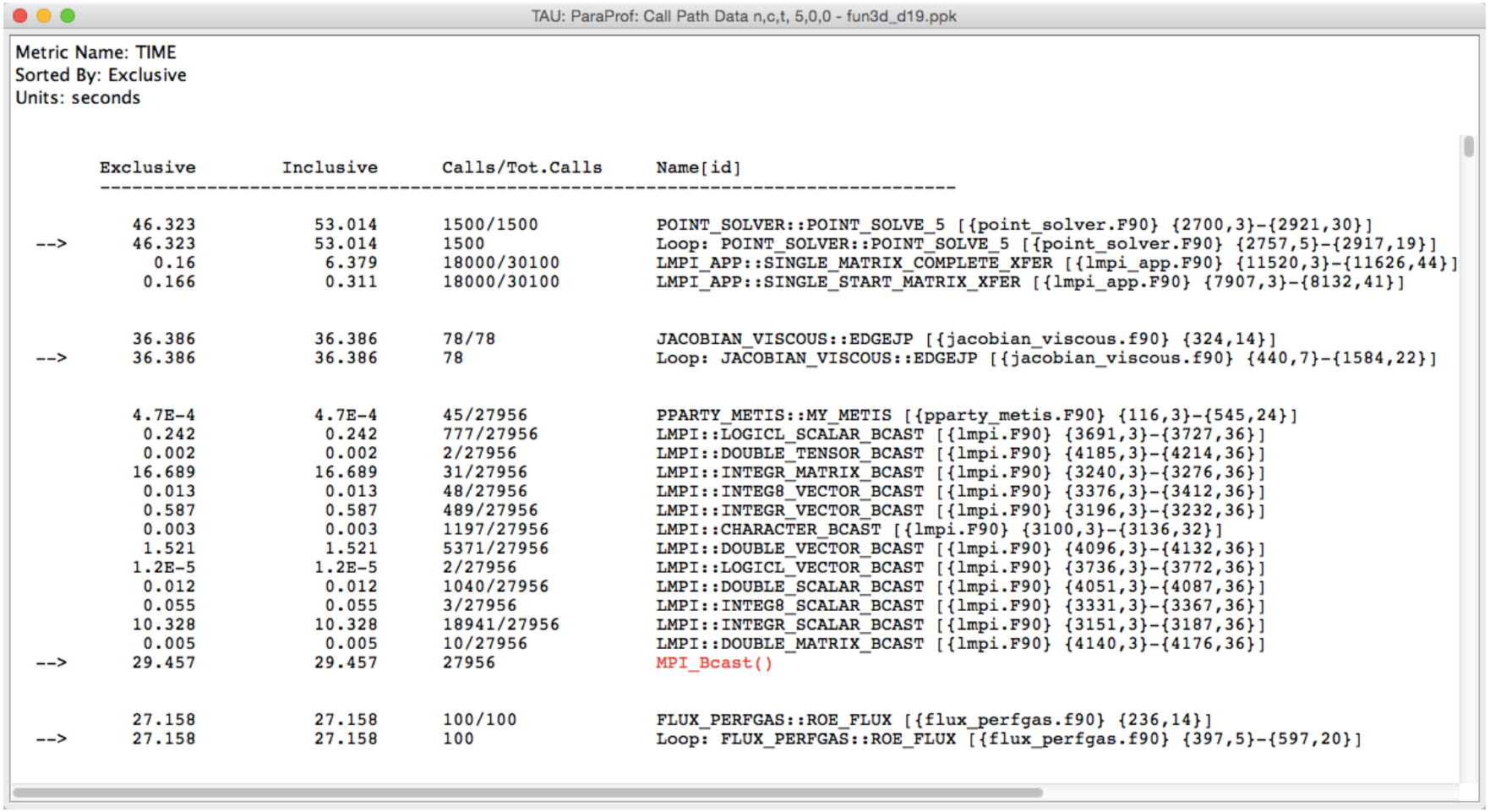
Name	Exclusive...	Inclusive...	Calls	Child...
.TAU application	0.001	221.305	1	1
NODET [{main.f90} {4,1}-{35,17}]	0	221.304	1	105
FLOW::INITIALIZE_PROJECT [{flow.F90} {366,14}]	0	0.517	1	9
FLOW::ITERATE [{flow.F90} {1692,14}]	0	197.989	100	500
FLOW::STEP_POST [{flow.F90} {2098,14}]	0.001	2.394	100	1,202
FLOW::STEP_SOLVER [{flow.F90} {1845,14}]	0.001	195.577	100	702
RELAX_STEADY::RELAX [{relax_steady.f90} {30,3}-{307,22}]	0.049	195.569	100	800
UPDATE_TURB::UPDATE_VALUES_TURB [{update_turb.f90} {854,3}-{877,35}]	0.479	0.737	100	300
RELAX_TURB::RELAX [{relax_turb.f90} {22,3}-{68,22}]	0.024	4.77	100	300
RELAX_MEAN::RELAX [{relax_mean.f90} {22,3}-{84,22}]	0.002	54.402	100	300
WU_DEFS::TIMES [{wu_defs.f90} {59,3}-{174,22}]	0.003	0.065	200	200
GCR_SOLVE::GCR_SOLVER_QSET [{gcr_solve.f90} {47,3}-{415,32}]	0.002	54.334	100	801
GCR_UTIL::RES_RMS_QSET [{gcr_util.f90} {375,3}-{395,29}]	0.001	0.15	100	100
GCR_UTIL::MATRIX_TO_GRID_RES [{gcr_util.f90} {313,3}-{336,35}]	0.001	0.536	100	100
GCR_UTIL::MATRIX_TO_GRID_DQ [{gcr_util.f90} {282,3}-{305,34}]	0.001	0.195	100	100
GCR_UTIL::GRID_TO_MATRIX_RES [{gcr_util.f90} {344,3}-{367,35}]	0	0.341	100	100
GCR_SOLVE_UTIL::GCR_PRECONDITIONER_QSET [{gcr_solve_util.f90} {40,3}-{131,40}]	0	53.104	100	100
LINEARSOLVE_NODIVCHECK::NODIVCHECK_RELAX_Q [{linearsolve_nodivcheck.F90} {56,14}]	0.008	53.103	100	4,900
WU_DEFS::TIMES [{wu_defs.f90} {59,3}-{174,22}]	0.02	0.34	3,200	3,200
POINT_SOLVER::POINT_SOLVE [{point_solver.F90} {31,3}-{214,28}]	0.004	52.751	1,500	1,500
POINT_SOLVER::POINT_SOLVE_5 [{point_solver.F90} {2700,3}-{2921,30}]	0.003	52.747	1,500	1,500
Loop: POINT_SOLVER::POINT_SOLVE_5 [{point_solver.F90} {2757,5}-{2917,19}]	43.649	52.744	1,500	36,000
LMPI_APP::SINGLE_START_MATRIX_XFER [{lmpi_app.F90} {7907,3}-{8132,41}]	0.271	0.512	18,000	85,500
LMPI_APP::SINGLE_MATRIX_COMPLETE_XFER [{lmpi_app.F90} {11520,3}-{11626,44}]	0.228	8.583	18,000	30,000
LMPI::LMPI_WAITALL [{lmpi.F90} {20175,3}-{20200,29}]	0.139	8.355	30,000	30,000
MPI_Waitall()	8.217	8.217	30,000	0
LMPI::INTEGR_SCALAR_REDUCE [{lmpi.F90} {4584,3}-{4611,37}]	0	0.002	100	100
LINEAR_SPECTRAL::SET_FIELD_POINTS [{linear_spectral.f90} {173,3}-{184,33}]	0	0.002	100	200

```
% export TAU_CALLPATH=1
% export TAU_CALLPATH_DEPTH=100
```

ParaProf Function Window



ParaProf Callpath Thread Relations Window



Shows the contribution of parents and children for each routine (marked by an arrow)

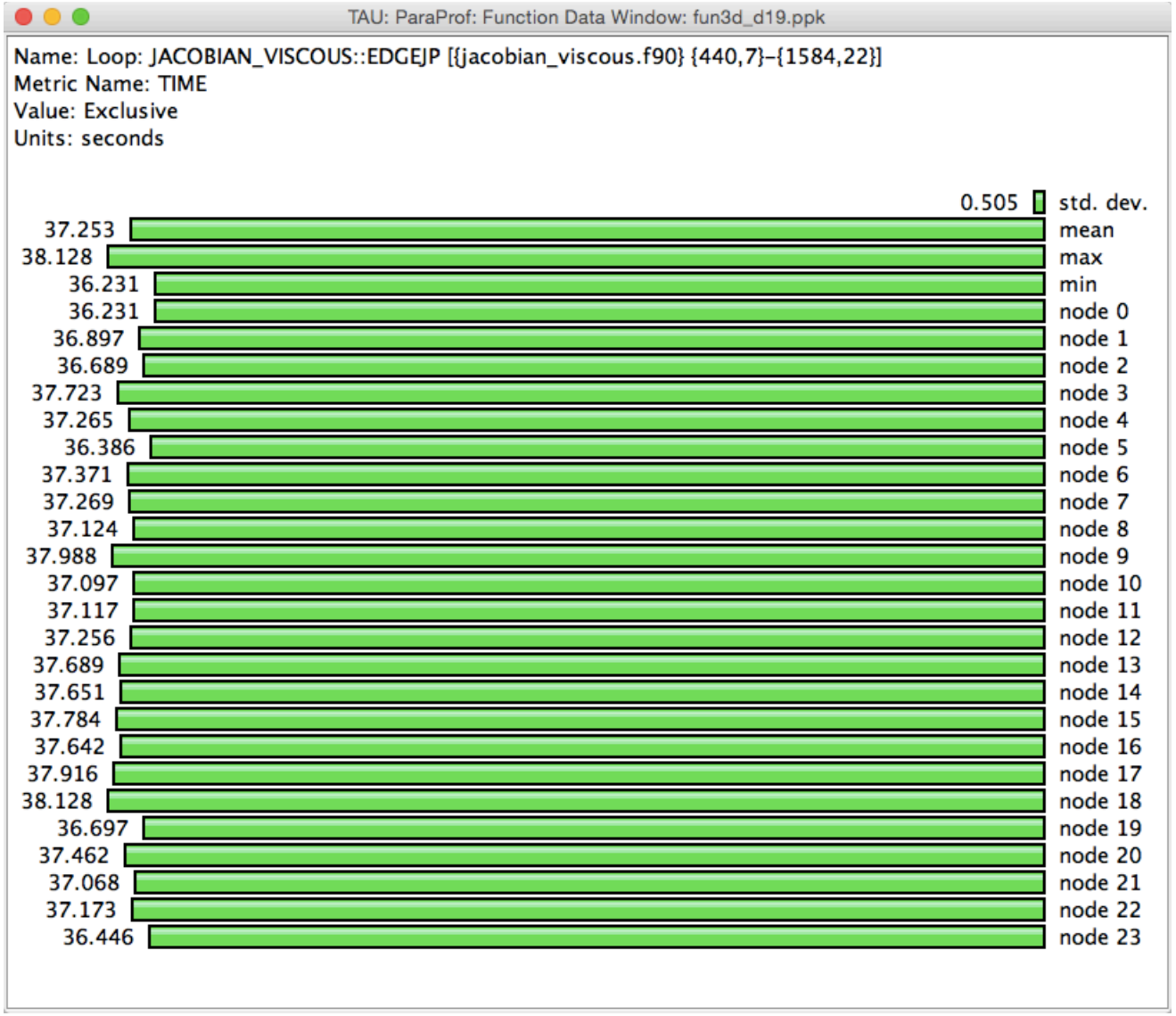
ParaProf Callpath Thread Relations Window

TAU: ParaProf: Call Path Data n,c,t, 13,0,0 - fun3d_d19.ppk

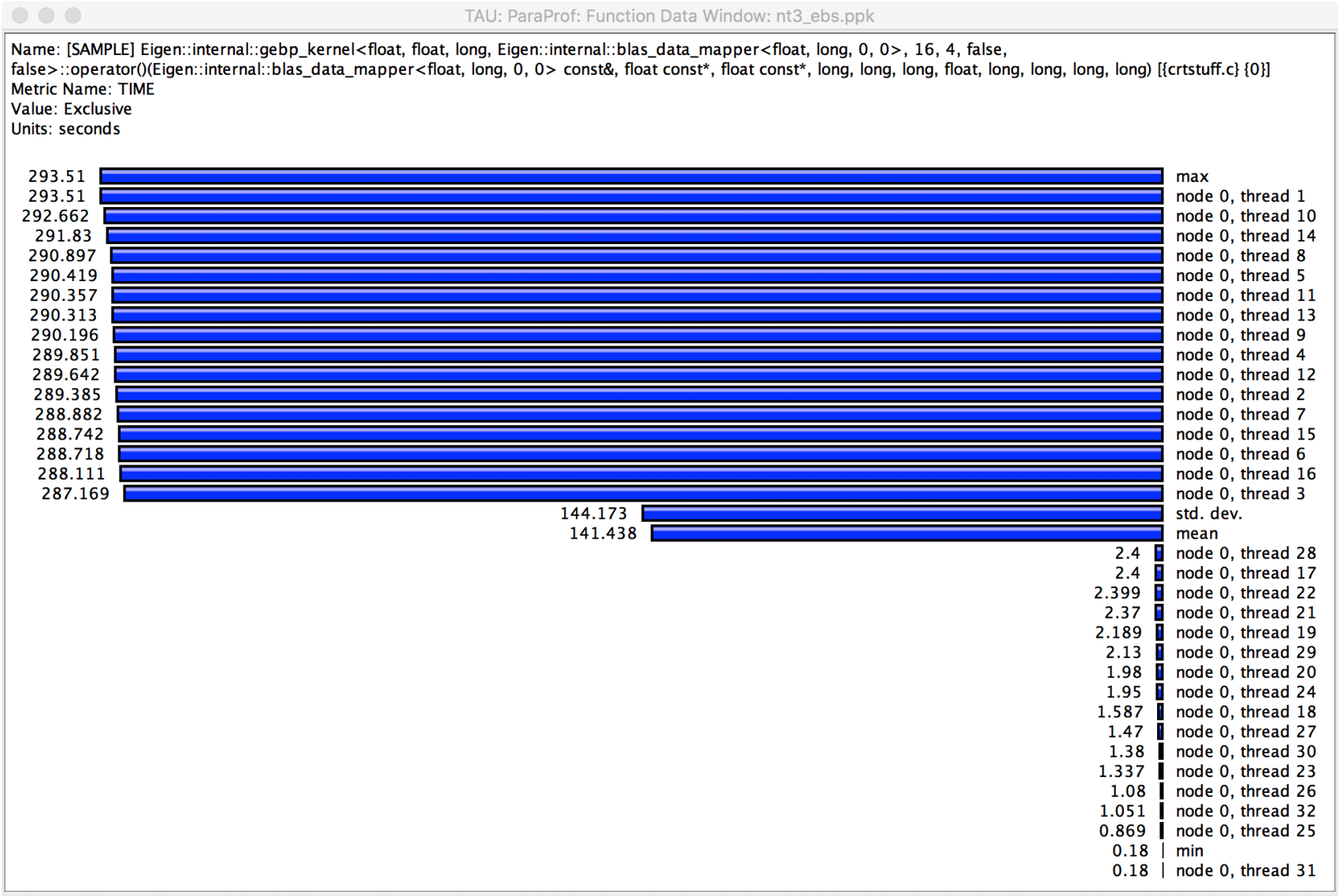
Metric Name: TIME
Sorted By: Exclusive
Units: seconds

	Exclusive	Inclusive	Calls/Tot.Calls	Name[id]
	45.642	52.774	1500/1500	POINT_SOLVER::POINT_SOLVE_5 [{point_solver.F90} {2700,3}-{2921,30}]
-->	45.642	52.774	1500	Loop: POINT_SOLVER::POINT_SOLVE_5 [{point_solver.F90} {2757,5}-{2917,19}]
	0.299	6.259	18000/30100	LMPI_APP::SINGLE_MATRIX_COMPLETE_XFER [{lmpi_app.F90} {11520,3}-{11626,44}]
	0.6	0.873	18000/30100	LMPI_APP::SINGLE_START_MATRIX_XFER [{lmpi_app.F90} {7907,3}-{8132,41}]
	37.689	37.689	78/78	JACOBIAN_VISCOUS::EDGEJP [{jacobian_viscous.f90} {324,14}]
-->	37.689	37.689	78	Loop: JACOBIAN_VISCOUS::EDGEJP [{jacobian_viscous.f90} {440,7}-{1584,22}]
	28.431	28.431	100/100	FLUX_PERFGAS::ROE_FLUX [{flux_perfgas.f90} {236,14}]
-->	28.431	28.431	100	Loop: FLUX_PERFGAS::ROE_FLUX [{flux_perfgas.f90} {397,5}-{597,20}]
	0.003	0.003	1197/27956	LMPI::CHARACTER_BCAST [{lmpi.F90} {3100,3}-{3136,32}]
	0.542	0.542	489/27956	LMPI::INTEGR_VECTOR_BCAST [{lmpi.F90} {3196,3}-{3232,36}]
	0.033	0.033	3/27956	LMPI::INTEG8_SCALAR_BCAST [{lmpi.F90} {3331,3}-{3367,36}]
	0.005	0.005	10/27956	LMPI::DOUBLE_MATRIX_BCAST [{lmpi.F90} {4140,3}-{4176,36}]
	16.724	16.724	31/27956	LMPI::INTEGR_MATRIX_BCAST [{lmpi.F90} {3240,3}-{3276,36}]
	0.032	0.032	1040/27956	LMPI::DOUBLE_SCALAR_BCAST [{lmpi.F90} {4051,3}-{4087,36}]
	1.48	1.48	5371/27956	LMPI::DOUBLE_VECTOR_BCAST [{lmpi.F90} {4096,3}-{4132,36}]
	1.5E-5	1.5E-5	2/27956	LMPI::LOGICL_VECTOR_BCAST [{lmpi.F90} {3736,3}-{3772,36}]
	0.002	0.002	2/27956	LMPI::DOUBLE_TENSOR_BCAST [{lmpi.F90} {4185,3}-{4214,36}]
	0.013	0.013	48/27956	LMPI::INTEG8_VECTOR_BCAST [{lmpi.F90} {3376,3}-{3412,36}]
	6.1E-4	6.1E-4	45/27956	PPARTY_METIS::MY_METIS [{pparty_metis.F90} {116,3}-{545,24}]
	5.481	5.481	18941/27956	LMPI::INTEGR_SCALAR_BCAST [{lmpi.F90} {3151,3}-{3187,36}]
	0.243	0.243	777/27956	LMPI::LOGICL_SCALAR_BCAST [{lmpi.F90} {3691,3}-{3727,36}]
-->	24.557	24.557	27956	MPI_Bcast()
	20.045	61.19	78/78	UPDATE_MEAN::UPDATE_JACOBIAN [{update_mean.F90} {513,3}-{588,32}]
-->	20.045	61.19	78	FILL_JACOBIANS::FILL_JACOBIAN [{fill_jacobians.f90} {19,3}-{341,30}]
	1.4E-4	1.4E-4	78/78	SOURCE::SOURCE_JACOBIAN [{source.f90} {93,3}-{168,32}]
	0.006	2.491	3822/16665	LMPI::LMPI_CONDITIONAL_STOP [{lmpi.F90} {611,3}-{672,38}]
	0.003	0.003	3822/8622	BC_NAMES::BC_HAS_PRESSURE_CLOSURE [{bc_names.f90} {1618,3}-{1693,38}]
	0.008	0.008	7644/17444	BC_NAMES::ELEMENT_BASED_BC [{bc_names.f90} {1390,3}-{1439,31}]
	3.2E-4	37.689	78/78	JACOBIAN_VISCOUS::VISCOUS_JACOBIAN [{jacobian_viscous.f90} {20,14}]
	0.443	0.445	78/123	TIMEACC::TIME_DIAG_NC [{timeacc.f90} {1067,3}-{1330,29}]

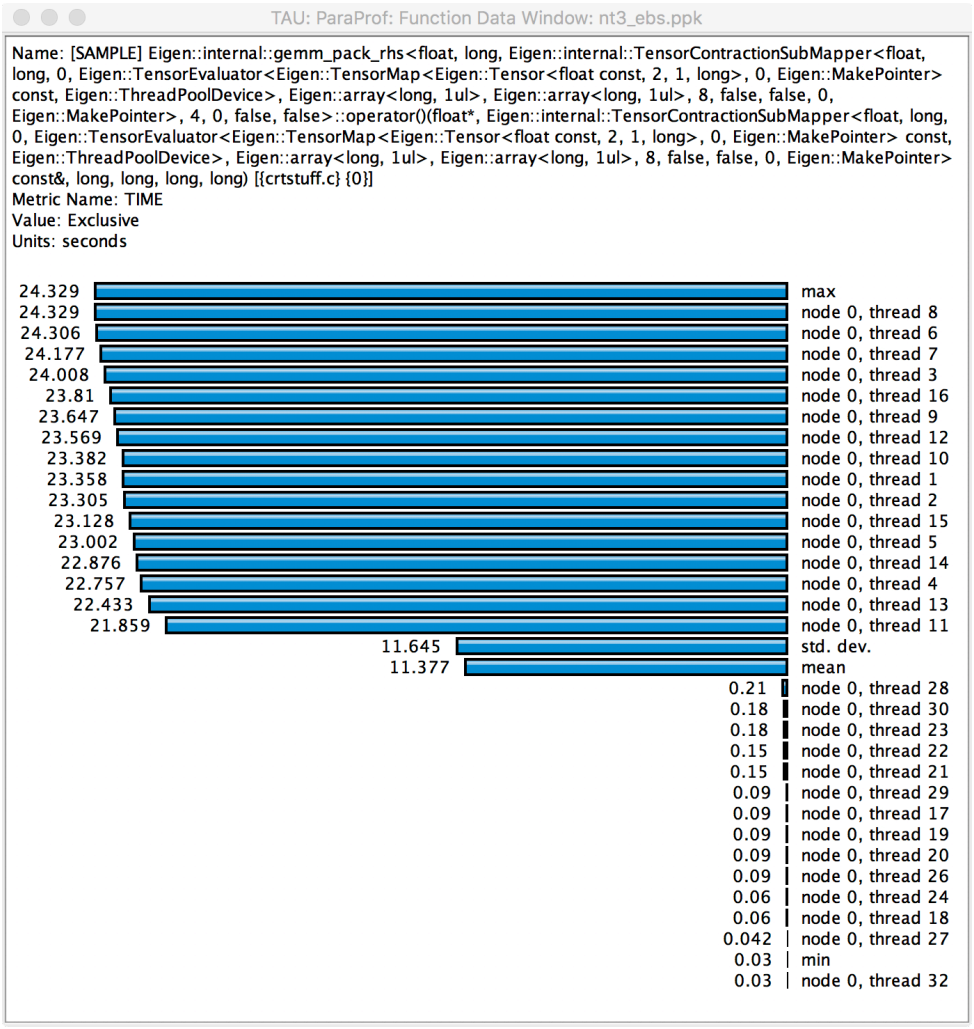
ParaProf Function Window



Event Based Sampling (EBS)



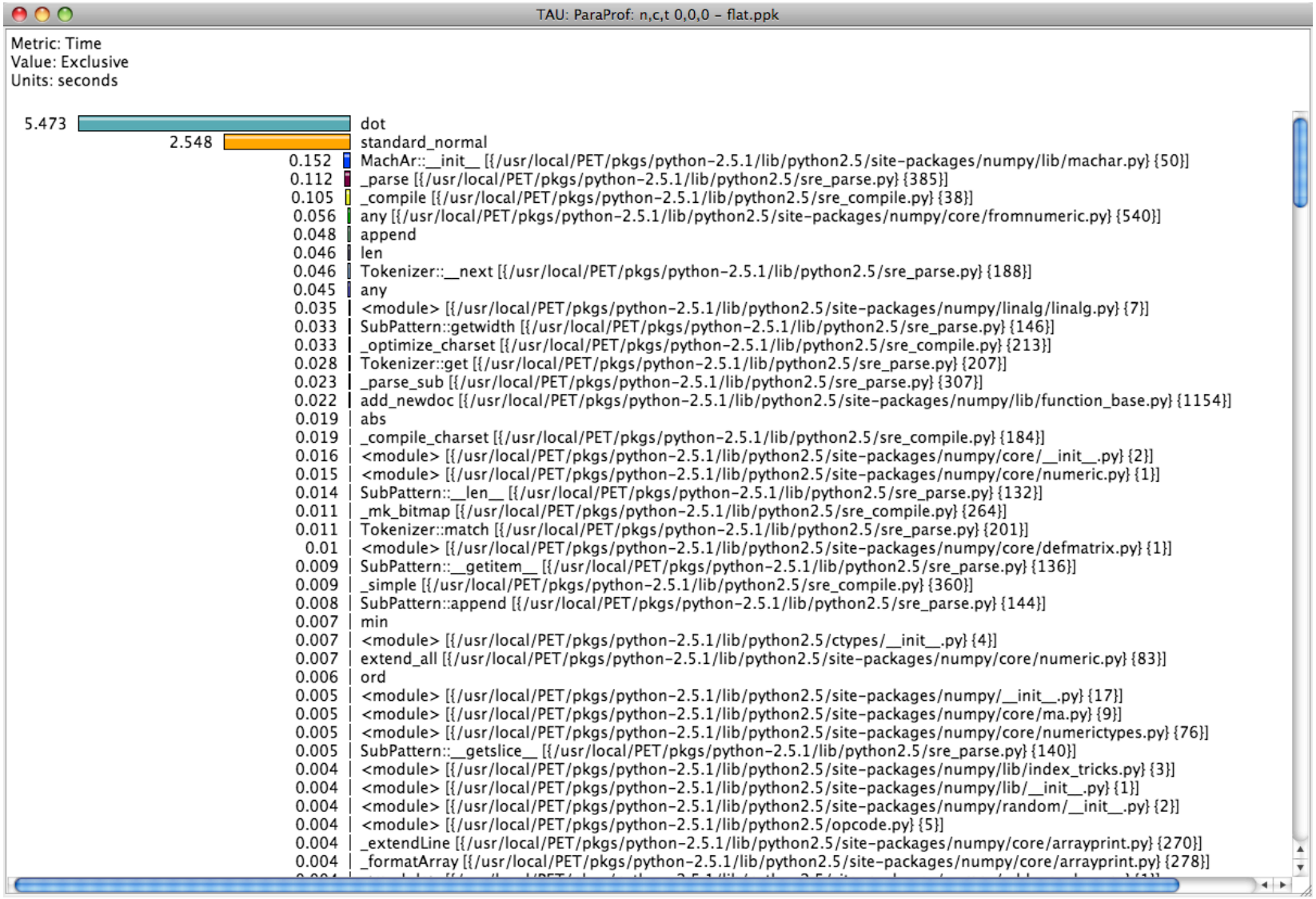
Python (TensorFlow) with EBS



Use tau_python instead of tau_exec to launch:

% tau_python −ebs −T pthread ./foo.py

Profiling Python using tau_python

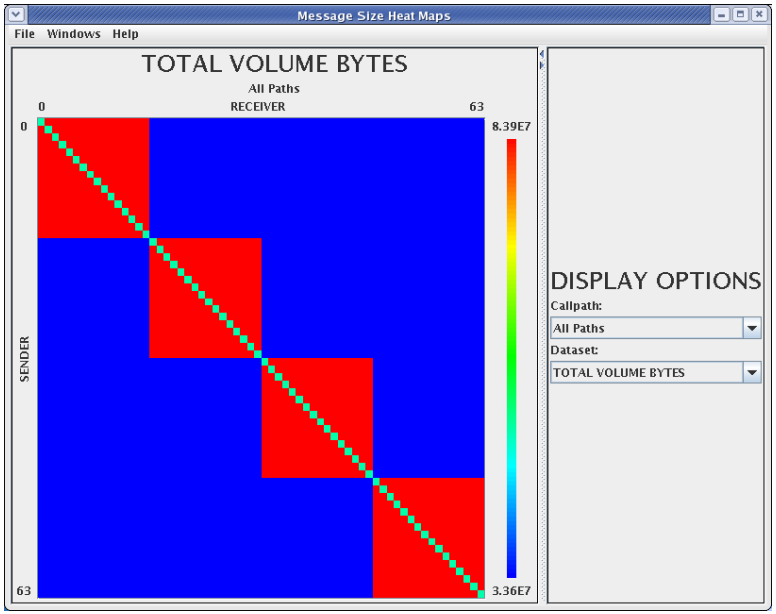


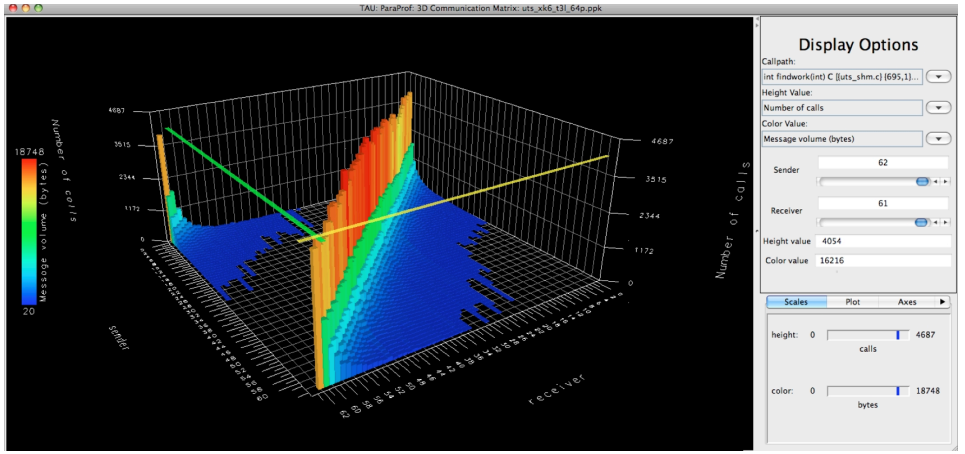
Generating Communication Matrix

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh
(Or edit Makefile and change CC=tau_cc.sh )

% export TAU_COMM_MATRIX=1
% srun -n 8./a.out

% paraprof
(Windows -> Communication Matrix)
(Windows -> 3D Communication Matrix)
```

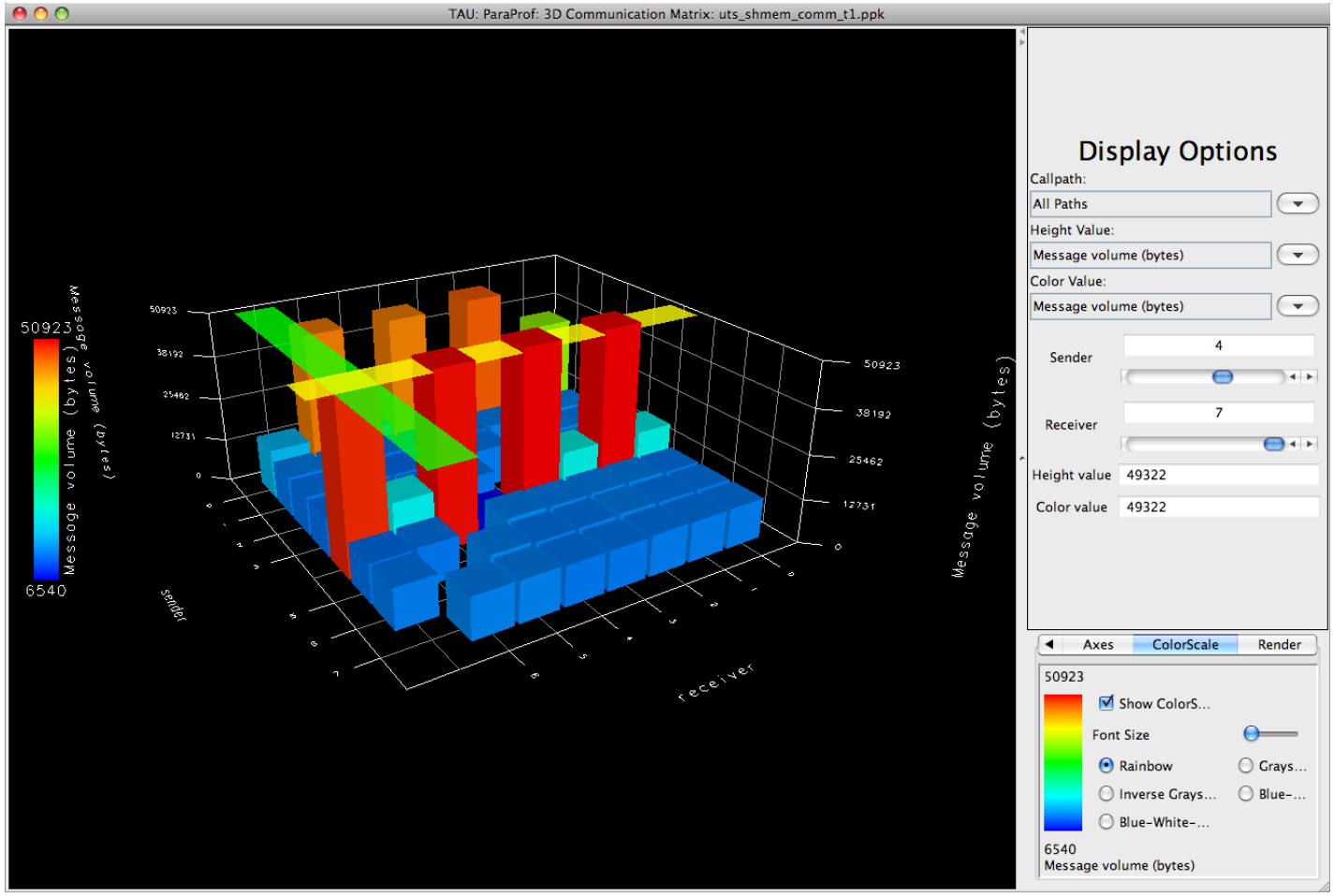
Communication Matrix Display

Goal: What is the volume of inter-process communication? Along which calling path?





SHMEM Communication Matrix



Compiler-based Instrumentation

- Compiler automatically **emits instrumentation calls** in the object code instead of parsing the source code using PDT
- To enable: export TAU_OPTIONS="-optCompInst"
- Configure TAU with "-bfd=download" for best results

Use Compiler-Based Instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_OPTIONS='-optCompInst -optQuiet'

% make CC=tau_cc.sh CXX=tau_cxx.sh CC=tau_cc.sh

NOTE: You may also use the short-hand scripts taucc, tauf90,
taucxx instead of specifying TAU_OPTIONS and using the
traditional tau_<cc,cxx,f90>.sh scripts. These scripts use
compiler-based instrumentation by default.

% make CC=taucc CXX=taucxx F90=tauf90

% srun -n 8 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
```

Compiler-based Instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_OPTIONS='-optCompInst -optQuiet'

% make CC=tau_cc.sh CXX=tau_cxx.sh CC=tau_cc.sh

NOTE: You may also use the short-hand scripts taucc, tauf90,
taucxx instead of specifying TAU_OPTIONS and using the
traditional tau_<cc,cxx,f90>.sh scripts. These scripts use
compiler-based instrumentation by default.

% make CC=taucc CXX=taucxx F90=tauf90

% srun -n 8 ./a.out
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk
```

Multi-language Application Debugging

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_OPTIONS='-optMemDbg -optVerbose'
% make CC=tau_cc.sh  CC=tau_cc.sh CXX=tau_cxx.sh

% export TAU_MEMDBG_PROTECT_ABOVE=1
% export TAU_MEMDBG_PROTECT_BELOW=1
% export TAU_MEMDBG_PROTECT_FREE=1
% srun -n 8  ./matmult
% paraprof
```

Memory Leak Detection

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% export TAU_OPTIONS='-optMemDbg -optVerbose'
% make CC=tau_cc.sh  CC=tau_cc.sh CXX=tau_cxx.sh


% export TAU_TRACK_MEMORY_LEAKS=1
% srun -n 8  ./matmult
% paraprof
```

Multi-language Memory Leak Detection

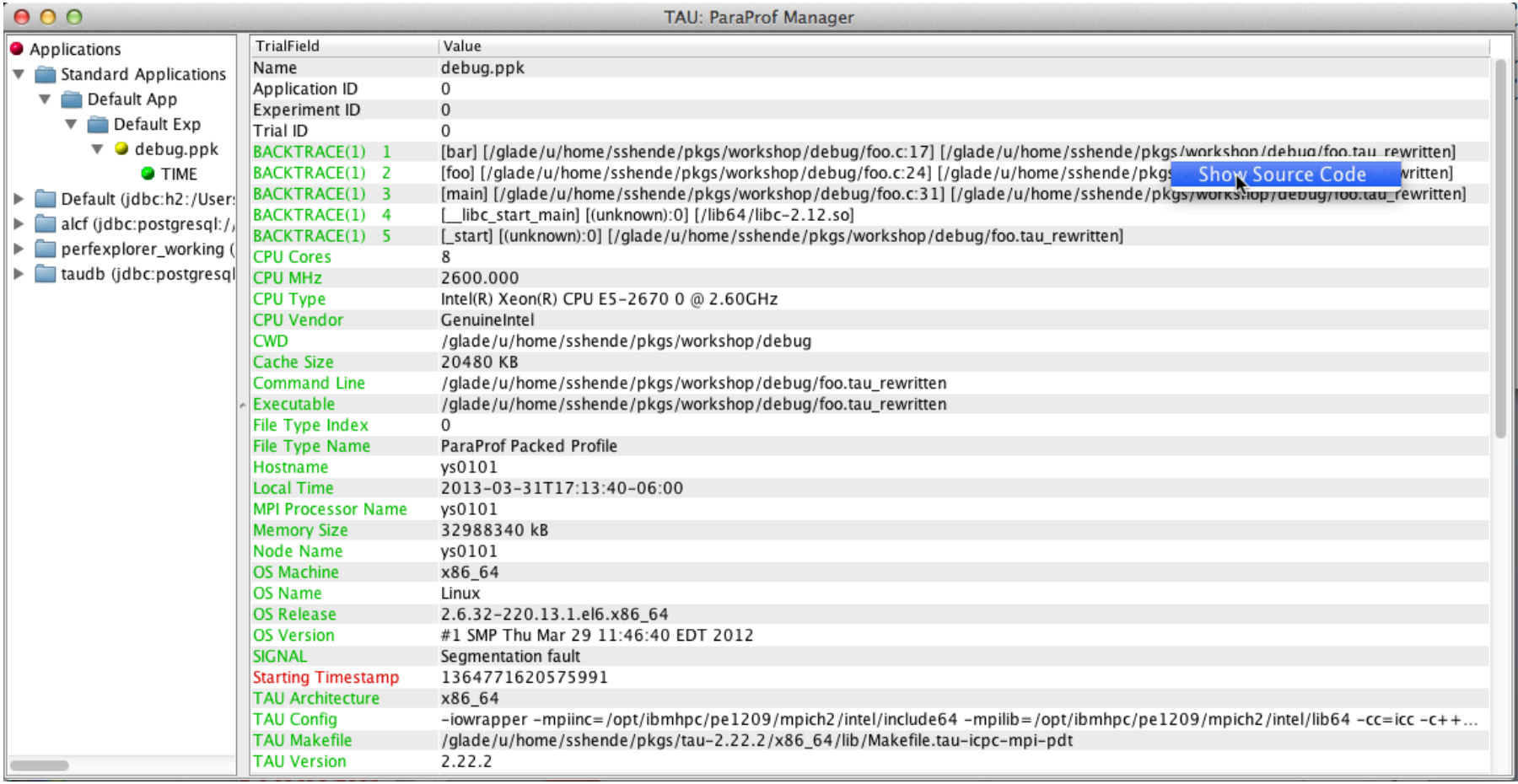
TAU: ParaProf: Context Events for: node 0 – memleak.ppk

Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
Heap Allocate	5,000,033	2	5,000,001	32	2,500,016.5	2,499,984.5
Heap Allocate <file=simple.c, line=15>	180	3	80	48	60	14.236
Heap Allocate <file=simple.c, line=23>	180	1	180	180	180	0
Heap Free <file=simple.c, line=18>	80	1	80	80	80	0
Heap Free <file=simple.c, line=25>	180	1	180	180	180	0
Heap Memory Used (KB)	4,884.829	8	4,883.196	0.047	610.604	1,614.888
▼ int foo(int) C [{simple.c} {36,1}-{44,1}]						
▼ int bar(int) C [{simple.c} {7,1}-{28,1}]						
Heap Allocate <file=simple.c, line=23>	180	1	180	180	180	0
Heap Free <file=simple.c, line=25>	180	1	180	180	180	0
▼ int g(int) C [{simple.c} {30,1}-{34,1}]						
▼ int bar(int) C [{simple.c} {7,1}-{28,1}]						
Heap Allocate <file=simple.c, line=15>	180	3	80	48	60	14.236
Heap Free <file=simple.c, line=18>	80	1	80	80	80	0
MEMORY LEAK! Heap Allocate <file=simple.c, line=15>	100	2	52	48	50	2
▼ int main(int, char **) C [{simple.c} {45,1}-{55,1}]						
▼ MPI_Finalize()						
Heap Allocate	5,000,033	2	5,000,001	32	2,500,016.5	2,499,984.5
MEMORY LEAK! Heap Allocate	5,000,033	2	5,000,001	32	2,500,016.5	2,499,984.5

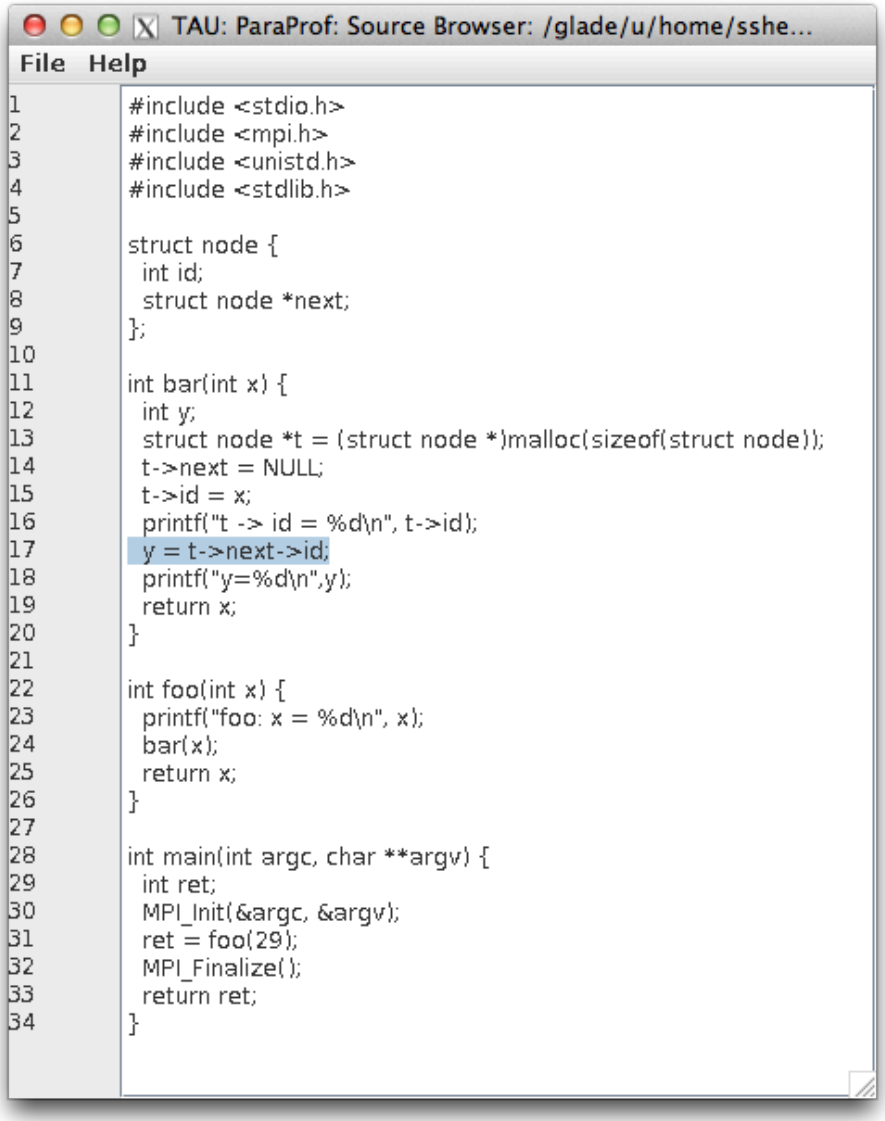
Tracking segfaults with TAU

```
% cd workshop/debug
% make
% srun -n 2 ./foo
% export TAU_TRACK_SIGNALS
% srun -n 2 tau_exec ./foo
% paraprof

For text output of backtrace:
% paraprof -d | grep BACK
```

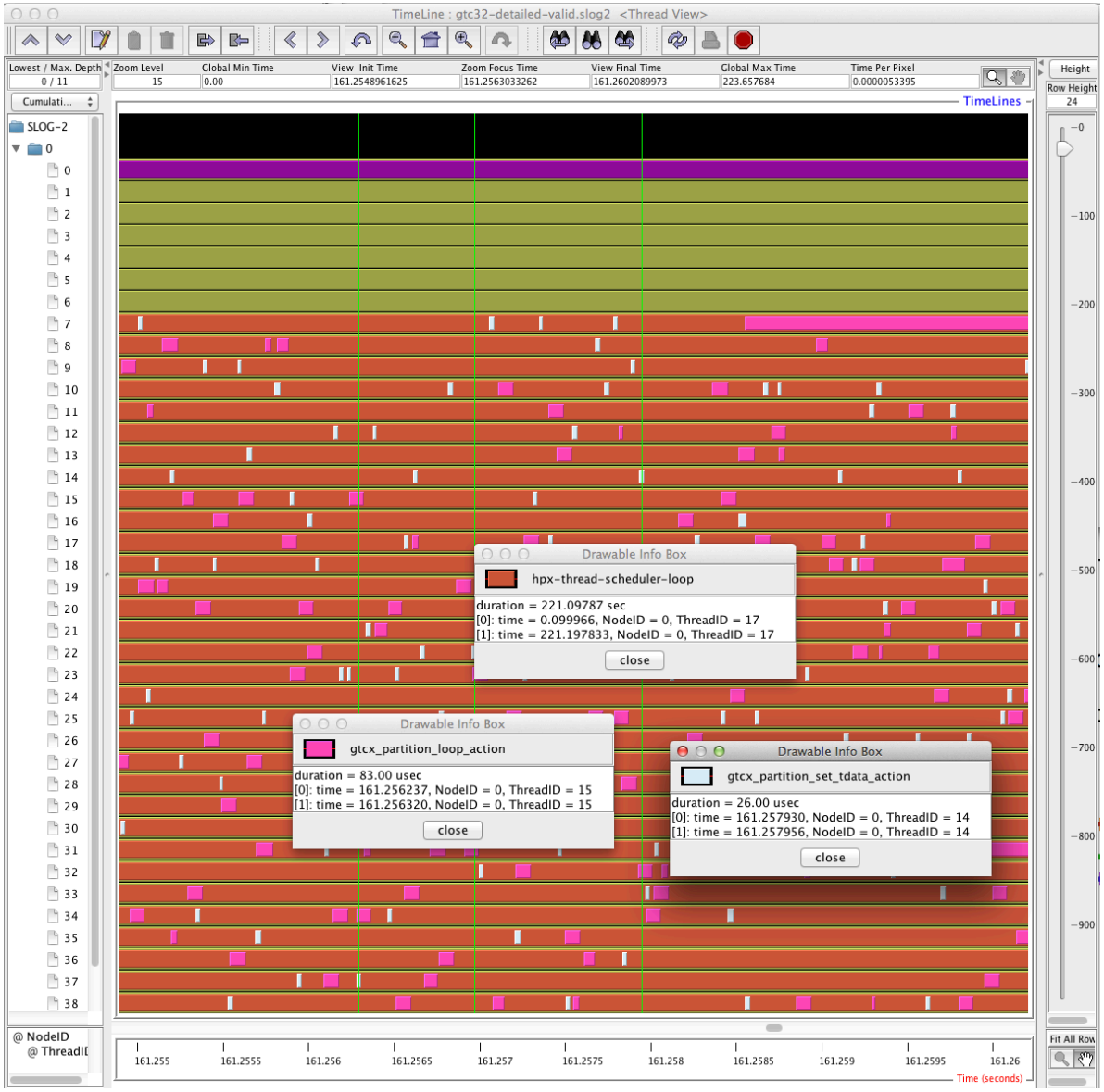
Multi-language Application Debugging



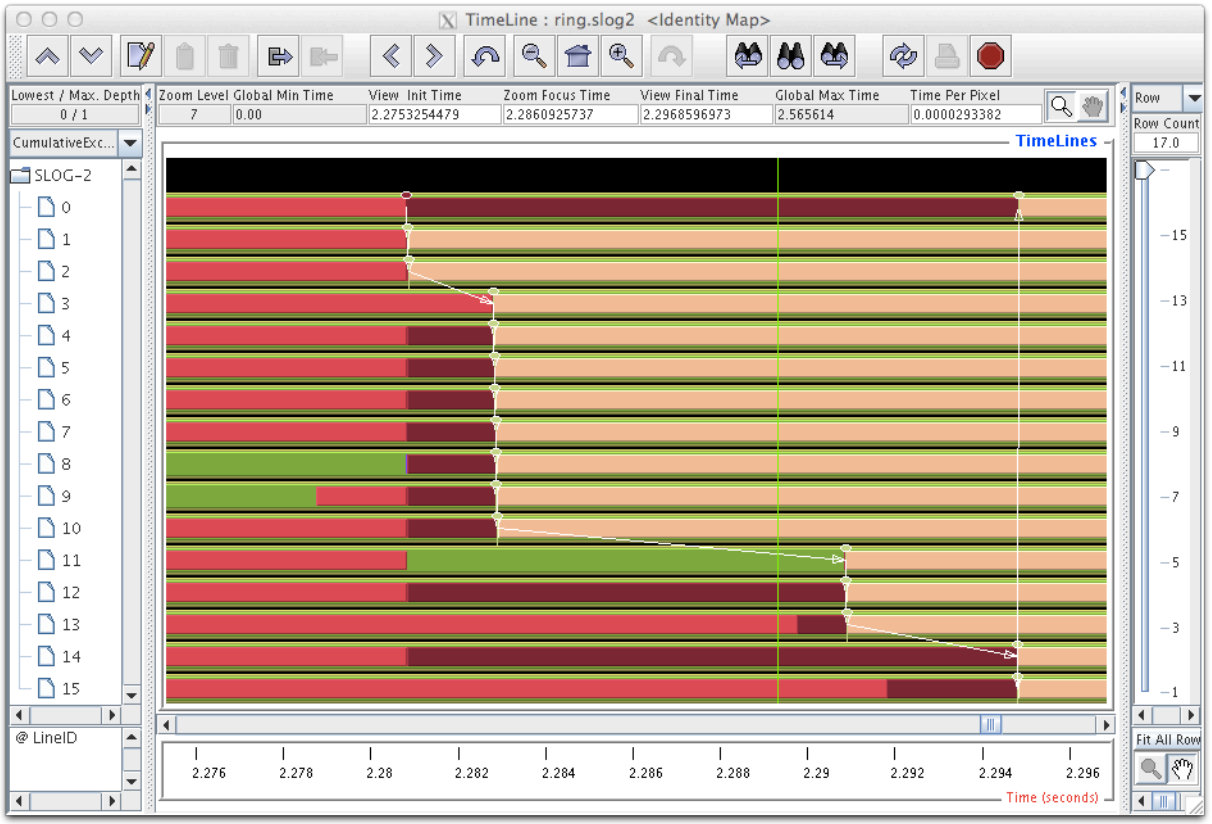
Location of segmentation violation



Jumpshot Trace Visualizer in TAU

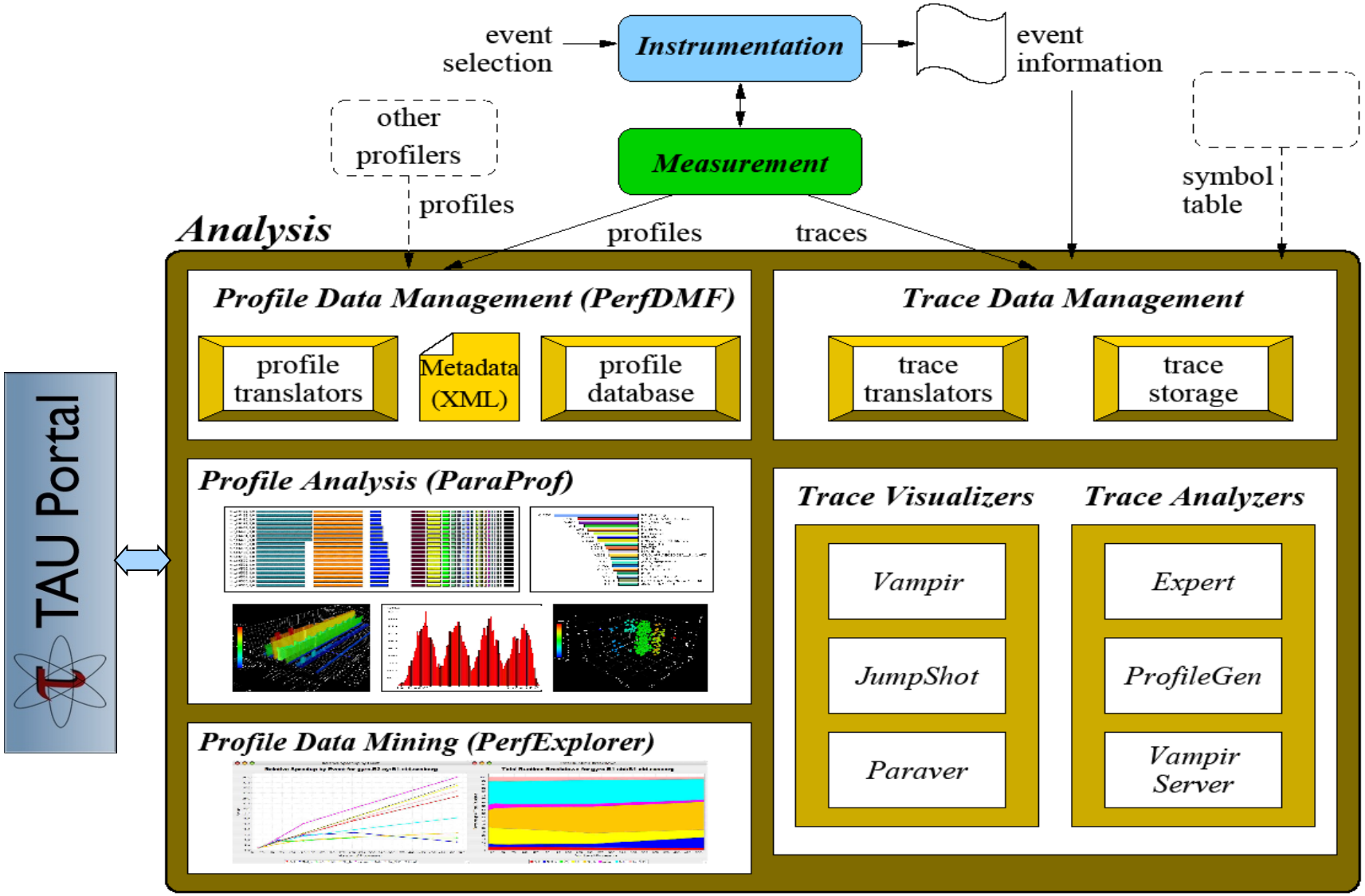


Tracing Communication in Jumpshot




```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% cmake –DCMAKE_CXX_COMPILER=tau_cxx.sh; make –j 8
% export TAU_TRACE=1
% mpirun –np 16 ./a.out ; tau_treemerge.pl; tau2slog2 tau.trc tau.edf –o a.slog2
% jumpshot a.slog2 &
```

Performance Analysis



Tools: Vampir

Vampir

Alternative and supplement to automatic analysis

Show dynamic run-time behavior graphically at any level of detail

Provide statistics and performance metrics

Timeline charts

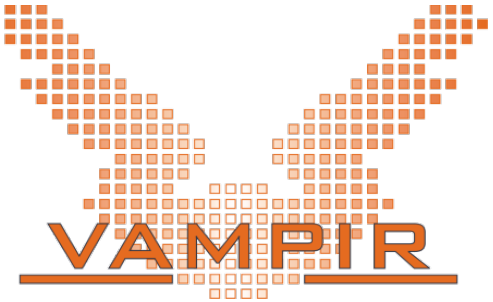
- **Show application activities and communication along a time axis**

Summary charts

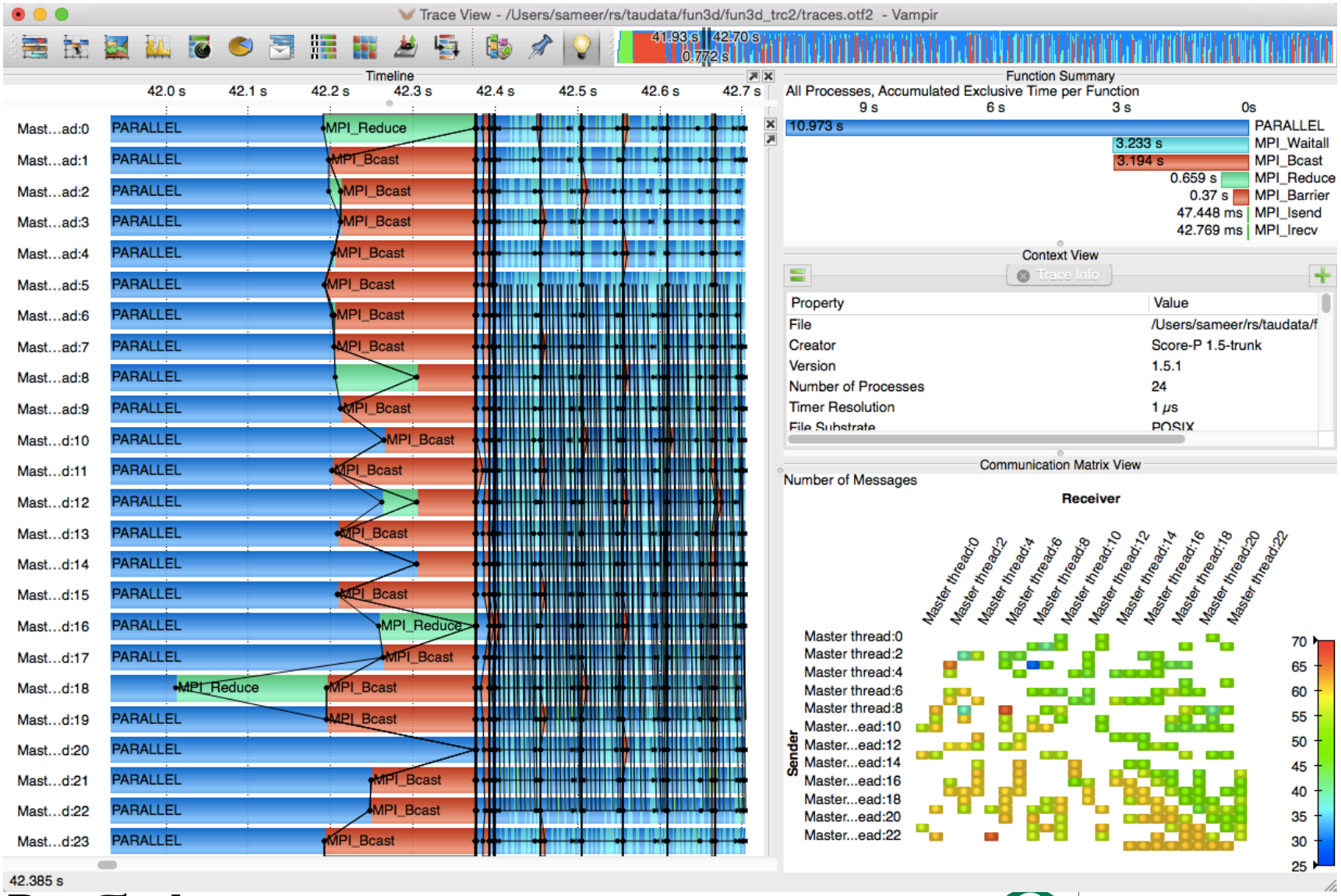
- **Provide quantitative results for the currently selected time interval**
- **Commercial trace visualization tool**

From TU Dresden, Germany

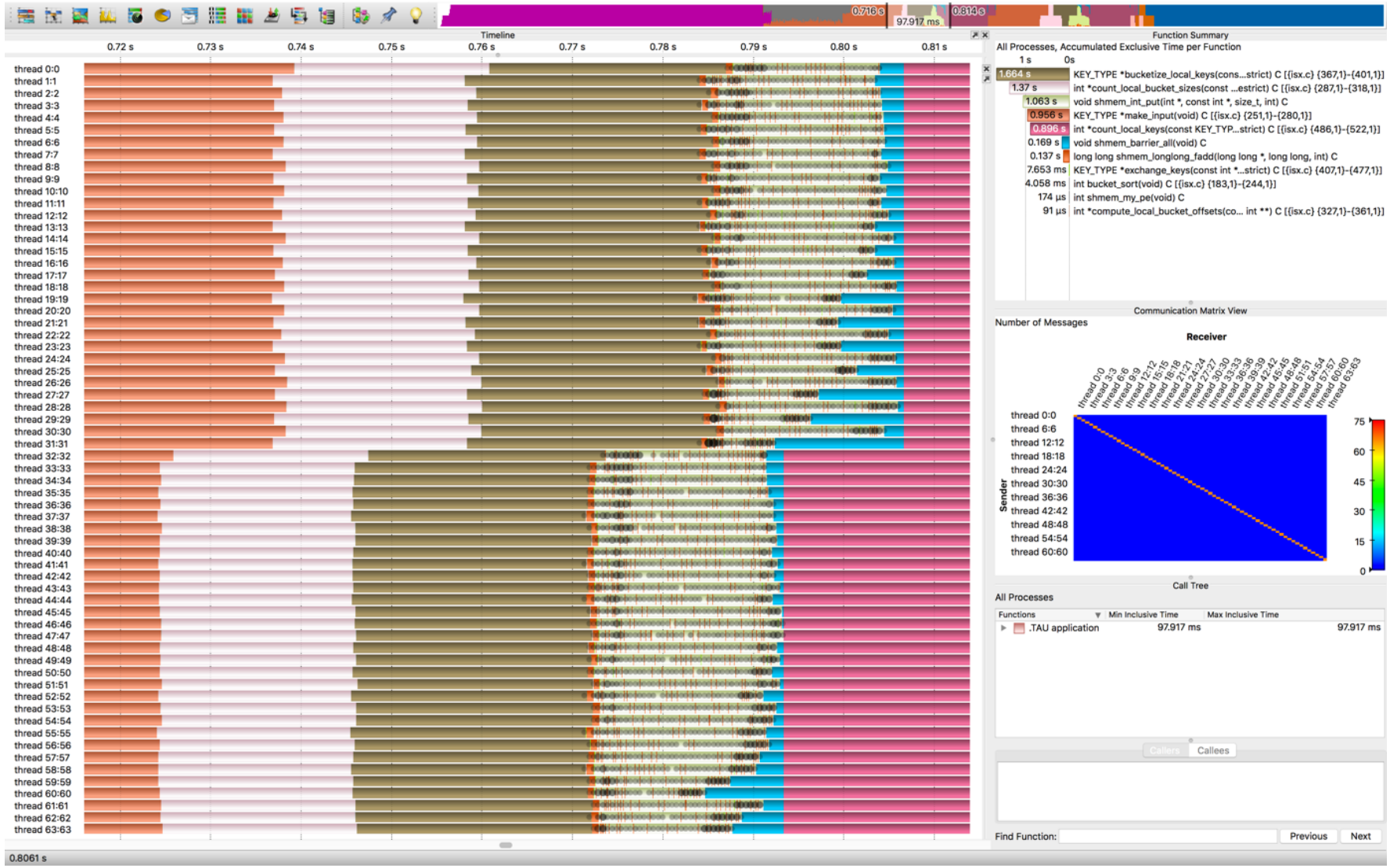
http://www.vampir.eu



Vampir – Trace Visualization

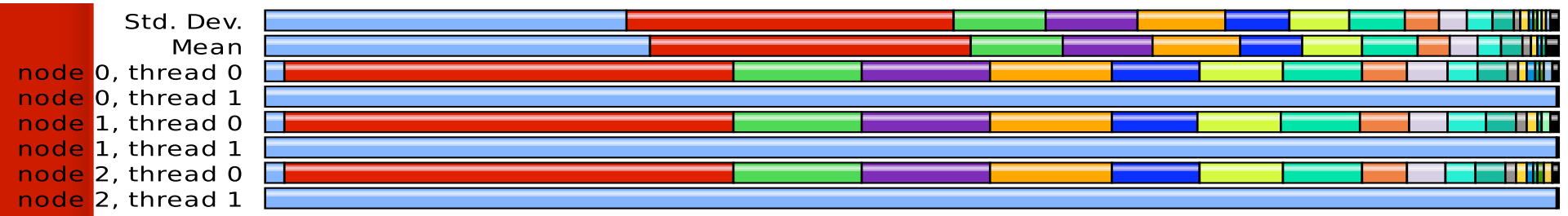


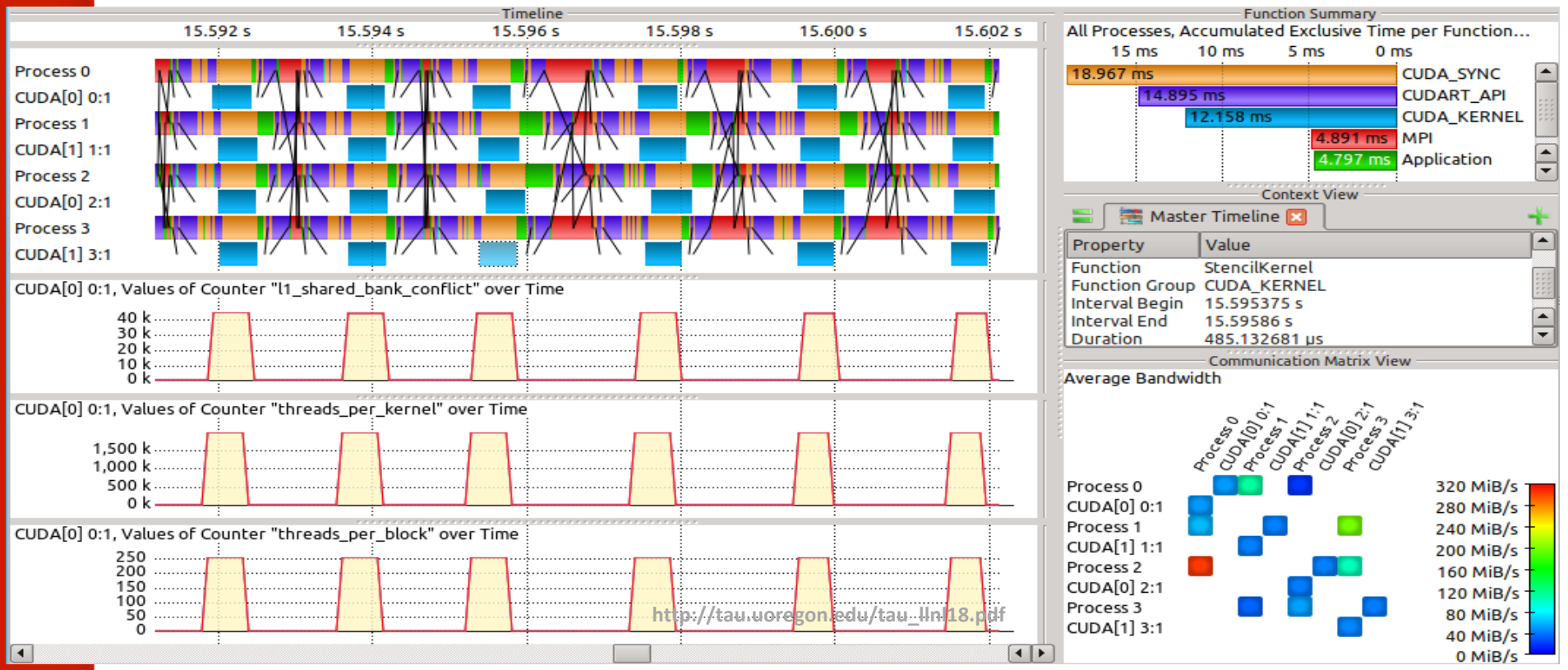
Vampir – Trace Visualization



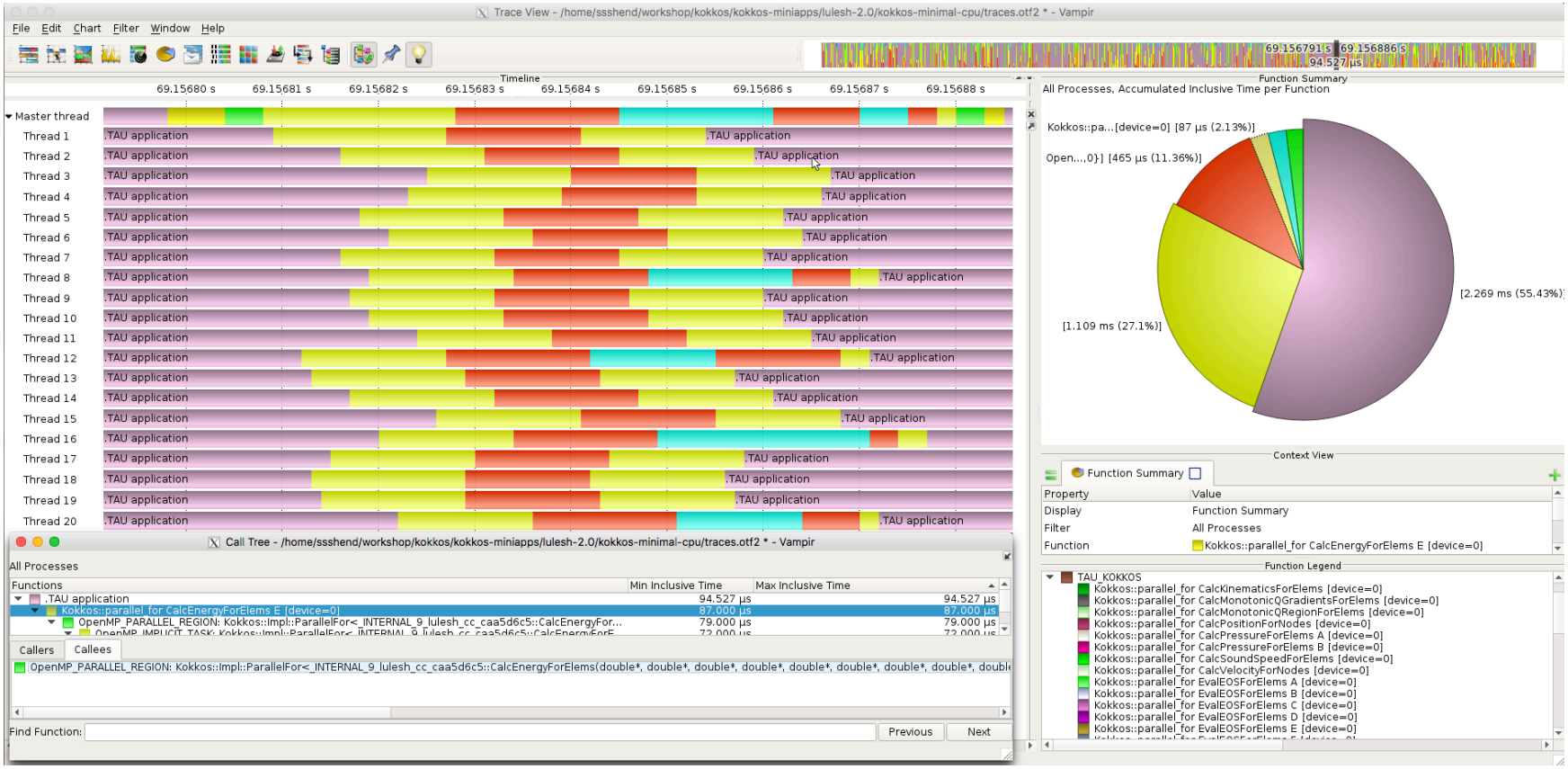
Stencil2D Parallel Profile / Trace in Vampir

Metric: TAUGPU_TIME
Value: Exclusive





Vampir – TAU's Kokkos Profiling Interface



Generating Event Traces

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh
(Or edit Makefile and change CC=tau_cc.sh )


% export TAU_TRACE=1
% export TAU_TRACE_FORMAT=otf2
# TAU's native OTF2 trace generation capability!
% mpirun -np 64   ./a.out
% vampir traces.otf2 &

For Jumpshot:
% export TAU_TRACE_FORMAT=default
% mpirun -np 64   ./a.out

% tau_treemerge.pl
% tau_treemerge; tau2slog2 tau.trc tau.edf -o app.slog2;
% jumpshot app.slog2

For ParaVer:
% tau_convert -paraver tau.trc tau.edf app.prv; paraver app.prv
```

Tools: Jumpshot

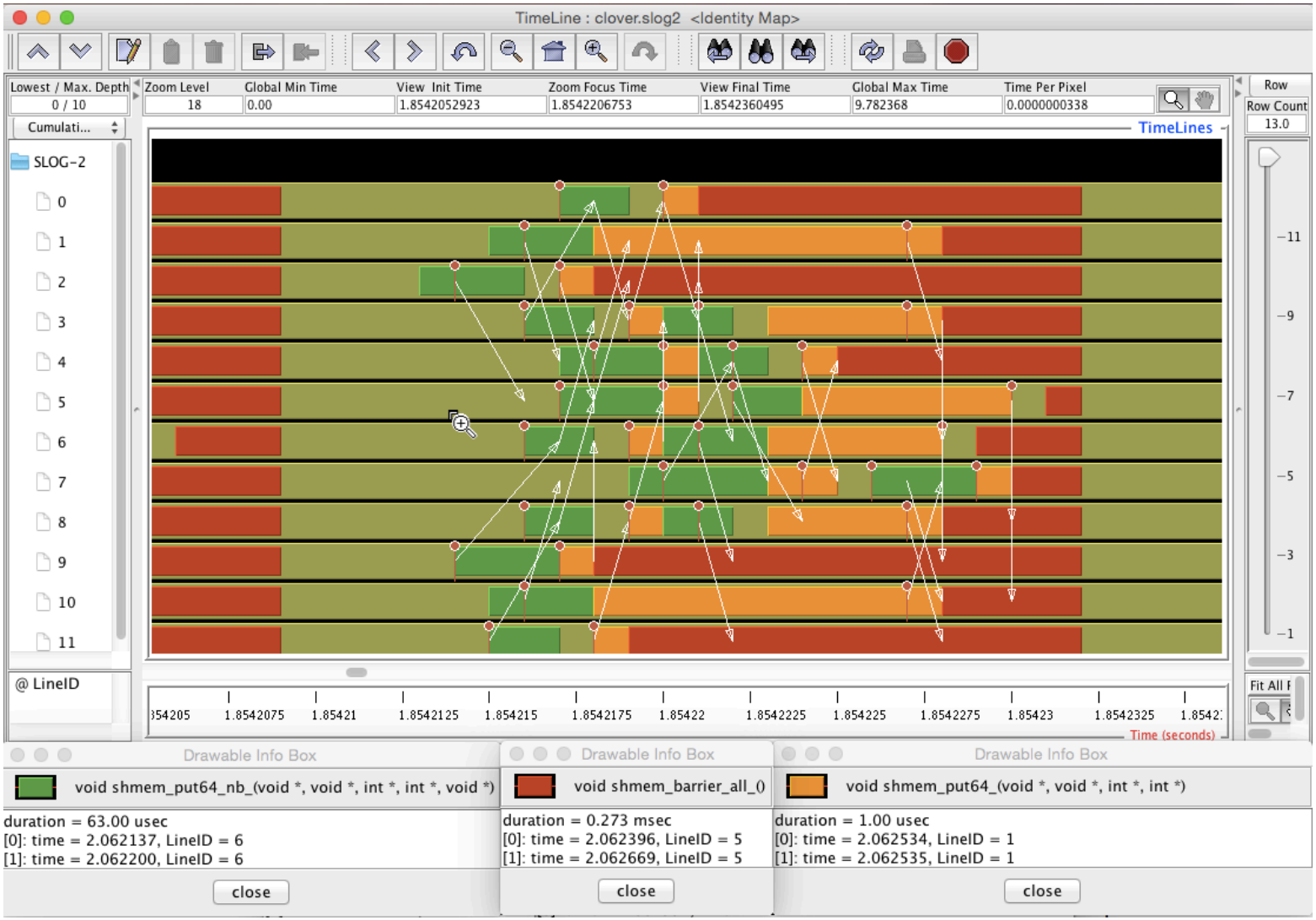
Jumpshot

- Open source alternative to Vampir
- Developed by Argonne National Laboratory
- Packaged with TAU

Timeline charts

- Show application activities and communication along a time axis
- Shows boxes within boxes to show nesting of events

Jumpshot


TimeLine : clover.slog2 <Identity Map>
Lowest / Max. Depth
0 / 10
Zoom Level
18
Global Min Time
0.00
View Init Time
1.8542052923
Zoom Focus Time
1.8542206753
View Final Time
1.8542360495
Global Max Time
9.782368
Time Per Pixel
0.0000000338
Row
Row Count
13.0
TimeLines
SLOG-2
0
1
2
3
4
5
6
7
8
9
10
11
@ LineID
Time (seconds)
Drawable Info Box
void shmem_put64_nb_(void *, void *, int *, int *, void *)
duration = 63.00 usec
[0]: time = 2.062137, LineID = 6
[1]: time = 2.062200, LineID = 6
close
Drawable Info Box
void shmem_barrier_all_()
duration = 0.273 msec
[0]: time = 2.062396, LineID = 5
[1]: time = 2.062669, LineID = 5
close
Drawable Info Box
void shmem_put64_(void *, void *, int *, int *)
duration = 1.00 usec
[0]: time = 2.062534, LineID = 1
[1]: time = 2.062535, LineID = 1
close

Tools:
TAU Commander

TAU Commander

Universal tool or integrated toolkit

Unbiased, accurate measurements

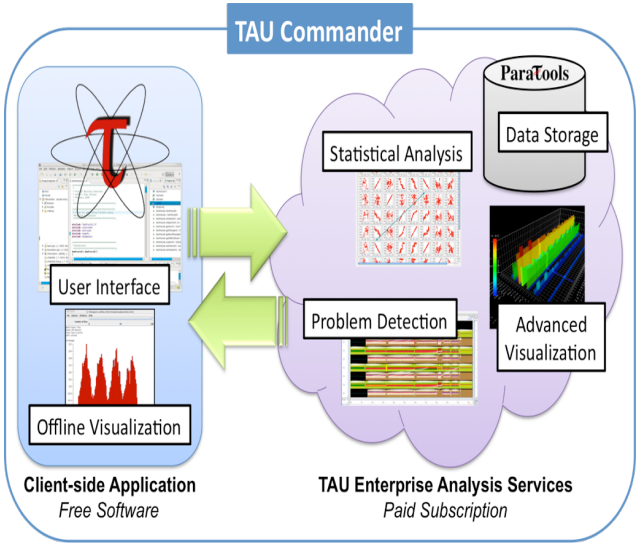
- File I/O: serial and parallel
- Communication: inter- and intra-node
- Memory: allocation and access
- CPU: vectorization, cache utilization, etc.

Minimal overhead

- Provide multiple measurement methods
- Focus on one performance aspect at a time

Easy to use

- Intuitive, systematic, and well documented
- Easy to understand and configure
- BSD style license, Github open source
- ***http://www.taucommander.com***

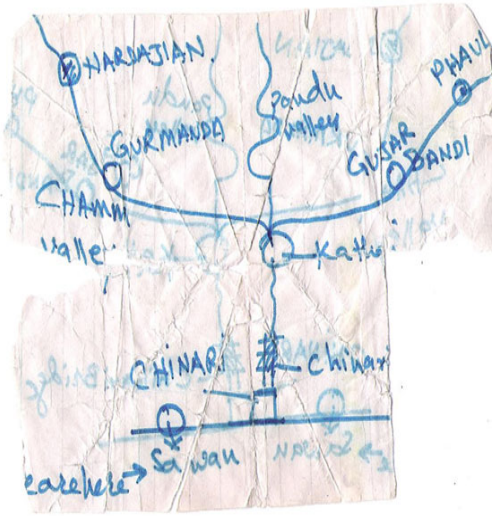


TAU Commander's Approach

Say where you're going, not how to get there

Experiments give context to the user's actions

- Defines desired metrics and measurement approach
- Defines operating environment
- Establishes a baseline for error checking



VS.

T-A-M Model for Performance Engineering

Target

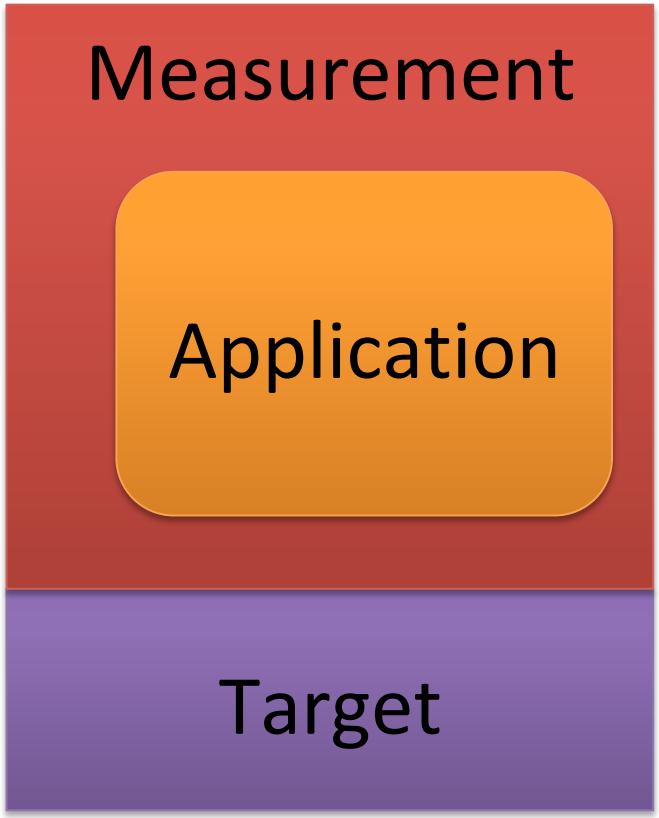
- Installed software
- Available compilers
- Host architecture/ OS

Application

- MPI, OpenMP, CUDA, OpenACC, etc.

Measurement

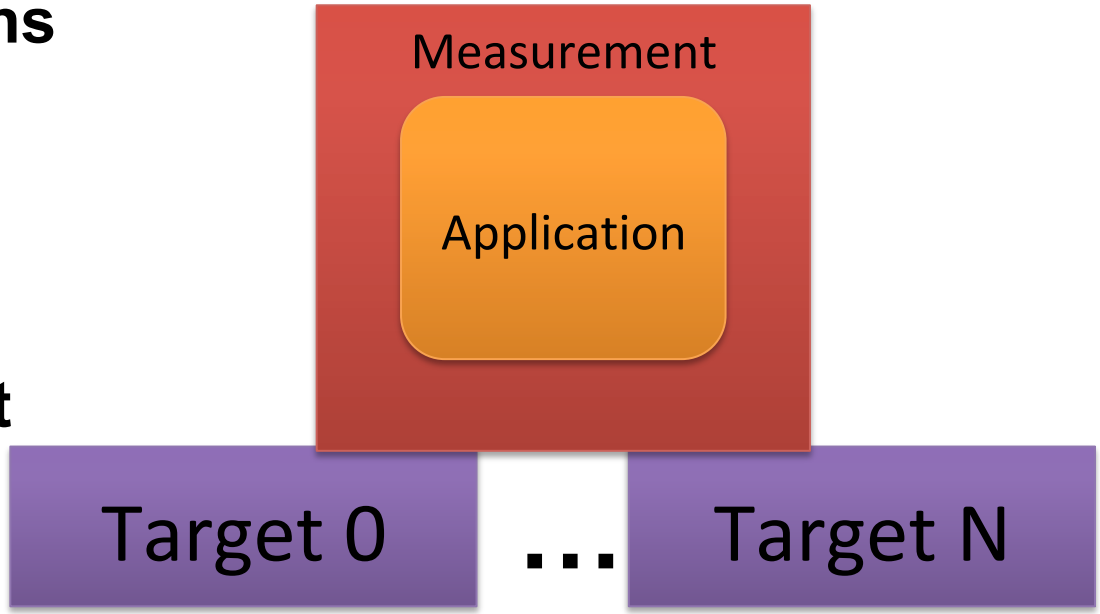
- Profile, trace, or both
- Sample, source inst…



Experiment = (Target, Application, Measurement)

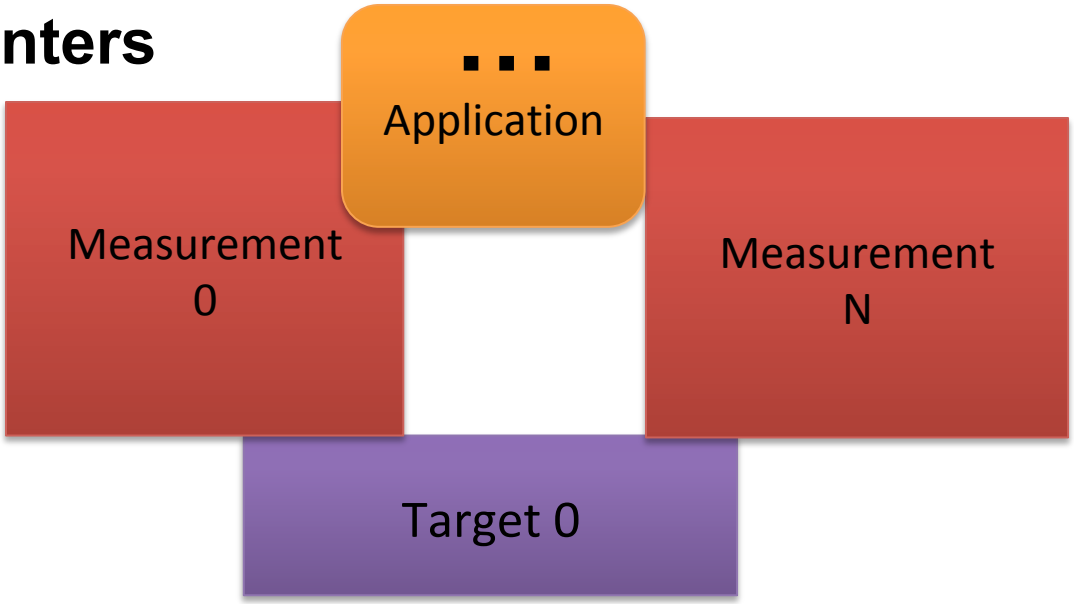
Which platform is best for my application?

- Many targets:
  - Different MPI implementations
  - Different CPU architectures
  - GPU vs MIC
  - Cray vs SGI
- One measurement
- One application



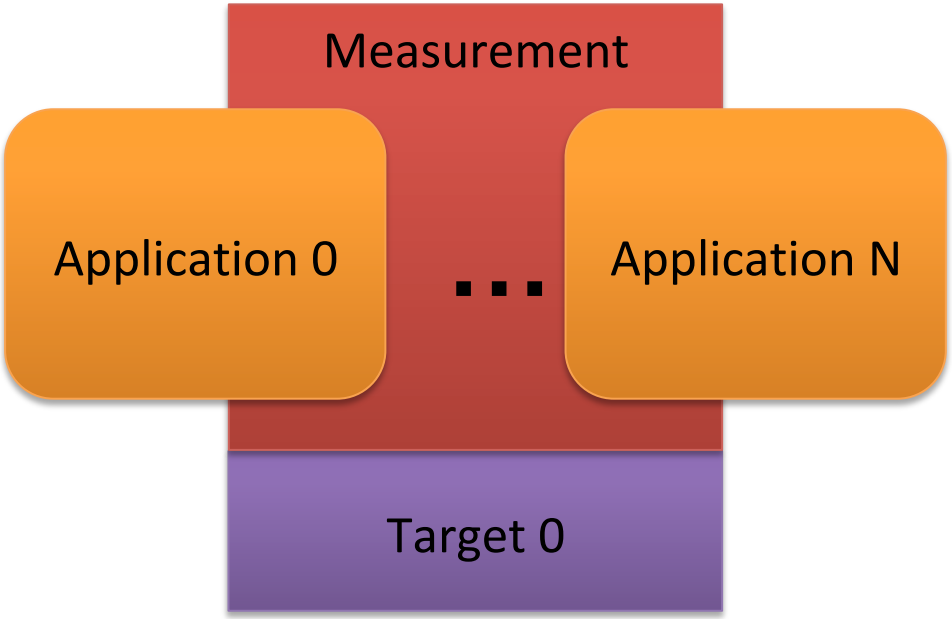
What are the performance characteristics of my application?

- One target
- Many measurements:
  - File I/O
  - Communication
  - Memory allocation
  - Performance counters
  - Vectorization
- One application



How well does my target perform various tasks?

- One target
- One measurement
- Many applications:
  - Compute bound
    - Dense LA
  - Memory bound
    - Sparse LA
    - Graph
  - Scaling
    - Thread-level
    - Process-level



Getting Started with TAU Commander

1. tau initialize --mpi --compilers Intel
2. tau mpicc *.c -o foo
3. tau srun –n 8 ./foo
4. tau show
5. tau dash

Just put `tau` in front of everything and see what happens.

This works on any supported system, even if TAU is not installed or has not been configured appropriately.

TAU and all its dependencies will be downloaded and installed if required.

TAU Commander Online Help

jlinford — ssh cori.nersc.gov — 80×47

```
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $ tau --help
usage: tau [arguments] <subcommand> [options]

TAU Commander 1.0a [ www.taucommander.com ]

Positional Arguments:
  <subcommand>    See subcommand descriptions below.
  [options]       Options to be passed to <subcommand>.

Optional Arguments:
  -V, --version   Show program's version number and exit.
  -h, --help      Show this help message and exit.
  -q, --quiet     Suppress all output except error messages.
  -v, --verbose   Show debugging messages.

Configuration Subcommands:
  application     Create and manage application configurations.
  experiment      Create and manage experiments.
  measurement     Create and manage measurement configurations.
  project         Create and manage project configurations.
  target          Create and manage target configurations.
  trial           Create and manage experiment trials.

Subcommands:
  build           Instrument programs during compilation and/or linking.
  configure       Configure TAU Commander.
  dashboard       Show all project components.
  help            Show help for a command or suggest actions for a file.
  initialize      Initialize TAU Commander.
  select          Create a new experiment or select an existing experiment.

Shortcuts:
  tau <compiler>  Execute a compiler command
                  - Example: tau gcc *.c -o a.out
                  - Alias for 'tau build <compiler>'
  tau <program>   Gather data from a program
                  - Example: tau ./a.out
                  - Alias for 'tau trial create <program>'
  tau metrics     Show metrics available in the current experiment
                  - Alias for 'tau target metrics'
  tau select      Select configuration objects to create a new experiment
                  - Alias for 'tau experiment create'
  tau show        Show data from the most recent trial
                  - Alias for 'tau trial show'

See 'tau help <subcommand>' for more information on <subcommand>.
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $
```

jlinford — ssh cori.nersc.gov — 80×35

```
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $ tau app cre --help
usage: tau application create <application_name> [arguments]

Create application configurations.

Optional Arguments:
  -@ <level>              Create the application at the specified storage
                          level.
                            - <level>: project, user, system
                            - default: project
  -h, --help              Show this help message and exit.

Application Arguments:
  <application_name>      Application configuration name.
  --cuda [T/F]            Application uses NVIDIA CUDA.
                            - default: False
  --linkage <linkage>     Application linkage.
                            - <linkage>: static, dynamic
                            - default: static
  --mpc [T/F]             Application uses MPC.
                            - default: False
  --mpi [T/F]             Application uses MPI.
                            - default: False
  --opencl [T/F]          Application uses OpenCL.
                            - default: False
  --openmp [T/F]          Application uses OpenMP.
                            - default: False
  --pthreads [T/F]        Application uses pthreads.
                            - default: False
  --select-file path      Specify selective instrumentation file.
  --shmem [T/F]           Application uses SHMEM.
                            - default: False
  --tbb [T/F]             Application uses Thread Building Blocks (TBB).
                            - default: False
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $
```

Step 1: Initialize TAU Project

```
$ tau initialize --mpi --compilers Intel
```

- Creates a new project configuration using defaults
- Project files exist in a directory named ".tau"
- Like git, all directories below the directory containing the ".tau" directory can access the project
  - E.g. `tau dashboard` works in miniapp1/baseline

Project Initialization

```
ParaTools — ssh cori.nersc.gov — 120×50
jlinford@cori09 ~/workspace/openshmem17/applications/ISx $ tau initialize --shmem
[TAU] Cray C++ compiler '/opt/cray/pe/craype/2.5.7/bin/CC' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/icpc'
[TAU] Cray Fortran compiler '/opt/cray/pe/craype/2.5.7/bin/ftn' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/ifort'
[TAU] Cray C compiler '/opt/cray/pe/craype/2.5.7/bin/cc' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/icc'
[TAU] Cray MPI C compiler '/opt/cray/pe/craype/2.5.7/bin/cc' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/icc'
[TAU] Cray MPI C++ compiler '/opt/cray/pe/craype/2.5.7/bin/CC' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/icpc'
[TAU] Cray MPI Fortran compiler '/opt/cray/pe/craype/2.5.7/bin/ftn' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/ifort'
[TAU] Cray SHMEM C compiler '/opt/cray/pe/craype/2.5.7/bin/cc' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/icc'
[TAU] Cray SHMEM C++ compiler '/opt/cray/pe/craype/2.5.7/bin/CC' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/icpc'
[TAU] Cray SHMEM Fortran compiler '/opt/cray/pe/craype/2.5.7/bin/ftn' wraps
[TAU]     '/opt/intel/compilers_and_libraries_2017.2.174/linux/bin/intel64/ifort'
[TAU] Created a new project named 'ISx'.
[TAU] Added application 'ISx' to project configuration 'ISx'.
[TAU] Added target 'cori09' to project configuration 'ISx'.
[TAU] Added measurement 'sample' to project configuration 'ISx'.
[TAU] Added measurement 'profile' to project configuration 'ISx'.
[TAU] Added measurement 'trace' to project configuration 'ISx'.
[TAU] Created a new experiment 'cori09-ISx-sample'
[TAU] Installing PDT to '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/pdt/77f947dd'
[TAU] Using PDT source archive '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/src/pdt.tgz'
[TAU] Checking contents of '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/src/pdt.tgz'
[TAU] Completed in 8.276 seconds
[TAU] Extracting '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/src/pdt.tgz' to create
[TAU]     '/dev/shm/tmpQl6qTD/./pdtoolkit-3.24'
[TAU] Completed in 5.216 seconds
[TAU] Configuring PDT...
[TAU] Completed in 17.439 seconds
[TAU] Compiling PDT...
[TAU] Completed in 6.394 seconds
[TAU] Installing PDT...
[TAU] Completed in 0.115 seconds
[TAU] Checking installed files...
[TAU] Completed in 0.115 seconds
[TAU] Setting file permissions...
[TAU] Completed in 0.136 seconds
[TAU] Verifying PDT installation...
[TAU] Installing TAU Performance System at
[TAU]     '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/tau/./tau-2.26.2'
[TAU] Configuring TAU...
[TAU] Completed in 26.358 seconds
[TAU] Compiling and installing TAU...
[TAU] 5.0 seconds [CPU: 32.4
```

Compiler detection

Project initialization

Download and install PDT

TAU installation progress

Project Dashboard (`tau dashboard`)

Terminal — ssh quartz — 186×56

```
quartz2306{shende1}55: tau dash
```

== Project Configuration (/g/g24/shende1/workshop/mm/.tau/project.json) ==

Name	Targets	Applications	Measurements	# Experiments
mm	quartz7	mm	baseline, sample, profile, source-inst, compiler-inst, trace	2

== Targets in project 'mm' ==

Name	Host OS	Host Arch	Host Compilers	MPI Compilers	SHMEM Compilers
quartz7	Linux	x86_64	Intel	System	Cray

== Applications in project 'mm' ==

Name	Linkage	OpenMP	Pthreads	TBB	MPI	CUDA	OpenCL	SHMEM	MPC
mm	dynamic	No	No	No	Yes	No	No	No	No

== Measurements in project 'mm' ==

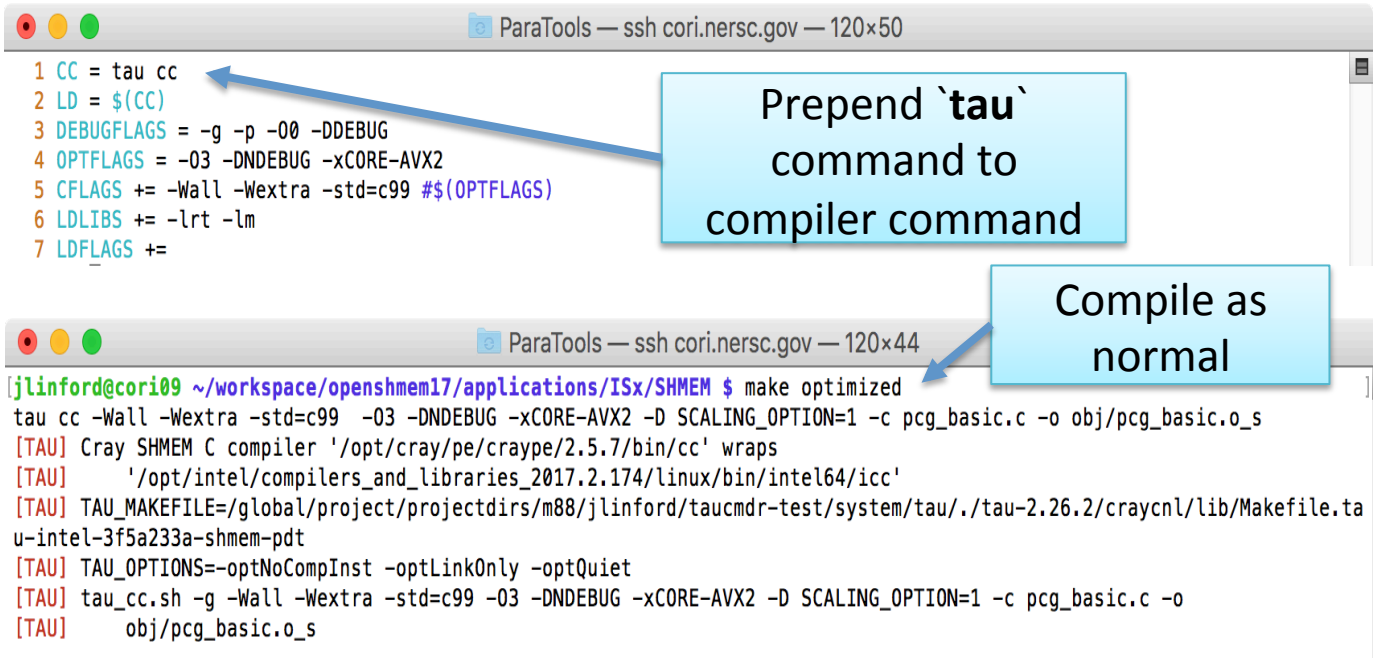
Name	Profile	Trace	Sample	Source Inst.	Compiler Inst.	OpenMP	CUDA	I/O	MPI	SHMEM
baseline	tau	none	No	never	never	ignore	No	No	No	No
sample	tau	none	Yes	never	never	ignore	No	No	Yes	No
profile	tau	none	No	automatic	fallback	ignore	No	No	Yes	No
source-inst	tau	none	No	automatic	never	ignore	No	No	Yes	No
compiler-inst	tau	none	No	never	always	ignore	No	No	Yes	No
trace	none	otf2	No	automatic	fallback	ignore	No	No	Yes	No

== Experiments in project 'mm' ==

Name	Trials	Data Size	Target	Application	Measurement	TAU Makefile
quartz7-mm-sample	1	15.5KiB	quartz7	mm	sample	Makefile.tau-icpc-papi-mpi-pdt
quartz7-mm-trace	1	3.0KiB	quartz7	mm	trace	Makefile.tau-icpc-papi-mpi-pdt

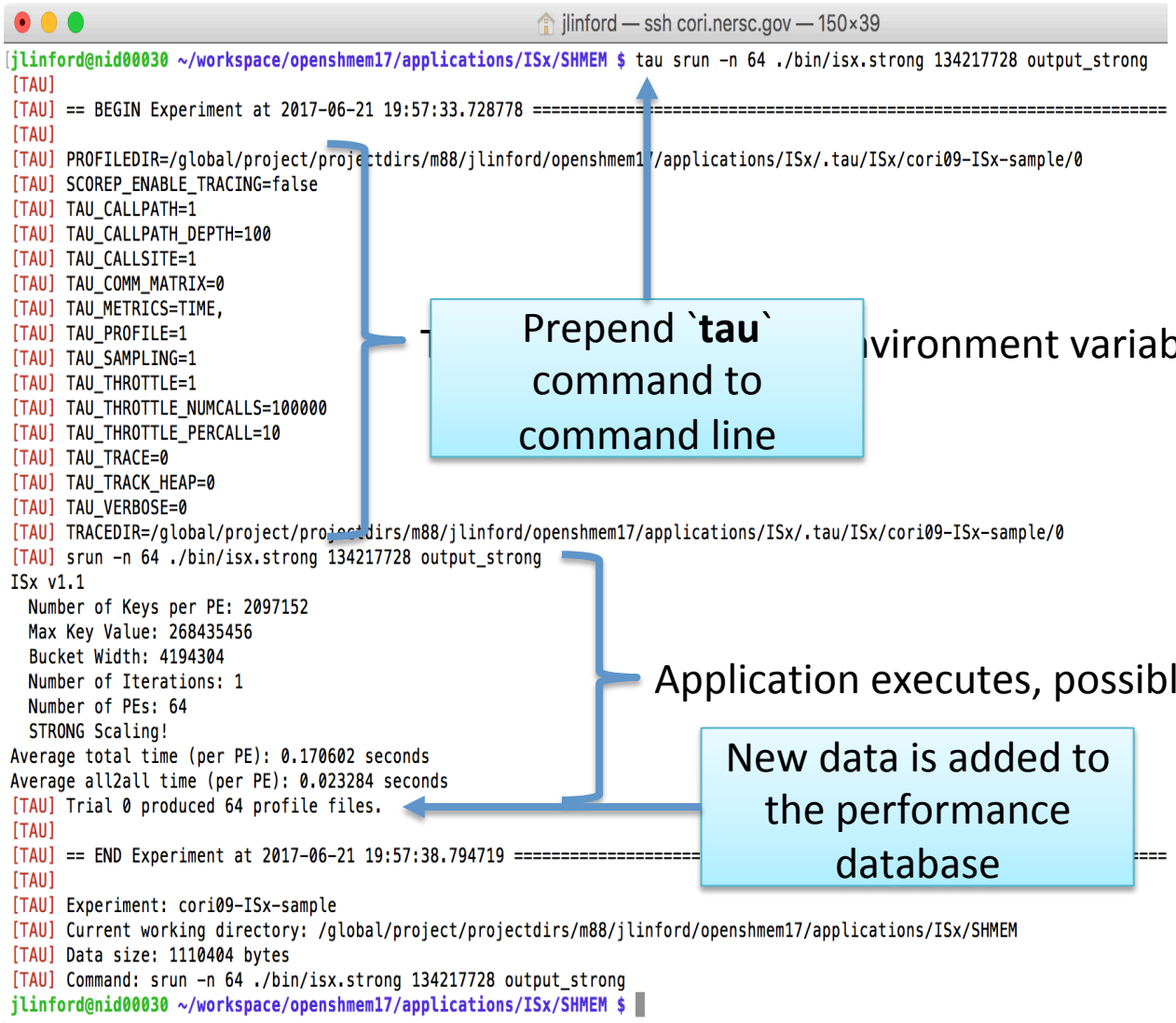
Selected Experiment: quartz7-mm-sample

Step 2: Use `tau` to compile



- TAU Commander constructs a new compilation command line.
  - May replace compiler commands with TAU's compiler wrapper scripts.
  - May set environment variables, parse configuration files, etc.
  - If no changes are required then nothing is changed.

Step 3: Use `tau` to run



Step 4: Use `tau` to view data (`tau show`)

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

Create a New Experiment

Select a new measurement to create a new experiment

ParaTools — ssh cori.nersc.gov — 120×17

```
jlinford@nid00073 ~/workspace/openshmem17/applications/ISx/SHMEM $ tau select profile
[TAU] Created a new experiment 'cori09-ISx-profile'
[TAU] Installing TAU Performance System at
[TAU]     '/global/project/projectdirs/m88/jlinford/taucmdr-test/system/tau/./tau-2.26.2'
[TAU] Configuring TAU...
[TAU] Completed in 155.459 seconds
[TAU] Compiling and installing TAU...
[TAU] Completed in 48.596 seconds
[TAU] Checking installed files...
[TAU] Completed in 10.551 seconds
[TAU] Setting file permissions...
[TAU] Completed in 2.556 seconds
[TAU] Verifying TAU Performance System installation...
[TAU] Selected experiment 'cori09-ISx-profile'.
[TAU] Application rebuild required:
[TAU]   - source_inst changed from 'never' to 'automatic'
jlinford@nid00073 ~/workspace/openshmem17/applications/ISx/SHMEM $
```

TAU Performance System® automatically reconfigured and recompiled.

User advised that an application rebuild is required to use source-based instrumentation.

TAU Commander at LLNL

```
On Quartz
% source /usr/global/tools/tau/training/tau.bashrc (or .cshrc)
% which tau
% cd workshop/matmult
% tau init --mpi  --compilers Intel
% make clean;
% make CC=‘tau mpicc’
% tau srun –n 8 ./matmult
% tau show
And try the examples. Try:
% tau --help
% tau meas edit --help
```

TAU and PDT for Source Instrumentation

```
On Quartz
% source /usr/global/tools/tau/training/tau.bashrc (or .cshrc)
% which tau
% cd workshop/matmult
% tau init --mpi --compilers Intel
% tau dash
% tau select profile
% make clean
% make F90='tau mpif90'
% tau srun -n 8 ./matmult
% tau show
```

Selective Instrumentation File

```
% tau dash
% tau application edit <app_name> --select-file select.tau
% cat select.tau
BEGIN_INCLUDE_LIST
int main#
int dgemm#
END_INCLUDE_LIST
BEGIN_FILE_INCLUDE_LIST
Main.c
Blas/*.f77
END_FILE_INCLUDE_LIST
# replace include with exclude list (BEGIN_EXCLUDE_LIST/END…)

BEGIN_INSTRUMENT_SECTION
loops routine="foo"
loops routine="int main#"
END_INSTRUMENT_SECTION
% export TAU_SELECT_FILE=select.tau    (to use at runtime)
```

Generating Event Traces

```
% cd workshop/matmult
% tau init –mpi --compilers Intel
% tau select trace
% make clean; make F90=‘tau mpif90’;
% tau srun –n 8 ./matmult;
% tau show
(Vampir OTF2)

To use Jumpshot:
% tau meas edit profile --trace slog2
    (if it is profiling is being used in another experiment, you may
    have to delete it:
    tau experiment delete <exp_name>
    and retry

% make F90=‘tau mpif90’
% tau srun –n 8 ./matmult
% tau show
```

Three Instrumentation Techniques for Wrapping External Libraries

Pre-processor based substitution by re-defining a call (e.g., read)

- Tool defined header file with same name <*unistd.h*> takes precedence
- Header redefines a routine as a different routine using macros
- Substitution: *read()* substituted by preprocessor as *tau_read()* at callsite

Preloading a library at runtime

- Library preloaded (*LD_PRELOAD* env var in Linux) in the address space of executing application intercepts calls from a given library
- Tool's wrapper library defines *read(),* gets address of global *read()* symbol (dlsym), internally calls timing calls around call to global read

Linker based substitution

- Wrapper library defines *__wrap_read* which calls *__real_read* and linker is passed *-Wl,-wrap,read* to substitute all references to read from application's object code with the *__wrap_read* defined by the tool

Preprocessor based substitution

Pre-processor based substitution by re-defining a call

- Compiler replaces read() with tau_read() in the body of the source code

Advantages:

- Simple to instrument
  - Preprocessor based replacement
  - A header file redefines the calls
  - No special linker or runtime flags required

Disadvantages

- Only works for C & C++ for replacing calls in the body of the code.
- Incomplete instrumentation: fails to capture calls in uninstrumented libraries (e.g., libhdf5.a)

Linker based substitution

Linker based substitution

- Wrapper library defines __wrap_read which calls __real_read and linker is passed -Wl,-wrap, read

Advantages

- Tool can intercept all references to a given call
- Works with static as well as dynamic executables
- No need to recompile the application source code, just re-link the application objects and libraries with the tool wrapper library

Disadvantages

- Wrapping an entire library can lengthen the linker command line with multiple –Wl,-wrap,<func> arguments. It is better to store these arguments in a file and pass the file to the linker
- Approach does not work with un-instrumented binaries

tau_gen_wrapper

Automates creation of wrapper libraries using TAU

Input:

- header file (foo.h)
- library to be wrapped (/path/to/libfoo.a)
- technique for wrapping
  - Preprocessor based redefinition (-d)
  - Runtime preloading (-r)
  - Linker based substitution (-w: default)
- Optional selective instrumentation file (-f select)
  - Exclude list of routines, or
  - Include list of routines

Output:

- wrapper library
- optional *link_options.tau* file (-w), pass –optTauWrapFile=<file> in TAU_OPTIONS environment variable

Design of wrapper generator (tau_gen_wrapper)

***tau_gen_wrapper* shell script:**

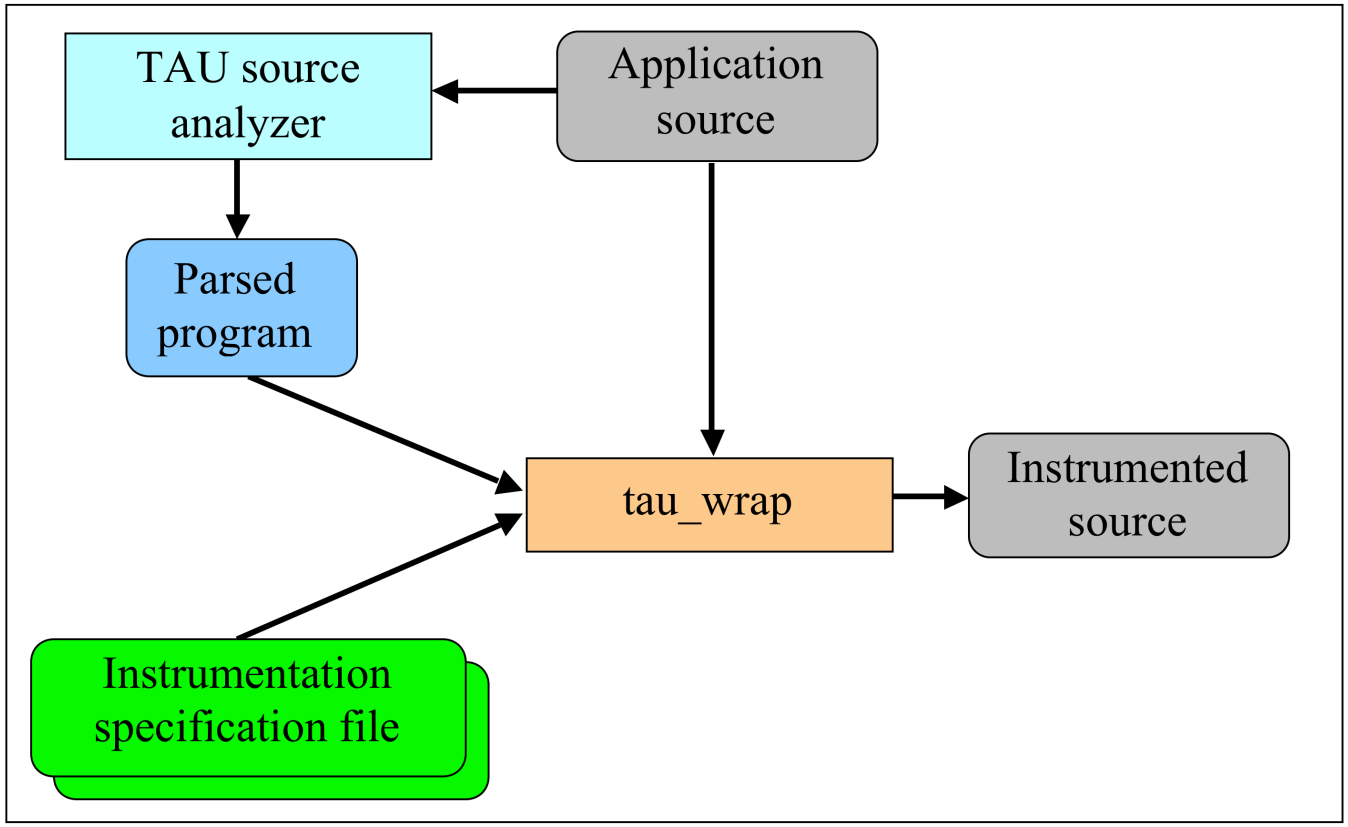
- parses source of header file using static analysis tool Program Database Toolkit (PDT)
- Invokes *tau_wrap*, a tool that generates
  - instrumented wrapper code,
  - an optional *link_options.tau* file (for linker-based substitution, -w)
  - Makefile for compiling the wrapper interposition library
- Builds the wrapper library using make

Use TAU_OPTIONS environment variable to pass location of link_options.tau file using

```
% export TAU_OPTIONS='-optTauWrapFile=<path/to/
link_options.tau> -optVerbose'
```

Use *tau_exec –loadlib=<wrapperlib.so>* to pass location of wrapper library for preloading based substitution

tau_wrap


TAU source analyzer
Application source
Parsed program
tau_wrap
Instrumented source
Instrumentation specification file

Using POSIX I/O wrapper library

Setting environment variable TAU_OPTIONS=-optTrackIO links in TAU's wrapper interposition library using linker-based substitution

Instrumented application generates bandwidth, volume data

Workflow:

- % export TAU_OPTIONS= '-optTrackIO –optVerbose'
- % export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
- % make CC=tau_cc.sh CXX=tau_cxx.sh CC=tau_cc.sh
- % mpirun –np 8 ./a.out
- % paraprof

Get additional data regarding individual arguments by setting environment variable TAU_TRACK_IO_PARAMS=1 prior to running

Preloading a wrapper library

Preloading a library at runtime

- Tool defines read(), gets address of global read() symbol (dlsym), internally calls timing calls around call to global read
- *tau_exec* tool uses this mechanism to intercept library calls

Advantages

- No need to re-compile or re-link the application source code
- Drop-in replacement library implemented using LD_PRELOAD environment variable under Linux, Cray CNL, IBM BG/P CNK, Solaris…

Disadvantages

- Only works with dynamic executables. Default compilation mode under Cray XE6 and IBM BG/P is to use static executables
- Not all operating systems support preloading of dynamic shared objects (DSOs)

TAU for Heterogeneous Measurement

Multiple performance perspectives

Integrate Host-GPU support in TAU measurement framework

- Enable use of each measurement approach
- Include use of PAPI and CUPTI
- Provide profiling and tracing support

Tutorial

- Use TAU library wrapping of libraries
- Use `tau_exec` to work with binaries

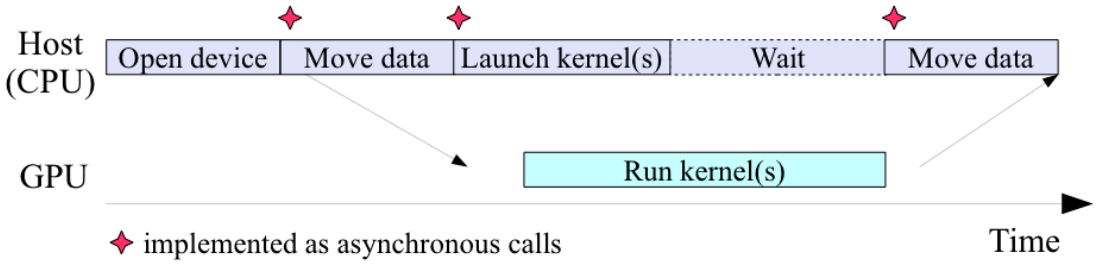
  % ./a.out (uninstrumented)

  % tau_exec –T <configuration tags> –cupti ./a.out

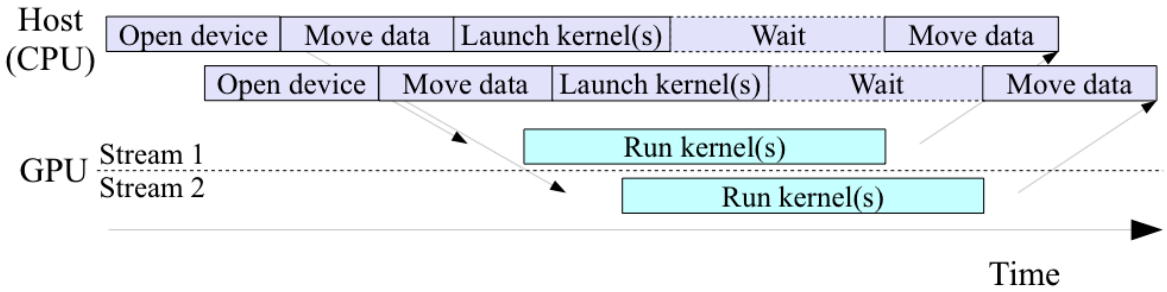
  % paraprof

Host (CPU) - GPU Scenarios

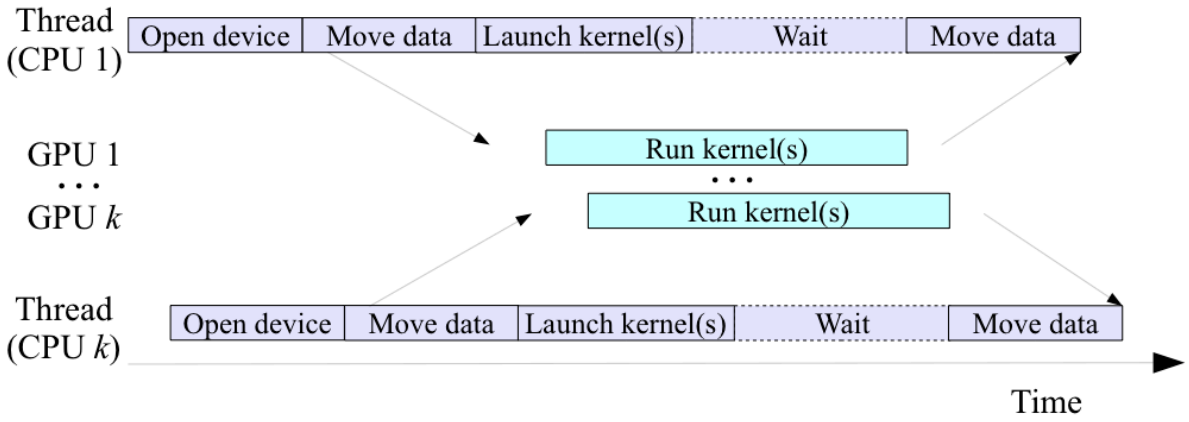
Single GPU



Multi-stream

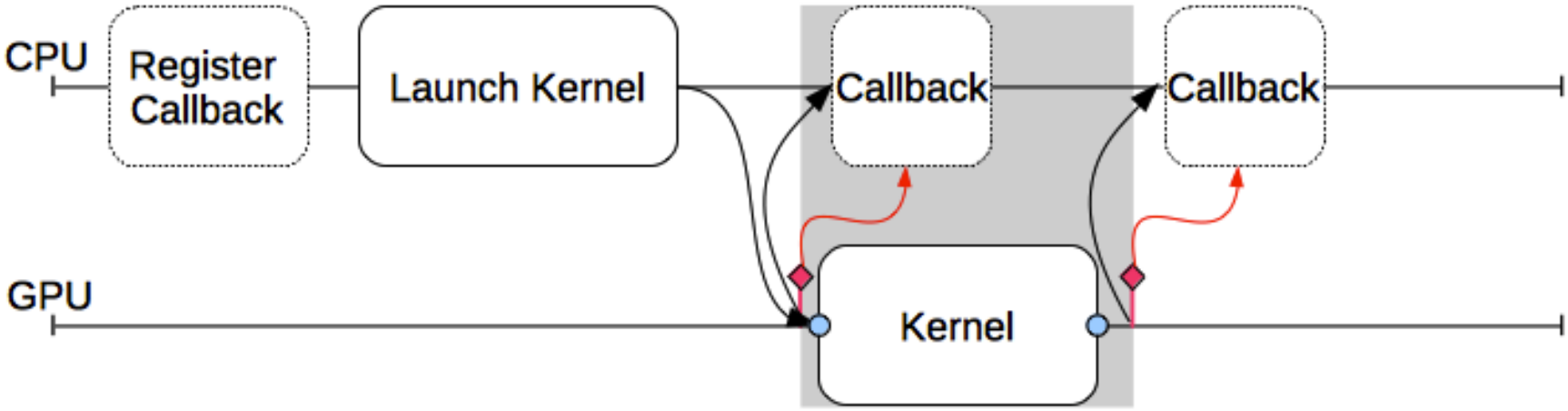


Multi-CPU, Multi-GPU



Host-GPU Measurement – Callback Method

- GPU driver libraries provide callbacks for certain routines and captures measurements
- Measurement tool registers the callbacks and processes performance data
- Application code is not modified



Method Support and Implementation

Synchronous method

- Place instrumentation appropriately around GPU calls (kernel launch, library routine, …)
- Wrap (synchronous) library with performance tool

Event queue method

- Utilize CUDA and OpenCL event support
- Again, need instrumentation to create and insert events in the streams with kernel launch and process events
- Can be implemented with driver library wrapping

Callback method

- Utilize language-level callback support in OpenCL
- Utilize NVIDIA CUDA Performance Tool Interface (CUPTI)
- Need to appropriately register callbacks

GPU Performance Measurement Tools

Support the Host-GPU performance perspective

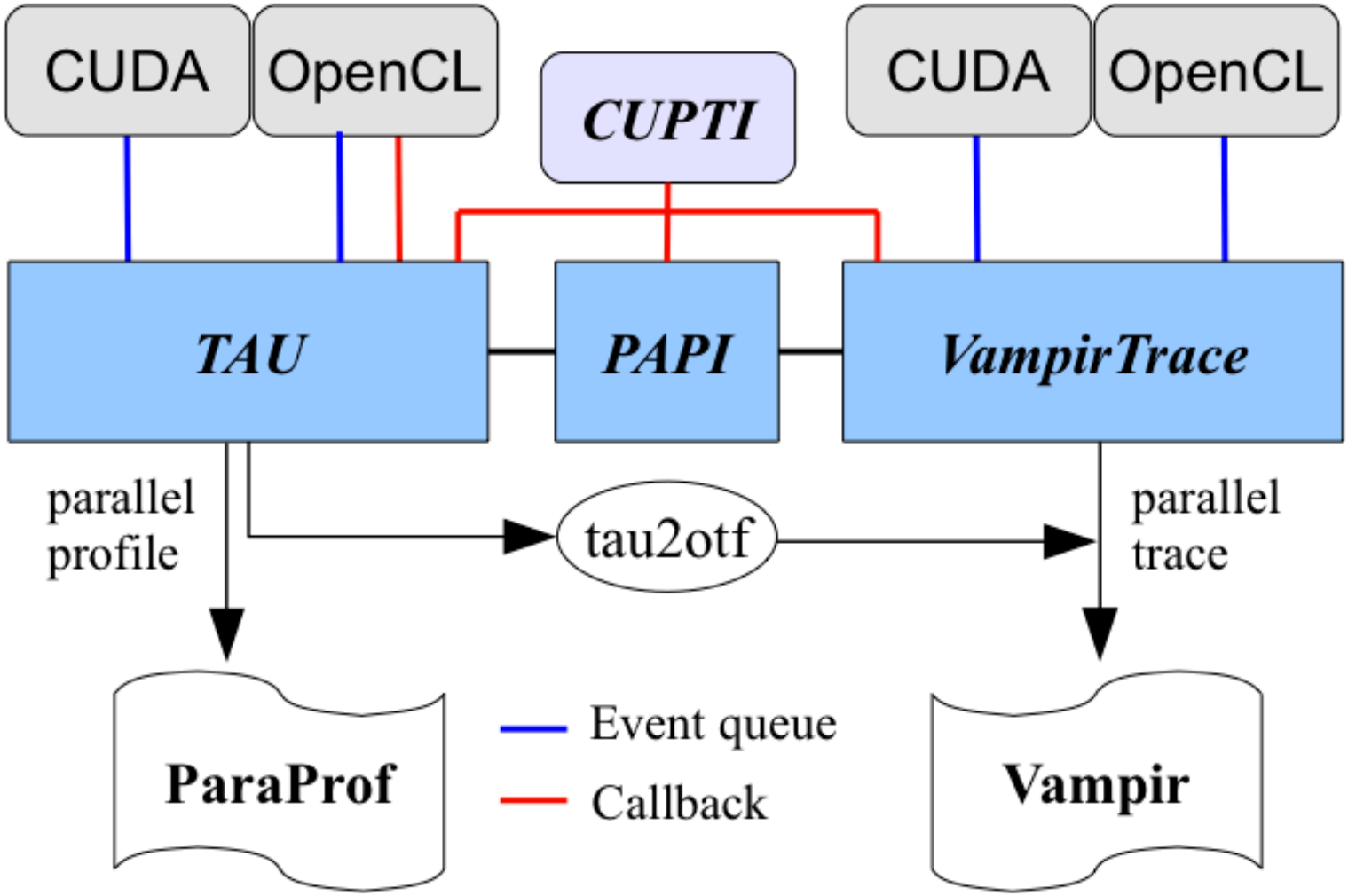
Provide integration with existing measurement system to facilitate tool use

Utilize support in GPU driver library and device

Tools

- TAU performance system
- Vampir
- PAPI
- NVIDIA CUPTI

GPU Performance Tool Interoperability



NVIDIA CUPTI

NVIDIA is developing CUPTI to enable the creation of profiling and tracing tools

Callback API

- Interject tool code at the entry and exist to each CUDA runtime and driver API call

Counter API

- Query, configure, start, stop, and read the counters on CUDA-enabled devices

CUPTI is delivered as a dynamic library

CUPTI is released with CUDA 4.0+

TAU for Heterogeneous Measurement

Multiple performance perspectives

Integrate Host-GPU support in TAU measurement framework

- Enable use of each measurement approach
- Include use of PAPI and CUPTI
- Provide profiling and tracing support

Tutorial

- Use TAU library wrapping of libraries
- Use `tau_exec` to work with binaries

```
% ./a.out    (uninstrumented)
% tau_exec –T serial,cupti –cupti ./a.out
% paraprof
```

Example: SDK simpleMultiGPU

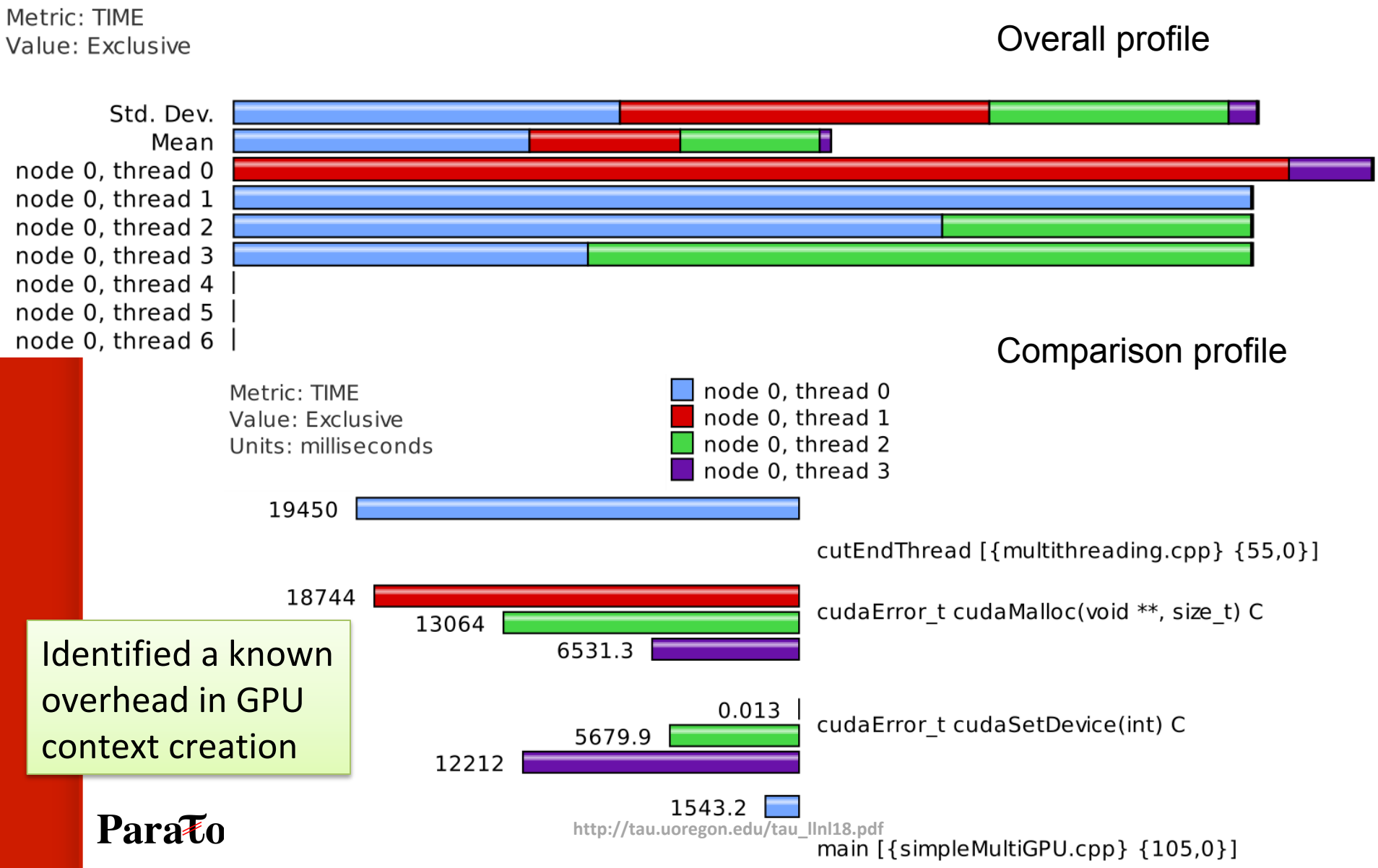
Demonstration of multiple GPU device use

main  ***solverThread***  ***reduceKernel***

Performance profile for:

- One *main* thread
- Three *solverThread* threads
- Three *reduceKernel* "threads"

simpleMultiGPU Profile



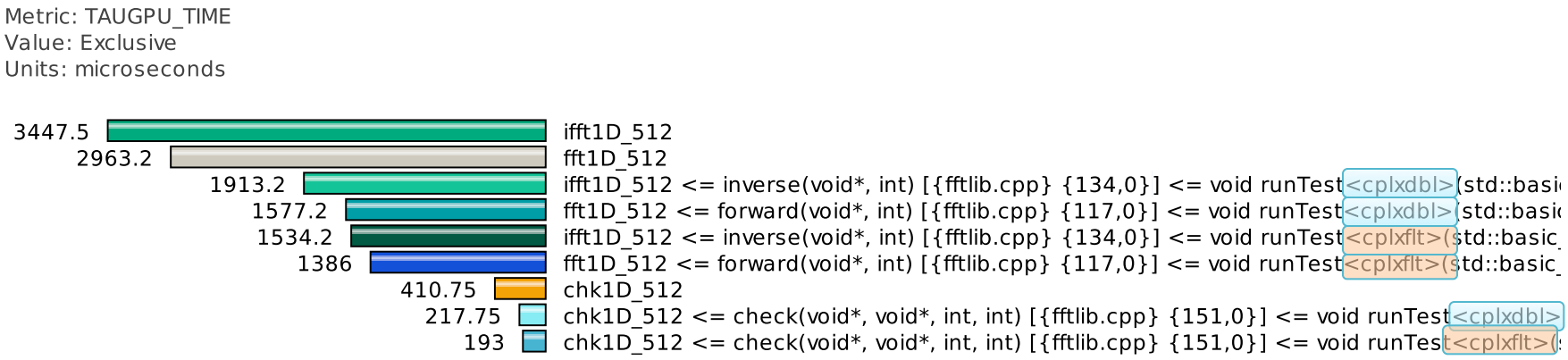
Identified a known overhead in GPU context creation

ParaTo

http://tau.uoregon.edu/tau_llnl18.pdf

SHOC FFT Profile with Callsite Info

TAU is able to associate callsite context information with kernel launch so that different kernel calls can be distinguished



Each kernel (ifft1D_512, fft1D_512 and chk1D_512) is broken down by call-site, either during the single precession or double precession step.

Example: SHOC Stencil2D

Compute 2D, 9-point stencil

- Multiple GPUs using MPI
- CUDA and OpenCL versions

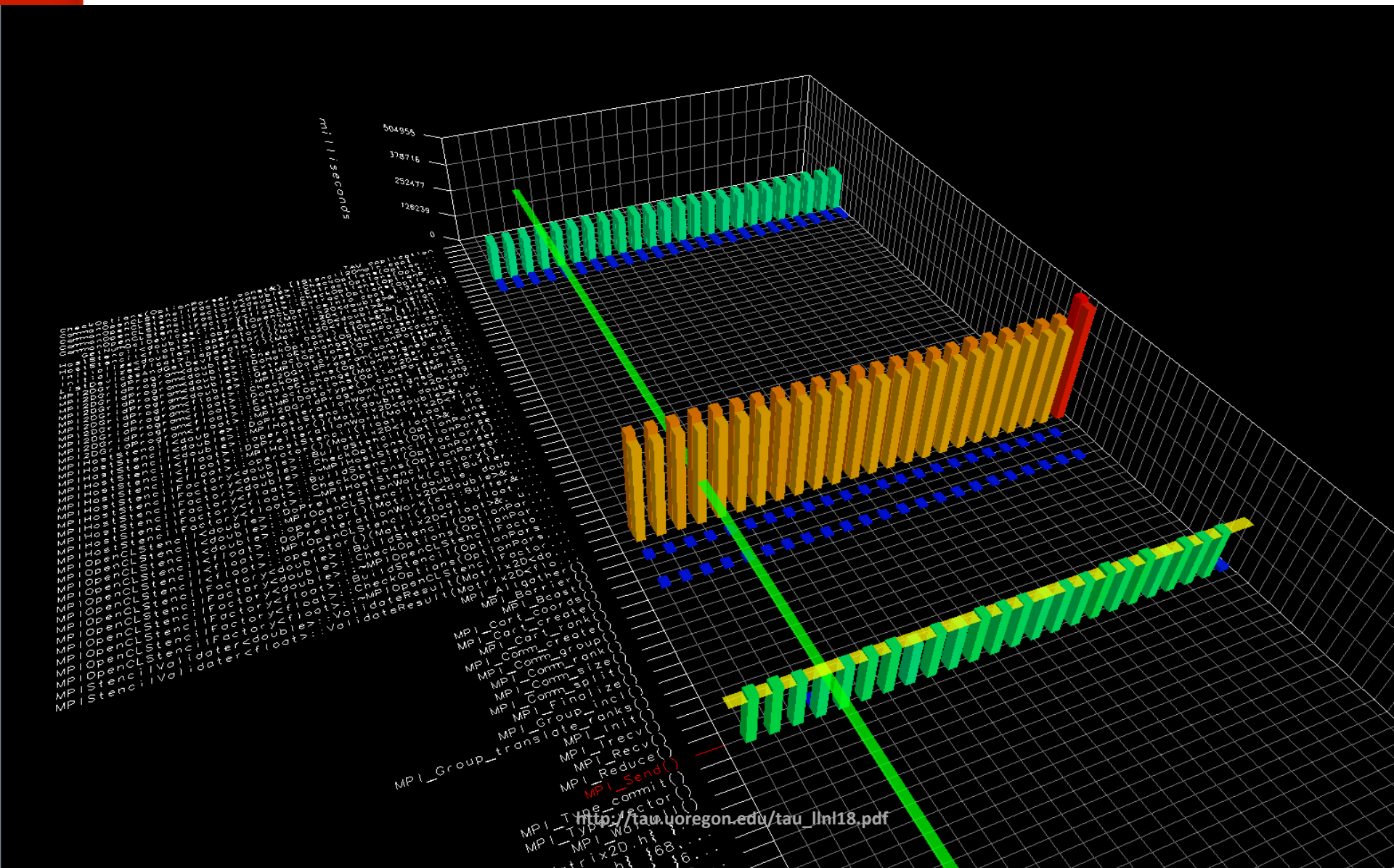
One Keeneland node with 3 GPUs

Eight Keeneland nodes with 24 GPUs

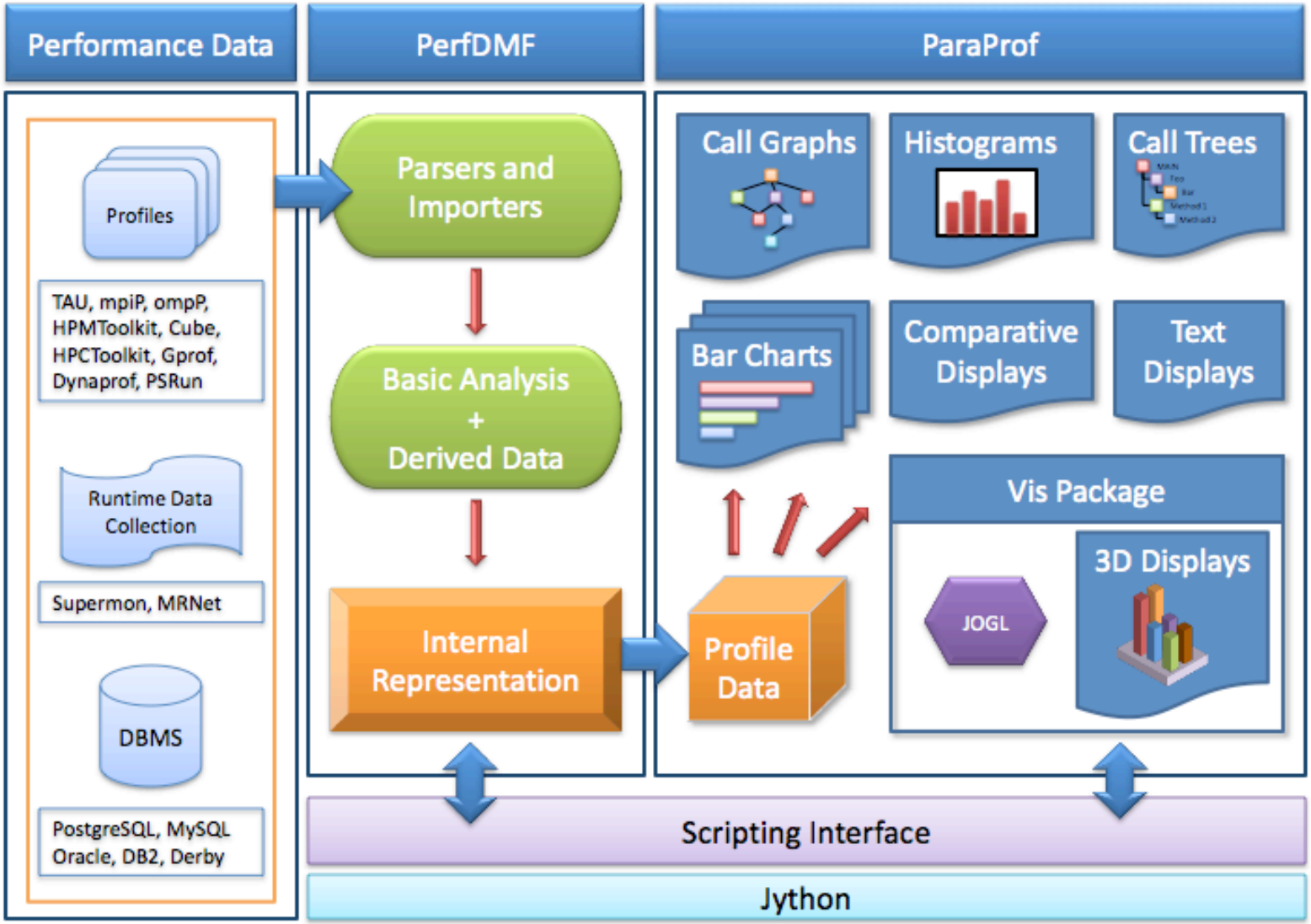
Performance profile and trace

- Application events
- Communication events
- Kernel execution

Stencil2D Parallel Profile

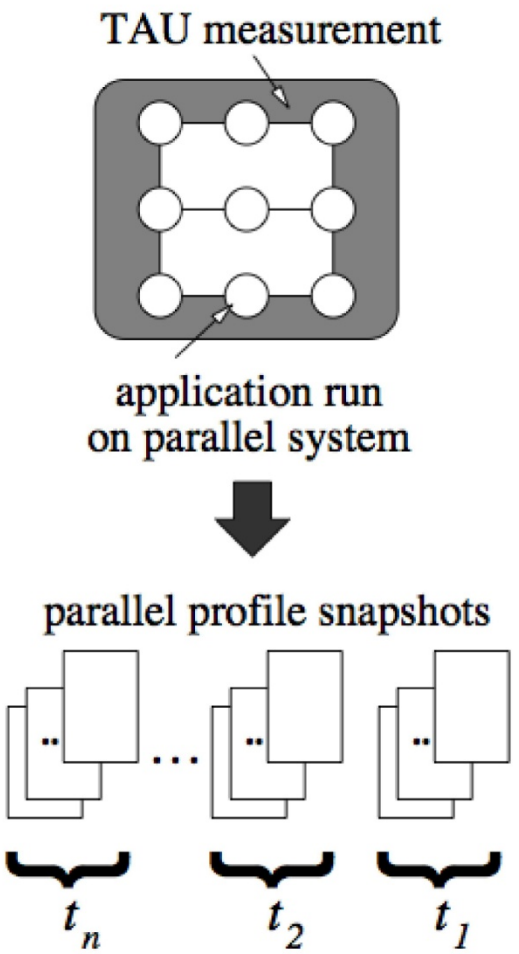


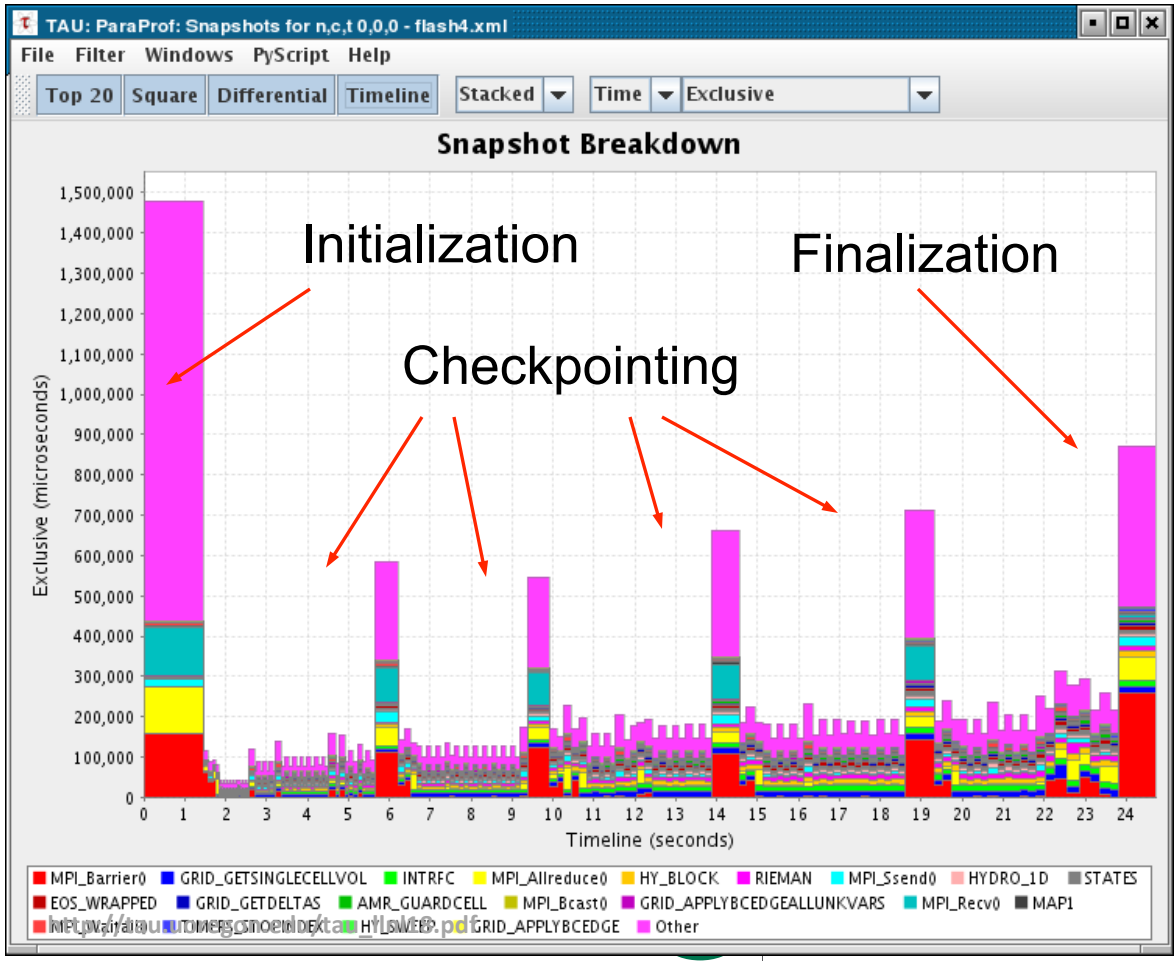
ParaProf Profile Analysis Framework



Profile Snapshots in ParaProf

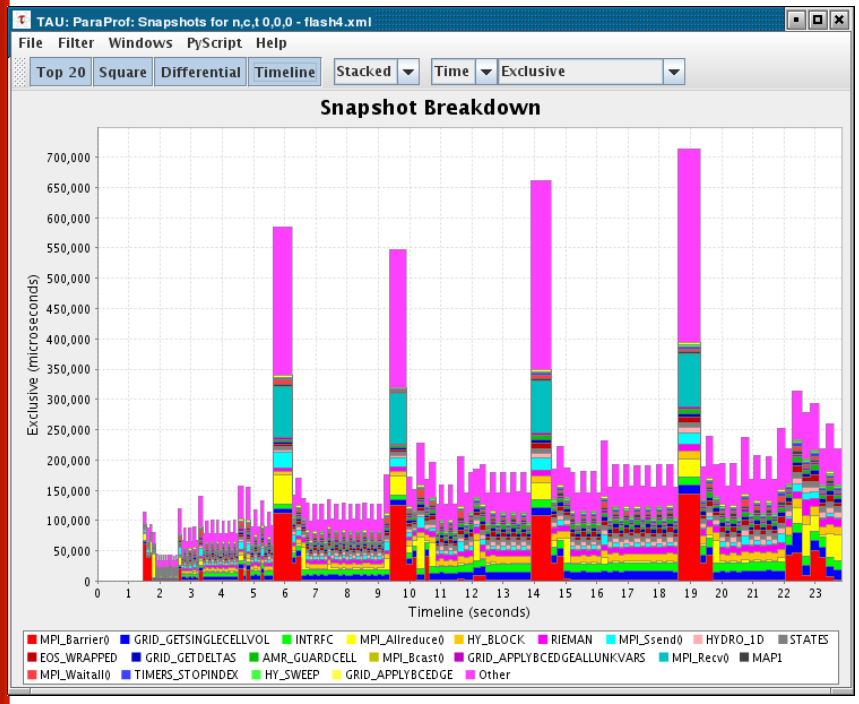
- Profile snapshots are profiles recorded at runtime
- Shows performance profile dynamics (all types allowed)



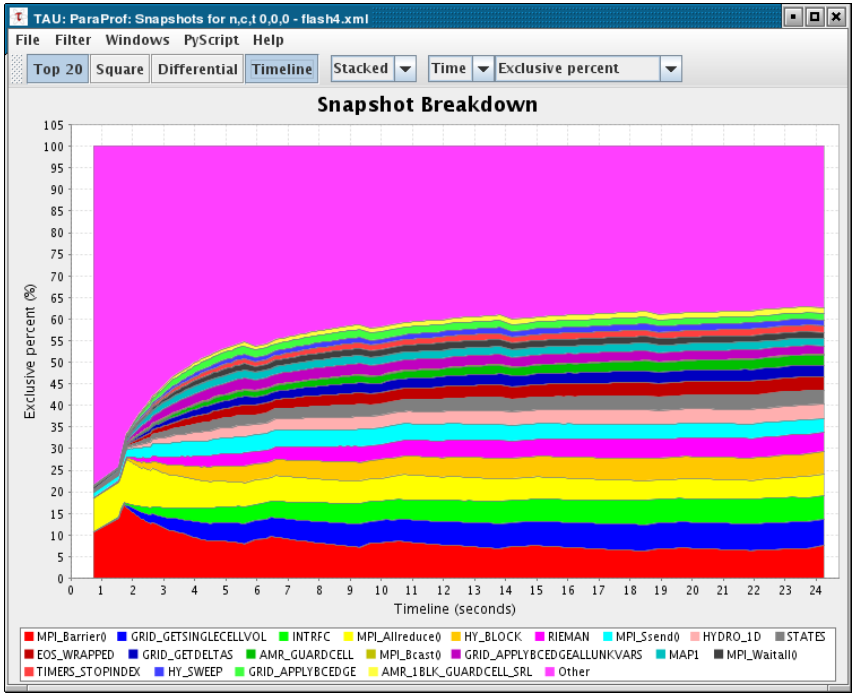


Profile Snapshot Views

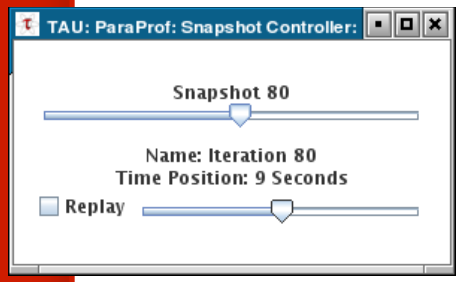
Percentage breakdown



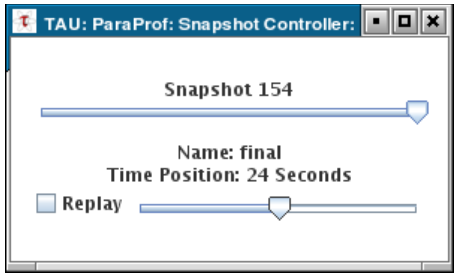
Only show main loop

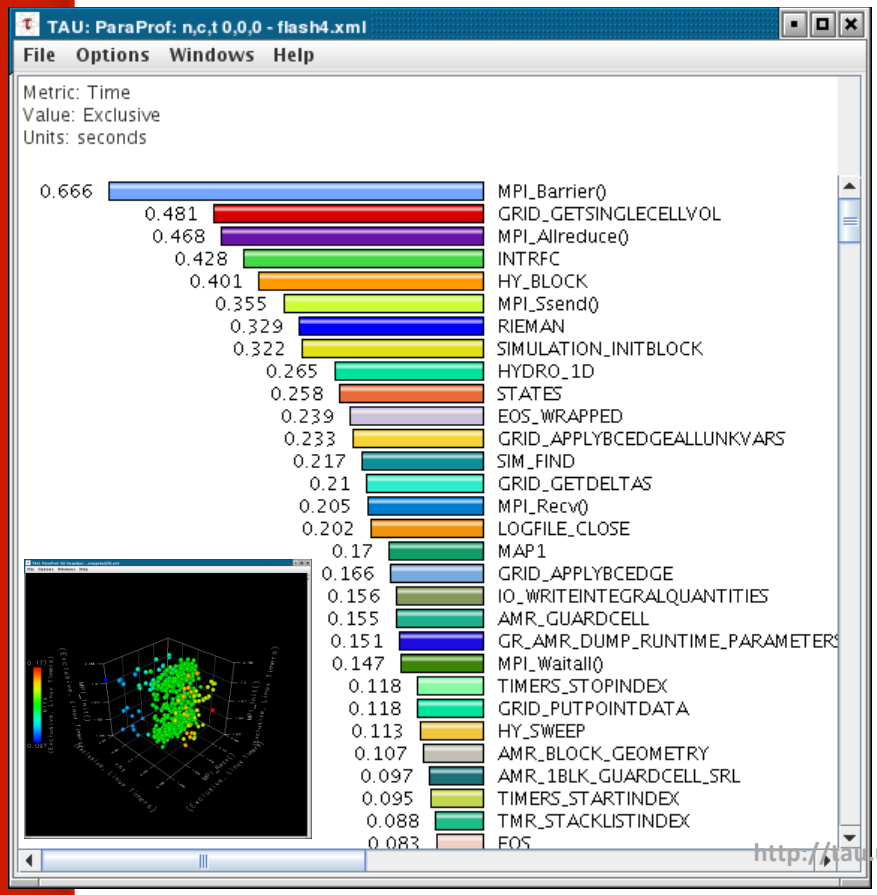


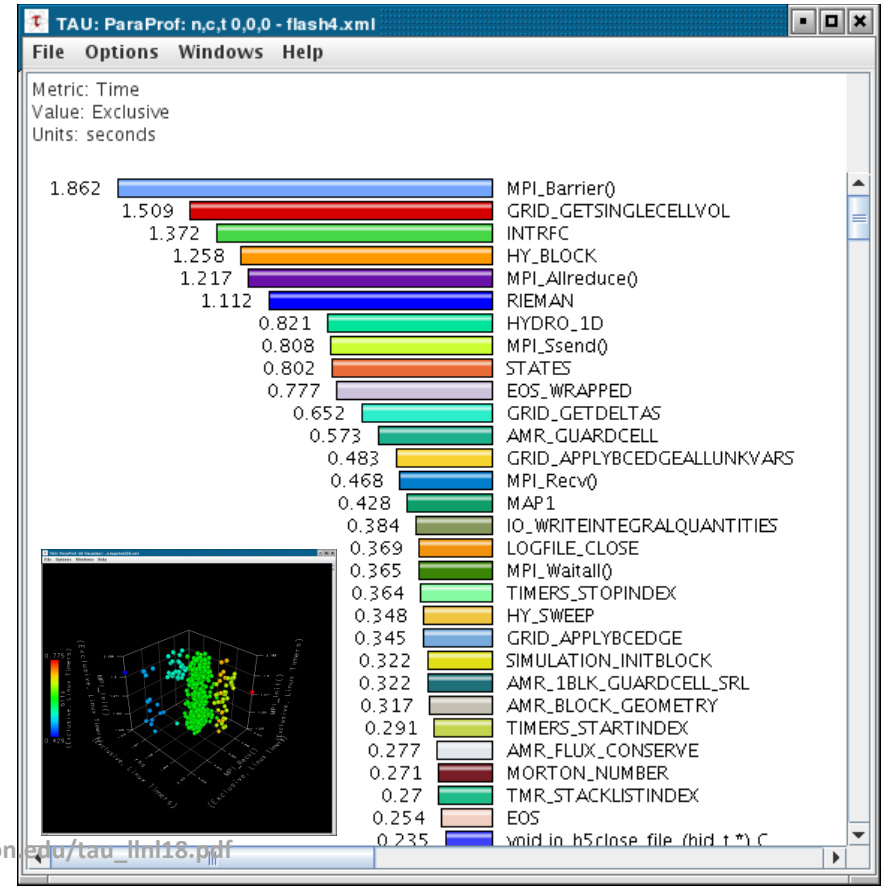
Snapshot Replay in ParaProf



All windows dynamically update

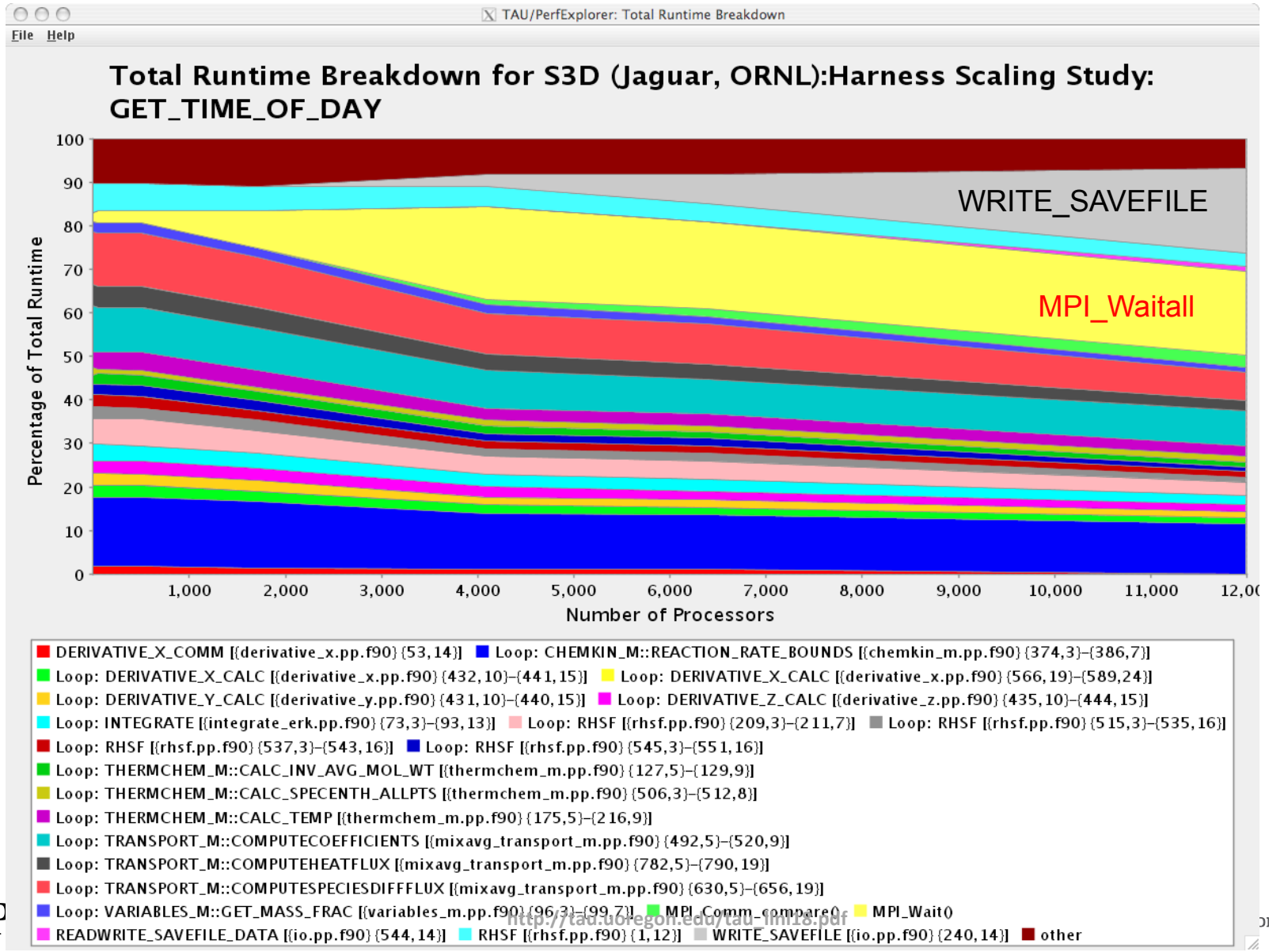




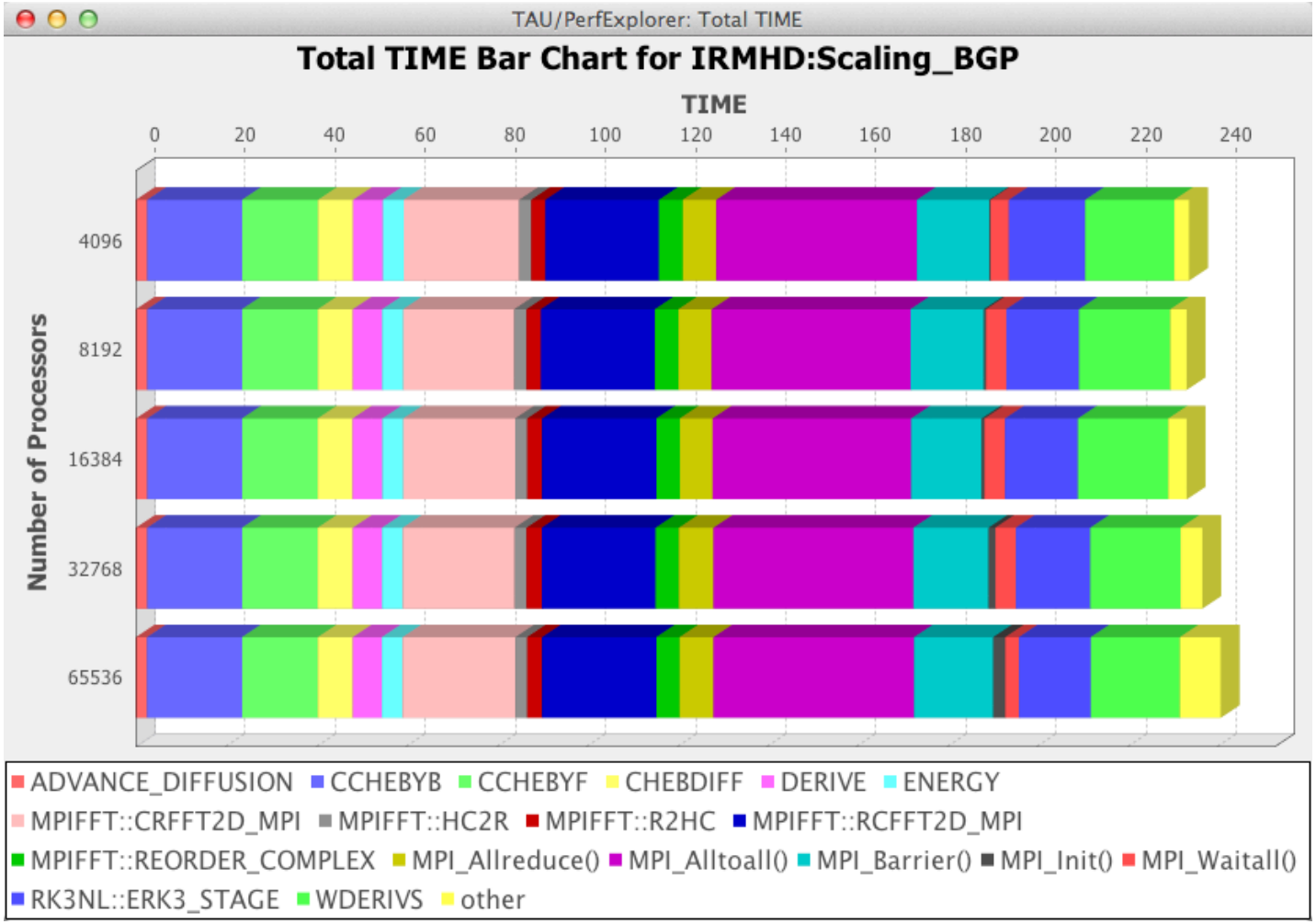


Tools: PerfExplorer

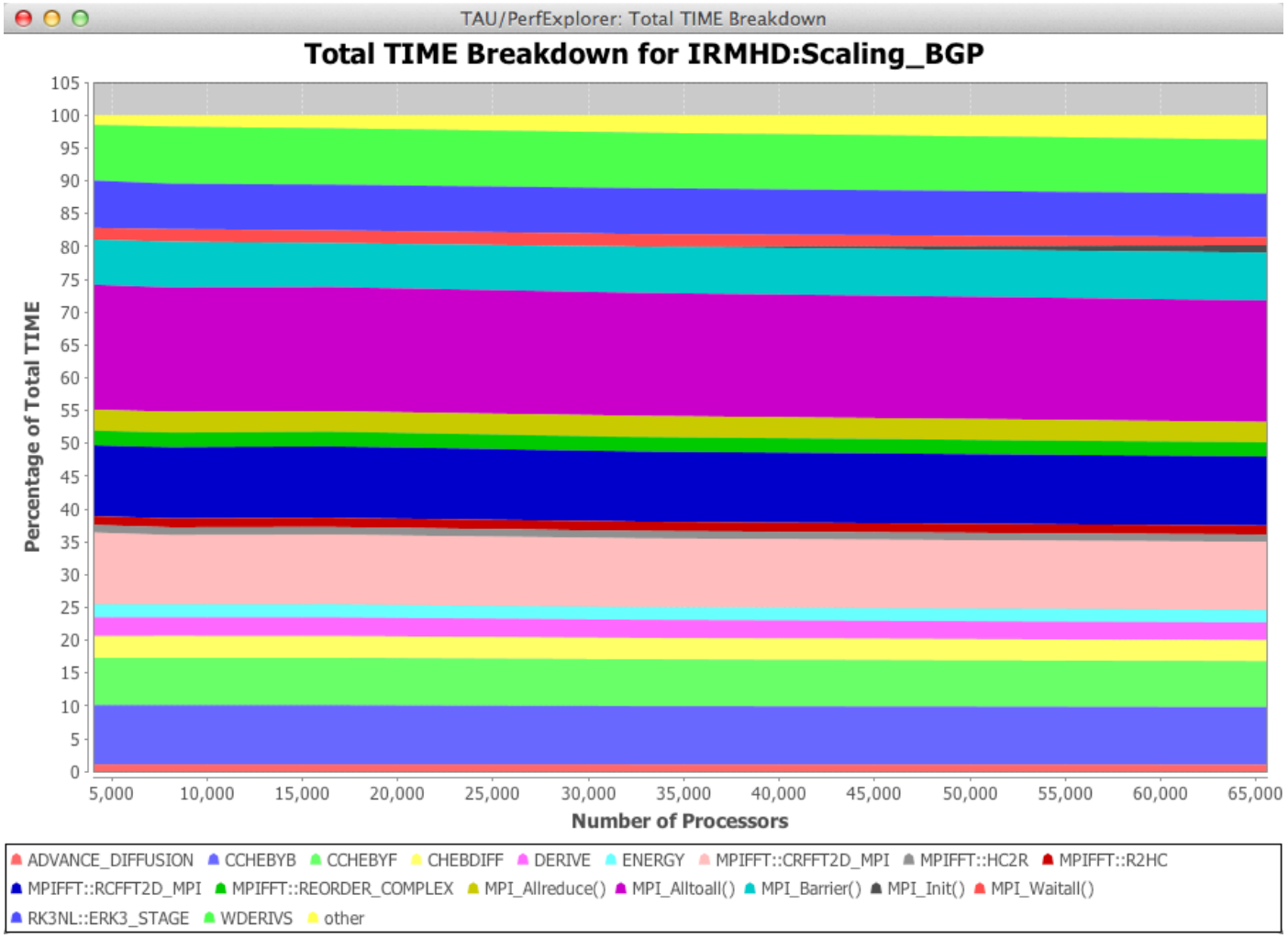
PerfExplorer – Runtime Breakdown



Evaluate Scalability

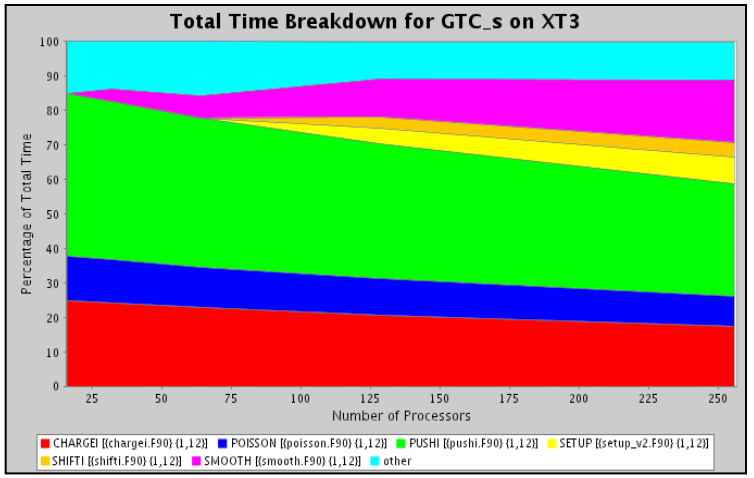


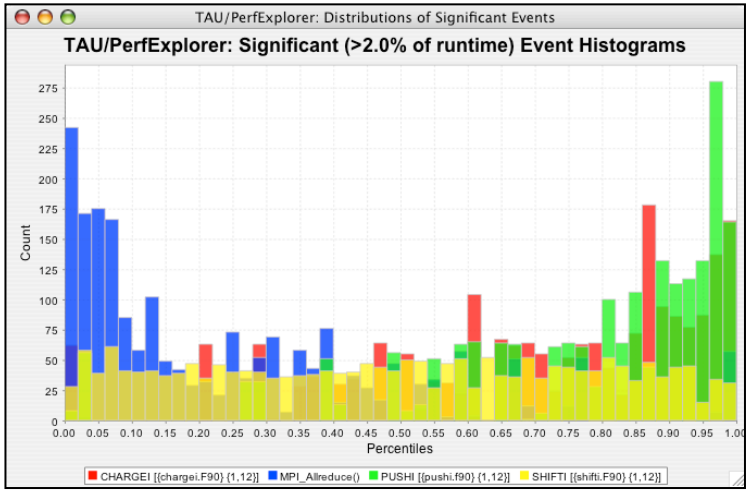
Runtime Breakdown



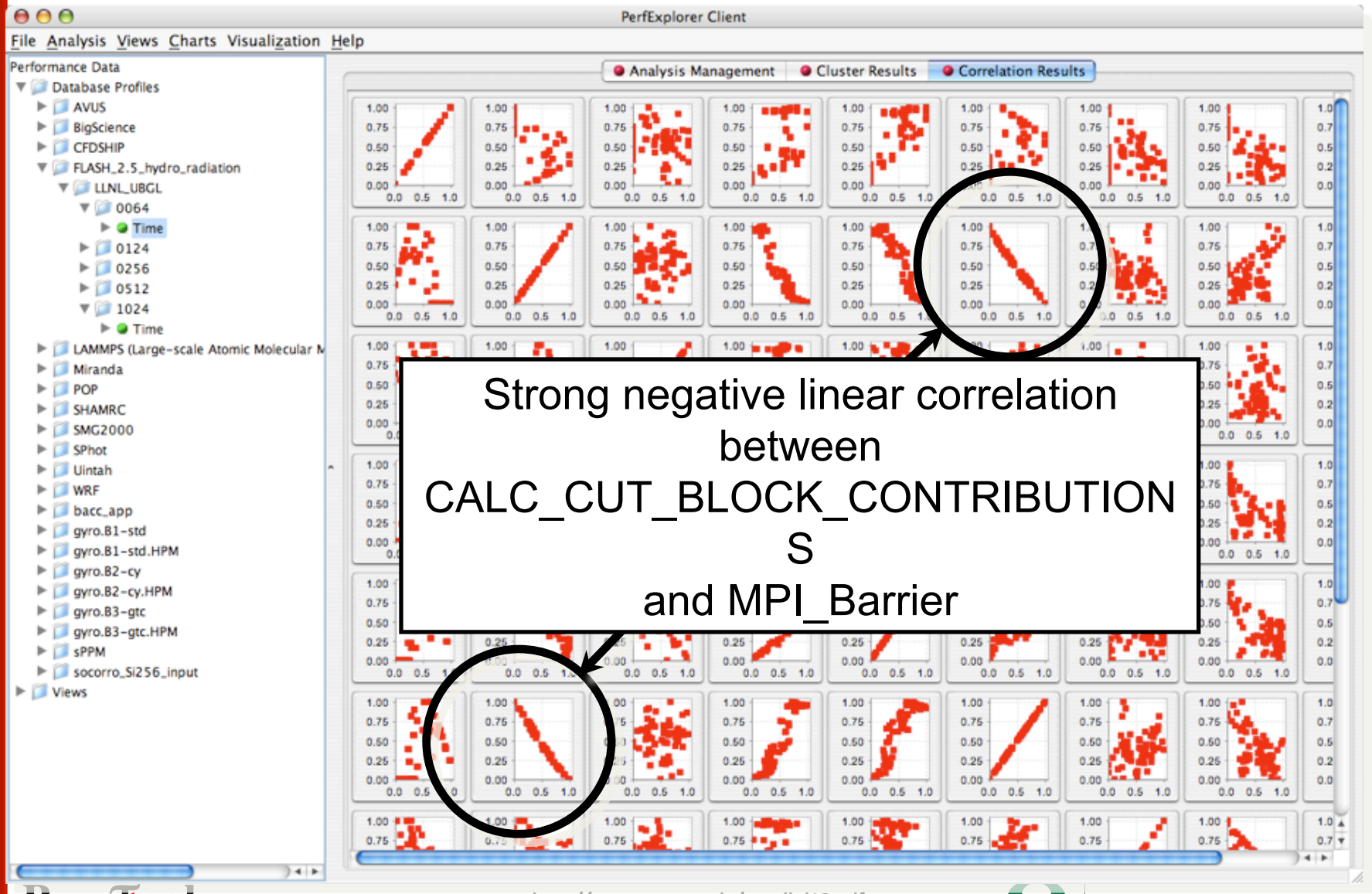
PerfExplorer – Relative Comparisons

- Total execution time
- Timesteps per second
- Relative efficiency
- Relative efficiency per event
- Relative speedup
- Relative speedup per event
- Group fraction of total
- Runtime breakdown
- Correlate events with total runtime
- Relative efficiency per phase
- Relative speedup per phase
- Distribution visualizations



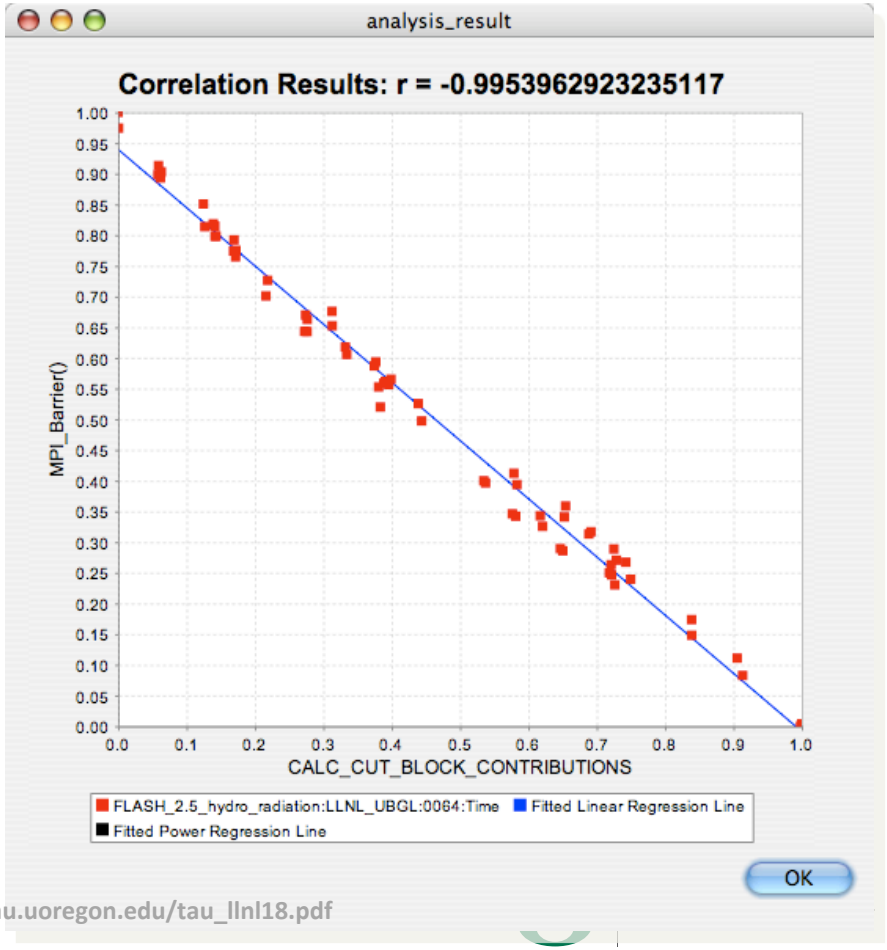


PerfExplorer – Correlation Analysis

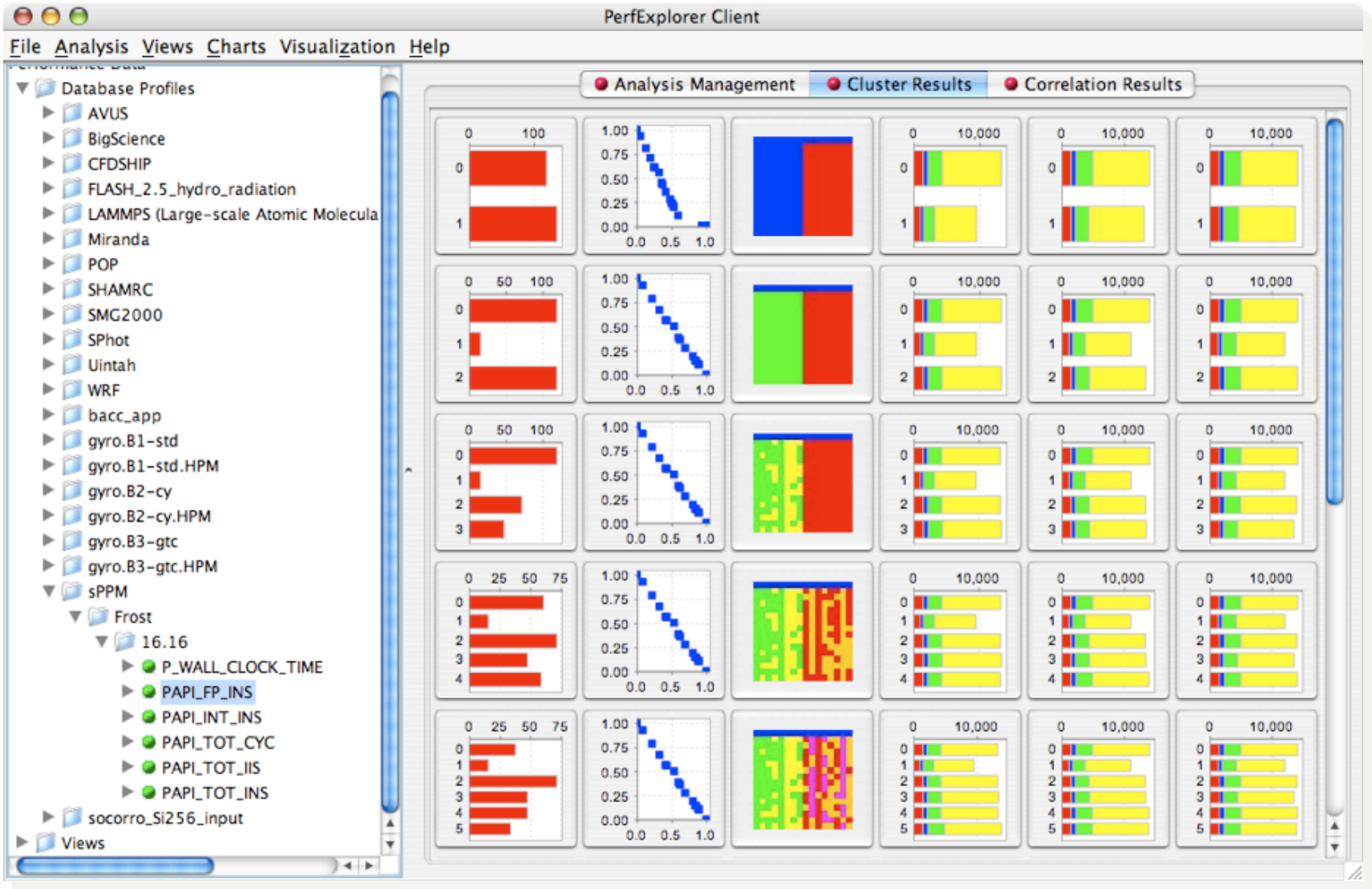


PerfExplorer – Correlation Analysis

-0.995 indicates strong, negative relationship. As CALC_CUT_BLOCK_CONTRIBUTIONS() increases in execution time, MPI_Barrier() decreases



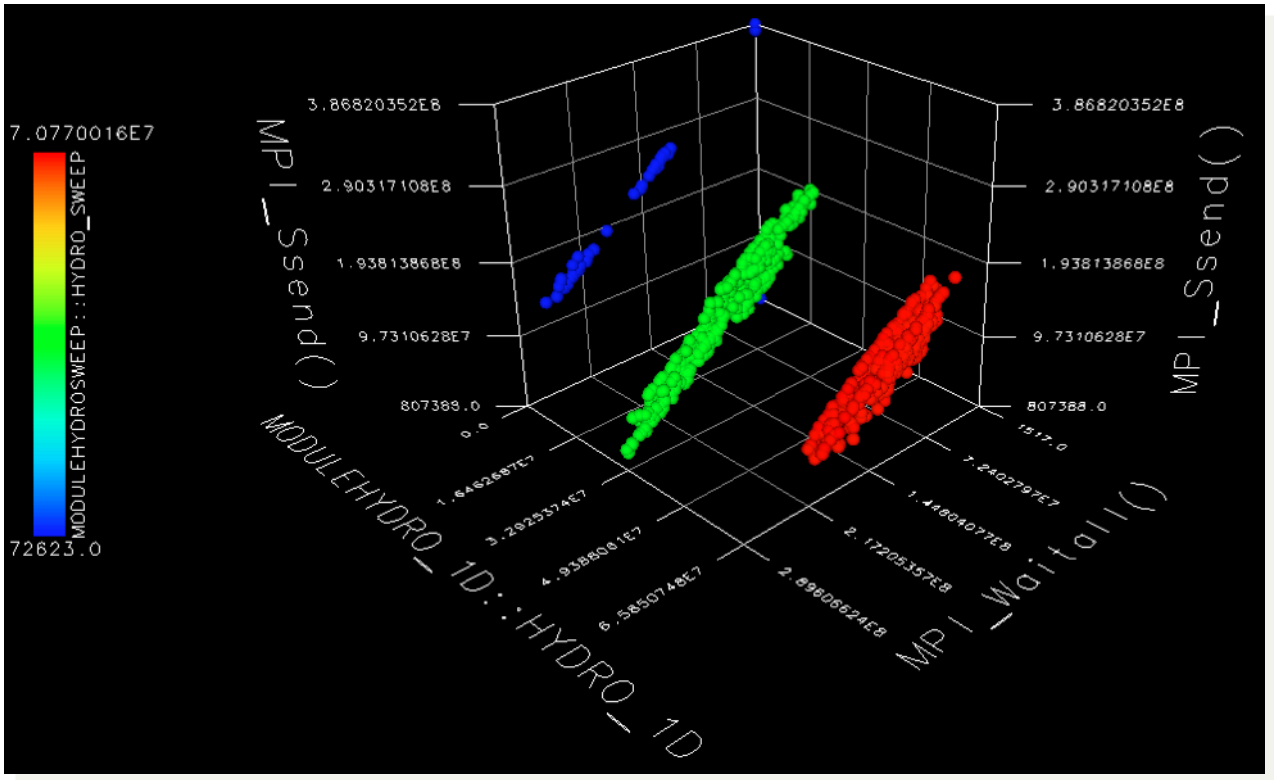
PerfExplorer – Cluster Analysis



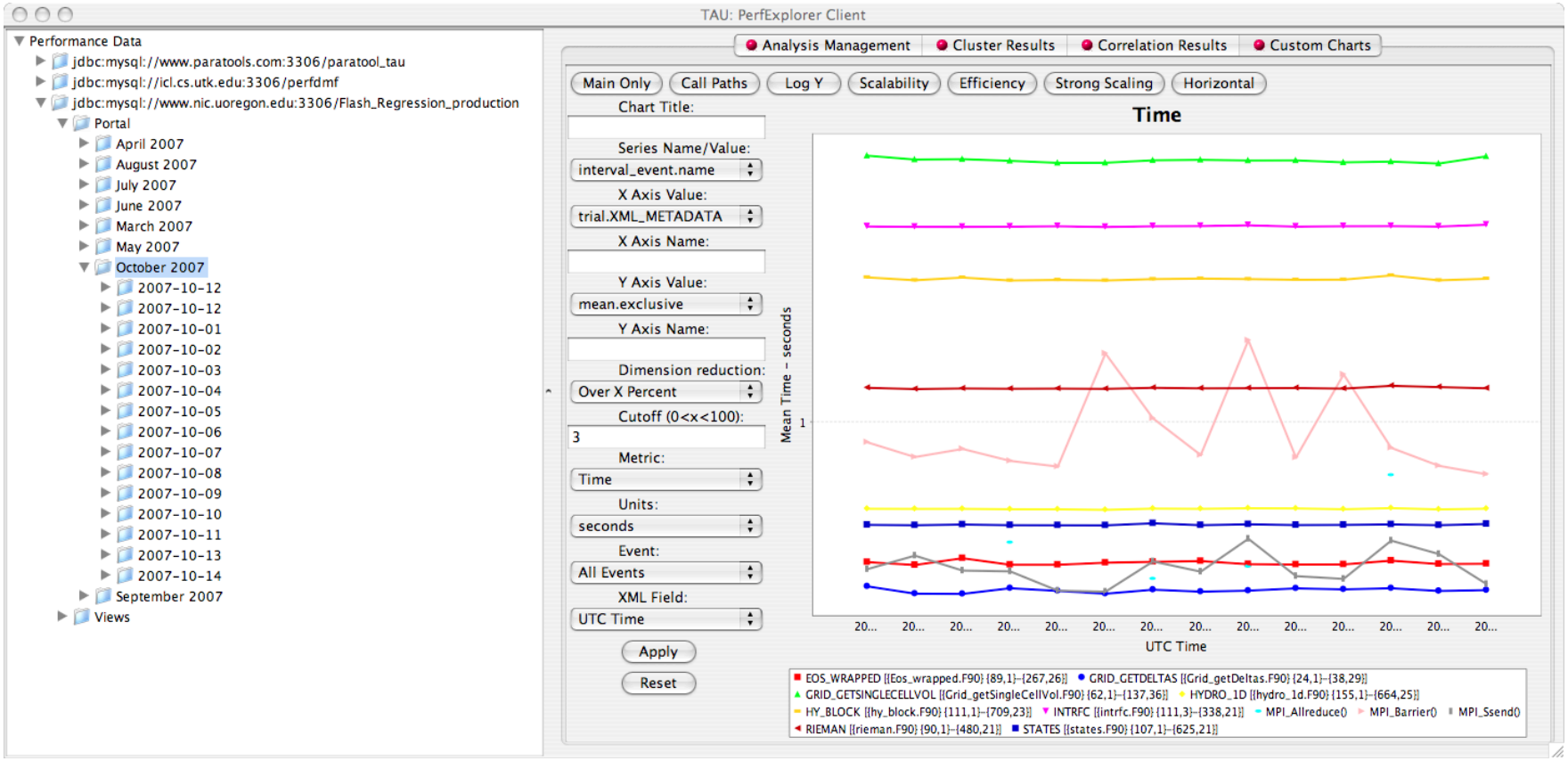
PerfExplorer – Cluster Analysis

Four significant events automatically selected

Clusters and correlations are visible



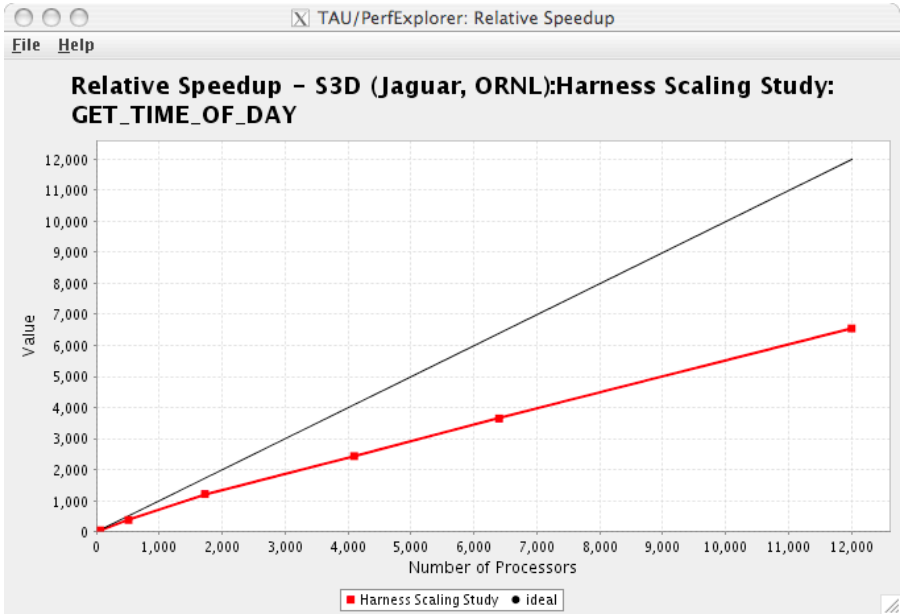
PerfExplorer – Performance Regression

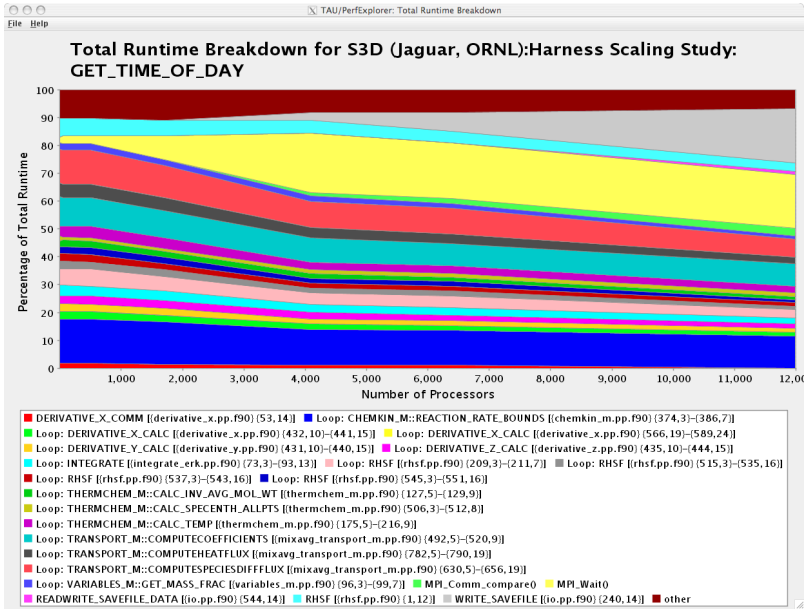


Evaluate Scalability

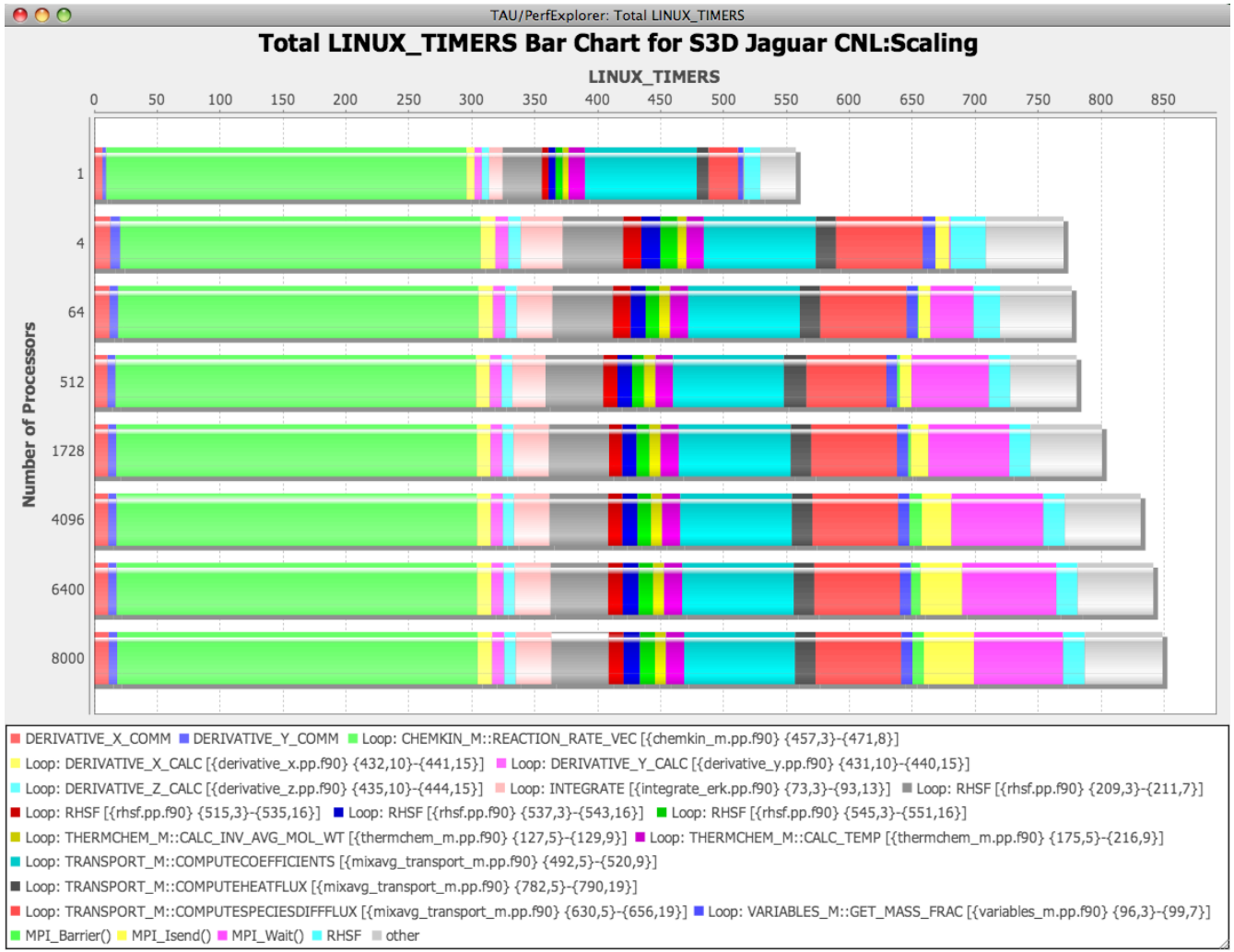
Goal: How does my application scale? What bottlenecks at what CPU counts?

Load profiles in PerfDMF database and examine with PerfExplorer

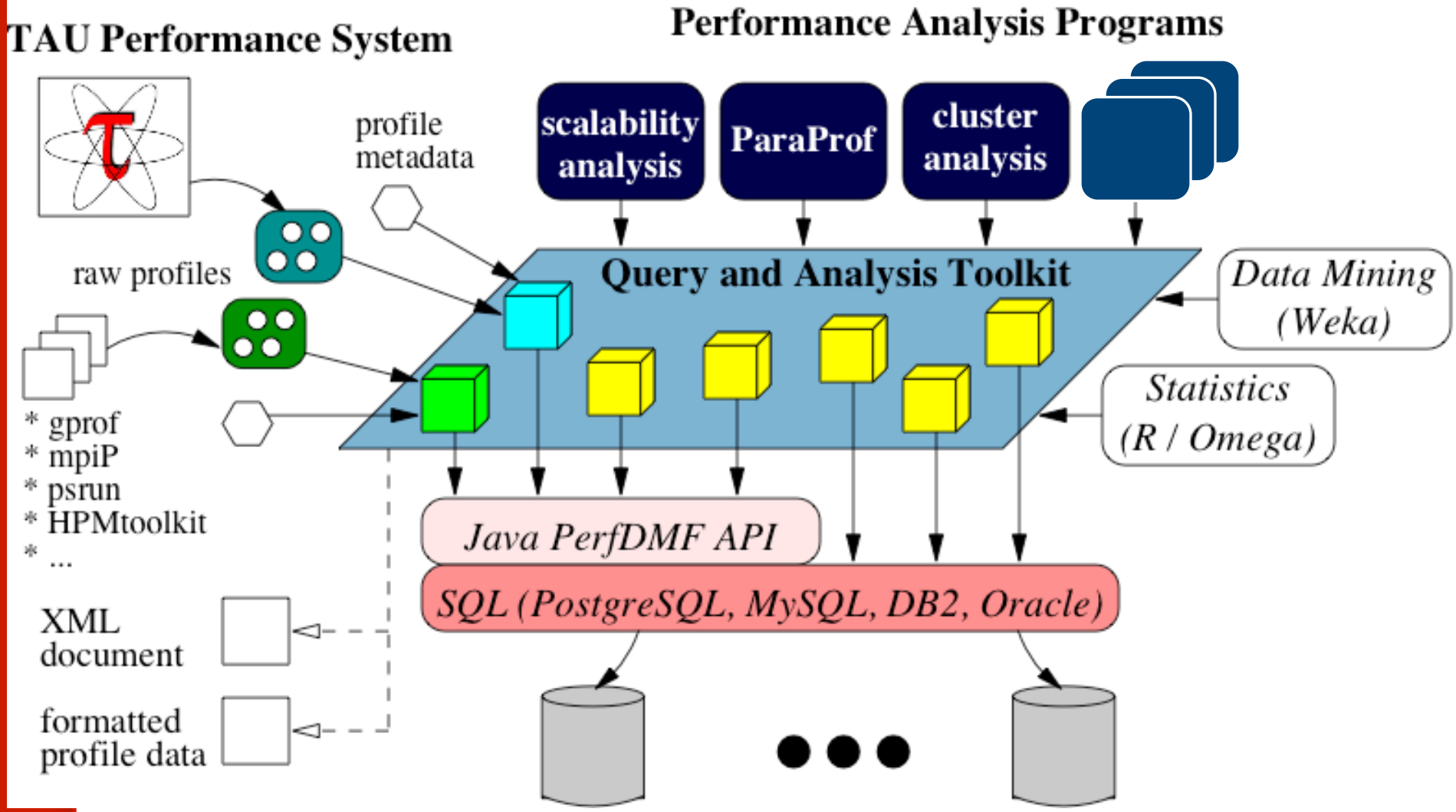




Usage Scenarios: Evaluate Scalability



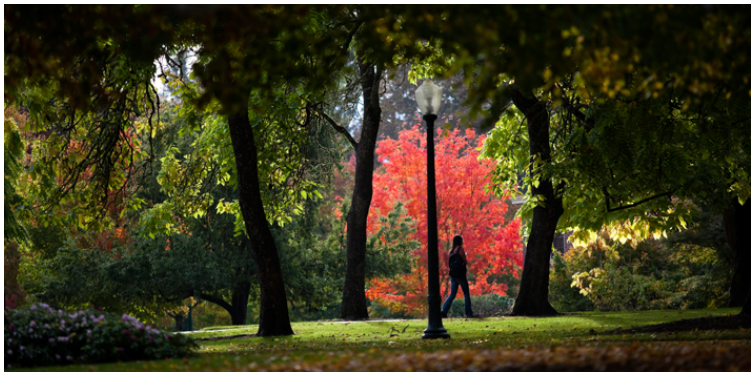
TAUdb: Framework for Managing Performance Data



Evaluate Scalability using PerfExplorer Charts

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt
% make CC=tau_cc.sh
(Or edit Makefile and change CC=tau_cc.sh )
% qsub  run1p.job
% paraprof --pack 1p.ppk
% qsub  run2p.job …
% paraprof --pack 2p.ppk … and so on.
On your client:
% taudb_configure --create-default
% perfexplorer_configure
(Enter, y to load schema, defaults)
% paraprof
(load each trial: DB -> Add Trial -> Type (Paraprof Packed
Profile) -> OK, OR use taudb_loadtrial on the commandline)
% taudb_loadtrial -a App -x MyExp -n 4p 4p.ppk
% perfexplorer
(Charts -> Speedup)
OR:
wget http://tau.uoregon.edu/data.tgz; cat README in data
```

PRL, University of Oregon, Eugene



www.uoregon.edu

Support Acknowledgments

US Department of Energy (DOE)

- ANL
- Office of Science contracts, ECP
- SciDAC, LBL contracts
- LLNL-LANL-SNL ASC/NNSA contract
- Battelle, PNNL and ORNL contract

CEA, France

Department of Defense (DoD)

- PETTT, HPCMP

National Science Foundation (NSF)

- SI2-SSI, Glassbox

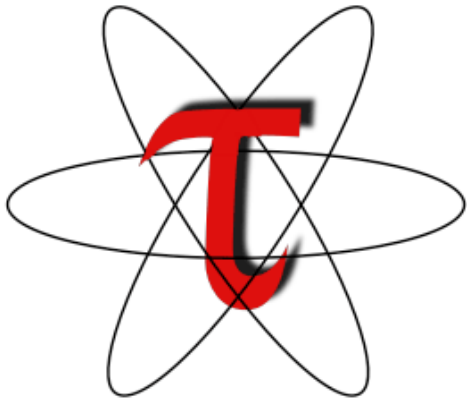
Intel Corporation

NASA

Partners:

- University of Oregon
- The Ohio State University
- ParaTools, Inc.
- University of Tennessee, Knoxville
- T.U. Dresden, GWT
- Jülich Supercomputing Center

Download TAU from U. Oregon



http://tau.uoregon.edu

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

http://tau.uoregon.edu/ecp
[ECP SDK Containers with TAU]

Free download, open source, BSD license

Reference

Installing and Configuring TAU

- **Installing PDT:**
  - wget tau.uoregon.edu/pdt_lite.tgz
  - ./configure –prefix=<dir>; make ; make install
- **Installing TAU:**
  - wget tau.uoregon.edu/tau.tgz; tar zxf tau.tgz; cd tau-2.<ver>
  - wget http://tau.uoregon.edu/ext.tgz
  - ./configure -bfd=download -pdt=<dir> -papi=<dir> ...
  - make install
- **Using TAU:**
  - export TAU_MAKEFILE=<taudir>/x86_64/ lib/Makefile.tau-<TAGS>
  - make CC=tau_cc.sh CXX=tau_cxx.sh CC=tau_cc.sh

Compile-Time Options

Optional parameters for the TAU_OPTIONS environment variable:
% tau_compiler.sh

-optVerbose	Turn on verbose debugging messages
-optCompInst	Use compiler based instrumentation
-optNoCompInst	Do not revert to compiler instrumentation if source instrumentation fails.
-optTrackIO	Wrap POSIX I/O call and calculates vol/bw of I/O operations (Requires TAU to be configured with *–iowrapper*)
-optTrackGOMP	Enable tracking GNU OpenMP runtime layer (used without –opari)
-optMemDbg	Enable runtime bounds checking (see TAU_MEMDBG_* env vars)
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optPreProcess	Preprocess sources (OpenMP, Fortran) before instrumentation
-optTauSelectFile=”<file>"	Specify selective instrumentation file for *tau_instrumentor*
-optTauWrapFile=”<file>"	Specify path to *link_options.tau* generated by *tau_gen_wrapper*
-optHeaderInst	Enable Instrumentation of headers
-optTrackUPCR	Track UPC runtime layer routines (used with tau_upc.sh)
-optLinking=""	Options passed to the linker. Typically $(TAU_MPI_FLIBS) $(TAU_LIBS) $(TAU_CXXLIBS)
-optCompile=""	Options passed to the compiler. Typically $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)
-optPdtF95Opts=""	Add options for Fortran parser in PDT (f95parse/gfparse) …

Compile-Time Options (contd.)

Optional parameters for the TAU_OPTIONS environment variable:
% tau_compiler.sh

-optShared	Use TAU's shared library (libTAU.so) instead of static library (default)
-optPdtCxxOpts=""	Options for C++ parser in PDT (cxxparse).
-optPdtF90Parser=""	Specify a different Fortran parser
-optPdtCleanscapeParser	Specify the Cleanscape Fortran parser instead of GNU gfparser
-optTau=""	Specify options to the tau_instrumentor
-optTrackDMAPP	Enable instrumentation of low-level DMAPP API calls on Cray
-optTrackPthread	Enable instrumentation of pthread calls

See tau_compiler.sh for a full list of TAU_OPTIONS.

…

Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_CALLPATH	0	Setting to 1 turns on callpath profiling
TAU_TRACK_MEMORY_FOOTPRINT	0	Setting to 1 turns on tracking memory usage by sampling periodically the resident set size and high water mark of memory usage
TAU_TRACK_POWER	0	Tracks power usage by sampling periodically.
TAU_CALLPATH_DEPTH	2	Specifies depth of callpath. Setting to 0 generates no callpath or routine information, setting to 1 generates flat profile and context events have just parent information (e.g., Heap Entry: foo)
TAU_SAMPLING	1	Setting to 1 enables event-based sampling.
TAU_TRACK_SIGNALS	0	Setting to 1 generate debugging callstack info when a program crashes
TAU_COMM_MATRIX	0	Setting to 1 generates communication matrix display using context events
TAU_THROTTLE	1	Setting to 0 turns off throttling. Throttles instrumentation in lightweight routines that are called frequently
TAU_THROTTLE_NUMCALLS	100000	Specifies the number of calls before testing for throttling
TAU_THROTTLE_PERCALL	10	Specifies value in microseconds. Throttle a routine if it is called over 100000 times and takes less than 10 usec of inclusive time per call
TAU_CALLSITE	0	Setting to 1 enables callsite profiling that shows where an instrumented function was called. Also compatible with tracing.
TAU_PROFILE_FORMAT	Profile	Setting to “merged” generates a single file. “snapshot” generates xml format
TAU_METRICS	TIME	Setting to a comma separated list generates other metrics. (e.g., ENERGY,TIME,P_VIRTUAL_TIME,PAPI_L1_DCM,PAPI_NATIVE_<event>:<subevent>)
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_TRACE_FORMAT	Default	Setting to “otf2” turns on TAU’s native OTF2 trace generation (configure with –otf=download)
TAU_EBS_UNWIND	0	Setting to 1 turns on unwinding the callstack during sampling (use with tau_exec –ebs or TAU_SAMPLING=1)
TAU_TRACK_LOAD	0	Setting to 1 tracks system load on the node
TAU_SELECT_FILE	Default	Setting to a file name, enables selective instrumentation based on exclude/include lists specified in the file.

Runtime Environment Variables (contd.

Environment Variable	Default	Description
TAU_TRACK_MEMORY_LEAKS	0	Tracks allocates that were not de-allocated (needs –optMemDbg or tau_exec –memory)
TAU_EBS_SOURCE	TIME	Allows using PAPI hardware counters for periodic interrupts for EBS (e.g., TAU_EBS_SOURCE=PAPI_TOT_INS when TAU_SAMPLING=1)
TAU_EBS_PERIOD	100000	Specifies the overflow count for interrupts
TAU_MEMDBG_ALLOC_MIN/MAX	0	Byte size minimum and maximum subject to bounds checking (used with TAU_MEMDBG_PROTECT_*)
TAU_MEMDBG_OVERHEAD	0	Specifies the number of bytes for TAU's memory overhead for memory debugging.
TAU_MEMDBG_PROTECT_BELOW/ ABOVE	0	Setting to 1 enables tracking runtime bounds checking below or above the array bounds (requires –optMemDbg while building or tau_exec –memory)
TAU_MEMDBG_ZERO_MALLOC	0	Setting to 1 enables tracking zero byte allocations as invalid memory allocations.
TAU_MEMDBG_PROTECT_FREE	0	Setting to 1 detects invalid accesses to deallocated memory that should not be referenced until it is reallocated (requires –optMemDbg or tau_exec –memory)
TAU_MEMDBG_ATTEMPT_CONTINUE	0	Setting to 1 allows TAU to record and continue execution when a memory error occurs at runtime.
TAU_MEMDBG_FILL_GAP	Undefined	Initial value for gap bytes
TAU_MEMDBG_ALINGMENT	Sizeof(int)	Byte alignment for memory allocations
TAU_EVENT_THRESHOLD	0.5	Define a threshold value (e.g., .25 is 25%) to trigger marker events for min/ max

Runtime Environment Variables

Environment Variable	Default	Description
TAU_OMPT_SUPPORT_LEVEL	basic	Setting to “full” improves resolution of OMPT TR6 regions on threads 1.. N-1. Also, “low overhead” option is available.
TAU_OMPT_RESOLVE_ADDRESS_EAGERLY	0	Setting to 1 is necessary for event based sampling to resolve addresses with OMPT TR6 (-ompt=download-tr6)