TAU Performance System®

Sameer Shende

ParaTools, Inc. and University of Oregon

Wednesday, May 20, 2015, 10am – 10:30am

Argonne Leadership Computing Facility

Building 240, Room 1416

Argonne National Laboratory

sameer@paratools.com

Download slides from:

http://tau.uoregon.edu/tau_mbc15.pptx

cetus.alcf.anl.gov:/soft/perftools/tau/ppt/tau_mbc15.pdf

ParaTools

UNIVERSITY OF OREGON

Outline

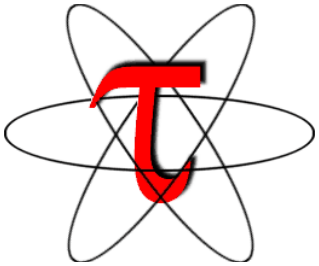
	Slide #
9:00 – 10:30: Introduction to TAU	**7**
• Instrumentation Options	
• Demonstration	
• Short hands-on session	
10:30 – 10:45: break	
10:45 – 12:00: Instrumentation and Measurement	**27**
• PAPI hardware Performance Counters	
• Measurement Options	
12:00 – 1:30pm: Lunch	
1:30 – 3:30pm: Runtime Systems	**98**
• I/O, OpenMP instrumentation, Accelerators (Xeon Phi)	
• Memory Instrumentation	
• Hands-On session	
3:30 – 3:45pm: break	
3:45 – 5:00pm: Analysis Tools	**114**
• TAUdb, ParaProf, PerfExplorer, Vampir	
• Hands-On Session	
Day 2 (Wed. Feb. 11th) : 9am – 5pm	
• Individual consulting	

Introduction to TAU

- Instrumentation Options
- Demonstration
- Short hands-on session

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?

Examples

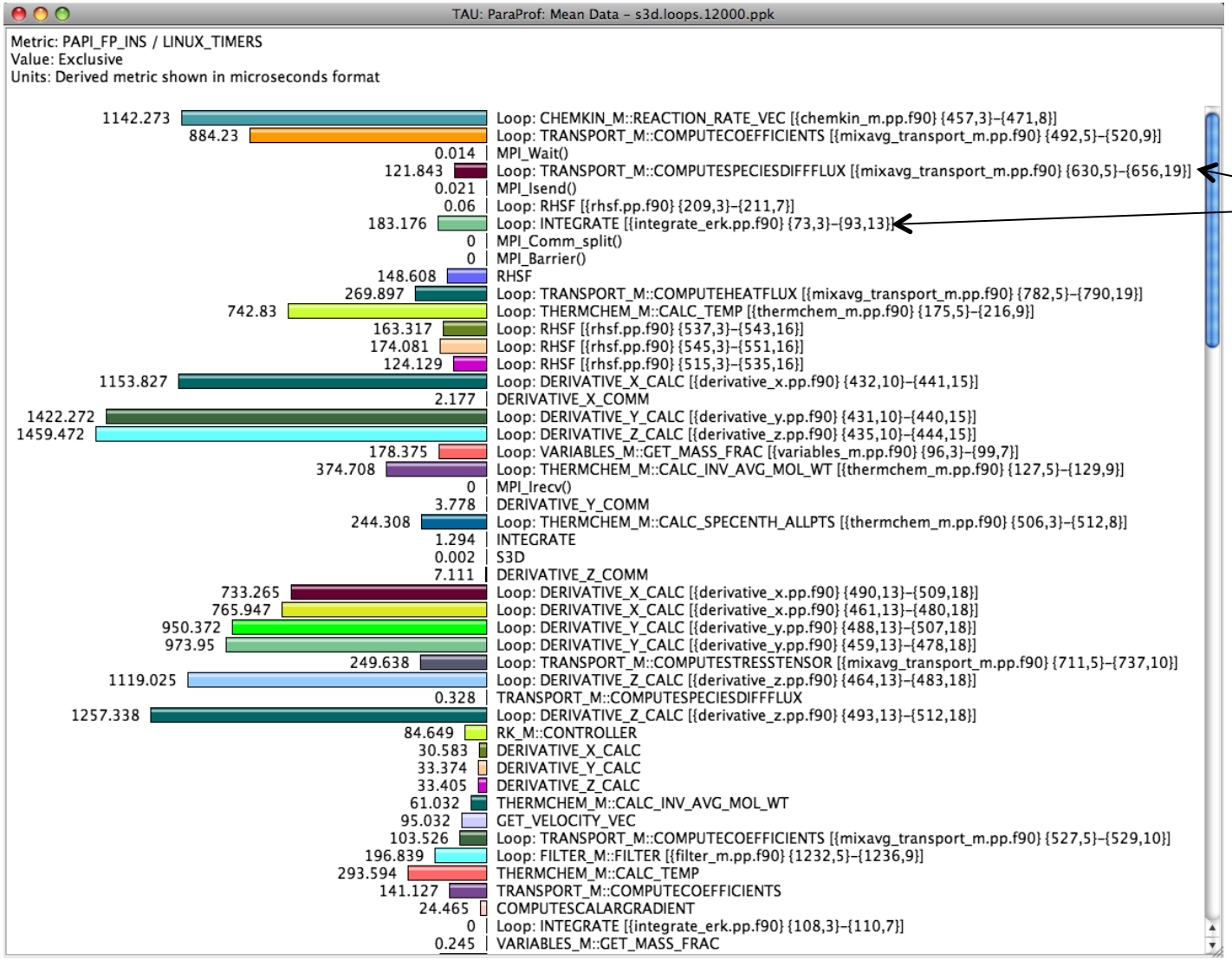
Workshop Examples

```
On Cetus and Mira(IBM BG/Q)
% tar zxf /soft/perftools/tau/workshop.tgz
% soft add +tau-latest
  to see how to load TAU and access the workshop tarball
% cd workshop
% cat README
And try the examples.

On Mira:
% qsub -c16 -n 128 -t 10 -q <Mira Boot Camp queue>./a.out
```

New Features in TAU

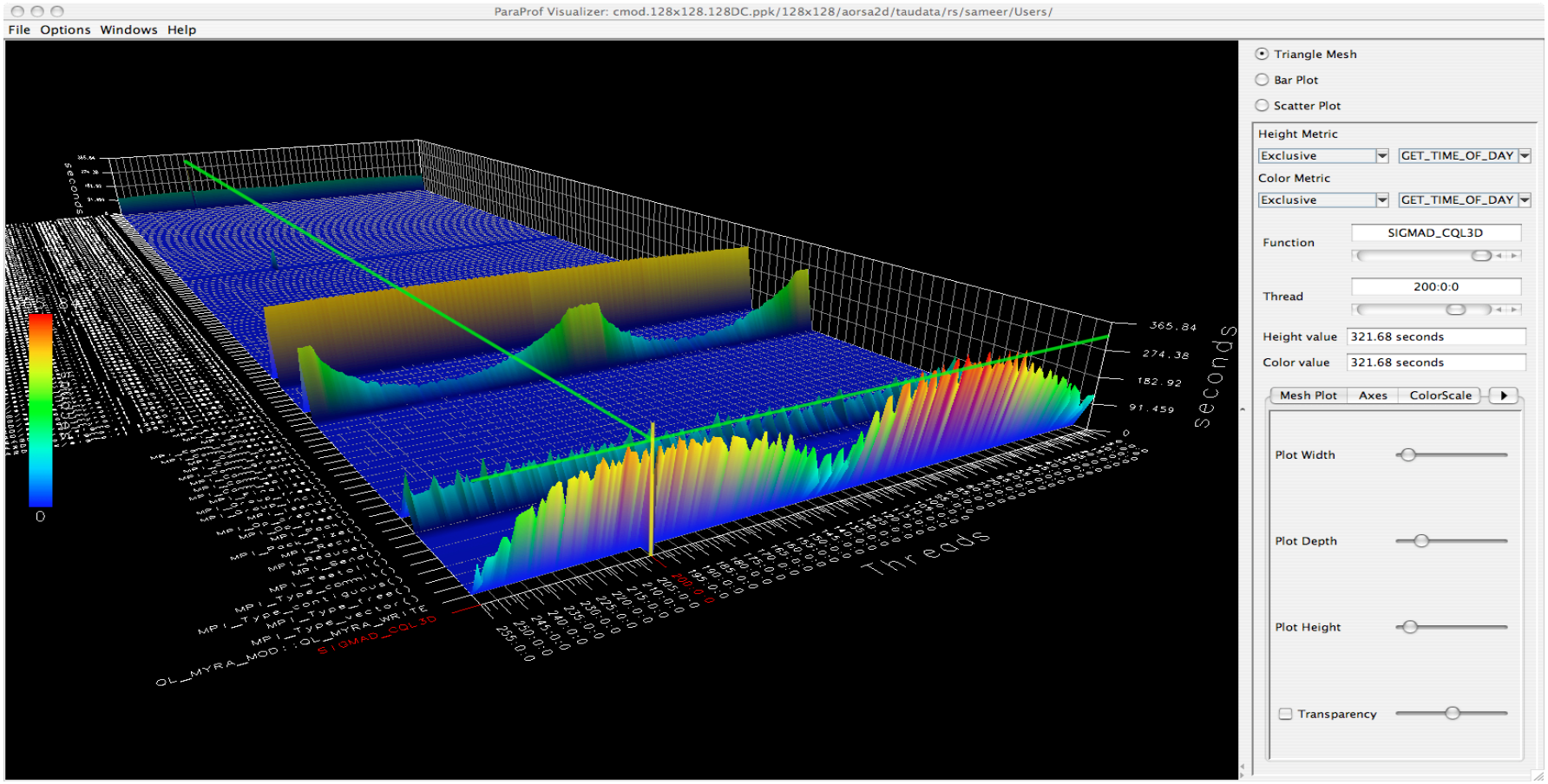
- *tau_exec*: A tool to simplify TAU's usage
- PDT: New parsers from ROSE Compiler
- Power profiling in TAU, Energy profiling on Cray XC40
- Memory footprint tracking on Linux
- Support for accelerators: OpenACC, CUDA, OpenCL, Intel Xeon Phi Co-processor
- Event based sampling
- OpenMP instrumentation using OpenMP Tools Interface with Intel compilers (OMPT)
- Support for Intel TBB, CILK, and MPC [paratools.com/mpc]
- Binary rewriting with MAQAO (Beta)
- ParaProf 3D topology displays
- TAUdb using H2 database

Identifying Potential Bottlenecks



low mflops in loops?

ParaProf 3D Profile Browser



What Can TAU Do?

- **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
- **Multi-language debugging**
  - Identify the source location of a crash by unwinding the system callstack
  - Identify memory errors (off-by-one, etc.)
- Profiling and tracing can measure **time** as well as **hardware performance counters** (cache misses, instructions) from your CPU
- TAU can **automatically instrument** your source code using a package called PDT for routines, loops, I/O, memory, phases, etc.
- TAU runs on **all HPC platforms** and it is free (BSD style license)
- TAU includes instrumentation, measurement and analysis tools

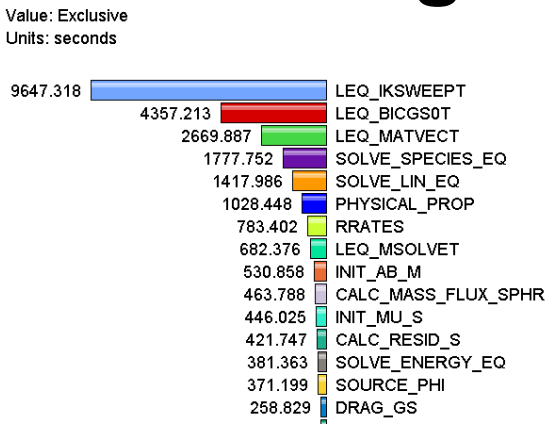
What does TAU support?

C/C++
Fortran
pthreads
Intel
GNU
MPC
CUDA
UPC
OpenACC
Intel MIC
PGI
LLVM
Linux
BlueGene
NVIDIA Kepler
OpenCL
GPI
Python
Java
MPI
OpenMP
Cray
Sun
Windows
AIX
Fujitsu
ARM
OS X

Insert yours here

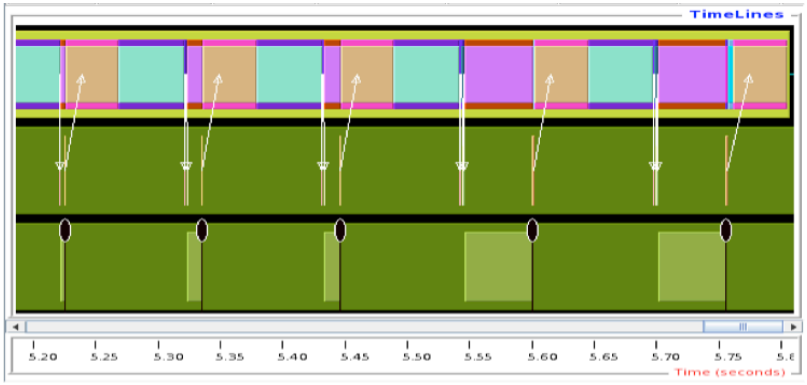
Profiling and Tracing

Profiling



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

Tracing

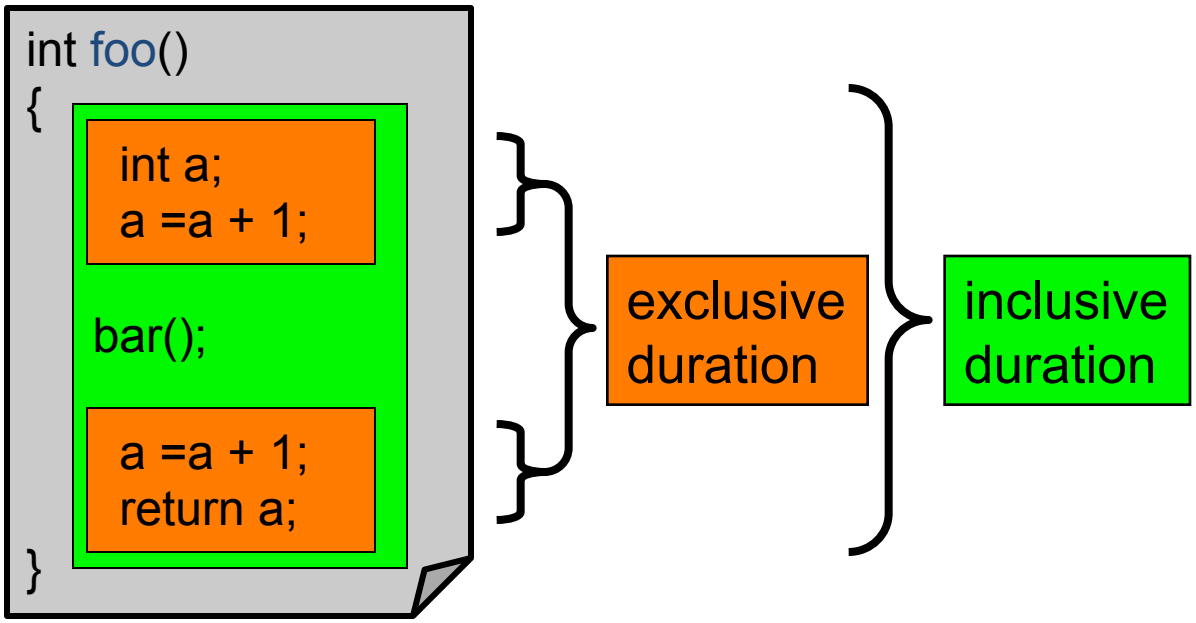


- **Tracing** shows you **when** the events take place on a timeline

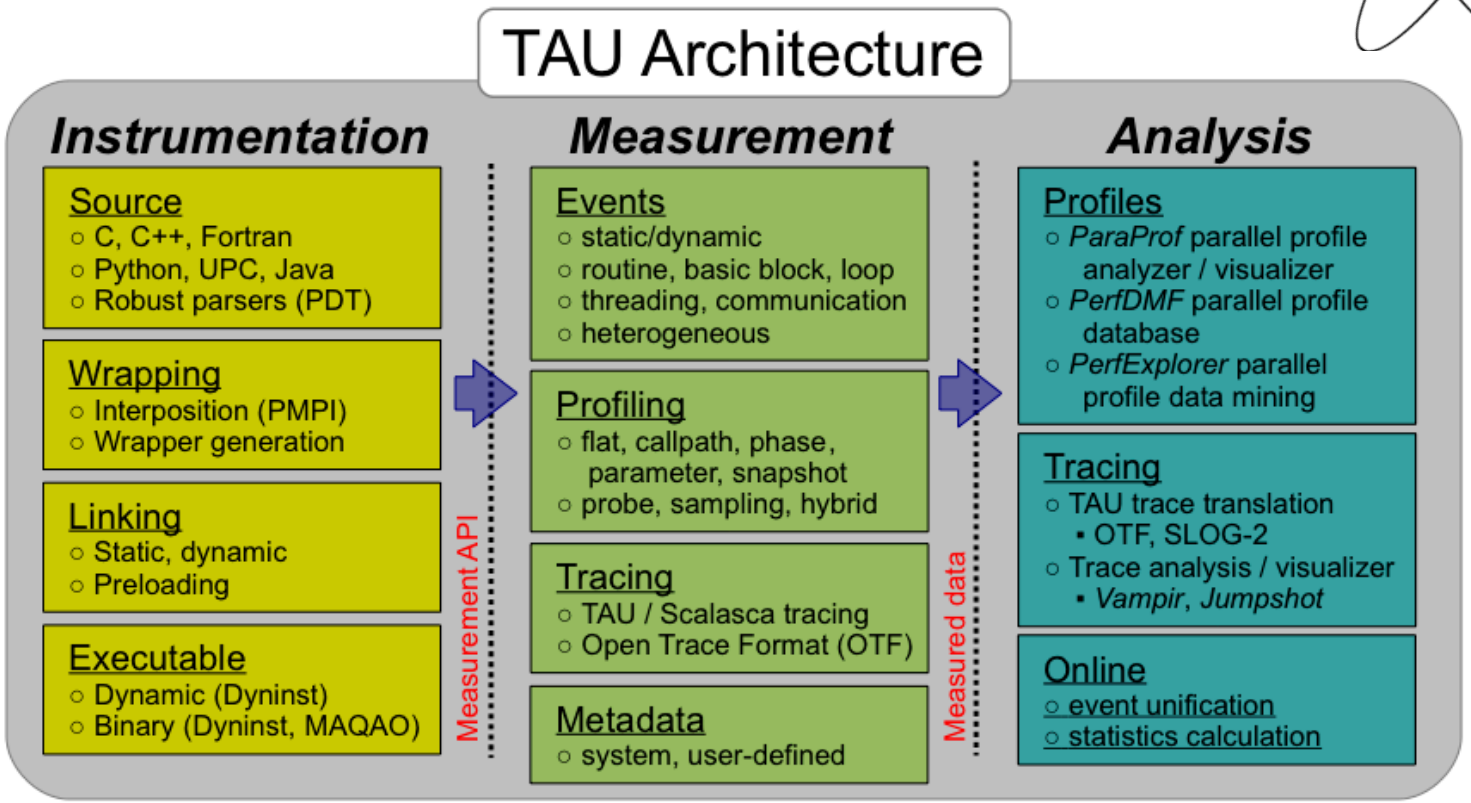
- Metrics can be time or hardware performance counters (cache misses, instructions)
- TAU can automatically instrument your source code using a package called PDT for routines, loops, I/O, memory, phases, etc.

Inclusive vs. Exclusive Measurements

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



TAU Architecture and Workflow



TAU Architecture and Workflow

Instrumentation: **Add probes to perform measurements**

- Source code instrumentation using pre-processors and compiler scripts
- Wrapping external libraries (I/O, MPI, Memory, CUDA, OpenCL, pthread)
- Rewriting the binary executable

Measurement: **Profiling or tracing using various metrics**

- Direct instrumentation (Interval events measure exclusive or inclusive duration)
- Indirect instrumentation (Sampling measures statement level contribution)
- Throttling and runtime control of low-level events that execute frequently
- Per-thread storage of performance data
- Interface with external packages (e.g. PAPI hw performance counter library)

Analysis: **Visualization of profiles and traces**

- 3D visualization of profile data in paraprof or perfexplorer tools
- Trace conversion & display in external visualizers (Vampir, Jumpshot, ParaVer)

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

Direct Observation Events

Interval events (begin/end events)

- Measures exclusive & inclusive durations between events
- Metrics monotonically increase
- Example: Wall-clock timer

Atomic events (trigger with data value)

- Used to capture performance data state
- Shows extent of variation of triggered values (min/max/mean)
- Example: heap memory consumed at a particular point

Code events

- Routines, classes, templates
- Statement-level blocks, loops
- Example: for-loop begin/end

Instrumentation and Measurement

- PAPI Hardware Performance Counters
- Instrumentation Options
- Short hands-on session

Direct Instrumentation Options in TAU

Source Code Instrumentation

- Automatic instrumentation using pre-processor based on static analysis of source code (PDT), creating an instrumented copy
- Compiler generates instrumented object code
- Manual instrumentation

Library Level Instrumentation

- Statically or dynamically linked wrapper libraries
  - MPI, I/O, memory, etc.
- Wrapping external libraries where source is not available

Runtime pre-loading and interception of library calls

Binary Code instrumentation

- Rewrite the binary, runtime instrumentation

Virtual Machine, Interpreter, OS level instrumentation

Examples

Workshop Examples

```
On Cetus and Mira(IBM BG/Q)
% tar zxf /soft/perftools/tau/workshop.tgz
% soft add +tau-latest
  to see how to load TAU and access the workshop tarball
% cd workshop
% cat README
And try the examples.

On Mira:
% qsub -c16 -n 128 -t 10 -q <queue>./a.out
```

Using TAU

TAU supports several measurement and thread options

Phase profiling, profiling with hardware counters, MPI library, CUDA…

Each measurement configuration of TAU corresponds to a unique stub makefile 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:

% soft add +tau-latest

% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt

% export TAU_OPTIONS= '-optVerbose …' (see tau_compiler.sh)

% export PATH=$TAUDIR/x86_64/bin:$PATH

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)

Choosing TAU_MAKEFILE

```
% ls $TAU/Makefile.*
```

Makefile.tau-bgqtimers-mpi-pdt

Makefile.tau-bgqtimers-papi-mpi-pdt

Makefile.tau-bgqtimers-papi-mpi-pdt-openmp-opari

Makefile.tau-bgqtimers-gnu-papi-mpi-pdt-openmp-opari

Makefile.tau-bgqtimers-papi-mpi-pthread-pdt

Makefile.tau-bgqtimers-pdt

Makefile.tau-papi-mpi-pdt-scorep

Makefile.tau-clang-papi-mpi-pdt-scorep

Makefile.tau-bgqtimers-clang-papi-mpi-pdt

…

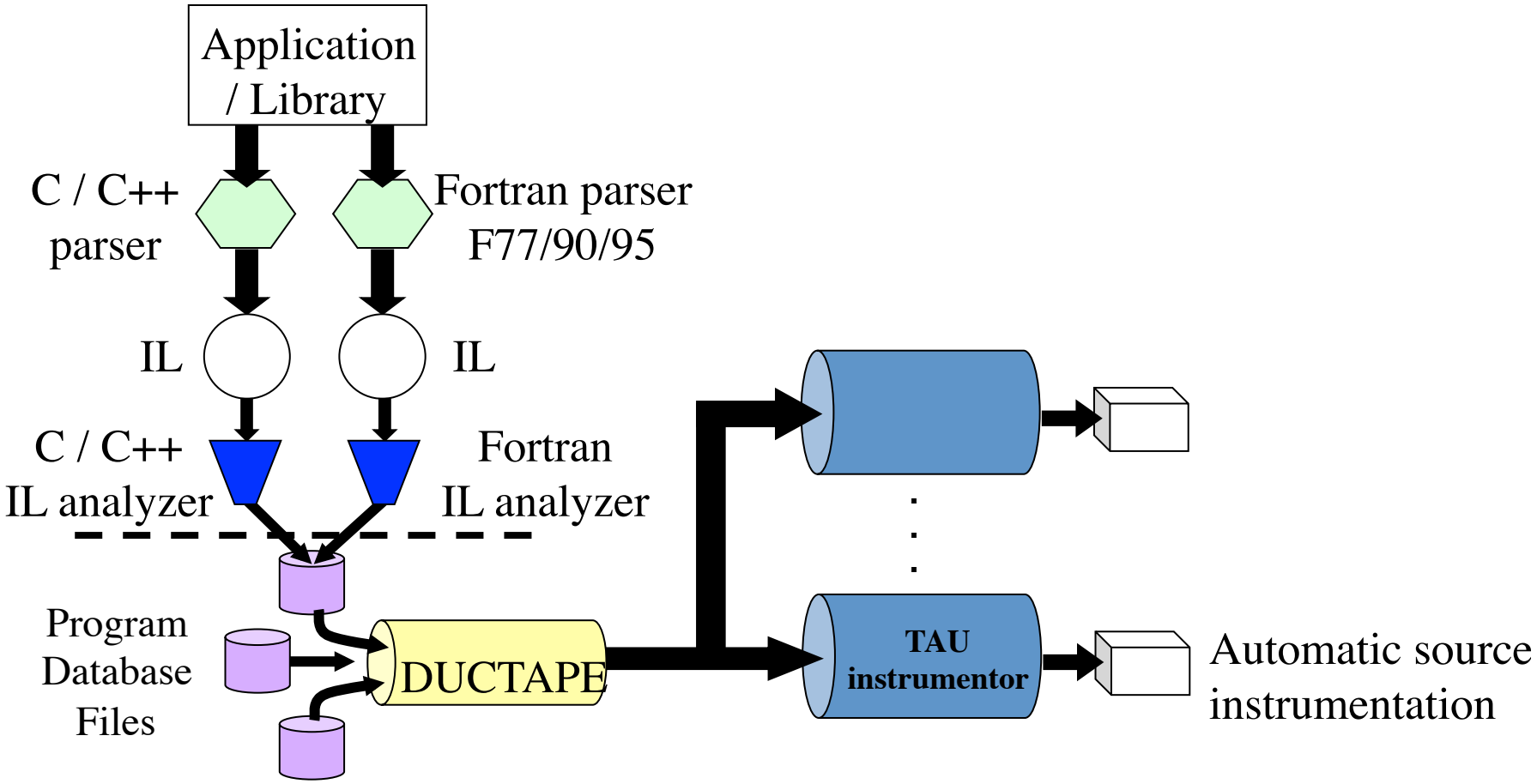
For an MPI+F90 application with Intel MPI, you may choose

Makefile.tau-bgqtimers-papi-mpi-pdt

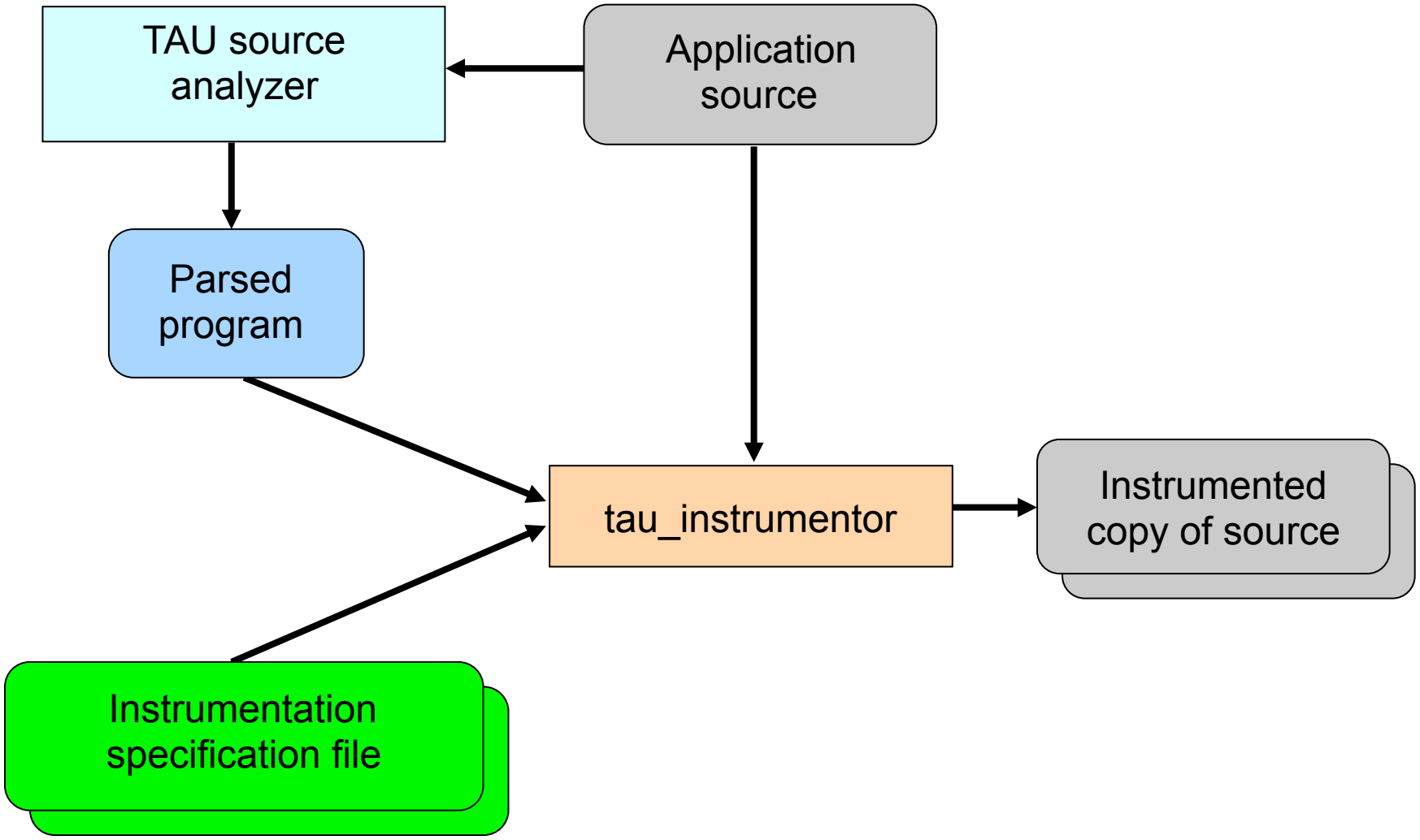
- Supports MPI instrumentation & PDT for automatic source instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% tau_f90.sh matrix.f90 -o matrix
% qsub -n 32 -q R.TAU-workshop -A <account> -t 10 ./matrix
% paraprof
```

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



Automatic Source Instrumentation using PDT



Automatic Instrumentation

- **Use TAU's compiler wrappers**
  - Simply replace `CXX` with `tau_cxx.sh, etc.`
  - Automatically instruments source code, links with TAU libraries.
- **Use `tau_cc.sh` for C, `tau_f90.sh` for Fortran, `tau_upc.sh` for UPC, etc.**

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

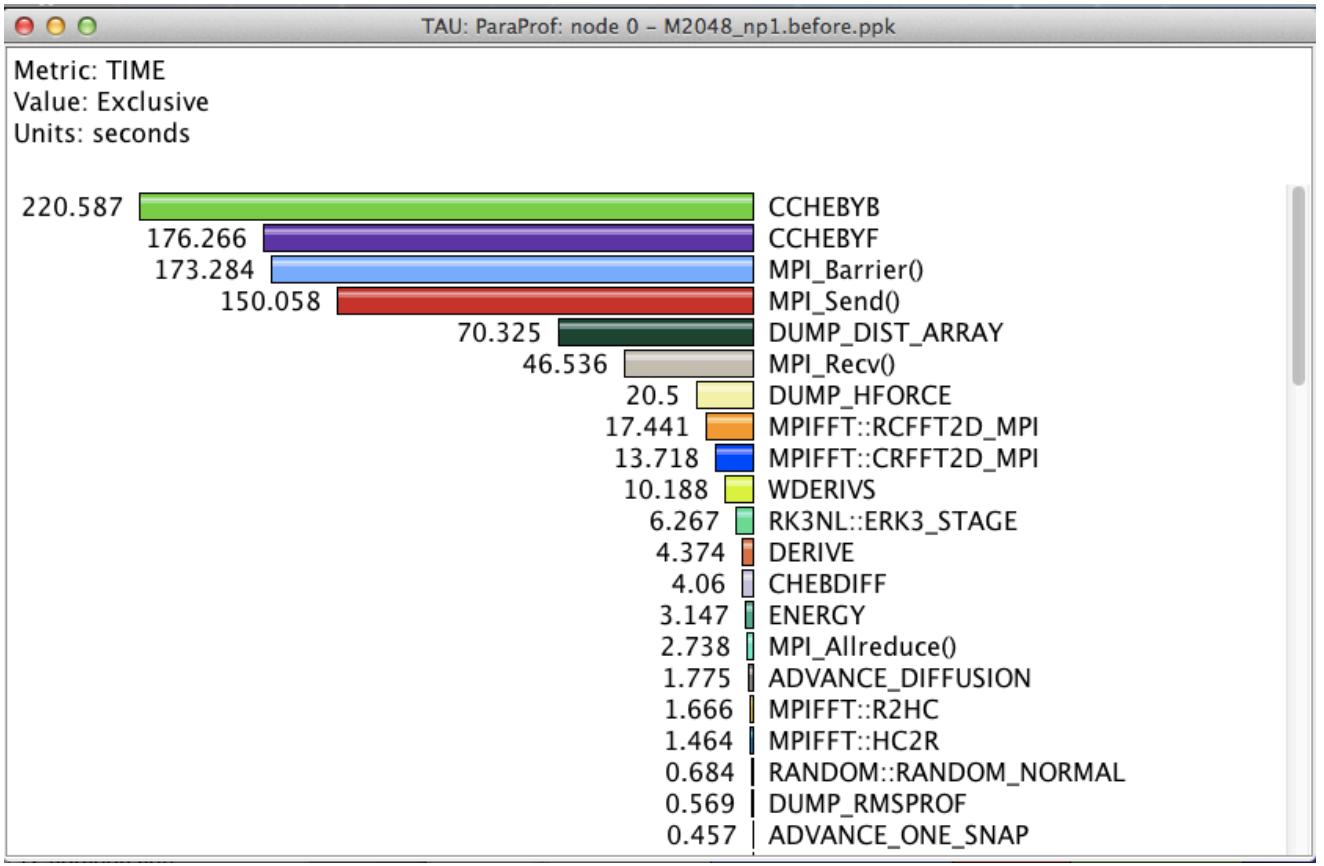
```
CXX = tau_cxx.sh
F90 = tau_f90.sh
CXXFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

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

Generating a flat profile with MPI

```
% soft add +tau-latest
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% make F90=tau_f90.sh
Or
% tau_f90.sh matmult.f90
% qsub -n 32 -q R.TAU-workshop -A <account> -t 10 ./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.
```

Routine Level Profile

How much time is spent in each application routine?



Profiling with multiple counters

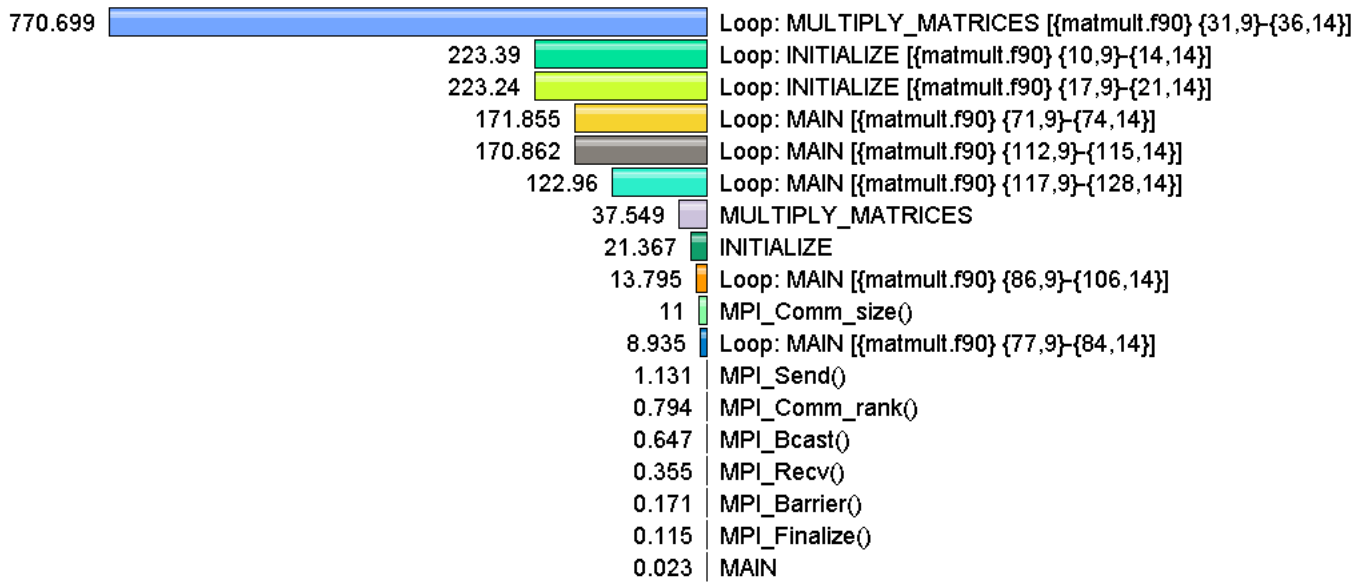
```
% soft add +tau-latest
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION
% make F90=tau_f90.sh

% export TAU_METRICS=TIME,PAPI_FP_INS,PAPI_L1_DCM
On MIC:
% export TAU_METRICS=TIME,PAPI_NATIVE_VPU_ELEMENTS_ACTIVE,
                      PAPI_NATIVE_VPU_INSTRUCTIONS_EXECUTED
% qsub -n 32 -q R.TAU-workshop -A <account> -t 10 ./matmult
% paraprof --pack app.ppk
  Move the app.ppk file to your desktop.
% paraprof app.ppk  (Choose Options -> Show Derived Panel -> Click
PAPI_FP_INS, Click "/",  Click TIME, Apply, Choose new metric by
double clicking.)
```

Computing FLOPS per loop

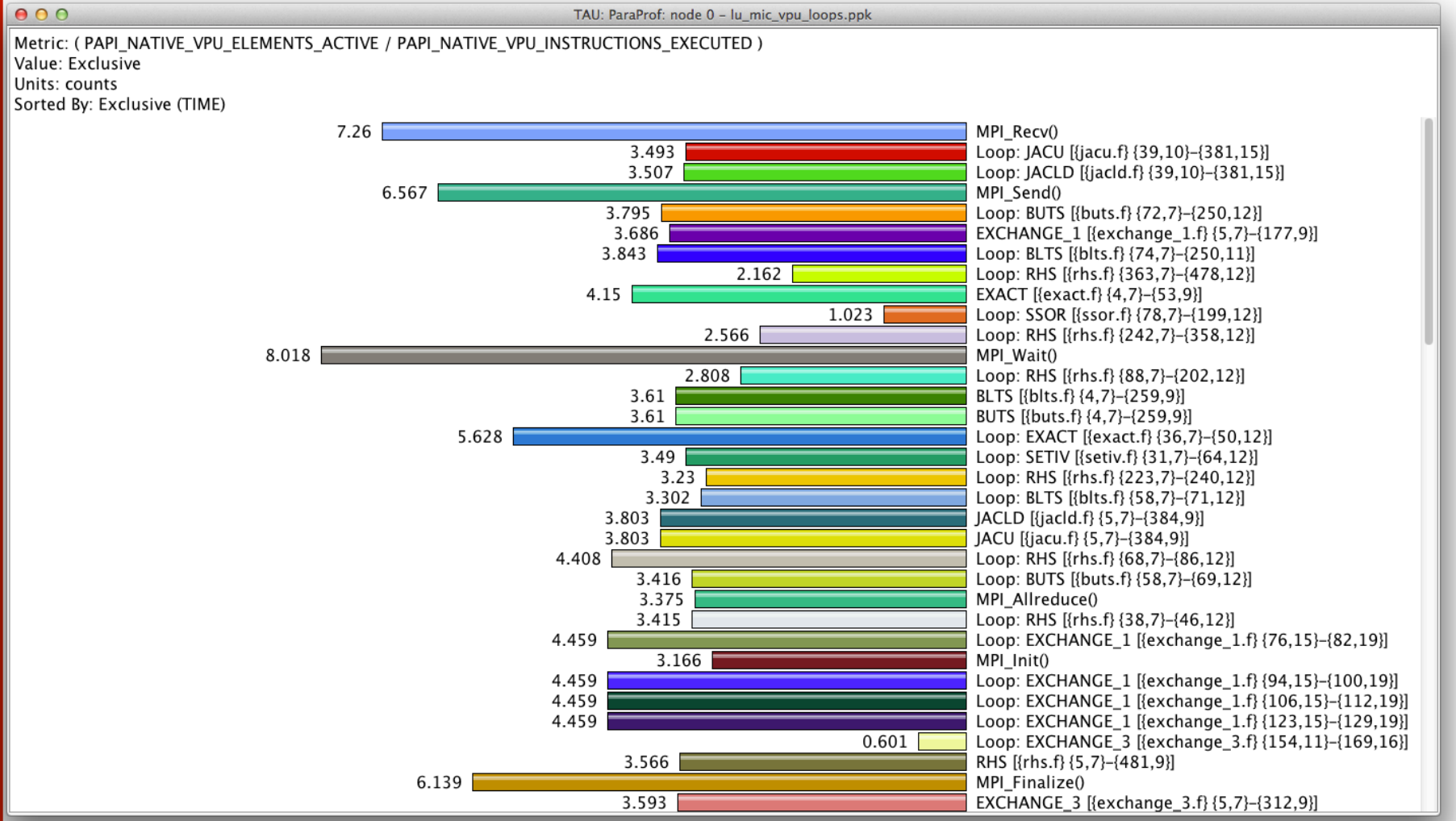
Goal: What is the execution rate of my loops in MFLOPS?

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



ParaProf's Derived Metric Window: MIC




```
% export TAU_MAKEFILE=$TAUROOT/mic_linux/lib/Makefile.tau-bgqtimers-papi-mpi-pdt
% export TAU_METRICS=TIME,
        PAPI_NATIVE_VPU_ELEMENTS_ACTIVE,PAPI_NATIVE_VPU_INSTRUCTIONS_EXECUTED
```

Steps to using TAU on BGQ

1. **Choose instrumentation and runtime by selecting TAU_MAKEFILE**
2. **Compile the program**
3. **Run it with**

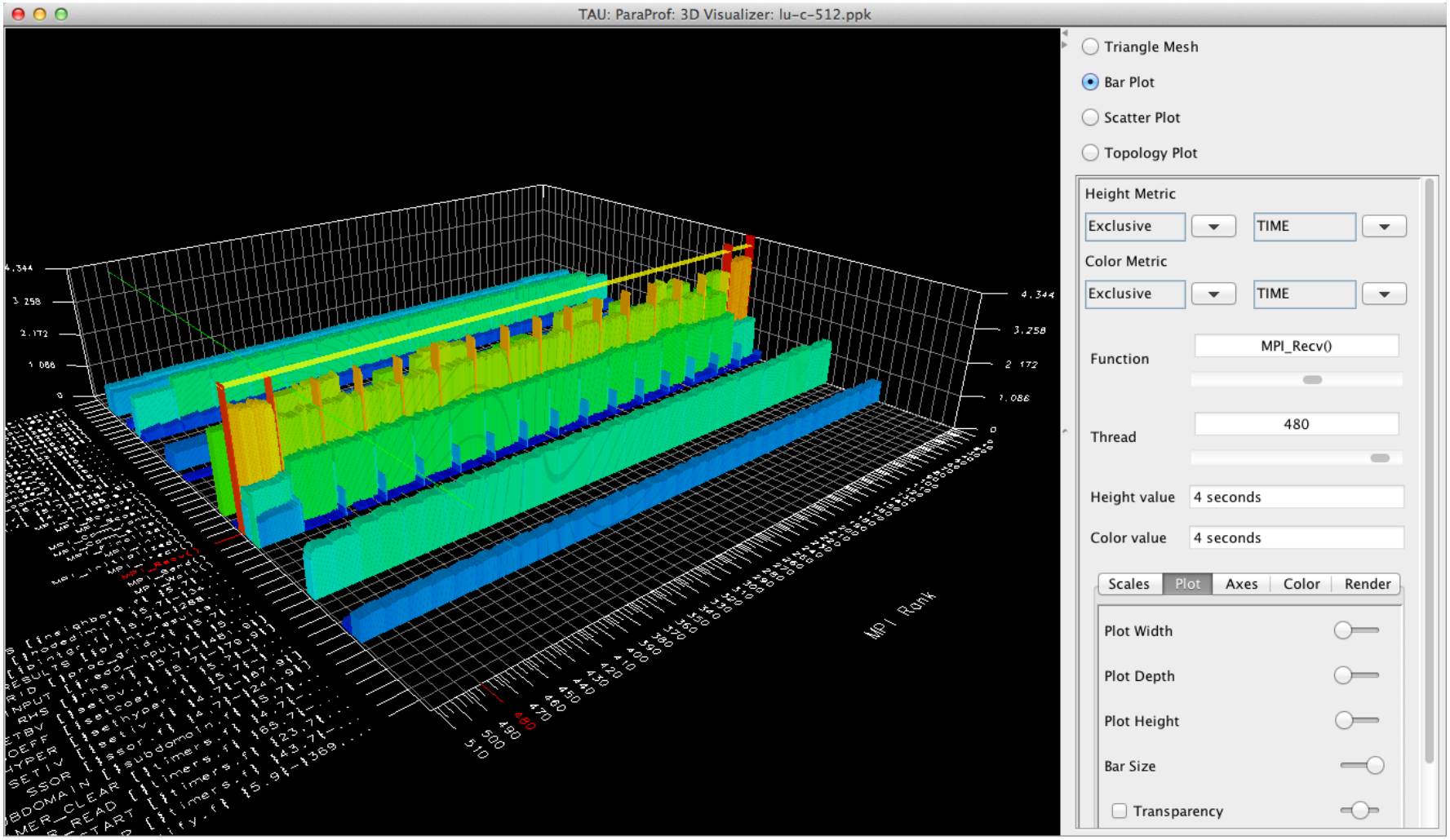
   export TAU_PROFILE_FORMAT=merged

   export TAU_METRICS=BGQ_TIMERS
4. **ParaProf can generate selective instrumentation file to eliminate throttled lightweight routines**
5. **Recompile the application using**

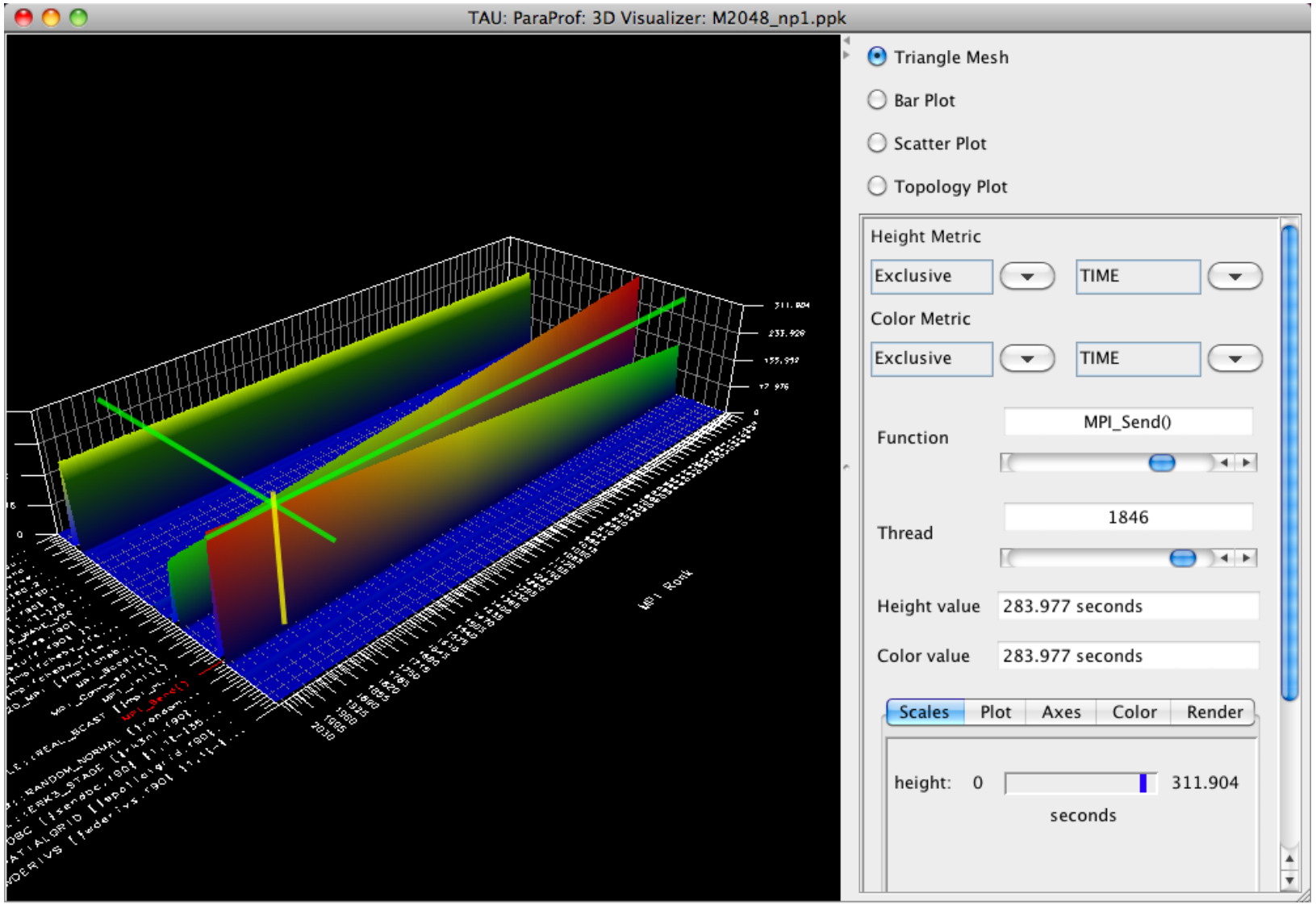
   export TAU_OPTIONS='–optTauSelectFile=select.tau –optVerbose'
6. **Run it with event-based sampling enabled**

   export TAU_OPTIONS='–optTauSelectFile=select.tau –optVerbose'
7. **Generate event traces using TAU_MAKEFILE set to a –scorep configuration and SCORE_ENABLE_TRACING=1**
8. **View traces.otf2 with Vampir**
9. **Choose other runtime/compile time options for other metrics**

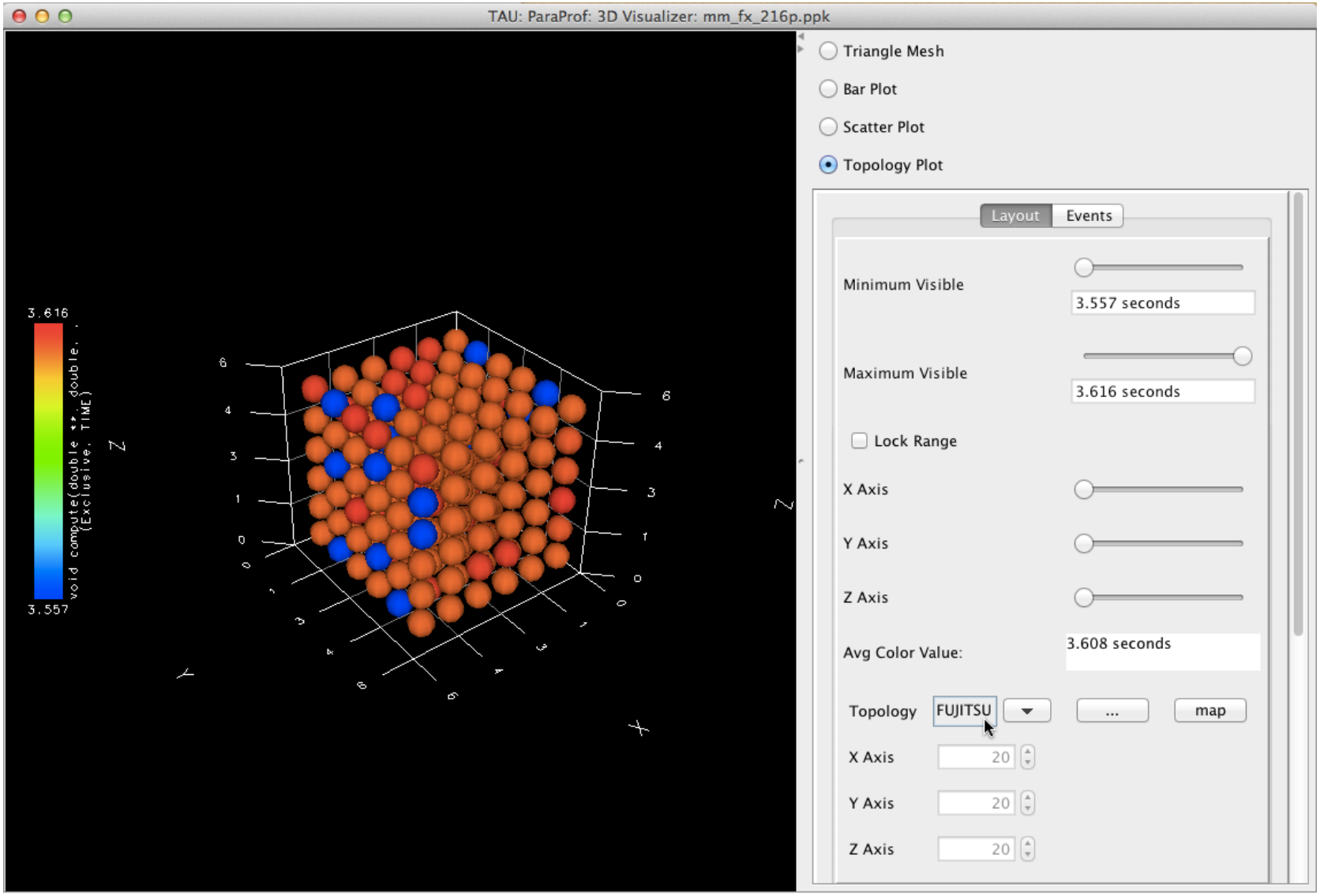
ParaProf 3D Profile Browser



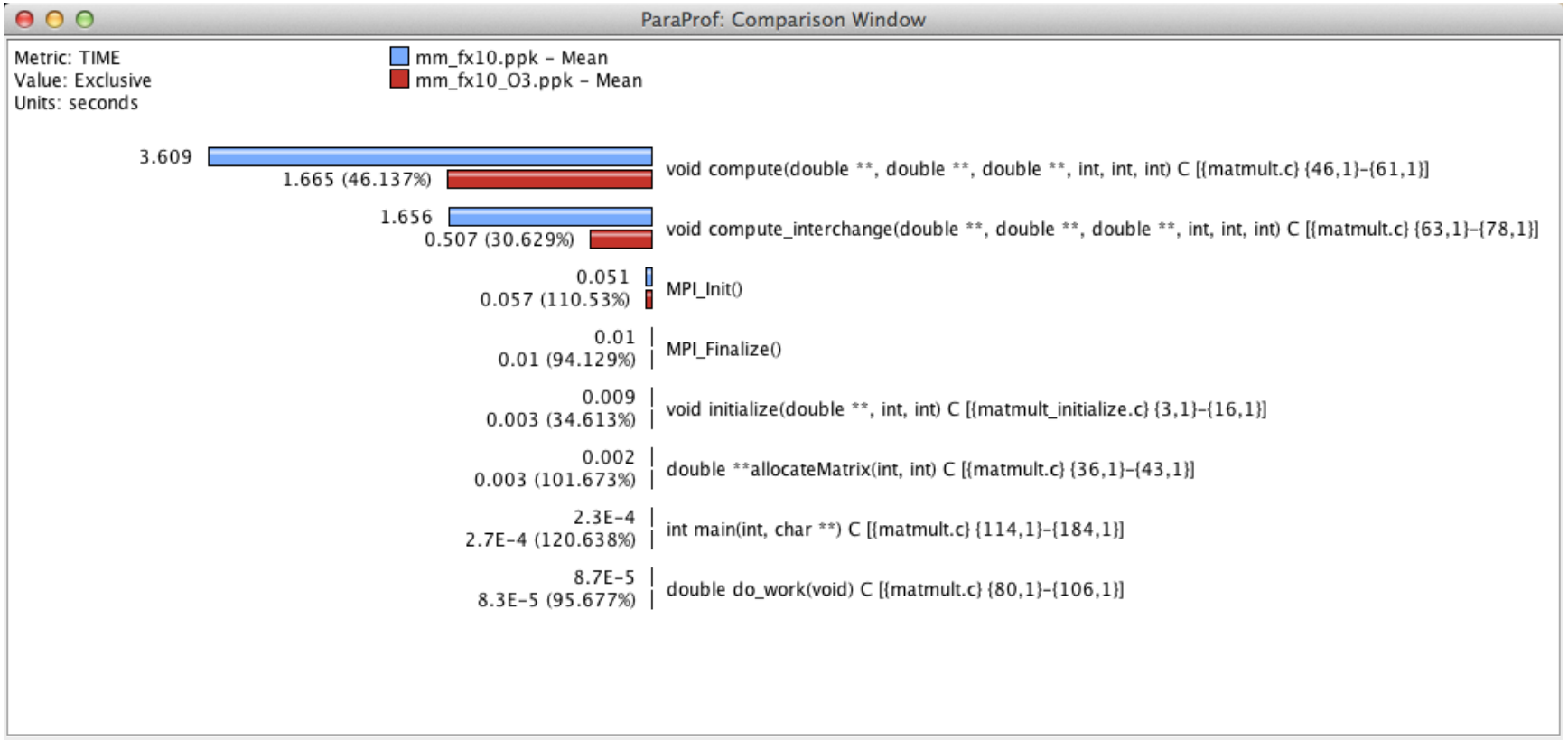
ParaProf


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

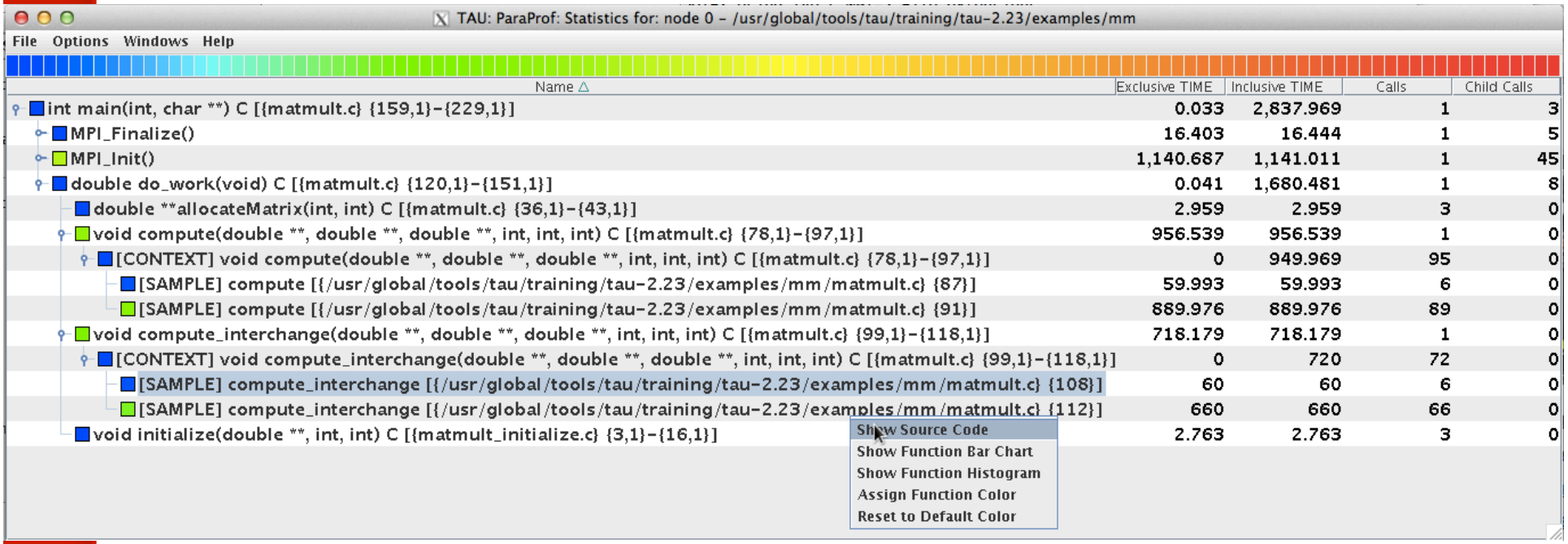
ParaProf 3D Topology Display



TAU's ParaProf Comparison Window



Event Based Sampling in TAU




```
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% make CC=tau_cc.sh CXX=tau_cxx.sh
% qsub –n 256 …   --env TAU_SAMPLING=1 ./a.out
% paraprof
```

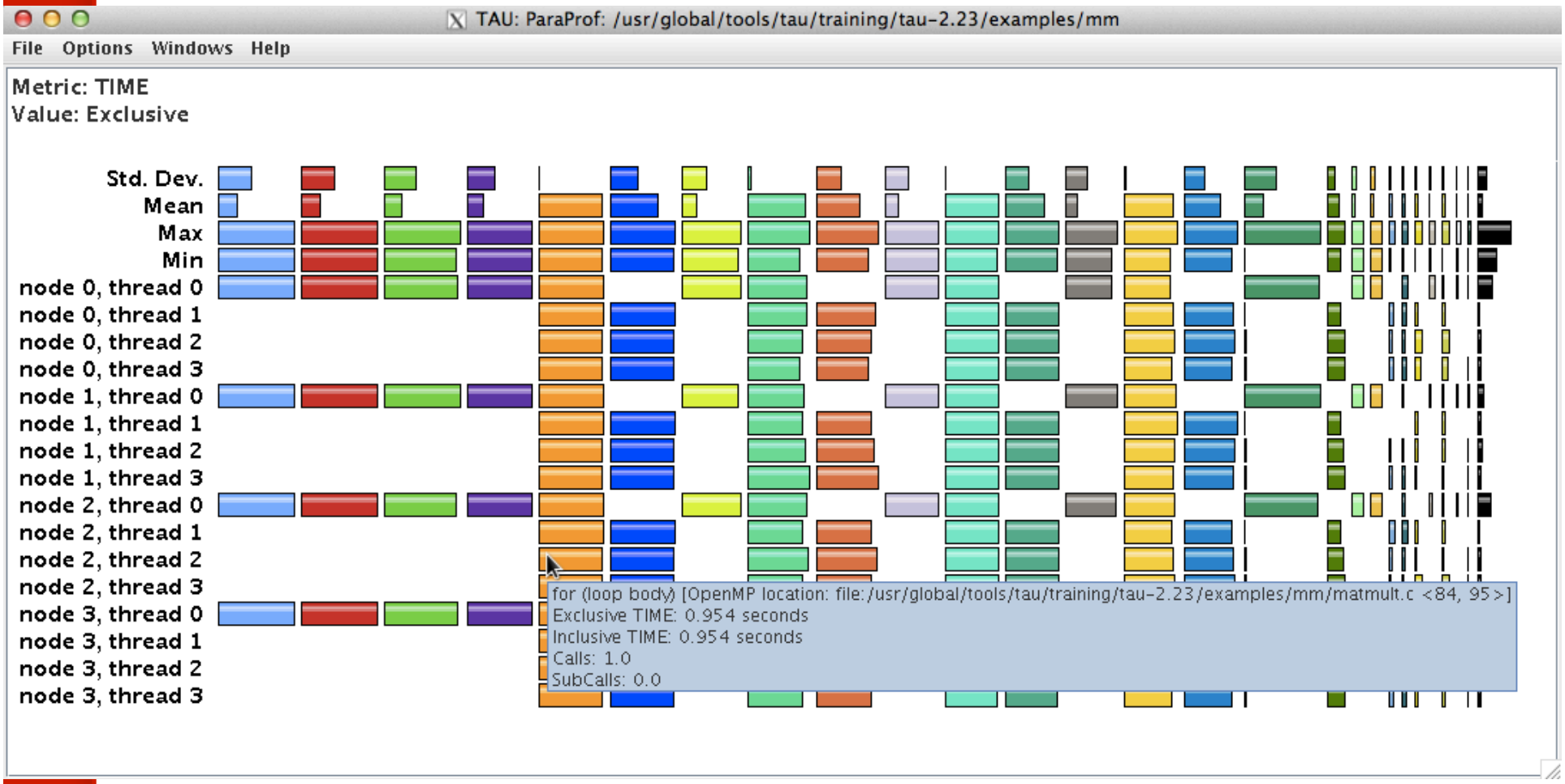
TAU_SAMPLING=1 on IBM BGQ

TAU: ParaProf: Statistics for: node 0, thread 0 - tauprofile.xml

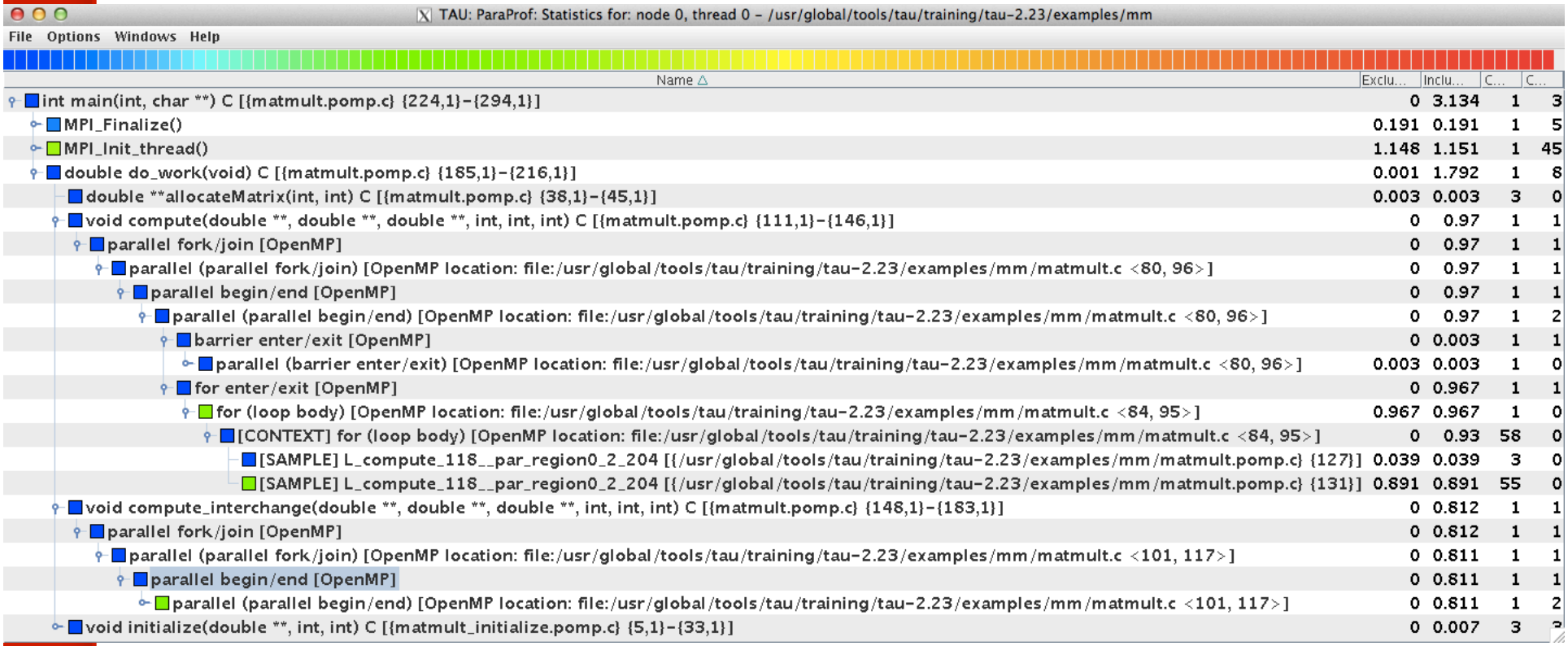
BGQ_TIMERS

Name	Exclusive BGQ_TIMERS	Inclusive BGQ_TIMERS	Exclusive PAPI_FMA_...	Inclusive PAPI_FMA_INS	Exclusive PAPI_FP_INS	Inclusive PAPI_FP_INS	Calls	Child Calls
LOCAL_GRAD3_T [{cg.pp.pomp.f} {280,7}-{304,9}]	17.66	17.66	2,831,360,000	2,831,360,000	3,492,959,736	3,492,959,736	204,800	0
[CONTEXT] LOCAL_GRAD3_T [{cg.pp.pomp.f} {280,7}-{304,9}]	0	16.35	0	2,097,153,789	0	2,690,896,651	269	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f}]	6.65	6.65	816,385,840	816,385,840	1,013,824,782	1,013,824,782	121	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {375}]	5.4	5.4	658,609,545	658,609,545	818,098,032	818,098,032	97	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {374}]	0.9	0.9	116,290,160	116,290,160	143,931,173	143,931,173	17	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {387}]	0.1	0.1	8,455,488	8,455,488	11,005,290	11,005,290	2	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {388}]	0.1	0.1	11,696,242	11,696,242	14,643,876	14,643,876	2	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {390}]	0.1	0.1	14,222,261	14,222,261	17,430,202	17,430,202	2	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {368}]	0.05	0.05	7,112,144	7,112,144	8,716,209	8,716,209	1	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f}]	5.95	5.95	800,377,037	800,377,037	1,070,818,351	1,070,818,351	90	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {289}]	4.9	4.9	659,781,065	659,781,065	883,058,544	883,058,544	75	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {288}]	0.6	0.6	82,374,650	82,374,650	109,915,220	109,915,220	9	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {298}]	0.2	0.2	24,704,818	24,704,818	33,062,613	33,062,613	3	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {299}]	0.2	0.2	26,485,272	26,485,272	35,360,198	35,360,198	2	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {282}]	0.05	0.05	7,031,232	7,031,232	9,421,776	9,421,776	1	0
[SAMPLE] add2 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/math.pp.pomp.inst.f}]	3.3	3.3	423,146,529	423,146,529	533,185,484	533,185,484	53	0
[SAMPLE] mxmf2 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f}]	0.2	0.2	23,398,488	23,398,488	30,501,747	30,501,747	2	0
[SAMPLE] UNRESOLVED UNKNOWN	0.2	0.2	26,738,171	26,738,171	33,855,336	33,855,336	2	0
[SAMPLE] local_grad3_t [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f}]	0.05	0.05	7,107,724	7,107,724	8,710,951	8,710,951	1	0
AX_E [{cg.pp.pomp.f} {230,7}-{258,9}]	14.383	46.211	1,509,580,800	7,172,300,800	2,278,339,598	9,109,896,356	204,800	409,600
[CONTEXT] AX_E [{cg.pp.pomp.f} {230,7}-{258,9}]	0	7.3	0	921,710,661	0	1,176,089,527	123	0
[SAMPLE] ax_e [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f}]	6.3	6.3	795,097,800	795,097,800	1,014,549,186	1,014,549,186	107	0
[SAMPLE] ax_e [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f} {296}]	3.45	3.45	430,468,957	430,468,957	549,588,408	549,588,408	57	0
[SAMPLE] ax_e [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f} {298}]	1.55	1.55	206,735,232	206,735,232	262,767,846	262,767,846	26	0
[SAMPLE] ax_e [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f} {297}]	1.25	1.25	150,778,613	150,778,613	193,473,629	193,473,629	23	0
[SAMPLE] ax_e [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f} {294}]	0.05	0.05	7,114,998	7,114,998	8,719,303	8,719,303	1	0
[SAMPLE] UNRESOLVED UNKNOWN	0.75	0.75	95,544,202	95,544,202	121,883,707	121,883,707	11	0
[SAMPLE] local_grad3_t [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f}]	0.15	0.15	18,605,626	18,605,626	23,740,675	23,740,675	3	0
[SAMPLE] local_grad3 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f}]	0.1	0.1	12,463,033	12,463,033	15,915,959	15,915,959	2	0
LOCAL_GRAD3 [{cg.pp.pomp.f} {260,7}-{278,9}]	14.168	14.168	2,831,360,000	2,831,360,000	3,338,597,022	3,338,597,022	204,800	0
[CONTEXT] LOCAL_GRAD3 [{cg.pp.pomp.f} {260,7}-{278,9}]	0	19.15	0	2,476,465,478	0	3,194,345,197	304	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f}]	9.7	9.7	1,159,211,477	1,159,211,477	1,443,834,446	1,443,834,446	152	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {375}]	7.45	7.45	883,564,086	883,564,086	1,101,349,172	1,101,349,172	114	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {374}]	1.45	1.45	187,591,921	187,591,921	231,911,233	231,911,233	26	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {387}]	0.3	0.3	26,825,496	26,825,496	34,521,647	34,521,647	6	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {390}]	0.3	0.3	36,936,757	36,936,757	45,911,471	45,911,471	3	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {388}]	0.15	0.15	21,335,325	21,335,325	26,147,379	26,147,379	2	0
[SAMPLE] mxf12 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f} {373}]	0.05	0.05	2,957,892	2,957,892	3,993,544	3,993,544	1	0
[SAMPLE] mxf9 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f}]	8.55	8.55	1,201,645,814	1,201,645,814	1,600,978,248	1,600,978,248	136	0
[SAMPLE] UNRESOLVED UNKNOWN	0.7	0.7	93,251,700	93,251,700	120,342,856	120,342,856	12	0
[SAMPLE] local_grad3 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/cg.pp.pomp.inst.f}]	0.15	0.15	15,246,925	15,246,925	20,476,422	20,476,422	3	0
[SAMPLE] mxmf2 [{/gpfs/mira-fs0/projects/BGQtools_esp/tau/scratch/workshop/nekbone-2.3.4/test/example1/mxm_std.pp.pomp.inst.f}]	0.05	0.05	7,109,562	7,109,562	8,713,225	8,713,225	1	0
MPI_Finalize()	6.058	6.058	0	0	25,437	25,437	1	0
AXI [{cg.pp.pomp.f} {173,7}-{212,9}]	4.751	52.22	0	7,235,200,000	22,140,492	9,195,065,464	400	214,000
ADD2S2I [{omp.pp.pomp.f} {123,7}-{136,20}]	3.204	3.204	188,697,600	188,697,600	188,721,777	188,721,777	1,200	0
GLSC3I [{omp.pp.pomp.f} {34,7}-{88,20}]	3.173	3.707	189,326,592	189,327,796	378,843,741	379,124,824	1,204	6,020
PROXY_SETUPDS [{prox_dssum.pp.pomp.f} {15,7}-{40,9}]	1.916	2.007	44	44	9,672	131,772	2	144
MPI_Init()	1.803	1.803	0	0	5,491	5,491	1	0
COPYI [{omp.pp.pomp.f} {18,7}-{30,20}]	1.076	1.076	0	0	6,143	6,143	404	0
ADD2S1I [{omp.pp.pomp.f} {106,7}-{119,20}]	1.067	1.067	62,899,200	62,899,200	62,906,644	62,906,644	400	0
MPI_Allreduce()	0.298	0.298	1,212	1,212	27,386	27,386	1,210	0
[CONTEXT] MPI_Allreduce()	0	0.35	0	45,702,325	0	60,154,493	5	0
[SAMPLE] PAMI::Device::MU::InjGroup::advance() [{/bgsys/source/srcV1R2M2.1830/comm/sys/buildtools/pami/components/devices/bgq/mu2/InjGroup.h}]	0.1	0.1	14,365,334	14,365,334	19,134,919	19,134,919	1	0
[SAMPLE] PAMI::Device::MU::InjGroup::advance() [{/bgsys/source/srcV1R2M2.1830/comm/sys/buildtools/pami/components/devices/bgq/mu2/InjGroup.h} {199}]	0.1	0.1	14,365,334	14,365,334	19,134,919	19,134,919	1	0
[SAMPLE] PAMI::Interface::PipeWorkQueue<PAMI::PipeWorkQueue>::configure(char*, unsigned long, unsigned long, PAMI::Type::TypeCode*, PAMI::Type::TypeCode*) [{/	0.1	0.1	14,406,270	14,406,270	19,189,739	19,189,739	1	0
[SAMPLE] PAMI::MultiQueue<2ul, 1ul>::peek_impl() [{/bgsys/source/srcV1R2M2.1830/comm/sys/buildtools/pami/util/queue/MultiQueue.h}]	0.05	0.05	6,531,239	6,531,239	8,735,276	8,735,276	1	0

Mixed MPI and OpenMP Instrumentation



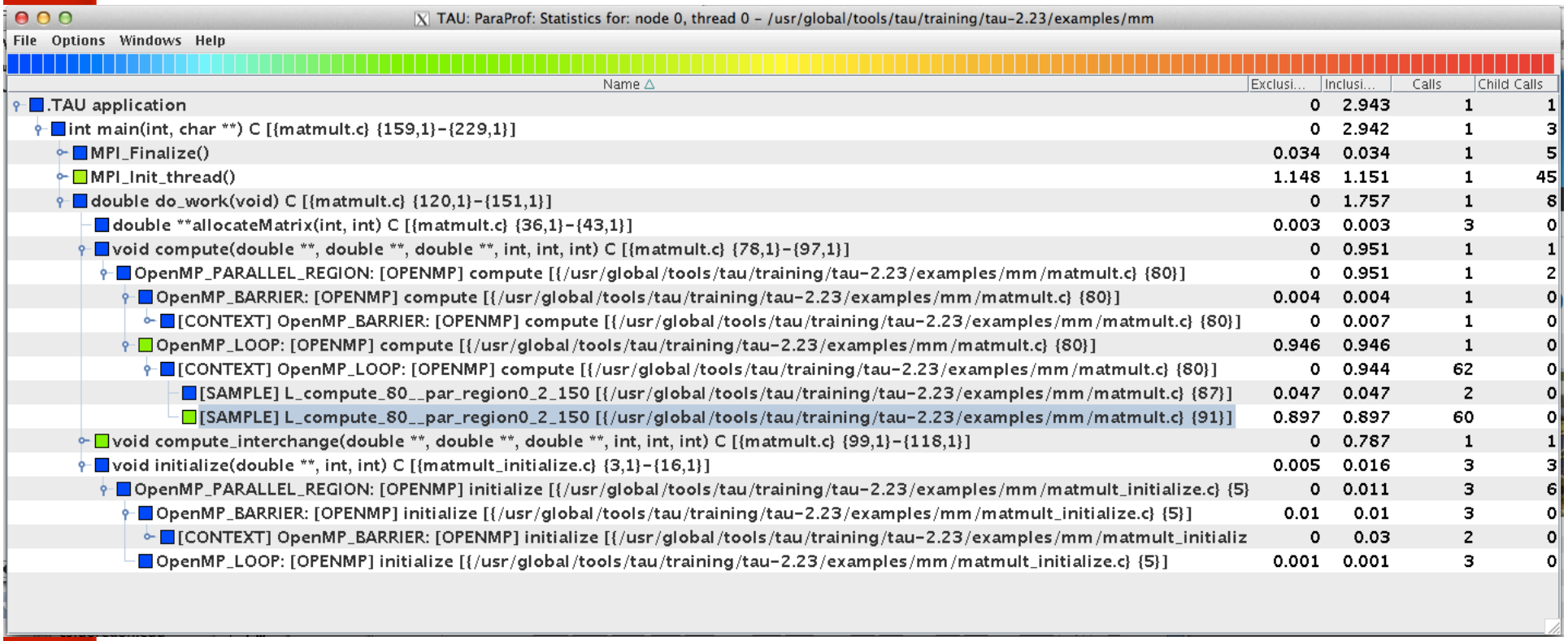
Opari OpenMP Instrumentation, Sampling



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-bgqtimers-papi-mpi-pdt-openmp-opari
% make CC=tau_cc.sh CXX=tau_cxx.sh
% qsub … --env TAU_SAMPLING=1:OMP_NUM_THREADS=4 ./a.out
% paraprof
```

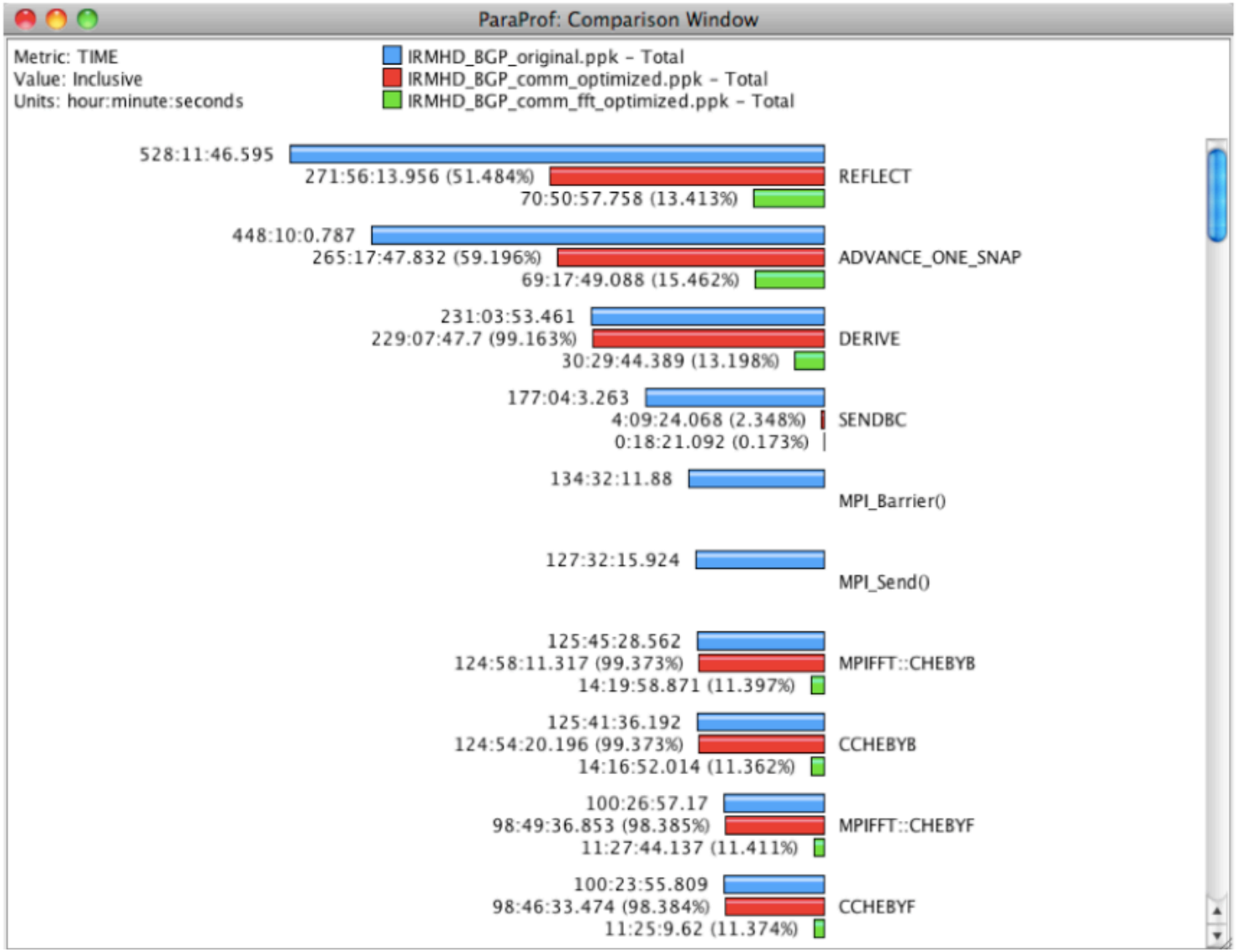
TAU's support for OMPT



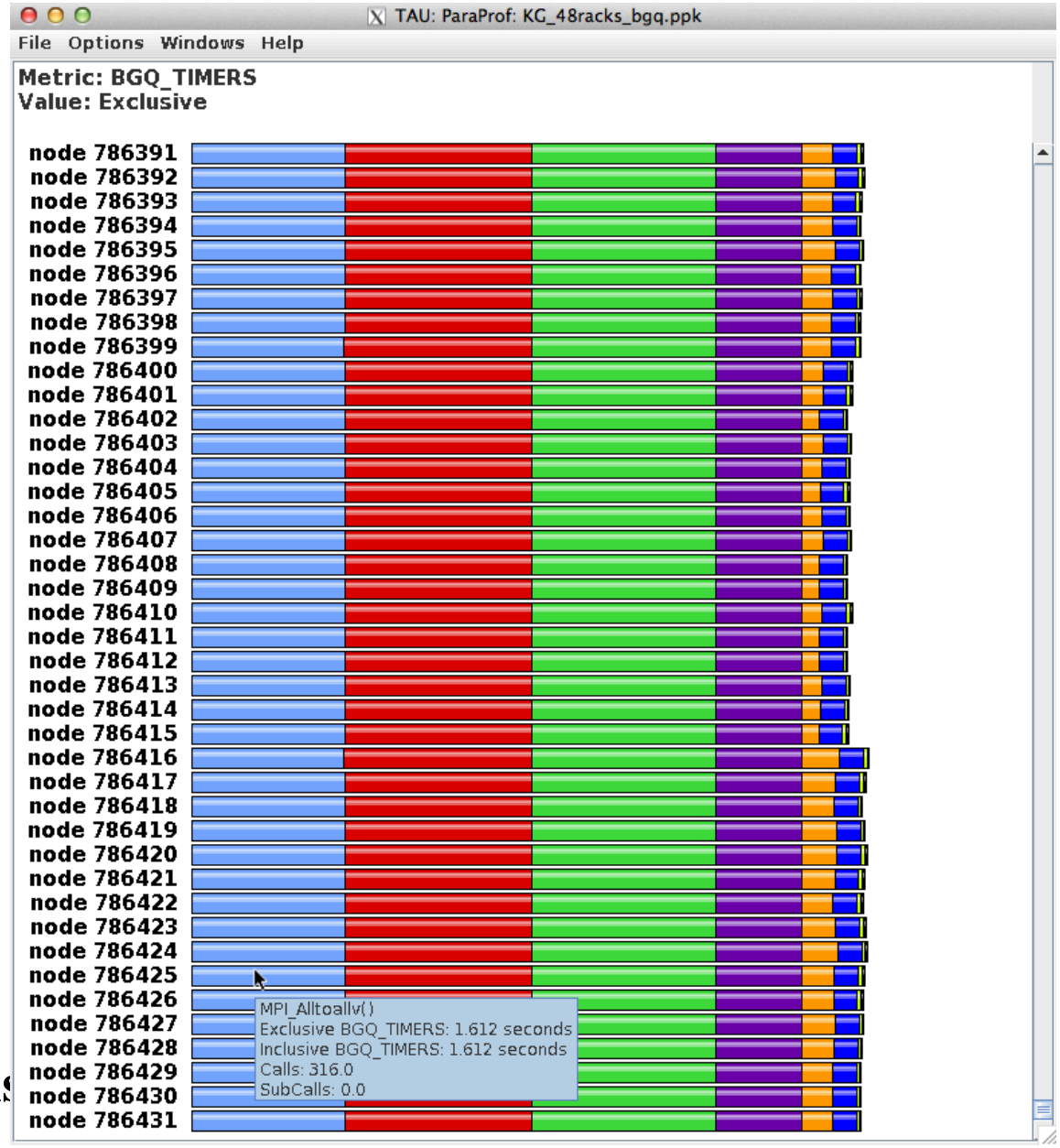
Name	Exclusi...	Inclusi...	Calls	Child Calls
.TAU application	0	2.943	1	1
int main(int, char **) C [{matmult.c} {159,1}-{229,1}]	0	2.942	1	3
MPI_Finalize()	0.034	0.034	1	5
MPI_Init_thread()	1.148	1.151	1	45
double do_work(void) C [{matmult.c} {120,1}-{151,1}]	0	1.757	1	8
double **allocateMatrix(int, int) C [{matmult.c} {36,1}-{43,1}]	0.003	0.003	3	0
void compute(double **, double **, double **, int, int, int) C [{matmult.c} {78,1}-{97,1}]	0	0.951	1	1
OpenMP_PARALLEL_REGION: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0	0.951	1	2
OpenMP_BARRIER: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0.004	0.004	1	0
[CONTEXT] OpenMP_BARRIER: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0	0.007	1	0
OpenMP_LOOP: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0.946	0.946	1	0
[CONTEXT] OpenMP_LOOP: [OPENMP] compute [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {80}]	0	0.944	62	0
[SAMPLE] L_compute_80__par_region0_2_150 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {87}]	0.047	0.047	2	0
[SAMPLE] L_compute_80__par_region0_2_150 [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult.c} {91}]	0.897	0.897	60	0
void compute_interchange(double **, double **, double **, int, int, int) C [{matmult.c} {99,1}-{118,1}]	0	0.787	1	1
void initialize(double **, int, int) C [{matmult_initialize.c} {3,1}-{16,1}]	0.005	0.016	3	3
OpenMP_PARALLEL_REGION: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initialize.c} {5}	0	0.011	3	6
OpenMP_BARRIER: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initialize.c} {5}]	0.01	0.01	3	0
[CONTEXT] OpenMP_BARRIER: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initializ	0	0.03	2	0
OpenMP_LOOP: [OPENMP] initialize [{/usr/global/tools/tau/training/tau-2.23/examples/mm/matmult_initialize.c} {5}]	0.001	0.001	3	0

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-mpi-pdt-ompt-openmp
% make CC=tau_cc.sh CXX=tau_cxx.sh
% qsub …
% paraprof
```

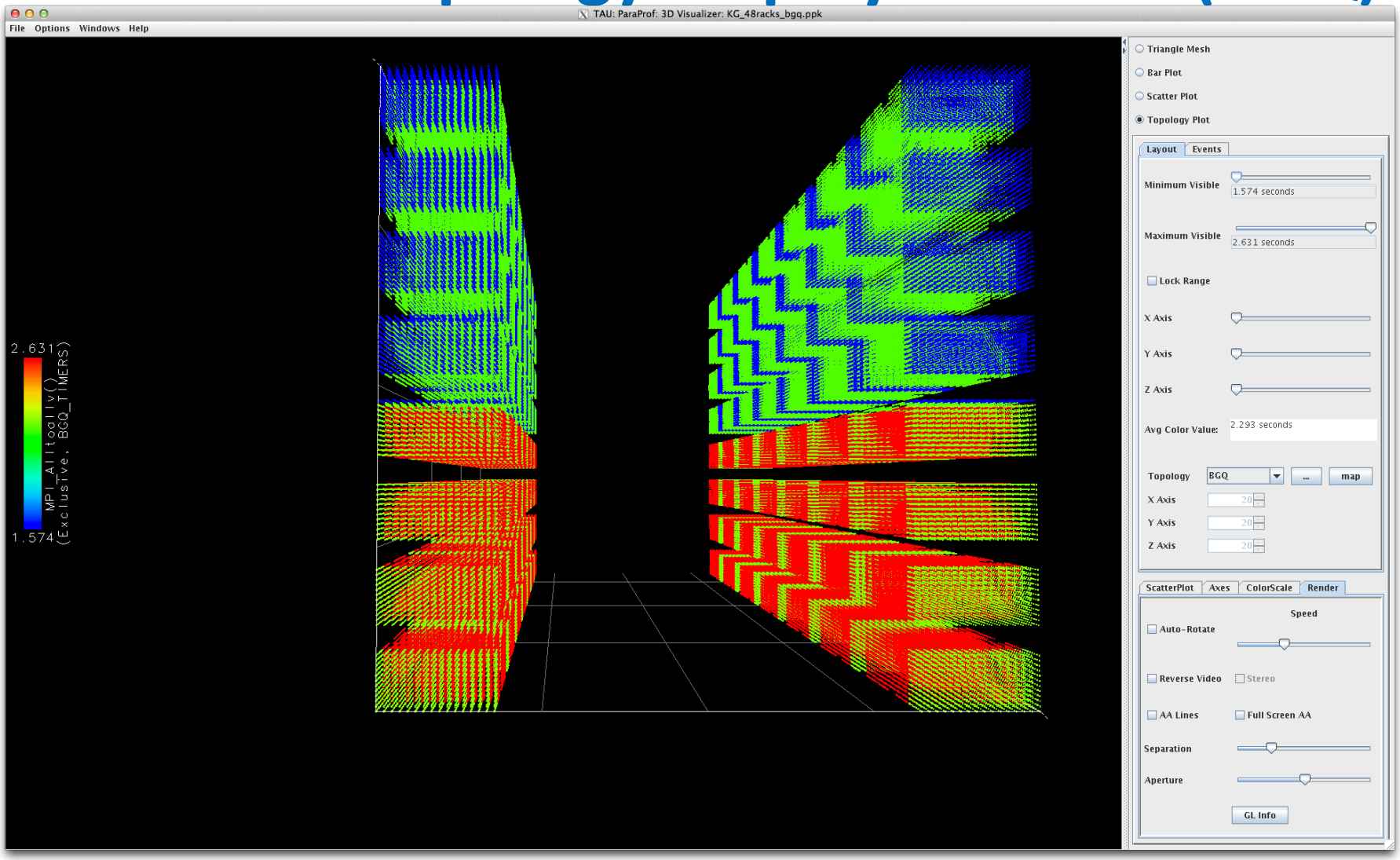
ParaProf Comparison Window



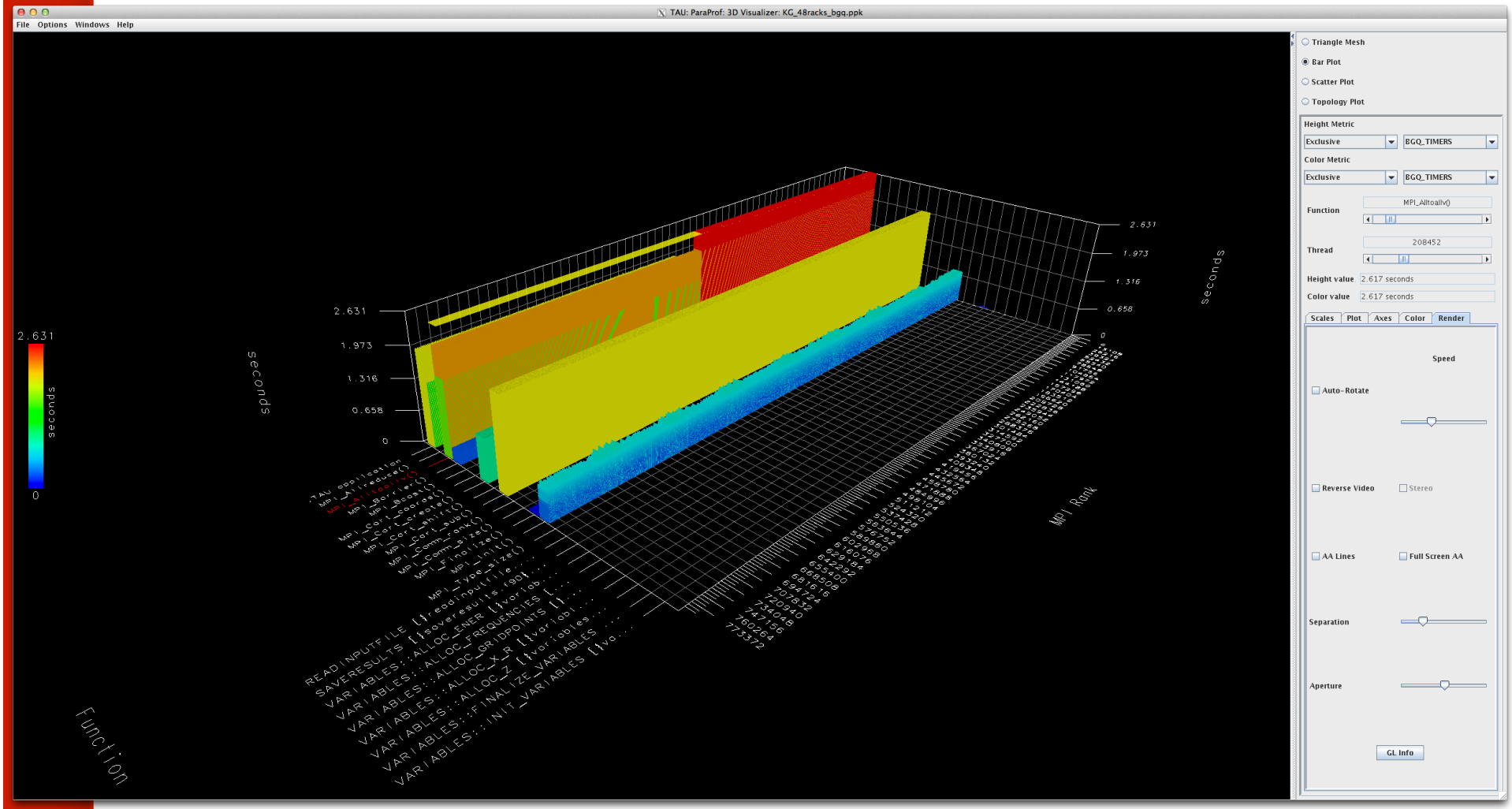
TAU's ParaProf Profile Browser



ParaProf's Topology Display Window (BGQ)

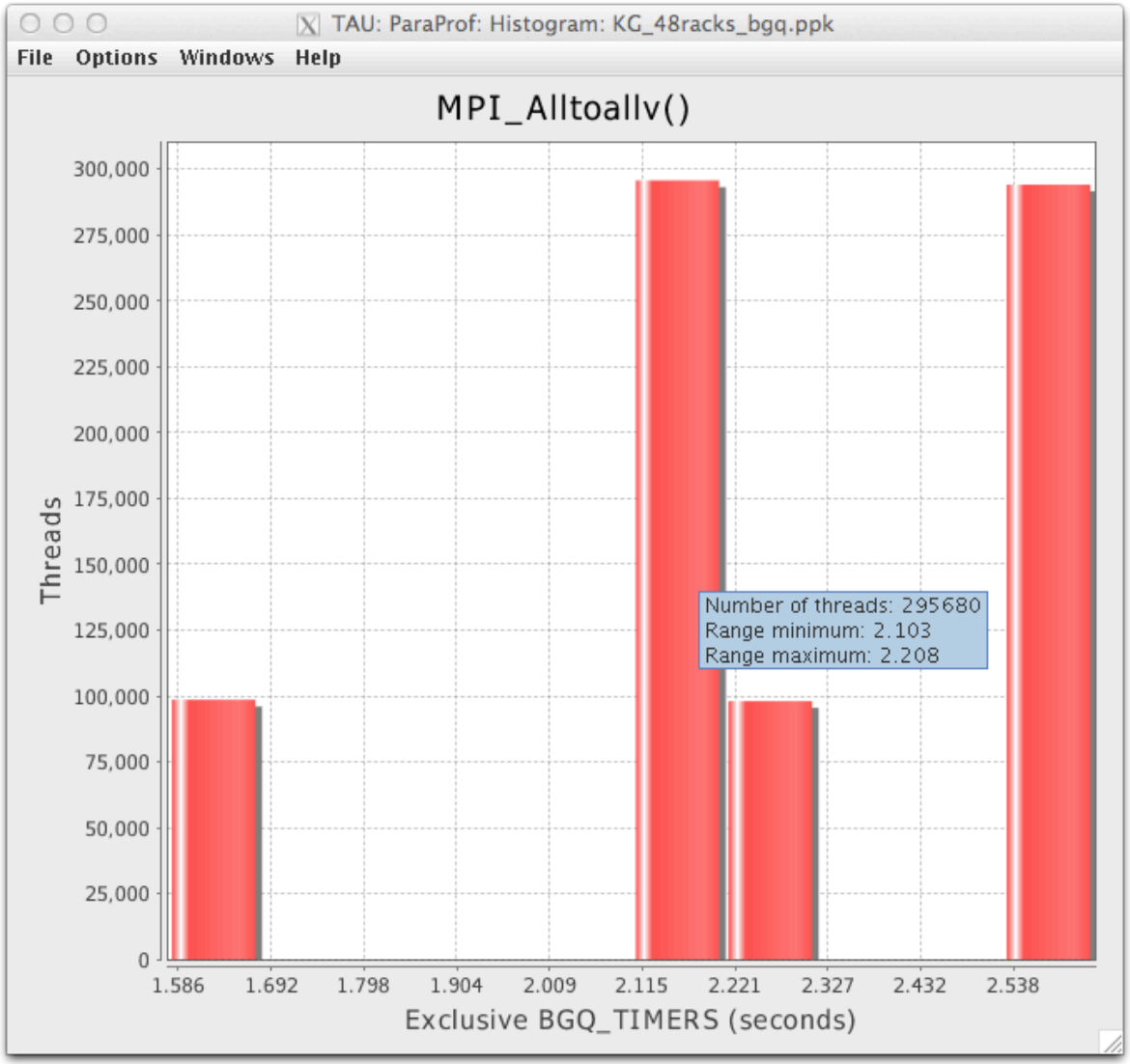


ParaProf's Scalable 3D Visualization (BGQ)



786,432 ranks

ParaProf Histogram Display



Runtime Preloading: tau_exec

- Runtime instrumentation by pre-loading the measurement library
- Works on dynamic executables (default under Linux)
- Can substitute I/O, MPI, SHMEM, CUDA, OpenCL, and memory allocation/deallocation routines with instrumented calls
- Track interval events (e.g., time spent in write()) as well as atomic events (e.g., how much memory was allocated) in wrappers
- Accurately measure I/O and memory usage
- Preload any wrapper interposition library in the context of the executing application

Preloading a TAU Library for x86_64

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

% ./configure -pdt=<dir> -mpi; make install
```
creates
```
Makefile.tau-bgqtimers-papi-mpi-pdt
shared-mpi-pdt/libTAU.so
```
To explicitly choose preloading of shared-<options>/libTAU.so change:
```
% mpirun -np 4 ./a.out      to
% mpirun -np 4  tau_exec -T <comma_separated_options> ./a.out

% mpirun -np 4 tau_exec -T papi,mpi,pdt ./a.out
```
Preloads `$TAU/shared-papi-mpi-pdt/libTAU.so`
```
% mpirun -np 4 tau_exec -T papi ./a.out
```
Preloads `$TAU/shared-papi-mpi-pdt/libTAU.so by matching.`
```
% mpirun -np 4 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.

TAU Execution Command (tau_exec)

Uninstrumented execution

- % mpirun –np 32 ./a.out

Track MPI performance

- % mpirun –np 32 tau_exec ./a.out

Track POSIX I/O and MPI performance (MPI enabled by default)

- % mpirun –np 32 tau_exec –io ./a.out

Track memory operations

- % export TAU_TRACK_MEMORY_LEAKS=1
- % mpirun –np 32 tau_exec –memory_debug ./a.out (bounds check)

Use event based sampling (compile with –g)

- % mpirun –np 32 tau_exec –ebs ./a.out
- Also –ebs_source=<PAPI_COUNTER> -ebs_period=<overflow_count>

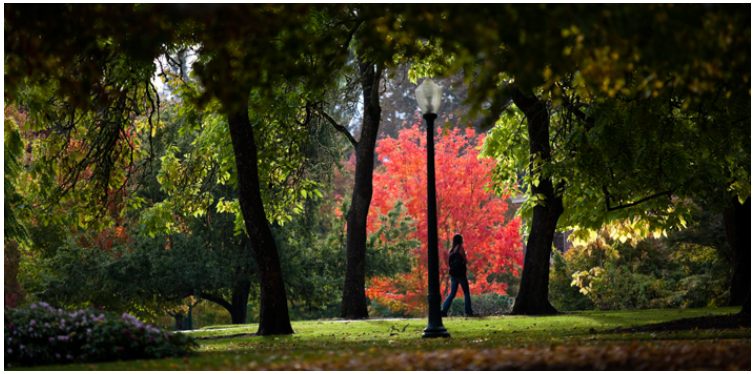
Load wrapper interposition library

- % mpirun –np 32 tau_exec –loadlib=<path/libwrapper.so> ./a.out

Track GPGPU operations

- % mpirun –np 32 tau_exec –cupti ./a.out
- % mpirun –np 32 tau_exec –openacc ./a.out

PRL, University of Oregon, Eugene



www.uoregon.edu

Support Acknowledgments

US Department of Energy (DOE)

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

Department of Defense (DoD)

- PETTT, HPCMP

National Science Foundation (NSF)

- Glassbox, SI-2

NASA

Partners:

University of Oregon

ParaTools, Inc.

University of Tennessee, Knoxville

T.U. Dresden, GWT

Juelich Supercomputing Center







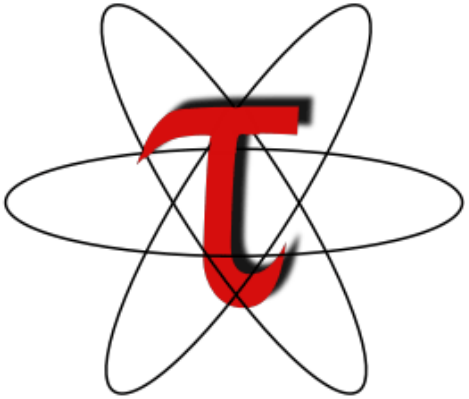








Download TAU from U. Oregon



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

Free download, open source, BSD license

Reference

Installing and Configuring TAU

- **Installing PDT:**
  - wget /pdt_lite.tgz
  - ./configure –prefix=<dir>; make ; make install
- **Installing TAU:**
  - wget /tau.tgz
  - ./configure -bfd=download -pdt=<dir> -papi=<dir> -arch=bgq...
  - See <taudir>/.all_configs file to see how TAU was configured
  - make install
- **Using TAU:**
  - export TAU_MAKEFILE=<taudir>/bgq/ lib/Makefile.tau-<TAGS>
  - make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
  - Also see TAU Commander in <taudir>/tau for managing projects and for simplifying TAU's workflow (Beta).

Compiling Fortran Codes with TAU

If your Fortran code uses free format in .f files (fixed is default for .f), you may use:

% export TAU_OPTIONS= '-optPdtF95Opts="-R free" -optVerbose '

To use the compiler based instrumentation instead of PDT (source-based):

% export TAU_OPTIONS= '-optCompInst -optVerbose'

If your Fortran code uses C preprocessor directives (#include, #ifdef, #endif):

% export TAU_OPTIONS= '-optPreProcess -optVerbose'

To use an instrumentation specification file:

% export TAU_OPTIONS= '-optTauSelectFile=select.tau -optVerbose -optPreProcess'

% cat select.tau

```
BEGIN_EXCLUDE_LIST
FOO
END_EXCLUDE_LIST

BEGIN_INSTRUMENT_SECTION
loops  routine="#"
# this statement instruments all outer loops in all routines. # is wildcard as well as comment in first column.
END_INSTRUMENT_SECTION
```

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

-optMICOffload	Links code for Intel MIC offloading, requires both host and MIC TAU libraries
-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_LEAKS	0	Setting to 1 turns on leak detection (for use with –optMemDbg or tau_exec)
TAU_TRACK_HEAP	0	Setting to 1 turns on collecting a sample of heap memory available at entry and exit of each routine and triggers a context event that records the callstack
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. Enabled by default to remove 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_COMPENSATE	0	Setting to 1 enables runtime compensation of instrumentation overhead
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., TIME,P_VIRTUAL_TIME,PAPI_FP_INS,PAPI_NATIVE_<event>:<subevent>)

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