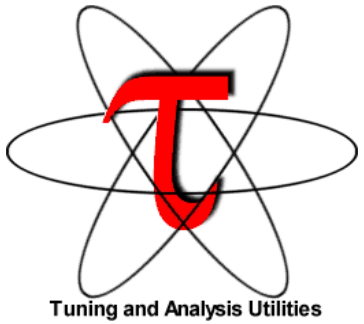
SALT-FM and TAU for Performance Analysis, Profiling and Beyond

Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, December 17th, 2024

https://www.nas.nasa.gov/pubs/ams.html

Prof. Sameer Shende
University of Oregon and ParaTools, Inc.
Eugene, OR

Izaak Beekman, ParaTools, Inc., MA



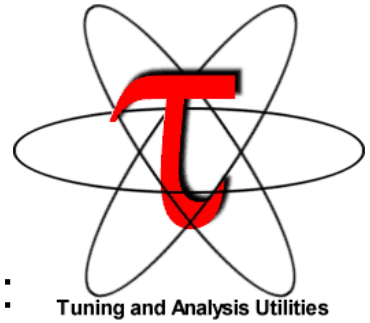


ParaTools

Motivation: Improving Productivity

- SALT-FM: Source Analysis Toolkit for Fortran and Mixed-Language
  - To harden and improve the usability of TAU and build a simplified interface on to LLVM technology
- TAU Performance System®:
  - Deliver a scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads
  - http://tau.uoregon.edu
- Extreme-scale Scientific Software Stack (E4S):
  - Delivering a modular, interoperable, and deployable software stack that includes TAU
  - https://e4s.io

TAU Performance System®





ParaTools

- Versatile profiling and tracing toolkit that supports:
  - MPI, CUDA, ROCm, DPC++/SYCL (Level Zero), OpenCL, and OpenMP (OpenMP Tools Interface for Target Offload), OpenACC
- Scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads that supports:
  - C++/C/DPC++, Fortran, Python
- Interfaces with Program Database Toolkit (PDT) and SALT-FM for instrumentation of source code.
- Supports PAPI, Likwid for hardware performance counter information
- Instrumentation includes support for PETSc (Perfstubs), XGC (CAMTIMERS), Kokkos, MPI, pthread, event-based sampling, GPU runtimes
- A single tool (tau_exec) is used to launch un-instrumented, un-modified binaries
- TAU's paraprof, pprof, perfexplorer for profile analysis; Vampir, Jumpshot, Perfetto.dev for traces
- http://tau.uoregon.edu

TAU: Quickstart Guide

Profiling:

```
MPI: % mpirun -np 16 tau_exec -ebs ./a.out
```

- `Pthread:  % mpirun -np 16 tau_exec -T mpi,pthread -ebs ./a.out`
- `CUDA: % mpirun -np 16 tau_exec -T cupti,mpi -cupti -ebs ./a.out`
- `Score-P: % mpirun -np 16 tau_exec -T scorep,mpi ./a.out`

```
Analysis: % pprof -a -m | more;  % paraprof (GUI)
```

Tracing:

- `Vampir: MPI: % export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2`
  `% mpirun -np 16 tau_exec ./a.out; vampir traces.otf2 &`
- `Chrome/Jumpshot: % export TAU_TRACE=1; mpirun -np 64 tau_exec ./a.out`
  `% tau_treemerge.pl;`

```
Chrome: % tau_trace2json tau.trc tau.edf -chrome -ignoreatomic -o app.json
   Chrome browser: chrome://tracing   (Load -> app.json) or Perfetto.dev
```

- `Jumpshot: tau2slog2 tau.trc tau.edf -o app.slog2; jumpshot app.slog2`

Using TAU’s Runtime Preloading Tool: tau_exec

- Preload a wrapper that intercepts the runtime system call and substitutes with another
  - **MPI**
  - **OpenMP**
  - **POSIX I/O**
  - **Memory allocation/deallocation routines**
  - **Wrapper library for an external package**
- No modification to the binary executable!
- Enable other TAU options (communication matrix, OTF2, event-based sampling)

TAU Execution Command (tau_exec)

Uninstrumented execution

```
% mpirun -np 256  ./a.out
```

Track GPU operations

```
% mpirun -np 256  tau_exec –cupti ./a.out
% mpirun -np 256  tau_exec –cupti  -um ./a.out  (for Unified Memory)
% mpirun -np 256  tau_exec –rocm ./a.out
% mpirun –np 256  tau_exec –l0      ./a.out
% mpirun –np 256 tau_exec –opencl ./a.out
% mpirun –np 256 tau_exec –openacc ./a.out
```

Track MPI performance

```
% mpirun -np 256   tau_exec ./a.out
```

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

```
% mpirun -np 256  tau_exec -io  ./a.out
```

Track OpenMP and MPI execution (using OMPT for Intel v19+ or Clang 8+)

```
% export TAU_OMPT_SUPPORT_LEVEL=full;
% mpirun –np 256  tau_exec –T ompt,mpi  -ompt  ./a.out
```

Track memory operations

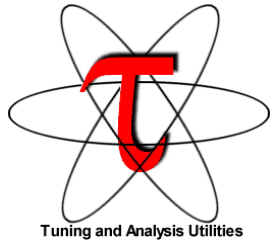
```
% export TAU_TRACK_MEMORY_LEAKS=1
% mpirun –np 256 tau_exec –memory_debug ./a.out (bounds check)
```

Use event based sampling (compile with –g)

```
% mpirun –np 256 tau_exec –ebs ./a.out
Also  export TAU_METRICS=TIME,PAPI_L1_DCM…  -ebs_resolution=<file | function | line>
```

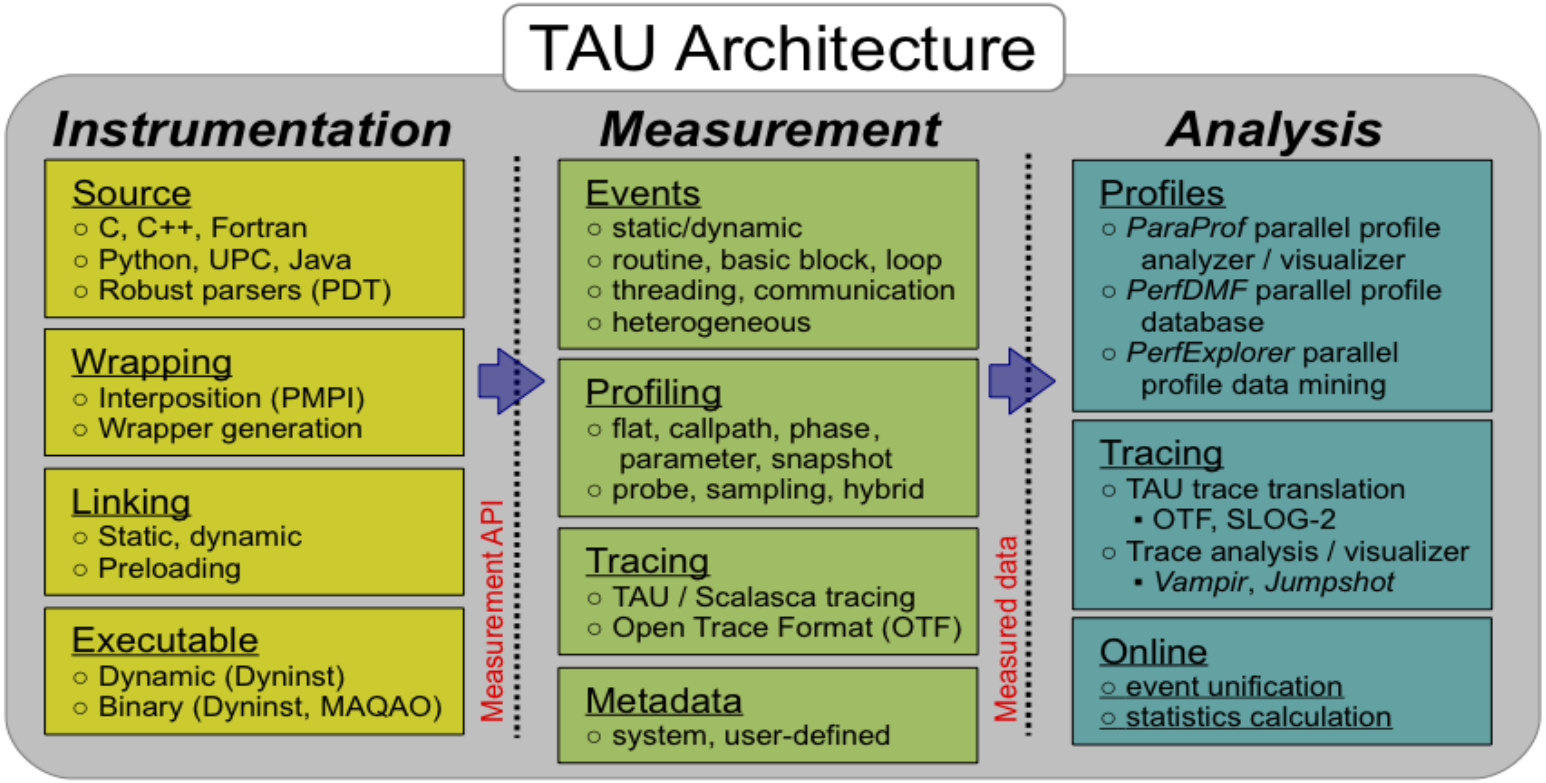
Advanced Instrumentation Features of TAU

TAU Performance System®



Parallel performance framework and toolkit

- Supports all HPC platforms, compilers, runtime system
- Provides portable instrumentation, measurement, analysis



Application Performance Engineering using TAU

- How much time is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*? What is the time spent in OpenMP loops? In kernels on GPUs. How long did it take to transfer data between host and device (GPU)?
- How many instructions are executed in these code regions?
  Floating point, Level 1 and 2 *data cache misses*, hits, branches taken? What is the extent of vectorization for loops?
- How much time did my application spend waiting at a barrier in MPI collective operations?
- What is the memory usage of the code? When and where is memory allocated/de-allocated? Are there any memory leaks? What is the memory footprint of the application? What is the memory high water mark?
- How much energy does the application use in Joules? What is the peak power usage?
- What are the I/O characteristics of the code? What is the peak read and write *bandwidth* of individual calls, total volume?
- How does the application *scale*? What is the efficiency, runtime breakdown of performance across different core counts?

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

Instrumentation

Add hooks in the code to perform measurements

- **Source instrumentation using a preprocessor**
  - Add timer start/stop calls in a copy of the source code.
  - Use SALT-FM or Program Database Toolkit (PDT) for parsing source code.
  - Requires recompiling the code using TAU shell scripts (tau_cc.sh, tau_f90.sh)
  - Selective instrumentation (filter file) can reduce runtime overhead and narrow instrumentation focus.
- **Compiler-based instrumentation**
  - Use system compiler to add a special flag to insert hooks at routine entry/exit.
  - TAU LLVM plugin for selective instrumentation at the LLVM IR level before codegen.
  - Requires recompiling using TAU compiler scripts (tau_cc.sh, tau_f90.sh…)
- **Runtime preloading of TAU's Dynamic Shared Object (DSO)**
  - No need to recompile code! Use **mpirun tau_exec ./app** with options.

Using Instrumentation of Source Code in TAU using PDT

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:

  (or module load tau…)

  % export TAU_MAKEFILE=$TAU/Makefile.tau-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:

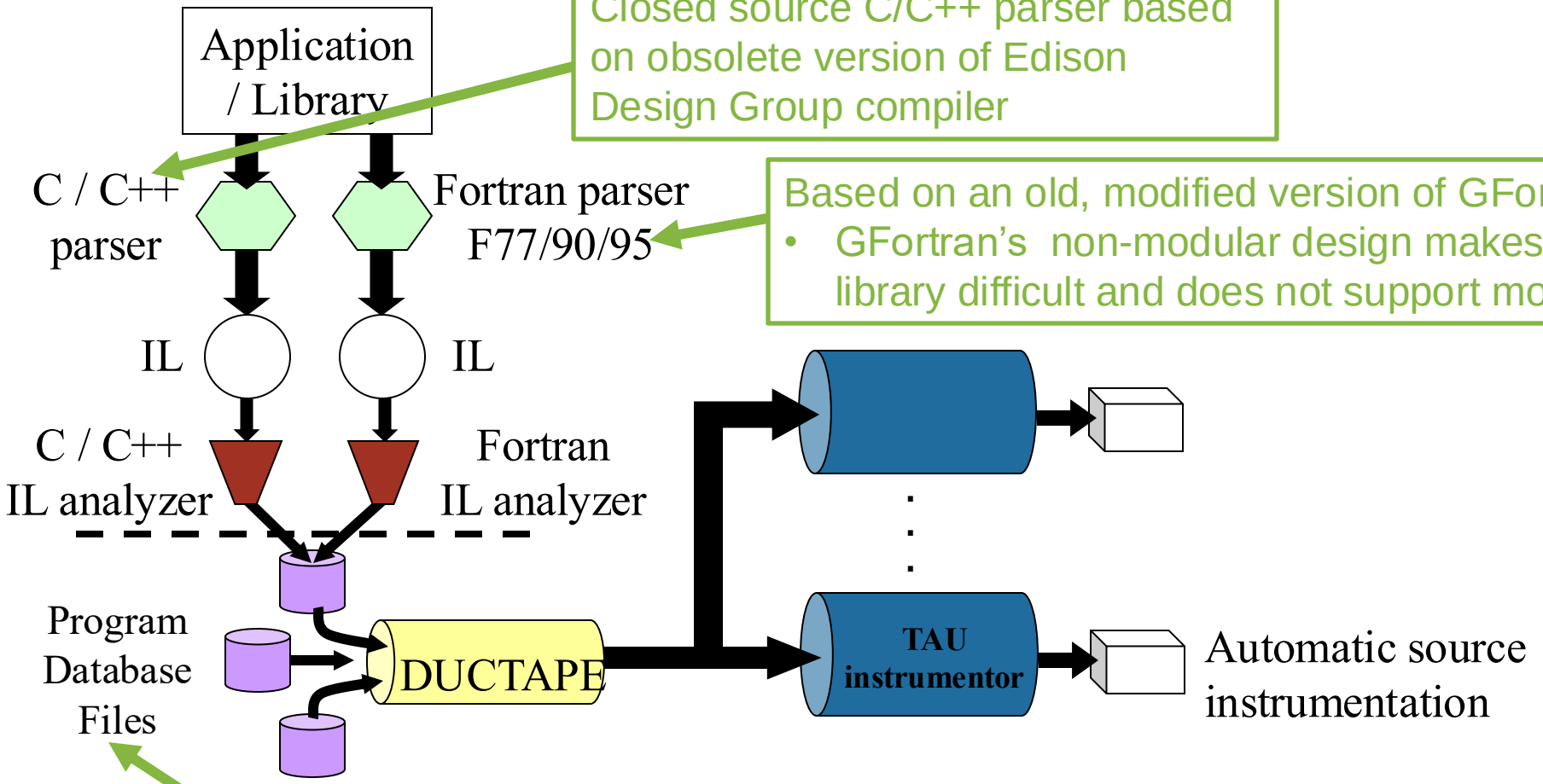
- **% pprof (for text based profile display)**
- **% paraprof (for GUI)**

TAU Configurations available

```
% module load tau
%  ls $TAU/Makefile*
/packages/tau-2.34/x86_64/lib/Makefile.tau-intel-papi-mpi-pthread-pdt
/packages/tau-2.34/x86_64/lib/Makefile.tau-intel-papi-ompt-mpi-pdt-openmp
/packages/tau-2.34/x86_64/lib/Makefile.tau-papi-mpi-pdt
/packages/tau-2.34/x86_64/lib/Makefile.tau-papi-pdt
/packages/tau-2.34/x86_64/lib/Makefile.tau-papi-pthread-pdt
/packages/tau-2.34/x86_64/lib/Makefile.tau-papi-tbb-pdt
For an uninstrumented binary:
% mpirun –np 16 tau_exec –T mpi,papi,pdt ./a.out
Picks the configuration represented by
/packages/tau-2.34/x86_64/lib/Makefile.tau-intel-papi-mpi-pdt
To use OpenMP instrumentation:
% export TAU_OMPT_SUPPORT_LEVEL=full
% export OMP_NUM_THREADS=<N>
% mpirun –np 16 tau_exec –T ompt,mpi –ompt –ebs ./a.out

To use TAU’s source code instrumentation:
% export TAU_MAKEFILE=/packages/tau-2.34/x86_64/lib/Makefile.tau-intel-papi-mpi-pdt
% make CC=tau_cc.sh F90=tau_f90.sh CXX=tau_cxx.sh ; mpirun –np 16 ./a.out
% pprof –a | more
% paraprof
% paraprof --pack foo.ppk
# Copy it to your local machine and launch: % paraprof foo.ppk
```

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



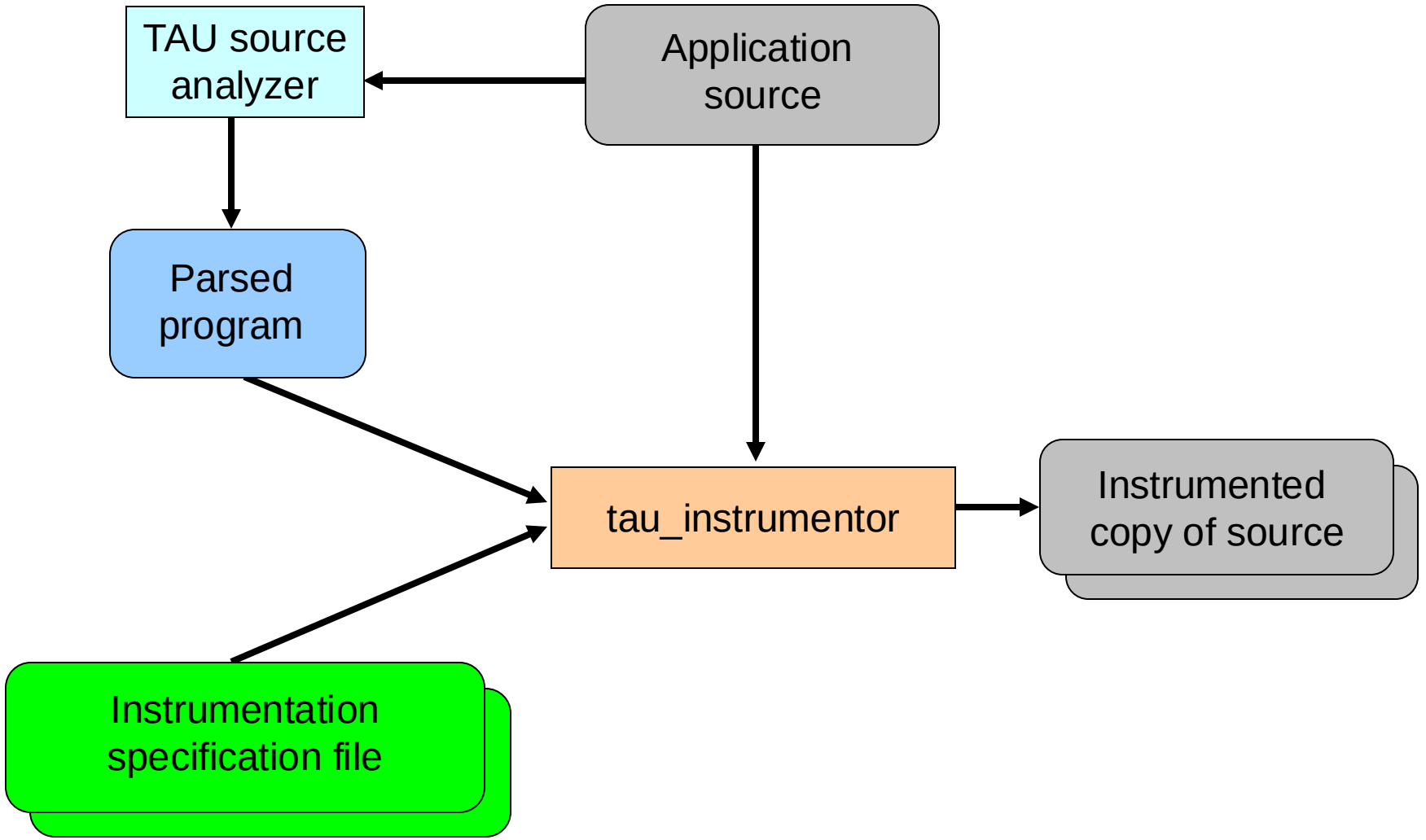
Closed source C/C++ parser based on obsolete version of Edison Design Group compiler

Based on an old, modified version of GFortran.

- GFortran's non-modular design makes use as a library difficult and does not support modern Fortran

Text-based database files enumerating all functions, statements, types

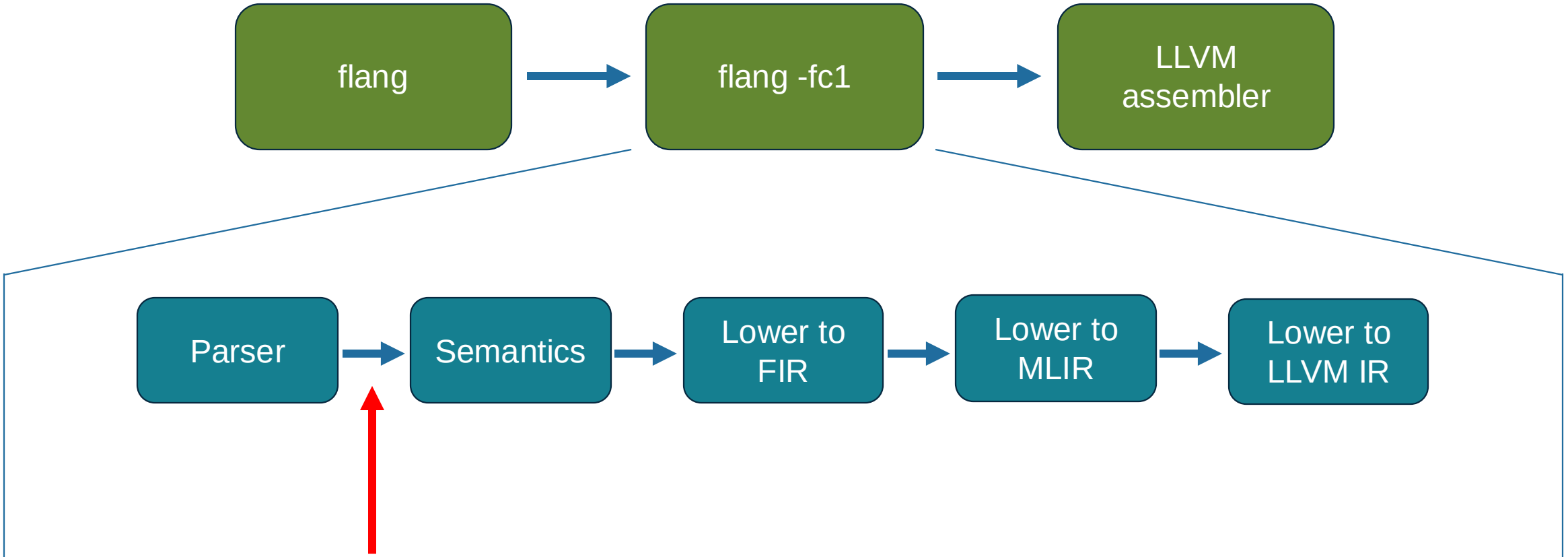
Automatic Source Instrumentation using PDT



Problems with PDT

- PDT has *usability issues*
  - Not compatible with any C++ features beyond C++11
  - Closed source — so have to ship large binary blobs to end users
  - PDB file requirement to serialize all types to textual description presents problems with modern C++ with deeply-templated types (large number of types with long description length)
- PDT has *maintainability issues*
  - Closed source – developers must sign NDA, limits number of developers involved
  - PDT instrumentor mixes EDG parser and PDT client code in same codebase — difficult to upgrade to new versions
  - Difficult to support for new C++ features
  - Major EDG release makes numerous breaking changes to API, which would require a from-scratch rewrite to adopt
- Fortran support
  - Based on older GFortran parser that does not support modern Fortran features

Flang-based Instrumentation



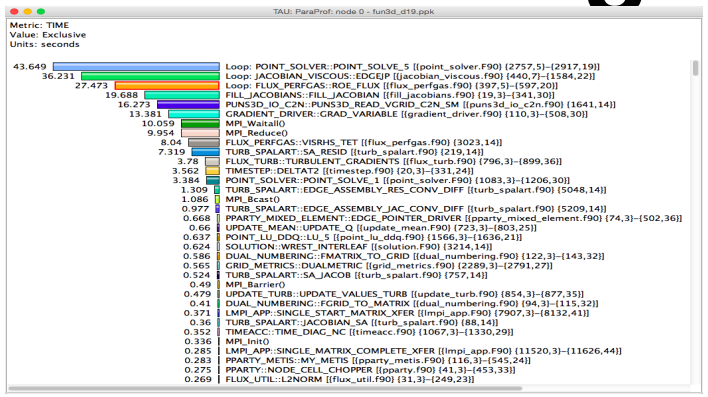
SALT is implemented as a Flang Frontend Plugin which receives the Fortran parse tree from flang –fc1 after parse tree generation and before the semantics stage.

SALT plugin for Flang: NASA SBIR Phase I

- Phase I proof-of-concept implements instrumentation for performance monitoring.
- SALT Visitor visits each node of parse tree
  - Identifies programs/subprograms for instrumentation
  - Identifies source locations for inserting variable declarations and timer start/stop code for TAU timers
    - Not all parse tree node types in Flang have associated source position data.
    - Scans into children of a given node to find source position data in subnodes.
- Phase I focused specifically on instrumentation.
- Phase II will expand focus to more general code analysis and transformation tasks.

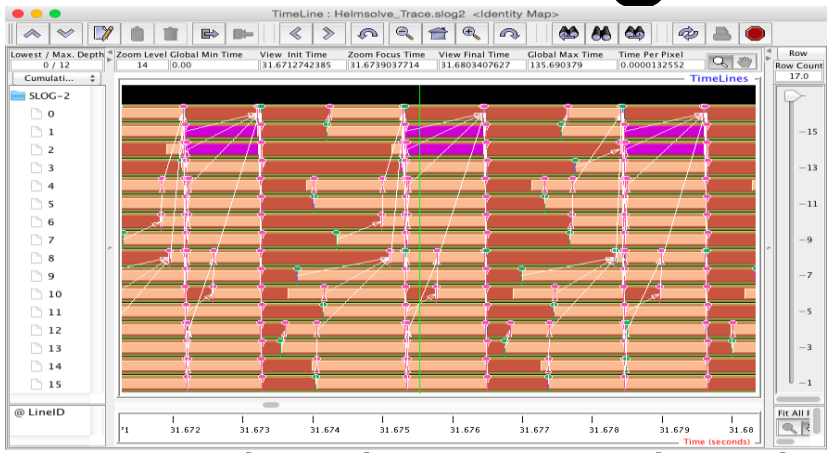
Profiling and Tracing

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

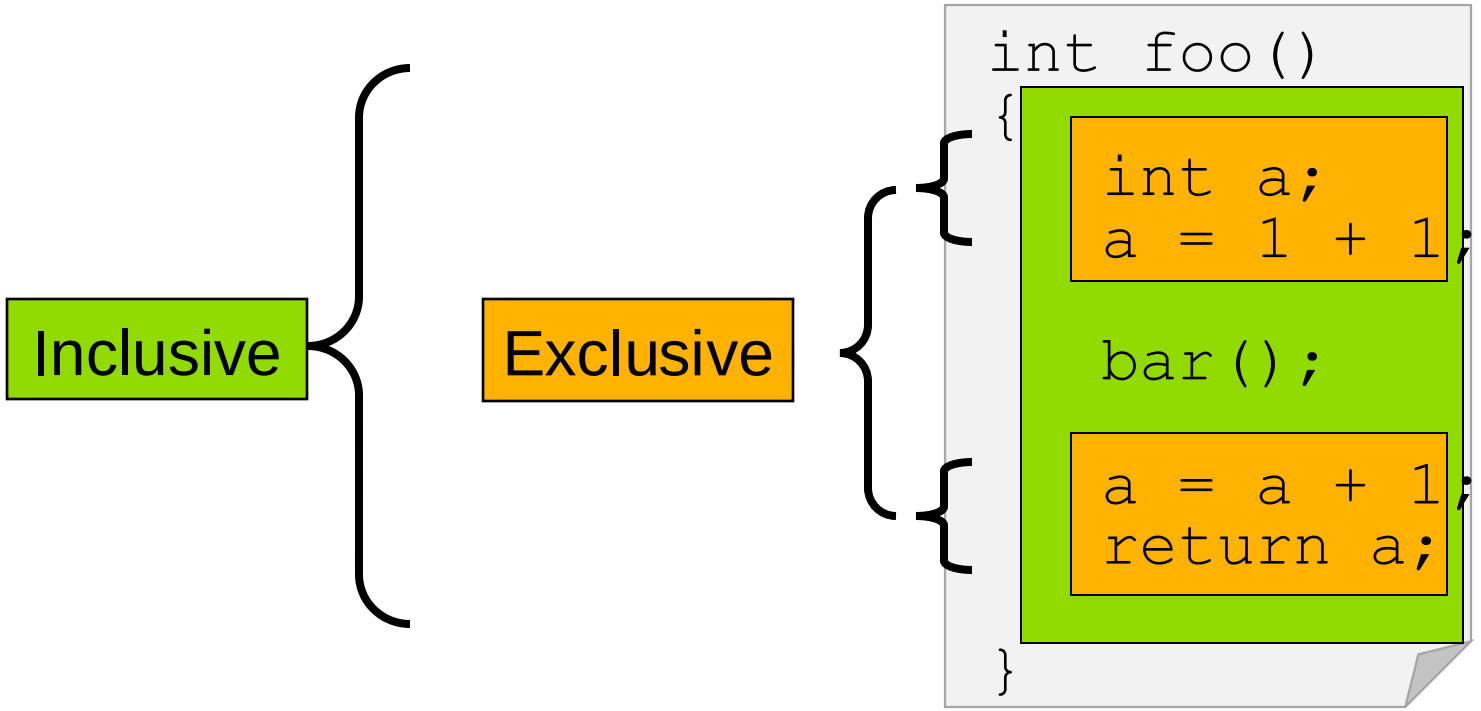
- Profiling and tracing

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

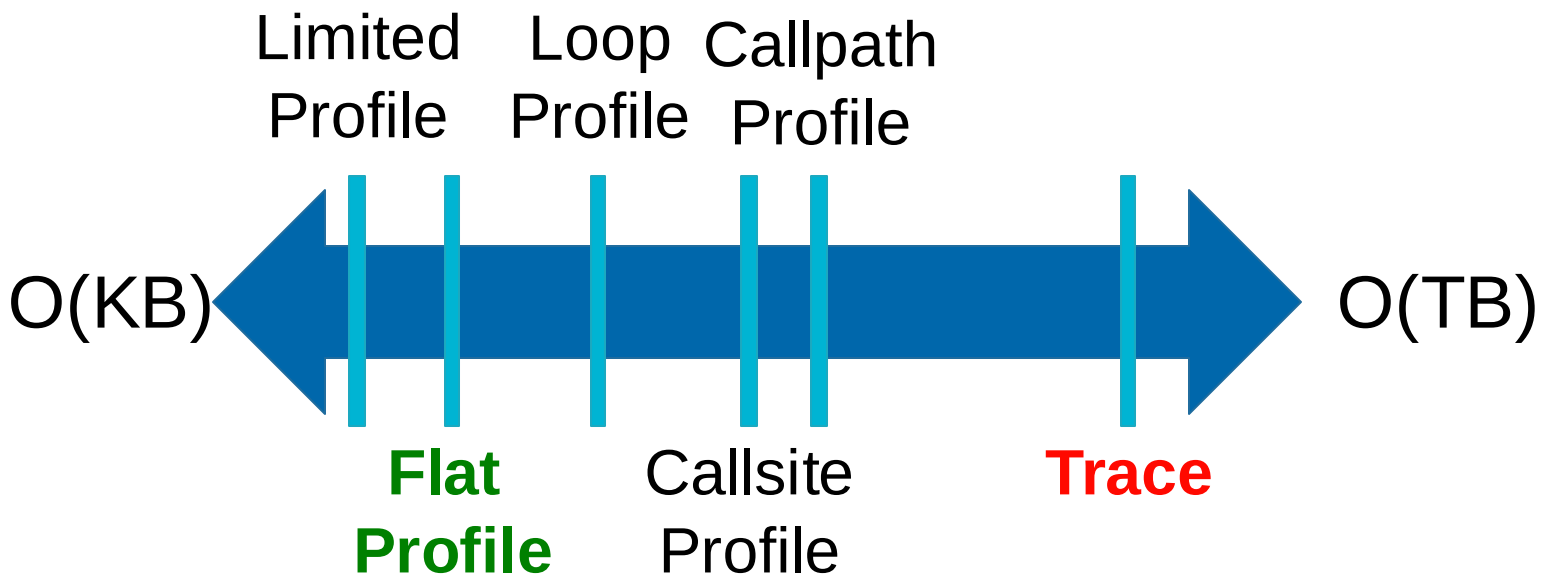
  Tracing shows you **when** the events take place on a timeline

Inclusive vs. Exclusive values

- Inclusive
  - Information of all sub-elements aggregated into single value
- Exclusive
  - Information cannot be subdivided further



How much data do you want?



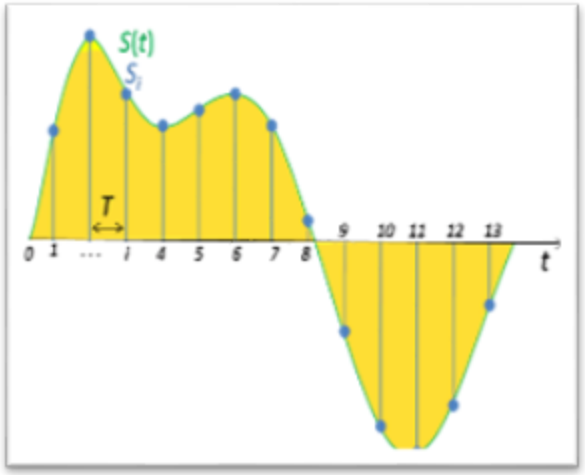
Performance Data Measurement

Direct via Probes

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

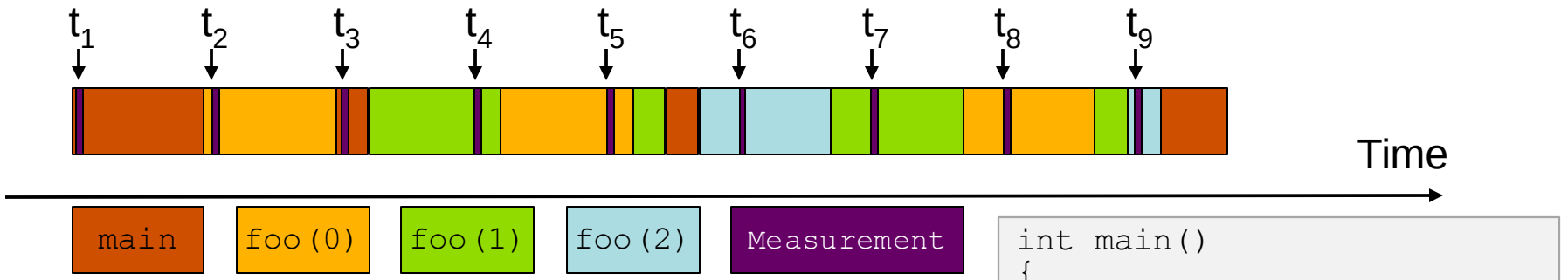
- Exact measurement
- Fine-grain control
- Calls inserted into code

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);

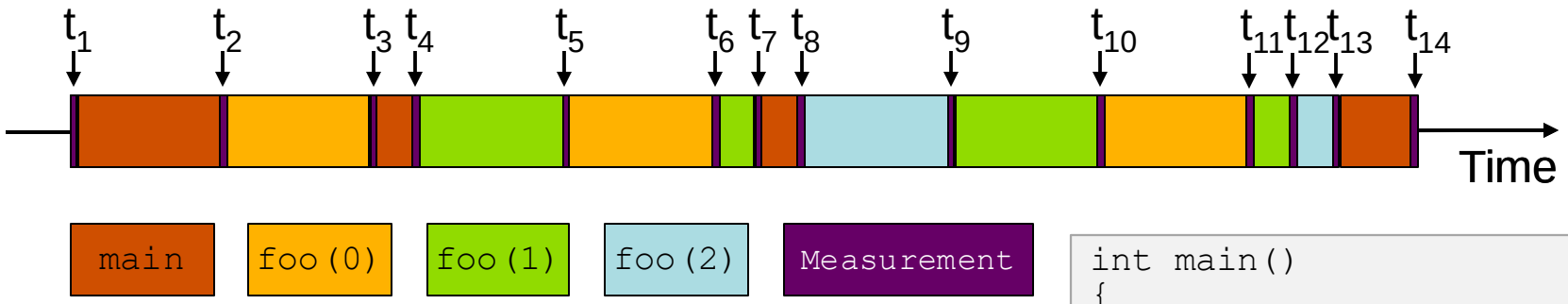
  return 0;
}

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;
  TAU_START("main");
  for (i=0; i < 3; i++)
    foo(i);
  TAU_STOP("main");
  return 0;
}

void foo(int i)
{
  TAU_START("foo");
  if (i > 0)
    foo(i - 1);
  TAU_STOP("foo");

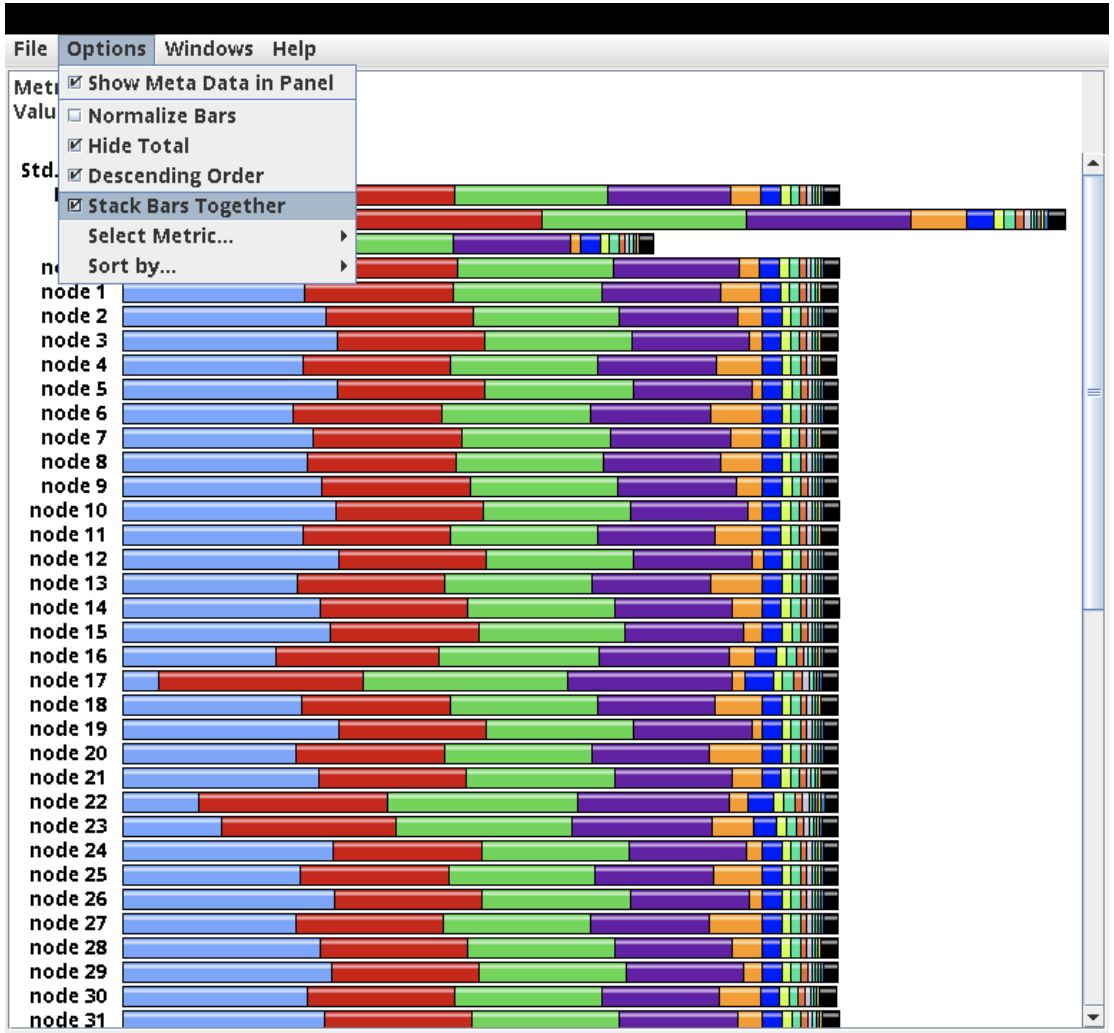
}
```

ParaProf Profile Browser

```
% paraprof
```

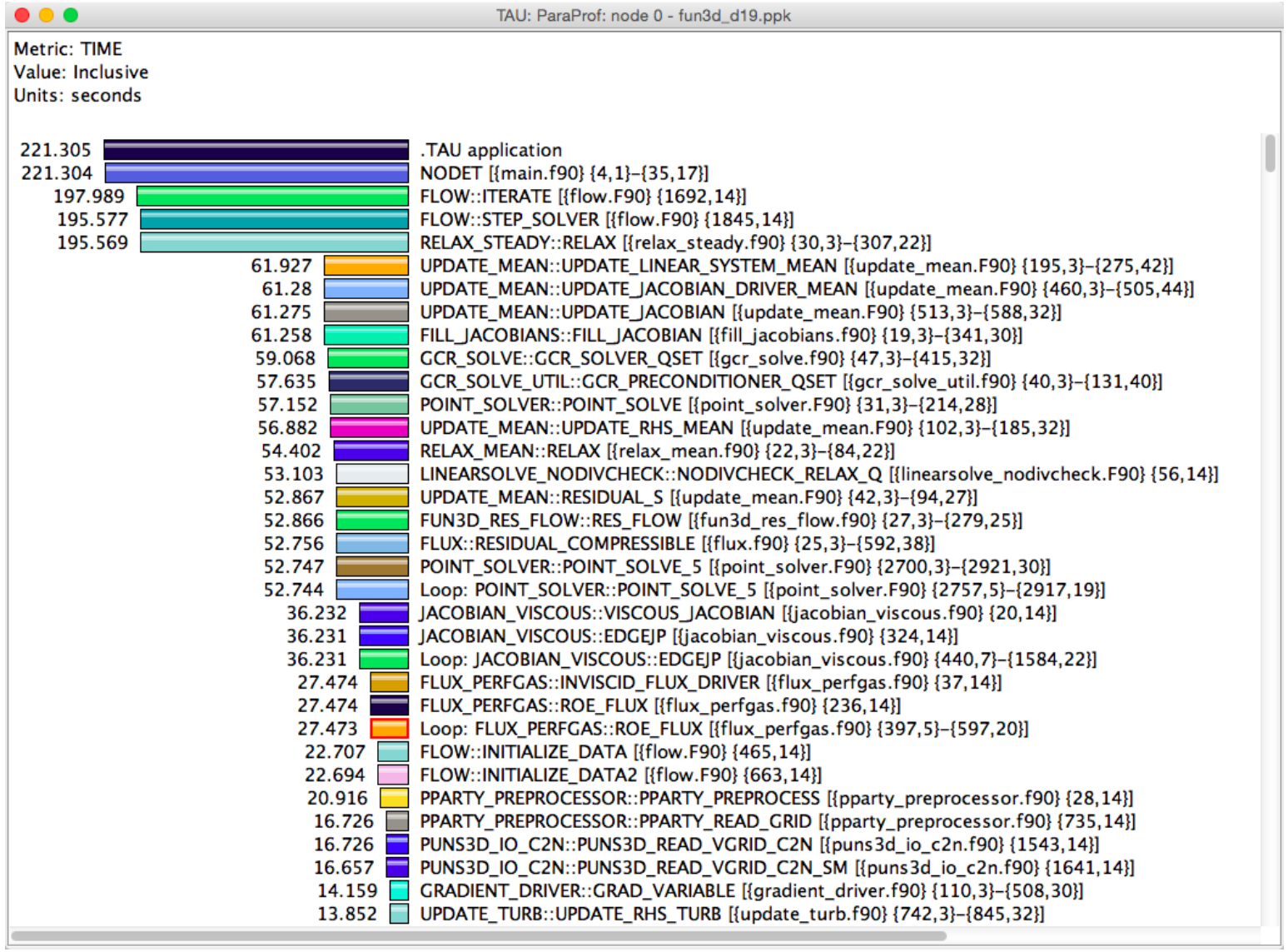


ParaProf Profile Browser

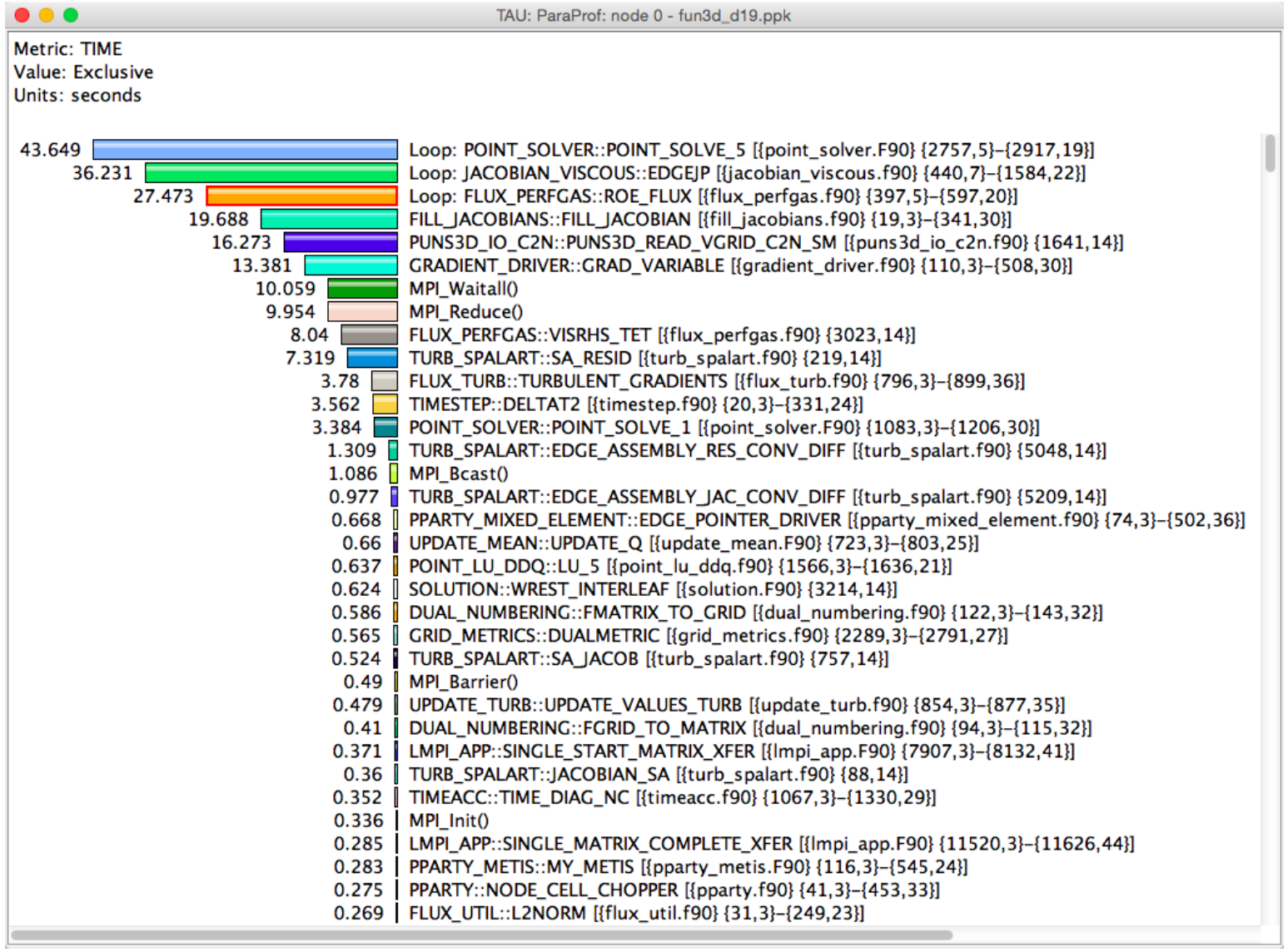




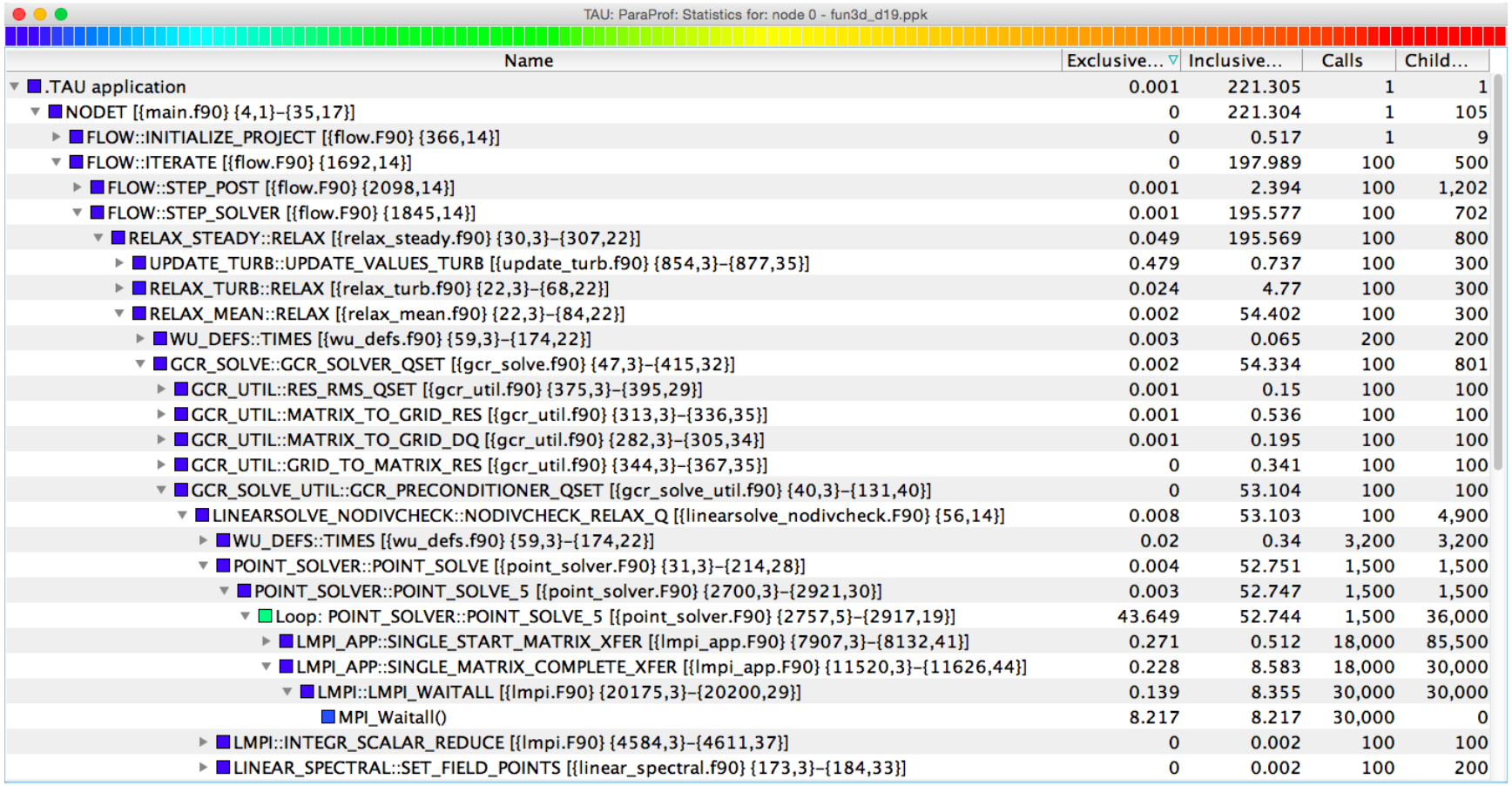
Inclusive Measurements



Exclusive Time



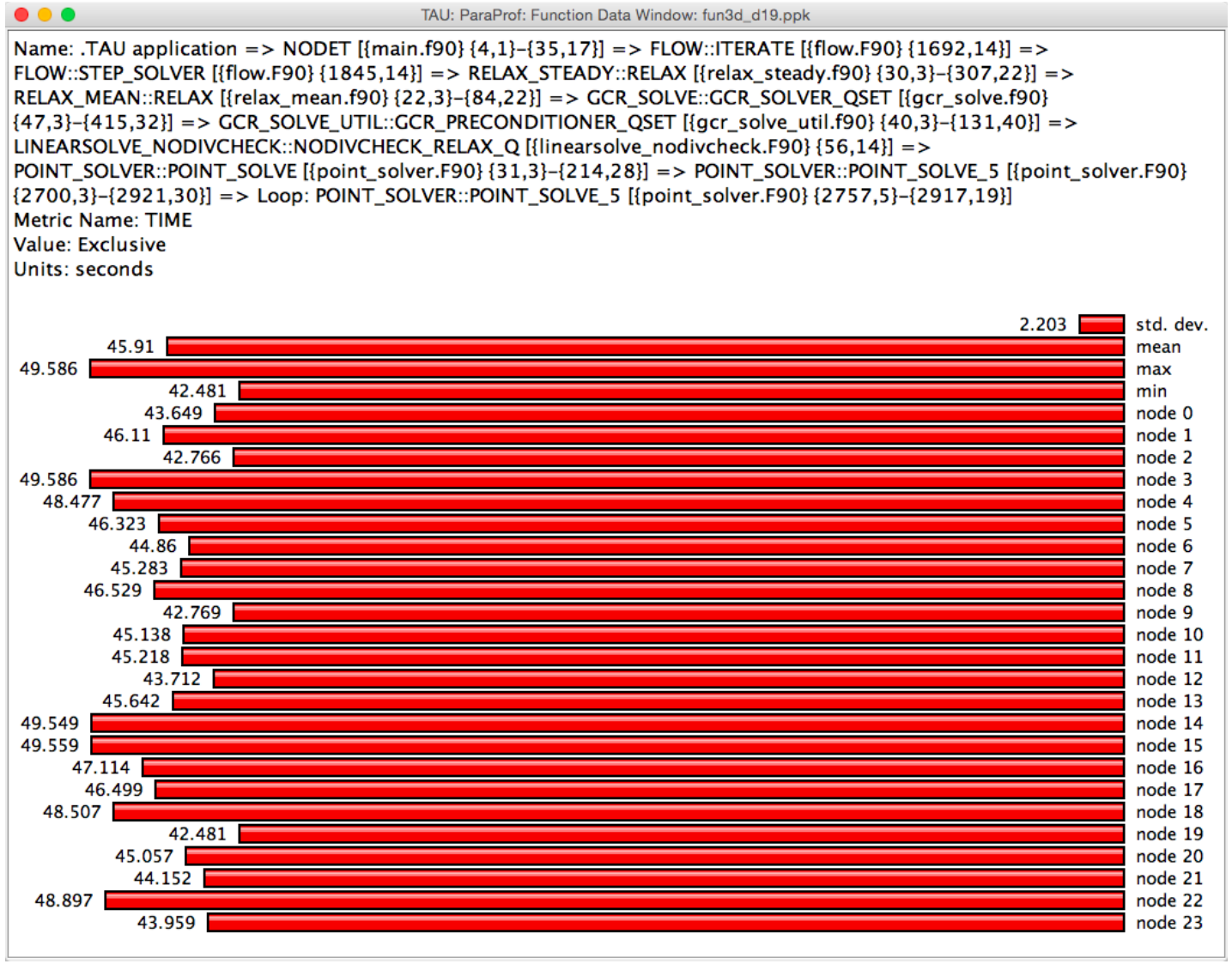
Callpath Profiling



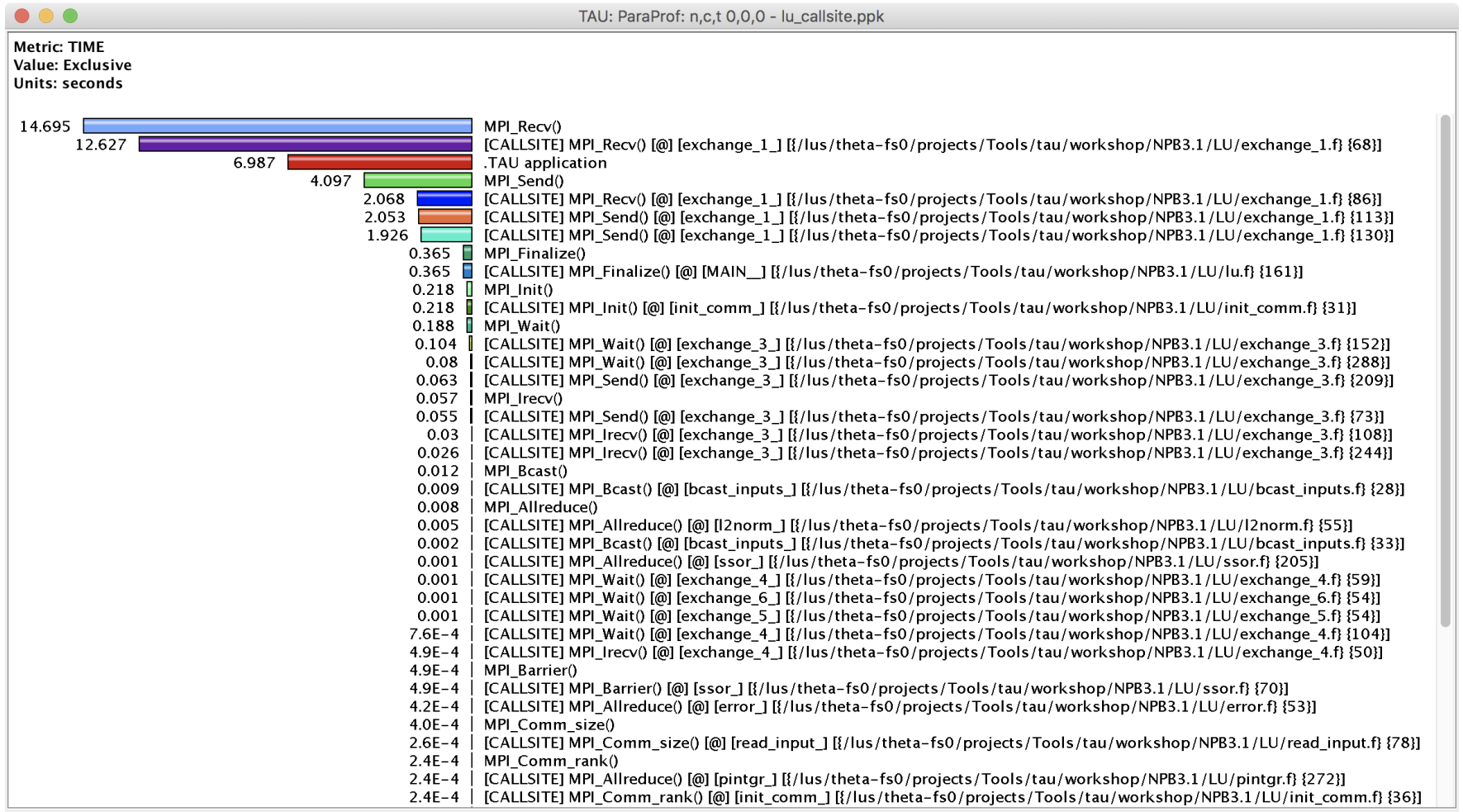
Name	Exclusive...	Inclusive...	Calls	Child...
.TAU application	0.001	221.305	1	1
NODET [{main.f90} {4,1}-{35,17}]	0	221.304	1	105
FLOW::INITIALIZE_PROJECT [{flow.F90} {366,14}]	0	0.517	1	9
FLOW::ITERATE [{flow.F90} {1692,14}]	0	197.989	100	500
FLOW::STEP_POST [{flow.F90} {2098,14}]	0.001	2.394	100	1,202
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

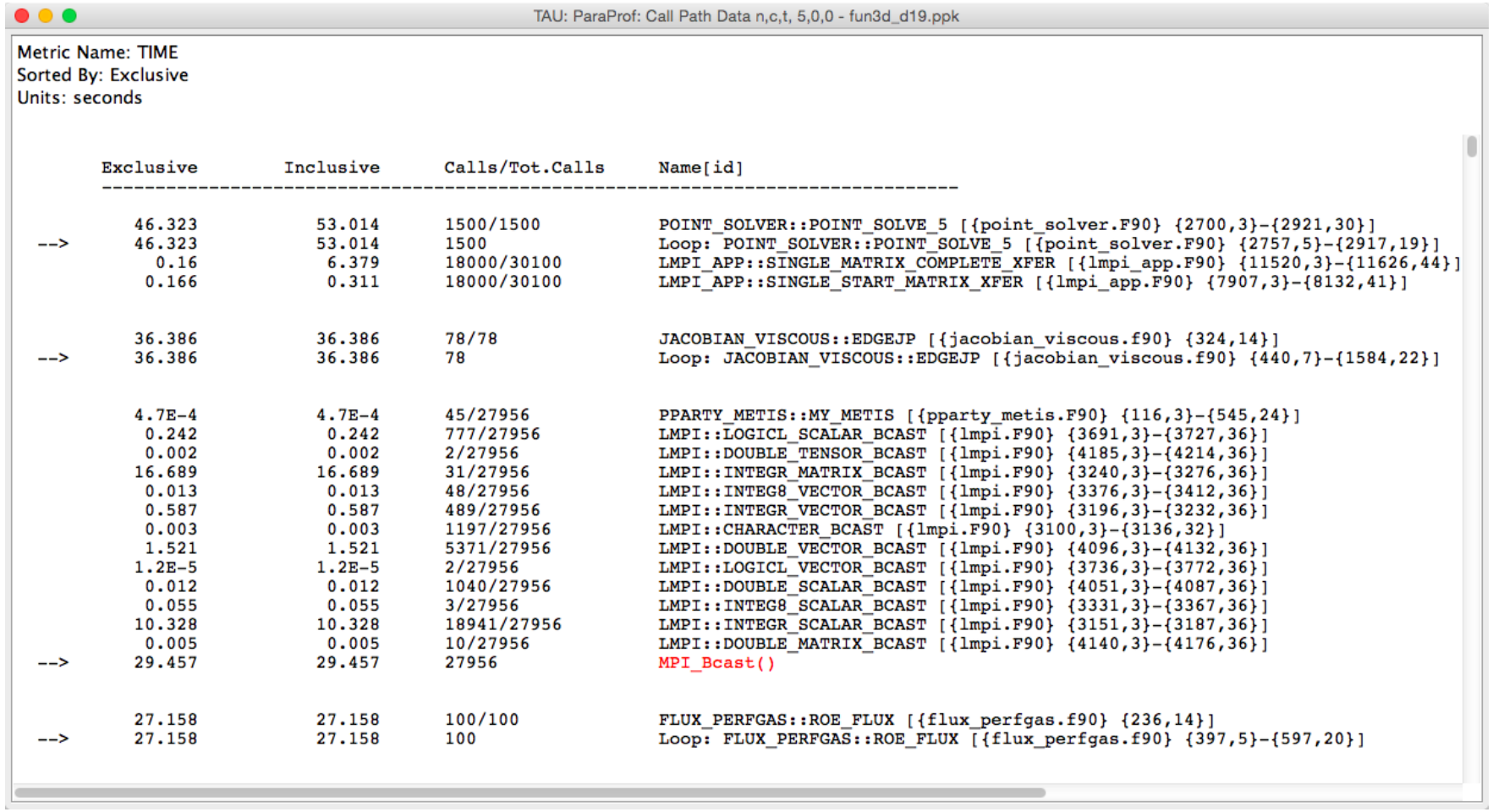


Callsite Profiling and Tracing



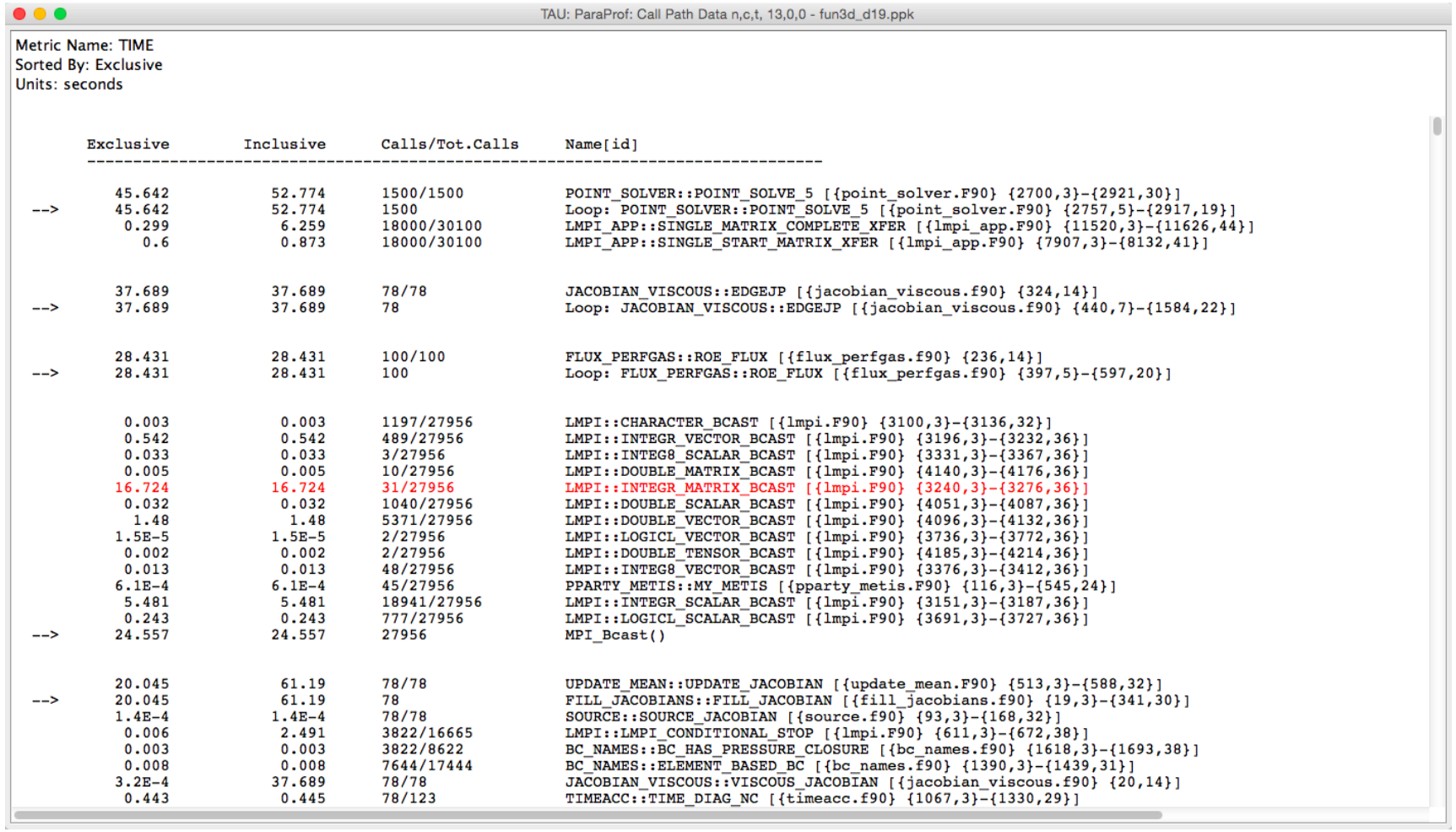
% export TAU_CALLSITE=1

ParaProf Callpath Thread Relations Window



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

ParaProf Callpath Thread Relations Window



Identifying Collective Wait States: Thread Callpath Relations Window

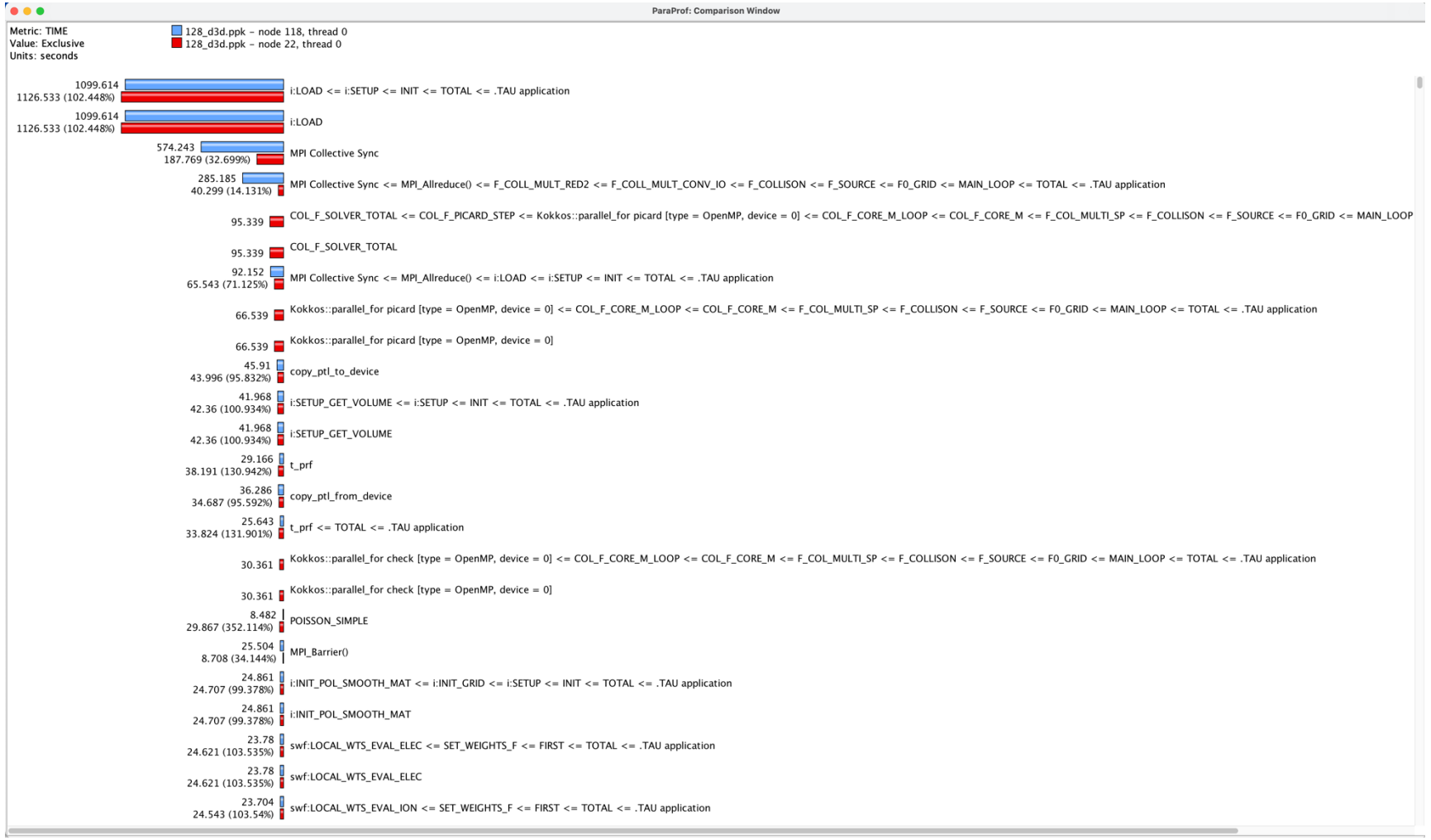
TAU: ParaProf: Call Path Data n,c,t, 118,0,0 - 128_d3d.ppk

Metric Name: TIME
Sorted By: Exclusive
Units: seconds

	Exclusive	Inclusive	Calls/Tot.Calls	Name[id]
	1099.614	1191.772	1/1	i:SETUP
-->	1099.614	1191.772	1	i:LOAD
	0.006	92.158	3/9543	MPI_Allreduce()
	9.8E-4	9.8E-4	11/15177	MPI_Gatherv()
	1.448	1.448	43/15177	MPI_Gather()
	15.353	15.353	46/15177	MPI_Alltoall()
	89.821	89.821	4311/15177	MPI_Bcast()
	6.777	6.777	195/15177	MPI_Allgather()
	68.678	68.678	991/15177	MPI_Reduce()
	9.179	9.179	12/15177	MPI_Comm_dup()
	0.125	0.125	25/15177	MPI_Allgatherv()
	382.861	382.861	9543/15177	MPI_Allreduce()
-->	574.243	574.243	15177	MPI Collective Sync
	2.507	2.508	10/186	DISTRIBUTE_F0G
	2.433	2.434	10/186	F_UPD_F0_SP
	5.156	5.158	20/186	F0_CHARGE_SEARCH_INDEX
	5.505	5.507	22/186	PULLBACK_WEIGHT
	24.86	24.872	102/186	UPDATE_PTL_WEIGHT
	0.473	0.473	2/186	MAIN_LOOP
	4.975	4.977	20/186	DIAG_f0_PORT1_PTL
-->	45.91	45.93	186	copy_ptl_to_device
	0.02	0.02	186/272	Kokkos::parallel_for set_buffer_particles_d [type = Cuda, device = 0]

MPI Collective Sync is the time spent in a barrier operation inside a collective

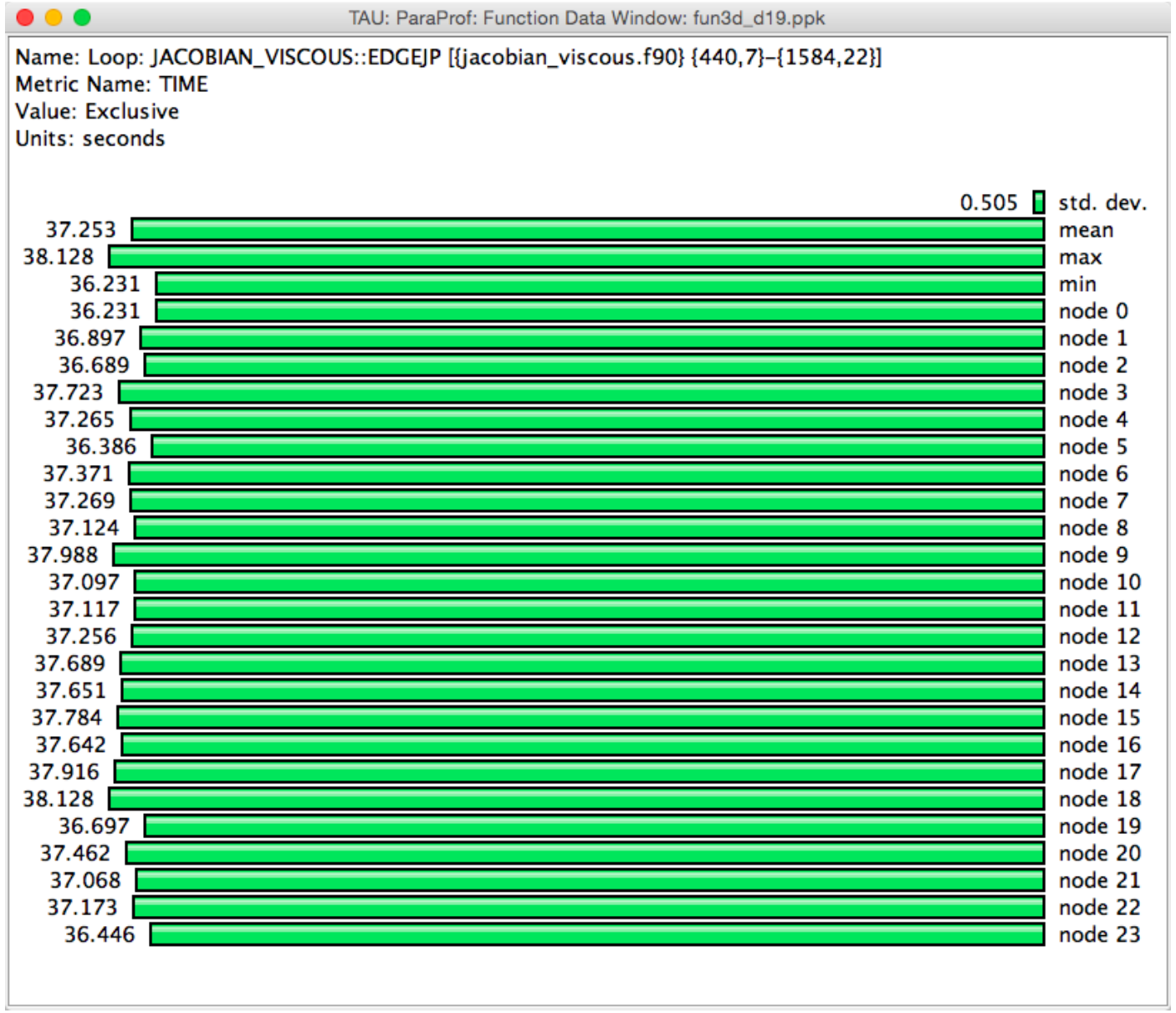
ParaProf Thread Comparison Window



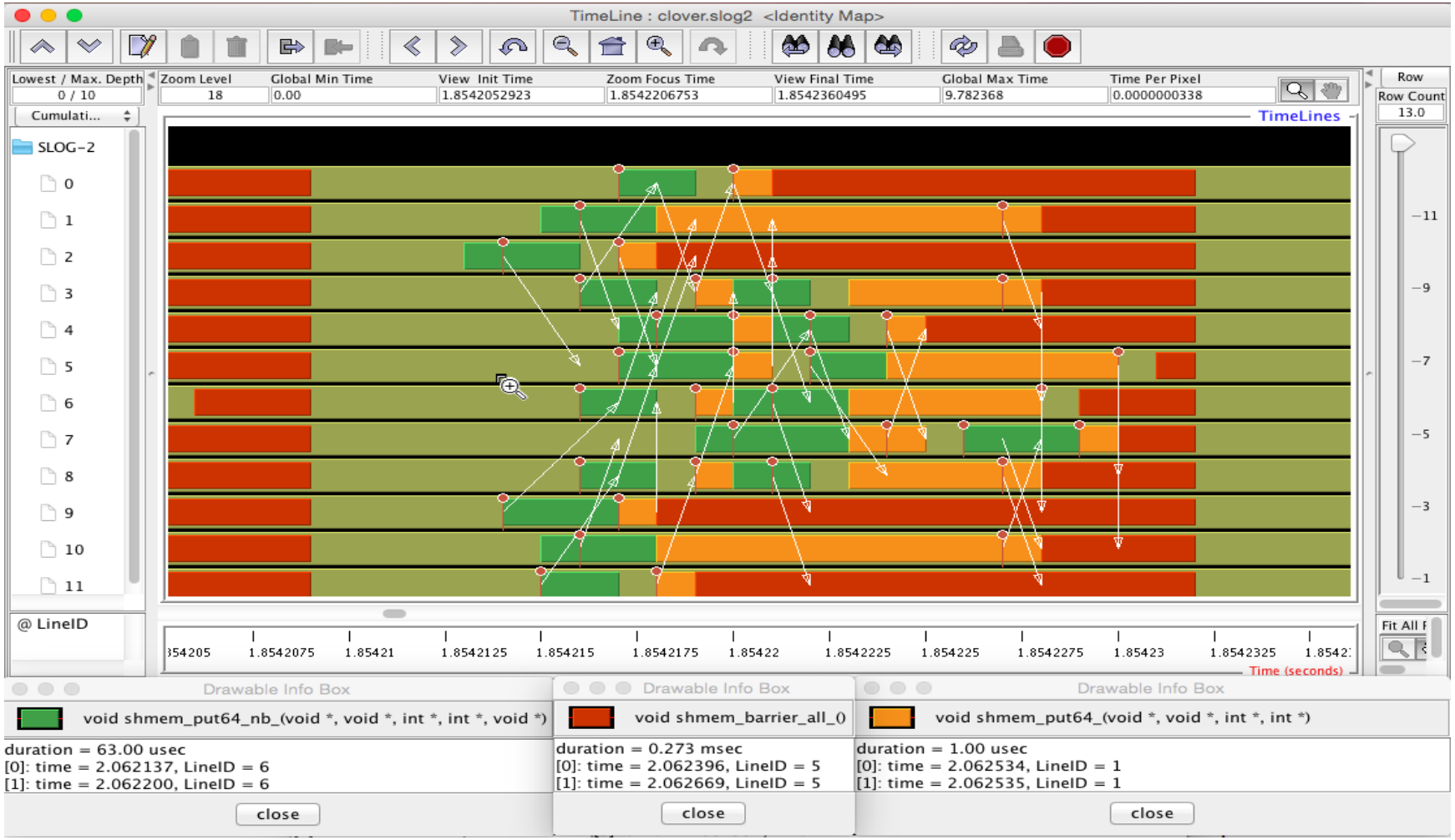
Comparing Rank 118 with 22.

Right click on “node 118” -> Add node to comparison window

ParaProf Function Window



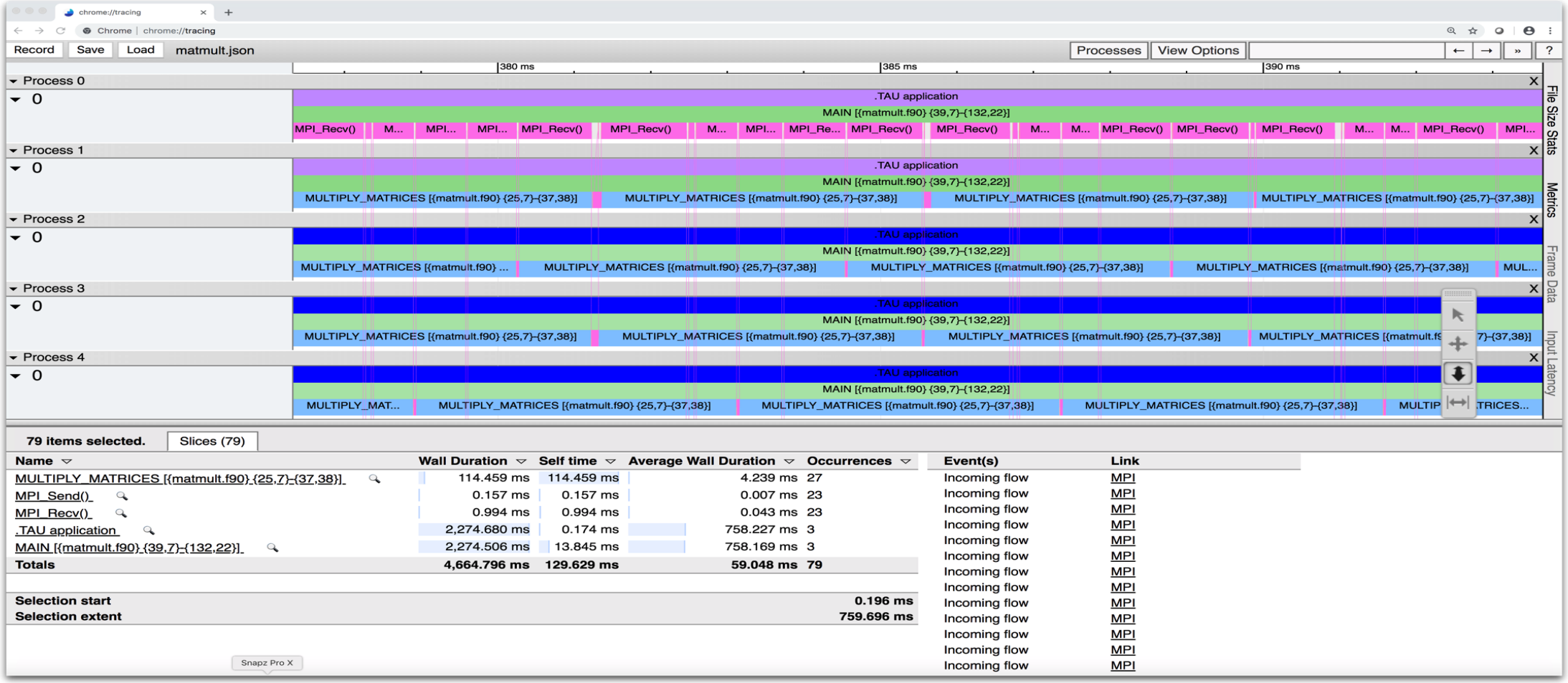
Tracing: Jumpshot (ships with TAU)



% export TAU_TRACE=1; mpirun –np 256 tau_exec ./a.out
% tau_treemerge.pl; tau2slog2 tau.trc tau.edf –o app.slog2

% jumpshot app.slog2

Tracing: Chrome Browser or Perfetto.dev

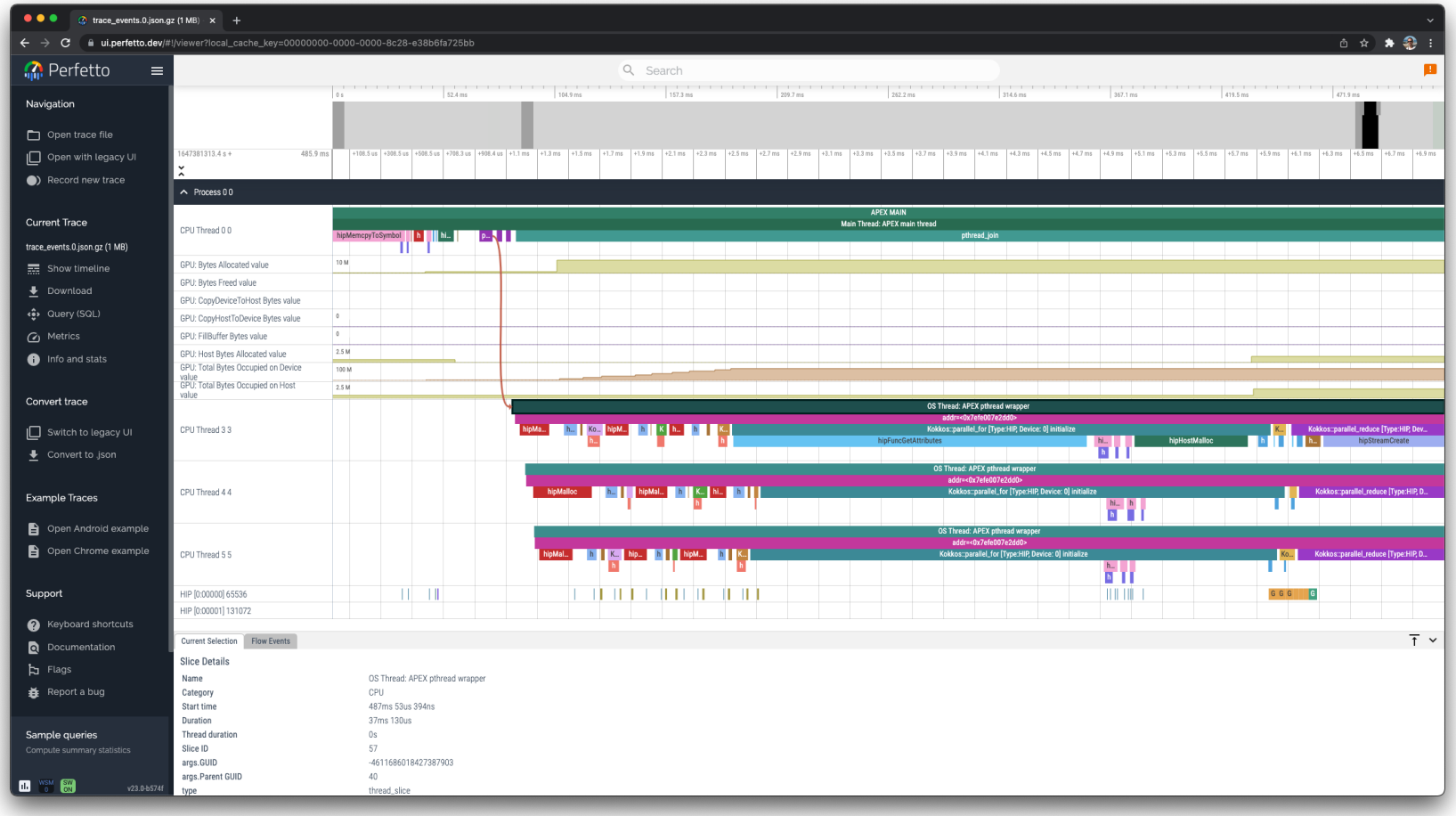


% export TAU_TRACE=1
% mpirun –np 256 tau_exec ./a.out
% tau_treemerge.pl; tau_trace2json tau.trc tau.edf –chrome –ignoreatomic –o app.json

Chrome browser: chrome://tracing (Load -> app.json)

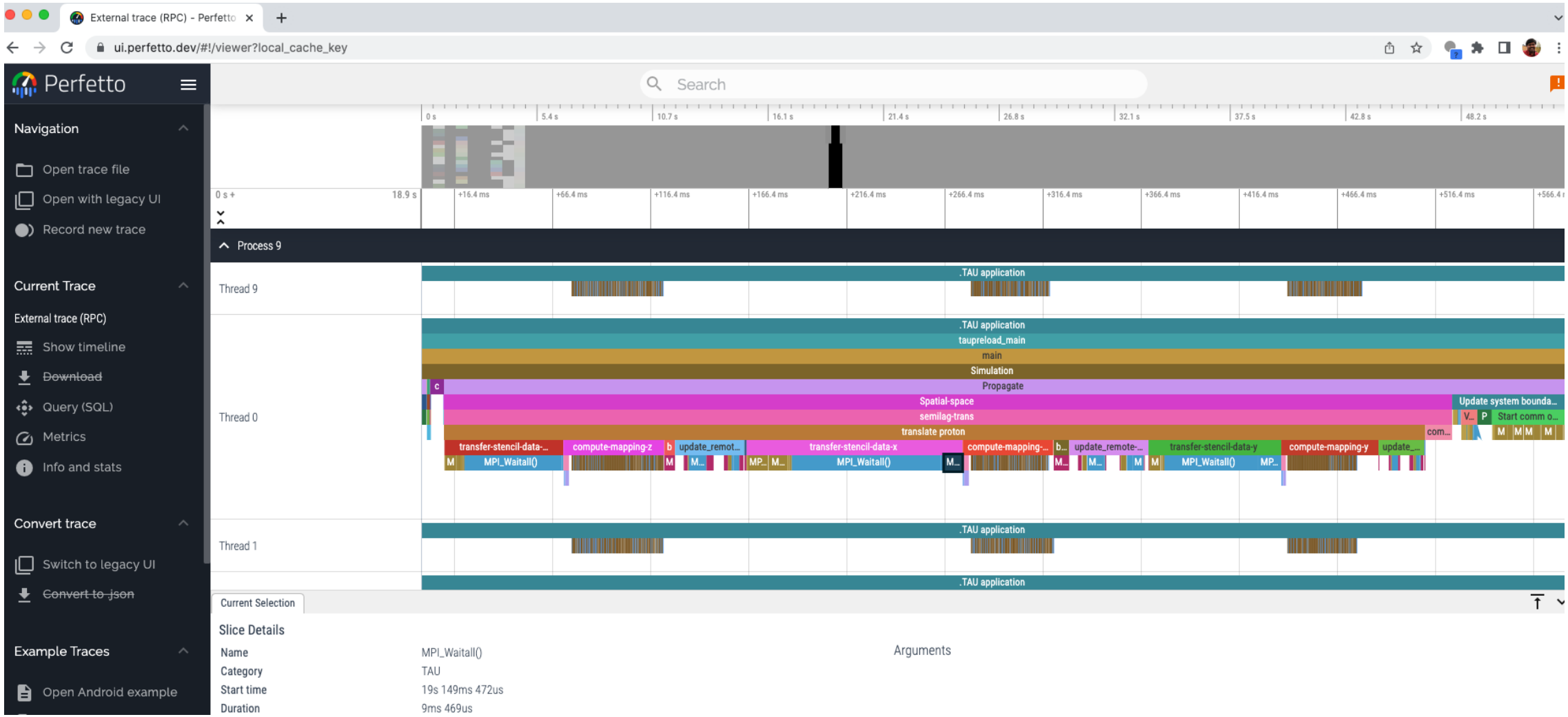
Perfetto.dev (open the UI)

Perfetto.dev Trace Browser: Kokkos

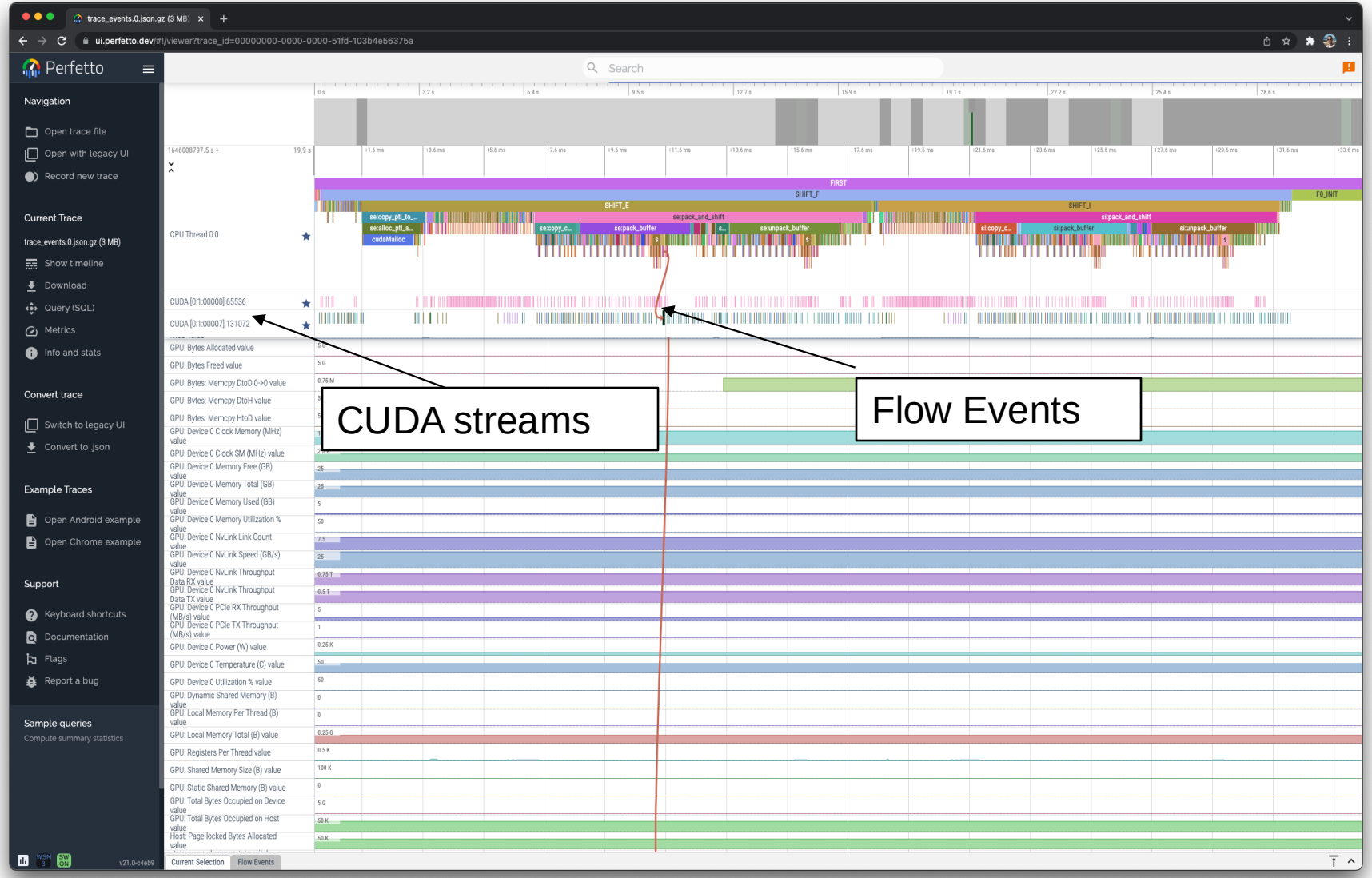



```
% export TAU_TRACE=1; mpirun -np 64 tau_exec ./a.out;
% tau_treemerge.pl;
% tau_trace2json tau.trc tau.edf -chrome -ignoreatomic -o app.json
```

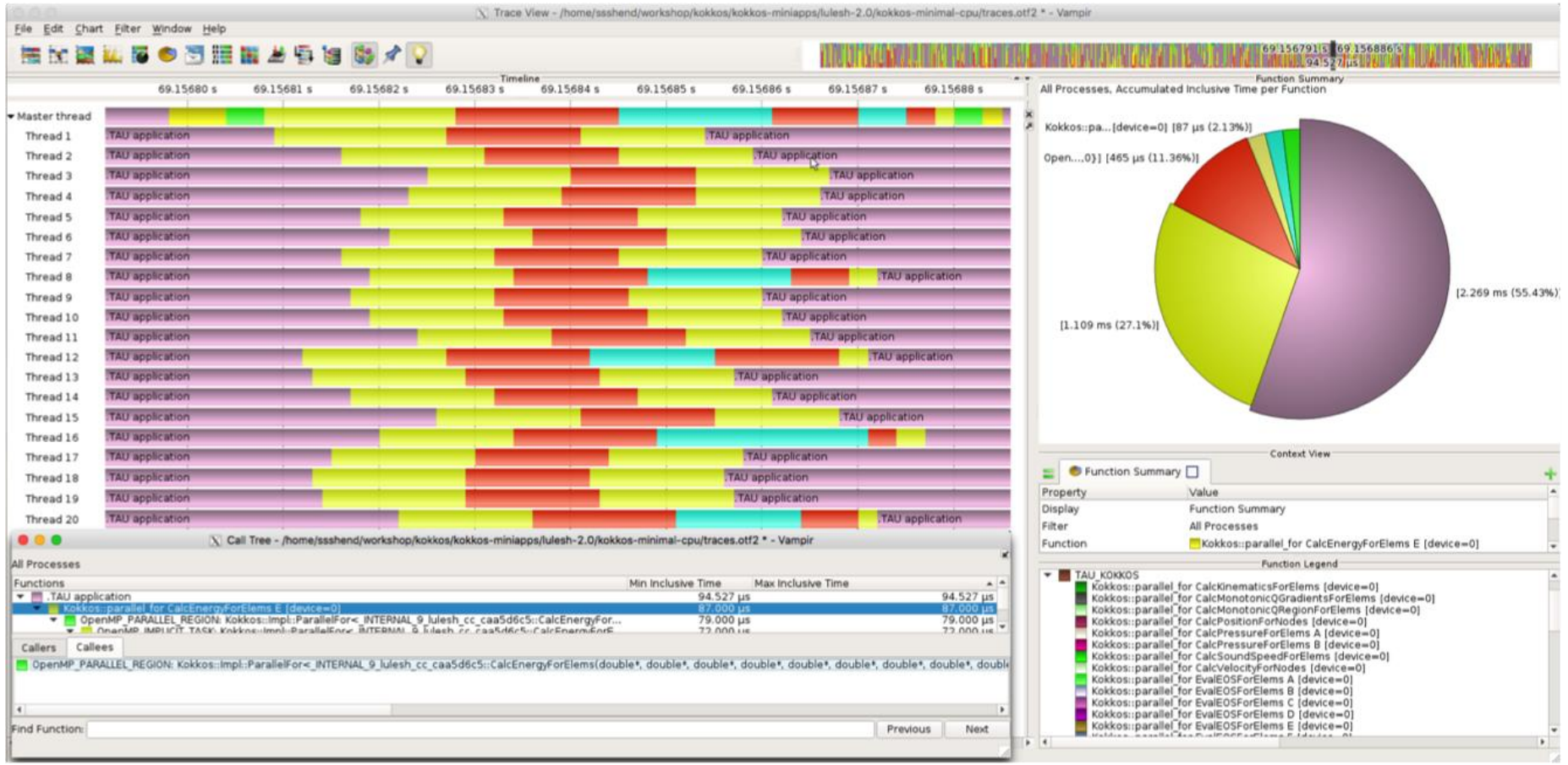
Perfetto.dev



Perfetto.dev Trace Browser

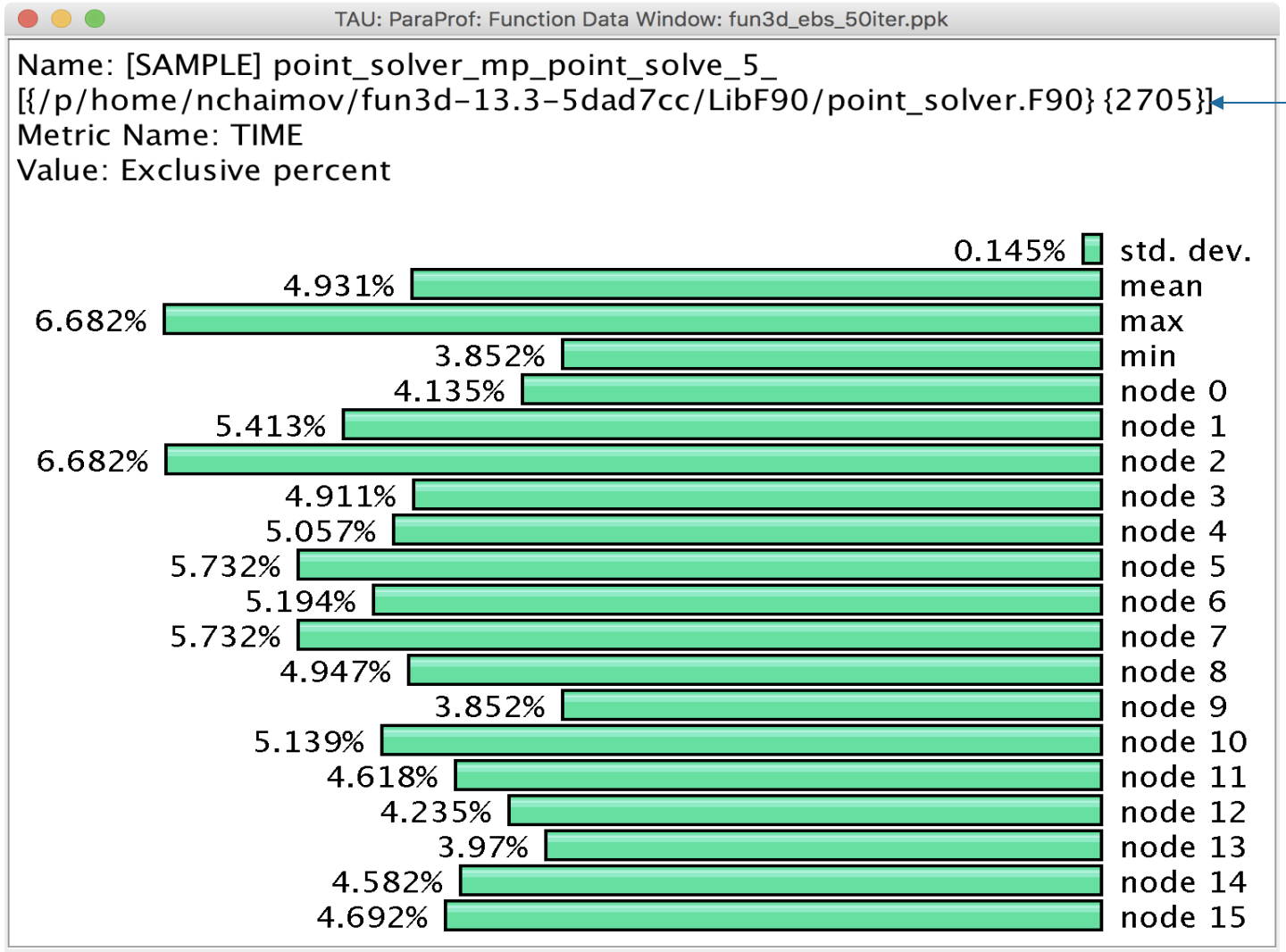


Vampir [TU Dresden] Timeline: Kokkos




```
% export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2
% tau_exec -T ompt  –ompt  ./a.out
% vampir traces.otf2 &
```

Event Based Sampling (EBS)



File: point_solver.F90
Line: 2705

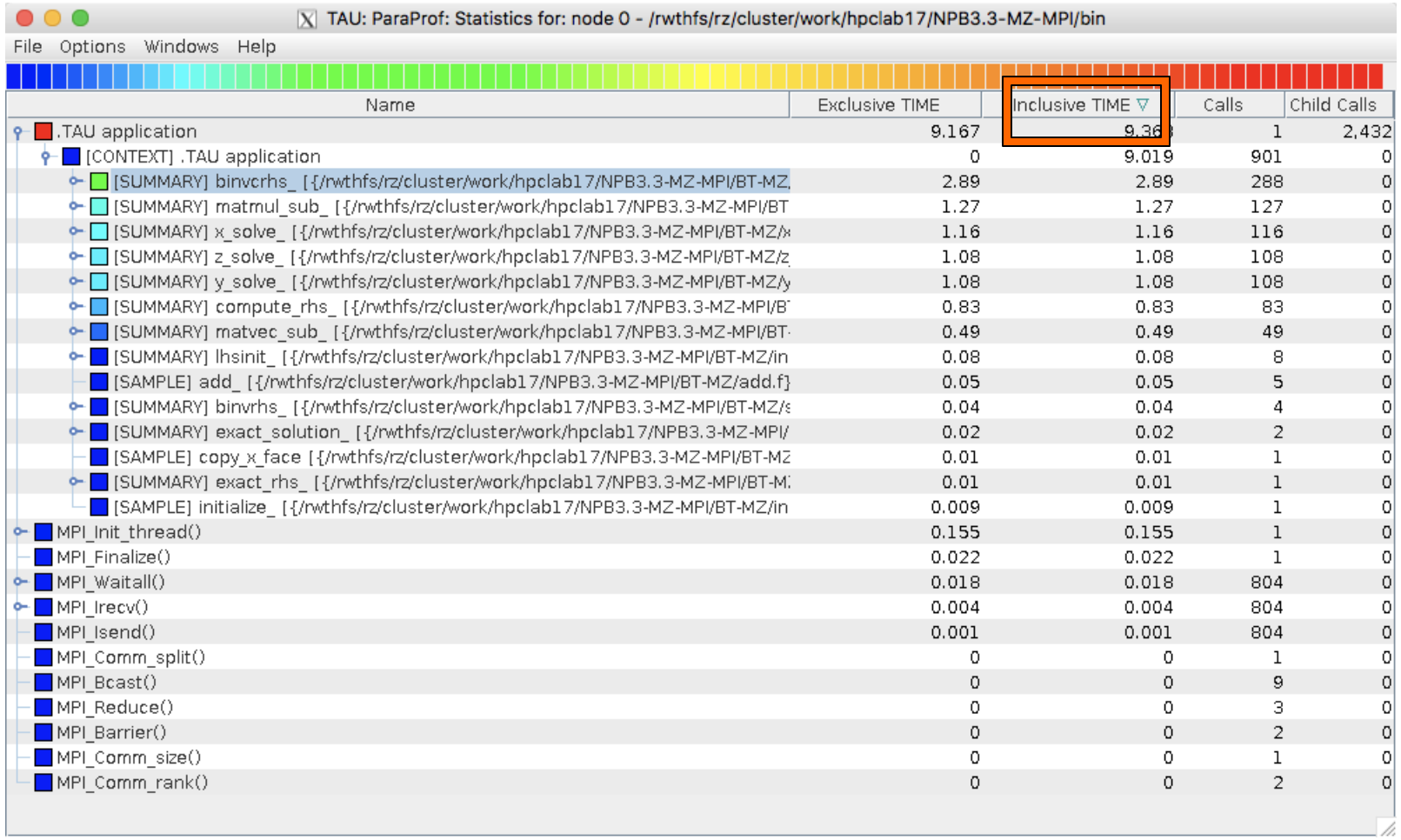
Uninstrumented!

```
% mpirun -n 16 tau_exec -ebs a.out
```

ParaProf

Click on Columns:
to sort by incl time

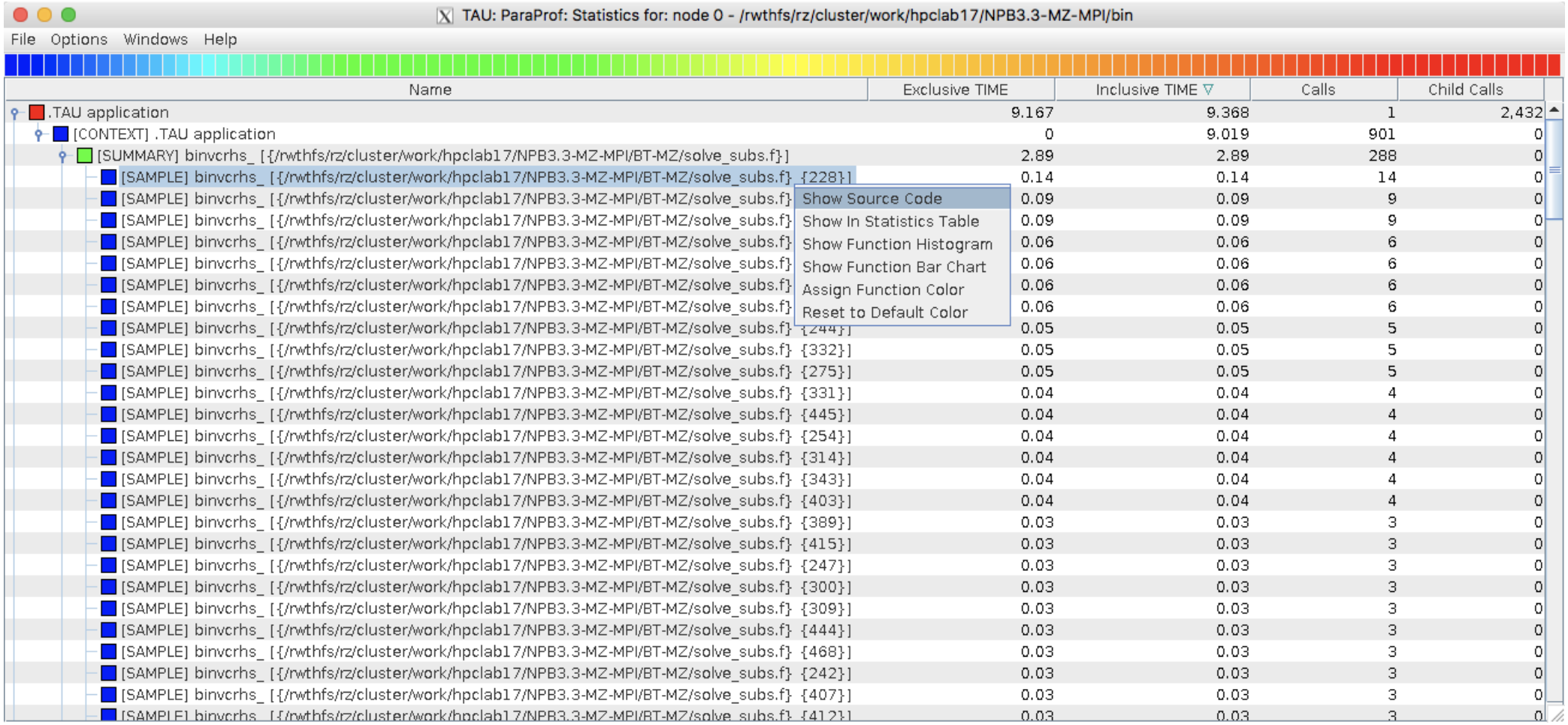
Open binvcrhs
Click on Sample



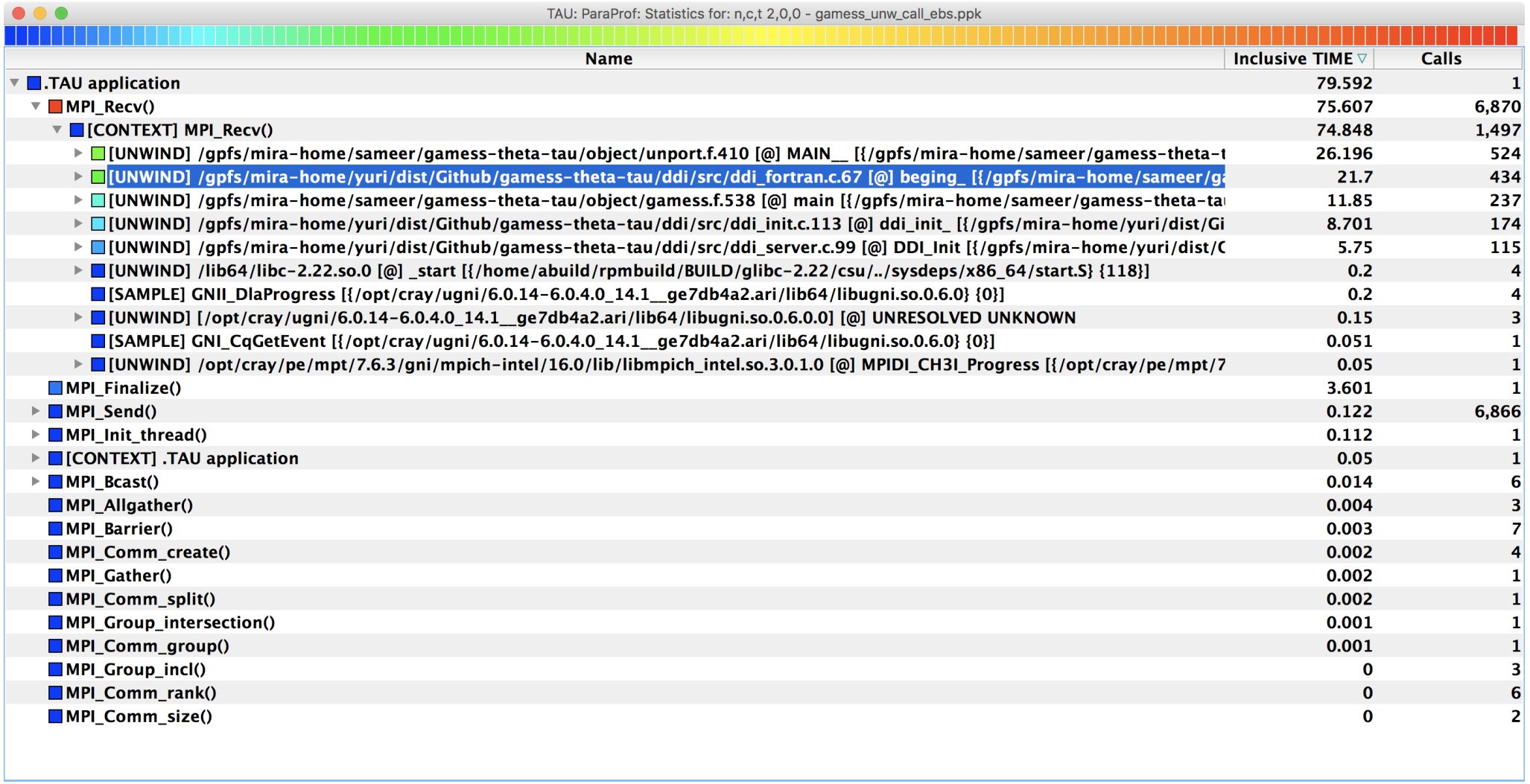
Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
.TAU application	9.167	9.36	1	2,432
[CONTEXT] .TAU application	0	9.019	901	0
[SUMMARY] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/	2.89	2.89	288	0
[SUMMARY] matmul_sub_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT	1.27	1.27	127	0
[SUMMARY] x_solve_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/x	1.16	1.16	116	0
[SUMMARY] z_solve_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/z	1.08	1.08	108	0
[SUMMARY] y_solve_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/y	1.08	1.08	108	0
[SUMMARY] compute_rhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/B	0.83	0.83	83	0
[SUMMARY] matvec_sub_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-	0.49	0.49	49	0
[SUMMARY] lhsinit_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/in	0.08	0.08	8	0
[SAMPLE] add_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/add.f}	0.05	0.05	5	0
[SUMMARY] binvrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/s	0.04	0.04	4	0
[SUMMARY] exact_solution_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/	0.02	0.02	2	0
[SAMPLE] copy_x_face [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ	0.01	0.01	1	0
[SUMMARY] exact_rhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-M	0.01	0.01	1	0
[SAMPLE] initialize_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/in	0.009	0.009	1	0
MPI_Init_thread()	0.155	0.155	1	0
MPI_Finalize()	0.022	0.022	1	0
MPI_Waitall()	0.018	0.018	804	0
MPI_Irecv()	0.004	0.004	804	0
MPI_Isend()	0.001	0.001	804	0
MPI_Comm_split()	0	0	1	0
MPI_Bcast()	0	0	9	0
MPI_Reduce()	0	0	3	0
MPI_Barrier()	0	0	2	0
MPI_Comm_size()	0	0	1	0
MPI_Comm_rank()	0	0	2	0

% export TAU_SAMPLING=1

ParaProf


TAU: ParaProf: Statistics for: node 0 - /rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/bin
File Options Windows Help
Name
Exclusive TIME
Inclusive TIME
Calls
Child Calls
.TAU application 9.167 9.368 1 2,432
[CONTEXT] .TAU application 0 9.019 901 0
[SUMMARY] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}] 2.89 2.89 288 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {228}] 0.14 0.14 14 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.09 0.09 9 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.09 0.09 9 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.06 0.06 6 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.06 0.06 6 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.06 0.06 6 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.06 0.06 6 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} 0.05 0.05 5 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {332}] 0.05 0.05 5 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {275}] 0.05 0.05 5 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {331}] 0.04 0.04 4 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {445}] 0.04 0.04 4 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {254}] 0.04 0.04 4 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {314}] 0.04 0.04 4 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {343}] 0.04 0.04 4 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {403}] 0.04 0.04 4 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {389}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {415}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {247}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {300}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {309}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {444}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {468}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {242}] 0.03 0.03 3 0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {407}] 0.03 0.03 3 0
Show Source Code
Show In Statistics Table
Show Function Histogram
Show Function Bar Chart
Assign Function Color
Reset to Default Color

Callstack Sampling in TAU



Name	Inclusive TIME ▽	Calls
.TAU application	79.592	1
MPI_Recv()	75.607	6,870
[CONTEXT] MPI_Recv()	74.848	1,497
[UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410 [@] MAIN__ [{/gpfs/mira-home/sameer/gamess-theta-t	26.196	524
[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_fortran.c.67 [@] beging_ [{/gpfs/mira-home/sameer/g	21.7	434
[UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/gamess.f.538 [@] main [{/gpfs/mira-home/sameer/gamess-theta-ta	11.85	237
[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_init.c.113 [@] ddi_init_ [{/gpfs/mira-home/yuri/dist/Gi	8.701	174
[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_server.c.99 [@] DDI_Init [{/gpfs/mira-home/yuri/dist/C	5.75	115
[UNWIND] /lib64/libc-2.22.so.0 [@] _start [{/home/abuild/rpmbuild/BUILD/glibc-2.22/csu/../sysdeps/x86_64/start.S} {118}]	0.2	4
[SAMPLE] GNII_DlaProgress [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1__ge7db4a2.ari/lib64/libugni.so.0.6.0} {0}]	0.2	4
[UNWIND] [/opt/cray/ugni/6.0.14-6.0.4.0_14.1__ge7db4a2.ari/lib64/libugni.so.0.6.0.0] [@] UNRESOLVED UNKNOWN	0.15	3
[SAMPLE] GNI_CqGetEvent [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1__ge7db4a2.ari/lib64/libugni.so.0.6.0} {0}]	0.051	1
[UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] MPIDI_CH3I_Progress [{/opt/cray/pe/mpt/7	0.05	1
MPI_Finalize()	3.601	1
MPI_Send()	0.122	6,866
MPI_Init_thread()	0.112	1
[CONTEXT] .TAU application	0.05	1
MPI_Bcast()	0.014	6
MPI_Allgather()	0.004	3
MPI_Barrier()	0.003	7
MPI_Comm_create()	0.002	4
MPI_Gather()	0.002	1
MPI_Comm_split()	0.002	1
MPI_Group_intersection()	0.001	1
MPI_Comm_group()	0.001	1
MPI_Group_incl()	0	3
MPI_Comm_rank()	0	6
MPI_Comm_size()	0	2

% export TAU_SAMPLING=1; export TAU_EBS_UNWIND=1

TAU – Callstack Sampling



Name	Inclusive...	Calls
.TAU application	34.979	1
[CONTEXT] .TAU application	31.647	632
void shmem_barrier_all_()	1.219	46,029
[CONTEXT] void shmem_barrier_all_()	1.599	32
[UNWIND] [/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90.41] [@] UNRESOLVED /lib64/libc-2.11.3.so	1.599	32
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.62 [@] main [{/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90} {41}]	0.85	17
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/advection.f90.102 [@] hydro_ [{/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90} {62}]	0.55	11
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/update_halo.f90.36 [@] __advection_module_MOD_advection [{/home/ssshend/CloverLeaf_Ope	0.55	11
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.292 [@] __update_halo_module_MOD_update_halo [{/home/ssshend/CloverLeaf_O	0.5	10
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.572 [@] __clover_module_MOD_clover_exchange [{/home/ssshend/CloverLeaf_(	0.5	10
[UNWIND] UNRESOLVED [@] __clover_module_MOD_clover_exchange_message [{/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90} {572}]	0.5	10
[UNWIND] [/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_barrier.c.118] [@] UNRESOLVED /nfsproje	0.45	9
[SAMPLE] _smai_smp_barrier_in [{/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_barrier.c} {118}]	0.45	9
[UNWIND] [/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_internal.h.88] [@] UNRESOLVED /nfsprojects/	0.05	1
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.461 [@] __update_halo_module_MOD_update_halo [{/home/ssshend/CloverLeaf_O	0.05	1
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/advection.f90.72 [@] hydro_ [{/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90} {62}]	0.15	3
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/advection.f90.55 [@] hydro_ [{/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90} {62}]	0.15	3
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.52 [@] main [{/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90} {41}]	0.5	10
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.54 [@] main [{/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90} {41}]	0.25	5
void start_pes_(int *)	0.508	1
void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.325	2,000
[CONTEXT] void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.5	10
[UNWIND] [/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90.41] [@] UNRESOLVED /lib64/libc-2.11.3.so	0.5	10
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.58 [@] main [{/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90} {41}]	0.45	9
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/PdV.f90.107 [@] hydro_ [{/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90} {58}]	0.45	9
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.740 [@] __pdv_module_MOD_pdv [{/home/ssshend/CloverLeaf_OpenSHMEM/PdV.f9(	0.45	9
[UNWIND] UNRESOLVED [@] __clover_module_MOD_clover_check_error [{/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90} {740}]	0.45	9
[UNWIND] [/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_reduction.h.207] [@] UNRESOLVED /nfsprojects/vol	0.45	9
[UNWIND] /notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_reduction.h.788 [@] pshmem_double_max_tc	0.45	9
[UNWIND] /notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_reduction.h.107 [@] _smai_opt_double_ma	0.45	9
[SAMPLE] _smai_smp_reduce_double_max [{/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_reduc	0.45	9
[UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.54 [@] main [{/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90} {41}]	0.05	1

% export TAU_SAMPLING=1; export TAU_EBS_UNWIND=1

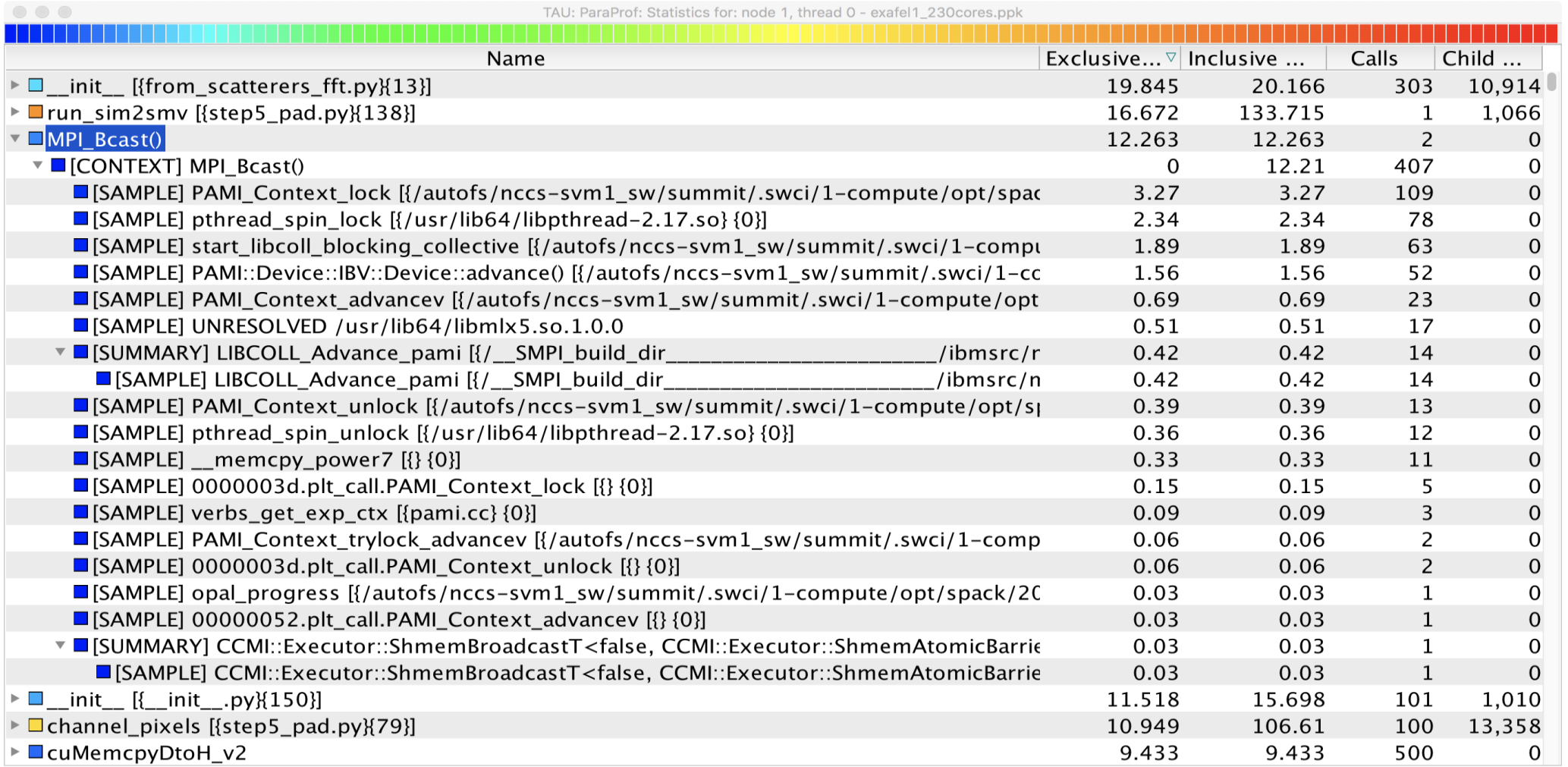
TAU Context Event Window

TAU: ParaProf: Context Events for: node 0, thread 0 - exafel1_230cores.ppk

Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
<module> [{step5_batch.py}{1}]						
tst_one [{step5_batch.py}{23}]						
run_sim2smv [{step5_pad.py}{138}]						
channel_pixels [{step5_pad.py}{79}]						
cudaMemcpy						
Bytes copied from Device to Host	15,300,000,000	500	36,000,000	9,000,000	30,600,000	10,800,000
Bytes copied from Host to Device	15,423,816,000	2,300	36,000,000	8	6,706,006.957	13,564,989.185
cuMemcpyHtoD_v2						
Bytes copied from Host to Device	15,423,816,000	2,300	36,000,000	8	6,706,006.957	13,564,989.185
cuMemcpyDtoH_v2						
Bytes copied from Device to Host	15,300,000,000	500	36,000,000	9,000,000	30,600,000	10,800,000
Bytes copied from Device to Host	30,600,000,000	1,000	36,000,000	9,000,000	30,600,000	10,800,000
Bytes copied from Host to Device	30,847,632,000	4,600	36,000,000	8	6,706,006.957	13,564,989.185
Message size for broadcast	827,971,798	2	827,971,794	4	413,985,899	413,985,895

TAU tracks the data transfers between the host and the GPU.

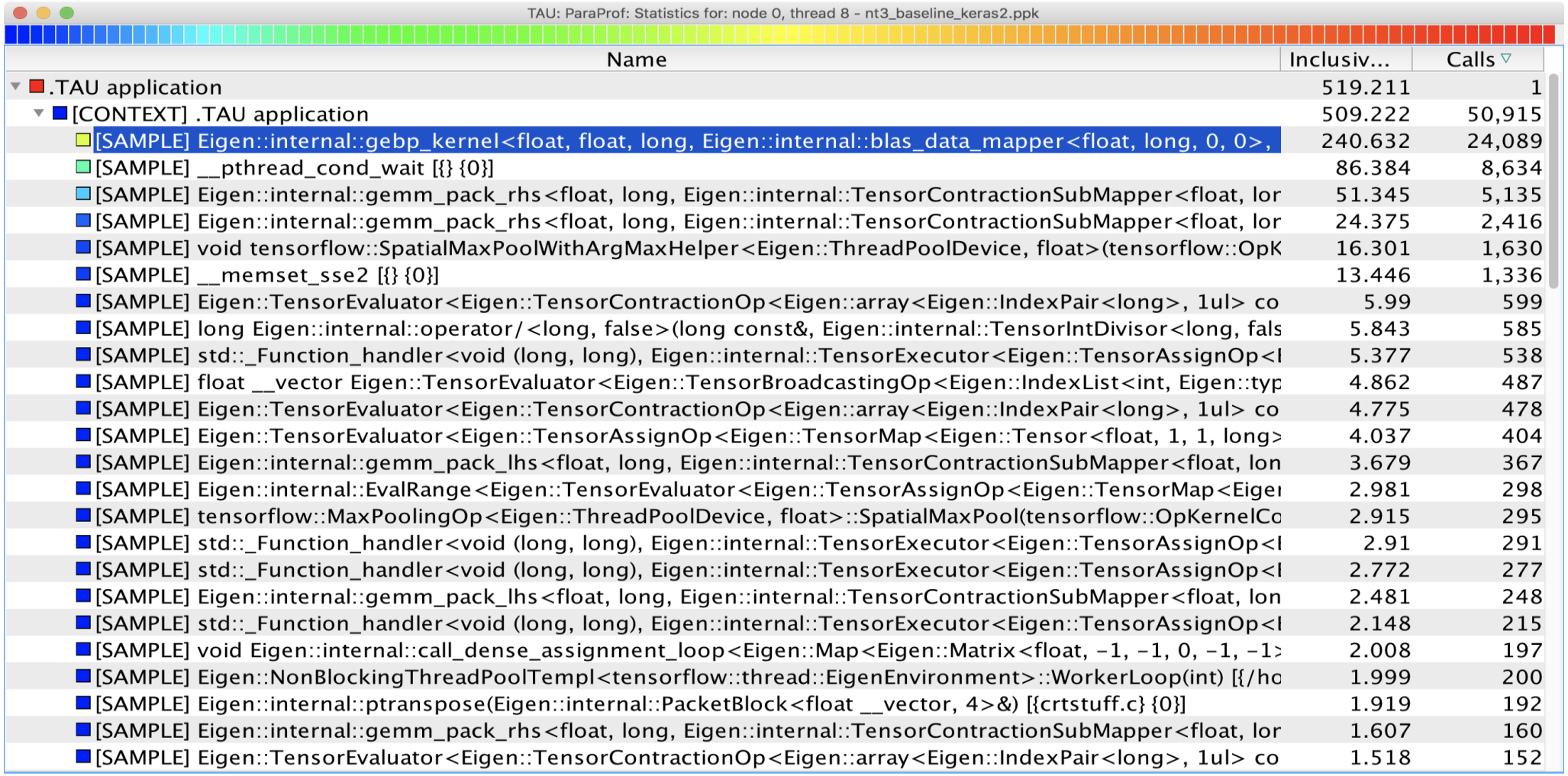
TAU's tracking of Python and MPI



Name	Exclusive...	Inclusive ...	Calls	Child ...
__init__ [{from_scatterers_fft.py}{13}]	19.845	20.166	303	10,914
run_sim2smv [{step5_pad.py}{138}]	16.672	133.715	1	1,066
MPI_Bcast()	12.263	12.263	2	0
[CONTEXT] MPI_Bcast()	0	12.21	407	0
[SAMPLE] PAMI_Context_lock [{/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spac	3.27	3.27	109	0
[SAMPLE] pthread_spin_lock [{/usr/lib64/libpthread-2.17.so}{0}]	2.34	2.34	78	0
[SAMPLE] start_libcoll_blocking_collective [{/autofs/nccs-svm1_sw/summit/.swci/1-compu	1.89	1.89	63	0
[SAMPLE] PAMI::Device::IBV::Device::advance() [{/autofs/nccs-svm1_sw/summit/.swci/1-cc	1.56	1.56	52	0
[SAMPLE] PAMI_Context_advancev [{/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt	0.69	0.69	23	0
[SAMPLE] UNRESOLVED /usr/lib64/libmlx5.so.1.0.0	0.51	0.51	17	0
[SUMMARY] LIBCOLL_Advance_pami [{/__SMPI_build_dir_____/ibmsrc/r	0.42	0.42	14	0
[SAMPLE] LIBCOLL_Advance_pami [{/__SMPI_build_dir_____/ibmsrc/n	0.42	0.42	14	0
[SAMPLE] PAMI_Context_unlock [{/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/s	0.39	0.39	13	0
[SAMPLE] pthread_spin_unlock [{/usr/lib64/libpthread-2.17.so}{0}]	0.36	0.36	12	0
[SAMPLE] __memcpy_power7 [{}{0}]	0.33	0.33	11	0
[SAMPLE] 0000003d.plt_call.PAMI_Context_lock [{}{0}]	0.15	0.15	5	0
[SAMPLE] verbs_get_exp_ctx [{pami.cc}{0}]	0.09	0.09	3	0
[SAMPLE] PAMI_Context_trylock_advancev [{/autofs/nccs-svm1_sw/summit/.swci/1-comp	0.06	0.06	2	0
[SAMPLE] 0000003d.plt_call.PAMI_Context_unlock [{}{0}]	0.06	0.06	2	0
[SAMPLE] opal_progress [{/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/2C	0.03	0.03	1	0
[SAMPLE] 00000052.plt_call.PAMI_Context_advancev [{}{0}]	0.03	0.03	1	0
[SUMMARY] CCMI::Executor::ShmemBroadcastT<false, CCMI::Executor::ShmemAtomicBarrie	0.03	0.03	1	0
[SAMPLE] CCMI::Executor::ShmemBroadcastT<false, CCMI::Executor::ShmemAtomicBarrie	0.03	0.03	1	0
__init__ [{__init__.py}{150}]	11.518	15.698	101	1,010
channel_pixels [{step5_pad.py}{79}]	10.949	106.61	100	13,358
cuMemcpyDtoH_v2	9.433	9.433	500	0

TAU can observe events in closed-source vendor libraries (e.g., in MPI_Bcast)!

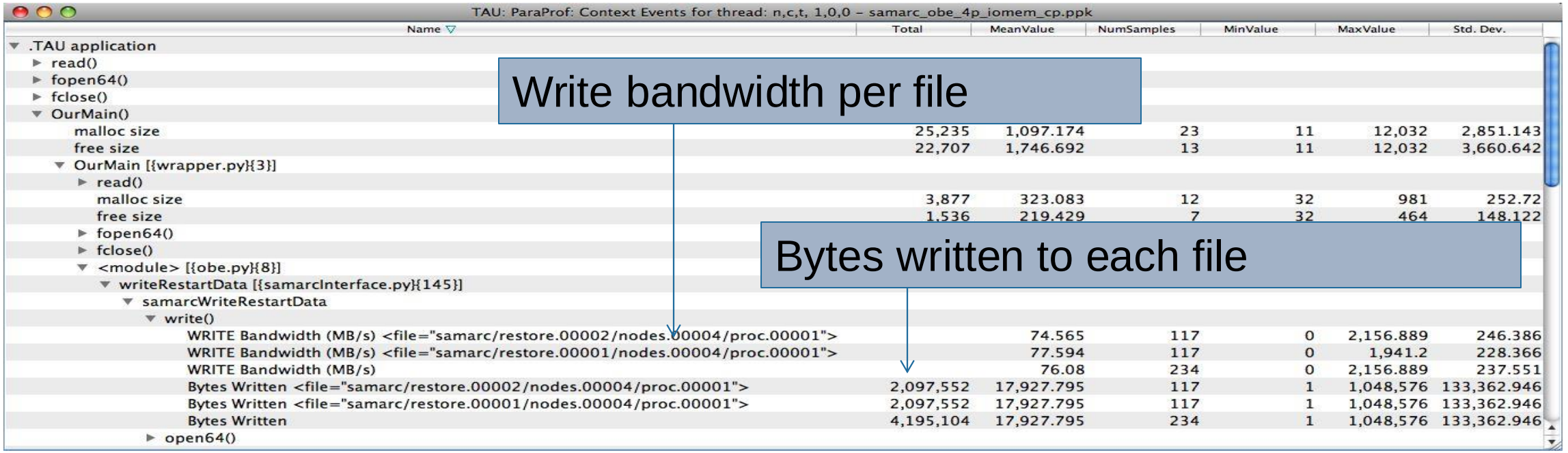
Deep Learning: Tensorflow



Name	Inclusiv...	Calls
.TAU application	519.211	1
[CONTEXT] .TAU application	509.222	50,915
[SAMPLE] Eigen::internal::gebp_kernel<float, float, long, Eigen::internal::blas_data_mapper<float, long, 0, 0>,	240.632	24,089
[SAMPLE] __pthread_cond_wait [{} {0}]	86.384	8,634
[SAMPLE] Eigen::internal::gemm_pack_rhs<float, long, Eigen::internal::TensorContractionSubMapper<float, lor	51.345	5,135
[SAMPLE] Eigen::internal::gemm_pack_rhs<float, long, Eigen::internal::TensorContractionSubMapper<float, lor	24.375	2,416
[SAMPLE] void tensorflow::SpatialMaxPoolWithArgMaxHelper<Eigen::ThreadPoolDevice, float>(tensorflow::OpK	16.301	1,630
[SAMPLE] __memset_sse2 [{} {0}]	13.446	1,336
[SAMPLE] Eigen::TensorEvaluator<Eigen::TensorContractionOp<Eigen::array<Eigen::IndexPair<long>, 1ul> co	5.99	599
[SAMPLE] long Eigen::internal::operator/<long, false>(long const&, Eigen::internal::TensorIntDivisor<long, fals	5.843	585
[SAMPLE] std::_Function_handler<void (long, long), Eigen::internal::TensorExecutor<Eigen::TensorAssignOp<I	5.377	538
[SAMPLE] float __vector Eigen::TensorEvaluator<Eigen::TensorBroadcastingOp<Eigen::IndexList<int, Eigen::typ	4.862	487
[SAMPLE] Eigen::TensorEvaluator<Eigen::TensorContractionOp<Eigen::array<Eigen::IndexPair<long>, 1ul> co	4.775	478
[SAMPLE] Eigen::TensorEvaluator<Eigen::TensorAssignOp<Eigen::TensorMap<Eigen::Tensor<float, 1, 1, long>	4.037	404
[SAMPLE] Eigen::internal::gemm_pack_lhs<float, long, Eigen::internal::TensorContractionSubMapper<float, lon	3.679	367
[SAMPLE] Eigen::internal::EvalRange<Eigen::TensorEvaluator<Eigen::TensorAssignOp<Eigen::TensorMap<Eiger	2.981	298
[SAMPLE] tensorflow::MaxPoolingOp<Eigen::ThreadPoolDevice, float>::SpatialMaxPool(tensorflow::OpKernelCo	2.915	295
[SAMPLE] std::_Function_handler<void (long, long), Eigen::internal::TensorExecutor<Eigen::TensorAssignOp<I	2.91	291
[SAMPLE] std::_Function_handler<void (long, long), Eigen::internal::TensorExecutor<Eigen::TensorAssignOp<I	2.772	277
[SAMPLE] Eigen::internal::gemm_pack_lhs<float, long, Eigen::internal::TensorContractionSubMapper<float, lon	2.481	248
[SAMPLE] std::_Function_handler<void (long, long), Eigen::internal::TensorExecutor<Eigen::TensorAssignOp<I	2.148	215
[SAMPLE] void Eigen::internal::call_dense_assignment_loop<Eigen::Map<Eigen::Matrix<float, -1, -1, 0, -1, -1>	2.008	197
[SAMPLE] Eigen::NonBlockingThreadPoolTempl<tensorflow::thread::EigenEnvironment>::WorkerLoop(int) [{/ho	1.999	200
[SAMPLE] Eigen::internal::ptranspose(Eigen::internal::PacketBlock<float __vector, 4>&) [{crtstuff.c} {0}]	1.919	192
[SAMPLE] Eigen::internal::gemm_pack_rhs<float, long, Eigen::internal::TensorContractionSubMapper<float, lor	1.607	160
[SAMPLE] Eigen::TensorEvaluator<Eigen::TensorContractionOp<Eigen::array<Eigen::IndexPair<long>, 1ul> co	1.518	152

% tau_python -ebs nt3_baseline_keras2.py (CANDLE)

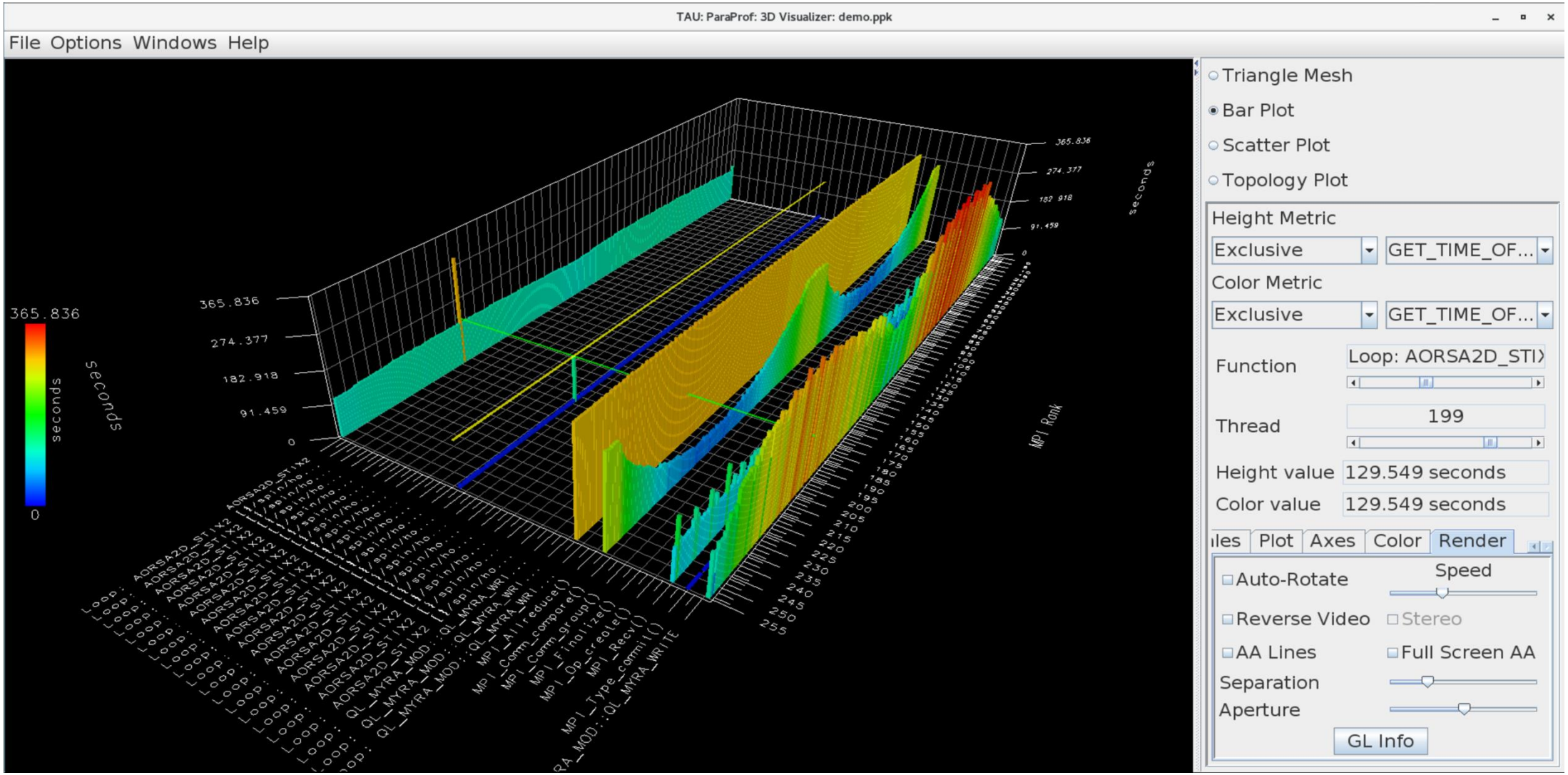
TAU – Context Events



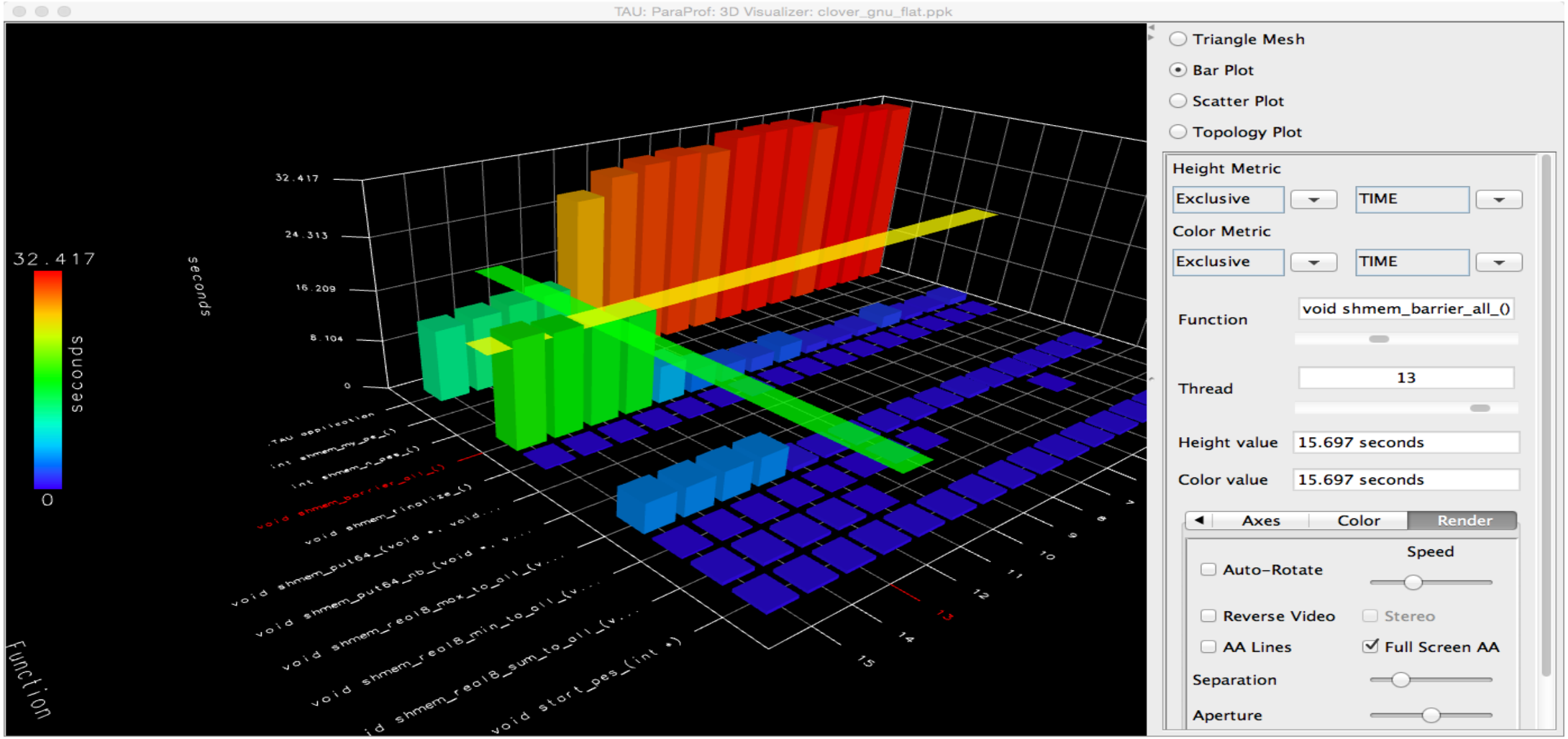
Name	Total	MeanValue	NumSamples	MinValue	MaxValue	Std. Dev.
.TAU application						
read()						
fopen64()						
fclose()						
OurMain()						
malloc size	25,235	1,097.174	23	11	12,032	2,851.143
free size	22,707	1,746.692	13	11	12,032	3,660.642
OurMain [{wrapper.py}{3}]						
read()						
malloc size	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()						

% mpirun –np 16 tau_python -io ./foo.py

ParaProf 3D Profile Browser

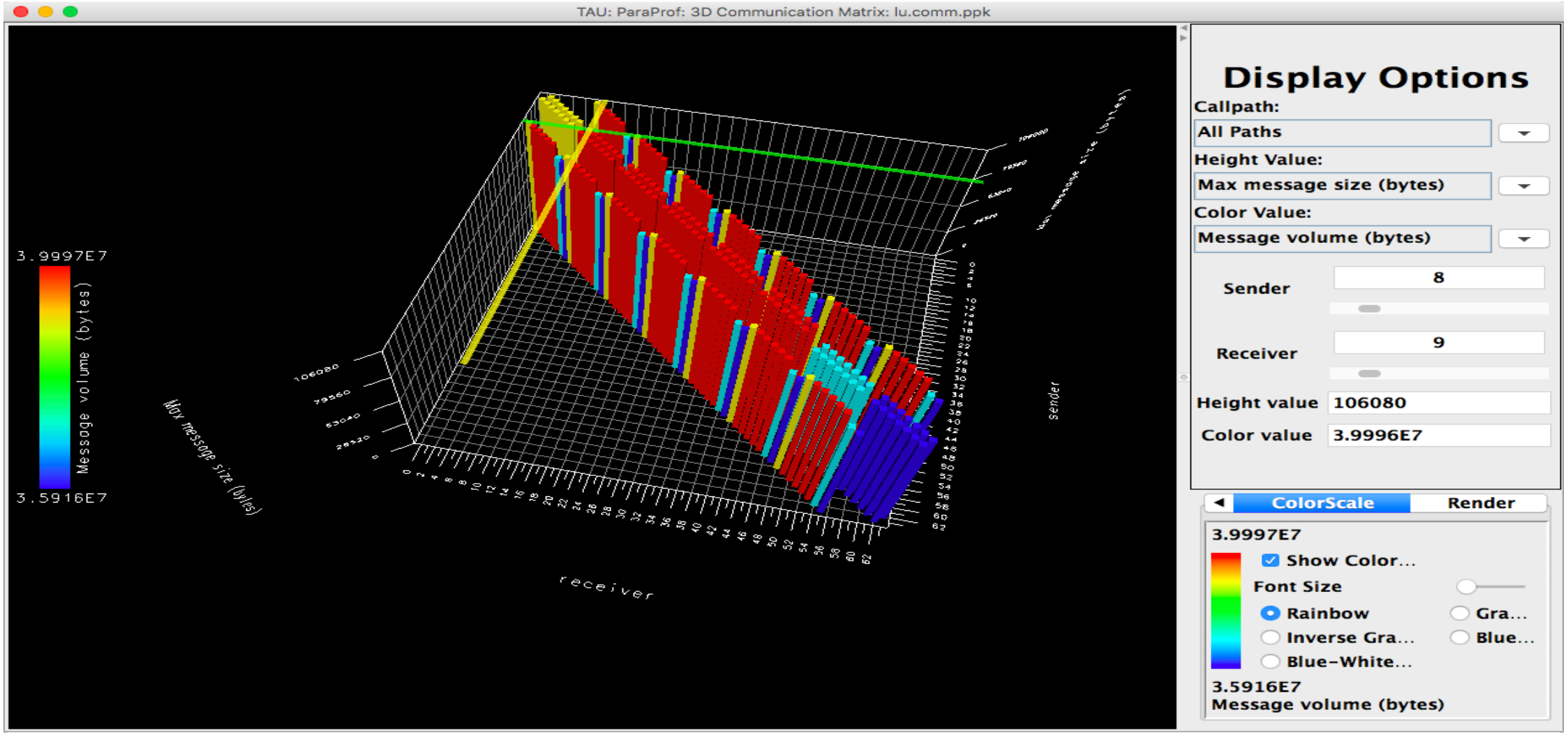


TAU – ParaProf 3D Visualization



% 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 ; Windows -> 3D Communication Matrix

TAU's Support for Runtime Systems

MPI

- PMPI profiling interface
- MPI_T tools interface using performance and control variables

Pthread

- Captures time spent in routines per thread of execution

OpenMP

- OMPT tools interface to track salient OpenMP runtime events
- Opari source rewriter
- Preloading wrapper OpenMP runtime library when OMPT is not supported

OpenACC

- OpenACC instrumentation API
- Track data transfers between host and device (per-variable)
- Track time spent in kernels

TAU's Support for Runtime Systems (contd.)

OpenCL
- OpenCL profiling interface
- Track timings of kernels

Intel® OneAPI
- Level Zero
- Track time spent in kernels executing on GPU
- Track time spent in OneAPI runtime calls

CUDA
- Cuda Profiling Tools Interface (CUPTI)
- Track data transfers between host and GPU
- Track access to uniform shared memory between host and GPU

ROCm
- Rocprofiler and Roctracer instrumentation interfaces
- Track data transfers and kernel execution between host and GPU

Kokkos
- Kokkos profiling API
- Push/pop interface for region, kernel execution interface

Python
- Python interpreter instrumentation API
- Tracks Python routine transitions as well as Python to C transitions

Examples of Multi-Level Instrumentation

MPI + OpenMP

MPI_T + PMPI + OMPT may be used to track MPI and OpenMP

MPI + HIP

PMPI + Roctracer interfaces

ROCprofiler+ MPI

ROCm Rocprofiler+ PMPI MPI interface

Kokkos + OpenMP

Kokkos profiling API + OMPT to transparently track events

Kokkos + pthread + MPI

Kokkos + pthread wrapper interposition library + PMPI layer

Python + ROCTracer + MPI

Python + ROCm Roctracer + MPI profiling interface

MPI + OpenCL

PMPI + OpenCL profiling interfaces

Binary instrumentation of libraries: Work in progress

`% tau_run a.out -o a.inst`
instruments a binary. Other flags –T <tags>, -f <selective instrumentation file>
`% tau_run -l /path/to/libhdf5.so.310 -o libhdf5.so.310`
instruments a DSO
`% tau_exec ./a.out`
executes the uninstrumented application with the instrumented shared object.

Works on x86_64. Issues with aarch64:
https://github.com/dyninst/dyninst/issues/1708 and https://github.com/dyninst/dyninst/pull/1712

To use with DyninstAPI 13 on x86_64:

1. Load spack: spack/share/spack/setup-env.sh
2. Install dyninst: spack install dyninst@13 %gcc@11
3. Configure tau with dyninst:
   3.1 spack find -p dyninst boost tbb elfutils
   3.2 Copy the paths for each package into the configure line

3.3 ./configure -bfd=download -dyninst=<dir> -tbb=<dir> -boost=<dir> -elf=<dir>; <set paths>; make install

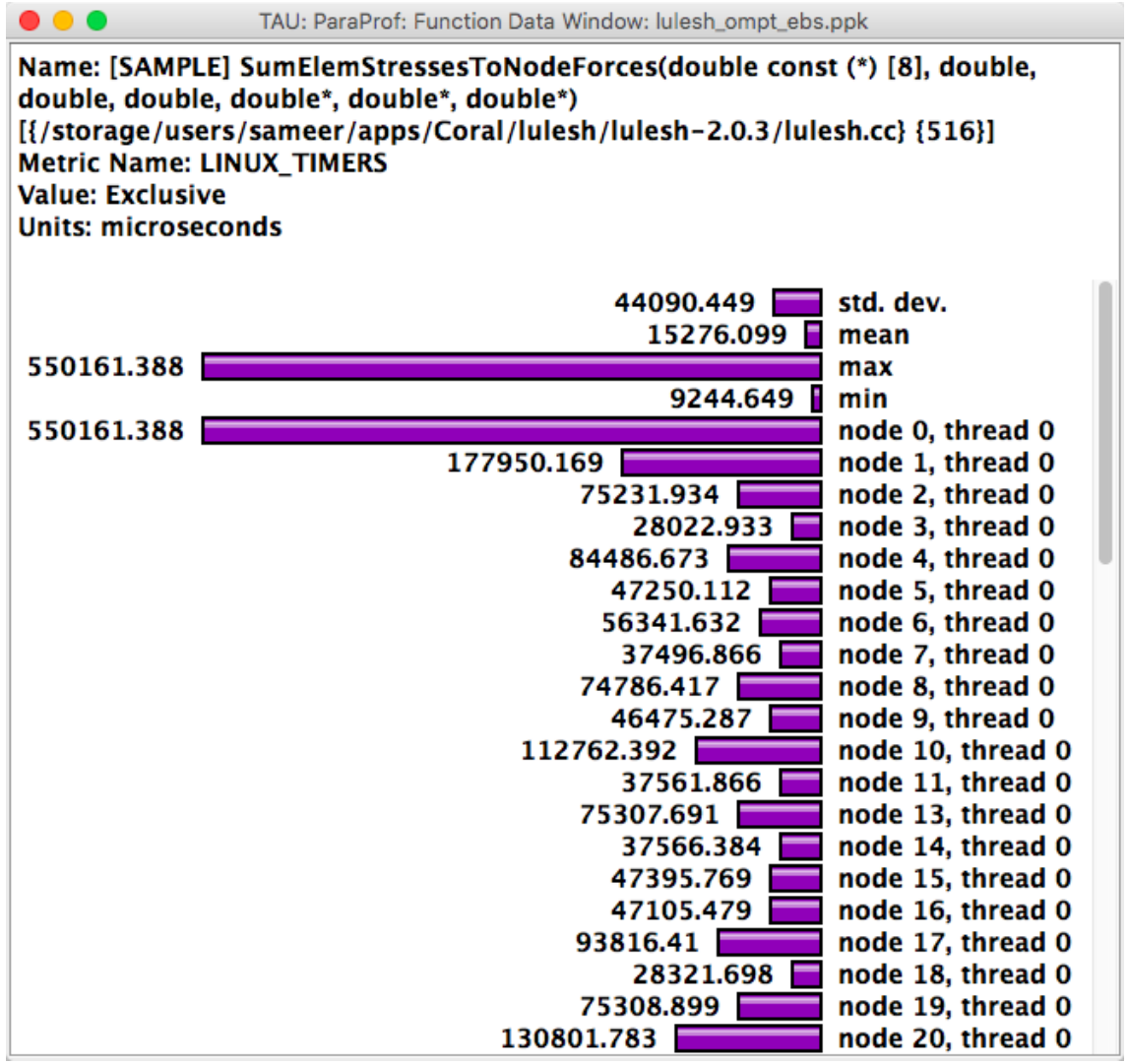
Binary instrumentation of libraries: HDF5




```
$ pprof
Reading Profile files in profile.*

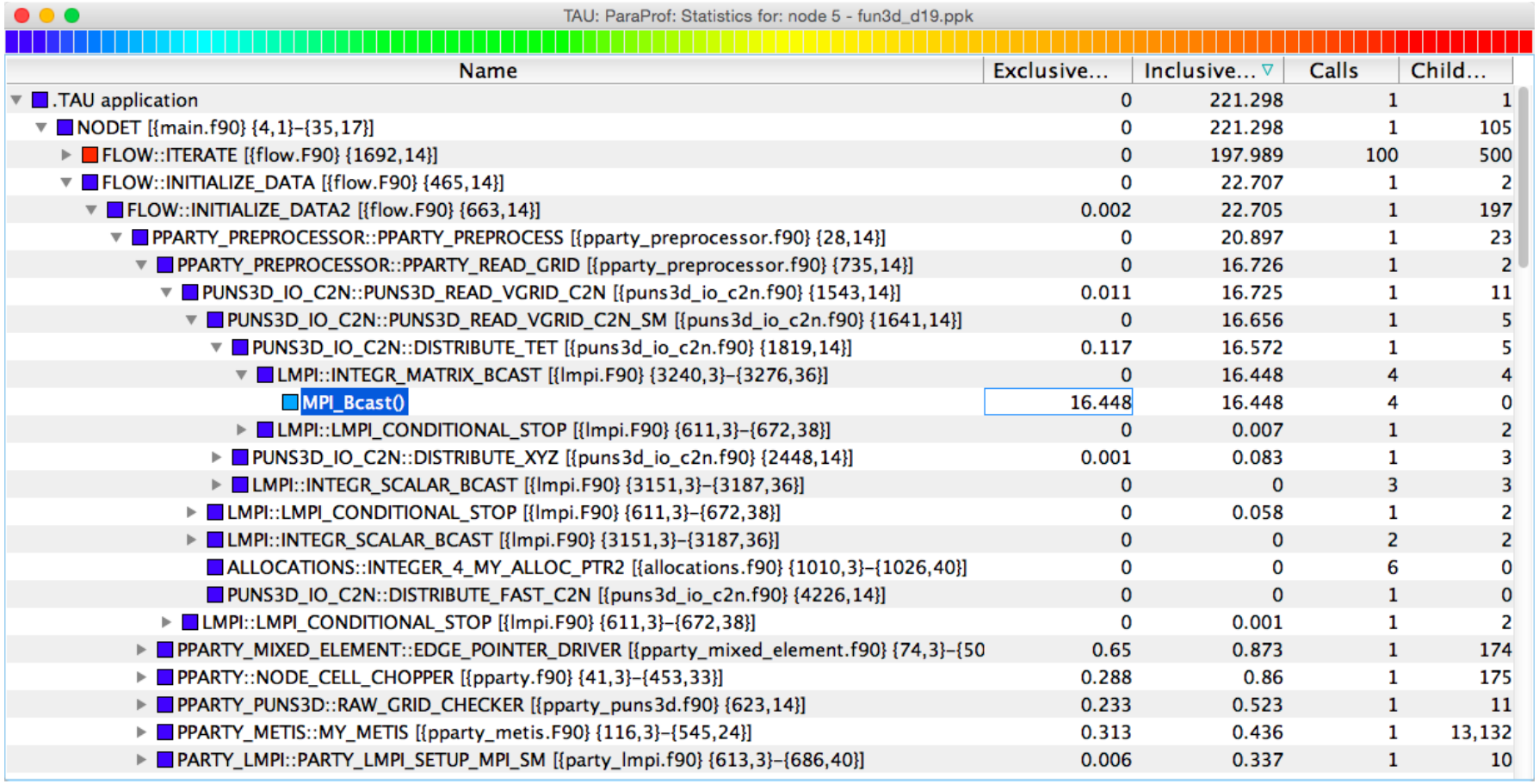
NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.272           68           1           1     68245 .TAU application
 99.6            1           67           1          26     67973 taupreload_main
 65.8        0.008           44           6           1      7484 H5open
 65.8            6           44           2          14     22448 H5_init_library
 36.0            4           24           1          12     24563 H5VL_init_phase2
 27.8            1           18           1         319     18943 H5T_init
 19.8        0.193           13         179         179        76 H5T__register_int
 19.5        0.302           13         179         310        74 H5T__register
 19.0            4           12         155        2555        84 H5T__path_find_real
 13.0            2            8           1          79      8857 H5P_init_phase1
 12.7        0.663            8           2          51      4349 H5F_open
 11.2        0.348            7           1           6      7610 H5Fcreate
 10.5        0.386            7           1           6      7138 H5F__create_api_common
  9.8        0.406            6           1           2      6707 H5VL_file_create
  9.2        0.005            6           1           1      6299 H5VL__native_file_create
  7.1            1            4         488         976        10 H5T_copy
  6.5            1            4           1         363      4452 H5E_init
  5.6        0.013            3           4          12       956 H5I_dec_app_ref
  5.6        0.013            3           2          10      1896 H5Fclose
  5.5        0.009            3           2           4      1878 H5F__close_cb
  5.5         0.01            3           2           6      1868 H5VL_file_close
  5.4        0.013            3           2           4      1852 H5VL__native_file_close
  5.4        0.019            3           4           8       924 H5F_try_close.localalias
```

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

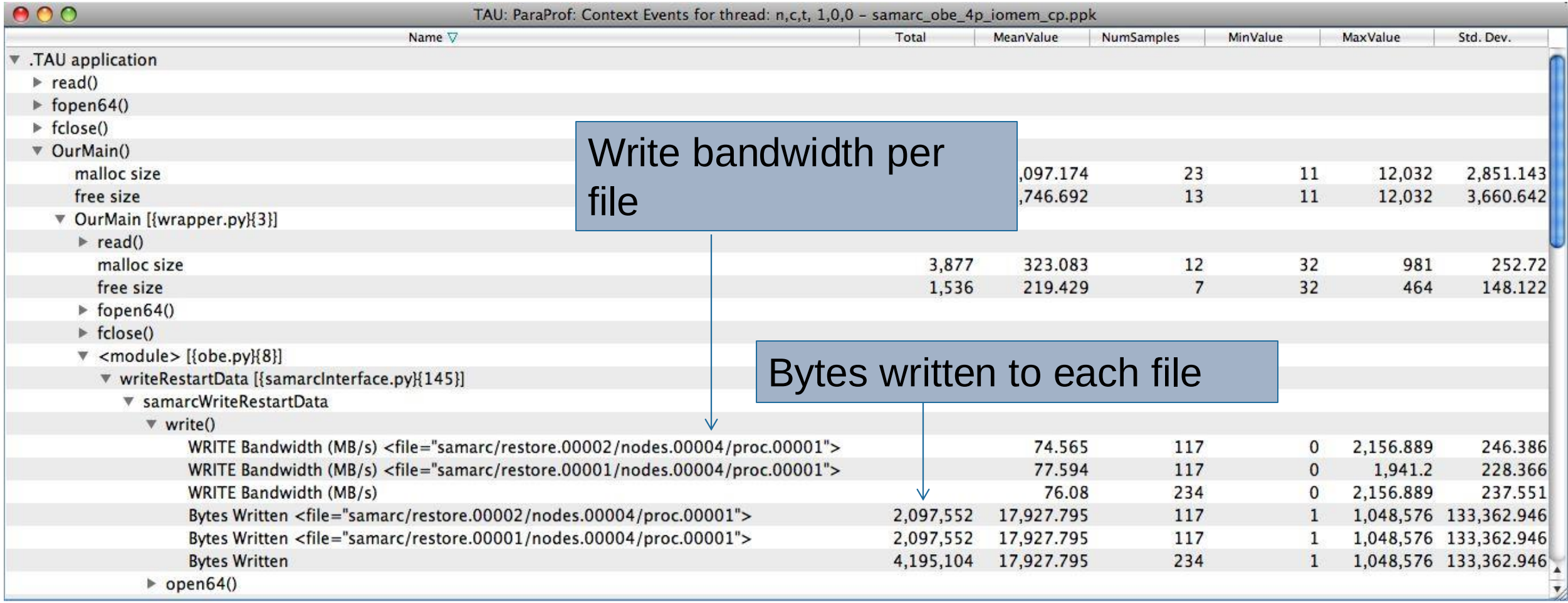
% export TAU_CALLPATH=1; export TAU_CALLPATH_DEPTH=100

TAU Atomic Events

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

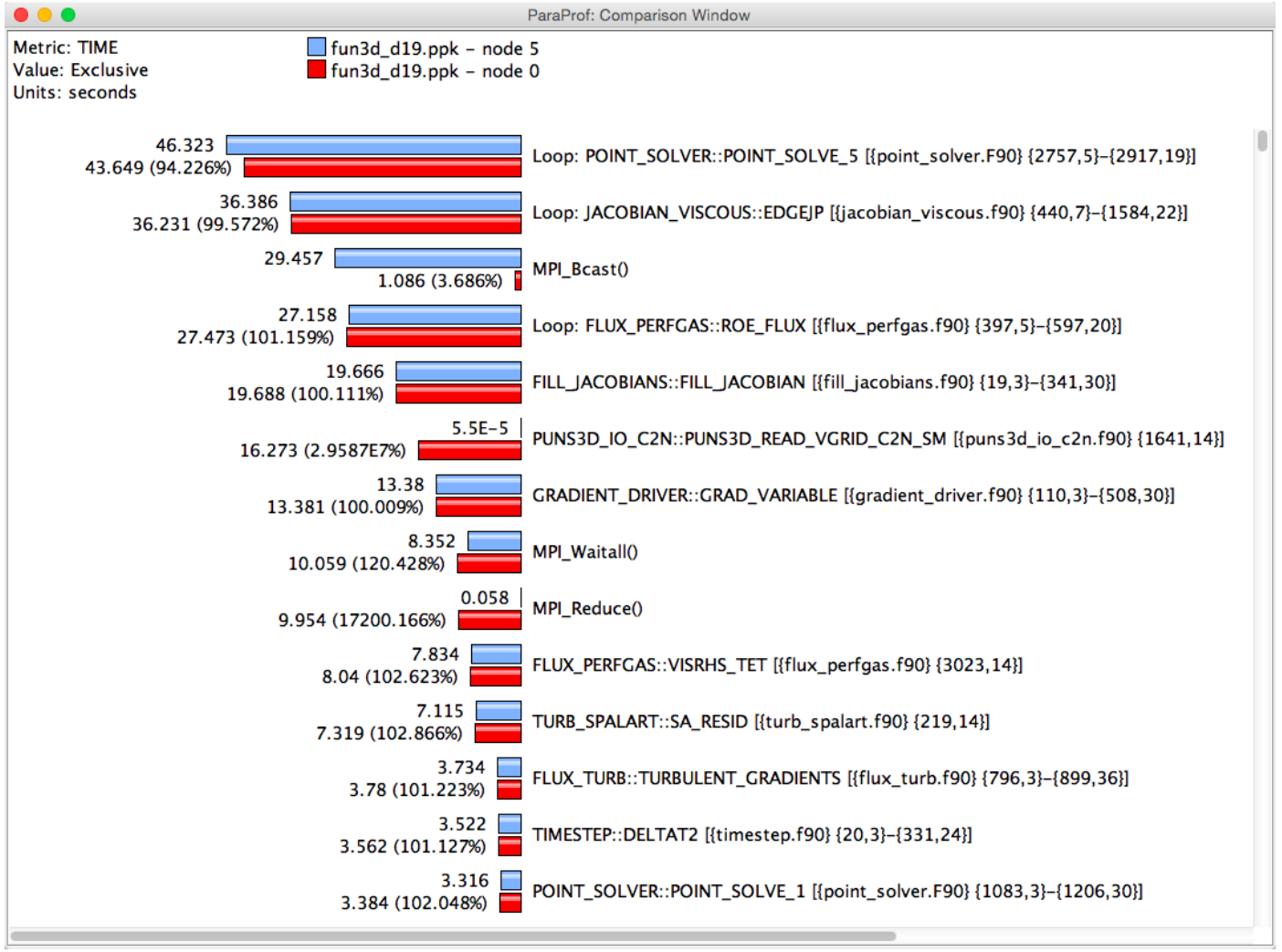
Name	Total	NumSamples	MaxValue	MinValue	MeanValue	Std. Dev.
Bytes Written <file=stdout>	911	62	21	1	14.694	7.441
Bytes Written <file=pipe>	22	22	1	1	1	0
Bytes Written <file=Process_Output/VelRsdl.dat>	7,826	100	302	76	78.26	22.487
Bytes Written <file=Process_Output/MomRsdl.dat>	7,826	100	302	76	78.26	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/NA	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



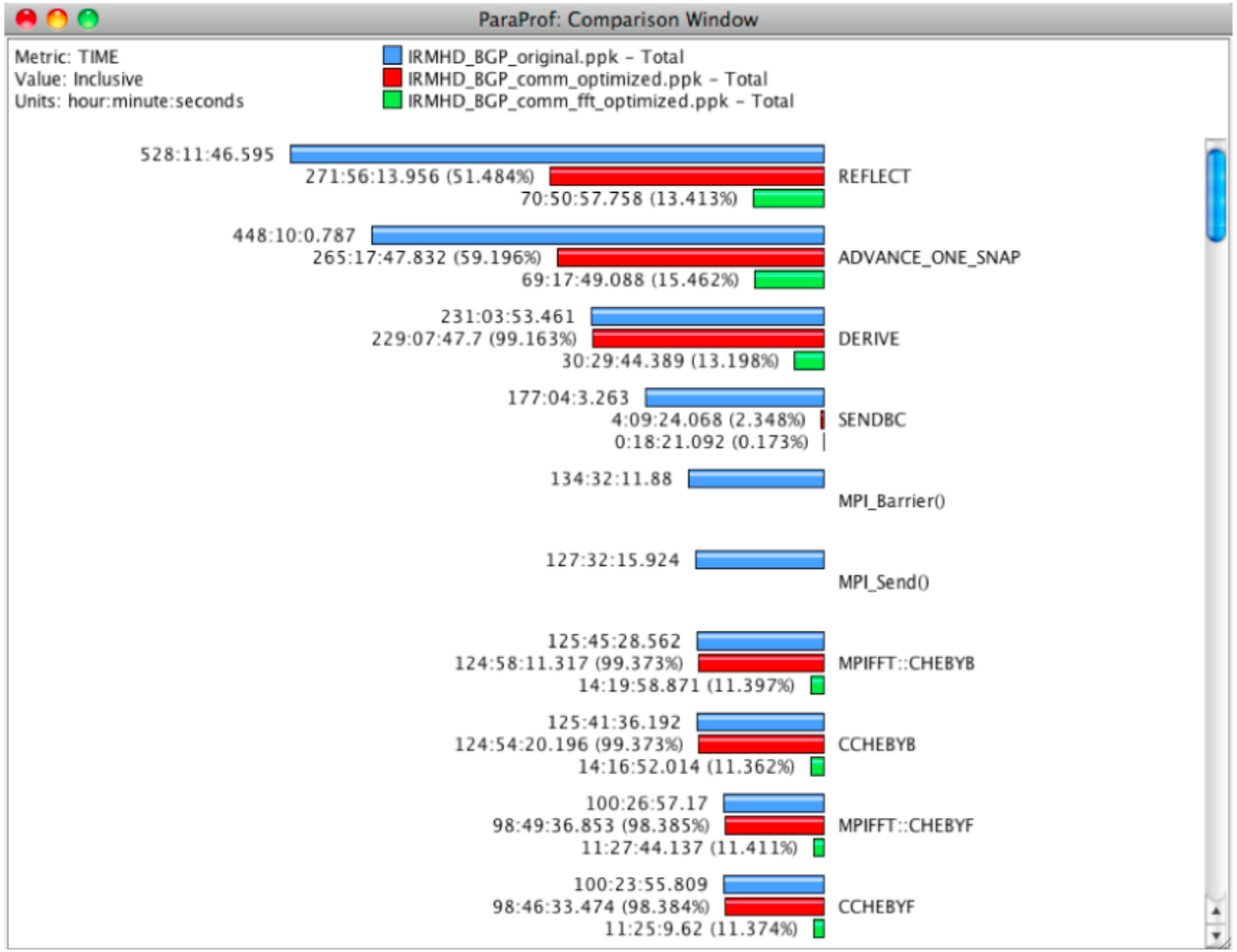
Name	Total	MeanValue	NumSamples	MinValue	MaxValue	Std. Dev.
.TAU application						
read()						
fopen64()						
fclose()						
OurMain()						
malloc size		,097.174	23	11	12,032	2,851.143
free size		,746.692	13	11	12,032	3,660.642
OurMain [{wrapper.py}{3}]						
read()						
malloc size	3,877	323.083	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()						

ParaProf Comparison Window

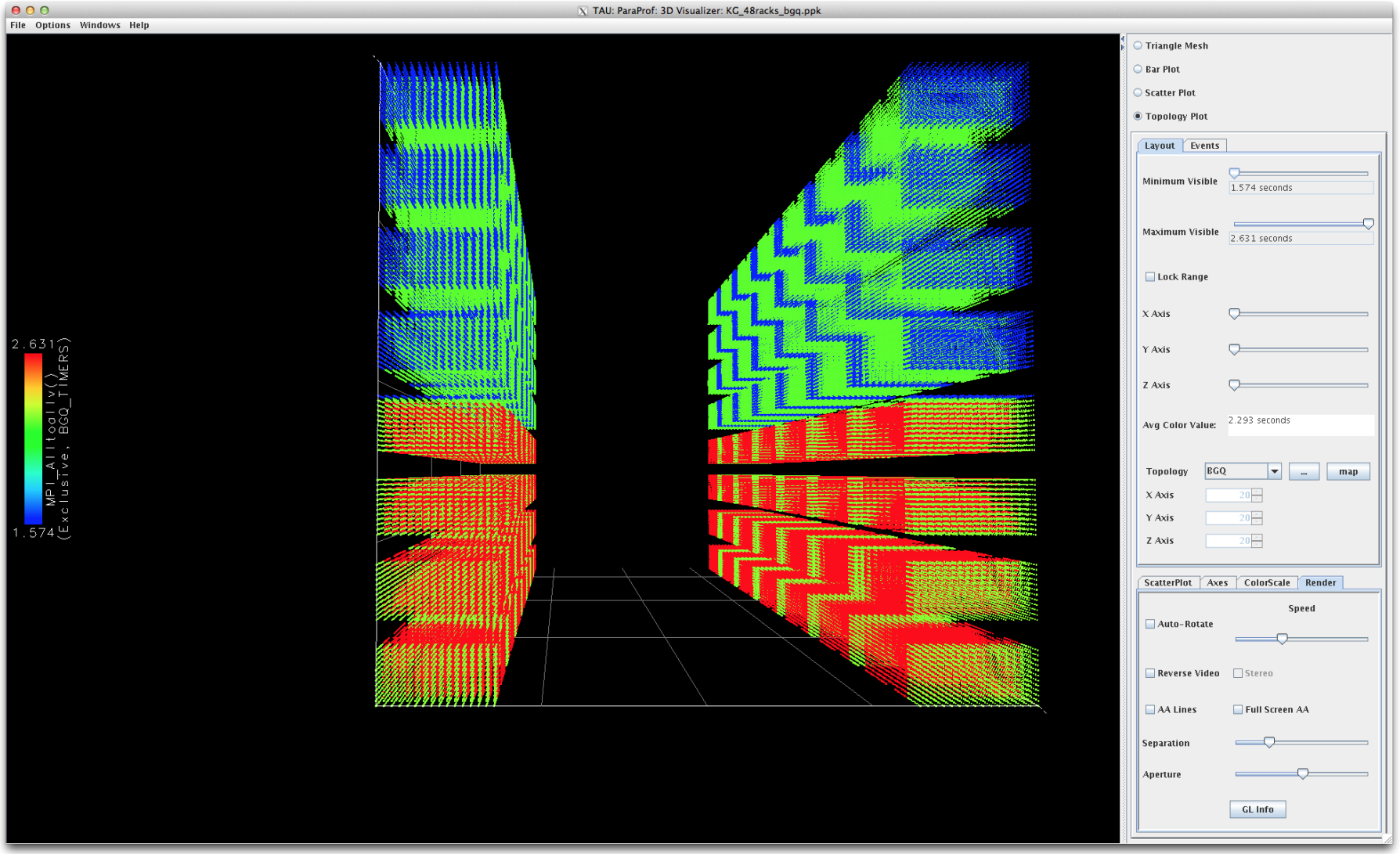


Comparing Rank 0 with 5.
Right click on “node 5” -> Add node to comparison window

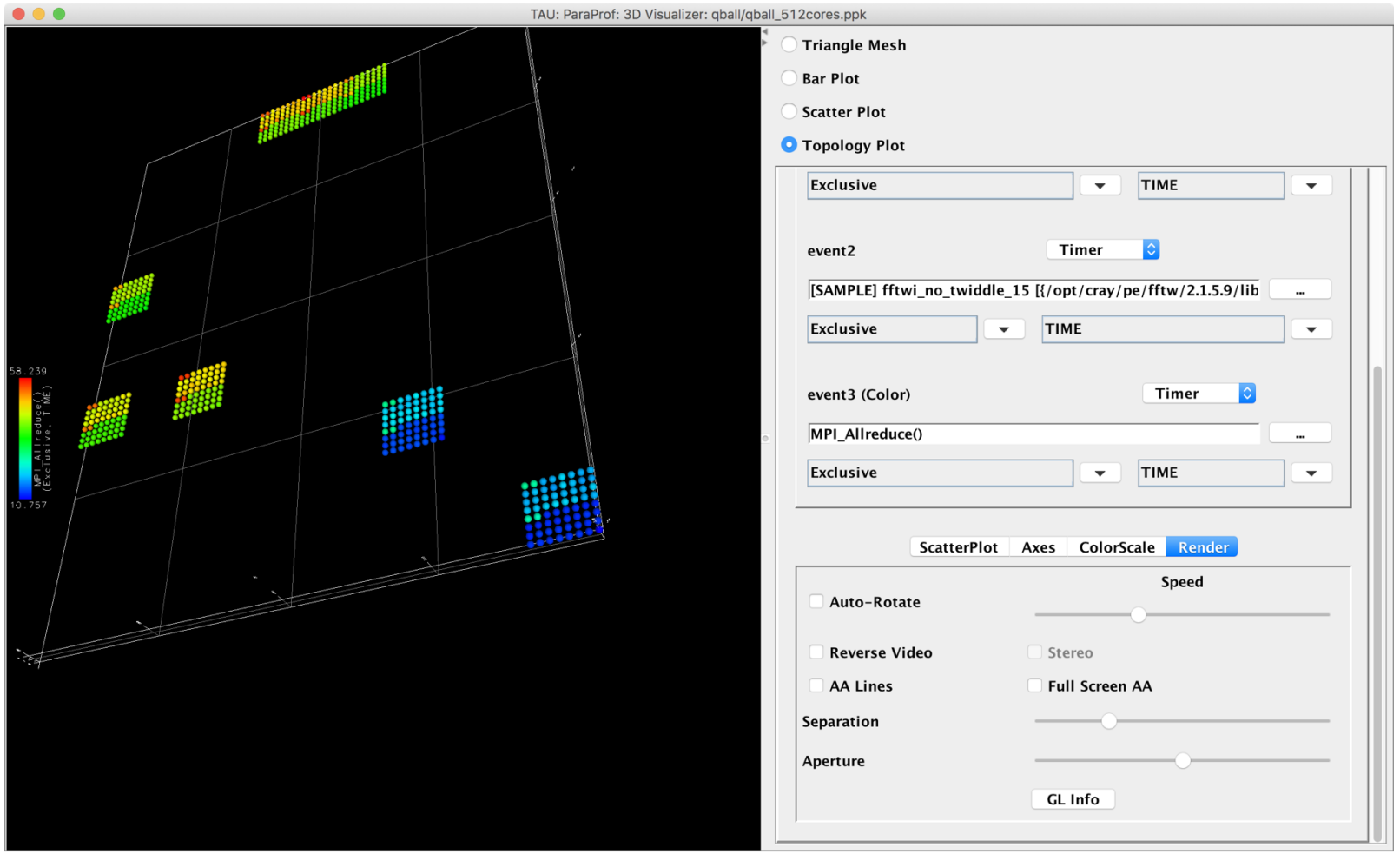
ParaProf Comparison Window



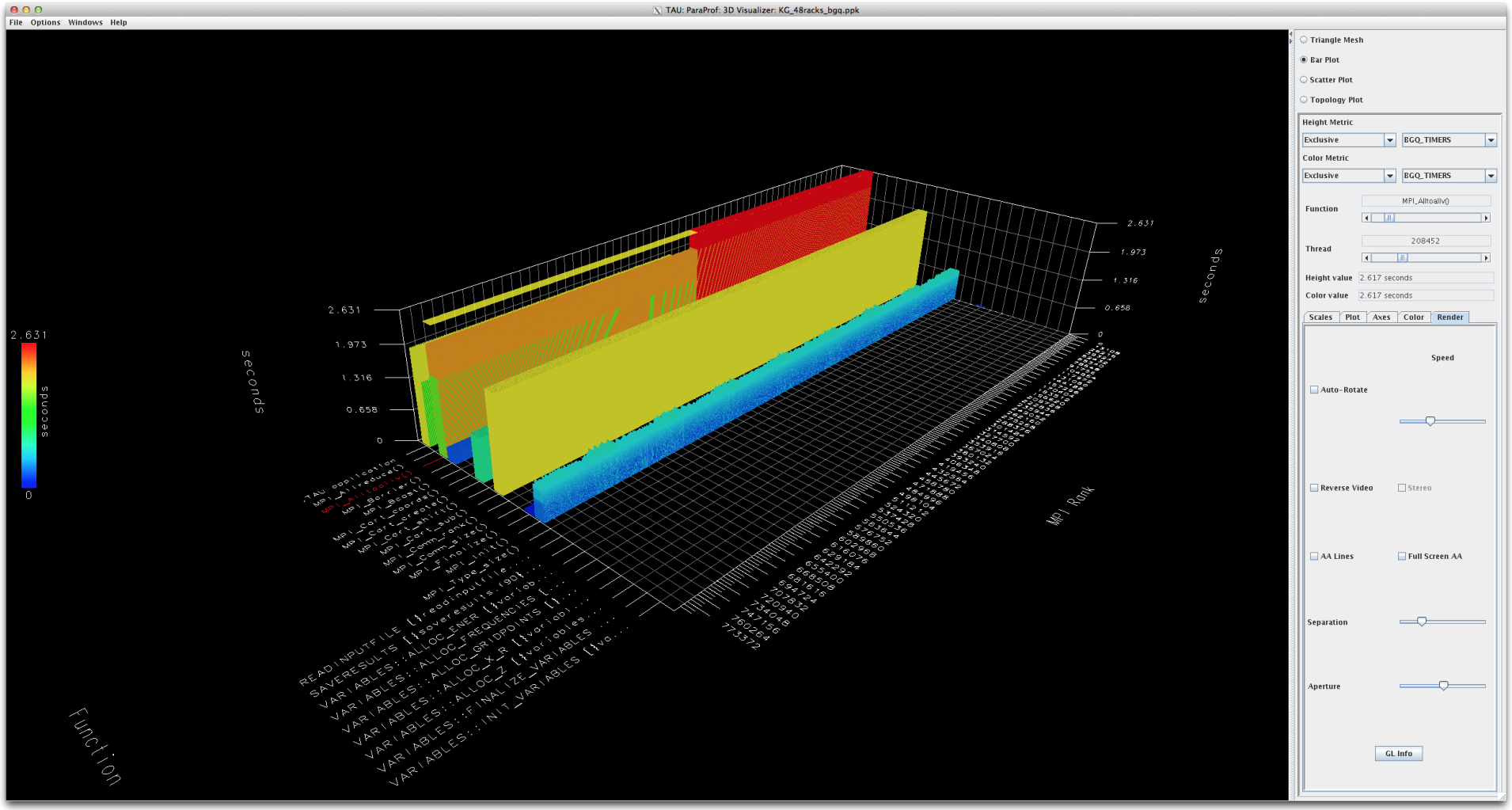
ParaProf's Topology Display Window



ParaProf Topology Display



ParaProf’s Scalable 3D Visualization



786,432 ranks

Conclusions

- TAU and SALT-FM target source code instrumentation of mixed language applications
- Fortran, C, and C++ are supported
- LLVM compiler infrastructure is used to instrument the source code
- Using an LLVM plugin, there is no need to patch the compiler source code
- Source instrumentation adds context to performance data generated
- Source instrumentation with instrumentation of runtime libraries (CUDA, ROCm, oneAPI, Kokkos, MPI, pthread) helps give a holistic view of performance on CPU and GPU activities
- Future work:
  - Generic static analysis and program transformation
  - Automatic generation of interfaces for use in mixed language programming
  - Porting portions of code to a different language (C++) or generating parallel code to be run on GPUs

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 ; tar xf ext.tgz
  - ./configure -bfd=download -pdt=<dir> -papi=<dir> -mpi –pthread –c++=mpicxx –cc=mpicc –fortran=mpif90 –dwarf=download –unwind=download –otf=download –iowrapper –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 F90=tau_f90.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.

…

TAU's 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_FP_INS,PAPI_NATIVE_<event>:<subevent>)

Runtime Environment Variables

Environment Variable	Default	Description
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_EBS_RESOLUTION	line	Setting to “function” or “file” changes the sampling resolution to function or file level respectively.
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.
TAU_OMPT_SUPPORT_LEVEL	basic	Setting to “full” improves resolution of OMPT TR6 regions on threads 1.. N-1. Also, “lowoverhead” option is available.
TAU_OMPT_RESOLVE_ADDRESS_EAGERLY	1	Setting to 1 is necessary for event based sampling to resolve addresses with OMPT. Setting to 0 allows the user to do offline address translation.

Runtime Environment Variables

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

Support Acknowledgements

- NASA SBIR Phase I 80NSSC24PB401
- 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
- Department of Defense (DoD)
  - PETTT, HPCMP
- National Science Foundation (NSF)
  - SI2-SSI, Glassbox
- CEA, France
- Industry: IBM, AMD, ARM, Intel, NVIDIA, AWS
- Partners:
  - University of Oregon
  - The Ohio State University
  - ParaTools, Inc.
  - University of Tennessee, Knoxville
  - T.U. Dresden, GWT
  - Jülich Supercomputing Center

Thank you

https://www.exascaleproject.org

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.



Thank you to all collaborators in the ECP and broader computational science communities. The work discussed in this presentation represents creative contributions of many people who are passionately working toward next-generation computational science.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR).



- https://science.osti.gov/ascr
- https://pesoproject.org
- https://ascr-step.org

