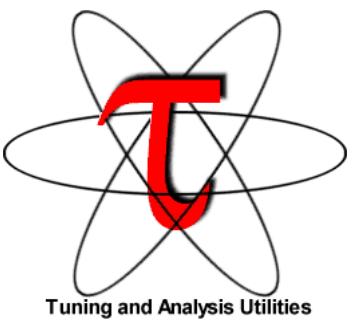


## TAU Performance System®



Tuning and Analysis Utilities

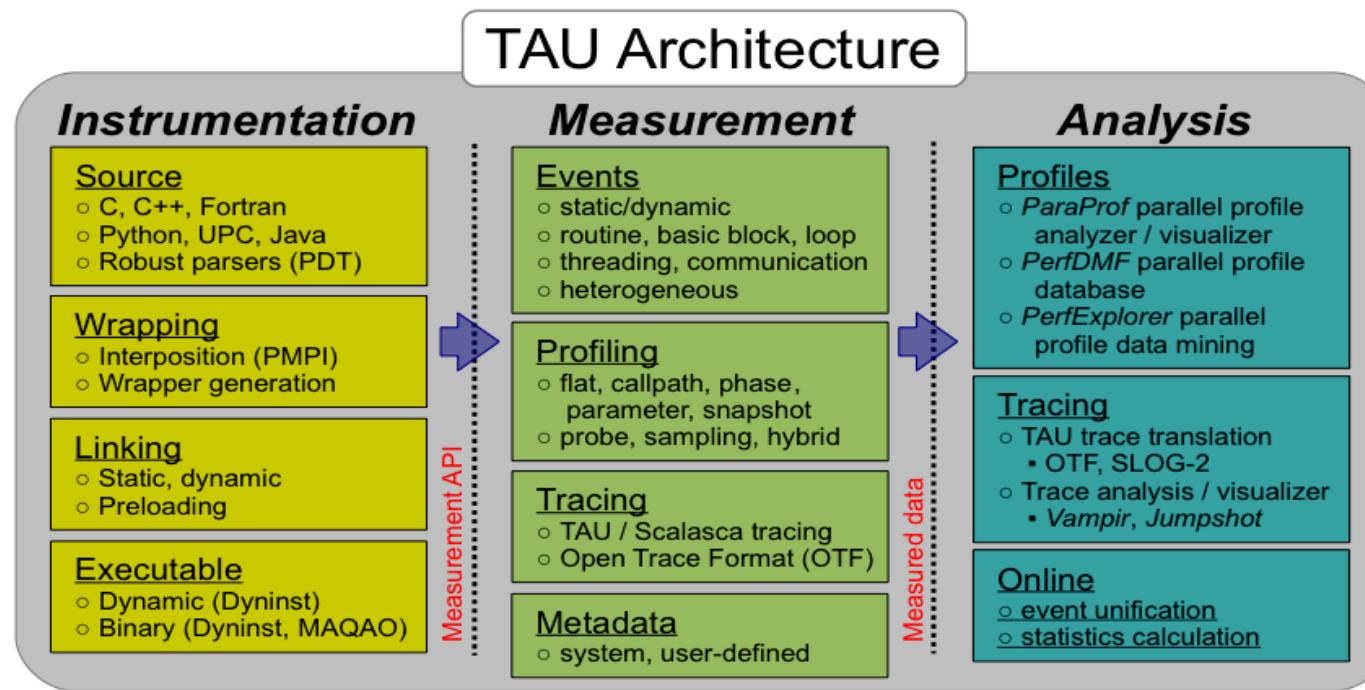
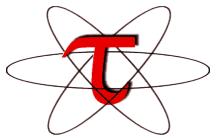
Sameer Shende  
[sameer@cs.uoregon.edu](mailto:sameer@cs.uoregon.edu)

University of Oregon  
[http://tau.uoregon.edu/TAU\\_TW37.pdf](http://tau.uoregon.edu/TAU_TW37.pdf)



# TAU Performance System®

- Parallel performance framework and toolkit
  - Supports all HPC platforms, compilers, runtime system
  - Provides portable instrumentation, measurement, analysis



# TAU Performance System

---

- Instrumentation
  - Fortran, C++, C, UPC, Java, Python, Chapel
  - Automatic instrumentation
- Measurement and analysis support
  - MPI, OpenSHMEM, ARMCI, PGAS, DMAPP
  - pthreads, OpenMP, OMPT interface, hybrid, other thread models
  - GPU, Intel OneAPI Level Zero, CUDA (CUPTI), OpenCL, OpenACC, ROCm (rocprofiler), HIP
  - Parallel profiling and tracing
  - Use of Score-P for native OTF2 and CUBEX generation
  - Efficient callpath profiles and trace generation using Score-P
- Analysis
  - Parallel profile analysis (ParaProf), data mining (PerfExplorer)
  - Performance database technology (TAUdb)
  - 3D profile browser

# TAU Performance System

---

- TAU supports both sampling and direct instrumentation
- Memory debugging as well as I/O performance evaluation
- Profiling as well as tracing
- Interfaces with Score-P for more efficient measurements
- TAU's instrumentation covers:
  - Runtime library interposition (`tau_exec`)
  - Compiler-based instrumentation
  - Native generation of OTF2 traces (`TAU_TRACE=1, TAU_TRACE_FORMAT=otf2`)
  - Callsite instrumentation with profiles and traces (`TAU_CALLSITE=1`)
  - PDT based Source level instrumentation: routine & loop
  - Event based sampling (`TAU_SAMPLING=1` or `tau_exec -ebs`)
  - Callstack unwinding with sampling (`TAU_EBS_UNWIND=1`)
  - OpenMP Tools Interface v5 (OMPT, `tau_exec -T ompt,v5`)
  - Intel Level Zero for Gen12LP GPUs (`tau_exec -T level_zero -l0` )
  - CUDA CUPTI, OpenCL (`tau_exec -T cupti -cupti`)

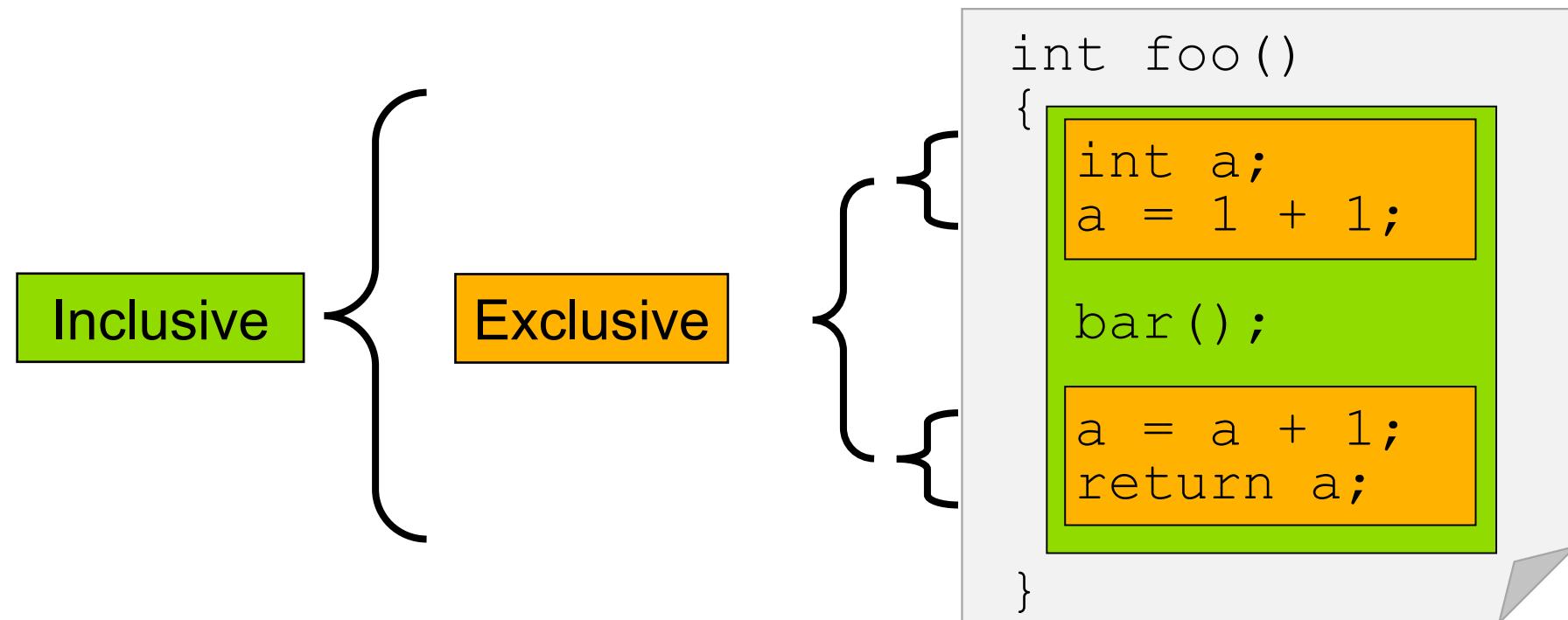
# Application Performance Engineering using TAU

---

- 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?
- How many instructions are executed in these code regions? Using Likwid or PAPI, TAU measures floating point, Level 1 and 2 *data cache misses*, hits, branches taken.
- What is the time taken in OS routines for thread scheduling? How much time is wasted?
- 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?
- 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?

# Inclusive vs. Exclusive values

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

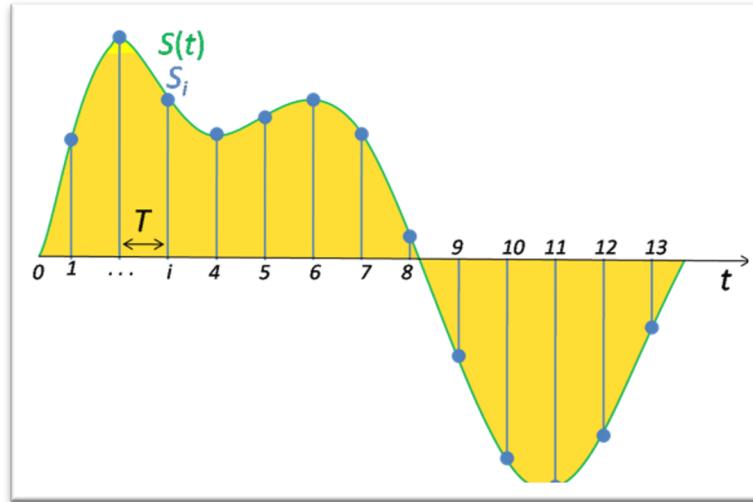


# Performance Data Measurement

## Direct via Probes

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

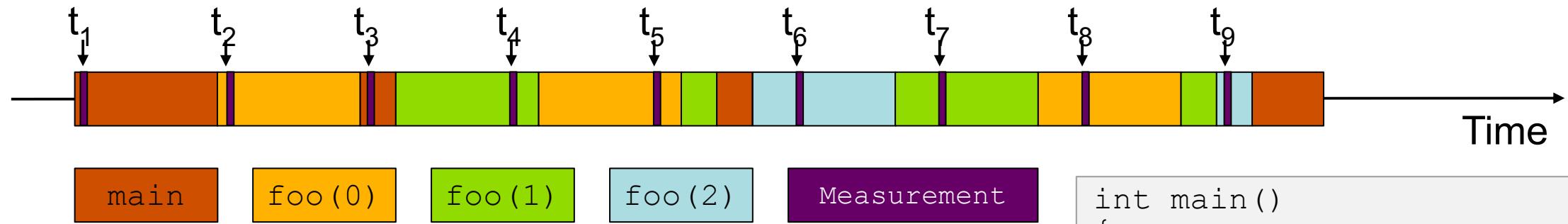
## Indirect via Sampling



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

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

# Event-Based Sampling (EBS)



- 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 (`tau_exec -ebs`)

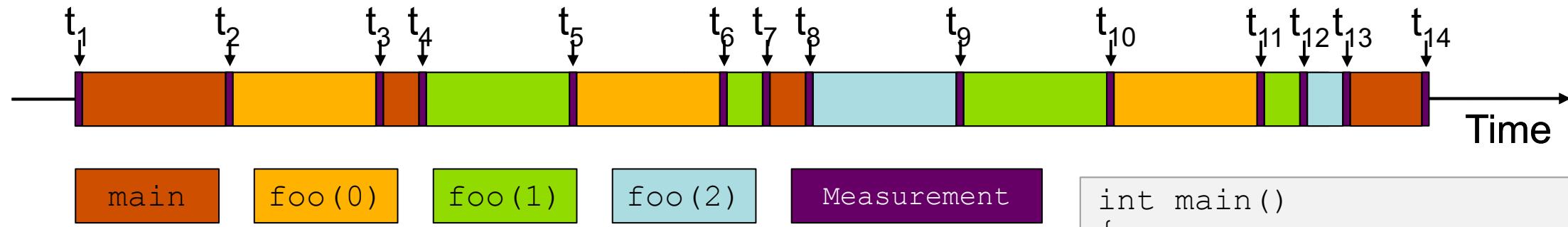
```
int main()
{
    int i;

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

    return 0;
}

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

# Instrumentation

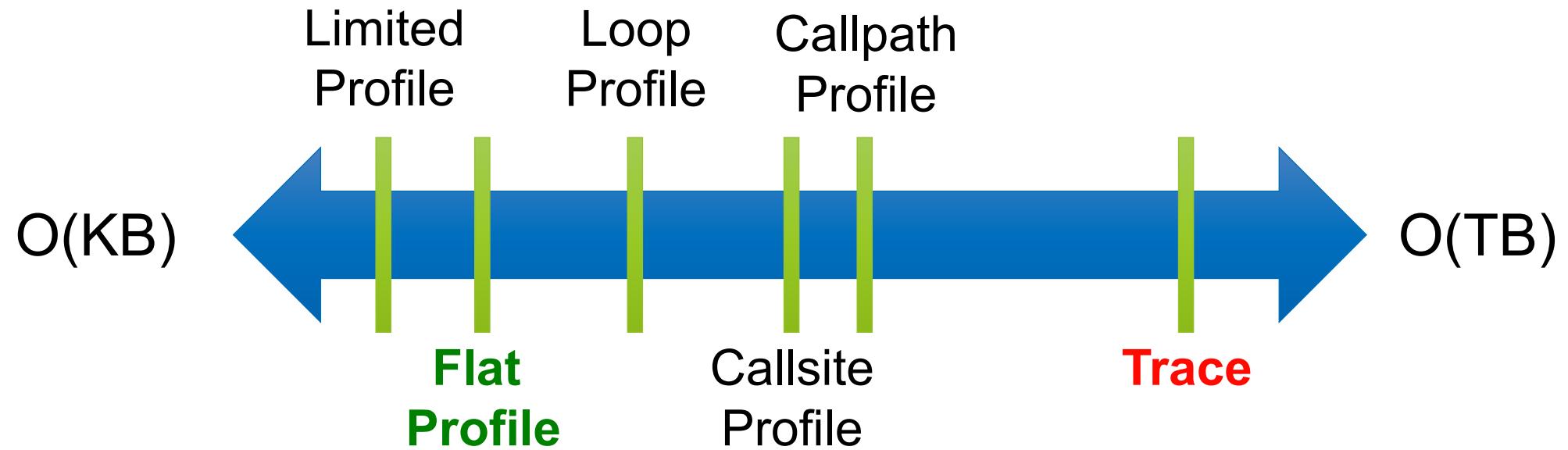


- Measurement code is inserted such that every event of interest is captured directly
  - Can be done in various ways
- Advantage:
  - Much more detailed information
- Disadvantage:
  - Processing of source-code / executable necessary
  - Large relative overheads for small functions

```
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");
}
```

# How much data do you want?



# Types of Performance Profiles

---

- *Flat* profiles

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

- *Callpath* profiles

- Time spent along a calling path (edges in callgraph)
  - “*main=> f1 => f2 => MPI\_Send*”
  - Set the `TAU_CALLPATH` and `TAU_CALLPATH_DEPTH` environment variables

- *Callsite* profiles

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

- *Phase* profiles

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

# 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)
- Add tau\_exec before the name of the binary
  - mpirun tau\_exec ./a.out
  - mpirun tau\_exec -T ompt,v5,mpi,papi -ompt ./a.out

# tau\_exec

```
$ tau_exec

Usage: tau_exec [options] [--] <exe> <exe options>

Options:
  -v          Verbose mode
  -s          Show what will be done but don't actually do anything (dryrun)
  -qsub       Use qsub mode (BG/P only, see below)
  -io         Track I/O
  -memory    Track memory allocation/deallocation
  -memory_debug Enable memory debugger
  -cuda       Track GPU events via CUDA
  -cupti     Track GPU events via CUPTI (Also see env. variable TAU_CUPTI_API)
  -opencl    Track GPU events via OpenCL
  -openacc   Track GPU events via OpenACC (currently PGI only)
  -ompt      Track OpenMP events via OMPT interface
  -armci     Track ARMCI events via PARMCI
  -ebs       Enable event-based sampling
  -ebs_period=<count> Sampling period (default 1000)
  -ebs_source=<counter> Counter (default itimer)
  -um        Enable Unified Memory events via CUPTI
  -T <DISABLE,GNU,ICPC,MPI,OMPT,OPENMP,PAPI,PDT,PROFILE,PTHREAD,SCOREP,SERIAL> : Specify TAU tags
  -loadlib=<file.so>  : Specify additional load library
  -XrunTAUsh-<options> : Specify TAU library directly
  -gdb       Run program in the gdb debugger

Notes:
  Defaults if unspecified: -T MPI
  MPI is assumed unless SERIAL is specified
```

- Tau\_exec preloads the TAU wrapper libraries and performs measurements.

No need to recompile the application!

# tau\_exec Example (continued)

Example:

```
mpirun -np 2 tau_exec -T icpc,ompt,mpi -ompt ./a.out  
mpirun -np 2 tau_exec -io ./a.out
```

Example - event-based sampling with samples taken every 1,000,000 FP instructions

```
mpirun -np 8 tau_exec -ebs -ebs_period=1000000 -ebs_source=PAPI_FP_INS ./ring
```

Examples - GPU:

```
tau_exec -T serial,cupti -cupti ./matmult (Preferred for CUDA 4.1 or later)  
tau_exec -openacc ./a.out  
tau_exec -T serial -opencl ./a.out (OPENCL)  
mpirun -np 2 tau_exec -T mpi,cupti,papi -cupti -um ./a.out (Unified Virtual Memory in CUDA 6.0+)
```

qsub mode (IBM BG/Q only):

Original:

```
qsub -n 1 --mode smp -t 10 ./a.out
```

With TAU:

```
tau_exec -qsub -io -memory -- qsub -n 1 ... -t 10 ./a.out
```

Memory Debugging:

-memory option:

Tracks heap allocation/deallocation and memory leaks.

-memory\_debug option:

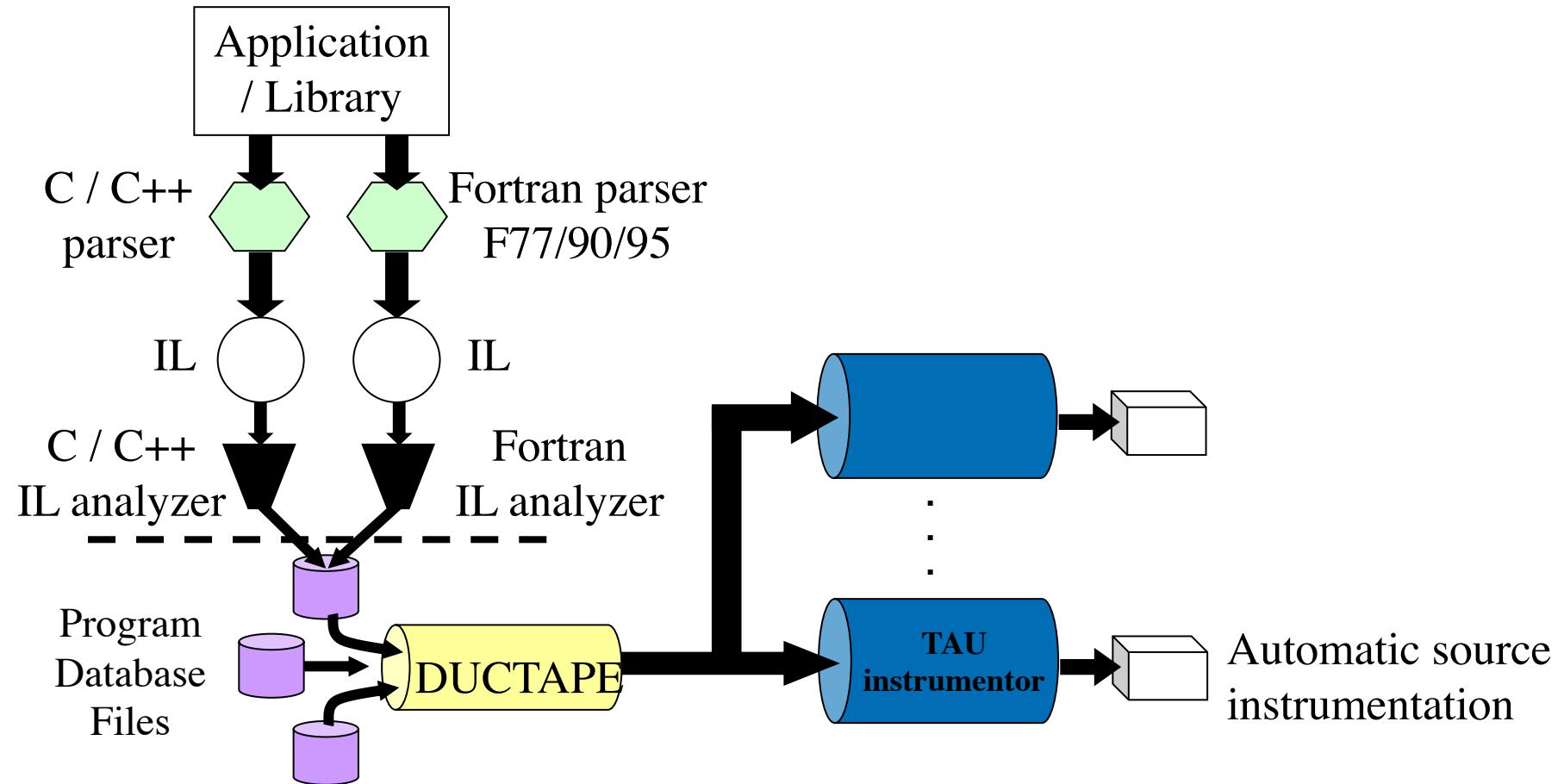
Detects memory leaks, checks for invalid alignment, and checks for array overflow. This is exactly like setting TAU\_TRACK\_MEMORY\_LEAKS=1 and TAU\_MEMDBG\_PROTECT\_ABOVE=1 and running with -memory

- tau\_exec can enable event based sampling while launching the executable using the **-ebs** flag!

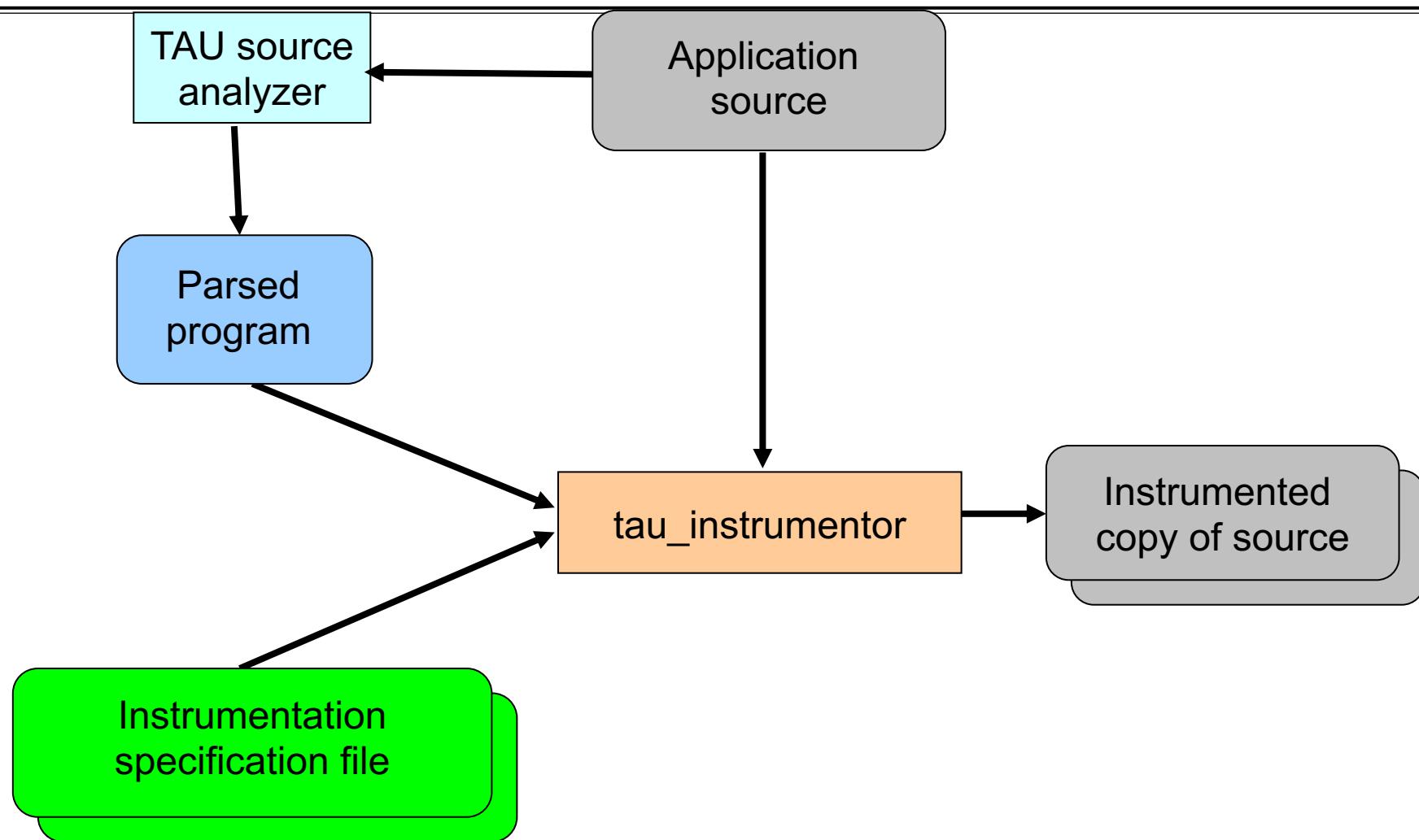
---

# TAU's Source Instrumentation

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



# Automatic Source Instrumentation using PDT



# Installing and Configuring TAU

---

- **Installing PDT:**

- wget <http://tau.uoregon.edu/pdt.tgz>
  - ./configure; make ; make install

- **Installing TAU :**

- wget <http://tau.uoregon.edu/tau.tgz>
  - ./configure --c++=mpiicpc --cc=mpiicc --fortran=mpiifort -mpi --tag=intelmpi -bfd=download --otf=download --unwind=download --dwarf=download -ompt -pdt=<dir> -papi=<dir>
  - make install; export PATH=<taudir>/x86\_64/bin:\$PATH
  - All configurations are stored in <taudir>/.all\_configs if you wish to see how TAU was configured!

- **Using TAU for source instrumentation:**

- export TAU\_MAKEFILE=<taudir>/x86\_64/lib/Makefile.tau-<TAGS>
  - make CC=tau\_cc.sh CXX=tau\_cxx.sh F90=tau\_f90.sh
  - Use tau\_exec with uninstrumented binaries instead of recompiling the source code.

# Installing TAU on your laptop for paraprof (GUI)

## ▪ Microsoft Windows

- Install Java from Oracle.com
- <http://tau.uoregon.edu/tau.exe>
- Install, click on a ppk file to launch paraprof

## ▪ macOS (x86\_64)

- Install Java 11.0.3:
  - Download <http://tau.uoregon.edu/java.dmg>
  - If you have multiple Java installations, add to your `~/.zshrc` (or `~/.bashrc` as appropriate):  
`export PATH=/Library/Java/JavaVirtualMachines/jdk-11.0.3.jdk/Contents/Home/bin:$PATH`
  - `java -version`
- Download and install TAU (copy to /Applications from dmg):
  - <http://tau.uoregon.edu/tau.dmg>
  - `export PATH=/Applications/TAU/tau/apple/bin:$PATH`
  - `paraprof app.ppk &`

## ▪ macOS (arm64, M1)

- [http://tau.uoregon.edu/tau\\_arm64.dmg](http://tau.uoregon.edu/tau_arm64.dmg)

## ▪ Linux (<http://tau.uoregon.edu/tau.tgz>)

- `./configure; make install; export PATH=<taudir>/x86_64/bin:$PATH`
- `paraprof app.ppk &`

# Using TAU on Goethe-HLR at CSC, Frankfurt

- Setup preferred program environment compilers using Intel compilers with OpenMP

```
% ssh -Y Goethe.hhlr-gu.de -l <USER>
Add to your ~/.bashrc:
# User specific aliases and functions
if [ -d /home/vihps/software/modulefiles ]; then
    module use /home/vihps/software/modulefiles
    module load tau
    module unload comp mpi
    module load comp/intel/2020.0 mpi/intel/2019.5
fi
% tar zxf /home/vihps/public/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI; make clean; make suite; cd bin; cp ../../jobsript/goethe-hlr/* .
% sbatch --reservation=VIHPS ./tau.1.sbatch
% pprof -a
...
```

# Using TAU's Source Code Instrumentation

- TAU supports several compilers, measurement, and thread options
  - Intel compilers, profiling with hardware counters using PAPI, MPI library, CUDA...
  - Each measurement configuration of TAU corresponds to a unique stub makefile (configuration file) and library that is generated when you configure it
- To instrument source code automatically using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:
    - % module use /home/vihps/software/modulefiles
    - % module load tau
    - % export TAU\_MAKEFILE=\$TAU/Makefile.tau-intelmpi-icpc-mpi-pdt  
% export TAU\_OPTIONS=' -optVerbose ...' (see tau\_compiler.sh )
  - Use tau\_f90.sh, tau\_f77.sh , tau\_cxx.sh, tau\_upc.sh, or tau\_cc.sh as F90, F77, 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)
```

# Makefiles for TAU Compiler and Runtime Options (Goethe-HLR)

```
% module use /home/vihps/software/modulefiles; module load tau
% echo $TAU
/home/vihps/software/tau/tau-2.30/x86_64/lib
% ls $TAU/Makefile.*
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-intelmpi-icpc-mpi-pdt
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-intelmpi-icpc-mpi-pdt-likwid
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-intelmpi-icpc-mpi-pthread-pdt-likwid
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-intelmpi-icpc-ompt-v5-mpi-pdt-openmp-likwid
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-intelmpi-icpc-papi-mpi-pthread-pdt
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-intelmpi-icpc-papi-ompt-v5-mpi-pdt-openmp
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-openmpi-mpi-pdt
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-openmpi-mpi-pdt-likwid
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-openmpi-mpi-pthread-pdt-likwid
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-openmpi-ompt-v5-mpi-pdt-openmp-likwid
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-openmpi-papi-mpi-pthread-pdt
/home/vihps/software/tau/tau-2.30/x86_64/lib/Makefile.tau-openmpi-papi-ompt-v5-mpi-pdt-openmp
```

For an MPI+Fortran with Intel MPI, you may choose

\$TAU/Makefile.tau-intelmpi-icpc-mpi-pdt

Supports MPI instrumentation & Opari and PDT for automatic source instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-intelmpi-icpc-mpi-pdt
% tau_f90.sh matmult.f90 -o matmult
% mpirun -np 16 ./matmult; paraprof
```

# Compile-Time Options

- Optional parameters for the TAU\_OPTIONS environment variable:

% tau\_compiler.sh

-optVerbose	Turn on verbose debugging messages
-optComInst	Use compiler based instrumentation
-optNoComInst	Do not revert to compiler instrumentation if source instrumentation fails.
-optTrackIO	Wrap POSIX I/O call and calculates vol/bw of I/O operations (configure TAU with <i>–iowrapper</i> )
-optTrackGOMP	Enable tracking GNU OpenMP runtime layer (used without <i>–opari</i> )
-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=" <i>&lt;file&gt;</i> "	Specify selective instrumentation file for <i>tau_instrumentor</i>
-optTauWrapFile=" <i>&lt;file&gt;</i> "	Specify path to <i>link_options.tau</i> generated by <i>tau_gen_wrapper</i>
-optHeaderInst	Enable Instrumentation of headers
-optTrackUPCR	Track UPC runtime layer routines (used with <i>tau_upc.sh</i> )
-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.

...

# Using TAU\_OPTIONS

---

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

```
% export TAU_OPTIONS= '-optComplInst -optVerbose'
```

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

- % export TAU\_OPTIONS= '-optPreProcess -optVerbose -optDetectMemoryLeaks'

- To use an instrumentation specification file:

- % export TAU\_OPTIONS= '-optTauSelectFile=select.tau -optVerbose -optPreProcess'
  - % cat select.tau
  - 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

# Selective Instrumentation File With Program Database Toolkit (PDT)

---

- To use an instrumentation specification file for source instrumentation:

```
% export TAU_OPTIONS=' -optTauSelectFile=/path/to/select.tau --optVerbose '  
% cat select.tau
```

```
BEGIN_EXCLUDE_LIST  
BINVCRHS  
MATMUL_SUB  
MATVEC_SUB  
EXACT_SOLUTION  
BINVRHS  
LHS#INIT  
TIMER_  
END_EXCLUDE_LIST
```

- **NOTE:** paraprof can create this file from an earlier execution for you based on throttling.
- File -> Create Selective Instrumentation File -> save
- Selective instrumentation at runtime:

```
% export TAU_SELECT_FILE=select.tau
```

# Simplifying TAU's usage (`tau_exec`)

- Uninstrumented execution linked with `-dynamic` (dynamic executables only!)  
% mpirun -np 16 ./a.out

- Track MPI performance

```
% mpirun -np 16 tau_exec ./a.out
```

- Track OpenMP, and MPI performance (MPI enabled by default; OMPT in Clang 9+, Intel 19+)

```
% export TAU_OMPT_SUPPORT_LEVEL=full;  
% mpirun -np 16 tau_exec -T mpi,pdt,ompt,v5,papi -ompt ./a.out
```

- Using pthread to track an OpenMP application with sampling:

```
% mpirun -np 16 tau_exec -T pthread -ebs ./a.out
```

- Track memory operations

```
% export TAU_TRACK_MEMORY_LEAKS=1  
% mpirun -np 16 tau_exec -memory_debug ./a.out (bounds check)
```

- Use event based sampling (compile with `-g`)

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

Also `-ebs_source=<PAPI_COUNTER>` `-ebs_period=<overflow_count>` `-ebs_resolution=<file|function|line>`

- Load wrapper interposition library

```
% mpirun -np 16 tau_exec -loadlib=<path/libwrapper.so> ./a.out
```

- Track GPU operations (`-rocm`, `-level_zero`, `-opencl`, `-cupti`, `-cupti -um`, `-openacc`):

```
% mpirun -np 16 tau_exec -T cupti,papi -cupti ./a.out
```

# Configuration tags for tau\_exec

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

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

To explicitly choose preloading of shared-<options>/libTAU.so change:

```
% mpirun -np 256 ./a.out      to  
% mpirun -np 256 tau_exec -T <comma_separated_options> ./a.out
```

```
% mpirun -np 256 tau_exec -T papi,mpi,pdt ./a.out
```

Preloads \$TAU/shared-papi-mpi-pdt/libTAU.so

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

Preloads \$TAU/shared-papi-mpi-pdt/libTAU.so by matching.

```
% mpirun -n 256 tau_exec -T papi,mpi,pdt -s ./a.out
```

Does not execute the program. Just displays the library that it will preload if executed without the **-s** option.

NOTE: -mpi configuration is selected by default. Use **-T serial** for Sequential programs.

# 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

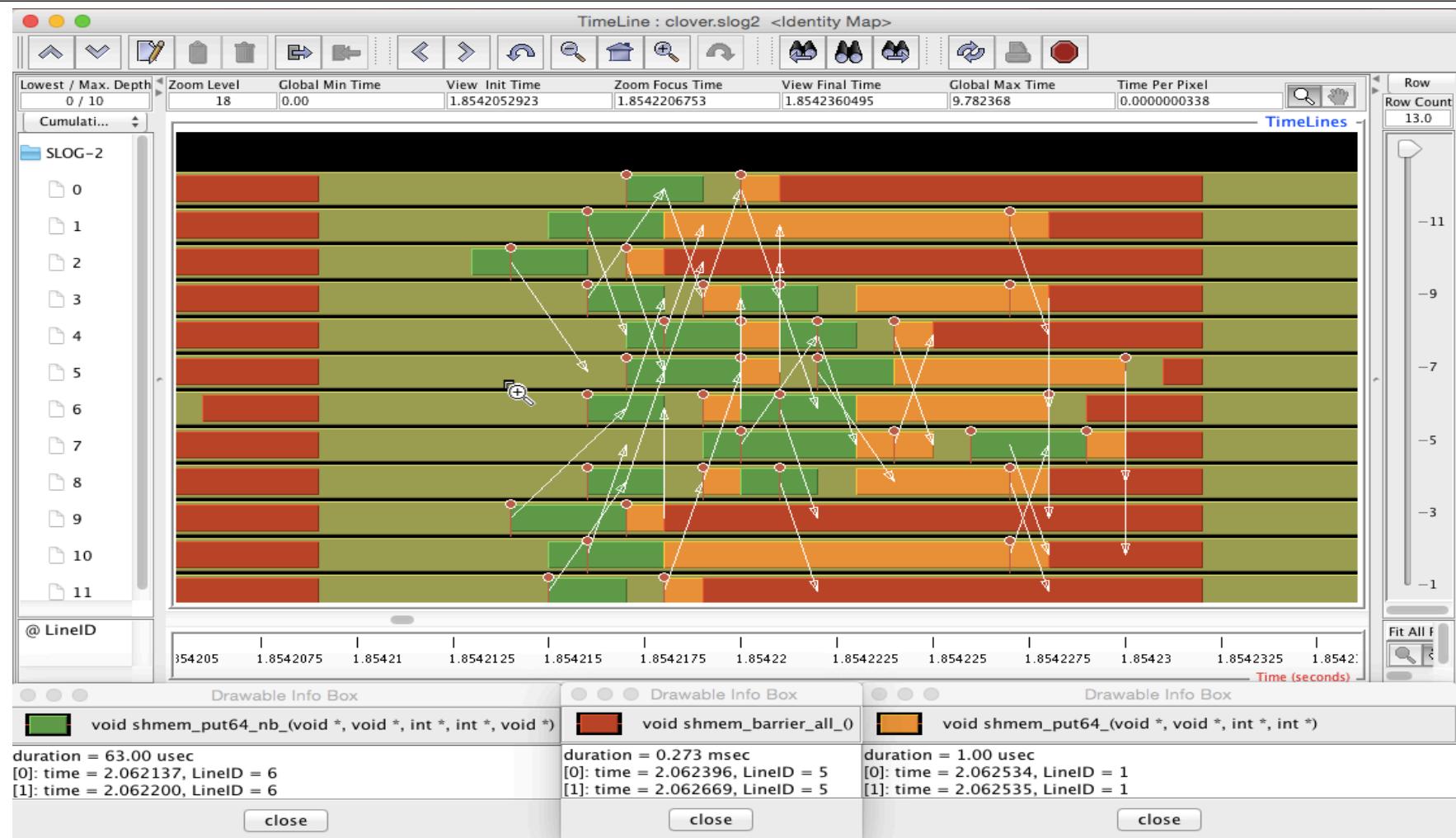
# Runtime Environment Variables

Environment Variable	Default	Description
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>)
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.
TAU_EVENT_THRESHOLD	0.5	Define a threshold value (e.g., .25 is 25%) to trigger marker events for min/max

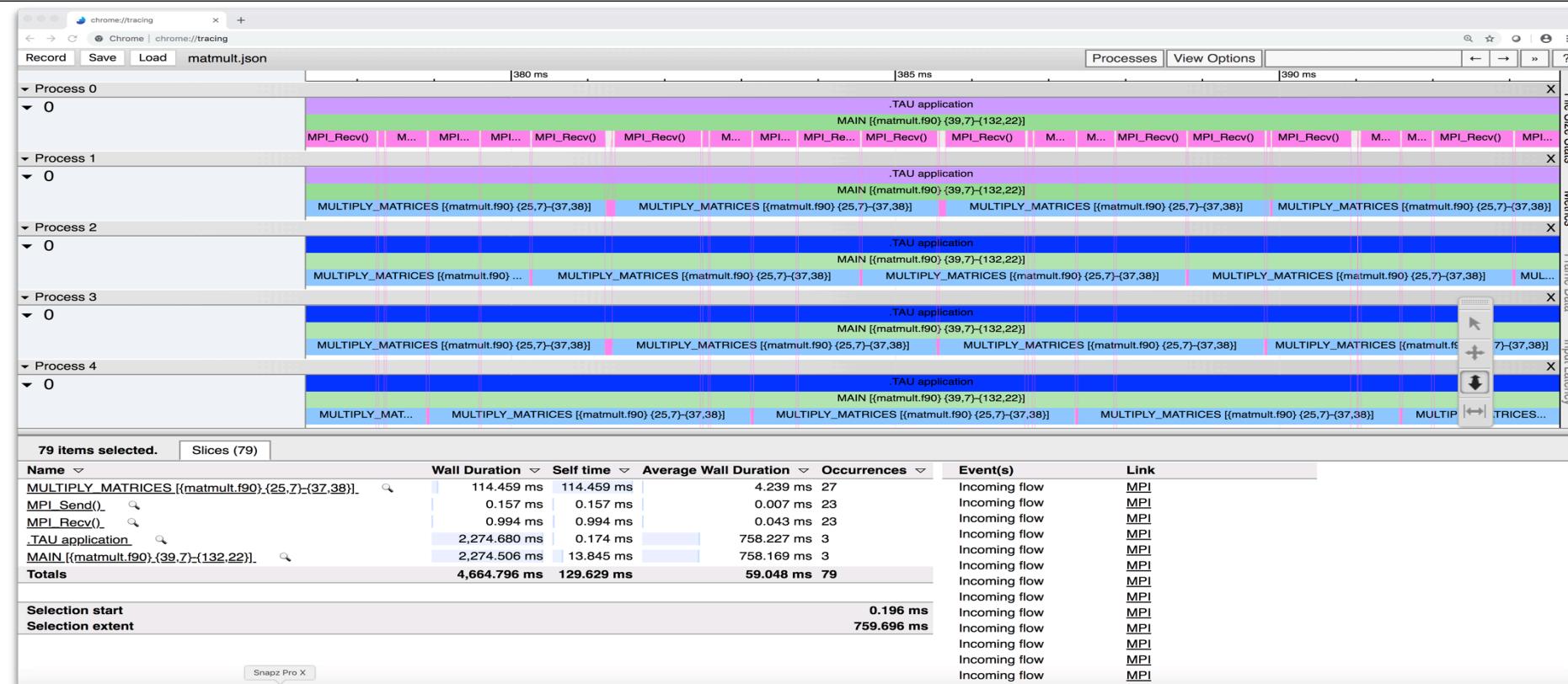
# Runtime Environment Variables

Environment Variable	Default	Description
TAU_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_ALIGNMENT	Sizeof(int)	Byte alignment for memory allocations

# Tracing: Jumpshot (ships with TAU)



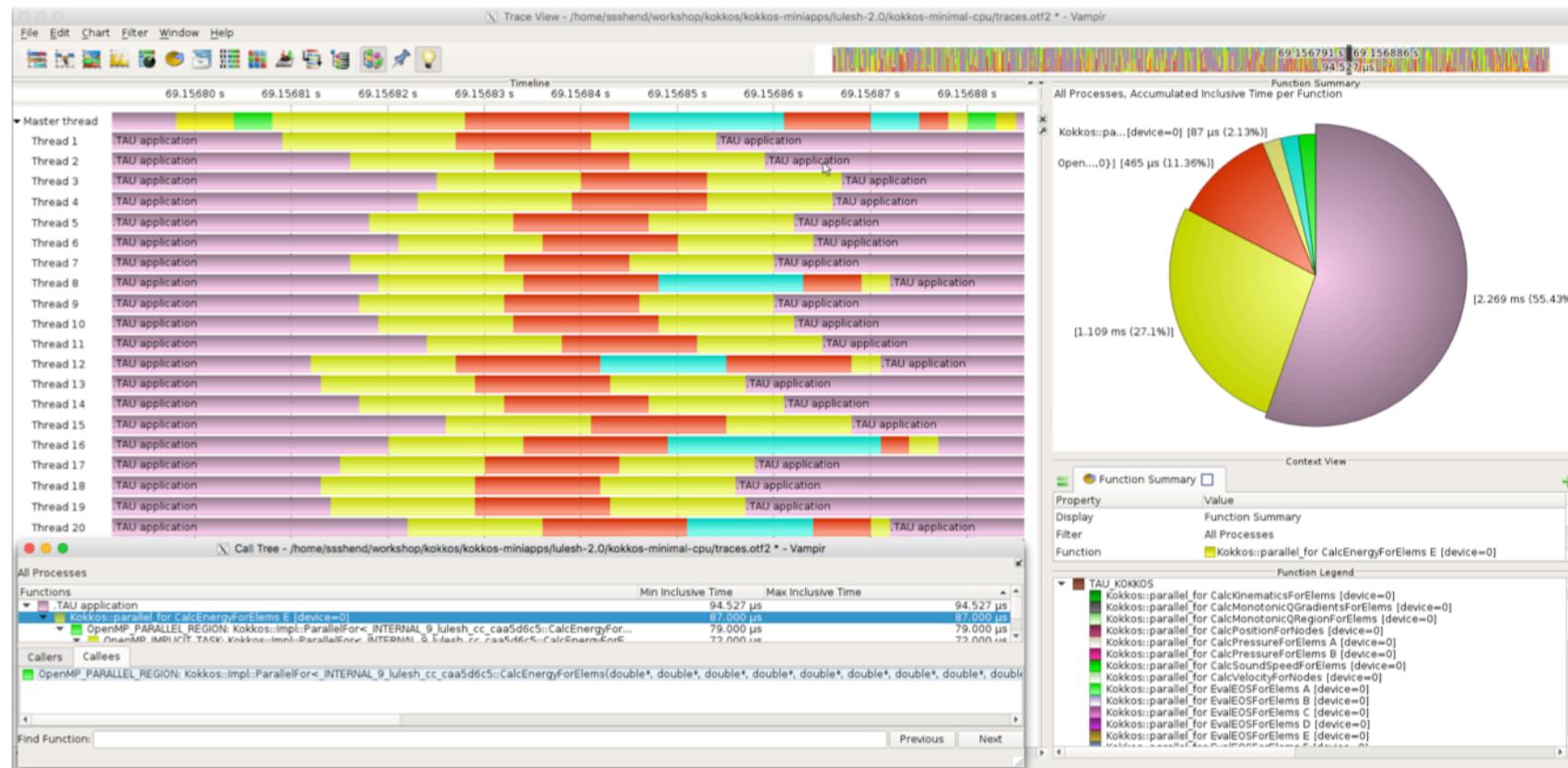
# Tracing: Chrome Browser



```
% 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)

# Vampir [TU Dresden] Timeline: Kokkos



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

# Kokkos

---

- Provides abstractions for node level parallelism (X in MPI+X)
- Productive, portable, and performant shared-memory programming model
- Helps you create single source performance portable codes
- Provides data abstractions
- C++ API for expressing parallelism in your program
- Aggressive compiler transformations using C++ templates
- Low level code targets backends such as OpenMP, Pthread, CUDA
- Creates a problem for performance evaluation tools
- Gap: performance data and higher-level abstractions
- Solution: Kokkos profiling API for mapping performance data

# Kokkos API use in ExaMiniMD

```
20. sameer@pegasus:~/pkgs/ORNL/DEMO/BUILD/ExaMiniMD-pthread/ExaMiniMD/src/comm_types (ssh)

void CommMPI::update_halo() {
    Kokkos::Profiling::pushRegion("Comm::update_halo"); ← pushRegion("Comm::update_halo")
    N_ghost = 0;
    s=*system;

    pack_buffer_update = t_buffer_update((T_X_FLOAT*)pack_buffer.data(),pack_indices_all.extent(1));
    unpack_buffer_update = t_buffer_update((T_X_FLOAT*)unpack_buffer.data(),pack_indices_all.extent(1));

    for(phase = 0; phase<6; phase++) {
        pack_indices = Kokkos::subview(pack_indices_all,phase,Kokkos::ALL());
        if(proc_grid[phase/2]>1) { ← Kokkos::parallel_for
            Kokkos::parallel_for("CommMPI::halo_update_pack",
                Kokkos::RangePolicy<TagHaloUpdatePack, Kokkos::IndexType<T_INT> >(0,proc_num_send[phase]),
                *this);
            MPI_Request request;
            MPI_Status status;
            MPI_Irecv(unpack_buffer.data(),proc_num_recv[phase]*sizeof(T_X_FLOAT)*3/sizeof(int),MPI_INT, proc_neighbors_recv[phase],100002,MPI_COMM_WORLD,&request);
            MPI_Send (pack_buffer.data(),proc_num_send[phase]*sizeof(T_X_FLOAT)*3/sizeof(int),MPI_INT, proc_neighbors_send[phase],100002,MPI_COMM_WORLD);
            s = *system;
            MPI_Wait(&request,&status);
            const int count = proc_num_recv[phase];
            if(unpack_buffer_update.extent(0)<count) {
                unpack_buffer_update = t_buffer_update((T_X_FLOAT*)unpack_buffer.data(),count);
            }
            Kokkos::parallel_for("CommMPI::halo_update_unpack",
                Kokkos::RangePolicy<TagHaloUpdateUnpack, Kokkos::IndexType<T_INT> >(0,proc_num_recv[phase]),
                *this); ← popRegion
        } else {
            //printf("HaloUpdateCopy: %i %i %i\n",phase,proc_num_send[phase],pack_indices.extent(0));
            Kokkos::parallel_for("CommMPI::halo_update_self",
                Kokkos::RangePolicy<TagHaloUpdateSelf, Kokkos::IndexType<T_INT> >(0,proc_num_send[phase]),
                *this);
        }
        N_ghost += proc_num_recv[phase];
    }
    Kokkos::Profiling::popRegion(); ←
};
```

# ExaMinIMD: TAU Phase

Name	Exclusive TIME	Inclusive TIME	Calls	Child Calls
.TAU application	0.143	96.743	1	832
Comm::exchange	0.001	0.967	6	142
Comm::exchange_halo	0.001	4.702	6	184
Comm::update_halo	0.004	31.347	95	1,330
Kokkos::parallel_for CommMPI::halo_update_pack [device=0]	0.002	0.506	190	190
Kokkos::parallel_for CommMPI::halo_update_self [device=0]	0.003	0.597	380	380
Kokkos::parallel_for CommMPI::halo_update_unpack [device=0]	0.002	0.97	190	190
MPI_Irecv()	0.001	0.001	190	0
MPI_Send()	29.268	29.268	190	0
MPI_Wait()	0.001	0.001	190	0
OpenMP_Implicit_Task	0.041	1.985	760	760
OpenMP_Parallel_Region parallel_for<Kokkos::RangePolicy<CommMPI::Ta	0	0.504	190	190
OpenMP_Parallel_Region parallel_for<Kokkos::RangePolicy<CommMPI::Ta	0.08	0.968	190	190
OpenMP_Parallel_Region void Kokkos::parallel_for<Kokkos::RangePolicy<	0.001	0.594	380	380
OpenMP_Sync_Region_Barrier parallel_for<Kokkos::RangePolicy<CommM	0.489	0.489	190	0
OpenMP_Sync_Region_Barrier parallel_for<Kokkos::RangePolicy<CommM	0.875	0.875	190	0
OpenMP_Sync_Region_Barrier void Kokkos::parallel_for<Kokkos::RangePol	0.58	0.58	380	0

Comm::update\_halo phase in TAU ParaProf's Thread Statistics Table

# ExaMinIMD: ParaProf Node Window

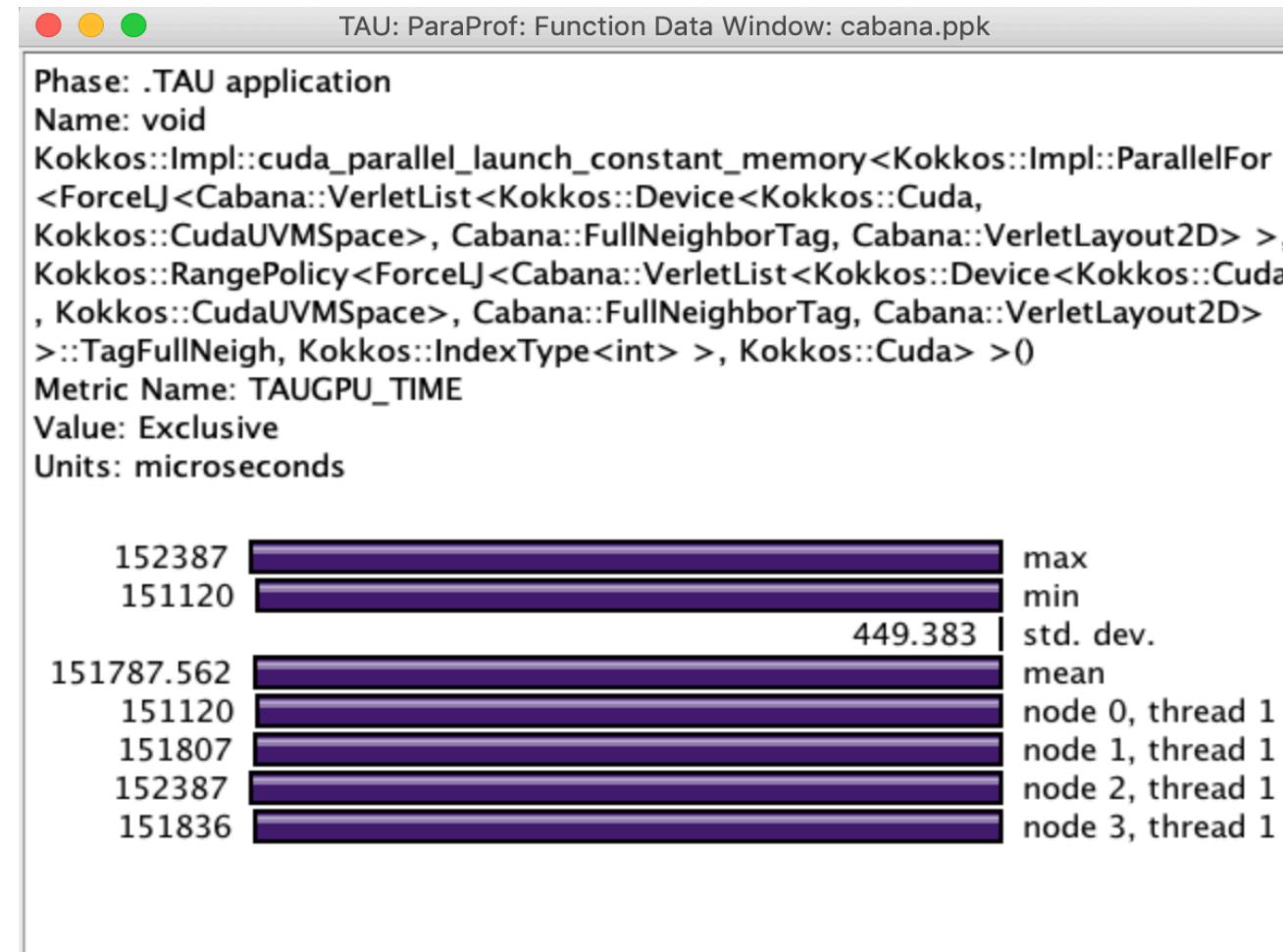


# Event-based Sampling (EBS): CabanaMD with Kokkos

TAU: ParaProf: Statistics for: node 0, thread 0 - cabana.ppk					
	Name	Exclusive...	Inclusive...	Calls	Child Calls
▼ .TAU application					
▼ Comm::update_halo					
▼ [CONTEXT] Comm::update_halo					
■ [SAMPLE] __strlen_power8 [{} {} 0]		0.129	1.634	95	21,755
■ [SAMPLE] Kokkos::Impl::SharedAllocationRecord<void, void>::increment(Kokkos::Impl::SharedAllocationRecord<void, void>*) [/g/g20/reeve5/bin/CabanaMD]		0.03	0.03	1	0
► cudaDeviceSynchronize		0.991	0.991	3,043	0
▼ [CONTEXT] .TAU application					
▼ [SUMMARY] LAMMPS_RandomVelocityGeom::reset(int, double*) [/g/g20/reeve5/pr/CabanaMD/src/input.h]		0.27	0.27	9	0
■ [SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [/g/g20/reeve5/pr/CabanaMD/src/input.h] {128}		0.09	0.09	3	0
■ [SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [/g/g20/reeve5/pr/CabanaMD/src/input.h] {129}		0.09	0.09	3	0
■ [SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [/g/g20/reeve5/pr/CabanaMD/src/input.h] {130}		0.06	0.06	2	0
■ [SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [/g/g20/reeve5/pr/CabanaMD/src/input.h] {140}		0.03	0.03	1	0
▼ [SUMMARY] Input::create_lattice(Comm*) [/g/g20/reeve5/pr/CabanaMD/src/input.cpp]		0.15	0.15	5	0
■ [SAMPLE] Input::create_lattice(Comm*) [/g/g20/reeve5/pr/CabanaMD/src/input.cpp] {745}		0.03	0.03	1	0
■ [SAMPLE] Input::create_lattice(Comm*) [/g/g20/reeve5/pr/CabanaMD/src/input.cpp] {665}		0.03	0.03	1	0
■ [SAMPLE] Input::create_lattice(Comm*) [/g/g20/reeve5/pr/CabanaMD/src/input.cpp] {721}		0.03	0.03	1	0
■ [SAMPLE] Input::create_lattice(Comm*) [/g/g20/reeve5/pr/CabanaMD/src/input.cpp] {713}		0.03	0.03	1	0
■ [SAMPLE] Input::create_lattice(Comm*) [/g/g20/reeve5/pr/CabanaMD/src/input.cpp] {714}		0.03	0.03	1	0
■ [SAMPLE] reference<unsigned int, unsigned int, unsigned int> [/g/g20/reeve5/build_v100/install/kokkos/include/impl/Kokkos_ViewMapping.hpp} {2740}]		0.06	0.06	2	0
■ [SAMPLE] unsigned long Kokkos::Impl::ViewOffset<Kokkos::Impl::ViewDimension<Oul, 16ul, 3ul>, Kokkos::LayoutCabanaSlice<176, 16, 3, 0, 0, 0, 0, 0>, void>::		0.03	0.03	1	0
▼ [SUMMARY] LAMMPS_RandomVelocityGeom::uniform0 [/g/g20/reeve5/pr/CabanaMD/src/input.h]		0.03	0.03	1	0
■ [SAMPLE] LAMMPS_RandomVelocityGeom::uniform0 [/g/g20/reeve5/pr/CabanaMD/src/input.h] {93}]		0.03	0.03	1	0
■ Comm::exchange		0.024	0.392	6	3,371
■ MPI_Finalize()		0.367	0.369	1	68
■ Comm::exchange_halo		0.026	0.351	6	4,772
■ MPI_Init()		0.323	0.323	1	0
■ Cabana::Verlet		0.004	0.256	6	438
■ Kokkos::parallel_for ForceLJCabanaNeigh::compute [device=0]		0.002	0.164	101	606
▼ MPI_Allreduce0		0.082	0.082	39	0
▼ [CONTEXT] MPI_Allreduce0		0	0.09	3	0
■ [SAMPLE] __GI_sched_yield [{} {} 0]		0.03	0.03	1	0
■ [SAMPLE] pthread_spin_unlock [/usr/lib64/libpthread-2.17.so] {0}}		0.03	0.03	1	0
■ [SAMPLE] pthread_spin_lock [/usr/lib64/libpthread-2.17.so] {0}}		0.03	0.03	1	0
■ Kokkos::parallel_for Kokkos::View::initialization [device=0]		0.001	0.072	35	170
■ Kokkos::parallel_for Kokkos::ViewFill-3D [device=0]		0.001	0.047	101	303
■ Kokkos::parallel_reduce ForceLJCabanaNeigh::compute_energy [device=0]		0	0.042	11	77
■ cudaLaunchKernel		0.015	0.028	527	1,581

Event-based sampling (EBS) with Kokkos API

# CabanaMD: CUDA Events



# 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

- *Level Zero (Intel GPUs e.g., Gen12LP in TigerLake)*

- Level Zero profiling interface
  - Track timings of kernels and Level Zero runtime functions

- *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 + CUDA*
  - PMPI + CUPTI interfaces
- *OpenCL + ROCm*
  - Rocprofiler + OpenCL instrumentation interfaces
- *Kokkos + OpenMP*
  - Kokkos profiling API + OMPT to transparently track events
- *Kokkos + pthread + MPI*
  - Kokkos + pthread wrapper interposition library + PMPI layer
- *Python + CUDA + MPI*
  - Python + CUPTI + pthread profiling interfaces (e.g., Tensorflow, PyTorch) + MPI
- *MPI + OpenCL*
  - PMPI + OpenCL profiling interfaces

# TAU Execution Command (tau\_exec)

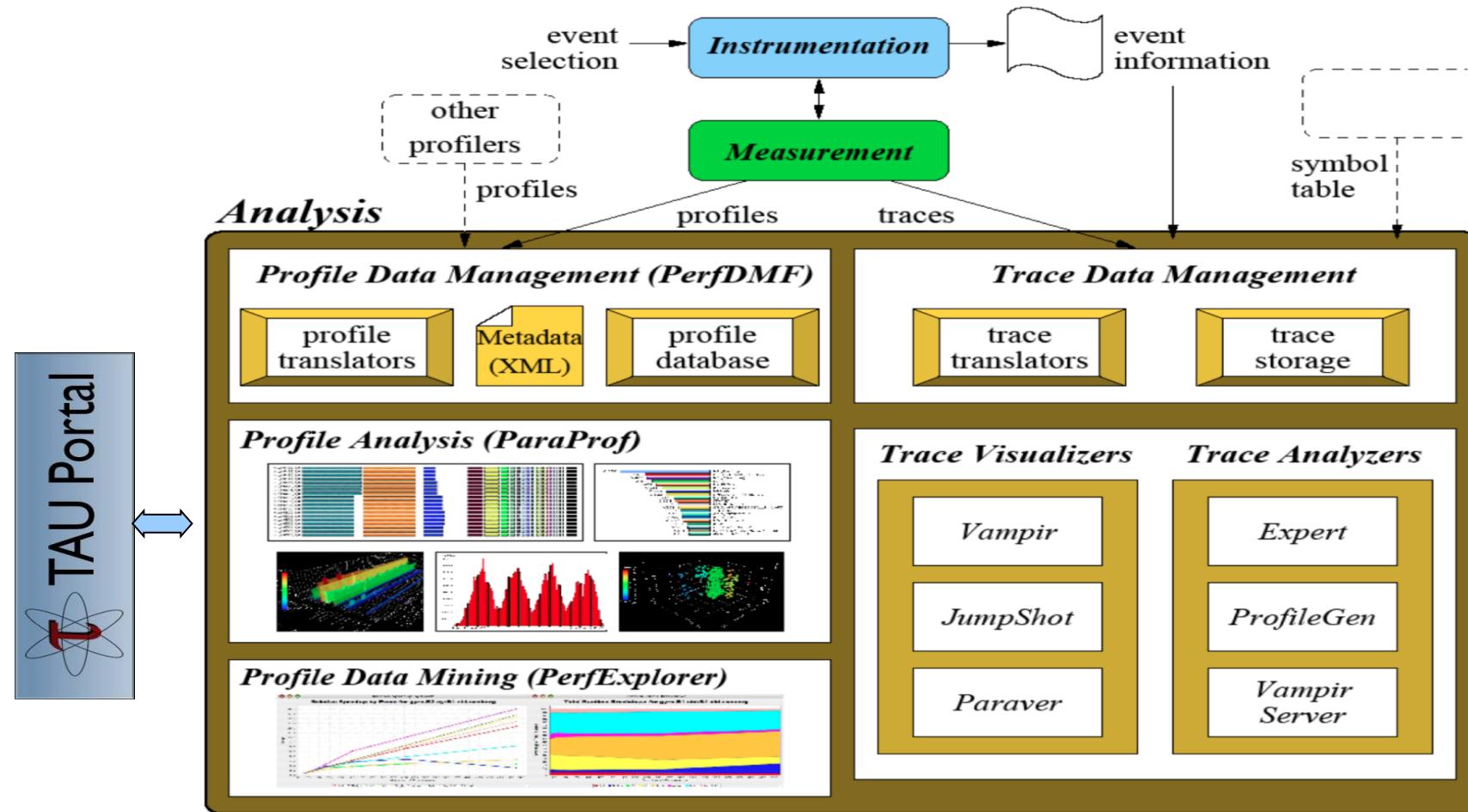
---

- Uninstrumented execution
  - % mpirun -np 256 ./a.out
- Track GPU operations
  - % mpirun -np 256 tau\_exec --rocm ./a.out
  - % mpirun -np 256 tau\_exec --level\_zero ./a.out
  - % mpirun -np 256 tau\_exec --cupti ./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,v5,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 -ebs\_source=<PAPI\_COUNTER> -ebs\_period=<overflow\_count> -ebs\_resolution=<file | function | line>

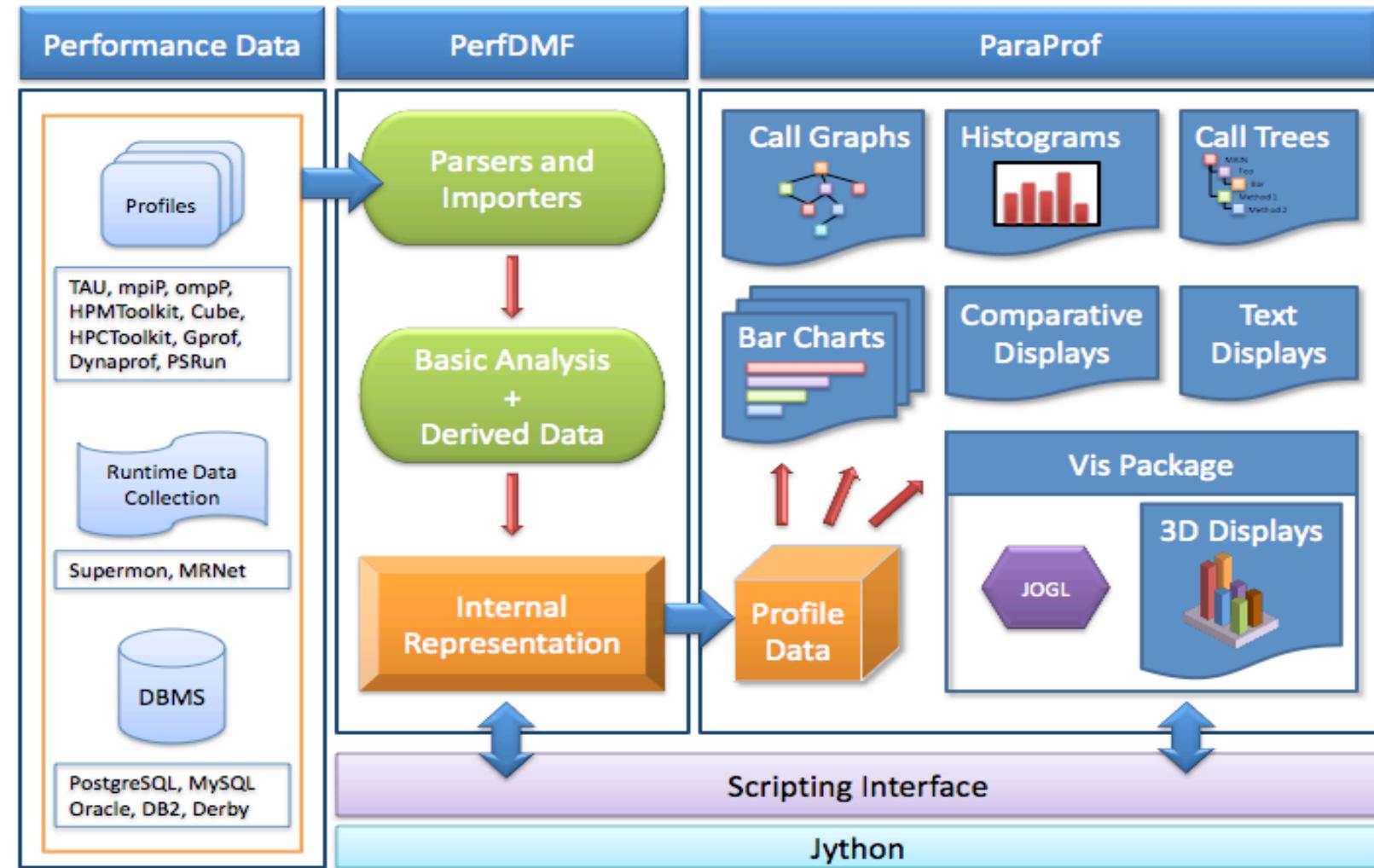
---

# TAU's Analysis Tools: ParaProf

# TAU Analysis



# ParaProf Profile Analysis Framework



# TAU Analysis Tools: paraprof

- Launch paraprof

```
% paraprof
```

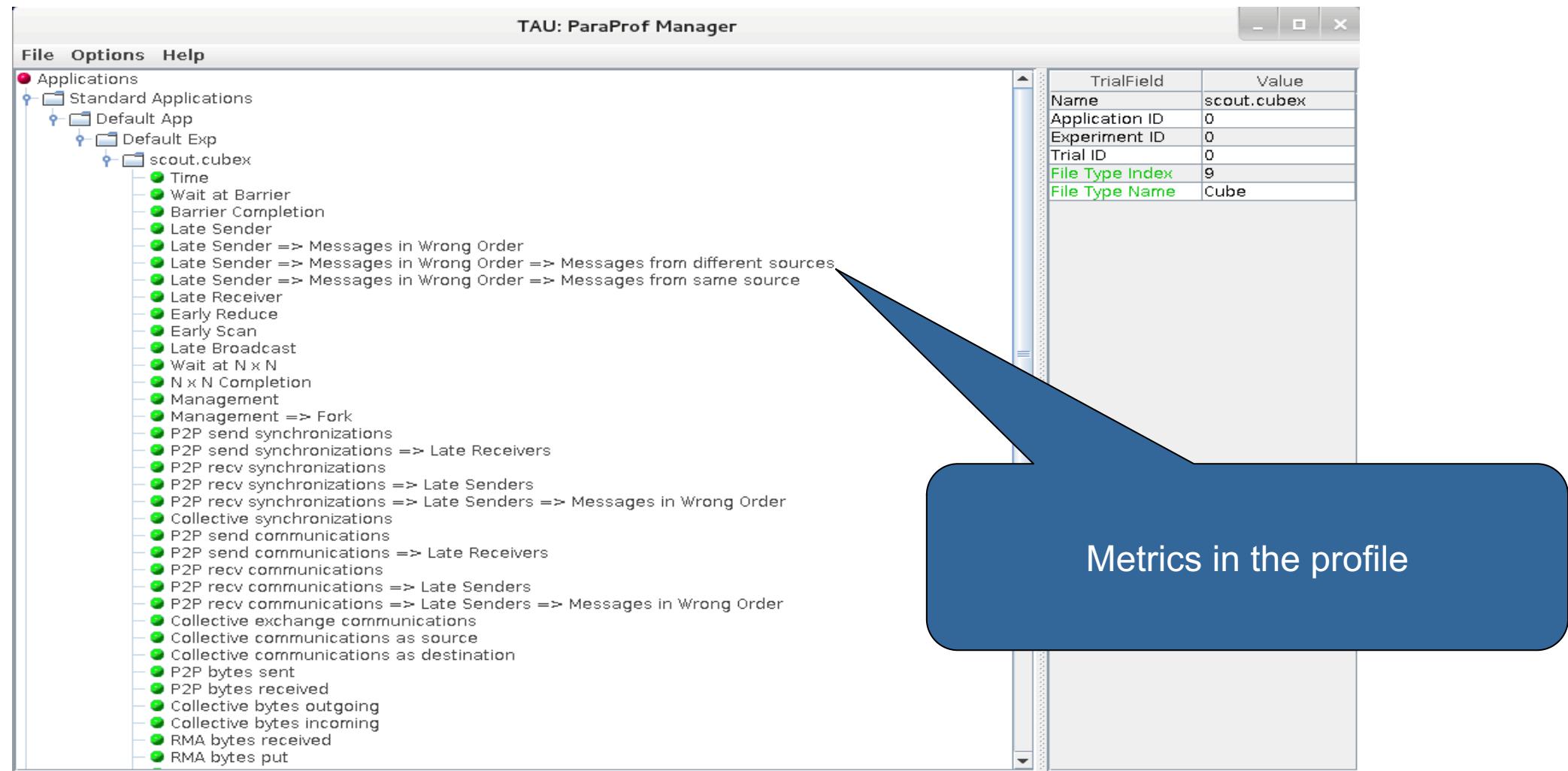
Metric

TAU: ParaProf Manager

The screenshot shows the TAU ParaProf Manager application window. On the left, there is a hierarchical file tree under the 'Applications' section. The tree includes 'Standard Applications', 'Default App', 'Default Exp', and a selected item 'bt\_ompt.ppk'. A blue arrow points from the word 'Metric' in the bottom-left corner towards this tree. To the right of the tree is a table titled 'TAU: ParaProf Manager' containing various experimental parameters.

TrialField	Value
Name	bt_ompt.ppk
Application ID	0
Experiment ID	0
Trial ID	0
CPU Cores	8
CPU MHz	2600.000
CPU Type	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
CPU Vendor	GenuineIntel
CWD	/scratch/sameer/NPB3.3-MZ-MPI/bin
Cache Size	20480 KB
Command Line	./bt-mz_C.8
Executable	/scratch/sameer/NPB3.3-MZ-MPI/bin/bt-mz_C.8
File Type Index	0
File Type Name	ParaProf Packed Profile
Hostname	frog9
Local Time	2015-05-18T00:37:38+02:00
MPI Processor Name	frog9
Memory Size	65944056 kB
Node Name	frog9
OMP_CHUNK_SIZE	1
OMP_DYNAMIC	off
OMP_MAX_THREADS	4
OMP_NESTED	off
OMP_NUM_PROCS	4
OMP_SCHEDULE	UNKNOWN
OS Machine	x86_64
OS Name	Linux
OS Release	2.6.32-279.5.2.b16.Bull.33.x86_64
OS Version	#1 SMP Sat Nov 10 01:48:00 CET 2012

# ParaProf Manager Widow: scout.cubex



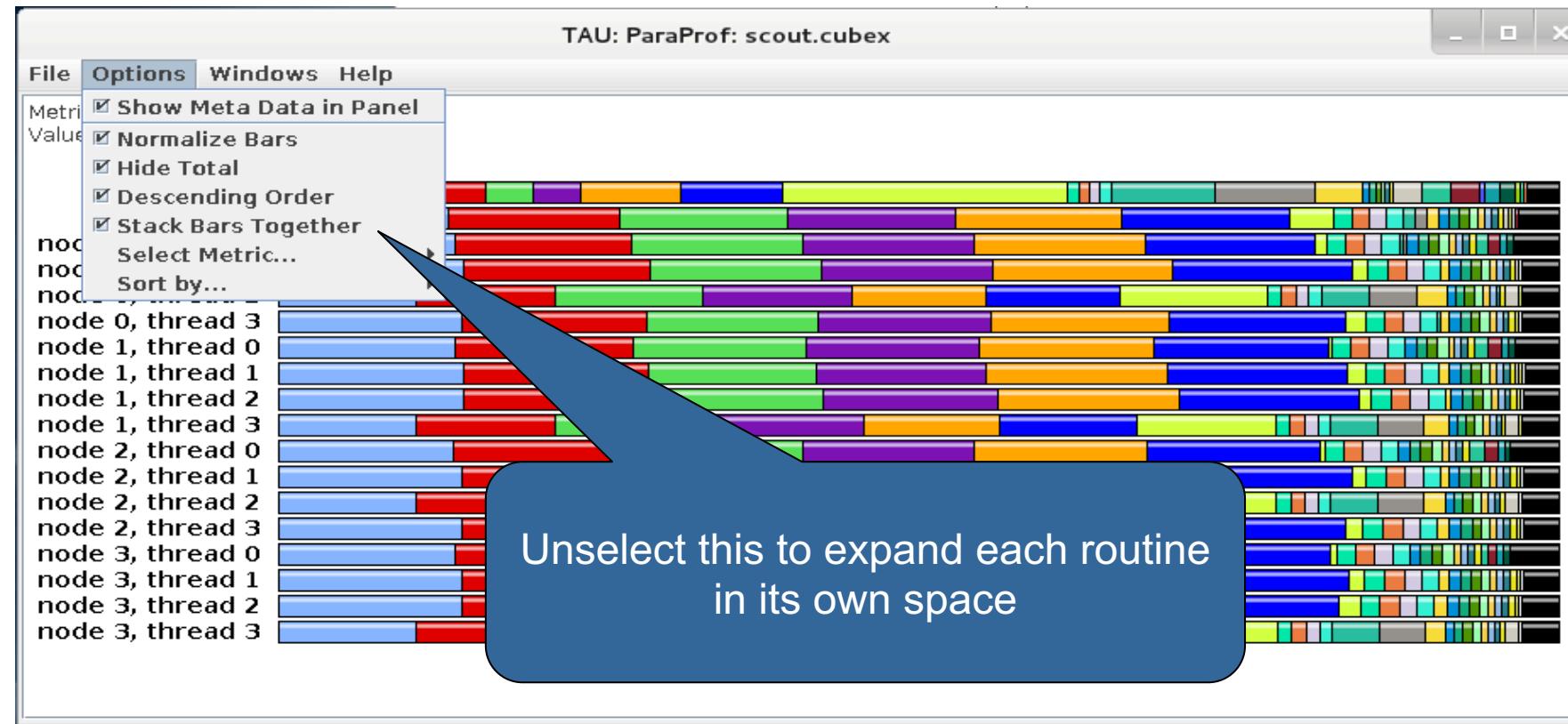
# Paraprof main window



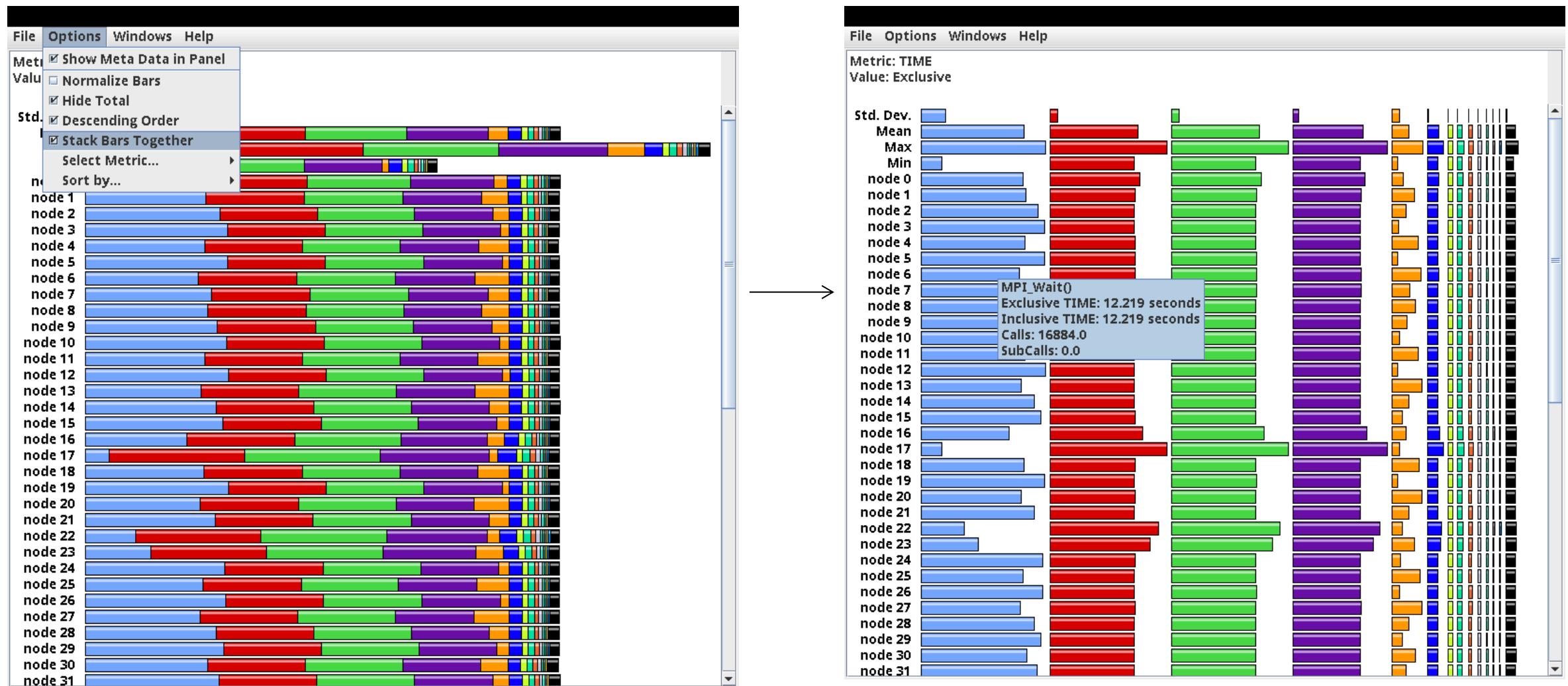
Colors represent code regions

Options -> uncheck Stack Bars Together

# Paraprof main window



# ParaProf Profile Browser

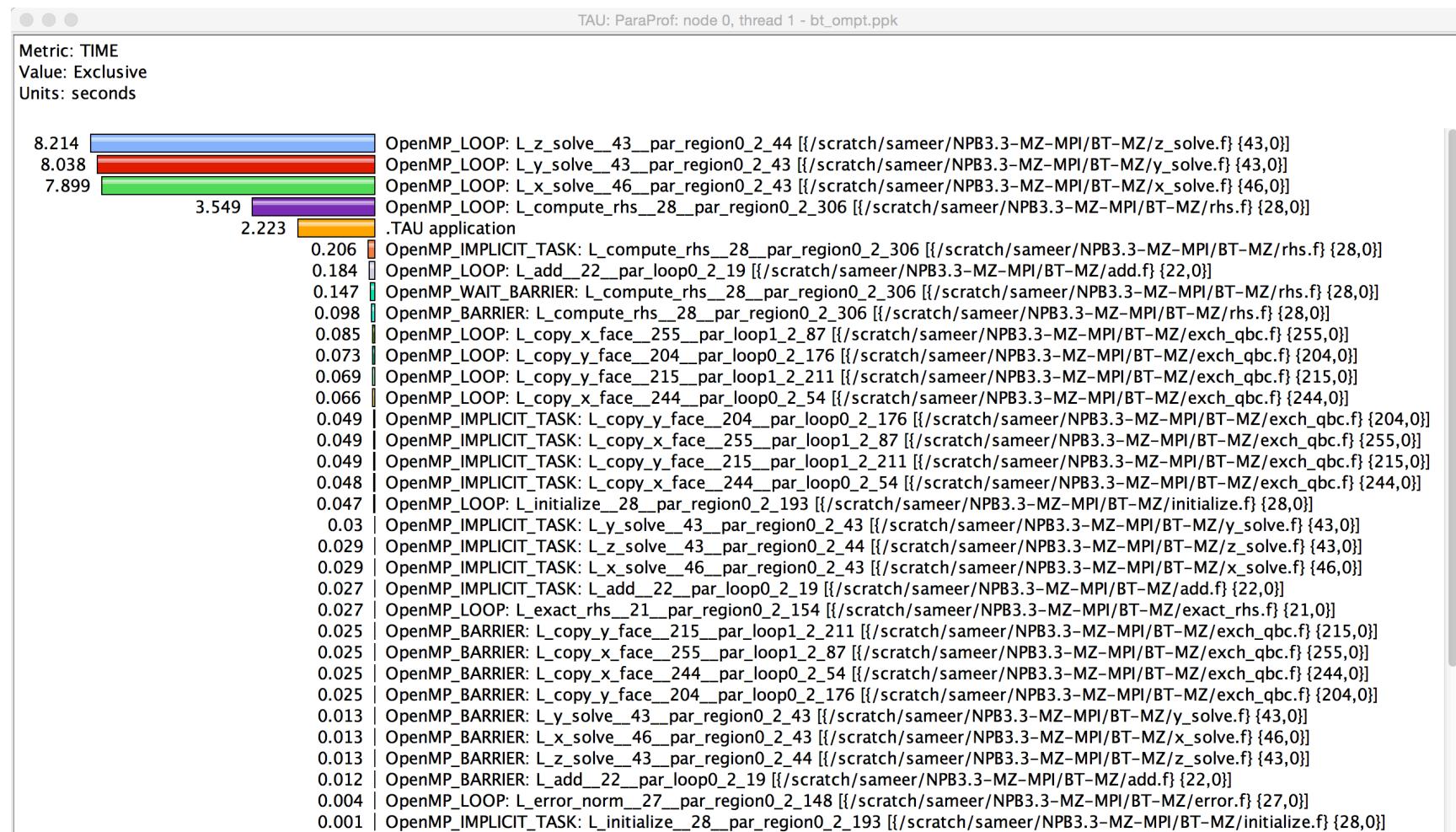


# Paraprof main window



Each routine occupies its own space.  
Can see the extent of imbalance  
across all threads.

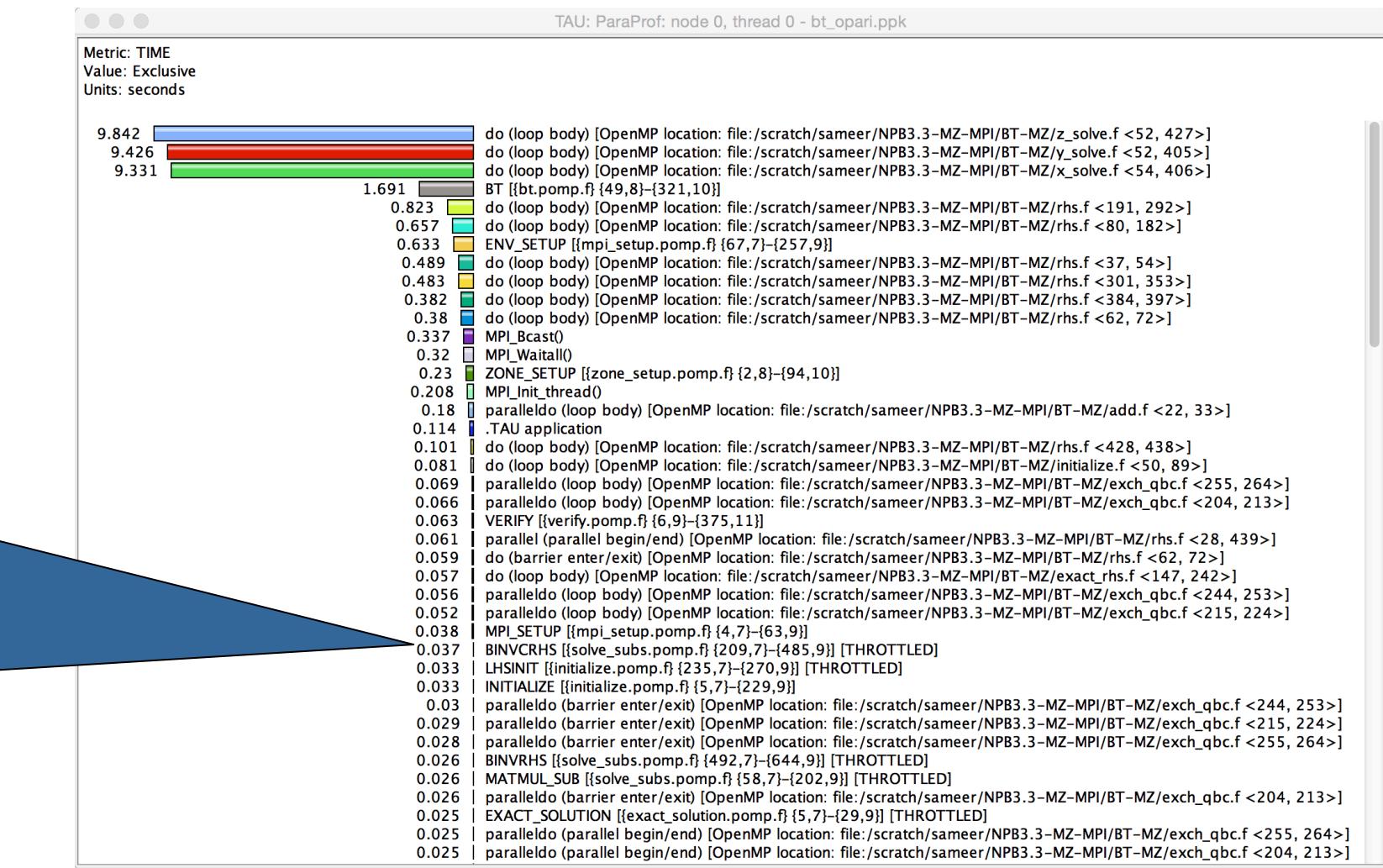
# Paraprof node window (function barchart window)



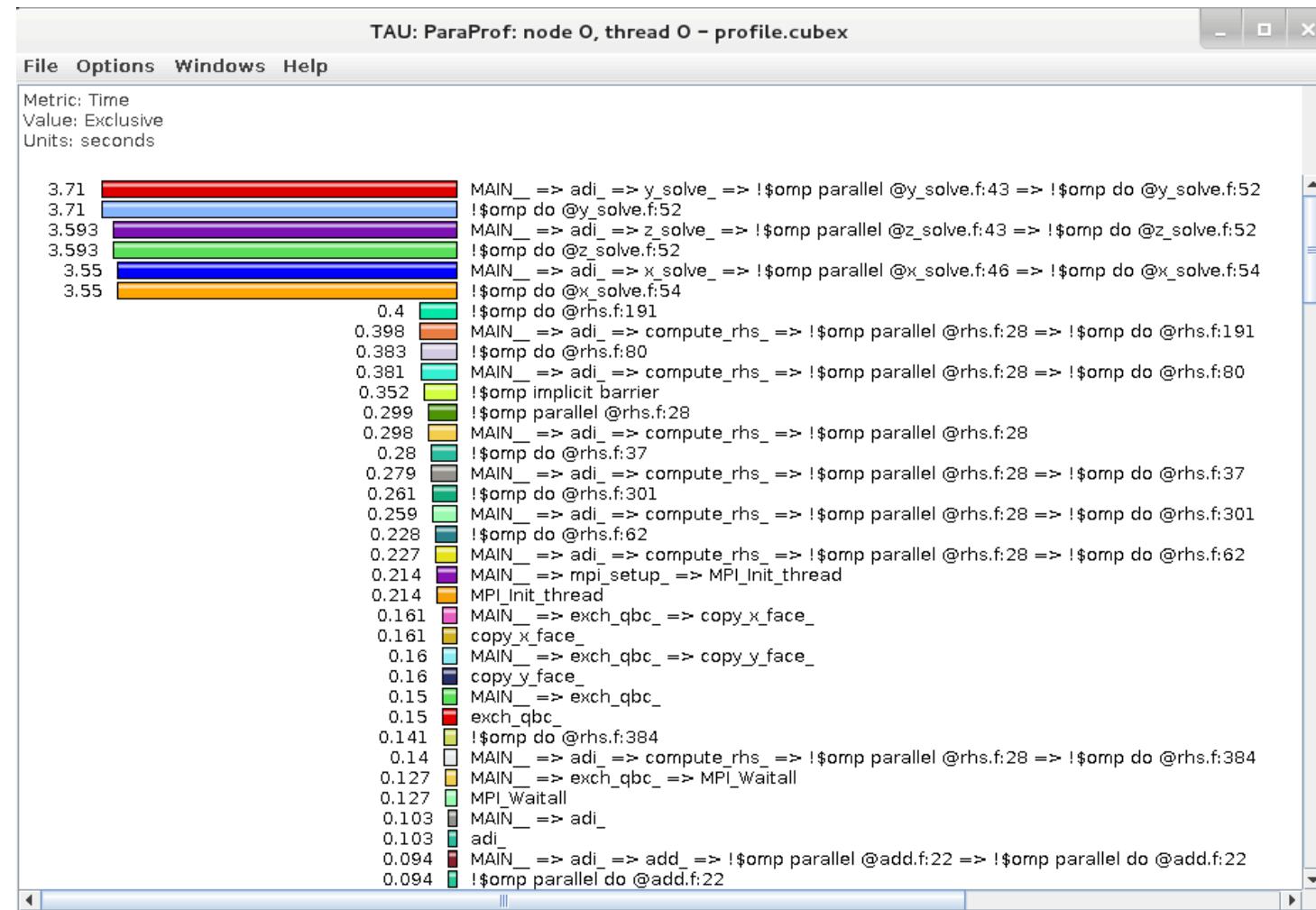
Exclusive time spent in each code region (OpenMP loop) is shown here for MPI rank 0 thread 1

# Instrumenting Source Code with PDT and Opari

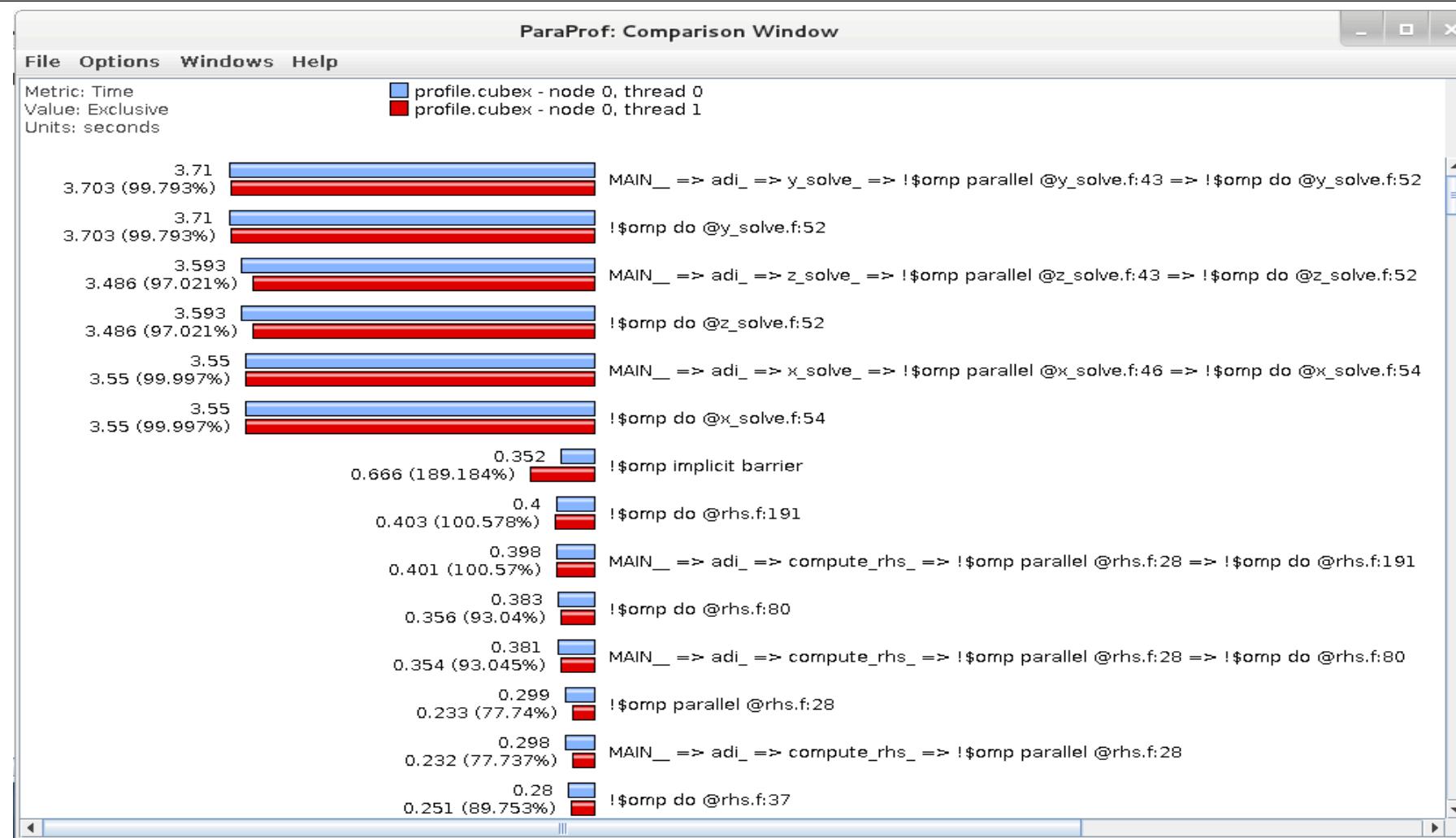
Frequently executing lightweight routines are automatically throttled at runtime. Reduces runtime dilation.



# ParaProf: Node view in a callpath profile



# ParaProf: Add thread to comparison window



# Paraprof Thread Statistics Table with TAU\_SAMPLING=1



# ParaProf: Thread Statistics Table

TAU: ParaProf: Statistics for: node 0, thread 0 – scout.cubex

File Options Windows Help

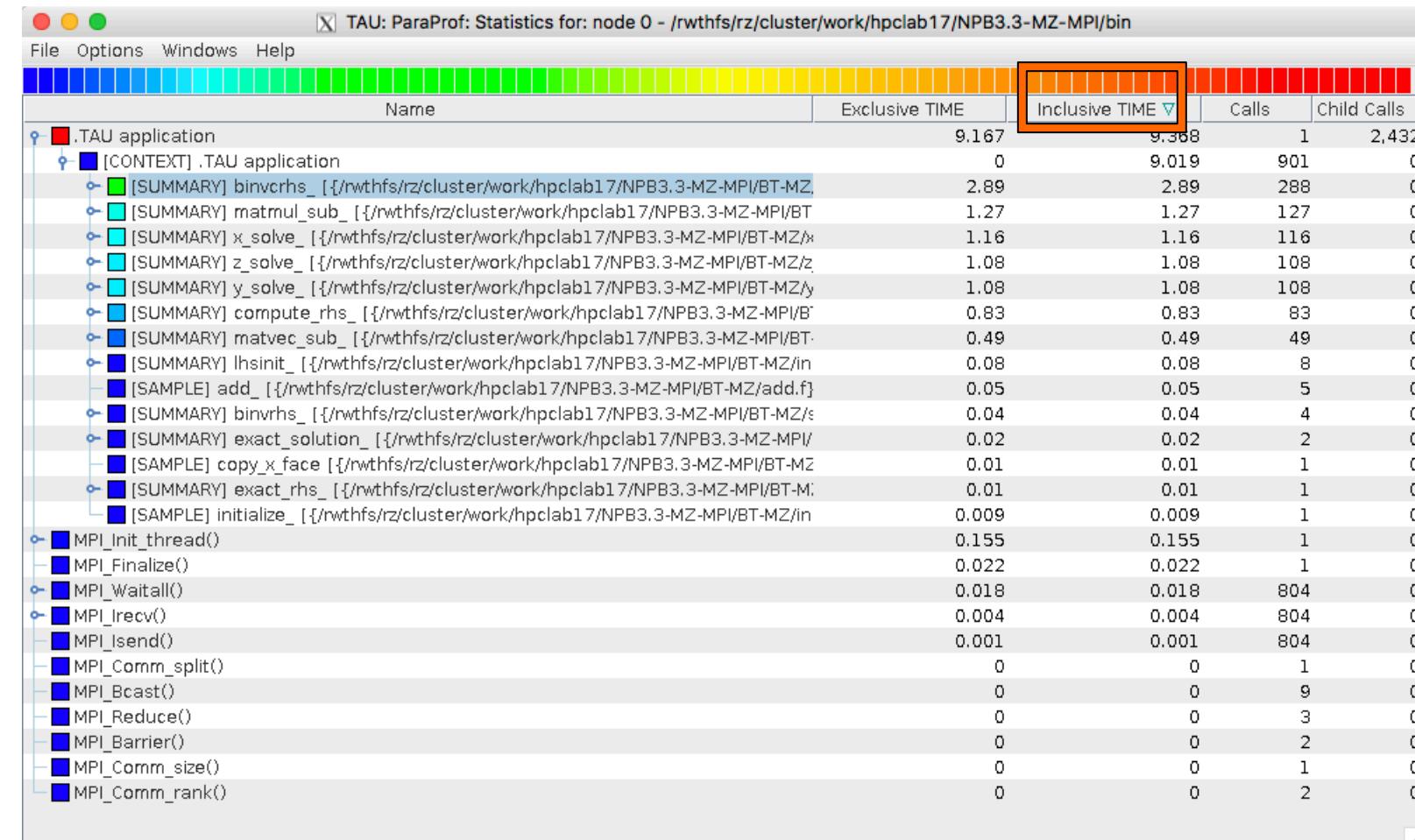
Time

Name	Exclusive Time ▼	Inclusive Time	Calls	Child Calls
!\$omp do @y_solve.f:52	5.817	5.817	3,216	0
!\$omp do @z_solve.f:52	5.657	5.657	3,216	0
!\$omp do @x_solve.f:54	5.609	5.609	3,216	0
!\$omp do @rhs.f:191	0.609	0.609	3,232	0
!\$omp do @rhs.f:80	0.583	0.583	3,232	0
MPI_Waitall	0.402	0.402	603	0
!\$omp implicit barrier	0.402	0.402	0	0
!\$omp do @rhs.f:301	0.36	0.36	0	0
!\$omp implicit barrier	0.026	0.026	0	0
!\$omp implicit barrier	0	0	0	0
!\$omp do @rhs.f:37	0.343	0.343	0	0
!\$omp do @rhs.f:62	0.225	0.225	0	0
!\$omp implicit barrier	0.004	0.004	3,216	0
!\$omp implicit barrier	0	0	16	0
MPI_Init_thread	0.218	0.218	1	0
!\$omp do @rhs.f:384	0.199	0.199	3,232	0
!\$omp parallel do @add.f:22	0.099	0.111	3,216	3,216
!\$omp do @rhs.f:428	0.069	0.069	3,232	0
MPI_Isend	0.043	0.043	603	0
!\$omp do @initialize.f:50	0.04	0.04	32	0
!\$omp parallel @rhs.f:28	0.03	2.536	3,232	51,712
!\$omp parallel do @exch_qbc.f:215	0.021	0.029	6,432	6,432
!\$omp parallel do @exch_qbc.f:255	0.02	0.033	6,432	6,432
!\$omp parallel @exch_qbc.f:255	0.02	0.053	6,432	6,432
!\$omp parallel @exch_qbc.f:244	0	0	0	0

Click to sort by a given metric, drag and move to rearrange columns

# ParaProf

- Click on Columns:
  - to sort by incl time
- Open binvcrhs
- Click on Sample

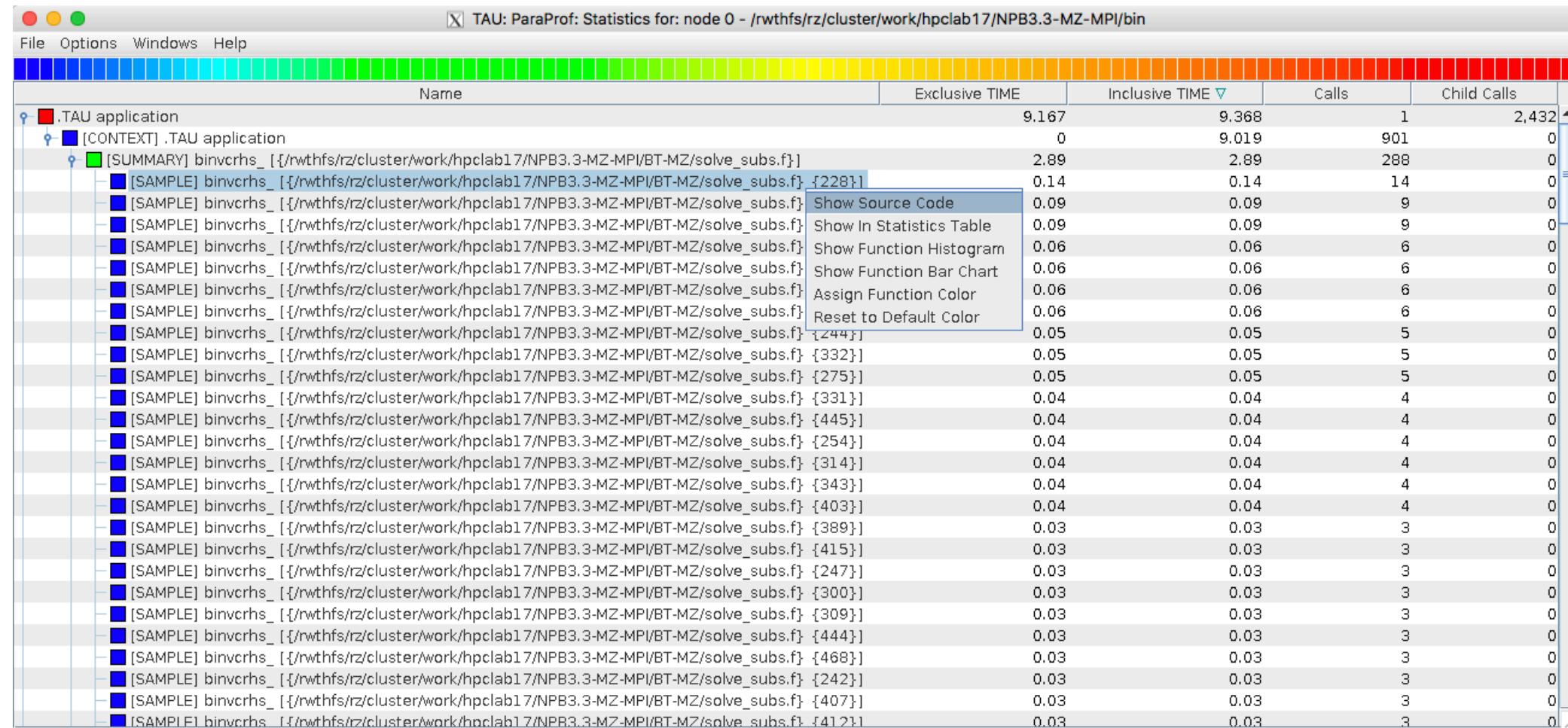


# Paraprof Thread Statistics Table

Name	Exclusive TIME	Inclusive TIME ▾	Calls	Child Calls
▼ .TAU application	1.754	36.26	1	88,049
▼ OpenMP_PARALLEL_REGION: L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {43,0}	0.061	8.692	6,432	12,864
▼ OpenMP_IMPLICIT_TASK: L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {43,0}	0.04	8.568	6,432	6,432
▼ OpenMP_LOOP: L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {43,0}	8.528	8.528	6,432	0
▼ [CONTEXT] OpenMP_LOOP: L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {43,0}	0	9.23	847	0
▼ [SUMMARY] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f]	3.67	3.67	340	0
▼ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f]	3.67	3.67	340	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {419}	0.22	0.22	21	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {58}	0.17	0.17	16	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {418}	0.16	0.16	12	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {123}	0.11	0.11	11	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {193}	0.08	0.08	5	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {126}	0.07	0.07	7	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {247}	0.07	0.07	6	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {158}	0.06	0.06	5	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {313}	0.06	0.06	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {230}	0.06	0.06	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {308}	0.05	0.05	3	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {191}	0.05	0.05	3	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {81}	0.05	0.05	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {301}	0.05	0.05	5	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {67}	0.05	0.05	5	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {175}	0.04	0.04	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {89}	0.04	0.04	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {55}	0.04	0.04	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {275}	0.04	0.04	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {129}	0.04	0.04	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {168}	0.04	0.04	4	0
█ [SAMPLE] L_z_solve_43_par_region0_2_44 [/scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/z_solve.f] {238}	0.04	0.04	4	0

Right click here and choose “Show Source Code” for a sample

# ParaProf

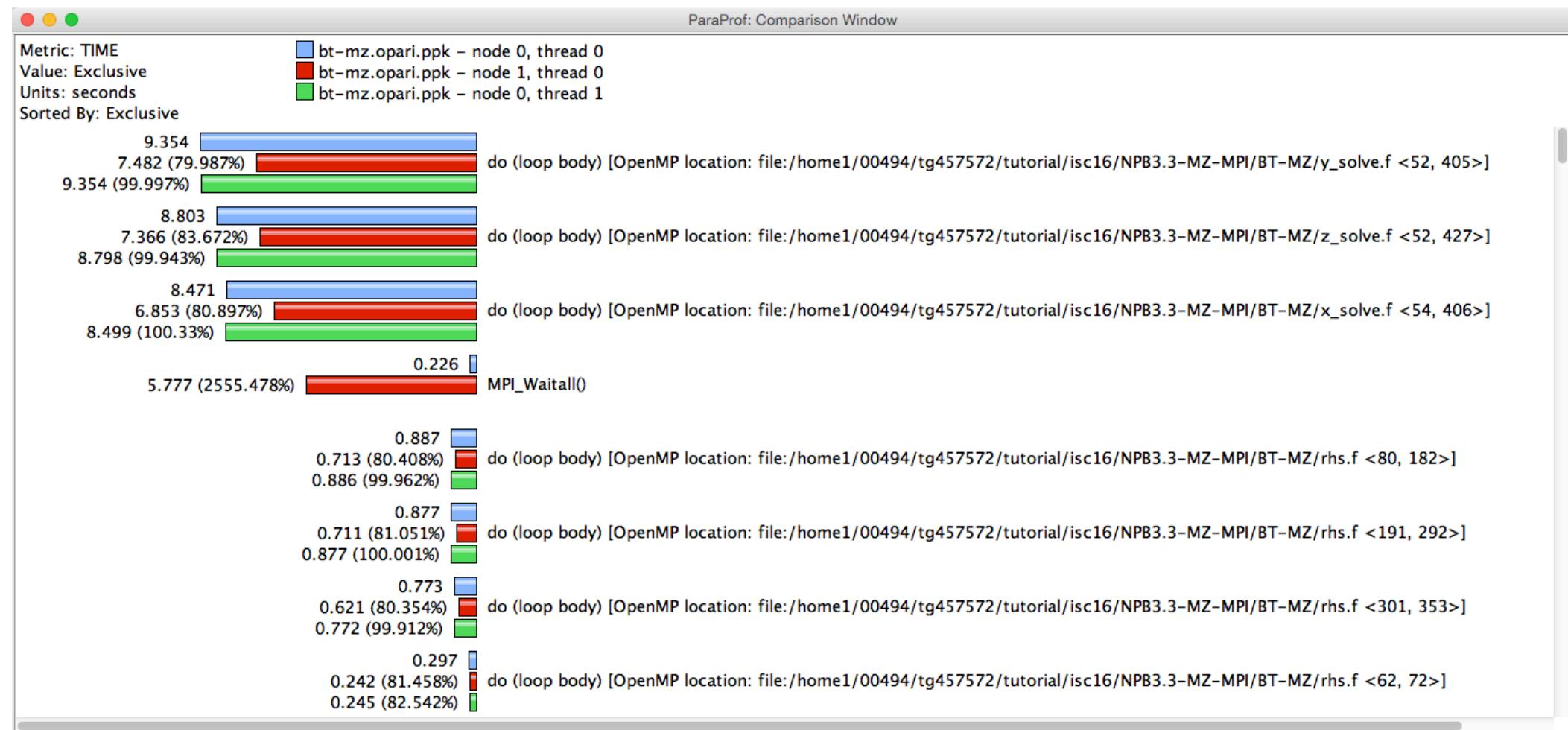


# Statement Level Profiling with TAU

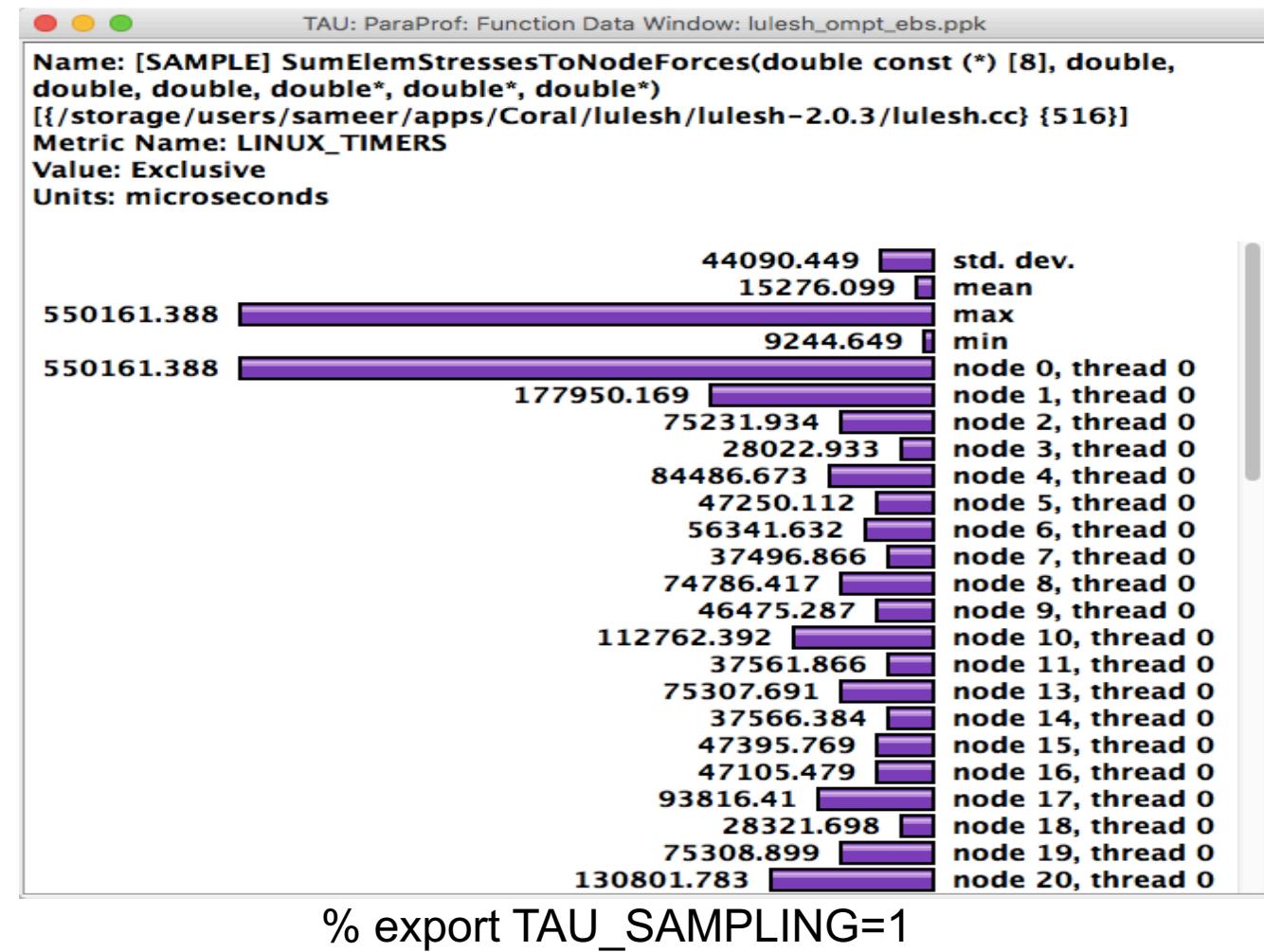
```
TAU: ParaProf: Source Browser: /scratch/sameer/NPB3.3-MZ-MPI/BT-MZ/x_solve.f
File Help
353         call matmul_sub(lhs(1,1,aa,i),
354             >                      lhs(1,1,cc,i-1),
355             >                      lhs(1,1,bb,i))
356
357
358 c-      multiply c(i,j,k) by b_inverse and copy back to c
359 c      multiply rhs(1,j,k) by b_inverse(1,j,k) and copy to rhs
360 c-
361         call binvcrhs( lhs(1,1,bb,i),
362             >                      lhs(1,1,cc,i),
363             >                      rhs(1,i,j,k) )
364
365         enddo
366
367 c-      rhs(isize) = rhs(isize) - A*rhs(isize-1)
368 c-
369         call matvec_sub(lhs(1,1,aa,isize),
370             >                      rhs(1,isize-1,j,k),rhs(1,isize,j,k))
371
372 c-      c(isize) = c(isize) - C(isize-1)*A(isize)
373 c-
374         call matmul_sub(lhs(1,1,aa,isize),
375             >                      lhs(1,1,cc,isize-1),
376             >                      lhs(1,1,bb,isize))
377
378 c-      multiply rhs() by b_inverse() and copy to rhs
379 c-
380         call binvrhs( lhs(1,1,bb,isize),
381             >                      rhs(1,isize,j,k) )
382
383
384 c-      back solve: if last cell, then generate U(isize)=rhs(isize)
385 c      else assume U(isize) is loaded in un pack backsolve_info
386 c      so just use it
387 c      after call u(istart) will be sent to next cell
388 c-
389
390         do i=isize-1,0,-1
391             do m=1,BLOCK_SIZE
392                 do n=1,BLOCK_SIZE
393                     rhs(m,i,j,k) = rhs(m,i,j,k)
394                     >                      - lhs(m,n,cc,i)*rhs(n,i+1,j,k)
395
396                 enddo
397             enddo
398         enddo
399
400
401
402
```

Source location where samples are taken.  
Compute intensive region.

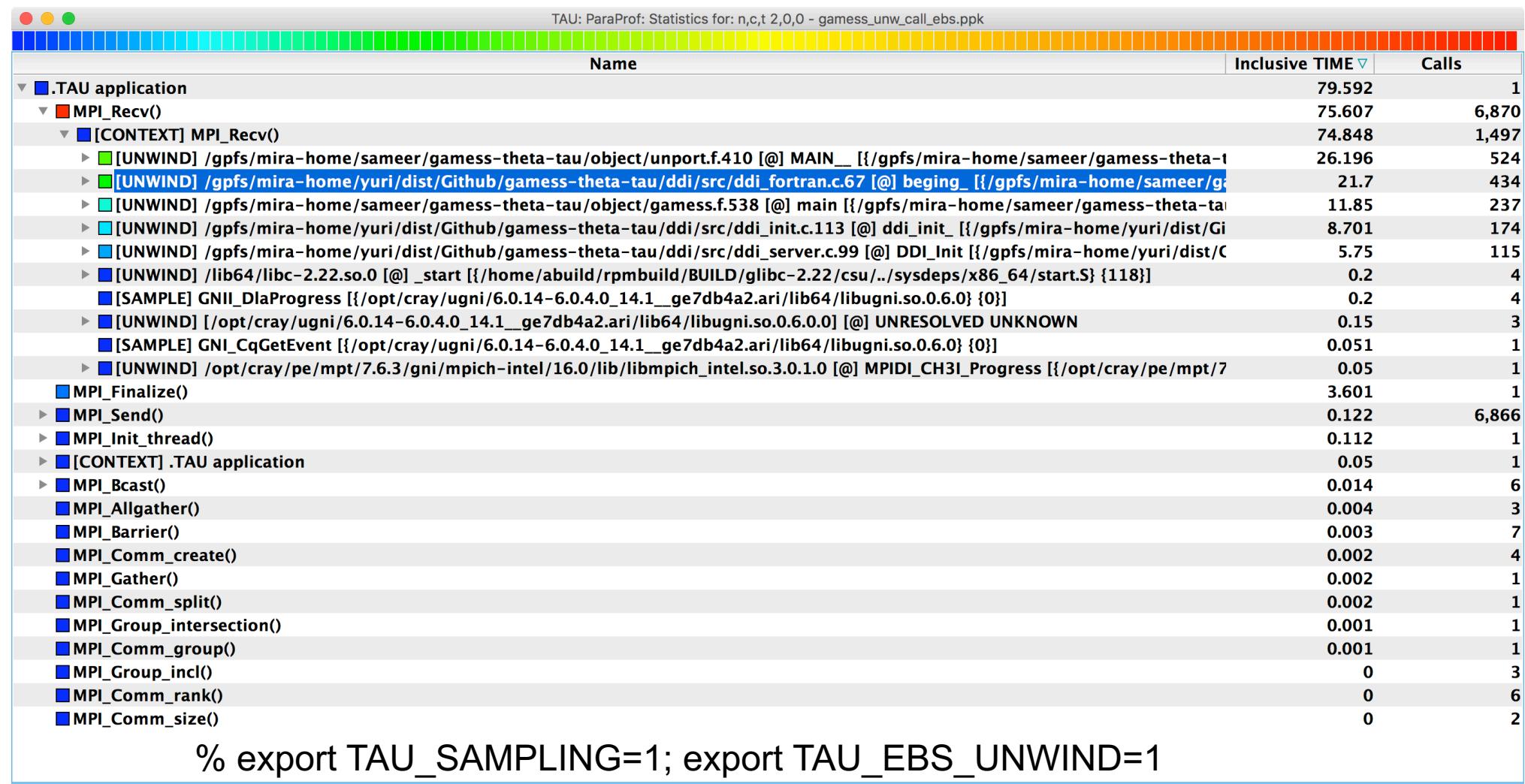
# ParaProf Comparison Window



# TAU – Event Based Sampling (EBS)



# Examples: Callstack Sampling in TAU



# UNWINDING CALLSTACKS

TAU: ParaProf: Statistics for: n,c,t 2,0,0 - gamess\_unw\_call\_ebs.ppk

	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-tau/object/unport.f.410}]	26.196	524	
▼ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_fortran.c.67 [@] begin_ [{/gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410}]	21.7	434	
▼ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_init.c.113 [@] ddi_init_ [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_init.c.113}]	21.7	434	
▼ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_server.c.99 [@] DDI_Init [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_server.c.99}]	21.7	434	
▼ [UNWIND] /lus/theta-fs0/software/perf-tools/tau/tau-2.26.3/src/Profile/TauMpi.c.2371 [@] DDI_Recv_request [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_recv.c.65}]	21.7	434	
▼ [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] MPI_Recv [{/lus/theta-fs0/software/perf-tools/tau/tau-2.26.3/src/Profile/TauMpi.c.2371}]	21.7	434	
▼ [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] PMPI_Recv [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	21.7	434	
▼ [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] MPIIDI_CH3I_Progress [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	21.45	429	
▼ [UNWIND] /opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0 [@] MPID_nem_gni_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	15.95	319	
█ [SAMPLE] GNI_SmsgGetNextWTag [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0}]	10.349	207	
█ [SAMPLE] GNI_CqGetEvent [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0} {0}]	5.6	112	
► [UNWIND] gni_poll.c.0 [@] MPID_nem_gni_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	5.25	105	
► [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] MPID_nem_gni_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	0.25	5	
► [UNWIND] UNRESOLVED [@] MPIIDI_CH3I_Progress [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	0.25	5	
► [UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/gamess.f.538 [@] main [{/gpfs/mira-home/sameer/gamess-theta-tau/object/gamess.f.538}]	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/Github/gamess-theta-tau/ddi/src/ddi_init.c.113}]	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/Github/gamess-theta-tau/ddi/src/ddi_server.c.99}]	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 [@] MPIIDI_CH3I_Progress [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	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	

```
% export TAU_SAMPLING=1; export TAU_EBS_UNWIND=1
```

# UNWINDING CALLSTACKS

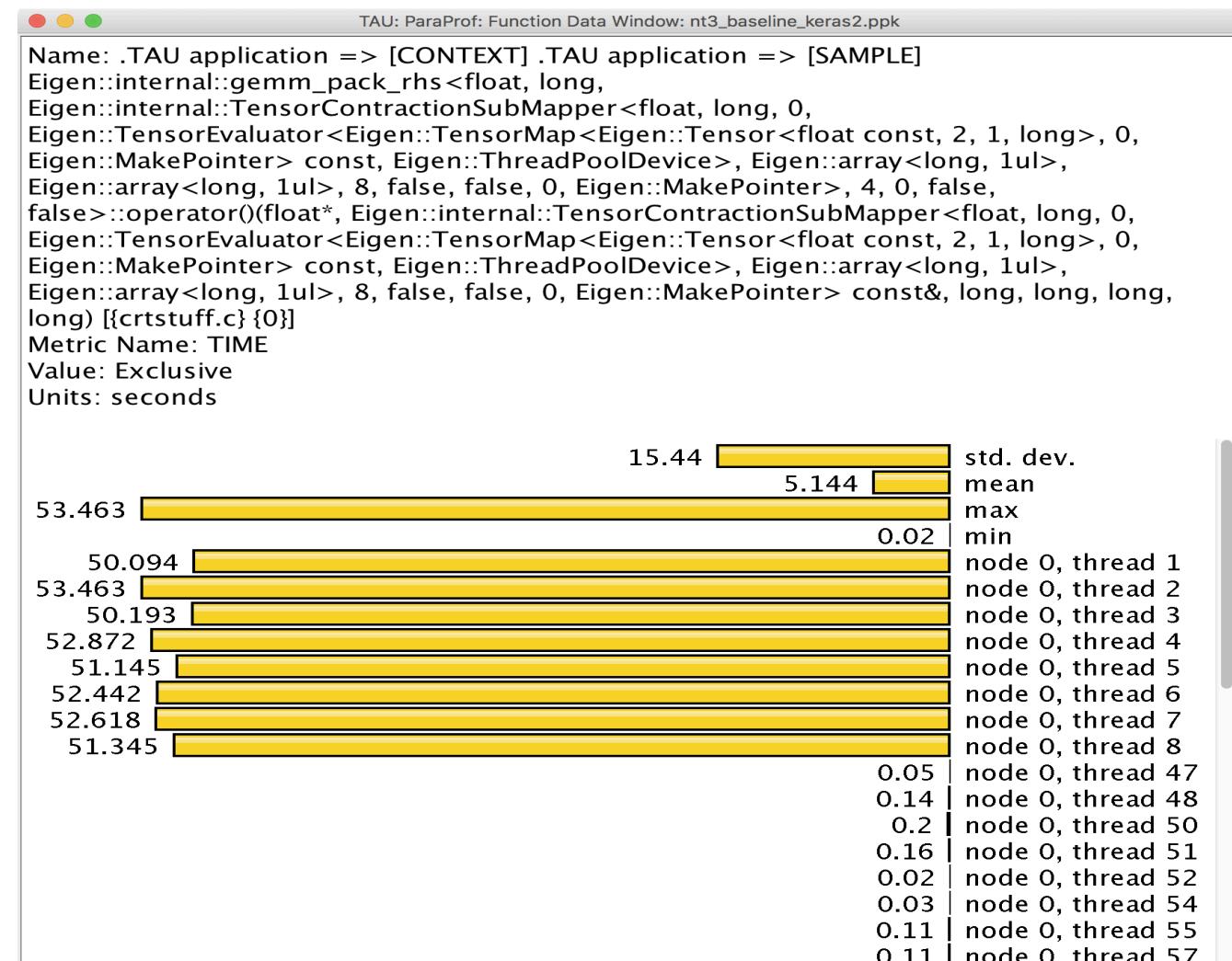
Name	Inclusive TIME ▼	Calls
▀ .TAU application	79.592	1
▀ MPI_Recv()	75.607	6,870
▀ [CONTEXT] MPI_Recv()	74.848	1,497
► █ [UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410 [@] MAIN__ [{/gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410}]	26.196	524
► █ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_fortran.c.67 [@] beging_ [{/gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410}]	21.7	434
▀ █ [UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/gamess.f.538 [@] main [{/gpfs/mira-home/sameer/gamess-theta-tau/object/gamess.f.538}]	11.85	237
▀ █ [UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410 [@] MAIN__ [{/gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410}]	11.85	237
▀ █ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_fortran.c.67 [@] beging_ [{/gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410}]	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/Github/gamess-theta-tau/ddi/src/ddi_init.c.113}]	11.85	237
▀ █ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_server.c.99 [@] DDI_Init [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_server.c.99}]	11.85	237
▀ █ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_recv.c.65 [@] DDI_Server [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_recv.c.65}]	11.85	237
▀ █ [UNWIND] /lus/theta-fs0/software/perftools/tau/tau-2.26.3/src/Profile/TauMpi.c.2371 [@] DDI_Recv_request [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_recv.c.65}]	11.85	237
▀ █ [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] MPI_Recv [{/lus/theta-fs0/software/perftools/tau/tau-2.26.3/src/Profile/TauMpi.c.2371}]	11.85	237
▀ █ [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [@] PMPI_Recv [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	11.7	234
█ [SAMPLE] MPIDI_CH3I_Progress [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1} {0}]	11.3	226
█ [SAMPLE] MPIDU_Sched_are_pending [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1} {0}]	0.2	4
█ [SAMPLE] MPID_nem_gni_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0} {0}]	0.15	3
█ [SAMPLE] MPID_nem_network_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1} {0}]	0.05	1
► █ [UNWIND] ch3_progress.c.0 [@] PMPI_Recv [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	0.15	3
► █ [UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_init.c.113 [@] ddi_init_ [{/gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_init.c.113}]	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/Github/gamess-theta-tau/ddi/src/ddi_server.c.99}]	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/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0}]	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

# Deep Learning: Tensorflow

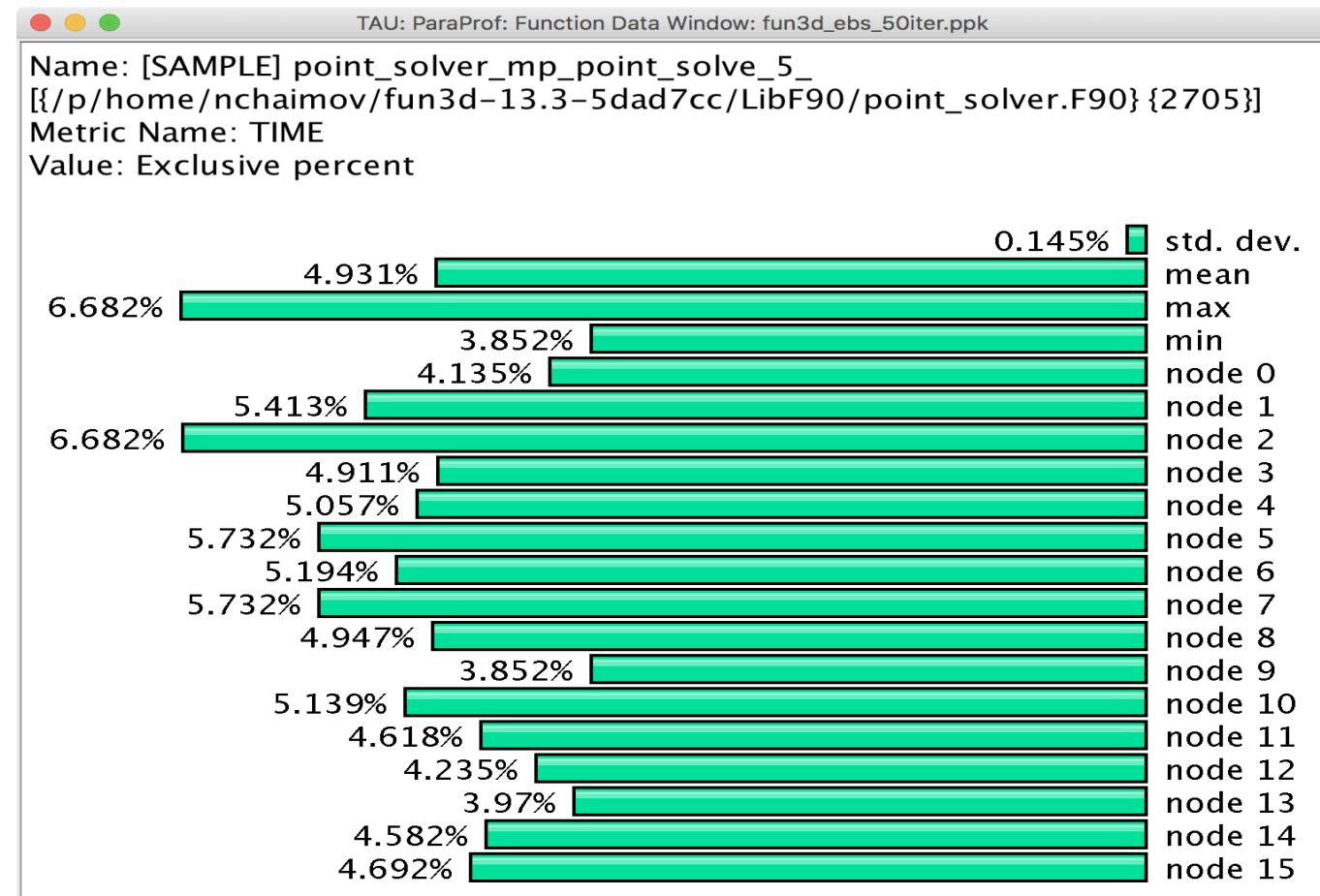


```
% tau_python -ebs nt3_baseline_keras2.py (CANDLE)
```

# Sampling Tensorflow



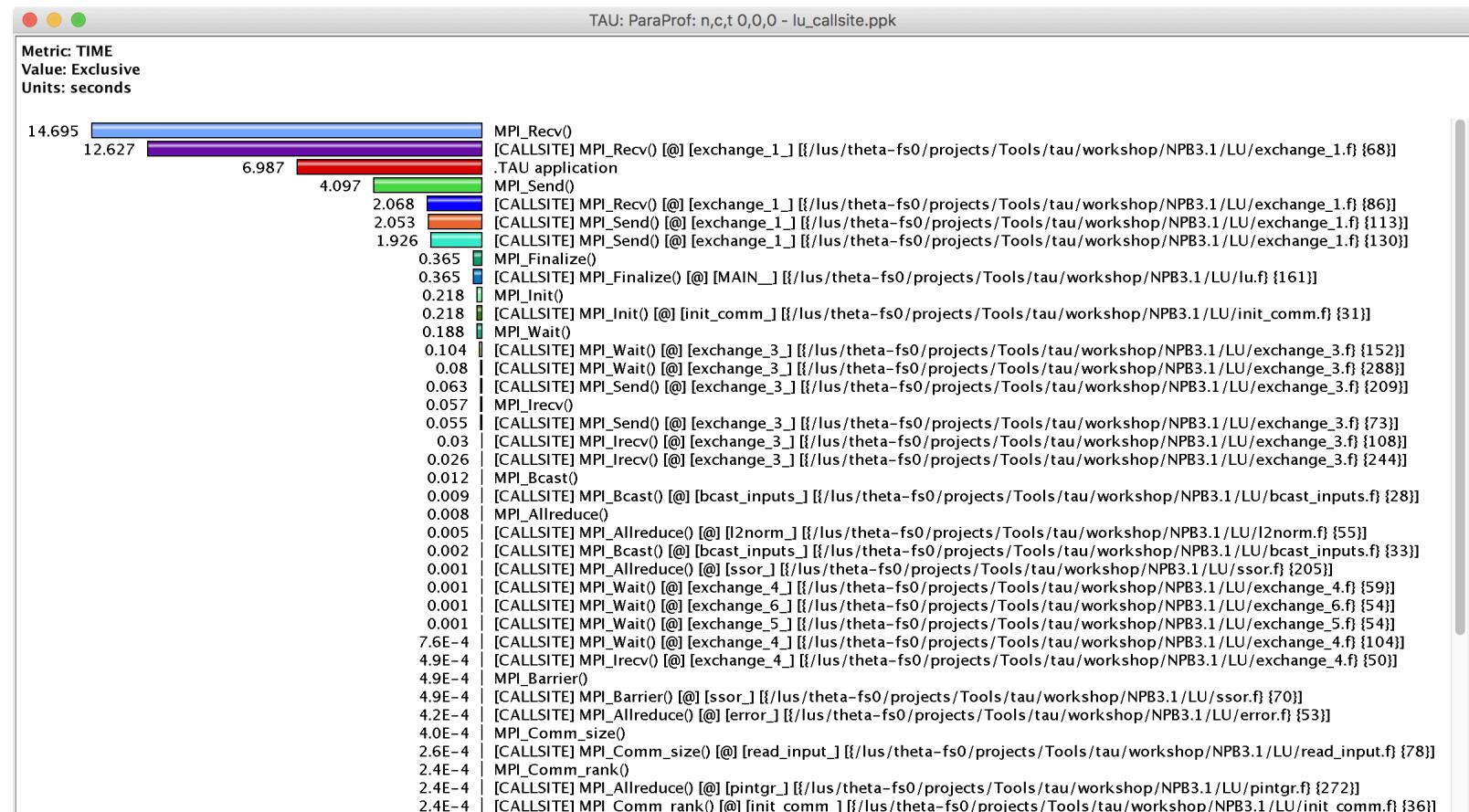
# Event Based Sampling (EBS)



Uninstrumented!

% mpirun -np 16 tau\_exec **-ebs** a.out

# Callsite Profiling and Tracing



% export TAU\_CALLSITE=1

# CALLPATH THREAD RELATIONS WINDOW

TAU: ParaProf: Call Path Data n,c,t, 2,0,0 - gamess\_unw\_call\_ebs.ppk

Metric Name: TIME  
Sorted By: Inclusive  
Units: seconds

	Exclusive	Inclusive	Calls/Tot.Calls	Name[id]
-->	0.121	79.592	1	.TAU application
	0.002	0.002	1/1	MPI_Gather()
	0.004	0.004	3/3	MPI_Allgather()
	0.122	0.122	6866/6866	MPI_Send()
	0.002	0.002	1/1	MPI_Comm_split()
	8.9E-5	8.9E-5	2/2	MPI_Comm_size()
	4.6E-4	4.6E-4	3/3	MPI_Group_incl()
	75.607	75.607	6870/6870	MPI_Recv()
	0.002	0.002	4/4	MPI_Comm_create()
	9.5E-5	9.5E-5	6/6	MPI_Comm_rank()
	5.4E-4	5.4E-4	1/1	MPI_Comm_group()
	0.003	0.003	7/7	MPI_BARRIER()
	0.112	0.112	1/1	MPI_Init_thread()
	6.3E-4	6.3E-4	1/1	MPI_Group_intersection()
	0	0.05	1/1	[CONTEXT] .TAU application
	3.601	3.601	1/1	MPI_Finalize()
	0.014	0.014	6/6	MPI_Bcast()
-->	75.607	75.607	6870/6870	.TAU application
-->	75.607	75.607	6870	MPI_Recv()
-->	0	74.848	1497/1497	[CONTEXT] MPI_Recv()
-->	0	74.848	1497	MPI_Recv()
-->	0	8.701	174/1371	[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_init.c.113 [ @] ddi_in
-->	0	26.196	524/763	[UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/unport.f.410 [ @] MAIN_ [ {/gpfs/mira-ho
-->	0.2	0.2	4/138	[SAMPLE] GNI_DlaProgress [ {/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so] [UNWIND] /lib64/libc-2.22.so.0 [ @] _start [ {/home/abuild/rpmbuild/BUILD/glibc-2.22/csua/.s
-->	0	5.75	115/1484	[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_server.c.99 [ @] DDI_
-->	0	0.2	4/5	[UNWIND] /lib64/libc-2.22.so.0 [ @] _start [ {/home/abuild/rpmbuild/BUILD/glibc-2.22/csua/.s
-->	0	11.85	237/239	[UNWIND] /gpfs/mira-home/sameer/gamess-theta-tau/object/gamess.f.538 [ @] main [ {/gpfs/mira-ho
-->	0.051	0.051	1/273	[SAMPLE] GNI_CqGetEvent [ {/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so] [UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [ @] MPID:
-->	0	0.05	1/1197	[UNWIND] /opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0) [ @] UNI
-->	0	0.15	3/7	[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_fortran.c.67 [ @] beg:
-->	0	21.7	434/1197	[UNWIND] /gpfs/mira-home/yuri/dist/Github/gamess-theta-tau/ddi/src/ddi_fortran.c.67 [ @] beg:

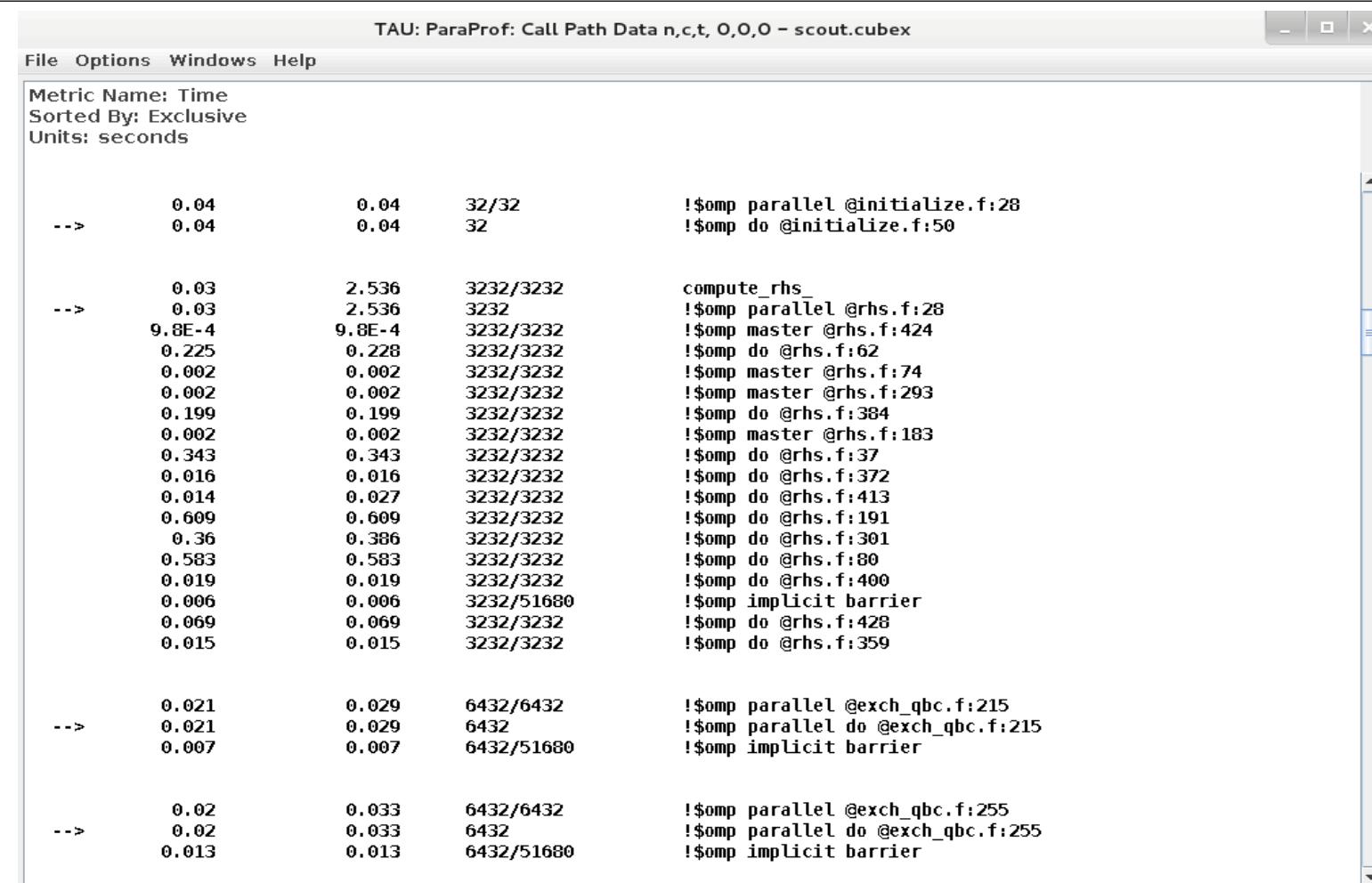
# CALLPATH THREAD RELATIONS WINDOW

TAU: ParaProf: Call Path Data n,c,t, 2,0,0 - gamess\_unw\_call\_ebs.ppk

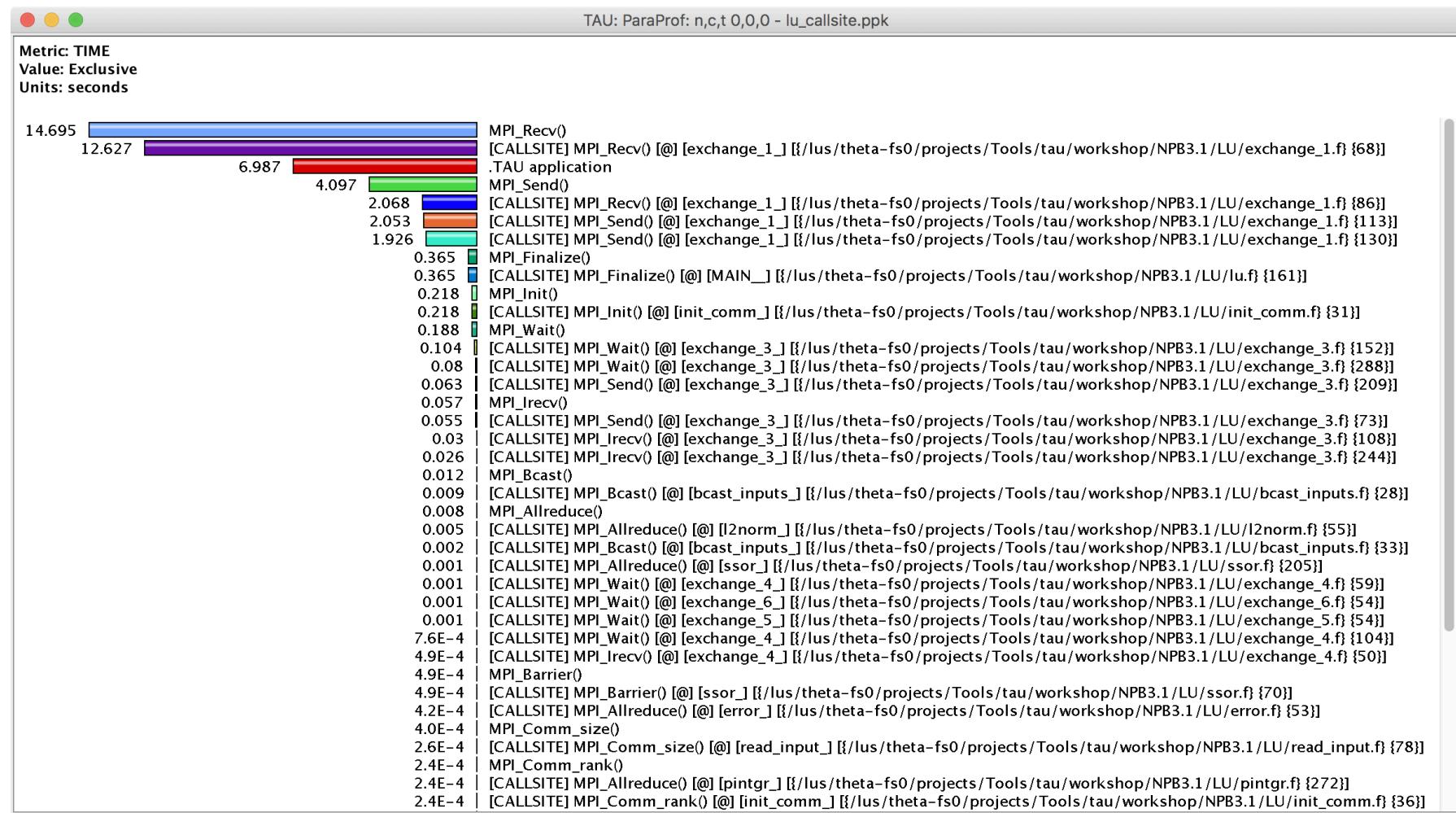
Metric Name: TIME  
Sorted By: Exclusive  
Units: seconds

	Exclusive	Inclusive	Calls/Tot.Calls	Name[id]
-->	75.607	75.607	6870/6870	.TAU application
-->	75.607	75.607	6870	MPI_Recv()
-->	0	74.848	1497/1497	[CONTEXT] MPI_Recv()
-->	0.15	0.15	3/444	[UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [0] PMPI_Recv
-->	22.046	22.046	441/444	[UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [0] MPIDI_CH3:
-->	22.196	22.196	444	[SAMPLE] MPID_nem_gni_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3
-->	5.6	5.6	112/273	[UNWIND] /opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0 [0] MPID_nem_
-->	0.051	0.051	1/273	[CONTEXT] MPI_Recv()
-->	7.651	7.651	153/273	[UNWIND] /opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0 [0] MPID_nem_
-->	0.35	0.35	7/273	[UNWIND] [/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0] [0] UNRESOLV
-->	13.652	13.652	273	[SAMPLE] GNI_CqGetEvent [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0]
-->	11.3	11.3	226/226	[UNWIND] /opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.3.0.1.0 [0] PMPI_Recv
-->	11.3	11.3	226	[SAMPLE] MPIDI_CH3I_Progress [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/libmpich_intel.so.
-->	10.349	10.349	207/207	[UNWIND] /opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0 [0] MPID_nem_
-->	10.349	10.349	207	[SAMPLE] GNI_SmsgGetNextWTag [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0]
-->	0.2	0.2	4/138	[CONTEXT] MPI_Recv()
-->	6.701	6.701	134/138	[UNWIND] /opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0 [0] GNI_CqGet
-->	6.901	6.901	138	[SAMPLE] GNII_DlaProgress [{/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64/libugni.so.0.6.0.0]
-->	5.25	5.25	105/109	[UNWIND] gni_poll.c.0 [0] MPID_nem_gni_poll [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/lib
-->	0.2	0.2	4/109	[UNWIND] gni_poll.c.0 [0] MPIDI_CH3I_Progress [{/opt/cray/pe/mpt/7.6.3/gni/mpich-intel/16.0/lib/
-->	5.45	5.45	109	[SAMPLE] MPID_nem_gni_check_localCQ [{gni_poll.c} {0}]
-->	3.601	3.601	1/1	.TAU application
-->	3.601	3.601	1	MPI_Finalize()

# ParaProf: Callpath Thread Relations Window



# Callsite Profiling and Tracing (TAU\_CALLSITE=1)



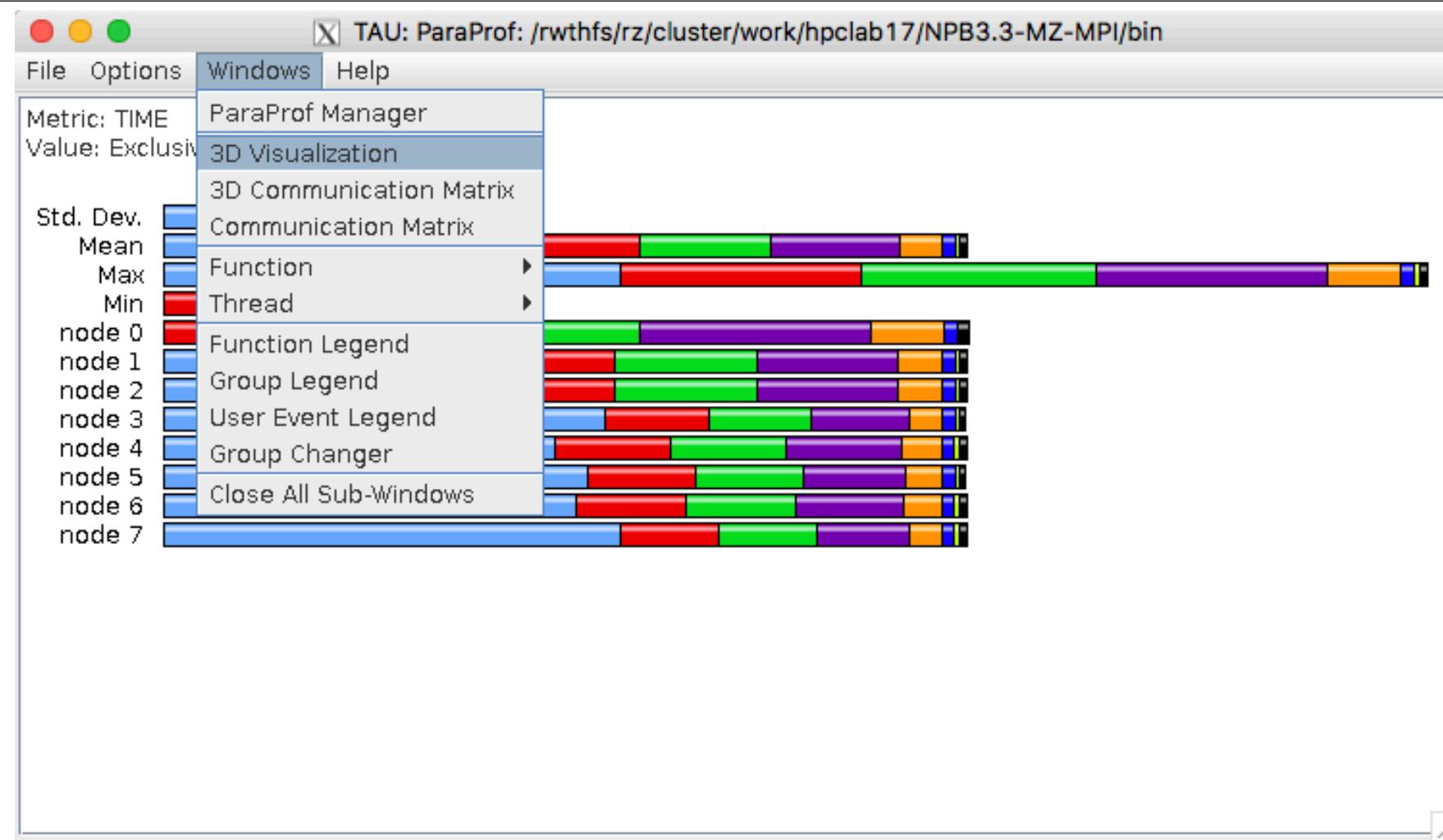
# TAU – Context Events

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						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()						

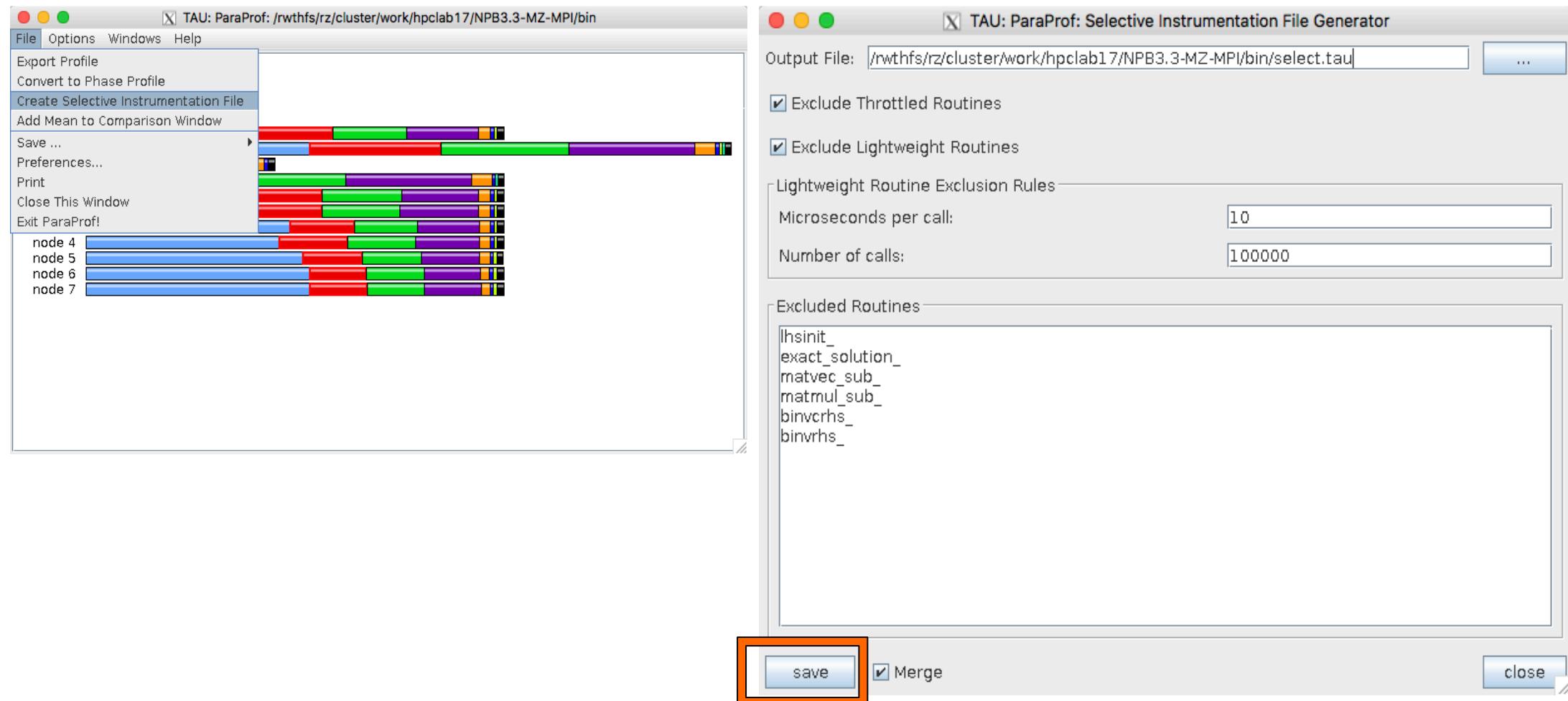
Write bandwidth per file

Bytes written to each file

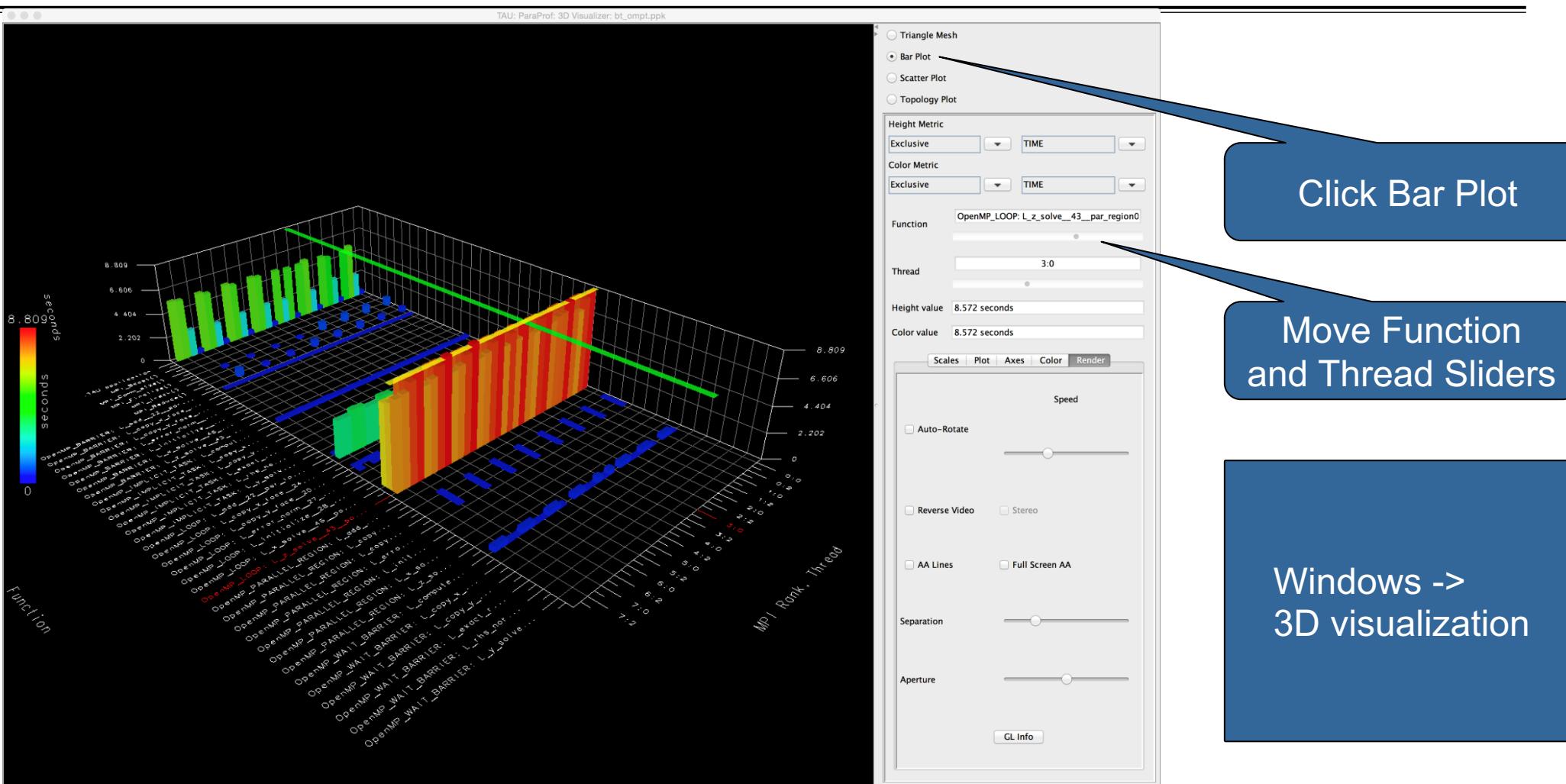
# ParaProf with Optimized Instrumentation



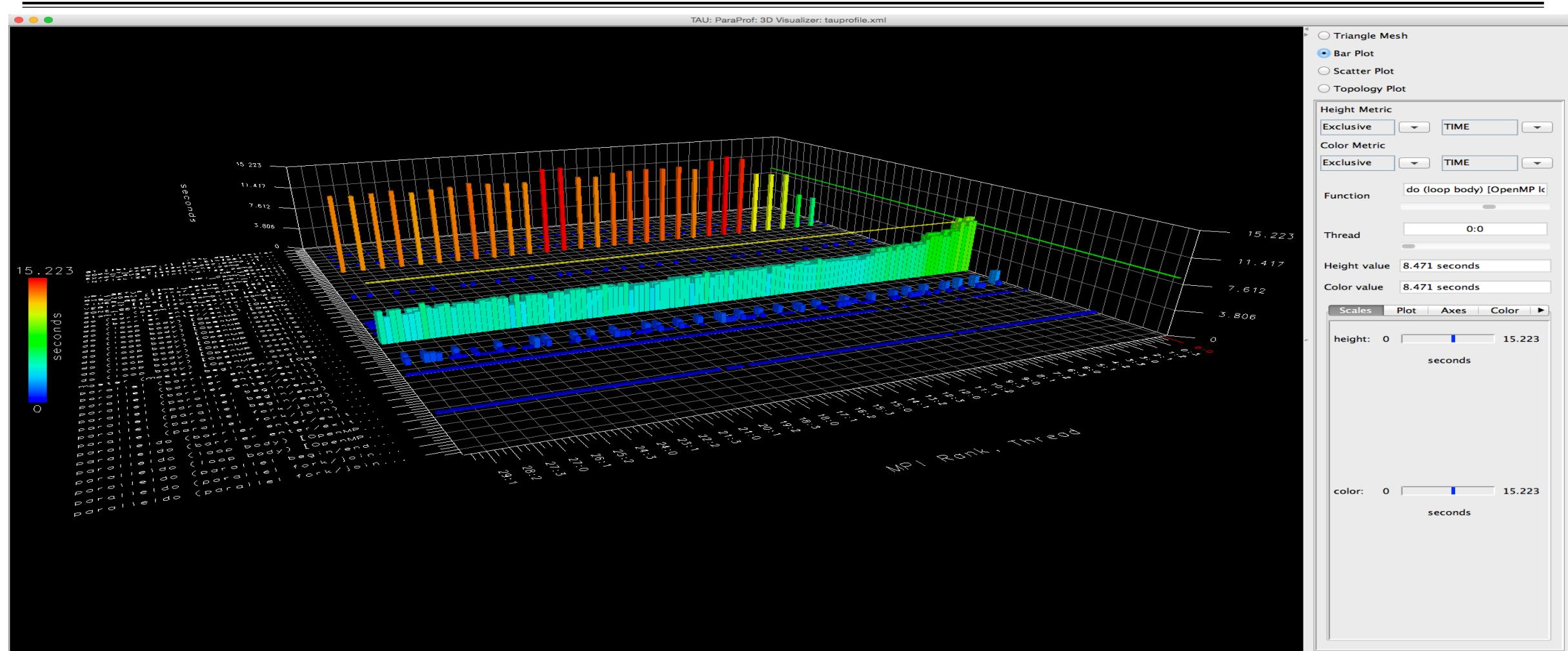
# Create a Selective Instrumentation File, Re-instrument, Re-run



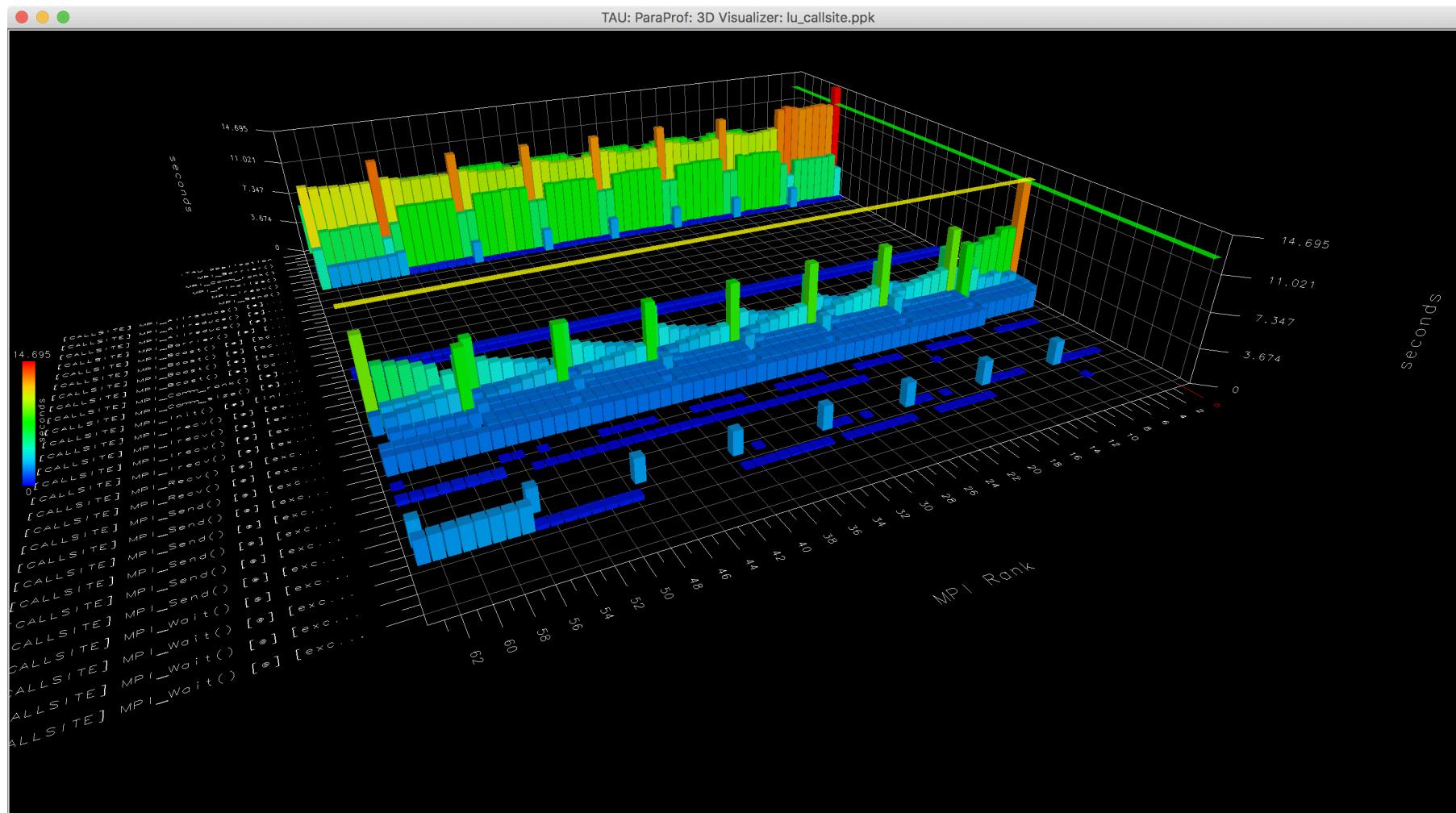
# Paraprof 3D visualization window



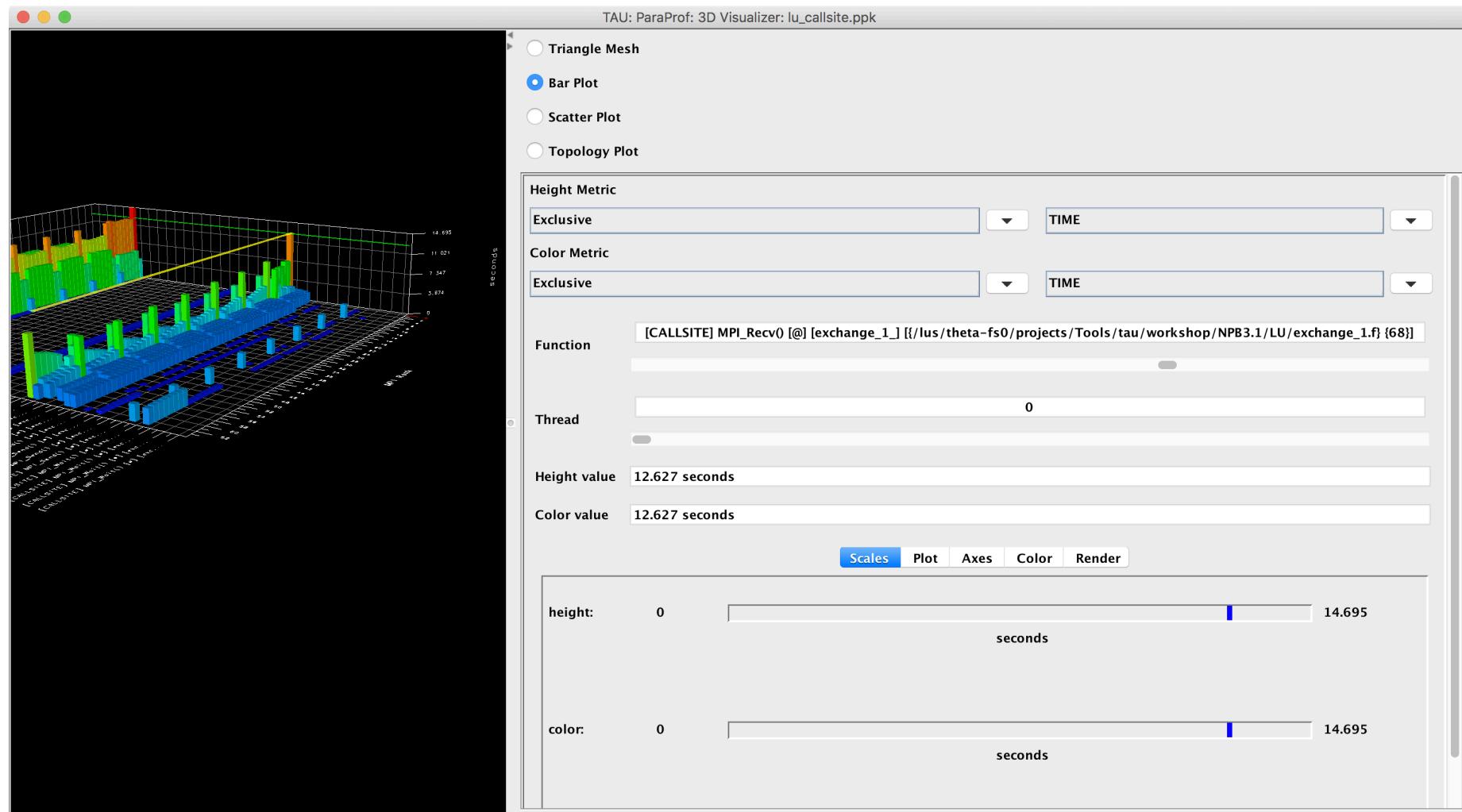
# ParaProf: 3D Visualization Window Showing Entire Profile



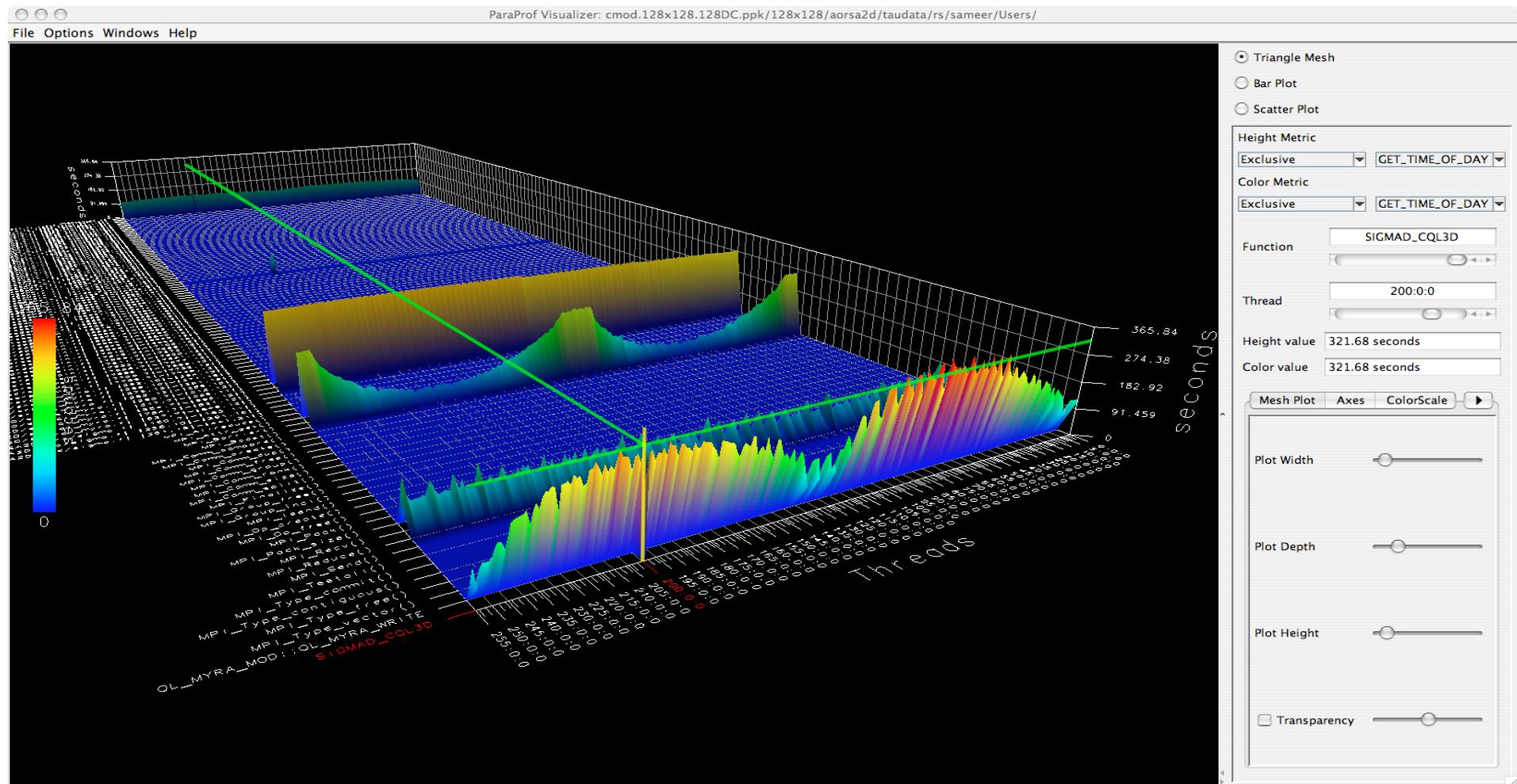
# Callsite Profiling and Tracing



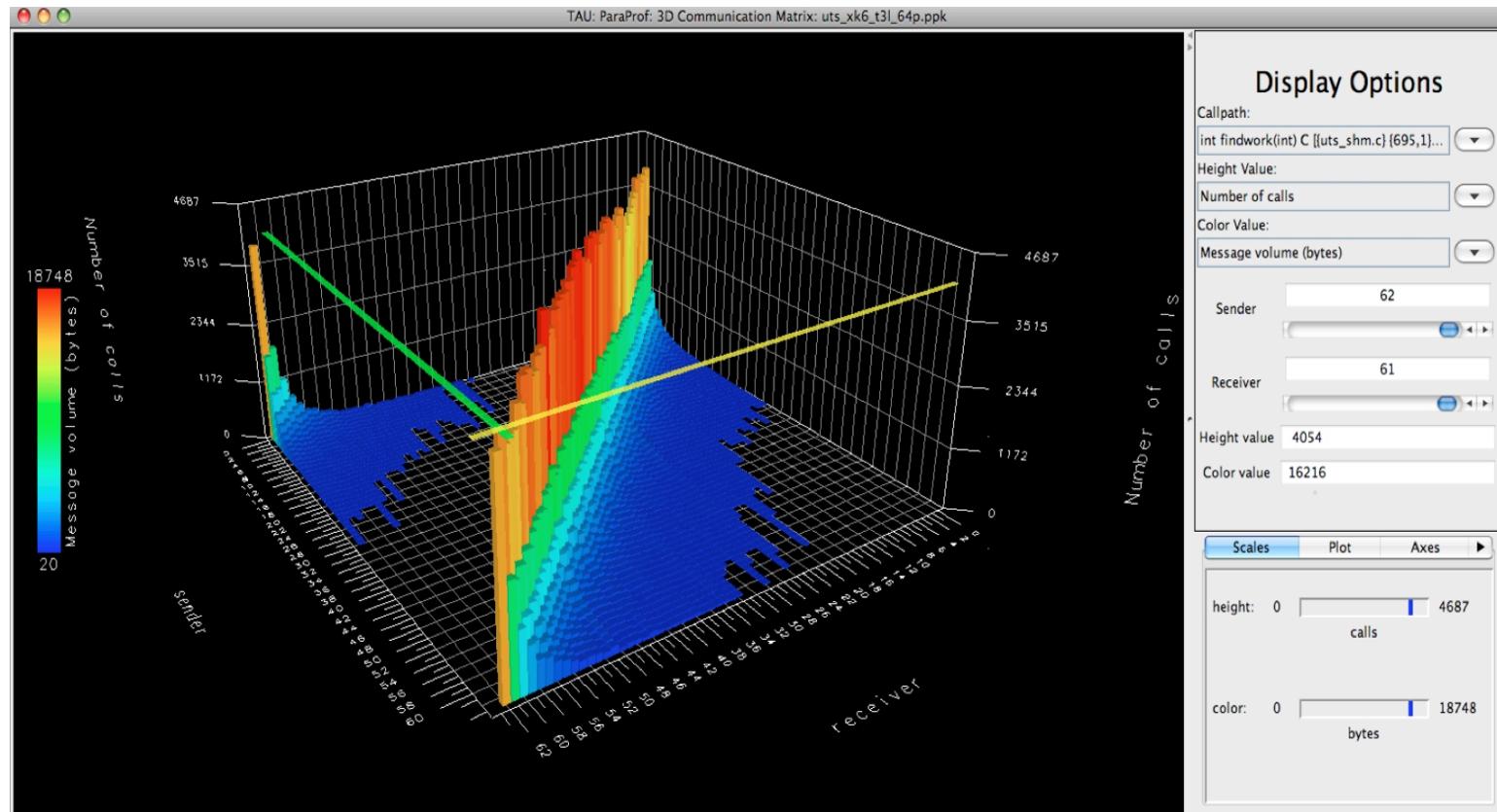
# Callsite Profiling and Tracing



# Parallel Profile Visualization: ParaProf

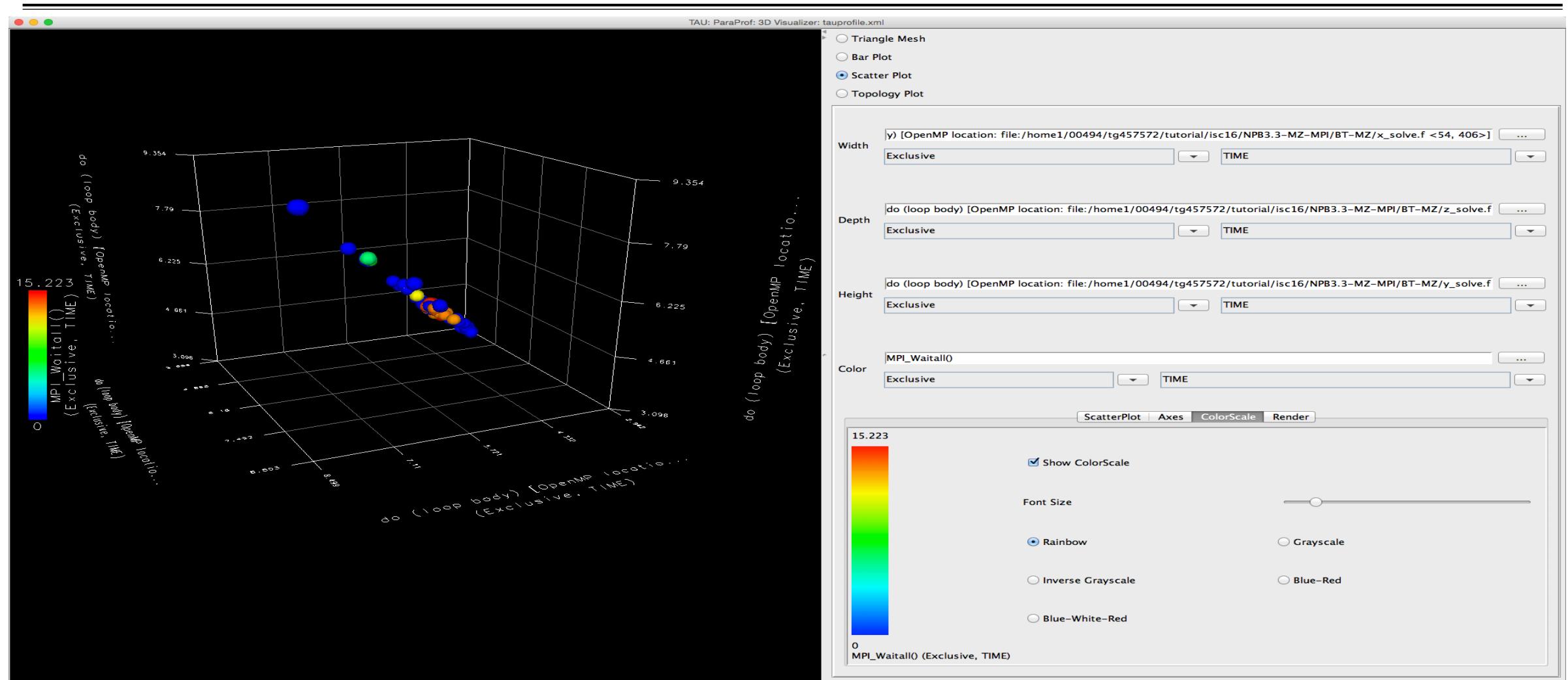


# ParaProf 3D Communication Matrix

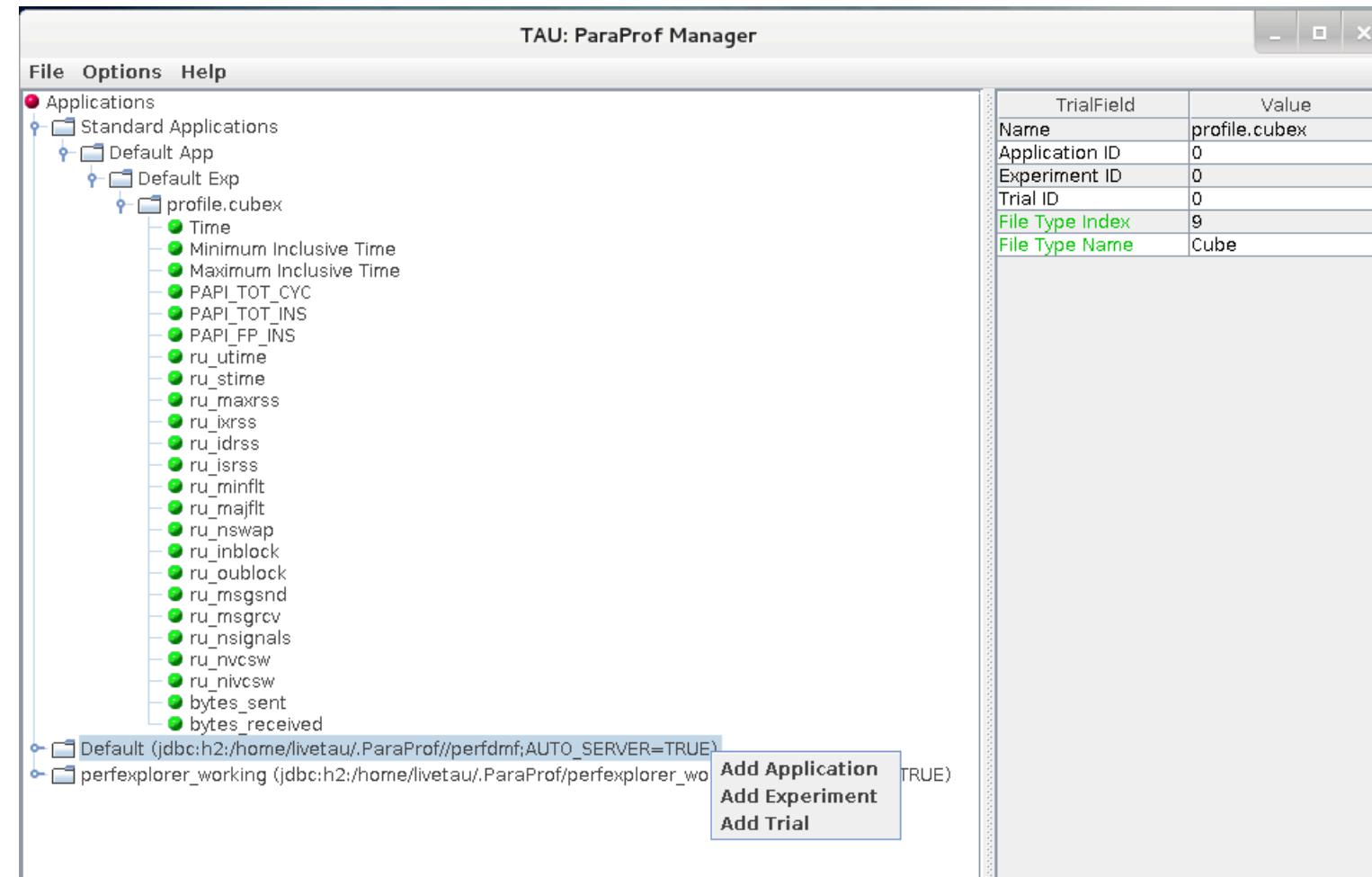


% export TAU\_COMM\_MATRIX=1

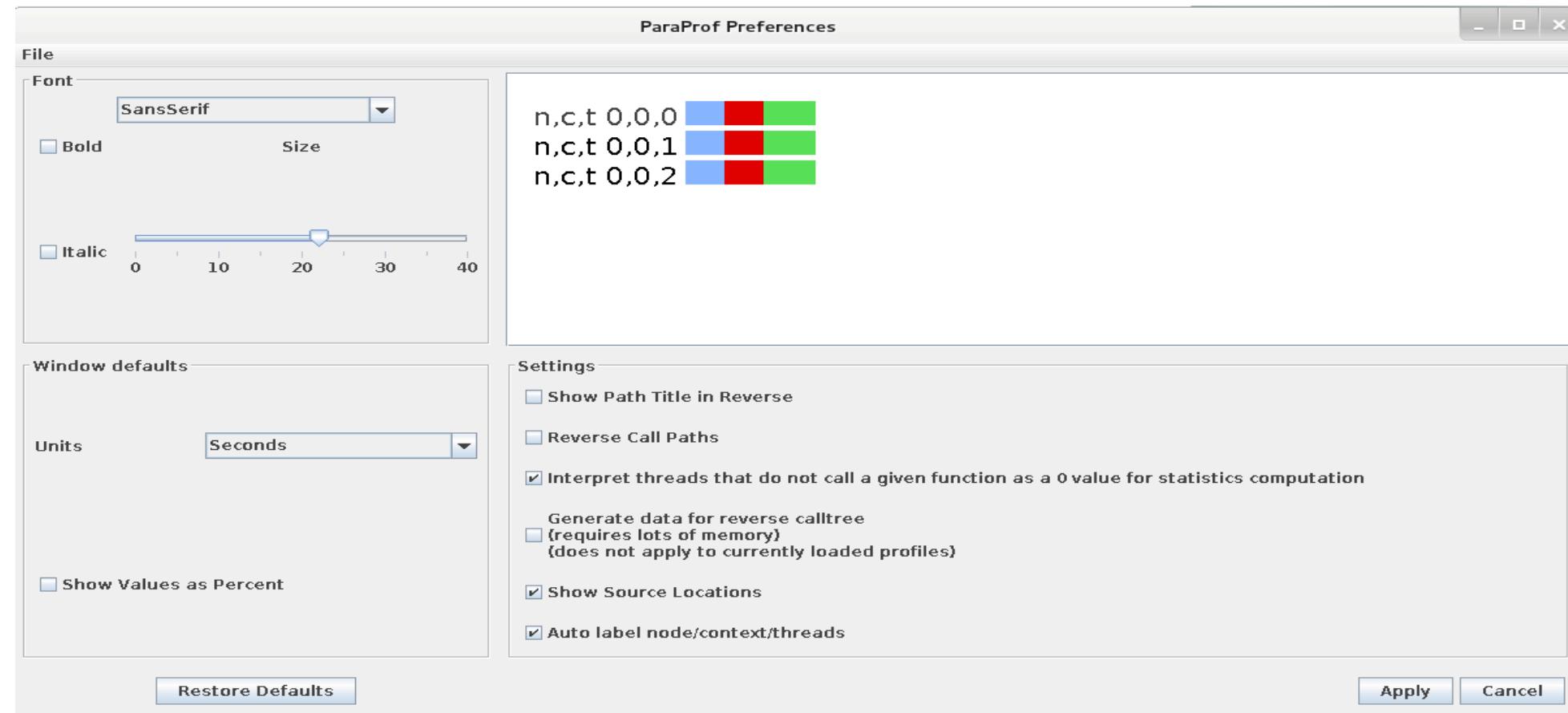
# ParaProf: 3D Scatter Plot



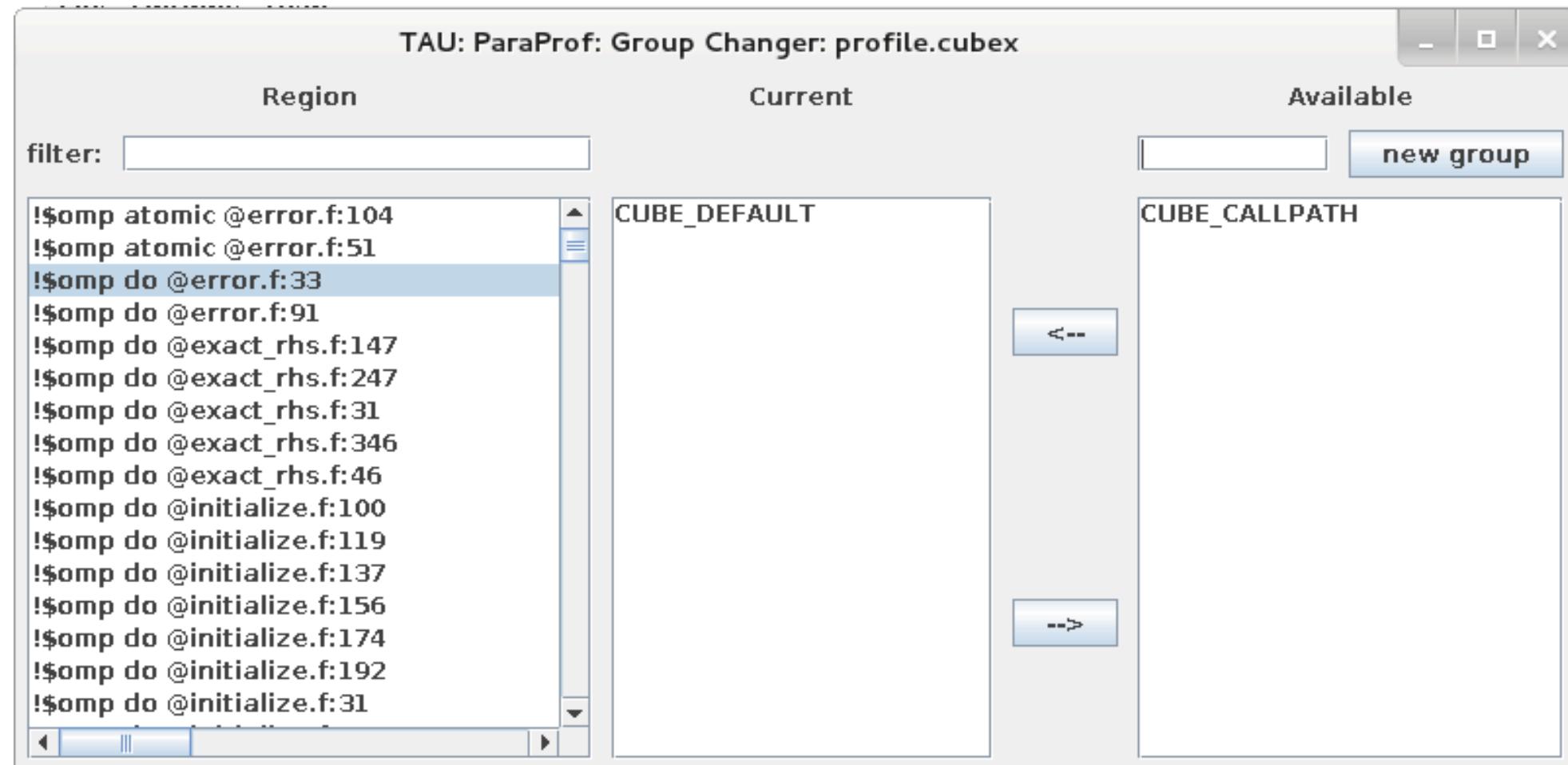
# ParaProf: Score-P Profile Files, Database



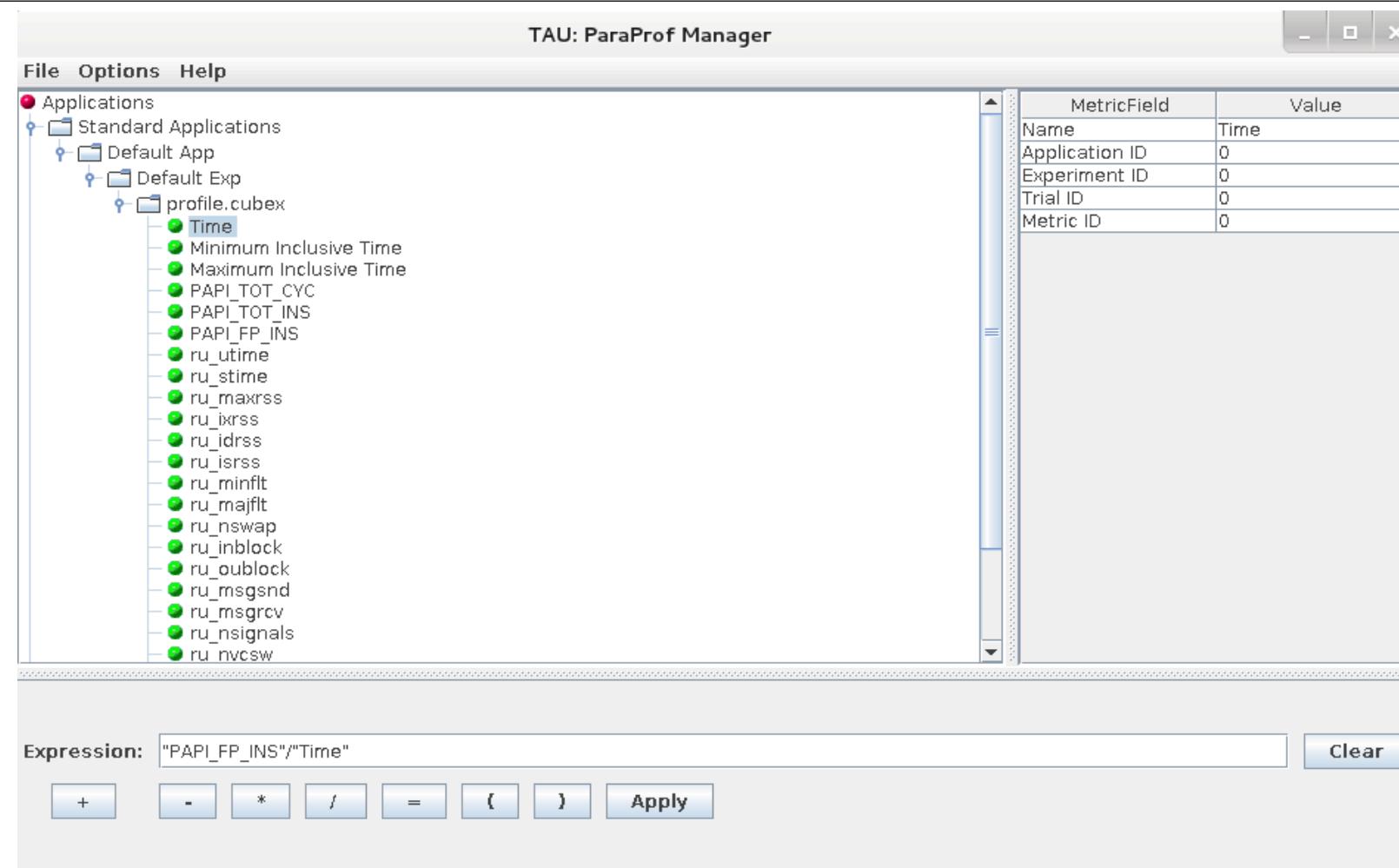
# ParaProf: File Preferences Window



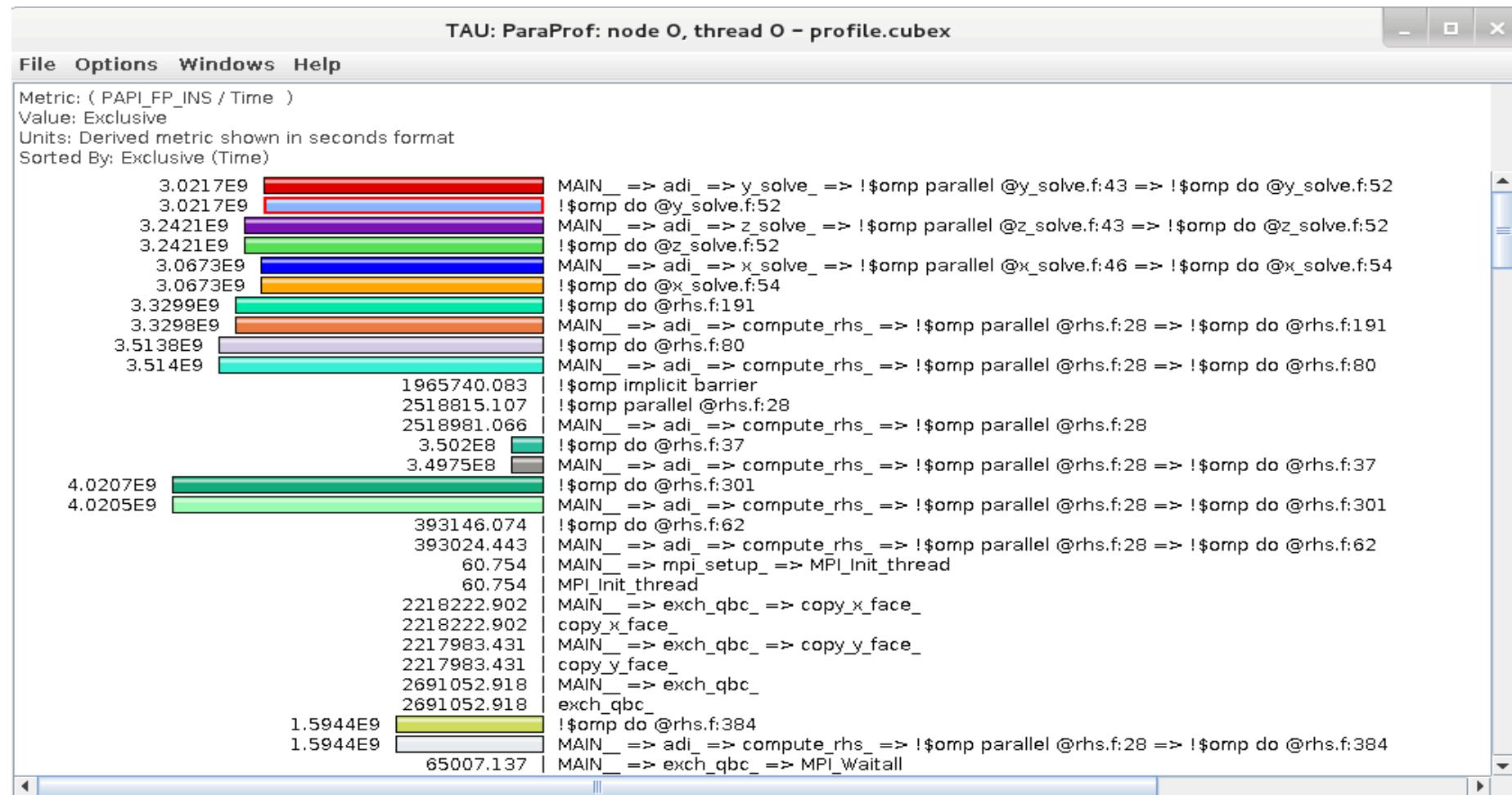
# ParaProf: Group Changer Window



# ParaProf: Derived Metric Panel in Manager Window



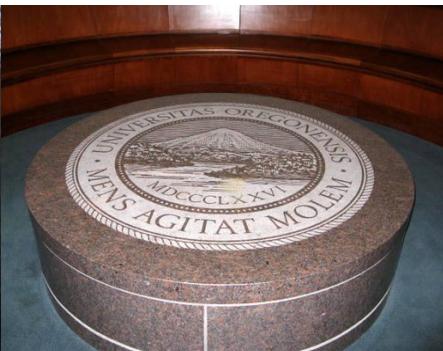
# Sorting Derived FLOPS metric by Exclusive Time



---

# TAU hands-on exercises

# Performance Research Lab, University of Oregon, Eugene, USA



# Support Acknowledgments

- US Department of Energy (DOE)
  - Office of Science contracts, ECP
  - SciDAC, LBL contracts
  - LLNL-LANL-SNL ASC/NNSA contract
  - Battelle, PNNL contract
  - ANL, ORNL contract
- Department of Defense (DoD)
  - PETTT, HPCMP
- National Science Foundation (NSF)
  - Glassbox, SI-2
  - NASA
  - CEA, France
- Partners:
  - University of Oregon
  - ParaTools, Inc., ParaTools, SAS
  - The Ohio State University
  - University of Tennessee, Knoxville
  - T.U. Dresden, GWT
  - Juelich Supercomputing Center

ParaTools



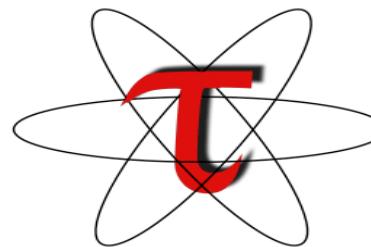


## Acknowledgement

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.

## Download TAU from U. Oregon

---



<http://tau.uoregon.edu>

<http://www.hpclinux.com> [LiveDVD, OVA]

<https://e4s.io> [Containers for Extreme-Scale Scientific Software Stack]

**Free download, open source, BSD license**