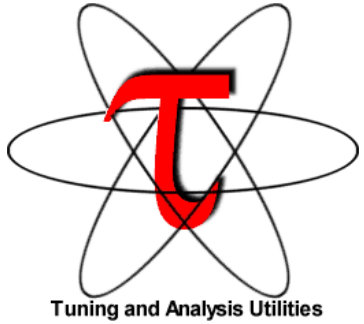
# TAU Performance System®

Tuesday, 11:45am – 12:15pm
MUG'24, OSC, OSU, Columbus, OH

Sameer Shende
Research Professor and Director,
Performance Research Laboratory, OACISS, University of Oregon
President and Director, ParaTools, Inc.

http://tau.uoregon.edu/TAU_MUG24.pdf

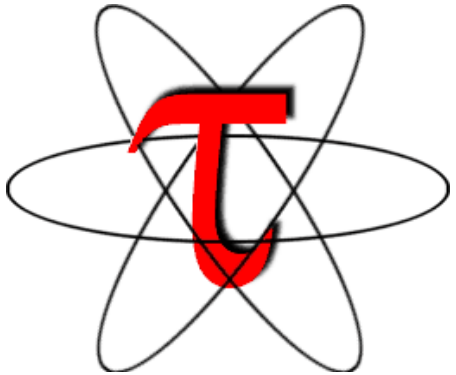



UNIVERSITY
OF OREGON


# Acknowledgments


- The MVAPICH2 team The Ohio State University
  - http://mvapich.cse.ohio-state.edu
- TAU team at the University of Oregon
  - http://tau.uoregon.edu





# Motivation and Challenges

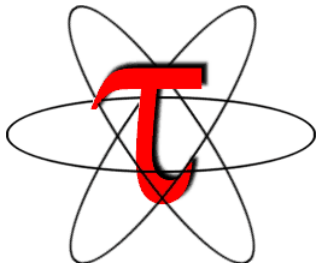
- With growing hardware complexity, it is getting harder to accurately measure and optimize the performance of our HPC and AI/ML workloads.
- TAU Performance System®:
  - Deliver a scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads.
  - http://tau.uoregon.edu

- It is getting harder to install our HPC and AI/ML tools.

# Motivation: Improving Productivity

- TAU Performance System®:
  - Deliver a scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads
  - http://tau.uoregon.edu

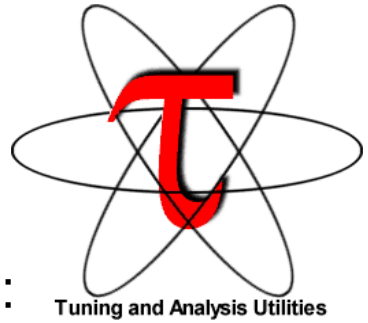
# TAU Performance System®



- Tuning and Analysis Utilities (25+ year project)
- Comprehensive performance profiling and tracing
  - Integrated, scalable, flexible, portable
  - Targets all parallel programming/execution paradigms

- Integrated performance toolkit
  - Instrumentation, measurement, analysis, visualization
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
  - Open source (BSD-style license)
  - Uses performance and control variables to interface with MVAPICH2
- Integrates with application frameworks
- http://tau.uoregon.edu

# 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)
- Scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads that supports:
  - C++/C/DPC++, Fortran, Python
- 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
- Supports Grace-Grace and Grace-Hopper (SVE aarch64) systems
- TAU's paraprof, pprof, perfexplorer for profile analysis; Vampir, Jumpshot, Perfetto.dev for traces
- http://tau.uoregon.edu

# 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 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?

# 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`
- `ROCm: % mpirun -np 16 tau_exec -T rocm,mpi -rocm -ebs ./a.out`
- `Python: % tau_python ./foo.py`

```
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
```

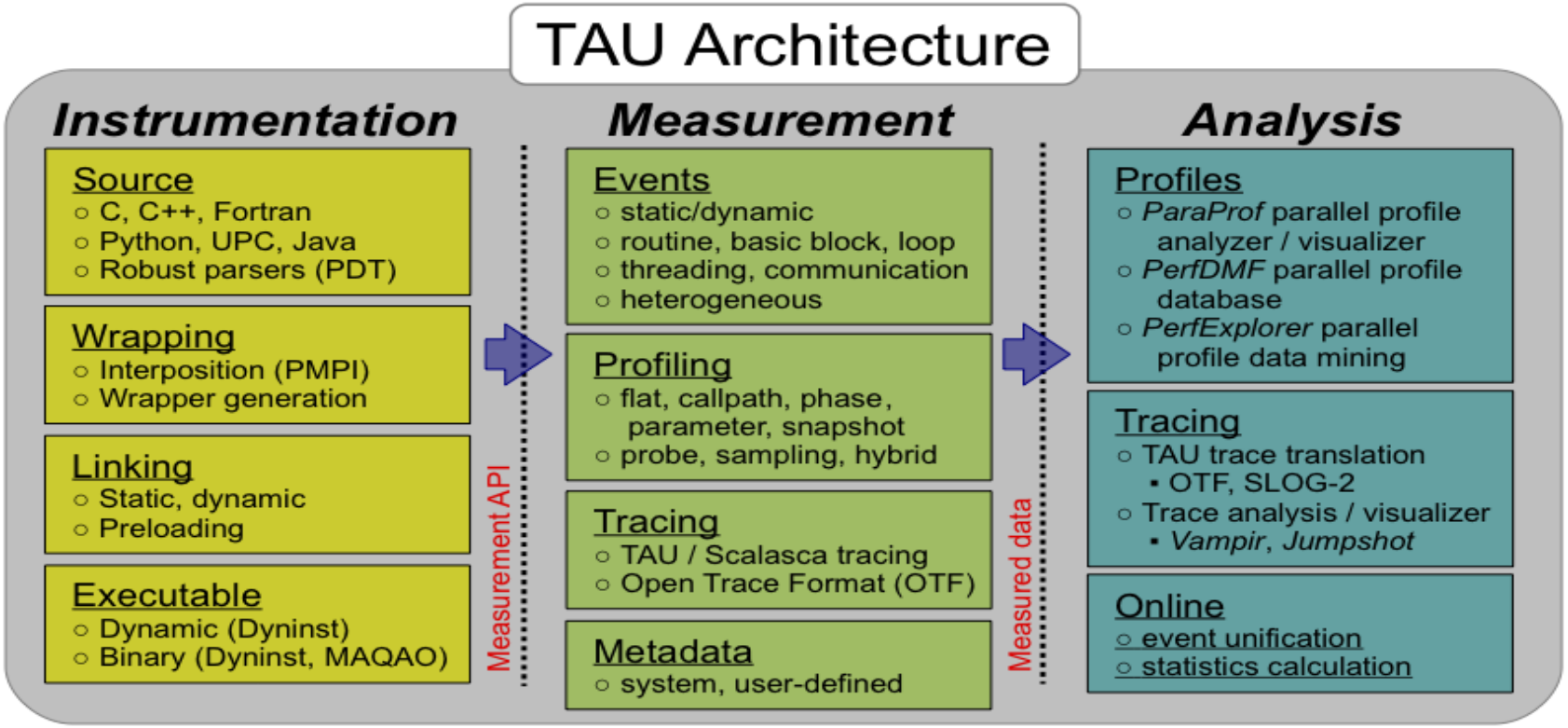
- `Jumpshot: tau2slog2 tau.trc tau.edf -o app.slog2; jumpshot app.slog2`

# TAU Performance System®

## Parallel performance framework and toolkit

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

Tuning and Analysis Utilities



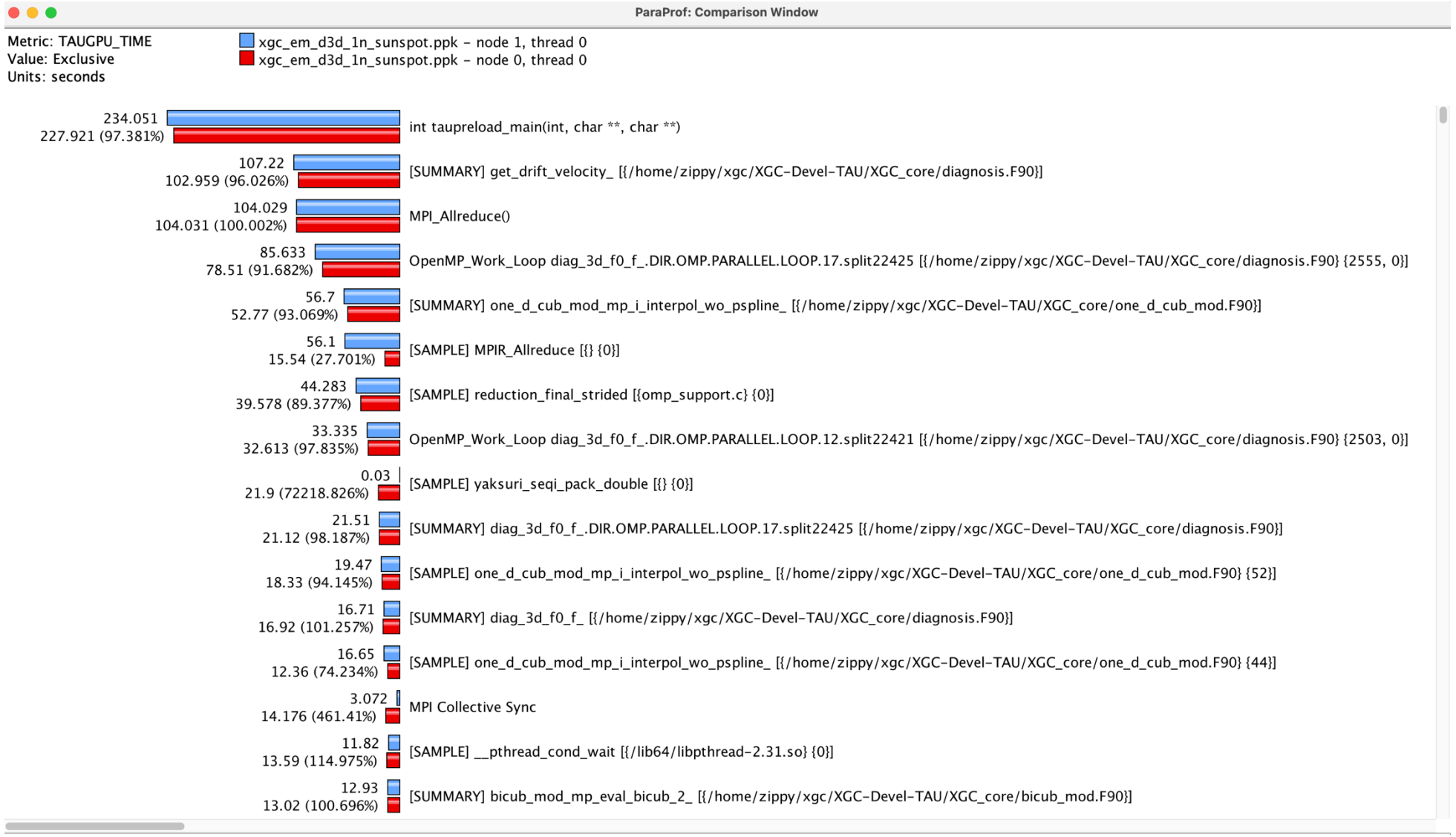
# TAU Performance System®

Instrumentation
- Fortran, C++, C, UPC, Java, Python, Chapel, Spark
- Automatic instrumentation

Measurement and analysis support
- MPI (MVAPICH), OpenSHMEM, ARMCI, PGAS, DMAPP
- Supports Intel oneAPI compilers
- pthreads, OpenMP, OMPT interface, hybrid, other thread models
- GPU: OpenCL, oneAPI DPC++/SYCL (Level Zero), OpenACC, Kokkos, RAJA
- Parallel profiling and tracing

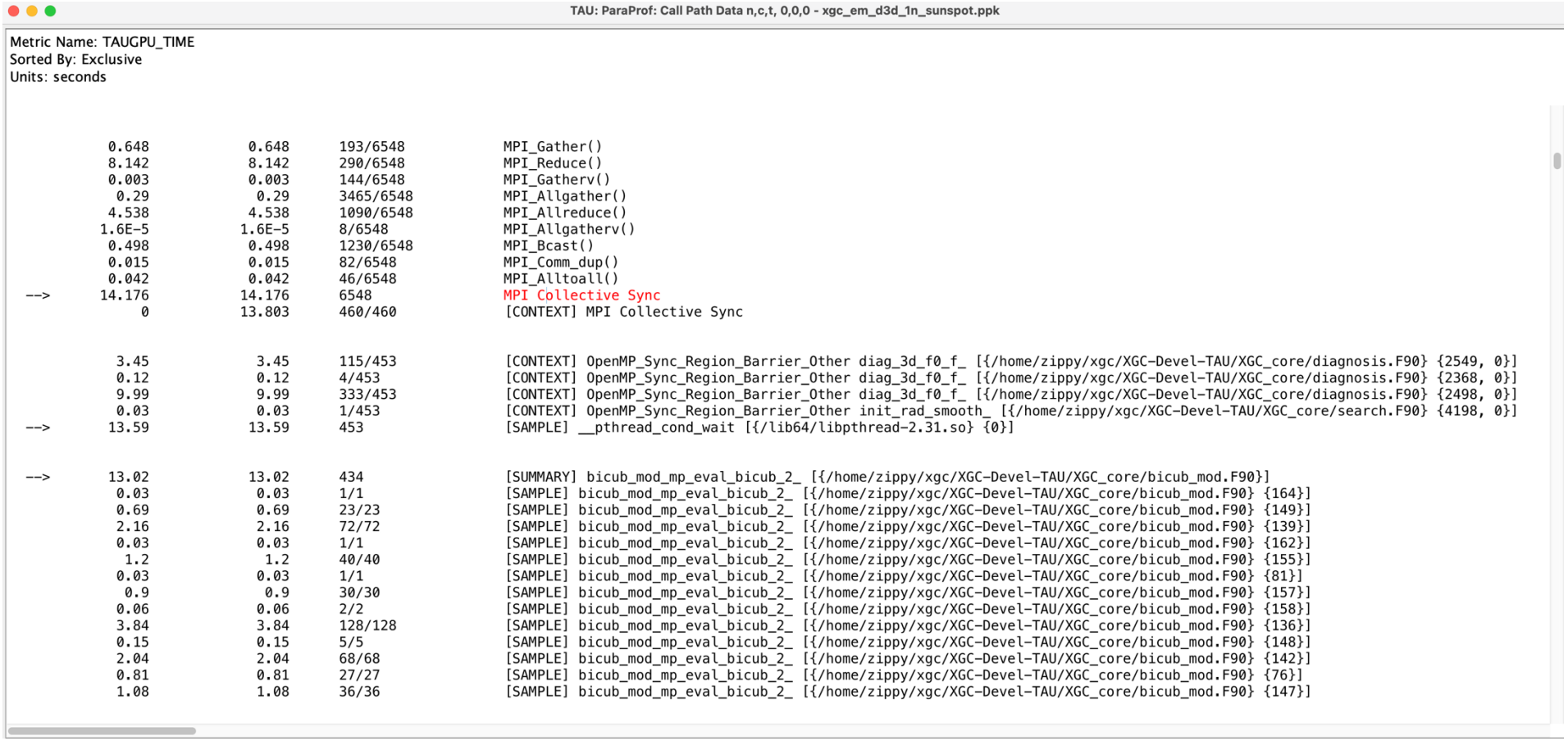
Analysis
- Parallel profile analysis (ParaProf), data mining (PerfExplorer)
- Performance database technology (TAUdb)
- 3D profile browser

# TAU and XGC on Sunspot (OMPT+MPI+Level Zero+Kokkos+EBS)




Comparing MPI rank 1 (thread 0) with MPI rank 0 (thread 0) with TAU's comparison window

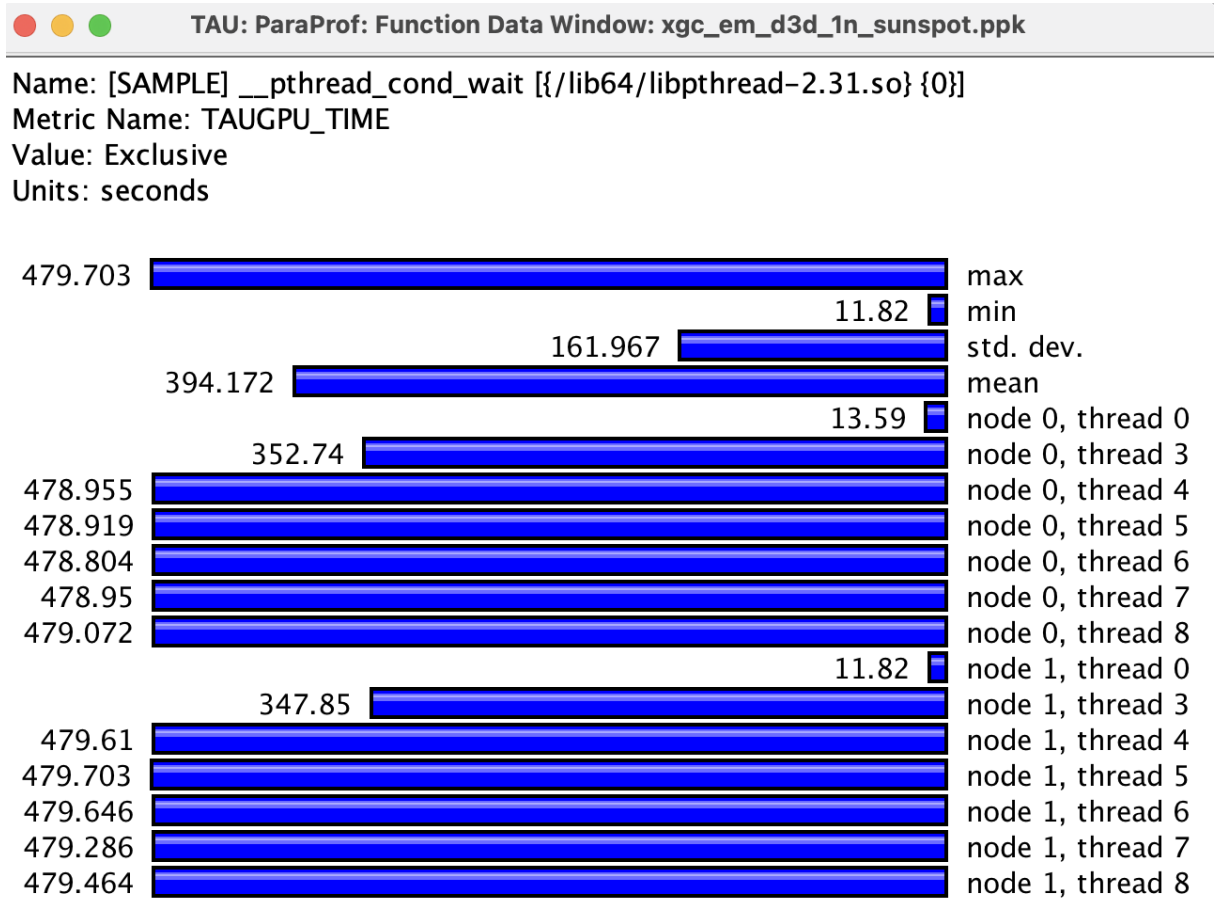
# TAU’s Callpath Thread Relations Window in ParaProf




MPI Collective Sync is the time wasted in an implicit barrier in a collective operation. In the TAU callpath Thread relations window – where we see immediate parents and children for a timer – we see that out of 14.176 seconds wasted in this sync operation, 8.142 seconds came from MPI_Reduce and 4.538 seconds came from MPI_Allreduce operations where implicit barriers were called on MPI rank 0 thread 0.

We also see that the pthread_condition_wait (13.59 seconds) was called from diag_3d_f0_f in diagnosis.F90 line 2368 which accounted for 9.99 s (out of 13.59 seconds).

# TAU and XGC on Sunspot (OMPT+MPI+Level Zero+Kokkos+EBS)




The time wasted in __pthread_cond_wait across different threads in XGC EM.
The contribution from MPI rank 0 thread 0 is small compared to other threads.

# 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 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.
  - 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.

# TAU's Support for Runtime Systems

- *MPI*
  - PMPI profiling interface
  - MPI_T tools interface using performance and control variables
  - MPI Collective Sync time: time in an implicit barrier in MPI collective operations
- *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
- *Intel Level Zero*
  - Captures time spent in kernels on GPUs using oneAPI Level Zero
  - Captures time spent in Intel Level Zero runtime calls
- *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
- *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*

Python + CUPTI + pthread profiling interfaces (e.g., Tensorflow, PyTorch)

*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.

To use with DyninstAPI 13 on x86_64:

1. Load spack: source 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

With AMD GPUs:

./configure -bfd=download -mpi -rocm=/opt/rocm-6.0.0 -rocprofiler=/opt/rocm-6.0.0 -dyninst=download; 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
```

# AWP-ODC [UCSD]: TAU +ROCm +DyninstAPI

```
NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0          42       23,298           1           1   23298737 .TAU application
 99.8           1       23,256           1           1   23256593 int taupreload_main(int, char **, char **)
 99.8      17,220       23,255           1      141471   23255517 main
 21.6       4,909        5,038        2001       12006       2518 topo_velocity_interior_H
  1.4         333          333          28           0      11920 Alloc3D
  0.9         208          208           1           0     208803 SetDeviceFilterParameters
  0.4           6           99        2002       14020         50 receivers_write
  0.3          81           81        6003           0         14 __device_stub__dtopo_vel_111_unroll<128, 1, 1, 1, 1>
  0.3           9           60        6006       30036         10 receiver_write
  0.2           3           56        2001       12006         28 sources_add_curvilinear
  0.2           7           53       12006       18000          4 source_add_curvilinear
  0.2          47           47        6003           0          8 __device_stub__dtopo_vel_112
  0.2          45           45        6006           0          8 cuinterp_interp_H
  0.2          43           43        6000           0          7 cusource_add_curvilinear_H
  0.2          43           43          18           0       2395 MPI_File_open()
  0.2          42           42           5           0       8457 MPI_Barrier()
  0.2          37           37           1           0      37563 init_texture
  0.2           5           36       12012       12036          3 source_read
  0.2           2           36        2001        4002         18 topo_stress_interior_H
  0.1           3           31        2002       10020         16 sources_read
  0.1       0.049           31           6          24       5183 mpi_io_idx_write
  0.1       0.097           28          12          48       2395 mpi_io_idx_read
  0.1           6           24           2           4      12450 _input_read
  0.1          17           18           2          18       9144 _read_header
  0.1          18           18          23           0        793 MPI_Send()
  0.1          17           17        2001           0          9 __device_stub__dtopo_str_111_index_unroll<128, 1, 2, 1, 1>
  0.1       0.028           17           1           9      17120 sources_init
```

# AWP-ODC [UCSD]: TAU +ROCm +DyninstAPI

```
NODE 0;CONTEXT 0;THREAD 2:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0       29,706       30,532           1       14006   30532377 .TAU application
  0.9          267          267        2001           0        134 void dtopo_str_111_index_unroll<128, 1, 2, 1, 1>(float*, float*, float*, float*, float*,
float*, float*, float*, float*, float*, float*, float*, float*, float*, float*, float const*, float const*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, int const*, float const*, int, int, int, int,
int, int, int, int, int, int) [clone .kd]
  0.6          197          197        2001           0         98 dtopo_buf_vel_111(float*, float*, float*, float const*, float const*, float const*,
float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*,
float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*,
float const*, float const*, float, float, int, int, int, int, int, int) [clone .kd]
  0.4          128          128        2000           0         64 dtopo_str_112(float*, float*, float*, float*, float*, float*, float*, float*, float*,
float*, float*, float*, float*, float*, float*, float*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, int const*, float const*, int, int, int, int, int, int, int, int, int, int, int)
[clone .kd]
  0.3           96           96        2001           0         48 dtopo_buf_vel_112(float*, float*, float*, float const*, float const*, float const*,
float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*,
float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*,
float const*, float const*, float, float, int, int, int, int, int, int) [clone .kd]
  0.3           81           81        2001           0         41 void dtopo_vel_111_unroll<128, 1, 1, 1, 1>(float*, float*, float*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float
const*, float, float, int, int, int, int, int, int, int) [clone .kd]
  0.1           28           28        2001           0         14 dtopo_vel_112(float*, float*, float*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float
const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float const*, float, float, int,
int, int, int, int, int, int) [clone .kd]
  0.1           26           26        2001           0         13 update_boundary_y(float*, float*, float*, float*, float*, float*, int, int, int)
[clone .kd]
```

# 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

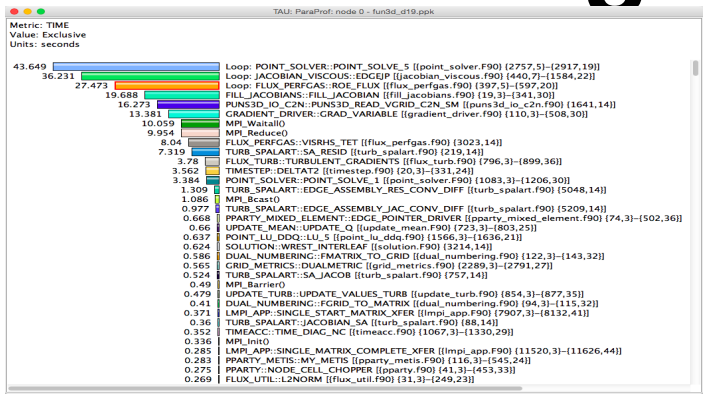
```
% export TAU_TRACK_MEMORY_LEAKS=1
% mpirun –np 256 tau_exec –memory_debug ./a.out (bounds check)
```

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>
```

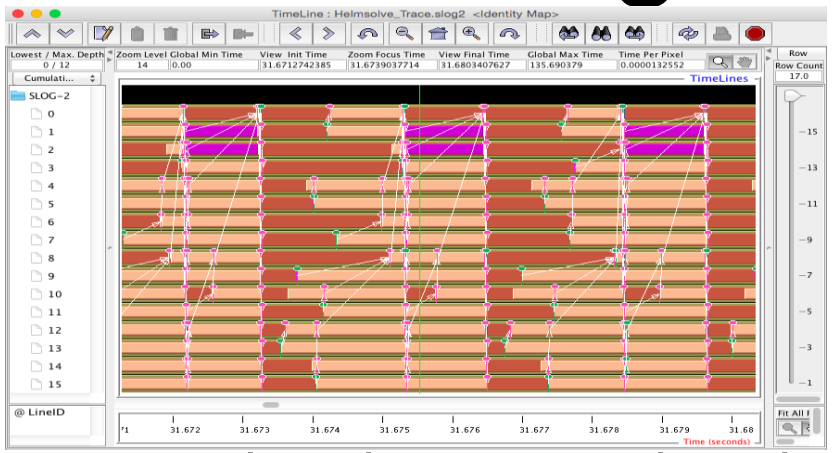
# Profiling and Tracing

## 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

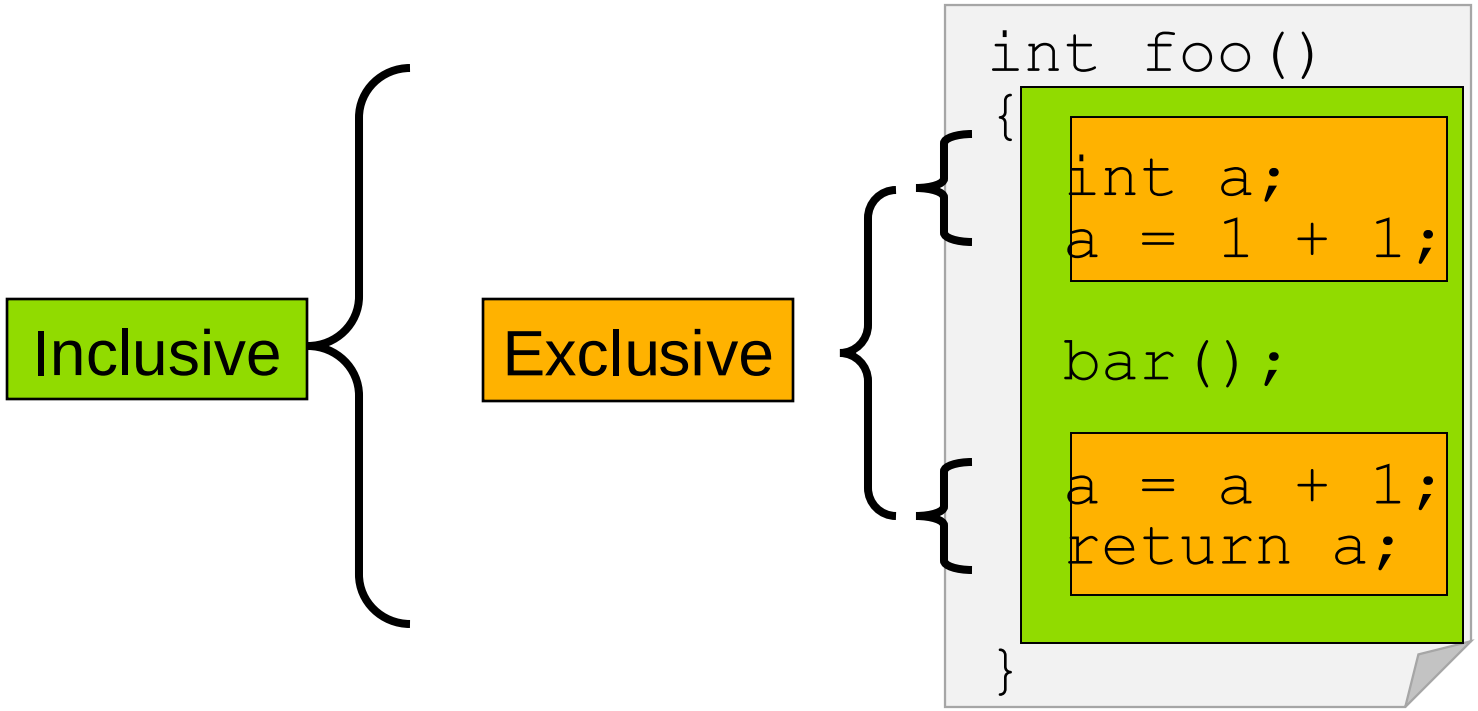
- Profiling and tracing

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

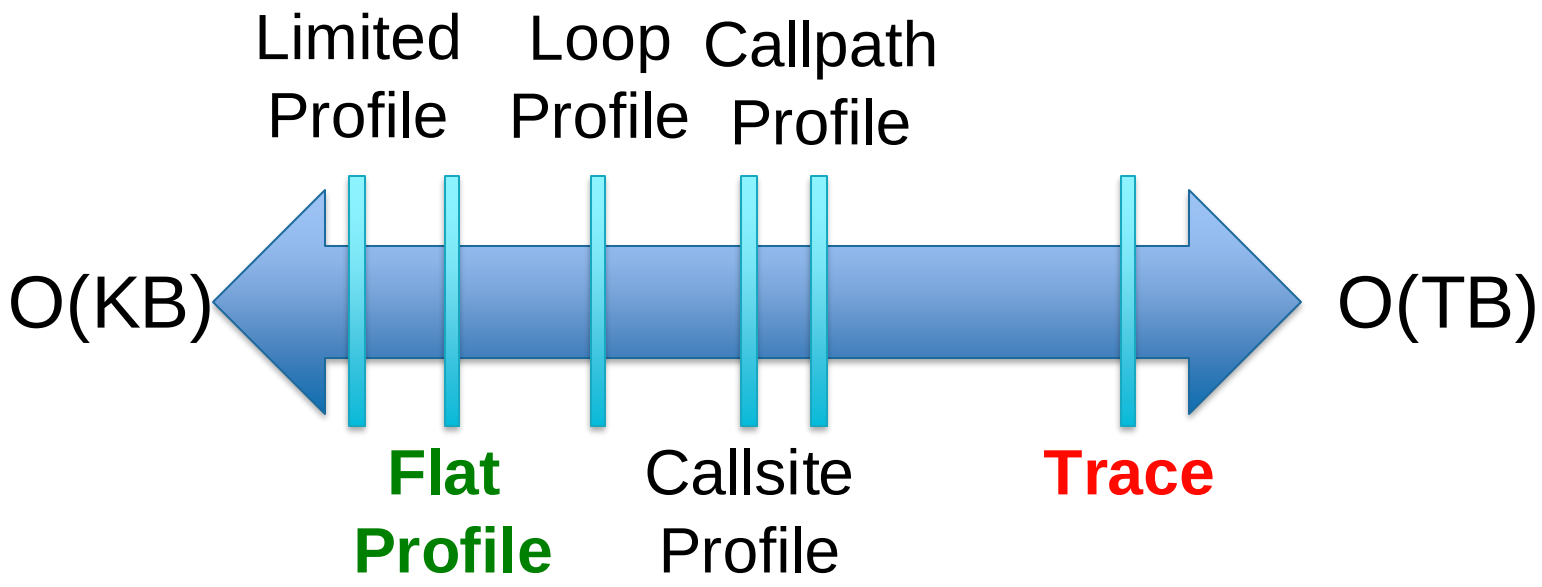
**Tracing** shows you **when** the events take place on a timeline

# 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?



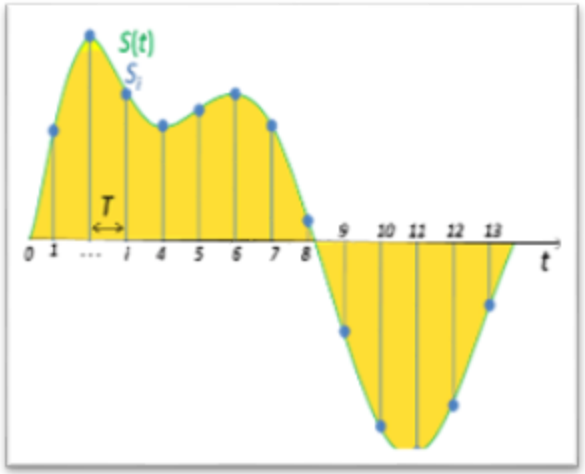
# Types of Performance Profiles

- *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

# Performance Data Measurement

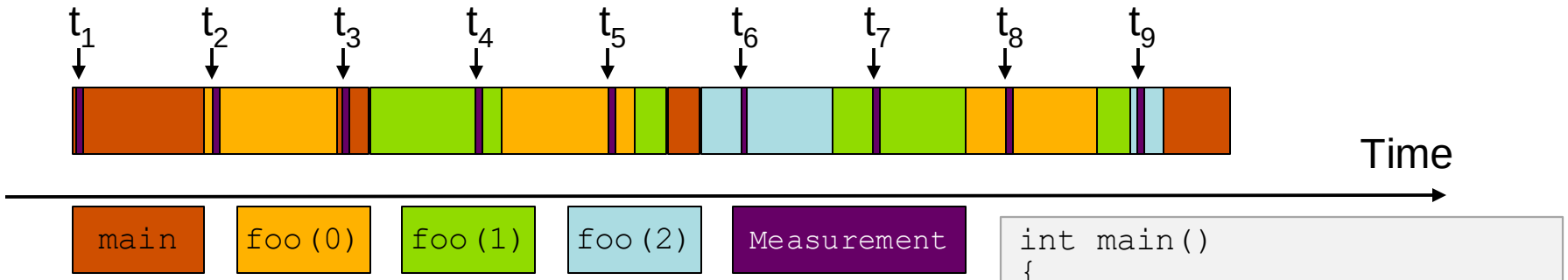
```
Call START('potential')
// code
Call STOP('potential')
```

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




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

# Sampling



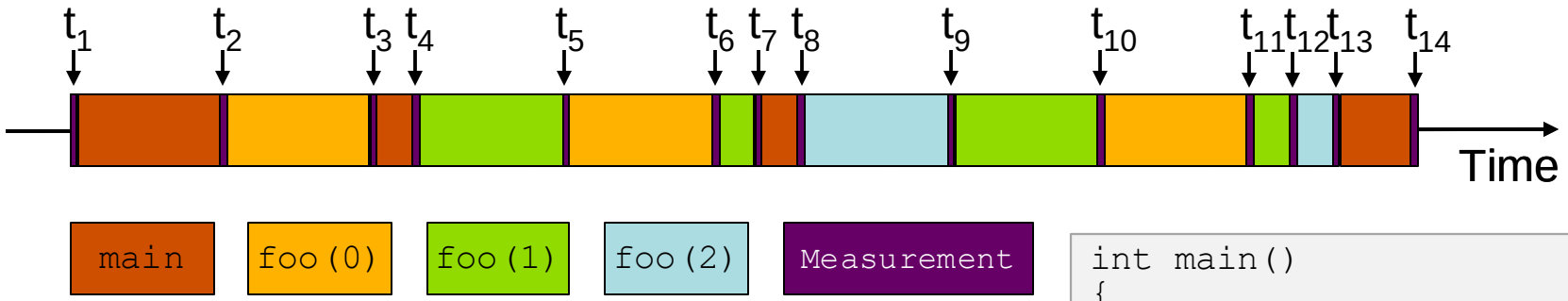

Running program is periodically interrupted to take measurement
- Timer interrupt, OS signal, or HWC overflow
- Service routine examines return-address stack
- Addresses are mapped to routines using symbol table information

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:

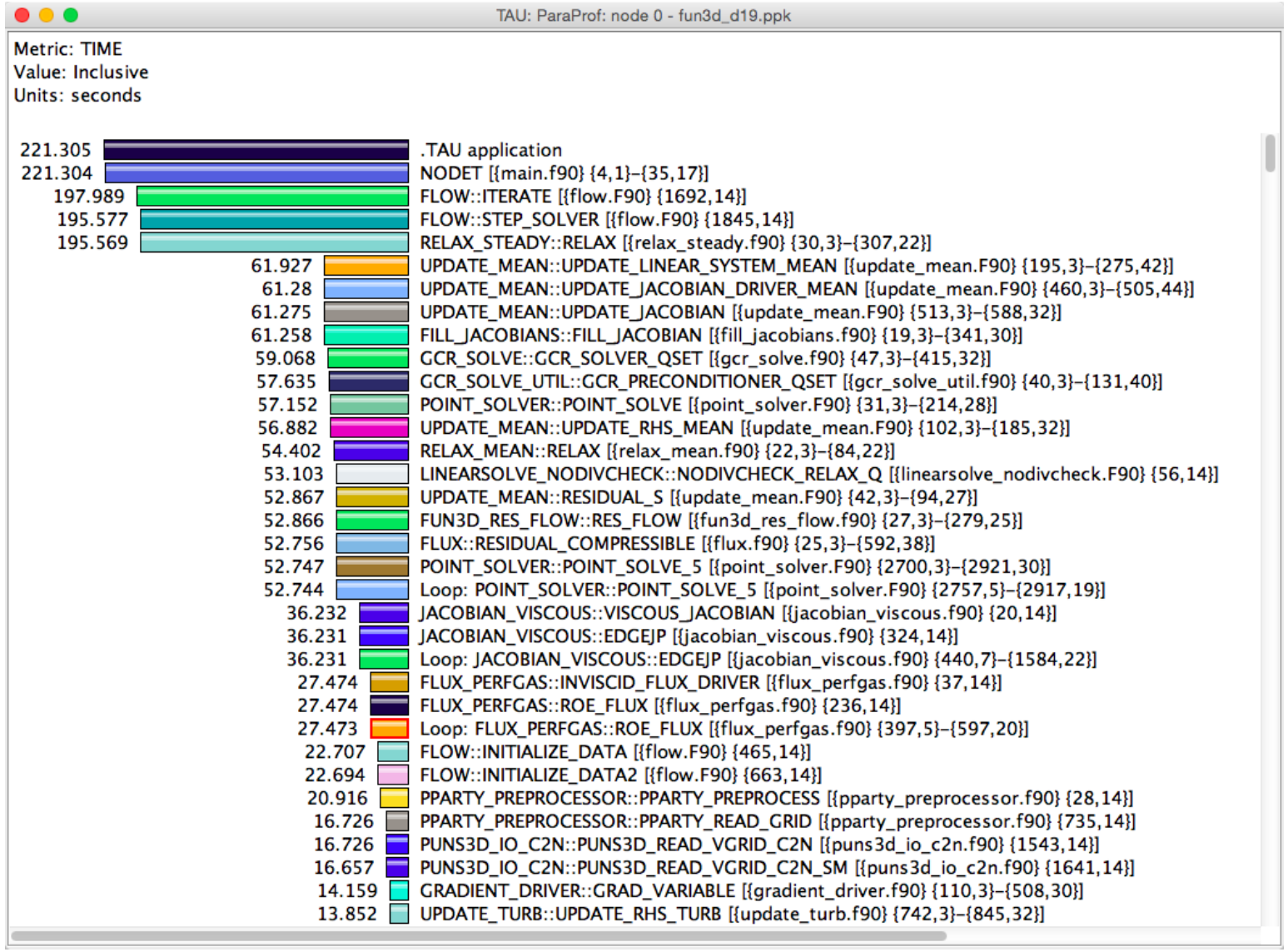
- Much more detailed information

Disadvantage:

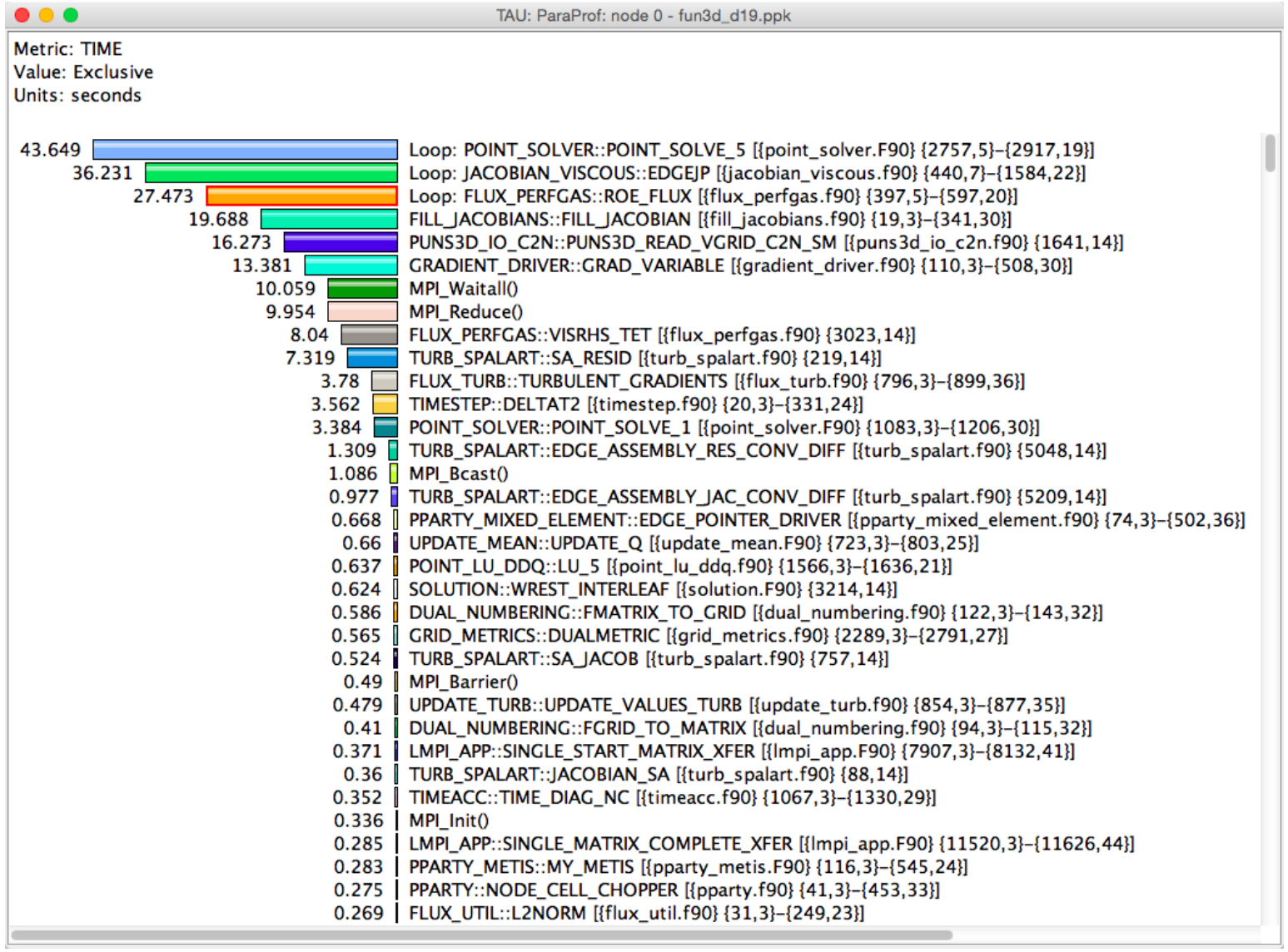
- Processing of source-code / executable necessary
- Large relative overheads for small functions

```
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");
}
```

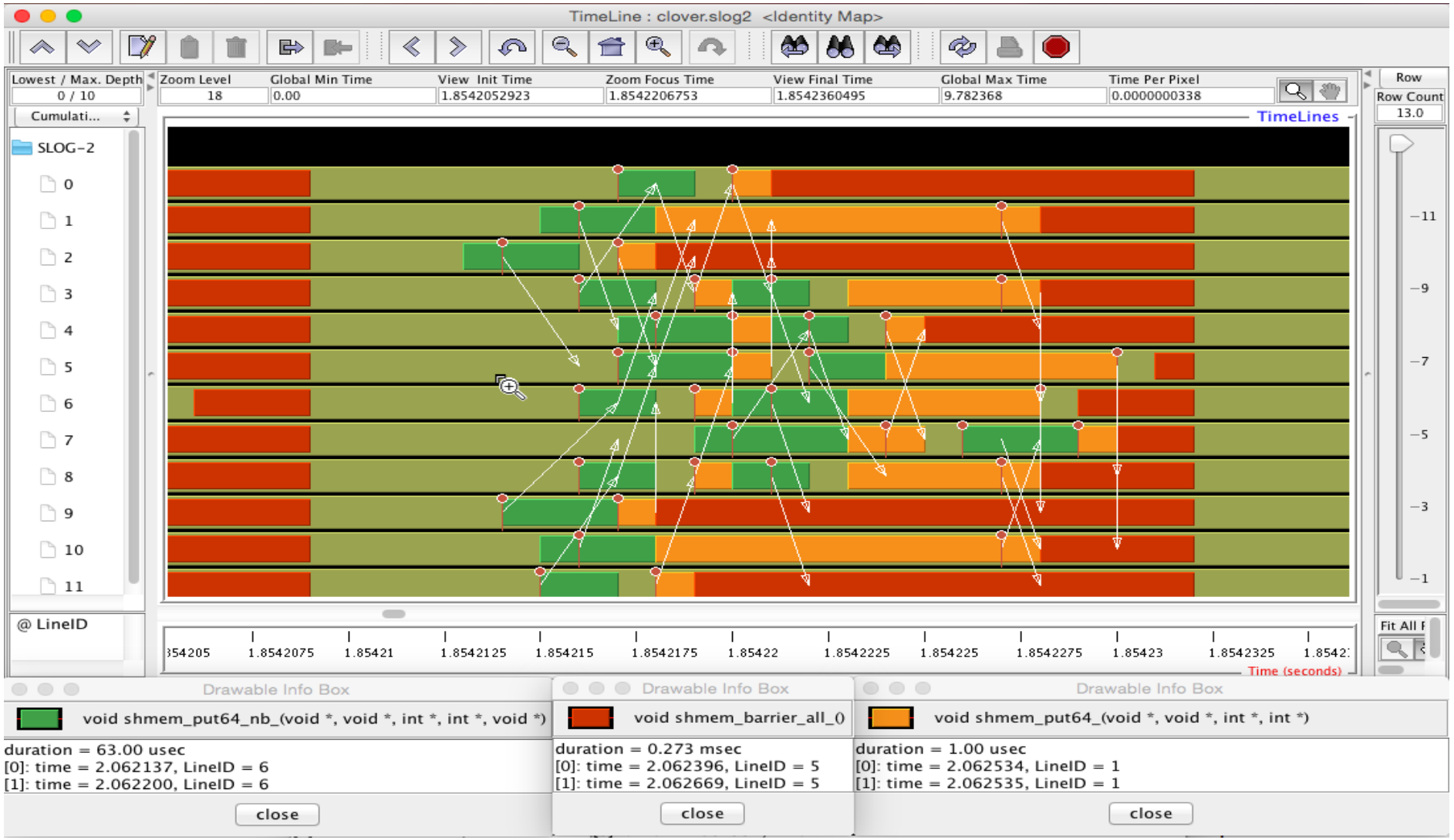
# Inclusive Measurements



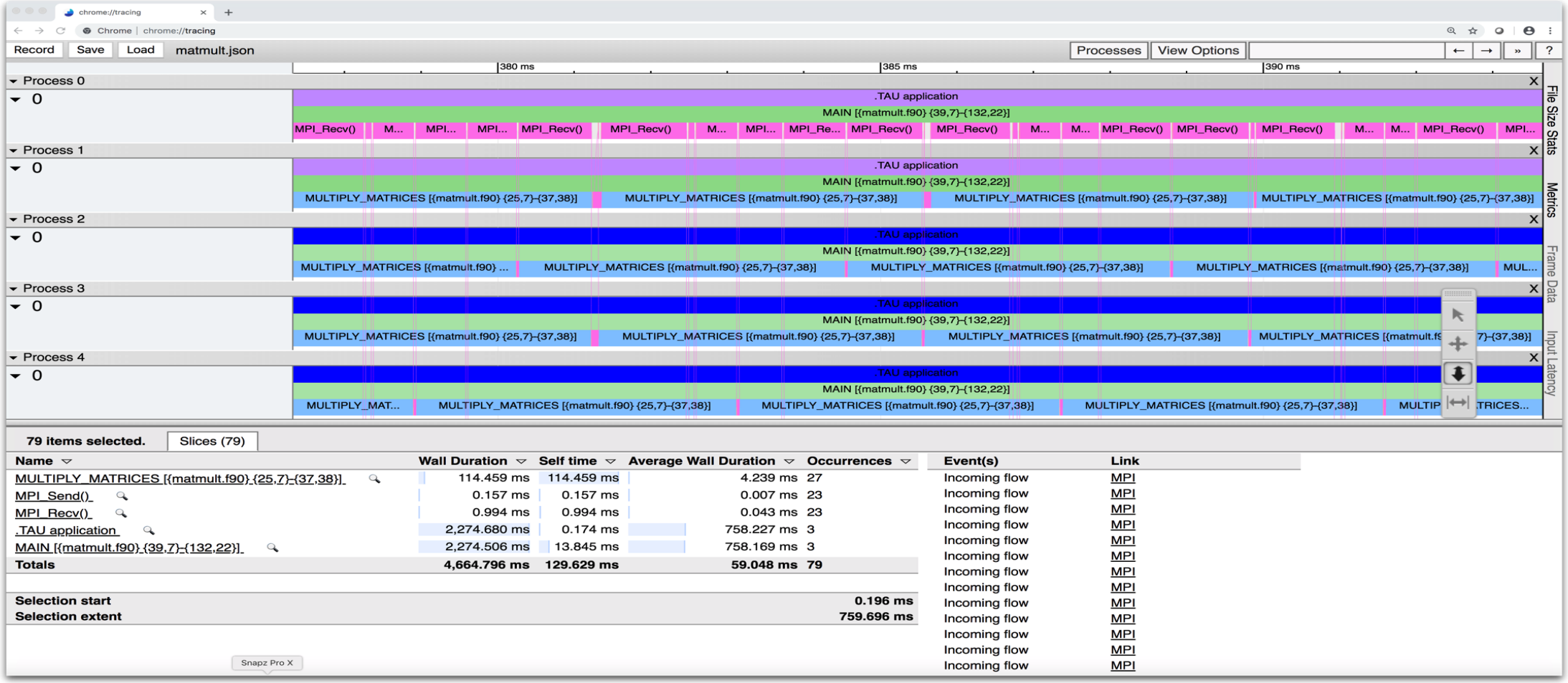
# Exclusive Time



# Tracing: Jumpshot (ships with TAU)



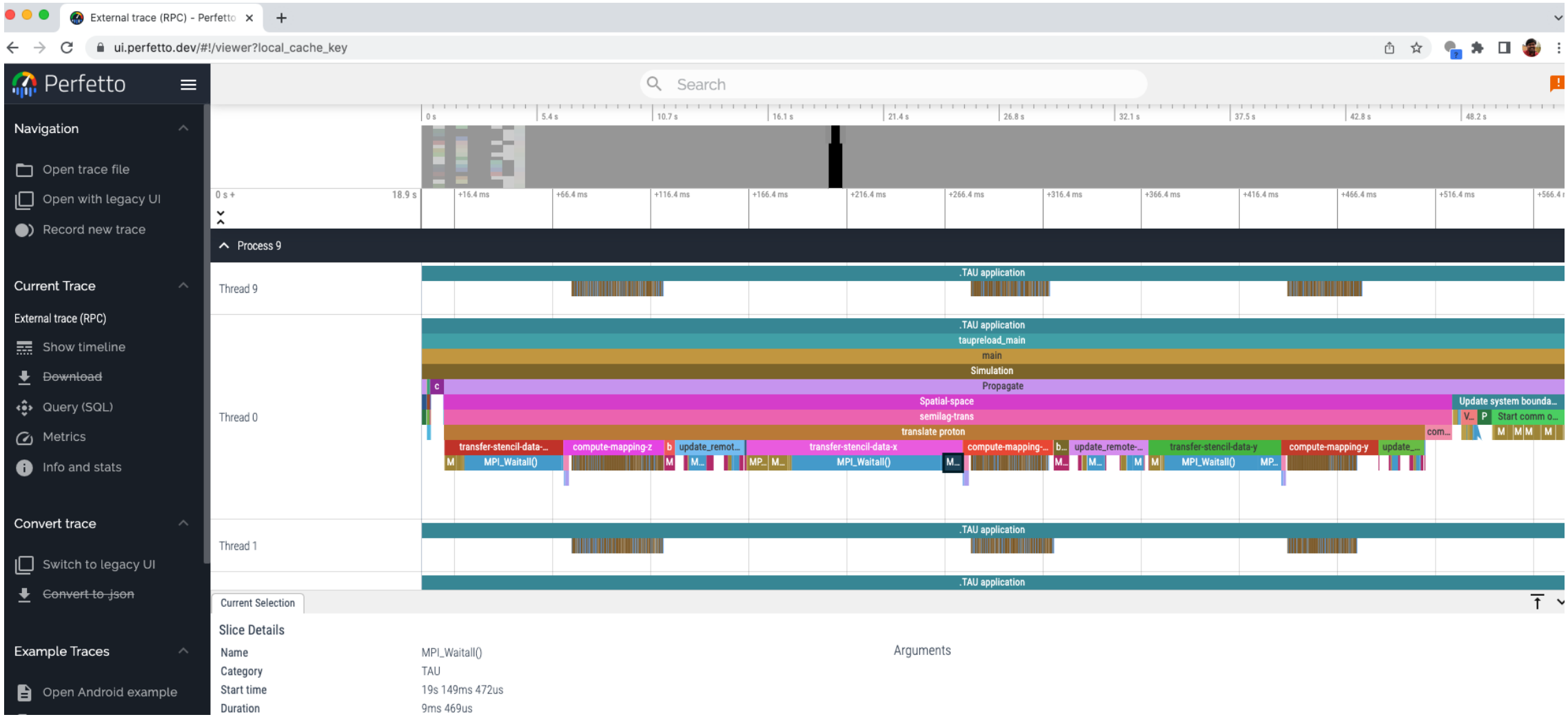
# Tracing: Chrome Browser



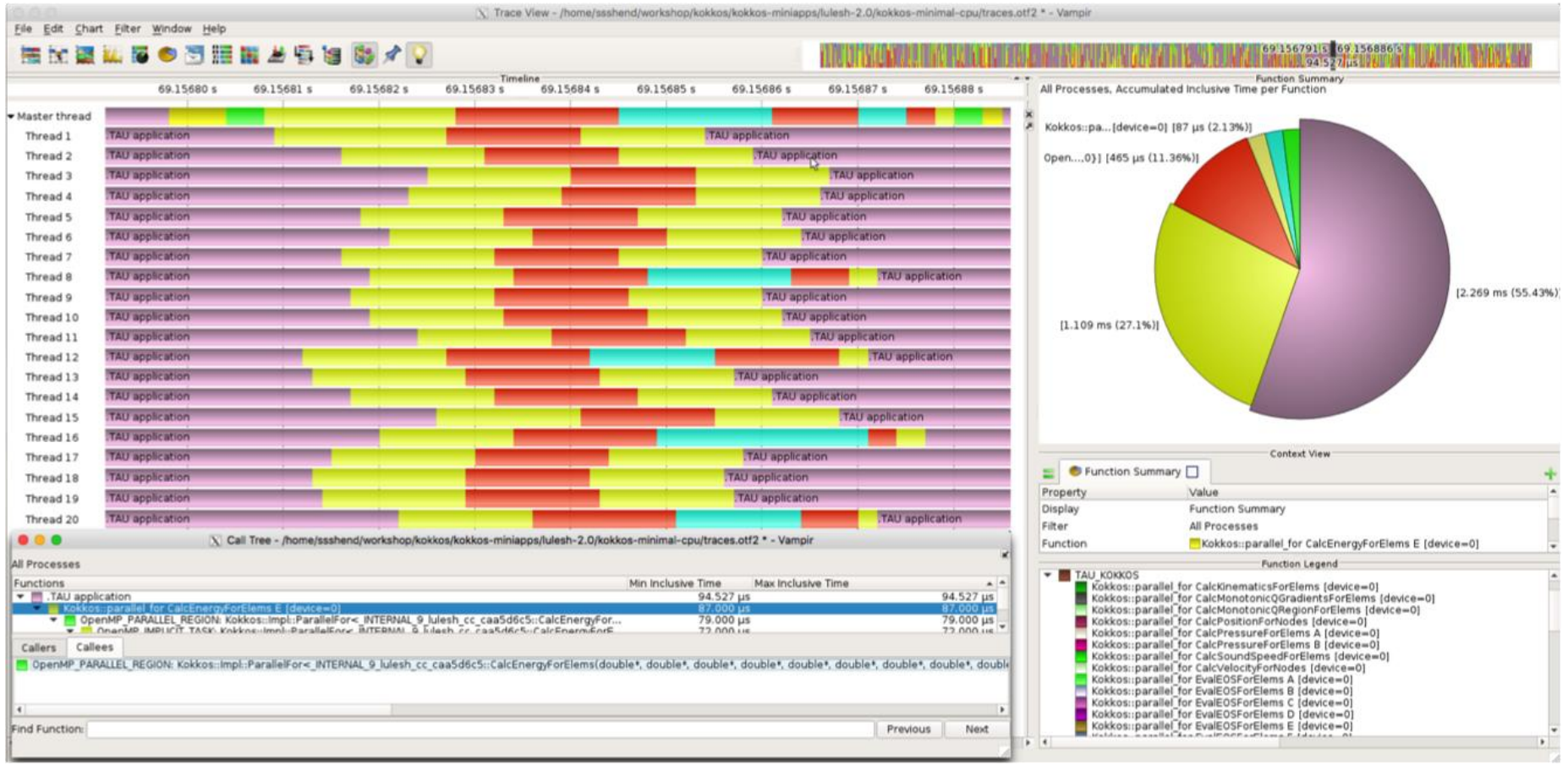

**% export TAU_TRACE=1**
**% mpirun –np 256 tau_exec ./a.out**
**% tau_treemerge.pl; tau_trace2json tau.trc tau.edf –chrome –ignoreatomic –o app.json**

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

# Perfetto.dev



# Vampir [TU Dresden] Timeline: Kokkos

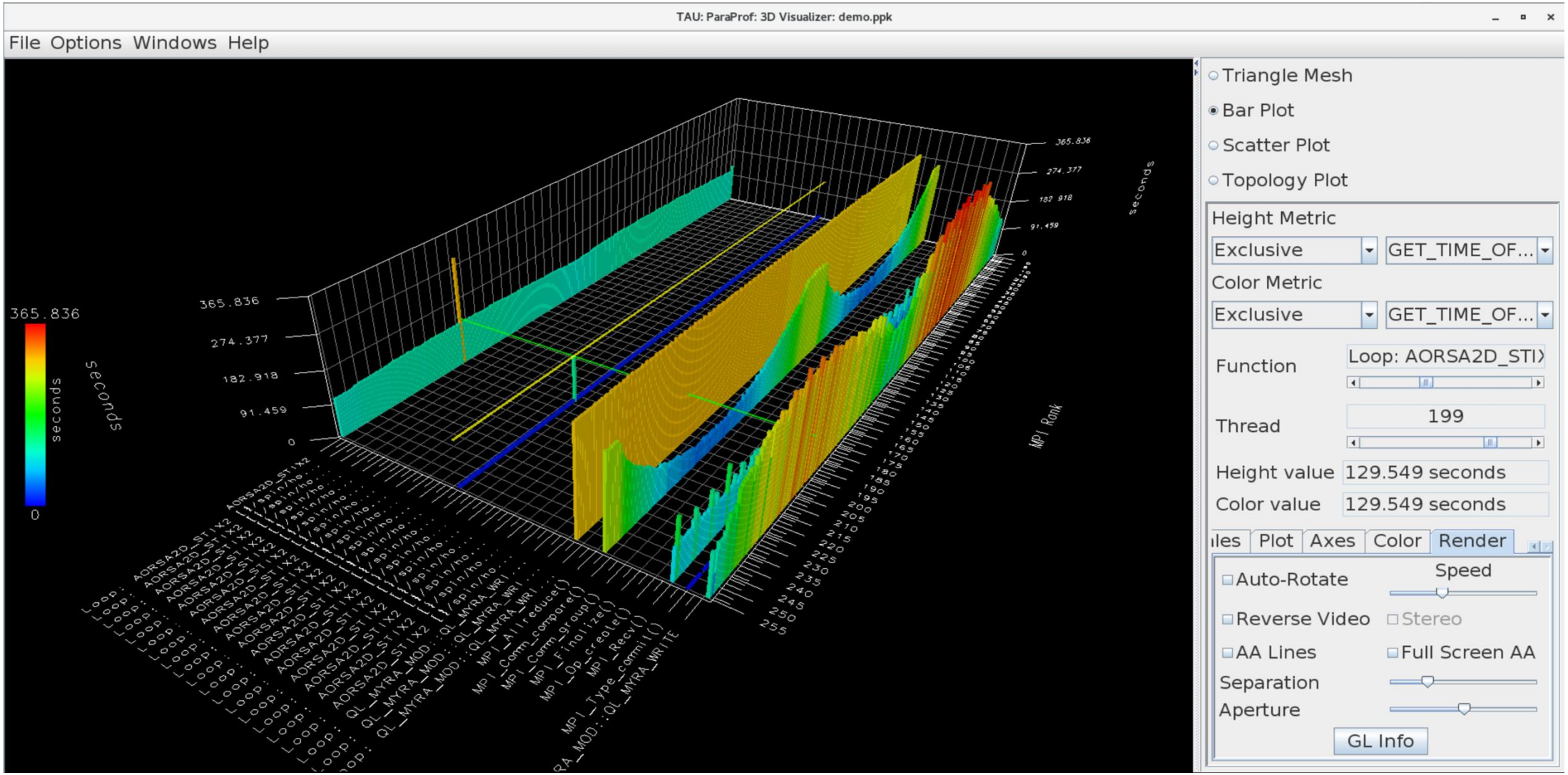



```
% export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2
% tau_exec -T ompt  –ompt  ./a.out
% vampir traces.otf2 &
```

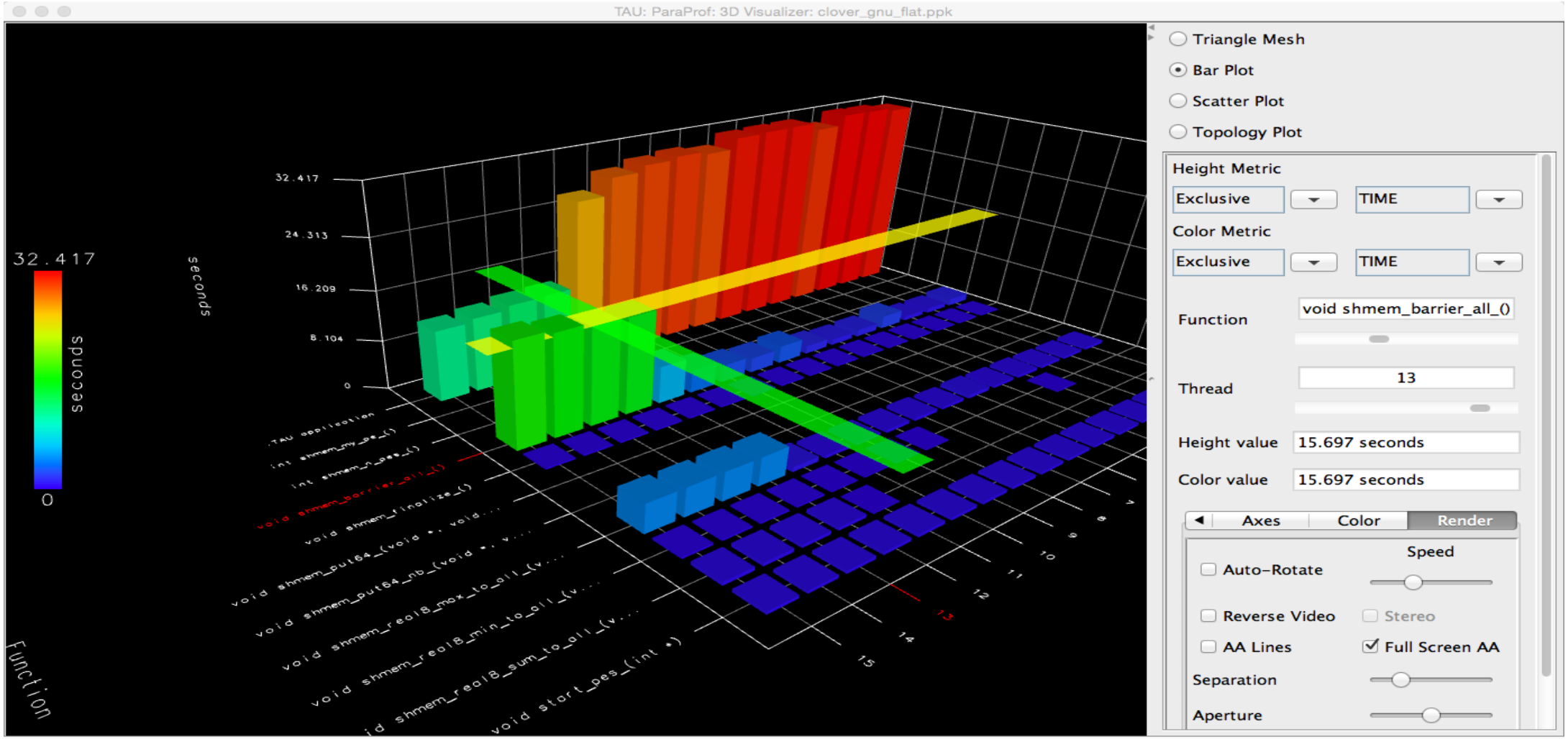
# ParaProf Profile Browser

```
% paraprof
```

# ParaProf 3D Profile Browser

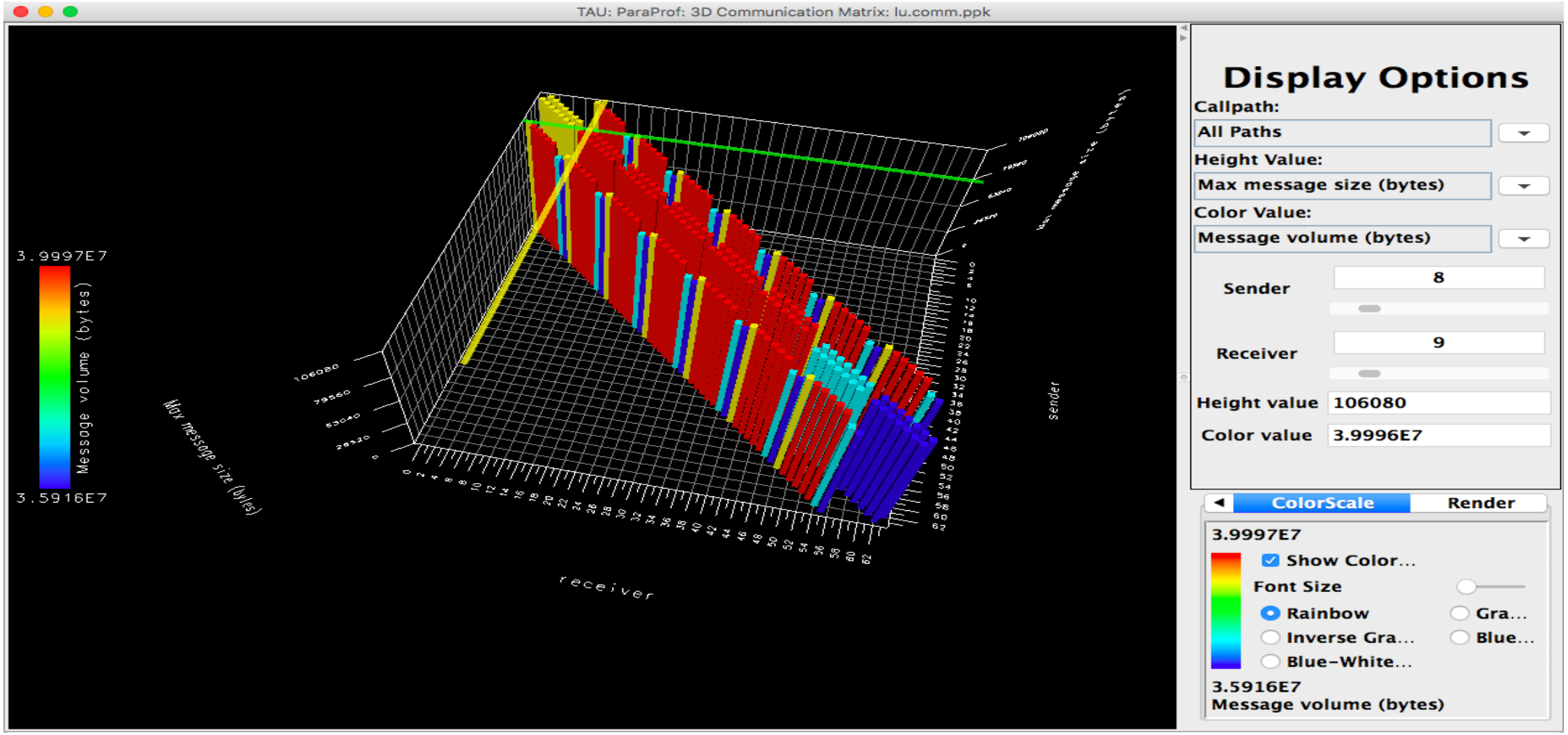


# TAU – ParaProf 3D Visualization



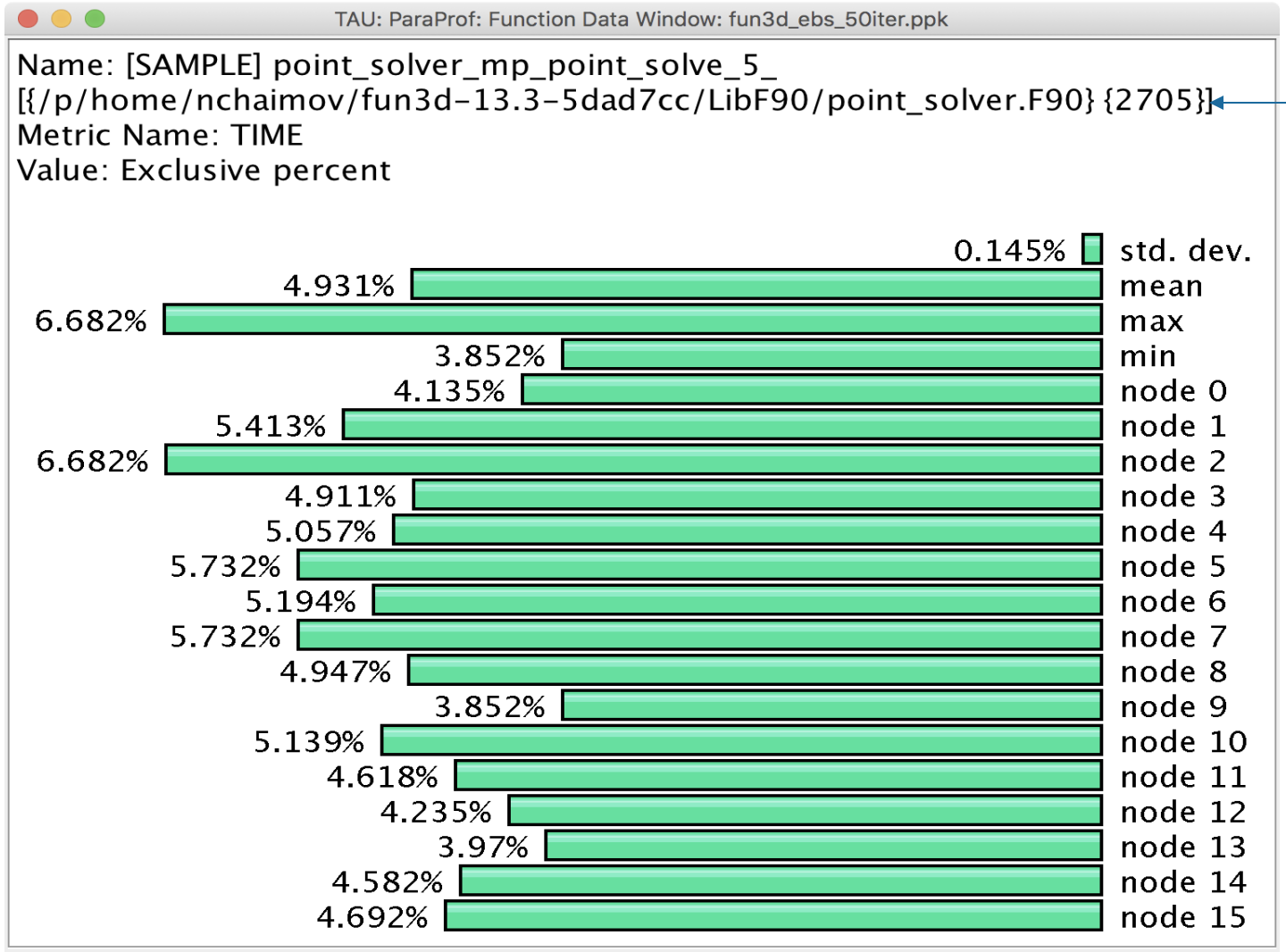

% paraprof app.ppk
Windows -> 3D Visualization -> Bar Plot (right pane)

# TAU – 3D Communication Window




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

# Event Based Sampling (EBS)

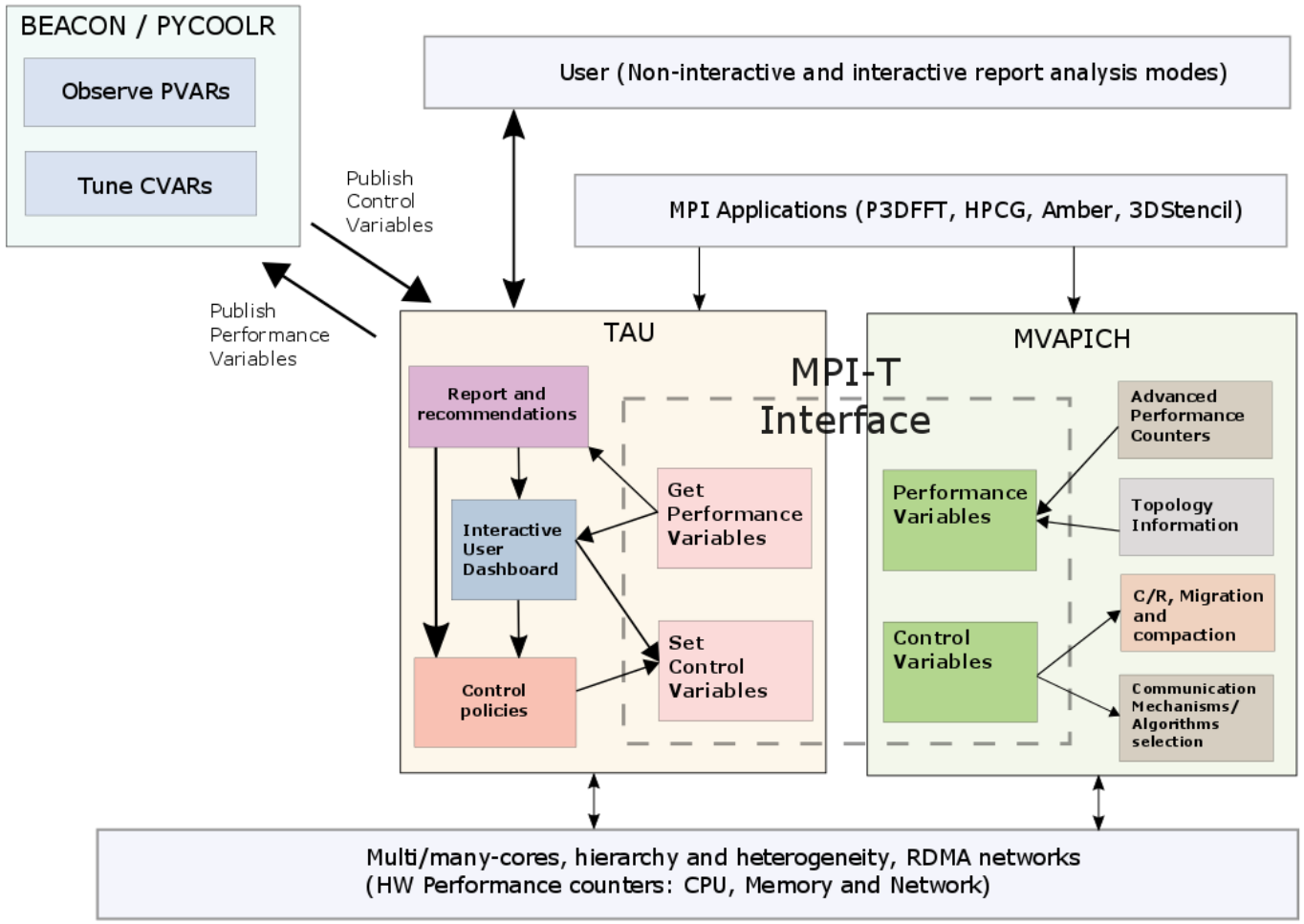



File: point_solver.F90
Line: 2705

Uninstrumented!

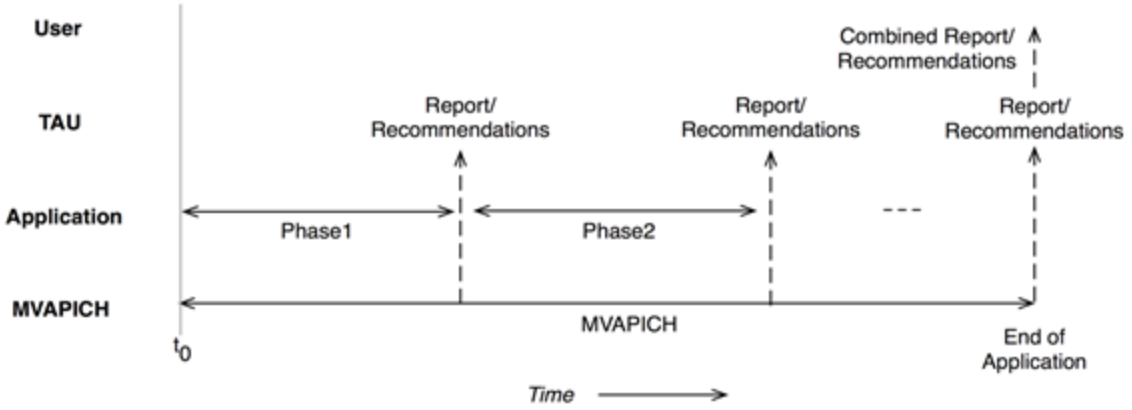
```
% mpirun -n 16 tau_exec -ebs a.out
```

# Integrating TAU with MVAPICH2 through MPI_T Interface

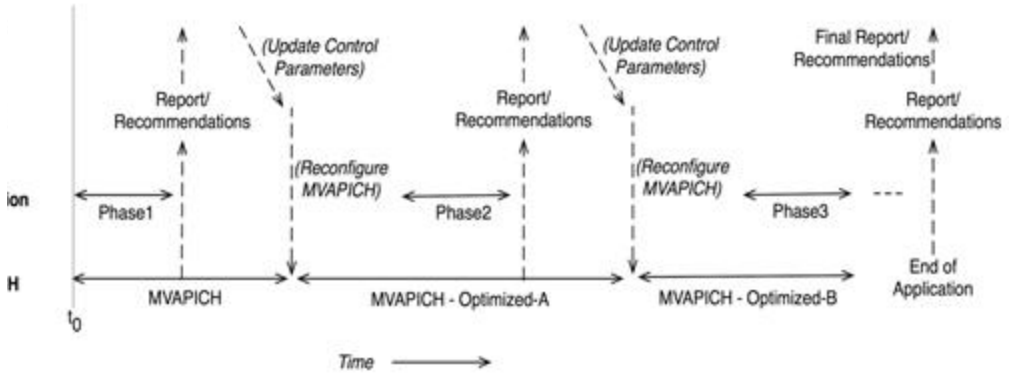



- Enhance existing support for MPI_T in MVAPICH2 to expose a richer set of performance and control variables
- Get and display MPI Performance Variables (PVARs) made available by the runtime in TAU
- Control the runtime's behavior via MPI Control Variables (CVARs)
- Add support to MVAPICH2 and TAU for interactive performance engineering sessions

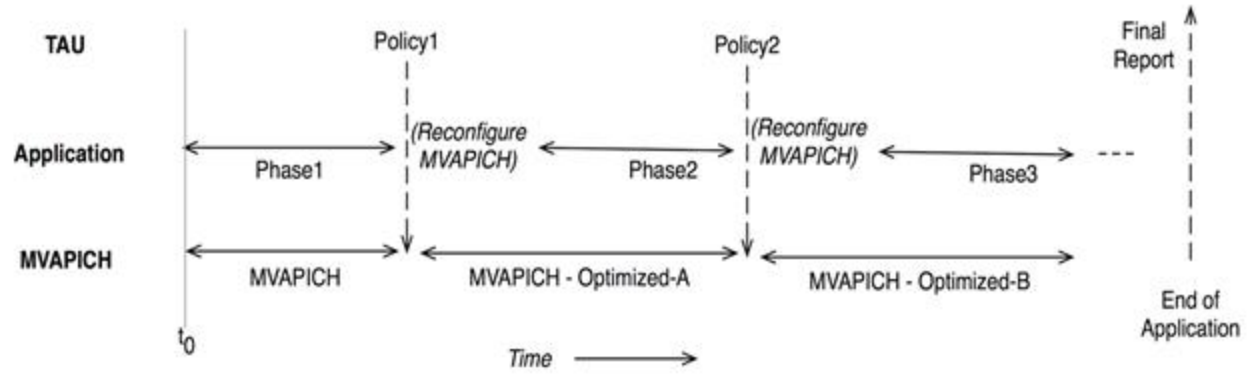
# Three Scenarios for Integration




Scenario 1: Non-interactive mode

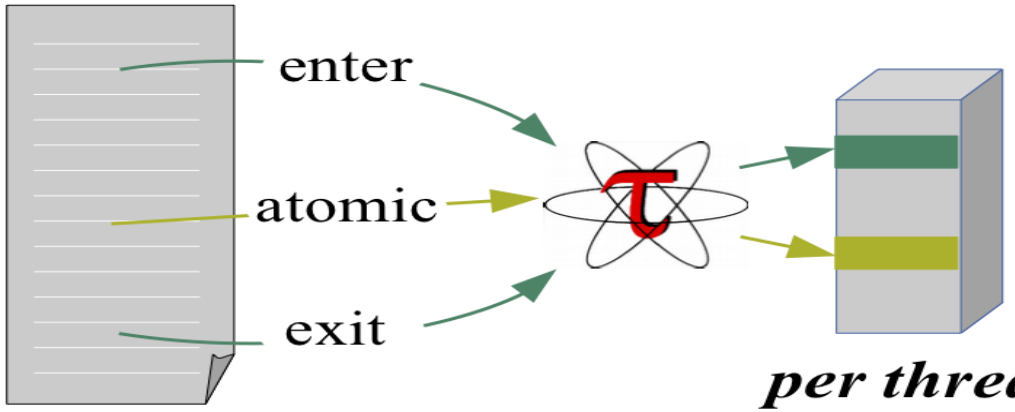


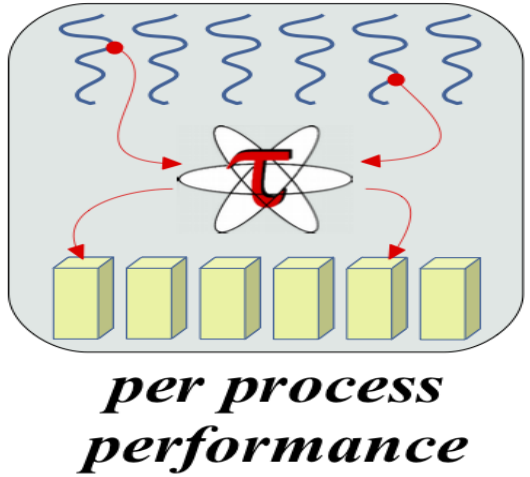

Scenario 2: User-interactive mode




Scenario 3: Policy driven mode

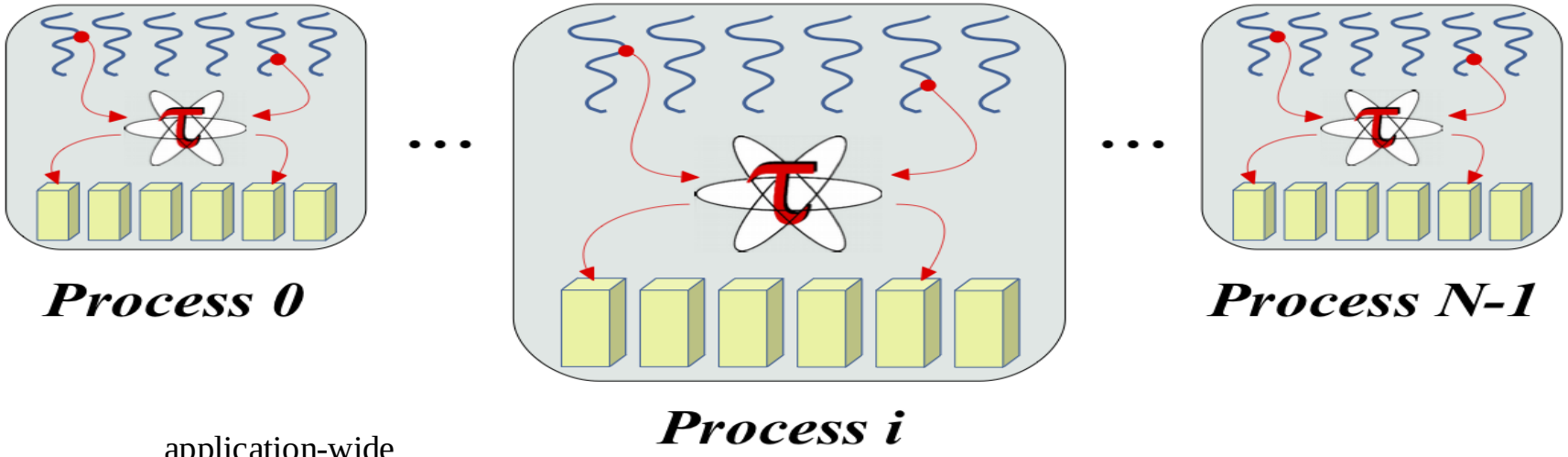
# TAU Performance Measurement Model




enter/exit events
are “interval” events

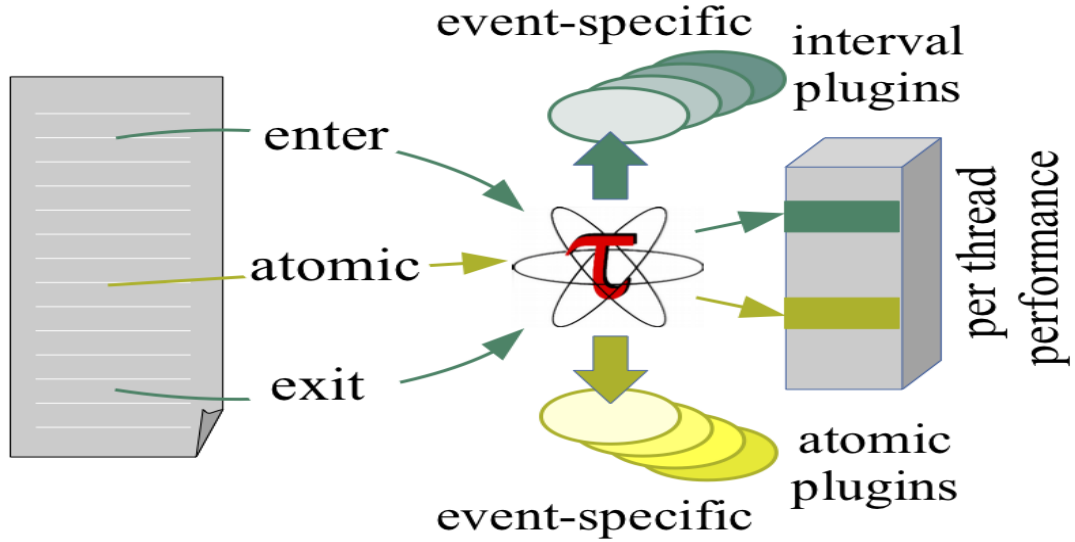
(in shared memory)

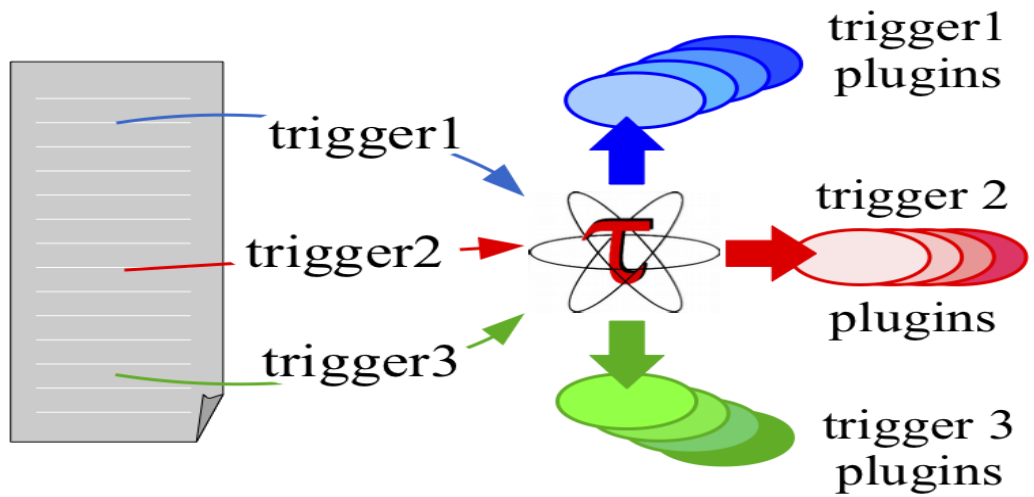



application-wide
performance data

# TAU Plugin Architecture

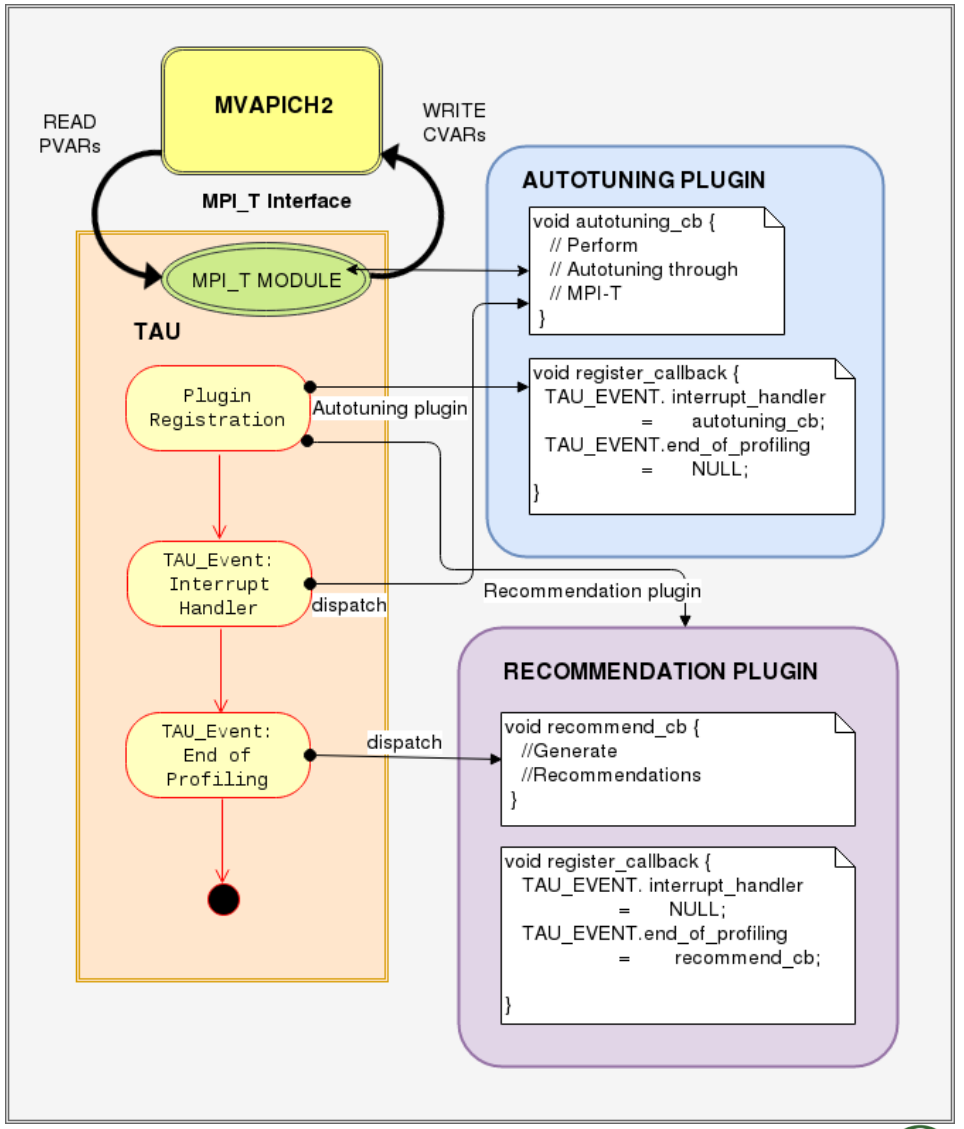
- Extend TAU *event* interface for plugins
  - Events: *interval*, *atomic*
  - Specialized on event ID
  - Synchronous operation
- Create TAU interface for *trigger* plugins
  - Named trigger
  - Pass application data
  - Synchronous
  - Asynchronous using agent plugin



# Plugin-based Infrastructure for Non-Interactive Tuning

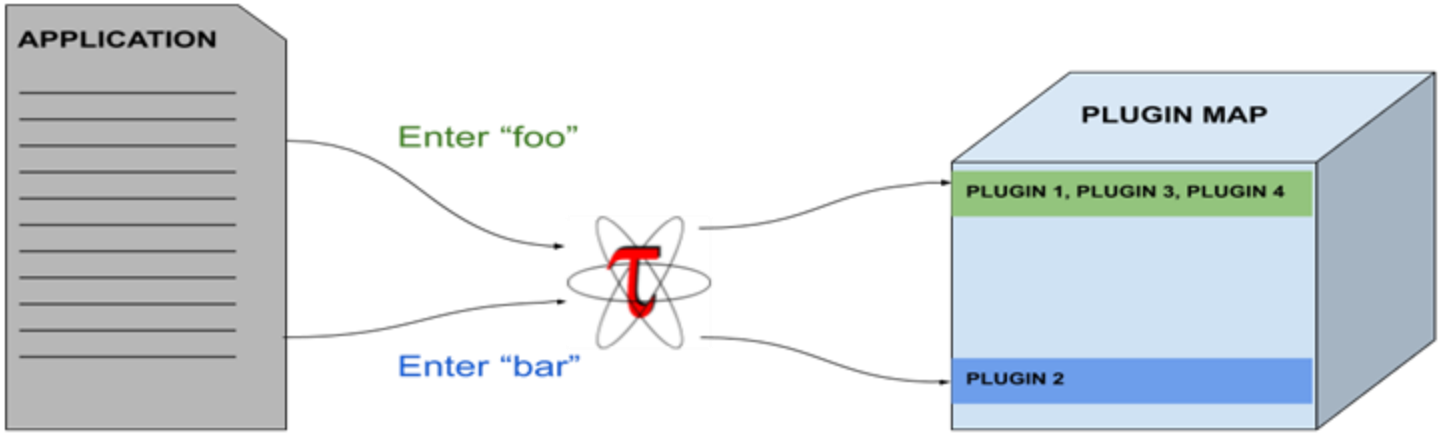
- TAU supports a *fully-customizable* plugin infrastructure based on callback event handler registration for salient states inside TAU:
  - Function Registration / Entry / Exit
  - Phase Entry / Exit
  - Atomic Event Registration / Trigger
  - Init / Finalize Profiling
  - Interrupt Handler
  - *MPI_T*
- Application can define its own “trigger” states and associated plugins
  - Pass arbitrary data to trigger state plugins



# TAU Customization

- TAU states can be *named* or *generic*
- TAU distinguishes named states in a way that allows for separation of occurrence of a state from the action associated with it

  Function entry for “foo” and “bar” represent distinguishable states in TAU
- TAU maintains an internal map of a list of plugins associated with each state



# TAU Runtime Control of Plugin

- TAU defines a plugin API to deliver access control to the internal plugin map
- User can specify a regular expression to control plugins executed for a class of named states at runtime

  Access to map on a process is serialized: application is expected to access map through main thread

# TAU Phase Based Recommendations

- MiniAMR: Benefits from hardware offloading using SHArP hardware offload protocol supported by MVAPICH2 for MPI_Allreduce operation
- Recommendation Plugin:
  Registers callback for *“Phase Exit”* event
  Monitors message size through PMPI interface
  If message size is low and execution time inside MPI_Allreduce is significant, a recommendation is generated on ParaProf (TAU’s GUI) for the user to set the CVAR enabling SHArP

# TAU Per-Phase Recommendations in ParaProf

Metadata for n,c,t 7,0,0

| Name | Value |
|---|---|
| TAU MEMDBG PROTECT BELOW | off |
| TAU MEMDBG PROTECT FREE | off |
| TAU MPI T ENABLE USER TUNING POLICY | off |
| TAU OPENMP RUNTIME | on |
| TAU OPENMP RUNTIME EVENTS | on |
| TAU OPENMP RUNTIME STATES | off |
| TAU OUTPUT CUDA CSV | off |
| TAU PAPI MULTIPLEXING | off |
| TAU PROFILE | on |
| TAU PROFILE FORMAT | profile |
| TAU RECOMMENDATION PHASE ALLOCATE | MPI T RECOMMEND SHARP USAGE: No perfomance benefit foreseen with SHArP usage |
| TAU RECOMMENDATION PHASE DEALLOCATE | MPI T RECOMMEND SHARP USAGE: You could see potential improvement in performance by enabling MV2 ENABLE SHARP in MVAPICH version 2.3a and above |
| TAU RECOMMENDATION PHASE DRIVER | MPI T RECOMMEND SHARP USAGE: You could see potential improvement in performance by enabling MV2 ENABLE SHARP in MVAPICH version 2.3a and above |
| TAU RECOMMENDATION PHASE INIT | MPI T RECOMMEND SHARP USAGE: No perfomance benefit foreseen with SHArP usage |
| TAU RECOMMENDATION PHASE PROFILE | MPI T RECOMMEND SHARP USAGE: You could see potential improvement in performance by enabling MV2 ENABLE SHARP in MVAPICH version 2.3a and above |
| TAU REGION ADDRESSES | off |
| TAU SAMPLING | off |
| TAU SHOW MEMORY FUNCTIONS | off |
| TAU SIGNALS GDB | off |
| TAU THROTTLE | on |
| TAU THROTTLE NUMCALLS | 100000 |
| TAU THROTTLE PERCALL | 10 |
| TAU TRACE | off |
| TAU TRACE FORMAT | tau |
| TAU TRACK CUDA CDP | off |
| TAU TRACK CUDA ENV | off |
| TAU TRACK CUDA INSTRUCTIONS | |
| TAU TRACK CUDA SASS | off |
| TAU TRACK HEADROOM | off |
| TAU TRACK HEAP | off |
| TAU TRACK IO PARAMS | off |
| TAU TRACK MEMORY FOOTPRINT | off |

# Enhancing MPI_T Support

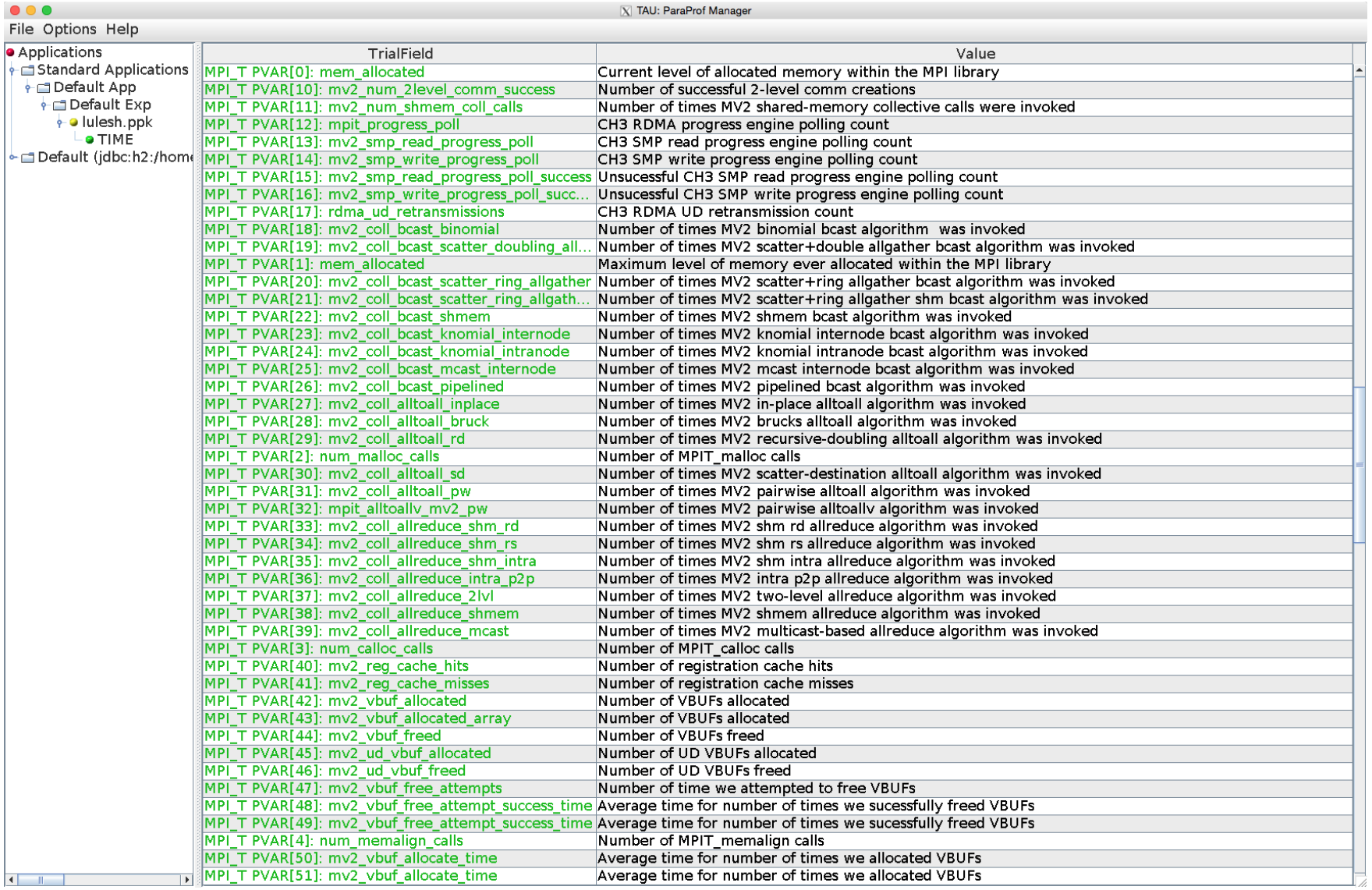
- Introduced support for new MPI_T based CVARs to MVAPICH2
  - MPIR_CVAR_MAX_INLINE_MSG_SZ
    - Controls the message size up to which “inline” transmission of data is supported by MVAPICH2
  - MPIR_CVAR_VBUF_POOL_SIZE
    - Controls the number of internal communication buffers (VBUFs) MVAPICH2 allocates initially. Also, MPIR_CVAR_VBUF_POOL_REDUCED_VALUE[1] ([2…n])
  - MPIR_CVAR_VBUF_SECONDARY_POOL_SIZE
    - Controls the number of VBUFs MVAPICH2 allocates when there are no more free VBUFs available
  - MPIR_CVAR_IBA_EAGER_THRESHOLD
    - Controls the message size where MVAPICH2 switches from eager to rendezvous protocol for large messages
- TAU enhanced with support for setting MPI_T CVARs in a non-interactive mode for uninstrumented applications

# MVAPICH2

- Several new MPI_T based PVARs added to MVAPICH2
  - mv2_vbuf_max_use, mv2_total_vbuf_memory etc
- Enhanced TAU with support for tracking of MPI_T PVARs and CVARs for uninstrumented applications
  - ParaProf, TAU's visualization front end, enhanced with support for displaying PVARs and CVARs
  - TAU provides tau_exec, a tool to transparently instrument MPI routines
    - Uninstrumented:
      % mpirun –np 1024 ./a.out
    - Instrumented:

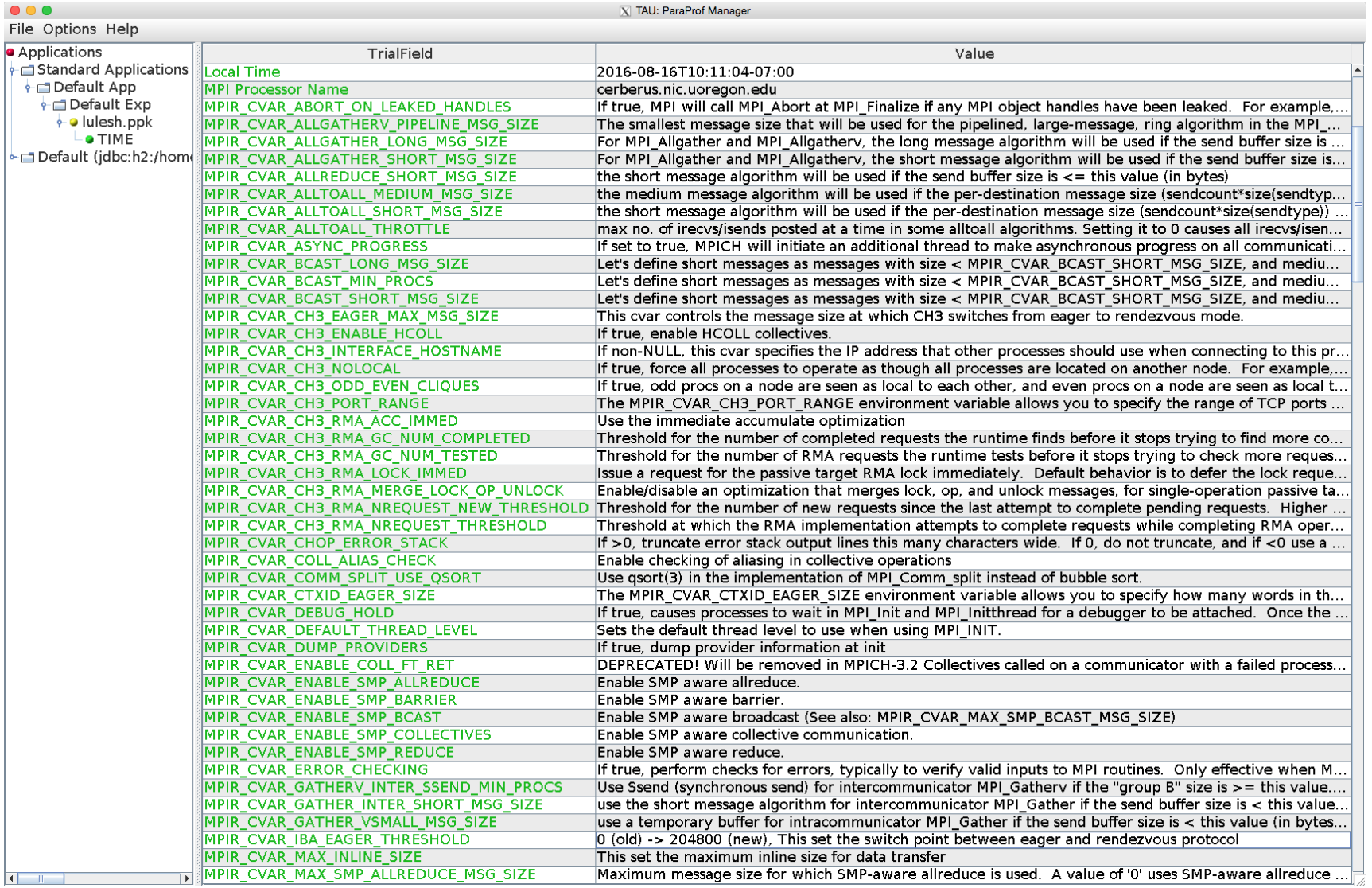
```
% export TAU_TRACK_MPI_T_PVARS=1
% export TAU_MPI_T_CVAR_METRICS=MPIR_CVAR_VBUF_POOL_SIZE
% export TAU_MPI_T_CVAR_VALUES=16
% mpirun -np 1024 tau_exec -T mvapich2,mpit ./a.out
```

# PVARs Exposed by MVAPICH2




| TrialField | Value |
|---|---|
| MPI_T PVAR[0]: mem_allocated | Current level of allocated memory within the MPI library |
| MPI_T PVAR[10]: mv2_num_2level_comm_success | Number of successful 2-level comm creations |
| MPI_T PVAR[11]: mv2_num_shmem_coll_calls | Number of times MV2 shared-memory collective calls were invoked |
| MPI_T PVAR[12]: mpit_progress_poll | CH3 RDMA progress engine polling count |
| MPI_T PVAR[13]: mv2_smp_read_progress_poll | CH3 SMP read progress engine polling count |
| MPI_T PVAR[14]: mv2_smp_write_progress_poll | CH3 SMP write progress engine polling count |
| MPI_T PVAR[15]: mv2_smp_read_progress_poll_success | Unsucessful CH3 SMP read progress engine polling count |
| MPI_T PVAR[16]: mv2_smp_write_progress_poll_succ... | Unsucessful CH3 SMP write progress engine polling count |
| MPI_T PVAR[17]: rdma_ud_retransmissions | CH3 RDMA UD retransmission count |
| MPI_T PVAR[18]: mv2_coll_bcast_binomial | Number of times MV2 binomial bcast algorithm was invoked |
| MPI_T PVAR[19]: mv2_coll_bcast_scatter_doubling_all... | Number of times MV2 scatter+double allgather bcast algorithm was invoked |
| MPI_T PVAR[1]: mem_allocated | Maximum level of memory ever allocated within the MPI library |
| MPI_T PVAR[20]: mv2_coll_bcast_scatter_ring_allgather | Number of times MV2 scatter+ring allgather bcast algorithm was invoked |
| MPI_T PVAR[21]: mv2_coll_bcast_scatter_ring_allgath... | Number of times MV2 scatter+ring allgather shm bcast algorithm was invoked |
| MPI_T PVAR[22]: mv2_coll_bcast_shmem | Number of times MV2 shmem bcast algorithm was invoked |
| MPI_T PVAR[23]: mv2_coll_bcast_knomial_internode | Number of times MV2 knomial internode bcast algorithm was invoked |
| MPI_T PVAR[24]: mv2_coll_bcast_knomial_intranode | Number of times MV2 knomial intranode bcast algorithm was invoked |
| MPI_T PVAR[25]: mv2_coll_bcast_mcast_internode | Number of times MV2 mcast internode bcast algorithm was invoked |
| MPI_T PVAR[26]: mv2_coll_bcast_pipelined | Number of times MV2 pipelined bcast algorithm was invoked |
| MPI_T PVAR[27]: mv2_coll_alltoall_inplace | Number of times MV2 in-place alltoall algorithm was invoked |
| MPI_T PVAR[28]: mv2_coll_alltoall_bruck | Number of times MV2 brucks alltoall algorithm was invoked |
| MPI_T PVAR[29]: mv2_coll_alltoall_rd | Number of times MV2 recursive-doubling alltoall algorithm was invoked |
| MPI_T PVAR[2]: num_malloc_calls | Number of MPIT_malloc calls |
| MPI_T PVAR[30]: mv2_coll_alltoall_sd | Number of times MV2 scatter-destination alltoall algorithm was invoked |
| MPI_T PVAR[31]: mv2_coll_alltoall_pw | Number of times MV2 pairwise alltoall algorithm was invoked |
| MPI_T PVAR[32]: mpit_alltoallv_mv2_pw | Number of times MV2 pairwise alltoallv algorithm was invoked |
| MPI_T PVAR[33]: mv2_coll_allreduce_shm_rd | Number of times MV2 shm rd allreduce algorithm was invoked |
| MPI_T PVAR[34]: mv2_coll_allreduce_shm_rs | Number of times MV2 shm rs allreduce algorithm was invoked |
| MPI_T PVAR[35]: mv2_coll_allreduce_shm_intra | Number of times MV2 shm intra allreduce algorithm was invoked |
| MPI_T PVAR[36]: mv2_coll_allreduce_intra_p2p | Number of times MV2 intra p2p allreduce algorithm was invoked |
| MPI_T PVAR[37]: mv2_coll_allreduce_2lvl | Number of times MV2 two-level allreduce algorithm was invoked |
| MPI_T PVAR[38]: mv2_coll_allreduce_shmem | Number of times MV2 shmem allreduce algorithm was invoked |
| MPI_T PVAR[39]: mv2_coll_allreduce_mcast | Number of times MV2 multicast-based allreduce algorithm was invoked |
| MPI_T PVAR[3]: num_calloc_calls | Number of MPIT_calloc calls |
| MPI_T PVAR[40]: mv2_reg_cache_hits | Number of registration cache hits |
| MPI_T PVAR[41]: mv2_reg_cache_misses | Number of registration cache misses |
| MPI_T PVAR[42]: mv2_vbuf_allocated | Number of VBUFs allocated |
| MPI_T PVAR[43]: mv2_vbuf_allocated_array | Number of VBUFs allocated |
| MPI_T PVAR[44]: mv2_vbuf_freed | Number of VBUFs freed |
| MPI_T PVAR[45]: mv2_ud_vbuf_allocated | Number of UD VBUFs allocated |
| MPI_T PVAR[46]: mv2_ud_vbuf_freed | Number of UD VBUFs freed |
| MPI_T PVAR[47]: mv2_vbuf_free_attempts | Number of time we attempted to free VBUFs |
| MPI_T PVAR[48]: mv2_vbuf_free_attempt_success_time | Average time for number of times we sucessfully freed VBUFs |
| MPI_T PVAR[49]: mv2_vbuf_free_attempt_success_time | Average time for number of times we sucessfully freed VBUFs |
| MPI_T PVAR[4]: num_memalign_calls | Number of MPIT_memalign calls |
| MPI_T PVAR[50]: mv2_vbuf_allocate_time | Average time for number of times we allocated VBUFs |
| MPI_T PVAR[51]: mv2_vbuf_allocate_time | Average time for number of times we allocated VBUFs |

# CVARs Exposed by MVAPICH2




| TrialField | Value |
|---|---|
| Local Time | 2016-08-16T10:11:04-07:00 |
| MPI Processor Name | cerberus.nic.uoregon.edu |
| MPIR_CVAR_ABORT_ON_LEAKED_HANDLES | If true, MPI will call MPI_Abort at MPI_Finalize if any MPI object handles have been leaked. For example,... |
| MPIR_CVAR_ALLGATHERV_PIPELINE_MSG_SIZE | The smallest message size that will be used for the pipelined, large-message, ring algorithm in the MPI_... |
| MPIR_CVAR_ALLGATHER_LONG_MSG_SIZE | For MPI_Allgather and MPI_Allgatherv, the long message algorithm will be used if the send buffer size is ... |
| MPIR_CVAR_ALLGATHER_SHORT_MSG_SIZE | For MPI_Allgather and MPI_Allgatherv, the short message algorithm will be used if the send buffer size is... |
| MPIR_CVAR_ALLREDUCE_SHORT_MSG_SIZE | the short message algorithm will be used if the send buffer size is <= this value (in bytes) |
| MPIR_CVAR_ALLTOALL_MEDIUM_MSG_SIZE | the medium message algorithm will be used if the per-destination message size (sendcount*size(sendtyp... |
| MPIR_CVAR_ALLTOALL_SHORT_MSG_SIZE | the short message algorithm will be used if the per-destination message size (sendcount*size(sendtype)) ... |
| MPIR_CVAR_ALLTOALL_THROTTLE | max no. of irecvs/isends posted at a time in some alltoall algorithms. Setting it to 0 causes all irecvs/isen... |
| MPIR_CVAR_ASYNC_PROGRESS | If set to true, MPICH will initiate an additional thread to make asynchronous progress on all communicati... |
| MPIR_CVAR_BCAST_LONG_MSG_SIZE | Let's define short messages as messages with size < MPIR_CVAR_BCAST_SHORT_MSG_SIZE, and mediu... |
| MPIR_CVAR_BCAST_MIN_PROCS | Let's define short messages as messages with size < MPIR_CVAR_BCAST_SHORT_MSG_SIZE, and mediu... |
| MPIR_CVAR_BCAST_SHORT_MSG_SIZE | Let's define short messages as messages with size < MPIR_CVAR_BCAST_SHORT_MSG_SIZE, and mediu... |
| MPIR_CVAR_CH3_EAGER_MAX_MSG_SIZE | This cvar controls the message size at which CH3 switches from eager to rendezvous mode. |
| MPIR_CVAR_CH3_ENABLE_HCOLL | If true, enable HCOLL collectives. |
| MPIR_CVAR_CH3_INTERFACE_HOSTNAME | If non-NULL, this cvar specifies the IP address that other processes should use when connecting to this pr... |
| MPIR_CVAR_CH3_NOLOCAL | If true, force all processes to operate as though all processes are located on another node. For example,... |
| MPIR_CVAR_CH3_ODD_EVEN_CLIQUES | If true, odd procs on a node are seen as local to each other, and even procs on a node are seen as local t... |
| MPIR_CVAR_CH3_PORT_RANGE | The MPIR_CVAR_CH3_PORT_RANGE environment variable allows you to specify the range of TCP ports ... |
| MPIR_CVAR_CH3_RMA_ACC_IMMED | Use the immediate accumulate optimization |
| MPIR_CVAR_CH3_RMA_GC_NUM_COMPLETED | Threshold for the number of completed requests the runtime finds before it stops trying to find more co... |
| MPIR_CVAR_CH3_RMA_GC_NUM_TESTED | Threshold for the number of RMA requests the runtime tests before it stops trying to check more reques... |
| MPIR_CVAR_CH3_RMA_LOCK_IMMED | Issue a request for the passive target RMA lock immediately. Default behavior is to defer the lock reque... |
| MPIR_CVAR_CH3_RMA_MERGE_LOCK_OP_UNLOCK | Enable/disable an optimization that merges lock, op, and unlock messages, for single-operation passive ta... |
| MPIR_CVAR_CH3_RMA_NREQUEST_NEW_THRESHOLD | Threshold for the number of new requests since the last attempt to complete pending requests. Higher ... |
| MPIR_CVAR_CH3_RMA_NREQUEST_THRESHOLD | Threshold at which the RMA implementation attempts to complete requests while completing RMA oper... |
| MPIR_CVAR_CHOP_ERROR_STACK | If >0, truncate error stack output lines this many characters wide. If 0, do not truncate, and if <0 use a ... |
| MPIR_CVAR_COLL_ALIAS_CHECK | Enable checking of aliasing in collective operations |
| MPIR_CVAR_COMM_SPLIT_USE_QSORT | Use qsort(3) in the implementation of MPI_Comm_split instead of bubble sort. |
| MPIR_CVAR_CTXID_EAGER_SIZE | The MPIR_CVAR_CTXID_EAGER_SIZE environment variable allows you to specify how many words in th... |
| MPIR_CVAR_DEBUG_HOLD | If true, causes processes to wait in MPI_Init and MPI_Initthread for a debugger to be attached. Once the ... |
| MPIR_CVAR_DEFAULT_THREAD_LEVEL | Sets the default thread level to use when using MPI_INIT. |
| MPIR_CVAR_DUMP_PROVIDERS | If true, dump provider information at init |
| MPIR_CVAR_ENABLE_COLL_FT_RET | DEPRECATED! Will be removed in MPICH-3.2 Collectives called on a communicator with a failed process... |
| MPIR_CVAR_ENABLE_SMP_ALLREDUCE | Enable SMP aware allreduce. |
| MPIR_CVAR_ENABLE_SMP_BARRIER | Enable SMP aware barrier. |
| MPIR_CVAR_ENABLE_SMP_BCAST | Enable SMP aware broadcast (See also: MPIR_CVAR_MAX_SMP_BCAST_MSG_SIZE) |
| MPIR_CVAR_ENABLE_SMP_COLLECTIVES | Enable SMP aware collective communication. |
| MPIR_CVAR_ENABLE_SMP_REDUCE | Enable SMP aware reduce. |
| MPIR_CVAR_ERROR_CHECKING | If true, perform checks for errors, typically to verify valid inputs to MPI routines. Only effective when M... |
| MPIR_CVAR_GATHERV_INTER_SSEND_MIN_PROCS | Use Ssend (synchronous send) for intercommunicator MPI_Gatherv if the "group B" size is >= this value.... |
| MPIR_CVAR_GATHER_INTER_SHORT_MSG_SIZE | use the short message algorithm for intercommunicator MPI_Gather if the send buffer size is < this value... |
| MPIR_CVAR_GATHER_VSMALL_MSG_SIZE | use a temporary buffer for intracommunicator MPI_Gather if the send buffer size is < this value (in bytes... |
| MPIR_CVAR_IBA_EAGER_THRESHOLD | 0 (old) -> 204800 (new), This set the switch point between eager and rendezvous protocol |
| MPIR_CVAR_MAX_INLINE_SIZE | This set the maximum inline size for data transfer |
| MPIR_CVAR_MAX_SMP_ALLREDUCE_MSG_SIZE | Maximum message size for which SMP-aware allreduce is used. A value of '0' uses SMP-aware allreduce ... |

# Using MVAPICH2 and TAU with Multiple CVARs

- To set CVARs or read PVARs using TAU for an uninstrumented binary:

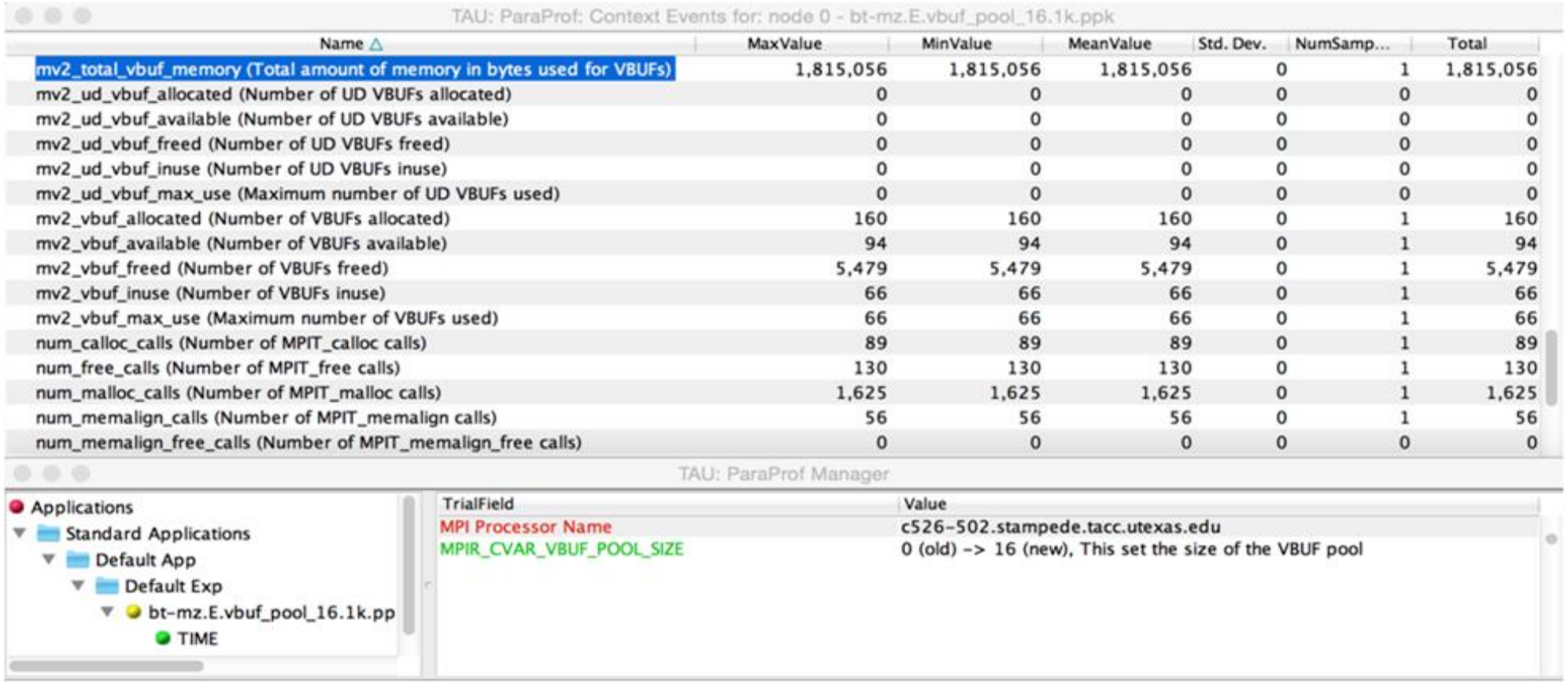
```
% export TAU_TRACK_MPI_T_PVARS=1
% export TAU_MPI_T_CVAR_METRICS=
    MPIR_CVAR_VBUF_POOL_REDUCED_VALUE[1],
    MPIR_CVAR_IBA_EAGER_THRESHOLD
% export TAU_MPI_T_CVAR_VALUES=32,64000
% export PATH=/path/to/tau/x86_64/bin:$PATH
% mpirun -np 1024 tau_exec -T mvapich2,mpit  ./a.out
% paraprof
```

# VBUF usage without CVARs

TAU: ParaProf: Context Events for: node 0 - mpit_withoutcvar_bt.C.1k.ppk

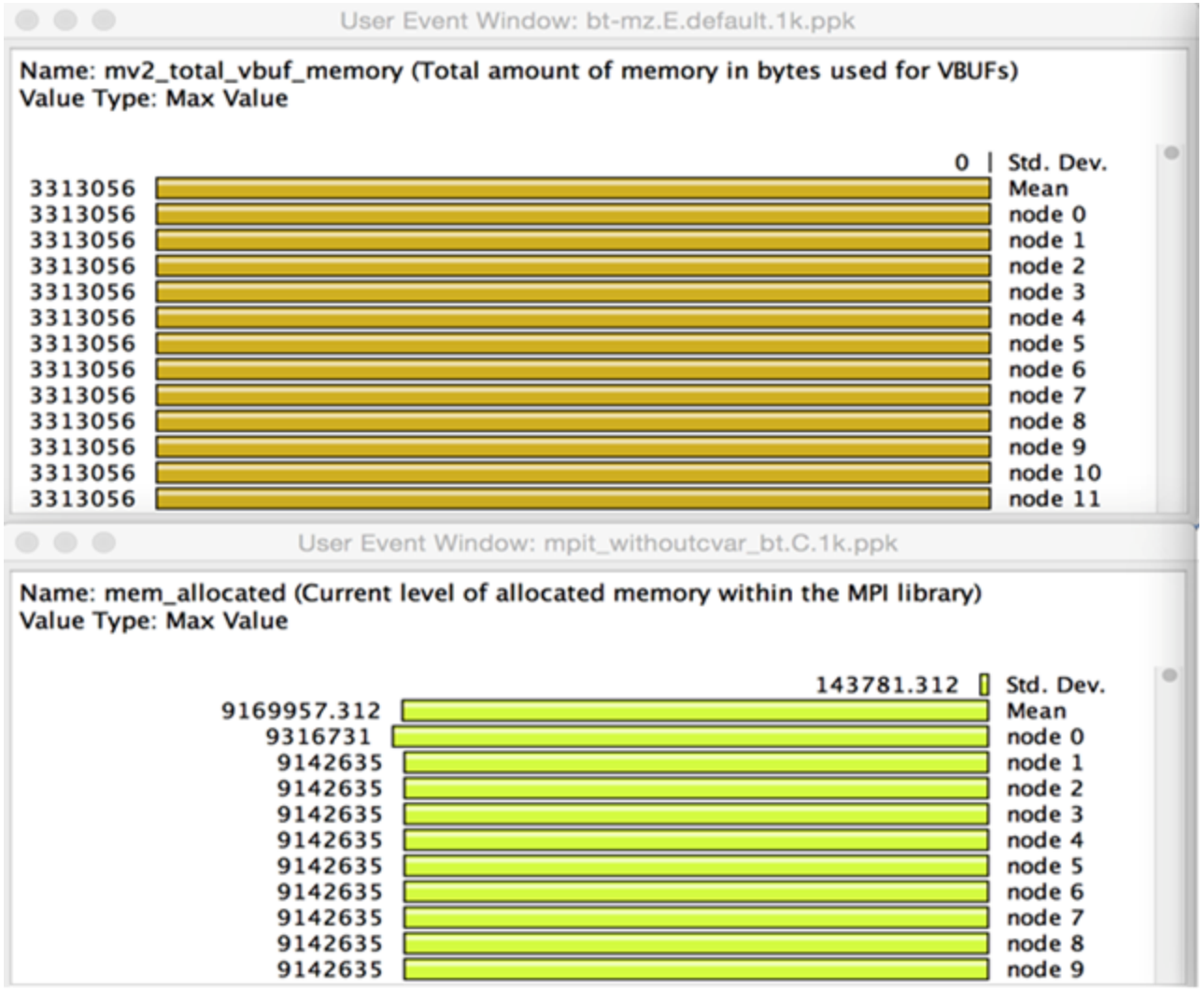
| Name △ | MaxValue | MinValue | MeanValue | Std. Dev. | NumSamples | Total |
|---|---|---|---|---|---|---|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs) | 3,313,056 | 3,313,056 | 3,313,056 | 0 | 1 | 3,313,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated) | 0 | 0 | 0 | 0 | 0 | 0 |
| mv2_ud_vbuf_available (Number of UD VBUFs available) | 0 | 0 | 0 | 0 | 0 | 0 |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed) | 0 | 0 | 0 | 0 | 0 | 0 |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse) | 0 | 0 | 0 | 0 | 0 | 0 |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used) | 0 | 0 | 0 | 0 | 0 | 0 |
| mv2_vbuf_allocated (Number of VBUFs allocated) | 320 | 320 | 320 | 0 | 1 | 320 |
| mv2_vbuf_available (Number of VBUFs available) | 255 | 255 | 255 | 0 | 1 | 255 |
| mv2_vbuf_freed (Number of VBUFs freed) | 25,545 | 25,545 | 25,545 | 0 | 1 | 25,545 |
| mv2_vbuf_inuse (Number of VBUFs inuse) | 65 | 65 | 65 | 0 | 1 | 65 |
| mv2_vbuf_max_use (Maximum number of VBUFs used) | 65 | 65 | 65 | 0 | 1 | 65 |
| num_calloc_calls (Number of MPIT_calloc calls) | 89 | 89 | 89 | 0 | 1 | 89 |
| num_free_calls (Number of MPIT_free calls) | 47,801 | 47,801 | 47,801 | 0 | 1 | 47,801 |
| num_malloc_calls (Number of MPIT_malloc calls) | 49,258 | 49,258 | 49,258 | 0 | 1 | 49,258 |
| num_memalign_calls (Number of MPIT_memalign calls) | 34 | 34 | 34 | 0 | 1 | 34 |
| num_memalign_free_calls (Number of MPIT_memalign_free calls) | 0 | 0 | 0 | 0 | 0 | 0 |

# VBUF usage with CVARs

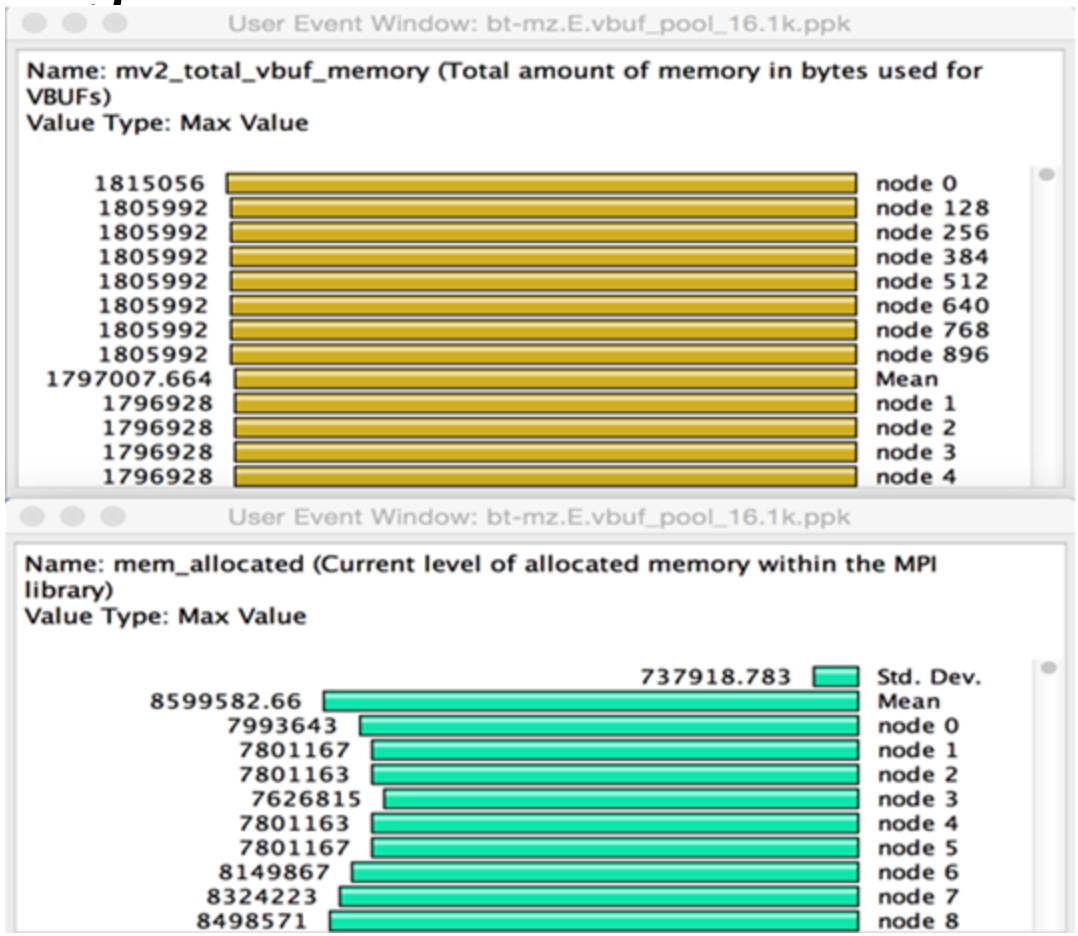



Total memory used by VBUFs is reduced from 3,313,056 to 1,815,056

# VBUF Memory Usage Without CVAR



# VBUF Memory Usage With CVAR

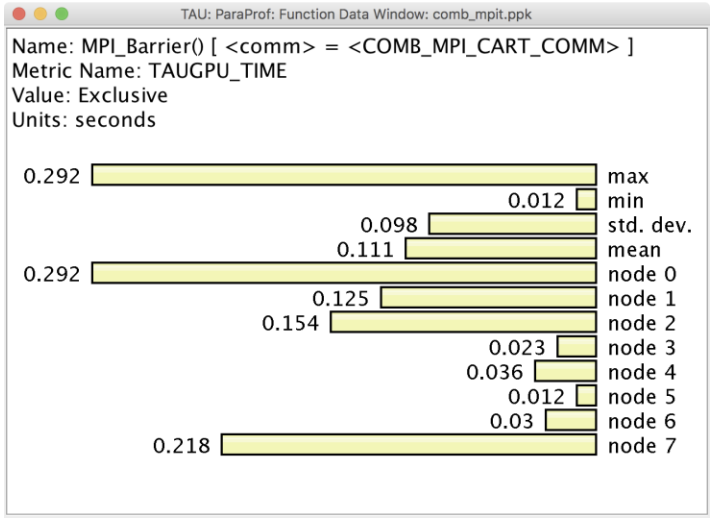



% export TAU_TRACK_MPI_T_PVARS=1
% export TAU_MPI_T_CVAR_METRICS=MPIR_CVAR_VBUF_POOL_SIZE
% export TAU_MPI_T_CVAR_VALUES=16
% mpirun -np 1024 *tau_exec -T mvapich2* ./a.out

# TAU: Extending Control Variables on a Per-Communicator Basis

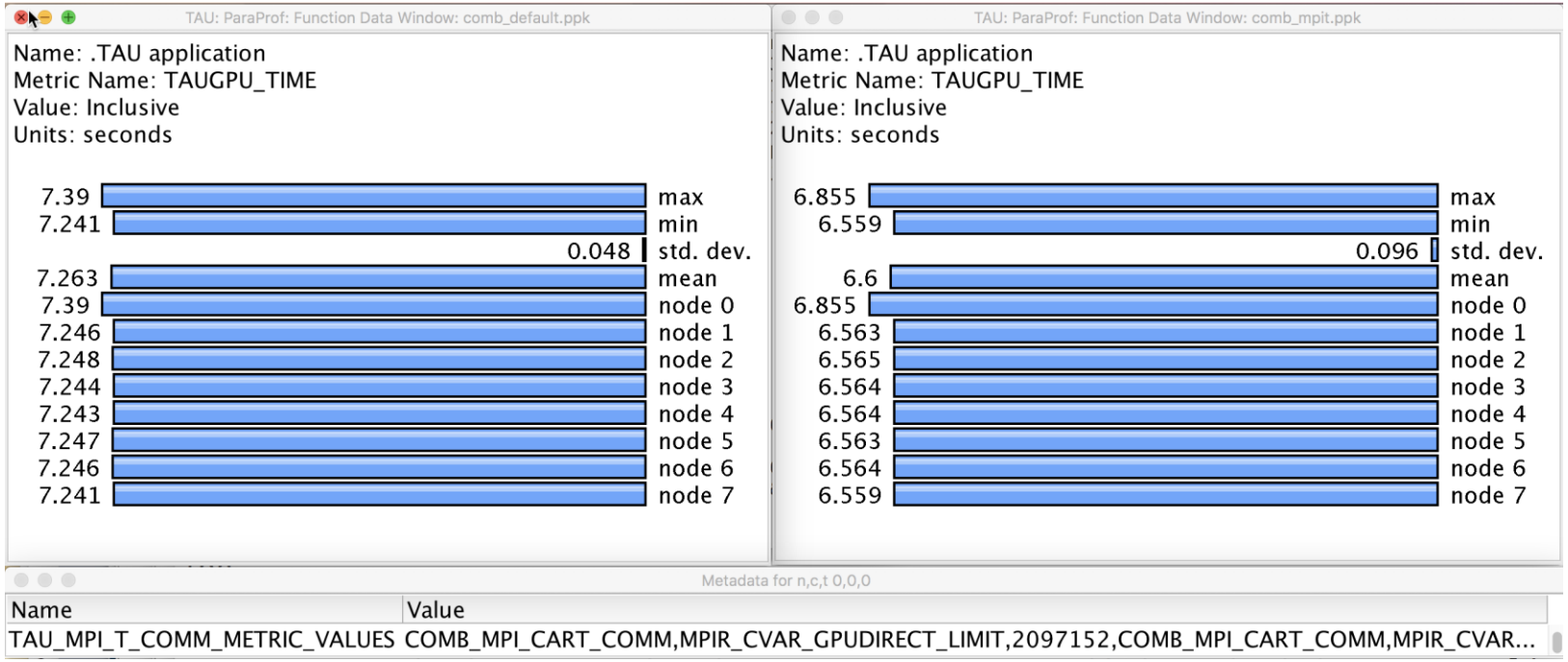
- Based on named communicators (MPI_Comm_set_name) in an application, TAU allows a user to specify triples to set MPI_T cvars for each communicator:
  Communicator name
  MPI_T CVAR name
  MPI_T CVAR value

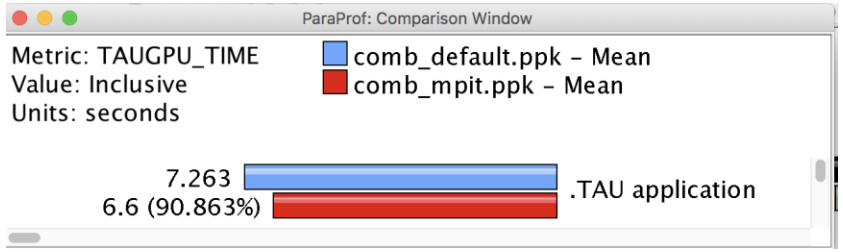
```
% ./configure –mpit –mpi –c++=mpicxx –cc=mpicc –fortran=mpif90 …
% make install
% export TAU_MPI_T_COMM_METRIC_VALUES=<comm, cvar,
    value>,…
% mpirun –np 64 tau_exec –T mvapich2,mpit    ./a.out
% paraprof
```



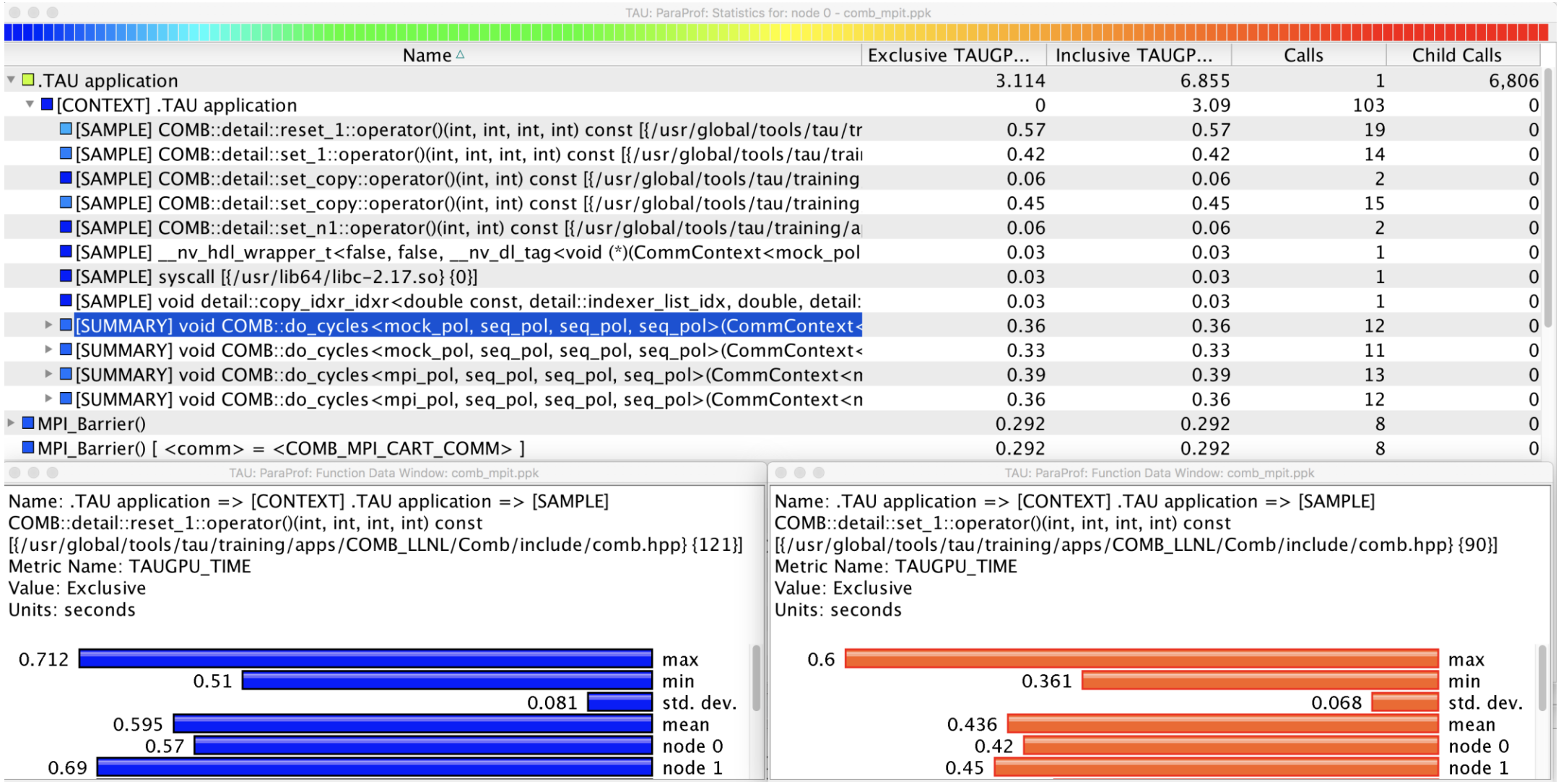
# COMB LLNL App MPI_T Tuning for COMB_MPI_CART_COMM

```
bash-4.2$
    TAU_MPI_T_COMM_METRIC_VALUES=COMB_MPI_CART_COMM,MPIR_CVAR_GPUDIRECT_LIMIT,2097152,COMB_MPI_CART_COMM,MPIR_CVAR_USE_GPUDIRECT_R
    ECEIVE_LIMIT,2097152,COMB_MPI_CART_COMM,MPIR_CVAR_CUDA_IPC_THRESHOLD,16384 MV2_USE_CUDA=1 mpirun -np 8 tau_exec -ebs -T
    mvapich2,mpit,cuda9,cupti,communicators,gnu -cupti ./comb -comm post_recv wait_all -comm post_send wait_all -comm wait_recv wait_all -comm wait_send wait_all 200_200_200
    -divide 2_2_2 -periodic 1_1_1 -ghost 1_1_1 -vars 3 -cycles 100 -comm cutoff 250 -omp_threads 1
Started rank 0 of 8
Node lassen710
Compiler COMB_COMPILER
Cuda compiler COMB_CUDA_COMPILER
GPU 0 visible undefined
Not built with openmp, ignoring -omp_threads 1.
Cart coords       0       0       0
Message policy cutoff 250
Post Recv using wait_all method
Post Send using wait_all method
Wait Recv using wait_all method
Wait Send using wait_all method
Num cycles       100
Num vars         3
ghost_widths      1       1       1
sizes           200     200     200
divisions         2       2       2
periodic          1       1       1
division map
map               0       0       0
map             100     100     100
map             200     200     200
Starting test memcpy seq dst Host src Host
Starting test Comm mock Mesh seq Host Buffers seq Host seq Host
Starting test Comm mpi Mesh seq Host Buffers seq Host seq Host
```



# COMB Profile



# CVARs Exposed by MVAPICH2

Metadata for n,c,t 0,0,0

| Name | Value |
|---|---|
| MPI Processor Name | lassen710 |
| MPIR_CVAR_CUDA_IPC_THRESHOLD | 16384 |
| MPIR_CVAR_GPUDIRECT_LIMIT | 2097152 |
| MPIR_CVAR_USE_GPUDIRECT_RECEIVE_LIMIT | 2097152 |
| MPI_T CVAR: MPIR_CVAR_ABORT_ON_LEAKED_HANDLES | If true, MPI will call MPI_Abort at MPI_Finalize if any MPI object handles ha... |
| MPI_T CVAR: MPIR_CVAR_ALLGATHERV_PIPELINE_MSG_SIZE | The smallest message size that will be used for the pipelined, large-mes... |
| MPI_T CVAR: MPIR_CVAR_ALLGATHER_COLLECTIVE_ALGORITHM | This CVAR selects proper collective algorithm for allgather operation. |
| MPI_T CVAR: MPIR_CVAR_ALLGATHER_LONG_MSG_SIZE | For MPI_Allgather and MPI_Allgatherv, the long message algorithm will be... |
| MPI_T CVAR: MPIR_CVAR_ALLGATHER_SHORT_MSG_SIZE | For MPI_Allgather and MPI_Allgatherv, the short message algorithm will b... |
| MPI_T CVAR: MPIR_CVAR_ALLREDUCE_COLLECTIVE_ALGORITHM | This CVAR selects proper collective algorithm for allreduce operation. |
| MPI_T CVAR: MPIR_CVAR_ALLREDUCE_SHORT_MSG_SIZE | the short message algorithm will be used if the send buffer size is <= th... |
| MPI_T CVAR: MPIR_CVAR_ALLTOALLV_COLLECTIVE_ALGORITHM | This CVAR selects proper collective algorithm for alltoallv operation. |
| MPI_T CVAR: MPIR_CVAR_ALLTOALL_COLLECTIVE_ALGORITHM | This CVAR selects proper collective algorithm for alltoall operation. |
| MPI_T CVAR: MPIR_CVAR_ALLTOALL_MEDIUM_MSG_SIZE | the medium message algorithm will be used if the per-destination messa... |
| MPI_T CVAR: MPIR_CVAR_ALLTOALL_SHORT_MSG_SIZE | the short message algorithm will be used if the per-destination message... |
| MPI_T CVAR: MPIR_CVAR_ALLTOALL_THROTTLE | max no. of irecvs/isends posted at a time in some alltoall algorithms. Set... |
| MPI_T CVAR: MPIR_CVAR_ASYNC_PROGRESS | If set to true, MPICH will initiate an additional thread to make asynchrono... |
| MPI_T CVAR: MPIR_CVAR_BCAST_COLLECTIVE_ALGORITHM | This CVAR selects proper collective algorithm for broadcast operation. |
| MPI_T CVAR: MPIR_CVAR_BCAST_LONG_MSG_SIZE | Let's define short messages as messages with size < MPIR_CVAR_BCAST_... |
| MPI_T CVAR: MPIR_CVAR_BCAST_MIN_PROCS | Let's define short messages as messages with size < MPIR_CVAR_BCAST_... |
| MPI_T CVAR: MPIR_CVAR_BCAST_SHORT_MSG_SIZE | Let's define short messages as messages with size < MPIR_CVAR_BCAST_... |
| MPI_T CVAR: MPIR_CVAR_CH3_EAGER_MAX_MSG_SIZE | This cvar controls the message size at which CH3 switches from eager to... |
| MPI_T CVAR: MPIR_CVAR_CH3_ENABLE_HCOLL | If true, enable HCOLL collectives. |
| MPI_T CVAR: MPIR_CVAR_CH3_INTERFACE_HOSTNAME | If non-NULL, this cvar specifies the IP address that other processes shoul... |
| MPI_T CVAR: MPIR_CVAR_CH3_NOLOCAL | If true, force all processes to operate as though all processes are located... |
| MPI_T CVAR: MPIR_CVAR_CH3_ODD_EVEN_CLIQUES | If true, odd procs on a node are seen as local to each other, and even pr... |
| MPI_T CVAR: MPIR_CVAR_CH3_PORT_RANGE | The MPIR_CVAR_CH3_PORT_RANGE environment variable allows you to s... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_ACTIVE_REQ_THRESHOLD | Threshold of number of active requests to trigger blocking waiting in op... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_DELAY_ISSUING_FOR_PIGGYBACKING | Specify if delay issuing of RMA operations for piggybacking LOCK/UNLOC... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_OP_GLOBAL_POOL_SIZE | Size of the Global RMA operations pool (in number of operations) that st... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_OP_PIGGYBACK_LOCK_DATA_SIZE | Specify the threshold of data size of a RMA operation which can be piggy... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_OP_WIN_POOL_SIZE | Size of the window-private RMA operations pool (in number of operation... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_POKE_PROGRESS_REQ_THRESHOLD | Threshold at which the RMA implementation attempts to complete reque... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_SCALABLE_FENCE_PROCESS_NUM | Specify the threshold of switching the algorithm used in FENCE from the ... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_SLOTS_SIZE | Number of RMA slots during window creation. Each slot contains a linked... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_TARGET_GLOBAL_POOL_SIZE | Size of the Global RMA targets pool (in number of targets) that stores inf... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_TARGET_LOCK_DATA_BYTES | Size (in bytes) of available lock data this window can provided. If current ... |
| MPI_T CVAR: MPIR_CVAR_CH3_RMA_TARGET_LOCK_ENTRY_WIN_POOL_SIZE | Size of the window-private RMA lock entries pool (in number of lock entr... |

# Path Aware Profiling in TAU and MVAPICH2

- To identify the path taken by an MPI message:
  - GPU memory to GPU memory
  - Unique send and receive path ids captured
- Configure TAU with -PROFILEPATHS:
- Partition the time in MPI pt-to-pt operations:

  MPI_Send and MPI_Recv

  Parameter based profiling identifies paths
- Path captured as metadata in TAU profiles

  PVARs based on CUPTI counters

  MVAPICH2 exports PVARs to TAU with MPI_T

Metadata for n,c,t 0,0,0

| Name | Value |
|---|---|
| TAU_PROFILE | on |
| TAU_PROFILE_FORMAT | profile |
| TAU_RECV_PATH_ID_\|_0 | gpu1-gpu0 |
| TAU_RECV_PATH_ID_\|_1 | gpu2-gpu0 |
| TAU_RECV_PATH_ID_\|_10 | internodelink-nic |
| TAU_RECV_PATH_ID_\|_2 | gpu3-gpu0 |
| TAU_RECV_PATH_ID_\|_3 | gpu2-gpu1 |
| TAU_RECV_PATH_ID_\|_4 | gpu3-gpu1 |
| TAU_RECV_PATH_ID_\|_5 | gpu3-gpu2 |
| TAU_RECV_PATH_ID_\|_6 | cpu-gpu0 |
| TAU_RECV_PATH_ID_\|_7 | cpu-gpu1 |
| TAU_RECV_PATH_ID_\|_8 | cpu-gpu2 |
| TAU_RECV_PATH_ID_\|_9 | cpu-gpu3 |
| TAU_RECYCLE_THREADS | off |
| TAU_REGION_ADDRESSES | off |
| TAU_SAMPLING | off |
| TAU_SEND_PATH_ID_\|_0 | gpu0-gpu1 |
| TAU_SEND_PATH_ID_\|_1 | gpu0-gpu2 |
| TAU_SEND_PATH_ID_\|_10 | nic-internodelink |
| TAU_SEND_PATH_ID_\|_2 | gpu0-gpu3 |
| TAU_SEND_PATH_ID_\|_3 | gpu1-gpu2 |
| TAU_SEND_PATH_ID_\|_4 | gpu1-gpu3 |
| TAU_SEND_PATH_ID_\|_5 | gpu2-gpu3 |
| TAU_SEND_PATH_ID_\|_6 | gpu0-cpu |
| TAU_SEND_PATH_ID_\|_7 | gpu1-cpu |
| TAU_SEND_PATH_ID_\|_8 | gpu2-cpu |
| TAU_SEND_PATH_ID_\|_9 | gpu3-cpu |

# Path Aware Profiling in TAU and MVAPICH2

Available for download in TAU v2.29.1

TAU: ParaProf: Statistics for: node 0 - path_3ranks.ppk

| Name | Exclusive ... ▽ | Inclusive ... | Calls | Child ... |
|---|---|---|---|---|
| main [{/g/g24/shende1/mpit/path_test_3ranks.c} {61,0}] | 40.332 | 42.472 | 1 | 12 |
| MPI_Init() | 0.86 | 0.86 | 1 | 0 |
| MPI_Send() | 0.746 | 0.746 | 4 | 2 |
| MPI_Send() [ <message send path id> = <1006> ] | 0.617 | 0.617 | 2 | 0 |
| init_accel [{/g/g24/shende1/mpit/path_test_3ranks.c} {42,0}] | 0.263 | 0.263 | 1 | 1 |
| MPI_Finalize() | 0.254 | 0.254 | 1 | 0 |
| MPI_Send() [ <message send path id> = <100600> ] | 0.129 | 0.129 | 2 | 0 |
| .TAU application | 0.033 | 42.505 | 1 | 1 |
| MPI_Barrier() | 0.017 | 0.017 | 3 | 0 |
| get_local_rank [{/g/g24/shende1/mpit/path_test_3ranks.c} {26,0}] | 0 | 0 | 1 | 0 |
| MPI_Get_processor_name() | 0 | 0 | 2 | 0 |
| MPI_Comm_rank() | 0 | 0 | 1 | 0 |
| MPI_Comm_size() | 0 | 0 | 1 | 0 |

# Identifying Collective Wait States

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

Metric Name: TIME
Sorted By: Exclusive
Units: seconds

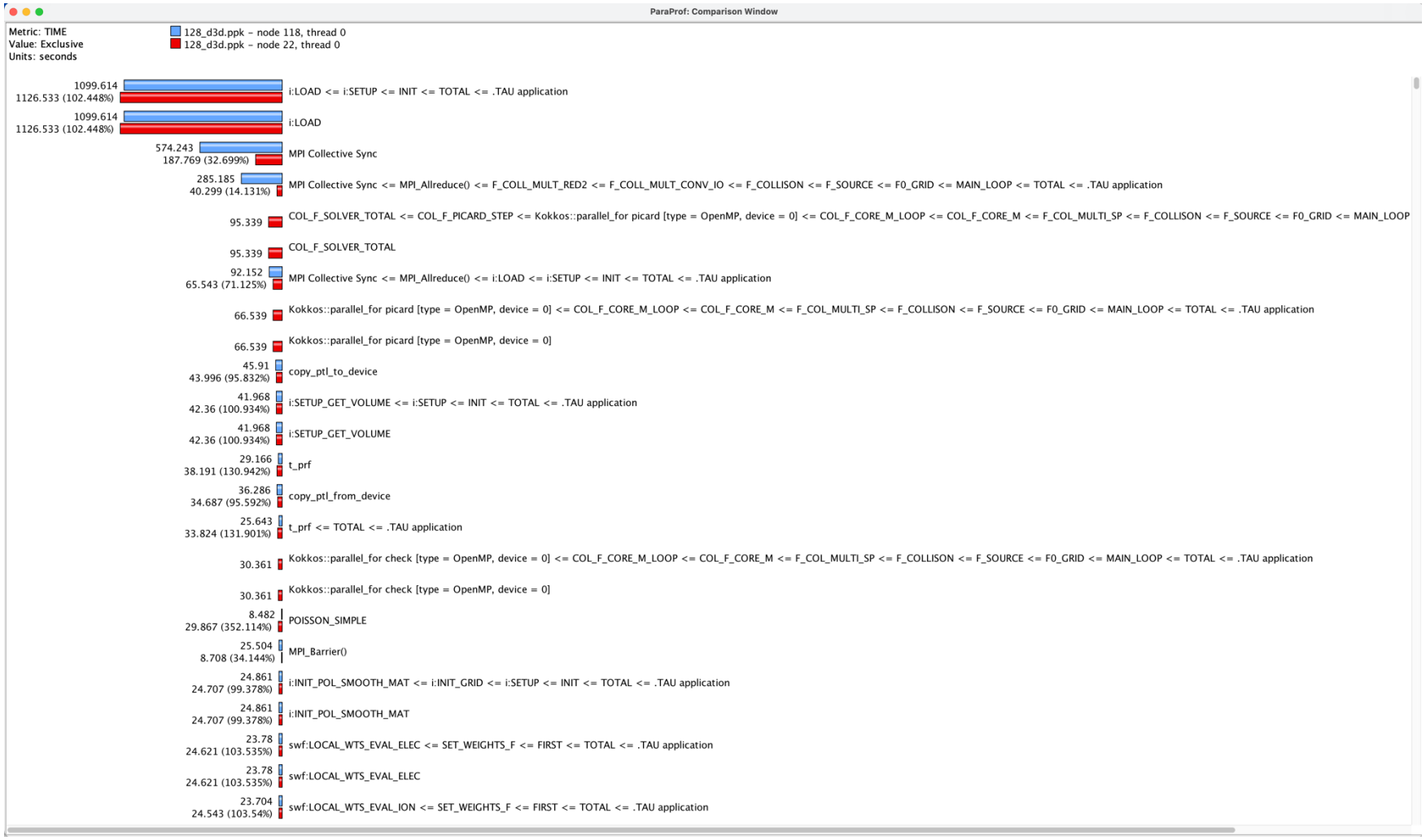
| | Exclusive | Inclusive | Calls/Tot.Calls | Name[id] |
|---|---|---|---|---|
| | 1099.614 | 1191.772 | 1/1 | i:SETUP |
| --> | 1099.614 | 1191.772 | 1 | i:LOAD |
| | 0.006 | 92.158 | 3/9543 | MPI_Allreduce() |
| | 9.8E-4 | 9.8E-4 | 11/15177 | MPI_Gatherv() |
| | 1.448 | 1.448 | 43/15177 | MPI_Gather() |
| | 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 comparison window

**Comparing Rank 118 with 22.**

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

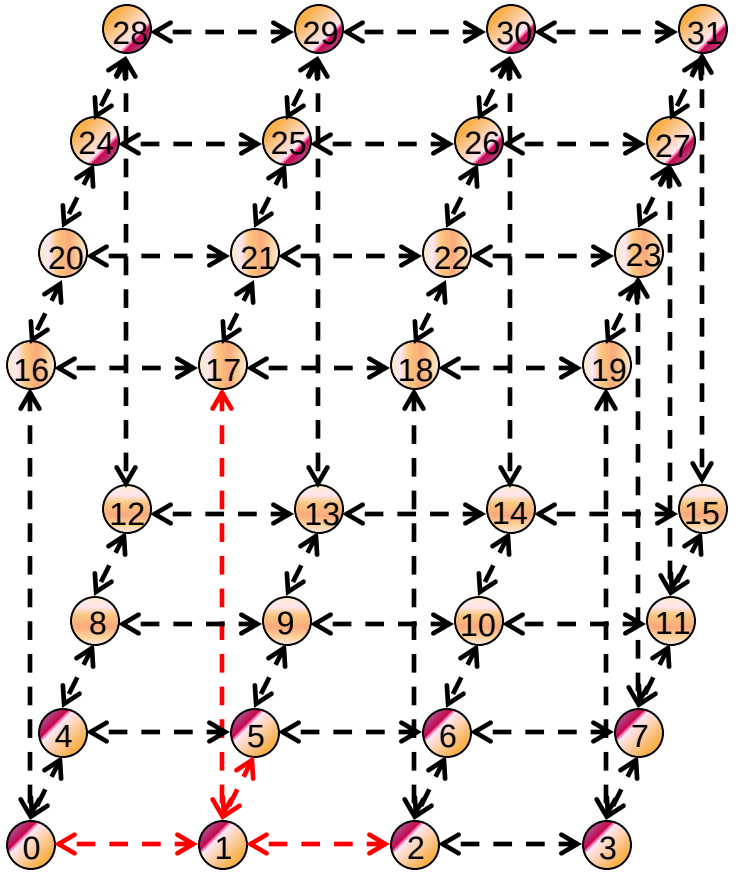


# Driving Example (3D Stencil)

## 3D Stencil benchmark

- Each process talks to at most six neighbors
- Two in each Cartesian dimension
  - X-right, X-left
  - Y-right, Y-left
  - Z-right, Z-left
- Repeat same communication pattern for multiple iterations

3D Stencil communication pattern for a 32 process job scheduled on 4 nodes




Process on Node 1

Process on Node 2

Process on Node 3

Process on Node 4

# Case Study: 3D Stencil

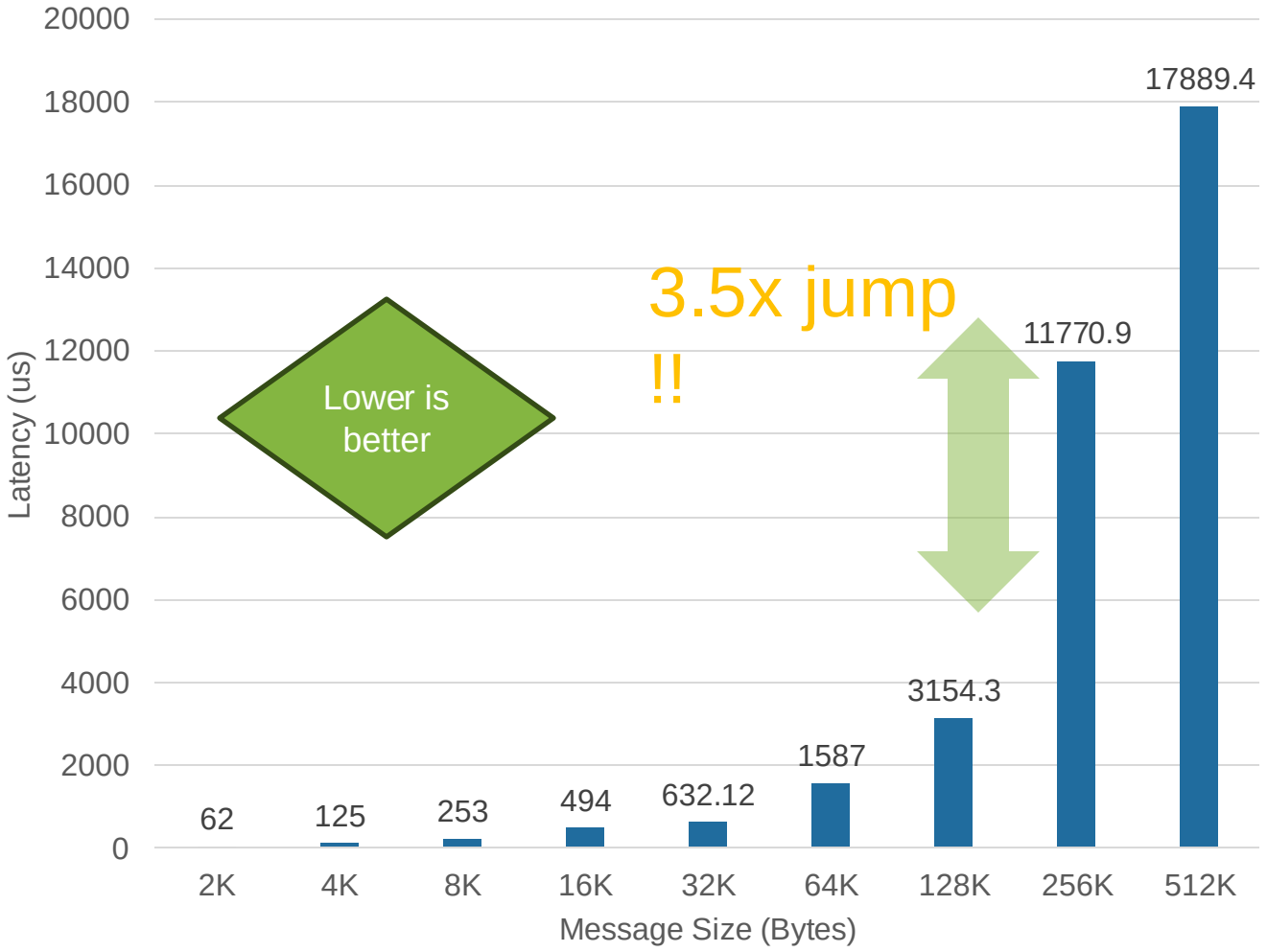
- Platform:
  - Broadcom RoCEv2 Thor Adapter
  - 64 Nodes x 2 x AMD EPYC 7713 64-Core Processor
- Application:
  - 3D Stencil HPC Benchmark
  - Dataset: 3000k-atoms dataset
- Raw run lines:

  - MVAPICH2-2.3.7-Broadcom

```
mpirun_rsh -np $NP -ppn $PPN ./3Dstencil_overlap 8 8 8
1000
```

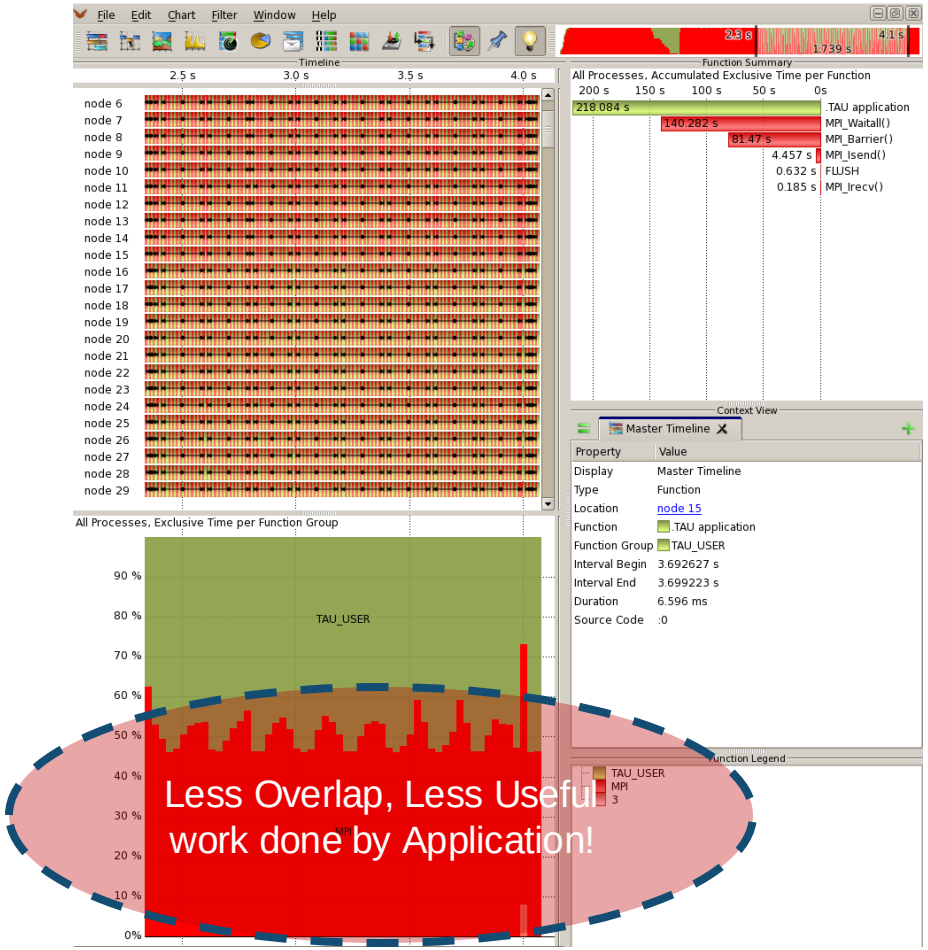
# 3D Stencil: Unoptimized Version

- Execution time tests on 2 Nodes x 128 PPN (512 ranks)
- We are measuring the latency
  - Lower is better
- Degradation observed at 256K message
- This is the unoptimized MVAPICH2-2.3.7 version
- Need to use TAU to see
  - what MPI calls are causing the degradation
  - What is the dominant communication pattern



# Understanding Basic Performance Trends with TAU-based Profiling

**Default**



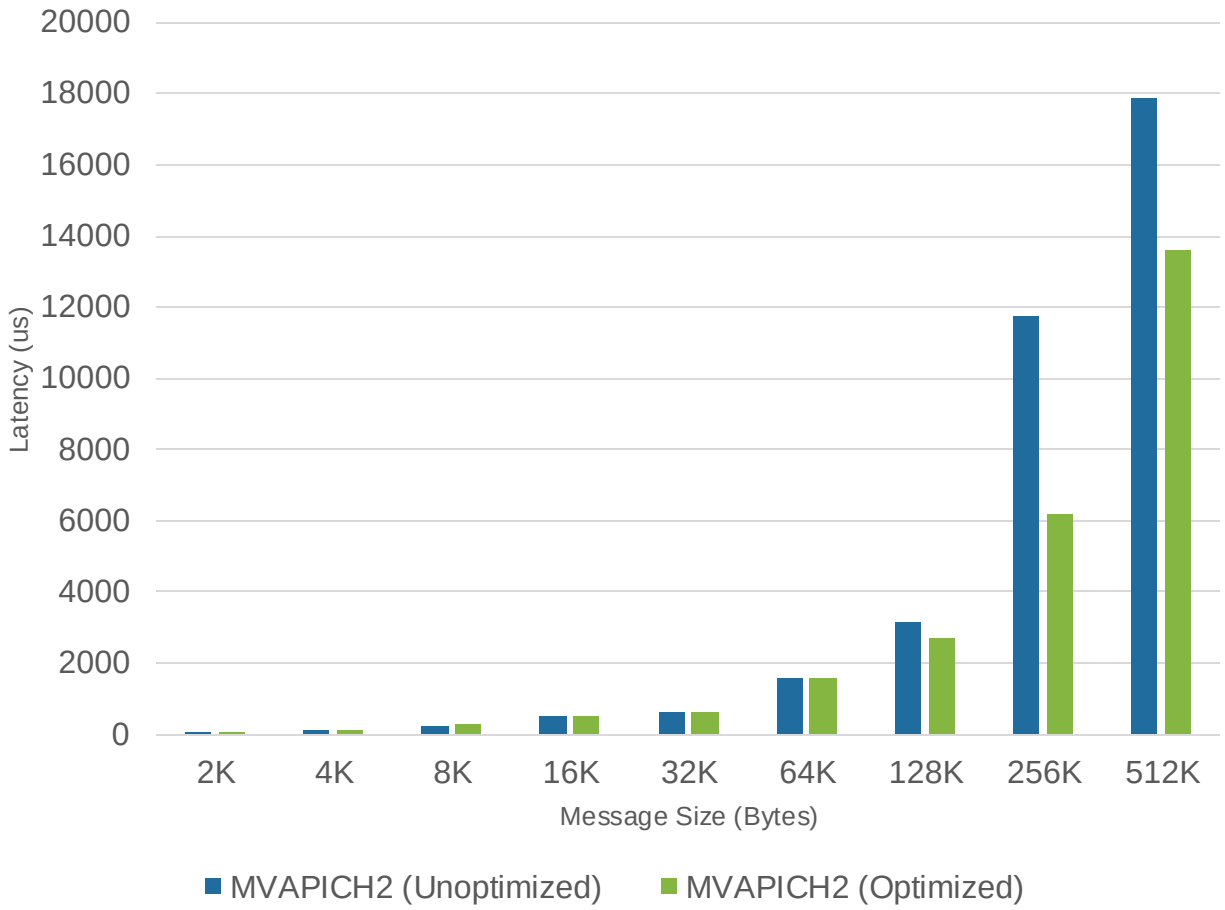

Visualized in Vampir [TUD Germany]

# 3D Stencil: Performance Engineered with TAU

- Diagnosis: more time is spent in inter-node pt-to-pt Rendezvous communication
- Solution: Use pt-to-pt eager communication
- Gains:
  - 2x reduction in latency
- Update the following parameter for the 3D Stencil runs

```
MV2_IBA_EAGER_THRESHOLD = 524288
```

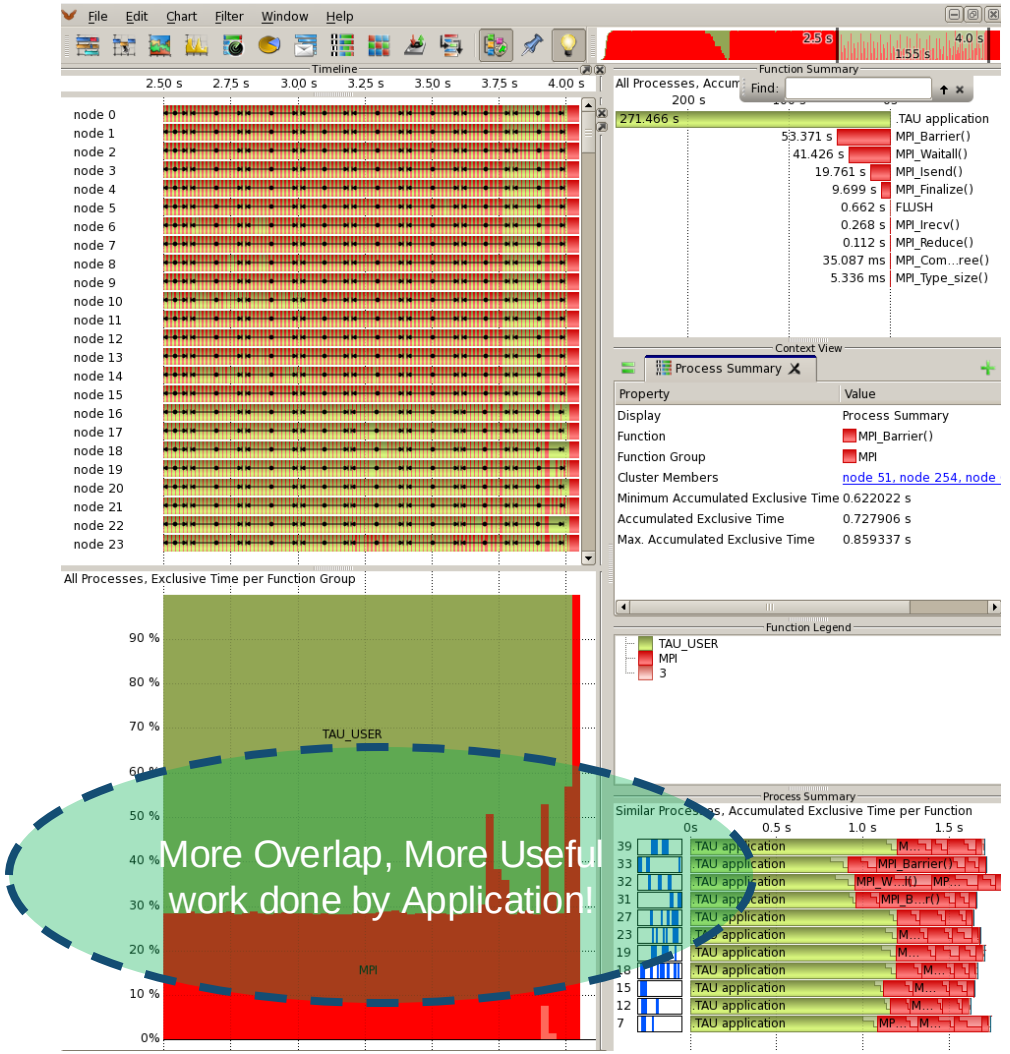
this will enable inter-node eager communication until the specified message size*




*For more details check user-guide:
https://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-userguide.html#:~:text=for%20the%20job.-,12.5,-MV2_IBA_EAGER_THRESHOLD

# Introspecting Impact of Eager Threshold on 3D Stencil Benchmark

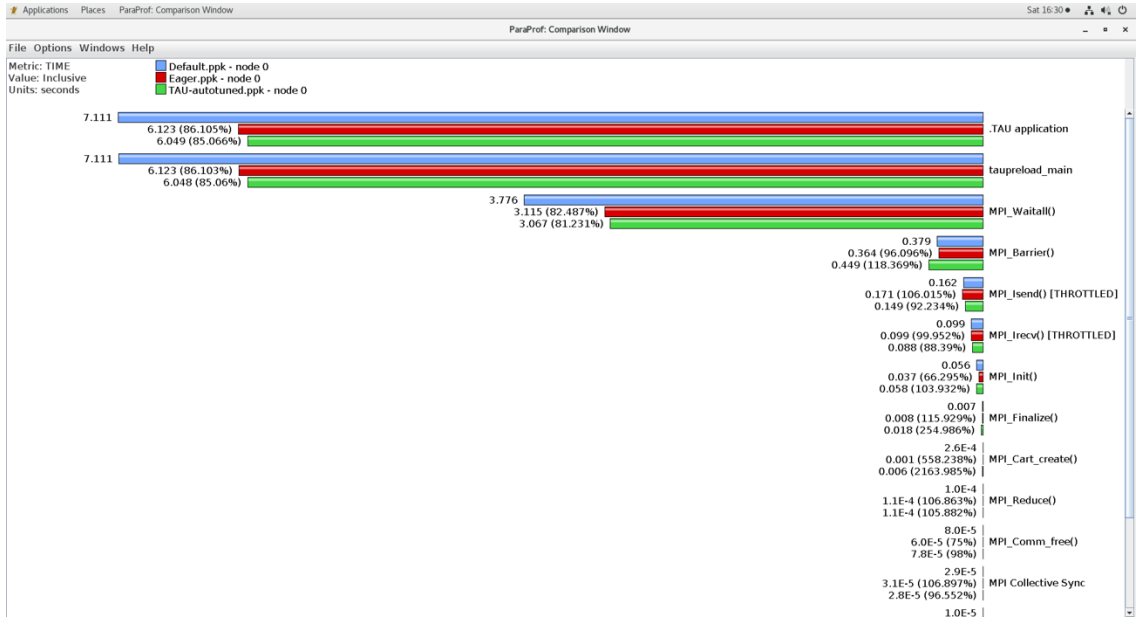
**Optimized**




Visualized in Vampir [TUD Germany]

# 3Dstencil on AWS

cd ~/SRC/demo/3Dstencil
./run.sh
ls  *.ppk
% paraprof *.ppk &
Right click “Add Thread to Comparison Window”
while clicking on Node 0 in each of the three trials
Options -> Select Metric -> Inclusive



# Usage Scenarios with MVAPICH2

- TAU measures the high water mark of total memory usage (TAU_TRACK_MEMORY_FOOTPRINT=1), finds that it is at 98% of available memory, and queries MVAPICH2 to find out how much memory it is using. Based on the number of pools allocated and used, it requests it to reduce the number of VBUF pools and controls the size of the these pools using the MPI-T interface. The total memory memory footprint of the application reduces.
- TAU tracks the message sizes of messages (TAU_COMM_MATRIX=1), detects excessive time spent in MPI_Wait and other synchronization operations. It compares the average message size with the eager threshold and sets the new eager threshold value to match the message size. This could be done offline by re-executing the application with the new CVAR setting for eager threshold or online.

# 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>) |

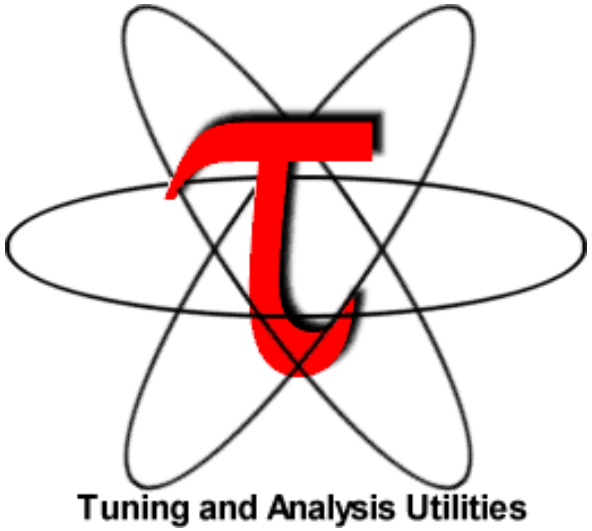
# Runtime Environment Variables

| 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 |

# Download TAU from U. Oregon



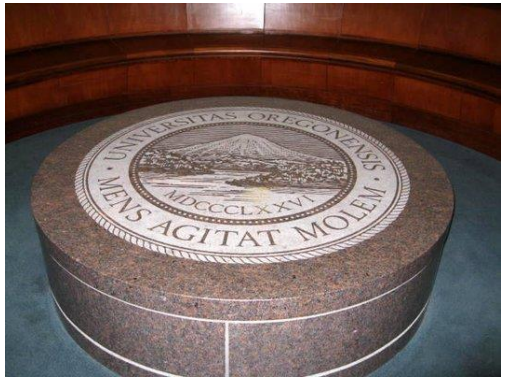

http://tau.uoregon.edu

https://e4s.io [TAU in Docker/Singularity containers]

for more information

Free download, open source, BSD license

# Performance Research Laboratory, University of Oregon, Eugene




www.uoregon.edu

# Support Acknowledgements


- 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, E4S Workshop
- NASA
- Intel, NVIDIA, AMD, IBM
- CEA, France
- 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