

Tuning and Analysis Utilities (TAU) Seminar

Sameer Shende

Director, Performance Research Laboratory, University of Oregon
President, ParaTools, Inc.

sameer@cs.uoregon.edu

<http://tau.uoregon.edu>

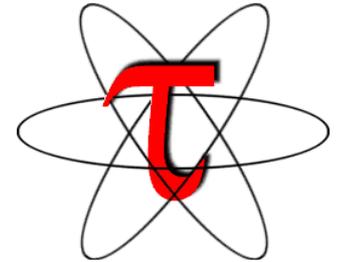
CERN, room 40/S2-D01 – Salle Dirac, Monday, October 14, 2019, 5pm CEST

<https://indico.cern.ch/event/845622>

Slides: <http://tau.uoregon.edu/CERN19.pdf>

TAU Performance System[®]

<http://tau.uoregon.edu>

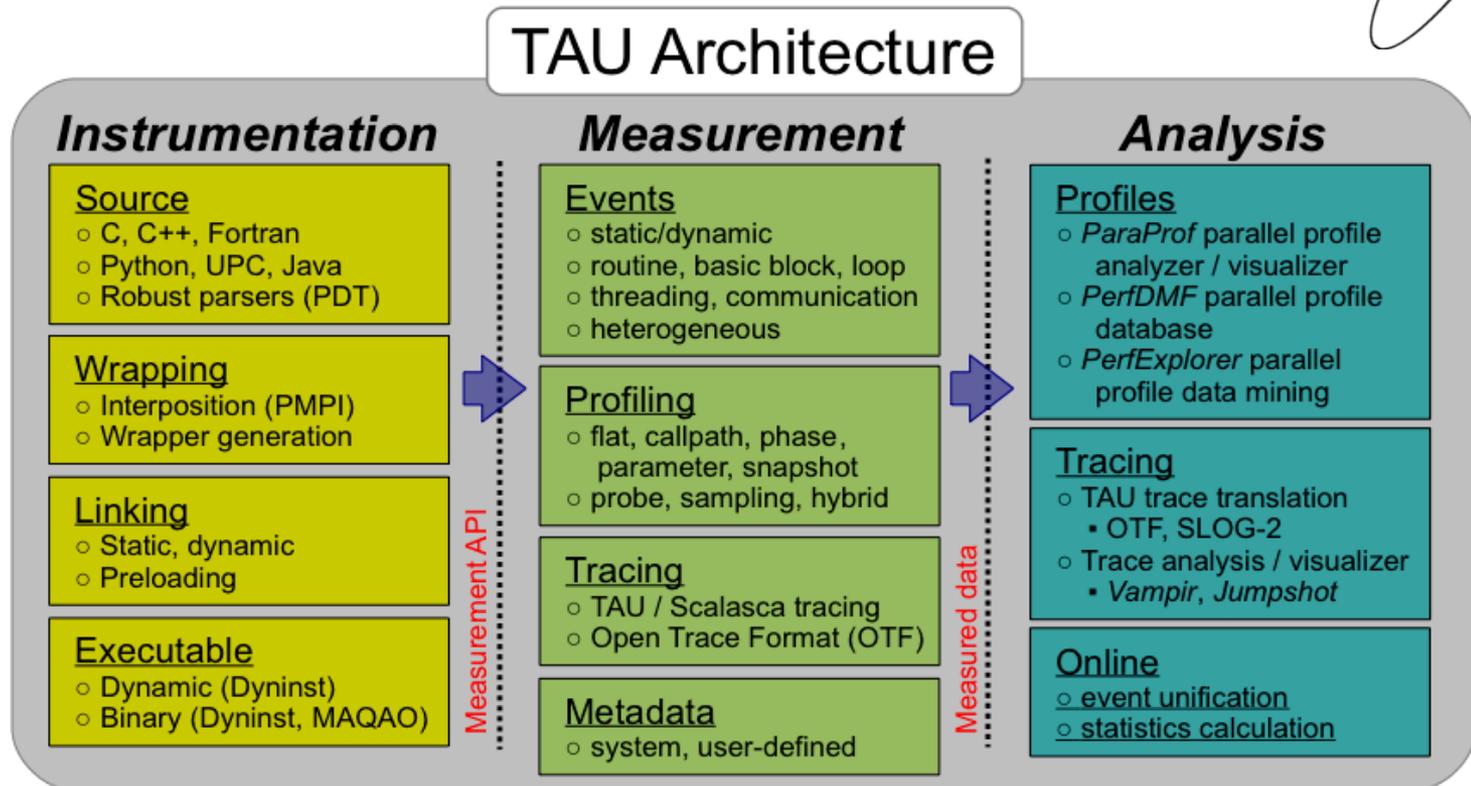
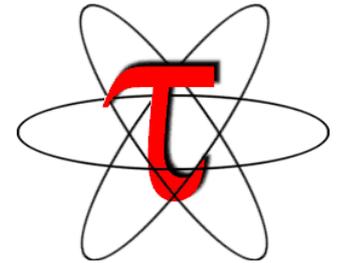


- **Tuning and Analysis Utilities (20+ year project)**
- **Comprehensive performance profiling and tracing**
 - Integrated, scalable, flexible, portable
 - Targets all parallel programming/execution paradigms
- **Integrated performance toolkit**
 - Instrumentation, measurement, analysis, visualization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

Understanding Application Performance using TAU

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

TAU Architecture and Workflow



Instrumentation

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

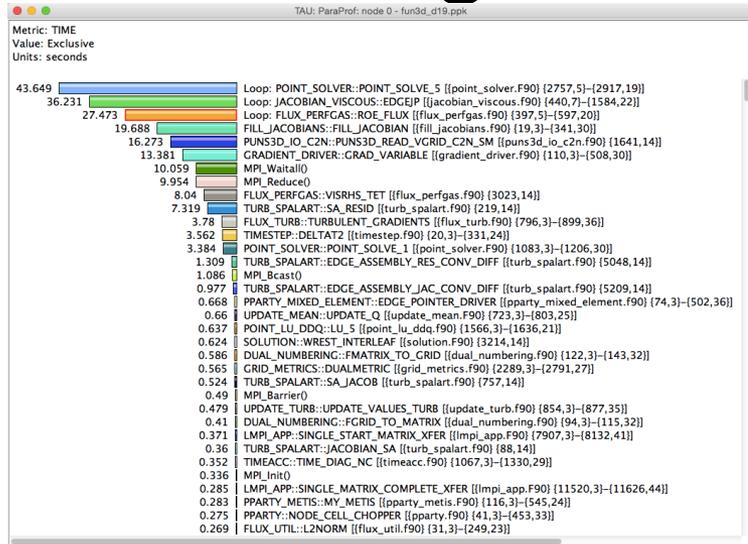
- Use system compiler to add a special flag to insert hooks at routine entry/exit.
- Requires recompiling using TAU compiler scripts (tau_cc.sh, tau_f90.sh...)

Runtime preloading of TAU's Dynamic Shared Object (DSO)

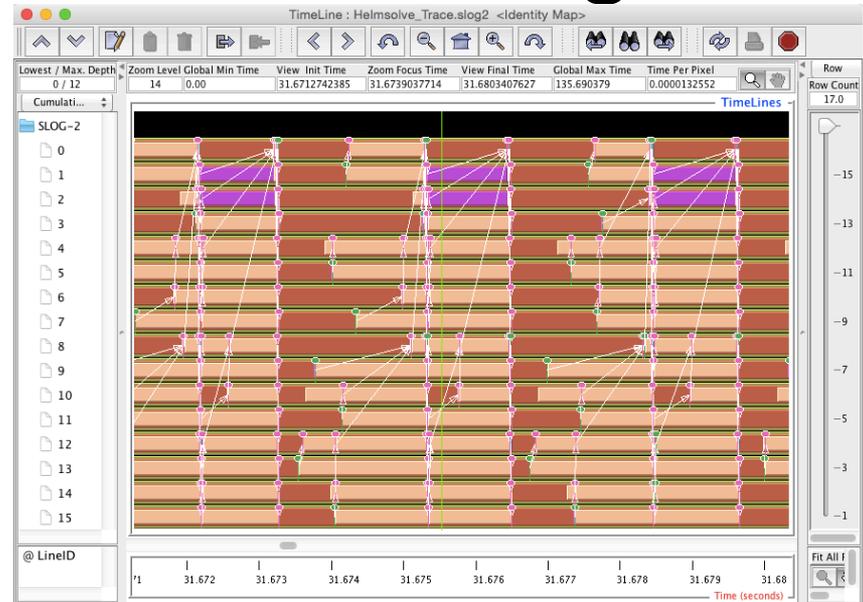
- No need to recompile code! Use **mpirun tau_exec ./app** with options.
- Requires dynamic executable (link using **-dynamic** on Cray systems).

Measurement: Profiling and Tracing

Profiling



Tracing



- Profiling and tracing

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

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

TAU: Quickstart Guide

Setup:

- `% module load tau`

Profiling with an un-instrumented application:

MPI: `% mpirun -np 64 tau_exec -ebs ./a.out`

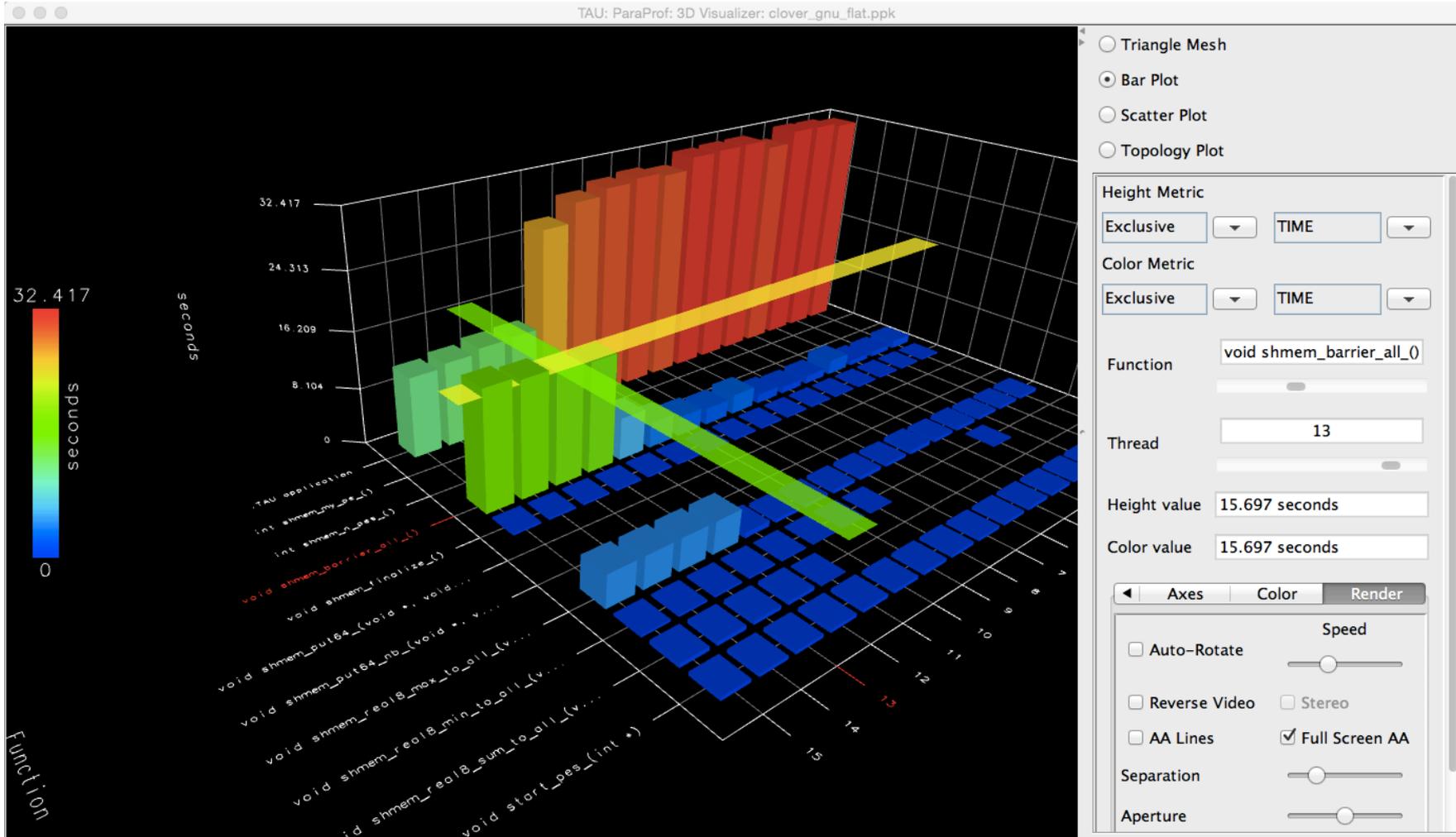
- MPI+OpenMP: `% export TAU_OMPT_SUPPORT_LEVEL=full;`
`% mpirun -np 64 tau_exec -T ompt,v5 -ompt ./a.out`
- Pthread: `% mpirun -np 64 tau_exec -T mpi,pthread -ebs ./a.out`
- Python+MPI+Sampling: `% mpirun -np 64 tau_python -ebs ./a.py`

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

Tracing:

- Vampir: MPI: `% export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2`
`% mpirun -np 64 tau_exec ./a.out; vampir traces.otf2 &`
 - Chrome: `% export TAU_TRACE=1; mpirun -np 64 tau_exec ./a.out`
`% tau_treemerge.pl;`
- `% tau_trace2json tau.trc tau.edf -chrome -ignoreatomic -o app.json`
Chrome browser: `chrome://tracing` (Load -> app.json)

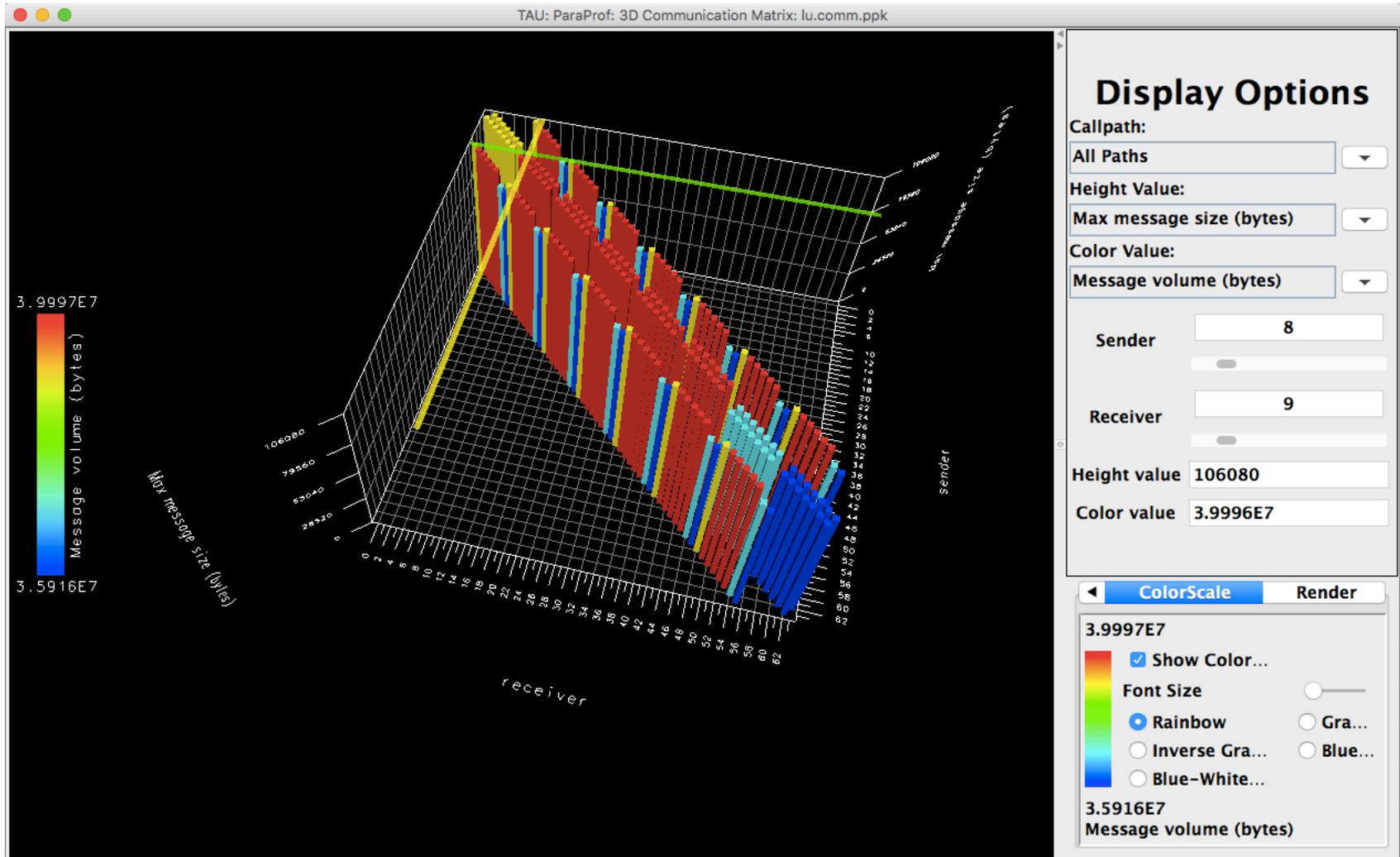
TAU – ParaProf 3D Visualization



% paraprof app.ppk

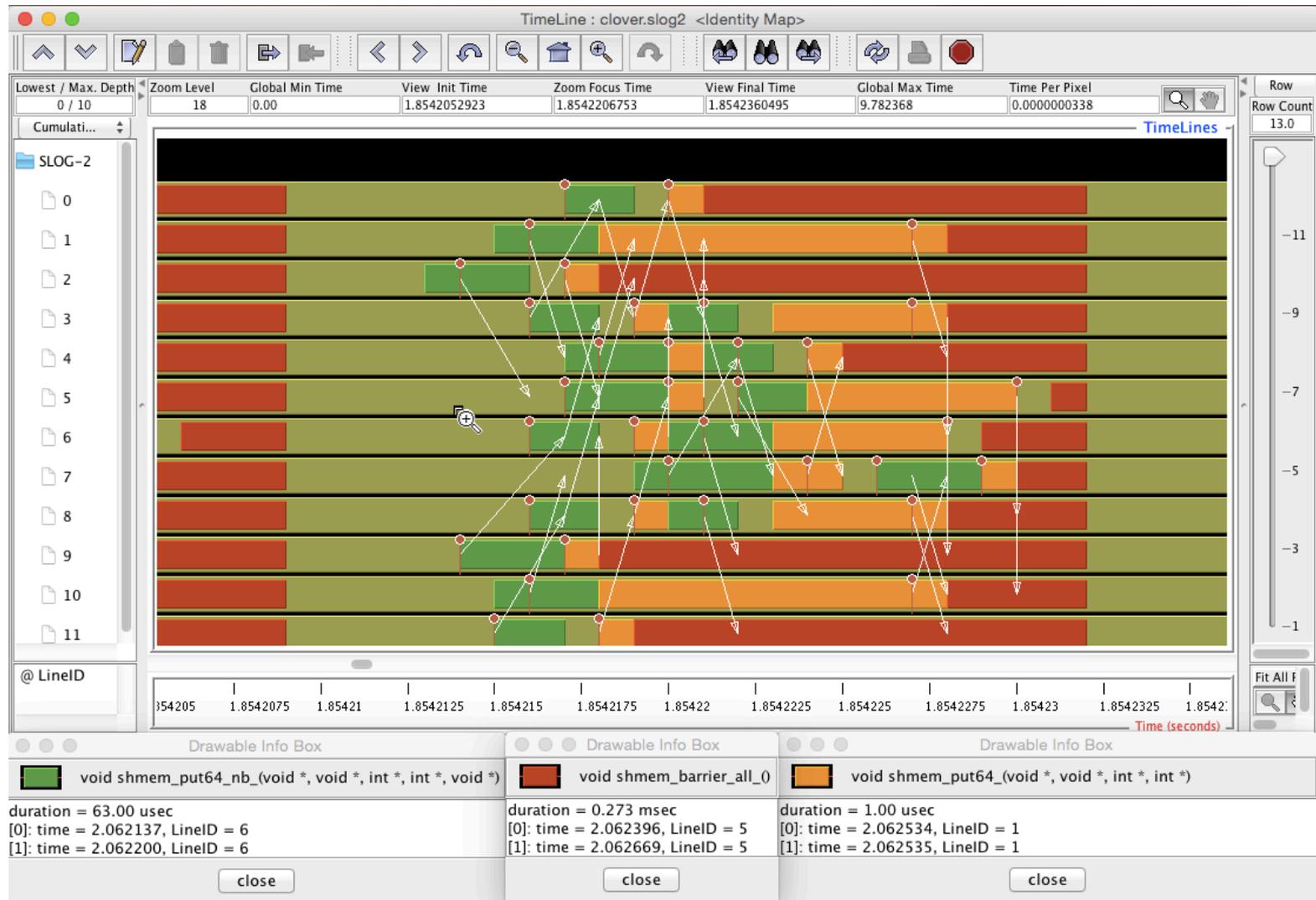
Windows -> 3D Visualization -> Bar Plot (right pane)

TAU – 3D Communication Window

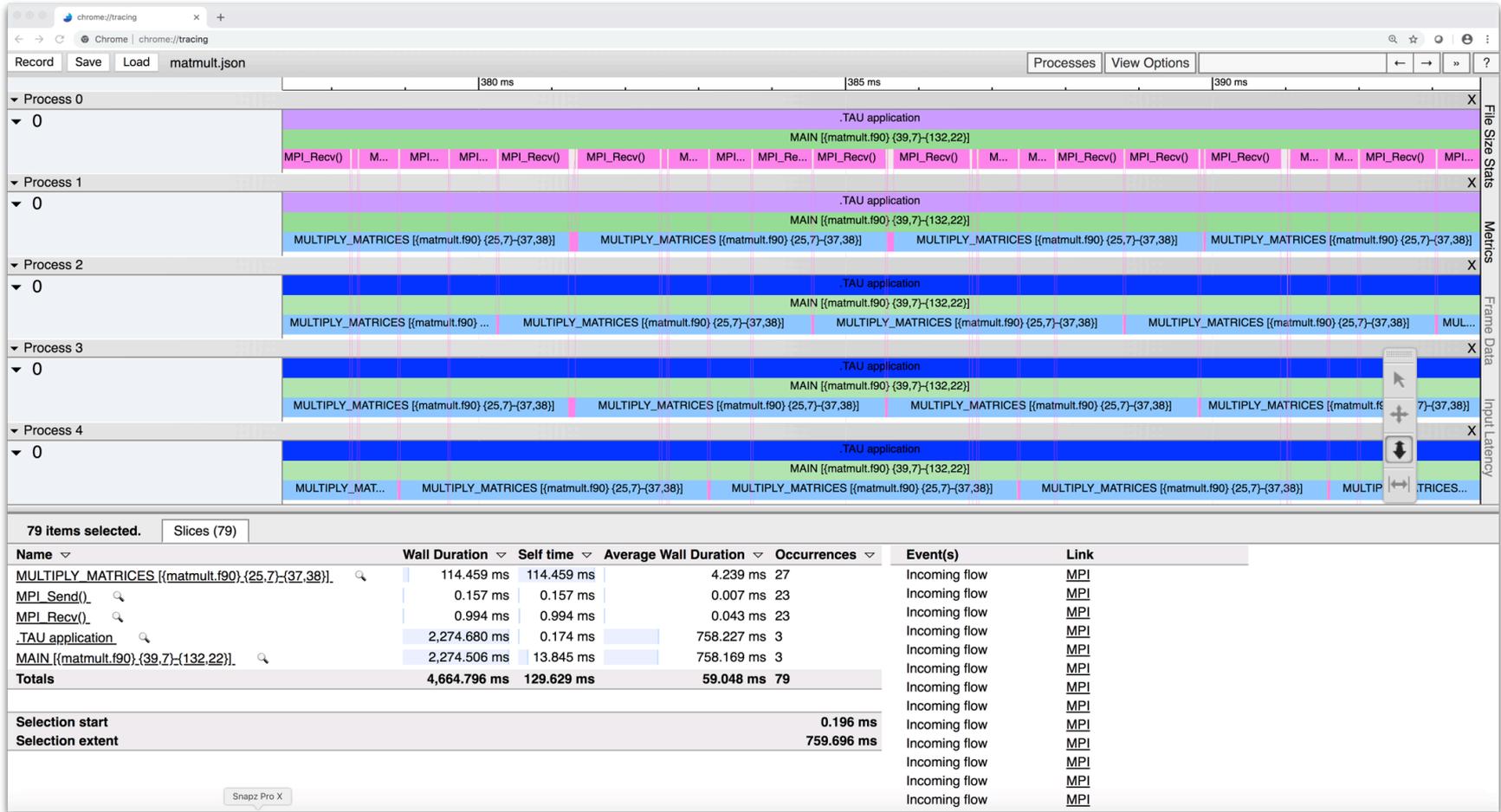


```
% export TAU_COMM_MATRIX=1; mpirun ... tau_exec ./a.out
% paraprof app.ppk; Windows -> 3D Communication Matrix
```

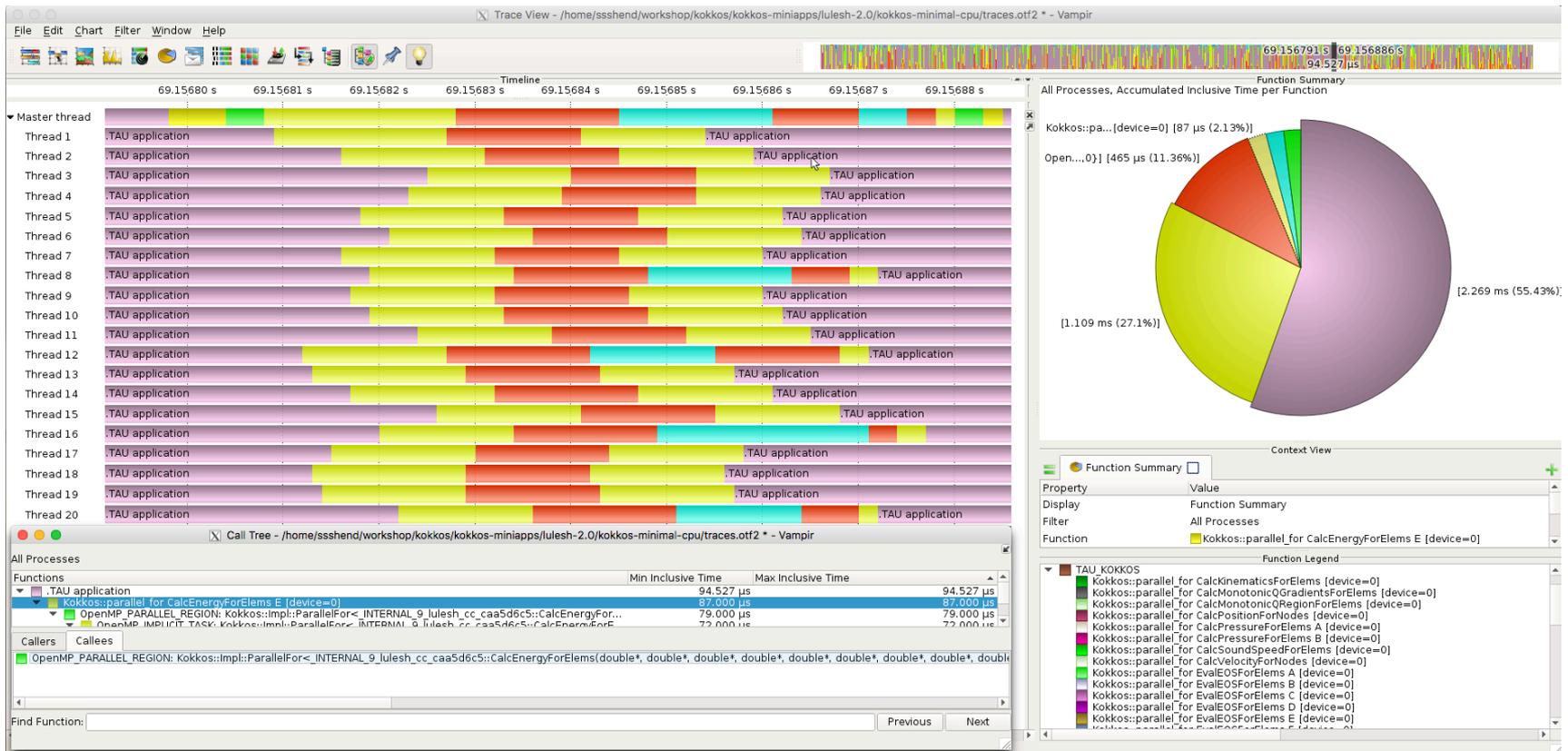
Jumpshot (ships with TAU)



Chrome Browser



Vampir [TU Dresden] Timeline: Kokkos



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

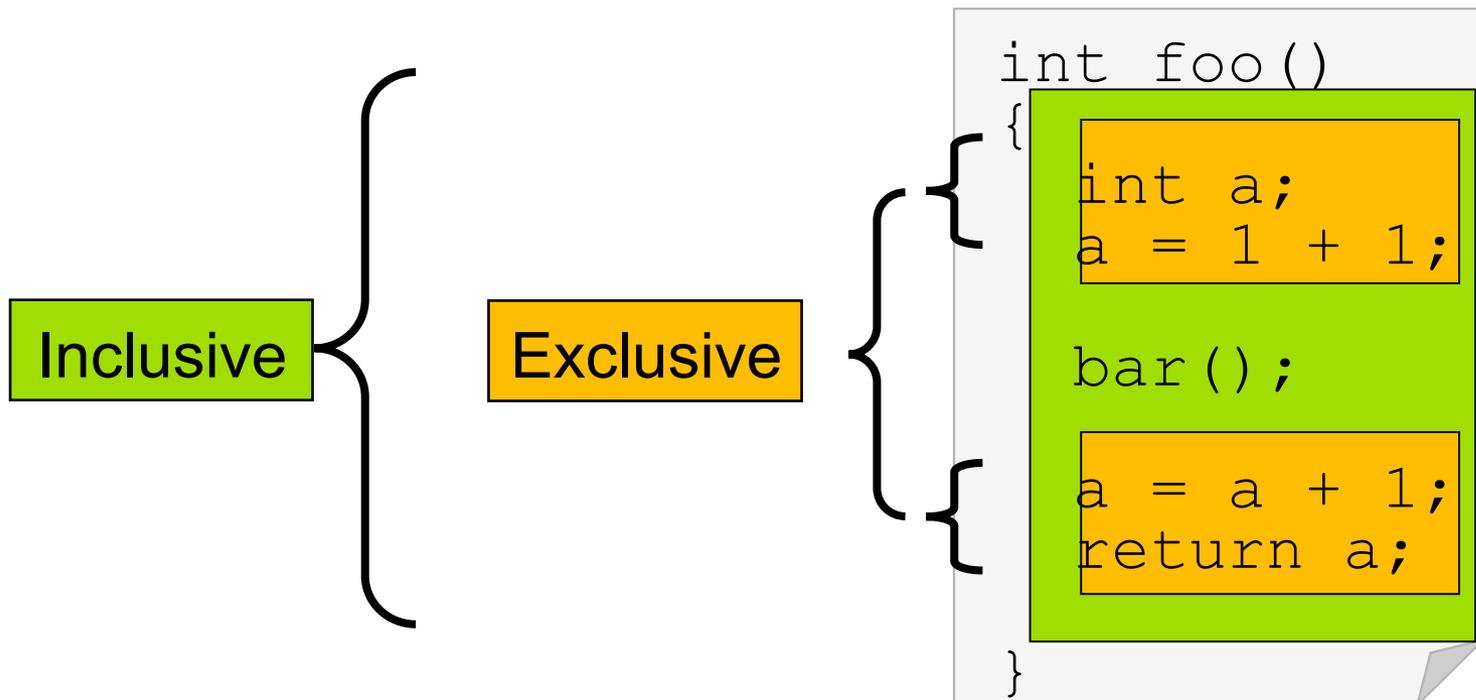
Inclusive vs. Exclusive values

- **Inclusive**

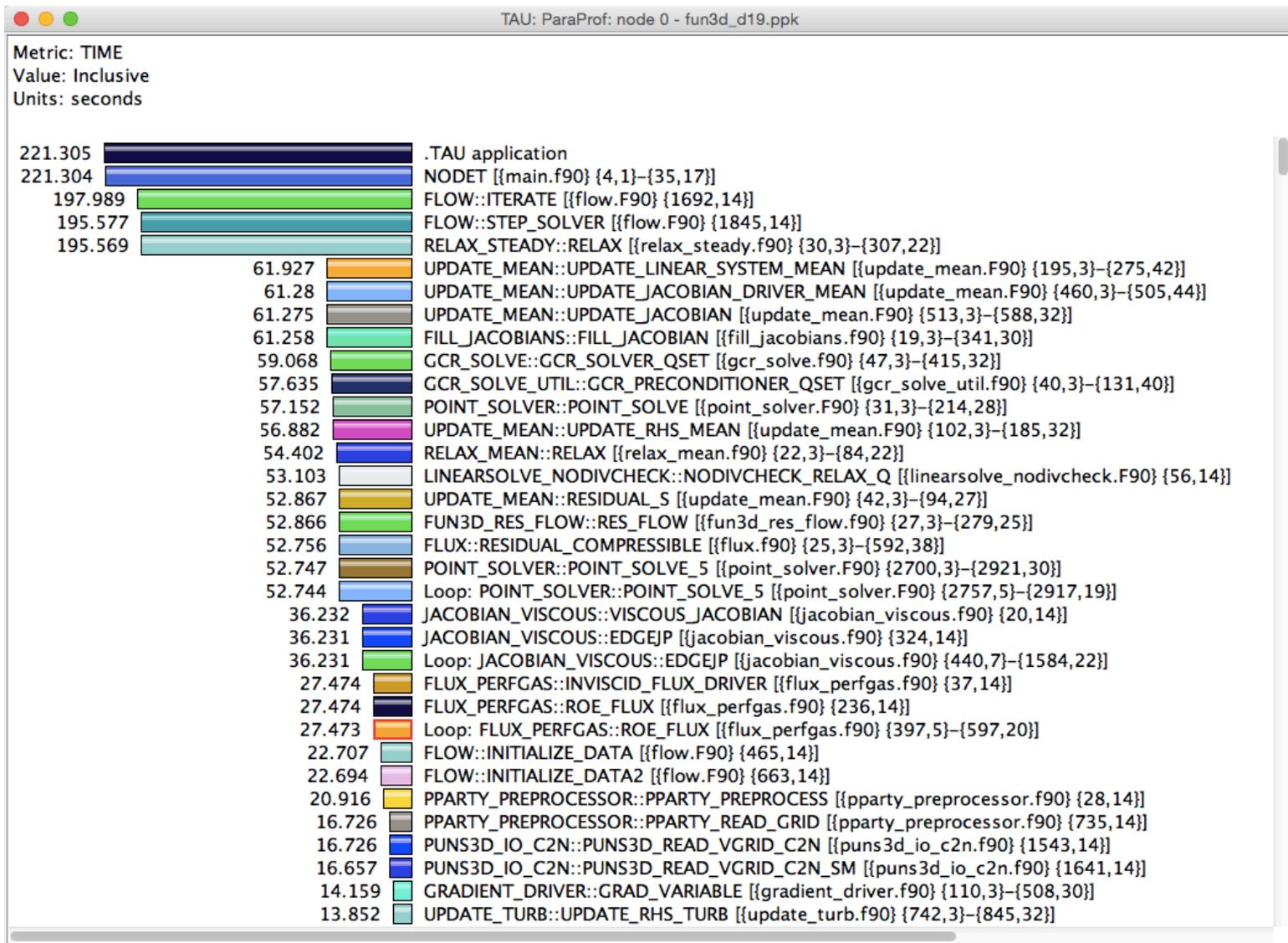
- Information of all sub-elements aggregated into single value

- **Exclusive**

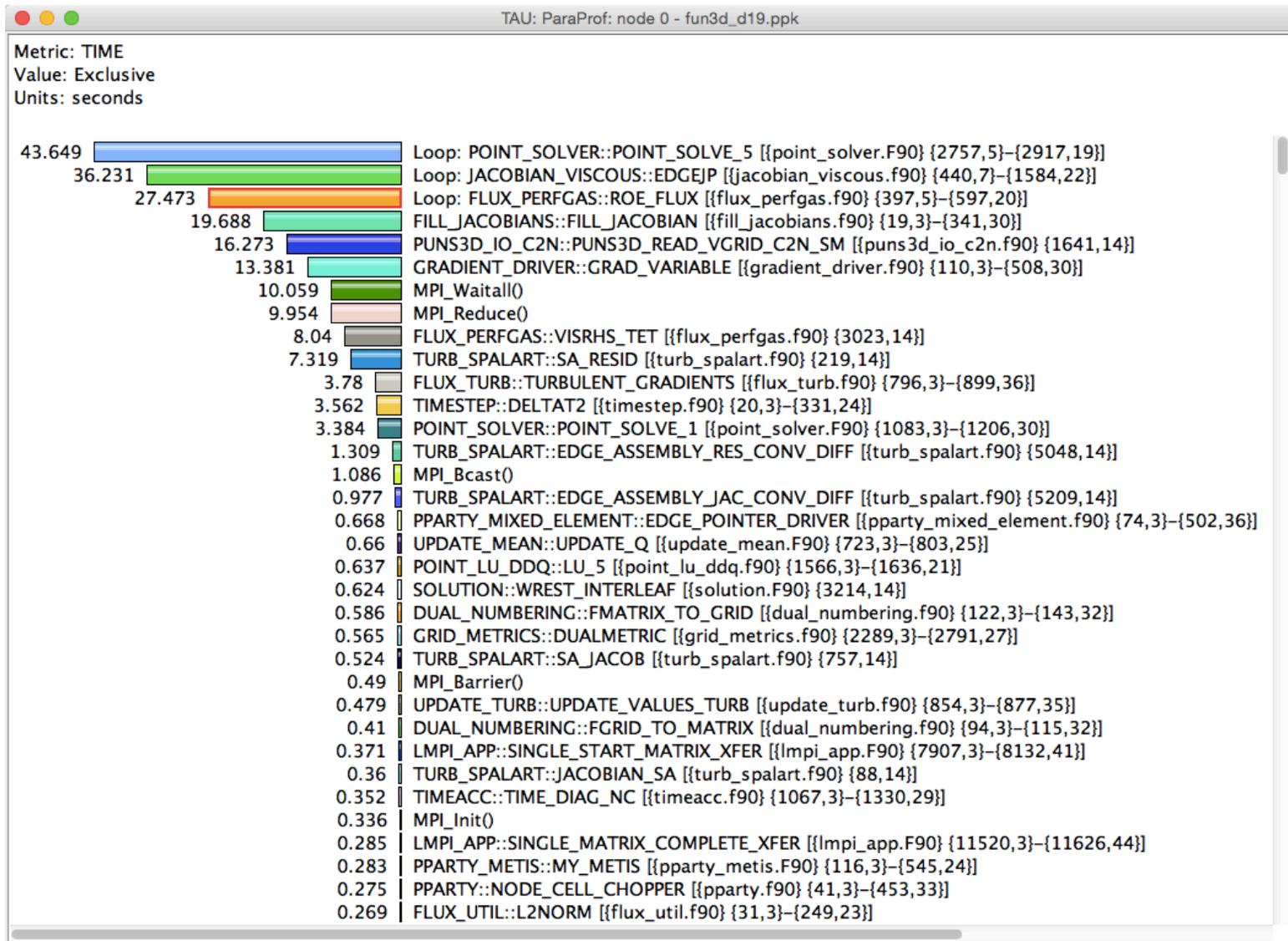
- Information cannot be subdivided further



Inclusive Measurements



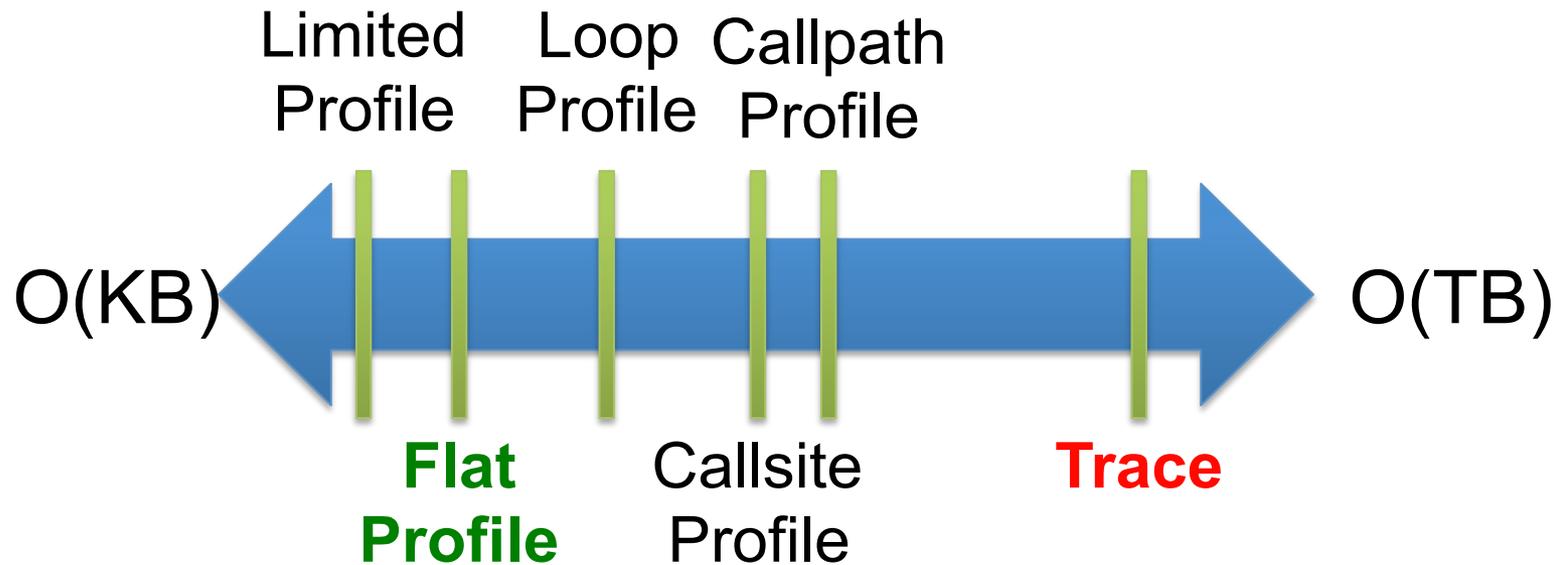
Exclusive Time



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_LOAD	0	Tracks system load on a node (e.g., also seen in tools like w, top, uptime)
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 (same as tau_exec -ebs).
TAU_EBS_RESOLUTION	Line	Setting to file function line will resolve addresses at the given resolution.
TAU_EBS_UNWIND	0	Setting to 1 enables callstack unwinding during 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_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>)

How much data do you want?



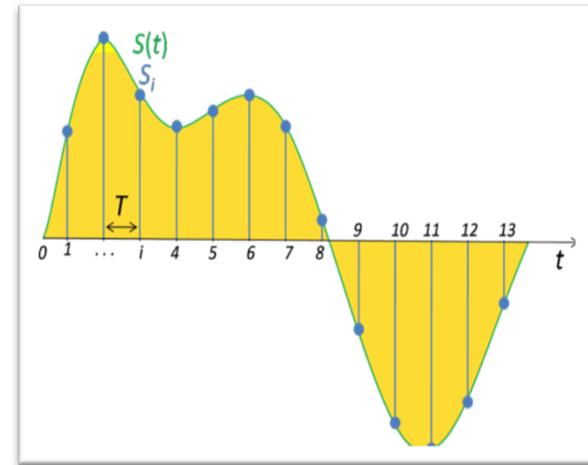
Performance Data Measurement

Direct via Probes

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

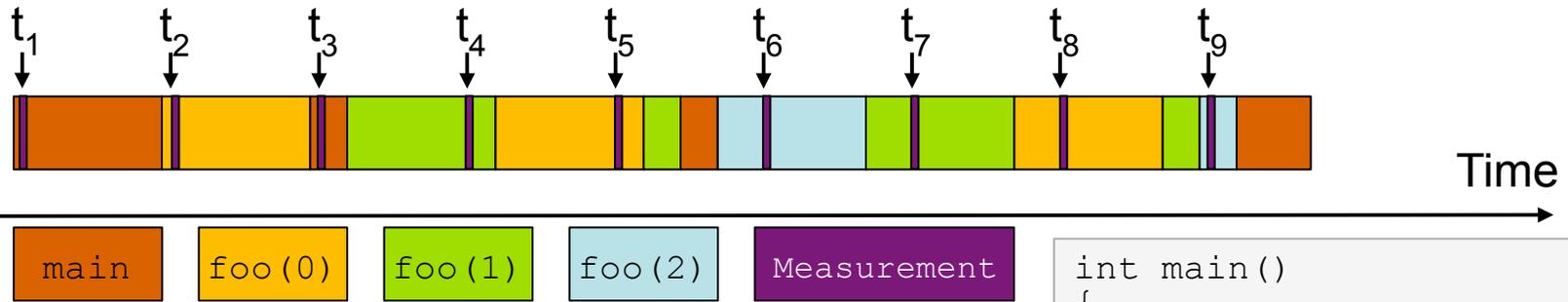
- Exact measurement
- Fine-grain control
- Calls inserted into code

Indirect via Sampling



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

Sampling



Running program is periodically interrupted to take measurement

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

Statistical inference of program behavior

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

Works with unmodified executables

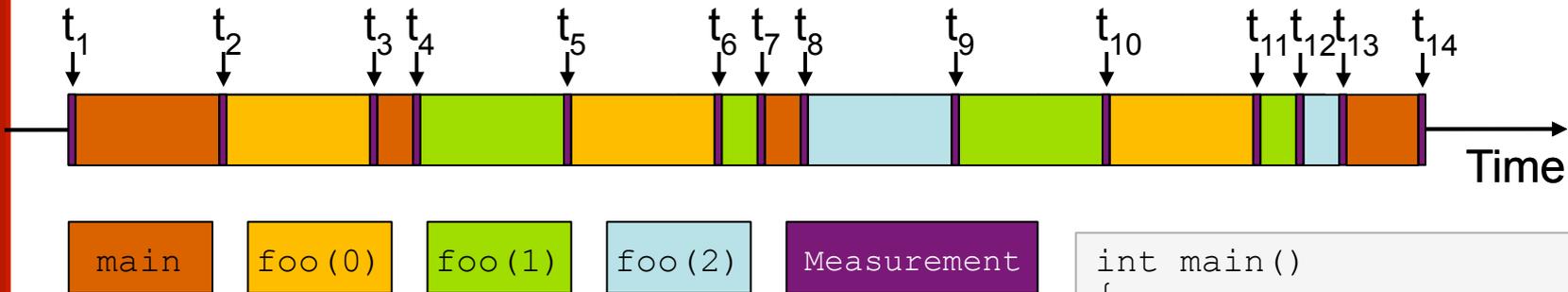
```
int main()
{
    int i;

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

    return 0;
}

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

Instrumentation



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

- Can be done in various ways

Advantage:

- Much more detailed information

Disadvantage:

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

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

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

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

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

Simplifying the use of TAU!

Uninstrumented code:

- `% make`
- `% mpirun -np 64 ./a.out`

With TAU using event based sampling (EBS):

- `% mpirun -np 64 tau_exec -ebs ./lu.B.64`
- `% paraprof` (GUI)
- `% pprof -a | more`

NOTE:

- Requires dynamic executables.
- Source code should be compiled with `-g` for access to symbol table.

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 -cupti ./a.out`
- `% mpirun -np 256 tau_exec -cupti -um ./a.out` (for Unified Memory)
- `% 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;`
`% export TAU_OMPT_RESOLVE_ADDRESS_EAGERLY=1`
- `% 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>`

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

TAU supports several measurement and thread options

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

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

To instrument source code automatically using PDT

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

(or module load tau...)

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

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

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

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

```
% mpif90 foo.f90      changes to
```

```
% tau_f90.sh foo.f90
```

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

```
% pprof (for text based profile display)
```

```
% paraprof (for GUI)
```

Choosing TAU_MAKEFILE

```
% module load tau
% echo $TAU
/usr/local/packages/tau_latest/x86_64/lib
% ls $TAU/Makefile*
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-mpi-pdt
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-papi-mpi-pdt-openmp-opari
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-papi-mpi-pthread-pdt
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-papi-mpi-pthread-python-pdt
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-papi-ompt-v5-mpi-pdt-openmp
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-papi-ompt-v5-pdt-openmp
/usr/local/packages/tau_latest/x86_64/lib/Makefile.tau-icpc-pdt
```

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

`$TAU/Makefile.tau-icpc-mpi-pdt`

- Supports MPI instrumentation & PDT for automatic source instrumentation

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-mpi-pdt
```

```
% tau_f90.sh matrix.f90 -o matrix
```

OR with build systems:

```
% make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
```

```
% cmake -DCMAKE_Fortran_COMPILER=tau_f90.sh
```

```
    -DCMAKE_C_COMPILER=tau_cc.sh -DCMAKE_CXX_COMPILER=tau_cxx.sh
```

```
% <ALLOCATE a NODE>
```

```
% mpirun -np 256 ./matrix
```

```
% paraprof
```

Configuration tags for tau_exec

```
% ./configure -pdt=<dir> -mpi -papi=<dir>; make install
```

Creates in \$TAU:

```
Makefile.tau-papi-mpi-pdt (Configuration parameters in stub makefile)  
shared-papi-mpi-pdt/libTAU.so
```

```
% ./configure -pdt=<dir> -mpi; make install creates
```

```
Makefile.tau-mpi-pdt  
shared-mpi-pdt/libTAU.so
```

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

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

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

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

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

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

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

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

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

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

Binary Rewriting Instrumentation

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

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

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

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

- **Pebil [SDSC]:**

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

- Execute the application to get measurement data:

```
% mpirun -np 4 ./a.inst
```

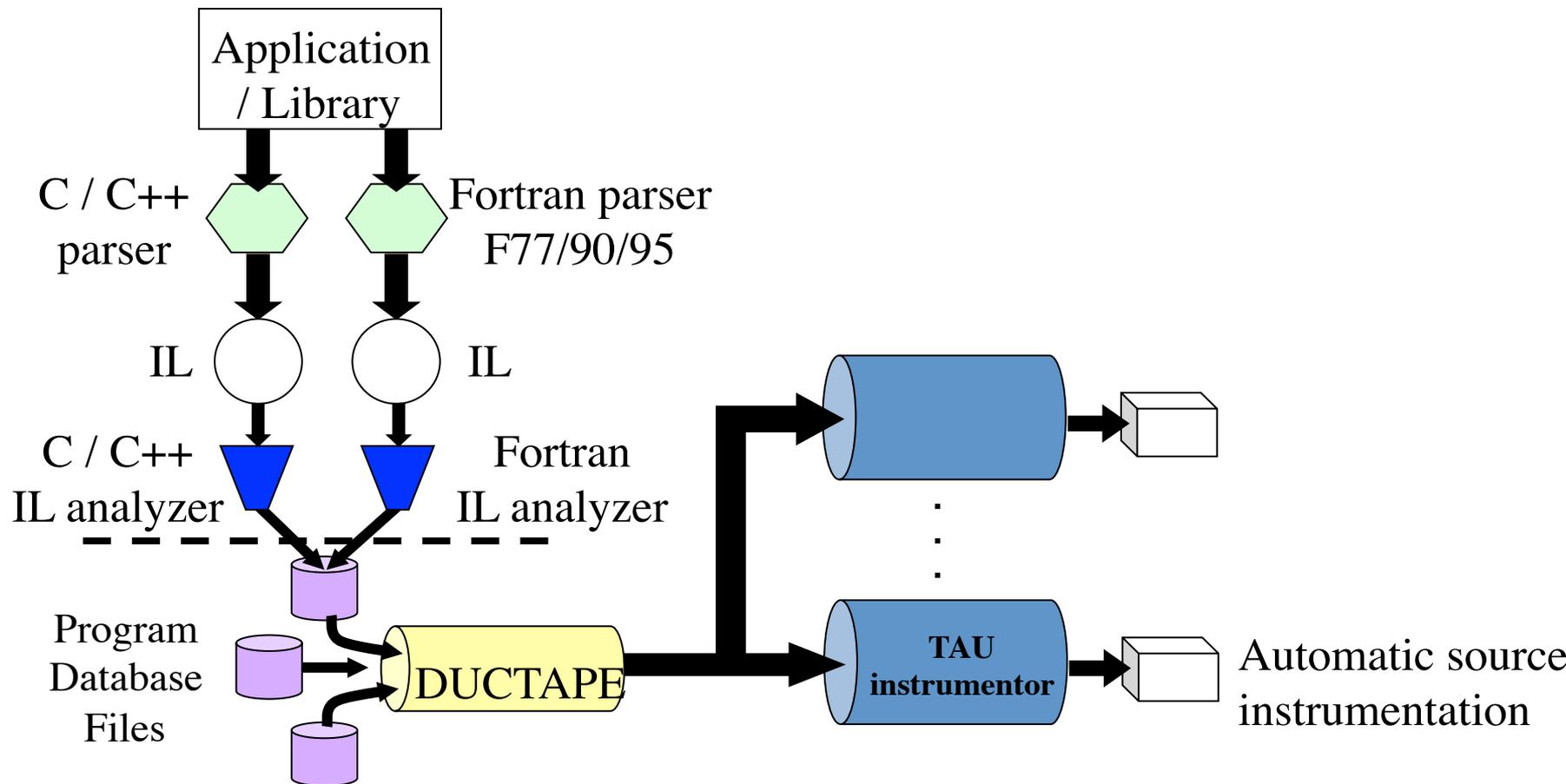
Binary Rewriting Instrumentation

```
% mpif90 -g matmult.f90 -o matmult
% tau_rewrite matmult matmult.i
```

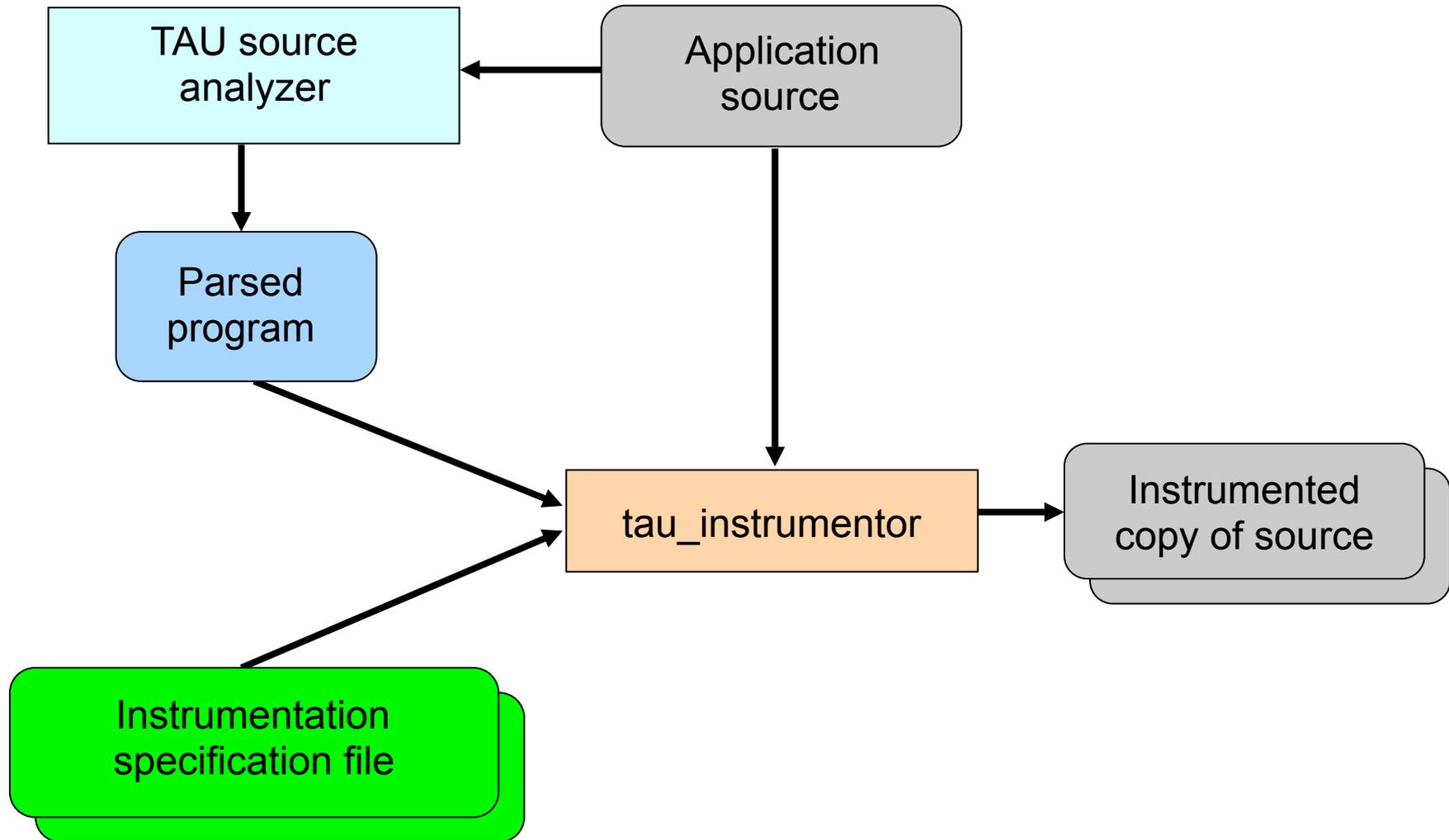
Or use a selective instrumentation file (include/exclude lists)

```
% tau_rewrite -f select.tau -T icpc,papi \
    ./matmult -o matmult.i
% mpirun -np 256 ./matmult.i
% paraprof
```

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



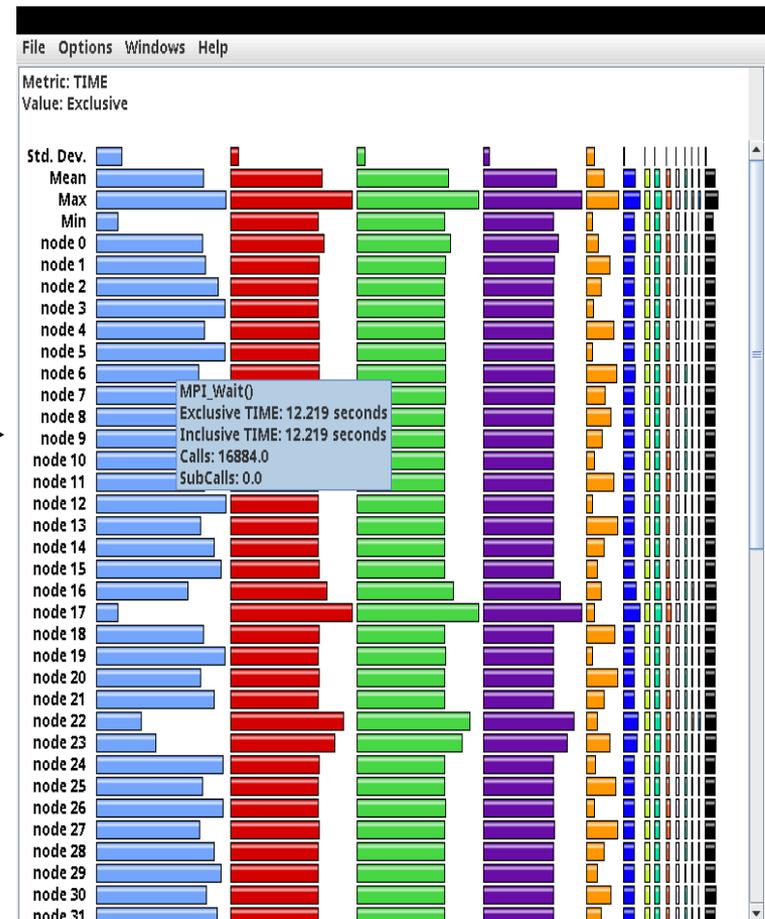
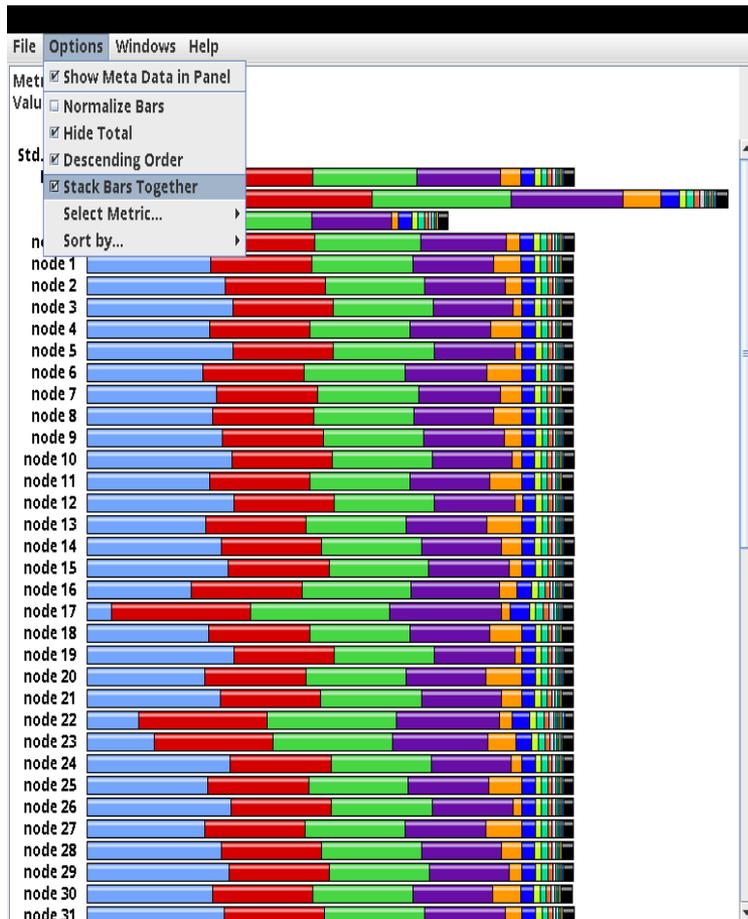
PDT: Automatic Source Instrumentation



Selective Instrumentation File

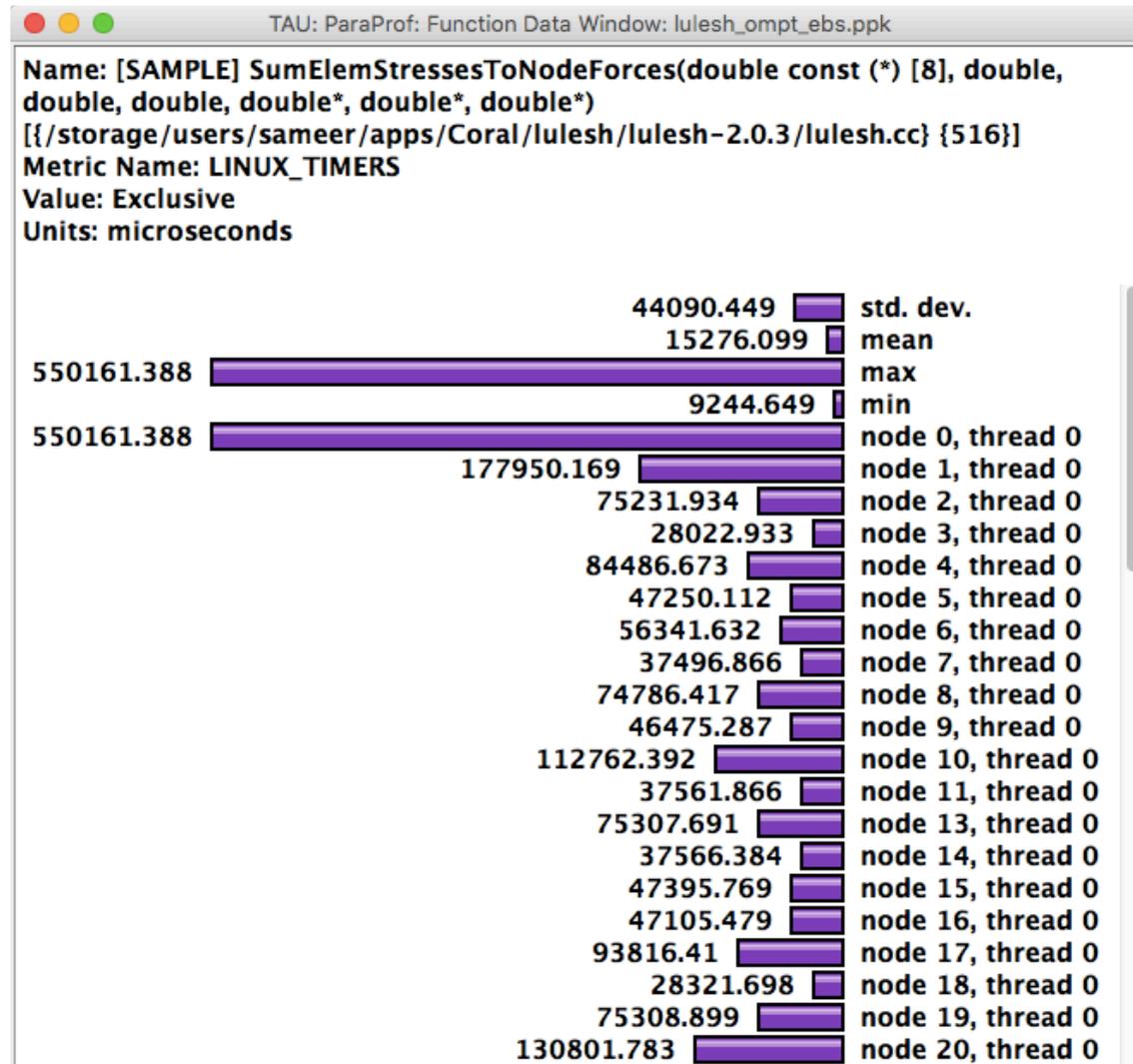
```
% export TAU_OPTIONS='-optTauSelectFile=select.tau ...'  
% make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh  
% cat select.tau  
BEGIN_INCLUDE_LIST  
int main#  
int dgemv#  
END_INCLUDE_LIST  
BEGIN_FILE_INCLUDE_LIST  
Main.c  
Blas/*.f77  
END_FILE_INCLUDE_LIST  
# replace include with exclude list  
  
BEGIN_INSTRUMENT_SECTION  
loops routine="foo"  
loops routine="int main#"  
END_INSTRUMENT_SECTION  
% export TAU_SELECT_FILE=select.tau      (to filter at runtime)
```

ParaProf Profile Browser



Click “node X” next to see details

TAU – Event Based Sampling (EBS)



% export TAU_SAMPLING=1 or tau_exec -ebs

Python Instrumentation

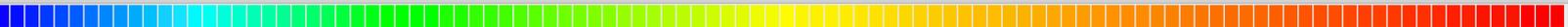
TAU: ParaProf: Statistics for: node 0, thread 0 - amd_resnet50_fp_1.ppk

Name	Exclusive TAUGPU_TIME	Inclusive TAUGPU_TIME	Calls	Child Calls
▾ .TAU application	0.575	182.783	1	6
▾ ▾ <module> [micro_benchmarking_pytorch.py{1}]	0.002	182.151	1	13
▾ ▾ ▾ main [micro_benchmarking_pytorch.py{81}]	0.002	168.702	1	1
▾ ▾ ▾ ▾ run_benchmarking [micro_benchmarking_pytorch.py{40}]	0.006	168.7	1	40
▾ ▾ ▾ ▾ ▾ forwardbackward [micro_benchmarking_pytorch.py{33}]	0.002	155.924	22	110
▾ ▾ ▾ ▾ ▾ ▾ backward [tensor.py{79}]	0.001	106.141	22	22
▾ ▾ ▾ ▾ ▾ ▾ ▾ backward [__init__.py{38}]	0.001	106.14	22	88
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ run_backward	106.135	106.135	22	3
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ pthread_create	0	0	3	0
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ _make_grads [__init__.py{20}]	0.001	0.004	22	110
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ isinstance	0	0	22	0
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ len	0	0	22	0
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ __call__ [module.py{485}]	0	49.77	22	110
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ forward [container.py{95}]	0	49.768	22	66
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ __call__ [module.py{485}]	0.001	49.767	44	220
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ forward [resnet.py{151}]	0.003	49.765	22	484
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ __call__ [module.py{485}]	0.006	49.759	220	1,100
▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ ▾ forward [container.py{95}]	0.002	45.622	88	440
▾ __call__ [module.py{485}]	0.007	45.616	352	1,760
▾ forward [resnet.py{78}]	0.071	45.598	352	6,600
▾ __call__ [module.py{485}]	0.07	45.495	3,256	16,280
▾ forward [conv.py{319}]	0.017	29.675	1,056	3,168
▾ conv2d	29.648	29.648	1,056	0
▾ __getattr__ [module.py{523}]	0.01	0.01	2,112	0
▾ forward [container.py{95}]	0.002	9.401	88	264
▾ forward [batchnorm.py{59}]	0.262	6.097	1,056	9,504

% tau_python ./foo.py

Identifying Wait States Using EBS

TAU: ParaProf: Statistics for: node 0, thread 0 - nt3_baseline_keras2.ppk



Name	Inclusive ...	Calls ▾
▣ _do_call [{}session.py{}1348]	512.135	82
▣ _run_fn [{}session.py{}1317]	512.134	82
▾ ▣ TF_Run	512.093	82
▾ ▣ [CONTEXT] TF_Run	512.173	51,211
▣ [SAMPLE] __pthread_cond_wait [{} {} 0]	511.273	51,123
▣ [SAMPLE] tensorflow::TensorBuffer* tensorflow::(anonymous namespace)::FromProtoField	0.42	42
▣ [SAMPLE] __memcpy_ssse3_back [{} {} 0]	0.28	28
▣ [SAMPLE] _int_free [{}malloc.c {} 0]	0.03	3
▣ [SAMPLE] __GI___libc_malloc [{} {} 0]	0.02	2
▣ [SAMPLE] std::basic_string<char, std::char_traits<char>, std::allocator<char> >::assign(s	0.02	1
▣ [SAMPLE] google::protobuf::internal::MapField<tensorflow::NodeDef::NodeDef_AttrEntry,	0.02	1
▣ [SAMPLE] __exchange_and_add [{} /home/msarahan/miniconda2/conda-bld/compiler_lin	0.01	1
▣ [SAMPLE] void google::protobuf::internal::RepeatedPtrFieldBase::MergeFromInnerLoop<g	0.01	1
▣ [SAMPLE] google::protobuf::internal::ArenaStringPtr::Destroy(std::basic_string<char, std::	0.01	1
▣ [SAMPLE] std::_Hashtable<std::basic_string<char, std::char_traits<char>, std::allocator<	0.01	1
▣ [SAMPLE] std::_Hashtable<tensorflow::Node*, std::pair<tensorflow::Node* const, tensorf	0.01	1
▣ [SAMPLE] std::basic_string<char, std::char_traits<char>, std::allocator<char> >::_Rep::_I	0.01	1
▣ [SAMPLE] std::_Hash_bytes(void const*, unsigned long, unsigned long) [{} /home/msaraha	0.01	1
▣ [SAMPLE] std::basic_string<char, std::char_traits<char>, std::allocator<char> >::_M_cop	0.01	1
▣ [SAMPLE] std::basic_string<char, std::char_traits<char>, std::allocator<char> >::size() cc	0.01	1
▣ [SAMPLE] PyObject_Malloc [{} /home/nwani/m2u/conda-bld/python_1500576437846/woi	0.01	1
▣ [SAMPLE] jemalloc_free [{} /home/nchaimov/candle/anaconda3/lib/python3.6/site-packag	0.01	1

```
% tau_python -ebs ./foo.py
```

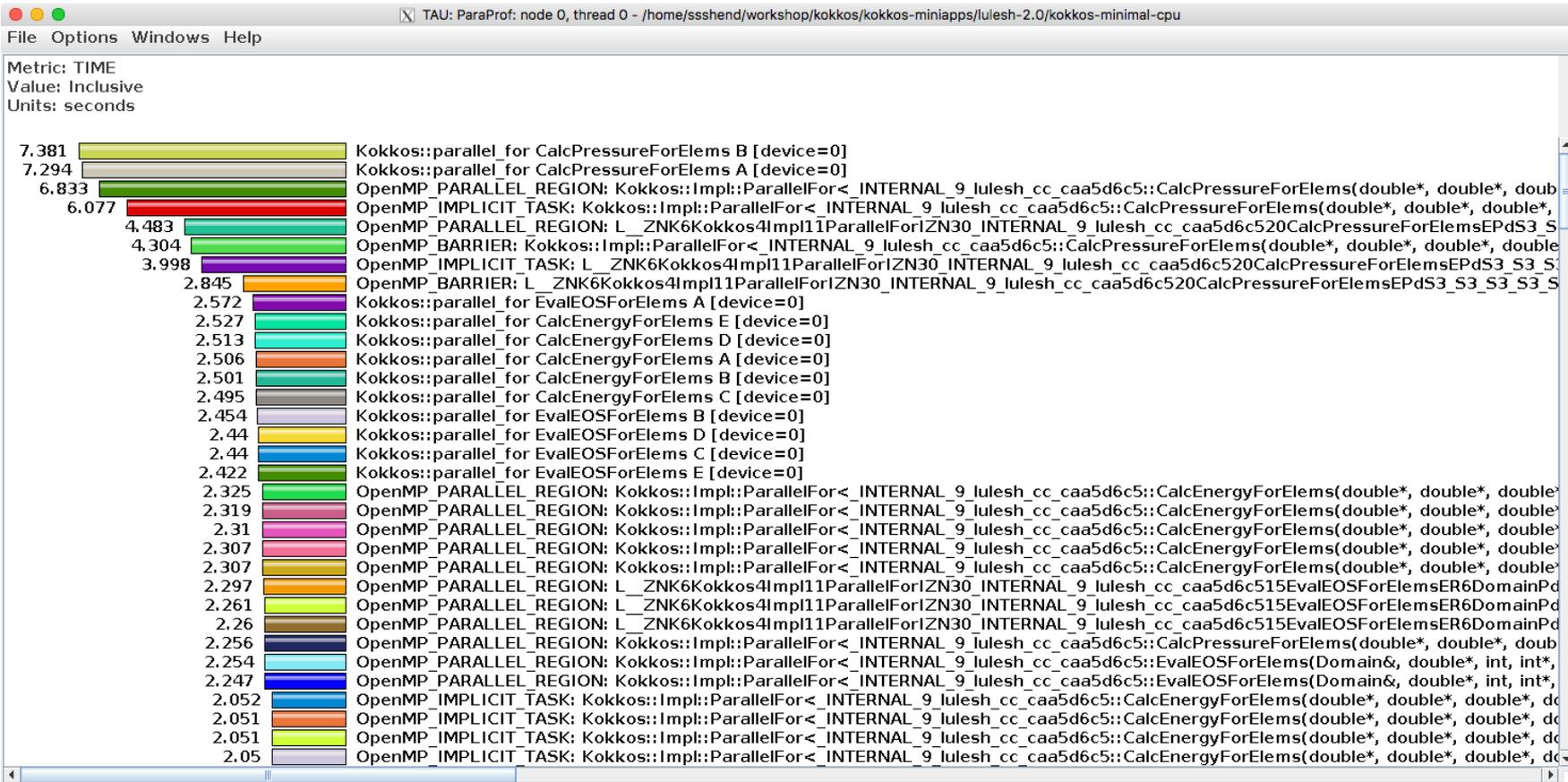
Kokkos and OpenMP Instrumentation

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

Name	Exclusive...	Inclusive...	Calls	Child C...
TAU application	1.796	16.719	1	27
Kokkos::parallel_for Kokkos::Example::BoxElemFixture<Kokkos::OpenMP, (Kokkos::Example::BoxElemPart::ElemOrder)0, ...	0.001	2.824	2	2
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::In	0	2.823	2	2
OpenMP_IMPLICIT_TASK: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Im	2.823	2.823	2	0
Kokkos::parallel_reduce Kokkos::Example::FiniteElementIntegration<Kokkos::Example::BoxElemFixture<Kokkos::OpenMP	0	4.027	1	1
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::In	0	4.027	1	1
OpenMP_IMPLICIT_TASK: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Im	4.027	4.027	1	0
Kokkos::parallel_reduce Kokkos::Example::FiniteElementIntegration<Kokkos::Example::BoxElemFixture<Kokkos::OpenMP	0.001	5.993	1	1
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::In	0	5.993	1	1
OpenMP_IMPLICIT_TASK: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Im	5.993	5.993	1	0
Kokkos::parallel_reduce Kokkos::Example::LumpElemToNode<Kokkos::View<double* [8], Kokkos::OpenMP>, Kokkos::Vie	0.001	1.266	1	1
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::In	0	1.265	1	1
OpenMP_IMPLICIT_TASK: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Im	1.265	1.265	1	0
Kokkos::parallel_reduce Kokkos::Example::LumpElemToNode<Kokkos::View<double* [8], Kokkos::OpenMP>, Kokkos::Vie	0.001	0.125	1	1
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::In	0	0.124	1	1
OpenMP_IMPLICIT_TASK: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Im	0.124	0.124	1	0
OpenMP_PARALLEL_REGION: Kokkos::Impl::OpenMPExec::clear_scratch() [clone _omp_fn.0] [{/home/users/sameer/apps	0	0	2	2
OpenMP_PARALLEL_REGION: Kokkos::Impl::OpenMPExec::resize_scratch(unsigned long, unsigned long) [clone _omp_fn.:	0	0	1	1
OpenMP_PARALLEL_REGION: Kokkos::OpenMP::initialize(unsigned int, unsigned int, unsigned int) [clone _omp_fn.2] [{/l	0.017	0.025	1	1
OpenMP_PARALLEL_REGION: Kokkos::OpenMP::initialize(unsigned int, unsigned int, unsigned int) [clone _omp_fn.3] [{/l	0	0	1	1
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Impl	0	0.514	6	6
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Impl	0	0.036	5	5
OpenMP_PARALLEL_REGION: std::enable_if<std::is_same<Kokkos::Static, Kokkos::Static>::value, void>::type Kokkos::Impl	0	0.113	5	5

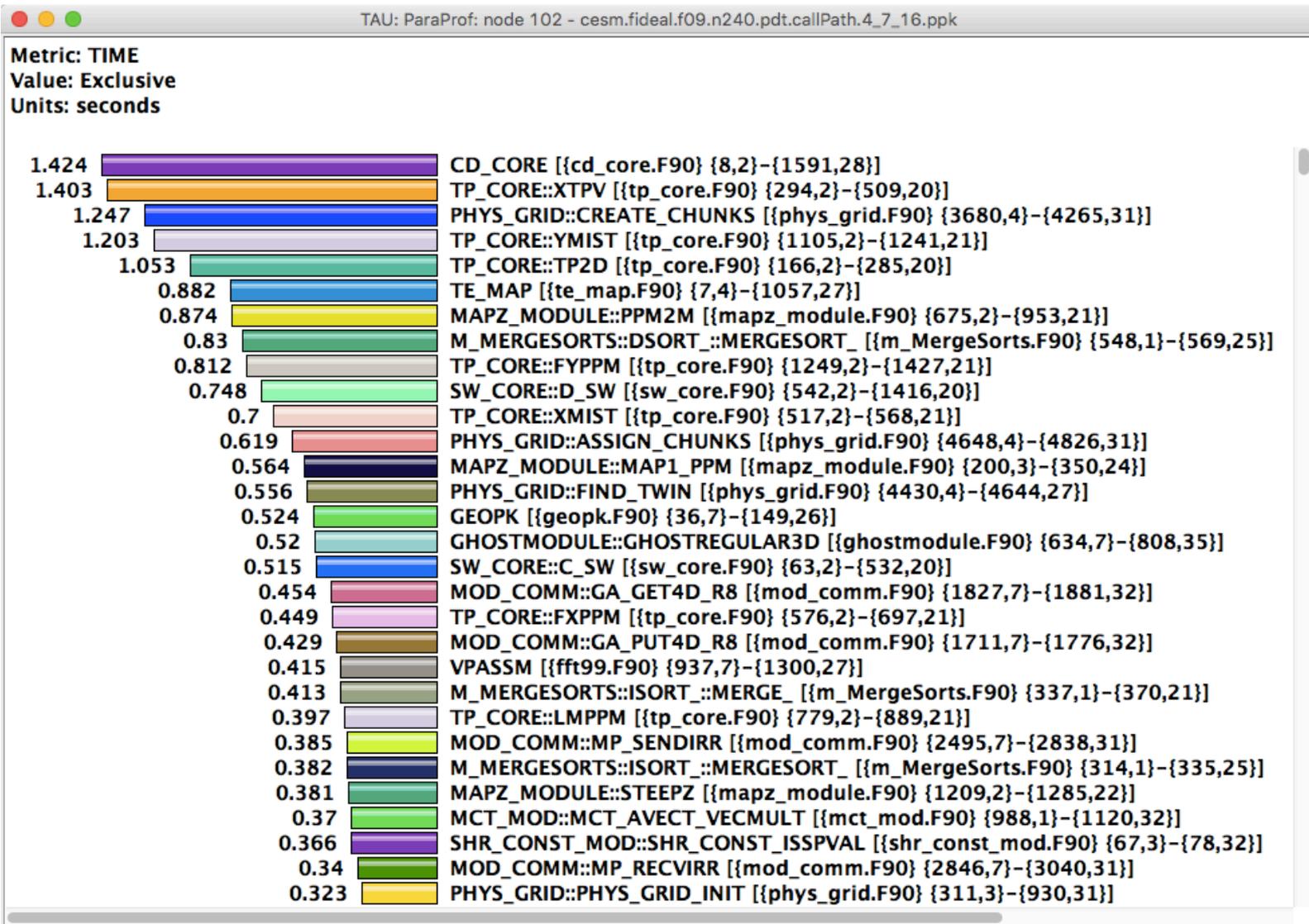
```
% tau_exec -ompt ./a.out
```

Kokkos Instrumentation with OpenMP

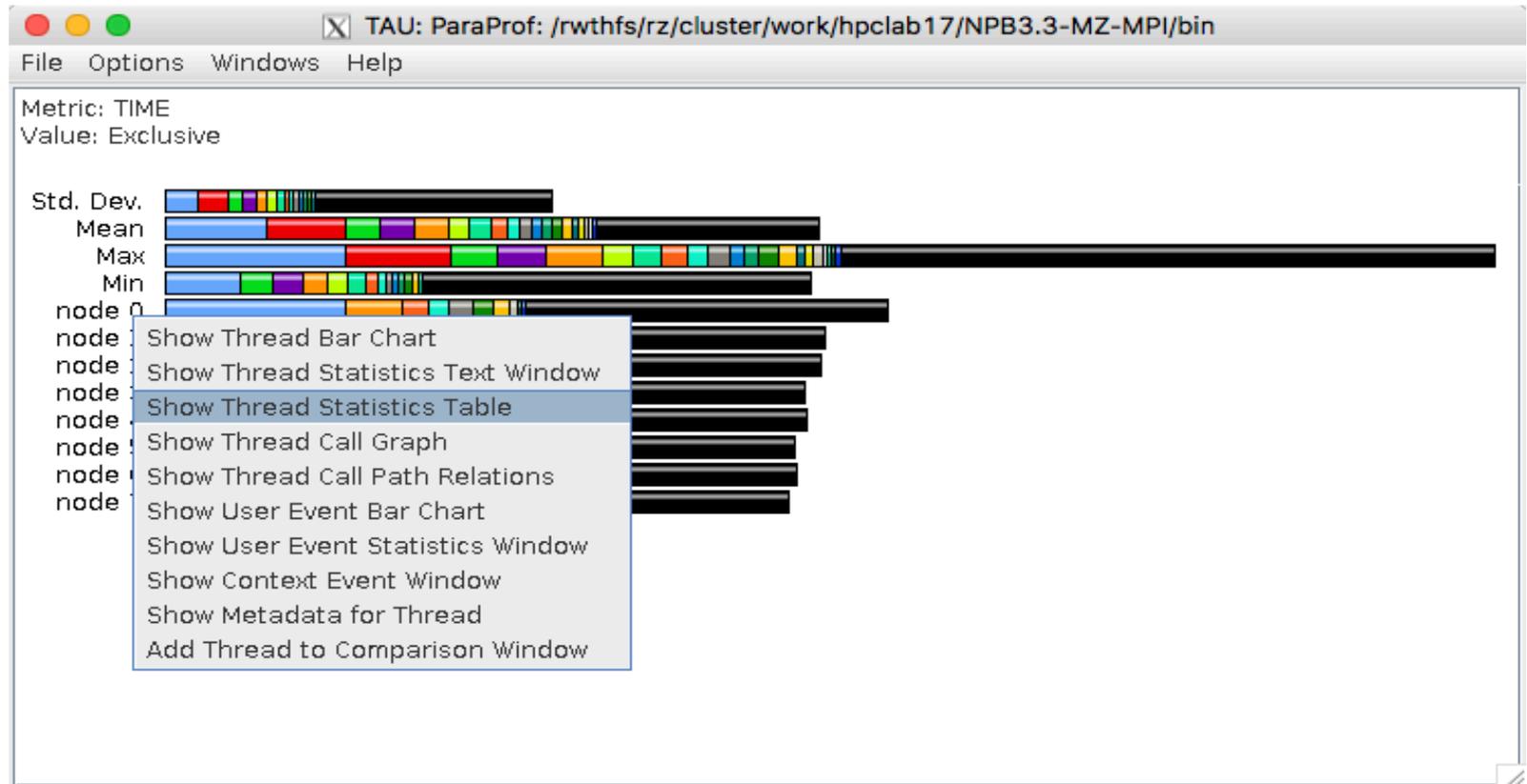


% tau_exec -ompt ./a.out

TAU – Flat Profile



ParaProf Thread Statistics Table



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

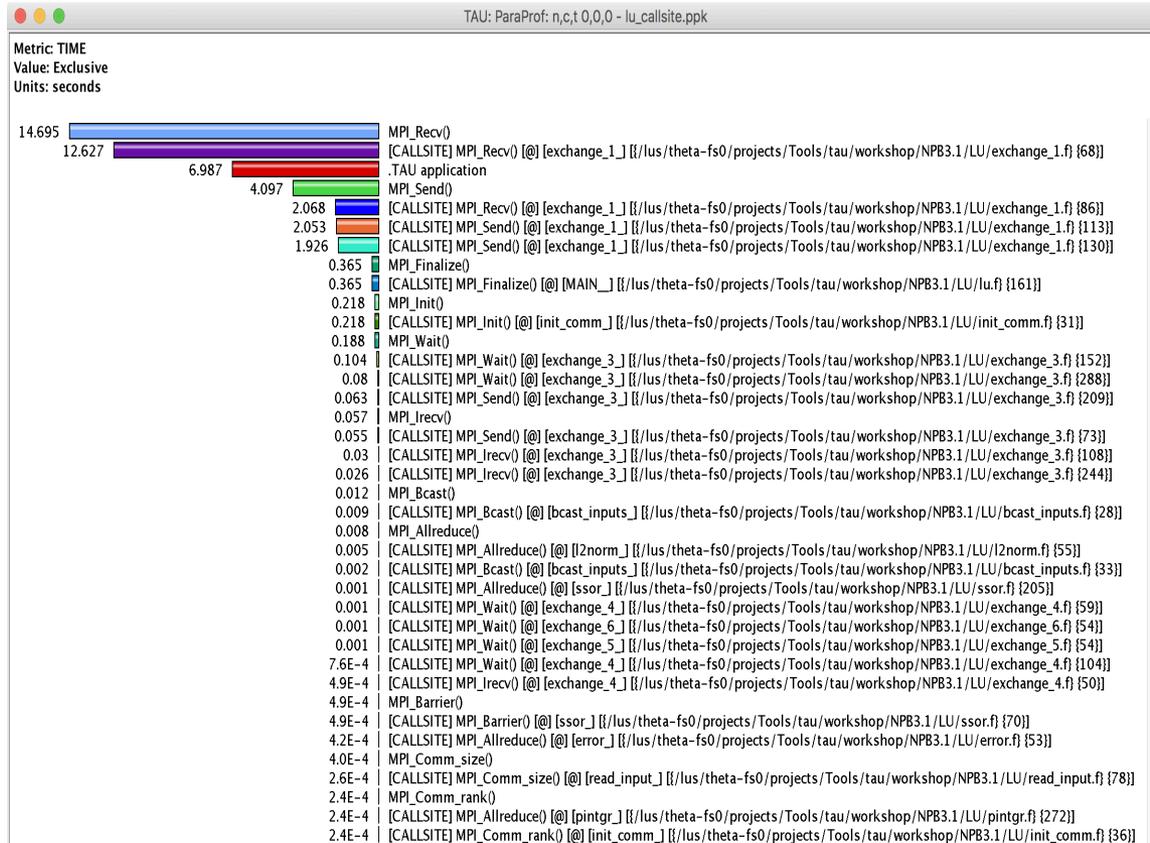
TAU – Callsite Profiling

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

Name	Excl...	Inclu...	Calls	Chil...
.TAU application	6.152	8.249	1	28,383
[CALLSITE] void start_pes_(int *) [@@] [/nfsprojects/volta-projects/tau/tau-2.24.1/craycnl/lib/libTAUsh-gnu-papi-shmem-pdt.so] UNRESOLVED ADDR	0.747	0.747	1	0
void start_pes_(int *)	0.747	0.747	1	0
void shmem_barrier_all_0	0.624	0.624	9,229	0
[CALLSITE] void shmem_barrier_all_0 [@@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90} {572}]	0.401	0.401	4,610	0
[CALLSITE] void shmem_finalize_0 [@@] [/nfsprojects/volta-projects/tau/tau-2.24.1/craycnl/lib/libTAUsh-gnu-papi-shmem-pdt.so] UNRESOLVED ADDR	0.314	0.314	1	0
void shmem_finalize_0	0.314	0.314	1	0
[CALLSITE] void shmem_barrier_all_0 [@@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90} {643}]	0.223	0.223	4,610	0
void shmem_put64_nb_(void *, void *, int *, int *, void *)	0.159	0.159	9,220	0
void shmem_put64_(void *, void *, int *, int *)	0.126	0.126	9,220	0
void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, int *, void *, long *)	0.081	0.081	400	0
[CALLSITE] void shmem_put64_nb_(void *, void *, int *, int *, void *) [@@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_...	0.07	0.07	4,610	0
[CALLSITE] void shmem_put64_(void *, void *, int *, int *) [@@] [__clover_module_MOD_clover_exchange_message] [{/home/ssshend/CloverLeaf_OpenSHMEM...	0.063	0.063	4,610	0
[CALLSITE] void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *) [@@] [hydro_] [{/home/ssshend/CloverLeaf_OpenSHMEM/hydr...	0.046	0.046	200	0
[CALLSITE] void shmem_real8_min_to_all_(void *, void *, int *, int *, int *, int *, void *, long *) [@@] [/nfsprojects/volta-projects/tau/tau-2.24.1/craycnl/lib/...	0.04	0.04	200	0
void shmem_real8_min_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.04	0.04	200	0
[CALLSITE] void shmem_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *) [@@] [hydro_] [{/home/ssshend/CloverLeaf_OpenSHMEM/hydr...	0.036	0.036	200	0
[CALLSITE] void shmem_put64_nb_(void *, void *, int *, int *, void *) [@@] [__clover_module_MOD_clover_exchange] [{/home/ssshend/CloverLeaf_OpenSHME...	0.028	0.028	601	0

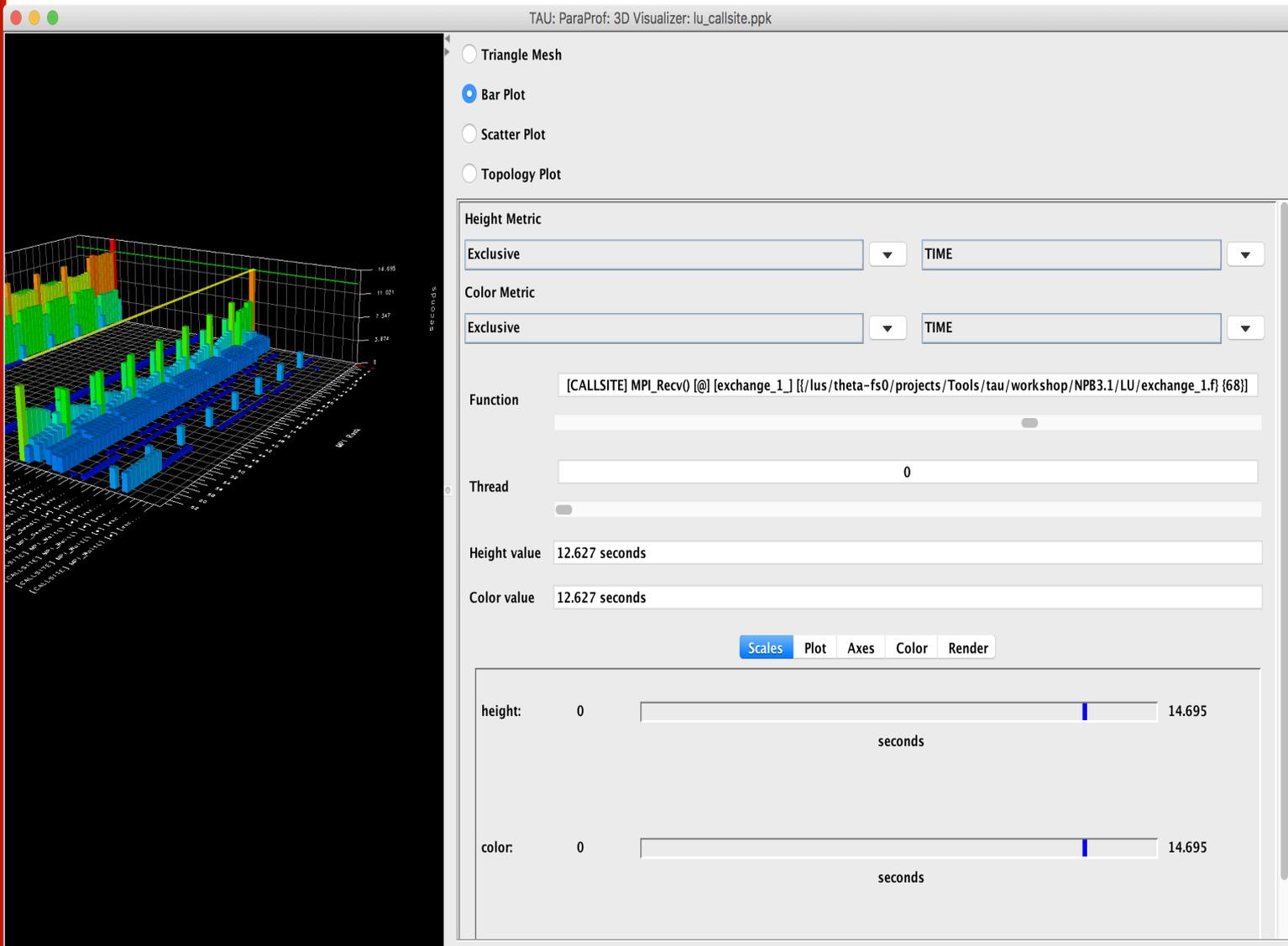
% export TAU_CALLSITE=1

Callsite Profiling and Tracing



% export TAU_CALLSITE=1

Callsite Profiling and Tracing



TAU – Callstack Sampling

TAU: ParaProf: Statistics for: n,c,t 0,0,0 - clover_gnu_ebs_unw_call.ppk

Name	Inclusive...	Calls
▾ .TAU application	34.979	1
▸ [CONTEXT] .TAU application	31.647	632
▾ void shmем_barrier_all_0	1.219	46,029
▾ [CONTEXT] void shmем_barrier_all_0	1.599	32
▾ [UNWIND] [/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90.41] [@@] UNRESOLVED /lib64/libc-2.11.3.so	1.599	32
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.62 [@@] main [[/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90] {41}]	0.85	17
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/advection.f90.102 [@@] hydro_ [[/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90] {62}]	0.55	11
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/update_halo.f90.36 [@@] __advection_module_MOD_advection [[/home/ssshend/CloverLeaf_Ope]	0.55	11
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.292 [@@] __update_halo_module_MOD_update_halo [[/home/ssshend/CloverLeaf_O	0.5	10
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.572 [@@] __clover_module_MOD_clover_exchange [[/home/ssshend/CloverLeaf_	0.5	10
▾ [UNWIND] UNRESOLVED [@@] __clover_module_MOD_clover_exchange_message [[/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90] {572}]	0.5	10
▾ [UNWIND] [/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_barrier.c.118] [@@] UNRESOLVED /nfsproje	0.45	9
▾ [SAMPLE] _smai_smp_barrier_in [[/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_barrier.c] {118}]	0.45	9
▸ [UNWIND] [/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_internal.h.88] [@@] UNRESOLVED /nfsprojects/\	0.05	1
▸ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.461 [@@] __update_halo_module_MOD_update_halo [[/home/ssshend/CloverLeaf_O	0.05	1
▸ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/advection.f90.72 [@@] hydro_ [[/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90] {62}]	0.15	3
▸ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/advection.f90.55 [@@] hydro_ [[/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90] {62}]	0.15	3
▸ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.52 [@@] main [[/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90] {41}]	0.5	10
▸ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.54 [@@] main [[/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90] {41}]	0.25	5
▸ void start_pes_(int *)	0.508	1
▾ void shmем_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.325	2,000
▾ [CONTEXT] void shmем_real8_max_to_all_(void *, void *, int *, int *, int *, int *, void *, long *)	0.5	10
▾ [UNWIND] [/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90.41] [@@] UNRESOLVED /lib64/libc-2.11.3.so	0.5	10
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.58 [@@] main [[/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90] {41}]	0.45	9
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/PdV.f90.107 [@@] hydro_ [[/home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90] {58}]	0.45	9
▾ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/clover.f90.740 [@@] __pdv_module_MOD_pdv [[/home/ssshend/CloverLeaf_OpenSHMEM/PdV.f9C	0.45	9
▾ [UNWIND] UNRESOLVED [@@] __clover_module_MOD_clover_check_error [[/home/ssshend/CloverLeaf_OpenSHMEM/clover.f90] {740}]	0.45	9
▾ [UNWIND] [/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_reduction.h.207] [@@] UNRESOLVED /nfsprojects/vol	0.45	9
▾ [UNWIND] /notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_reduction.h.788 [@@] pshmem_double_max_tc	0.45	9
▾ [UNWIND] /notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_reduction.h.107 [@@] _smai_opt_double_m	0.45	9
▾ [SAMPLE] _smai_smp_reduce_double_max [[/notbackedup/tmp/ulib/mpt/nightly/7.2/062215-RC/sma_dmapp/src/shmem_opt_reduc]	0.45	9
▸ [UNWIND] /home/ssshend/CloverLeaf_OpenSHMEM/hydro.f90.54 [@@] main [[/home/ssshend/CloverLeaf_OpenSHMEM/clover_leaf.f90] {41}]	0.05	1

% export TAU_SAMPLING=1; export TAU_EBS_UNWIND=1

TAU – Callpath Profiling

TAU: ParaProf: Statistics for: node 5 - fun3d_d19.ppk

Name	Exclusive...	Inclusive...	Calls	Child...
▼ .TAU application	0	221.298	1	1
▼ NODET [{{main.f90} {4,1}-{35,17}}	0	221.298	1	105
▶ FLOW::ITERATE [{{flow.F90} {1692,14}}	0	197.989	100	500
▼ FLOW::INITIALIZE_DATA [{{flow.F90} {465,14}}	0	22.707	1	2
▼ FLOW::INITIALIZE_DATA2 [{{flow.F90} {663,14}}	0.002	22.705	1	197
▼ PPARTY_PREPROCESSOR::PPARTY_PREPROCESS [{{pparty_preprocessor.f90} {28,14}}	0	20.897	1	23
▼ PPARTY_PREPROCESSOR::PPARTY_READ_GRID [{{pparty_preprocessor.f90} {735,14}}	0	16.726	1	2
▼ PUNS3D_IO_C2N::PUNS3D_READ_VGRID_C2N [{{puns3d_io_c2n.f90} {1543,14}}	0.011	16.725	1	11
▼ PUNS3D_IO_C2N::PUNS3D_READ_VGRID_C2N_SM [{{puns3d_io_c2n.f90} {1641,14}}	0	16.656	1	5
▼ PUNS3D_IO_C2N::DISTRIBUTE_TET [{{puns3d_io_c2n.f90} {1819,14}}	0.117	16.572	1	5
▼ LMPI::INTEGR_MATRIX_BCAST [{{lmpi.F90} {3240,3}-{3276,36}}	0	16.448	4	4
■ MPI_Bcast0	16.448	16.448	4	0
▶ LMPI::LMPI_CONDITIONAL_STOP [{{lmpi.F90} {611,3}-{672,38}}	0	0.007	1	2
▶ PUNS3D_IO_C2N::DISTRIBUTE_XYZ [{{puns3d_io_c2n.f90} {2448,14}}	0.001	0.083	1	3
▶ LMPI::INTEGR_SCALAR_BCAST [{{lmpi.F90} {3151,3}-{3187,36}}	0	0	3	3
▶ LMPI::LMPI_CONDITIONAL_STOP [{{lmpi.F90} {611,3}-{672,38}}	0	0.058	1	2
▶ LMPI::INTEGR_SCALAR_BCAST [{{lmpi.F90} {3151,3}-{3187,36}}	0	0	2	2
■ ALLOCATIONS::INTEGER_4_MY_ALLOC_PTR2 [{{allocations.f90} {1010,3}-{1026,40}}	0	0	6	0
■ PUNS3D_IO_C2N::DISTRIBUTE_FAST_C2N [{{puns3d_io_c2n.f90} {4226,14}}	0	0	1	0
▶ LMPI::LMPI_CONDITIONAL_STOP [{{lmpi.F90} {611,3}-{672,38}}	0	0.001	1	2
▶ PPARTY_MIXED_ELEMENT::EDGE_POINTER_DRIVER [{{pparty_mixed_element.f90} {74,3}-{50}}	0.65	0.873	1	174
▶ PPARTY::NODE_CELL_CHOPPER [{{pparty.f90} {41,3}-{453,33}}	0.288	0.86	1	175
▶ PPARTY_PUNS3D::RAW_GRID_CHECKER [{{pparty_puns3d.f90} {623,14}}	0.233	0.523	1	11
▶ PPARTY_METIS::MY_METIS [{{pparty_metis.F90} {116,3}-{545,24}}	0.313	0.436	1	13,132
▶ PARTY_LMPI::PARTY_LMPI_SETUP_MPI_SM [{{party_lmpi.f90} {613,3}-{686,40}}	0.006	0.337	1	10

% export TAU_CALLPATH=1; export TAU_CALLPATH_DEPTH=100

TAU Atomic Events

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

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

TAU – Context Events

TAU: ParaProf: Context Events for thread: n,c,t, 1,0,0 – samarc_obe_4p_iomem_cp.ppk

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

Write bandwidth per file

Bytes written to each file

Packaging TAU: Exascale Preparation

- SDK Exascale platform preparation is focused on **interoperable delivery**.
- ST products from SDKs are released in the Extreme-scale Scientific Software Stack (E4S) [<https://e4s.io>].
 - E4S: a community effort to provide open source software packages for developing, deploying, and running scientific applications on HPC platforms
- **E4S containers and Spack based builds currently support the following pre-exascale systems:**
 - Theta at ALCF (Cray XC).
 - Cori at NERSC (Cray XC).
 - Summit, Sierra, Butte, RZAnsel (IBM Power 9 AC922).
 - Linux x86_64 systems at LANL (Grizzly), Sandia (Voltrino), LLNL (Quartz).
 - Other NSF platforms including Frontera (TACC).
- **E4S preparation for future Exascale systems includes testing on AMD and Intel systems.**

**E4S is led and funded out of the PMR SDK, but is the delivery vehicle for all SDK efforts



Integration and Interoperability

• **SDK integration efforts focus on establishing SDKs and adding ST products into E4S.**

• **Challenges include:**

- Greatly varying software quality and related practices
- Establishing and **maintaining** common versions of software dependencies
- Establishing **workflows** that allow autonomy for individual teams, yet maintain an interoperable ecosystem
- **Sustaining** interoperability without established processes and mature CI tools

• **Strategy:**

- Begin with what is currently feasible given the state of CI testing support
 - Periodic integrations
 - Intermittent testing
- Balance achieving & maintaining interoperability with making progress on pre-exascale platforms
- Develop workflows and proposed policies related to sustaining interoperability
- Initial engagement with facilities involved
 - Education on SDKs and E4S and their utility
 - An effort at each facility to install the xSDK using the Spack meta-package
 - Smaller scale, more robust than trying everything
 - Gathering feedback
- Next (with the Spack team) we will introduce the Spack manifest / build cache approach

Integration and Interoperability: E4S

- E4S is released twice a year. Two versions have been released to date and we are planning for a release at SC19. The E4S 0.2 release supports:
 - Containers and turn-key, from-source builds of 80+ popular HPC software packages
 - 40 full release ECP ST products including:
 - MPI: MPICH and OpenMPI
 - Development tools: TAU, HPCToolkit, and PAPI
 - Math libraries: PETSc and Trilinos
 - Data and Viztools: Adios, HDF5, and Paraview
 - Limited access to 10 additional ECP ST products
 - Docker
 - Singularity
 - Shifter
 - Charliecloud
 - Inception
 - Open Virtualization Appliance (OVA) for VirtualBox features Spack, E4S containers, and support for container environments

Integration and Interoperability: E4S on AWS

- E4S AWS public image ami-063e830287b86155c (US-West-2 Oregon) has following container runtimes:
 - Docker
 - Shifter
 - Singularity
 - Charliecloud
- Spack with base PMR components
- E4S full featured Singularity image
 - (exascaleproject/sdk:AHM19)
- Used in ISC-HPC 2019 tutorials
- **Used as base image for NASA GEOS-Chem E4S public image**
- Resources provided by AWS AI/ML team



The screenshot shows the AWS Management Console interface. The top navigation bar includes 'Services', 'Resource Groups', and a search icon. The left sidebar contains various navigation options like 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'INSTANCES', 'Launch Templates', 'Spot Requests', 'Reserved Instances', 'Dedicated Hosts', 'Scheduled Instances', 'Capacity Reservations', 'IMAGES', 'AMIs', 'Bundle Tasks', 'ELASTIC BLOCK STORE', 'Volumes', 'Snapshots', 'Lifecycle Manager', 'NETWORK & SECURITY', and 'Security Groups'. The main content area displays a table of AMIs owned by the user. The table has columns for Name, AMI Name, AMI ID, Source, Owner, Visibility, and Status. The first row is selected, showing 'GEOS Chem E4S container with Spack, Docker, ...' with AMI ID 'ami-016555a769a29afeb'. Below the table, the 'Details' tab for this AMI is expanded, showing fields like AMI ID, Owner, Status, Creation date, Architecture, Virtualization type, Root Device Name, RAM disk ID, Product Codes, AMI Name, Source, State Reason, Platform, Image Type, Description, Root Device Type, Kernel ID, and Block Devices.

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status
GEOS Chem E4S container with Spack, Docker, ...	E4S-GEOS-Chem	ami-016555a769a29afeb	792568971918/E4S-GEOS-Chem	792568971918	Public	available
SuperLU Tutorial E4S Singularity	E4S_SC_SuperLU_Tutorial	ami-07d0b5dab32444ff	792568971918/E4S_SC_SuperLU_Tutorial	792568971918	Public	available
ECP E4S image with AI and HPC software stacks...	import-ami-07af056b52139562	ami-063e830287b86155c	792568971918/import-ami-07af056b52139562	792568971918	Public	available
E4S container with Spack, Docker, Singularity, SH...	import-ami-0d68aa1dc4496567c	ami-0997e7525abb44f	792568971918/import-ami-0d68aa1dc4496567c	792568971918	Public	available

Field	Value	Field	Value
AMI ID	ami-016555a769a29afeb	AMI Name	E4S-GEOS-Chem
Owner	792568971918	Source	792568971918/E4S-GEOS-Chem
Status	available	State Reason	-
Creation date	July 30, 2019 at 2:04:49 PM UTC-7	Platform	Other Linux
Architecture	x86_64	Image Type	machine
Virtualization type	hvm	Description	E4S GEOS-Chem AWS AMI
Root Device Name	/dev/sda1	Root Device Type	eb6
RAM disk ID	-	Kernel ID	-
Product Codes	-	Block Devices	/dev/sda1-snap-06f38fc656d7760cb:80:false:gp2

Reproducible, Customizable Container Builds & Spack Mirrors

- E4S provides base images and recipes for building Docker containers based on SDKs
 - Git: <https://github.com/UO-OACISS/e4s>
 - Base images released (September 2019):
 - UBI 7.6 (RHEL Universal Binary Image for container builds) for x86_64
 - Centos 7.6 for x86_64
 - Ubuntu 18.04 for x86_64
 - UBI 7.6 (RHEL) for ppc64le
- E4S provides **build caches for Spack for native bare-metal as well as container builds based installation** of ECP Software Technology products
 - Build caches: <https://oaciss.uoregon.edu/e4s>
 - **The build cache model can be extended to target platforms**, and can be managed by facilities staff when appropriate.

Support Acknowledgments

US Department of Energy (DOE)

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



US Department of Defense (DoD)

- HPCMP



National Science Foundation (NSF)

- SI2-SSI, CSSI



NASA

CEA, France



Partners:

- University of Oregon
- The Ohio State University
- ParaTools, Inc.
- University of Tennessee, Knoxville



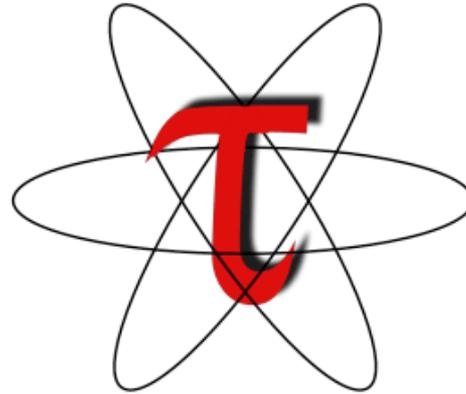
Acknowledgment



“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.”

<http://exascaleproject.org>

Download TAU



<http://tau.uoregon.edu>

<http://taucommander.com>

<http://www.hpclinux.com> [OVA for VirtualBox]

<https://e4s.io>

[Extreme-Scale Scientific Software Stack, Containers for HPC]

Free download, open source, BSD license

Reference

Installing and Configuring TAU

•Installing PDT:

- `wget tau.uoregon.edu/pdt_lite.tgz`
- `./configure --prefix=<dir>; make ; make install`

•Installing TAU:

- `wget tau.uoregon.edu/tau.tgz; tar xzf tau.tgz; cd tau-2.<ver>`
- `wget http://tau.uoregon.edu/ext.tgz ; tar xf ext.tgz`
- `./configure -bfd=download -pdt=<dir>`
`-iowrapper -mpi -dwarf=download -unwind=download -`
`otf=download -papi=<dir>`
`make install`

•Using TAU:

- `export TAU_MAKEFILE=<taudir>/<arch>/lib/Makefile.tau-<TAGS>`
- `make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh`

Compile-Time Options

Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

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

Compile-Time Options (contd.)

Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

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

See tau_compiler.sh for a full list of TAU_OPTIONS.

...

TAU's Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_CALLPATH	0	Setting to 1 turns on callpath profiling
TAU_TRACK_MEMORY_FOOTPRINT	0	Setting to 1 turns on tracking memory usage by sampling periodically the resident set size and high water mark of memory usage
TAU_TRACK_POWER	0	Tracks power usage by sampling periodically.
TAU_CALLPATH_DEPTH	2	Specifies depth of callpath. Setting to 0 generates no callpath or routine information, setting to 1 generates flat profile and context events have just parent information (e.g., Heap Entry: foo)
TAU_SAMPLING	1	Setting to 1 enables event-based sampling.
TAU_TRACK_SIGNALS	0	Setting to 1 generate debugging callstack info when a program crashes
TAU_COMM_MATRIX	0	Setting to 1 generates communication matrix display using context events
TAU_THROTTLE	1	Setting to 0 turns off throttling. Throttles instrumentation in lightweight routines that are called frequently
TAU_THROTTLE_NUMCALLS	100000	Specifies the number of calls before testing for throttling
TAU_THROTTLE_PERCALL	10	Specifies value in microseconds. Throttle a routine if it is called over 100000 times and takes less than 10 usec of inclusive time per call
TAU_CALLSITE	0	Setting to 1 enables callsite profiling that shows where an instrumented function was called. Also compatible with tracing.
TAU_PROFILE_FORMAT	Profile	Setting to "merged" generates a single file. "snapshot" generates xml format
TAU_METRICS	TIME	Setting to a comma separated list generates other metrics. (e.g., ENERGY,TIME,P_VIRTUAL_TIME,PAPI_FP_INS,PAPI_NATIVE_<event>:<subevent>)

Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_TRACE_FORMAT	Default	Setting to "otf2" turns on TAU's native OTF2 trace generation (configure with --otf=download)
TAU_EBS_UNWIND	0	Setting to 1 turns on unwinding the callstack during sampling (use with tau_exec -ebs or TAU_SAMPLING=1)
TAU_EBS_RESOLUTION	line	Setting to "function" or "file" changes the sampling resolution to function or file level respectively.
TAU_TRACK_LOAD	0	Setting to 1 tracks system load on the node
TAU_SELECT_FILE	Default	Setting to a file name, enables selective instrumentation based on exclude/include lists specified in the file.
TAU_OMPT_SUPPORT_LEVEL	basic	Setting to "full" improves resolution of OMPT TR6 regions on threads 1.. N-1. Also, "lowoverhead" option is available.
TAU_OMPT_RESOLVE_ADDRESS_EAGERLY	1	Setting to 1 is necessary for event based sampling to resolve addresses with OMPT. Setting to 0 allows the user to do offline address translation.

Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACK_MEMORY_LEAKS	0	Tracks allocates that were not de-allocated (needs <code>-optMemDbg</code> or <code>tau_exec -memory</code>)
TAU_EBS_SOURCE	TIME	Allows using PAPI hardware counters for periodic interrupts for EBS (e.g., <code>TAU_EBS_SOURCE=PAPI_TOT_INS</code> when <code>TAU_SAMPLING=1</code>)
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 <code>TAU_MEMDBG_PROTECT_*</code>)
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 <code>-optMemDbg</code> while building or <code>tau_exec -memory</code>)
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 <code>-optMemDbg</code> or <code>tau_exec -memory</code>)
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