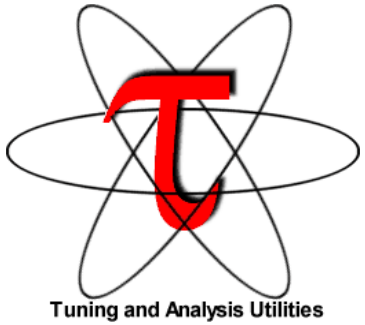
Performance Evaluation using TAU Performance System®

July 28, 2020, 10am PT
ORNL via Zoom



Sameer Shende
Performance Research Laboratory, OACISS, University of Oregon
http://tau.uoregon.edu/tau_ornl20.pdf

Challenges

- With growing hardware complexity, it is getting harder to accurately measure and optimize the performance of our HPC and AI/ML workloads.

- As our software gets more complex, it is getting harder to install tools and libraries correctly in an integrated and interoperable software stack.

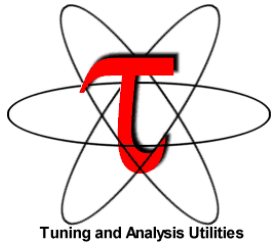
Motivation: Improving Productivity

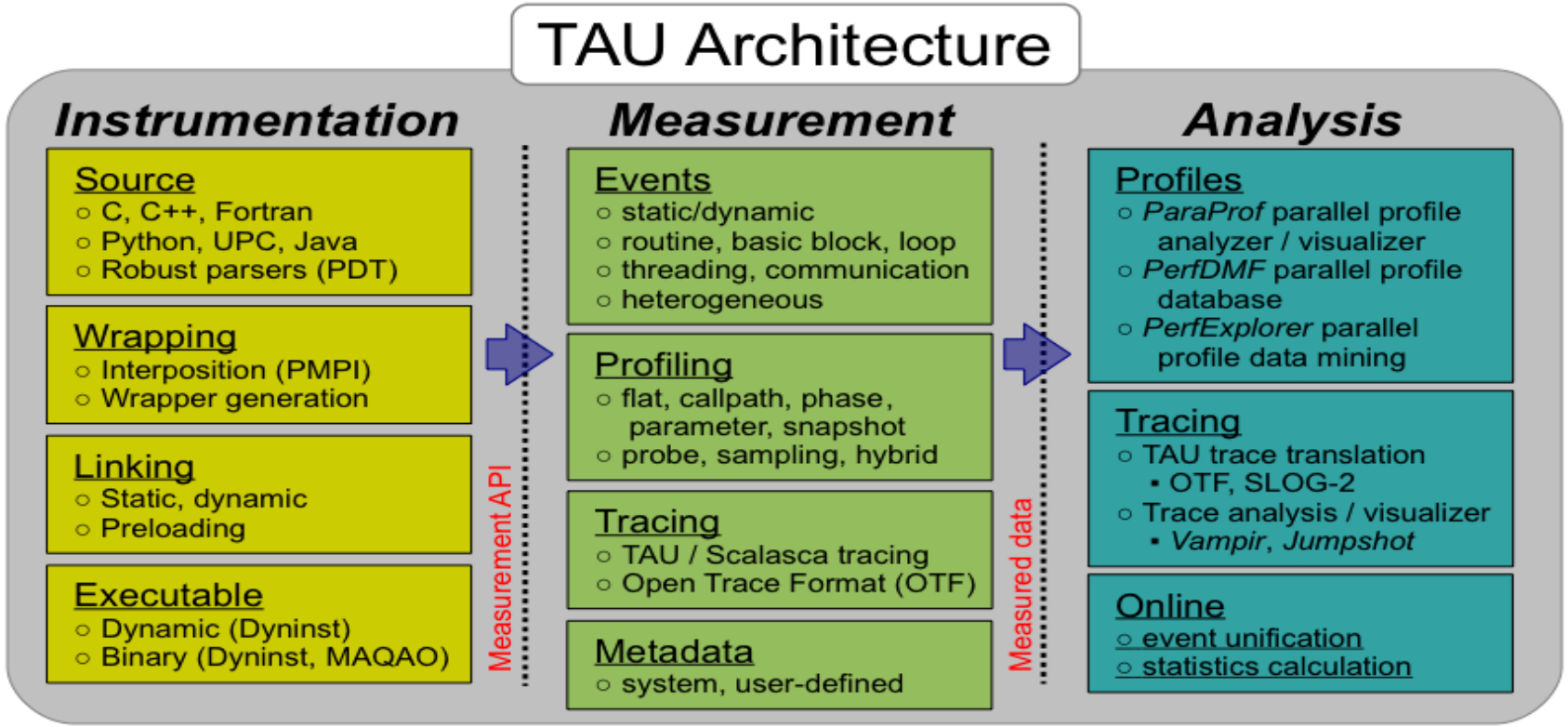
- TAU Performance System®:
  - Deliver a scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads
- Extreme-scale Scientific Software Stack (E4S):
  - Delivering a modular, interoperable, and deployable software stack
  - Deliver expanded and vertically integrated software stacks to achieve full potential of extreme-scale computing
  - Lower barrier to using software technology (ST) products from ECP
  - Enable uniform APIs where possible

TAU Performance System®

Parallel performance framework and toolkit

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





TAU Performance System

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

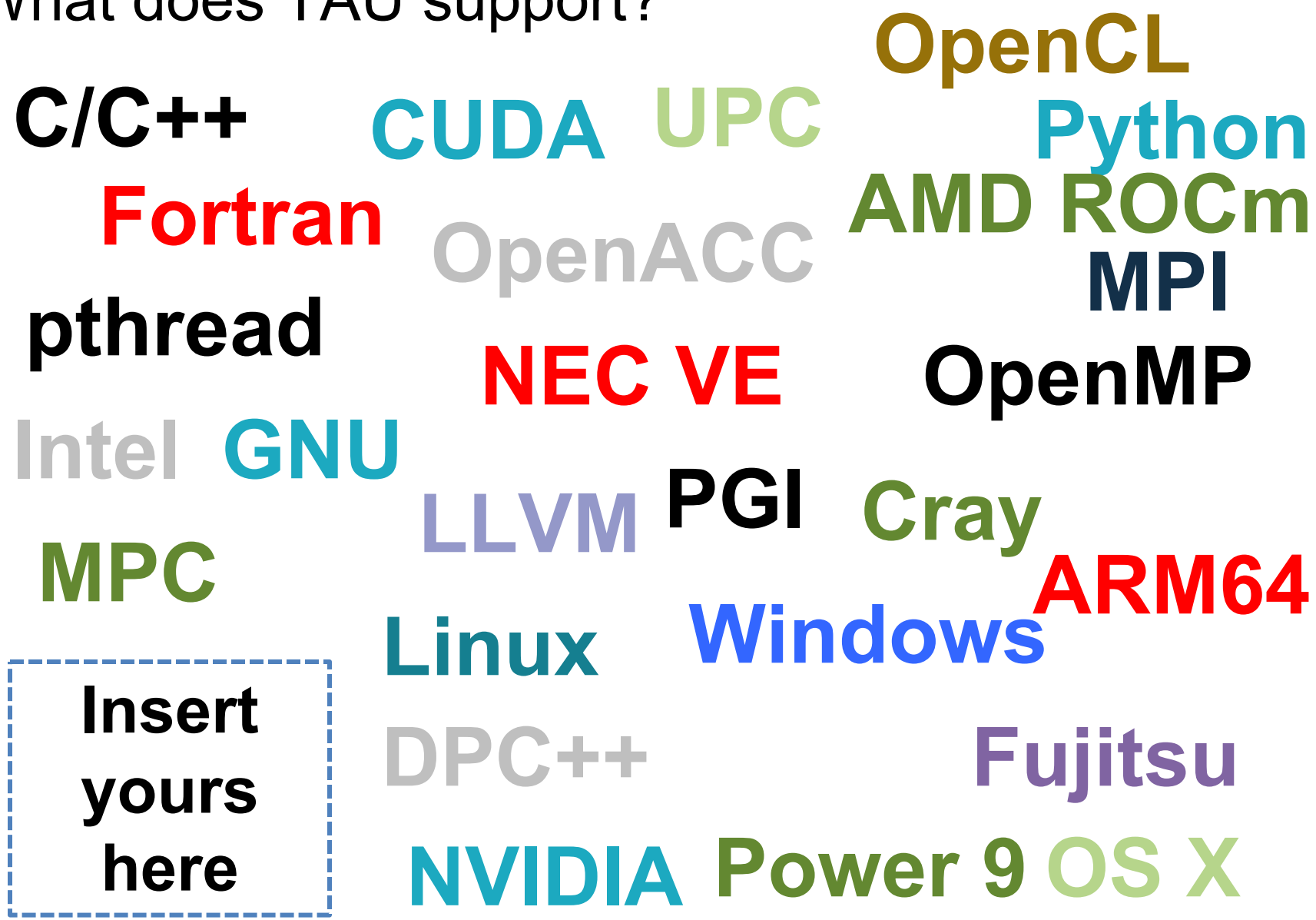
Measurement and analysis support
- MPI, OpenSHMEM, ARMCI, PGAS, DMAPP
- pthreads, OpenMP, OMPT interface, hybrid, other thread models
- GPU, ROCm, CUDA, OpenCL, OpenACC
- Parallel profiling and tracing

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

Application Performance Engineering using TAU

- How much time is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*? What is the time spent in OpenMP loops? In kernels on GPUs. How long did it take to transfer data between host and device (GPU)?
- How many instructions are executed in these code regions?
  Floating point, Level 1 and 2 *data cache misses*, hits, branches taken? What is the extent of vectorization for loops?
- 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?

What does TAU support?



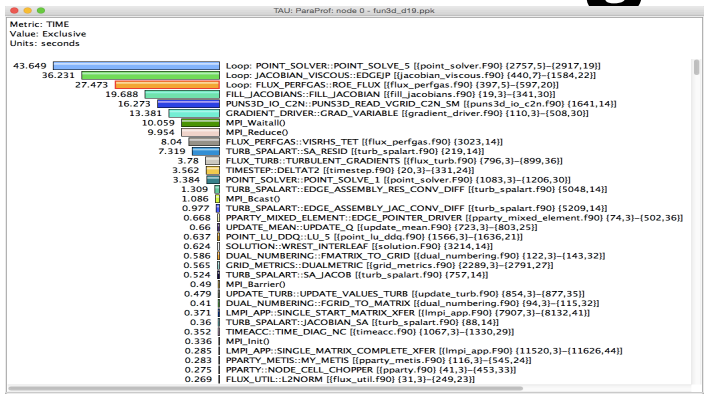
Instrumentation

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.

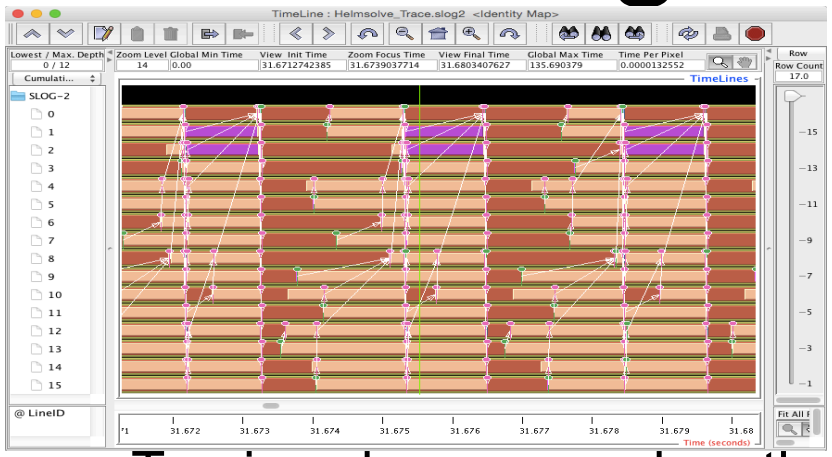
Profiling and Tracing

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

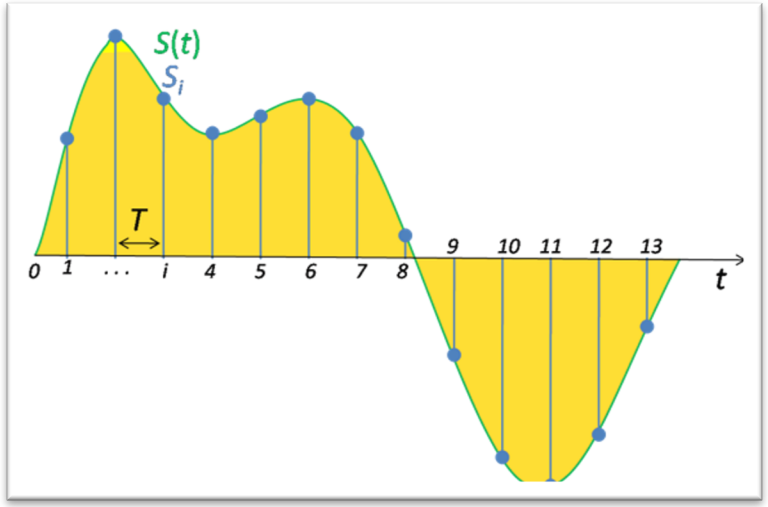
Performance Data Measurement

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**)

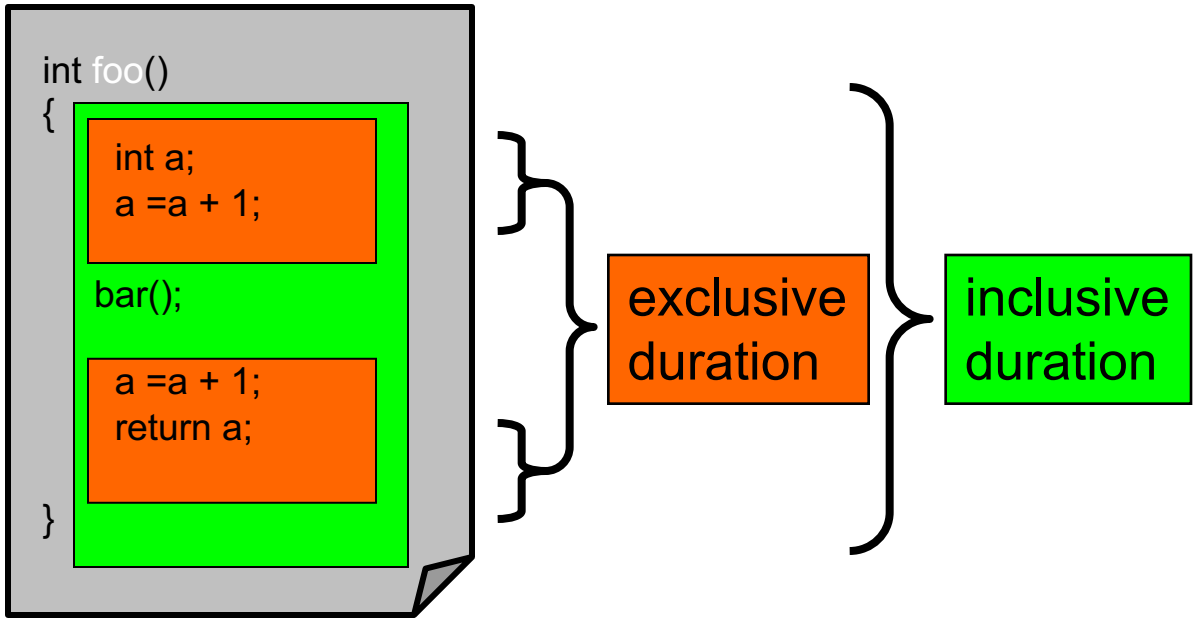
Instrumentation

- Direct and indirect performance observation
- Instrumentation invokes performance measurement
- Direct measurement with *probes*
- Indirect measurement with periodic sampling or hardware performance counter overflow interrupts
- Events measure performance data, metadata, context, etc.

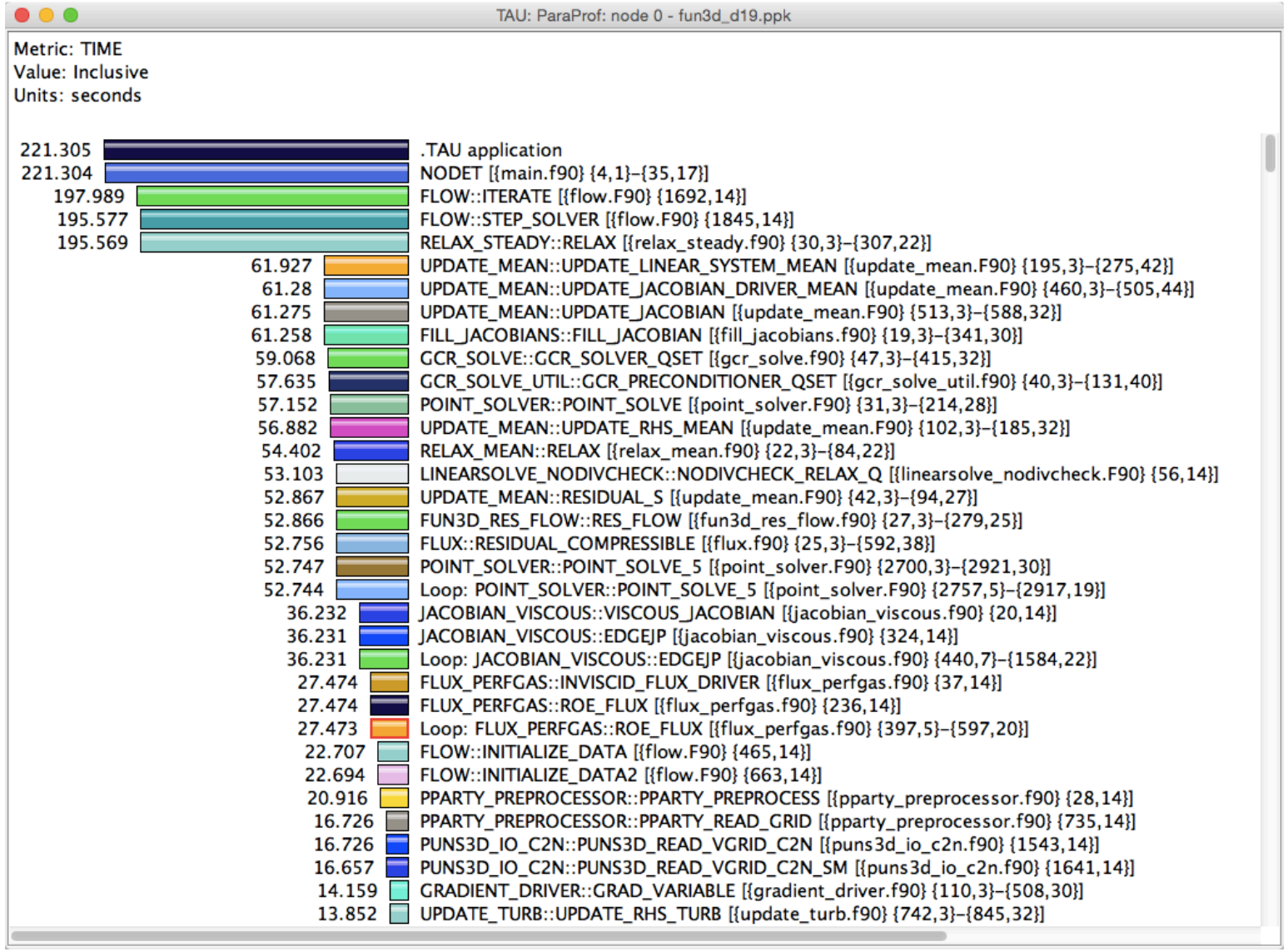
- User-defined events
  - ***Interval*** (start/stop) events to measure exclusive & inclusive duration
  - ***Atomic events*** take measurements at a single point
    - Measures total, samples, min/max/mean/std. deviation statistics
  - ***Context events*** are atomic events with executing context
    - Measures above statistics for a given calling path

Inclusive vs. Exclusive Measurements

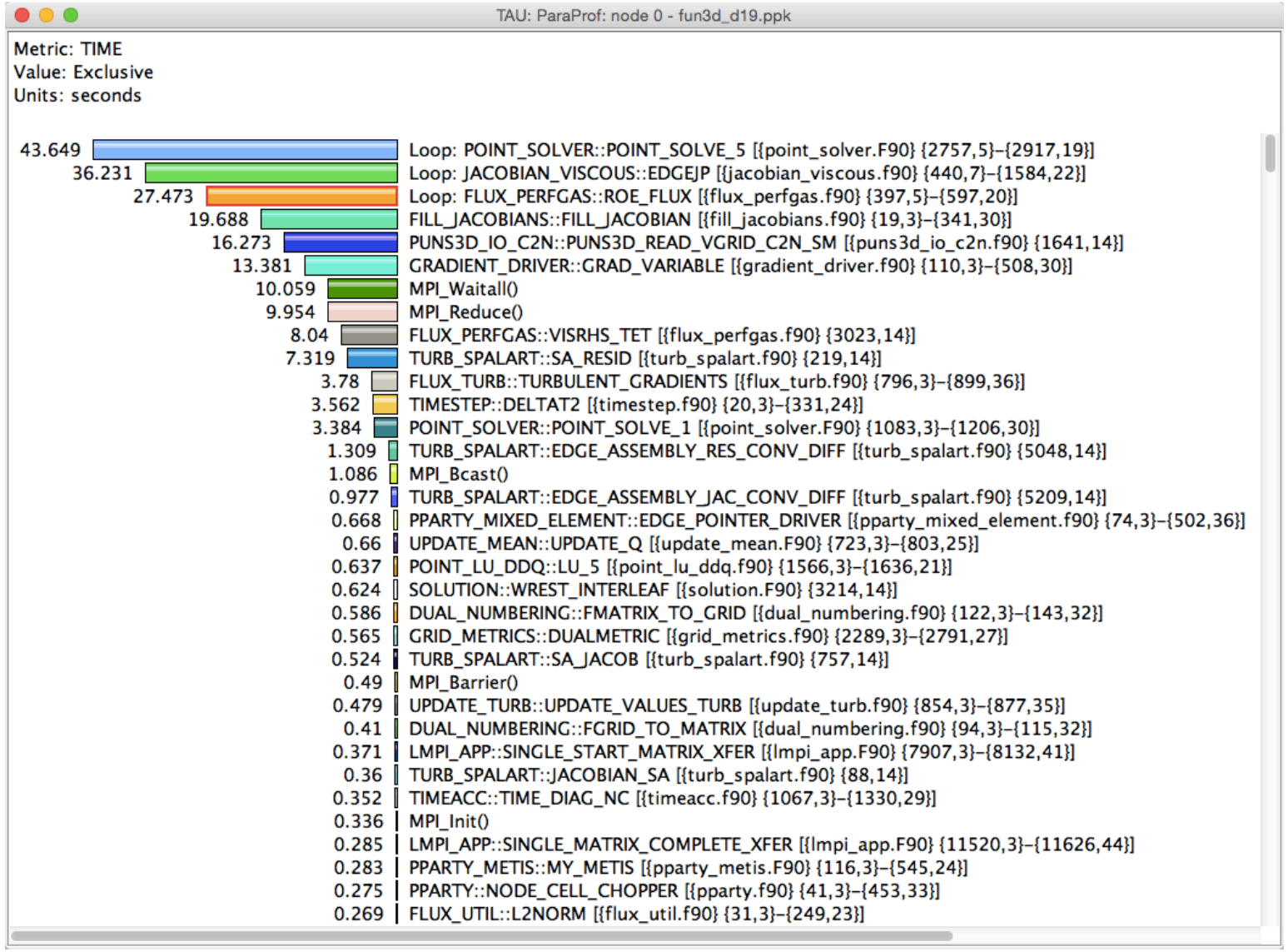
- Performance with respect to code regions
- Exclusive measurements for region only
- Inclusive measurements includes child regions



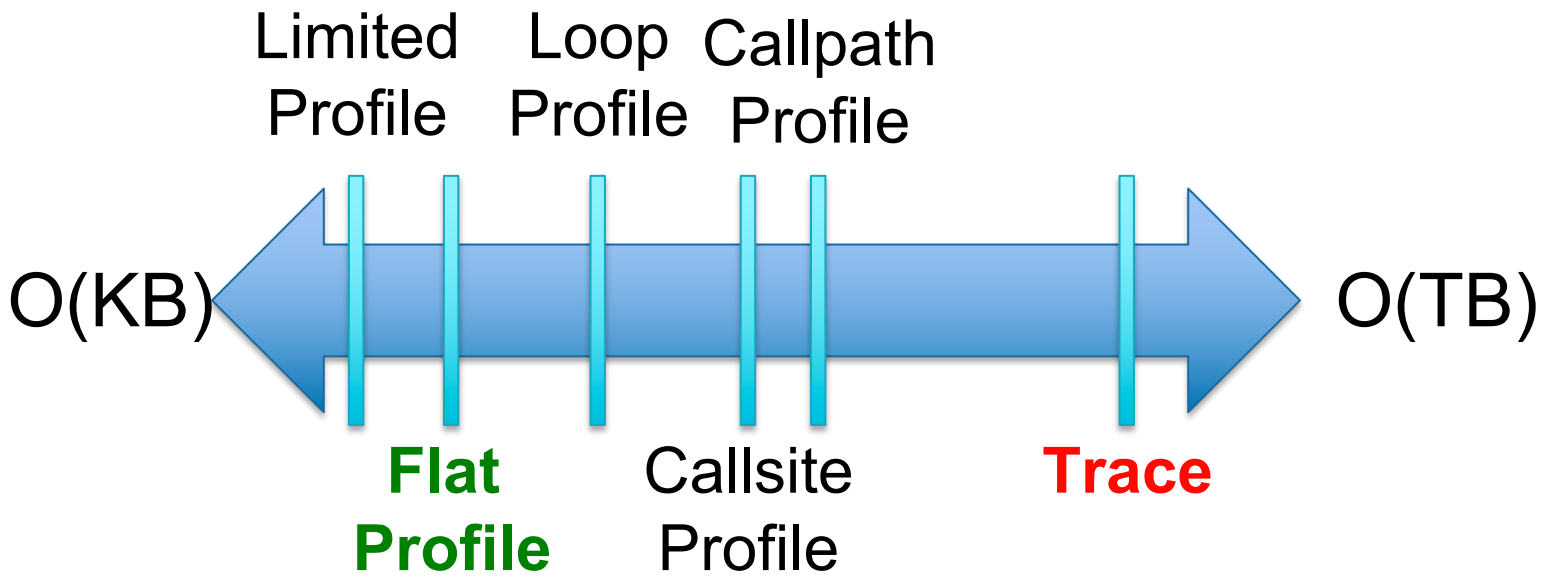
Inclusive Measurements



Exclusive Time



How much data do you want?



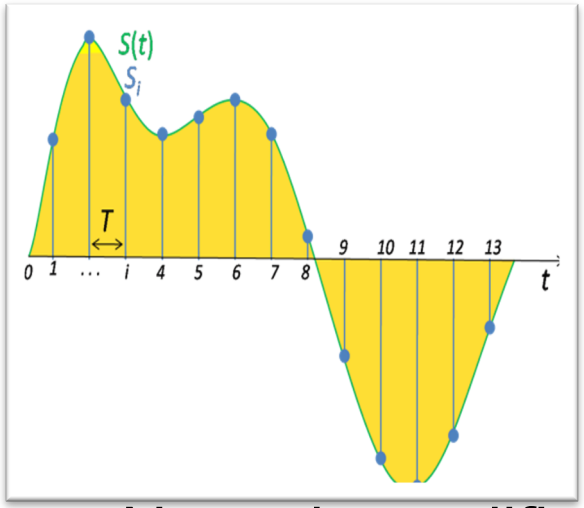
Performance Data Measurement

Direct via Probes

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

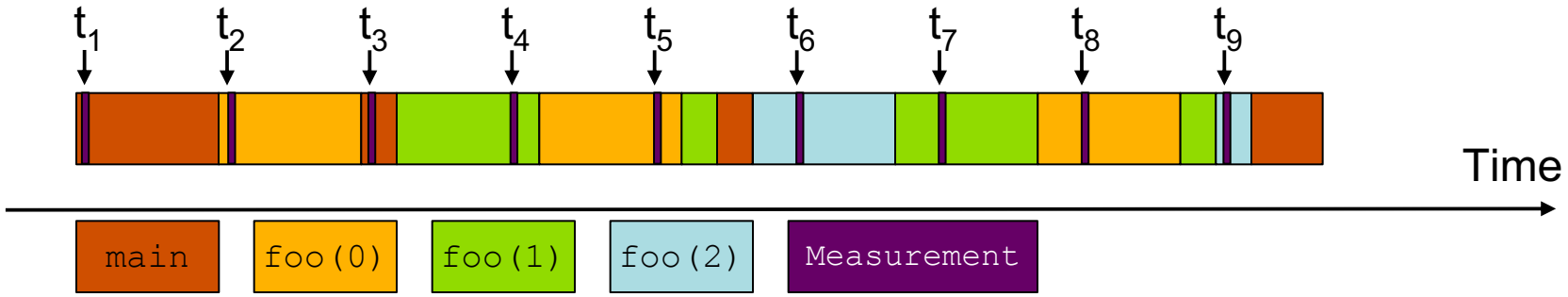
- Exact measurement
- Fine-grain control
- Calls inserted into code

Indirect via Sampling



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

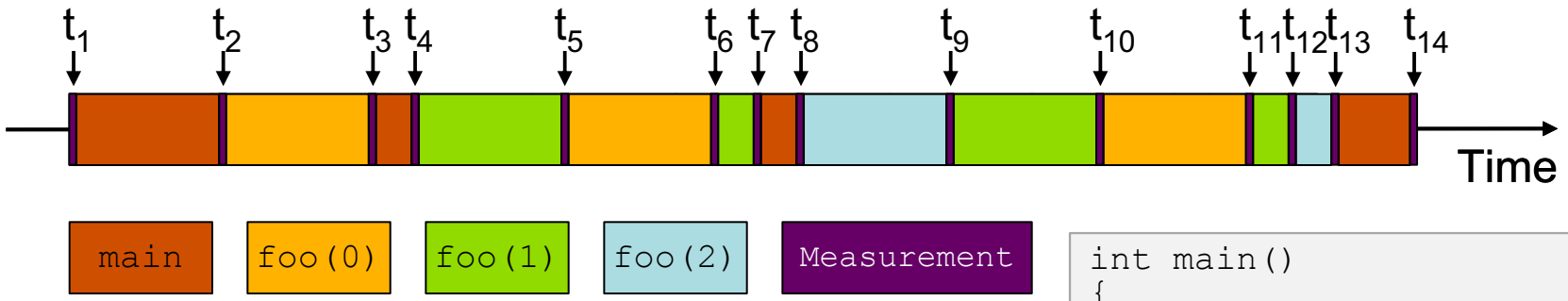
Sampling



- 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");
}
```

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 Hands-On: Quickstart Guide

Setup:

- `% module load tau ; tar zxf ~sameer/workshop.tgz; cd workshop; cat README`

Profiling with an un-instrumented application:

`MPI: % mpirun -np 64 tau_exec -T <tag> -ebs ./a.out`

- `MPI+OpenMP with Clang 9+: % 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`
- `Python+MPI+CUDA+Sampling: % mpirun -np 64 tau_python -cupti -ebs ./a.py`
- `C+CUDA (no MPI): % tau_exec -T cupti,serial -cupti ./a.out`

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

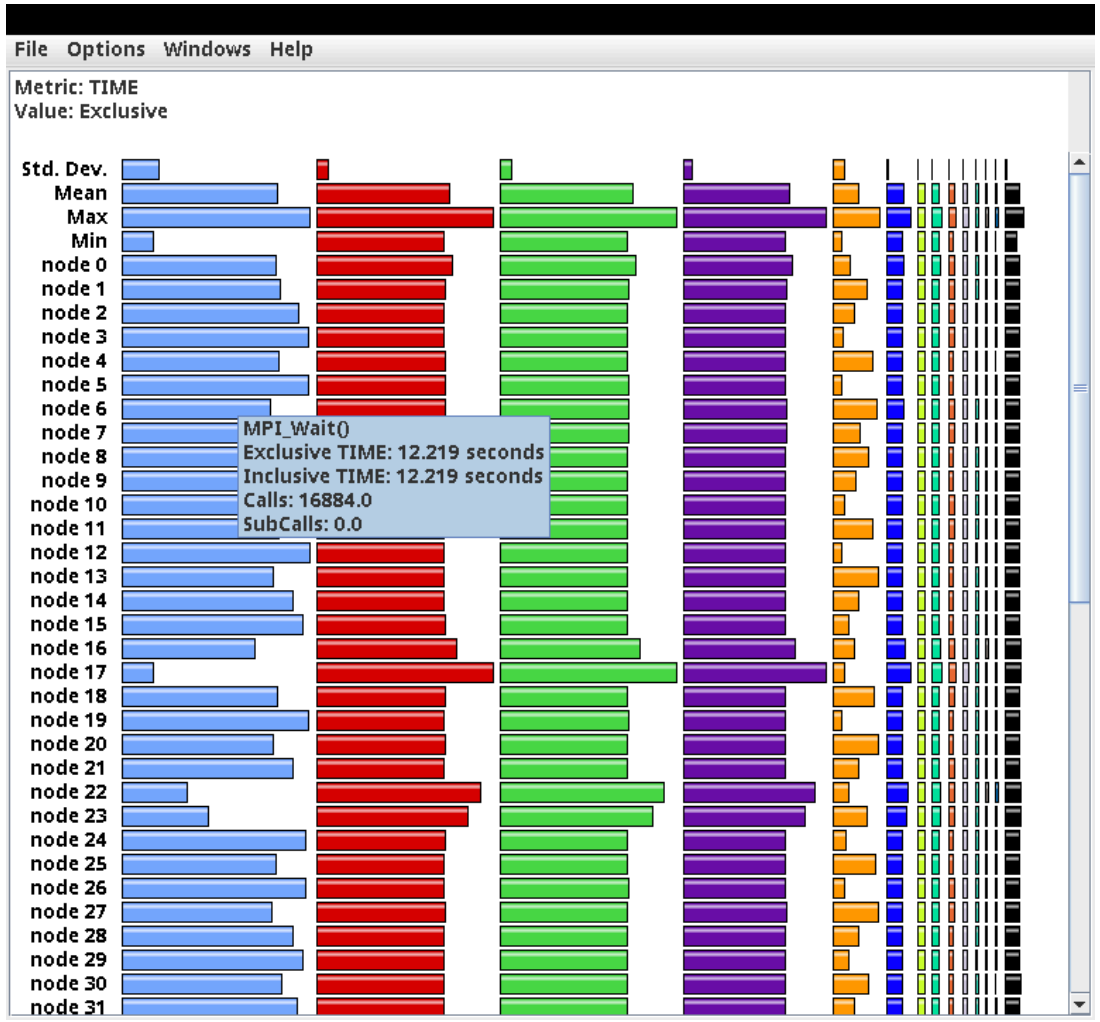
Tracing:

- `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)`
- `Jumpshot: % export TAU_TRACE=1; mpirun -np 64 tau_exec ./a.out; tau_treemerge.pl;`
  `% tau2slog2 tau.trc tau.edf -o app.slog2; jumpshot app.slog2 &`

ParaProf Profile Browser

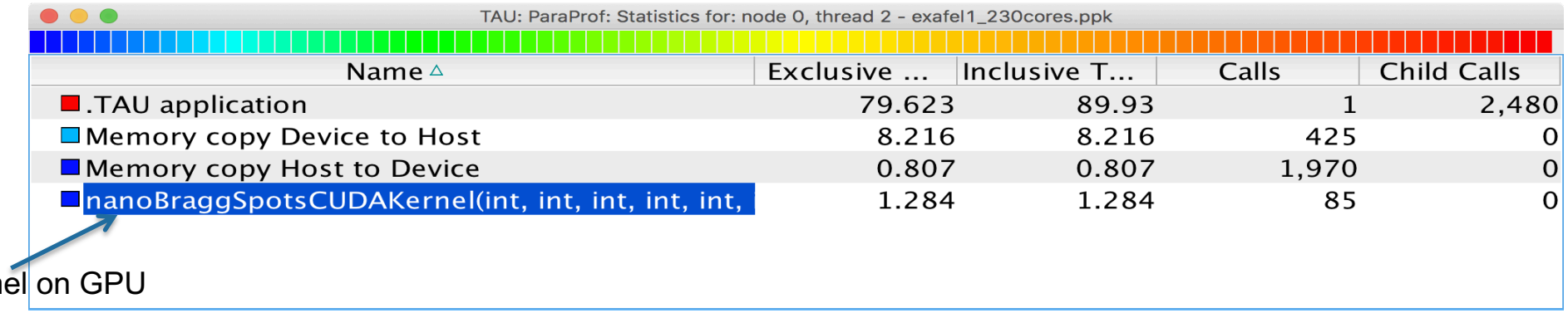
```
% paraprof
```

ParaProf Profile Browser



TAU supports Python, MPI, and CUDA

Without any modification to the source code or DSOs or interpreter, it instruments and samples the application using Python, MPI, and CUDA instrumentation.



% mpirun –np 230 tau_python –T cupti,mpi,pdt –ebs –cupti ./exafel.py
Instead of:
% mpirun –np 230 python ./exafel.py

TAU Thread Statistics Table

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

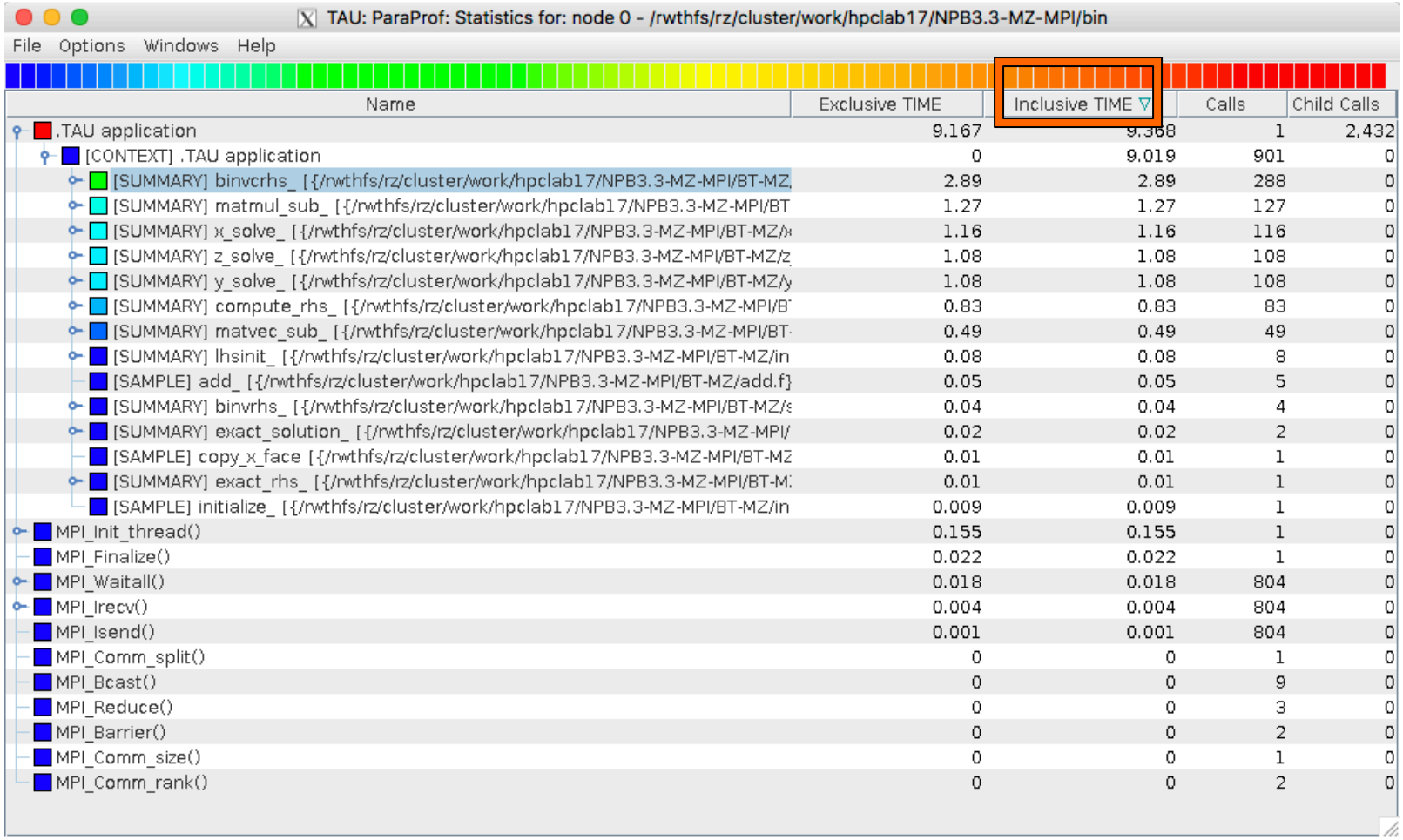
Name	Exclusive...	Inclusive ...	Calls	Child Calls
__init__ [{from_scatterers_fft.py}{13}]	20.036	20.362	303	10,914
run_sim2smv [{step5_pad.py}{138}]	16.78	134.9	1	1,066
__init__ [{__init__.py}{150}]	11.669	15.909	101	1,010
channel_pixels [{step5_pad.py}{79}]	11.029	107.657	100	13,358
[CONTEXT] channel_pixels [{step5_pad.py}{79}]	0	9.345	312	0
[SAMPLE] nanoBraggSpotsCUDA [{/autofs/nccs-svm1_home1/iris/adse13_161/psana-legion/simtbx/sun	4.755	4.755	159	0
[SAMPLE] simtbx::nanoBragg::nanoBragg::add_nanoBragg_spots_cuda() [{/autofs/nccs-svm1_home1/iris/	4.08	4.08	136	0
[SAMPLE] __memset_power8 [{} {0}]	0.3	0.3	10	0
[SAMPLE] UNRESOLVED /usr/lib64/libc-2.17.so	0.181	0.181	6	0
[SUMMARY] Tau_handle_driver_api_memcpy(void*, CUpti_CallbackDomain, unsigned int, CUpti_CallbackDa	0.03	0.03	1	0
cuMemcpyDtoH_v2	9.483	9.483	500	0
expand_to_p1_iselection [{__init__.py}{1376}]	7.349	7.35	101	606
load	7.004	7.009	2	2,251
reset_wavelength [{util_fmodel.py}{121}]	6.197	6.553	100	47,550
is_unique_set_under_symmetry [{__init__.py}{790}]	5.913	5.915	202	808
__import__	5.782	15.766	382	78
fp_fdp_at_wavelength [{fdp_plot.py}{44}]	5.616	5.723	800	1,600
MPI_Init_thread()	4.987	4.987	1	0
cuDevicePrimaryCtxRetain	4.735	4.735	2	0
<module> [{__init__.py}{1}]	4.255	23.888	85	756
MPI_Finalize()	3.829	3.829	1	1
match_bijvoet_mates [{__init__.py}{1032}]	3.146	3.684	101	707
bcast	3.073	3.448	1	9
__init__ [{__init__.py}{20}]	3.011	3.399	101	149,196
compute_f_mask [{__init__.py}{299}]	2.897	18.853	101	707

Python, MPI, CUDA, and samples from DSOs are all integrated in a single view

ParaProf

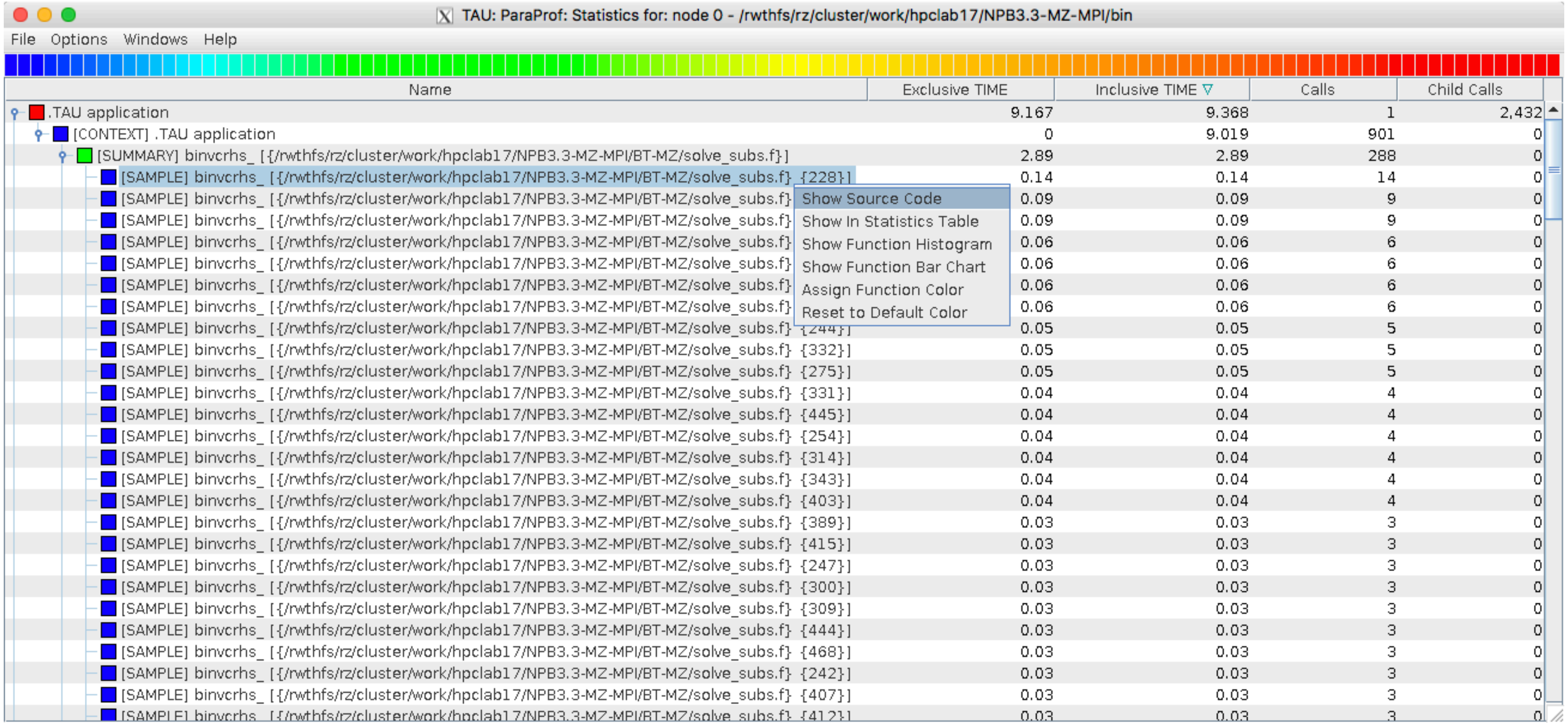
Click on Columns:
to sort by incl time

Open binvcrhs
Click on Sample



Name	Exclusive TIME	Inclusive TIME ▽	Calls	Child Calls
.TAU application	9.167	9.368	1	2,432
[CONTEXT] .TAU application	0	9.019	901	0
[SUMMARY] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ	2.89	2.89	288	0
[SUMMARY] matmul_sub_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT	1.27	1.27	127	0
[SUMMARY] x_solve_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/x	1.16	1.16	116	0
[SUMMARY] z_solve_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/z	1.08	1.08	108	0
[SUMMARY] y_solve_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/y	1.08	1.08	108	0
[SUMMARY] compute_rhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/B	0.83	0.83	83	0
[SUMMARY] matvec_sub_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT	0.49	0.49	49	0
[SUMMARY] lhsinit_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/in	0.08	0.08	8	0
[SAMPLE] add_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/add.f}	0.05	0.05	5	0
[SUMMARY] binvrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/s	0.04	0.04	4	0
[SUMMARY] exact_solution_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/	0.02	0.02	2	0
[SAMPLE] copy_x_face [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ	0.01	0.01	1	0
[SUMMARY] exact_rhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-M	0.01	0.01	1	0
[SAMPLE] initialize_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/in	0.009	0.009	1	0
MPI_Init_thread()	0.155	0.155	1	0
MPI_Finalize()	0.022	0.022	1	0
MPI_Waitall()	0.018	0.018	804	0
MPI_Irecv()	0.004	0.004	804	0
MPI_Isend()	0.001	0.001	804	0
MPI_Comm_split()	0	0	1	0
MPI_Bcast()	0	0	9	0
MPI_Reduce()	0	0	3	0
MPI_Barrier()	0	0	2	0
MPI_Comm_size()	0	0	1	0
MPI_Comm_rank()	0	0	2	0

ParaProf



Name	Exclusive TIME	Inclusive TIME	Calls	Child Calls
.TAU application	9.167	9.368	1	2,432
[CONTEXT] .TAU application	0	9.019	901	0
[SUMMARY] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}]	2.89	2.89	288	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {228}]	0.14	0.14	14	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}	0.09	0.09	9	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}	0.09	0.09	9	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}	0.06	0.06	6	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}	0.06	0.06	6	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}	0.06	0.06	6	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f}	0.06	0.06	6	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {244}]	0.05	0.05	5	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {332}]	0.05	0.05	5	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {275}]	0.05	0.05	5	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {331}]	0.04	0.04	4	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {445}]	0.04	0.04	4	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {254}]	0.04	0.04	4	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {314}]	0.04	0.04	4	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {343}]	0.04	0.04	4	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {403}]	0.04	0.04	4	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {389}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {415}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {247}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {300}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {309}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {444}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {468}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {242}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {407}]	0.03	0.03	3	0
[SAMPLE] binvcrhs_ [{/rwthfs/rz/cluster/work/hpclab17/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f} {412}]	0.03	0.03	3	0

TAU Context Event Window

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

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

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

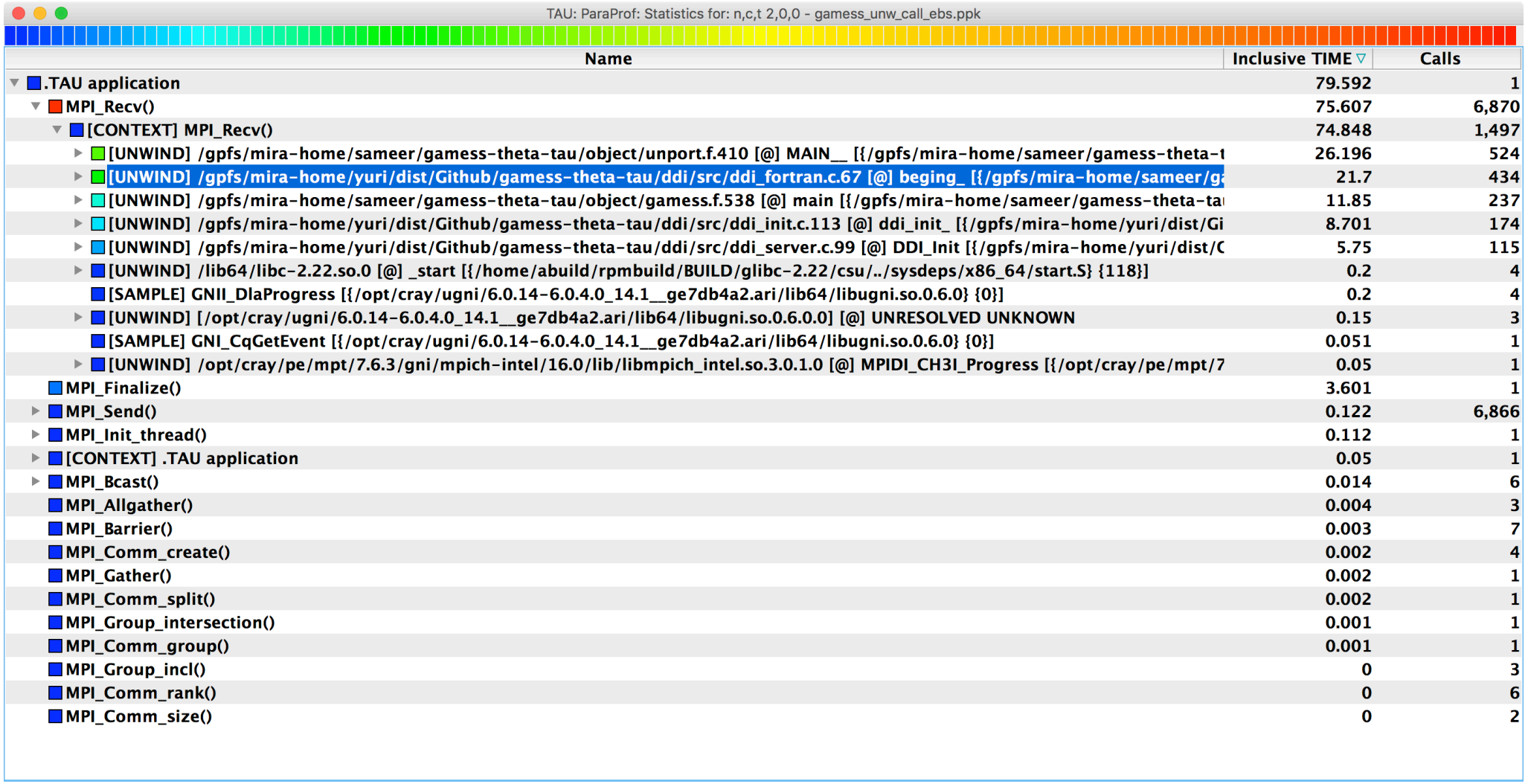
TAU's tracking of Python and MPI

TAU: ParaProf: Statistics for: node 1, thread 0 - exafel1_230cores.ppk

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

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

Callstack Sampling in TAU



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

% export TAU_SAMPLING=1; export TAU_EBS_UNWIND=1

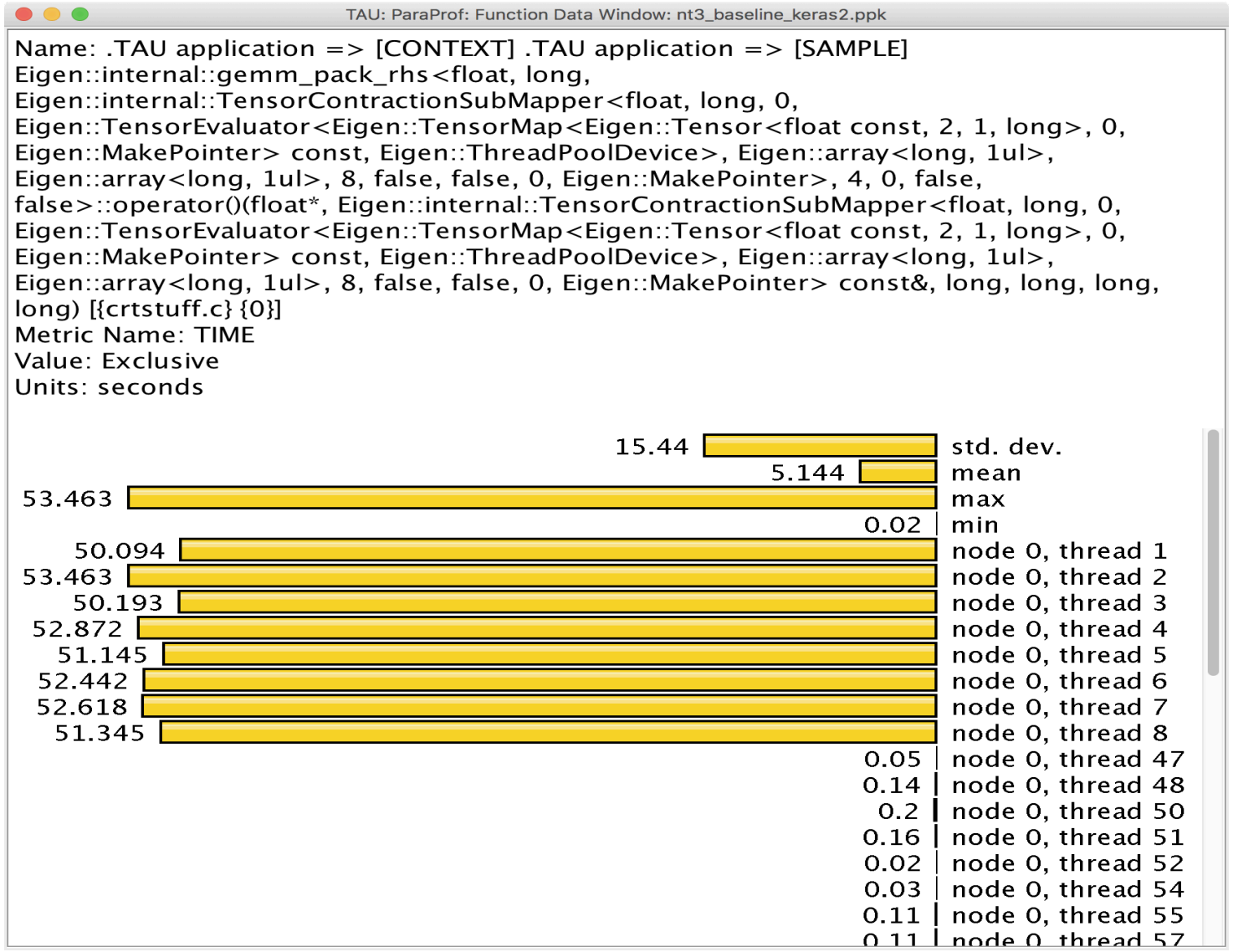
Deep Learning: Tensorflow

TAU: ParaProf: Statistics for: node 0, thread 8 - nt3_baseline_keras2.ppk

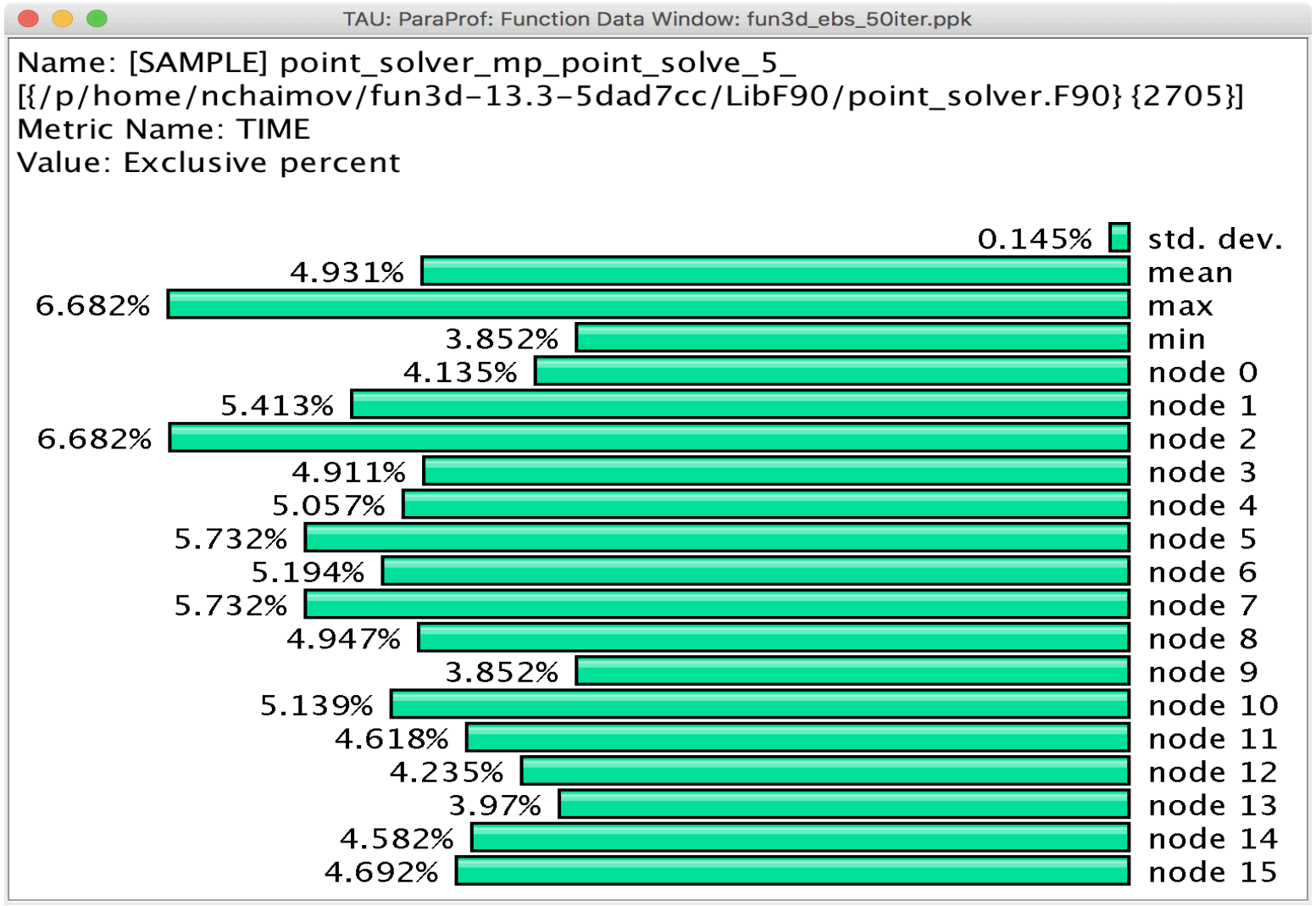
Name	Inclusiv...	Calls
.TAU application	519.211	1
[CONTEXT] .TAU application	509.222	50,915
[SAMPLE] Eigen::internal::gebp_kernel<float, float, long, Eigen::internal::blas_data_mapper<float, long, 0, 0>,	240.632	24,089
[SAMPLE] __pthread_cond_wait [{} {0}]	86.384	8,634
[SAMPLE] Eigen::internal::gemm_pack_rhs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

% tau_python –ebs nt3_baseline_keras2.py (CANDLE)

Sampling Tensorflow



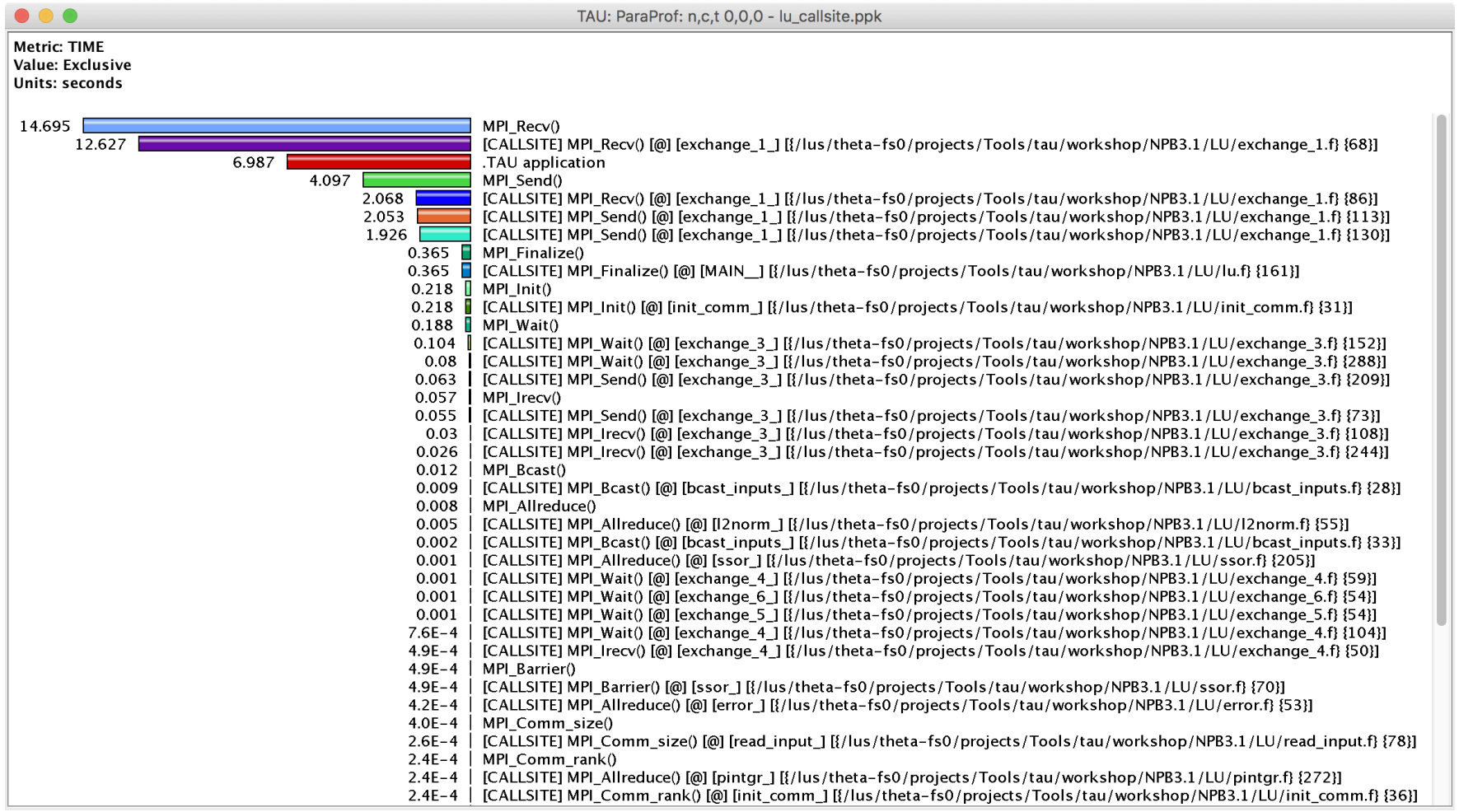
Event Based Sampling (EBS)



Uninstrumented!

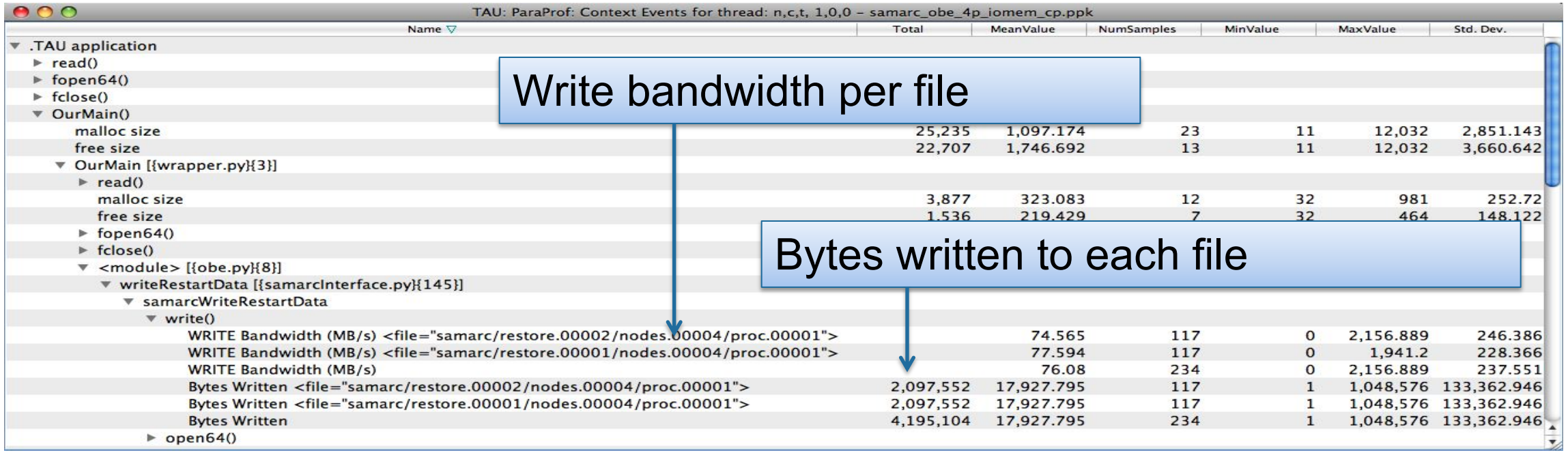
```
% mpirun -n 16 tau_exec –ebs a.out
```

Callsite Profiling and Tracing



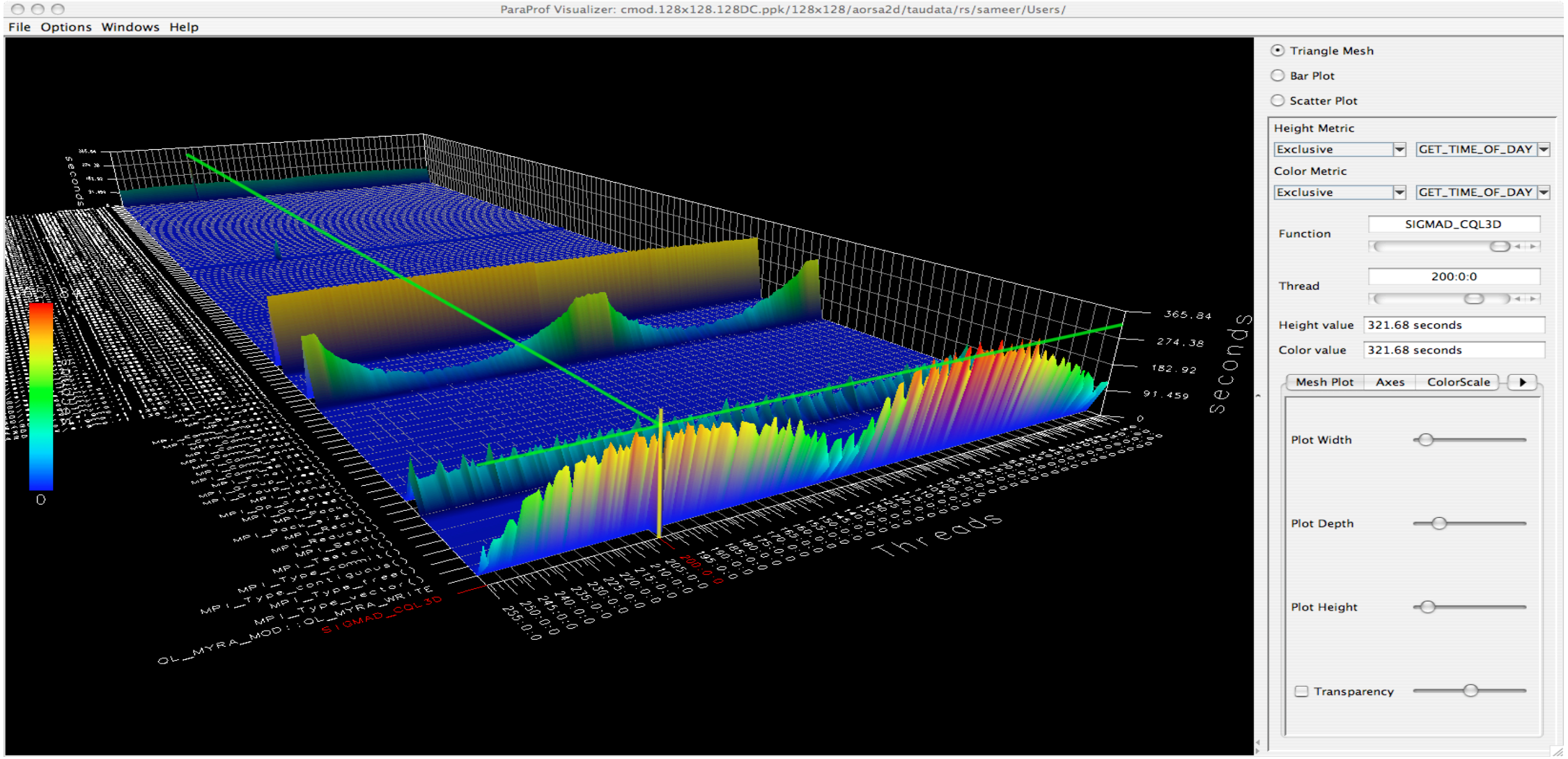
% export TAU_CALLSITE=1

TAU – Context Events

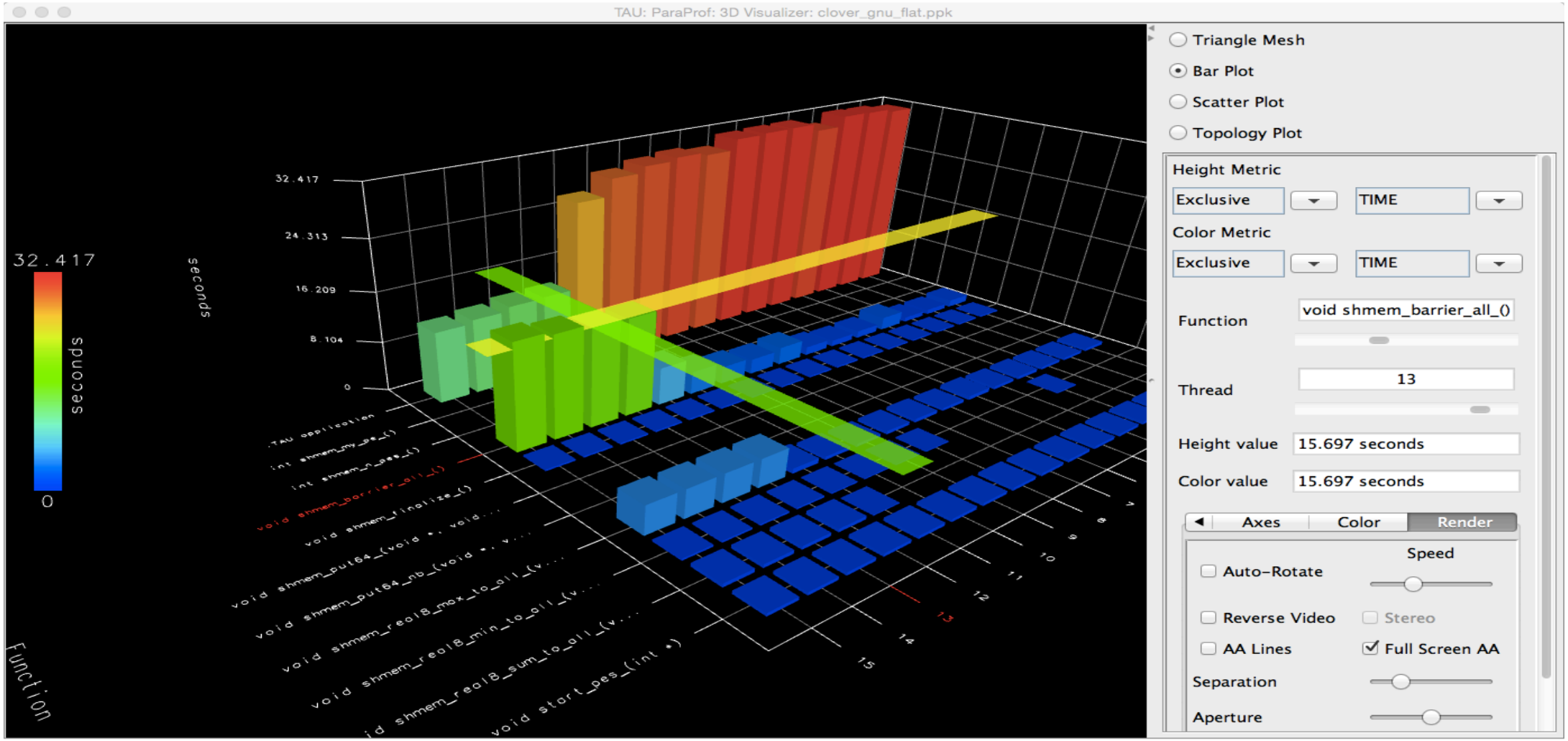


Name	Total	MeanValue	NumSamples	MinValue	MaxValue	Std. Dev.
.TAU application						
read()						
fopen64()						
fclose()						
OurMain()						
malloc size	25,235	1,097.174	23	11	12,032	2,851.143
free size	22,707	1,746.692	13	11	12,032	3,660.642
OurMain [{wrapper.py}{3}]						
read()						
malloc size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

ParaProf 3D Profile Browser

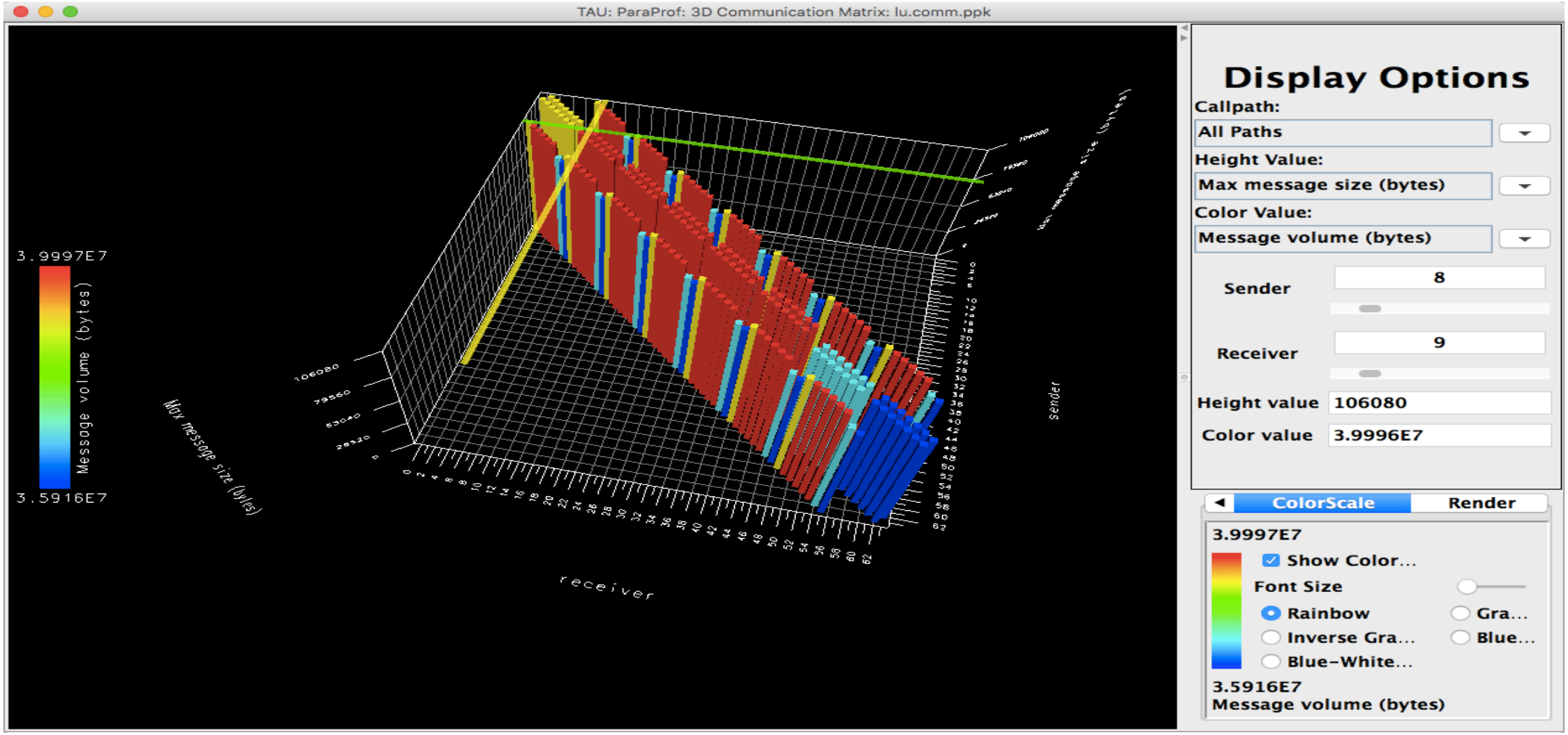


TAU – ParaProf 3D Visualization



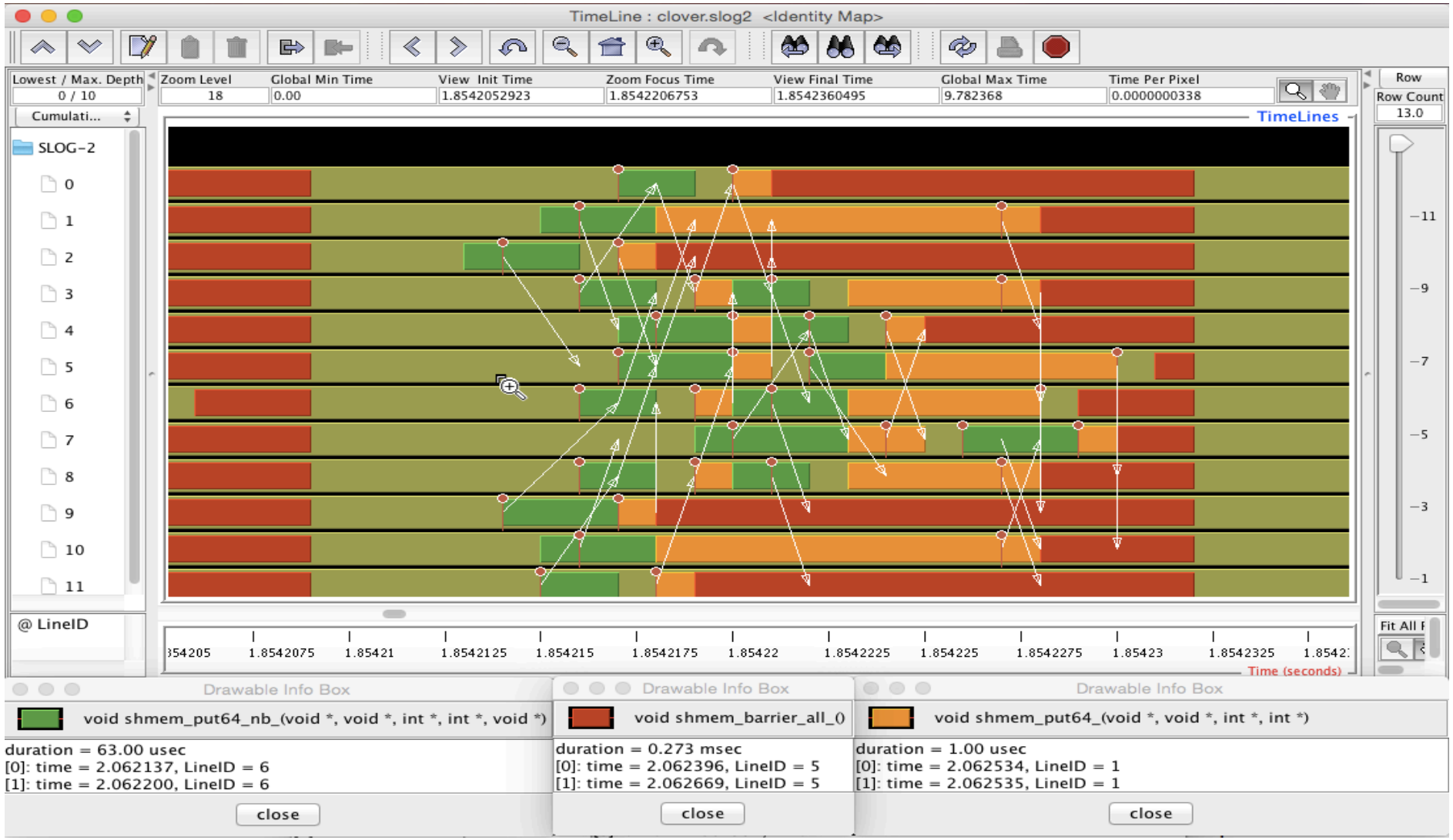
% paraprof app.ppk
Windows -> 3D Visualization -> Bar Plot (right pane)

TAU – 3D Communication Window

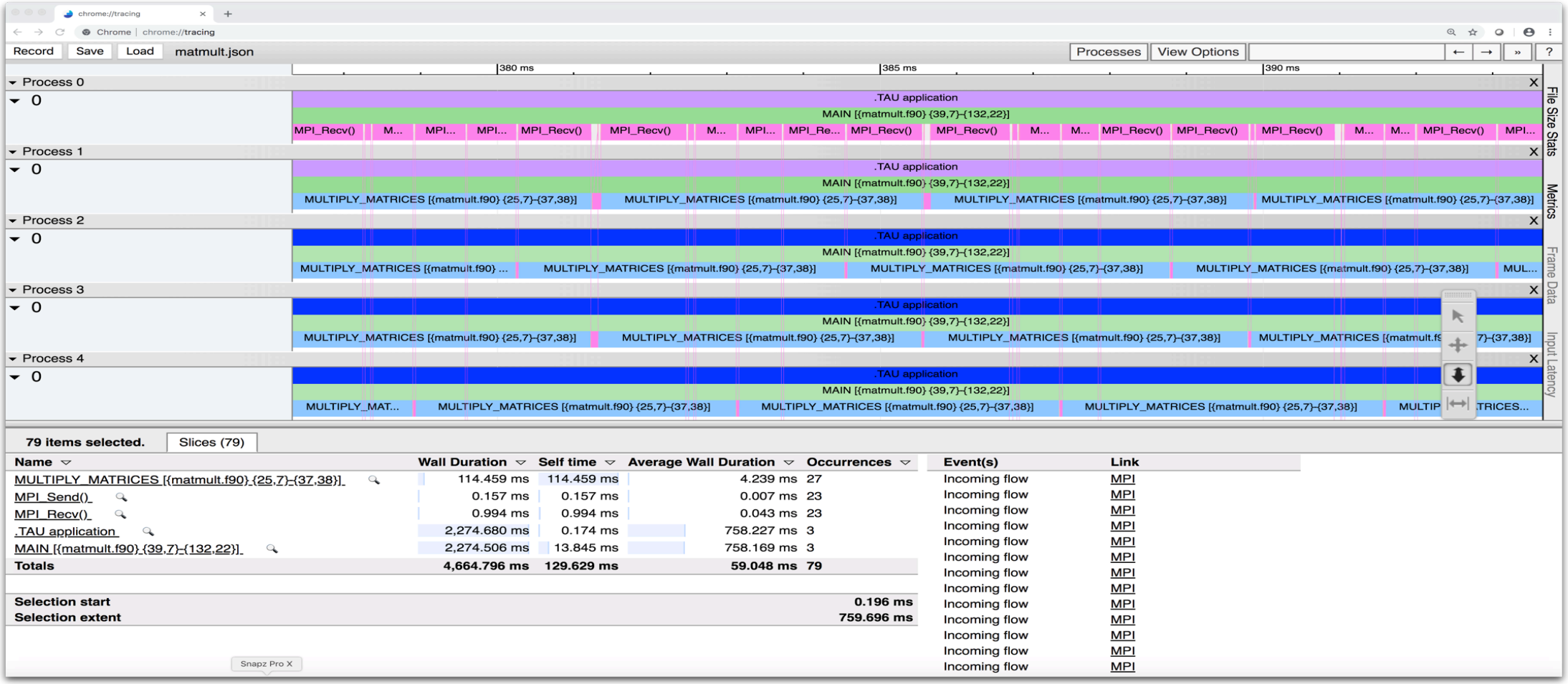


% export TAU_COMM_MATRIX=1; mpirun … tau_exec ./a.out
% paraprof ; Windows -> 3D Communication Matrix

Tracing: Jumpshot (ships with TAU)



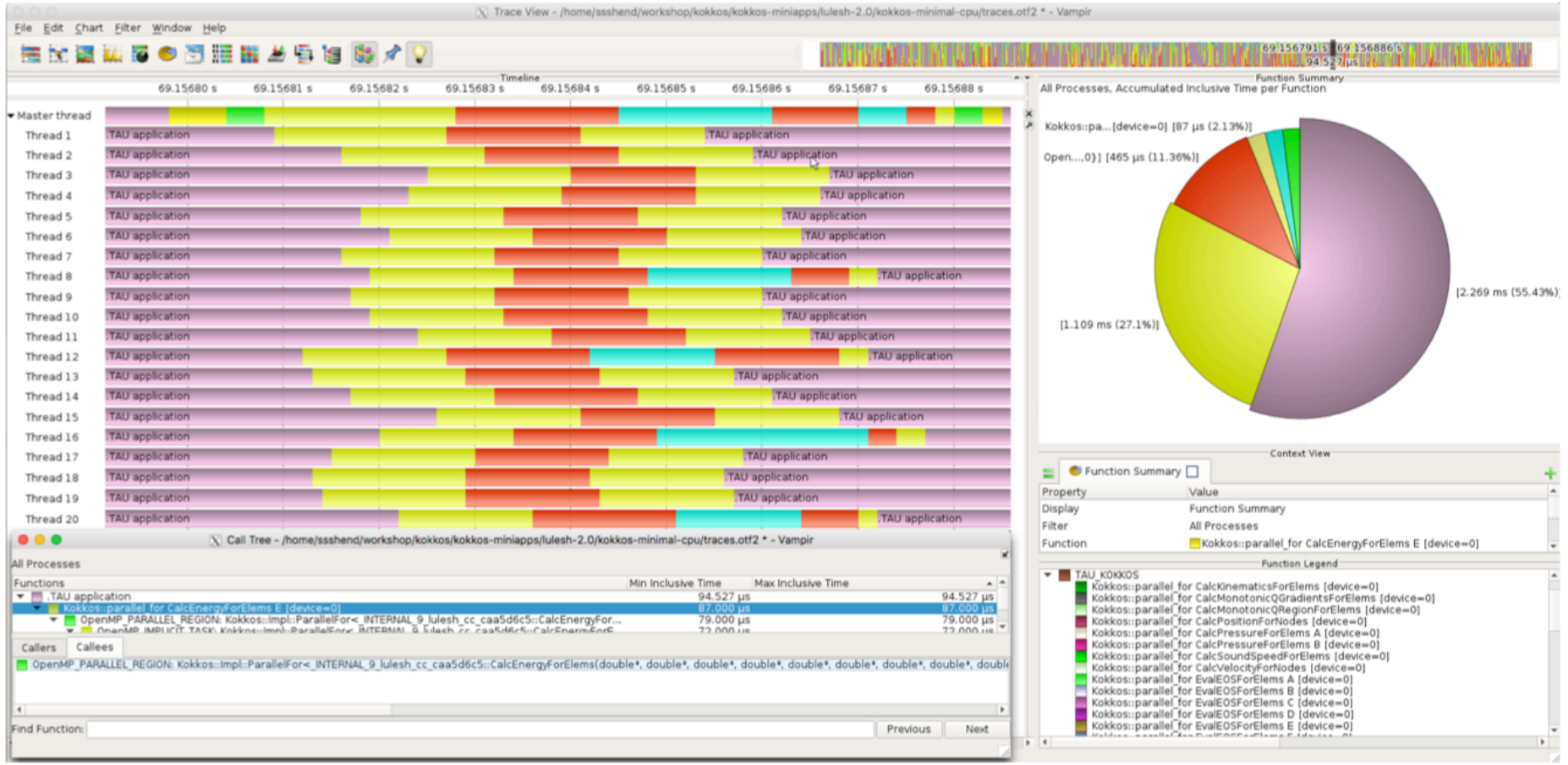
Tracing: Chrome Browser



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

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

Vampir [TU Dresden] Timeline: Kokkos




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

Kokkos

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

Kokkos API use in ExaMiniMD

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

```
void CommMPI::update_halo() {

  Kokkos::Profiling::pushRegion("Comm::update_halo");

  N_ghost = 0;
  s=*system;

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

  for(phase = 0; phase<6; phase++) {
    pack_indicies = Kokkos::subview(pack_indicies_all,phase,Kokkos::ALL());
    if(proc_grid[phase/2]>1) {

      Kokkos::parallel_for("CommMPI::halo_update_pack",
         Kokkos::RangePolicy<TagHaloUpdatePack, Kokkos::IndexType<T_INT> >(0,proc_num_send[phase]),
         *this);
      MPI_Request request;
      MPI_Status status;
      MPI_Irecv(unpack_buffer.data(),proc_num_recv[phase]*sizeof(T_X_FLOAT)*3/sizeof(int),MPI_INT, proc_neighbors_recv[phase],100002,MPI_COMM_WORLD,&request);
      MPI_Send (pack_buffer.data(),proc_num_send[phase]*sizeof(T_X_FLOAT)*3/sizeof(int),MPI_INT, proc_neighbors_send[phase],100002,MPI_COMM_WORLD);
      s = *system;
      MPI_Wait(&request,&status);
      const int count = proc_num_recv[phase];
      if(unpack_buffer_update.extent(0)<count) {
        unpack_buffer_update = t_buffer_update((T_X_FLOAT*)unpack_buffer.data(),count);
      }
      Kokkos::parallel_for("CommMPI::halo_update_unpack",
                Kokkos::RangePolicy<TagHaloUpdateUnpack, Kokkos::IndexType<T_INT> >(0,proc_num_recv[phase]),
                *this);

    } else {
      //printf("HaloUpdateCopy: %i %i %i\n",phase,proc_num_send[phase],pack_indicies.extent(0));
      Kokkos::parallel_for("CommMPI::halo_update_self",
        Kokkos::RangePolicy<TagHaloUpdateSelf, Kokkos::IndexType<T_INT> >(0,proc_num_send[phase]),
        *this);
    }
    N_ghost += proc_num_recv[phase];
  }

  Kokkos::Profiling::popRegion();
};
```

pushRegion("Comm::update_halo")

Kokkos::parallel_for

popRegion

ExaMiniMD: TAU Phase

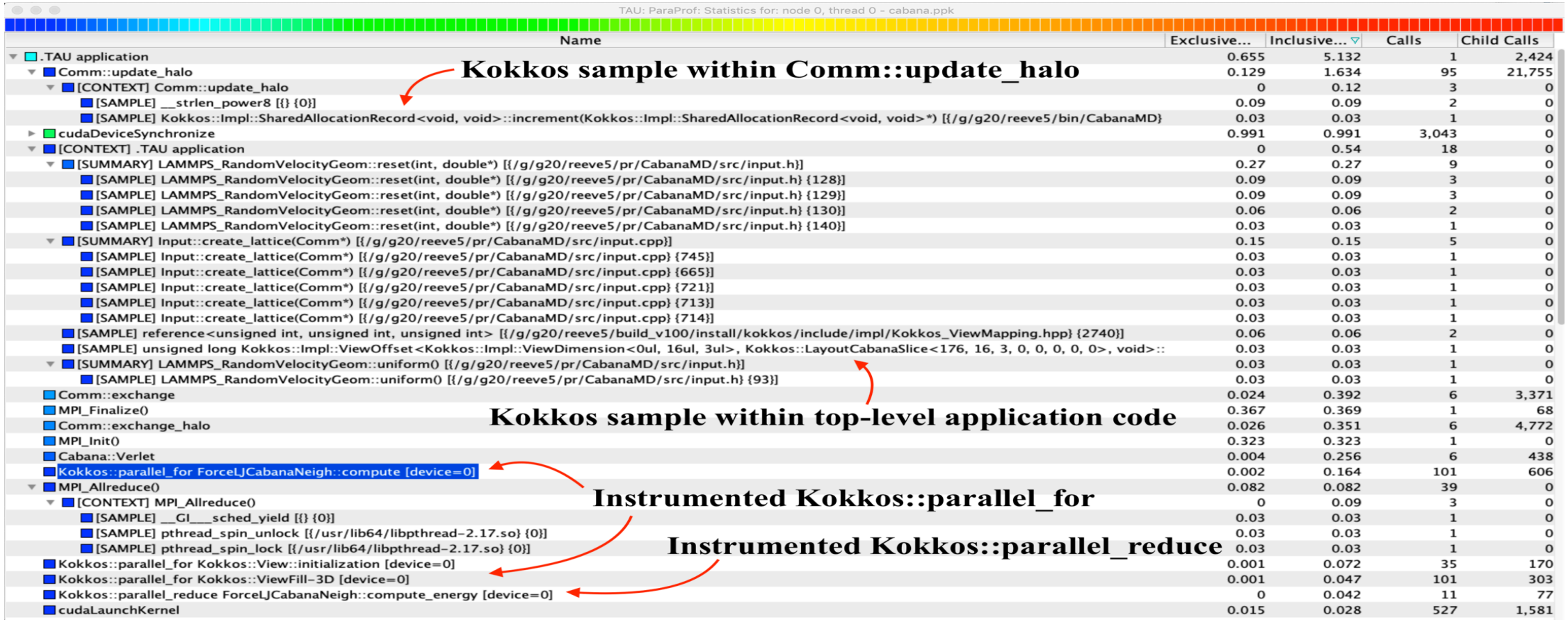
TAU: ParaProf: Statistics for: node 0, thread 0 - examinimd_ompt_phase.ppk

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

Comm::update_halo phase in TAU ParaProf's Thread Statistics Table

ExaMiniMD: ParaProf Node Window

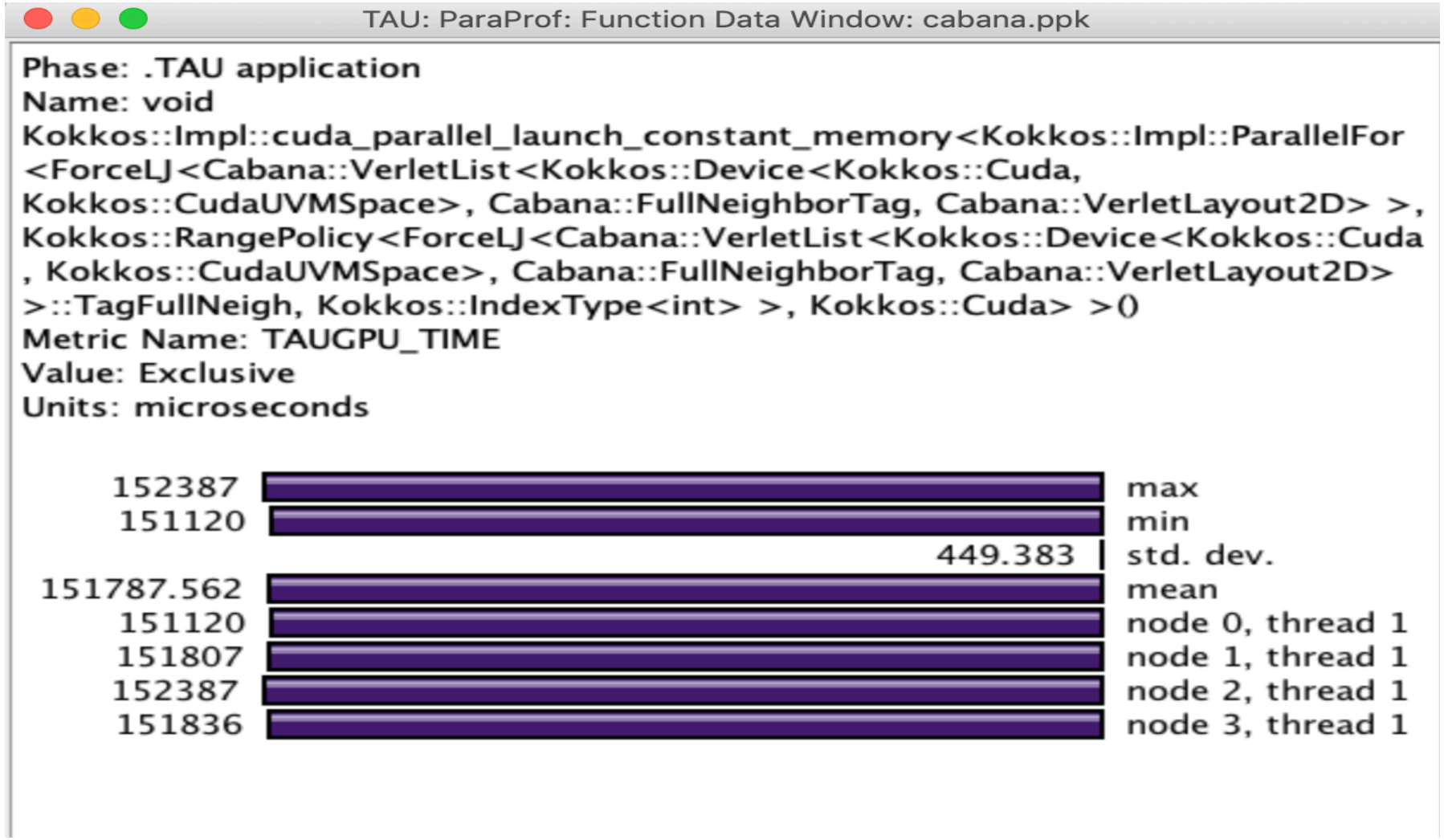
Event-based Sampling (EBS): CabanaMD on an IBM AC922 with NVIDIA V100 GPUs



Name	Exclusive...	Inclusive...	Calls	Child Calls
.TAU application	0.655	5.132	1	2,424
Comm::update_halo	0.129	1.634	95	21,755
[CONTEXT] Comm::update_halo	0	0.12	3	0
[SAMPLE] __strlen_power8 [{} {0}]	0.09	0.09	2	0
[SAMPLE] Kokkos::Impl::SharedAllocationRecord<void, void>::increment(Kokkos::Impl::SharedAllocationRecord<void, void>*) [{/g/g20/reeve5/bin/CabanaMD}	0.03	0.03	1	0
cudaDeviceSynchronize	0.991	0.991	3,043	0
[CONTEXT] .TAU application	0	0.54	18	0
[SUMMARY] LAMMPS_RandomVelocityGeom::reset(int, double*) [{/g/g20/reeve5/pr/CabanaMD/src/input.h}]	0.27	0.27	9	0
[SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [{/g/g20/reeve5/pr/CabanaMD/src/input.h} {128}]	0.09	0.09	3	0
[SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [{/g/g20/reeve5/pr/CabanaMD/src/input.h} {129}]	0.09	0.09	3	0
[SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [{/g/g20/reeve5/pr/CabanaMD/src/input.h} {130}]	0.06	0.06	2	0
[SAMPLE] LAMMPS_RandomVelocityGeom::reset(int, double*) [{/g/g20/reeve5/pr/CabanaMD/src/input.h} {140}]	0.03	0.03	1	0
[SUMMARY] Input::create_lattice(Comm*) [{/g/g20/reeve5/pr/CabanaMD/src/input.cpp}]	0.15	0.15	5	0
[SAMPLE] Input::create_lattice(Comm*) [{/g/g20/reeve5/pr/CabanaMD/src/input.cpp} {745}]	0.03	0.03	1	0
[SAMPLE] Input::create_lattice(Comm*) [{/g/g20/reeve5/pr/CabanaMD/src/input.cpp} {665}]	0.03	0.03	1	0
[SAMPLE] Input::create_lattice(Comm*) [{/g/g20/reeve5/pr/CabanaMD/src/input.cpp} {721}]	0.03	0.03	1	0
[SAMPLE] Input::create_lattice(Comm*) [{/g/g20/reeve5/pr/CabanaMD/src/input.cpp} {713}]	0.03	0.03	1	0
[SAMPLE] Input::create_lattice(Comm*) [{/g/g20/reeve5/pr/CabanaMD/src/input.cpp} {714}]	0.03	0.03	1	0
[SAMPLE] reference<unsigned int, unsigned int, unsigned int> [{/g/g20/reeve5/build_v100/install/kokkos/include/impl/Kokkos_ViewMapping.hpp} {2740}]	0.06	0.06	2	0
[SAMPLE] unsigned long Kokkos::Impl::ViewOffset<Kokkos::Impl::ViewDimension<0ul, 16ul, 3ul>, Kokkos::LayoutCabanaSlice<176, 16, 3, 0, 0, 0, 0, 0>, void>::	0.03	0.03	1	0
[SUMMARY] LAMMPS_RandomVelocityGeom::uniform() [{/g/g20/reeve5/pr/CabanaMD/src/input.h}]	0.03	0.03	1	0
[SAMPLE] LAMMPS_RandomVelocityGeom::uniform() [{/g/g20/reeve5/pr/CabanaMD/src/input.h} {93}]	0.03	0.03	1	0
Comm::exchange	0.024	0.392	6	3,371
MPI_Finalize()	0.367	0.369	1	68
Comm::exchange_halo	0.026	0.351	6	4,772
MPI_Init()	0.323	0.323	1	0
Cabana::Verlet	0.004	0.256	6	438
Kokkos::parallel_for ForceLJCabanaNeigh::compute [device=0]	0.002	0.164	101	606
MPI_Allreduce()	0.082	0.082	39	0
[CONTEXT] MPI_Allreduce()	0	0.09	3	0
[SAMPLE] __GI___sched_yield [{} {0}]	0.03	0.03	1	0
[SAMPLE] pthread_spin_unlock [{/usr/lib64/libpthread-2.17.so} {0}]	0.03	0.03	1	0
[SAMPLE] pthread_spin_lock [{/usr/lib64/libpthread-2.17.so} {0}]	0.03	0.03	1	0
Kokkos::parallel_for Kokkos::View::initialization [device=0]	0.001	0.072	35	170
Kokkos::parallel_for Kokkos::ViewFill-3D [device=0]	0.001	0.047	101	303
Kokkos::parallel_reduce ForceLJCabanaNeigh::compute_energy [device=0]	0	0.042	11	77
cudaLaunchKernel	0.015	0.028	527	1,581

Event-based sampling (EBS) with Kokkos API

CabanaMD: CUDA Events



TAU's Support for Runtime Systems

MPI

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

Pthread

- Captures time spent in routines per thread of execution

OpenMP

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

OpenACC

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

TAU's Support for Runtime Systems (contd.)

OpenCL
- OpenCL profiling interface
- Track timings of kernels

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

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

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

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

Examples of Multi-Level Instrumentation

MPI + OpenMP

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

MPI + CUDA

PMPI + CUPTI interfaces

OpenCL + ROCm

Rocprofiler + OpenCL instrumentation interfaces

Kokkos + OpenMP

Kokkos profiling API + OMPT to transparently track events

Kokkos + pthread + MPI

Kokkos + pthread wrapper interposition library + PMPI layer

Python + CUDA + MPI

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

MPI + OpenCL

PMPI + OpenCL profiling interfaces

TAU Execution Command (tau_exec)

Uninstrumented execution

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

Track GPU operations

```
% mpirun –np 256  tau_exec –rocm ./a.out
% mpirun –np 256  tau_exec –cupti ./a.out
% mpirun –np 256 tau_exec –opencl ./a.out
% mpirun –np 256 tau_exec –openacc ./a.out
```

Track MPI performance

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

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

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

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

```
% export TAU_OMPT_SUPPORT_LEVEL=full;
% mpirun –np 256  tau_exec –T ompt,v5,mpi  -ompt  ./a.out
```

Track memory operations

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

Use event based sampling (compile with –g)

```
% mpirun –np 256 tau_exec –ebs ./a.out
Also  -ebs_source=<PAPI_COUNTER> -ebs_period=<overflow_count>   -ebs_resolution=<file | function | line>
```

tau_exec

```
$ tau_exec

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

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

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

tau_exec
- preloads the TAU wrapper libraries and performs measurements

No need to recompile the application!

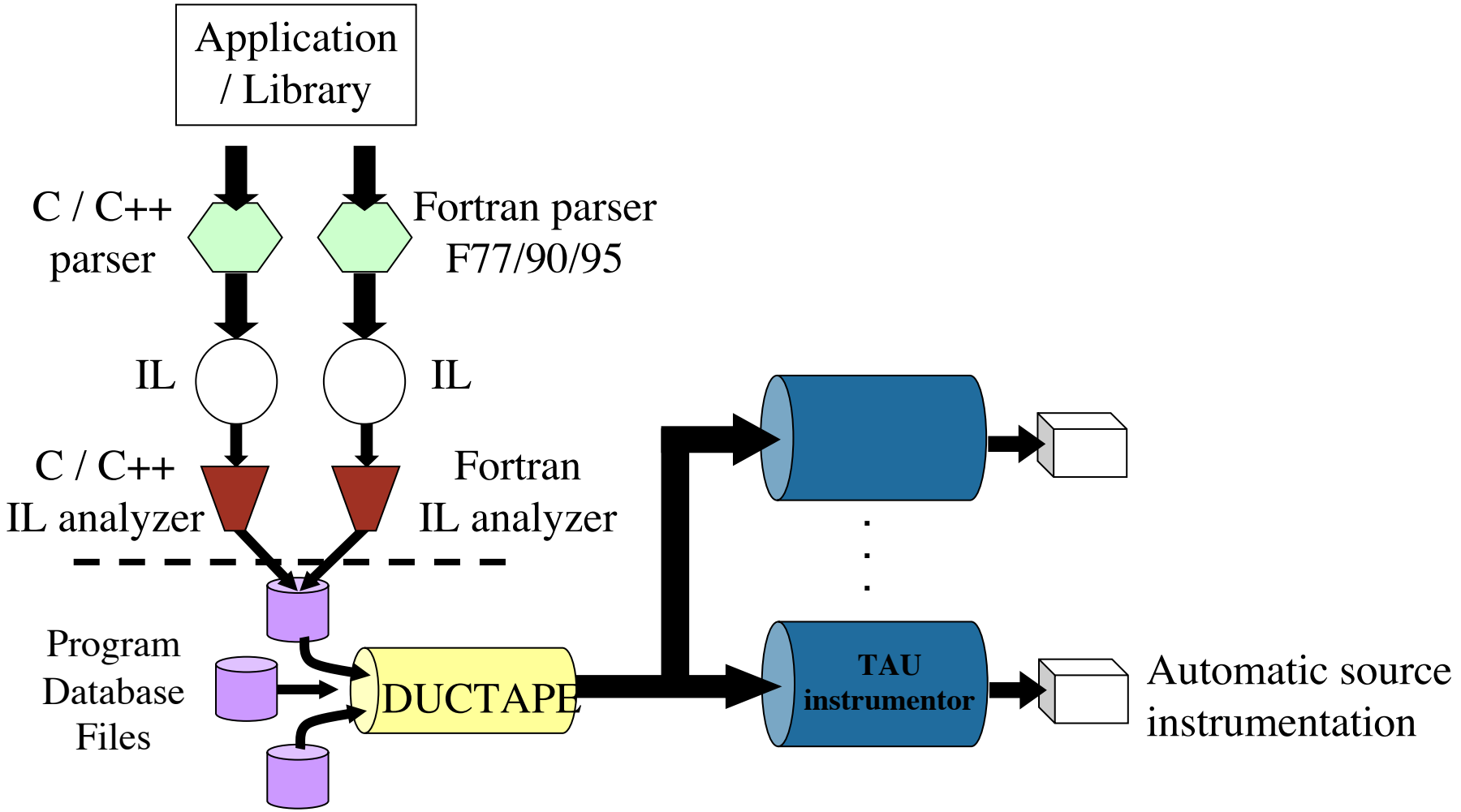
tau_exec Example (continued)

```
Example:
    mpirun -np 2 tau_exec -T icpc,ompt,mpi  -ompt ./a.out
    aprun -n  2 tau_exec -io ./a.out
Example - event-based sampling with samples taken every 1,000,000 FP instructions
    aprun -n  8 tau_exec -ebs -ebs_period=1000000 -ebs_source=PAPI_FP_INS ./ring
Examples - GPU:
    tau_exec -T serial,cupti -cupti ./matmult (Preferred for CUDA 4.1 or later)
    tau_exec -openacc ./a.out
    tau_exec -T serial -opencl ./a.out (OPENCL)
    mpirun -np 2 tau_exec -T mpi,cupti,papi -cupti -um ./a.out (Unified Virtual Memory in CUDA 6.0+)

qsub mode (IBM BG/Q only):
    Original:
      qsub -n 1 --mode smp -t 10 ./a.out
    With TAU:
      tau_exec -qsub -io -memory -- qsub -n 1 … -t 10 ./a.out

Memory Debugging:
    -memory option:
      Tracks heap allocation/deallocation and memory leaks.
    -memory_debug option:
      Detects memory leaks, checks for invalid alignment, and checks for
      array overflow.  This is exactly like setting TAU_TRACK_MEMORY_LEAKS=1
      and TAU_MEMDBG_PROTECT_ABOVE=1 and running with -memory
```

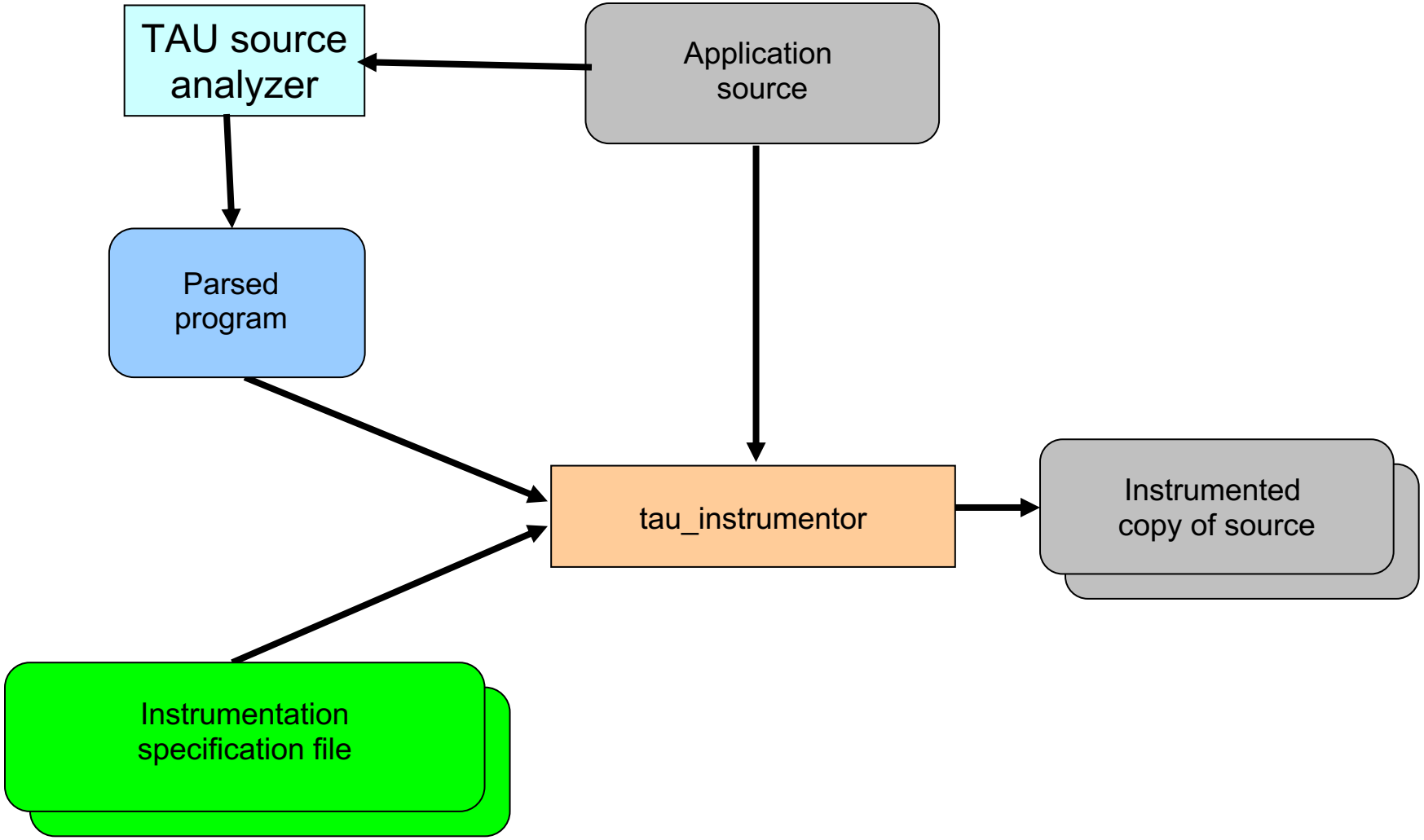
tau_exec can enable event based sampling while launching the executable using environment var TAU_SAMPLING=1 or tau_exec -ebs

Automatic Source Instrumentation in TAU using PDT

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



PDT: automatic source instrumentation



Installing TAU

- Installing PDT:
  - wget http://tau.uoregon.edu/pdt_lite.tgz
  - ./configure –prefix=<dir>; make ; make install
- Installing TAU:
  - wget http://tau.uoregon.edu/tau.tgz
  - ./configure -mpi –pdt=<dir> -bfd=download –unwind=download –iowrapper …
  - make install
- Using TAU:
  - export TAU_MAKEFILE=<taudir>/ibm64linux/lib/Makefile.tau-<TAGS>
  - make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh

Installing TAU on Laptops

- Installing TAU under Mac OS X:
  - wget http://tau.uoregon.edu/tau.dmg
  - Install tau.dmg
- Installing TAU under Linux
  - http://tau.uoregon.edu/tau.exe
- Installing TAU under Linux
  - http://tau.uoregon.edu/tau.tgz
  - ./configure; make install
  - export PATH=<taudir>/x86_64/bin:$PATH

Source Instrumentation in TAU

- TAU supports several compilers, measurement, and thread options
  - IBM XL, Clang, PGI compilers, profiling with hardware counters using PAPI, MPI library, OpenMP…
  - Each measurement configuration of orresponds to a unique stub makefile (configuration file) and library that is generated when you configure it
- To instrument source code automatically using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:
    - **% module load tau**
    - **% export TAU_MAKEFILE=<tau_root>/**ibm64linux/lib/Makefile.tau-<options>
    - **% export TAU_OPTIONS= '-optVerbose …' (see tau_compiler.sh)**
  - Use tau_f90.sh, tau_cxx.sh, tau_upc.sh, tau_caf.sh, or tau_cc.sh as F90, C++, UPC, CAF, 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)**

Different Makefiles for TAU Compiler

```
% module load tau
% ls $TAU_DIR/lib/Makefile*
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-5-papi-cupti
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-5-papi-cupti-openmp-opari
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-5-papi-mpi
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-5-papi-mpi-cupti
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-5-papi-mpi-cupti-openmp-opari
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-5-papi-mpi-openmp-opari
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-6-papi-cupti
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-6-papi-cupti-openmp-opari
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-6-papi-mpi
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-6-papi-mpi-cupti
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-6-papi-mpi-cupti-openmp-opari
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-xl_16.1.1-6-papi-mpi-openmp-opari
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-gcc_8.1.1-papi-gnu-cupti
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-gcc_8.1.1-papi-gnu-cupti-pdt
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-gcc_8.1.1-papi-gnu-mpi
/sw/summit/tau/2.29.1/ibm64linux/lib/Makefile.tau-gcc_8.1.1-papi-gnu-mpi-cupti
…
```

Configuration tags for tau_exec

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

Creates in <taudir>/<arch>/lib:

Makefile.tau-papi-mpi-pdt

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 8 ./a.out to

% mpirun -np 8 tau_exec -T <comma_separated_options> ./a.out

% mpirun -np 8 tau_exec -T papi,mpi,pdt ./a.out

Preloads **<taudir>/<arch>/shared-papi-mpi-pdt/libTAU.so**

% mpirun -np 8 tau_exec -T papi ./a.out

Preloads **<taudir>/<arch>/shared-papi-mpi-pdt/libTAU.so by matching.**

% mpirun -np 8 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.

Compile-Time Options

Optional parameters for the TAU_OPTIONS environment variable:
% tau_compiler.sh

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

Compile-Time Options (contd.)

Optional parameters for the TAU_OPTIONS environment variable:
% tau_compiler.sh

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

See tau_compiler.sh for a full list of TAU_OPTIONS.

…

Using TAU_OPTIONS

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

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

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

```
% export TAU_OPTIONS= '-optPreProcess -optVerbose -optDetectMemoryLeaks'
```

To use an instrumentation specification file:

```
% export TAU_OPTIONS= '-optTauSelectFile=select.tau -optVerbose -optPreProcess'
% cat select.tau
BEGIN_INSTRUMENT_SECTION
loops  routine="#"
# this statement instruments all outer loops in all routines. # is wildcard as well as comment in first column.
END_INSTRUMENT_SECTION
```

Selective Instrumentation File With Program Database Toolkit (PDT)

To use an instrumentation specification file for source instrumentation:

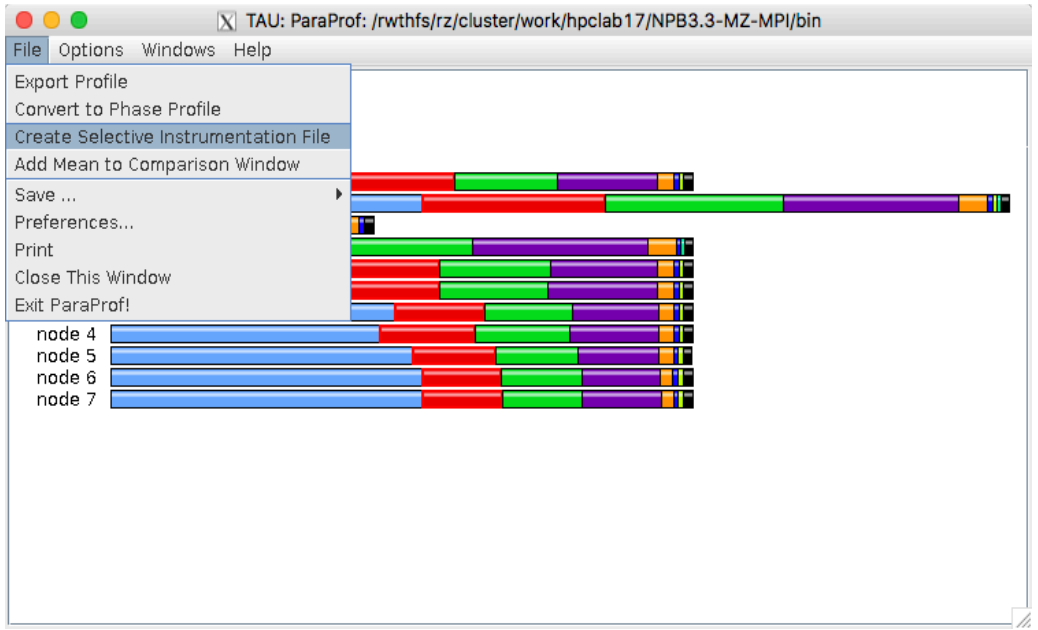
% export TAU_OPTIONS= '-optTauSelectFile=/path/to/select.tau –optVerbose '

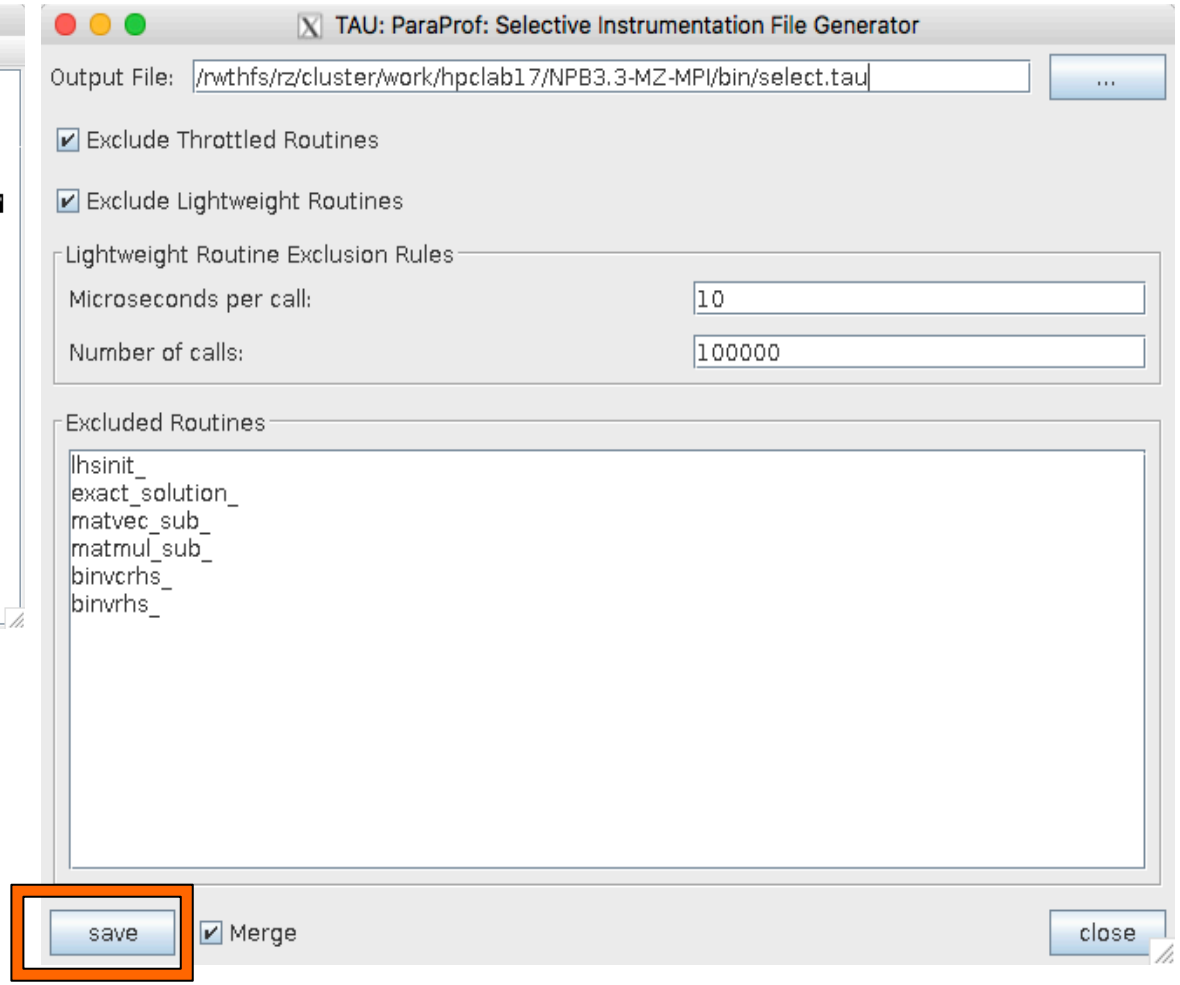
% cat select.tau

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

```
NOTE: paraprof can create this file from an earlier execution for you.
File -> Create Selective Instrumentation File -> save
Selective instrumentation at runtime:
% export TAU_SELECT_FILE=select.tau
```

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





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

Extreme-scale Scientific Software Stack (E4S)

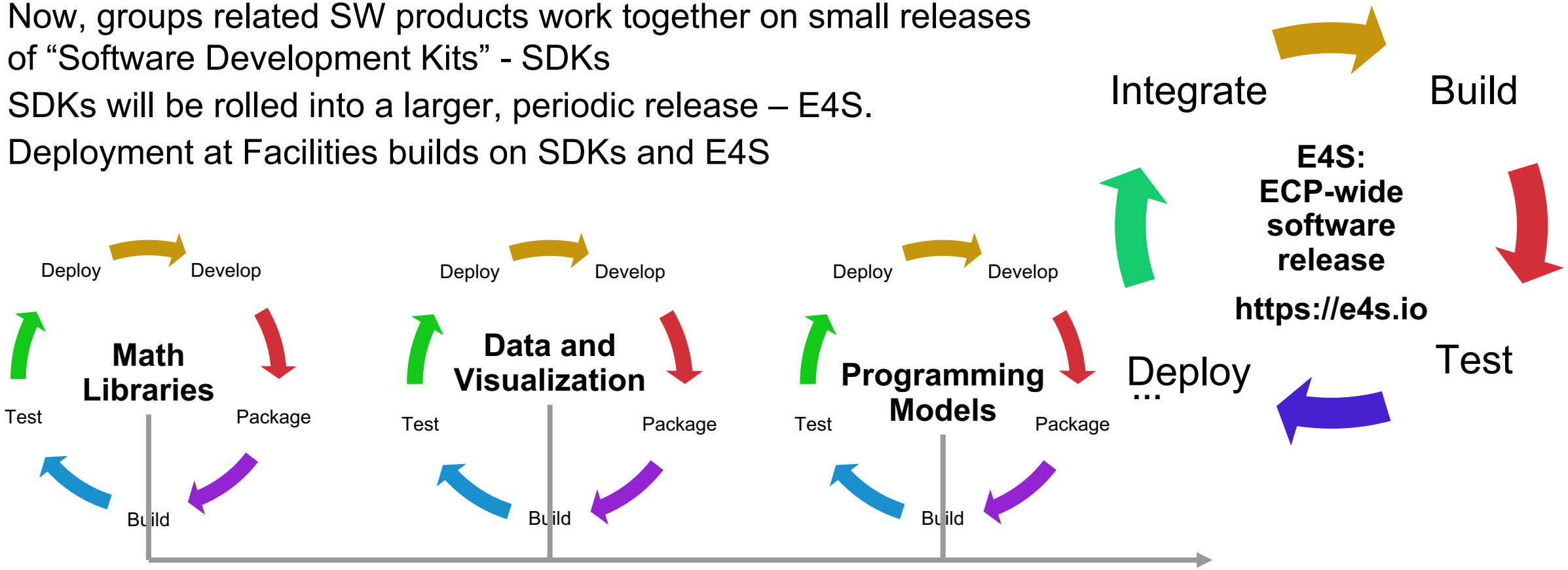
https://e4s.io

Extreme-scale Scientific Software Stack (E4S)
https://e4s.io

- E4S is a community effort to provide open source software packages for developing, deploying, and running scientific applications on HPC platforms.
- E4S provides both source builds and containers of a broad collection of HPC software packages.
- E4S exists to accelerate the development, deployment and use of HPC software, lowering the barriers for HPC users.
- E4S provides containers and turn-key, from-source builds of 80+ popular HPC software packages:
  - MPI: MPICH and OpenMPI
  - Development tools: TAU, HPCToolkit, and PAPI
  - Math libraries: PETSc and Trilinos
  - Data and Viz tools: Adios, HDF5, and Paraview

ECP is working towards a periodic, hierarchical release process

- In ECP, teams increasingly need to ensure that their libraries and components work together
  - Historically, HPC codes used very few dependencies
- Now, groups related SW products work together on small releases of “Software Development Kits” - SDKs
- SDKs will be rolled into a larger, periodic release – E4S.
- Deployment at Facilities builds on SDKs and E4S



E4S: Extreme-scale Scientific Software Stack

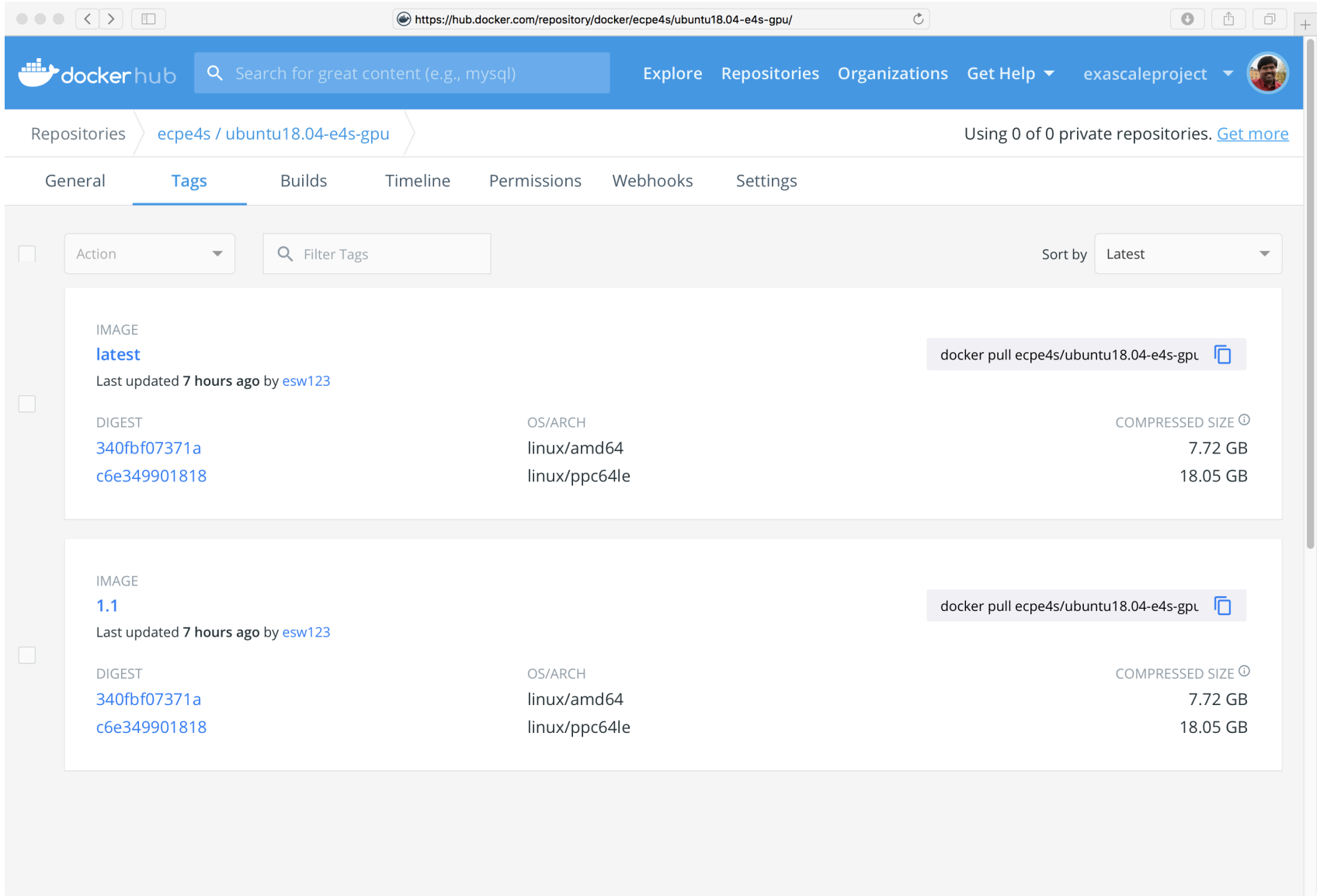
- Curated release of ECP ST products based on Spack [http://spack.io] package manager
- Spack binary build caches for bare-metal installs
  - x86_64, ppc64le (IBM Power 9), and aarch64 (ARM64)
- Container images on DockerHub and E4S website of pre-built binaries of ECP ST products
- Base images and full featured containers (GPU support)
- GitHub recipes for creating custom images from base images
- GitLab integration for building E4S images
- E4S validation test suite on GitHub
- E4S VirtualBox image with support for container runtimes
  - Docker
  - Singularity
  - Shifter
  - Charliecloud
- AWS image to deploy E4S on EC2

https://e4s.io

Integration and Interoperability: E4S

- E4S is released twice a year. ECP Annual Meeting release of E4S v1.1:
  - Containers and turn-key, from-source builds popular HPC software packages
  - 45+ 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
  - Limited access to 10 additional ECP ST products
  - GPU support

E4S v1.1 Release: GPU



- 40+ ECP ST Products
- Support for GPUs
  - NVIDIA (CUDA 10.1.243)
  - ppc64le and x86_64

% docker pull ecpe4s/ubuntu18.04-e4s-gpu

E4S v1.1 GPU Release

```
1: adios        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/adios-1.13.1-ryfsxpqaab6rkefscdy3qdba47eeh5fe
2: aml  /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/aml-0.1.0-ftizegmvpbweuyzg75g3ndzhdyjx37op
3: argobots     /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/argobots-1.0rc1-xjjrxwo2molgdesfwjyojaiwj24rxp3h
4: caliper      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/caliper-2.0.1-lzff2xp2eg7jqrmgy4k2ifbiii7zmpas
5: darshan-runtime      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/darshan-runtime-3.1.7-jgyzjuukvxpwjrzeaweaukygyg7q6jxz
6: dyninst      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/dyninst-10.1.0-2ayefefqh74xwuadlbs3d6p2zyammmsf
7: flecsi       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/flecsi-develop-3v4qqtzsnja6dt4lr5pps3cxv3hy6to2
8: gasnet       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/gasnet-2019.3.0-n4tdzzd4hvy6zexy7bhe4qryzbcfkoif
9: globalarrays         /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/globalarrays-5.7-ilibm5yzx6draxpkfo5t3d4csx5g64nu
10: gotcha      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/gotcha-1.0.2-h3qq2jz6a2ihwibdqndbfehh4pnmjr3z
11: hdf5        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/hdf5-1.10.5-lgnifkbzs3p6ojothkjg53wf6hut4xqv
12: hpctoolkit  /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/hpctoolkit-2019.08.14-vljantrum36jbdlzz37tynacw3ee2yhk
13: hypre       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/hypre-2.18.1-pj2krvlyc5bnjpq742a5xvk6k53r5ita
14: kokkos      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/kokkos-2.9.00-7h2cg7o3kdu23qymnfyqwsewruvwkxgr
15: legion      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/legion-19.06.0-7ksiuqzq57y24t5kl6s2n4ffqdpnlupg
16: libnrm      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/libnrm-0.1.0-5yet3aafh2rp3wuapixphq3mwykiqyqn
17: libquo      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/libquo-1.3-ul26vaqh6kl6vmqh4pj4esm7ztelbpfo
18: mercury     /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/mercury-1.0.1-akxk7fk6mdbp2m5wxdaum3lehlb7dt24
19: mfem        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/mfem-4.0.0-bdte6oktrcnaktwi7qjj72pisial5krb
20: mpich       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/mpich-3.2.1-sb2fna3i4s4feofy6swrr6wivzyyu7he
21: mpifileutils        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/mpifileutils-develop-r3s7t6qzgqb4y4hpd5mzldbtskhvlqfj
22: ninja       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/ninja-1.9.0-pxjz2hyrn24ar65hqwcou7rvgafvylos
23: openmpi     /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/openmpi-3.1.4-cty2bl5wikxipkgzlpu32xtmpqxzua2v
24: papi        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/papi-5.7.0-eaabymcif7c5dvjdejevnqkgat7recfp
25: papyrus     /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/papyrus-develop-nu56mejweub56zjuexidml2t4o664zug
26: parallel-netcdf     /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/parallel-netcdf-1.11.2-zxyjkdz4yxmyxcgyarcym5fqec4paj56
27: pdt         /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/pdt-3.25.1-czegb3ismtstyml2wugkifojybw4cacs
28: petsc       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/petsc-3.12.1-q6pm6wyb43hlc6ndpvrkphvwntbt3xul
29: py-jupyter-notebook         /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/py-jupyter-notebook-4.2.3-qky73g3q6kapsva7ishirewe3qtapxhk
30: py-libensemble      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/py-libensemble-0.5.2-gz4rxbjd5lku5lt6ji6lyf2f6qoafuzt
31: qthreads    /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/qthreads-1.14-jjpfrqevnacp6dhd5or6vz4obckmy4ah
32: raja        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/raja-0.8.0-mhamgpur67esmyekjpclfvlv6fdngod6
33: rempi       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/rempi-1.1.0-feg2oeofieeanoloaxulawxdeaqnqftj
34: scr         /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/scr-1.2.2-5sepujyr7dp5eioymphnqva4atcjfc4e
35: strumpack   /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/strumpack-3.1.1-jd7ytfdr4saggdbxx3khjy2jamlfbru4
36: sundials    /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/sundials-5.0.0-cjkp7zluit5jtm6eqd74mavwko5thr2w
37: superlu-dist        /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/superlu-dist-6.1.1-6gklgz5snhaipunjjtirlo46zq76zd2d
38: sz  /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/sz-1.4.12.3-7bnst7vnpuajazlagxtot46oih4caoqa
39: tasmanian   /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/tasmanian-7.0-e3r2dvv5nnpg7yafs5hzcocvkmxaf7pn
40: tau         /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/tau-2.29-3eb5tlvcnukpkd2pls4z7ivhtc2ld7jm
41: trilinos    /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/trilinos-12.14.1-jcikf6anps63huaohpymnm4xohgrrral
42: umpire      /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/umpire-0.3.3-bnxsv2cla4yc42mvfhvwlk7suylmbcou
43: unifyfs     /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/unifyfs-develop-fqzlm2dhtbnb2aeggadfcue2mkv74sco
44: upcxx       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/upcxx-2019.9.0-ndrqyemtolmjk7i3dduekjbt4i3uo2jz
45: veloc       /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/veloc-1.1-h723o37b5mowzewdzqburtrawnb5ynhi
46: zfp         /spack/opt/spack/linux-ubuntu18.04-ppc64le/gcc-7.3.0/zfp-0.5.5-3r4a4s3qdeqbdabvlwlswrgig62yc6yj
```

- 46 ECP ST products
- Ubuntu v18.04 ppc64le
- Support for GPUs
  - NVIDIA

E4S v1.1 GPU Support

sameer@cyclops:~ — ssh zorak — 137×51

```
Singularity> which python
/usr/bin/python
Singularity> python
Python 3.6.10 |Anaconda, Inc.| (default, Jan  7 2020, 21:47:07)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>> import torch
>>> import cv2
>>> import pandas
>>> import sklearn
>>> import keras
Using TensorFlow backend.
>>> import matplotlib; import numpy; import scipy
>>> torch.cuda.current_device()
0
>>> tensorflow.test.is_gpu_available()
2020-02-20 19:46:00.714206: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 0 with properties:
name: Tesla V100-SXM2-32GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0004:04:00.0
totalMemory: 31.72GiB freeMemory: 24.06GiB
2020-02-20 19:46:00.828521: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 1 with properties:
name: Tesla V100-SXM2-32GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0004:05:00.0
totalMemory: 31.72GiB freeMemory: 31.41GiB
2020-02-20 19:46:00.937000: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 2 with properties:
name: Tesla V100-SXM2-32GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0035:03:00.0
totalMemory: 31.72GiB freeMemory: 31.41GiB
2020-02-20 19:46:01.043356: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 3 with properties:
name: Tesla V100-SXM2-32GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0035:04:00.0
totalMemory: 31.72GiB freeMemory: 31.41GiB
2020-02-20 19:46:01.043434: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1512] Adding visible gpu devices: 0, 1, 2, 3
2020-02-20 19:46:09.555187: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] Device interconnect StreamExecutor with strength 1 ed
ge matrix:
2020-02-20 19:46:09.555313: I tensorflow/core/common_runtime/gpu/gpu_device.cc:990]      0 1 2 3
2020-02-20 19:46:09.555326: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 0:   N Y Y Y
2020-02-20 19:46:09.555334: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 1:   Y N Y Y
2020-02-20 19:46:09.555341: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 2:   Y Y N Y
2020-02-20 19:46:09.555349: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 3:   Y Y Y N
2020-02-20 19:46:09.555942: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/device:GPU:0 with 22986
MB memory) -> physical GPU (device: 0, name: Tesla V100-SXM2-32GB, pci bus id: 0004:04:00.0, compute capability: 7.0)
2020-02-20 19:46:09.556909: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/device:GPU:1 with 30132
MB memory) -> physical GPU (device: 1, name: Tesla V100-SXM2-32GB, pci bus id: 0004:05:00.0, compute capability: 7.0)
2020-02-20 19:46:09.557139: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/device:GPU:2 with 30132
MB memory) -> physical GPU (device: 2, name: Tesla V100-SXM2-32GB, pci bus id: 0035:03:00.0, compute capability: 7.0)
2020-02-20 19:46:09.558067: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/device:GPU:3 with 30132
MB memory) -> physical GPU (device: 3, name: Tesla V100-SXM2-32GB, pci bus id: 0035:04:00.0, compute capability: 7.0)
True
>>>
```

Download E4S v1.1 GPU image

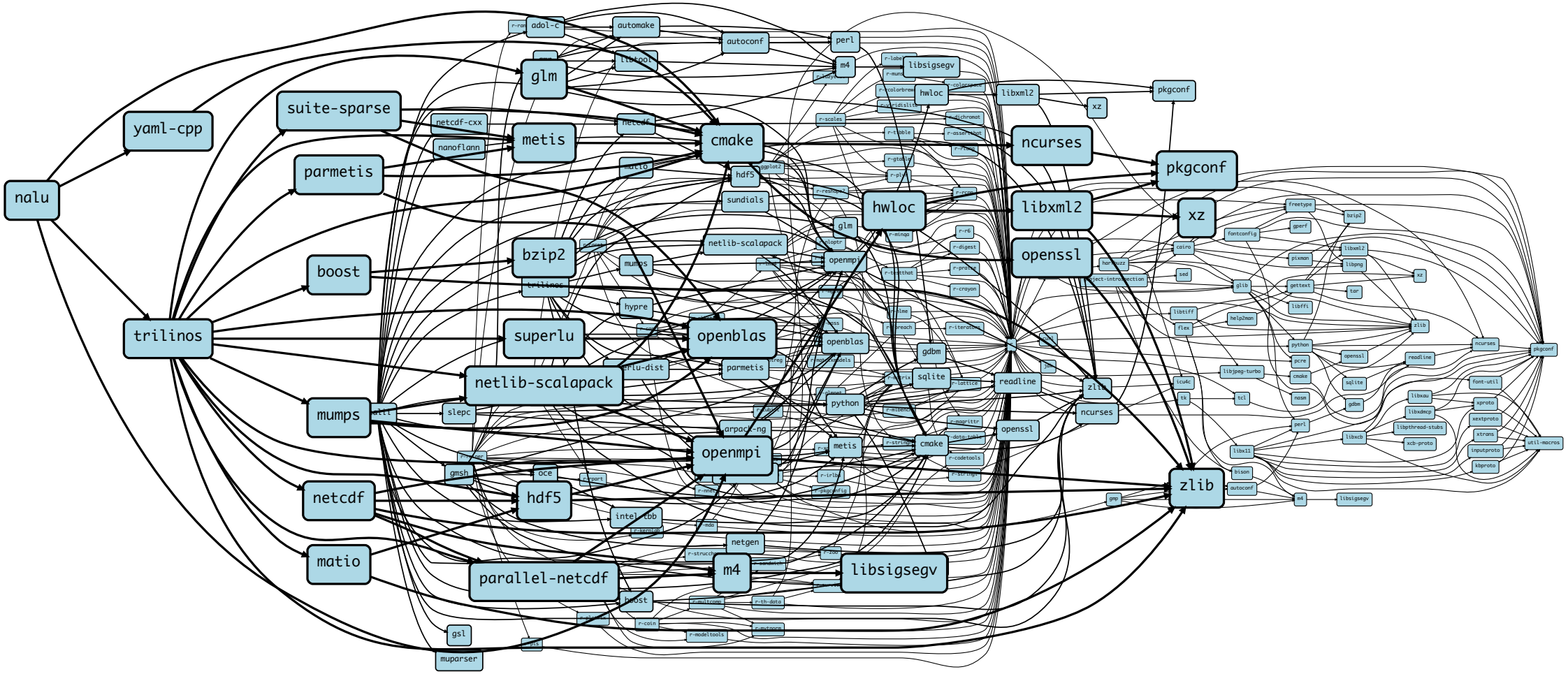
https://e4s.io

Spack

Spack

- E4S uses the Spack package manager for software delivery
- Spack provides the ability to specify versions of software packages that are and are not interoperable.
- Spack is a build layer for not only E4S software, but also a large collection of software tools and libraries outside of ECP ST.
- Spack supports achieving and maintaining interoperability between ST software packages.

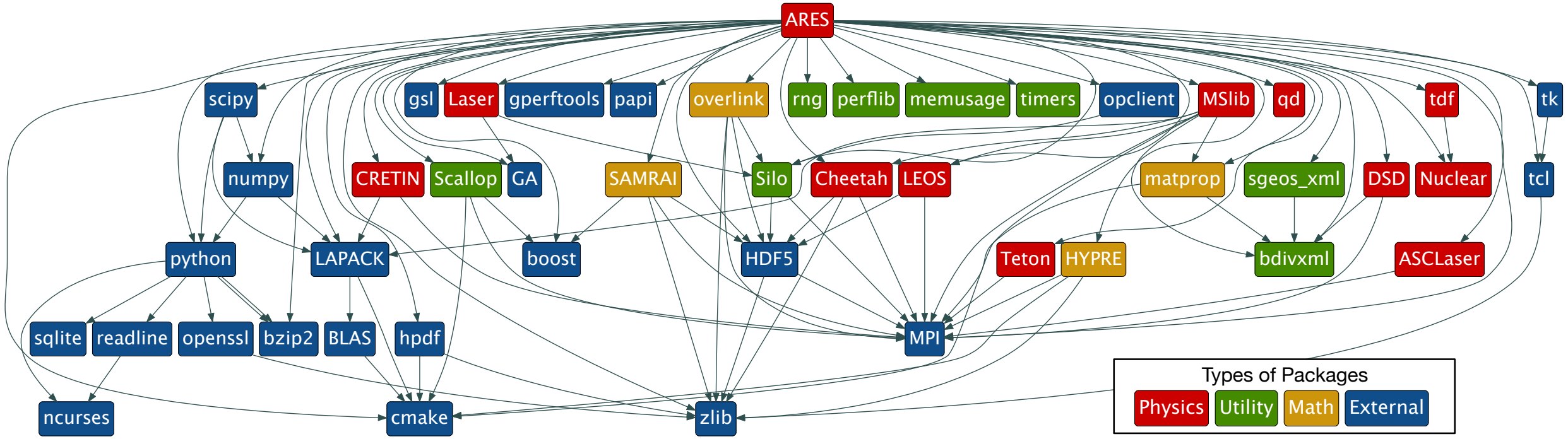
Scientific software is becoming extremely complex



Nalu: Generalized Unstructured Mesh Navier-Stokes Low Mach Flow

deal.II: C++ Finite Element Library

Even proprietary codes are based on many open source libraries



- Half of this DAG is external (blue); *more* than half of it is open source
- Nearly *all* of it needs to be built specially for HPC to get the best performance

The Exascale Computing Project is building an entire *ecosystem*

15+ applications x **80+ software packages** x **5+ target architectures/platforms**
Xeon Power KNL
NVIDIA ARM Laptops?

x **Up to 7 compilers**
Intel GCC Clang XL
PGI Cray NAG

x **10+ Programming Models**
OpenMPI MPICH MVAPICH OpenMP CUDA
OpenACC Dharma Legion RAJA Kokkos

x **2-3 versions of each package + external dependencies**

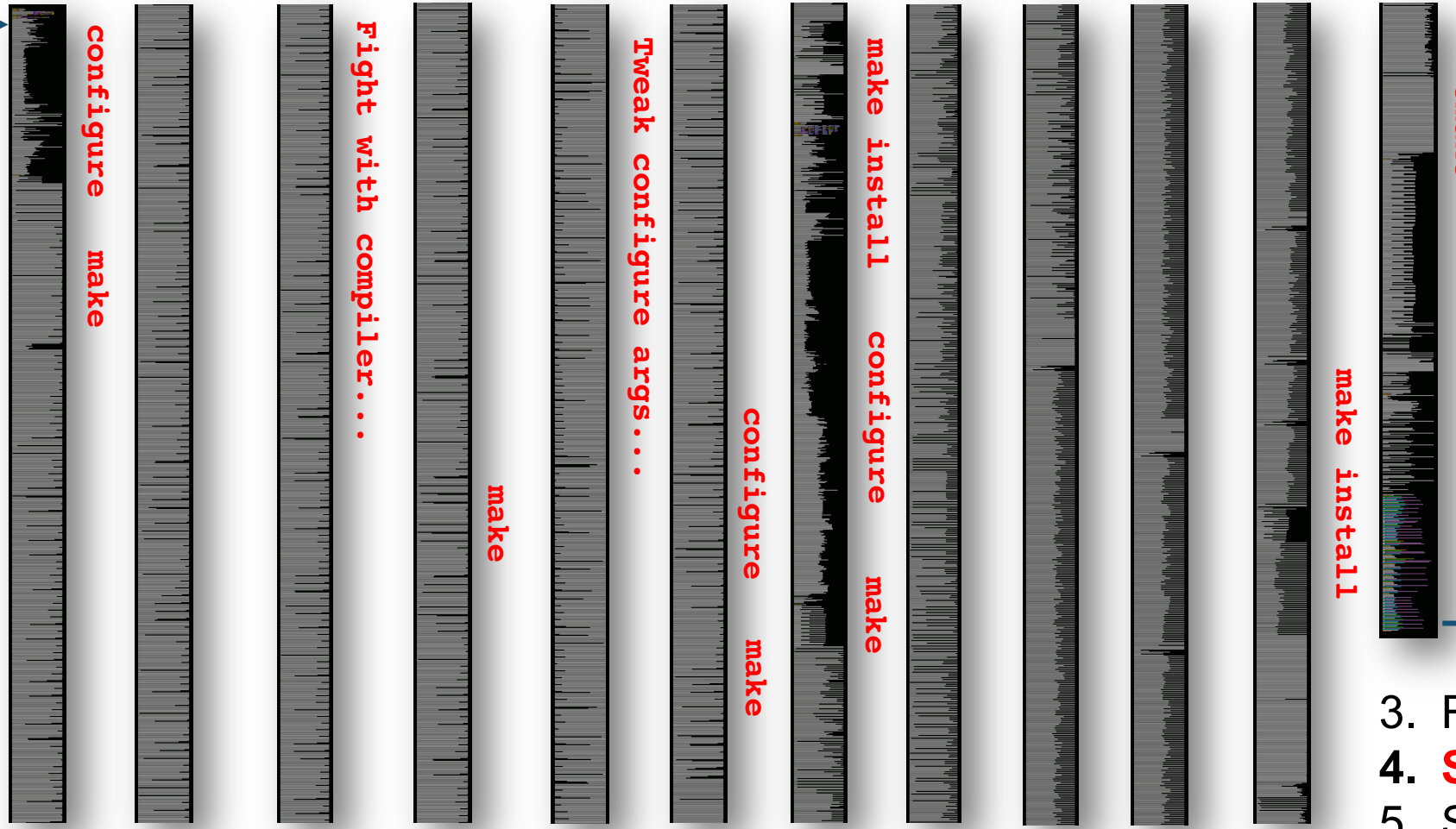
= up to **1,260,000** combinations!

- Every application has its own stack of dependencies.
- Developers, users, and facilities dedicate (many) FTEs to building & porting.
- Often trade reuse and usability for performance.

We must make it easier to rely on others' software!

How to install software on a supercomputer

1. Download all 16 tarballs you need
2. Start building!



3. Run code
4. **Segfault!?**
5. Start over…

What about modules?

- Most supercomputers deploy some form of *environment modules*
  - TCL modules (dates back to 1995) and Lmod (from TACC) are the most popular

```
$ gcc
- bash: gcc: command not found

$ module load gcc/7.0.1
$ gcc –dumpversion
7.0.1
```

- Modules don’t handle installation!
  - They only modify your environment (things like PATH, LD_LIBRARY_PATH, etc.)
- Someone (likely a team of people) has already installed gcc for you!
  - Also, you can *only* `module load` the things they’ve installed

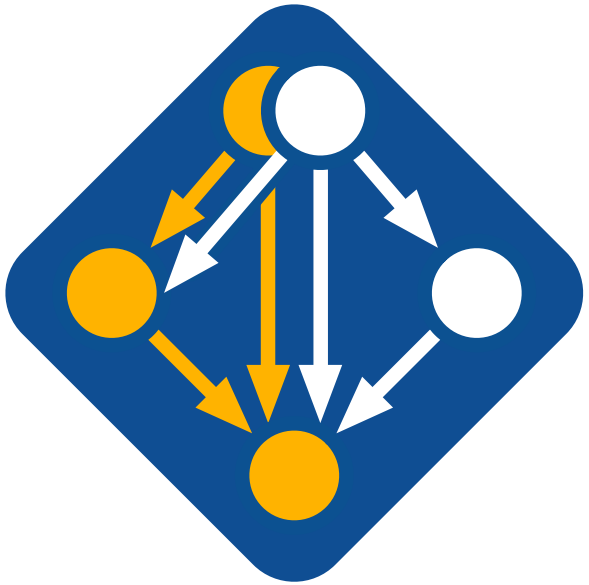
Spack is a flexible package manager for HPC

```
$ git clone https://github.com/spack/spack
$ . spack/share/spack/setup-env.sh
```

- How to install a package:

```
$ spack install tau
```

- TAU and its dependencies are installed within the Spack directory.
- Unlike typical package managers, Spack can also install many variants of the same build.
  - Different compilers
  - Different MPI implementations
  - Different build options



Visit spack.io

github.com/spack/spack

@spackpm

Spack provides the *spec* syntax to describe custom configurations

```
$ spack install tau                                                  unconstrained
$ spack install tau@2.29                                             @ custom version
$ spack install tau@2.29 %gcc@7.3.0                                  % custom compiler
$ spack install tau@2.29 %gcc@7.3.0 +mpi+python+pthreads             +/- build option
$ spack install tau@2.29 %gcc@7.3.0 +mpi ^mvapich2@2.3~wrapperrpath ^ dependency information
```

- Each expression is a ***spec*** for a particular configuration
  - Each clause adds a constraint to the spec
  - Constraints are optional – specify only what you need.
  - Customize install on the command line!
- Spec syntax is recursive
  - Full control over the combinatorial build space

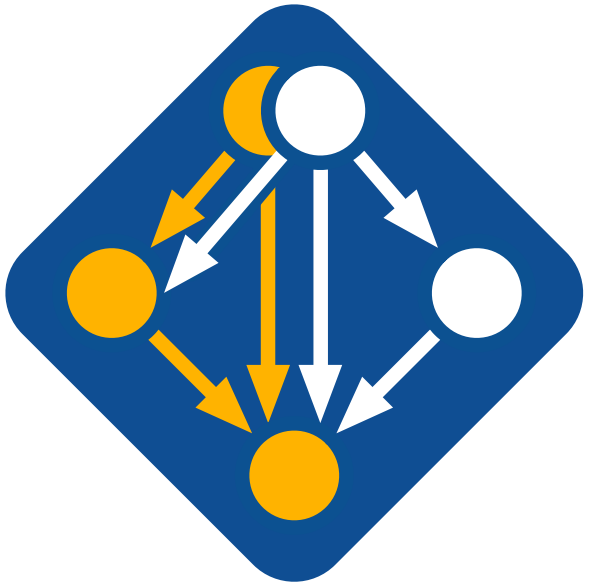
`spack find` shows what is installed

```
5. root@3b983c57f123: / (ssh)
sameer@minotaur:~$ docker run -v $HOME:/home/sameer -it ecpe4s/ubuntu18.04-ppc64le-e4s
root@3b983c57f123:/# which spack
/spack/bin/spack
root@3b983c57f123:/# spack find
==> 235 installed packages
-- linux-ubuntu18.04-ppc64le / gcc@7.3.0 ------------------------
adios@1.13.1            hdf5@1.10.5             lua-luafilesystem@1_7_0_2  py-async-generator@1.10  py-pexpect@4.6.0          raja@0.8.0
aml@0.1.0               hdf5@1.10.5             lua-luaposix@33.4.0        py-babel@2.7.0           py-pickleshare@0.7.4      rankstr@0.0.2
argobots@1.0rc1         hdf5@1.10.5             lwgrp@1.0.2                py-backcall@0.1.0        py-pillow@6.2.0           readline@8.0
autoconf@2.69           hpctoolkit@2019.08.14   lz4@1.9.2                  py-blinker@1.4           py-prometheus-client@0.7.1 redset@0.0.3
automake@1.16.1         hwloc@1.11.11           lzo@2.10                   py-certifi@2019.9.11     py-prompt-toolkit@2.0.9   rempi@1.1.0
axl@0.1.1               hypre@2.18.1            m4@1.4.18                  py-certipy@0.1.3         py-ptyprocess@0.5.1       scr@1.2.2
binutils@2.32           intel-tbb@2019.4        margo@0.4.3                py-cffi@1.13.0           py-py@1.8.0               shuffile@0.0.3
bmi@develop             kokkos@2.7.00           matio@1.5.13               py-chardet@3.0.4         py-pycparser@2.19         snappy@1.1.7
boost@1.70.0            kokkos@2.9.00           mercury@1.0.1              py-cryptography@2.3.1    py-pygments@2.4.2         sqlite@3.30.1
boost@1.70.0            kokkos-kernels@2.7.00   mercury@1.0.1              py-cycler@0.10.0         py-pyjwt@1.7.1            strumpack@3.1.1
boost@1.70.0            kvtree@1.0.2            metis@5.1.0                py-cython@0.29.13        py-pyopenssl@19.0.0       suite-sparse@5.3.0
bzip2@1.0.8             leveldb@1.22            mfem@4.0.0                 py-decorator@4.4.0       py-pyparsing@2.4.2        sundials@5.0.0
c-blosc@1.17.0          libarchive@3.3.2        mpich@3.2.1                py-entrypoints@0.3       py-python-dateutil@2.8.0  superlu@5.2.1
caliper@2.0.1           libbsd@0.9.1            mpifileutils@develop       py-idna@2.8              py-python-dateutil@2.8.0  superlu-dist@6.1.1
cmake@3.15.4            libcircle@0.2.1-rc.1    mumps@5.2.0                py-ipykernel@5.1.0       py-python-editor@1.0.4    sz@1.4.12.3
curl@7.63.0             libdwarf@20180129       nasm@2.14.02               py-ipython@7.3.0         py-python-oauth2@1.1.1    tar@1.32
darshan-runtime@3.1.7   libfabric@1.8.1         ncurses@6.1                py-ipython-genutils@0.2.0 py-pytz@2019.3           tasmanian@7.0
darshan-util@3.1.7      libffi@3.2.1            netcdf@4.7.1               py-jinja2@2.10.3         py-requests@2.22.0        tcl@8.6.8
diffutils@3.7           libiberty@2.31.1        netlib-scalapack@2.0.2     py-joblib@0.14.0         py-scikit-learn@0.21.3    texinfo@6.5
dtcmp@1.1.0             libiconv@1.16           nettle@3.4.1               py-jsonschema@2.6.0      py-scikit-optimize@0.5.2  trilinos@12.14.1
dyninst@10.1.0          libjpeg-turbo@2.0.3     ninja@1.9.0                py-jupyter-client@4.4.0  py-scipy@1.3.1            umpire@0.3.3
elfutils@0.177          libmonitor@2018.07.18   numactl@2.0.12             py-jupyter-console@5.2.0 py-setuptools@41.4.0      unifyfs@develop
emacs@26.2              libnrm@0.1.0            openblas@0.3.7             py-jupyter-core@4.4.0    py-setuptools@41.4.0      unzip@6.0
er@0.0.3                libpciaccess@0.13.5     openmpi@3.1.4              py-jupyter-notebook@4.2.3 py-simplegeneric@0.8.1   upcxx@2019.9.0
expat@2.2.9             libpfm4@4.10.1          openssl@1.1.1d             py-jupyterhub@1.0.0      py-six@1.12.0             util-macros@1.19.1
findutils@4.6.0         libpng@1.6.37           papi@5.7.0                 py-kiwisolver@1.1.0      py-six@1.12.0             veloc@1.1
flatcc@0.5.3            libpthread-stubs@0.4    papyrus@develop            py-libensemble@0.5.2     py-sqlalchemy@1.3.9       xerces-c@3.2.2
flecsi@develop          libquo@1.3              parallel-netcdf@1.11.2     py-mako@1.0.4            py-tornado@6.0.3          xz@5.2.4
freetype@2.10.1         libsigsegv@2.12         parmetis@4.0.3             py-markupsafe@1.1.1      py-traitlets@4.3.3        zeromq@4.3.2
gdbm@1.18.1             libsodium@1.0.17        pcre@8.42                  py-matplotlib@3.1.1      py-urllib3@1.25.6         zfp@0.5.5
gettext@0.20.1          libtool@2.4.6           pcre@8.42                  py-mistune@0.7.1         py-vcversioner@2.16.0.0   zlib@1.2.11
git@2.21.0              libunwind@1.2.1         pdsh@2.31                  py-mpi4py@3.0.1          py-wcwidth@0.1.7          zstd@1.4.3
glm@0.9.7.1             libunwind@2018.10.12    perl@5.30.0                py-nbconvert@4.2.0       py-ytopt@0.1.0
globalarrays@5.7        libxml2@2.9.9           petsc@3.12.1               py-nbformat@4.4.0        py-zmq@17.1.2
gmp@6.1.2               libyogrt@1.24           pkgconf@1.6.3              py-numpy@1.17.3          python@3.7.3
gotcha@0.0.2            lmod@8.1.5              py-alembic@1.0.7           py-oauthlib@3.1.0        python@3.7.4
gotcha@1.0.2            lua@5.3.5               py-asn1crypto@0.22.0       py-pamela@1.0.0          qthreads@1.14
```

- All the versions coexist!
  - Multiple versions of same package are ok.
- Packages are installed to automatically find correct dependencies.
- Binaries work *regardless of user's environment*.
- Spack also generates module files.
  - Don't *have* to use them.

The Spack community is growing rapidly

- **Spack simplifies HPC software for:**
  - Users
  - Developers
  - Cluster installations
  - The largest HPC facilities
- **Spack is central to ECP's software strategy**
  - Enable software reuse for developers and users
  - Allow the facilities to consume the entire ECP stack
- **The roadmap is packed with new features:**
  - Building the ECP software distribution
  - Better workflows for building containers
  - Stacks for facilities
  - Chains for rapid dev workflow
  - Optimized binaries
  - Better dependency resolution



Visit spack.io

github.com/spack/spack

@spackpm

E4S Spack Build Cache and Container Build Pipeline

Reproducible, Customizable Container Builds & Spack Mirrors

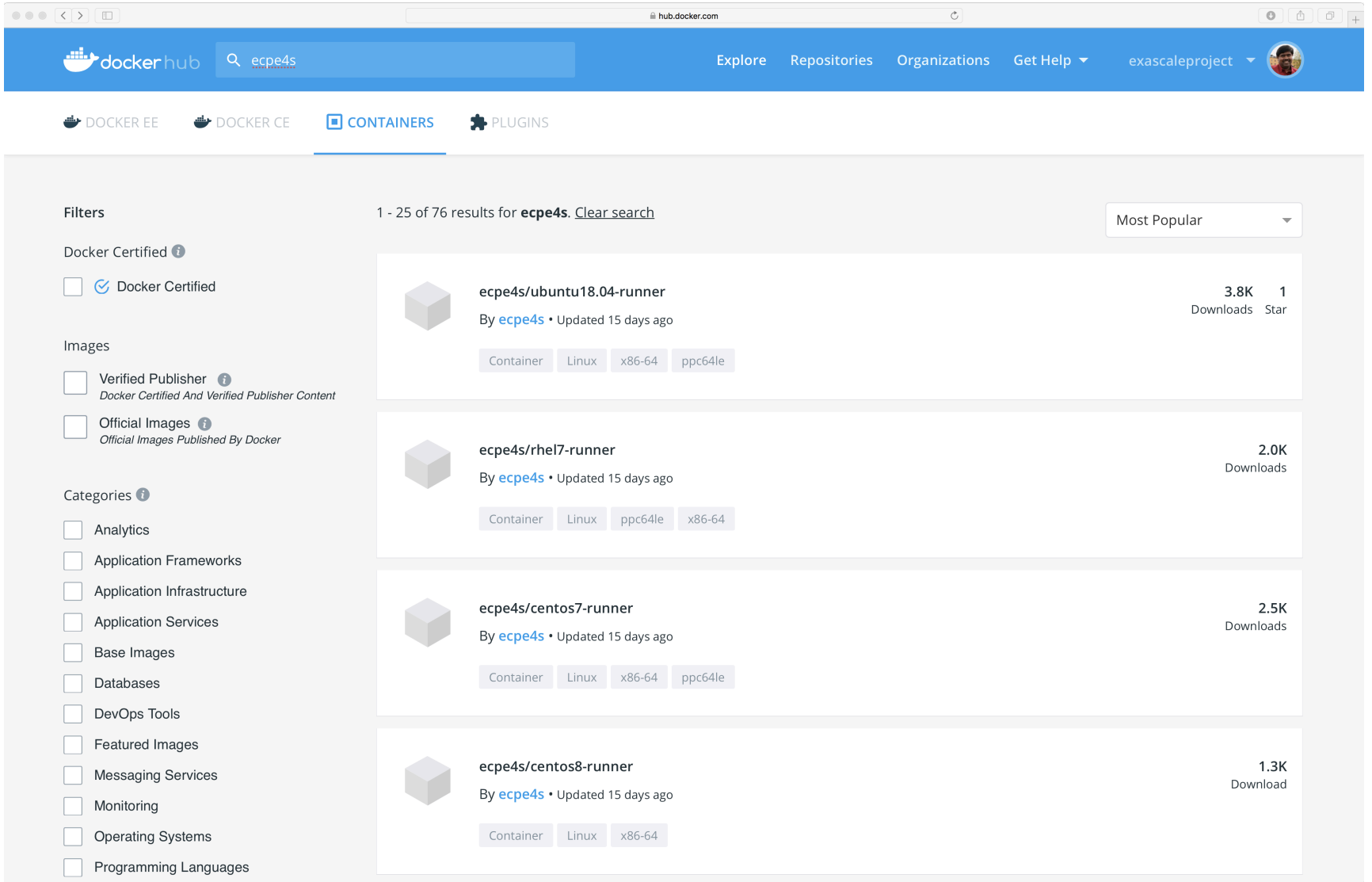
- E4S provides base images and recipes for building Docker containers based on SDKs
  - Git: https://github.com/UO-OACISS/e4s
  - E4S provides build caches for Spack for native bare-metal as well as container builds based installation of ST products
  - Build caches: https://oaciss.uoregon.edu/e4s/inventory.html
    - The build cache model can be extended to target platforms, and can be managed by facilities staff when appropriate.

E4S: Spack Build Cache at U. Oregon

- 10,000+ binaries
- S3 mirror
- No need to build from source code!

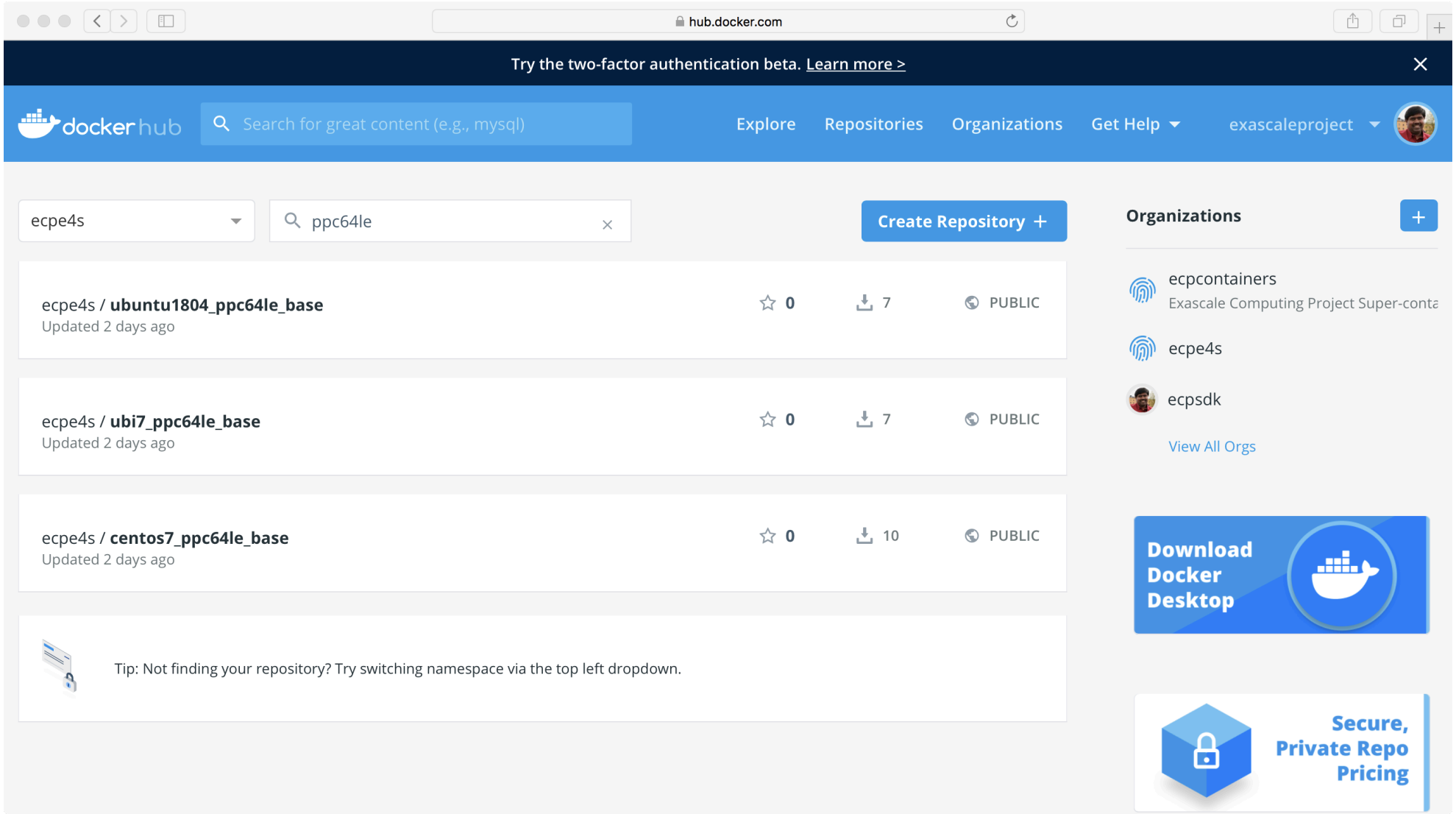
- https://oaciss.uoregon.edu/e4s/inventory.html

E4S: GitLab Runner Images



- Dockerhub
- Bare-bones
- Multi-platfrom
- Build E4S

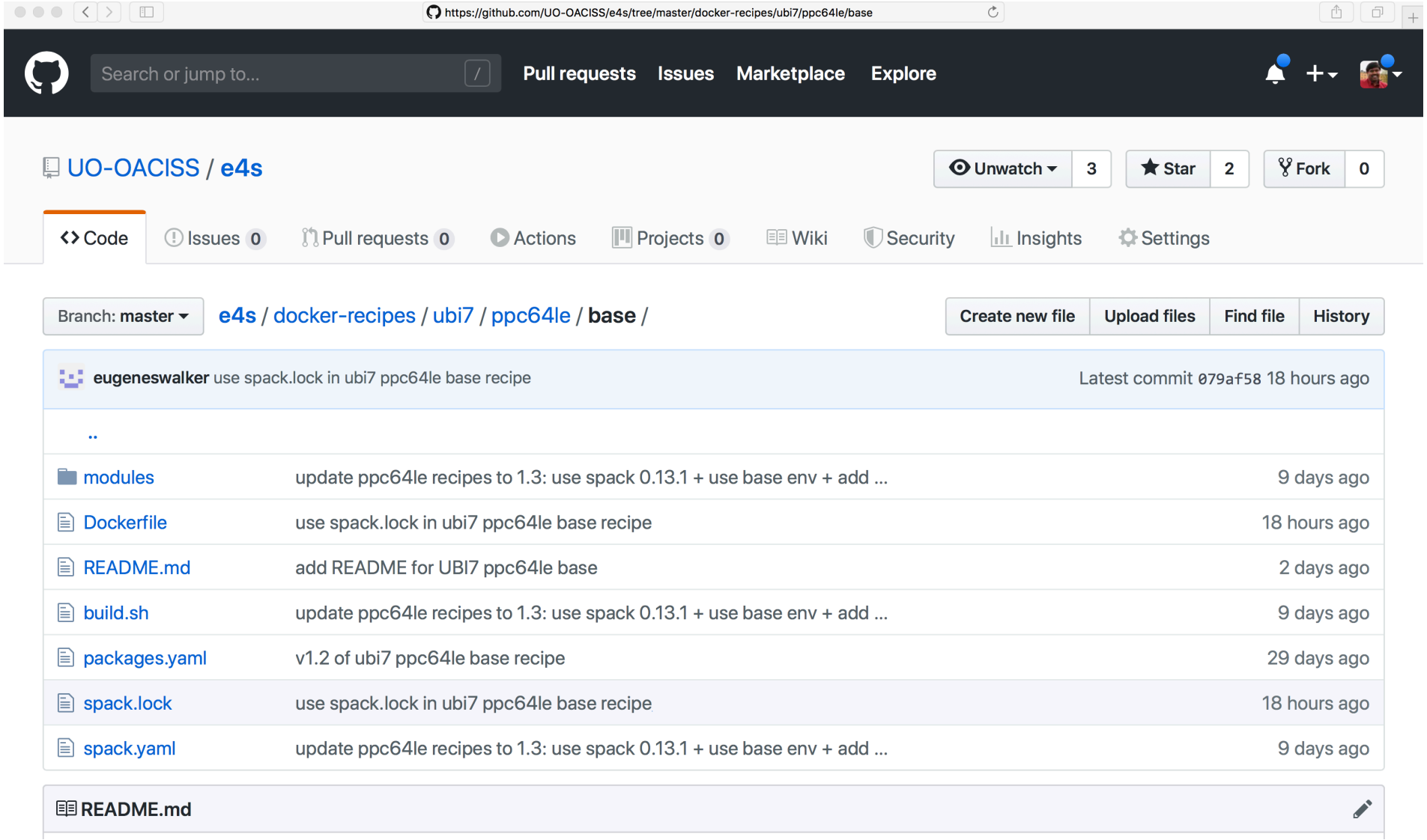
E4S: ppc64le Base Container Images



- Hub.docker.com
- ecpe4s

- Ubuntu 18.04
- RHEL/UBI 7.6
- Centos 7.6

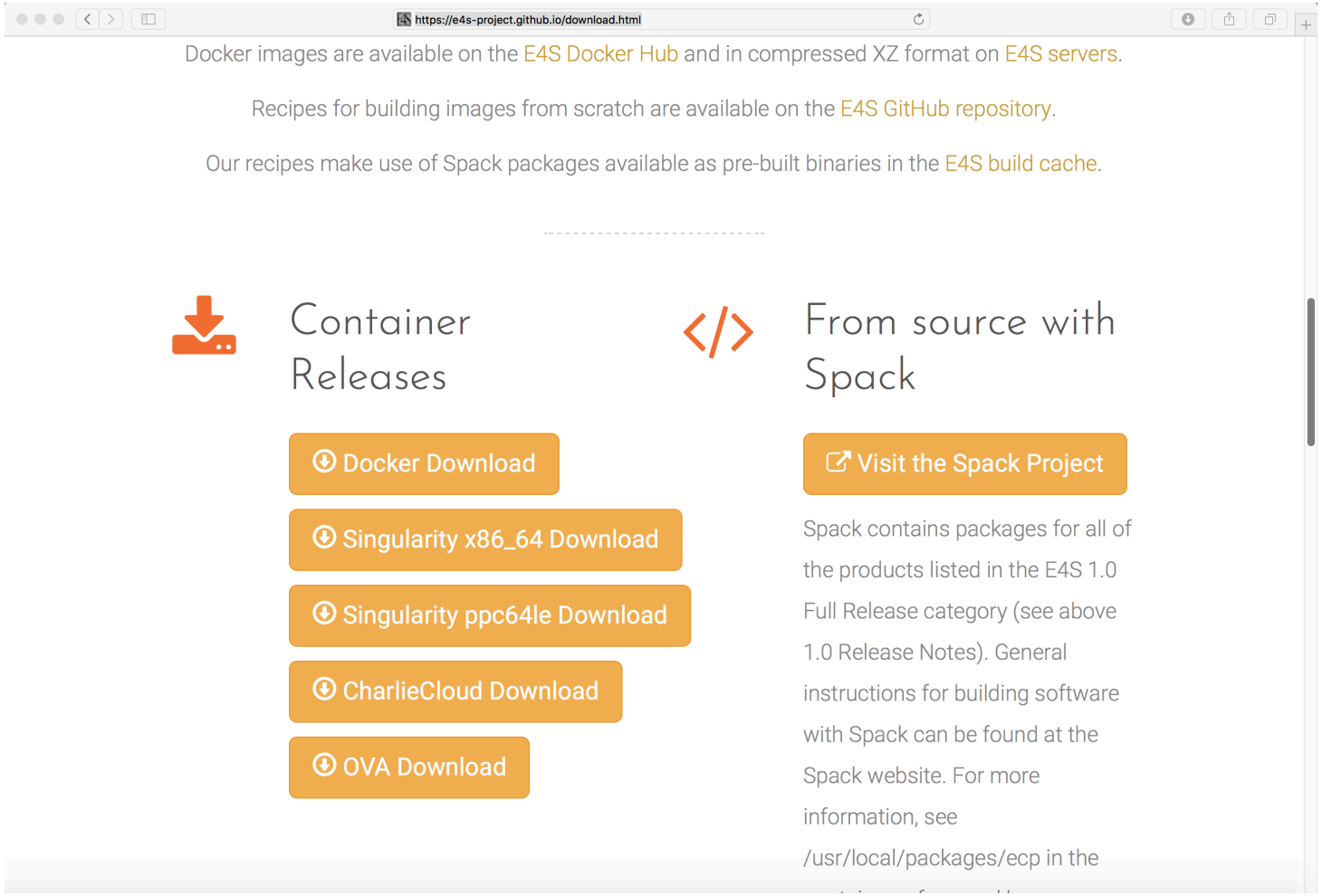
E4S: Multi-platform Reproducible Docker Recipes



E4S
- x86_64
- ppc64le
- aarch64

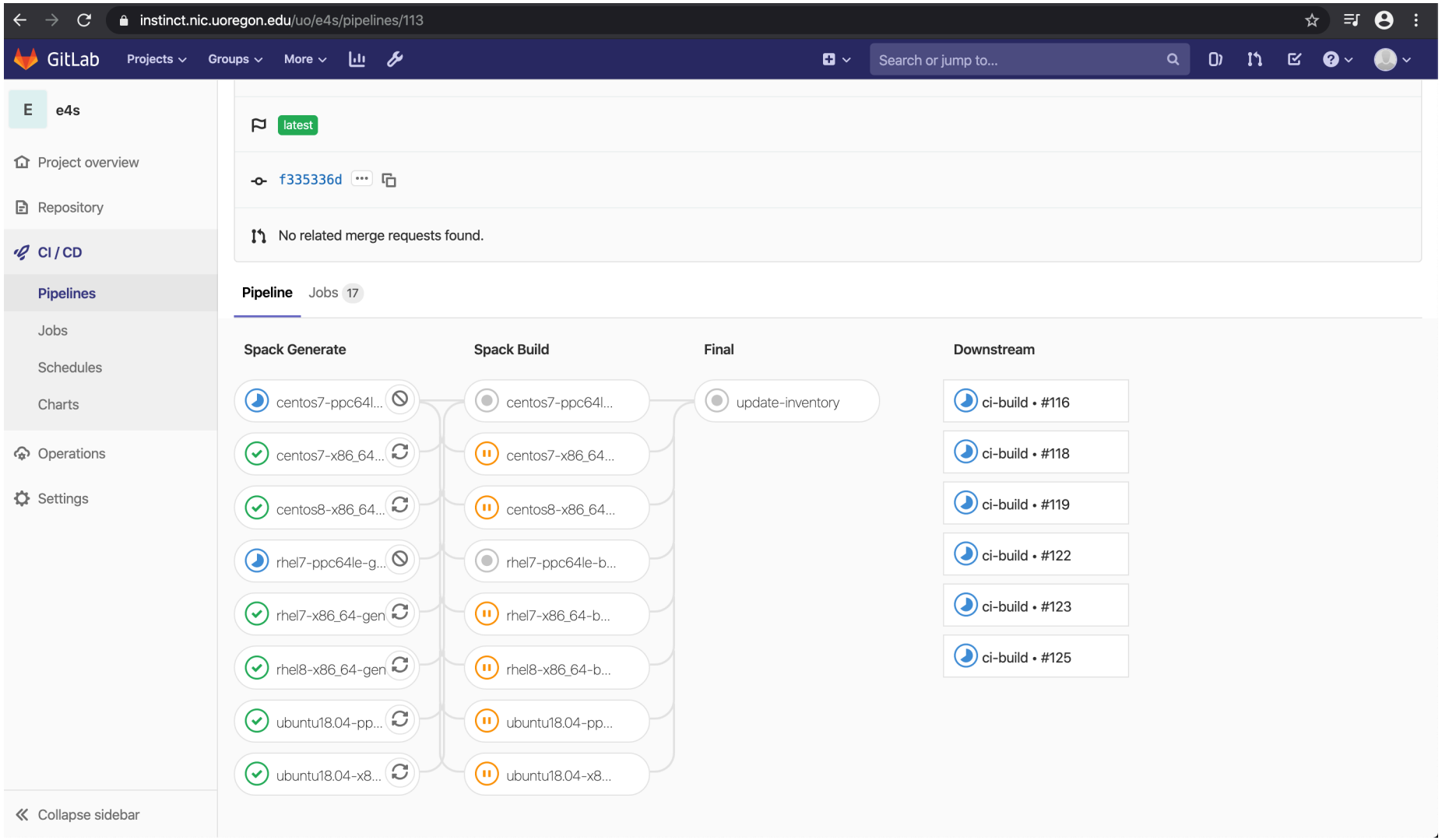
https://e4s.io

E4S Support for Singularity Container Runtime [Sylabs.io]



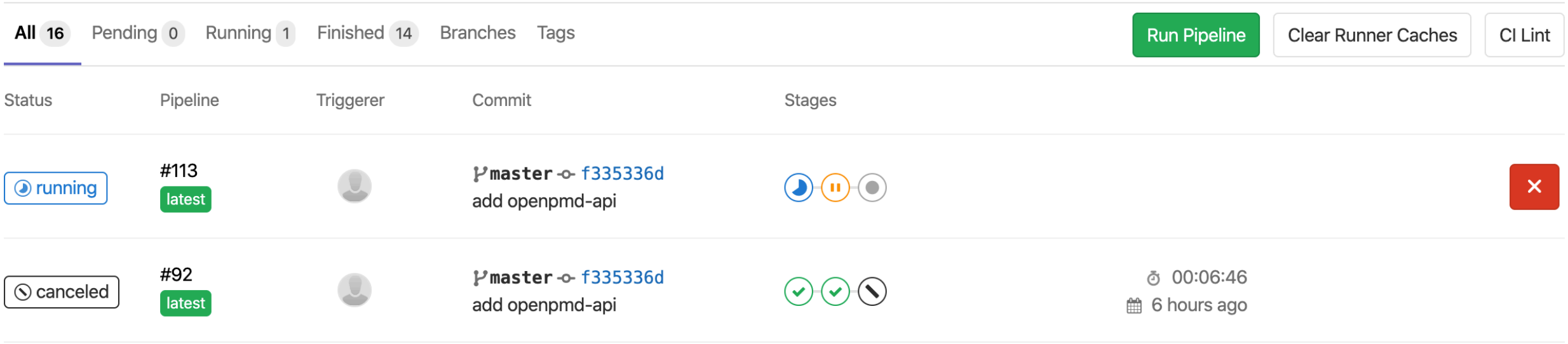
- wget http://oaciss.uoregon.edu/e4s/images/e4s_ubuntu1804_gpu_ppc64le_1.1.simg
- singularity exec --nv e4s_ubuntu1804_gpu_ppc64le_1.1.simg /bin/bash --rcfile /etc/bashrc
- spack find

E4S: Build Pipeline for Creating Spack Build Cache at U. Oregon



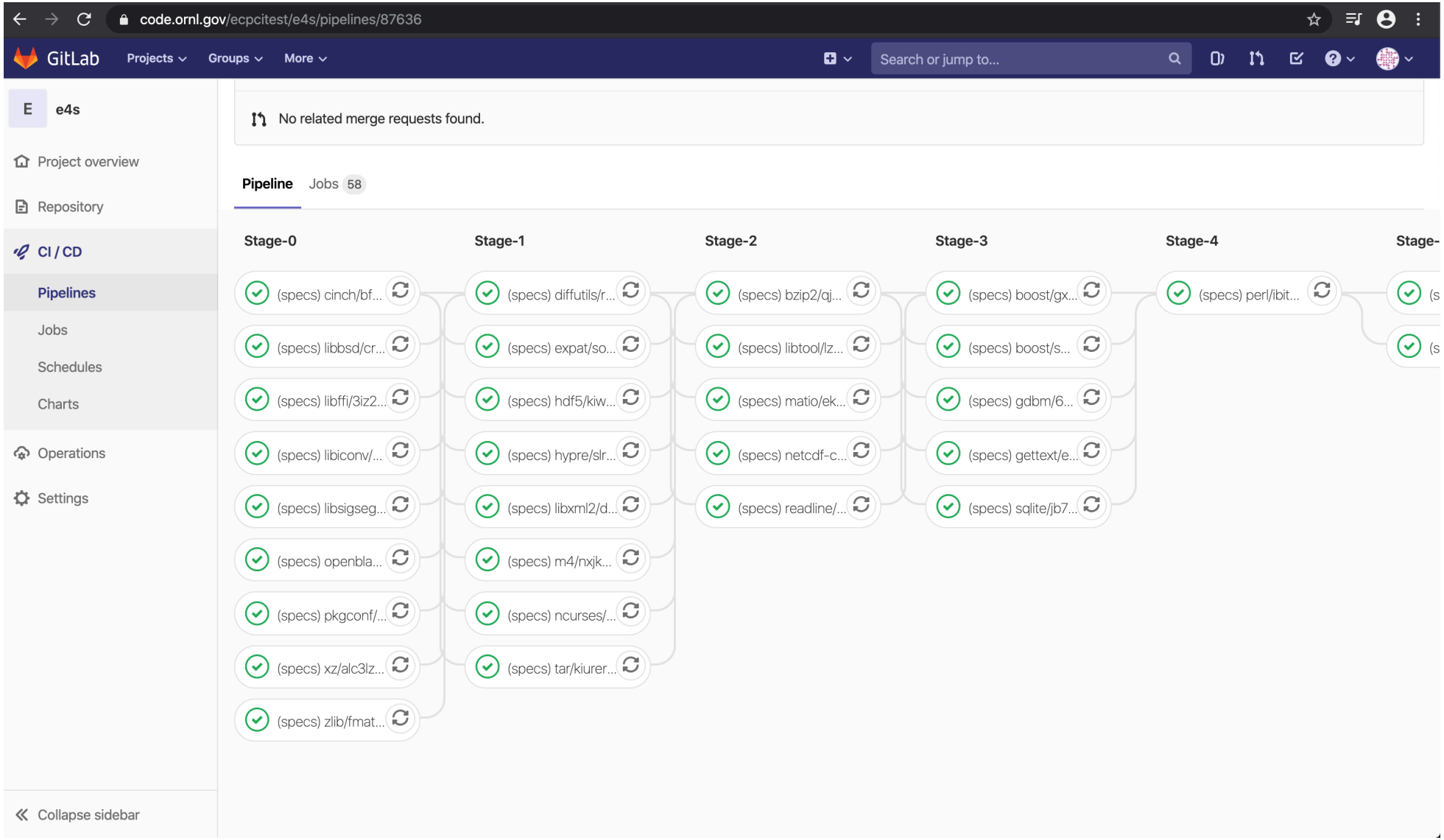
- GitLab
- Multi-platform builds

E4S: Build Pipeline at U. Oregon



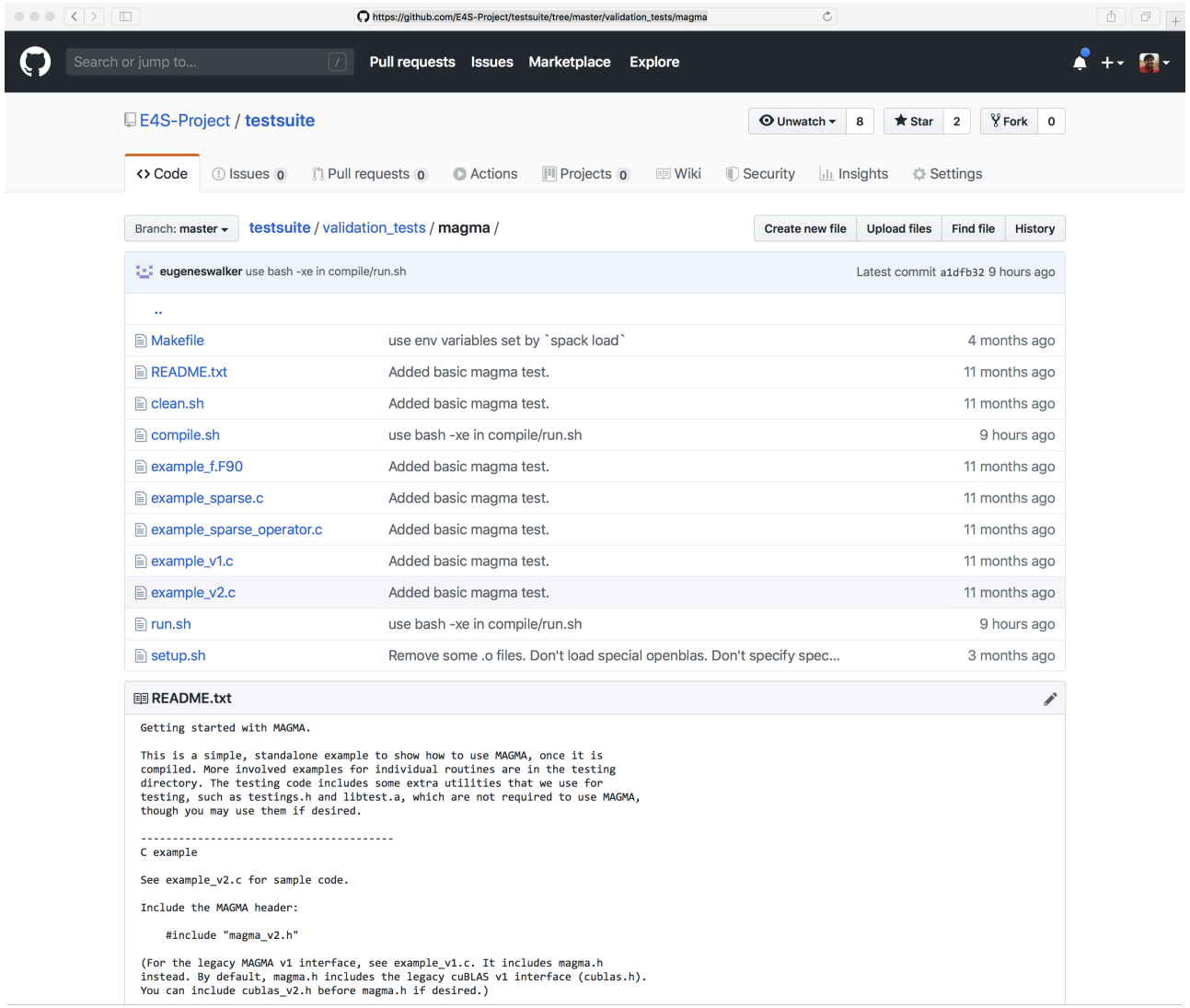
- Adding another package (openpmd-api) to E4S
- Automated build using GitLab

ORNL GitLab Build Pipeline for E4S Spack Build Cache



- ppc64le (Ascent @ ORNL)
- Reproducible container builds

E4S Validation Test Suite



To validate packages built in E4S

- git clone https://github.com/E4S-Project/testsuite.git

Reproducible Container Builds using E4S Base Images

UO-OACISS / e4s

Unwatch 3 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

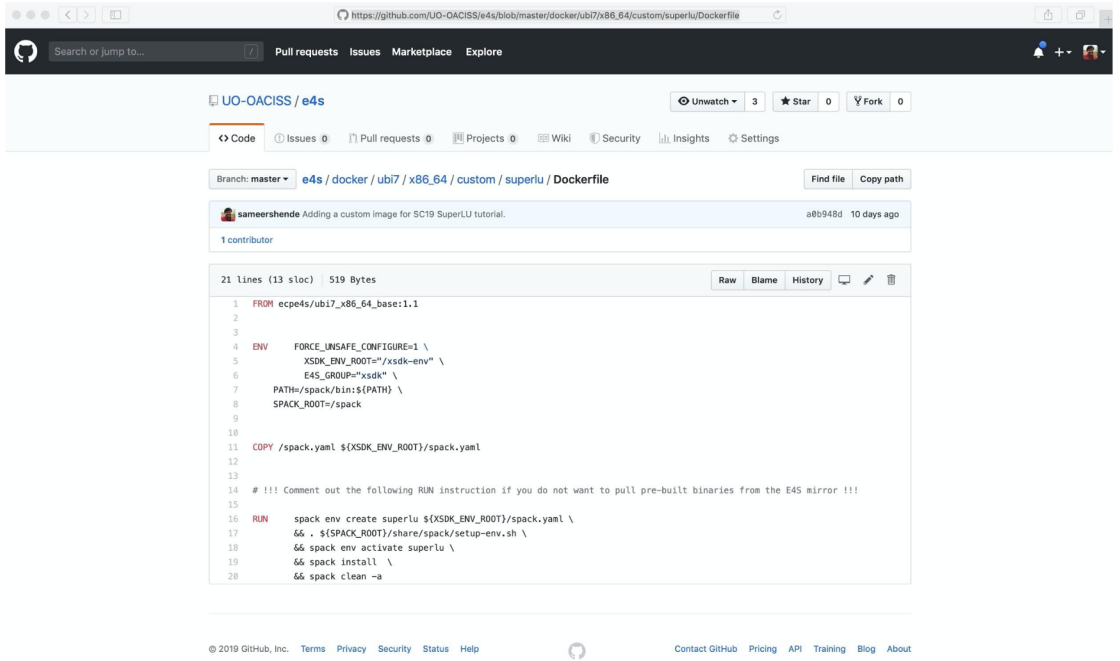
Branch: master — e4s / docker / ubi7 / x86_64 / custom / superlu / **spack.yaml** — Find file | Copy path

sameershende Adding a custom image for SC19 SuperLU tutorial. a0b948d 10 days ago

1 contributor

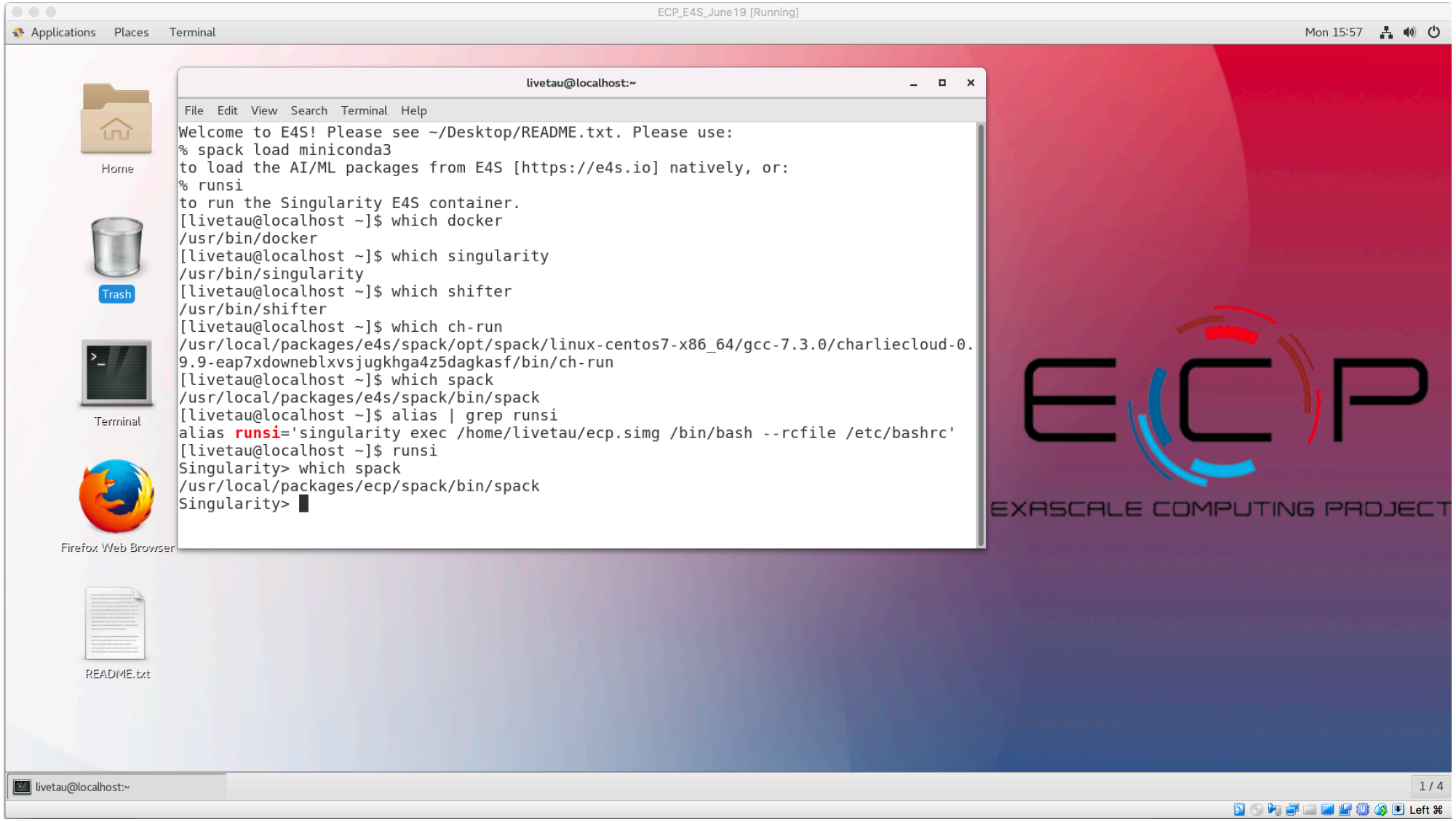
35 lines (34 sloc) | 579 Bytes — Raw | Blame | History

```
spack:
  packages:
    all:
      compiler: [gcc@7.3.0]
      variants: +mpi
      providers:
        mpi: [mpich]
      buildable: true
      version: []
      paths: {}
      modules: {}
    mpich:
      version: [3.2.1]
      variants: ~wrapperrpath
      buildable: true
      providers: {}
      paths: {}
      modules: {}
      compiler: []
    gcc:
      version: [7.3.0]
      buildable: true
      providers: {}
      paths: {}
      modules: {}
      compiler: []
  specs:
  - superlu-dist
  - petsc
  - mfem
  - strumpack
  - butterflypack
  - openblas
  view: false
```



- PMR SDK base image has Spack build cache mirror and GPG key installed.
- Base image has GCC and MPICH configured for MPICH ABI level replacement (with system MPI).
- Customized container build using binaries from E4S Spack build cache for fast deployment.
- No need to rebuild packages from the source code.
- Same recipe for container and native bare-metal builds with Spack!

E4S VirtualBox Image



Container Runtimes

- Docker
- Shifter
- Singularity
- Charliecloud

https://e4s.io

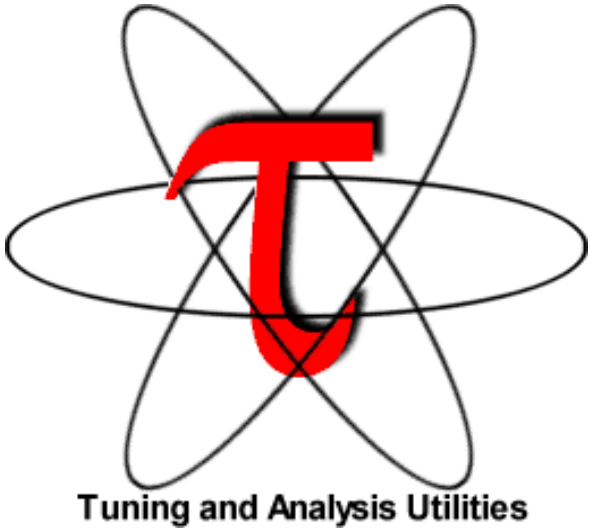
Future work, issues…

- Improved support for GPUs and visualization tools
- DOE LLVM
- Addition of CI testing
- Facility deployment
- Scalable startup with full-featured “Supercontainers”
- Improving the launch of MPI applications

Conclusions

- TAU provides a versatile tool to measure performance of HPC and AI/ML workloads
- TAU is part of the bigger ECP E4S project
- E4S provides a well integrated software stack for AI/ML and HPC for IBM ppc64le
- From-source builds assisted by a binary build cache or containers
- Docker and Singularity images are available for download
  - https://e4s.io

Download TAU from U. Oregon

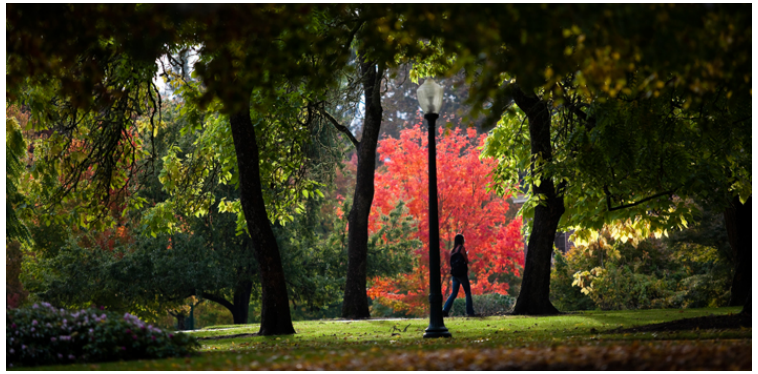


http://tau.uoregon.edu

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
- NASA
- CEA, France
- Partners:
  - –University of Oregon
  - –The Ohio State University
  - –ParaTools, Inc.
  - –University of Tennessee, Knoxville
  - –T.U. Dresden, GWT
  - –Jülich Supercomputing Center

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.”