

ALCF Hands on HPC Workshop 2024

TAU

<http://tau.uoregon.edu>

Sameer Shende

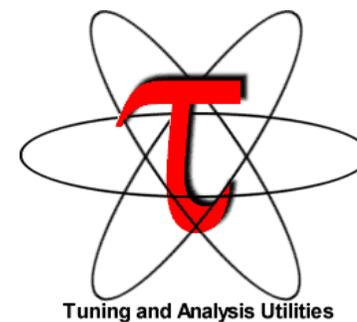
Research Professor and Director, Performance Research Laboratory

OACISS, U. Oregon

President and Director, ParaTools, Inc.

Track 1, TCS 1404, 4:00 – 5:15pm, Wednesday, October 30, 2024

sameer@cs.uoregon.edu



UNIVERSITY
OF OREGON



<https://www.alcf.anl.gov/events/2024-alc-f-hands-hpc-workshop>



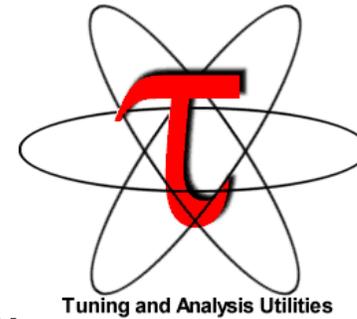
Motivation and Challenges

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

Motivation: Improving Productivity

- TAU Performance System[®]:
 - Deliver a scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads
 - <http://tau.uoregon.edu>
- 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
 - <https://e4s.io>

TAU Performance System®



UNIVERSITY
OF OREGON

ParaTools

- Versatile profiling and tracing toolkit that supports:
 - MPI, DPC++/SYCL (Level Zero), OpenCL, and OpenMP (OpenMP Tools Interface for Target Offload)
- Scalable, portable, performance evaluation toolkit for HPC and AI/ML workloads that supports:
 - C++/C/DPC++, Fortran, Python
- Supports PAPI, Likwid for hardware performance counter information
- Instrumentation includes support for Kokkos, MPI, pthread, event-based sampling, GPU runtimes
- A single tool (tau_exec) is used to launch un-instrumented, un-modified binaries
- TAU's paraprof, pprof, perfexplorer for profile analysis; Vampir, Jumpshot, Perfetto.dev for traces
- <http://tau.uoregon.edu>
- module load tau
 - to load TAU on Sunspot and other ALCF systems

Application Performance Engineering using TAU

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

Using TAU on Polaris at ALCF

```
% qsub -I -l select=1 -l filesystems=home:eagle -l walltime=1:00:00 -q R2035675 -A ATPESC2024 -X
```

```
% module use /soft/modulefiles; module load tau
```

```
% wget http://tau.uoregon.edu/workshop\_atpesc24.tgz; tar workshop_atpesc24.tgz
```

```
% cd workshop; cat README; cd TeaLeaf_CUDA; make; cd bin; ./run.sh; pprof -a
```

```
% cd ../../petsc-tau; ./clean.sh; ./compile.sh; ./run.sh
```

```
% paraprof --pack petsc_ex19.ppk ; <SCP to AWS>; paraprof petsc_ex19.ppk
```

- Un-instrumented run with MPI % aprun -n N ./a.out
- Profiling an un-instrumented application (use tau_exec -ebs with any of the following for event-based sampling):
- MPI without GPUs: % aprun -n N tau_exec -ebs ./a.out
- CUDA with MPI: % aprun -n N tau_exec -T cupti,mpi -cupti -ebs ./a.out

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

Tracing:

- Vampir: % export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2
 % aprun -n N tau_exec [options] ./a.out; vampir traces.otf2 &

- Chrome: % export TAU_TRACE=1; aprun -n N 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) or https://Perfetto.dev

- Jumpshot: % export TAU_TRACE=1; aprun -n N tau_exec [Options] ./a.out;
 % tau_treemerge.pl; tau2slog2 tau.trc tau.edf -o app.slog2; jumpshot app.slog2

TAU: Quickstart Guide for Sunspot.alcf.anl.gov

```
% tar xf /soft/perftools/tau/tar/workshop.tgz; cd workshop; cat README
```

```
% qsub -I -l select=<N> -l walltime=1:00:00 -A <Project_id>
```

```
% module use /soft/modulefiles; module load tau
```

- Un-instrumented run with MPI `% mpirun -np N ./a.out`
- Profiling an un-instrumented application (use `tau_exec -ebs` with any of the following for event-based sampling):
- MPI without GPUs: `% mpirun -np N tau_exec -ebs ./a.out`
- DPC++/SYCL with MPI: `% mpirun -np N tau_exec -T level_zero -10 ./a.out`
- DPC++/SYCL without MPI: `% TAU_SET_NODE=0 tau_exec -T level_zero -10 ./a.out`
- OpenMP with MPI: `% mpirun -np N tau_exec -T ompt -ompt ./a.out`
- OpenMP without MPI: `% TAU_SET_NODE=0 tau_exec -T ompt -ompt ./a.out`
- OpenCL with MPI: `% mpirun -np N tau_exec -T level_zero -opencl ./a.out`
- OpenCL without MPI: `% TAU_SET_NODE=0 tau_exec -T level_zero -opencl ./a.out`

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

Tracing:

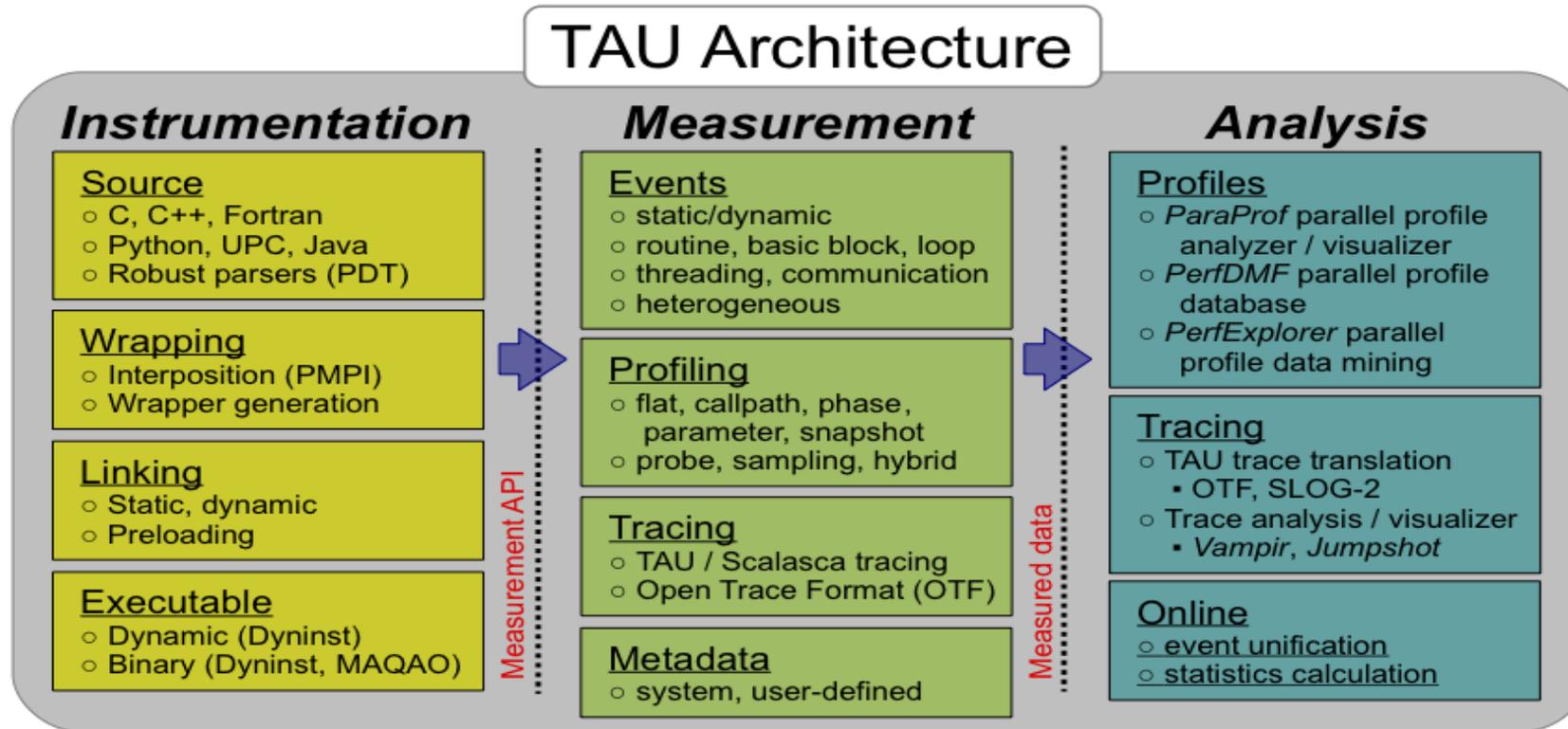
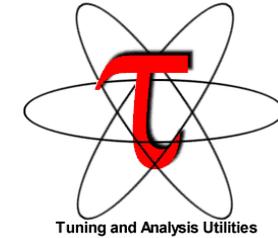
- Vampir: `% export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2`
`% mpirun -np N tau_exec [Options] ./a.out; vampir traces.otf2 &`
- Chrome: `% export TAU_TRACE=1; mpirun -np N tau_exec ./a.out; tau_treemerge.pl;`
`% tau_trace2json tau.trc tau.edf -chrome -ignoreatomic -o app.json`
Chrome browser: `chrome://tracing` (Load -> app.json) or `https://Perfetto.dev`
- Jumpshot: `% export TAU_TRACE=1; mpirun -np N tau_exec [Options] ./a.out;`
`% tau_treemerge.pl; tau2slog2 tau.trc tau.edf -o app.slog2; jumpshot app.slog2 &`

TAU Performance System[®]

Parallel performance framework and toolkit

Supports all HPC platforms, compilers, runtime system

Provides portable instrumentation, measurement, analysis



TAU Performance System[®]

Instrumentation

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

Measurement and analysis support

- MPI (MVAPICH2, Intel MPI), OpenSHMEM, ARMCI, PGAS, DMAPP
- Supports Intel oneAPI compilers
- pthreads, OpenMP, OMPT interface, hybrid, other thread models
- GPU: OpenCL, oneAPI DPC++/SYCL (Level Zero), OpenACC, Kokkos, RAJA
- Parallel profiling and tracing

Analysis

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

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.

Configure TAU on Polaris

Support for MPI and CUDA

- `./configure -c++=CC -cc=cc -fortran=ftn -
cuda=/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2 -pdt=/soft/perftools/tau/pdtoolkit-
3.25.2 -bfd=download -otf=download -dwarf=download -iowrapper -mpi -
papi=/opt/cray/pe/papi/7.0.1.2/ -pthread`
- `make install -j`

Builds `craycnl/lib/Makefile.tau-nvidia-papi-mpi-cupti-pdt`
and `craycnl/lib/shared-nvidia-papi-mpi-cupti-pdt/libTAU.so`

- We can build multiple configurations of TAU with `PrgEnv-nvhpc` and `PrgEnv-gnu`

Configurations of TAU installed on Polaris

```
module use /soft/modulefiles; module load tau; ls $TAU/Makefile*  
  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-gnu-papi-mpi-cupti-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-gnu-papi-mpi-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-gnu-papi-mpi-pthread-cupti-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-gnu-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-gnu-tbb-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-nvidia-papi-mpi-cupti-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-nvidia-papi-mpi-pdt  
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-nvidia-pdt
```

```
aprun -n 16 tau_exec -T gnu-papi-mpi-pthread-cupti-pdt -ebs ./a.out  
will choose a configuration represented by:
```

```
/soft/perftools/tau/tau-2.33.2/craycnl/lib/Makefile.tau-gnu-papi-mpi-pthread-cupti-pdt
```

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:

```
% module load tau
```

```
% export TAU_MAKEFILE=/soft/perftools/tau/tau-2.33.2/craycn1/lib/Makefile.tau-gnu-papi-mpi-pthread-cupti-pdt
```

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

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

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

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

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

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

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

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

TAU's Support for Runtime Systems

- *MPI*
 - PMPI profiling interface
 - MPI_T tools interface using performance and control variables
- *Pthread*
 - Captures time spent in routines per thread of execution
- *OpenMP*
 - OMPT tools interface to track salient OpenMP runtime events
 - Opari source rewriter
 - Preloading wrapper OpenMP runtime library when OMPT is not supported
- *OpenACC*
 - OpenACC instrumentation API
 - Track data transfers between host and device (per-variable)
 - Track time spent in kernels

TAU's Support for Runtime Systems (contd.)

- *OpenCL*
 - OpenCL profiling interface
 - Track timings of kernels
- *Intel® OneAPI*
 - Level Zero
 - Track time spent in kernels executing on GPU
 - Track time spent in OneAPI runtime calls
- *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 + pthread*
 - PMPI + pthread interfaces
- *MPI + Intel[®] oneAPI DPC++/SYCL*
 - PMPI + Level Zero interfaces
- *OpenCL + Python*
 - OpenCL + Python instrumentation interfaces
- *Kokkos + OpenMP*
 - Kokkos profiling API + OMPT to transparently track events
- *Kokkos + pthread + MPI*
 - Kokkos + pthread wrapper interposition library + PMPI layer
- *MPI + OpenCL*
 - PMPI + OpenCL profiling interfaces

Binary instrumentation of libraries: Work in progress

```
% tau_run a.out -o a.inst
```

instruments a binary. Other flags `-T <tags>`, `-f <selective instrumentation file>`

```
% tau_run -l /path/to/libhdf5.so.310 -o libhdf5.so.310
```

instruments a DSO

```
% tau_exec ./a.out
```

executes the uninstrumented application with the instrumented shared object.

To use with DyninstAPI 13 on x86_64:

1. Load spack: `source spack/share/spack/setup-env.sh`

2. Install dyninst: `spack install dyninst@13 %gcc@11`

3. Configure tau with dyninst:

3.1 `spack find -p dyninst boost tbb elfutils`

3.2 Copy the paths for each package into the configure line

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

Binary instrumentation of libraries: HDF5



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

```
NODE 0;CONTEXT 0;THREAD 0:
```

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

Using TAU's Runtime Preloading Tool: tau_exec

Preload a wrapper that intercepts the runtime system call and substitutes with another

MPI

OpenMP

POSIX I/O

Memory allocation/deallocation routines

Wrapper library for an external package

No modification to the binary executable!

Enable other TAU options (communication matrix, OTF2, event-based sampling)

TAU Execution Command (tau_exec)

Uninstrumented execution

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

Track GPU operations

```
% mpirun -np 256 tau_exec -l0 ./a.out
```

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

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

Track MPI performance

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

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

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

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

```
% export TAU_OMPT_SUPPORT_LEVEL=full;
```

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

Track memory operations

```
% export TAU_TRACK_MEMORY_LEAKS=1
```

```
% mpirun -np 256 tau_exec -memory_debug ./a.out (bounds check)
```

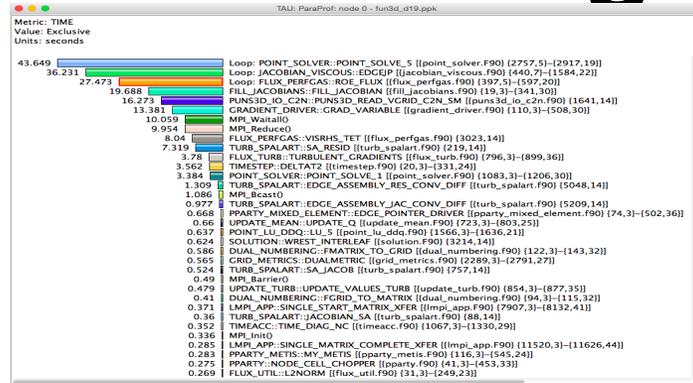
Use event based sampling (compile with -g)

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

```
Also export TAU_METRICS=TIME,PAPI_L1_DCM... -ebs_resolution=<file | function | line>
```

Profiling and Tracing

Profiling



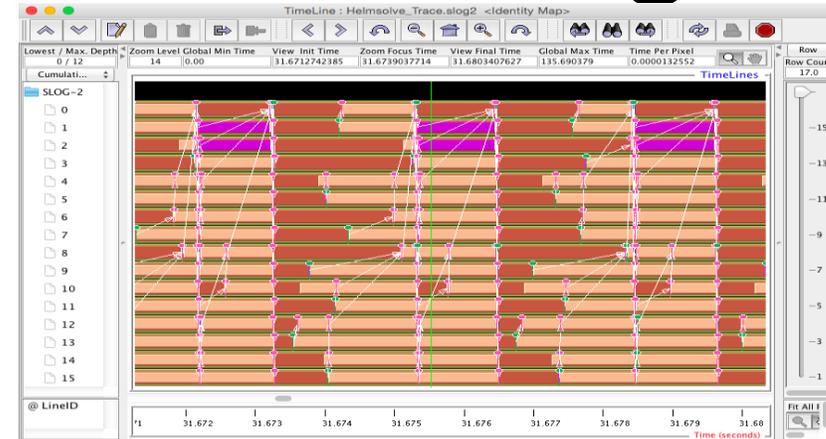
- **Profiling** shows you **how much** (total) time was spent in each routine

- 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

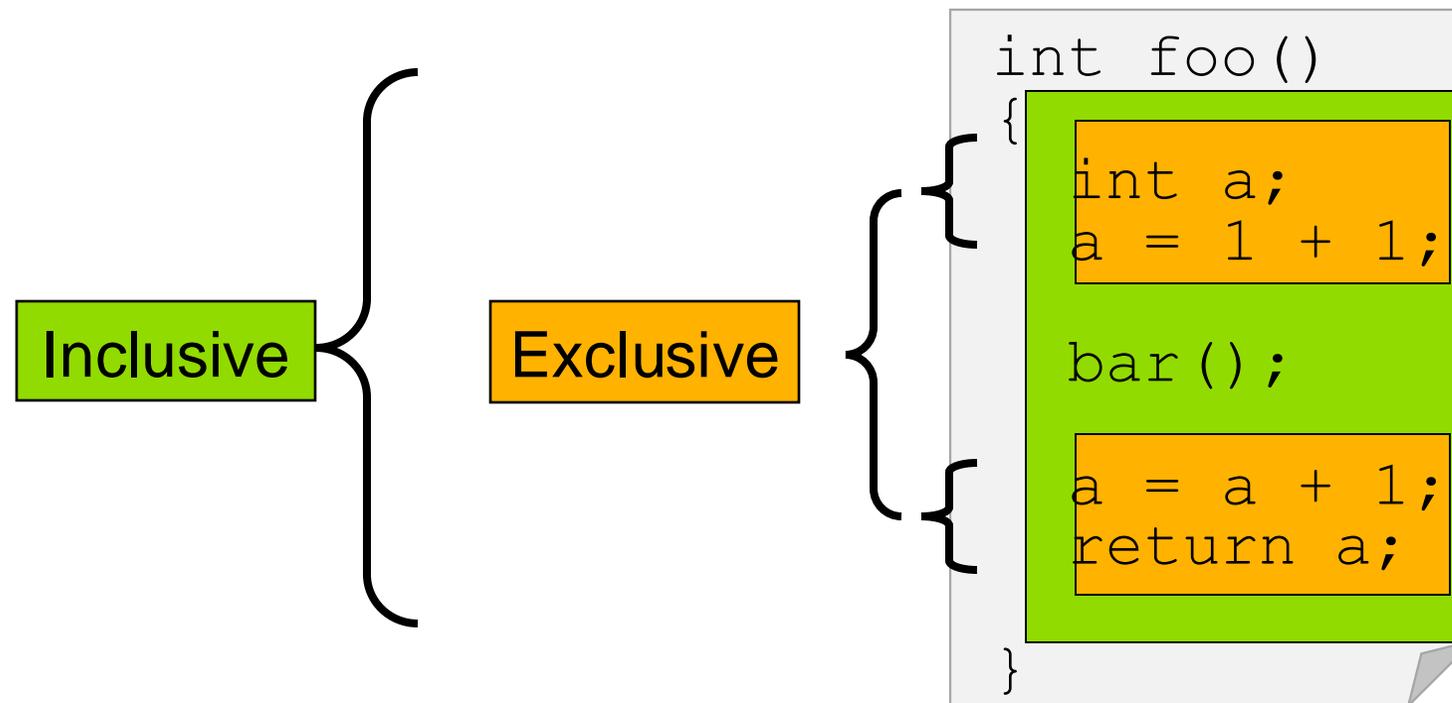
Tracing



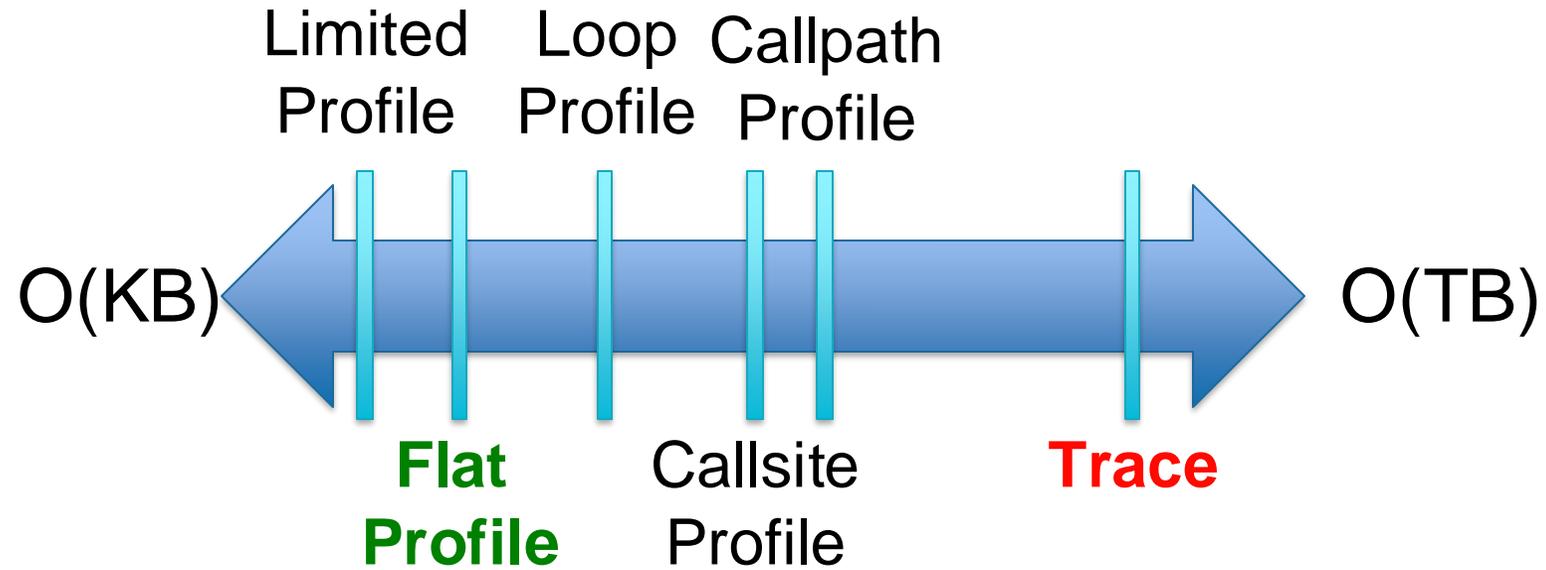
- **Tracing** shows you **when** the events take place on a timeline

Inclusive vs. Exclusive values

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



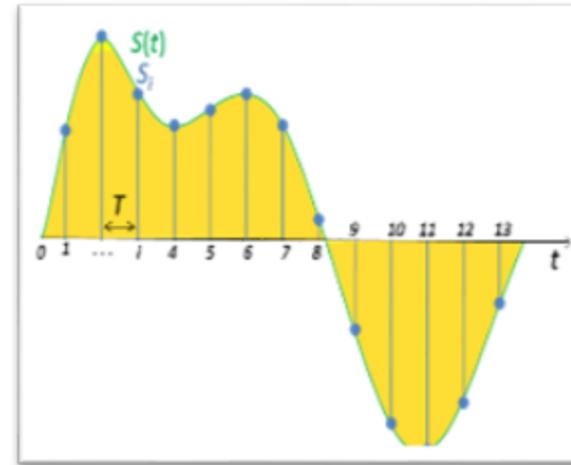
How much data do you want?



Performance Data Measurement

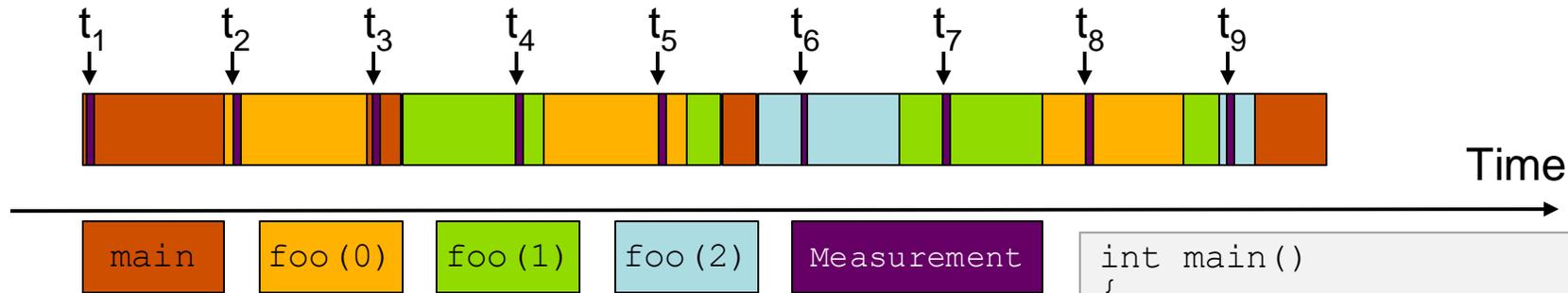
```
Call START('potential')  
// code  
Call STOP('potential')
```

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



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

Sampling



Running program is periodically interrupted to take measurement

Timer interrupt, OS signal, or HWC overflow

Service routine examines return-address stack

Addresses are mapped to routines using symbol table information

Statistical inference of program behavior

Not very detailed information on highly volatile metrics

Requires long-running applications

Works with unmodified executables

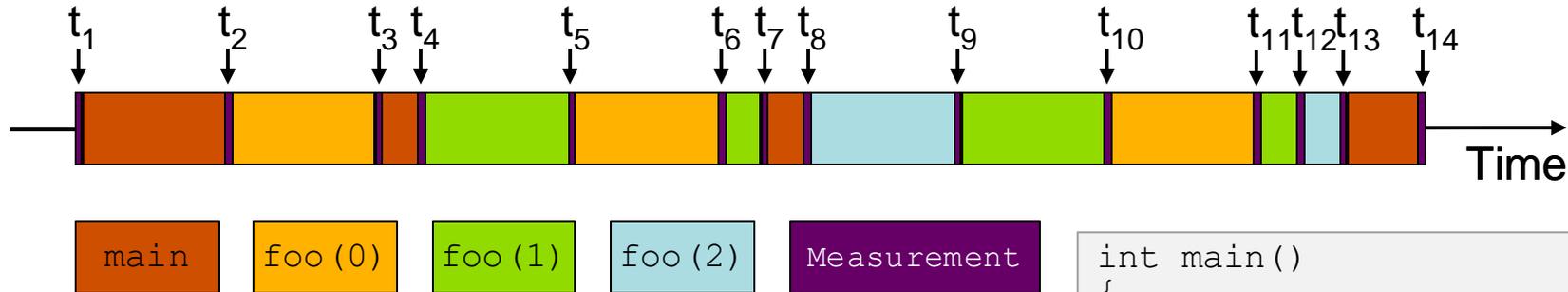
```
int main()
{
    int i;

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

    return 0;
}

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

Instrumentation



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

Can be done in various ways

Advantage:

Much more detailed information

Disadvantage:

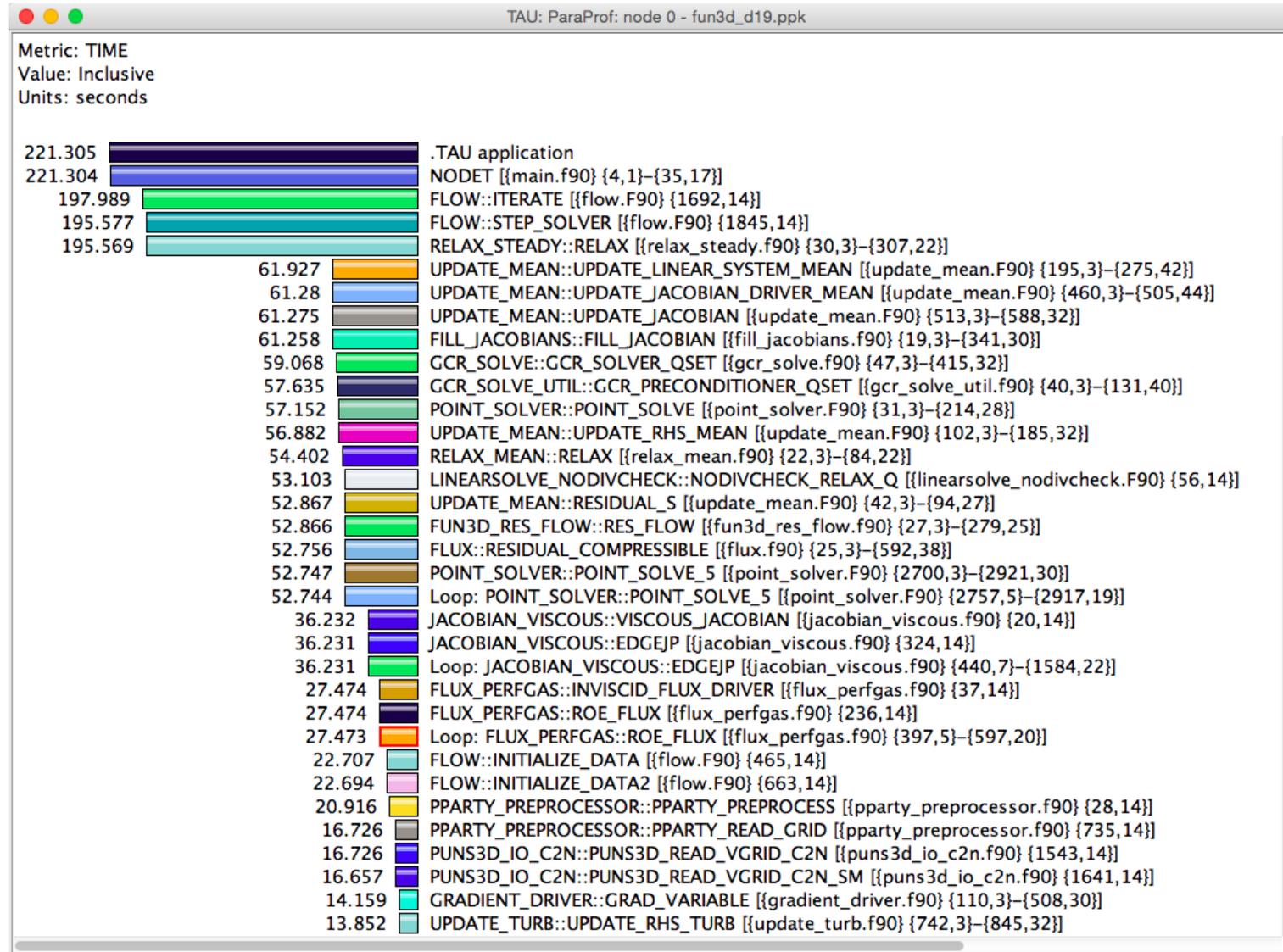
Processing of source-code / executable necessary

Large relative overheads for small functions

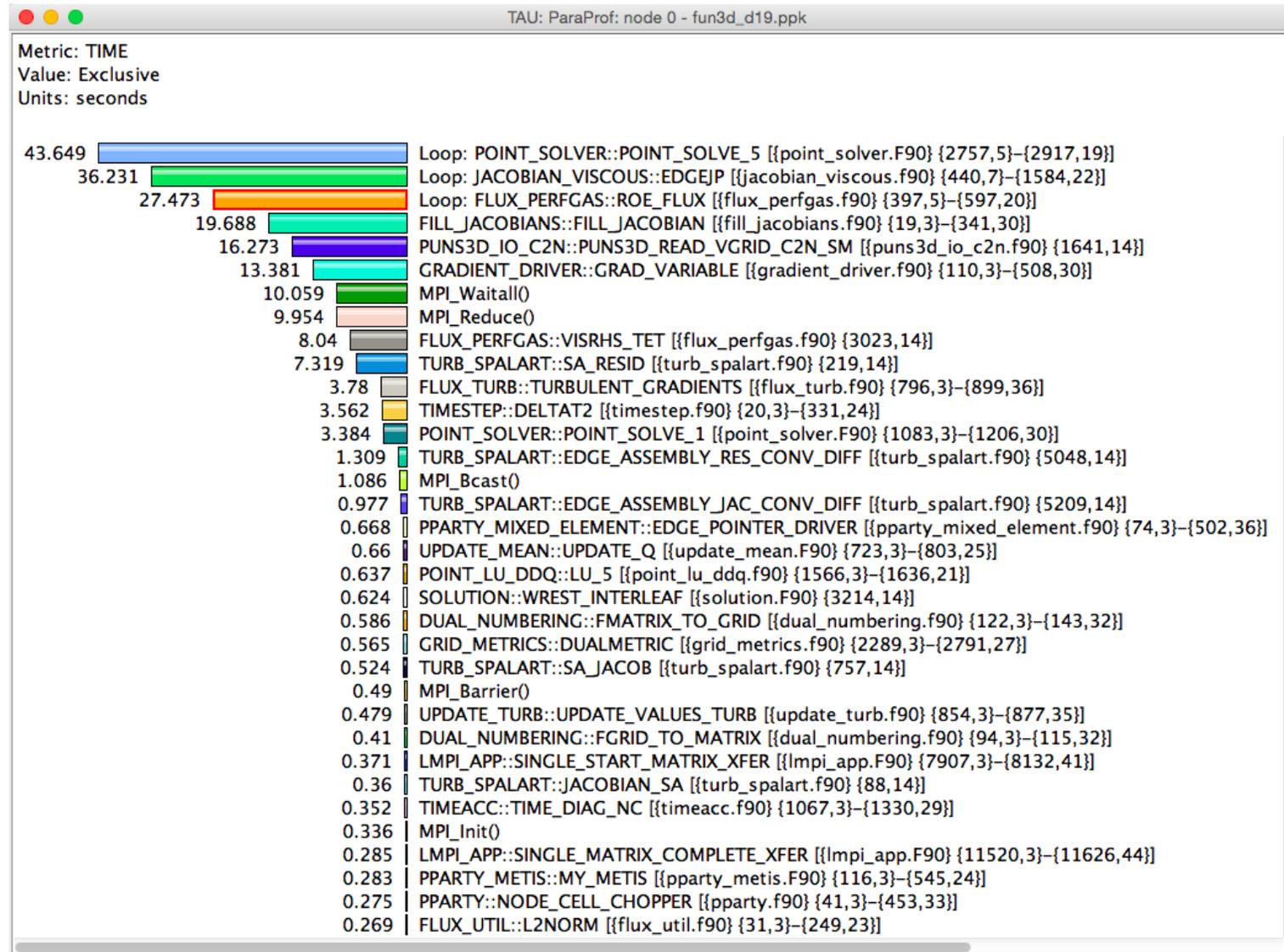
```
int main()
{
    int i;
    TAU_START("main");
    for (i=0; i < 3; i++)
        foo(i);
    TAU_STOP("main");
    return 0;
}

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

Inclusive Measurements



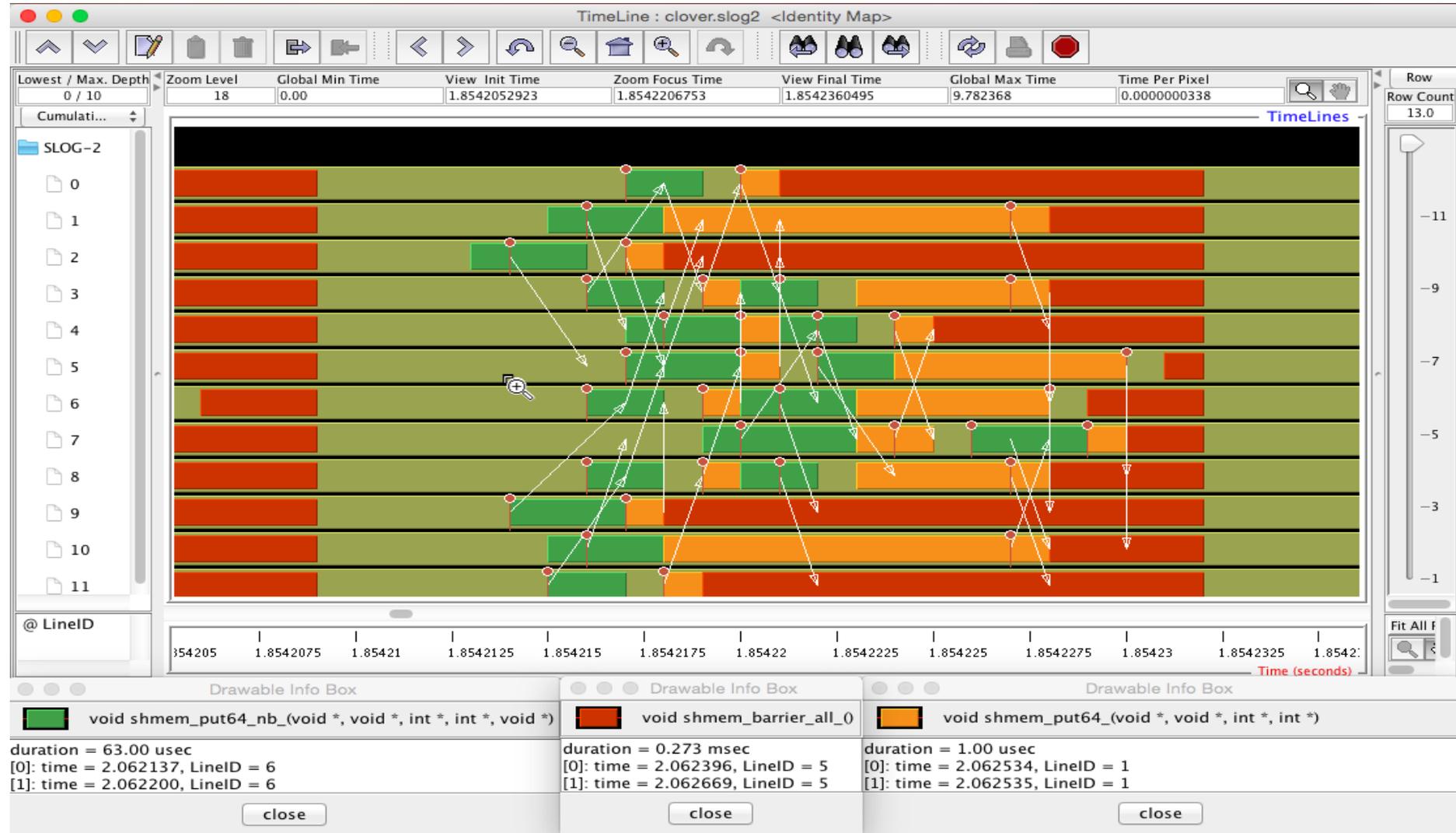
Exclusive Time



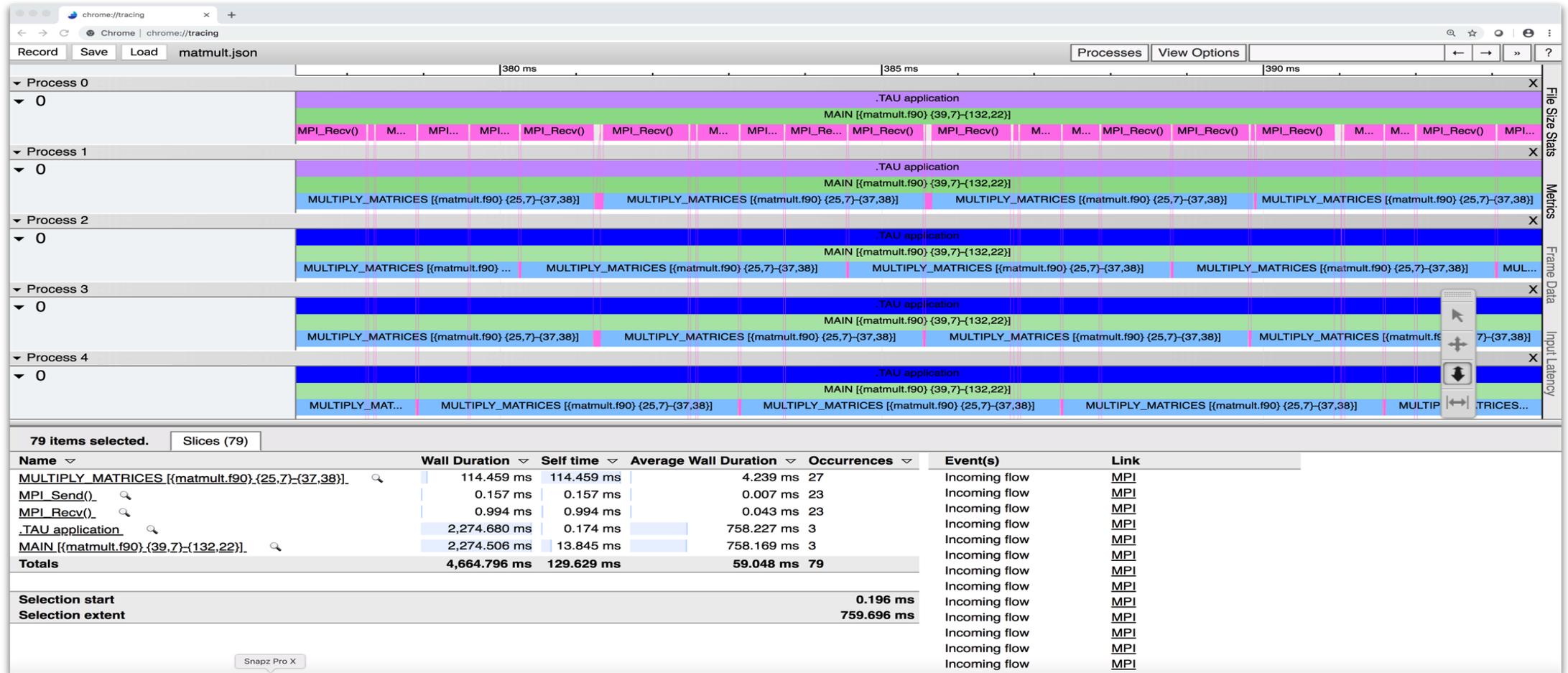
TAU's Runtime Environment Variables

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>)

Tracing: Jumpshot (ships with TAU)



Tracing: Chrome Browser



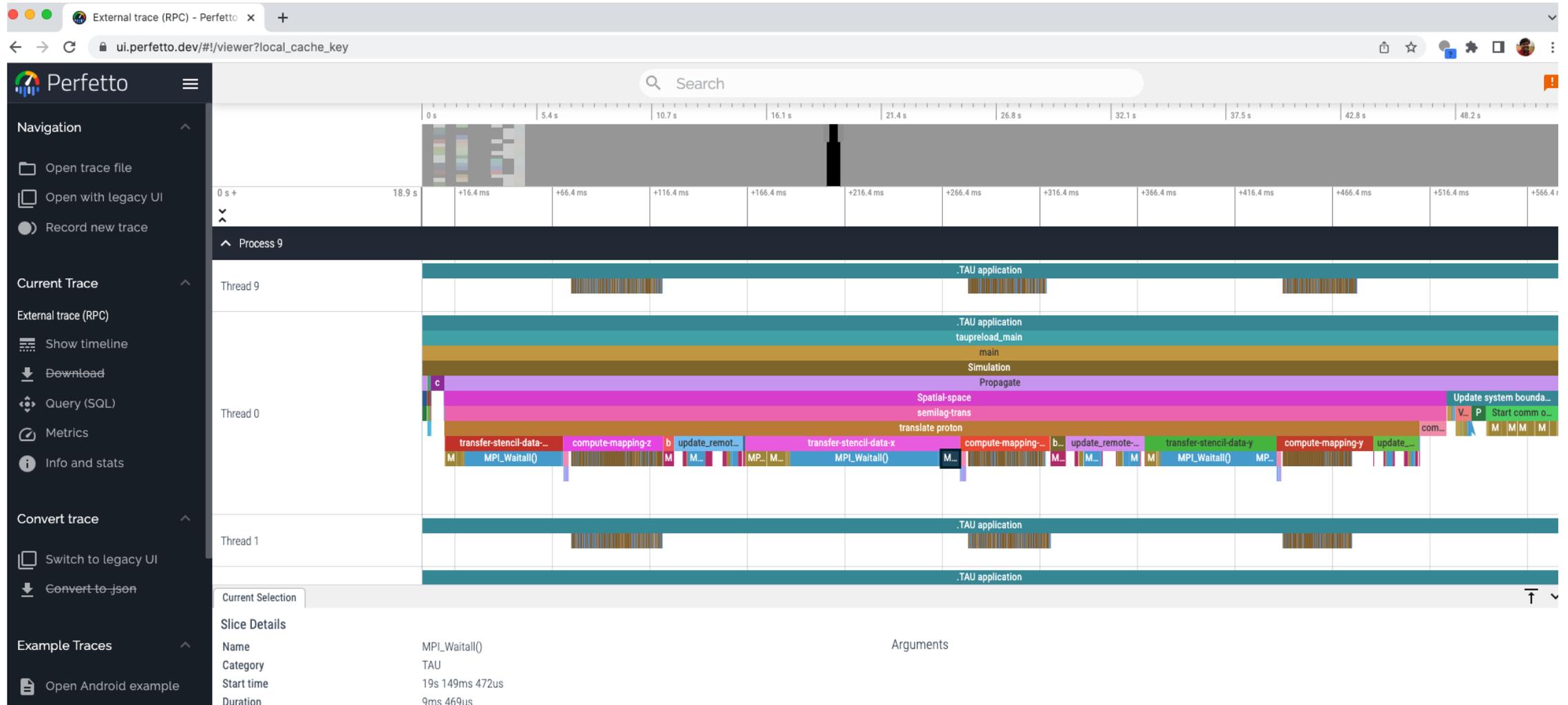
```
% export TAU_TRACE=1
```

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

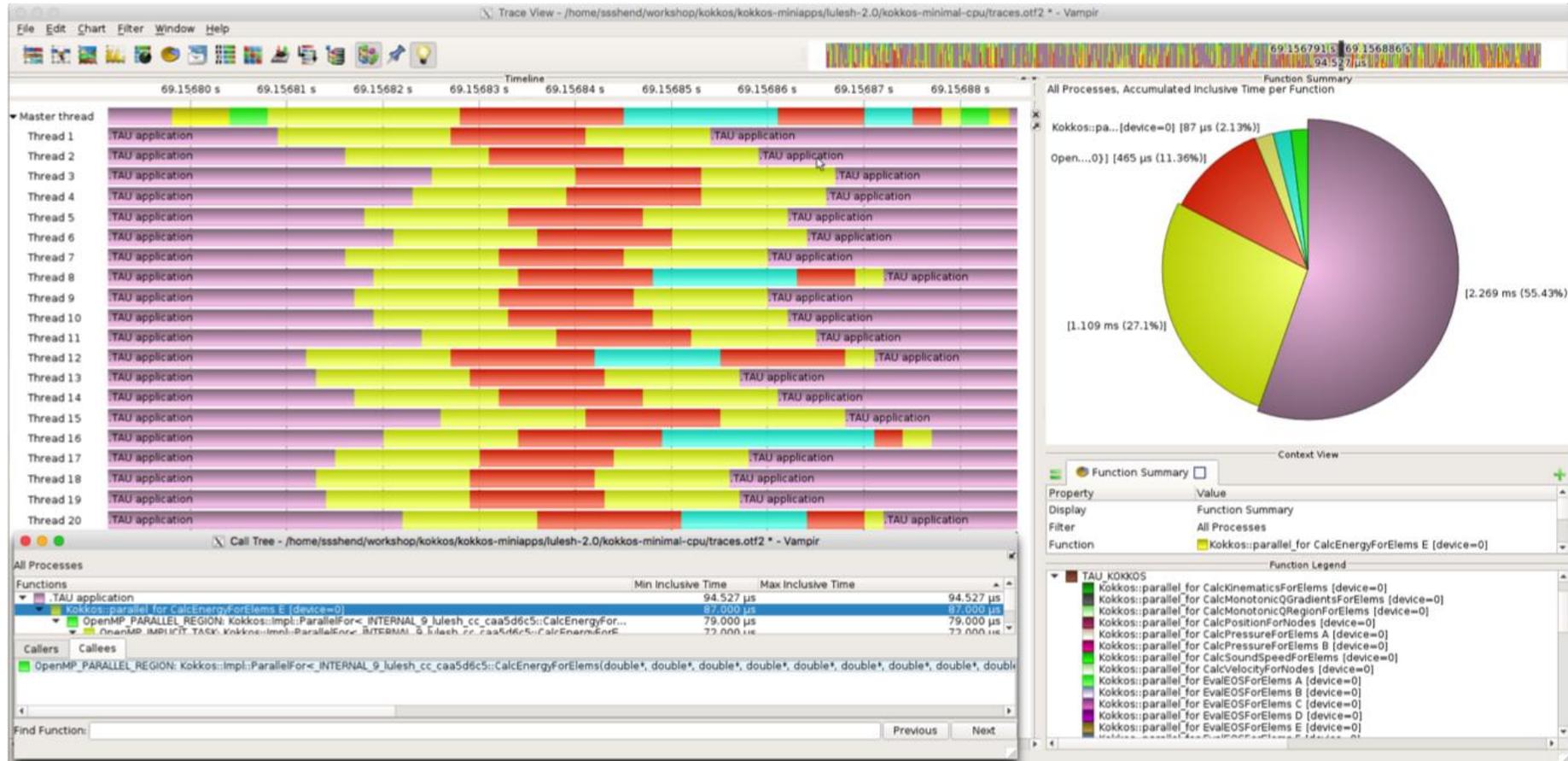
```
% tau_treemerge.pl; tau_trace2json tau.trc tau.edf -chrome -ignoreatomic -o app.json
```

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

Perfetto.dev



Vampir [TU Dresden] Timeline: Kokkos



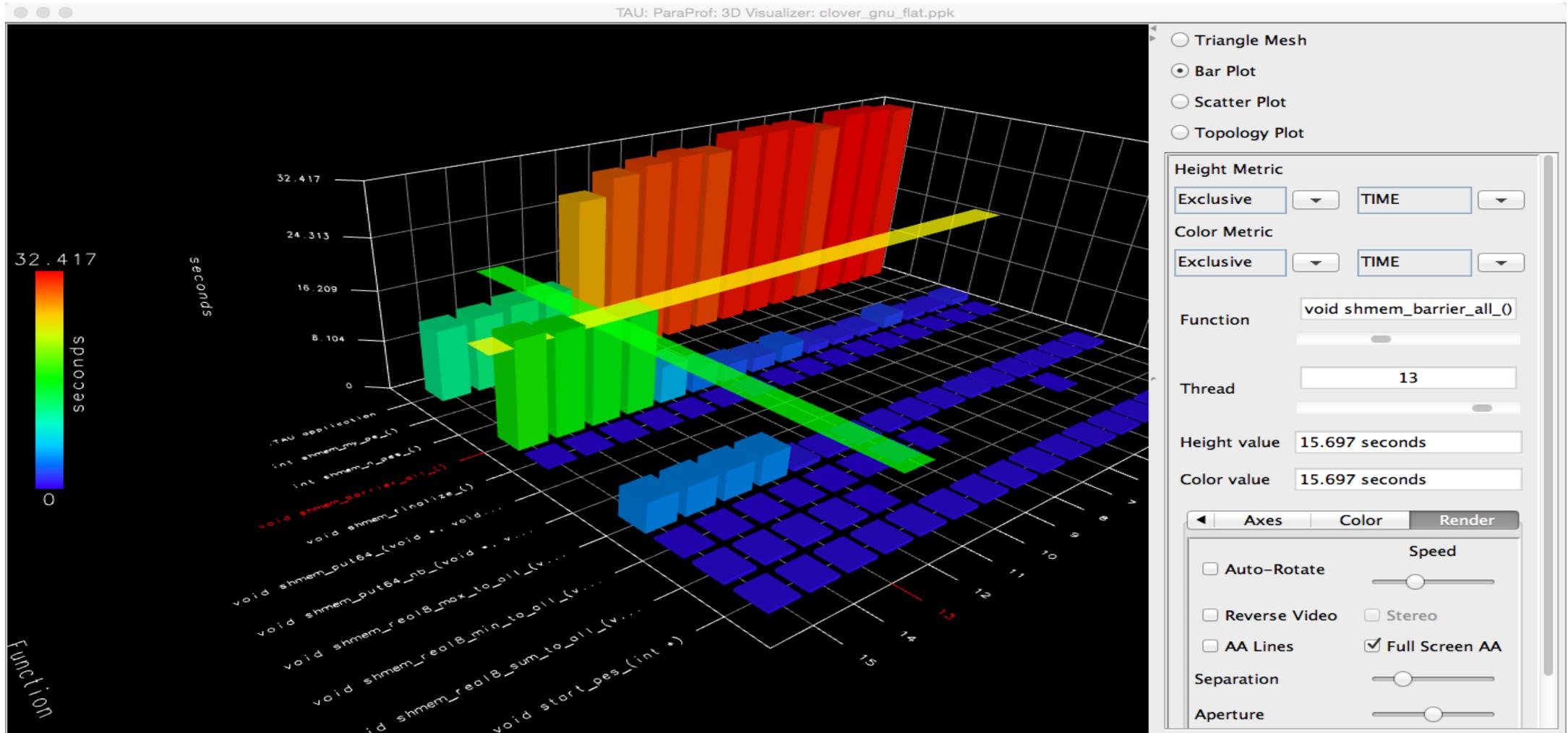
```
% export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2
% tau_exec -T ompt -ompt ./a.out
% vampir traces.otf2 &
```

ParaProf Profile Browser



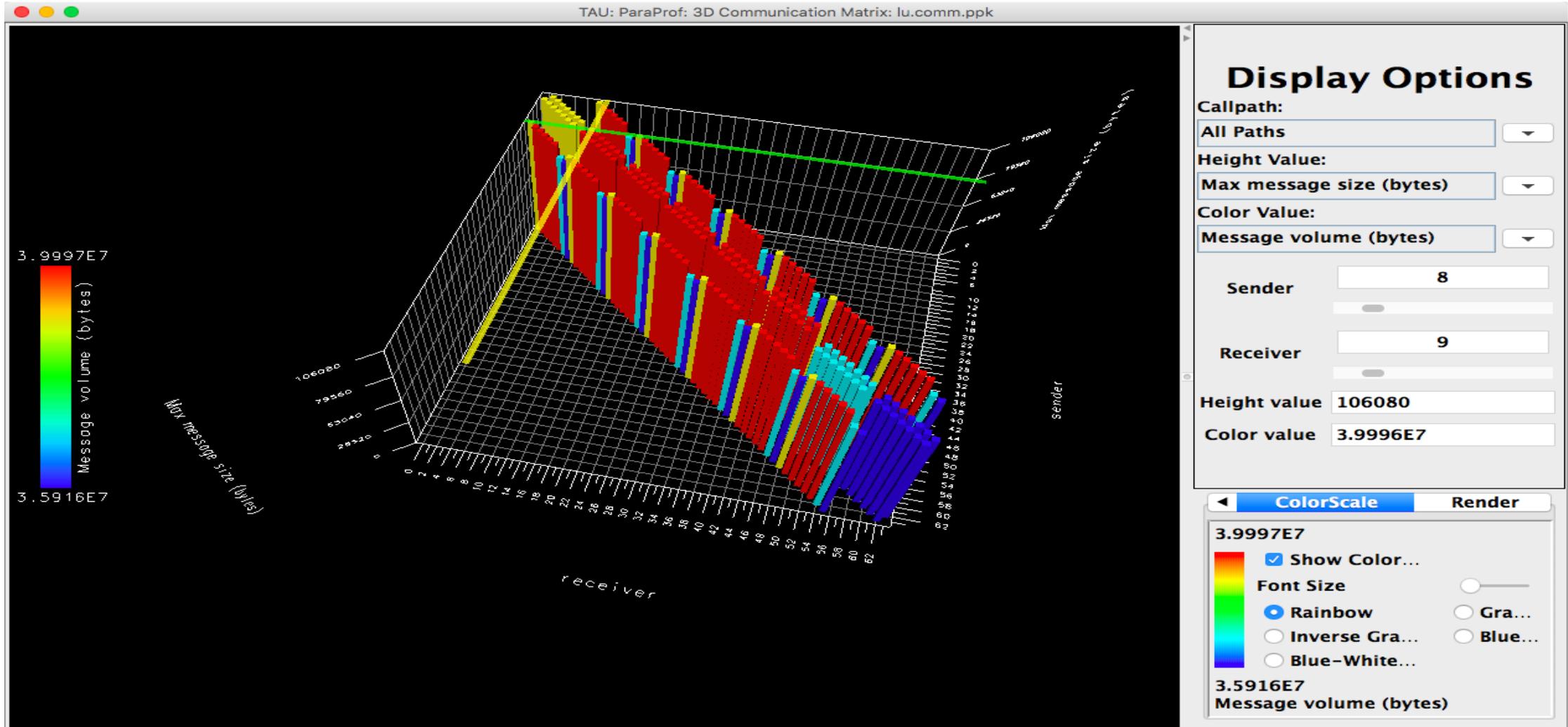
% paraprof

TAU – ParaProf 3D Visualization



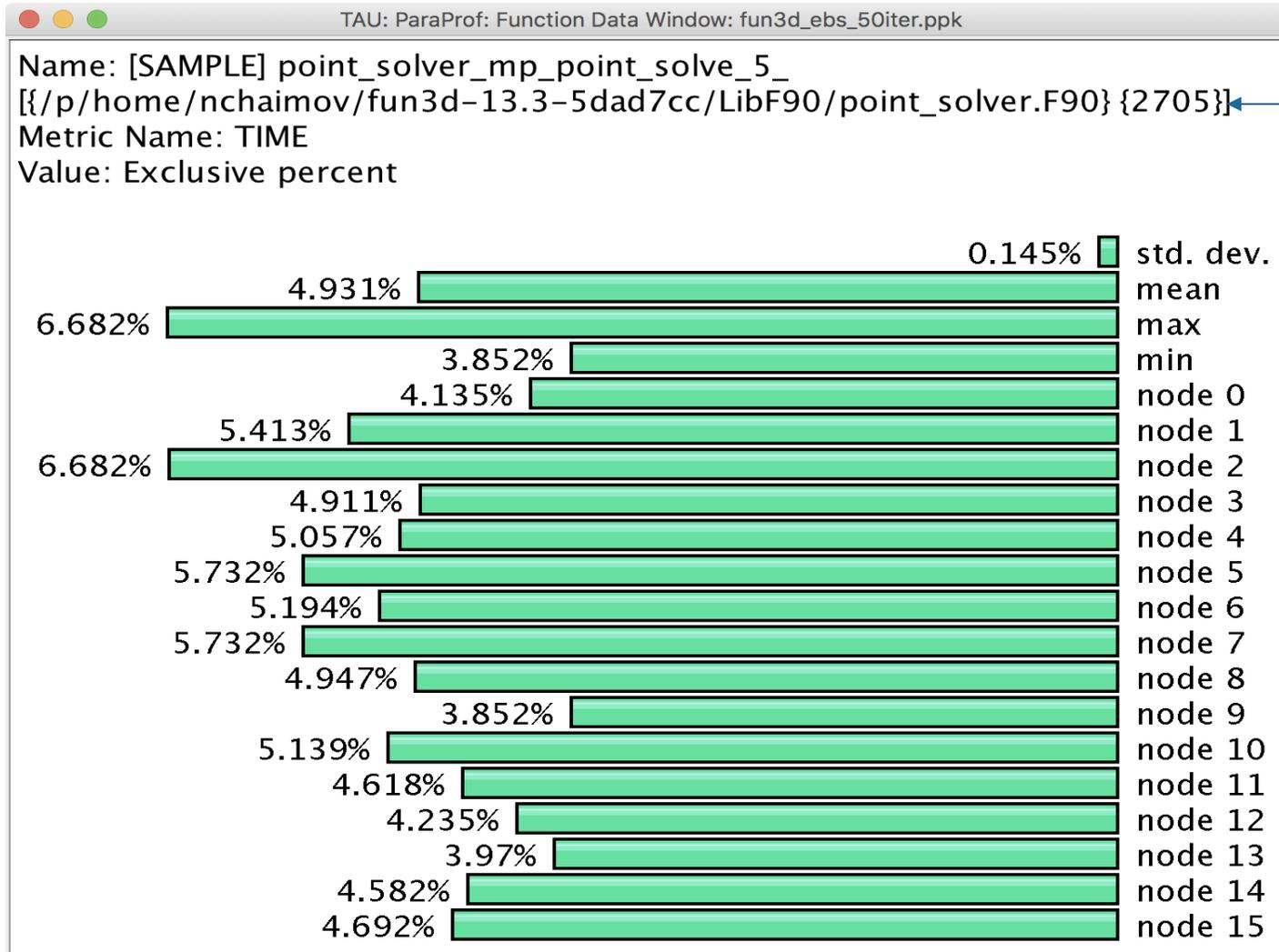
% paraprof app.ppk
Windows -> 3D Visualization -> Bar Plot (right pane)

TAU – 3D Communication Window



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

Event Based Sampling (EBS)



File: point_solver.F90
Line: 2705

Uninstrumented!

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

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>)

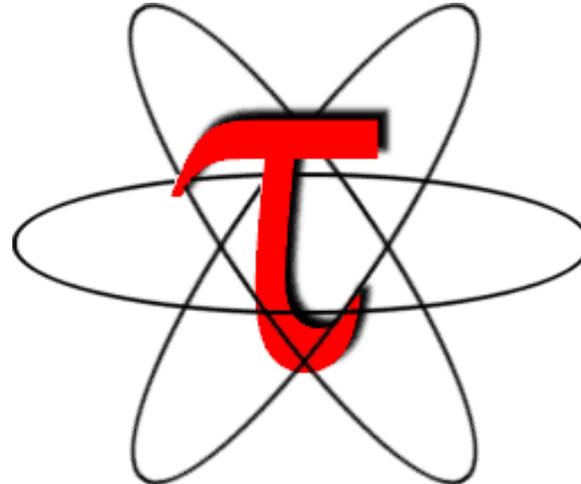
Runtime Environment Variables

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 <code>-otf=download</code>)
TAU_EBS_UNWIND	0	Setting to 1 turns on unwinding the callstack during sampling (use with <code>tau_exec -ebs</code> or <code>TAU_SAMPLING=1</code>)
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

Download TAU from U. Oregon



Tuning and Analysis Utilities

<http://tau.uoregon.edu>

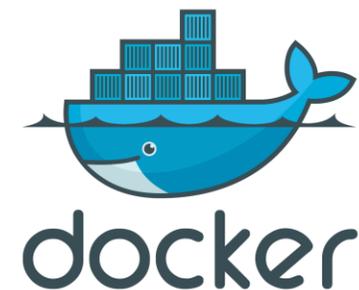
<https://e4s.io> [TAU in Docker/Singularity containers]

for more information

Free download, open source, BSD license

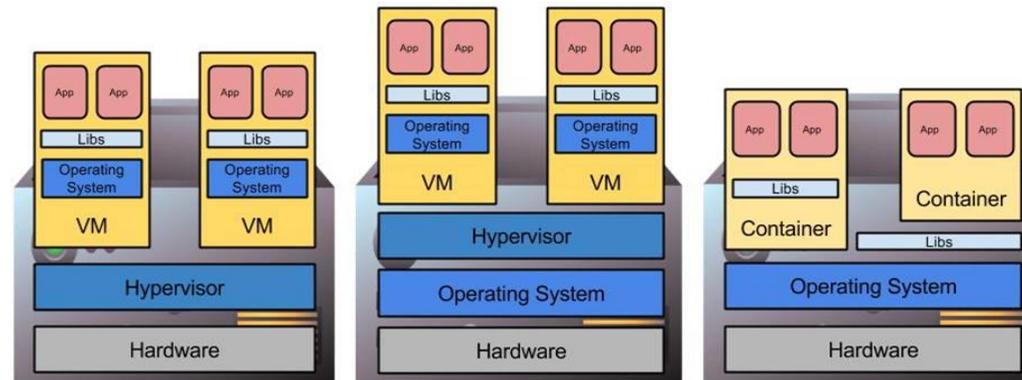
What are containers

- A lightweight collection of executable software that encapsulates everything needed to run a single specific task
 - Minus the OS kernel
 - Based on Linux only
- Processes and all user-level software is isolated
- Creates a portable* software ecosystem
- Think `chroot` on steroids
- Docker most common tool today
 - Available on all major platforms
 - Widely used in industry
 - Integrated container registry via Dockerhub



Hypervisors and Containers

- Type 1 hypervisors insert layer below host OS
- Type 2 hypervisors work as or within the host OS
- Containers do not abstract hardware, instead provide “enhanced chroot” to create isolated environment
- Location of abstraction can have impact on performance
- All enable custom software stacks on existing hardware



Type 1 Hypervisor

Type 2 Hypervisor

Containers

E4S: Extreme-scale Scientific Software Stack

- E4S is a community effort to provide open-source software packages for developing, deploying and running scientific applications on HPC platforms.
- E4S has built a comprehensive, coherent, curated software stack based on Spack [<https://spack.io>] that enables application developers to productively develop highly parallel applications that effectively target diverse exascale architectures.
- E4S provides a curated, Spack based software distribution of 120+ HPC (OpenFOAM, Gromacs, Nek5000, LAMMPS), EDA (e.g., Xyce), and AI/ML packages (e.g., TensorFlow, PyTorch, TorchBraid, Scikit-Learn, Pandas, JAX, Horovod, and LBANN).
- With E4S Spack binary build caches, E4S supports both bare-metal and containerized deployment for GPU based platforms.
 - X86_64, ppc64le (IBM Power 10), aarch64 (ARM64) with support for GPUs from NVIDIA, AMD, and Intel
 - HPC and AI/ML packages are optimized for GPUs and CPUs.
- Container images on DockerHub and E4S website of pre-built binaries of ECP ST products.
- Base images and full featured containers (with GPU support) and DOE LLVM containers.
- Commercial support for E4S through ParaTools, Inc. for installation, maintaining an issue tracker, and ECP AD engagement.
- E4S for commercial cloud platforms: Adaptive Computing's ODDC with ParaTools Pro for E4S image with support for AWS.
 - <https://adaptivecomputing.com>
- e4s-chain-spack.sh to chain two Spack instances allows us to install new packages in home directory and use other tools.
- e4s-cl container launch tool allows binary distribution of applications by swapping MPI in the containerized app w/ system MPI.
- e4s-alc is an à la carte tool to customize container images by adding system and Spack packages to an existing image.

E4S Community Policies Version 1

A Commitment to Quality Improvement

- Will serve as membership criteria for E4S
 - Membership is not required for *inclusion* in E4S
 - Also includes forward-looking draft policies
- Purpose: enhance sustainability and interoperability
- Topics cover building, testing, documentation, accessibility, error handling and more
- Multi-year effort led by SDK team
 - Included representation from across ST
 - Multiple rounds of feedback incorporated from ST leadership and membership
- Modeled after xSDK Community Policies
- <https://e4s-project.github.io/policies.html>

P1 Spack-based Build and Installation Each E4S member package supports a scriptable *Spack* build and production-quality installation in a way that is compatible with other E4S member packages in the same environment. When E4S build, test, or installation issues arise, there is an expectation that teams will collaboratively resolve those issues.

P2 Minimal Validation Testing Each E4S member package has at least one test that is executable through the E4S validation test suite (<https://github.com/E4S-Project/testsuite>). This will be a post-installation test that validates the usability of the package. The E4S validation test suite provides basic confidence that a user can compile, install and run every E4S member package. The E4S team can actively participate in the addition of new packages to the suite upon request.

P3 Sustainability All E4S compatibility changes will be sustainable in that the changes go into the regular development and release versions of the package and should not be in a private release/branch that is provided only for E4S releases.

P4 Documentation Each E4S member package should have sufficient documentation to support installation and use.

P5 Product Metadata Each E4S member package team will provide key product information via metadata that is organized in the *E4S DocPortal* format. Depending on the filenames where the metadata is located, this may require *minimal setup*.

P6 Public Repository Each E4S member package will have a public repository, for example at GitHub or Bitbucket, where the development version of the package is available and pull requests can be submitted.

P7 Imported Software If an E4S member package imports software that is externally developed and maintained, then it must allow installing, building, and linking against a functionally equivalent outside copy of that software. Acceptable ways to accomplish this include (1) forsaking the internal copied version and using an externally-provided implementation or (2) changing the file names and namespaces of all global symbols to allow the internal copy and the external copy to coexist in the same downstream libraries and programs. This pertains primarily to third party support libraries and does not apply to key components of the package that may be independent packages but are also integral components to the package itself.

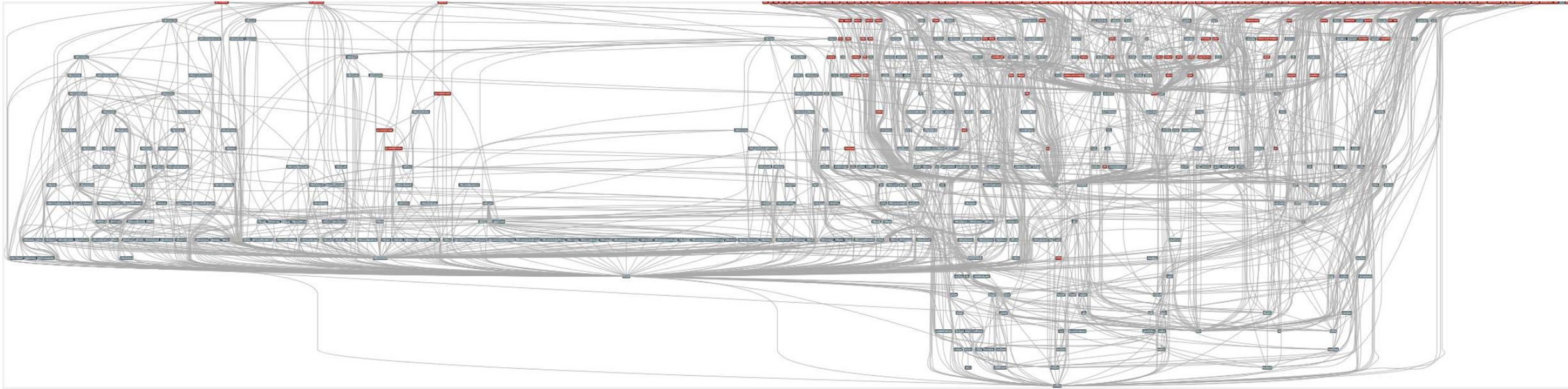
P8 Error Handling Each E4S member package will adopt and document a consistent system for signifying error conditions as appropriate for the language and application. For e.g., returning an error condition or throwing an exception. In the case of a command line tool, it should return a sensible exit status on success/failure, so the package can be safely run from within a script.

P9 Test Suite Each E4S member package will provide a test suite that does not require special system privileges or the purchase of commercial software. This test suite should grow in its comprehensiveness over time. That is, new and modified features should be included in the suite.

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.

Managing large software installations: E4S



- Red boxes are the packages in it (about 100)
- Blue boxes are what *e/se* you need to build it (about 600)
- It's infeasible to build and integrate all of this manually

Spack enables software distribution for HPC

No installation required: clone and go

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

```
$ spack install tau
$ spack install tau@2.33.2
$ spack install tau@2.33.2 %gcc@11.2.0
$ spack install tau@2.33.2 %gcc@11.2.0 +mpi+python+threads
$ spack install tau@2.33.2 %gcc@11.2.0 +mpi ^mvapich2@2.3~wrapperrpath ^ dependency information
```

unconstrained
@ custom version
% custom compiler
+/- build option

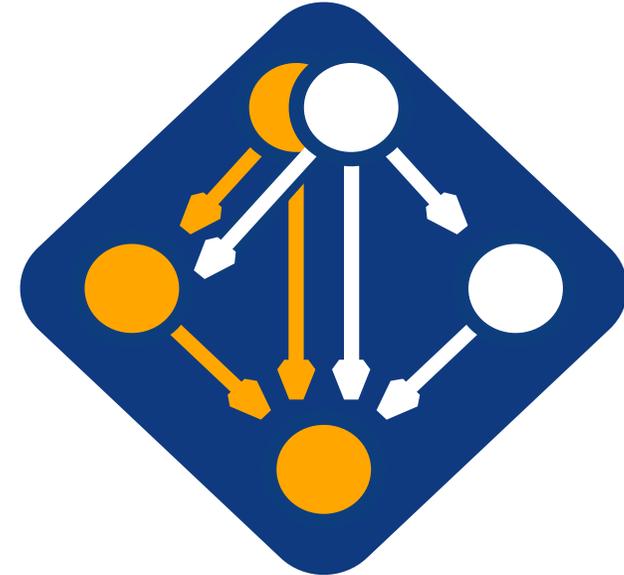
- Each expression is a **spec** for a particular configuration
 - Each clause adds a constraint to the spec
 - Constraints are optional – specify only what you need.
 - Customize install on the command line!
- Spec syntax is recursive
 - Full control over the combinatorial build space



github.com/spack/spack

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](https://twitter.com/spackpm)

Download E4S 24.05 Container for Intel oneAPI: Docker or Singularity

https://e4s-project.github.io/download.html

Note on Container Images

Container images contain binary versions of the Full Release packages listed above. Full-featured GPU-enabled container images are available from Dockerhub:

```
# docker pull ecpe4s/e4s-cuda:24.05
```

```
# docker pull ecpe4s/e4s-rocm:24.05
```

```
# docker pull ecpe4s/e4s-oneapi:24.05
```

E4S Full GPU Images

These images contain a full Spack-based deployment of E4S, including GPU-enabled packages for NVIDIA, AMD, or Intel GPUs.

These images also contain TensorFlow, PyTorch, and TAU.

AMD ROCm (x86_64)	NVIDIA CUDA (X86_64, PPC64LE, AARCH64)	Intel OneAPI (x86_64)
ecpe4s/e4s-rocm:24.05	ecpe4s/e4s-cuda:24.05	ecpe4s/e4s-oneapi:24.05
e4s-rocm90a-x86_64-24.05.sif mirror 1	e4s-cuda80-x86_64-24.05.sif mirror 1	e4s-oneapi-x86_64-24.05.sif mirror 1
e4s-rocm908-x86_64-24.05.sif mirror 1	e4s-cuda90-x86_64-24.05.sif mirror 1	
	e4s-cuda70-ppc64le-24.05.sif mirror 1	
	e4s-cuda75-aarch64-24.05.sif mirror 1	
	e4s-cuda80-aarch64-24.05.sif mirror 1	
	e4s-cuda90-aarch64-24.05.sif mirror 1	

E4S base container images allow users to customize their containers

The screenshot shows a web browser window with the URL <https://e4s-project.github.io/download.html>. The page is titled "GPU Base Images" and contains the following content:

These images come with MPICH, CMake, and the relevant GPU SDK – either AMD ROCm, NVIDIA CUDA Toolkit and NVHPC, or Intel OneAPI.

AMD ROCM (X86_64)

- ecpe4s/e4s-base-rocm:24.05
- e4s-base-rocm-24.05.sif

NVIDIA Multi-Arch (X86_64, PPC64LE, AARCH64)

- ecpe4s/e4s-base-cuda:24.05
- e4s-base-cuda-x86_64-24.05.sif
- e4s-base-cuda-aarch64-24.05.sif
- e4s-base-cuda-ppc64le-24.05.sif

Intel OneAPI (X86_64)

- ecpe4s/e4s-base-oneapi:24.05
- e4s-base-oneapi-24.05.sif

Minimal Spack

This image contains a minimal setup for using Spack 0.22.0 w/ GNU compilers

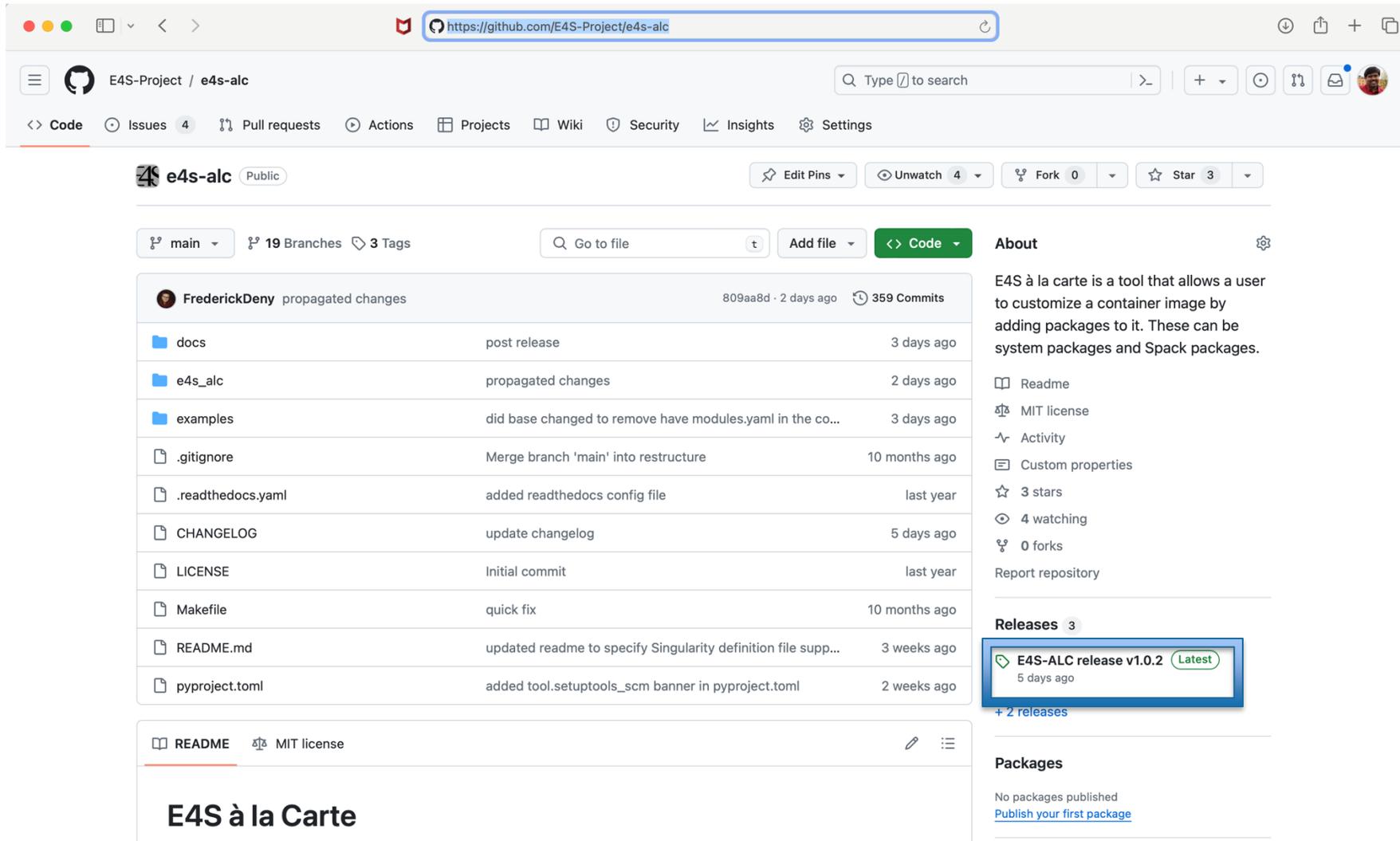
X86_64, PPC64LE, AARCH64

- ecpe4s/ubuntu20.04
- ecpe4s-ubuntu20.04-x86_64-24.02.sif
- ecpe4s-ubuntu20.04-ppc64le-24.02.sif
- ecpe4s-ubuntu20.04-aarch64-24.02.sif

- Intel oneAPI



e4s-alc: a new tool to customize container images. Version 1.0.2



The screenshot displays the GitHub repository page for `E4S-Project/e4s-alc`. The repository is public and has 3 stars and 4 watchers. The main content area shows a list of files and folders, including `docs`, `e4s_alc`, `examples`, `.gitignore`, `.readthedocs.yaml`, `CHANGELOG`, `LICENSE`, `Makefile`, `README.md`, and `pyproject.toml`. The `About` section on the right provides a description of the tool: "E4S à la carte is a tool that allows a user to customize a container image by adding packages to it. These can be system packages and Spack packages." Below the description, there are links for `Readme`, `MIT license`, `Activity`, `Custom properties`, `3 stars`, `4 watching`, and `0 forks`. The `Releases` section shows 3 releases, with the latest release, `E4S-ALC release v1.0.2`, highlighted with a blue box. The `Packages` section indicates that no packages have been published and provides a link to `Publish your first package`.

Add to a base image:

- Spack packages
- OS packages
- Tarballs
- Can create a Dockerfile
- Can create Singularity definition file

E4S 24.05 oneAPI release on Dockerhub

The screenshot shows the Docker Hub interface for the repository `ecpe4s/e4s-oneapi`. The page is viewed on a browser with the URL `https://hub.docker.com/repository/docker/ecpe4s/e4s-oneapi/tags`. The navigation bar includes the Docker Hub logo, 'Explore', 'Repositories', and 'Organizations' tabs, along with a search bar and user profile icon. The breadcrumb trail is `ecpe4s / Repositories / e4s-oneapi / Tags`. Below the breadcrumb, there are tabs for 'General', 'Tags', 'Permissions', 'Webhooks', and 'Settings', with 'Tags' being the active tab. The main content area shows a list of tags. At the top, there is a 'Sort by' dropdown set to 'Newest', a 'Filter Tags' search box, and a 'Delete' button. Two tags are listed:

- latest**: Last pushed 2 days ago by `esw123`. Includes a terminal snippet `docker pull ecpe4s/e4s-oneapi:latest` and a 'Copy' button.
- 24.05**: Last pushed 2 days ago by `esw123`. Includes a terminal snippet `docker pull ecpe4s/e4s-oneapi:24.05` and a 'Copy' button.

Digest	OS/ARCH	Last pull	Compressed Size
ba29a2094e8e	linux/amd64	---	20.53 GB
ba29a2094e8e	linux/amd64	---	20.53 GB



`docker pull ecpe4s/e4s-oneapi`

Using E4S on Cloud Platforms



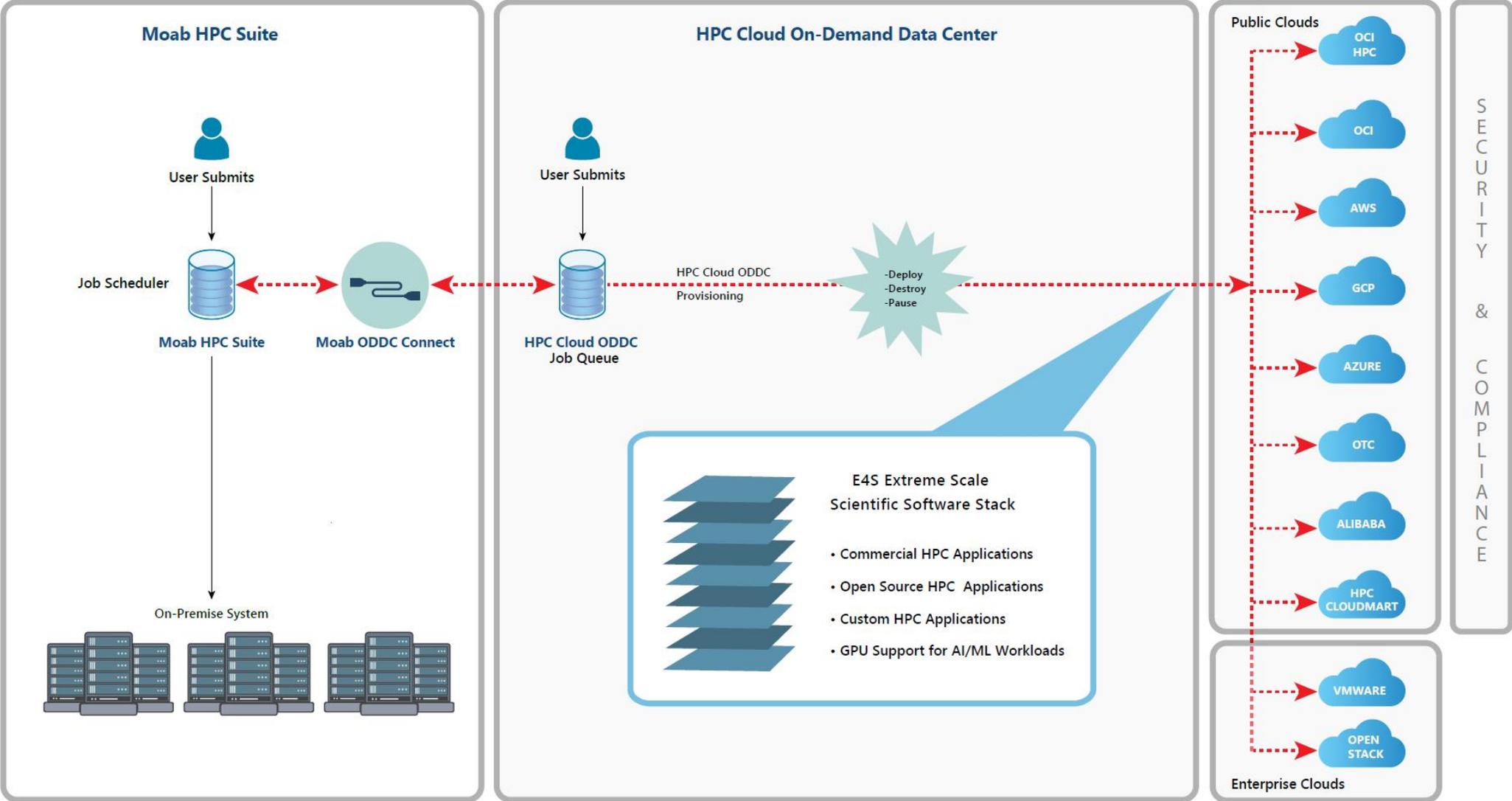
Considerations while deploying HPC/AI workloads to the cloud

- Which cloud provider?
 - AWS, OCI, GCP, Azure, ...
 - Why not all?
- HPC and AI/ML workloads need low latency, high bandwidth
 - Which MPI?
- Which image?
 - Base Ubuntu without HPC tools or libraries? Too steep a learning curve
- Provisioning and building the image on different cloud providers
 - Command line interfaces can be cumbersome to use
- Bursting to the cloud from on-prem clusters using batch submission scripts?

Key considerations for cloud-based deployment for E4S

- MPI - the core inter-node communication library has several implementations
 - Intel MPI, MVAPICH2-X, OpenMPI
 - Interfacing MPI with the job scheduling package (MOAB, Torque, SLURM)
- Cloud providers have different inter-node network adapters:
 - Elastic Fabric Adapter (EFA) on AWS
 - Infiniband on Azure
 - Mellanox Connect-X 5 Ethernet (ROCE) on Oracle Cloud Infrastructure (OCI)
 - IPU on Google Cloud (GCP)
- Intra-node communication with XPMEM (driver and kernel module support is critical)
- GPU Direct Async (GDR) support for communication between GPUs in MVPICH-Plus release
- ParaTools, Inc. building E4S optimized with MVAPICH-Plus for AWS, OCI, GCP, and Azure
- Using Adaptive Computing's ODDC interface to launch E4S jobs on multiple cloud providers!

Adaptive Computing's ODDC with ParaTools Pro for E4S



ParaTools Pro for E4S on AWS Marketplace

The screenshot shows the AWS Marketplace search results for "ParaTools Pro". The search results are sorted by relevance and show 6 results. The first result is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ParallelCluster (arm64)". The second result is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Node (x86)". The third result is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Server (x86)". The fourth result is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ParallelCluster (x86)". The fifth result is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Node (ARM)".

Refine results

- Categories
 - Infrastructure Software (6)
 - Machine Learning (6)
 - Industries (6)
- Delivery methods
 - Amazon Machine Image (6)
 - ParaTools Inc. (6)
- Publisher
 - ParaTools Inc. (6)
- Pricing model
 - Usage Based (6)
- Operating system
 - All Linux/Unix
- Free trial
 - Free Trial (6)
- Contract type
 - Standard Contract (6)
- Architecture
 - 64-bit (x86) (3)
 - 64-bit (Arm) (3)
- Region
 - Africa (Cape Town) (6)
 - Asia Pacific (Hong Kong) (6)
 - Asia Pacific (Tokyo) (6)
 - Asia Pacific (Seoul) (6)
 - Asia Pacific (Osaka) (6)
 - Asia Pacific (Mumbai) (6)
 - Asia Pacific (Hyderabad) (6)
 - Asia Pacific (Singapore) (6)
 - Asia Pacific (Sydney) (6)
 - Asia Pacific (Jakarta) (6)

ParaTools Pro (6 results) showing 1 - 6
Did you mean [paratool pre?](#) Sort By: Relevance

ParaTools Pro for E4S™: AI/ML & HPC Tools on ParallelCluster (arm64)
By [ParaTools Inc.](#) | Ver v2024.05.16.1748-pcluster-3.9.1-e4s-24.05-arm64
Free Trial
Starting from \$0.99 to \$0.99/hr for software + AWS usage fees
E4S Pro[1] - the Extreme-scale Scientific Software Stack E4S[2] hardened for commercial clouds and supported by ParaTools, Inc. provides a platform for developing and deploying HPC and AI/ML applications. It features a performant remote desktop environment (based on DCV) on the login node and...

ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Node (x86)
By [ParaTools Inc.](#) | Ver v17159706-oddc-v1.0-e4s-24.05-node-amd64
Free Trial
Starting from \$0.99 to \$0.99/hr for software + AWS usage fees
E4S Pro[1] - the Extreme-scale Scientific Software Stack E4S[2] hardened for commercial clouds and supported by ParaTools, Inc. provides a platform for developing and deploying HPC and AI/ML applications. It features a performant remote desktop environment (based on VNC) on the login node and...

ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Server (x86)
By [ParaTools Inc.](#) | Ver v17159706-oddc-v1.0-e4s-24.05-server-amd64
Free Trial
Starting from \$0.99 to \$0.99/hr for software + AWS usage fees
E4S Pro[1] - the Extreme-scale Scientific Software Stack E4S[2] hardened for commercial clouds and supported by ParaTools, Inc. provides a platform for developing and deploying HPC and AI/ML applications. It features a performant remote desktop environment (based on VNC) on the login node and...

ParaTools Pro for E4S™: AI/ML & HPC Tools on ParallelCluster (x86)
By [ParaTools Inc.](#) | Ver v2024.05.16.1904-pcluster-3.9.1-e4s-24.05-amd64
Free Trial
Starting from \$0.99 to \$0.99/hr for software + AWS usage fees
E4S Pro[1] - the Extreme-scale Scientific Software Stack E4S[2] hardened for commercial clouds and supported by ParaTools, Inc. provides a platform for developing and deploying HPC and AI/ML applications. It features a performant remote desktop environment (based on DCV) on the login node and...

ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Node (ARM)
By [ParaTools Inc.](#) | Ver v17159835-oddc-v1.0-e4s-24.05-node-arm64
Free Trial
Starting from \$0.99 to \$0.99/hr for software + AWS usage fees

ParaTools



Choosing an instance on AWS to run the image

The screenshot displays the Adaptive Computing Cluster Manager interface. A modal window is open for configuring an Amazon Web Services instance. The modal title is "e4s-22.11-mvapich2-xyce-aws" with a price of "\$0.28 per Hour". The configuration fields include:

- Name: e4s-22.11-mvapich2-xyce-aws
- OS Type: centos-7
- Prefix: e4s-xyce-aws
- Credential: (empty)
- Head Node Size: t2.xlarge - vCPU: 4, Mem (GB): 16
- Manager: Torque
- Region: US West 1
- Availability Zone: (empty)

Bursting Configuration is set to "Off". Under "Compute Nodes", the size is "t2.xlarge - vCPU: 4, Mem (GB): 16" and the count is "2".

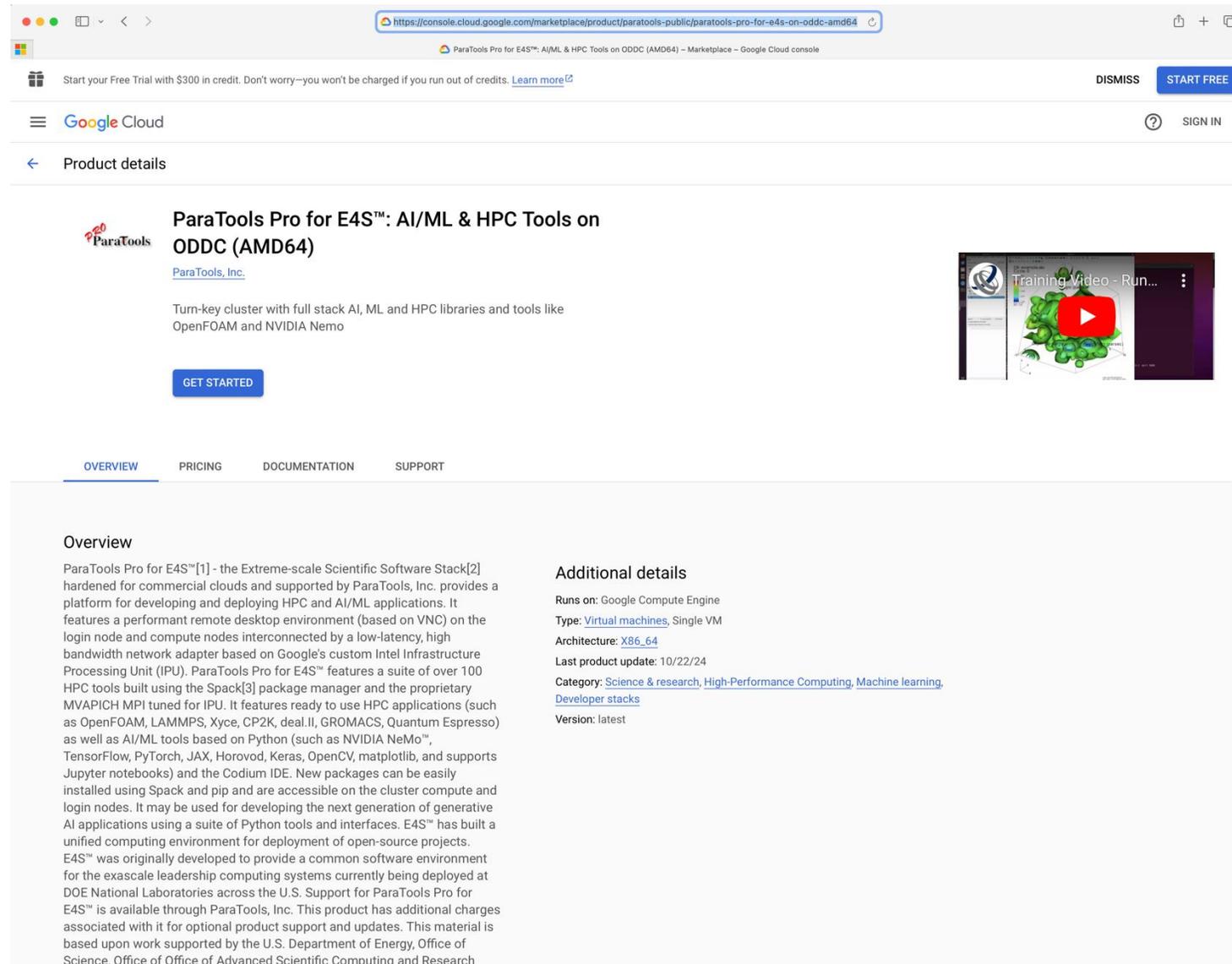
The background interface shows a sidebar with "APPLICATIONS" including Cluster Manager, Stack Manager, Credentials Manager, Job Manager, File Manager, Accounting, and Instance Prices. The "Cloud Providers" section lists All Cloud Providers, Alibaba Cloud, Oracle Cloud HPC, Oracle Cloud, Amazon Web Services, Google Cloud, Microsoft Azure, and Open Telekom Cloud.

Credential	Uptime	Actions
Not Set	N/A	ⓘ ...

Rows per page: 10 | 1-1 of 1

Copyright © 2022. Adaptive Computing Enterprises, Inc., All rights reserved.

ParaTools Pro for E4S™ on Google Cloud Marketplace



Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#) **DISMISS** **START FREE**

Google Cloud **?** SIGN IN

Product details

 **ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC (AMD64)**
[ParaTools, Inc.](#)

Turn-key cluster with full stack AI, ML and HPC libraries and tools like OpenFOAM and NVIDIA Nemo

GET STARTED



OVERVIEW PRICING DOCUMENTATION SUPPORT

Overview

ParaTools Pro for E4S™[1] - the Extreme-scale Scientific Software Stack[2] hardened for commercial clouds and supported by ParaTools, Inc. provides a platform for developing and deploying HPC and AI/ML applications. It features a performant remote desktop environment (based on VNC) on the login node and compute nodes interconnected by a low-latency, high bandwidth network adapter based on Google's custom Intel Infrastructure Processing Unit (IPU). ParaTools Pro for E4S™ features a suite of over 100 HPC tools built using the Spack[3] package manager and the proprietary MVAPICH MPI tuned for IPU. It features ready to use HPC applications (such as OpenFOAM, LAMMPS, Xyce, CP2K, deal.II, GROMACS, Quantum Espresso) as well as AI/ML tools based on Python (such as NVIDIA NeMo™, TensorFlow, PyTorch, JAX, Horovod, Keras, OpenCV, matplotlib, and supports Jupyter notebooks) and the Codium IDE. New packages can be easily installed using Spack and pip and are accessible on the cluster compute and login nodes. It may be used for developing the next generation of generative AI applications using a suite of Python tools and interfaces. E4S™ has built a unified computing environment for deployment of open-source projects. E4S™ was originally developed to provide a common software environment for the exascale leadership computing systems currently being deployed at DOE National Laboratories across the U.S. Support for ParaTools Pro for E4S™ is available through ParaTools, Inc. This product has additional charges associated with it for optional product support and updates. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Office of Advanced Scientific Computing and Research

Additional details

Runs on: Google Compute Engine
Type: [Virtual machines](#), Single VM
Architecture: [X86_64](#)
Last product update: 10/22/24
Category: [Science & research](#), [High-Performance Computing](#), [Machine learning](#), [Developer stacks](#)
Version: latest

ParaTools



<https://console.cloud.google.com/marketplace/product/paratools-public/paratools-pro-for-e4s-on-oddc-amd64>

E4S Exercise: TAU

Using ParaTools Pro for E4S image on AWS with Adaptive Computing's On-Demand Data Center (ODDC)

STEP 1: Click on Training tab at:
<https://paratools.adaptivecomputing.com>
Firefox recommended.

Adaptive Computing's ODDC: Go to Training session tab and enter email and 11090 as session id

The screenshot shows a web browser window at paratools.adaptivecomputing.com. The main content area features the Adaptive Computing logo and the title "HPC Cloud On-Demand Data Center". Below the title, there is a paragraph describing the platform's capabilities: "Adaptive Computing's On-Demand Data Center platform gives companies the ability to spin up temporary or persistent data center infrastructure resources quickly, inexpensively, and on-demand. This intelligent cloud management platform gives immediate access to all computational resources, whether on premise or in the cloud on any leading cloud provider." Another paragraph states: "Teams can automatically deploy and build clusters in the cloud, automatically run applications on those clusters, and then terminate the cloud resources on a daily, weekly, or even hourly basis." A third paragraph mentions: "The HPC Cloud On-Demand Data Center includes all of the necessary tools to provision compute power and run workloads in the cloud or on-premise. Access to all major cloud providers is pre-configured and built into the interface (CLI or GUI)." At the bottom left of the main content, it says "On-Demand Data Center 8.2.1".

Overlaid on the right side of the browser window is a white registration form titled "TRAINING SESSION". At the top of the form, there are two tabs: "LOGIN" and "TRAINING SESSION". The "TRAINING SESSION" tab is selected and highlighted with a blue underline. An arrow points from the text "Click here" to this tab. Below the tabs, the form has the title "TRAINING SESSION" and two input fields: "Email*" with the value "sameer@paratools.com" and "Session Code*" with the value "11090". Below these fields is a dark blue button labeled "REGISTER".

Adaptive Computing's ODDC: Login with session code

The screenshot shows a web browser at `paratools.adaptivecomputing.com`. The main page features the Adaptive Computing logo and the title "HPC Cloud On-Demand Data Center". Below the title, there are three paragraphs of text describing the platform's capabilities. A modal window is overlaid on the right side of the page, titled "TRAINING SESSION". The modal has two tabs: "LOGIN" and "TRAINING SESSION", with the latter being selected. The "TRAINING SESSION" tab contains two input fields: "Email *" with the value "sameer@paratools.com" and "Session Code *" with the value "11090". A "REGISTER" button is located below the input fields. Blue arrows point from external text labels to the "TRAINING SESSION" tab and the email input field.

Adaptive
COMPUTING
HPC Cloud On-Demand Data Center

Adaptive Computing's On-Demand Data Center platform gives companies the ability to spin up temporary or persistent data center infrastructure resources quickly, inexpensively, and on-demand. This intelligent cloud management platform gives immediate access to all computational resources, whether on premise or in the cloud on any leading cloud provider.

Teams can automatically deploy and build clusters in the cloud, automatically run applications on those clusters, and then terminate the cloud resources on a daily, weekly, or even hourly basis.

The HPC Cloud On-Demand Data Center includes all of the necessary tools to provision compute power and run workloads in the cloud or on-premise. Access to all major cloud providers is pre-configured and built into the interface (CLI or GUI).

On-Demand Data Center 8.2.1

LOGIN TRAINING SESSION

TRAINING SESSION

Email *
sameer@paratools.com

Session Code *
11090

REGISTER

Click on
Training Session

Enter email
and session
Code 11090

Launch VNC Viewer from ODDC's Configuration Tab

The screenshot shows a web browser window at `paratools.adaptivecomputing.com`. The page title is "Cluster Manager". In the top right corner, it displays "Session 11090" and "Training1 - User". Below the header is a search bar and a refresh button. The main content is a table with the following data:

State	Name	Provider	Nodes
 Available 18:43 10:22	e4s-24-05-aws		2

An arrow points from the text "Click here" to the cluster name "e4s-24-05-aws". At the bottom right of the table, there is pagination information: "Rows per page: 10", "1-1 of 1", and navigation arrows.



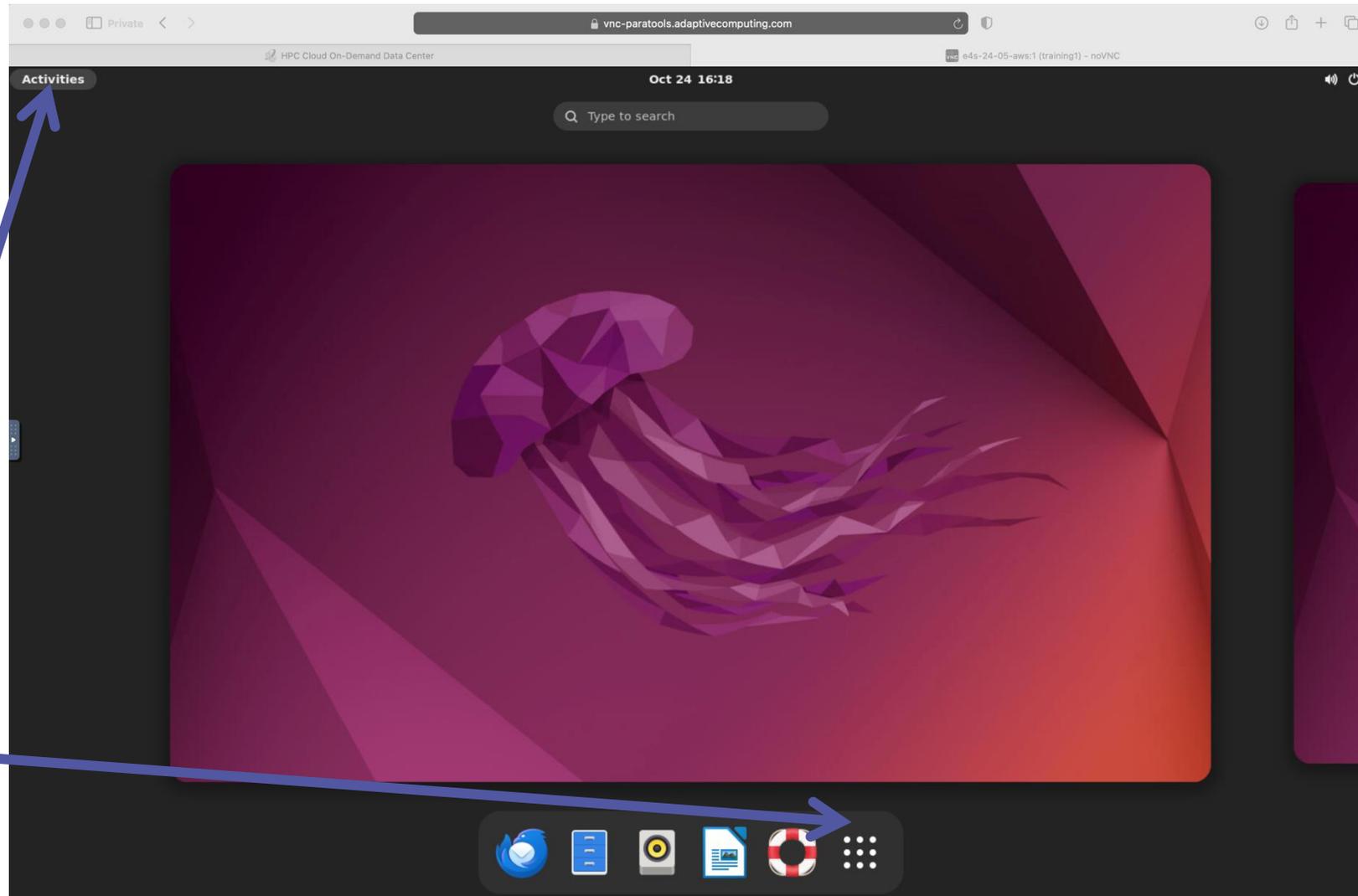
Launch VNC Viewer from ODDC and allow popups

The screenshot shows a Firefox browser window in private browsing mode. The address bar displays the URL: `https://paratools.adaptivecomputing.com/clusters/6626dd00a6069602bd755901`. A notification bar at the top indicates that Firefox prevented a pop-up window from opening. A 'Preferences' popup is open, showing options to allow pop-ups for the current site, manage settings, and show the URL of the blocked pop-up. The main content area shows the 'Configuration' tab for a cluster named 'e4s-24-05-aws'. The 'Cluster Information' section includes details such as Cluster ID, Head Node, Size, Image Name, Cluster IP, SSH Username, Created time, and Availability Zone. A blue 'OPEN VNC VIEWER' button is visible, with a checkbox for 'Use popup' selected. The 'Participant Credentials' section shows the Username, Password, and IP Address, along with a 'DOWNLOAD SSH KEY' button. The footer contains the copyright notice: 'Copyright © 2024. Adaptive Computing Enterprises, Inc., All rights reserved.'

Click here

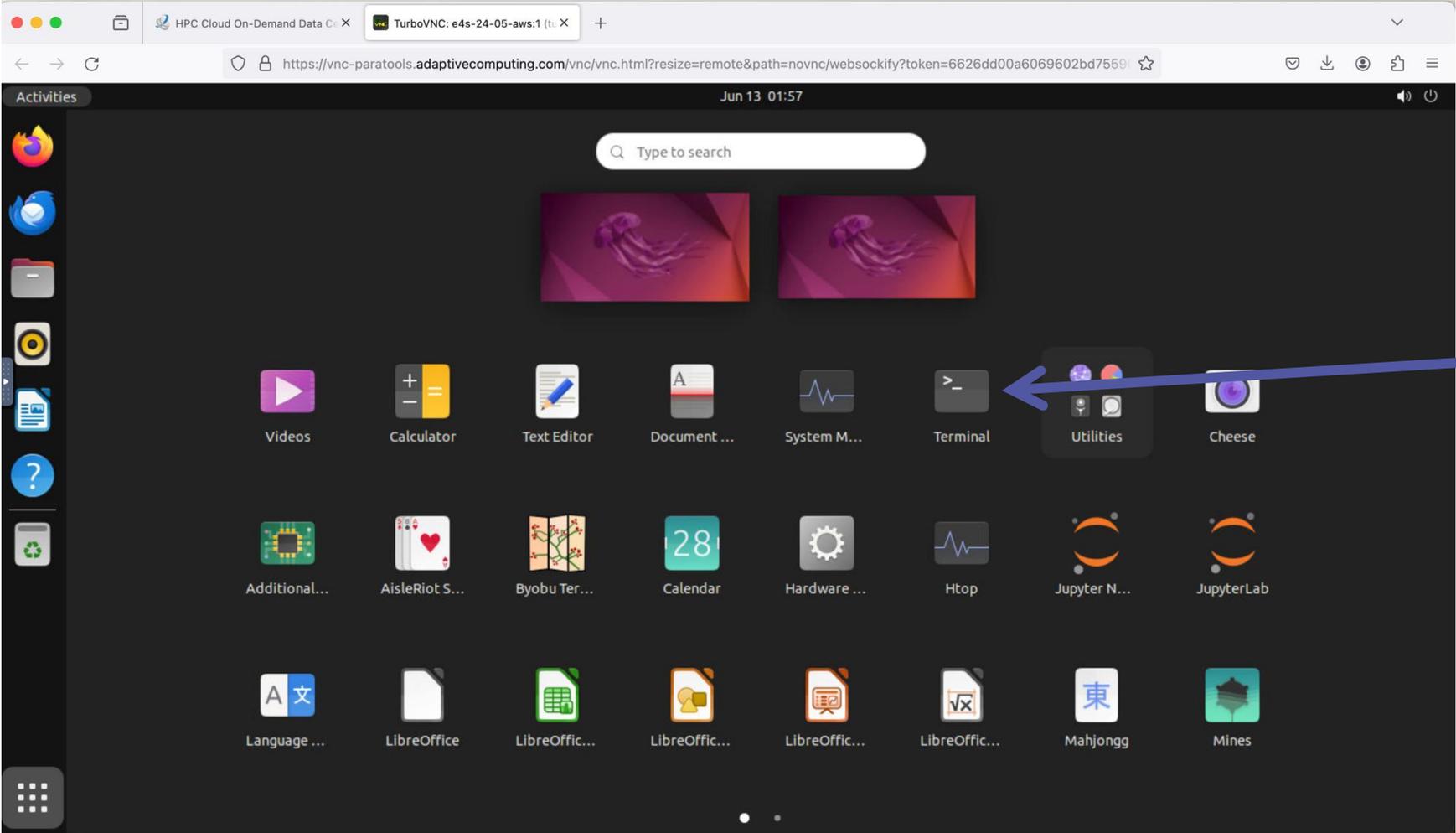


Remote Desktop from the ParaTools Pro for E4S™ image on AWS and GCP



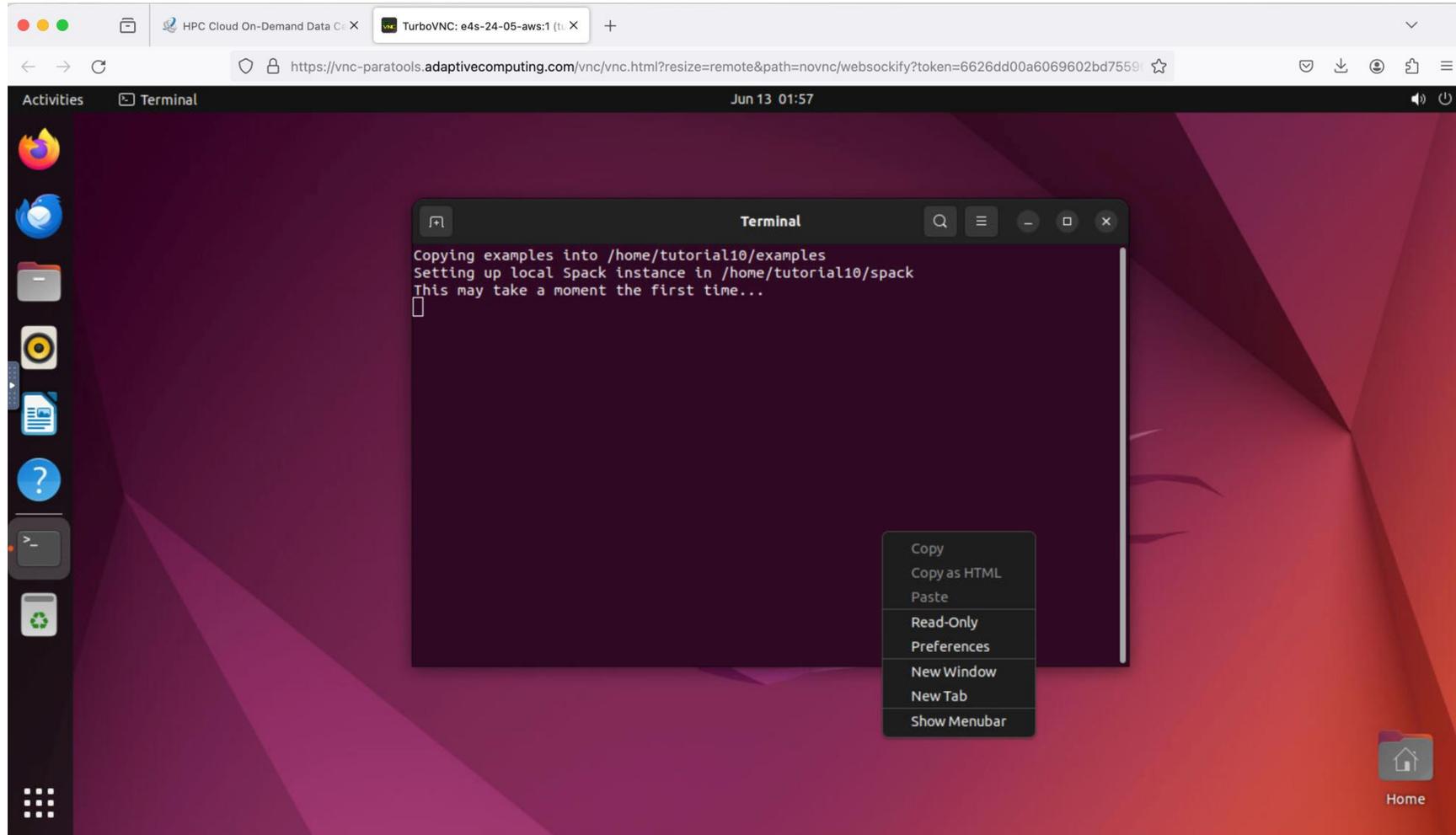
Click here

Launch Terminal in Remote Desktop

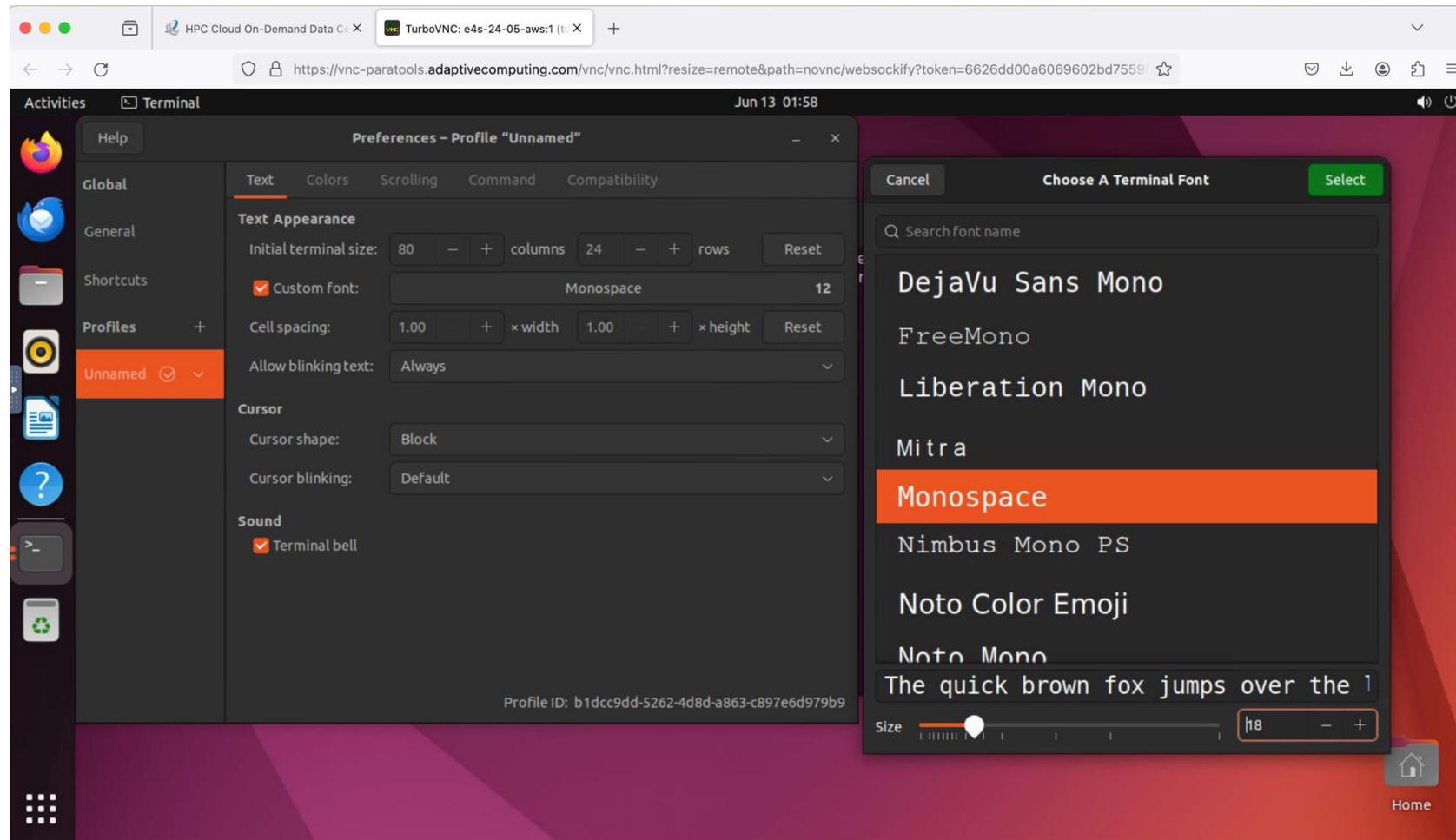


Click here

To increase font size right click and choose preferences



Choose font size after clicking Custom Font for Terminal

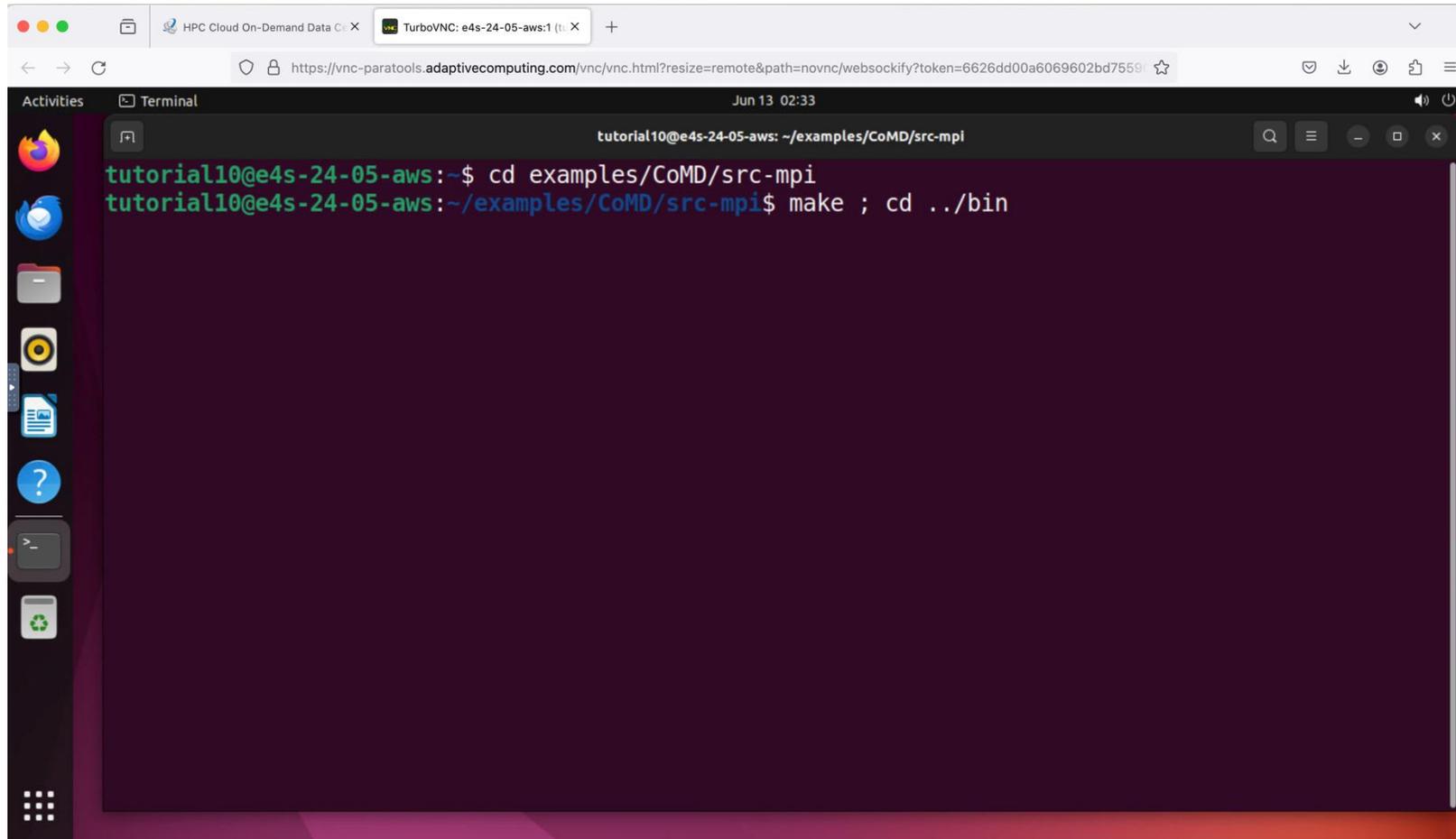


Spack Package Manager

```
TurboVNC: e4s-24-02-aws:1 (paratoolsadmin) - noVNC
https://vnc-paratools.adaptivecomputing.ai/vnc/vnc.html?resize=remote&path=novnc/websockify?token=65e3202d2aa75102b14de767-paratoolsadmin
Activities Terminal Apr 17 14:54
paratoolsadmin@e4s-24-02-aws: ~/examples
paratoolsadmin@e4s-24-02-aws:~/examples$ spack find -x
-- linux-ubuntu22.04-x86_64 / gcc@11.4.0 -----
adios@1.13.1      dealii@9.5.1      heffte@2.4.0      mercury@2.3.1    phist@1.12.0     superlu@5.3.0
adios2@2.9.2     dealii@9.5.1      heffte@2.4.0      metall@0.25      plasma@23.8.2    superlu-dist@8.2.1
alquimia@1.1.0  dyninst@12.3.0    hive@4.0.0-beta-1  mfem@4.6.0       plumed@2.9.0     superlu-dist@8.2.1
aml@0.2.1        e4s-cl@1.0.1      hpctoolkit@2023.08.1  mfem@4.6.0       precice@2.5.0    swig@4.0.2-fortran
amrex@24.01      ecp-data-vis-sdk@1.0  hpctoolkit@2023.08.1  mgard@2023-03-31  pruners-ninja@1.0.1  sz@2.1.12.5
amrex@24.01      exago@1.6.0        hpctoolkit@2023.08.1  mgard@2023-03-31  pumi@2.2.8        sz3@3.1.7
arborx@1.5       flecsi@2.2.1      hpx@1.9.1          mpark-variant@1.4.0  py-cinemasci@1.3    tasmanian@8.0
arborx@1.5       flecsi@2.2.1      hpx@1.9.1          mpiutils@0.11.1    py-deephyper@0.4.2  tasmanian@8.0
argobots@1.1     flit@2.1.0         hypre@2.30.0       nccmp@1.9.1.0     py-h5py@3.8.0      tau@2.33.1
ascent@0.9.2     flux-core@0.58.0   kokkos@4.2.00      nco@5.1.6         py-jupyterhub@1.4.1  tau@2.33.1
axom@0.8.1       flux-core@0.58.0   kokkos@4.2.00      netcdf-fortran@4.6.1  py-libensemble@1.1.0  trilinos@13.0.1
boost@1.79.0     fortrilinos@2.3.0  kokkos-kernels@4.2.00  netlib-scalapack@2.2.0  py-petsc4py@3.20.2  trilinos@14.4.0
bricks@2023.08.25  fpm@0.10.0        kokkos-kernels@4.2.00  nrm@0.1.0         py-warpx@23.08      trilinos@15.0.0
bricks@2023.08.25  gasnet@2023.9.0   laghos@3.1          nvhpc@23.11       qthreads@1.18      umap@2.1.0
butterflypack@2.4.0  ginkgo@1.7.0      lammps@20230802.2    omega-h@9.34.13    quantum-espresso@7.3  umpire@2022.10.0
cabana@0.6.0      ginkgo@1.7.0      lammps@20230802.2    openfoam@2312     raja@2022.10.4      umpire@2022.10.0
cabana@0.6.0      globalarrays@5.8.2  lbann@0.104         openmpi@5.0.1     raja@2022.10.4      unifyfs@2.0
caliper@2.10.0    gmp@6.2.1         legion@23.06.0      openpmd-api@0.15.2  rempi@1.1.0         upcxx@2023.9.0
caliper@2.10.0    gotcha@1.0.5      libcatalyst@2.0.0-rc4  papi@7.1.0        scr@3.0.1           upcxx@2023.9.0
chai@2022.03.0    gptune@4.0.0      libnm@0.1.0         papyrus@1.0.2     slate@2023.08.25    variorum@0.7.0
chai@2022.03.0    gromacs@2023.3    libpressio@0.95.1   parallel-netcdf@1.12.3  slate@2023.08.25    veloc@1.7
charliecloud@0.35  h5bench@1.4       libpressio@0.95.1   paraview@5.11.2   slepc@3.20.1       vtk-m@2.0.0
conduit@0.8.8     hdf5@1.12.2       libquo@1.3.1        parsec@3.0.2209   slepc@3.20.1       wannier90@3.1.0
cp2k@2024.1       hdf5@1.14.3       libunwind@1.6.2     pdt@3.25.2        strumpack@7.2.0     xyce@7.8.0
cusz@0.3.1        hdf5-vol-async@1.7  loki@0.1.7          petsc@3.20.3      strumpack@7.2.0    zfp@0.5.5
darshan-runtime@3.4.4  hdf5-vol-cache@v1.1  magma@2.7.2         petsc@3.20.3      sundials@6.7.0
darshan-util@3.4.4  hdf5-vol-log@1.4.0
==> 160 installed packages
paratoolsadmin@e4s-24-02-aws:~/examples$
```



CoMD: TAU with event-based sampling (EBS)

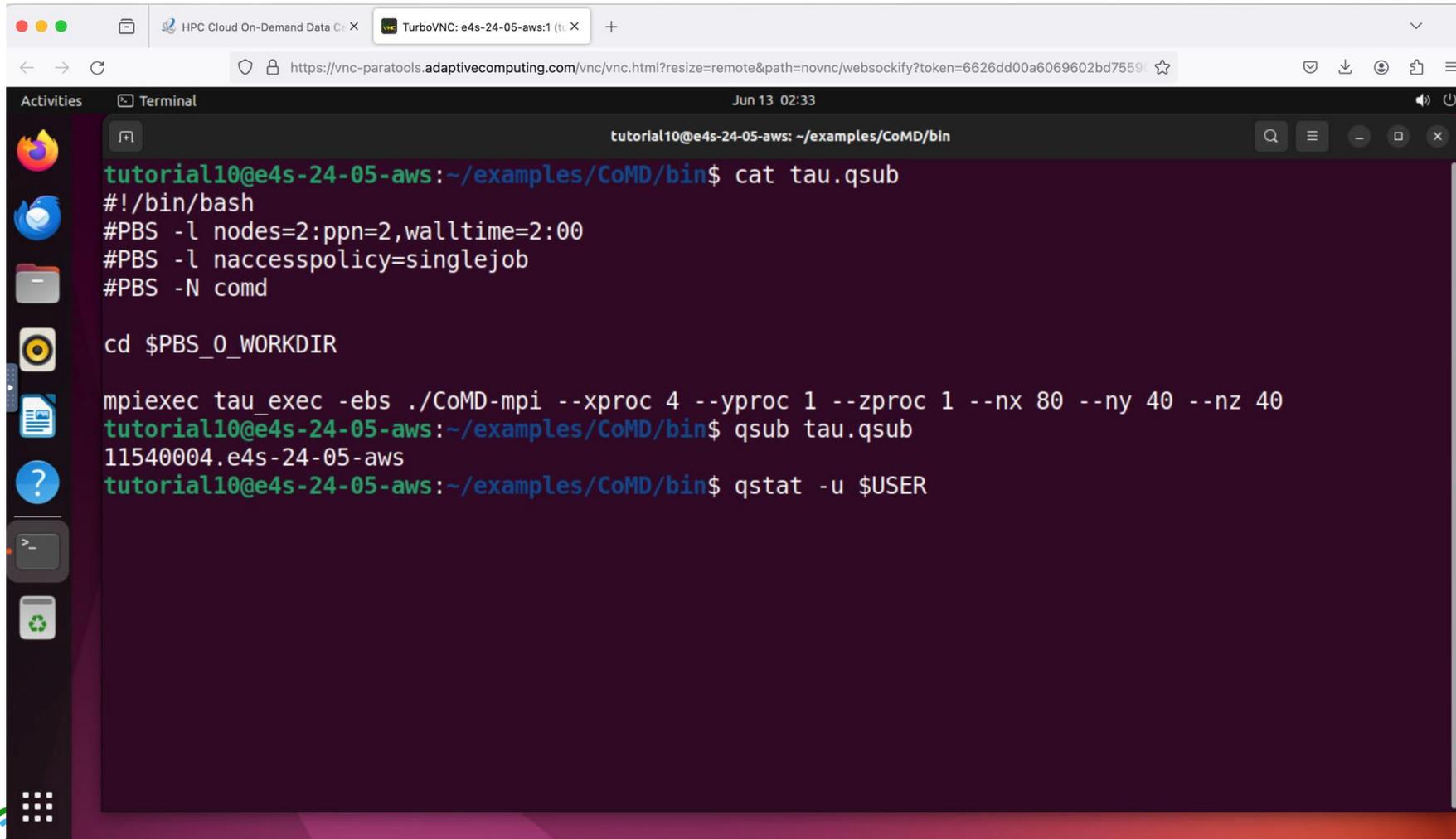


The screenshot shows a TurboVNC terminal window with the following content:

```
tutorial10@e4s-24-05-aws: ~$ cd examples/CoMD/src-mpi
tutorial10@e4s-24-05-aws: ~/examples/CoMD/src-mpi$ make ; cd ../bin
```

```
% cd examples/CoMD/src-mpi
% make; cd ../bin
```

CoMD: TAU with event-based sampling (EBS)



```
tutorial10@e4s-24-05-aws: ~/examples/CoMD/bin
tutorial10@e4s-24-05-aws:~/examples/CoMD/bin$ cat tau.qsub
#!/bin/bash
#PBS -l nodes=2:ppn=2,walltime=2:00
#PBS -l naccesspolicy=singlejob
#PBS -N comd

cd $PBS_0_WORKDIR

mpiexec tau_exec -ebs ./CoMD-mpi --xproc 4 --yproc 1 --zproc 1 --nx 80 --ny 40 --nz 40
tutorial10@e4s-24-05-aws:~/examples/CoMD/bin$ qsub tau.qsub
11540004.e4s-24-05-aws
tutorial10@e4s-24-05-aws:~/examples/CoMD/bin$ qstat -u $USER
```

```
% qsub tau.qsub
% qstat -u $USER
```

CoMD: TAU's paraprof visualizer

```
tutorial10@e4s-24-05-aws: ~/examples/CoMD/bin$ ls
CoMD-mpi          comd.qsub        profile.1.0.1    r.sh            tau.sbatch
CoMD-mpi.2024:06:13-02:33:42.yaml  comd.sbatch     profile.2.0.0    romp.sh
clean.sh         profile.0.0.0   profile.2.0.1    rt.sh
comd.e11540004  profile.0.0.1  profile.3.0.0    select.tau
comd.o11540004  profile.1.0.0  profile.3.0.1    tau.qsub
tutorial10@e4s-24-05-aws: ~/examples/CoMD/bin$ paraprof &
```

% paraprof &

CoMD: TAU's paraprof visualizer

The screenshot shows the TAU ParaProf Manager interface. On the left, a tree view shows the trial field structure. The main window displays a table of trial fields and a detailed view of thread statistics for Node 0, Thread 0.

TrialField	
Name	bin/CoM
Application ID	0
Experiment ID	0
Trial ID	0
CPU Cores	16
CPU MHz	3100.22
CPU Type	Intel(R)
CPU Vendor	Genuine
CPU Allowed	000000
CPU Allowed List	0-1
CWD	/home/t
Cache Size	36608 K
Command Line	./CoMD-
Ending Timestamp	171824
Executable	/home/t
File Type Index	1
File Type Name	TAU pro
Hostname	ac-5901
Local Time	2024-06
MPI Processor Name	ac-5901
Memories Allowed	000000
Memories Allowed List	0
Memory Size	130390
Node Name	ac-5901
OS Machine	x86_64
OS Name	Linux
OS Release	5.19.0-1029-aws
OS Version	#30~22.04.1-Ubuntu SMP Thu Jul ...
Starting Timestamp	1718246022396106

TAU: ParaProf: /home/tutorial10/examples/CoMD/bin	
Metric:	TIME
Value:	Exclusive
Std. Dev.	[Bar Chart]
Mean	[Bar Chart]
Max	[Bar Chart]
Min	[Bar Chart]
node 0, thr	[Bar Chart]
node 0, thr	[Bar Chart]
node 1, thr	[Bar Chart]
node 1, thr	[Bar Chart]
node 2, thr	[Bar Chart]
node 2, thr	[Bar Chart]
node 3, thr	[Bar Chart]
node 3, thr	[Bar Chart]

Right click on Node 0, Thread 0 and choose Show Thread Statistics Table (third option)

TAU's ParaProf Profile Browser: Thread Statistics Table

TAU: ParaProf: Statistics for: node 0, thread 0 - /home/tutorial10/examples/CoMD/bin

Name	Exclusive TIME	Inclusive TIME	Calls	Child Calls
.TAU application	0	26.582	1	1
taupreload_main	26.174	26.582	1	1,255
[CONTEXT] taupreload_main	0	25.65	855	0
[SUMMARY] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c}]	24.27	24.27	809	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {198}]	4.8	4.8	160	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {209}]	3.78	3.78	126	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {199}]	3.66	3.66	122	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {189}]	2.76	2.76	92	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {202}]	1.98	1.98	66	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {193}]	1.56	1.56	52	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {208}]	1.47	1.47	49	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {207}]	1.38	1.38	46	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {224}]	0.66	0.66	22	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {206}]	0.63	0.63	21	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {223}]	0.39	0.39	13	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {185}]	0.27	0.27	9	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {210}]	0.24	0.24	8	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {220}]	0.18	0.18	6	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {214}]	0.15	0.15	5	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {181}]	0.12	0.12	4	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {175}]	0.09	0.09	3	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {159}]	0.06	0.06	2	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {187}]	0.06	0.06	2	0
[SAMPLE] jfForce [{/home/tutorial10/examples/CoMD/src-mpi/jfForce.c} {156}]	0.03	0.03	1	0
[SUMMARY] getBoxFromCoord [{/home/tutorial10/examples/CoMD/src-mpi/linkCells.c}]	0.36	0.36	12	0
[SAMPLE] UNRESOLVED /usr/lib/x86_64-linux-gnu/libc.so.6	0.21	0.21	7	0
[SUMMARY] sortAtomsInCell [{/home/tutorial10/examples/CoMD/src-mpi/haloExchange}]	0.21	0.21	7	0
[SUMMARY] advancePosition [{/home/tutorial10/examples/CoMD/src-mpi/advance}]	0.12	0.12	4	0

Click on columns to sort (e.g., Inclusive)

Expand nodes and right click on a sample and

Select "Show Source Code"

TAU's ParaProf Profile Browser: Source Code Browser

The screenshot displays the TAU ParaProf Source Browser interface. On the left, a source code editor shows the file `/home/tutorial10/examples/CoMD/src-mpi/ljForce.c`. The code is a C function with several nested loops. Line 198 is highlighted in blue, showing the calculation of a distance vector `dr` and its squared magnitude `r2`.

On the right, a performance table is visible, showing the time spent at each line of code. The table has four columns: Exclusive TIME, Inclusive TIME, Calls, and Child Calls. The row corresponding to line 198 is highlighted in yellow, indicating it is the current selection.

Line	Exclusive TIME	Inclusive TIME	Calls	Child Calls
166	0	26.582	1	1
167	26.174	26.582	1	1,255
168	0	25.65	855	0
169	24.27	24.27	809	0
170	4.8	4.8	160	0
171	3.78	3.78	126	0
172	3.66	3.66	122	0
173	2.76	2.76	92	0
174	1.98	1.98	66	0
175	1.56	1.56	52	0
176	1.47	1.47	49	0
177	1.38	1.38	46	0
178	0.66	0.66	22	0
179	0.63	0.63	21	0
180	0.39	0.39	13	0
181	0.27	0.27	9	0
182	0.24	0.24	8	0
183	0.18	0.18	6	0
184	0.15	0.15	5	0
185	0.12	0.12	4	0
186	0.09	0.09	3	0
187	0.06	0.06	2	0
188	0.06	0.06	2	0
189	0.03	0.03	1	0
190	0.36	0.36	12	0
191	0.21	0.21	7	0
192	0.21	0.21	7	0
193	0.12	0.12	4	0

The application spent 4.8 seconds at line 198 in `ljForce.c` in MPI rank 0. TAU collected 160 samples at this line of code.

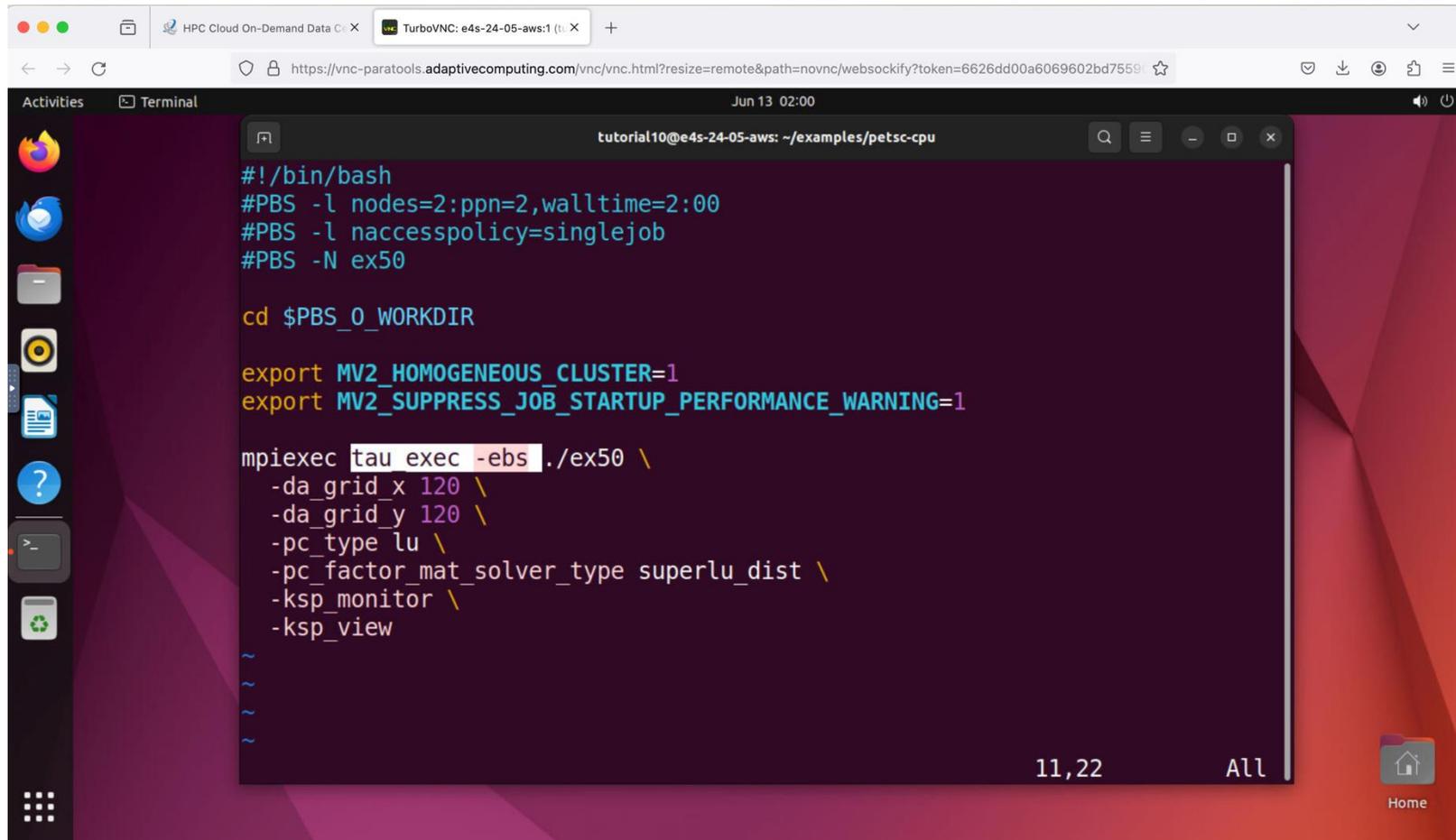
It is within five levels of for loops!

There was no change to source code, build system, or the application binary!

TAU Exercise #2:

**Instrumenting PETSc application using TAU's
Perfstubs interface**

Launching the binary using tau_exec -ebs



The screenshot shows a terminal window with the following content:

```
#!/bin/bash
#PBS -l nodes=2:ppn=2,walltime=2:00
#PBS -l naccesspolicy=singlejob
#PBS -N ex50

cd $PBS_0_WORKDIR

export MV2_HOMOGENEOUS_CLUSTER=1
export MV2_SUPPRESS_JOB_STARTUP_PERFORMANCE_WARNING=1

mpiexec tau_exec -ebs ./ex50 \
  -da_grid_x 120 \
  -da_grid_y 120 \
  -pc_type lu \
  -pc_factor_mat_solver_type superlu_dist \
  -ksp_monitor \
  -ksp_view
```

cd ~/examples/petsc-cpu
vi ex50.qsub

Add tau_exec -ebs
before ./ex50 in the launch
command. Save the file.

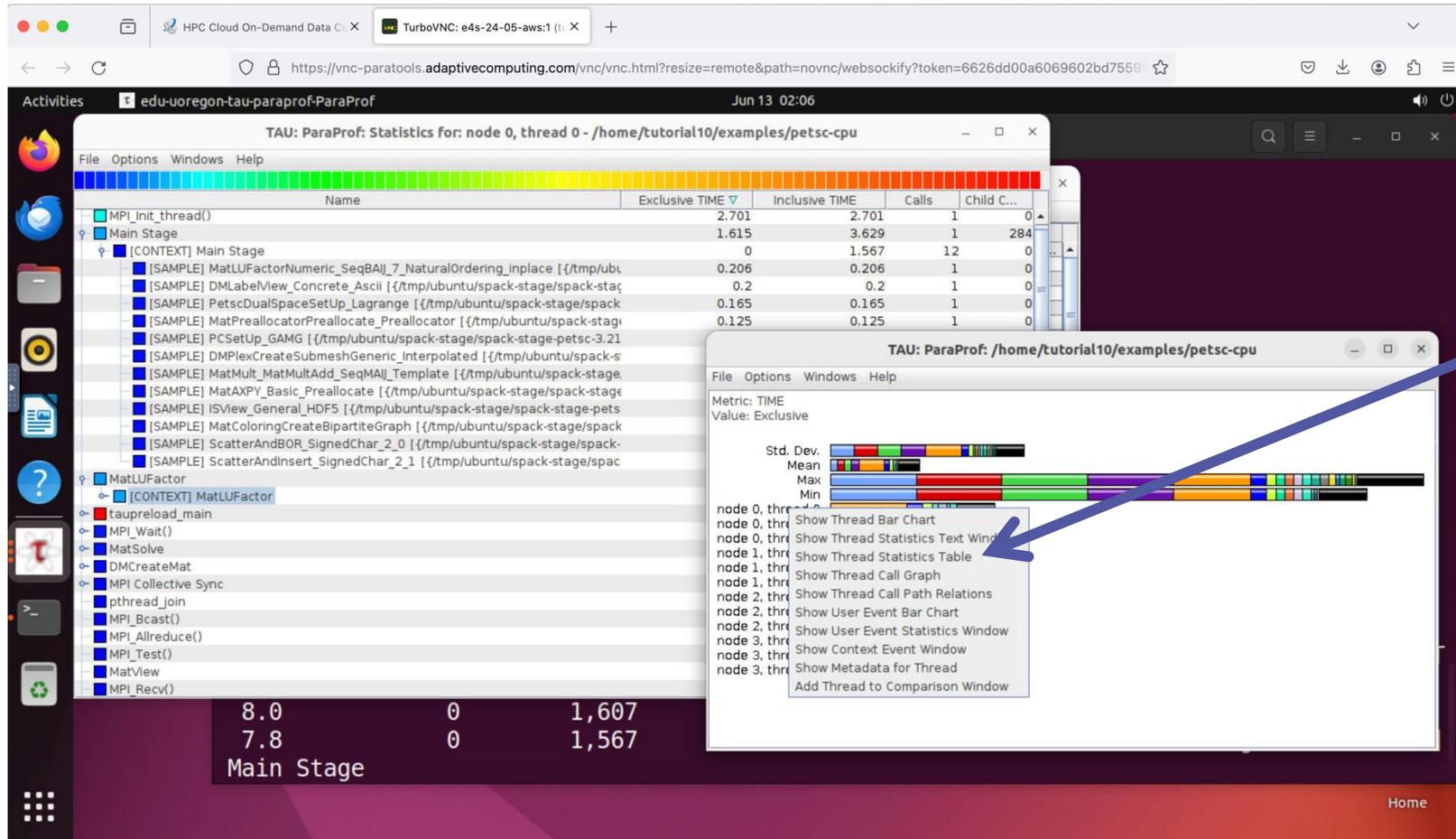
TAU's ParaProf Profile Browser: Source Code Browser

```
tutorial10@e4s-24-05-aws: ~/examples/petsc-cpu
-----
11540000.e4s-24-05-aws tutorial10 e4s-24-0 ex50          3795    2    4    --
   00:02:00 R   00:00:26
tutorial10@e4s-24-05-aws:~/examples/petsc-cpu$ qstat -u $USER
e4s-24-05-aws:
      Req'd      Elap
Job ID  Time      S  Time  Username  Queue  Jobname      SessID  NDS  TSK  Req'd
-----
11540000.e4s-24-05-aws tutorial10 e4s-24-0 ex50          3795    2    4    --
   00:02:00 C   --
tutorial10@e4s-24-05-aws:~/examples/petsc-cpu$ ls
clean.sh      ex50.o11540000  profile.0.0.1  profile.2.0.0  profile.3.0.2
compile.sh    ex50.qsub       profile.0.0.2  profile.2.0.1  run-single-node.sh
ex50          ex50.sbatch     profile.1.0.0  profile.2.0.2  run-tau-oddc.sh
ex50.c        makefile        profile.1.0.1  profile.3.0.0
ex50.e11540000 profile.0.0.0   profile.1.0.2  profile.3.0.1
tutorial10@e4s-24-05-aws:~/examples/petsc-cpu$ paraprof &
```

qsub ex50.qsub
qstat -u \$USER

After it completes
ls
paraprof &

TAU's paraprof browser with PETSc performance profile



paraprof

Choose
Show thread statistics table by
right clicking on
node 0, thread 0.

Using pprof: TAU's text based profile browser

```
tutorial10@e4s-24-05-aws: ~/examples/petsc-cpu
tutorial10@e4s-24-05-aws:~/examples/petsc-cpu$ paraprof &
[1] 152885
tutorial10@e4s-24-05-aws:~/examples/petsc-cpu$ pprof -a | more
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
-----
%Time   Exclusive   Inclusive   #Call   #Subrs   Inclusive Name
      msec    total msec
-----
100.0    0.266      20,080     1       1       20080268 .TAU application
100.0      597       20,080     1       159      20080002 taupreload_main
65.2    13,080     13,095     1       2       13095901 MPI_Finalize()
18.1     1,615     3,629     1       284      3629490 Main Stage
13.4     2,700     2,700     1       0       2700776 MPI_Init_thread()
8.7       1         1,756     1       27       1756523 PCSetUp
8.7     1,512     1,754     1      10258     1754558 MatLUFactor
8.0       0         1,607     44       0         36527 MatLUFactor => [CONTEXT
] MatLUFactor
8.0       0         1,607     44       0         36527 [CONTEXT] MatLUFactor
7.8       0         1,567     12       0        130617 Main Stage => [CONTEXT]
Main Stage
```

pprof -a | more

Here we see PETSc timers translated into TAU timers using the Perfstubs library.

No modification to the source, build system, or the binary!

Generating callpath profiles

```
Edit ex50.qsub
# add
export TAU_CALLPATH=1
export TAU_CALLPATH_DEPTH=100

export TAU_PROFILE_FORMAT=merged
mpirun ...

% qsub ex50.qsub
% paraprof tauprofile.xml
```

Generating Traces

```
# cd ~/examples/petsc-cuda; vi ex50.qsub
# Comment out previous CALLPATH options
export TAU_TRACE=1

% qsub ex50.qsub
% tau_treemerge.pl
% tau_trace2json tau.trc tau.edf -chrome \
  -ignoreatomic -o ex50.json
Open Firefox, load Perfetto.dev
trace visualizer and open ex50.json
wasd keys to widen/shrink/left/right
```

Using TAU on Polaris at ALCF

```
% qsub -I -l select=1 -l filesystems=home:eagle -l walltime=1:00:00 -q R2035675 -A ATPESC2024 -X
```

```
% module use /soft/modulefiles; module load tau
```

```
% wget http://tau.uoregon.edu/workshop_atpesc24.tgz; tar xf workshop_atpesc24.tgz
```

```
% cd workshop; cat README; cd TeaLeaf_CUDA; make; cd bin; ./run.sh
```

```
% cd ../../petsc-tau; ./clean.sh; ./compile.sh; ./run.sh
```

```
% paraprof --pack petsc_ex19.ppk ; <SCP to AWS>; paraprof petsc_ex19.ppk
```

```
Un-instrumented run with MPI          % aprun -n N ./a.out
```

Profiling an un-instrumented application (use `tau_exec -ebs` with any of the following for event-based sampling):

```
MPI without GPUs:                    % aprun -n N tau_exec -ebs ./a.out
```

```
CUDA with MPI:                        % aprun -n N tau_exec -T cupti,mpi -cupti -ebs ./a.out
```

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

Tracing:

```
Vampir:                               % export TAU_TRACE=1; export TAU_TRACE_FORMAT=otf2
```

```
                                       % aprun -n N tau_exec [options] ./a.out; vampir traces.otf2 &
```

```
Chrome:                               % export TAU_TRACE=1; aprun -n N 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`) or `https://Perfetto.dev`

```
Jumpshot:                             % export TAU_TRACE=1; aprun -n N tau_exec [Options] ./a.out;
```

```
                                       % tau_treemerge.pl; tau2slog2 tau.trc tau.edf -o app.slog2; jumpshot app.slog2
```

TAU Exercise #3:

paraprof 3D display

TAU paraprof

terminal10@e4s-24-05-aws: ~/examples/tau

```
tutorial10@e4s-24-05-aws: ~/examples/tau$ ls
demo.ppk  fetch.sh
tutorial10@e4s-24-05-aws: ~/examples/tau$ paraprof demo.ppk &
[1] 309831
tutorial10@e4s-24-05-aws: ~/examples/tau$
```

TrialField	
Name	demo.ppk
Application ID	0
Experiment ID	0
Trial ID	0
File Type Index	0
File Type Name	ParaProf Pa

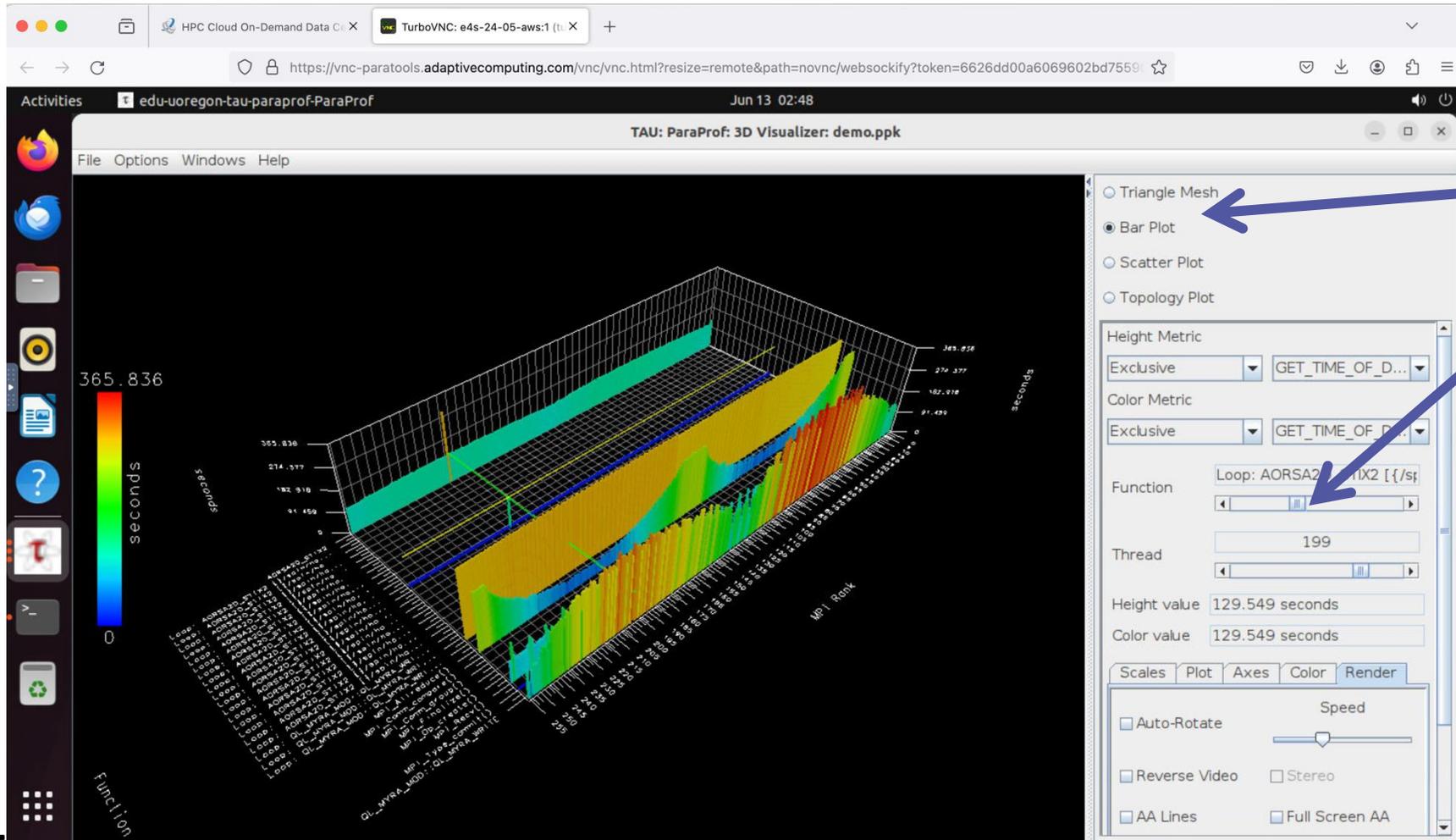
TAU: ParaProf Manager

TAU: ParaProf: demo.ppk

Choose 3D Visualization

cd ~/examples/tau
paraprof demo.ppk &

TAU paraprof 3D visualization

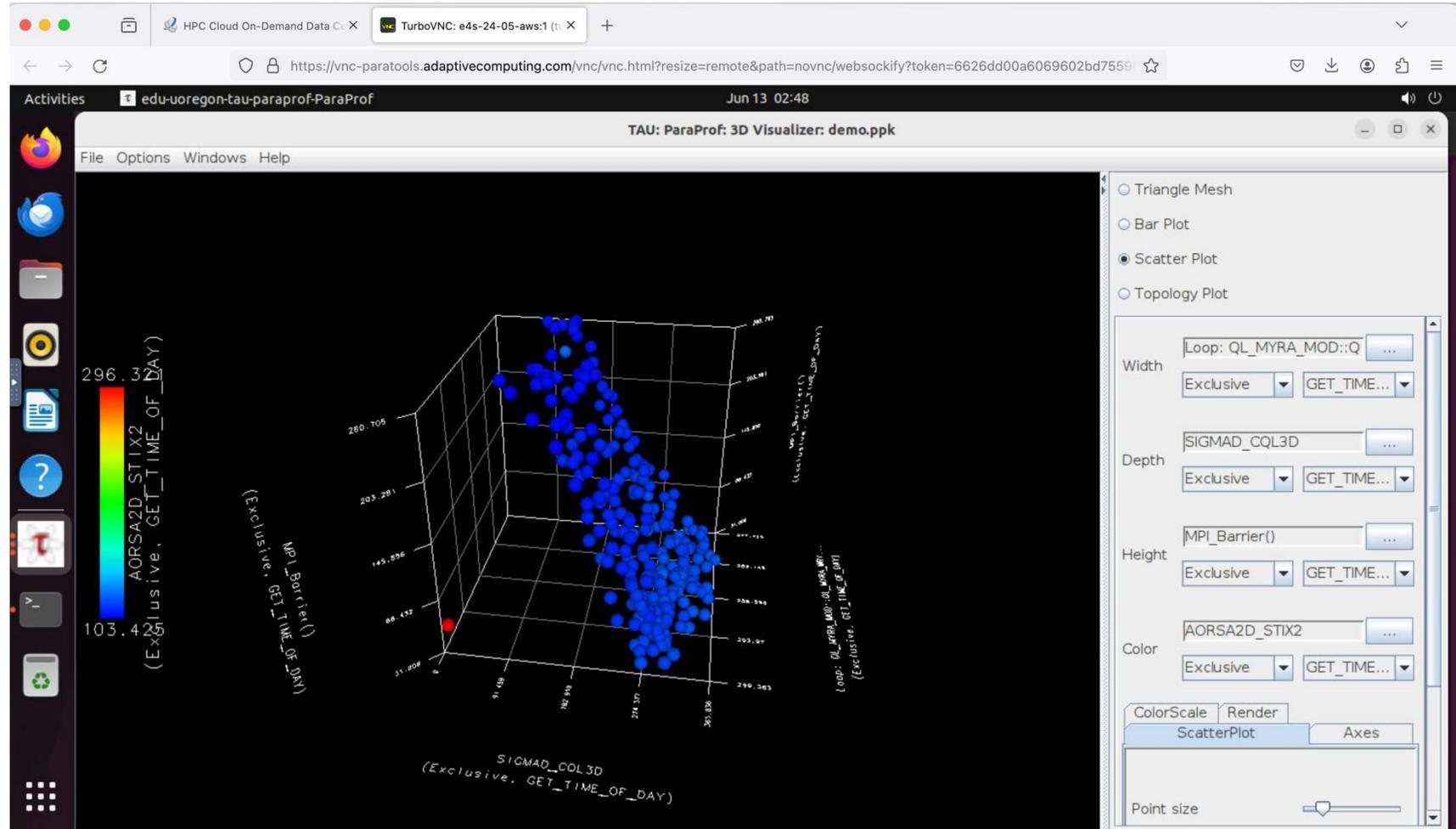


Choose Bar Plot and move
Function and Thread
Sliders

First mouse button to rotate
Second mouse button to
translate (left to right)
Scroll wheel (or +/- keys) to
zoom in.

Try Scatter plot next

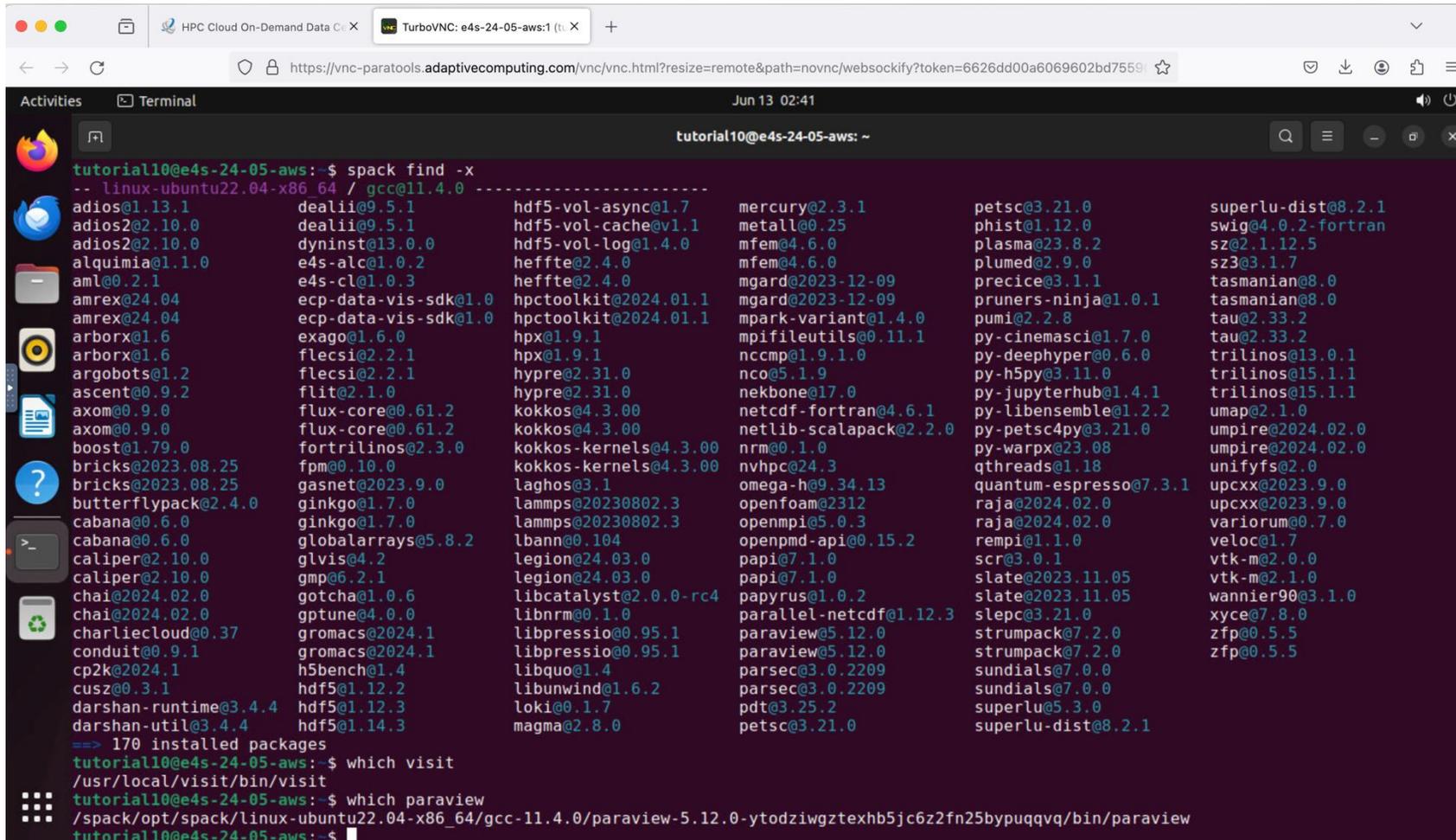
TAU paraprof: 3D Scatter Plot



Exercise #7

E4S: Extreme-scale Scientific Software Stack

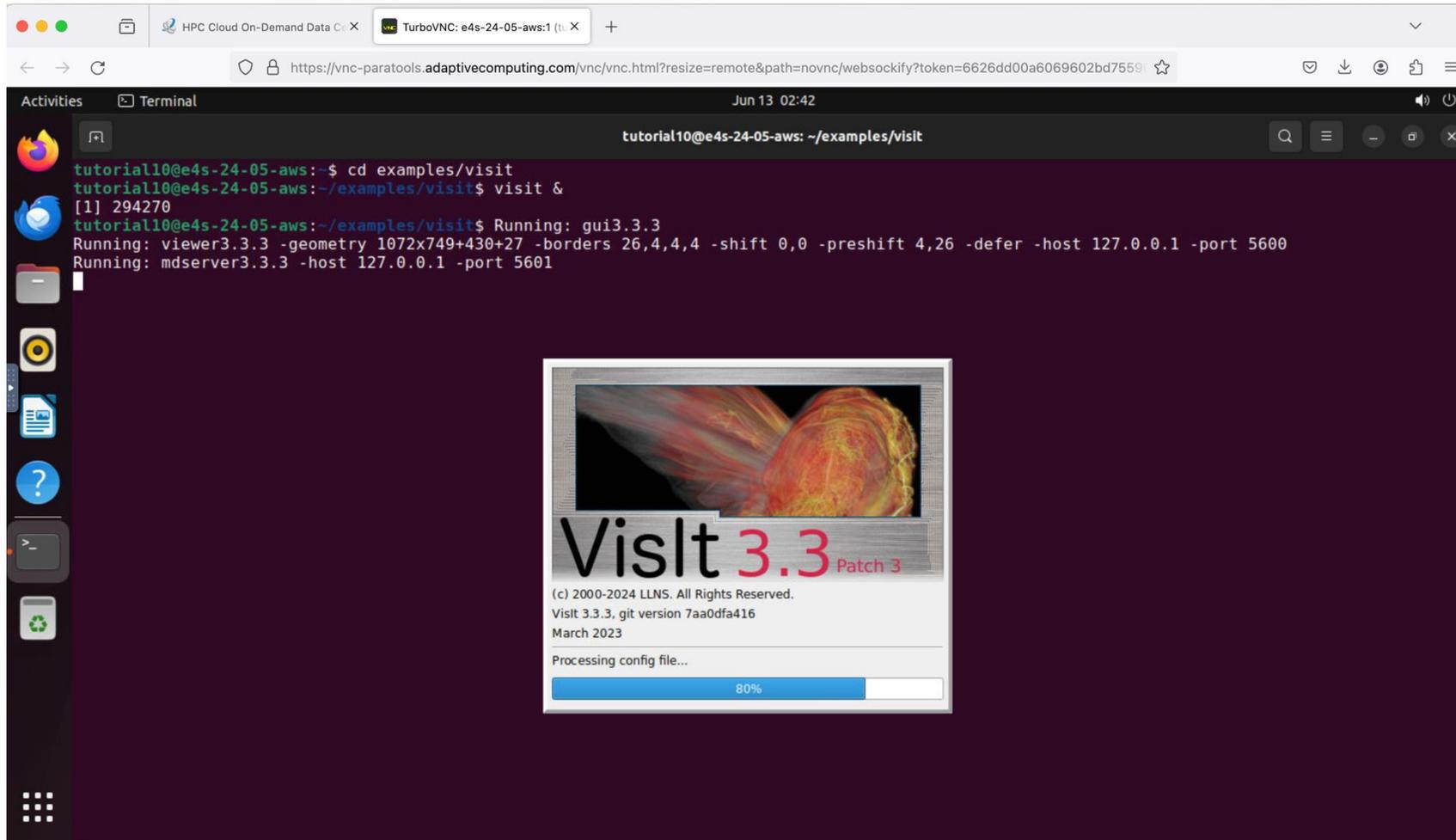
Spack package manager [https://spack.io]



```
tutorial10@e4s-24-05-aws: ~  
tutorial10@e4s-24-05-aws:~$ spack find -x  
-- linux-ubuntu22.04-x86_64 / gcc@11.4.0 -----  
adios@1.13.1      dealii@9.5.1      hdf5-vol-async@1.7      mercury@2.3.1      petsc@3.21.0      superlu-dist@8.2.1  
adios2@2.10.0    dealii@9.5.1      hdf5-vol-cache@v1.1    metall@0.25        phist@1.12.0      swig@4.0.2-fortran  
adios2@2.10.0    dyninst@13.0.0    hdf5-vol-log@1.4.0     mfem@4.6.0         plasma@23.8.2     sz@2.1.12.5  
alquimia@1.1.0   e4s-alc@1.0.2     heffte@2.4.0           mfem@4.6.0         plumed@2.9.0     sz3@3.1.7  
aml@0.2.1        e4s-cl@1.0.3      heffte@2.4.0           mgard@2023-12-09  precice@3.1.1     tasmanian@8.0  
amrex@24.04      ecp-data-vis-sdk@1.0.0 hpctoolkit@2024.01.1  mgard@2023-12-09  pruners-ninja@1.0.1 pumi@2.2.8     tasmanian@8.0  
amrex@24.04      ecp-data-vis-sdk@1.0.0 hpctoolkit@2024.01.1  mpark-variant@1.4.0 pumil@2.2.8      tau@2.33.2  
arborx@1.6       exago@1.6.0       hpx@1.9.1              mpifileutils@0.11.1 py-cinemasci@1.7.0 tau@2.33.2  
arborx@1.6       flecsi@2.2.1      hpx@1.9.1              nccmp@1.9.1.0     py-deephyper@0.6.0 trilinos@13.0.1  
argobots@1.2     flecsi@2.2.1      hypre@2.31.0           nco@5.1.9          py-h5py@3.11.0   trilinos@15.1.1  
ascent@0.9.2     flit@2.1.0        kokkos@4.3.00          nekbone@17.0      py-jupyterhub@1.4.1 py-libensemble@1.2.2 umap@2.1.0  
axom@0.9.0       flux-core@0.61.2  kokkos@4.3.00          netcdf-fortran@4.6.1 py-petsc4py@3.21.0 umpire@2024.02.0  
axom@0.9.0       flux-core@0.61.2  kokkos@4.3.00          netlib-scalapack@2.2.0 nrm@0.1.0        umphire@2024.02.0  
boost@1.79.0     fortrilinos@2.3.0 lbann@0.104            nvhpc@24.3         py-warpx@23.08   unifyfs@2.0  
bricks@2023.08.25 fpm@0.10.0        gasnet@2023.9.0        omega-h@9.34.13   quantum-espresso@7.3.1 upcxx@2023.9.0  
bricks@2023.08.25 ginkgo@1.7.0      gromacs@2024.1         lammps@20230802.3 openfoam@2312     raja@2024.02.0   upcxx@2023.9.0  
butterflypack@2.4.0 ginkgo@1.7.0     gromacs@2024.1         lammps@20230802.3 openmpi@5.0.3     raja@2024.02.0   variorum@0.7.0  
cabana@0.6.0     globalarrays@5.8.2 lbann@0.104            libcat@2.0.0-rc4  openpmd-api@0.15.2 rempi@1.1.0     veloc@1.7  
cabana@0.6.0     glvis@4.2         libarmadillo@11.8.0    libcat@2.0.0-rc4  papi@7.1.0       scr@3.0.1        vtk-m@2.0.0  
caliper@2.10.0   gmp@6.2.1        libarmadillo@11.8.0    libcat@2.0.0-rc4  papi@7.1.0       slate@2023.11.05  vtk-m@2.1.0  
caliper@2.10.0   gotcha@1.0.6     libarmadillo@11.8.0    libcat@2.0.0-rc4  papyrus@1.0.2    slate@2023.11.05  wannier90@3.1.0  
chai@2024.02.0   gptune@4.0.0     libarmadillo@11.8.0    libcat@2.0.0-rc4  paraview@5.12.0  slepc@3.21.0     xyce@7.8.0  
chai@2024.02.0   gromacs@2024.1   libarmadillo@11.8.0    libcat@2.0.0-rc4  paraview@5.12.0  strumpack@7.2.0   zfp@0.5.5  
charliecloud@0.37 gromacs@2024.1   libarmadillo@11.8.0    libcat@2.0.0-rc4  paraview@5.12.0  strumpack@7.2.0   zfp@0.5.5  
conduit@0.9.1    gromacs@2024.1   libarmadillo@11.8.0    libcat@2.0.0-rc4  parsec@3.0.2209  sundials@7.0.0   sundials@7.0.0  
cp2k@2024.1      h5bench@1.4       libarmadillo@11.8.0    libcat@2.0.0-rc4  parsec@3.0.2209  superlu@5.3.0    superlu@5.3.0  
cusz@0.3.1       hdf5@1.12.2      libarmadillo@11.8.0    libcat@2.0.0-rc4  pdt@3.25.2       superlu-dist@8.2.1  
darshan-runtime@3.4.4 hdf5@1.12.3      loki@0.1.7             magma@2.8.0        petsc@3.21.0  
darshan-util@3.4.4 hdf5@1.14.3      magma@2.8.0  
==> 170 installed packages  
tutorial10@e4s-24-05-aws:~$ which visit  
/usr/local/visit/bin/visit  
tutorial10@e4s-24-05-aws:~$ which paraview  
/spack/opt/spack/linux-ubuntu22.04-x86_64/gcc-11.4.0/paraview-5.12.0-ytodziwgtztxhb5jc6z2fn25byuqqvq/bin/paraview  
tutorial10@e4s-24-05-aws:~$
```

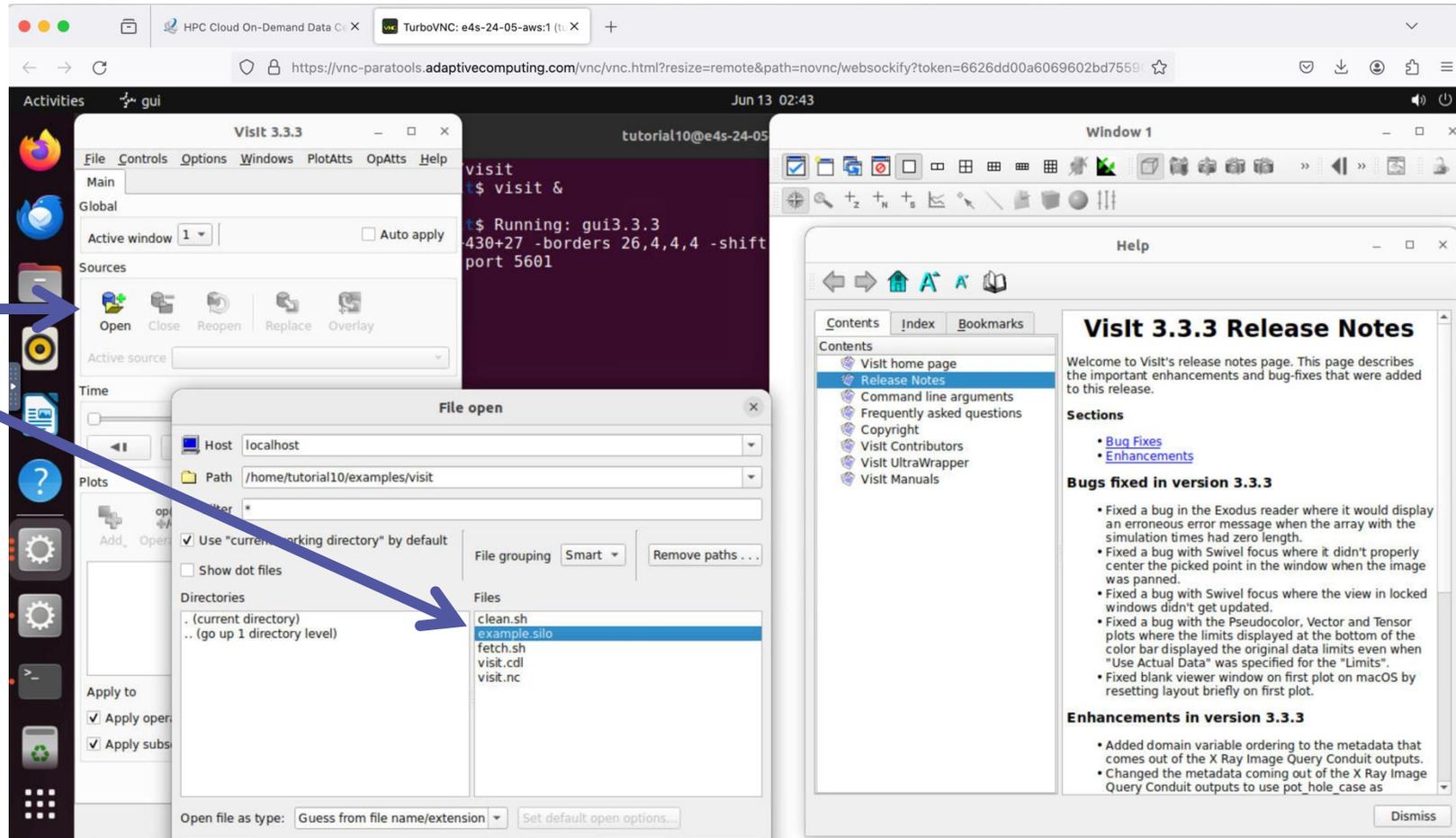
spack find -x
spack find

VisIt visualizer



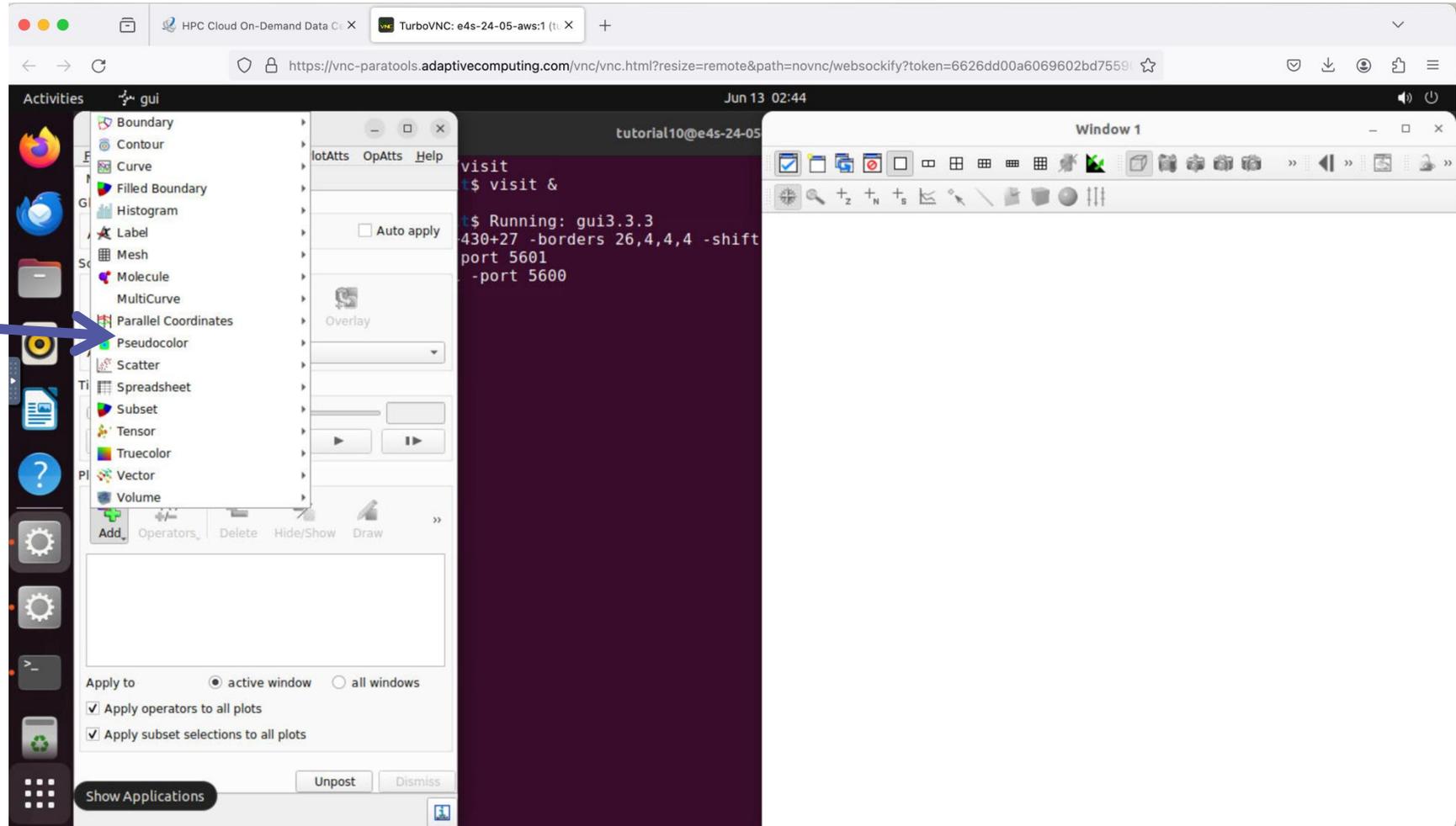
cd ~/examples/visit
visit &

VisIt visualizer



VisIt visualizer

Add Pseudocolor ->
Pressure



VisIt visualizer

Add Pseudocolor ->
Pressure
Click Draw
Rotate image

Activities viewer Jun 13 02:44

Visit 3.3.3

File Controls Options Windows PlotAtts OpAtts Help

Main

Global

Active window 1 Auto apply

Sources

Open Close Reopen Replace Overlay

Active source example.silo

Time

Plots

Add Operators Delete Hide/Show Draw

Pseudocolor - pressure

Apply to active window all windows

Apply operators to all plots

Apply subset selections to all plots

Unpost Dismiss

DB: example.silo
Cycle: 0

Pseudocolor
Var: pressure
Units: Pa

4.610
3.442
2.273
1.104
Max: 5.779
Min: 1.104

Height (parsec)

Width (parsec)

Depth (parsec)

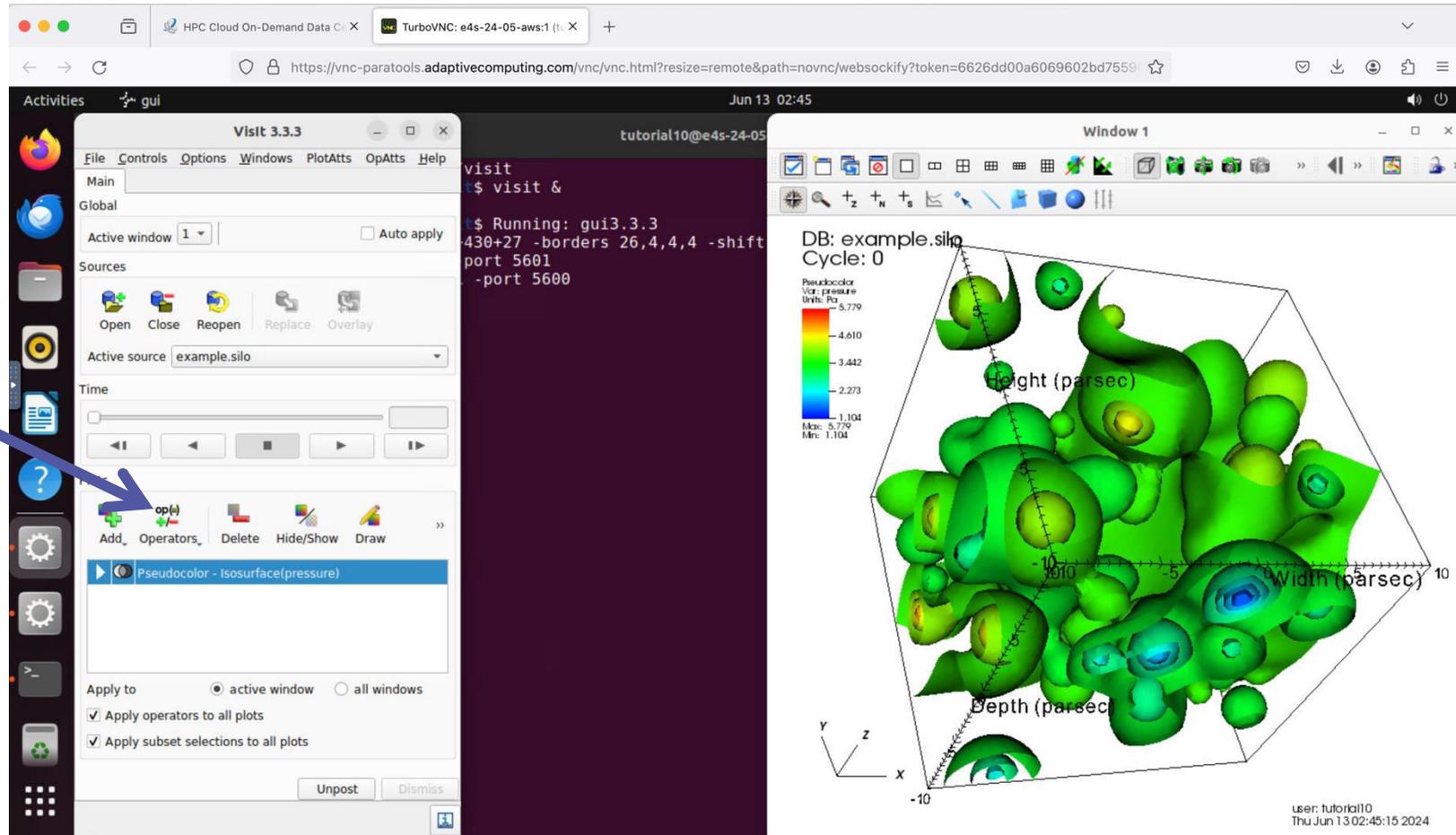
10
5
0
-5
-10

10
5
0
-5
-10

10
5
0
-5
-10

user: tutorial10
Thu Jun 13 02:44:24 2024

VisIt visualizer



Add Operators ->
Isosurface
Click Draw
Rotate image

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> -papi=<dir> -mpi
-pthread -c++=mpicxx -cc=mpicc -fortran=mpif90
-dwarf=download -unwind=download -otf=download
-iowrapper -papi=<dir>`
- `make install`

•Using TAU for source instrumentation (not needed with `tau_exec`):

- `export TAU_MAKEFILE=<taudir>/x86_64/lib/Makefile.tau-<TAGS>`
- `make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh`

Compile-Time Options

- Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

-optVerbose	Turn on verbose debugging messages
-optCompInst	Use compiler based instrumentation
-optNoCompInst	Do not revert to compiler instrumentation if source instrumentation fails.
-optTrackIO	Wrap POSIX I/O call and calculates vol/bw of I/O operations (Requires TAU to be configured with <i>-iowrapper</i>)
-optTrackGOMP	Enable tracking GNU OpenMP runtime layer (used without <i>-opari</i>)
-optMemDbg	Enable runtime bounds checking (see TAU_MEMDBG_* env vars)
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optPreProcess	Preprocess sources (OpenMP, Fortran) before instrumentation
-optTauSelectFile="<file>"	Specify selective instrumentation file for <i>tau_instrumentor</i>
-optTauWrapFile="<file>"	Specify path to <i>link_options.tau</i> generated by <i>tau_gen_wrapper</i>
-optHeaderInst	Enable Instrumentation of headers
-optTrackUPCR	Track UPC runtime layer routines (used with 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)	

Compile-Time Options (contd.)

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

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

See tau_compiler.sh for a full list of TAU_OPTIONS.

...

TAU's Runtime Environment Variables

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

Runtime Environment Variables

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

Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACK_MEMORY_LEAKS	0	Tracks allocates that were not de-allocated (needs <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_ALIGNMENT	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

Performance Research Laboratory, University of Oregon, Eugene



Support Acknowledgements

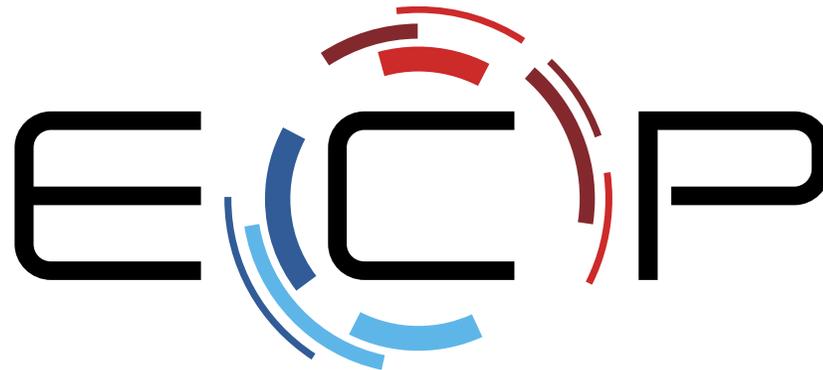
- US Department of Energy (DOE)
 - ANL
 - Office of Science contracts, ECP
 - SciDAC, LBL contracts
 - LLNL-LANL-SNL ASC/NNSA contract
 - Battelle, PNNL and ORNL contract
- Department of Defense (DoD)
 - PETTT, HPCMP
- National Science Foundation (NSF)
 - SI2-SSI, Glassbox, E4S Workshop
- NASA
- Intel
- CEA, France
- Partners:
 - University of Oregon
 - The Ohio State University
 - ParaTools, Inc.
 - University of Tennessee, Knoxville
 - T.U. Dresden, GWT
 - Jülich Supercomputing Center



Thank you

<https://www.exascaleproject.org>

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



EXASCALE COMPUTING PROJECT

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

Acknowledgment

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



U.S. DEPARTMENT OF
ENERGY

Office of
Science

<https://science.osti.gov/ascr>

<https://pesoproject.org>

<https://ascr-step.org>



