Improving productivity using TAU and E4S

October 22, 2020, 11am PT
via Zoom

Sameer Shende
Research Associate Professor and Director
Performance Research Laboratory, OACISS, University of Oregon
http://tau.uoregon.edu/tau_xpert20.pdf

Tuning and Analysis Utilities

E4S

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

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.

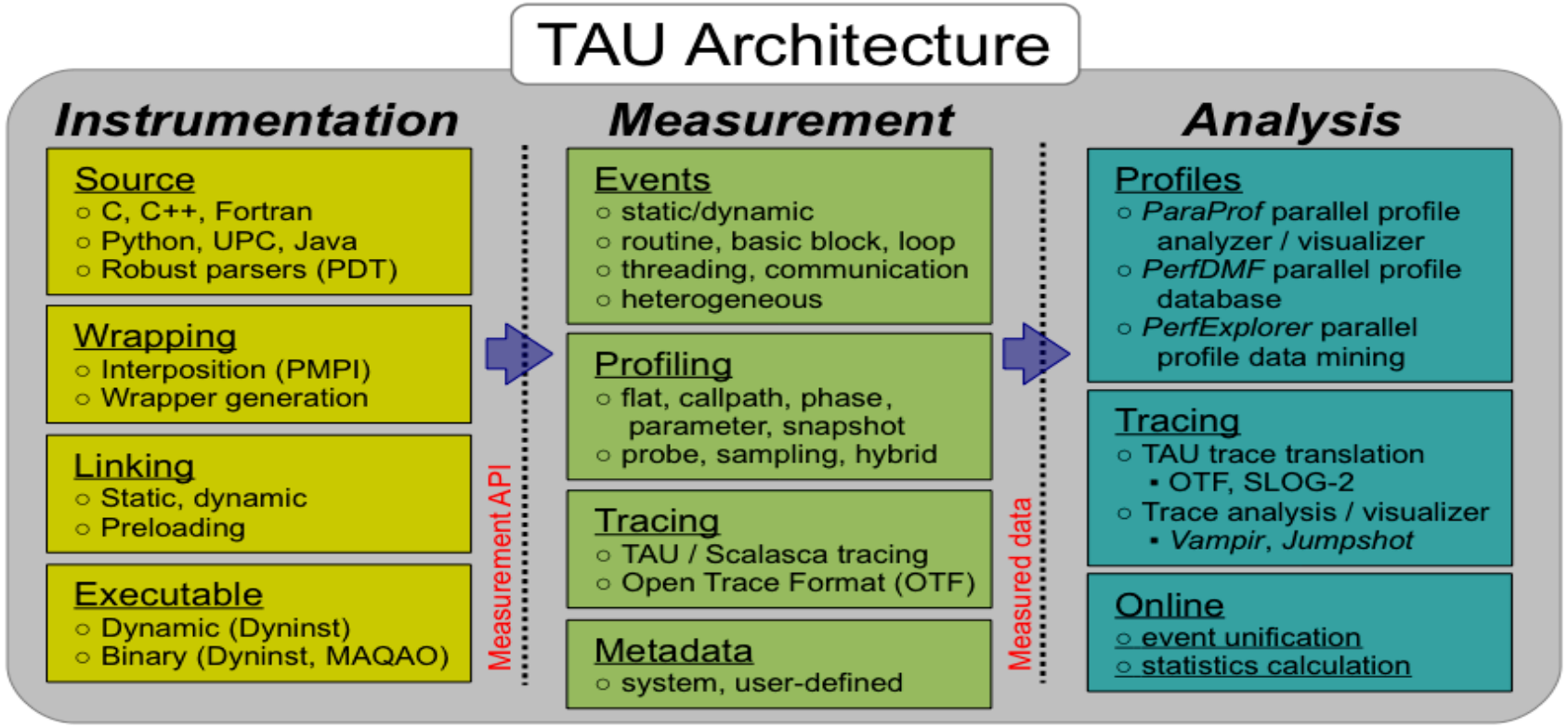
Motivation: Improving Productivity

- 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®

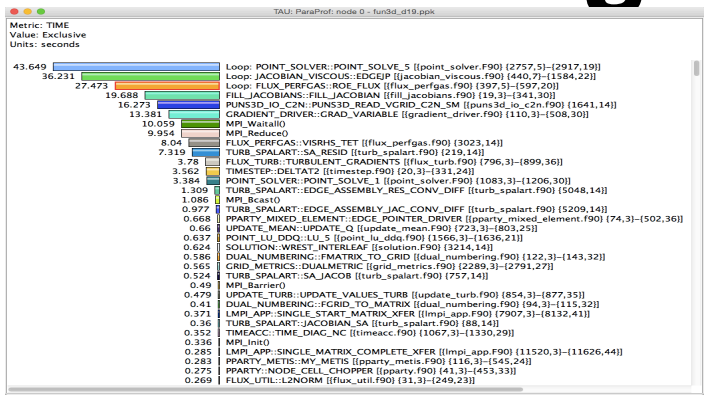
Parallel performance framework and toolkit

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



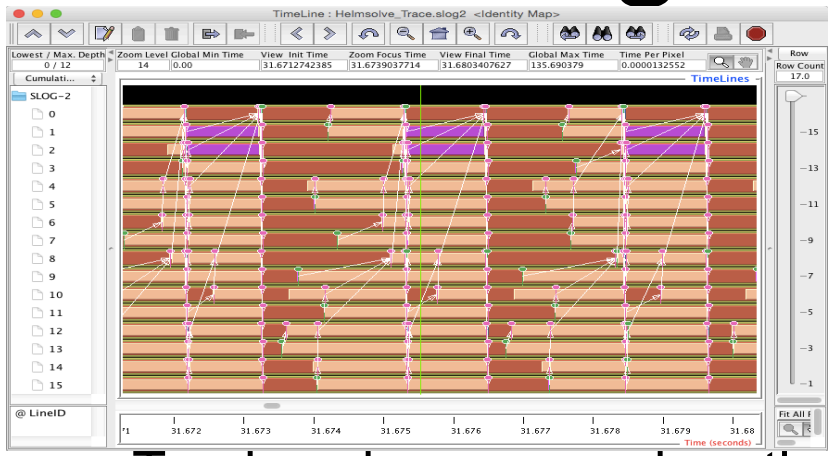
Profiling and Tracing

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

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?

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>
```

Extreme-scale Scientific Software Stack (E4S)

https://e4s.io

EXASCALE COMPUTING PROJECT

E4S: Extreme-scale Scientific Software Stack

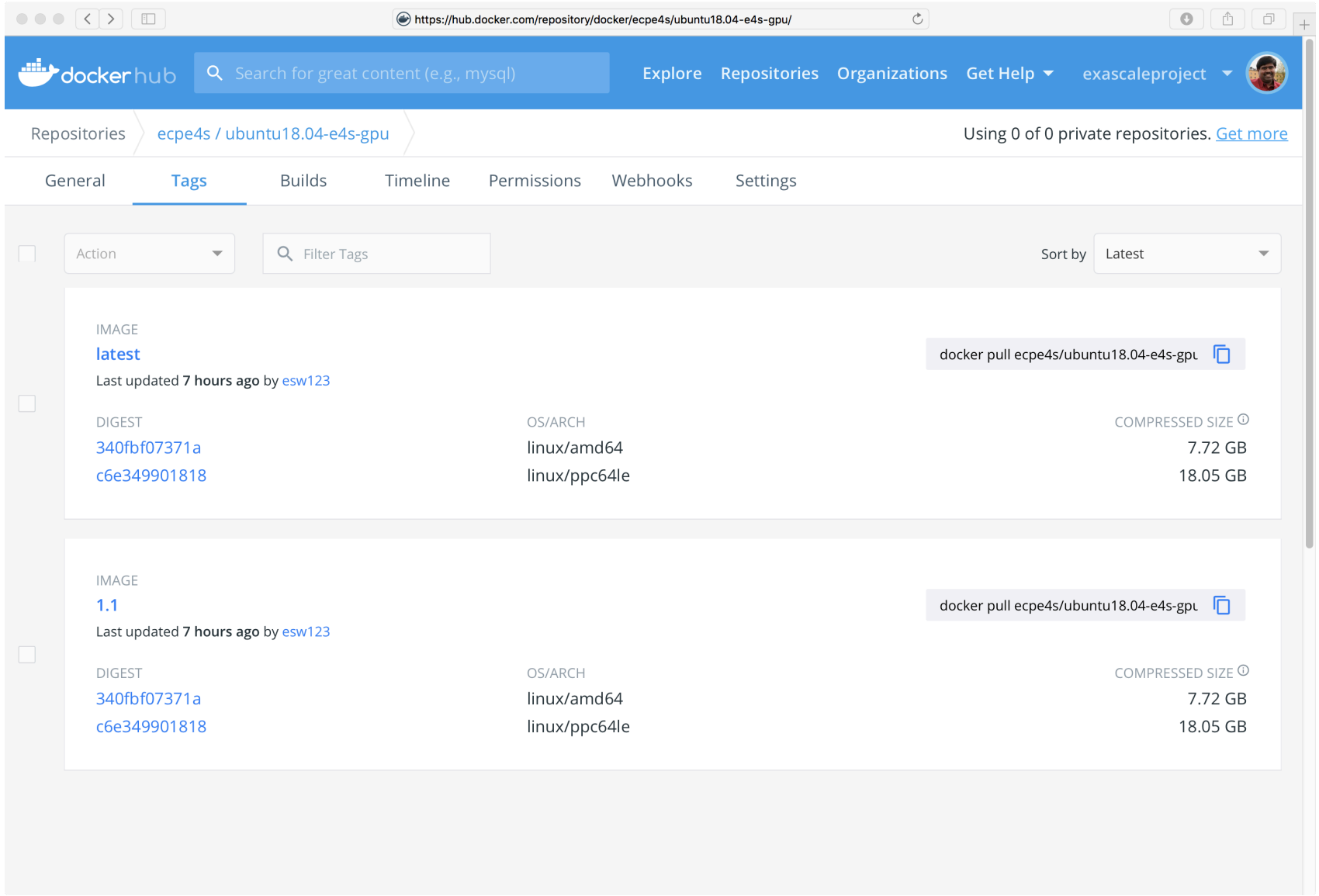
- Curated release of ECP ST products based on Spack [http://spack.io] package manager
- Spack binary build caches for bare-metal installs
  - x86_64, ppc64le (IBM Power 9), and aarch64 (ARM64)
- Container images on DockerHub and E4S website of pre-built binaries of ECP ST products
- Base images and full 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

Download E4S v1.1 GPU image

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

```
# docker pull ecpe4s/ubuntu18.04-e4s-gpu
```

RHEL 7

SPACK MINIMAL

ecpe4s/rhel7-spack docker GitHub

E4S COMPREHENSIVE

ecpe4s/rhel7-e4s docker GitHub

CUSTOM

ecpe4s/superlu_sc docker GitHub

Ubuntu 18.04

E4S GPU IMAGE

- ecpe4s/ubuntu18.04-e4s-gpu docker GitHub

x86_64 version: CUDA and ROCM

ppc64le version: CUDA

SPACK MINIMAL

ecpe4s/ubuntu18.04-spack docker GitHub

E4S COMPREHENSIVE

ecpe4s/ubuntu18.04-e4s docker GitHub

CentOS 7

SPACK MINIMAL

ecpe4s/centos7-spack docker GitHub

E4S COMPREHENSIVE

ecpe4s/centos7-e4s docker GitHub

CUSTOM

- - - -

https://e4s.io

Spack provides the *spec* syntax to describe custom configurations

No installation required: clone and go

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

```
$ 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

E4S: Spack Build Cache at U. Oregon

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

E4S Build Cache for Spack 0.15.4

To use this build cache, just add it to your Spack

spack mirror add E4S https://cache.e4s.io

spack buildcache keys -it

Click on one of the packages below to see a list of all available variants.

All Architectures | PPC64LE | X86_64

All Operating Systems | Centos 7 | Centos 8 | RHEL 7 | RHEL 8 | Ubuntu 18.04 | Ubuntu 20.04

Last updated: 09-10-2020 08:45 PDT

21905 Spack packages

Search

adiak@0.1.1 adios2@2.5.0 adios2@2.6.0 adios@1.13.1 adlbx@0.9.2 amg@1.2 aml@0.1.0 amrex@20.07 ant@1.10.0 ant@1.10.7

argobots@1.0 argobots@1.0rc1 argobots@1.0rc2 arpack-ng@3.7.0 ascent@develop autoconf@2.69 automake@1.16.1 automake@1.16.2

axl@0.1.1 axl@0.3.0 axom@0.3.3 bdftopcf@1.0.5 berkeley-db@18.1.40 berkeley-db@6.2.32 binutils@2.31.1 binutils@2.32 binutils@2.33.1

binutils@2.34 bison@3.4.2 bmi@develop bolt@1.0 bolt@1.0rc2 bolt@1.0rc3 boost@1.70.0 boost@1.72.0 boost@1.73.0 boost@1.74.0

butterflypack@1.1.0 butterflypack@1.2.0 bzip2@1.0.8 c-blosc@1.17.0 caliper@2.0.1 caliper@2.2.0 caliper@2.3.0 caliper@2.4.0

camtimers@master catalyst@5.6.0 cinch@develop cinch@master cmake@3.13.4 cmake@3.14.5 cmake@3.14.7 cmake@3.15.4 cmake@3.16.2

- 20,000+ binaries
- S3 mirror
- No need to build from source code!

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

Conclusions

- Avoid DIY approaches!
- Use prebuilt E4S Singularity and Docker containers with software packages preinstalled
- TAU helps evaluate the performance of applications and is installed in E4S containers
- Docker and Singularity images are available for download
  - https://e4s.io

Download TAU from U. Oregon

Tuning and Analysis Utilities

http://tau.uoregon.edu

for more information

Free download, open source, BSD license

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."