

E4S: A Platform for HPC-AI Tool Interoperability

Scalable Tools Workshop 2025

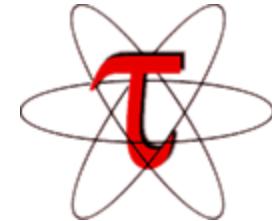
3:30pm – 4pm PDT, July 7, 2025
Granlibakken Resort, Lake Tahoe, CA

Sameer Shende
Research Professor and Director,
Performance Research Laboratory, OACISS, University of Oregon
President and Director, ParaTools, Inc.
sameer@cs.uoregon.edu
https://oaciss.uoregon.edu/e4s/talks/E4S_Scalable_Tools25.pdf



HPSF

HIGH PERFORMANCE
SOFTWARE FOUNDATION



Motivation

- As our software gets more complex, it is getting harder and harder to install and operate our performance evaluation tools and libraries for HPC-AI workflows on GPUs correctly in an integrated and interoperable software stack!

Solutions

- E4S: An HPC-AI software platform for tool integration
- Frank@UO: A hardware platform for CI for performance tools
- ParaTools Pro for E4S™: A cloud image for tool integration

Extreme-scale Scientific Software Stack (E4S)

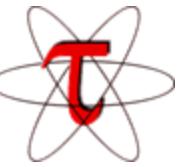
<https://e4s.io>



Motivation

- What are the requirements for performance evaluation tools?

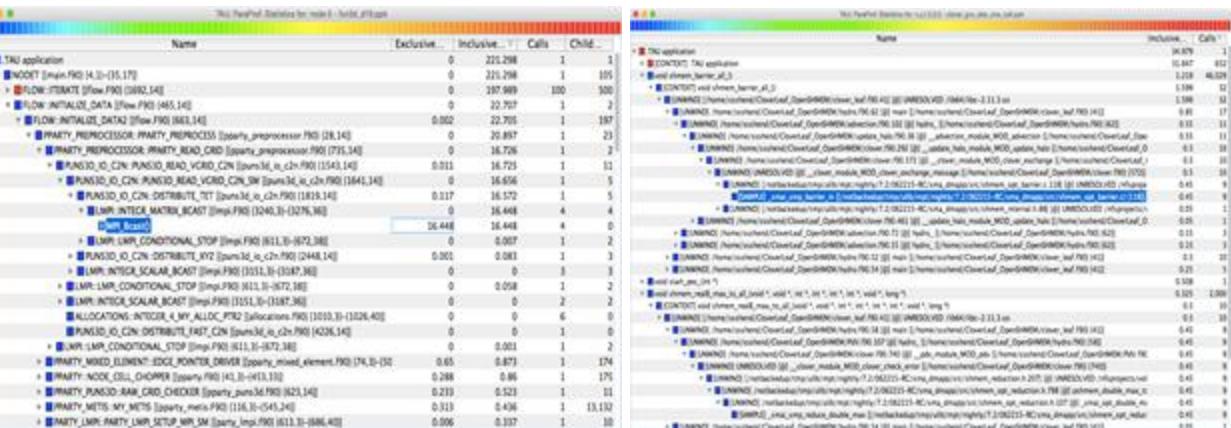
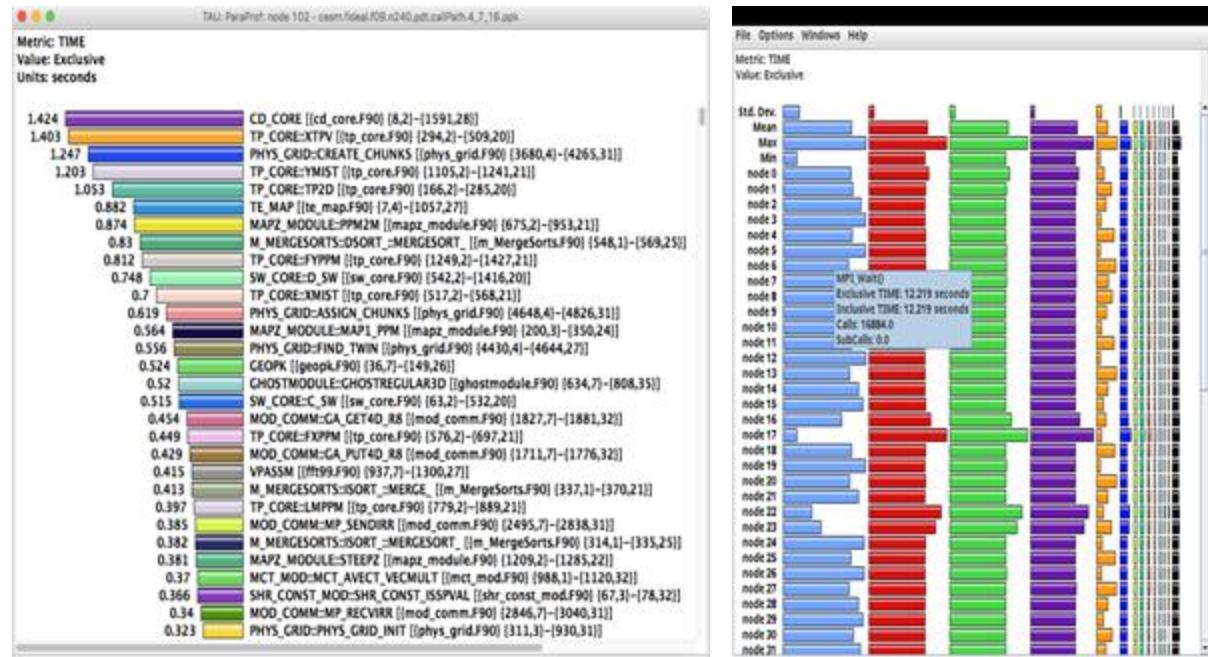
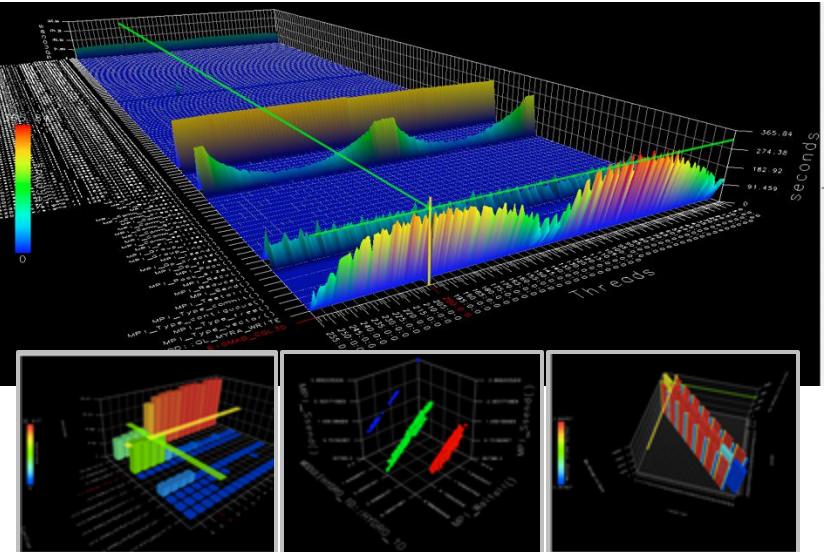
TAU Performance System®



Portable profiling and tracing toolkit for performance analysis of HPC parallel programs

- Supports most parallel execution models
- Provides instrumentation and measurement
- Parallel profiling analysis and data mining
- Open source: <http://tau.uoregon.edu>

TAU runs on most HPC platforms



<https://tau.uoregon.edu>



Our performance evaluation tools are getting complex to install!

- GPU Runtimes: ROCm, CUDA/CUPTI, Intel® oneAPI
- Tool interfaces: ROCprofiler-sdk, CUPTI, Intel Level Zero, OpenCL, MPI_T, OMPT, Kokkos, RAJA, Caliper, CAMTimers, PerfStubs, GPTL, phiprof, ittnotify, ...
- Tool dependencies:
 - DyninstAPI, binutils, libunwind, libdwarf installed just right (-fPIC used to compile .o files in DSOs)
 - Languages: C++/C/Fortran, Python, Rust, Chapel
 - Tools: Qt5, Java, perl, ruby, autotools, sed, awk, cmake...
 - PAPI, Likwid for hardware performance counter access
 - Compilers: LLVM, GNU, Intel, AMD, NVHPC, PrgEnv-{cray,amd,nvidia,intel,gnu-amd} on HPE CPE
 - MPI (MVAPICH with MPI_T, zfp compression, GDR, OFI/UCX/EFA)
 - Intel TBB
 - Boost
 - Other third-party libraries (e.g., AWS NeuronX SDK for AWS Trainium and Inferentia nodes) ...
- Installing these tools and their dependencies by hand is hard!

Tool dependencies: HPCToolkit, DyninstAPI, TAU, OpenFOAM

```
Singularity> spack find -d hpctoolkit+rocm
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 -----
hpctoolkit@2024.01.1
  autoconf@2.72
    m4@1.4.19
      libsigsegv@2.14
    perl@5.40.0
      berkeley-db@18.1.40
        gdbm@1.23
          readline@8.2
    automake@1.16.5
    boost@1.86.0
    bzip2@1.0.8
      diffutils@3.10
    dyninst@13.0.0
      cmake@3.31.6
      curl@8.11.1
        nghttp2@1.65.0
        openssl@3.4.1
          ca-certificates-mozilla@2025-02-25
        perl@5.40.0
          berkeley-db@18.1.40
        gdbm@1.23
          readline@8.2
      ncurses@6.5
      zlib-ng@2.2.3
    elfutils@0.190
      libiconv@1.17
      pkgconf@2.3.0
      zstd@1.5.6
  gcc-runtime@11.4.0
  glibc@2.35
  gmake@4.4.1
  hip@6.3.2
  hpcviewer@2025.01
    openjdk@17.0.11_9
  hsa-rocr-dev@6.3.2
  intel-tbb@2022.0.0
    hwloc@2.11.1
      libpciaccess@0.17
      util-macros@1.20.1
      libxml2@2.13.5
  intel-xed@2024.05.20
    python@3.10.12
  liberty@2.41
  libmonitor@2023.03.15
  libtool@2.4.7
    findutils@4.10.0
    gettext@0.23.1
    tar@1.35
    pigz@2.8
  libunwind@1.8.1
  mpich@4.2.3
  papi@7.1.0
  rocprofiler-dev@6.3.2
  roctracer-dev@6.3.2
  xerces-c@3.3.0
    curl@8.11.1
      nghttp2@1.65.0
      openssl@3.4.1
        ca-certificates-mozilla@2025-02-25
  xz@5.6.3
  yaml-cpp@0.8.0
  zlib-ng@2.2.3
```

```
Singularity> spack find -d dyninst
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 -----
dyninst@13.0.0
  boost@1.86.0
    mpich@4.2.3
  cmake@3.31.6
  curl@8.11.1
    nghttp2@1.65.0
    openssl@3.4.1
      ca-certificates-mozilla@2025-02-25
    perl@5.40.0
      berkeley-db@18.1.40
    gdbm@1.23
      readline@8.2
  ncurses@6.5
  zlib-ng@2.2.3
  elfutils@0.190
    bzip2@1.0.8
      diffutils@3.10
    libiconv@1.17
    m4@1.4.19
      libsigsegv@2.14
    xz@5.6.3
  gcc-runtime@11.4.0
  glibc@2.35
  gmake@4.4.1
  hip@6.3.2
  hsa-rocr-dev@6.3.2
  hwloc@2.11.1
    libpciaccess@0.17
    util-macros@1.20.1
    libxml2@2.13.5
  intel-tbb@2022.0.0
    hwloc@2.11.1
      libpciaccess@0.17
      util-macros@1.20.1
      libxml2@2.13.5
  liberty@2.41
```

```
Singularity> spack find -d tau+rocm
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 -----
tau@2.34.1
  binutils@2.43.1
  diffutils@3.10
  pkgconf@2.3.0
  zstd@1.5.6
  comgr@6.3.2
  elfutils@0.192
    bzip2@1.0.8
    gettext@0.23.1
    tar@1.35
    pigz@2.8
  libiconv@1.17
  m4@1.4.19
    libsigsegv@2.14
    xz@5.6.3
  gcc-runtime@11.4.0
  glibc@2.35
  gmake@4.4.1
  hip@6.3.2
  hsa-rocr-dev@6.3.2
  hwloc@2.11.1
    libpciaccess@0.17
    util-macros@1.20.1
    libxml2@2.13.5
    ncurses@6.5
  libdwarf@0.11.0
  cmake@3.31.6
  curl@8.11.1
    nghttp2@1.65.0
    openssl@3.4.1
      ca-certificates-mozilla@2025-02-25
    perl@5.40.0
      berkeley-db@18.1.40
    gdbm@1.23
      readline@8.2
  libunwind@1.8.1
  mpich@4.2.3
  openjdk@17.0.11_9
  otf2@3.0.3
    python@3.10.12
    python-venv@1.0
  papi@7.1.0
  pdt@3.25.2
  rocm-core@6.3.2
  rocm-smi-lib@6.3.2
  rocprofiler-sdk@6.3.2
  zlib-ng@2.2.3
```

```
Singularity> spack find -d openfoam
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 -----
openfoam@2412
  adios2@2.10.2
    bzip2@1.0.8
    c-blosc2@2.15.1
    lizard@2.0
    lz4@1.10.0
    snappy@1.2.1
    gmake@4.4.1
    libcatayst@2.0.0
    libfabric@1.22.0
    libffl@3.4.6
    libpng@1.6.39
    mgard@2023-12-09
      libarchive@3.7.6
      lz@2.10
      protobuf@3.28.2
        obseil-cpp@20240722.0
      python@3.10.12
      sed@4.9
      yaml-cpp@0.8.0
    pkgconf@2.3.0
    sz@2.1.12.5
    zfp@0.5.5
  boost@1.86.0
    icu4c@74.2
    xz@5.6.3
    zstd@1.5.6
  cgal@5.6
    eigen@3.4.0
    gmp@6.3.0
    mpfr@4.2.1
    autoconf-archive@2023.02.20
    texinfo@7.1
  cmake@3.31.6
    curl@8.11.1
      nghttp2@1.65.0
      openssl@3.4.1
        ca-certificates-mozilla@2025-02-25
      ncurses@6.5
    fftw@3.3.10
    flex@2.6.4
      autoconf@2.72
      perl@5.40.0
        berkeley-db@18.1.40
        gdbm@1.23
      automake@1.16.5
      bison@3.8.2
      diffutils@3.10
      libiconv@1.17
      findutils@4.10.0
      gettext@0.23.1
      libxml2@2.13.5
      tar@1.35
      pigz@2.8
      help2man@1.49.3
      libtool@2.4.7
    gcc-runtime@11.4.0
    glibc@2.35
    json-c@0.18
    libyaml@0.2.5
    m4@1.4.19
      libsigsegv@2.14
    mpich@4.2.3
    readline@8.2
    scotch@7.0.6
    zlib-ng@2.2.3
```



How can we build a productive software stack to help tool developers?

- Tool dependencies should be pre-installed
- A consistent environment that we can share with other tool developers to report bugs
- Can containers help here?
- Build your tools with the dependencies inside a container
 - Same kernel as the host OS
 - Can support a different OS
 - Docker/podman and Singularity/Aptainer are popular container runtimes
- Need a base container that can provide the dependencies
- E4S provides a rich set of containers with tools and libraries
- Spack is a versatile package manager for HPC-AI tools and libraries

E4S: Extreme-scale Scientific Software Stack



<https://e4s.io>

About E4S

- E4S, an HPSF project, is an HPC-AI software **ecosystem for science** and 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 software stack 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 100+ HPC (OpenFOAM, Gromacs, Nek5000, LAMMPS), EDA (e.g., Xyce), and AI/ML packages (e.g., NVIDIA NeMo™, NVIDIA BioNeMo™, Vllm, HuggingFace CLI, TensorFlow, PyTorch, OpenCV, TorchBraid, Scikit-Learn, Pandas, JAX, LBANN with support for GPUs where available).
- Base images and full featured containers (with GPU support).
- Commercial support for E4S through ParaTools, Inc. for installation, maintaining an issue tracker, and application engagement.
- E4S for commercial clouds: Adaptive Computing's Heidi AI with ParaTools Pro for E4S™ image for **AWS, GCP, Azure, OCI**.
- 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 CPUs and GPUs from NVIDIA, AMD, and Intel
 - Container images on DockerHub and E4S website of pre-built binaries of ECP ST products.
- 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 25.06 released on June 6, 2025: https://e4s.io/talks/E4S_25.06.pdf



UNIVERSITY
OF OREGON

Argonne
NATIONAL LABORATORY



Brookhaven
National Laboratory

kitware

Lawrence Livermore
National Laboratory

Los Alamos
NATIONAL LABORATORY

OAK RIDGE
National Laboratory
Pacific Northwest
National Laboratory

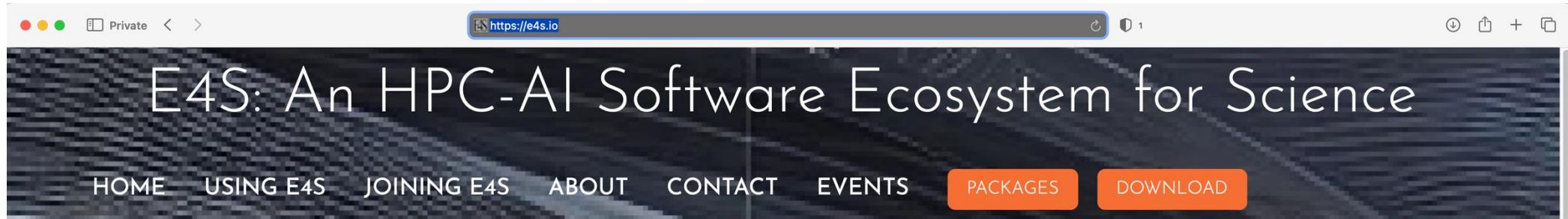
ParaTools

National
Laboratories

Sustainable Horizons Institute

HPSF
HIGH PERFORMANCE SOFTWARE FOUNDATION

E4S Download from <https://e4s.io>



E4S is a community effort to provide and support an open-source software ecosystem for science. E4S provides a curated collection of scientific libraries and tools (packages) that form the foundation for hundreds of the world's most advanced scientific applications.

E4S packages support developing, deploying and running scientific applications on high-performance computing (HPC) and AI platforms sponsored by the **US Department of Energy (DOE)** Office of Advanced Scientific Computing Research. E4S is also used as a foundation for applications on leadership-class computing systems at the **US Department of Defense, US National Science Foundation, and other federal agencies**. It is used on numerous high-performance computing systems at universities and at collaborating international organizations.

E4S provides from-source builds, containers, and pre-installed versions of a broad collection of HPC and AI software packages ([E4S 25.06 release announcement](#)). E4S includes contributions from many organizations, including national laboratories, universities, and industry. E4S is one of the key legacies of the **US Exascale Computing Project (ECP)**, a collaborative effort of the US Department of Energy Office of Advanced Scientific Computing Research and the National Nuclear Security Administration.

E4S Container Download from <https://e4s.io>

The current E4S container offerings include Docker and Singularity images capable of running on X86_64, PPC64LE, and AARCH64 architectures. Our full E4S Release images are based on Ubuntu 22.04 (x86_64, aarch64, ppc64le). In addition to offering a full E4S image containing a comprehensive selection of E4S software released on a bi-annual cycle, we also offer a set of minimal base images suitable for use in Continuous Integration (CI) pipelines where Spack is used to build packages.

Docker images are available on the [E4S Docker Hub](#).

Please see the [E4S 25.06 Release Notes](#).

Container Releases

- [Docker Downloads - CPU only](#)
- [Docker Downloads - CUDA](#)
- [Docker Downloads - ROCm](#)
- [Docker Downloads - OneAPI](#)
- [Singularity x86_64 Download - CPU only](#)
- [Singularity x86_64 Download - CUDA 80](#)
- [Singularity x86_64 Download - CUDA 90](#)
- [Singularity x86_64 Download - CUDA 120](#)
- [Singularity ppc64le Download - CUDA 70](#)
- [Singularity aarch64 Download - CPU only](#)
- [Singularity aarch64 Download - CUDA 75](#)
- [Singularity aarch64 Download - CUDA 80](#)
- [Singularity aarch64 Download - CUDA 90](#)
- [Singularity x86_64 Download - ROCm gfx942](#)
- [Singularity x86_64 Download - ROCm gfx90a](#)
- [Singularity x86_64 Download - ROCm gfx908](#)
- [Singularity x86_64 Download - OneAPI](#)
- [OVA Download](#)

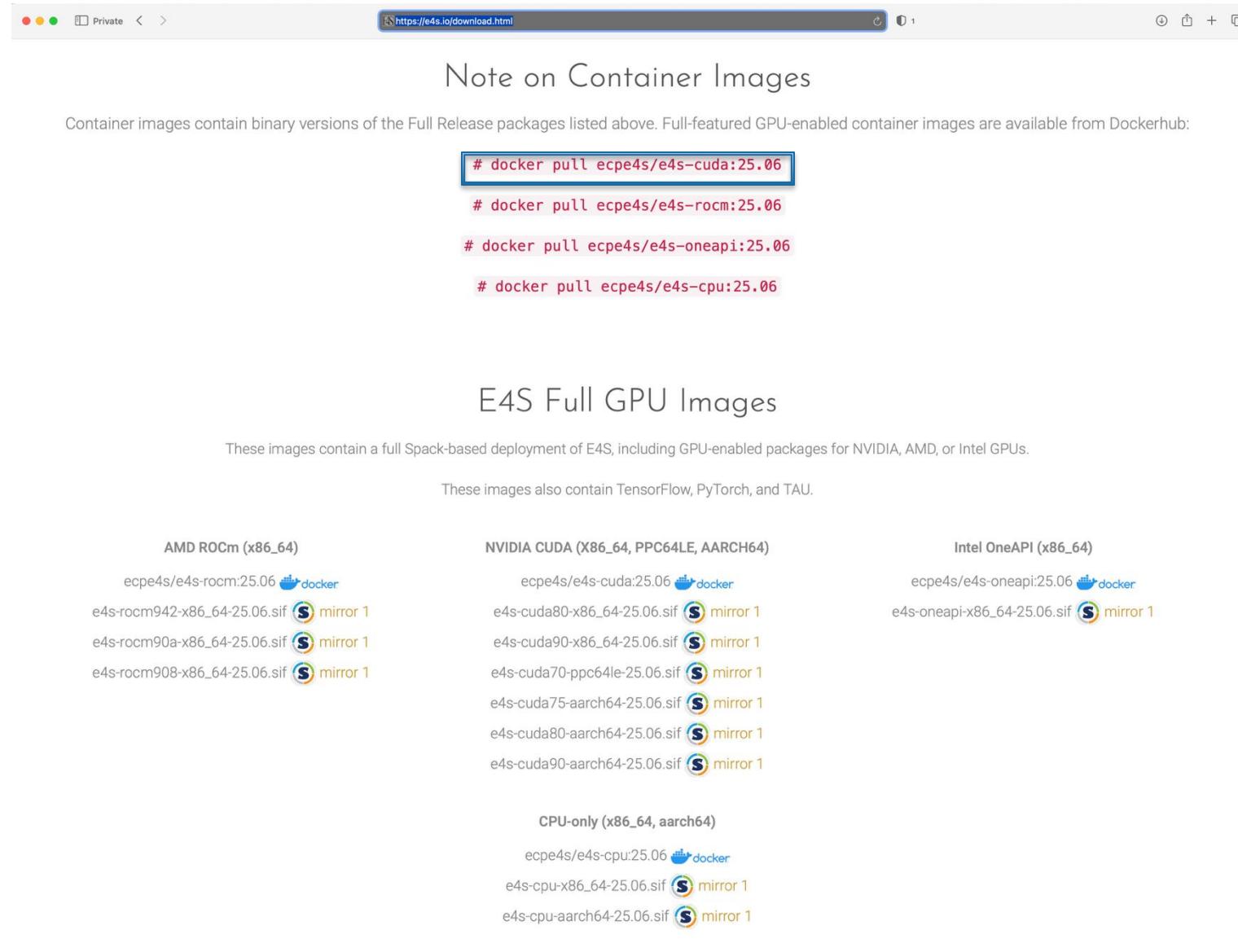
From source with Spack

[Visit the Spack Project](#)

Spack contains packages for all of the products listed in the E4S 25.06 Full Release category (see above Release Notes). General instructions for building software with Spack can be found at the Spack website. Questions concerning building those packages are deferred to the associated package development team.

- Separate full featured Singularity images for 3 GPU architectures
- GPU full featured images for
 - x86_64 (Intel, AMD, NVIDIA)
 - ppc64le (NVIDIA)
 - aarch64 (NVIDIA)
- Full featured images available on Dockerhub
- 130+ products on 3 architectures

Download E4S 25.06 GPU Container Images: AMD, Intel, and NVIDIA



The screenshot shows a web browser window with the URL <https://e4s.io/download.html>. The page title is "Note on Container Images". Below the title, a note states: "Container images contain binary versions of the Full Release packages listed above. Full-featured GPU-enabled container images are available from Dockerhub:". Below this note, there is a code block with four Docker pull commands:

```
# docker pull ecpe4s/e4s-cuda:25.06
# docker pull ecpe4s/e4s-rocm:25.06
# docker pull ecpe4s/e4s-oneapi:25.06
# docker pull ecpe4s/e4s-cpu:25.06
```

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:25.06  	ecpe4s/e4s-cuda:25.06  	ecpe4s/e4s-oneapi:25.06  
e4s-rocm942-x86_64-25.06.sif 	e4s-cuda80-x86_64-25.06.sif 	e4s-oneapi-x86_64-25.06.sif 
e4s-rocm90a-x86_64-25.06.sif 	e4s-cuda90-x86_64-25.06.sif 	
e4s-rocm908-x86_64-25.06.sif 	e4s-cuda70-ppc64le-25.06.sif 	
	e4s-cuda75-aarch64-25.06.sif 	
	e4s-cuda80-aarch64-25.06.sif 	
	e4s-cuda90-aarch64-25.06.sif 	
CPU-only (x86_64, aarch64)		
	ecpe4s/e4s-cpu:25.06  	
	e4s-cpu-x86_64-25.06.sif 	
	e4s-cpu-aarch64-25.06.sif 	

<https://e4s.io>

E4S base container images allow users to customize their containers

GPU Base Images

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:25.06

e4s-base-rocm-25.06.sif mirror 1

NVIDIA Multi-Arch (X86_64, PPC64LE, AARCH64)

ecpe4s/e4s-base-cuda:25.06

e4s-base-cuda-x86_64-25.06.sif mirror 1

e4s-base-cuda-aarch64-25.06.sif mirror 1

e4s-base-cuda-ppc64le-25.06.sif mirror 1

Intel OneAPI (X86_64)

ecpe4s/e4s-base-oneapi:25.06

e4s-base-oneapi-25.06.sif mirror 1

• Intel oneAPI
• AMD ROCm
• NVIDIA CUDA

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 mirror 1

ecpe4s-ubuntu20.04-ppc64le-24.02.sif mirror 1

ecpe4s-ubuntu20.04-aarch64-24.02.sif mirror 1

DOE LLVM E4S Image

This multi-architecture image contains E4S products compiled with DOE LLVM 16 and Flang using Spack

Multi-Arch (X86_64, PPC64LE, AARCH64)

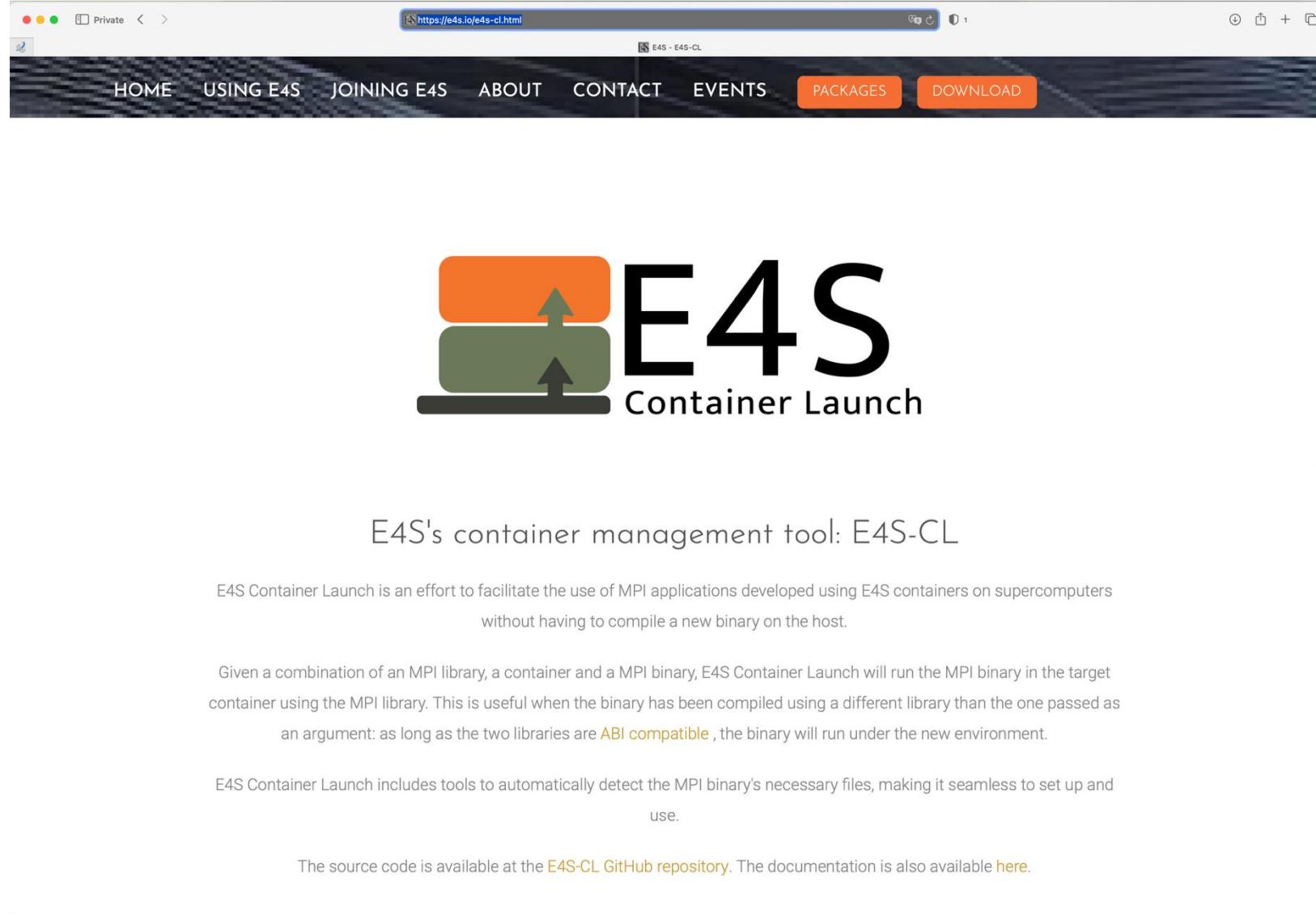
ecpe4s/e4s-doe-llvm:23.05

e4s-doe-llvm-x86_64-23.05.sif mirror 1

e4s-doe-llvm-aarch64-23.05.sif mirror 1

e4s-doe-llvm-ppc64le-23.05.sif mirror 1

E4S Tools: e4s-cl: Container Launch tool for MPI applications



The screenshot shows the homepage of the E4S Container Launch website. At the top, there is a navigation bar with links for HOME, USING E4S, JOINING E4S, ABOUT, CONTACT, EVENTS, PACKAGES (which is highlighted in orange), and DOWNLOAD. Below the navigation bar is the E4S logo, which consists of a stylized orange and green block icon followed by the text "E4S Container Launch". The main heading on the page is "E4S's container management tool: E4S-CL". Below this, there is a paragraph of text explaining the purpose of E4S Container Launch. Further down, there is another paragraph about how it runs MPI binaries in target containers using MPI libraries. The footer contains a link to the GitHub repository and documentation.

E4S Container Launch is an effort to facilitate the use of MPI applications developed using E4S containers on supercomputers without having to compile a new binary on the host.

Given a combination of an MPI library, a container and a MPI binary, E4S Container Launch will run the MPI binary in the target container using the MPI library. This is useful when the binary has been compiled using a different library than the one passed as an argument: as long as the two libraries are **ABI compatible**, the binary will run under the new environment.

E4S Container Launch includes tools to automatically detect the MPI binary's necessary files, making it seamless to set up and use.

The source code is available at the [E4S-CL GitHub repository](#). The documentation is also available [here](#).

- Distribute your MPI application as a binary with an E4S image
- While deploying on a system substitute the embedded containerized MPI in application with the system/vendor MPI
- Use inter-node network interfaces efficiently for near native performance!

e4s-cl: A tool to simplify the launch of MPI jobs in E4S containers

- E4S containers support replacement of MPI libraries using MPICH ABI compatibility layer and Wi4MPI [CEA] for OpenMPI replacement.
- Applications binaries built using E4S can be launched with Singularity using MPI library substitution for efficient inter-node communications.
- e4s-cl is a new tool that simplifies the launch and MPI replacement.
 - e4s-cl init --backend [singularity|shifter|docker] --image <file> --source <startup_cmds.sh>
 - e4s-cl mpirun -np <N> <command>

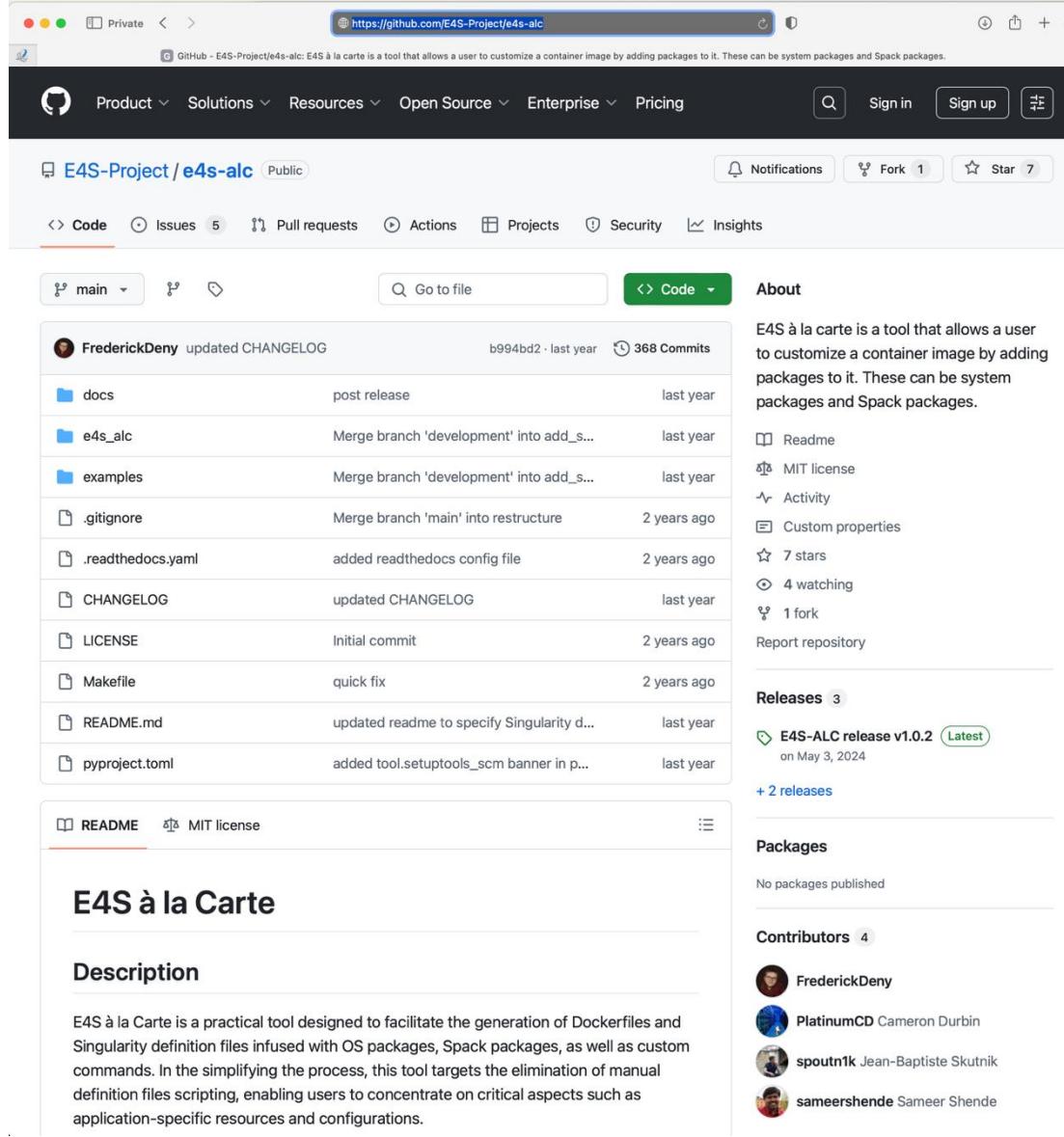
- Usage:

```
% e4s-cl init --backend singularity --image ~/images/e4s-gpu-x86.sif --source ~/source.sh  
% cat ~/source.sh  
  . /spack/share/spack/setup-env.sh  
  spack load trilinos+cuda cuda_arch=90  
% e4s-cl mpirun -np 4 ./a.out
```



<https://github.com/E4S-Project/e4s-cl>

E4S Tools: E4S à la carte or e4s-alc: Customize container images

A screenshot of a GitHub repository page for "E4S-Project/e4s-alc". The repository has 368 commits, 5 issues, and 1 fork. The README and MIT license are visible. The repository description explains E4S à la carte as a tool for customizing container images by adding packages. It also lists releases, packages, and contributors.

GitHub - E4S-Project/e4s-alc: 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.

E4S-Project / e4s-alc Public

Code Issues 5 Pull requests Actions Projects Security Insights

main Go to file Code About

FrederickDeny updated CHANLOG b994bd2 · last year 368 Commits

docs post release last year

e4s_alc Merge branch 'development' into add_s... last year

examples Merge branch 'development' into add_s... last year

.gitignore Merge branch 'main' into restructure 2 years ago

.readthedocs.yaml added readthedocs config file 2 years ago

CHANGELOG updated CHANLOG last year

LICENSE Initial commit 2 years ago

Makefile quick fix 2 years ago

README.md updated readme to specify Singularity d... last year

pyproject.toml added tool.setupools_scm banner in p... last year

README MIT license

E4S à la Carte

Description

E4S à la Carte is a practical tool designed to facilitate the generation of Dockerfiles and Singularity definition files infused with OS packages, Spack packages, as well as custom commands. In the simplifying the process, this tool targets the elimination of manual definition files scripting, enabling users to concentrate on critical aspects such as application-specific resources and configurations.

Notifications Fork 1 Star 7

Readme MIT license Activity Custom properties 7 stars 4 watching 1 fork Report repository

Releases 3

E4S-ALC release v1.0.2 (Latest) on May 3, 2024 + 2 releases

Packages No packages published

Contributors 4

FrederickDeny PlatinumCD Cameron Durbin spoutn1k Jean-Baptiste Skutnik sameershende Sameer Shende

- Add new system packages
- Add new Spack packages
- Add new tarballs
- Customize the container image
- Start with a base image
- Add packages
- Create a new container image!

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.
- <https://spack.io>

Spack is a flexible package manager for HPC

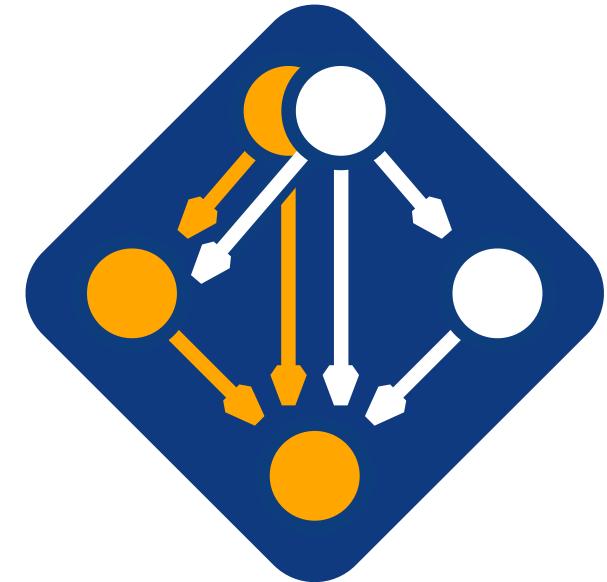
- How to install Spack (works out of the box):

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

- How to install a package:

```
$ spack install tau
```

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



Visit spack.io



github.com/spack/spack



Spack provides the *spec* syntax to describe custom configurations

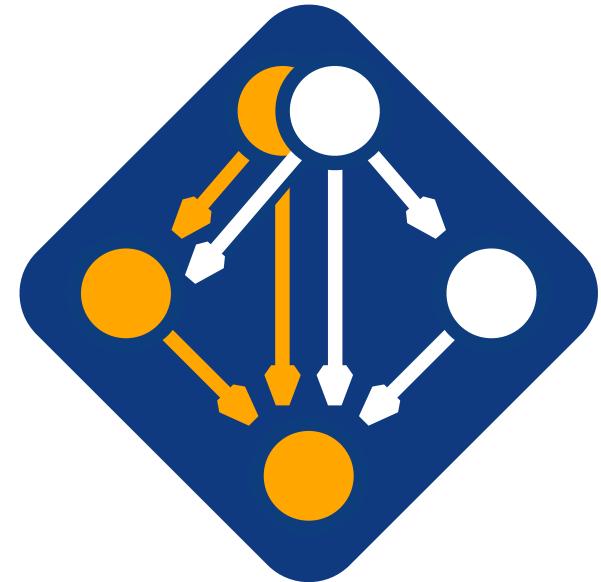
```
$ git clone https://github.com/spack/spack
$ . spack/share/spack/setup-env.sh
$ spack compiler find                                # set up compilers
$ spack external find                               # set up external packages
```

\$ spack install tau	unconstrained
\$ spack install tau@2.34.1	@ custom version
\$ spack install tau@2.34.1 %gcc@12.4.0	% custom compiler
\$ spack install tau@2.34.1 %gcc@12.4.0 +rocm	+/- build option
\$ spack install tau@2.34.1 %gcc@12.4.0 +mpi ^mvapich2@4.0	^ 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

The Spack community is growing rapidly

- **Spack simplifies HPC software for:**
 - Users
 - Developers
 - Cluster installations
 - The largest HPC facilities
- **Spack is central to HPSF's software strategy**
 - Enable software reuse for developers and users
 - Allow the facilities to consume the entire E4S
- **The roadmap is packed with new features:**
 - Building the software distribution
 - Better workflows for building containers
 - Stacks for facilities
 - Chains for rapid dev workflow
 - Optimized binaries
 - Better dependency resolution



Visit spack.io
hpsf.io



E4S Tools: e4s-chain-spack.sh to customize software stack

```
sameer@mothra:~$ ls ~/images  
e4s-cuda80-x86_64-25.06.sif  
sameer@mothra:~$ singularity run --nv ~/images/e4s-cuda80-x86_64-25.06.sif  
Singularity> /etc/e4s/e4s-chain-spack.sh ~/spack  
Cloning into '/home/sameer/spack'...  
remote: Enumerating objects: 686113, done.  
remote: Counting objects: 100% (976/976), done.  
remote: Compressing objects: 100% (463/463), done.  
remote: Total 686113 (delta 772), reused 518 (delta 510), pack-reused 685137 (from 3)  
Receiving objects: 100% (686113/686113), 230.82 MiB | 37.06 MiB/s, done.  
Resolving deltas: 100% (326280/326280), done.
```

Configuration SUCCESS!

```
Downstream: /home/sameer/spack  
Upstream: /spack
```

To use the downstream Spack instance, run the following command in your shell:

```
. /home/sameer/spack/share/spack/setup-env.sh  
-----
```

```
Singularity> . /home/sameer/spack/share/spack/setup-env.sh
```

```
Singularity> spack find valgrind  
==> Error: No package matches the query: valgrind  
Singularity> spack install valgrind  
[+] /usr/local/mpich/install/mpich (external mpich-4.2.3-47excoypwhfmhx57rfs6reouvninugcf)  
[+] /usr (external glibc-2.35-a7drdl4tlx4bu3mzhor75pskvd3pdot6)  
[+] /spack/opt/spack/linux-ubuntu22.04-x86_64_v3/gcc-11.4.0/gcc-runtime-11.4.0-f63c77kavzjtpmnhud2oyfaxagwjzla  
[+] /spack/opt/spack/linux-ubuntu22.04-x86_64_v3/gcc-11.4.0/boost-1.86.0-6qkv24gbidwxhllgah6jrkytm5ev2cng5  
[+] /spack/opt/spack/linux-ubuntu22.04-x86_64_v3/gcc-11.4.0/gmake-4.4.1-qp5blvcyuzghqsrp2ew6gq2nlos34b2  
==> Installing valgrind-3.23.0-feuxx36lsqp7quzmhmo4opbiadwpsars [6/6]  
==> No binary for valgrind-3.23.0-feuxx36lsqp7quzmhmo4opbiadwpsars found: installing from source  
==> Fetching https://mirror.spack.io/_source-cache/archive/c5/c5c34a3380457b9b75606df890102e7df2c702b9420c2ebef9540f8b5d56264d.tar.bz2  
==> Ran patch() for valgrind  
==> valgrind: Executing phase: 'autoreconf'  
==> valgrind: Executing phase: 'configure'  
==> valgrind: Executing phase: 'build'  
==> valgrind: Executing phase: 'install'  
==> valgrind: Successfully installed valgrind-3.23.0-feuxx36lsqp7quzmhmo4opbiadwpsars  
  Stage: 3.78s. Autoreconf: 0.01s. Configure: 48.56s. Build: 37.71s. Install: 2.97s. Post-install: 0.60s. Total: 1m 33.97s  
[+] /home/sameer/spack/opt/spack/linux-ubuntu22.04-x86_64_v3/gcc-11.4.0/valgrind-3.23.0-feuxx36lsqp7quzmhmo4opbiadwpsars  
Singularity> spack load valgrind  
Singularity> which valgrind  
/home/sameer/spack/opt/spack/linux-ubuntu22.04-x86_64_v3/gcc-11.4.0/valgrind-3.23.0-feuxx36lsqp7quzmhmo4opbiadwpsars/bin/valgrind
```

Specify location of downstream
Spack installation directory

Source downstream Spack's
setup-env.sh

Install a new Spack package
in downstream Spack directory

Load new package (valgrind)
using spack load



E4S Tools: e4s-chain-spack.sh to customize software stack

```
Singularity> which valgrind
/home/sameer/spack/opt/spack/linux-ubuntu22.04-x86_64_v3/gcc-11.4.0/valgrind-3.23.0-feuxx36lsqp7quzmhmo4opbiadwpsars/bin/valgrind
Singularity> valgrind --help
usage: valgrind [options] prog-and-args

tool-selection option, with default in [ ]:
--tool=<name>          use the Valgrind tool named <name> [memcheck]
                        available tools are:
                        memcheck cachegrind callgrind helgrind drd
                        massif dhat lackey none exp-bbv

basic user options for all Valgrind tools, with defaults in [ ]:
-h --help                show this message
--help-debug              show this message, plus debugging options
--help-dyn-options        show the dynamically changeable options
--version                 show version
-q --quiet                run silently; only print error msgs
-v --verbose               be more verbose -- show misc extra info
--trace-children=no|yes   Valgrind-ise child processes (follow execve)? [no]
--trace-children-skip=patt1,patt2,...  specifies a list of executables
                                         that --trace-children=yes should not trace into
--trace-children-skip-by-arg=patt1,patt2,... same as --trace-children-skip=
                                         but check the argv[] entries for children, rather
                                         than the exe name, to make a follow/no-follow decision
--child-silent-after-fork=no|yes omit child output between fork & exec? [no]
--vgdb=no|yes|full        activate gdbserver? [yes]
                          full is slower but provides precise watchpoint/step
--vgdb-error=<number>     invoke gdbserver after <number> errors [99999999]
                          to get started quickly, use --vgdb-error=0
                          and follow the on-screen directions
--vgdb-stop-at=event1,event2,... invoke gdbserver for given events [none]
                                         where event is one of:
                                         startup exit abexit valgrindabexit all none
--track-fds=no|yes|all    track open file descriptors? [no]
                                         all includes reporting stdin, stdout and stderr
--time-stamp=no|yes       add timestamps to log messages? [no]
--log-fd=<number>         log messages to file descriptor [2=stderr]
--log-file=<file>         log messages to <file>
--log-socket=ipaddr:port  log messages to socket ipaddr:port
--enable-debuginfod=no|yes query debuginfod servers for missing
                             debuginfo [yes]

user options for Valgrind tools that report errors:
--xml=yes                emit error output in XML (some tools only)
--xml-fd=<number>         XML output to file descriptor
--xml-file=<file>         XML output to <file>
--xml-socket=ipaddr:port  XML output to socket ipaddr:port
--xml-user-comment=STR    copy STR verbatim into XML output
--demangle=no|yes          automatically demangle C++ names? [yes]
--num-callers=<number>    show <number> callers in stack traces [12]
--error-limit=no|yes      stop showing new errors if too many? [yes]
--exit-on-first-error=no|yes exit code on the first error found? [no]
--error-exitcode=<number>  exit code to return if errors found [0=disable]
--error-markers=<begin>,<end> add lines with begin/end markers before/after
                                         each error output in plain text mode [none]
--show-error-list=no|yes|all show detected errors list and
                                         suppression counts at exit [no].
                                         all means to also print suppressed errors.
                                         same as --show-error-list=yes
-s
```

Downstream Spack's package
is loaded in your environment

e4s-chain-spack.sh helps
customize the software stack
using upstream /spack
(read-only in the container) for
package dependencies while
installing a new package in the
downstream Spack in your
writable home directory.



E4S 25.06 image for NVIDIA H100 GPU on x86_64

```
$ singularity run --nv e4s-cuda90-x86_64-25.06.sif
Singularity> ls /opt/demo/e4s-cloud-examples/
bionemo      cuda      julia-cuda    lammps       nalu        openfoam      pytorch      superlu-dist-cpu  vllm
clean-all.sh  fetch-all.sh  julia-mpi   machine-learning  nemo       osu-benchmarks  pytorch-gpu  tau          xyce
CoMD         horovod     jupyter-notebook  matmult     nemo-speech_to_text  petsc-cpu   pytorch-image-classifier tensorflow
containers   jax        laghos       mpi-procname  neuronx    petsc-cuda     qe          visit
Singularity> ls /opt/demo/e4s-cloud-examples/machine-learning/
clean.sh  gemini  openai  perplexity  pytorch  tensorflow
Singularity> ls /opt/demo/e4s-cloud-examples/vllm
gradio_openai_chatbot_webserver.py  llama2_template.jinja  README.md  run.sh  run-smaller.sh
Singularity> python
Python 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import nemo
>>> import bionemo
>>> import torch
>>> import openai
>>> import google.generativeai
>>> import huggingface_hub
>>> import jax
>>> import pandas
>>> import cv2
>>> import sklearn
>>> import mpi4py
>>> import matplotlib
>>> import seaborn
>>> import plotly
>>> import vllm
>>> vllm.__version__
'0.8.3.dev0+g25f560a62.d20250520'
>>> nemo.__version__
'2.3.0rc0'
>>> tensorflow.__version__
'2.19.0'
>>> torch.__version__
'2.6.0'
>>> torch.cuda.get_arch_list()
['sm_80', 'sm_90', 'sm_120']
>>>
```

E4S 25.06 image for NVIDIA GPUs (x86_64)

Singularity> spack find

```
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 -----
abseil-cpp@20240722.0          expat@2.7.0           libceed@0.12.0        openssl@3.4.1
adiak@0.4.1                     fftw@3.3.10          libdwarf@0.11.0       otf2@3.0.3
adios2@2.10.2                   fftw@3.3.10          libedit@3.1-20240808  papi@7.1.0
amrex@25.03                    fftx@1.2.0           libevent@2.1.12      papi@7.1.0
arborx@1.5                      findutils@4.10.0     libfabric@1.22.0      parmetis@4.0.3
arpack-ng@3.9.1                 flecsi@2.3.2         libfffi@3.4.6         parmetis@4.0.3
asio@1.32.0                     flex@2.6.3           libiberty@2.41       parsec@3.0.2209
autoconf@2.72                   flux-core@0.67.0    libiconv@1.17        pcre2@10.44
automake@1.16.5                 fmt@11.1.4          libidn2@2.3.7        pdt@3.25.2
axom@0.10.1                     fmt@11.1.4          libintl@2.9.0        perl@5.40.0
bc@1.07.1                       gasnet@2024.5.0     libmd@1.1.0          perl@5.40.0
berkeley-db@18.1.40             gcc-runtime@11.4.0  libmonitor@2023.03.15 perl-data-dumper@2.173
binutils@2.43.1                 gdbm@1.23           libpciaccess@0.17   petsc@3.22.4
bison@3.8.2                      gettext@0.23.1     libpng@1.6.39         petsc@3.22.4
blaspp@2024.10.26              ginkgo@1.9.0        libsigsev@2.14      pigz@2.8
blt@0.7.0                        git@2.48.1          libsodium@1.0.20     pkgconf@2.3.0
blt@0.7.0                        glibc@2.35          libtool@2.4.7        protobuf@3.28.2
boost@1.86.0                     gmake@4.4.1         libunistring@1.2     protobuf@3.29.3
boost@1.86.0                     gmp@6.3.0           libunwind@1.8.1      py-calver@2022.6.26
boost@1.86.0                     gperftools@2.15    libxcrypt@4.4.38    py-certifi@2023.7.22
boost@1.86.0                     gromacs@2024.4     libxml2@2.13.5      py-cffi@1.17.1
boost@1.86.0                     hdf5@1.8.23        libyaml@0.1.7        py-charset-normalizer@3.3.0
bricks@2023.08.25               hdf5@1.14.5        libyaml@0.2.5        py-cython@3.0.11
butterflypack@3.2.0              heffte@2.4.1        lizard@2.0          py-editables@0.5
bzip2@1.0.8                      hpctoolkit@2024.01.1 llvm@19.1.7        py-flit-core@3.10.1
c-blosc2@2.15.1                 hpcviewer@2025.01   lua@5.3.6          py-fypp@3.1
ca-certificates-mozilla@2025-02-25 hpx@1.10.0          lua@5.4.6          py-hatchling@1.25.0
cabana@0.7.0                     hwloc@2.11.1       lua-luaposix@36.1   py-idna@3.4
caliper@2.12.1                  hwloc@2.11.1       lz4@1.10.0         py-meson-python@0.16.0
camp@2024.07.0                  hypre@2.32.0        lzo@2.10           py-numpy@2.2.4
camp@2024.07.0                  hypre@2.32.0        m4@1.4.19          py-packaging@24.2
chai@2024.07.0                  icu4c@74.2         py-pathspec@0.11.1  py-pip@24.3.1
chapel@2.4.0
```

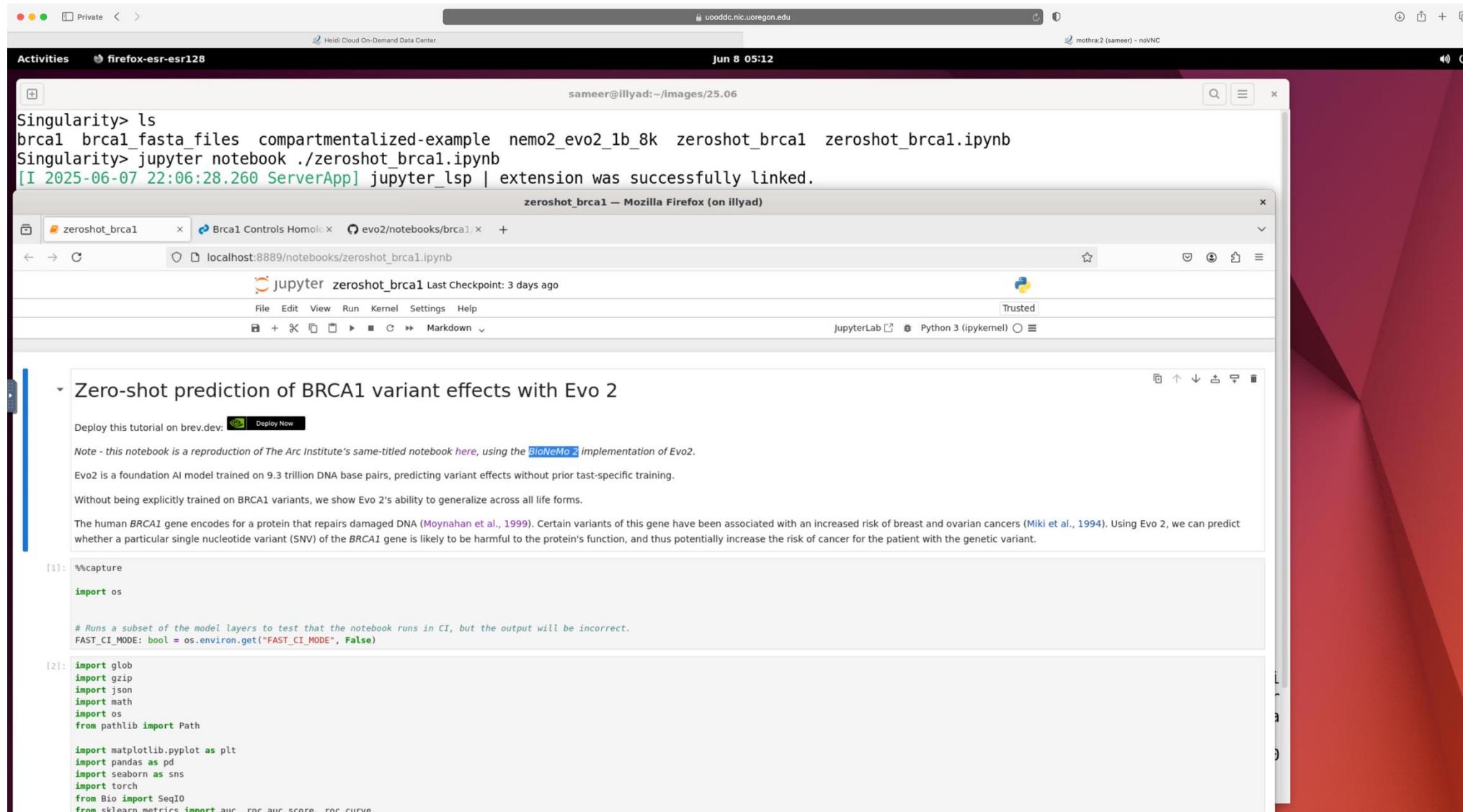


E4S 25.06 image for NVIDIA GPUs (x86_64)

```
Singularity> spack find +cuda
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 --
adios2@2.10.2   camp@2024.07.0   gromacs@2024.4      kokkos@4.6.01      nvcomp@2.2.0      strumpack@8.0.0      upcxx@2023.9.0
amrex@25.03    chai@2024.07.0   heffte@2.4.1       kokkos-kernels@4.6.01  papi@7.1.0      sundials@7.2.1      vtk-m@2.2.0
arbork@1.5     chapel@2.4.0    hpctoolkit@2024.01.1  lammps@20240829.1  parsec@3.0.2209  superlu-dist@9.1.0  zfp@1.0.0
axom@0.10.1    cp2k@2025.1    hpx@1.10.0       legion@24.12.0    petsc@3.22.4     tasmanian@8.1
blaspp@2024.10.26 cusz@0.14.0   hwloc@2.11.1     libceed@0.12.0    petsc@3.22.4     tau@2.34.1
bricks@2023.08.25 fftx@1.2.0    hypre@2.32.0     magma@2.9.0     raja@2024.07.0   trilinos@16.1.0
cabana@0.7.0    flecsi@2.3.2   kokkos@4.5.01    mfem@4.7.0      raja@2024.07.0   umpire@2024.07.0
caliper@2.12.1   flux-core@0.67.0  kokkos@4.6.01    mgard@2023-12-09 slate@2024.10.29  umpire@2024.07.0
camp@2024.07.0   ginkgo@1.9.0   kokkos@4.6.01    slepc@3.22.2    slate@2024.10.29  umpire@2024.07.0
==> 57 installed packages
Singularity> nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2025 NVIDIA Corporation
Built on Fri_Feb_21_20:23:50_PST_2025
Cuda compilation tools, release 12.8, V12.8.93
Build cuda_12.8.r12.8/compiler.35583870_0
Singularity> which huggingface-cli
/usr/local/bin/huggingface-cli
Singularity> which firefox
/usr/bin/firefox
Singularity> which codium
/usr/bin/codium
Singularity> which jupyter
/usr/local/bin/jupyter
Singularity> nvidia-smi
Sat Jun  7 22:02:16 2025
+-----+
| NVIDIA-SMI 570.124.06      Driver Version: 570.124.06      CUDA Version: 12.8 |
+-----+
| GPU  Name                  Persistence-M | Bus-Id        Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp     Perf          Pwr:Usage/Cap |                 Memory-Usage | GPU-Util  Compute M. |
|                                         |                MIG M.       |                      |
+-----+
| 0  NVIDIA H100 PCIe        On           00000000:E1:00.0 Off  |                    0 |
+-----+
```



NVIDIA® BioNeMo™ Framework on E4S 25.06 CUDA x86_64



E4S 25.06 image for CUDA and x86_64 with VSCodium IDE

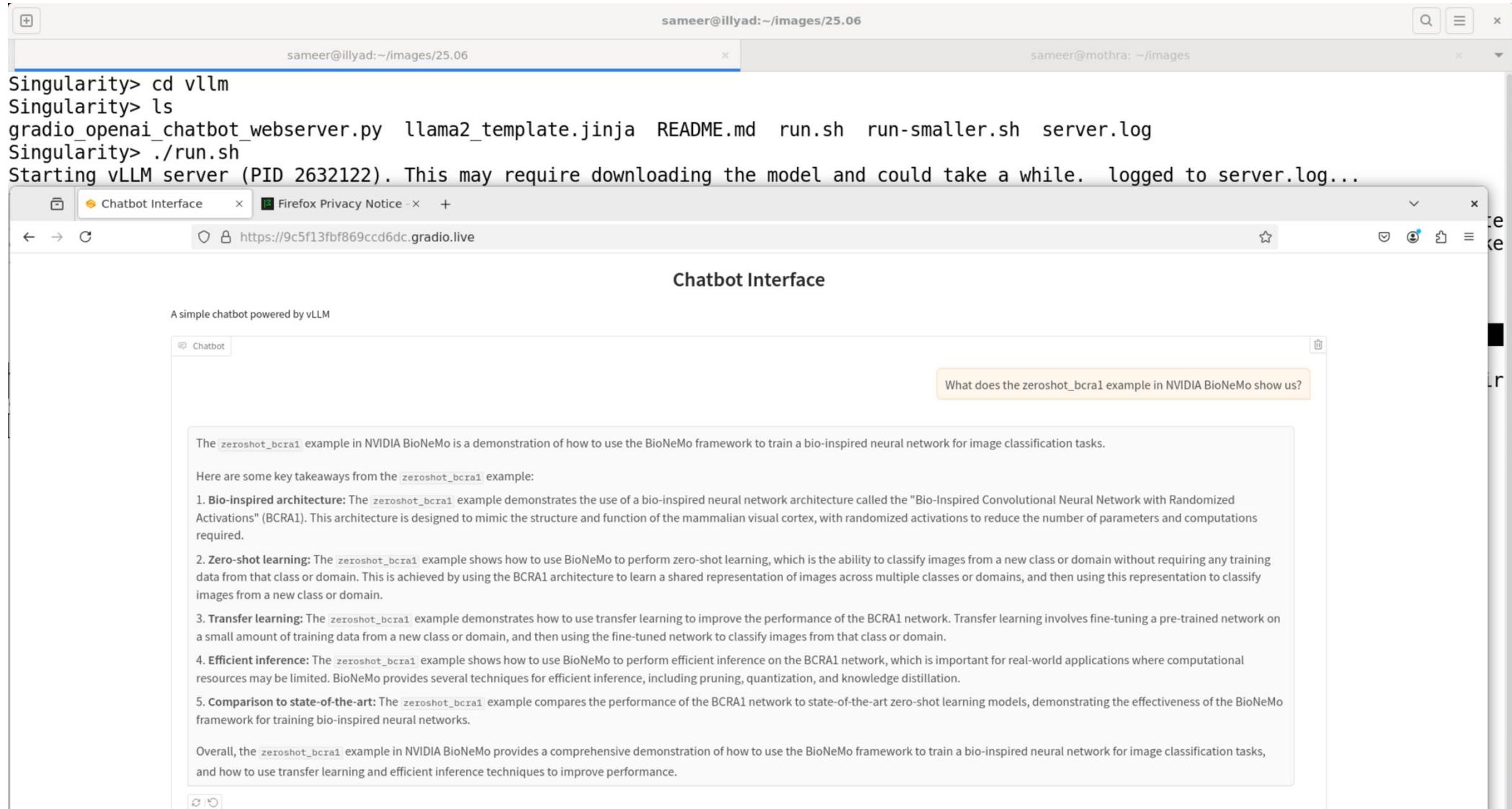
```
sameer@illyad:~/images/25.06
Singularity> codium zeroshot brca1.ipynb
zeroshot_brca1.ipynb - bionemo - VSCodium (on illyad)
File Edit Selection View Go Run Terminal Help
EXPLORER ... zeroShot_brca1.ipynb M ...
OPEN EDITORS zeroShot_brca1.ipynb M zeroShot_brca1.ipynb > m Zero-shot prediction of BRCA1 variant effects with Evo 2
BIONEMO + Code + Markdown ...
.ipynb_checkpoints
> brca1
> brca1_fasta_files
> compartmentalized-example
> nemo2_evo2_1b_8k
> zeroshot_brca1
zeroShot_brca1.ipynb M
plot_strip_with_means(brca1_df, x_col="evo2_delta_score", class_col="class")
Select Kernel Python
Distribution of Delta Likelihood Scores
Comparing Evo 2 likelihood scores for different BRCA1 SNV classes
BRCA1 SNV Class
FUNC/INT
LOF
Delta Likelihood Score, Evo 2
-0.004 -0.003 -0.002 -0.001 -0.0001 0.001 0.002
Cell 1 of 44
```

- NVIDIA H100 (cuda90) GPU on x86_64
- Jupyter Notebook in VSCodium IDE
- Running NVIDIA® BioNeMo™ Framework for biopharma workflows

We can also calculate the area under the receiver operating characteristic curve (AUROC) of this zero-shot prediction method. Note that the results are nearly random unless you are on one of the following configurations:

- `--fp8` on an fp8 enabled GPU with either the 1b or 7b models. The 40b likely works as well.
- the 7b model uniquely seems to work well without `--fp8` so if you are on an older device, the 7b model should produce robust results. Change the `MODEL_SIZE` earlier in this tutorial and rerun for good results in that case.

Creating a Chatbot using Vllm using E4S 25.06 image for x86_64



```
sameer@illyad:~/images/25.06
sameer@illyad:~/images/25.06
sameer@mothra: ~/images

Singularity> cd vllm
Singularity> ls
gradio_openai_chatbot_webserver.py  llama2_template.jinja  README.md  run.sh  run-smaller.sh  server.log
Singularity> ./run.sh
Starting vLLM server (PID 2632122). This may require downloading the model and could take a while. logged to server.log...
Chatbot Interface
https://9c5f13fbf869cccd6dc.gradio.live

Chatbot Interface

A simple chatbot powered by vLLM

Chatbot

What does the zeroshot_bcra1 example in NVIDIA BioNeMo show us?

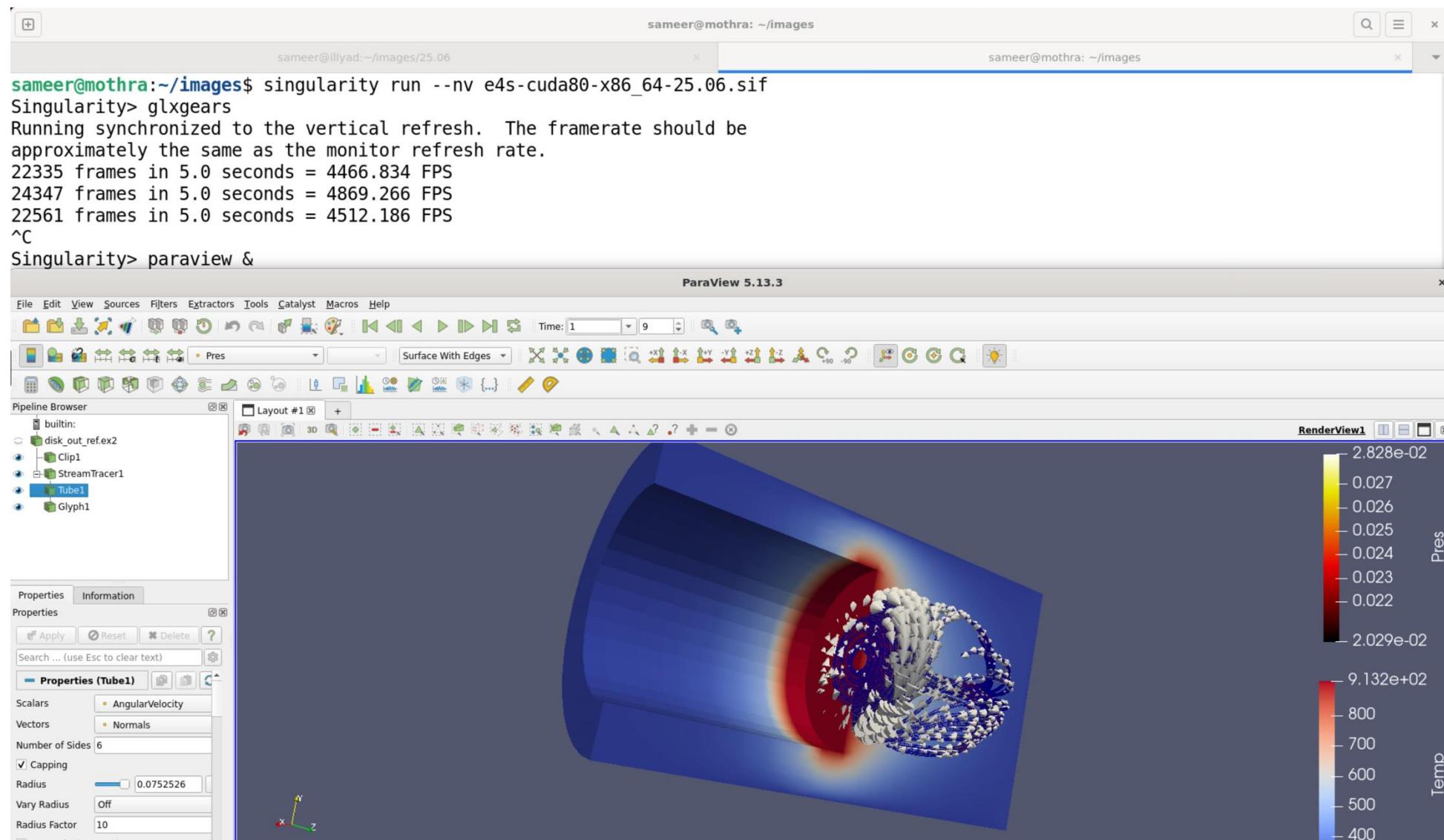
The zeroshot_bcra1 example in NVIDIA BioNeMo is a demonstration of how to use the BioNeMo framework to train a bio-inspired neural network for image classification tasks.

Here are some key takeaways from the zeroshot_bcra1 example:
1. Bio-inspired architecture: The zeroshot_bcra1 example demonstrates the use of a bio-inspired neural network architecture called the "Bio-Inspired Convolutional Neural Network with Randomized Activations" (BCRA1). This architecture is designed to mimic the structure and function of the mammalian visual cortex, with randomized activations to reduce the number of parameters and computations required.
2. Zero-shot learning: The zeroshot_bcra1 example shows how to use BioNeMo to perform zero-shot learning, which is the ability to classify images from a new class or domain without requiring any training data from that class or domain. This is achieved by using the BCRA1 architecture to learn a shared representation of images across multiple classes or domains, and then using this representation to classify images from a new class or domain.
3. Transfer learning: The zeroshot_bcra1 example demonstrates how to use transfer learning to improve the performance of the BCRA1 network. Transfer learning involves fine-tuning a pre-trained network on a small amount of training data from a new class or domain, and then using the fine-tuned network to classify images from that class or domain.
4. Efficient inference: The zeroshot_bcra1 example shows how to use BioNeMo to perform efficient inference on the BCRA1 network, which is important for real-world applications where computational resources may be limited. BioNeMo provides several techniques for efficient inference, including pruning, quantization, and knowledge distillation.
5. Comparison to state-of-the-art: The zeroshot_bcra1 example compares the performance of the BCRA1 network to state-of-the-art zero-shot learning models, demonstrating the effectiveness of the BioNeMo framework for training bio-inspired neural networks.

Overall, the zeroshot_bcra1 example in NVIDIA BioNeMo provides a comprehensive demonstration of how to use the BioNeMo framework to train a bio-inspired neural network for image classification tasks, and how to use transfer learning and efficient inference techniques to improve performance.
```

- NVIDIA H100 (cuda90) GPU on x86_64
- Vllm chatbot running after huggingface-cli login
- Using local H100 GPU

GPU accelerated 3D graphics using E4S 25.06 image for x86_64



- Rendering on an NVIDIA A100 (cuda80) GPU on x86_64
- ParaView
- Using Adaptive Computing's Heidi/ODDC remote desktop

Frank@UO: A hardware platform for CI for performance tools

<https://oaciss.uoregon.edu/frank>



Why is Continuous Integration (CI) infrastructure critical?

- Faster development and delivery
 - Integrate code changes frequently
 - Receive immediate feedback on their work
 - Deploy code to production faster
- Improved Code Quality
 - Automated testing
 - Consistent builds
 - Early bug detection
- Continuous improvement
- Risk reduction

What hardware do our performance tools projects need for CI testing?

- GPUs from multiple vendors: NVIDIA, AMD, Intel
 - Different GPU architectures:
 - NVIDIA GH200, H100, A100, V100, ...
 - AMD MI300A, MI210, MI100, MI50, ...
 - Intel Data Center Max GPU 1100 series (PVC), DG1, A770, B580, ...
- Operating Systems and Programming Environments
 - HPE Cray Programming Environment
 - RHEL, Debian, Ubuntu, SLES, ...
 - GNU, AMD, NVIDIA, Intel, LLVM compilers with MVAPICH, Cray-mpich, Intel MPI, ...
- Network interconnects
 - NDR, HDR, EDR Infiniband
 - 100Gbps Ethernet
 - Two identical nodes for multi-node testing

Private < > https://oaciss.uoregon.edu/frank Log in

Category Discussion Read View source View history Search OACISS Systems Wiki ? Help

Category:Servers

This is a list of all OACISS servers in the Frank cluster at UO. You may also select a section header to view the Wiki-generated category index for systems of that type.



Some relevant pages:

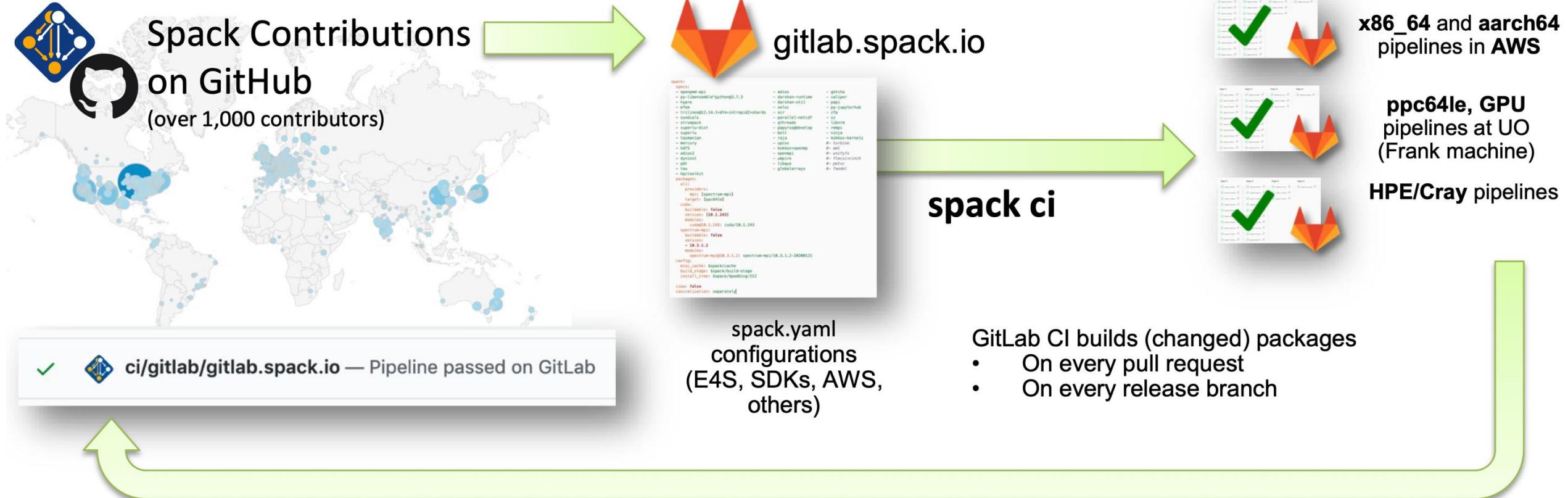
- [Introduction](#) and about <- click me
- The [NetworkInfrastructure](#) page describes the host naming (dns) conventions, as well as documenting the physical setup and connections within the OACISS racks in the machine room. All OACISS systems automatically search .nic.uoregon.edu for DNS, so only the short hostname is needed for ssh internally.
- The [Service:storage](#) describes available storage for users of OACISS systems. OACISS currently has a total of just under 350TB of online storage available.
- The new [HowtoMPI](#) page describes various tested-working MPI setups and the steps

Click on the server links to access more information about individual machines. Note that only the three machines designated as login gateways (orthus, sphinx, cerberus) are accessible by machines outside of nic.uoregon.edu.

IMPORTANT: Cerberus is in the process of being decommissioned and replaced with a new and much more capable system, Sphinx. Sphinx is the new secondary login node.

Nodes in Computing Center datacenter						
Name	Description	OS	Model	Processors	Local Network	Physical location
Compute: Orthus	Primary login gateway	RHEL-8.8	VM	6 x Cascade Lake	10GbE	
Compute: Odyssey	AMD quad MI300A system	RHEL 8.10	SM AS-4145-GT	4 x MI300a, 24cpu	10GbE	R81.U27
Compute: Pinwheel	2xMI210 Debian 12 system	Debian 12	SM AS-2024US-TRT	2 x Epyc Milan 7413	10GbE	R81.U32
Compute: Headroom	Intel Data Center Max 1100 (Ponte Vecchio)	Ubuntu 22	Supermicro	2 x Xeon 4410T @ 3GHz	10GbE	R82.U8
Compute: Hopper1/2	NVIDIA Grace-Hopper GH200	RHELF 9.3	Quanta S74G-2U	Grace (72c)+GH200 Hopper GPU @ 3.4GHz	10GbE + Connect X-7 200 Gbps (pt2pt w/hopper1/2.eth)	R81.U33-36
Compute: Grace1/2	NVIDIA Grace-Grace	Ubuntu 22.04 LTS	Supermicro ARS-221GL-NR01 (x2)	Grace-Grace (144c) Superchip @ 3.4GHz	10GbE + EDR 100 Gbps (pt2pt w/grace1/2)	R81.U4-8
Compute: Roberta	HPE CPE Epyc Genoa	RHEL 8.10	DL385 gen 11	2 x Epyc 9654 Genoa 96c @ 2.4GHz	10GbE + NDR 400 Gbps (pt2pt w/roberta)	R82.U6
Compute: Gary	HPE CPE Epyc Genoa	RHEL 8.10	DL345 gen 11	Epyc 9124 Genoa 16c @ 3GHz	10GbE + NDR 400 Gbps (pt2pt w/roberta)	R82.U4
Compute: Picard	Atipa Sapphire Rapids + 2 A2000	RHEL 8.10	R283-S91	2 x 32c Xeon 6430 @ 3.4GHz	100GbE	R82.U6
Compute: Athena	4xA100 nvlink box	RHEL 8.10	Gigabyte RS292	2 x Epyc Milan 7763 5	10GbE	R85.U39

Spack relies on cloud CI to ensure that builds continue working



Nightly Trilinos CI using E4S containers on 12 GPU architectures

Screenshot of a GitLab pipeline interface showing a CI job for Trilinos using E4S containers across 12 GPU architectures.

The pipeline has passed, created by Administrator for commit 82afc03c, 10 hours ago, finished 10 hours ago.

For master, there are 13 jobs scheduled, with 24 minutes 25 seconds queued for 1 second.

Group jobs by Stage:

- Prep**: Repo-Sync (Passed)
- 1-NVIDIA GPU**: NVIDIA-A100, NVIDIA-A2000, NVIDIA-GH200, NVIDIA-H100, NVIDIA-RTX5080 (All Passed)
- 2-AMD GPU**: AMD-MI50, AMD-MI100, AMD-MI210, AMD-MI300a (All Passed)
- 3-INTEL GPU**: INTEL-A770, INTEL-DATA-CENTER-MAX-1100, INTEL-DG1 (All Passed)

Nightly Trilinos CI using E4S on MI300A GPU: 449 tests

The screenshot shows a browser window displaying a GitLab CI job log for Trilinos. The URL is gitlab.e4s.io/uu-public/Trilinos/Jobs/#392916. The page title is "Explore". On the right side, there are details about the job: Duration: 11 minutes 55 seconds, Finished: 10 hours ago, Queued: 3 seconds, Timeout: 6h (from job), Runner: #380 (VjFvZwQY) odyssey-docker-amd-mi300-trilinos, Source: Schedule, and Tags: trilinos-amd-mi300a. Below this, there is a section for Job artifacts with Download and Browse buttons. At the bottom, there is a Commit link (82afc03c) and a Pipeline link (#25190, Passed for master).

uu-public / Trilinos / Jobs / #392916

Search visible log output

2583 448/449 Test #1281: Zoltan2_kokkosBlock_MPI_4
..... Passed 0.95 sec
2584 Start 1282: Zoltan2_rcb_C_MPI_4
2585 449/449 Test #1282: Zoltan2_rcb_C_MPI_4
..... Passed 0.92 sec
2586 100% tests passed, 0 tests failed out of 449
2587 Subproject Time Summary:
2588 Ifpack2 = 2.60 sec*proc (1 test)
2589 Tpetra = 887.07 sec*proc (271 tests)
2590 Zoltan2 = 845.30 sec*proc (177 tests)
2591 Total Test time (real) = 537.05 sec
2592 \$ find . -type f -name LastTest.log -exec cp {} \$ARTIFACTS/LastTest.log \
2593 \$ exit \$RC
2594 Uploading artifacts for successful job 00:01
2595 Uploading artifacts...
2596 artifacts: found 4 matching artifact files and directories
2597 WARNING: processPath: artifact path is not a subpath of project directory: /Trilinos/spack-config
ure-args.txt
2598 WARNING: processPath: artifact path is not a subpath of project directory: /Trilinos/spack-build-
01-cmake-out.txt
2599 Uploading artifacts as "archive" to coordinator... 201 Created id=392916 responseStatus=201 Crea
ted token=glcbt-64
2600 Cleaning up project directory and file based variables 00:01
2601 Job succeeded

Duration: 11 minutes 55 seconds
Finished: 10 hours ago
Queued: 3 seconds
Timeout: 6h (from job) [?](#)
Runner: #380 (VjFvZwQY) odyssey-docker-amd-mi300-trilinos
Source: Schedule
Tags: [trilinos-amd-mi300a](#)

Job artifacts [?](#)
These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Download [Browse](#)

Commit 82afc03c [Archive](#)
add container-image recipes to this repo

Pipeline #25190 [Passed](#) for master [Archive](#)

Nightly Trilinos CI using E4S containers on NVIDIA Blackwell GPU

The screenshot shows a GitLab CI job log for Trilinos. The log output is as follows:

```
2697 521/521 Test #1289: Zoltan2_rcb_C_MPI_4
..... Passed 0.92 sec
2698 100% tests passed, 0 tests failed out of 521
2699 Subproject Time Summary:
2700 Amesos2    =  9.23 sec*proc (5 tests)
2701 Belos      = 412.17 sec*proc (59 tests)
2702 Ifpack2     = 286.20 sec*proc (69 tests)
2703 Sacado      =  8.15 sec*proc (2 tests)
2704 Tpetra      = 742.49 sec*proc (258 tests)
2705 Zoltan2     = 602.69 sec*proc (128 tests)
2706 Total Test time (real) = 595.61 sec
2707 $ find . -type f -name LastTest.log -exec cp {} $ARTIFACTS/LastTest.log \;
2708 $ exit $RC
2709 Uploading artifacts for successful job
2710 Uploading artifacts...
2711 artifacts: found 4 matching artifact files and directories
2712 WARNING: processPath: artifact path is not a subpath of project directory: /Trilinos/spack-config
ure-args.txt
2713 WARNING: processPath: artifact path is not a subpath of project directory: /Trilinos/spack-build-
01-cmake-out.txt
2714 Uploading artifacts as "archive" to coordinator... 201 Created id=392910 responseStatus=201 Crea
ted token=64_8FBMSq
2715 Cleaning up project directory and file based variables
2716 Job succeeded
```

Job details:

- Duration: 12 minutes 9 seconds
- Finished: 10 hours ago
- Queued: 1 second
- Timeout: 6h (from job)
- Runner: #400 (VxbUQr48) heimdall-docker-nvidia-rtx5080-trilinos
- Source: Schedule
- Tags: trilinos-nvidia-rtx5080

Job artifacts:

Commit 82afc03c

Pipeline #25190 Passed for master

1-NVIDIA GPU

Related links

Monitoring Spack PR jobs on UO and AWS runners

The screenshot shows a web browser window with the URL stats.e4s.io in the address bar. The page title is "Summary". Below the title, there are three key statistics: "Beginning: 2021-09-22 12:48 AM PDT", "Ending: 2025-07-07 10:01 AM PDT", and "Total Jobs: 9,214,501". A "Navigation" section follows, containing links to various monitoring dashboards. The links are organized into sections separated by double-dash symbols (---).
- Pipeline failures over time
- Jobs per pipeline, overview
- Summary of Pipeline Errors

- UO Frank Node Descriptions

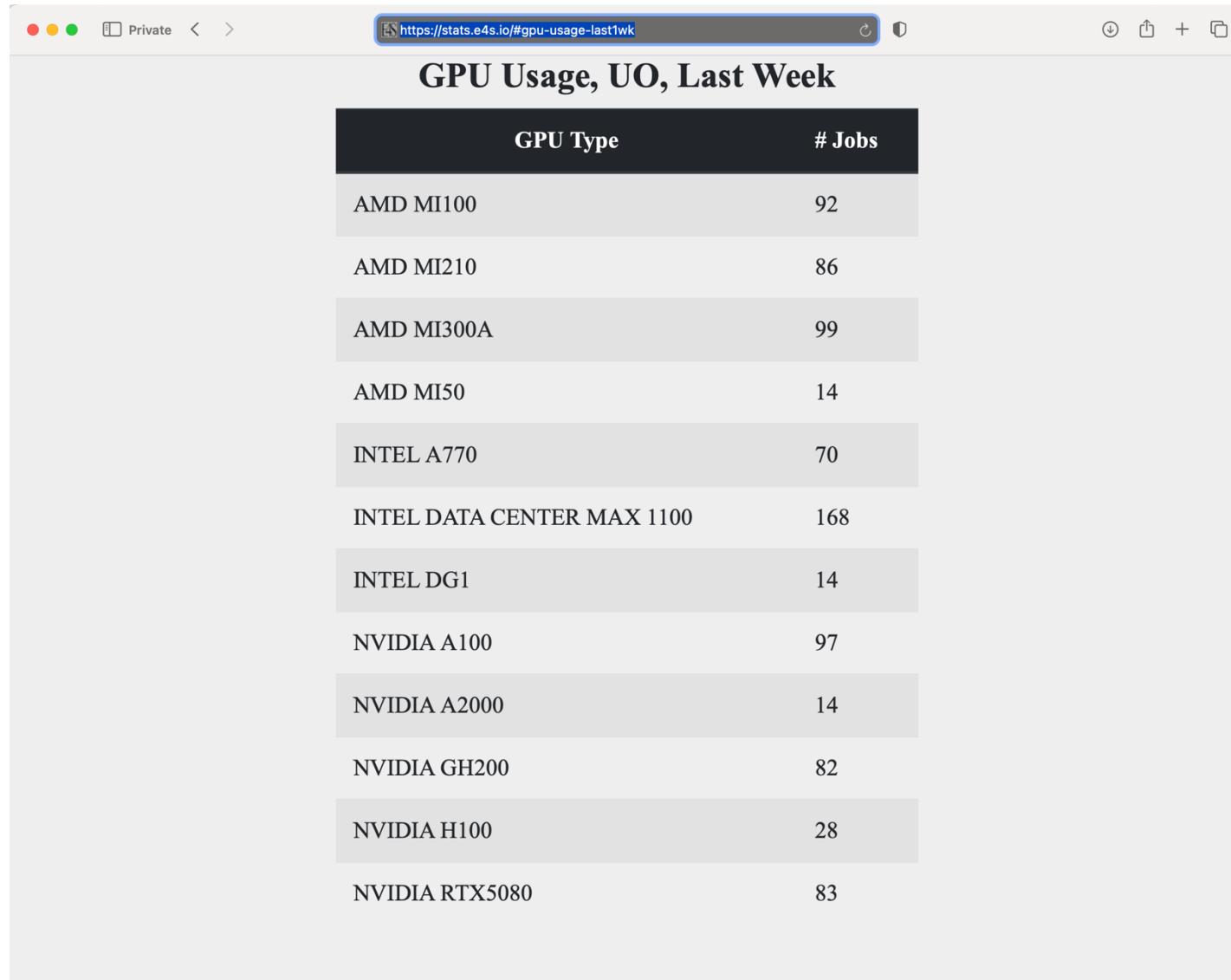
- UO Runners, Last 100 Completed Jobs
- AWS Runners, Last 100 Completed Jobs

- GPU Usage, UO, Last Week
- Job Times, Last 24 Hours
- Job Times, Last Week
- Job Times, Overview, All
- Runner System Failures, by Type, Last 24 Hours
- Runner System Failures, by Runner, Last 24 Hours

<https://stats.e4s.io>



GPU jobs run on Frank@UO last week by GPU architecture



A screenshot of a web browser displaying a table titled "GPU Usage, UO, Last Week". The table lists GPU types and the number of jobs run on them. The browser interface includes standard controls like back, forward, and search at the top.

GPU Type	# Jobs
AMD MI100	92
AMD MI210	86
AMD MI300A	99
AMD MI50	14
INTEL A770	70
INTEL DATA CENTER MAX 1100	168
INTEL DG1	14
NVIDIA A100	97
NVIDIA A2000	14
NVIDIA GH200	82
NVIDIA H100	28
NVIDIA RTX5080	83

<https://stats.e4s.io>

Monitoring Spack PR jobs on UO and AWS runners

Job Times, Last 24 Hours

Job Type	Runtime	% Total Runtime	Average Runtime	N	% UO	% AWS
AWS Packages	5.94 hr	1.4%	8.91 min	40	63%	38%
App CI - E4S - Adios2	2.57 hr	0.6%	0.54 min	288	100%	0%
App CI - E4S - ExaGO	.09 hr	0.0%	1.81 min	3	100%	0%
App CI - E4S - ExaWind	1.34 hr	0.3%	80.26 min	1	100%	0%
App CI - E4S - HPCToolkit	1.8 hr	0.4%	5.15 min	21	100%	0%
App CI - E4S - Kokkos	12.33 hr	2.8%	18.97 min	39	100%	0%
App CI - E4S - TAU	.45 hr	0.1%	1.81 min	15	100%	0%
App CI - E4S - Trilinos	2.5 hr	0.6%	11.56 min	13	100%	0%
Build Systems	.03 hr	0.0%	1.54 min	1	100%	0%
Data and Vis SDK	17.09 hr	3.9%	6.07 min	169	88%	12%
Developer Tools aarch64-linux-gnu	10.34 hr	2.4%	2.26 min	275	87%	13%
Developer Tools x86_64_v3-linux-gnu	19.74 hr	4.5%	3.98 min	298	88%	12%
Developer Tools, Darwin	6.58 hr	1.5%	2.67 min	148	100%	0%
E4S	67.22 hr	15.4%	11.27 min	358	73%	27%

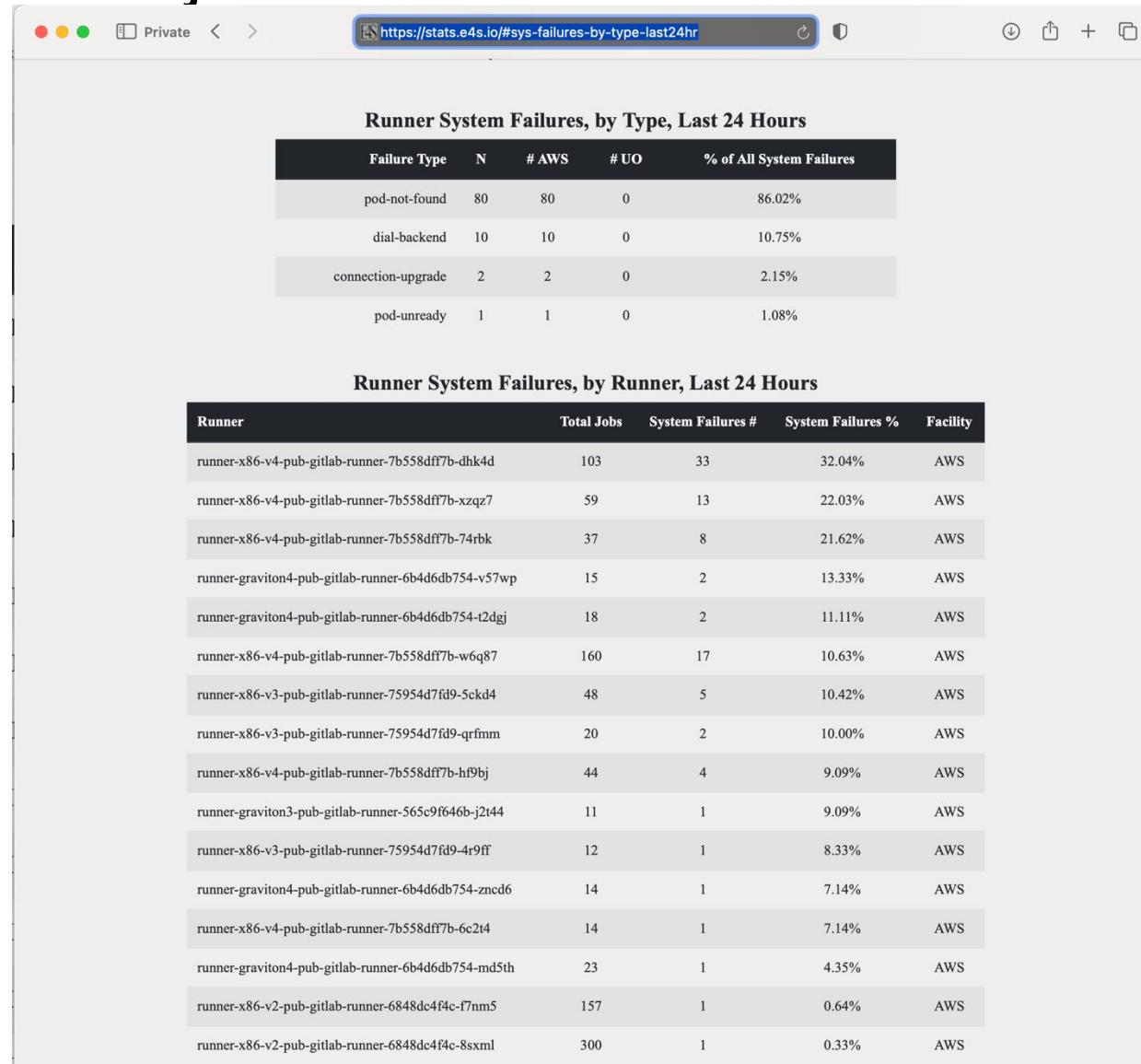
https://stats.e4s.io/#job-times-last24hr

E4S	67.22 hr	15.4%	11.27 min	358	73%	27%
E4S ARM Neoverse V2	9.24 hr	2.1%	4.20 min	132	76%	24%
E4S Cray	4.75 hr	1.1%	4.68 min	61	100%	0%
E4S OneAPI	2.55 hr	0.6%	4.37 min	35	71%	29%
E4S ROCm External	15.08 hr	3.5%	9.53 min	95	69%	31%
High Energy Physics	44.31 hr	10.2%	14.37 min	185	94%	6%
Machine Learning	88.66 hr	20.4%	11.56 min	460	70%	30%
RADIUSS	.76 hr	0.2%	2.67 min	17	71%	29%
Spack CI	8.55 hr	2.0%	2.55 min	201	94%	6%
Spack Tutorial	1.91 hr	0.4%	3.96 min	29	69%	31%
generate	104.65 hr	24.0%	6.85 min	917	63%	37%
no-specs-to-rebuild	5.19 hr	1.2%	0.43 min	718	0%	100%
other	.34 hr	0.1%	0.58 min	35	0%	100%
rebuild-index	1.27 hr	0.3%	0.83 min	92	0%	100%
sign-pkgs	.35 hr	0.1%	5.26 min	4	0%	100%
TOTAL	435.66 hr	--	--	4,650	65%	35%

<https://stats.e4s.io>

Data collected for past 24 hours on 7/7/2025
UO: 65% jobs, AWS: 35%

Monitoring Spack PR job failures on UO and AWS runners



The screenshot shows a web browser window with the URL <https://stats.e4s.io/#sys-failures-by-type-last24hr>. The page displays two tables of data about runner system failures over the last 24 hours.

Runner System Failures, by Type, Last 24 Hours

Failure Type	N	# AWS	# UO	% of All System Failures
pod-not-found	80	80	0	86.02%
dial-backend	10	10	0	10.75%
connection-upgrade	2	2	0	2.15%
pod-unready	1	1	0	1.08%

Runner System Failures, by Runner, Last 24 Hours

Runner	Total Jobs	System Failures #	System Failures %	Facility
runner-x86-v4-pub-gitlab-runner-7b558dff7b-dhk4d	103	33	32.04%	AWS
runner-x86-v4-pub-gitlab-runner-7b558dff7b-xzqz7	59	13	22.03%	AWS
runner-x86-v4-pub-gitlab-runner-7b558dff7b-74rbk	37	8	21.62%	AWS
runner-graviton4-pub-gitlab-runner-6b4d6db754-v57wp	15	2	13.33%	AWS
runner-graviton4-pub-gitlab-runner-6b4d6db754-t2dgi	18	2	11.11%	AWS
runner-x86-v4-pub-gitlab-runner-7b558dff7b-w6q87	160	17	10.63%	AWS
runner-x86-v3-pub-gitlab-runner-75954d7fd9-5ckd4	48	5	10.42%	AWS
runner-x86-v3-pub-gitlab-runner-75954d7fd9-qrfmm	20	2	10.00%	AWS
runner-x86-v4-pub-gitlab-runner-7b558dff7b-hf9bj	44	4	9.09%	AWS
runner-graviton3-pub-gitlab-runner-565c9f646b-j2t44	11	1	9.09%	AWS
runner-x86-v3-pub-gitlab-runner-75954d7fd9-4r9ff	12	1	8.33%	AWS
runner-graviton4-pub-gitlab-runner-6b4d6db754-zncd6	14	1	7.14%	AWS
runner-x86-v4-pub-gitlab-runner-7b558dff7b-6c2t4	14	1	7.14%	AWS
runner-graviton4-pub-gitlab-runner-6b4d6db754-md5th	23	1	4.35%	AWS
runner-x86-v2-pub-gitlab-runner-6848dc4f4c-f7nm5	157	1	0.64%	AWS
runner-x86-v2-pub-gitlab-runner-6848dc4f4c-8sxml	300	1	0.33%	AWS

Job failures in the last 24 hours
Data collected on 7/7/2025

<https://stats.e4s.io>

ParaTools Pro for E4S™:
A cloud image for tool
interoperability

<https://paratoolspro.com>



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, Inc.'s Heidi/ODDC to launch E4S jobs on multiple cloud providers!

E4S on Commercial Cloud Platforms: ParaTools Pro for E4S™

The screenshot shows a web browser displaying the ParaTools website. The main page features a header with navigation links for PRODUCTS, SERVICES, SOLUTIONS, TUTORIALS, TRAINING, NEWS, and MORE. Below the header, a section titled "PARATOOLS PRO FOR E4S™" contains a detailed description of the software stack, mentioning OpenFOAM, LAMMPS, CP2K, Xyce, and Quantum Espresso. A video player below the text shows a simulation of a flame kernel. To the right, there is a sidebar with a "PRO ParaTools" logo and a "RELATED POSTS" section featuring a link to "Launch Your HPC Workflows With HPC E4S". At the bottom, a table lists "Cloud Platform" and "ParaTools Pro for E4S™ Images" for various providers.

Cloud Platform	ParaTools Pro for E4S™ Images
Amazon Web Services	ParaTools Pro for E4S™ on ODDC Node (AWS, x86-64) ParaTools Pro for E4S™ on ODDC Server (AWS, x86-64) ParaTools Pro for E4S™ on ODDC Node (AWS, arm64) ParaTools Pro for E4S™ on ODDC Server (AWS, arm64)
Google Cloud Platform	ParaTools Pro for E4S™ on ODDC (GCP, x86-64)
Microsoft Azure	ParaTools Pro for E4S™ on ODDC (Azure, x86-64)
Oracle Cloud Infrastructure	ParaTools Pro for E4S™ on ODDC (OCI, x86-64)

* Acknowledgment:
Supported by
DOE SBIR Phase I and II
DE-SC0022502



Office of
Science



<https://paratoolspro.com> and <https://www.energy.gov/technologytransitions/sbirstrr>

- ParaTools Pro for E4S™*
images in vendor marketplaces
support:
 - AWS
 - Azure
 - Google Cloud (GCP)
 - Oracle Cloud Infrastructure (OCI)
- Supports SLURM and Torque for scheduling jobs on multi-node GPU accelerated nodes
- Shared GPU accelerated login node with a VNC based remote desktop
 - Adaptive Computing's Heidi/ODDC
 - AWS PCS and PC (x86, ARM64)
 - Azure Cyclecloud
 - Google GCluster

E4S on Adaptive Computing's Heidi AI/On Demand Data Center (ODDC)

The screenshot shows the Heidi AI website on a Mac OS X browser. The main navigation bar includes 'Private', 'adaptivecomputing.com', and a search icon. The page content starts with the 'Our Mission' section, which states: "Heidi AI's mission is to provide every student with access to their own personal supercomputer, ensuring that all students, regardless of their economic background, have the tools they need to succeed and reach their full potential." Below this is the 'Heidi for Grades K-12 & Higher Education' section, which describes Heidi as a "Cloud-Based Personal AI Supercomputer for Grades K-12 & Higher Education". It highlights that Heidi provides access to cutting-edge technology like HPC and AI, and is designed to be accessible and affordable for educational institutions. The final section shown is 'Cloud-Based Supercomputing for Education: HPC, AI, and STEM Solutions with Heidi', which details the platform's features, including its "How it works" and "Heidi Technology Stack". It also mentions the "ParaTools Pro for E4S™ - Extreme-scale Scientific Software Stack", which includes over 150 preloaded applications for HPC, AI, and STEM.

<https://adaptivecomputing.com>

- ParaTools Pro for E4S™ images in commercial cloud marketplaces launched using Heidi
- Supports Torque for scheduling jobs on multi-node GPU accelerated nodes
- Shared GPU accelerated login node with a VNC based remote desktop

ParaTools Pro for E4S™ on Commercial Clouds: AWS Marketplace

The screenshot shows the AWS Marketplace search results for "ParaTools Pro for E4S". The search bar at the top has "ParaTools Pro for E4S" entered. The results page lists four items, all from ParaTools Inc. and hardened for E4S. The items are:

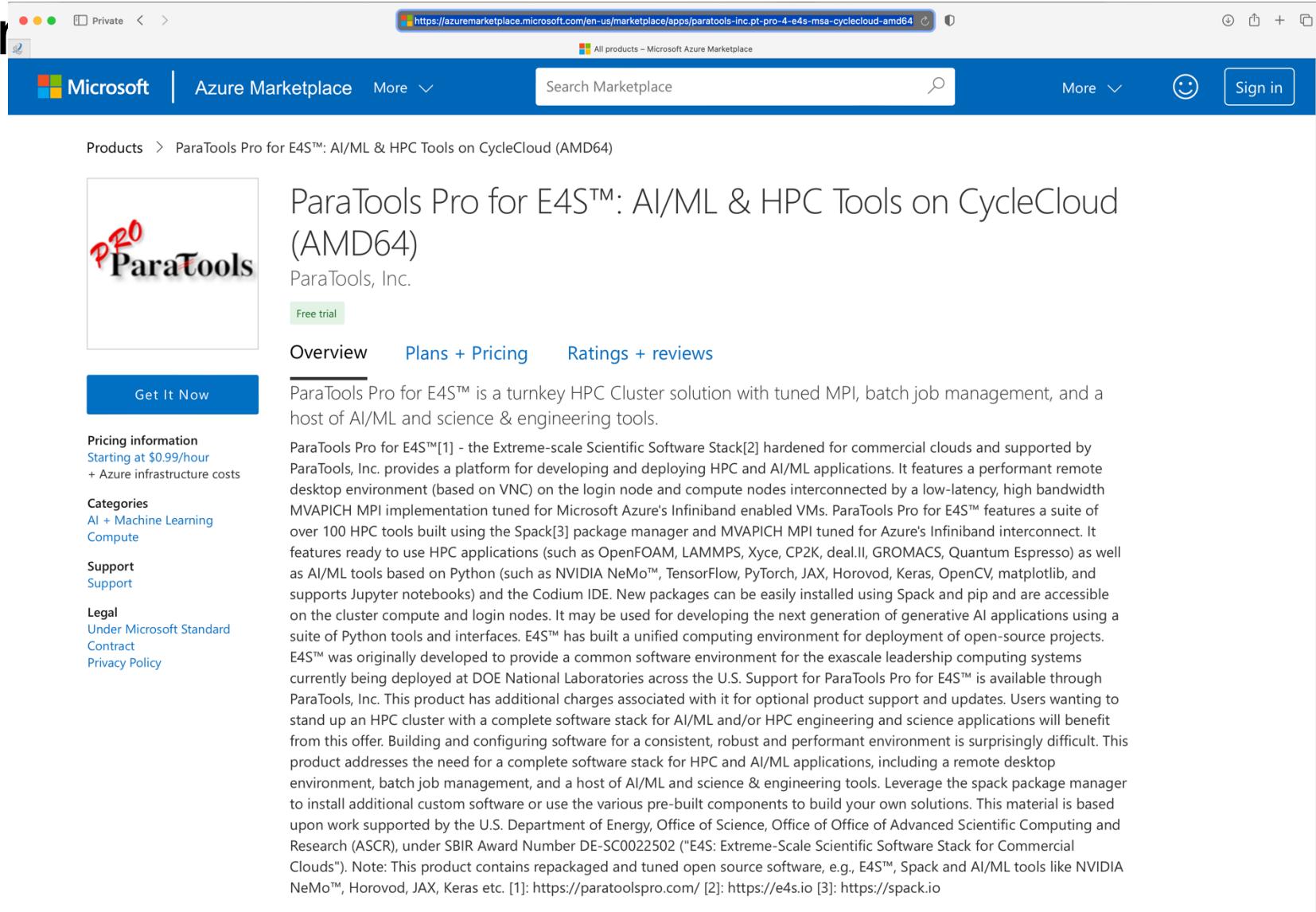
- ParaTools Pro for E4S™: AI/ML & HPC Tools on ParallelCluster (arm64)**: Starting from \$0.99 to \$0.99/hr for software + AWS usage fees.
- ParaTools Pro for E4S™: AI/ML & HPC Tools on ParallelCluster (x86)**: Starting from \$0.99 to \$0.99/hr for software + AWS usage fees.
- ParaTools Pro for E4S™: AI/ML & HPC Tools on AWS PCS (x86)**: Starting from \$0.10 to \$0.99/hr for software + AWS usage fees.
- ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC Node (x86)**: Starting from \$0.99 to \$0.99/hr for software + AWS usage fees.

The sidebar on the left provides refine results options for Categories, Delivery methods, Publisher, Pricing model, Operating system, Free trial, Contract type, Architecture, and Region.

ParaTools Pro for E4S™ on AWS supports

- AWS Trainium and Inferentia custom AI hardware with NeuronX SDK
 - AWS PCS and PC on x86_64 and aarch64 nodes
 - NVIDIA GPUs
 - SLURM (PCS and PC) and Torque (ODDC node/server)
 - Also on AWS Marketplace in GovCloud (US East & West)
 - Elastic Fabric Adapter (EFA)
 - MVAPICH MPI
- [X-ScaleSolutions, LLC and The Ohio State University]

ParaTools Pro for E4S™ on Commercial Clouds: Azure



The screenshot shows the Microsoft Azure Marketplace product page for ParaTools Pro for E4S™. The page title is "ParaTools Pro for E4S™: AI/ML & HPC Tools on CycleCloud (AMD64)". It features a "Get It Now" button, pricing information starting at \$0.99/hour, and links to "Overview", "Plans + Pricing", and "Ratings + reviews". The main content describes ParaTools Pro for E4S™ as a turnkey HPC Cluster solution with tuned MPI, batch job management, and a host of AI/ML and science & engineering tools. It highlights the use of the Extreme-scale Scientific Software Stack (E4S) hardened for commercial clouds, supported by ParaTools, Inc. The page also mentions the use of MVAPICH MPI implementation and the Spack package manager.

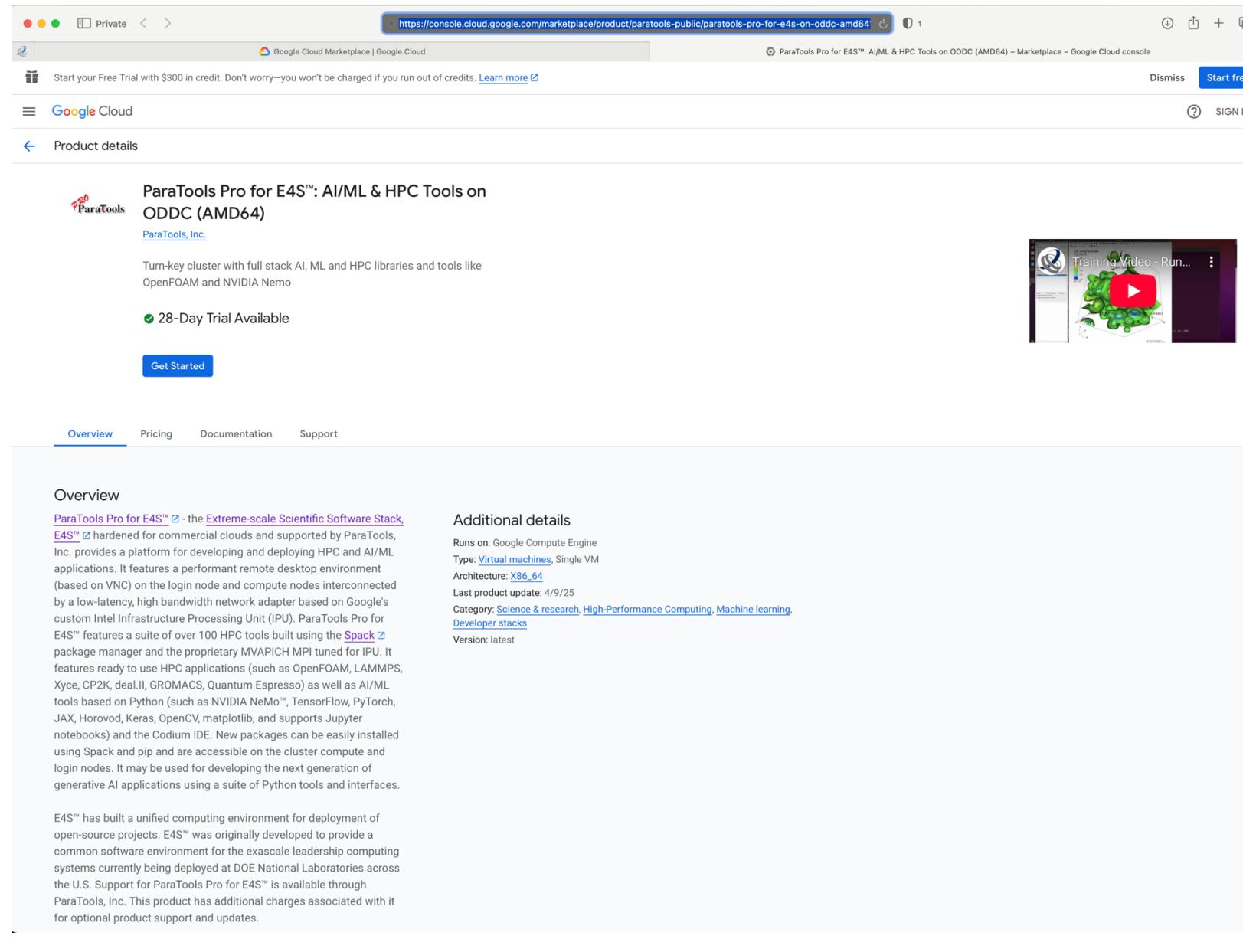
ParaTools Pro for E4S™ is a turnkey HPC Cluster solution with tuned MPI, batch job management, and a host of AI/ML and science & engineering tools.

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 MVAPICH MPI implementation tuned for Microsoft Azure's Infiniband enabled VMs. ParaTools Pro for E4S™ features a suite of over 100 HPC tools built using the Spack[3] package manager and MVAPICH MPI tuned for Azure's Infiniband interconnect. 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. Users wanting to stand up an HPC cluster with a complete software stack for AI/ML and/or HPC engineering and science applications will benefit from this offer. Building and configuring software for a consistent, robust and performant environment is surprisingly difficult. This product addresses the need for a complete software stack for HPC and AI/ML applications, including a remote desktop environment, batch job management, and a host of AI/ML and science & engineering tools. Leverage the spack package manager to install additional custom software or use the various pre-built components to build your own solutions. 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 (ASCR), under SBIR Award Number DE-SC0022502 ("E4S: Extreme-Scale Scientific Software Stack for Commercial Clouds"). Note: This product contains repackaged and tuned open source software, e.g., E4S™, Spack and AI/ML tools like NVIDIA NeMo™, Horovod, JAX, Keras etc. [1]: <https://paratoolspro.com/> [2]: <https://e4s.io> [3]: <https://spack.io>

ParaTools Pro for E4S™
on Azure Marketplace supports

- SLURM (Azure CycleCloud) and Torque schedulers (Adaptive Computing's ODDC)
- Support for Infiniband Network adapter

ParaTools Pro for E4S™ on Google Cloud Marketplace



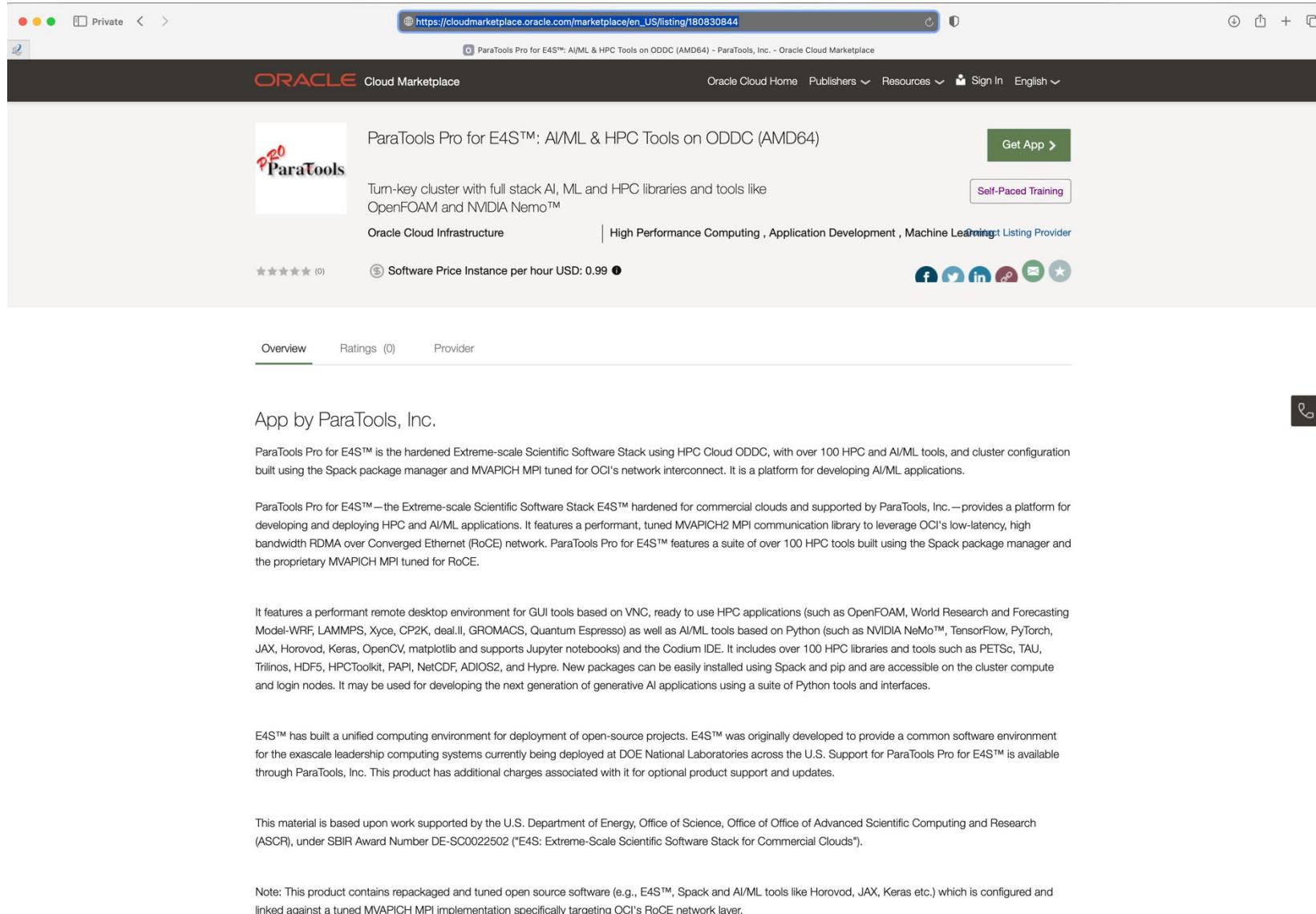
The screenshot shows the Google Cloud Marketplace product page for ParaTools Pro for E4S™. The page includes a banner for a free trial, navigation links like 'Google Cloud' and 'Product details', and a main section for 'ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC (AMD64)'. It highlights a 28-day trial available and features a 'Get Started' button. Below this, there are tabs for 'Overview', 'Pricing', 'Documentation', and 'Support'. The 'Overview' tab contains detailed information about the product, mentioning E4S™ as the Extreme-scale Scientific Software Stack, its support for ParaTools, Inc., and its features. The 'Additional details' section provides technical specifications such as running on Google Compute Engine, being a virtual machine, using X86_64 architecture, and having a last update on 4/9/25. It also lists categories like Science & research, High-Performance Computing, Machine learning, and Developer stacks. A video thumbnail titled 'Training Video - Run...' is shown on the right side.

ParaTools Pro for E4S™

Google Cloud Marketplace supports

- SLURM (GCluster) and Torque schedulers (Adaptive Computing's ODDC)
- Support for Google IPU network adapters

ParaTools Pro for E4S™ on Oracle Cloud Marketplace



The screenshot shows the Oracle Cloud Marketplace listing for ParaTools Pro for E4S. The page title is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC (AMD64) - ParaTools, Inc. - Oracle Cloud Marketplace". The main heading is "ParaTools Pro for E4S™: AI/ML & HPC Tools on ODDC (AMD64)". Below it, there's a brief description: "Turn-key cluster with full stack AI, ML and HPC libraries and tools like OpenFOAM and NVIDIA Nemo™". A "Get App" button is visible. To the right, there's a "Self-Paced Training" button. The Oracle Cloud Infrastructure logo is present, along with category tags: "High Performance Computing", "Application Development", and "Machine Learning". Below the description, it says "Software Price Instance per hour USD: 0.99". There are social sharing icons and a star rating section. At the bottom, there are tabs for "Overview", "Ratings (0)", and "Provider". The "Overview" tab is selected.

App by ParaTools, Inc.

ParaTools Pro for E4S™ is the hardened Extreme-scale Scientific Software Stack using HPC Cloud ODDC, with over 100 HPC and AI/ML tools, and cluster configuration built using the Spack package manager and MVAPICH MPI tuned for OCI's network interconnect. It is a platform for developing AI/ML applications.

ParaTools Pro for E4S™—the Extreme-scale Scientific Software Stack E4S™ 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, tuned MVAPICH2 MPI communication library to leverage OCI's low-latency, high bandwidth RDMA over Converged Ethernet (RoCE) network. ParaTools Pro for E4S™ features a suite of over 100 HPC tools built using the Spack package manager and the proprietary MVAPICH MPI tuned for RoCE.

It features a performant remote desktop environment for GUI tools based on VNC, ready to use HPC applications (such as OpenFOAM, World Research and Forecasting Model-WRF, 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. It includes over 100 HPC libraries and tools such as PETSc, TAU, Trilinos, HDF5, HPCToolkit, PAPI, NetCDF, ADIOS2, and Hypre. 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 (ASCR), under SBIR Award Number DE-SC0022502 ("E4S: Extreme-Scale Scientific Software Stack for Commercial Clouds").

Note: This product contains repackaged and tuned open source software (e.g., E4S™, Spack and AI/ML tools like Horovod, JAX, Keras etc.) which is configured and linked against a tuned MVAPICH MPI implementation specifically targeting OCI's RoCE network layer.

ParaTools Pro for E4S™
for Oracle Cloud Infrastructure
(OCI) Marketplace
supports Torque (ODDC) and
RDMA over Converged Ethernet
(RoCE) network adapters and
GPUs on login and compute nodes

Can E4S help provide a stable platform for tool development?

- Bare-metal installation as well as containers built with Spack
- Base containers that can be customized with e4s-alc
- Replace MPI in containerized E4S application with system MPI using e4s-cl
- Support for commercial cloud platforms
- What are we missing?

E4S: An HPC-AI Software Ecosystem for Science!

Singularity> spack find +cuda
-- linux-ubuntu22.04-x86_64_v3 / gcc@11.4.0 -----
adios2@2.10.2 camp@2024.07.0 flux-core@0.67.0 kokkos@4.5.01 libceed@0.12.0 petsc@3.22.4 tasmanian@8.1 zfp@1.0.0
amrex@25.03 camp@2024.07.0 ginkgo@1.9.0 kokkos@4.6.01 magma@2.9.0 raja@2024.07.0 tau@2.34.1
arborx@1.5 chai@2024.07.0 gromacs@2024.4 kokkos@4.6.01 mfem@4.7.0 raja@2024.07.0 trilinos@16.1.0
axom@0.10.1 chapel@2.4.0 heffte@2.4.1 kokkos@4.6.01 mgard@2023-12-09 slate@2024.10.29 umpire@2024.07.0
blaspp@2024.10.26 cp2k@2025.1 hpctoolkit@2024.01.1 kokkos-kernels@4.6.01 nvcomp@2.2.0 slepc@3.22.2 umpire@2024.07.0
bricks@2023.08.25 cusz@0.14.0 hpx@1.10.0 lammps@20240829.1 papi@7.1.0 strumpack@8.0.0 umpire@2024.07.0
cabana@0.7.0 fftx@1.2.0 hwloc@2.11.1 lapackpp@2024.10.26 parsec@3.0.2209 sundials@7.2.1 upcxx@2023.9.0
caliper@2.12.1 flecsi@2.3.2 hypre@2.32.0 legion@24.12.0 petsc@3.22.4 superlu-dist@9.1.0 vtk-m@2.2.0
==> 57 installed packages
Singularity> paraprof demo.ppk &

foo.py - nemo-speech_to_text - VSCode (on illyad)

```
File Edit Selection View Go Run Terminal Help  
EXPLORER ... foo.py x  
OPEN EDITORS x foo.py  
NEMO-SPEECH_TO_TEXT > an4  
an4_sphere.tar.gz u  
foo.py  
run.sh  
# Untar and convert .sph to .wav (using sox)  
tar = tarfile.open(an4_path)  
tar.extractall(path=data_dir)  
  
print("Converting .sph to .wav...")  
sph_list = glob.glob(data_dir + '/an4/**/*.*.sph', recursive=True)  
for sph_path in sph_list:  
    wav_path = sph_path[:-4] + '.wav'  
    cmd = ["sox", sph_path, wav_path]  
    subprocess.run(cmd)  
print("Finished conversion.\n*****")  
  
# NeMo's "core" package  
import nemo  
# NeMo's ASR collection - this collections contains complete ASR models and  
# building blocks (modules) for ASR  
import nemo.collections.asr as nemo_asr  
  
# This line will download pre-trained QuartzNet15x5 model from NVIDIA's NGC cloud  
quartznet = nemo_asr.models.EncDecCTCModel.from_pretrained(model_name="QuartzNet15x5")  
files = [os.path.join(data_dir, 'an4/wav/an4_clstkr/mgah/cen2-mgah-b.wav')]  
for fname, transcription in zip(files, quartznet.transcribe(audio=files)):  
    print(f"Audio in {fname} was recognized as: {transcription}")
```

TAU: ParaProf: 3D Visualizer: demo.ppk (on illyad)

File Options Windows Help
365.836
Seconds
0
Min node 0
node 1
node 2
node 3

Acknowledgment

- *This work was supported by the U.S. Department of Energy, Office of Science, Advanced Computing Research, through the Next-Generation Scientific Software Technologies (NGSST) under contract DE-AC02-AC05-000R22725, DOE SBIR DE-SC0022502, and NNSA Sandia contract 2542488.*



U.S. DEPARTMENT OF
ENERGY

Office of
Science



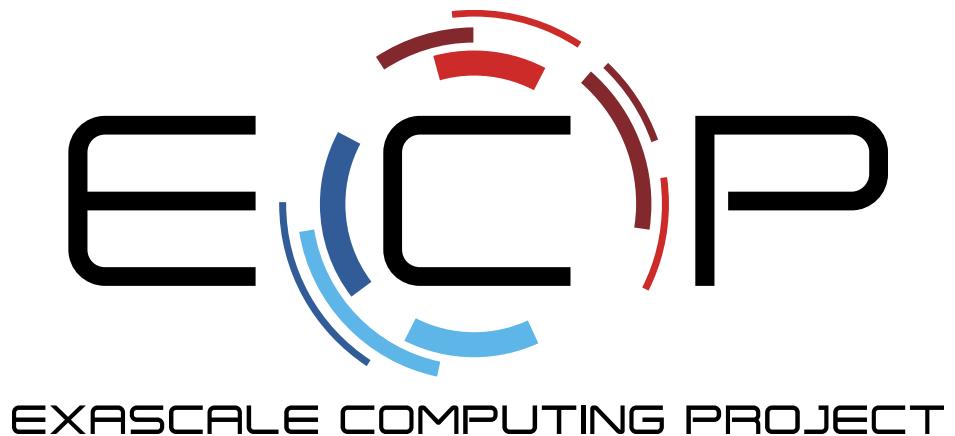
- <https://science.osti.gov/ascr>
- <https://pesoproject.org>
- <https://ascr-step.org>
- <https://hpsf.io>
- <https://www.energy.gov/technologytransitions/sbirstr>



Thank you

<https://www.exascaleproject.org>

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



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

