What is E4S?

E4S at NERSC 2022 workshop:
https://www.nersc.gov/users/training/events/e4s-at-nersc-2022/
Thursday, August 25, 2022

https://e4s.io/talks/E4S_at_NERSC_Shende_1.pptx

Sameer Shende
Research Professor and Director, Performance Research Lab, U. Oregon
E4S: Extreme-scale Scientific Software Stack

- Curated, Spack based software distribution
- 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 (with GPU support)
- GitHub recipes for creating custom images from base images
- GitLab integration for building E4S images
- E4S validation test suite on GitHub
- E4S-cl container launcher tool for MPI substitution in applications using MPICH ABI
- E4S VirtualBox image with support for container runtimes
  - Docker
  - Singularity
  - Shifter
  - Charliecloud
- AWS and GCP images to deploy E4S

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

- **E4S**: HPC Software Ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures with support for GPUs from NVIDIA, AMD, and Intel in a single distribution
- Available from **source, containers, cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Oct 2018: E4S 0.1 - 24 full, 24 partial release products
- Jan 2019: E4S 0.2 - 37 full, 10 partial release products
- Nov 2019: E4S 1.0 - 50 full, 5 partial release products
- Feb 2020: E4S 1.1 - 61 full release products
- Nov 2020: E4S 1.2 (aka, 20.10) - 67 full release products
- Feb 2021: E4S 21.02 - 67 full release, 4 partial release
- May 2021: E4S 21.05 - 76 full release products
- Aug 2021: E4S 21.08 - 88 full release products
- Nov 2021: E4S 21.11 - 91 full release products
- Feb 2022: E4S 22.02 – 100 full release products
- May 2022: E4S 22.05 – 101 full release products

Also include other products e.g.,
AI: PyTorch, TensorFlow (CUDA, ROCm)
Co-Design: AMReX, Cabana, MFEM

https://e4s.io
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
Scientific software is becoming extremely complex
Even proprietary codes are based on many open source libraries

• Half of this DAG is external (blue); more than half of it is open source
• Nearly all of it needs to be built specially for HPC to get the best performance
The Exascale Computing Project is building an entire ecosystem

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

- Up to 7 compilers
  - Intel
  - GCC
  - Clang
  - XL
  - PGI
  - Cray
  - NAG

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

- 2-3 versions of each package + external dependencies

= up to 1,260,000 combinations!

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

We must make it easier to rely on others’ software!
How to install software on a supercomputer

1. Download all 16 tarballs you need
2. Start building!
3. Run code
4. Segfault!?
5. Start over…
What about modules?

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

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

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

• Modules don’t handle installation!
  – They only modify your environment (things like PATH, LD_LIBRARY_PATH, etc.)

• Someone (likely a team of people) has already installed gcc for you!
  – Also, you can only `module load` the things they’ve installed
Spack is a flexible package manager for HPC

- How to install Spack (works out of the box):
  
  ```bash
  $ git clone https://github.com/spack/spack
  $ . spack/share/spack/setup-env.sh
  ```

- How to install a package:
  
  ```bash
  $ spack install tau
  ```

- TAU and its dependencies are installed within the Spack directory.

- Unlike typical package managers, Spack can also install many variants of the same build.
  - Different compilers
  - Different MPI implementations
  - Different build options

Visit spack.io

github.com/spack/spack

@spackpm
Spack provides the spec syntax to describe custom configurations

- 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

```plaintext
$ 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@2.31 @ custom version
$ spack install tau@2.31 %gcc@9.3.0 % custom compiler
$ spack install tau@2.31 %gcc@9.3.0 +rocm +/- build option
$ spack install tau@2.31 %gcc@9.3.0 +mpi ^mvapich2@2.3~wrapperrpath ^ dependency information
```

```bash
# set up compilers
# set up external packages
```
`spack find` shows what is installed

- All the versions coexist!
- Packages are installed to automatically find correct dependencies.
- Binaries work regardless of user’s environment.
- Spack also generates module files.

- Multiple versions of same package are ok.
- Packages are installed to automatically find correct dependencies.
- Binaries work regardless of user’s environment.
- Spack also generates module files.

```
Terminal — ssh: crusher — 198+57

[spack]@crusher-1crusher ~/spack find
>> 742 installed packages
| (spack-env:0) / spack/lib/spack/env
| (spack-env:1) / spack/lib/spack/env

... Output trimmed for brevity...

```

```bash
spack find
```

```bash
• All the versions coexist!
• Packages are installed to automatically find correct dependencies.
• Binaries work regardless of user’s environment.
• Spack also generates module files.

```
```

- Don’t have to use them.
The Spack community is growing rapidly

- **Spack simplifies HPC software for:**
  - Users
  - Developers
  - Cluster installations
  - The largest HPC facilities

- **Spack is central to ECP’s software strategy**
  - Enable software reuse for developers and users
  - Allow the facilities to consume the entire ECP stack

- **The roadmap is packed with new features:**
  - Building the ECP software distribution
  - Better workflows for building containers
  - Stacks for facilities
  - Chains for rapid dev workflow
  - Optimized binaries
  - Better dependency resolution

Visit spack.io

github.com/spack/spack

@spackpm
Facility Deployment: https://dashboard.e4s.io

<table>
<thead>
<tr>
<th>System</th>
<th>Deployment</th>
<th>Spack Details</th>
<th>Result</th>
<th>Spack Environment</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polaris</td>
<td>E4S 22.05, PregEnv-gnu, MVAPICH2</td>
<td>/usr/local/ParaTools/E4S/22.05/pregenv-gnu/spack</td>
<td>108/139</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/usr/local/ParaTools/E4S/22.05/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polaris</td>
<td>E4S 22.05, PregEnv-gnu</td>
<td>/usr/local/ParaTools/E4S/22.05/pregenv-gnu/spack</td>
<td>108/139</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/usr/local/ParaTools/E4S/22.05/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td>E4S 22.05, PregEnv-gnu, MVAPICH2</td>
<td>/global/cdf/crext4/shared-ParaTools/E4S/22.05/pregenv-gnu/spack</td>
<td>111/142</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/cdf/crext4/shared-ParaTools/E4S/22.05/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td>E4S 22.05, PregEnv-gnu</td>
<td>/global/cdf/crext4/shared-ParaTools/E4S/22.05/pregenv-gnu/spack</td>
<td>125/144</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/cdf/crext4/shared-ParaTools/E4S/22.05/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.05, PregEnv-gnu, MVAPICH2</td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-gnu/spack</td>
<td>112/139</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.05, PregEnv-gnu</td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-gnu/spack</td>
<td>117/130</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.05, PregEnv-amd</td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-amd/spack</td>
<td>99/131</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-amd/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.05, PregEnv-cray</td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-cray/spack</td>
<td>96/130</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.05/pregenv-cray/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JLSE</td>
<td>E4S 22.05, oneAPI</td>
<td>/usr/local/ParaTools/E4S/22.05/spack</td>
<td>79/110</td>
<td>-restricted-</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/usr/local/ParaTools/E4S/22.05/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.02, PregEnv-gnu</td>
<td>/global/alpine/xce49/world-shared/E4S/22.02/pregenv-gnu/spack</td>
<td>107/121</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.02/pregenv-gnu/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.02, PregEnv-cray</td>
<td>/global/alpine/xce49/world-shared/E4S/22.02/pregenv-cray/spack</td>
<td>83/119</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.02/pregenv-cray/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crusher</td>
<td>E4S 22.02, PregEnv-amd</td>
<td>/global/alpine/xce49/world-shared/E4S/22.02/pregenv-amd/spack</td>
<td>71/117</td>
<td>spack yml</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/global/alpine/xce49/world-shared/E4S/22.02/pregenv-amd/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JLSE</td>
<td>E4S 22.03, oneAPI</td>
<td>/usr/local/ParaTools/E4S/22.02/spack</td>
<td>62/105</td>
<td>-restricted-</td>
<td>Testsuite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/usr/local/ParaTools/E4S/22.02/module-use.sh</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Default modules for E4S on JLSE and Polaris at ALCF for all users:

% module avail e4s/22.05
E4S 22.05 on Perlmutter using PrgEnv-gnu

```bash
sameer@perlmutter:login13:--> module use $CFS/m3896/shared/modulefiles
goinger@perlmutter:login13:--> module avail e4s

----------- /global/cfs/cdirs/m3896/shared/modulefiles -----------
e4s/mvapich2/22.05 e4s/PrgEnv-gnu/22.05
----------- /global/common/software/nersc/pm-2022.05.0/extra_modulefiles -----------
e4s/spack-develop e4s/21.11-1mod e4s/21.11-tcl (D) spack/e4s-22.02 (D)

Where:
D: Default Module

Use "module spider" to find all possible modules and extensions.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".

sameer@perlmutter:login13:--> module load e4s/PrgEnv-gnu

Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.15

sameer@perlmutter:login13:--> spack find +cuda cuda_arch=80

```
Perlmutter @ NERSC: E4S 22.05 with PrgEnv-gnu
E4S Project

The Extreme-scale Scientific Software Stack

E4S 2.05 is now available!
See Downloads for more information.

What is E4S?

The Extreme-scale Scientific Software Stack (E4S) is a community effort to provide open source software packages for developing, deploying and running scientific applications on high-performance computing (HPC) platforms. E4S provides from-source builds and containers of a broad collection of HPC software packages.
Download E4S 22.05 GPU Container Images: NVIDIA, AMD, Intel

- Separate full featured Singularity images for 3 GPU architectures
- GPU base images for
  - x86_64 (Intel, AMD, NVIDIA)
  - ppc64le
  - aarch64
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
Download E4S 22.05 GPU Container Images: NVIDIA, AMD, Intel

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:

```bash
# docker pull ecpe4s/e4s-cuda:22.05
# docker pull ecpe4s/e4s-rocm:22.05
# docker pull ecpe4s/e4s-oneapi:22.05
```

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)**
  - ecpe4s/e4s-rocm:22.05
  - e4s-rocm-22.05.sif

- **NVIDIA CUDA (x86_64, ppc64le)**
  - ecpe4s/e4s-cuda:22.05
  - e4s-cuda-x86_64-22.05.sif
  - e4s-cuda-ppc64le-22.05.sif

- **Intel OneAPI (x86_64)**
  - ecpe4s/e4s-oneapi:22.05
  - e4s-oneapi-22.05.sif

https://e4s.io
Download E4S 22.05 Base GPU Container Images

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.

NVIDIA Multi-Arch (X86_64, PPC64LE, AARCH64)
- ecpe4s/e4s-base-cuda:22.05
- e4s-base-cuda-x86_64-22.05.sif
- e4s-base-cuda-aarch64-22.05.sif
- e4s-base-cuda-ppc64le-22.05.sif

ROCM X86_64
- ecpe4s/e4s-base-rocm:22.05
- e4s-base-rocm-22.05.sif

Intel OneAPI X86_64
- ecpe4s/e4s-base-oneapi:22.05
- e4s-base-oneapi-22.05.sif

https://e4s.io
Minimal Spack base image on Dockerhub

- Create custom container images
- 1M+ downloads!
22.05 Release: 101 Official Products + dependencies (gcc, x86_64)
22.05 Release: 101 Official Products + dependencies (gcc, x86_64)

34: hefftte
35: hptcutil
36: hp
37: hypre
38: kokkos
39: kokkos-kernels
40: lammps
41: legion
42: libnrm
43: Libqio
44: lok
45: magma
46: mercury
47: metall
48: mfem
49: mpich
50: mpirunlinutil
51: netlib-sacleapack
52: nccmp
53: nco
54: nrm
55: omega-h
56: openpmd-api
57: openmpi
58: papi
59: papyrus
60: parallel-netcdf
61: paraview
62: parsec
63: pdb
64: petsc
65: phist
66: plasma

/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/hefftte-2.2.0-hqigomw3nlexreke2rgwqwdazenvhb2
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/hptcutil-2022.04.15-cqz4vlmjcq6qmp4sonlqzig4twlsh
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/hpx-1.7.4-0y77frbq7yj2cvcfhhgsdtckzkhw3v
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/hpyre-2.2.0-c7v7qorxpxf6enrj42x632txsu64ax
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/kokkos-3.6.0-fy3onzyijziq2x2laoxu7trtrwqaqfuj
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/kokkos-kernels-3.6.0-gn7fvgn67ekhwt3240x451lizuea327n
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/lammps-20220107-prm51nxruxpzkucuk7qqr6q5siri
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/legion-21.03.0-0g3tu5cdevyuxrsrvcqinanxrgduil3ea54
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/libnrm-0.1.0-gjllhjve7in4rp3kvawyrw7uyjpjr
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/libqio-1.3.1-207zqn3e2f7e303z4hcwsgkn6bymm
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/loci-0.1.0-67nnzpehmoidifzfm6klyxhnbfgmpoaoa
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/magma-2.6.2-0mxm2xmmz7bcjbc2jv3vn6e0xrmviv24
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/mercury-2.1.0-yz4pmkdsnnmc5dgse5ine4ek2c07eaag763
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/metall-0.20-60ewbucx2xbs7j4f5qdeunqjkmogta
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/mfem-4.4.0-ra7q4j0335kqgwow7futluc5v5h5f4bu6b
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/mpich-4.0.2-ssbm22cbzxxbx7fzrlekupe23r0kwo2
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/mpirunlinutil-0.1.11-zvlai7bnjegjipd5zelxwvmwvmus6yesje5
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/netlib-sacleapack-2.0.0-12lic66vprvnc6rdsldqzz2rtrgdkib
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/nccmp-1.9.0.1-ef6vnl4s3jl7iwuvgip5c21trzzpvo
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/nco-5.0.1-1lx5wvw6bprieree7rtyknegskv54jsho
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/nrm-0.1.0-qiyhjsurt3gq4bn37u1vylzhdnmu
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/omega-h-9.34.1-3d2vshz6t7tw3nqchebl54zwrzym
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/openpmd-api-0.14.4-6tkbump7eqm673vflkx3japn3kanommm
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/openmpi-4.1.3-0p52yygp07hc2m26f205ymx3tt3hdwt
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/papi-6.0.0-1-fxfybh2varhhxnu6jcrqsg5ixxrxcpx2
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/papyrus-1.0.2-r2we2f2jd24lq0odc43h3n23f0xowzet2
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/parallel-netcdf-1.12.2-7ruhgcy35ho6fg4i6c3d2tteuhlqlw
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/paraview-5.10.1-o0hruf5ntfp2yeyw6lslszry5jbfq5pww7
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/parsec-3.0.2012-dcaz21w6nrz716kcwpwsabh3t3uvr6v
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/pdb-3.25.1-s5yjifrocerrc15cxqwsfrvbl51776d7r
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/petsc-3.17.1-vff66bykiky34a1uxoe7diz42ggbrrdqb
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/phist-1.9.5-vmwxjbnbs3kvwtopglldhmouoehg7l7m
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/plasma-21.8.29-hwpchbnbscxsmsmguyoieepejwddgnunxm
22.05 Release: 101 Official Products + dependencies (gcc, x86_64)

- **GPU runtimes**
  - Intel (oneAPI)
    - 2022.1.0
  - AMD (ROCm)
    - 5.1.1
  - NVIDIA (CUDA)
    - 11.4
  - NVPHC
    - 22.3

**76:** qthreads

**77:** raja

**78:** rempi

**79:** scr

**80:** slate

**81:** slepc

**82:** stc

**83:** strumpack

**84:** sundials

**85:** superlu-dist

**86:** swig

**87:** sz

**88:** tasmanian

**89:** tau

**90:** trilinos

**91:** turbine

**92:** umpar

**93:** umpire

**94:** unifyfs

**95:** upcxx

**96:** variorum

**97:** veloc

**98:** vtk-m

**99:** wannier90

**100:** warpx

**101:** zfp
E4S 22.05 container deployment on Perlmutter using Shifter

- E4S containers
- Accessing A100 GPUs
- CUDA 11.5
E4S 22.05 packages built with support for A100 GPUs in a container

• AI/ML

• HPC
E4S 22.05 Release: Support for NVIDIA GPUs

Singularity> python
Python 3.9.7 (default, Sep 16 2021, 13:09:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.9.1'
>>> tf.config.list_physical_devices('GPU')
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'), PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU')]
>>> import torch
>>> torch.__version__
'1.11.0+cu113'
>>> torch.cuda.get_device_name(torch.cuda.current_device())
'NVIDIA A100-PCIe-40GB'

Singularity> spack find -l +cuda cuda_arch=80
==> 27 installed packages
-- linux-ubuntu20.04-x86_64 / gcc@9.4.0

Singularity> spack find -l +cuda cuda_arch=70
==> 27 installed packages
-- linux-ubuntu20.04-x86_64 / gcc@9.4.0

Singularity>
E4S 22.05 Release: GUI Tools

Singularity> module load paraview
Singularity> which paraview
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/paraview-5.10.1-ohru5tnfp2yeywlszszy5jfbq6pjw7/bin/paraview
Singularity> module load visit
Singularity> which visit
/usr/local/visit/bin/visit
Singularity> which paraprof
/spack/opt/spack/linux-ubuntu20.04-x86_64/gcc-9.4.0/tau-2.31.1-ikx2r2pk34hej3rmjaoam5ma6rvzbuqm/bin/paraprof
Singularity> module load julia
Singularity> julia

Documentation: https://docs.julia-lang.org
Type "?" for help, "!?” for Pkg help.
Version 1.7.3 (2022-05-06)
Official https://julia-lang.org/ release

julia>
Singularity> nvidia-smi
Tue May 31 23:33:39 2022

<table>
<thead>
<tr>
<th>NVIDIA-SMI 470.57.02</th>
<th>Driver Version: 470.57.02</th>
<th>CUDA Version: 11.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Name</td>
<td>Persistence-M</td>
<td>Bus-Id</td>
</tr>
<tr>
<td>Fan Temp</td>
<td>Perf Wpw:Usage/Cap</td>
<td>Memory-Usage</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>0 NVIDIA A100-PCI...</td>
<td>Off</td>
<td>00000000:25:00.0 Off</td>
</tr>
<tr>
<td>N/A</td>
<td>42C</td>
<td>P0</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>1 NVIDIA A100-PCI...</td>
<td>Off</td>
<td>00000000:E1:00.0 Off</td>
</tr>
<tr>
<td>N/A</td>
<td>38C</td>
<td>P0</td>
</tr>
</tbody>
</table>
E4S 22.02 bare-metal Spack installation environments on GitHub

```yaml
packages:
  all:
    compiler:
      - gcc:9.3.0
    providers:
      blas:
        - openblas
      mpi:
        - epipich
      target:
        - x86_64
    variants: +mpi
  binutils:
    variants: +ld +gold +headers +libiberty +nls
    version:
      - 2.36.1
  cuds:
    version:
      - 11.4.2
  djpeg:
    version:
      - 1.8.20
  elfutils:
    variants: +gzip +nls +xz
  hdf5:
    variants: +fortran +h1 +shared
    version:
      - 1.18.7
  libfabric:
    variants: +fabric=sockets,tcp,udp,rsm
  libgmem:
    variants: +pic +xz
  meso:
    variants: -llvm
  mesa:
    variants: -llvm
  mpi:
    variants: +wrapcraypath
  nurees:
  variants: +stelwib
  openblas:
  variants: +threads=nopenmp
  python:
    version:
      - 3.8.32
  trilinos:
    variants: +amesos +amesos2 +amasso2 +scalapack +blas +boost +petra +petraext
    +fpack +mpack2 +intrepid +intrepid2 +isorropia +kokkos +ml +minicondor +mlu
    +mx +pico +phalans +ral +pyromos + sacado +atk +shards +shyhu +astokos +stratisklos
    +teko +tempus +tpetra +trilinoscouplings +zoltan +zoltan2 +superlu-dist gotype=long_long
```

spack.yaml
E4S 22.02 bare-metal installation spack.yaml recipe

- E4S products built with CUDA for A100
- Built with ROCm for MI100 and MI250X
- Built with oneAPI
E4S Validation Test Suite

- Provides automated build and run tests
- Validate container environments and products
- New LLVM validation test suite for DOE LLVM

- git clone https://github.com/E4S-Project/testsuite.git
E4S MPI Tests (OMB) using Buildtest @ NERSC

- CI/CD using GitLab
- Buildtest integration
Multi-platform E4S Docker Recipes
E4S: Multi-platform Reproducible Docker Recipes

https://e4s.io

- x86_64
- ppc64le
- aarch64
WDMApp: Speeding up bare-metal installs using E4S build cache

- E4S Spack build cache
- Adding E4S mirror
- WDMApp install speeds up!
Pantheon and E4S build cache support end-to-end ECP examples

**Overview:** The Exascale Computing Project (ECP) is a complex undertaking, involving a myriad of technologies working together. An outstanding need is a way to capture, curate, communicate and validate workflows that cross all of these boundaries.

The **Pantheon** and **E4S** projects are collaborating to advance the integration and testing of capabilities, and to promote understanding of the complex workflows required by the ECP project. Utilizing a host of ECP technologies (spack, Ascent, Cinema, among others), this collaboration brings curated workflows to the fingertips of ECP researchers.

**Contributions**

- Curated end-to-end application/in-situ analysis examples can be run quickly by anyone on Summit. ([https://github.com/pantheonscience/ECP-E4S-Examples](https://github.com/pantheonscience/ECP-E4S-Examples))

- Pantheon/E4S integration speeds up build/setup times over source builds due to cached binaries (**approx.10x speed up**).
E4S Build Cache at U. Oregon

E4S Build Cache for Spack 0.18.0

To add this mirror to your Spack:
- `spack mirror add e4s https://cache.e4s.io`
- `spack buildcache keys -i`

88,011 total packages

Last updated 2022-05-30 16:42 PDT

All Arch  | PPC64LE  | X86_64

All OS  | CentOS 7  | CentOS 8  | RHEL 7  | RHEL 8  | Ubuntu 18.04 | Ubuntu 20.04

Search

Over 88,000 binaries!
E4S Support for AD teams: ExaWind

**ExaWind Project Engagement**

- Daily development builds of the ExaWind software stack are being containerized and distributed to ExaWind developers via Docker Hub.
- These container images contain Spack-based development builds of AMR-Wind, Nalu-Wind, Trilinos and other elements of the ExaWind software stack.
- The build process for these containers are integrated via a meta-build tool developed in-house by the ExaWind team called Spack Manager.
- Container images are posted daily to the ecpe4s/exawind-snapshot DockerHub repository
- GitLab Repository for exawind-snapshot project
- Exawind-Driver CI using ecpe4s/exawind-snapshot
- Nalu-Wind CI using ecpe4s/exawind-snapshot
- AMR-Wind CI using ecpe4s/exawind-snapshot
E4S Custom Docker Images using E4S Build Cache: ExaWind
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 and MPICH variants.

- 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.

- Usage:

  ```
  . /opt/intel/oneapi/setvars.sh
e4s-cl init --backend singularity --image /home/tutorial/ecp.simg --source /home/tutorial/source.sh
cat ~/source.sh
  . /spack/share/spack/setup-env.sh
  spack load trilinos+cuda cuda_arch=80

e4s-cl mpirun -np 4 ./a.out
  ```

  [https://github.com/E4S-Project/e4s-cl](https://github.com/E4S-Project/e4s-cl)
e4s-cl Container Launcher

Login host

e4s-cl launch

Work hosts

e4s-cl execute

library resolution

container launch

Containers

Host libraries

MPI program

 MPI program

https://e4s.io
E4S VirtualBox Image

Container Runtimes
- Docker
- Shifter
- Singularity
- Charliecloud

https://e4s.io
Private E4S 22.05 image
- Build cache
- Nalu-X demonstration
- Singularity and Docker runtimes
- Discussing a compelling demo in E4S iteration 42
- Was demonstrated at ISC22 and ATPESC.
# E4S Summary

## What E4S is not

- A closed system taking contributions only from DOE software development teams.
- A monolithic, take-it-or-leave-it software behemoth.
- A commercial product.
- A simple packaging of existing software.

## What E4S is

- Extensible, open architecture software ecosystem accepting contributions from US and international teams. Framework for collaborative open-source product integration.
- A full collection of compatible software capabilities **and** A manifest of a la carte selectable software capabilities.
- Vehicle for delivering high-quality reusable software products in collaboration with others.
- The conduit for future leading edge HPC software targeting scalable next-generation computing platforms. A hierarchical software framework to enhance (via SDKs) software interoperability and quality expectations.
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.”