SAND2020-10318PE

ECP Container Status - 2020

EXASCALE COMPUTING PROJECT

Approved for public release

Andrew J. Younge (SNL)

September 23rd, 2019

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

ECP Supercontainers Effort

- Joint DOE effort - LANL, LBNL, LLNL, Sandia, U. of Oregon
- Ensure container runtimes will be scalable, interoperable, and well integrated across DOE
  - Enable container deployments from laptops to Exascale
  - Assist Exascale applications and facilities leverage containers most efficiently
- Three-fold approach
  - Scalable R&D activities
  - Collaboration with related ST and AD projects
  - Training, Education, and Support
- Activities conducted in the context of interoperability
  - Portable solutions
    - Optimized E4S container images for each machine type
    - Containerized ECP that runs on Astra, A21, El-Capitan, …
  - Work for multiple container implementations
    - Not picking a “winning” container runtime
  - Multiple DOE facilities at multiple scales

SUPERCONTAINERS

HPC container runtimes are rapidly emerging at DOE sites

ALCF

- Theta: Singularity
- Aurora: Singularity (TBD)

OLCF

- Summit: Singularity (trial)
- Frontier: Singularity (2022)

NERSC

- Cori: Shifter
- Perlmutter: Shifter or Singularity (2020)

LLNL

- Sierra/Lassen: Singularity (trial)
- Linux clusters: Singularity
- El Capitan: Singularity (2023)

LANL

- Trinity: Charliecloud
- Linux clusters: Charliecloud
- Crossroads: Charliecloud (2021)

Sandia

- Astra: Singularity, Charliecloud, & Podma
  Linux clusters: Singularity

Many sites are rolling out container runtimes for users.
We are developing resources to facilitate consistent, performant deployment across sites.

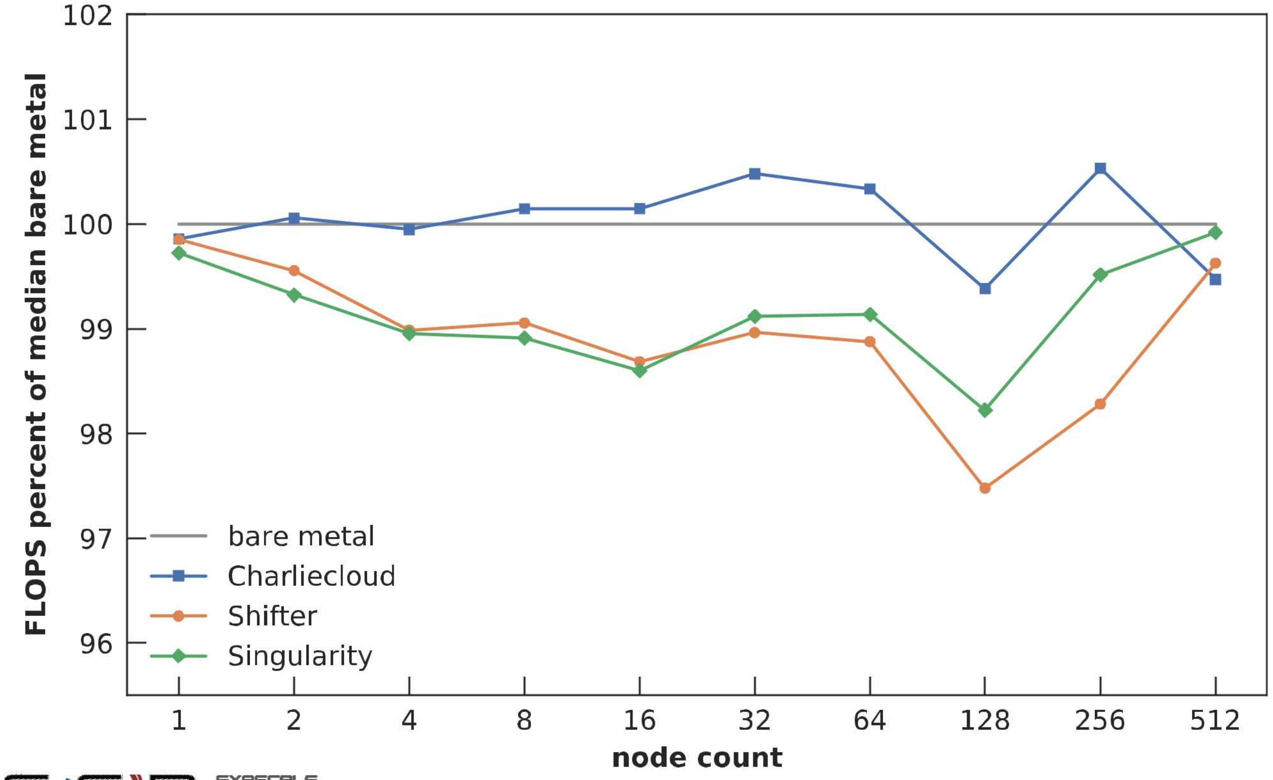
Focus on OCI-spec Container Images

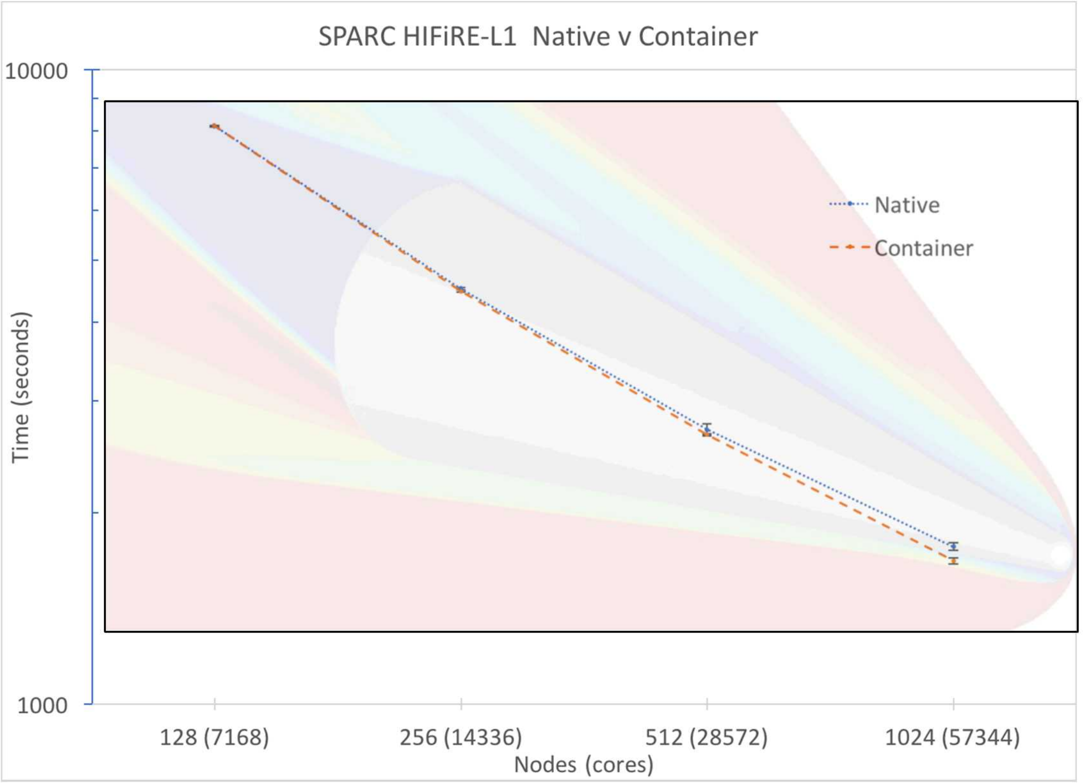
- Diversity in container runtimes does not mean diversity in container image types!
- Directed focus only on Open Container Initiative (OCI) images
- Effectively build from Docker v2.2 format
  - Uses Dockerfiles
  - Follows community-driven image conventions
- Can be *built* with several modern container runtimes
  - Docker, Podman, Buildah, …
- Can be *run* on several HPD container runtimes
  - Singularity, Shifter, Charliecloud, SARUS, …
- Can be *stored* across many DOE container registry services:
  - Gitlab, OpenShift, Harbor, …
- Allow for ECP to integrate and share containers across wider community
  - Deploy ECP software in the cloud?

https://github.com/opencontainers/image-spec

Container performance overhead and scalability well known

- Detailed performance study on LANL commodity cluster (left)
- Scaled ATDM app to 2048 nodes on Astra with Singularity = near-native performance (right)
- Several Shifter experiments on Cori confirm near-native performance



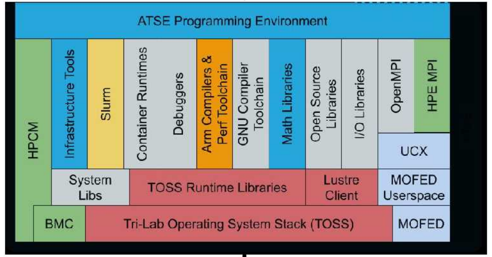


Credit: Reid Priedhorsky (LANL)

Podman for Un-privileged Container Builds

- Build containers directly on HPC nodes
  - Doing so w/ Docker requires root
  - Need user functionality for building containers
- Leverage user namespaces for _building_ containers
- Podman and Buildah to provide container builds functionality while maintaining user-level permissions
  - User namespaces
  - Set uid/gid mappers
  - TBD Overlay & FUSE for mount
- NEXT: E4S enablement for ECP

```
podman build -t "gitlab.sandia.gov/atse/astra:1.2.4" .
```



ATSE Programming Environment

HPCM

Infrastructure Tools

Slurm

Container Runtimes

Debuggers

Arm Compilers & Perf Toolchain

GNU Compiler Toolchain

Math Libraries

Open Source Libraries

I/O Libraries

OpenMPI

HPE MPI

UCX

System Libs

TOSS Runtime Libraries

Lustre Client

MOFED Userspace

BMC

Tri-Lab Operating System Stack (TOSS)

MOFED

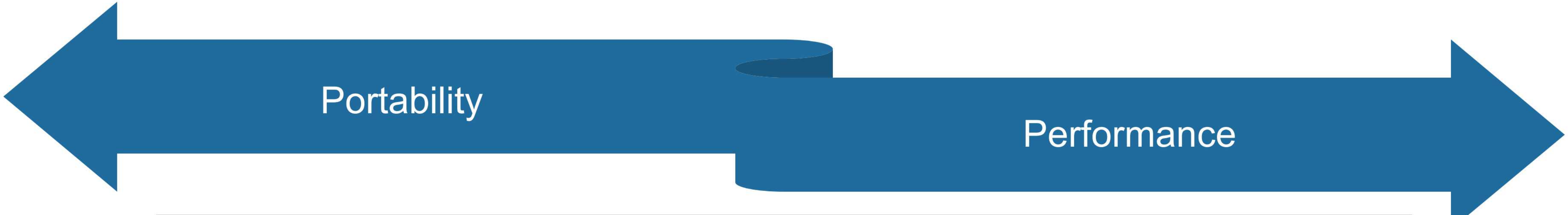
```
podman push gitlab.sandia.gov/atse/astra:1.2.4
```

```
singularity build atse-astra-1.2.4.sif docker://gitlab.sandia.gov/atse/astra:1.2.4
```

```
salloc -N 2048 && mpirun -np $NP singularity exec atse-astra-1.2.4.sif /app
```

Pedretti, Younge, et. al, *Chronicles of Astra: Challenges and Lessons from the First Petascale Arm Supercomputer* , to appear in SC20

There is a container performance-portability continuum



How do we strike the right balance?

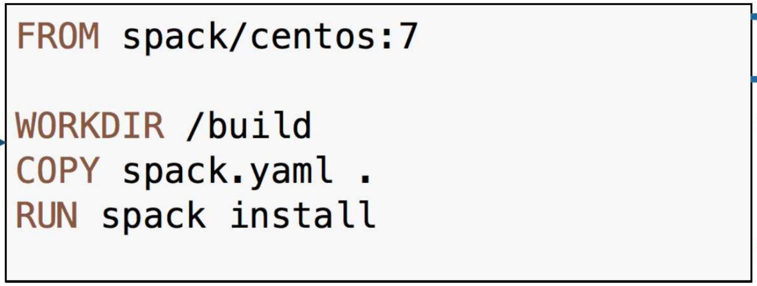
- Portable container images can be moved form one resource deployment to another with ease
- Reproducibility is possible
  - Everything (minus kernel) is self-contained
  - Traceability is possible via build manuscripts
  - No image modifications
- **Performance can suffer – no optimizations**
  - Can’t build for AVX512 and run on Haswell
  - Unable to leverage latest GPU drivers

- Performant container images can run at near-native performance compared to natively build applications
- Requires targeted builds for custom hardware
  - Specialized interconnect optimizations
  - Vendor-proprietary software
- Host libraries are mounted into containers
  - Load system MPI library (glibc issues!?)
  - Match accelerator libs to host driver
- **Not portable across multiple systems**

Simplified container builds using Spack Environments

- We recently started providing base images on DockerHub with Spack preinstalled.
- **Very** easy to build a container with some Spack packages in it:

```
spack-docker-demo/
        Dockerfile
        spack.yaml
```

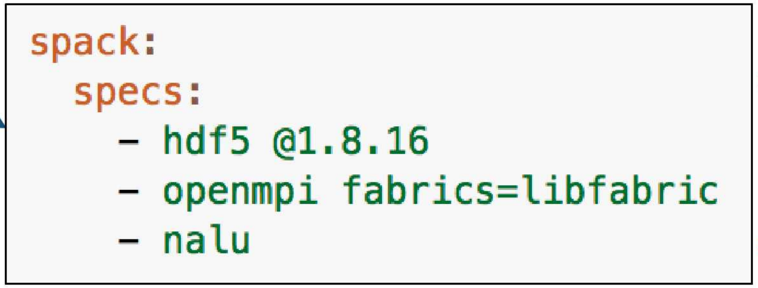


```
FROM spack/centos:7

WORKDIR /build
COPY spack.yaml .
RUN spack install
```

Base image with Spack in PATH

Copy in spack.yaml
Then run `spack install`



```
spack:
  specs:
    - hdf5 @1.8.16
    - openmpi fabrics=libfabric
    - nalu
```

List of packages to install, with constraints

Build with **`docker build .`**

Run with Singularity (or some other tool)

Credit: Todd Gamblin (LLNL)

E4S: Extreme-scale Scientific Software Stack

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

E4S

https://e4s.io

Credit: Sameer Shende (U of Oregon)