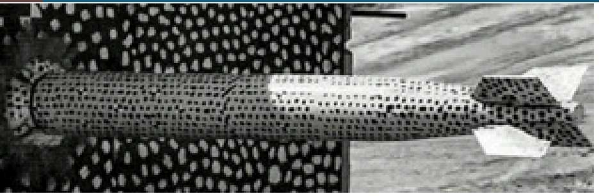
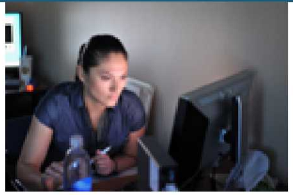
SAND2020-10317C

Towards Containerized HPC Applications at Exascale





PRESENTED BY

Andrew J. Younge

Sandia National Laboratories

ajyoung@sandia.gov

E4S Forum 2020

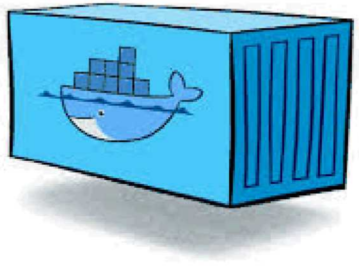
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

- Container background
- Introduction to *Supercontainers*
- Building HPC Containers
- Running HPC Containers
- Conclusion

What is a Container?

- Unit of software which packages up all code and dependencies necessary to execute single process or task
- Encapsulates the entire software ecosystem (minus the kernel)
- OS-level virtualization mechanism
  - Different than Virtual Machines
  - Think ”chroot” on steroids, BSD Jails
  - Dependent on host OS, which is (usually) Linux
  - Uses namespaces (user, mount, pid, etc)
- Docker is the leading container runtime
  - Used extensively in industry/cloud enterprise
  - Foundation for Kubernetes and Google cloud
  - Supported in Amazon AWS cloud



High Performance Computing

- DOE has a long history of investment in HPC
  - Stockpile Stewardship & simulation-based science
    - ASCI Red – first Teraflop Supercomputer
    - Red Storm - 100 Tflops and MPP
    - Summit/Sierra - 200 Pflops
  - Bulk synchronous parallel computing ~ HPC
    - 1 application spanning thousands of CPUs concurrently
    - Large-scale capability simulations
    - Also many capacity workloads
  - HPC represents the pinnacle of computing today

- Mission workloads computational requirements demand scale
  - Tightly coupled BSP simulation codes typically use MPI for communication
  - Many workload ensembles quickly expanding to ML/DL/AI

The Cloudy relationship with HPC

- Public cloud computing is often prohibitive
  - Cost – expensive to run millions of CPU hours
  - Security – Can we trust public clouds?
- However, HPC is not traditionally as flexible as “the cloud”
  - Shared resource models
  - Static software environments
  - Requires modification of many COTS tools
- Containers have become a primary cloud deployment model

- What about Containers in HPC?
  - Can we support containers in HPC in the same way as clouds do? *Yes and No*
  - Does this model fit for _both_ HPC and emerging workloads across DOE? *Yes and No*
  - Can we adapt our current programming environments into container images? *Yes*

HPC Container Vision

- Support HPC software development and testing on laptops/workstations
  - Create working container builds that can run on supercomputers
  - Minimize dev time on supercomputers
- Scale containers to leadership-class supercomputing resources
- Developers specify how to build the environment AND the application
  - Users/analysts just import and run on a supercomputer
  - Many containers, but with different manifests target platforms & deployments.
  - Not bound to vendor release cycles, sysadmin reqs, etc
- Enable full-scale software ecosystems = E4S
- Performance matters
  - Use mini-apps to “shake out” container implementations on HPC
  - Expand to exascale workloads & applications
  - Enable features to support emerging workflows (ML/DL/in-situ analytics)

ECP Supercontainers

- 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

ECP
EXASCALE COMPUTING PROJECT

Building HPC Container Images

Have to start from somewhere...

E4S: Extreme-scale Scientific Software Stack

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

https://e4s.io

Credit: Sameer Shende (U of Oregon)

E4S Container Images Available

- Providing E4S container images on Dockerhub
  - https://hub.docker.com/u/ecpe4s
- Multiple ISAs
  - x86_64, ppc64le, and aarch64
- Multiple OS distros
  - Centos, UBI (Red Hat), and Ubuntu images
- Enabling containerized CI in Gitlab (see T. Gamblin's talk)
- Additional GPU support incoming

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**

Spack environments help with building containers

- We recently started providing base images 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

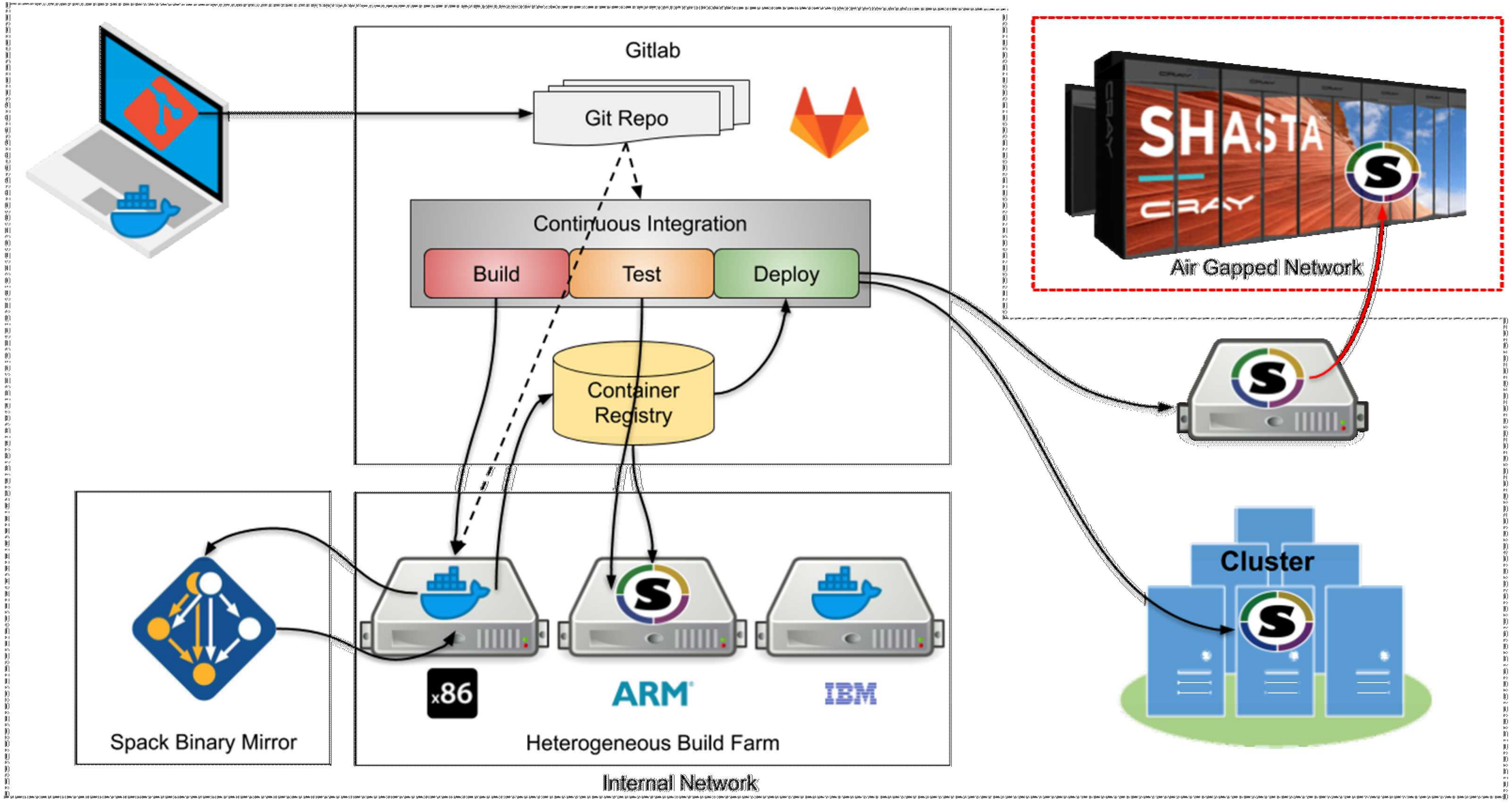
Build with docker build .

Run with Singularity (or some other tool)

Credit: Todd Gamblin (LLNL)

A Containerized CI Pipeline

- As a *developer* I want to *generate container builds from code* ~~*pull requests*~~ *releases* so that *containers are used to test & deploy new code on target HPC machines.*



Focus on OCI-spec Container Images

- Container runtime diversity 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?

OPEN CONTAINER INITIATIVE

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

Custom OCI Image Labels

SHIFTER

- HPC apps require special system libraries
  - CrayMPI linked in at runtime
- Fix: Leverage OCI-compatible image LABELs
  - Insert directly in Dockerfile
  - Embedding metadata into spec
- Labels specify expectations from the host
  - HPC container runtime intercepts labels, makes appro
  - Specify MPI version, Glibc expectation, etc
- Implemented prototype solution in Shifter
- Working with OCI container community

```
FROM centos:7

LABEL org.supercontainers.mpi=mpich
LABEL org.supercontainers.glibc=2.17

RUN yum -y update && \
    yum -y install gcc make gcc-gfortran \
           gcc-c++ wget curl

RUN B=mpich.org/static/downloads && V=3.2 && \
    wget $B/$V/mpich-$V.tar.gz && \
    tar xf mpich-$V.tar.gz && \
    cd mpich-$V && \
    ./configure && \
    make && \
    make install

ADD helloworld.c /src/helloworld.c

RUN mpicc -o /bin/hello /src/helloworld.c
```

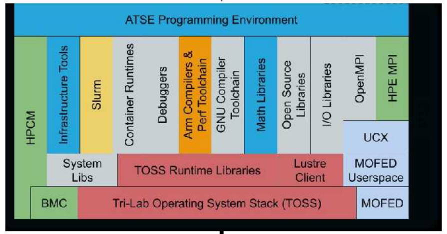
Label	Values	Comment
org.supercontainers.mpi	{mpich,openmpi}	Required MPI support, ABI compatibility
org.supercontainers.gpu	{cuda,opencl,rocm, etc}	Required GPU library support
org.supercontainers.glibc	Semantic version: XX.YY.Z	Specific version of GLIBC

Credit: Shane Canon (LBNL)

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: Enhanced E4S builds for ECP*

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




```
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

Running Containers at scale

Petascale? Exascale?

HPC Container Runtimes

- Docker is not good fit for running HPC workloads
  - Building with Docker on my laptop is ok
  - Security issues, no HPC integration
- Several different container options in HPC

SHIFTER Charliecloud

- All 3 HPC container runtimes are usable in HPC today!
- Each runtime offers different designs and OS mechanisms
  - Storage & mgmt of images
  - User, PID, Mount namespaces
  - Security models
  - OCI vs Docker vs Singularity images
  - Image signing, validation, registries, etc

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)

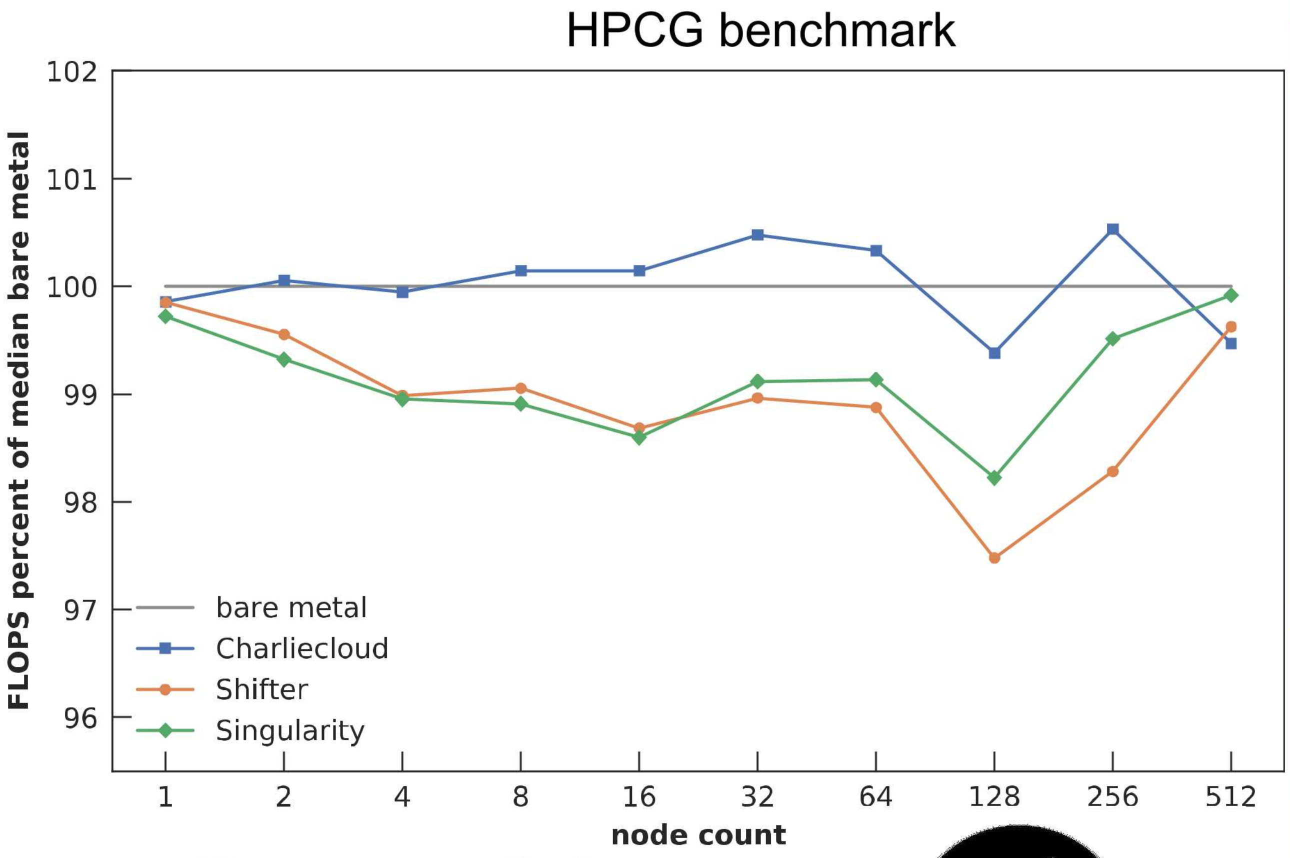
LANL
- Trinity: Charliecloud
- Linux clusters: Charliecloud
- Crossroads: Charliecloud (2021)

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.

"HPC container runtimes have minimal or no performance impact"

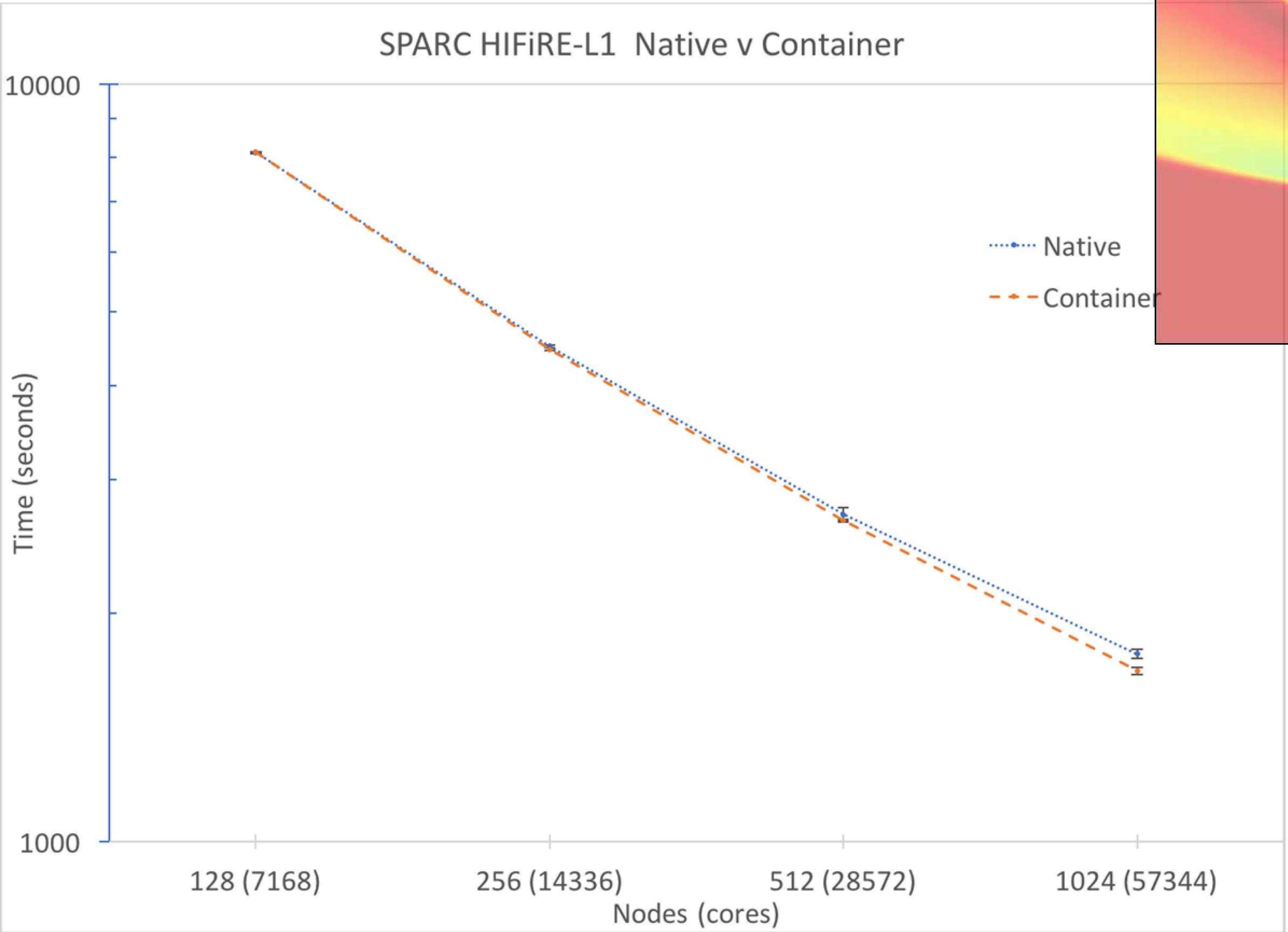
- LANL team confirms all HPC container runtimes perform well
  - Performance delta < 1% @ 512 nodes
  - "*we hypothesize that the performance impact of containerization itself is nil*."
  - Memory consumption may differ
- Pick a container runtime, any runtime!
  - More about features and experience
- Need to confirm experiments to Exascale

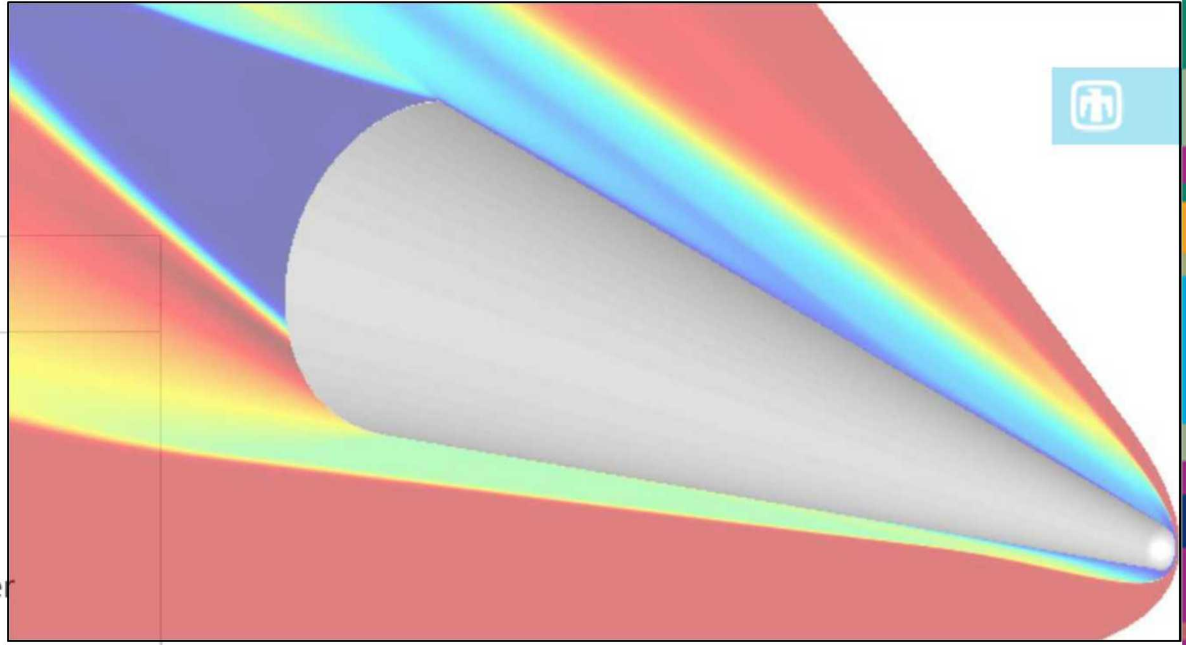


Charliecloud

From: Alfred Torrez, Tim Randles, and Reid Priedhorsky, *"HPC Container Runtimes Have Minimal or No Performance Impact"*, IEEE CANOPIE-HPC Workshop @ SC19, Nov 2019.

Case Study 1: SNL ATDM App





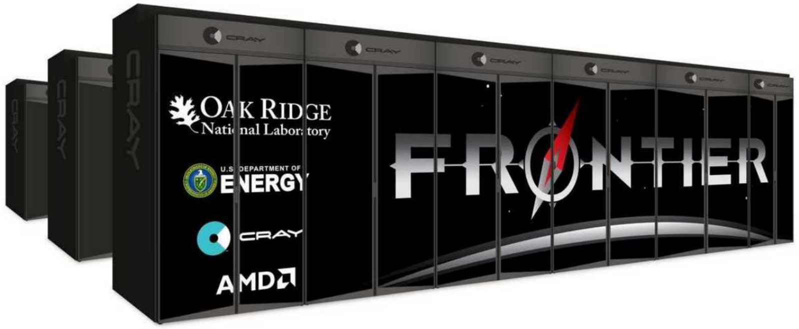
Points:

- Supporting SPARC containerized build & deployment
- Deployed on **Astra** with **Singularity**
- Near-native performance using a container
  - Container faster due to testing new optimizations for TX2
- Testing HIFiRE-1 Experiment (MacLean et al. 2008)
  - UNCLASSIFIED problem

Younge, et. al, *(U) Practice and Experience with Containers and ARM64 on Astra*, NNSA/AWE JOWOG 34

Vendor Engagement is a continual activity

- Starting working with Vendors & HW providers to enable container deployment
- Solutions must be both functional AND performant!
  - Scale is critical to success
  - Handle the growing heterogeneity in the HPC marketplace
- Working with vendors to provide optimized base container images
  - How do I build a my container to use CrayPE tools?
  - How do I use the latest GPU toolchain for my exa-app in a container?
- Enable new capabilities for controlling HPC software stack management [facilities]
  - Release testing, Rollback debugging, and cross-system software synchronization
- Fix some previous container issues moving forward
  - Can we stop runtime library hot-swapping?
  - Can we avoid `glibc` compatibility issues indefinitely?



Containers and Reproducibility?

- Reproducibility is a cornerstone of science!
  - Consistent results across studies aimed at answering the same **scientific** question
  - Critically important in conducting computational science today
- DOE/NNSA must extend the lifecycle without underground testing
- Rely on modeling and simulation apps to perform this task
  - Incorporate a multitude of physics and engineering models
  - Executed on leadership-class supercomputers
- Long-term studies take years
  - Any particular simulation may not seem important at the time
  - Later analysis may prove to demonstrate value in an old simulation
  - Need to reproduce & reevaluate runs many months or years later!
- Containerized builds can help future reproducibility efforts
- Containers alone are not the answer
  - Can be part of the solution

Conclusion

- Containers *will* be a viable software deployment model for the first Exascale platforms
- Demonstrated value of container models
  - Deployments in testbeds to production Petascale
  - Modern DevOps approach with containers useful
- Supercontainers = Container enablement at Exascale
  - Enable efficient *build* and *execution* of container images
  - Simplify HPC application development via modern DevOps
- E4S will prove-out containerized SW deployments at Exascale
  - Support next generation AI & ML apps
- Containers can increase software flexibility in HPC
- Could be the new-default mechanism for software deployment?

Acknowledgements:
Kevin Pedretti (1422)
Si Hammond (1422)
Jim Laros (1422)
Anthony Agelastos (9326)
Stephen Olivier (1423)
Justin Lamb (9326)
Jay Lofstead (1423)
Aron Warren (9327)
Erik Illescas (9327)
Ron Brightwell (1423)
Mike Heroux (1400)

Collaborators:
Shane Canon (NERSC)
Todd Gamblin (LLNL)
Reid Priedhorsky (LANL)
Sameer Shende (Oregon)
Todd Munson (ANL)
Angel Beltre (Binghamton)
Greg Kurtzer (CtrlIQ)
Eduardo Arango (Red Hat)

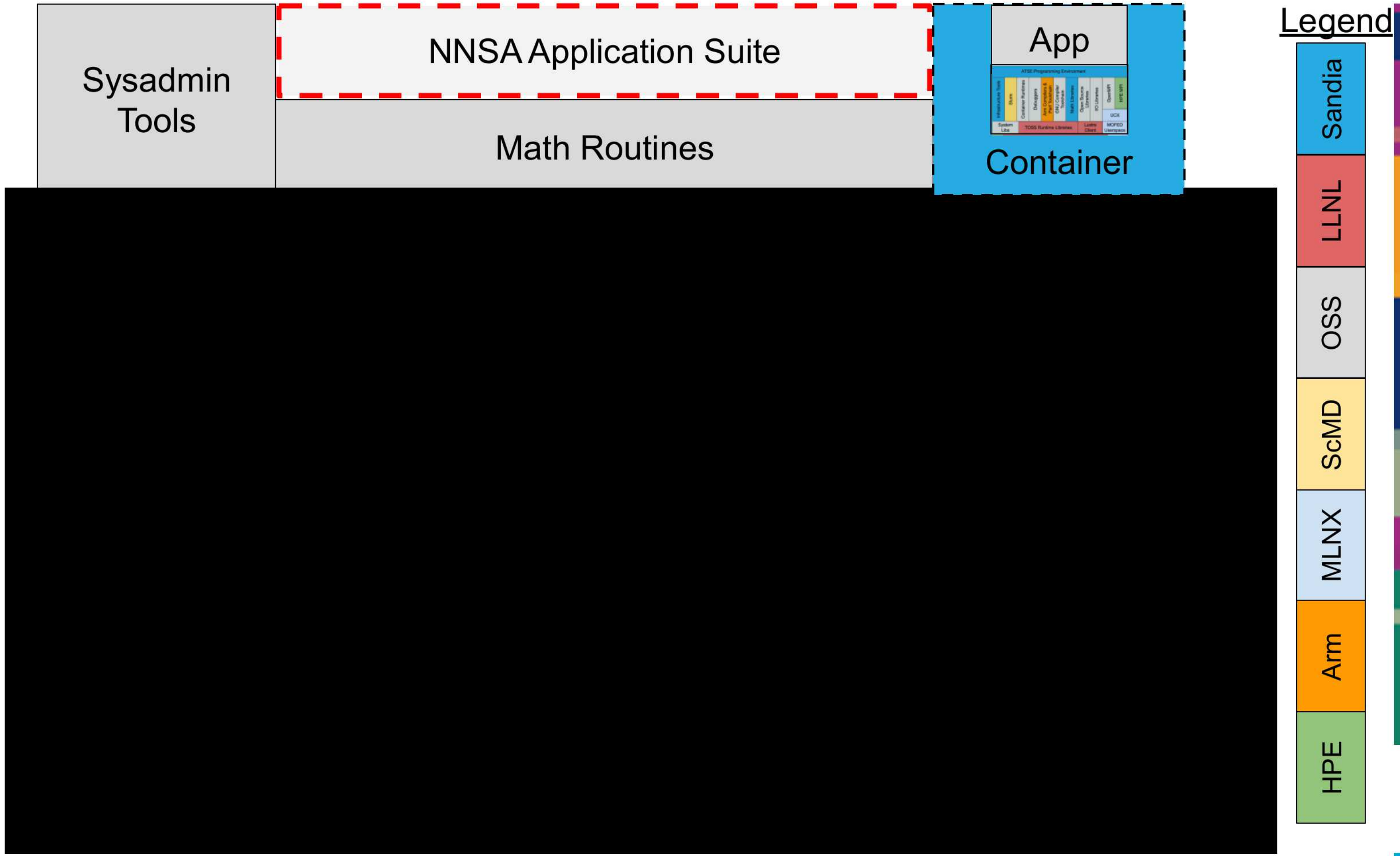
Questions?

ajyounge.com - ajyoung@sandia.gov


ARM SUPERCOMPUTER
ASTRA
"Per aspera ad astra"
Hewlett Packard
Enterprise
ATSE
stack
VANGUARD
2.3 PFLOPs peak
>5000 TX2 ARM CPUs, ~150k cores
885 TB/s memory bandwidth peak
332 TB memory
1.2 MW

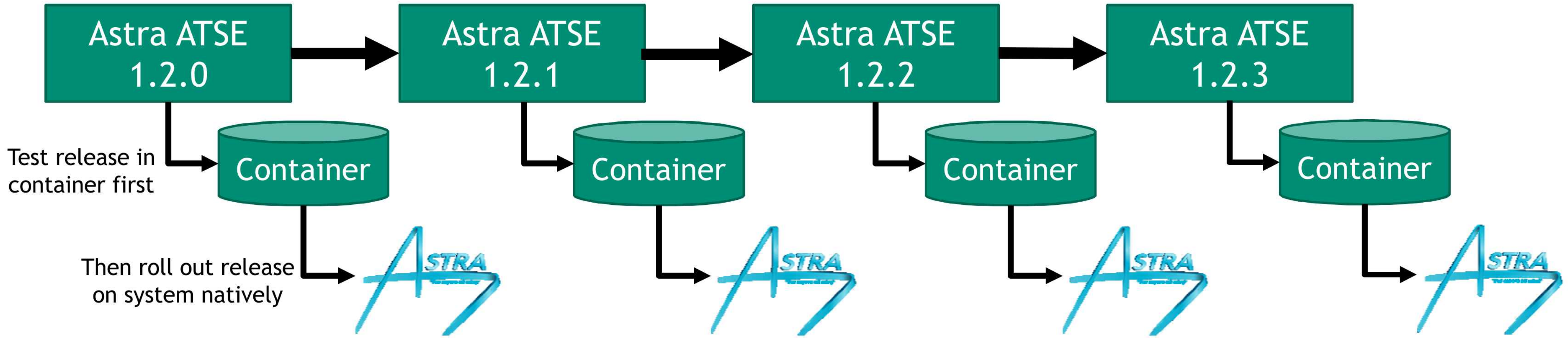
ATSE & Collaboration with HPE, OpenHPC, and ARM

- Advanced Tri-lab Software Environment = ATSE
- Many pieces to the software stack puzzle
- HPE's HPC Software Stack
  - HPE Cluster Manager
  - HPE MPI (+ XPMEM)
- Arm
  - Arm HPC Compilers
  - Arm Math Libraries
  - Allinea Tools
- Open source tools - OpenHPC
  - Slurm, OpenMPI, etc
- Mellanox-OFED & HPC-X
- RedHat 7.x for aarch64 - TOSS



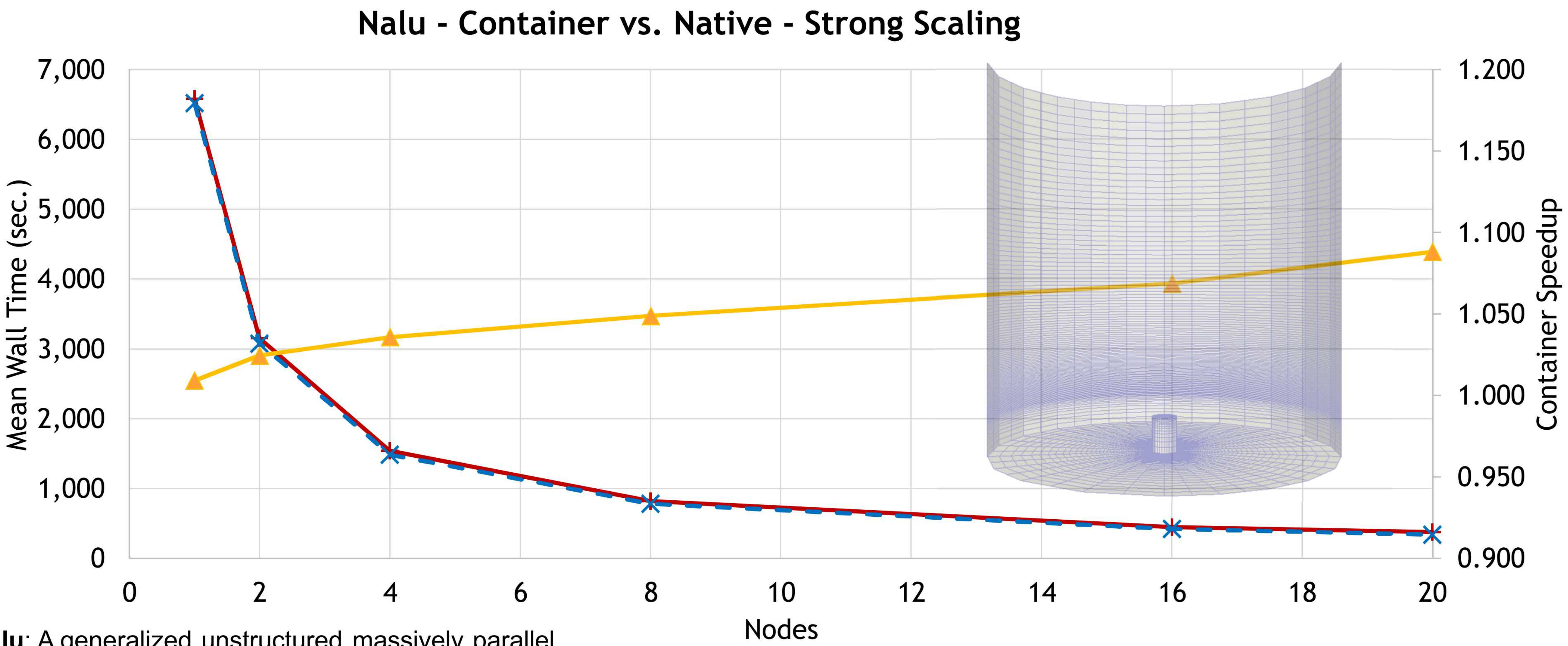
System Software Stack Testing & Debugging

- Astra ATSE programming environment release consists of:
  - TOSS base operating system + Mellanox InfiniBand stack
  - {2 compilers} * {3 mpi implementations} * {~25 libraries} = 150 packages
  - Each release packaged as a container for testing and archival purposes



- ATSE Container use cases:
  - **Release testing:** Enables full applications to be built and run at scale (2048+ nodes) before rolling out natively
  - **Rollback debug:** If issues are identified, ability to easily go back to a prior software release and test
  - **Cross-system synchronization**: Move full user-level software environments between systems. In one instance, this allowed an Astra InfiniBand library bug to be debugged off platform on another Arm cluster.

Case Study 2: Nalu CFD

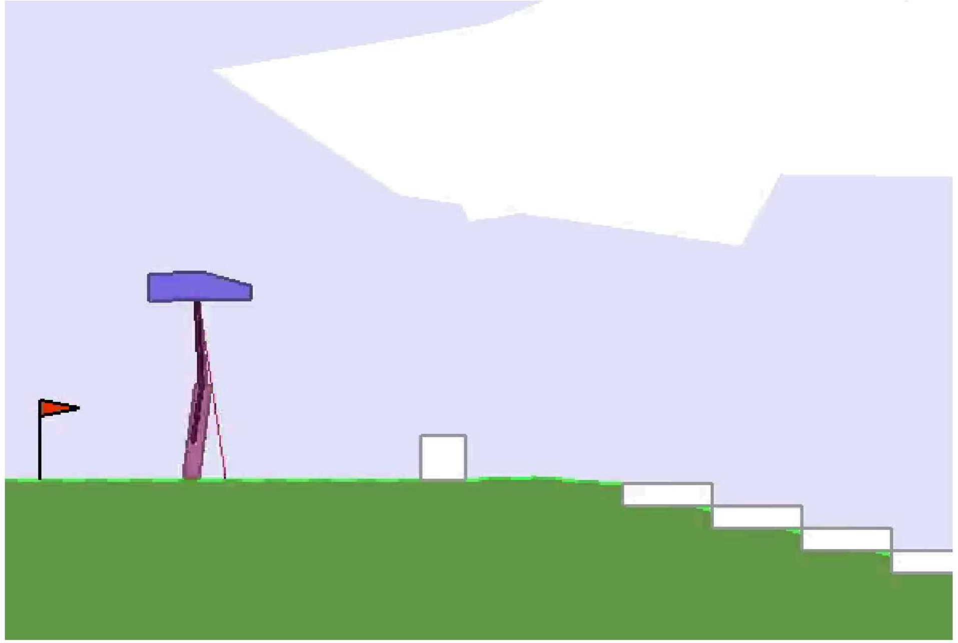


Nalu: A generalized unstructured massively parallel low Mach CFD flow code designed to support energy applications of interest

Agelastos, Younge, et. al, *(U) Quantifying Metrics to Evaluate Containers for Deployment and Usage of NNSA Production Applications,* NECDC 2018

Case Study 3: Reinforcement Learning Algorithms

- An evolutionary approach for multi-objective optimization
  - Evolutionary Algorithms are gradient-free population-based methods
  - EA benefits from parallelization and does not require GPU acceleration
    - Population of agents is generated and attempts a problem in parallel
    - High performance agents are used for next population generation
- Astra has been ideal for experimenting with EA
- We are using Astra for scaling of ASTool
  - Coevolves an agent's decision making policy and body
- Built Singularity container
  - Ubuntu 16.04, NumPy, PyBullet, …
  - Simple to use and modify
- 500 nodes - 7.5 hours



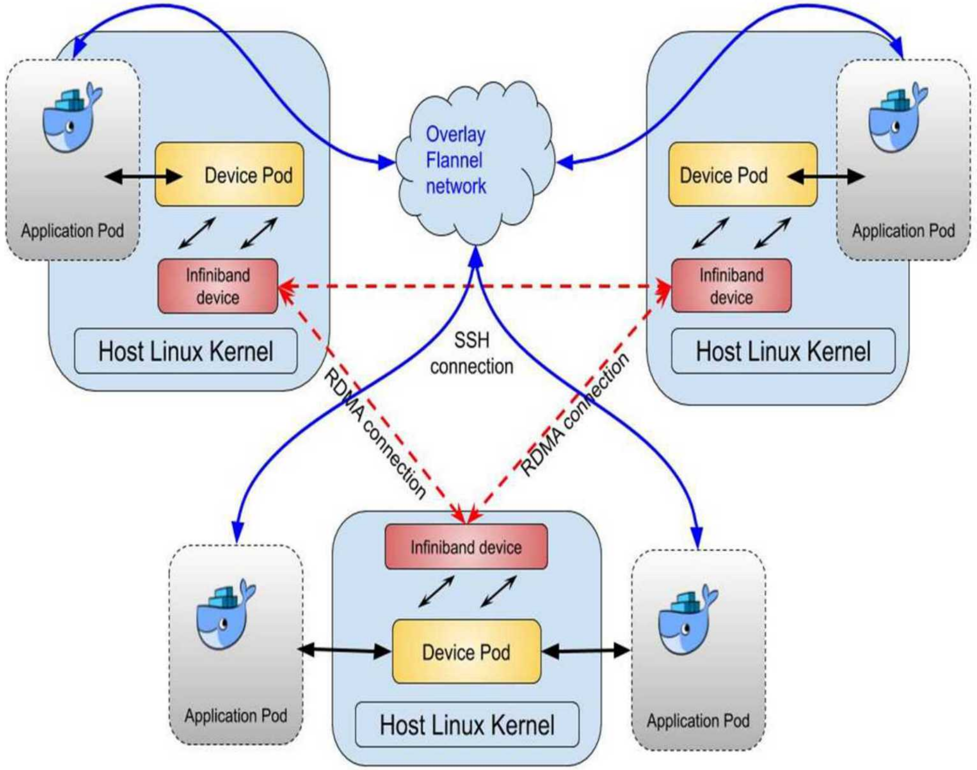
Credit: https://designrl.github.io/

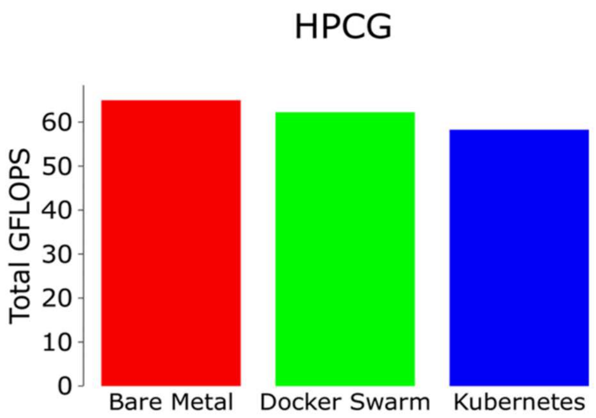
Takeaway: Containers help support Emerging HPC workloads like Reinforcement Learning

Kubernetes is coming…

kubernetes

- Containers are changing the software ecosystem for application deployment
- Container orchestration tools are now mainstream
  - VERY different than traditional HPC
  - No batch job scheduler, no jobs, just services and and orchestrator
- **Study Opportunities container orchestration frameworks in HPC**
  - Performance, Usability, and Constraints
- Orchestration and batch?
  - Separate solutions deployed today
  - Orchestration _and_ batch!

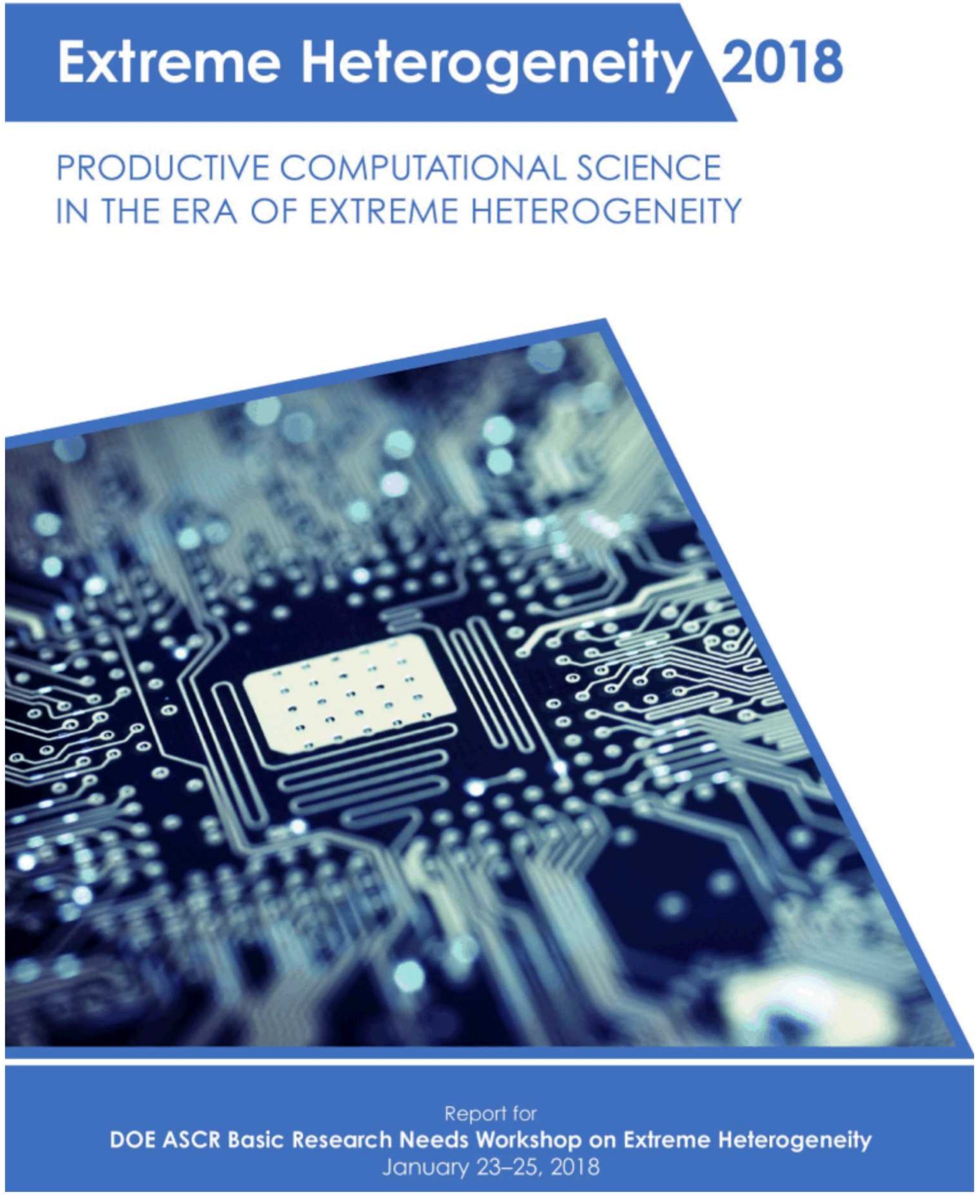




Credit: Angel Beltre (Binghamton University)

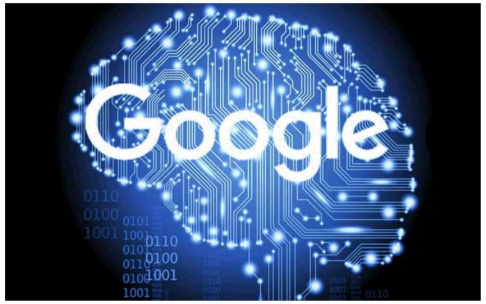
Position 1: Heterogeneity is the Future of HPC…

- HPC workloads are becoming more diverse
  - It's not just BSP simulations any more
  - Data is a cornerstone in ML, analytics, …
- And HPC hardware will be more diverse
  - "Era of predictable [hardware] improvements is ending."
  - Expecting custom aggregated components at the system level
- How will system software cope and support system-level heterogeneity?
  - How will programmers be efficient in such a landscape?
  - Will abstractions help or hinder performance?
- We need more APIs…



Position 2: … and so is the Cloud

- The hyperscalers are finally paying attention to HPC
  - *"The physical network topology does affect performance; particularly important is the performance of MPI Allreduce, grouped by splitting the mesh by a subset of the dimensions, which can be very efficient [5] [6] if each such group is physically connected."* – Shazeer et al Google Brain, Mesh-TensorFlow: Deep Learning for Supercomputers.
  - As learning techniques grow in scale, HPC becomes more important.



- HPC cannot compete with the hyperscalers
  - Let's stop trying and start *integrating*
    - *That doesn't mean adopting Cloud as-is*
    - *That doesn't dissolving HPC either*
  - The closer HPC and cloud paradigms get, the better we all are
    - Encourage open source infrastructure
    - Collaborative partnerships
  - Avoid boutique solutions without sacrificing performance