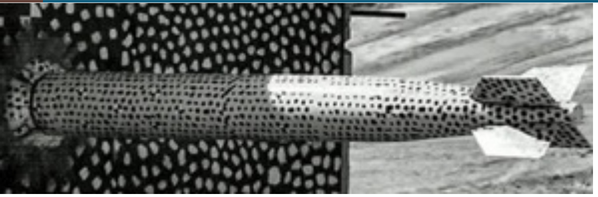
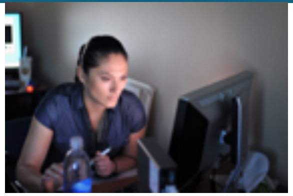
Supercontainers in HPC





PRESENTED BY Andrew J. Younge

Sandia National Laboratories

ajyoung@sandia.gov

E4S Forum – IEEE Cluster 2019

September 23rd, 2019

SAND2019-11335 C

Unclassified Unlimited Release

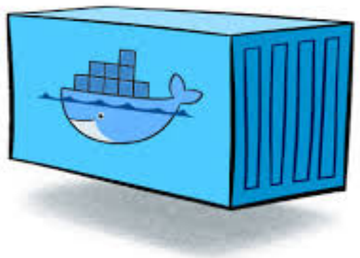
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Motivation

- DOE/NNSA and Sandia have long history of investment in HPC
- 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
- Public cloud computing is often prohibitive
  - Both in cost and security models
- However, HPC is not traditionally as flexible as “the cloud”
  - Shared resource models
  - Static software environments
  - Not always best fit for emerging apps and workflows
- What about Containers?
  - Can we support containers in HPC in the same way as industry?
  - Does this model fit for HPC and emerging workloads across DOE?
  - Can we adapt our programming environments into container images?

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



Initial 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
- Developers specify how to build the environment AND the application
  - Users just import a container and run on target platform
  - Have many containers, but with different manifests for arch, compilers, etc.
  - Not bound to vendor and sysadmin software release cycles
- Performance matters
  - Use mini-apps to “shake out” container implementations on HPC
  - Envision features to support future workflows (ML/DL/in-situ analytics)

Containers in HPC

Wanted Features

- **BYOE - Bring-Your-Own-Environment**
  - Developers define the operating environment and system libraries in which their application runs
- **Composability**
  - Developers have control over how their software environment is composed of modular components as container images
  - Enable reproducible environments that can potentially span different architectures
- **Portability**
  - Containers can be rebuilt, layered, or shared across multiple different computing systems
  - Potentially from laptops to clouds to advanced supercomputing resources
- **DevOps**
  - Integrate with revision control systems like Git
  - Include build manifests and container images using container registries

Conflicting Features

- **Overhead**
  - HPC applications cannot incur significant overhead from containers
- **Micro-Services**
  - Micro-services container methodology does not apply to current HPC workloads
  - 1 app/node with multiple processes or threads per container
- **On-node Partitioning**
  - On-node partitioning with cgroups unnecessary
- **Root Operation**
  - Containers allow root-level access control to users
  - Root is a significant security risk for HPC facilities
- **Commodity Networking**
  - Common network control mechanisms are built around commodity networking (TCP/IP)
  - Supercomputers utilize custom interconnects w/ OS kernel bypass operations

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

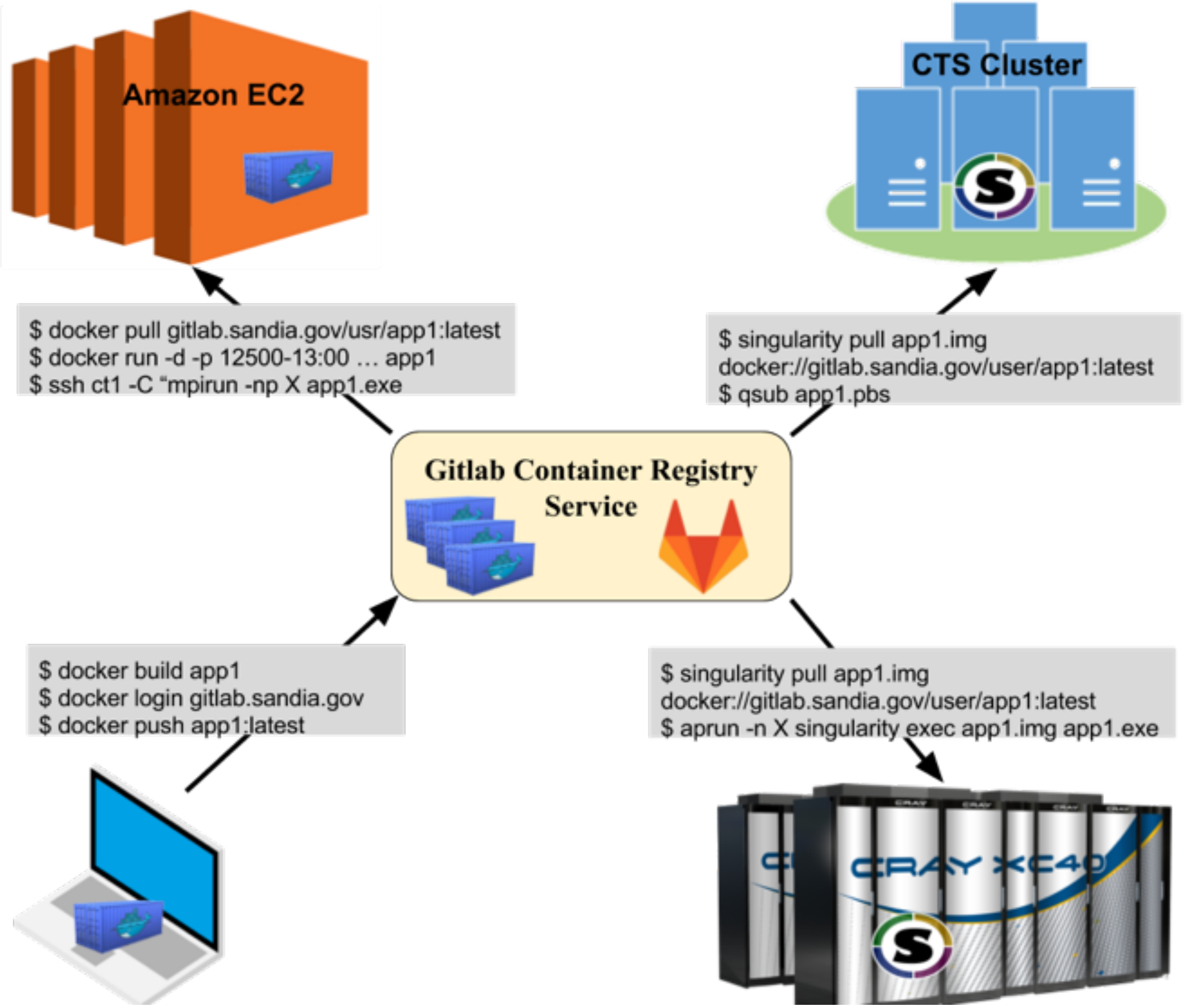
ECP Supercontainers

- Joint DOE effort - Sandia, LANL, LBNL, LLNL, U. of Oregon
- Ensure container runtimes will be scalable, interoperable, and well integrated across DOE
  - Enable container deployments from laptops to Exascale
  - Assist ECP 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

EXASCALE COMPUTING PROJECT

Container DevOps

- Impractical to use large-scale supercomputers for DevOps and testing
  - HPC resources have long batch queues
  - Large effort to port to each new machine
- Deployment portability with containers
  - Develop Docker containers on your laptop or workstation
  - Leverage registry services
  - Import container to target deployment
  - Integrate with vendor libs (via ABI compat)
  - Leverage local resource manager (SLURM)
  - Separate networks maintain separate registries



Singularity Runtime at Sandia

- Singularity fit for current needs
  - OSS, publicly available, support backed by Sylabs
  - Simple image plan, support for many HPC systems
  - Docker image support
  - Multiple architectures
    - X86_64, ARM64, POWER9
  - Initial GPU support
    - `singualrity exec --nv app1.simg /opt/bin/app`
  - Large community involvement
- Singularity deployed across Sandia
  - CTS-1 and TLCC clusters
  - Astra – First Petascale ARM supercomputer
- Ongoing collaboration with Sylabs

Sylabs Remote Container Builder

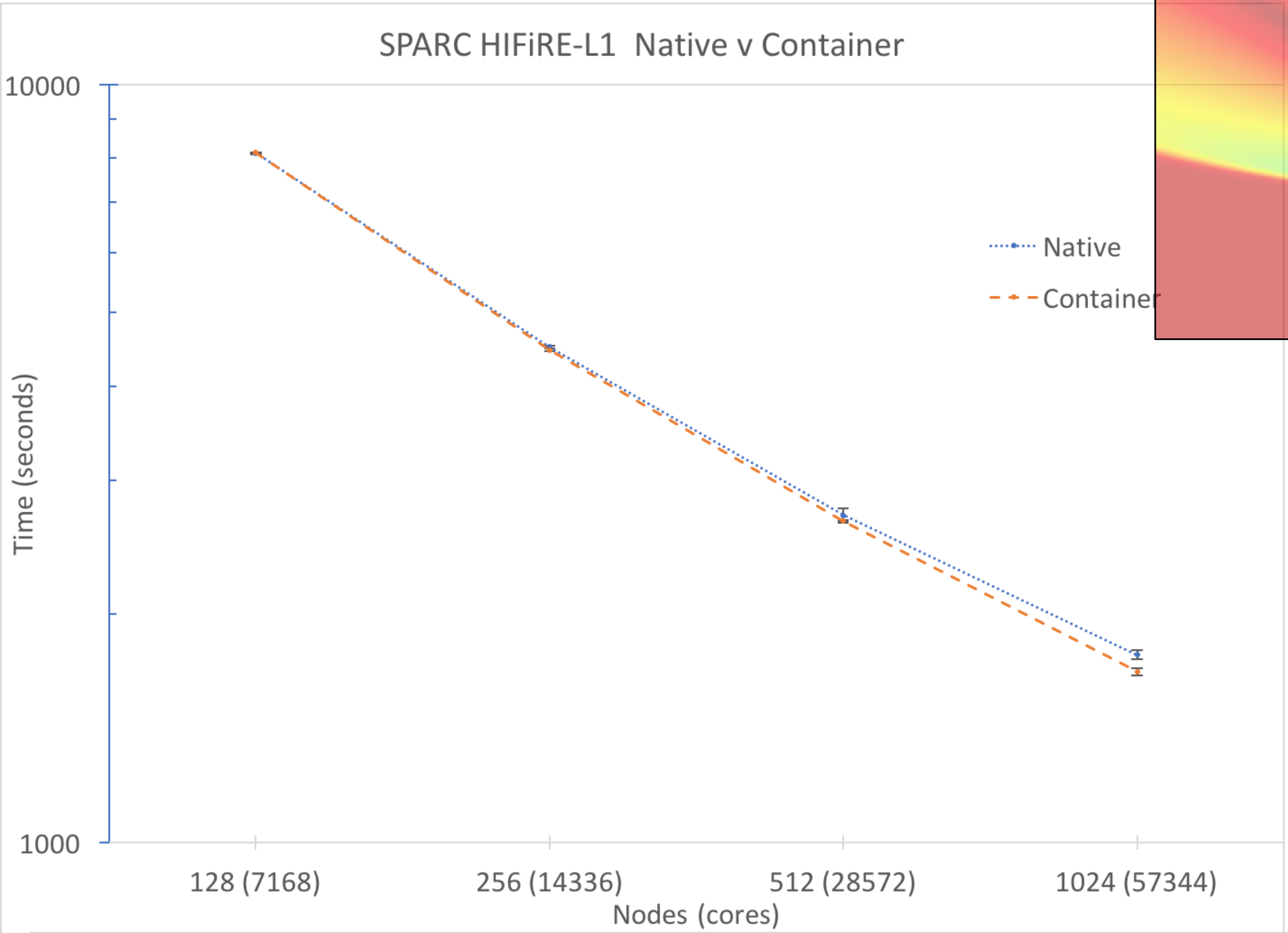
Separated container build workstations for various architectures

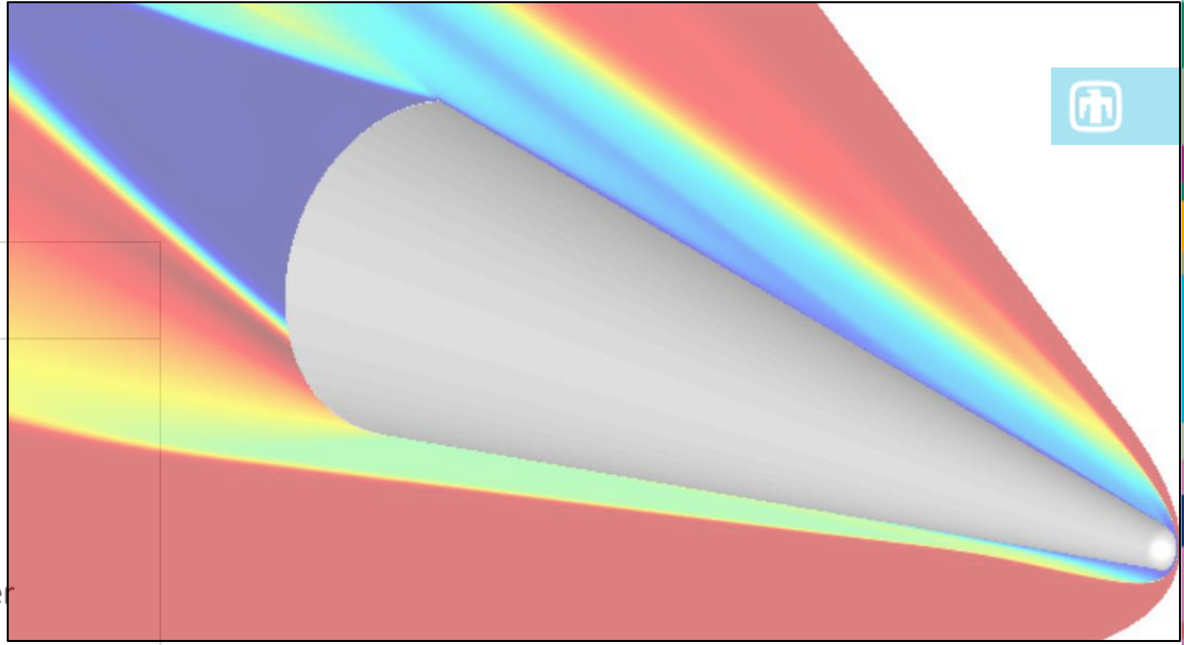
- Can’t use a laptop to build ARM64 or POWER9 CPUs
- Inflexible, clunky, isolated

Working with Sylabs on new solution – Remote Builder

- Enables users to build for alternate architectures:
  - Ex. build AARCH64 container from AMD64 workstation
  - Can be used as part of CI/CD process (GitHub, etc.)
- Builds run natively on alternate architecture, giving great performance
- Centralized resource pool:
  - Lowers TCO by decreasing the need for workstations of multiple architectures
- Enables users to build containers without privilege
- Native integration with Singularity CLI
- Can be deployed on-premise via Singularity Enterprise
  - More info: https://sylabs.io/singularity-enterprise/

Case Study 1: SNL ATDM App



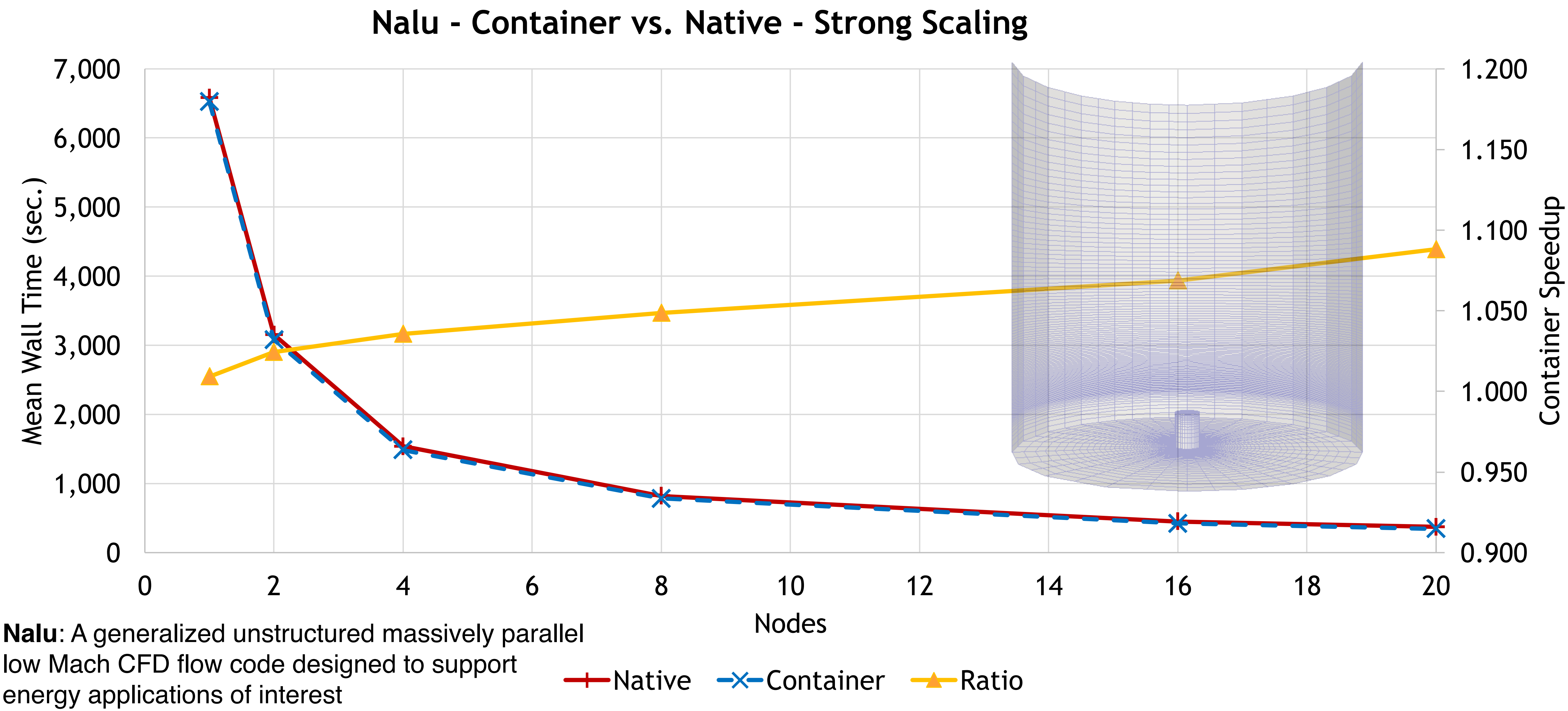


Points:

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

Takeaway: Production HPC Applications can be deployed with containers

Case Study 2: Nalu CFD



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

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

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)

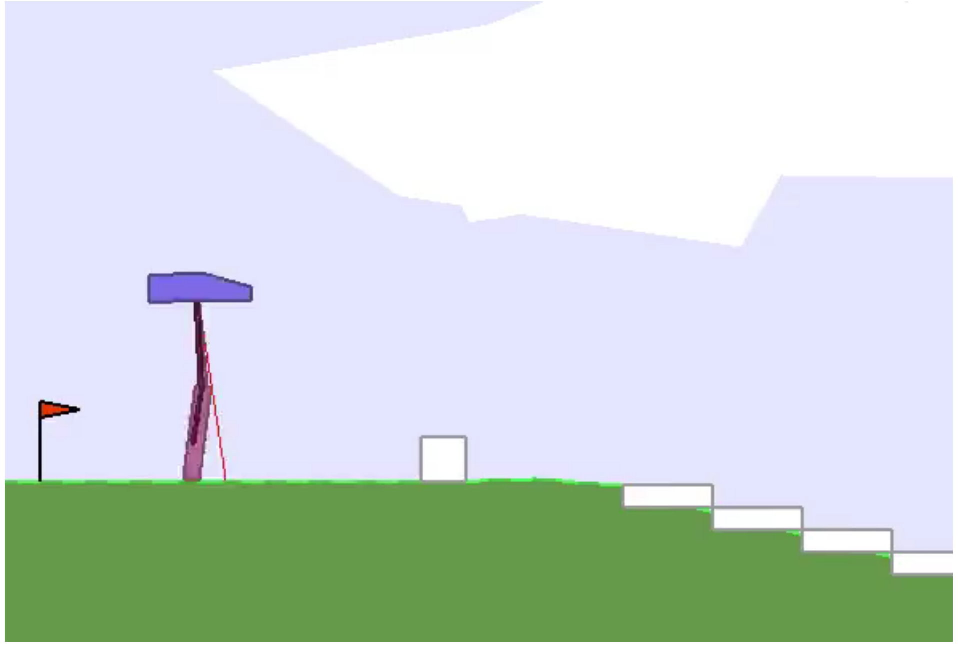
Emerging workloads on HPC with Containers

- Support emerging AI/ML/DL frameworks on HPC
  - Containers useful to adapt ML software to HPC
  - Already supported and heavily utilized in industry
- Extreme-scale Scientific Software Stack (**E4S**)
  - Includes TensorFlow & Pytorch in container image
  - Find Sameer Shende for more details! – e4s.io
- Working with DOE app teams to deploy custom ML tools in containers
- Investigating scalability challenges and opportunities

E4S TensorFlow PYTORCH

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
- We are using Astra for scaling of ASTool[1]
  - 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 to complete
- Next steps:
  - Eliminate major software performance inefficiencies and bottlenecks
  - Apply lessons-learned to our own multi-objective optimization problem

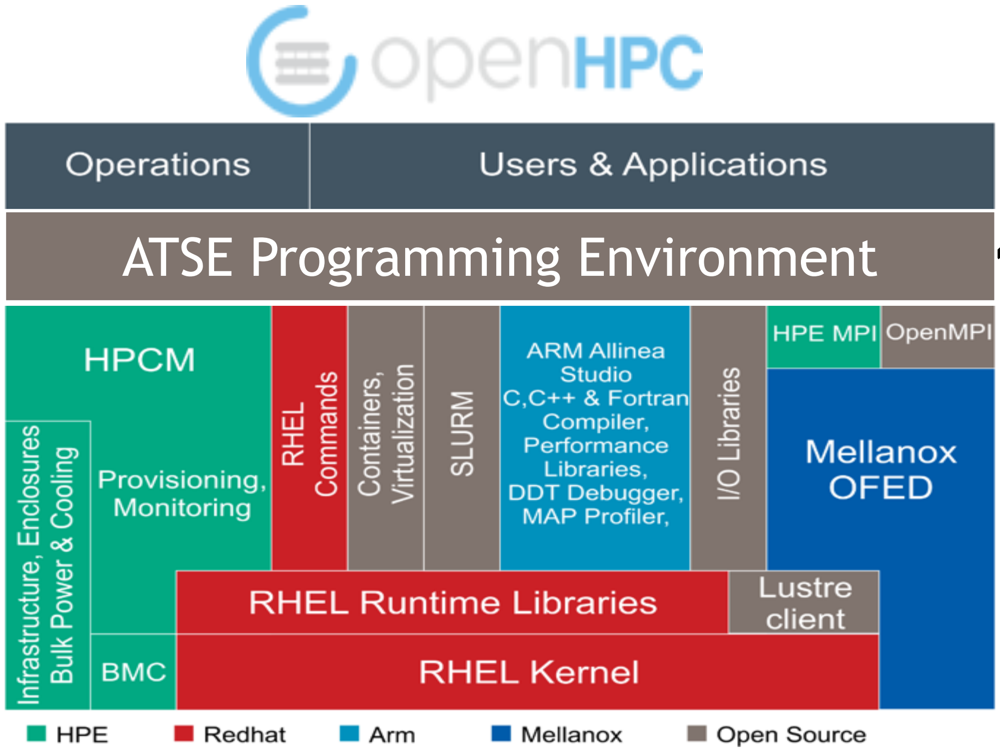


Credit: https://designrl.github.io/

1. https://github.com/hardmaru/astool

Takeaway: Containers can support Emerging HPC workloads like Reinforcement Learning

Case Study 4: Containerized ATSE



On workstation where user has root:

1. `docker build -t "gitlab.sandia.gov/atse/astra:1.2.2" .`
2. `docker push gitlab.sandia.gov/atse/astra:1.2.2`

Sandia GitLab Container Registry

3. `singularity build atse-astra-1.2.2.simg docker://gitlab.sandia.gov/atse/astra:1.2.2`
4. `salloc -N 2048 -t 4:00:00`
   `mpirun -np 114688 -npernode 56 singularity exec atse-astra-1.2.2.simg /home/user/myapp`

Takeaway: Deployed & validated upgraded ATSE in a container before machine upgrade

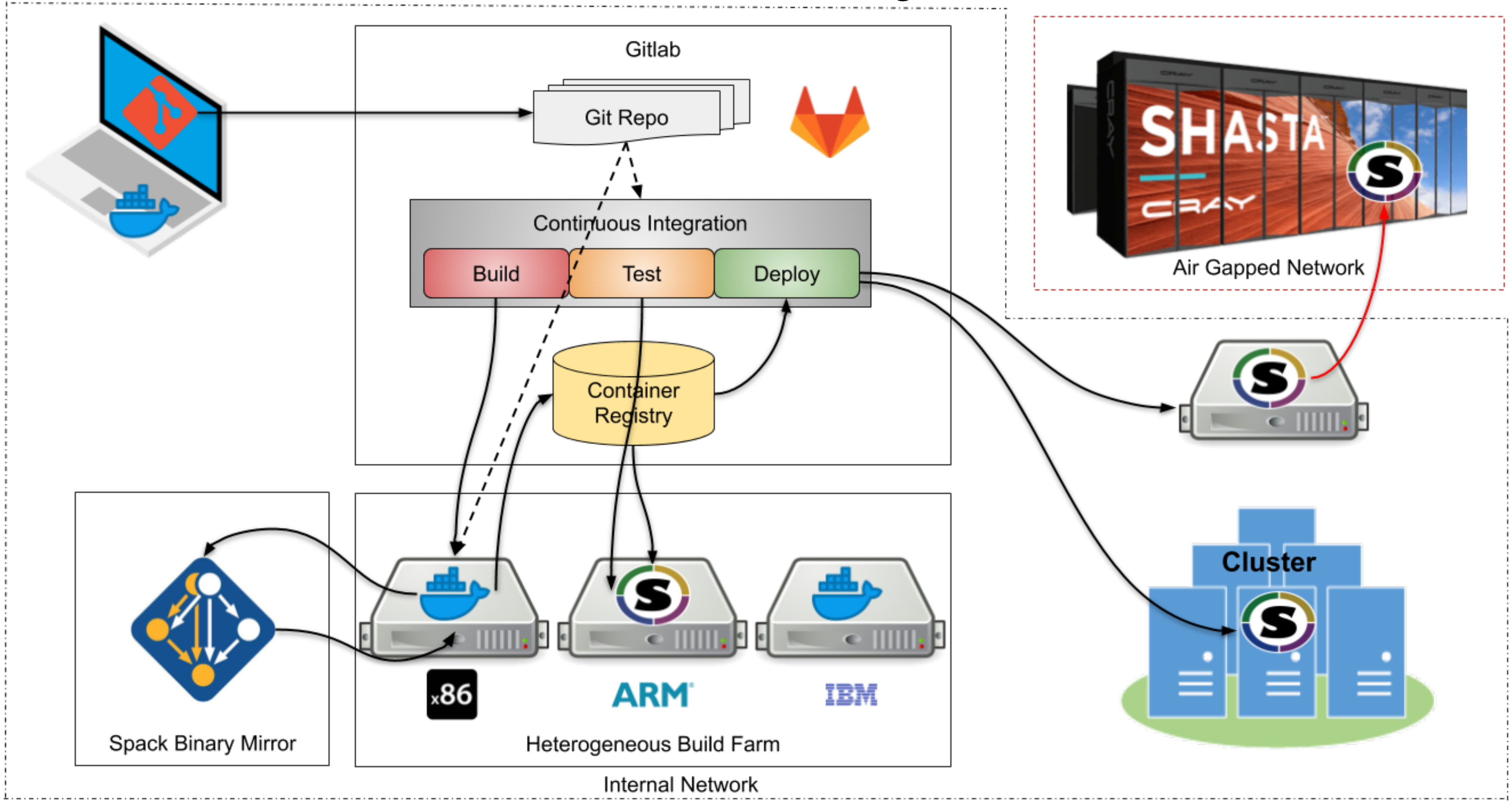
Containers on Secure Networks

- SNL containers are primarily built on unclassified systems then moved to air gapped networks via automated transfers
- Cybersecurity approvals in place to run containers on all networks
- Security controls used in running containers on HPC systems
  - Working to validate software compliance
- Automated Transfer Services to air gapped networks
- Challenges of automated transfers
  - Size – 5GB-10GB are ideal
  - Integrity – md5 is enough
  - Transfer policies – executables, code, etc.

Containers will fully work with automated transfers for use in air gapped networks

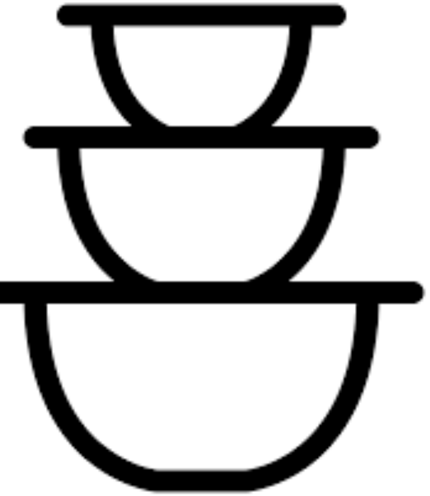
Future Containerized CI Pipeline

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



Container Takeaways (aka tupperware?)

- Use Docker to build manifests to assemble full app suites from scratch
    - Developers specify base OS, configuration, TPLs, compiler installs, etc
    - Leverage base or intermediate container images (eg: TOSS RPMs in a container)
  - Leverage container registry services for storing images
  - Import/flatten Docker images into Singularity & run on HPC resources
    - Also works for Charliecloud compatibility
- Advantages
  - Simplify deployment to analysts (just run this container image)
  - Simplify new developer uptake (just develop FROM my base container image)
  - Decouple development from software release cycle issues
  - Reproducibility has a new hope?
- Caveats
  - ABI compatibility with MPI an ongoing issue
  - Focus is on x86_64 images, alternative archs require more work
    - Can't build an ARM64 container image from my Mac laptop w/ x86_64
  - Containers are an option in HPC, not a mandate



Conclusion

- Demonstrated value of container models in HPC
  - Deployments in testbeds to production HPC
  - Initial performance is promising
  - Modern DevOps approach with containers
  - Deployed on several Sandia systems
- ECP Supercontainers
  - Enable containers at Exascale
  - Embrace software diversity while insuring interoperability
  - Simplify HPC application deployment
  - Enable next-gen computing ecosystems
- Containers can increase software flexibility in HPC

Acknowledgements:
Kevin Pedretti (1423)
Anthony Agelastos (9326)
Si Hammond (1422)
Doug Pase (9326)
Aron Warren (9327)
Stephen Olivier (1423)
Justin Lamb (9326)
Erik Illescas (9327)
Ron Brightwell (1423)

Collaborators:
Shane Canon (LBNL/NERSC)
Todd Gamblin (LLNL)
Reid Priedhorsky (LANL)
Sameer Shende (Oregon)

Thanks!

ajyoung@sandia.gov

Want to learn more about containers?

Attend the Container Tutorial @ SC19

Interested in helping & collaborating?

Students, Postdocs, Collaborators... email!

CANOPIE-HPC WORKSHOP

Containers and New Orchestration Paradigms for Isolated Environments in HPC

canopie-hpc.org

- In coordination with Supercomputing 2019 (SC19) in Denver
- Proceedings published in IEEE TCHPC
- Submission Deadline: Monday, September 2nd, 2019
- Conference Date: Monday, November 18th, 2019
- SC19 workshop dedicated to containers & software environments

Backup Slides