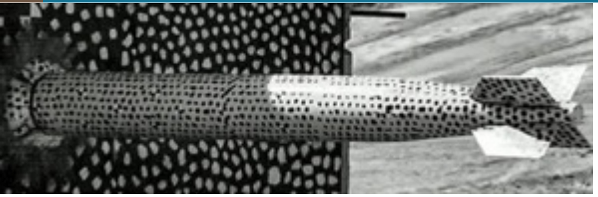
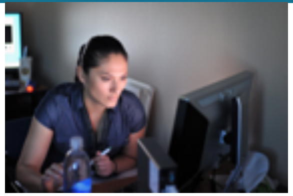
Containers in HPC





PRESENTED BY Andrew J. Younge

Sandia National Laboratories

ajyoung@sandia.gov

Workshop on NSF and DOE High Performance Computing Tools

July 11th, 2019

SAND2019-7980 C

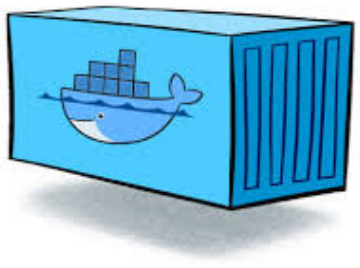
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Motivation

- DOE/NNSA and Sandia have long history of investment in HPC
- Mission workloads computational requirements demand scale
  - Tightly coupled BSP simulation codes eg: MPI
  - Extensive computing capacity - CTS cluster resources
  - Intermediate computing capability – ATS advanced supercomputing
- Public cloud computing is often prohibitive for Sandia
  - 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?

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
- Want to manage permutations of architectures and compilers
  - x86 & KNL, ARMv8, POWER9, etc.
  - Intel, GCC, LLVM, etc
- 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

- **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
- **Version Control Integration**
  - Containers integrate with revision control systems like Git
  - Include not only build manifests but also with complete container images using container registries like Docker Hub

Container features not wanted in HPC

- **Overhead**
  - HPC applications cannot incur significant overhead from containers
- **Micro-Services**
  - Micro-services container methodology does not apply to HPC workloads
  - 1 application per node with multiple processes or threads per container
- **On-node Partitioning**
  - On-node partitioning with cgroups is not necessary (yet?)
- **Root Operation**
  - Containers allow root-level access control to users
  - On supercomputers this is unnecessary and a significant security risk for facilities
- **Commodity Networking**
  - Containers and their 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

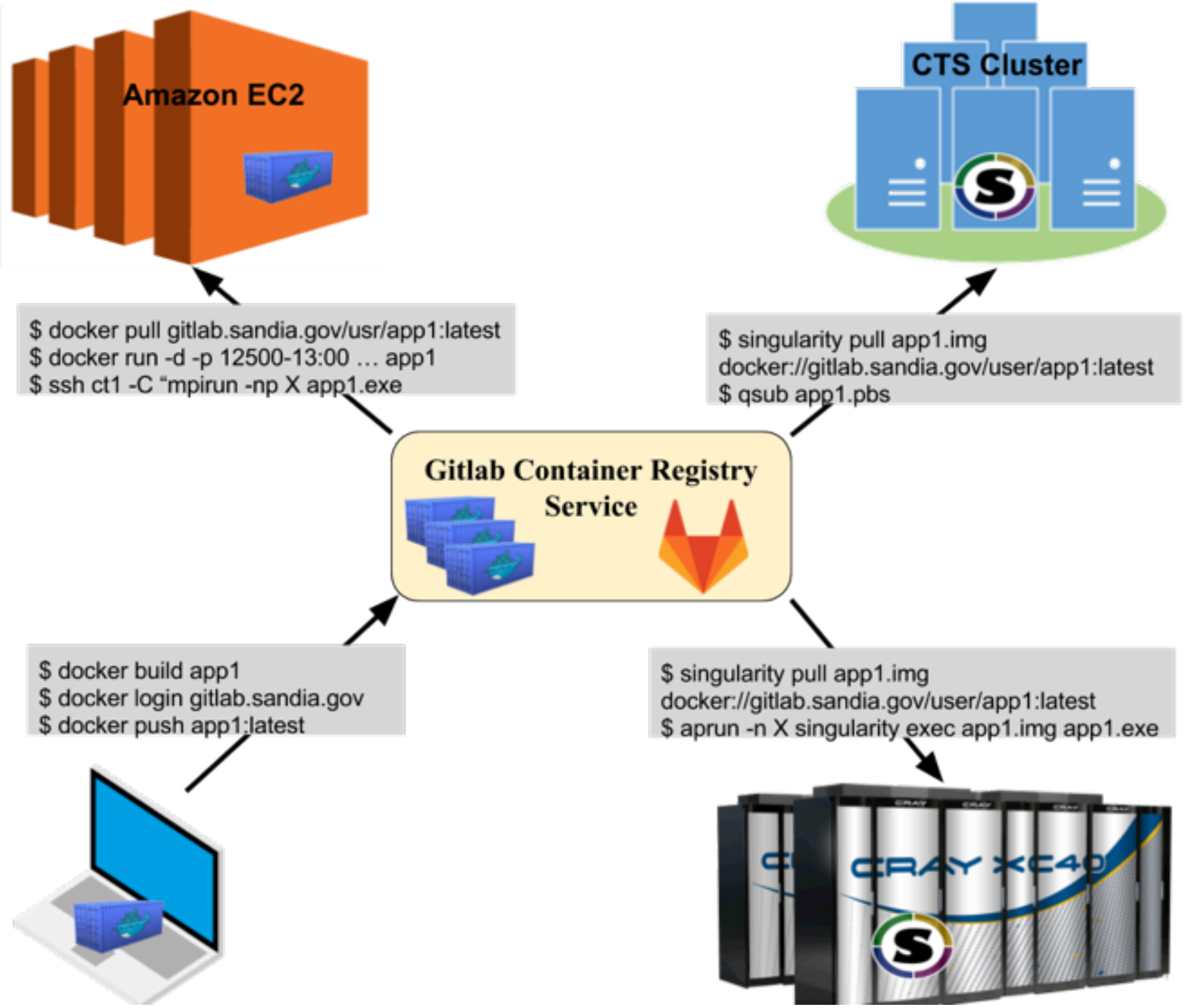
- All 3 HPC container runtimes are usable in HPC today
- Each runtime offers different designs and OS mechanisms
  - Storage & mgmt of images
  - User, PID, Mount namespaces
  - Security models
  - OCI vs Docker vs Singularity images
  - Image signing, validation, registries, etc

Singularity Runtime at Sandia

- Singularity best 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 at Sandia
  - CTS-1 and TLCC clusters
  - Astra – First Petascale ARM supercomputer
- Ongoing collaboration with Sylabs

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



Example Muelu Dockerfile

```
FROM ajyounge/dev-tpl

WORKDIR /build/trilinos

# Download Trilinos
COPY do-configure /build/trilinos/
RUN wget -nv https://trilinos.org/...\
    /files/trilinos-12.8.1-Source.tar.gz \
    -O /build/trilinos/trilinos.tar.gz

# Extract Trilinos source file
RUN tar xf /build/trilinos/trilinos.tar.gz
RUN mv /build/trilinos/trilinos-12.8.1-Source \
    /build/trilinos/trilinos
RUN mkdir /build/trilinos/trilinos-build

# Compile Trilinos
RUN /build/trilinos/do-configure
RUN cd /build/trilinos/trilinos-build && \
    make -j 3

# Link Muelu tutorial
RUN ln -s /build/trilinos/trilinos-build/pkgs/...\
        /opt/muelu-tutorial
WORKDIR /opt/muelu-tutorial
```

- Example Trilinos container build
  - Muelu Tutorial
  - Trilinos on version 12.8.1
- Uses ajyounge/dev-tpl as base container
  - Contains necessary third party libraries for building
  - Parmetis, NetCDF, compilers, etc.
- This is a simple version, more complex Dockerfile allows various features and versions to be selected

A Tale of Two Systems

Run a series of benchmarks and Sandia mini-apps to evaluate each system.
Use *same* container images, built using Docker & deployed to both Volta and Amazon cloud.

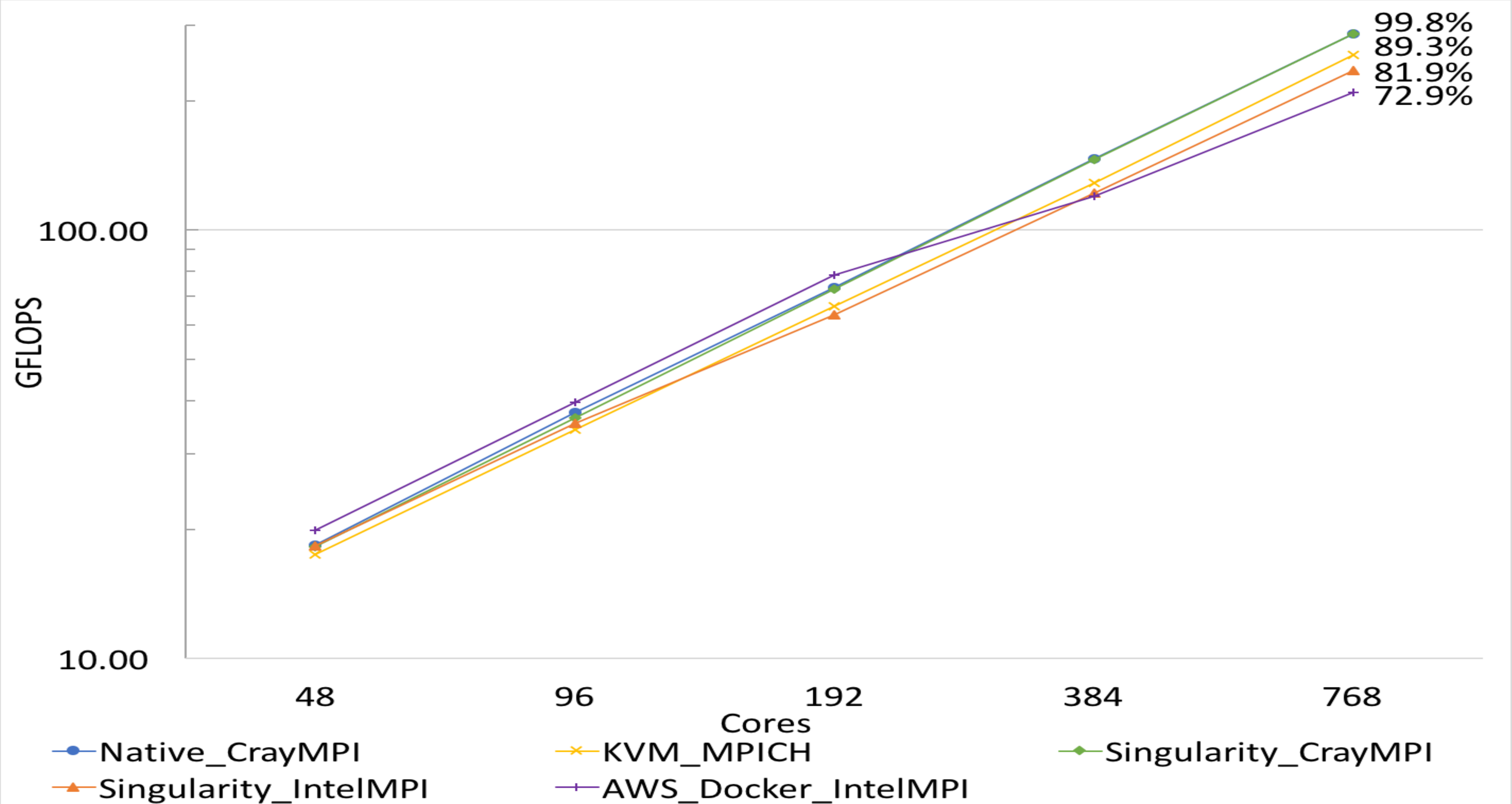
Volta

- Cray XC30 HPC system
- 56 nodes:
  - 2x Intel ”IvyBridge” E5-2695v2 CPUs
  - 24 cores total, 2.4Ghz
  - 64GB DDR3 RAM
- Cray Aries Interconnect
- Shared DVS filesystem
- Cray CNL ver. 5.2.UP04
  - 3.0.101 kernel
  - ***Running custom Singularity***
- 32 nodes used to keep equal core count
- NNSA ASC testbed at Sandia

Amazon EC2

- Common public cloud service from AWS
- 48 c3.8xlarge instances:
  - 2x Intel “IvyBridge” E5-2680 CPUs
  - 16 cores total 32 vCPUs (HT), 2.8Ghz
  - 10 core chip (2 cores reserved by AWS)
  - 60 GB RAM
- 10 Gb Ethernet network w/ SR-IOV
- 2x320 SSD EBS storage per node
- RHEL7 compute image
  - ***Running Docker 1.19***
- Run in dedicated host mode
- 48 node virtual cluster = $176.64/hour

HPCG Weak Scaling (log log scale)



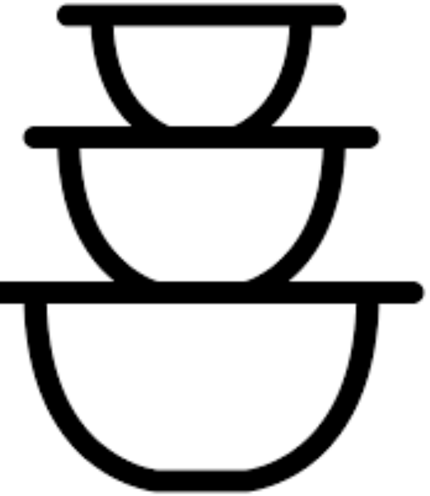
Containers on Secure Networks

- 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

Container Takeaways

- Use Docker to build a 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
- 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



ECP Supercontainers Project

- Joint effort across 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

Supercontainer R&D Activities

- Containers must work at Exascale!
  - Embrace architectural diversity

R&D Topics:

- Advanced Container Runtimes
  - Efficient container launch
  - Comparison studies
- Optimized Images
  - E4S environment
  - Use Spack!
  - Vendor images
- Expand interoperability
  - Decrease reliance on MPI ABI compatibility
  - Foster community standards
- Other opportunities
  - Service container orchestration
  - Workflow ensemble support
  - Reproducibility?

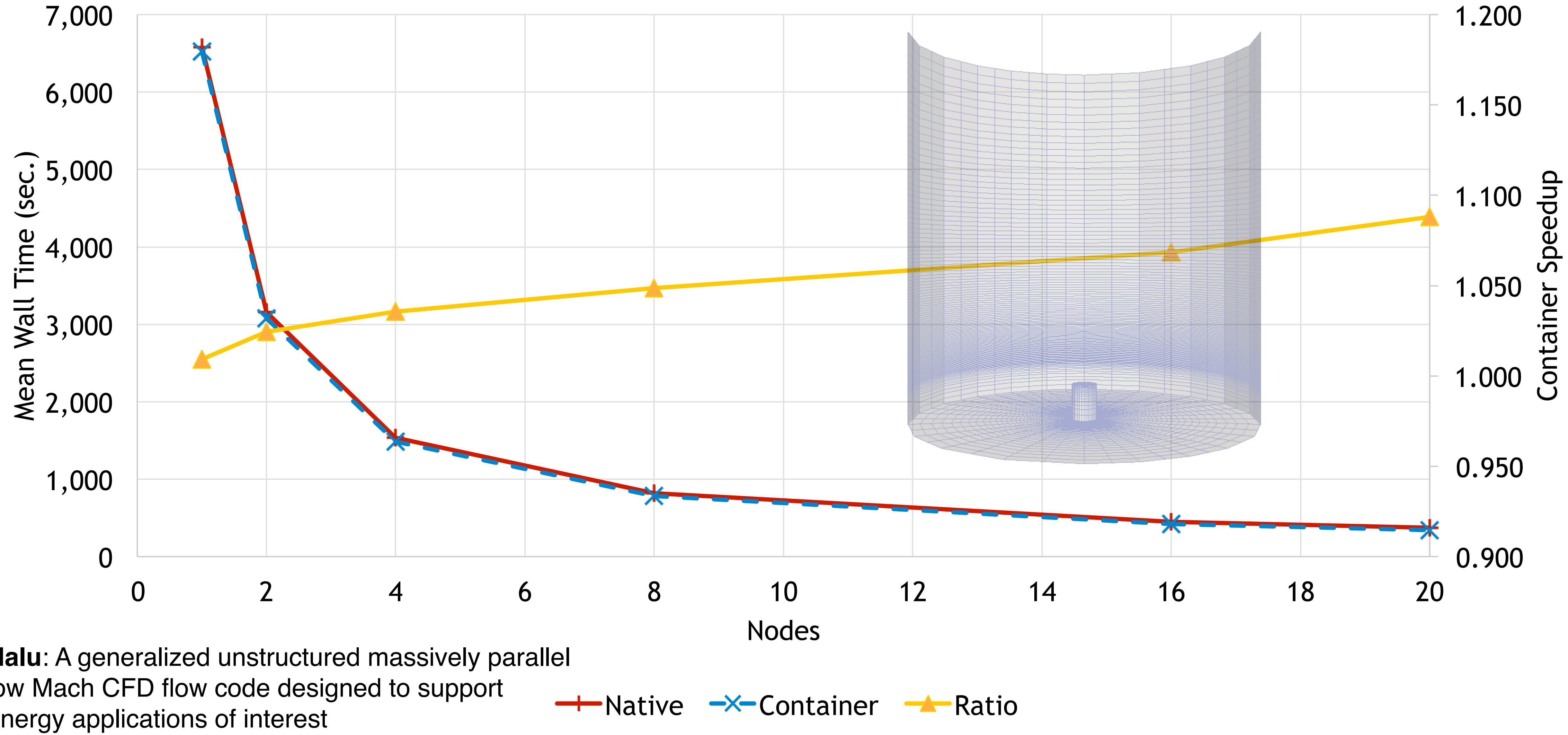
Spack

E4S

Containerized Application Support

- Interface with key ECP ST and AD app development areas
- Advise and support the container usage models necessary for deploying first Exascale apps
- Initiate deep-dive sessions with app groups
  - Use provided base images
  - Enable Spack in container images
  - Tuned & supported by facilities via HI
- Leverage DOE Gitlab Continuous Integration mechanisms
  - Integrate containers into current CI plan

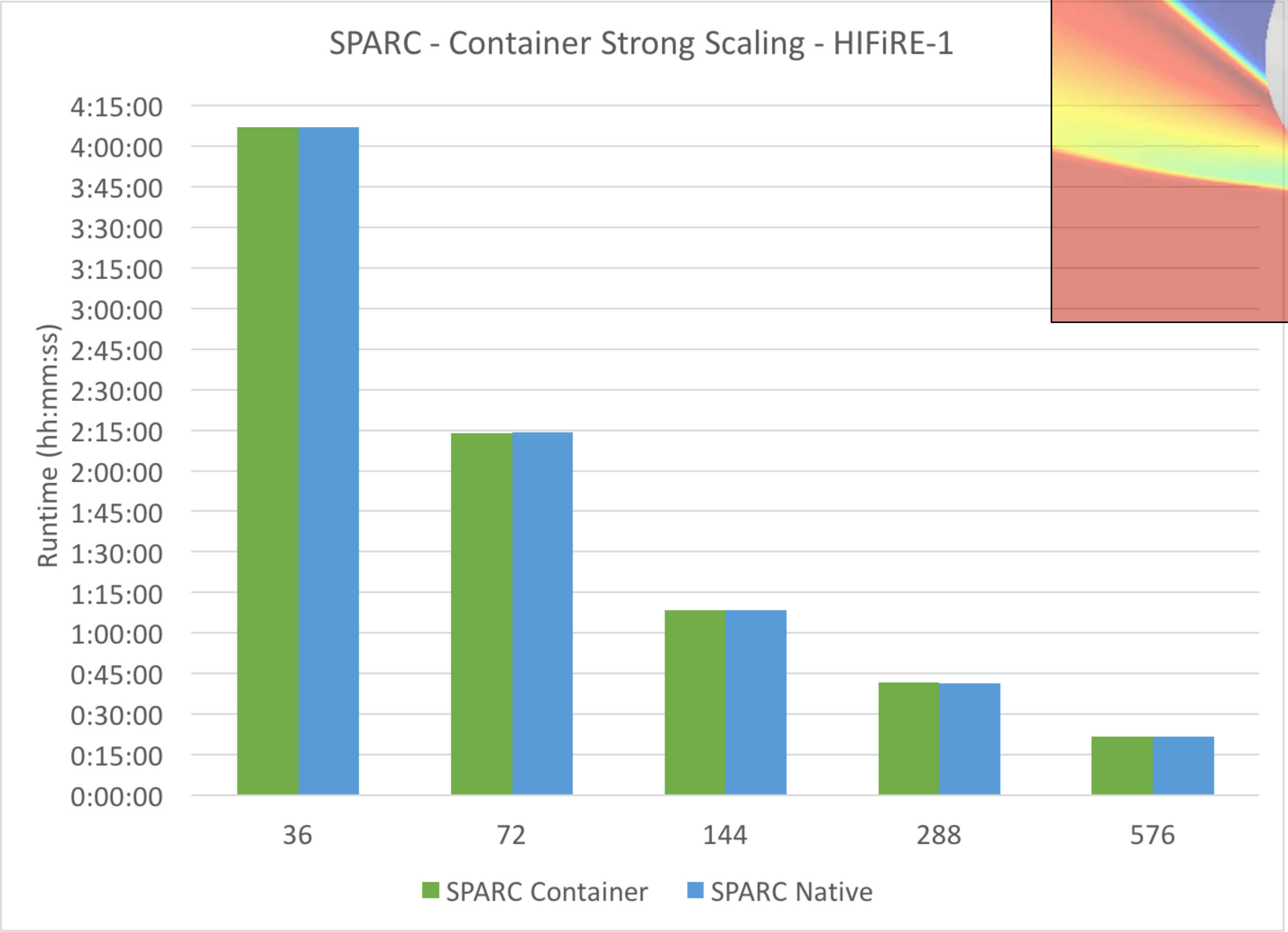
Nalu - Container vs. Native - Strong Scaling

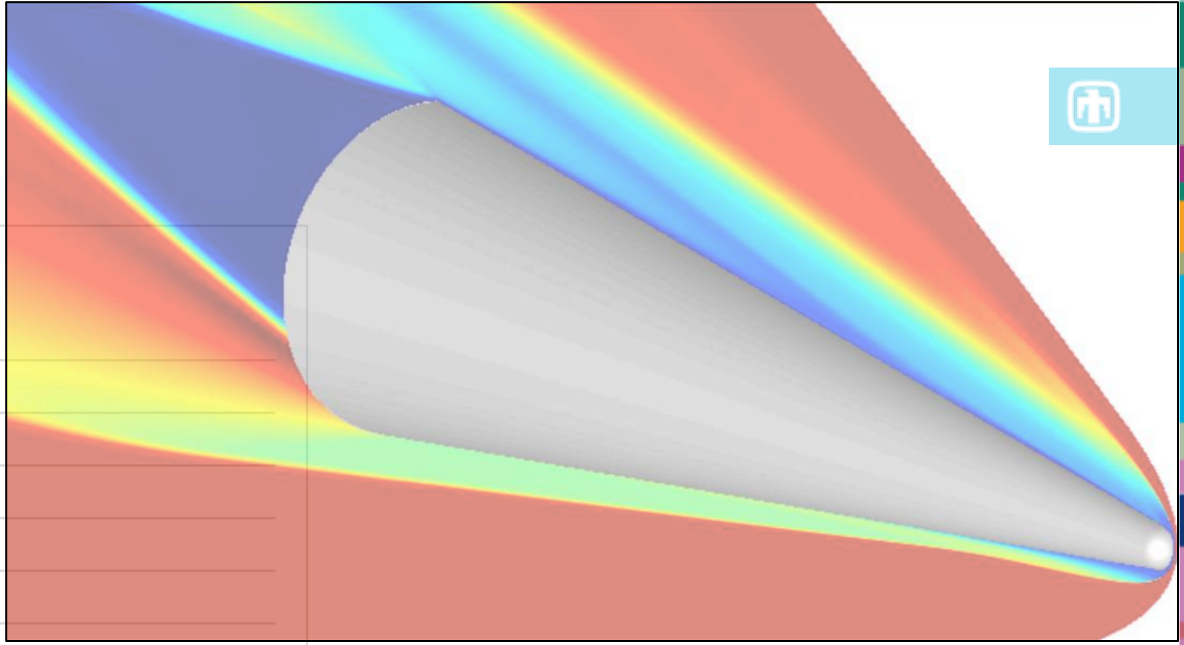


Nalu: A generalized unstructured massively parallel low Mach CFD flow code designed to support energy applications of interest

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

SNL ATDM Mission App





Points:

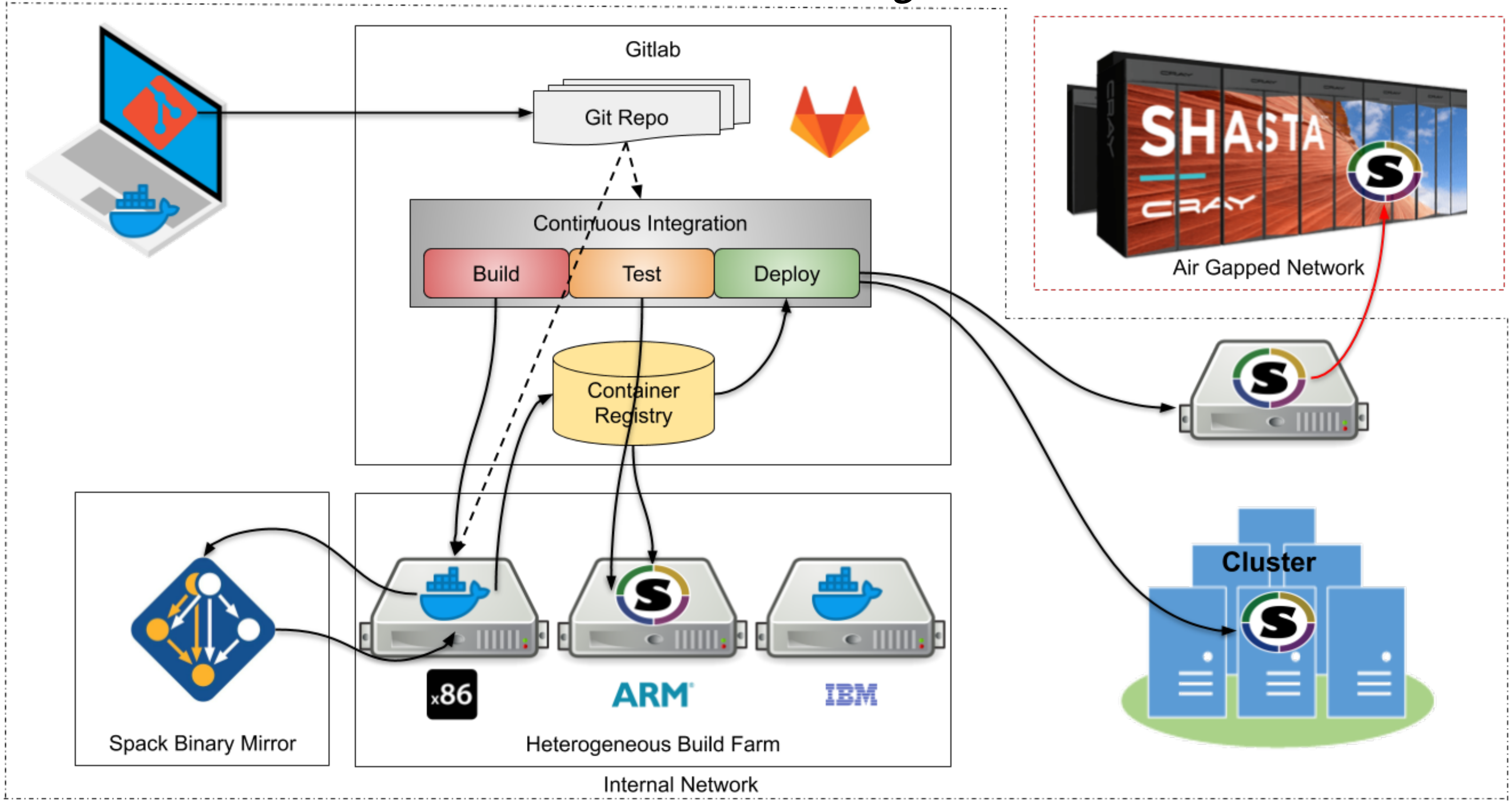
- Supporting SPARC containerized build & deployment
- Deployed on Sandia CTS-1
- Near-native performance using a container
- Testing HIFiRE-1 Experiment (MacLean et al. 2008)

Emerging workloads on HPC with Containers

- Support merging AI/ML/DL frameworks on HPC
  - Containers may be 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
- Working with DOE app teams to deploy custom ML tools in containers
- Investigating scalability challenges and opportunities

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.*



Training Education & Support

- Containers involve new software deployment methodology
- Training and education is needed to help ECP community to best utilize new functionality
- Technical Reports
  - Best Practices for building software using containers
  - Taxonomy survey to survey current state of the practice
- Training sessions
  - International Supercomputing Conference 2019
  - IEEE/ACM Supercomputing 2019
  - ECP All-Hands Meeting
- Provide single source of knowledge for groups interested in containers

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 CTS systems
- ECP Supercontainers Project
  - Enable containers Exascale
  - Embrace software diversity while insuring interoperability
  - Simplify HPC applciation deployment
- Containers can increase software flexibility in HPC

EXASCALE COMPUTING PROJECT

ASC™

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)

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

Thanks!

ajyoung@sandia.gov

Want to learn more about containers?

Attend the Container Tutorial @ SC19

Interested in helping & collaborating?

Students, Postdocs, Collaborators...

Backup Slides

THE WORLD'S FIRST PETASCALE ARM SUPERCOMPUTER

Sandia National Laboratories

ASTRA

"Per aspera ad astra"

ATSE stack

2.3 PFLOPs peak
>5000 TX2 ARM CPUs, ~150k cores
885 TB/s memory bandwidth peak
332 TB memory
1.2 MW

- ATSE – Advanced Tri-lab Software Environment
- Supports Singularity container runtime
- Building ATSE container images
- Developing Pytorch ARM containers

U.S. DEPARTMENT OF ENERGY NNSA