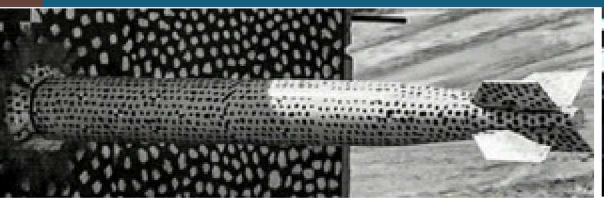
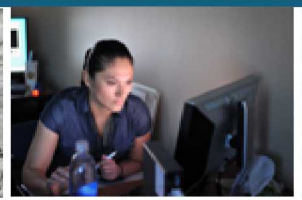
SAND2019-7351C

Containers in HPC and Beyond





PRESENTED BY

Andrew J. Younge

Sandia National Laboratories

ajyoung@sandia.gov

CEA/NNSA Meeting

June 26th, 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.

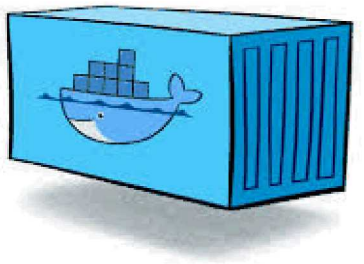
Motivation

- Need HPC infrastructure to better support diverse software stacks
- Utilize, modify, adapt industry tools to HPC only when applicable
- Provide increased software flexibility to HPC
- Sandia has long history of OSR & virtualization research in HPC
  - Hobbes, Kitten, Palacios, KVM, …
  - Latest efforts also focus on containers & containerized workloads

- Can HPC support containers in the same way as industry?
- Does this model fit for HPC and emerging workloads at Sandia?

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



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

Initial Container Vision

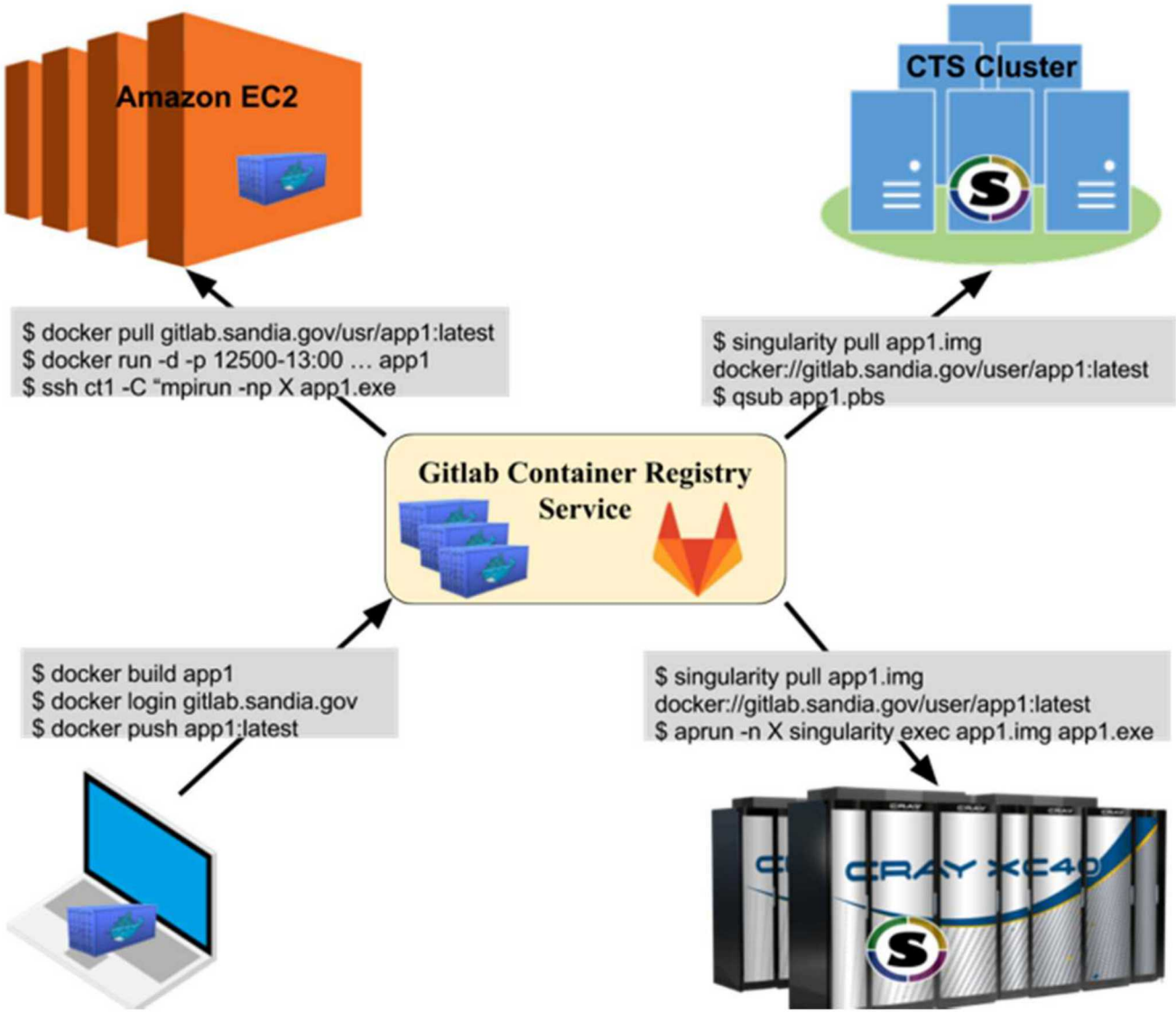
- Support HPC software development and testing on laptops/workstations
  - Create working container builds that can run on supercomputers
  - Minimize dev time on supercomputers
- Developers specify how to build the environment AND the application
  - Users just import a container and run on target platform
  - 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)

Singularity Containers

- Docker is not good fit for running HPC workloads
  - Security issues, no HPC integration
- Many different container runtimes
  - Docker, Shifter, Singularity, Charliecloud, etc. etc.
- 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, ARM, POWER)
  - Large community involvement

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



A Tale of Two Systems

Volta

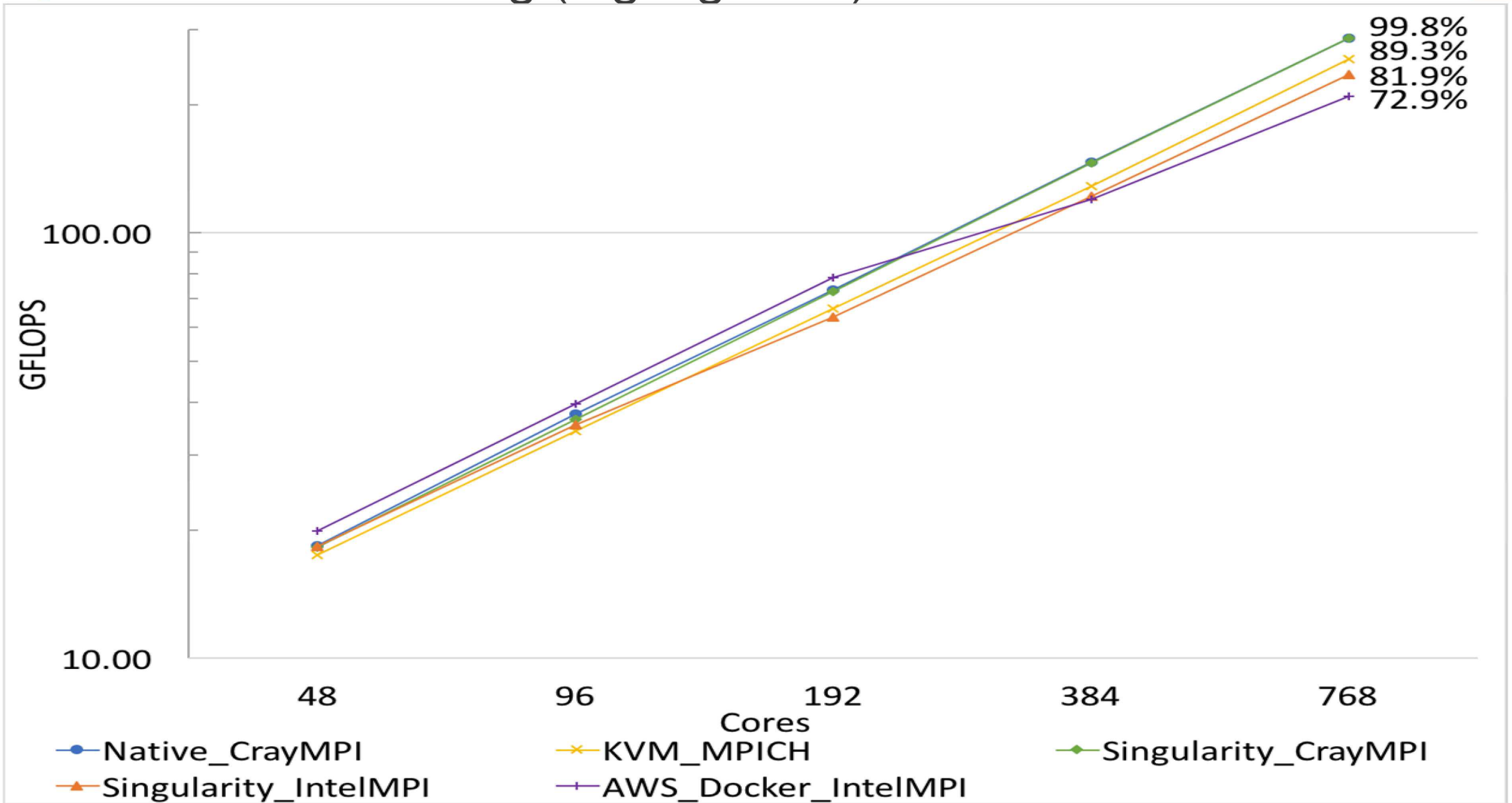
- Cray XC30 HPC system
- 56 nodes:
  - 2x Intel ”IvyBridge” E5-2695v2 CPUs
  - 24 cores total, 2.4Ghz
  - 64GB DDR3 RAM
- Cray Aries Interconnect
- Shared DVS filesystem
- Cray CNL ver. 5.2.UP04
  - 3.0.101 kernel
  - ***Running custom Singularity***
- 32 nodes used to keep equal core count
- NNSA ASC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

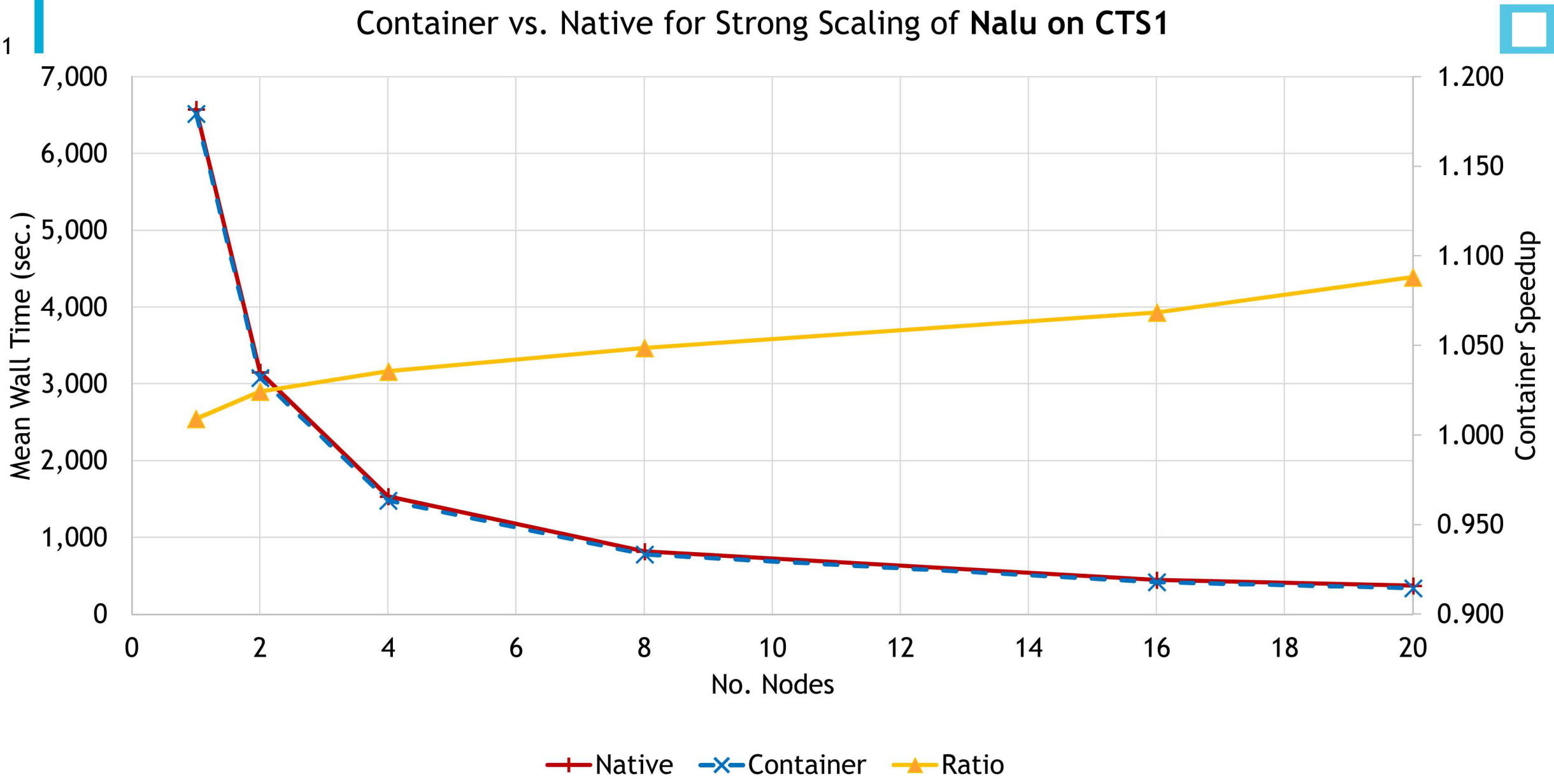
Amazon EC2

- 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

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.

HPCG Weak Scaling (log log scale)





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

Containers on Secure Networks

- 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
  - Availability – who are you competing against?
  - Transfer policies – executables, code, etc.

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

ASTRA

"Per aspera ad astra"

RM SUPERCOMPUTER

885 TB/s memory bandwidth peak
332 TB memory
1.2 MW

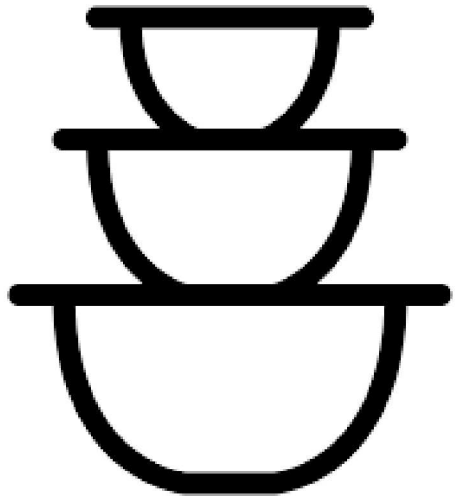
ATSE stack

ATSE - Advanced Tri-lab Software Environment
Supports Singularity container runtime
- Working on Charliecloud

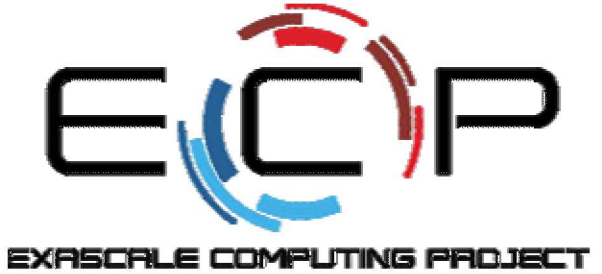
Building ATSE container images
Developing Pytorch ARM containers

Container Takeaways

- Use case today:
  - Use Docker to build a manifests to assemble full application suites from scratch
    - Developers specify TPLs, base OS, configuration, etc
    - Leverage base or intermediate container images (eg: TOSS base, SEMS image, etc)
  - Leverage container registry services for storing images
  - Import/flatten Docker images into Singularity & run on HPC resources
    - Singularity being rolled out across CTS1, TLCC, and Astra
- Advantages:
  - Simplify deployment to analysts (just run this container image)
  - Simplify new developer uptake (just develop FROM my base container image)
  - Decouple development from TPL or vendor 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, U. of Oregon, LLNL
- 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

Efforts with Supercontainers

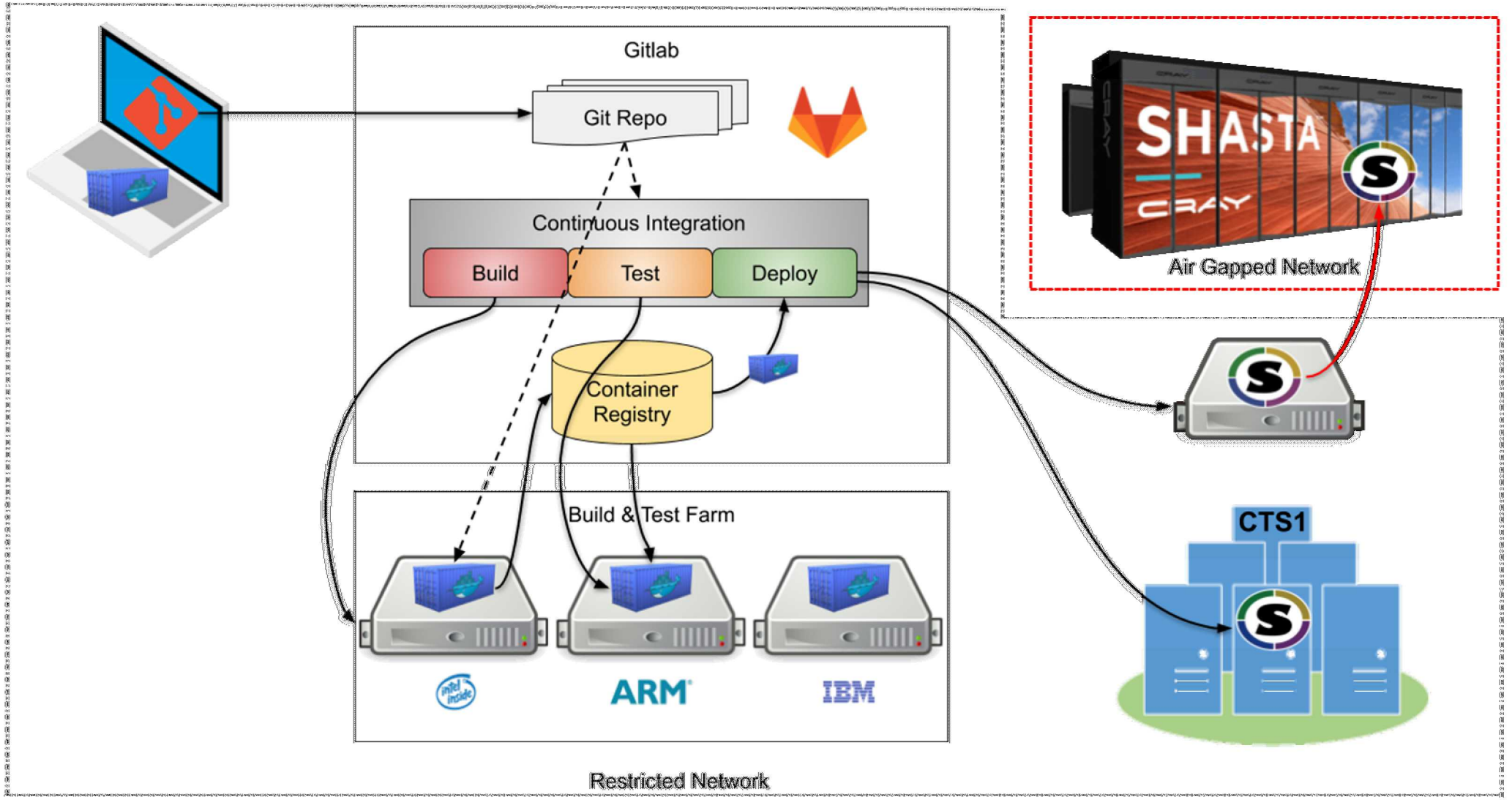
- Containers must work at Exascale
  - DOE ECP efforts are depending on it
  - Embrace architectural diversity
  - Advanced GPU support
- Containerized CI/CD pipeline with Gitlab
- HPC service orchestration
- Build-time optimizations w/ Spack
  - E4S software stack in containers
  - Multi-stage builds, multi-env builds
  - Use Spack pkg mgmt & orchestration
- Further integration with larger community needed
  - Decrease reliance on MPI ABI compatibility
  - Vendor support for base containers images
  - Foster standards that increase reliability
- Workflow ensemble support
- Reproducibility?

Spack

E4S

Future Containerized CI/CD 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 machines*.



Emerging workloads on HPC with Containers

- Need to 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
  - ATDM ML4SS project
  - ECP ExaLEARN project
- Investigating scalability challenges and opportunities

TensorFlow

PYTORCH

Conclusion

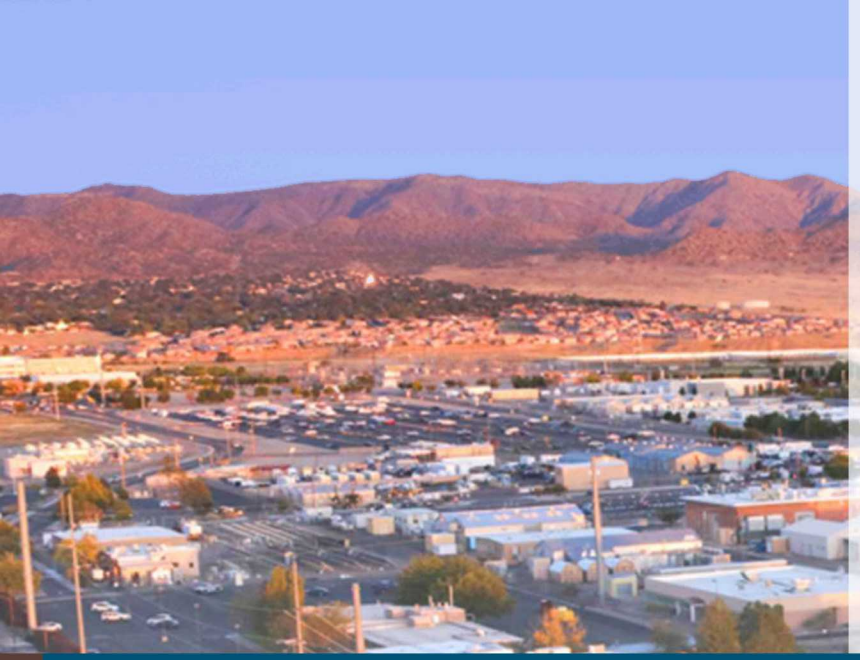
- Demonstrated value of container models in HPC
  - Deployments in testbeds to production HPC
  - Initial performance is promising
  - Modern DevOps approach with containers
  - Deployed on CTS systems
- ECP Supercontainers Project
  - Performance to be validated at Exascale
  - Embrace diversity while insuring interoperability
  - Larger integration with CI pipeline
- Containers 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)
Reid Priedhorsky (LANL)
Sameer Shende (UofOregon)
Todd Gamblin (LLNL)

Questions?

ajyoung@sandia.gov



Backup Slides

Adaptive HPC Infrastructure

- Sandia and DOE/NNSA 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

Container features not wanted in HPC

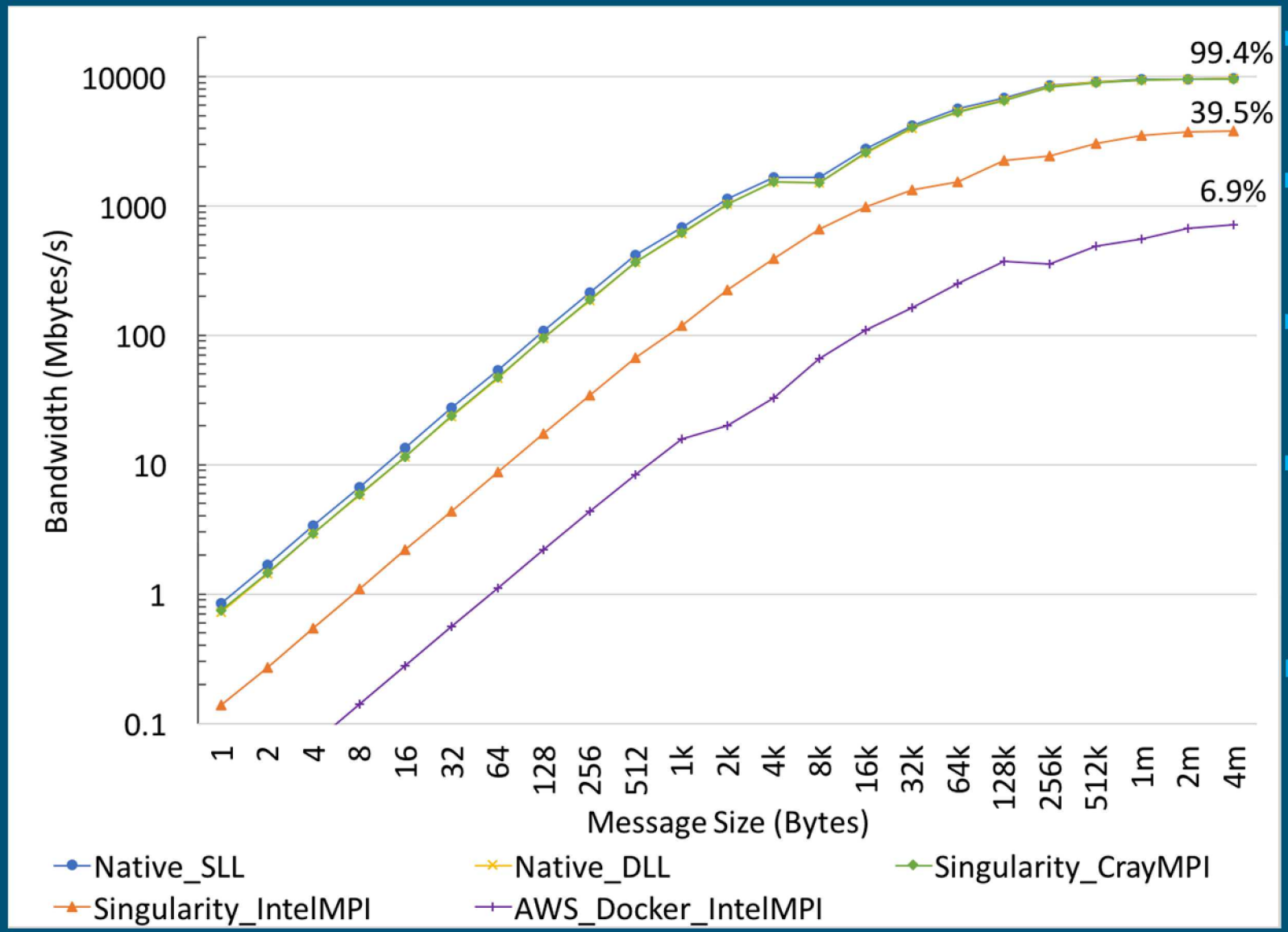
- **Overhead**
  - HPC applications cannot incur significant overhead from containers
- **Micro-Services**
  - Micro-services container methodology does not apply to HPC workloads
  - 1 application per node with multiple processes or threads per container
- **On-node Partitioning**
  - On-node partitioning 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

Example Muelu Dockerfile

```
FROM ajyounge/dev-tpl

WORKDIR /opt/trilinos

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

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

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

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

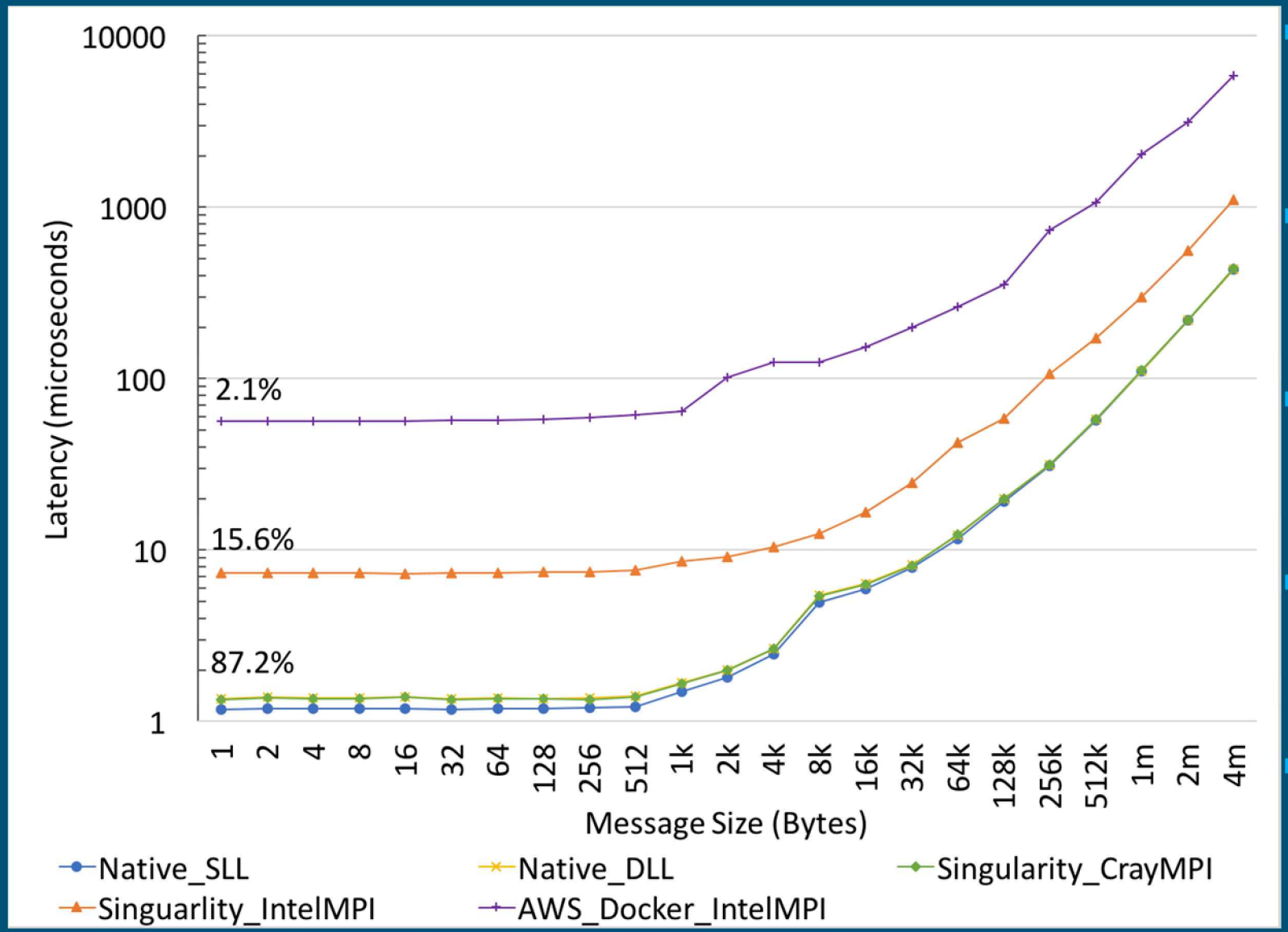
- Example Trilinos container build
  - Muelu Tutorial
  - Trilinos on version 12.8.1
- Uses ajyounge/dev-tpl as base container
  - Contains necessary third party libraries for building
  - PETSc, NetCDF, compilers, etc.
- This is a simple version, more complex Dockerfile allows various features and versions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

IMB PingPong Bandwidth (log scale)



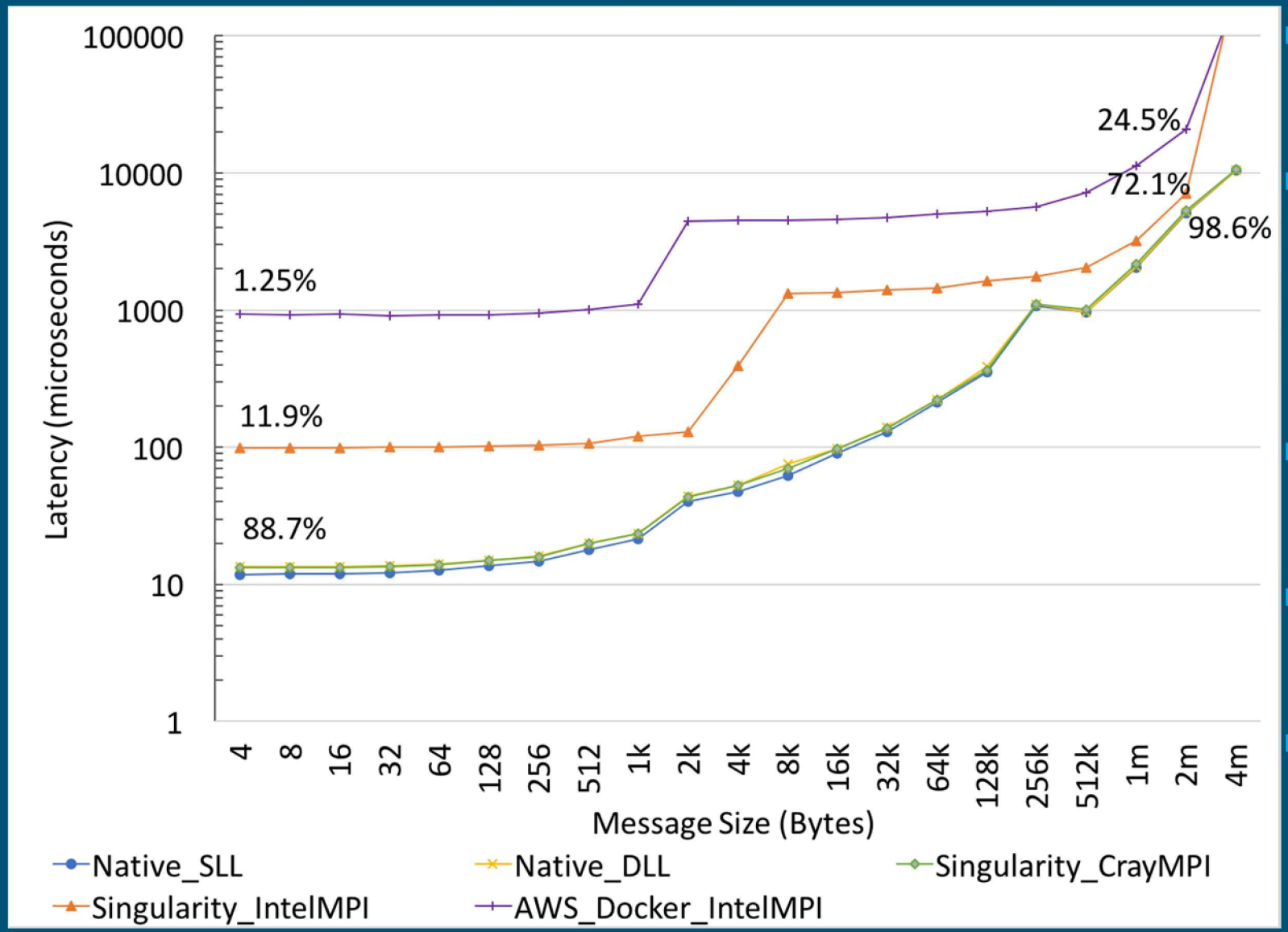
- Aries Interconnect provides peak 96 Gbps bandwidth
- Singularity w/ CrayMPI provides 99.4% of native efficiency
- Singularity w/ IntelMPI drops to only 39.5% of native efficiency
- Amazon EC2 provides peak 7 Gbps bandwidth
  - Just 6.9% of native Aries
- Illustrates massive difference between commodity and HPC interconnects

IMB PingPong Latency (log scale)



- Aries native latencies around 1.1 to 1.3 microseconds
  - ~200 ns = static linking
- Singularity w/ CrayMPI achieves 87.2% of native latency
  - Overhead = Native DLL
- Singularity w/ IntelMPI only achieves 15.6% efficiency
  - ~7 microseconds
- Amazon EC2 offers just 2.1% of native efficiency
  - ~55 microseconds
- Large difference depending on MPI library and interconnect

IMB All-Reduce across 768 ranks (log scale)



- IMPI All-Reduce benchmark
- Some overhead in dynamic linking of apps
  - ~1.6us latency overhead
  - Independent of containers
- Large messages hide latency
- 1 order of magnitude difference with Intel MPI
- 2 orders of magnitude difference with Amazon EC2

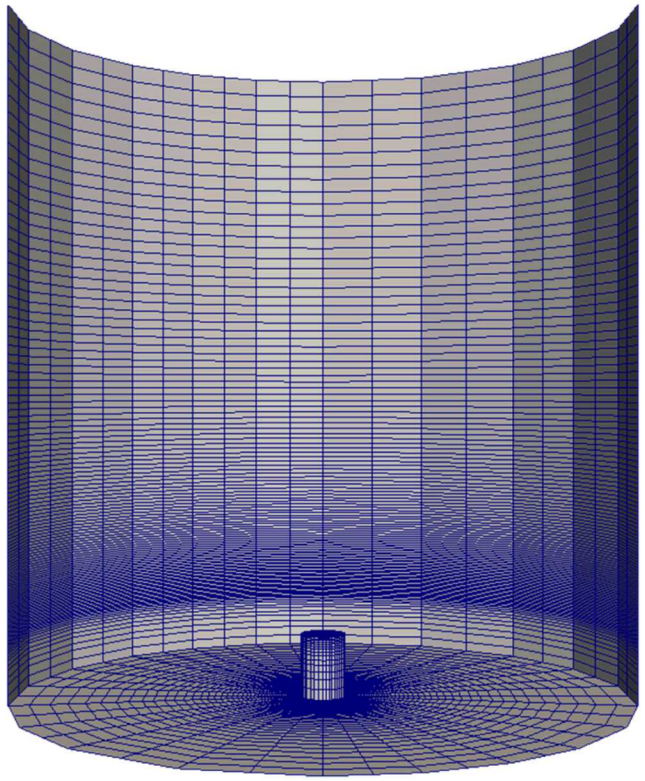
From Testbeds to Production

- Demonstrated Singularity containers on a Cray XC30
  - Performance *can* be near native
    - Leveraging vendor libraries within a container is critical
    - Cray MPI on Aries is performant
  - Confirming similar results from Shifter runtime
- Container and library interoperability is key moving forward
  - Vendor provided base containers desired
  - Community effort on library ABI compatibility is necessary
- Initial benchmarks and mini-apps, what about real apps?
  - Can mission applications use containers?
  - Can production/facilities teams build container images?
  - What are key metrics for success?
  - How will containers work in “air gapped” environments?

Initiated joint container investigation with Gunite Falls team in 9320

Problem Description

- **SNL Nalu:**
  - A generalized unstructured massively parallel low Mach flow code designed to support energy applications of interest [1]
  - Distributed on GitHub under 3-Clause BSD License [2]
  - Leverages the Trilinos libraries
    - Similar to bulk of SNL Advanced Simulation and Computing (ASC) Integrated Codes (IC) and Advanced Technology, Development, and Mitigation (ATDM) project applications
- **Milestone Simulation:**
  - Based on “milestoneRun” regression test [3] with 3 successive levels of uniform mesh refinement (17.2M elem.), 50 fixed time steps, and no file system output
  - Problem used for Trinity Acceptance [4] and demonstrated accordingly on Trinity HSW [5] and KNL [6], separately, at near-full scale



1. S. P. Domino, "Sierra Low Mach Module: Nalu Theory Manual 1.0", SAND2015-3107W, Sandia National Laboratories Unclassified Unlimited Release (UUR), 2015. https://github.com/NaluCFD/NaluDoc
2. “NaluCFD/Nalu,” https://github.com/NaluCFD/Nalu, Sep. 2018.
3. “Nalu/milestoneRun.i at master,” https://github.com/NaluCFD/Nalu/blob/master/reg_tests/test_files/milestoneRun/milestoneRun.i, Sep. 2018.
4. A. M. Agelastos and P. T. Lin, “Simulation Information Regarding Sandia National Laboratories’ Trinity Capability Improvement Metric,” Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, Technical report SAND2013-8748, October 2013.
5. M.Rajan, N.Wichmann, R.Baker, E.W.Draeger, S.Domino, C. Nuss, P. Carrier, R. Olson, S. Anderson, M. Davis, and A. Agelastos, “Performance on Trinity (a Cray XC40) with Acceptance Applications and Benchmarks,” in Proc. Cray User’s Group, 2016.
6. A. M. Agelastos, M. Rajan, N. Wichmann, R. Baker, S. Domino, E.W. Draeger, S. Anderson, J. Balma, S. Behling, M. Berry, P. Carrier, M. Davis, K. McMahon, D. Sandness, K. Thomas, S. Warren, and T. Zhu, “Performance on Trinity Phase 2 (a Cray XC40 utilizing Intel Xeon Phi processors) with Acceptance Applications and Benchmarks,” in Proc. Cray User’s Group, 2017.

Nalu Build & Environment Description

- **Doom Software Stack:**
  - TOSS 3.3-1 (~RHEL 7.5)
  - gnu-7.3.1, OpenMPI 2.1.1
  - hwloc-1.11.8
- **Container Software Stack:**
  - CentOS 7.5.1804 (~RHEL 7.5)
  - gnu-7.2.0, OpenMPI 2.1.1
  - hwloc-1.11.1
  - olv-plugin

Nalu Dependencies:

- zlib-1.2.11
- bzip2-1.0.6
- boost-1.65.0
- hdf5-1.8.19
- pnetcdf-1.8.1
- netcdf-4.4.1
- parmetis-4.0.3
- superlu_dist-5.2.2
- superlu-4.3
- suitesparse-5.1.0
- matio-1.5.9
- yaml-cpp-0.5.3
- Trilinos-develop-7c67b929
- Nalu-master-11899aff

- **SNL Doom:**
  - CTS-1 HPC platform
  - Dual E5-2695 v4 (Broadwell) processors, with AVX2, per node
  - 18 cores (36 threads) per processor, 36 cores (72 threads) total per node
  - 512GB DDR4 2400 MHz/s, 4 channels per socket
  - Intel Omni-Path HFI Silicon 100 Series (100 Gb/s adapter)

Open Source Software Stack Enables Greater Collaboration and Testing Across Networks and Systems

Nalu Container Analysis

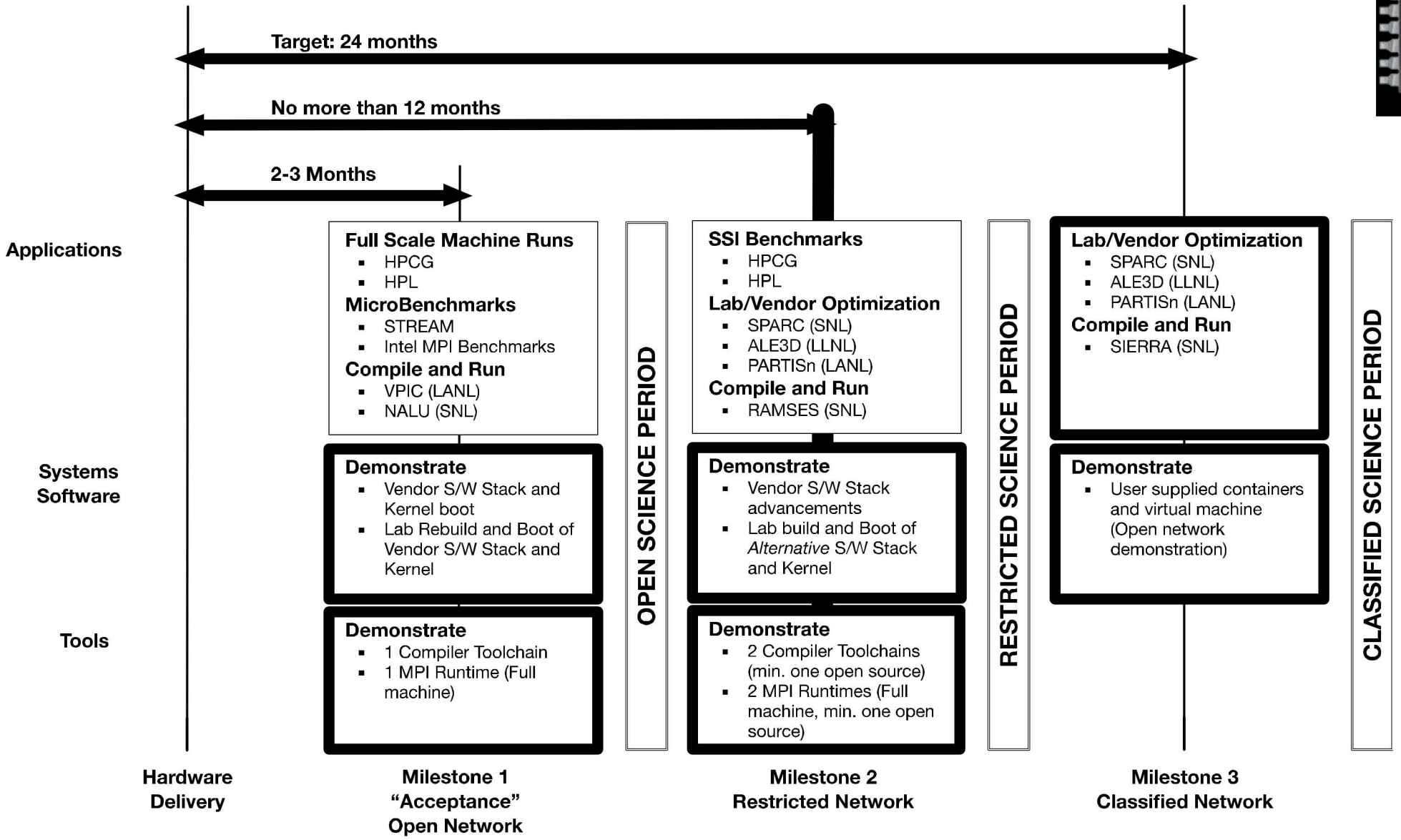
- The container was faster, but used more memory
- Dynamic linking of GCC 7.2 in container vs system GCC 4.8 Memory usage
  - Memory differences: gfortran & stdlibc++ libraries
    - GCC 7.2 libs much larger, ~18MB total
  - Performance differences: OpenMPI libs
    - Container's OpenMPI w/ GCC7 provides usempif08 in OpenMPI
    - usempif08 includes MPI3 optimizations vs MPI2 with usempi
- Position Independent Code (-fPIC) used throughout container compiles
  - Provides larger .GOT in memory, but often slightly improved performance on x86_64
- Overhead with using bash in container to load LD_LIBRARY_PATH before exec
  - Constant but small, depends on .bashrc file

Demonstrates both the power and pitfalls of building your own HPC application environment in containers

Containers at Exascale

- Containers have gained significant interest throughout the ECP
- There exists several container runtimes for HPC today
  - Shifter, Singularity, Charliecloud
  - Diversity is good!
- Containers can provide greater software flexibility, reliability, ease of deployment, and portability
- Several likely challenges to containers at Exascale:
  - Scalability
  - Resource management
  - Interoperability
  - Security
  - Further integration with HPC (batch jobs, Lustre, etc)

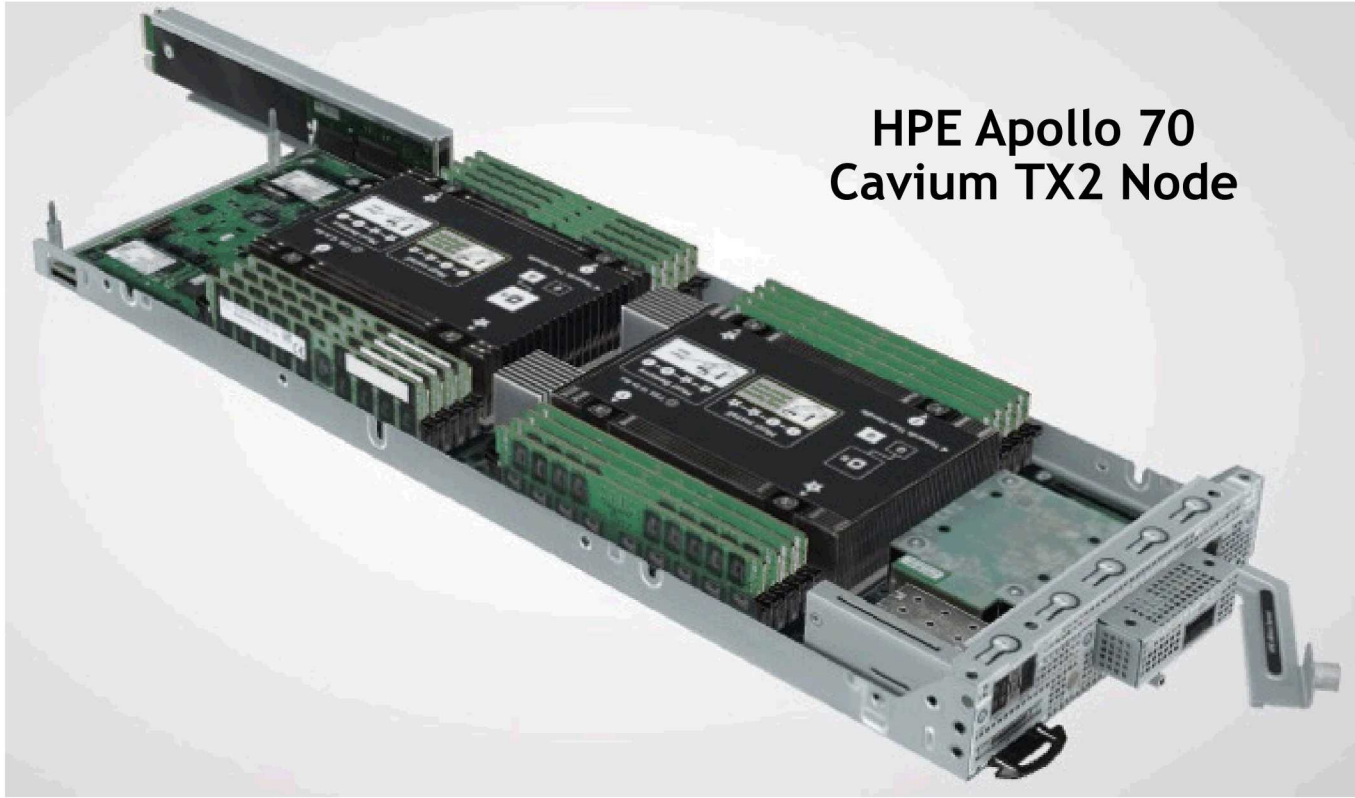
Acceptance Plan – Maturing the Stack



Vanguard-Astra Compute Node Building Block

Hewlett Packard Enterprise · arm · CAVIUM · Mellanox TECHNOLOGIES · redhat

- Dual socket Cavium Thunder-X2
  - CN99xx
  - 28 cores @ 2.0 GHz
- 8 DDR4 controllers per socket
- One 8 GB DDR4-2666 dual-rank DIMM per controller
- Mellanox EDR InfiniBand ConnectX-5 VPI OCP
- Tri-Lab Operating System Stack based on RedHat 7.5+



HPE Apollo 70
Cavium TX2 Node

Astra – the First Petscale Arm based Supercomputer

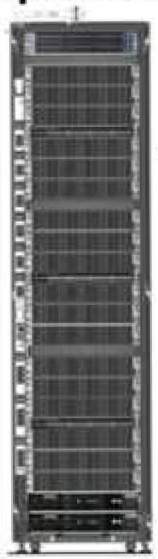
HPE Apollo 70 Chassis: 4 nodes

18 chassis/rack

72 nodes/rack

3 IB switches/rack
(one 36-port switch
per 6 chassis)

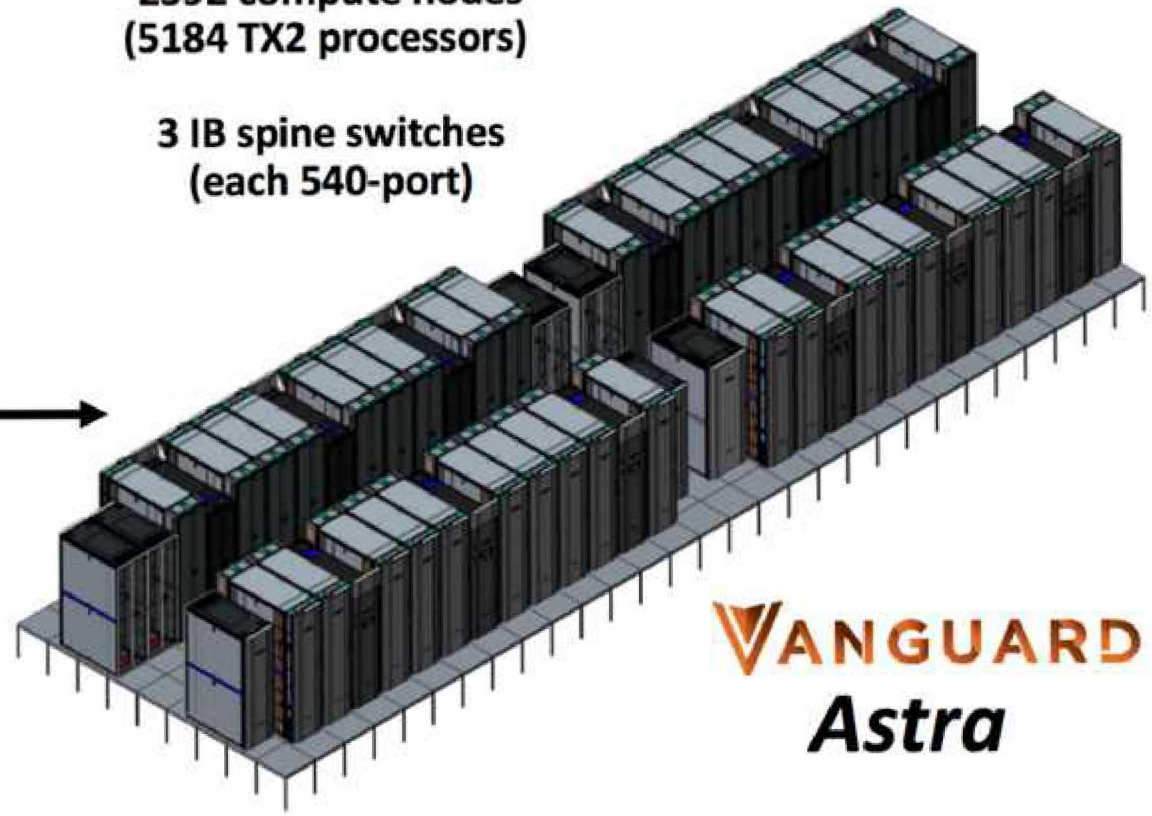
HPE Apollo 70 Rack



36 compute racks
(9 scalable units, each 4 racks)

2592 compute nodes
(5184 TX2 processors)

3 IB spine switches
(each 540-port)



VANGUARD
Astra

Advanced Trilab Software Environment (ATSE)

- Advanced Tri-lab Software Environment
  - Sandia leading development with input from Tri-lab Arm team
  - Provide a user programming environment for Astra
  - Partnership across the NNSA/ASC Labs and with HPE
- Lasting value for Vanguard effort
  - Documented specification of:
    - Software components needed for HPC production applications
    - How they are configured (i.e., what features and capabilities are enabled) and interact
    - User interfaces and conventions
  - Reference implementation:
    - Deployable on multiple ASC systems and architectures with common look and feel
    - Tested against real ASC workloads
    - Community inspired, focused and supported
    - Leveraging OpenHPC effort

Hewlett Packard Enterprise

Sandia National Laboratories

Lawrence Livermore National Laboratory

Los Alamos NATIONAL LABORATORY

openHPC

ATSE stack

ATSE is an integrated software environment for ASC workloads

Supercontainer Collaboration

- Interface with key ST and AD development areas
- Advise and support the container usage models necessary for deploying first Exascale apps and ecosystems
- Initiate deep-dive sessions with interested AD groups
  - ExaLEARN or CANDLE good first targets
  - Activities which can best benefit from container runtimes
- Develop advanced container DevOps models
  - Work with DOE Gitlab CI team to integrate containers into current CI plan
  - Leverage Spack to enable advanced multi-stage container builds
  - Integrate with ECP SDK effort to provide optimized container builds which benefit multiple AD efforts

Scalable R&D Activities

- Several Topics:
  - Container and job launch, including integration with resource managers
  - Distribution of images at scale
  - Use of storage resources (parallel file systems, burst buffers, on-node storage)
  - Efficient and portable MPI communications, even for proprietary networks
  - Accelerators e.g. GPUs
  - Integration with novel hardware and systems software associated with pre-Exascale and Exascale platforms
- Activities conducted in the context of interoperability
  - Portable solutions
  - Work for multiple container implementations
  - Multiple facilities at multiple scales

Future Integration

- For project to be successful, need to provide support for deploying container runtimes at individual facilities
- Facilities Integration ideas:
  - Help integrate with facilities on pre-Exa and Exa machine deployments
  - Include systems level support for efficient configuration, and interoperability across ECP
  - Demonstrate exemplar ECP application deployed with containers at scale
  - Work with HPC vendors today to ensure designs meet container criteria
  - Support upstream container projects when applicable (Docker, Singularity)

Training Education & Support

- Containers are a new SW mechanism, training and education is needed to help ECP community to best utilize new functionality
- Reports:
  - Best Practices for building and using containers
  - Taxonomy survey to survey current state of the practice
- Training activities:
  - Run tutorial sessions at prominent venues
    - ISC, SC, and ECP annual meetings
    - Already have several activities underway
  - Online training and outreach sessions
- Provide single source of knowledge for groups interested in containers