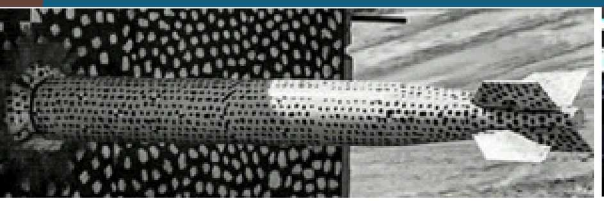
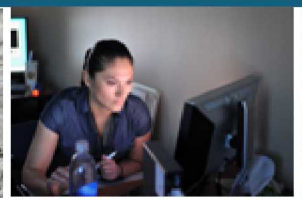
SAND2019-2750C

From Containerizing Testbeds for HPC Applications to Exascale Supercontainers





PRESENTED BY

Andrew J. Younge

Sandia National Laboratories

ajyoung@sandia.gov

Unclassified Unlimited Release
SAND2019-XXXXP

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

- Motivation
  - Why containers in HPC
  - Container vision at Sandia w/ Singularity
- Initial investigation of Singularity on Cray
- Mission HPC apps on production DOE/NNSA clusters
- DOE ECP Supercontainers Project
  - Supporting containers at Exascale
- Conclusions & Future Directions

Containers in HPC

- **BYOE - Bring-Your-Own-Environment**
  - Developers define the operating environment and system libraries in which their application runs
- **Composability**
  - Developers explicitly define 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

Initial Container Vision

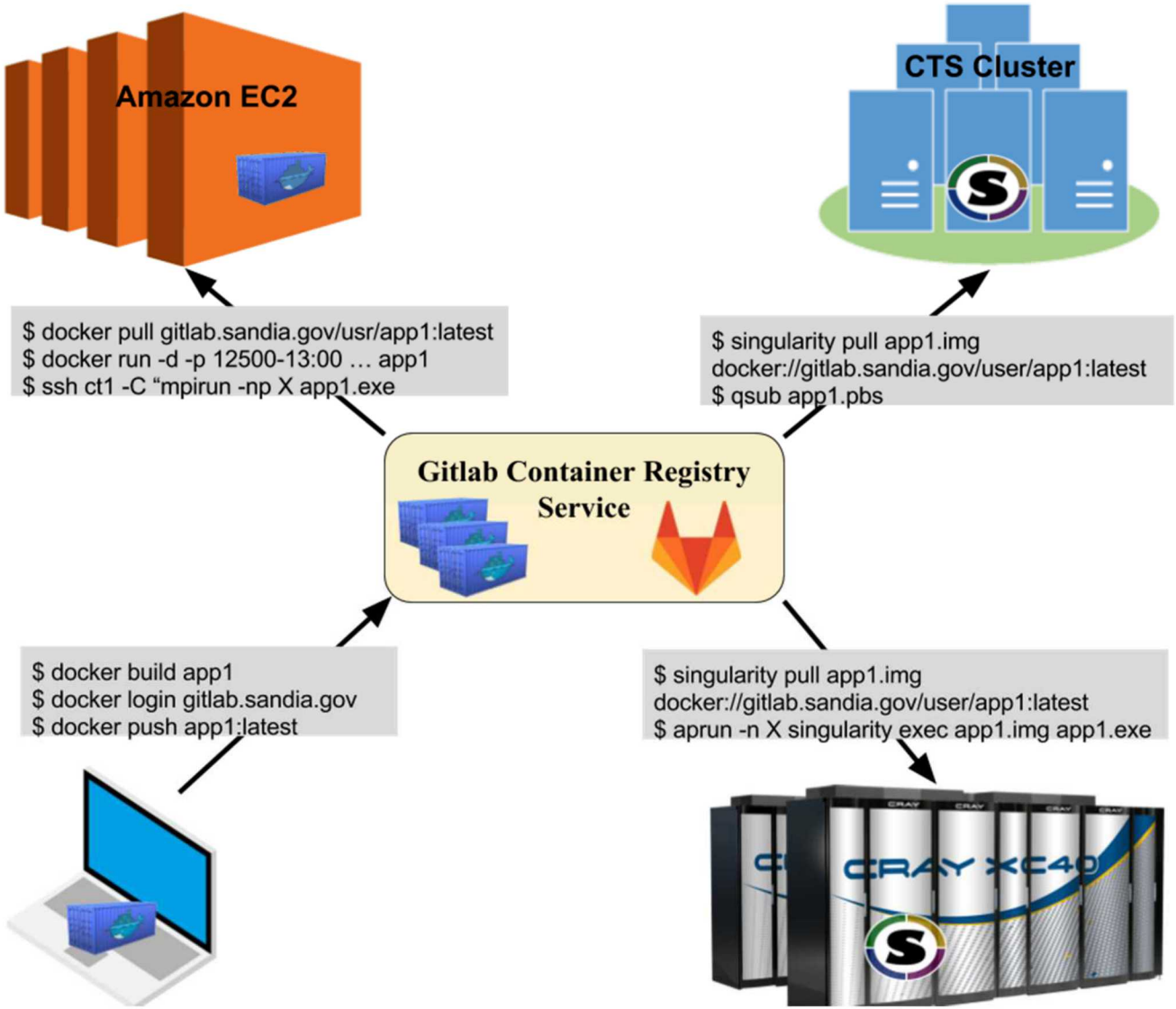
- Support software dev and testing on laptops
  - Working builds that can run on supercomputers
  - Dev time on supercomputers is extremely expensive
  - May also leverage VM/binary translation
- Let developers specify how to build the environment AND the application
  - Users just import a container and run on target platform
  - Many containers, but can have different code “branches” 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
- Performance matters!
  - Use HPC to “shake out” container implementations on HPC
  - Keep features to support future complete workflows

Singularity Containers

- Many different container options
  - Docker, Shifter, Singularity, Charliecloud, etc. etc.
- Docker is not good fit for running HPC workloads
  - Security issues, no HPC integration
- Singularity best for current mission needs
  - OSS, publicly available, support backed by Sylabs
  - Simple image plan, support for HPC systems
  - Docker image support, as well as custom Singularity builds
  - Support for multiple architectures (x86, ARM, POWER)
  - Large community support

Container DevOps

- Impractical for apps to use large-scale supercomputers for DevOps and testing
  - HPC resources have long batch queues
  - Dev time commonly delayed as a result
- Create deployment portability with containers
  - Develop Docker containers on your laptop or workstation
  - Leverage registry services
  - Separate networks maintain separate registries
  - Import to target deployment
  - Leverage local resource manager



Singularity on a Cray

- Crays can represent pinnacle of HPC
  - 4 of the 10 fastest supercomputers are Cray (Nov 17 Top500)
- Cray systems are different than Linux clusters
  - Specialized compute OS, no node-local storage, custom interconnect, specialized and tuned libraries, etc
- Modified Cray CNL kernel to build in necessary features
  - Loop mounting and EXT3 support, soon SquashFS and Overlay
- Create /opt/cray and /var/opt/cray mounts on all images
- Use LD_LIBRARY_PATH to link in Cray system software
  - XPMEM, CrayMPI, uGNI, etc

How does running Singularity on a Cray compare to Docker on a Cloud?

Tale of Two Systems

Volta

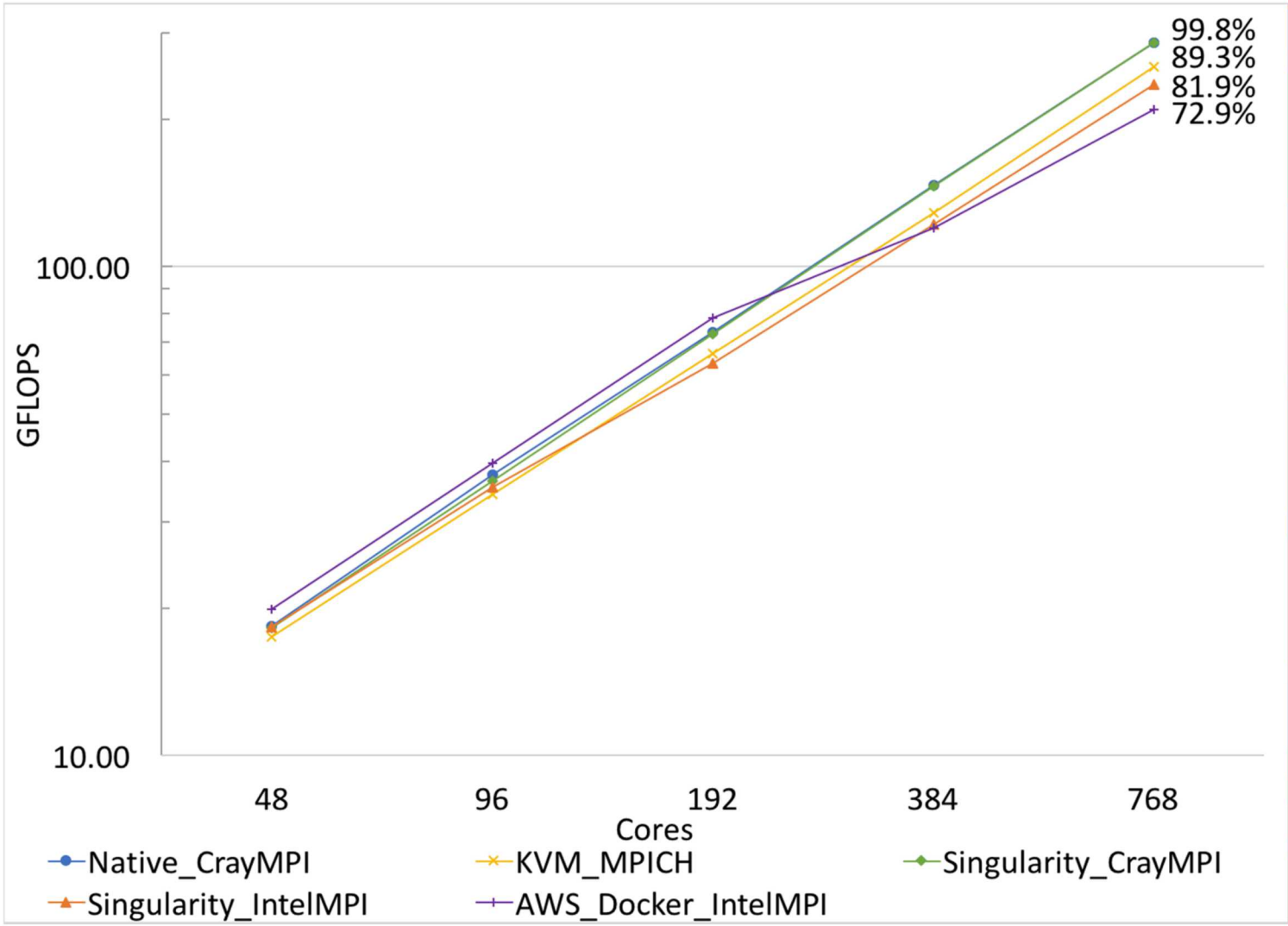
- Cray XC30 system
- NNSA ASC testbed at Sandia
- 56 nodes:
  - 2x Intel ”IvyBridge” E5-2695v2 CPUs
  - 24 cores total, 2.4Ghz
  - 64GB DDR3 RAM
- Cray Aries Interconnect
- No local storage, Shared DVS filesystem
- Singularity 2.X
- Cray CNL ver. 5.2.UP04
  - Based on SUSE 11
  - 3.0.101 kernel
- 32 nodes used to keep equal core count

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
  - Docker 1.19
- Run in dedicated host mode
- 48 node virtual cluster = $176.64/hour

HPCG VMs and container performance

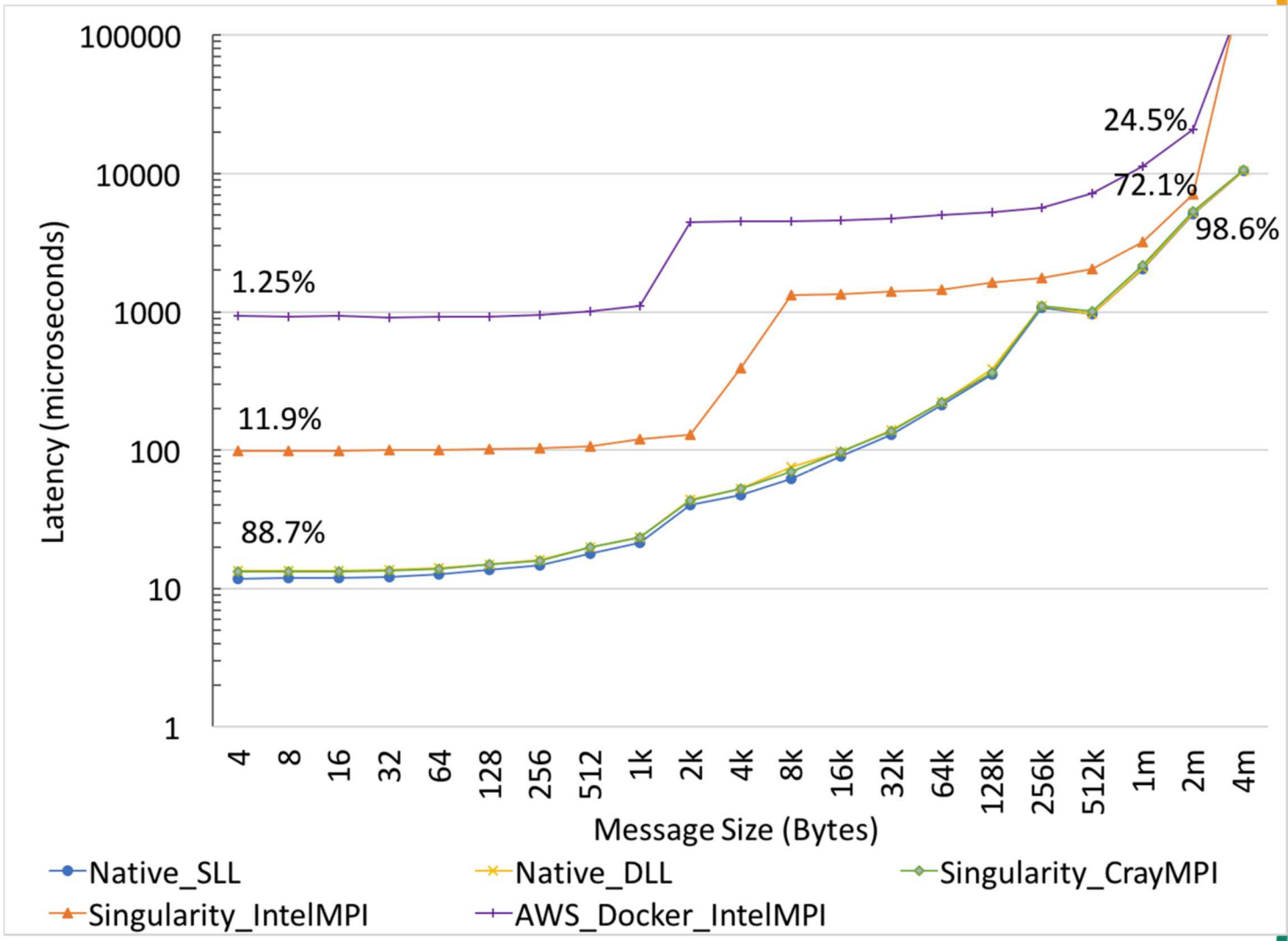
- Modified Cray XC testbed to run Singularity containers
- Create `/opt/cray` and `/var/opt/cray` on all images
- Link in Cray system software
  - XPMEM, CrayMPI, uGNI, etc
- HPCG Benchmark in Container
  - Compare Singularity on Cray
  - Compare KVM on Cray
  - Compare Amazon EC2



Younge et. al, *A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds*, IEEE CloudCom 2017

Container MPI Performance

- IMPI All-Reduce benchmark
- Some overhead in dynamic linking of apps
  - ~1.6us latency
  - 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

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 most performant
- 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 production apps?
  - Can NNSA 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?

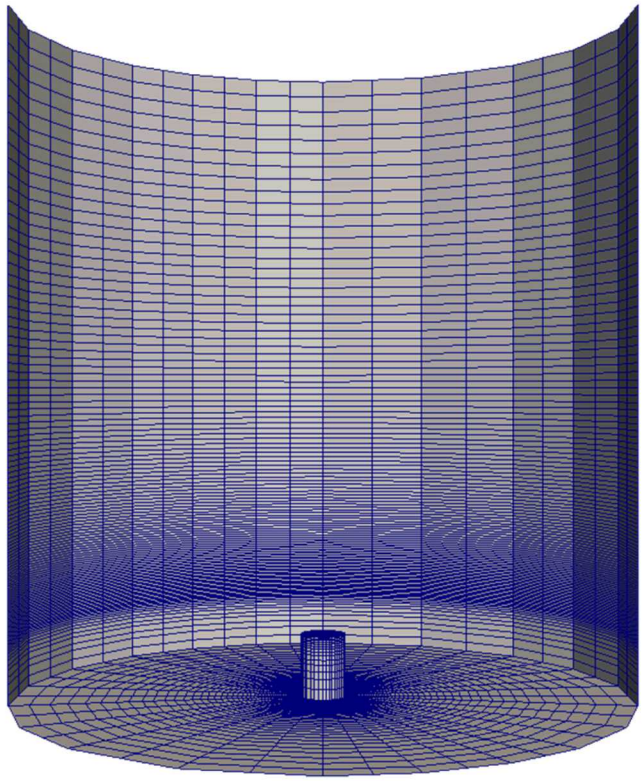
System Description

- **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
  - Core base frequency 2.1 GHz, 3.3 GHz max boost frequency
  - 32 KiB instruction, 32 KiB data L1 cache per core
  - 256 KiB unified (instruction + data) L2 cache per core
  - 2.5 MB shared L3, 45 MiB L3 per processor
  - 4 memory channels per processor (8 per node)
  - DDR4 2400 MHz/s
  - 512 GB per node
  - Intel Omni-Path HFI Silicon 100 Series (100 Gb/s adapter) for MPI communications

CTS-1 Platform Is Relevant to the Tri-Labs for Production HPC Workloads

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 SNL Sierra Toolkit and 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.

Performance Testing Description

- **Test Characteristics:**
  - Strong scale problem
    - 36 MPI ranks per node
    - No threading
    - Bind ranks to socket
  - Measure the the `mpiexec` process wall time for both native and container simulations
  - Extract the maximum resident set size (MaxRSS) for the `mpiexec` process and all of its sub-processes on the head node for both native and container simulations
    - We want to gather all overhead the Singularity container runtime imposes over a native simulation
    - The tested methodology for the Singularity container simulation is:

```
mpiexec -> singularity exec -> bash -> nalu
```

  - Extract the maximum resident set size (MaxRSS) for all of the Nalu MPI processes across all nodes and compute the “average” MaxRSS for Nalu
    - This value is computed for both the native and container simulations so that the former can be subtracted from the latter to compute the container overhead
    - This was extracted via LDPXI using LD_PRELOAD to attach to the native and containerized Nalu processes; LDPXI extracts this via `ru_maxrss` from `getrusage()` at the end of the simulation

Wall Time and Memory Are Key Performance Parameters for Production Workloads

Build & Environment Description

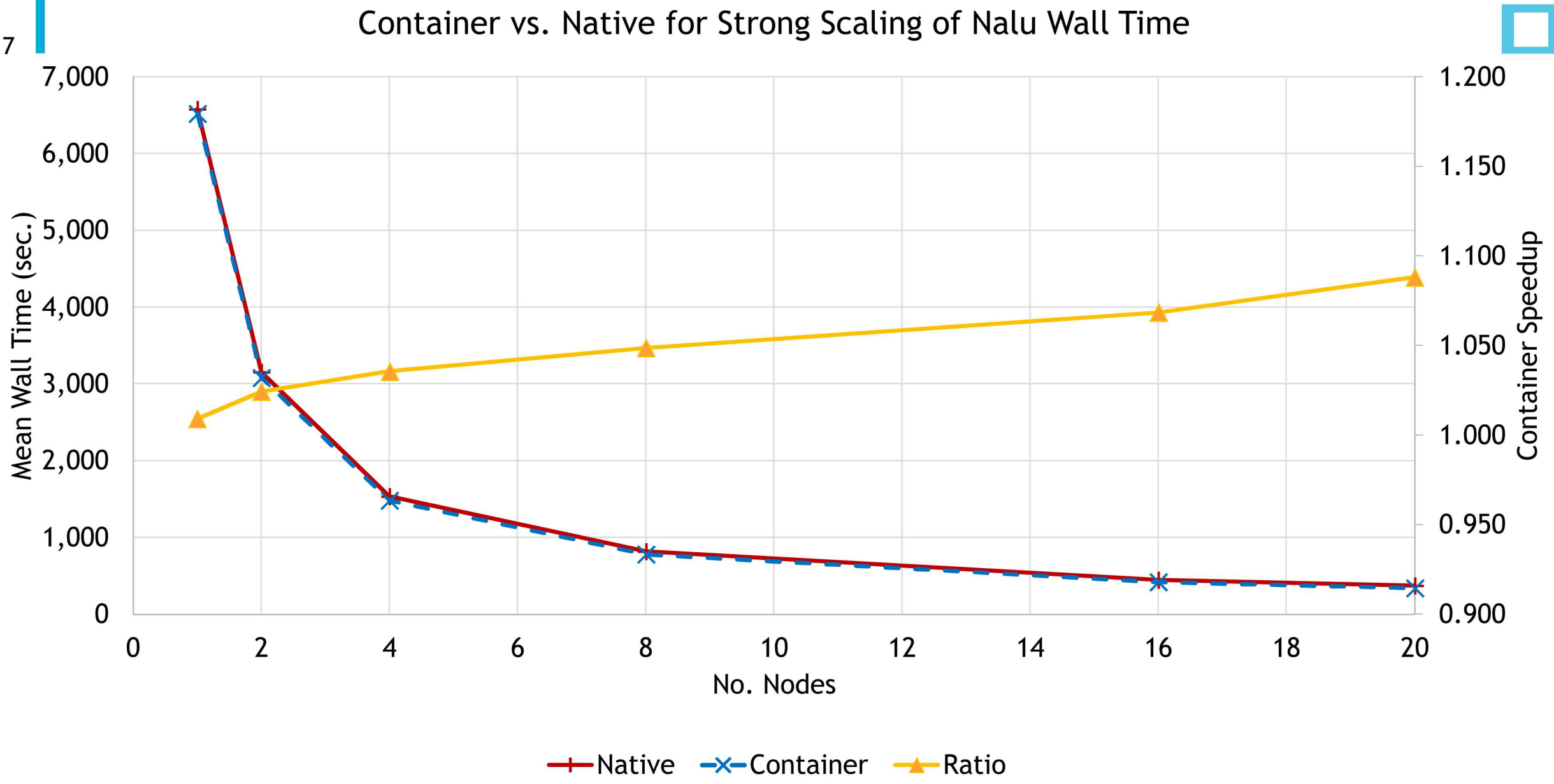
- **Doom Software Stack:**
  - TOSS 3.3-1 (~RHEL 7.5)
  - gnu-7.3.1, OpenMPI 2.1.1
  - hwloc-1.11.8
- **Container Software Stack:**
  - CentOS 7.5.1804 (~RHEL 7.5)
  - gnu-7.2.0, OpenMPI 2.1.1
  - hwloc-1.11.1
  - olv-plugin

Nalu Dependencies:

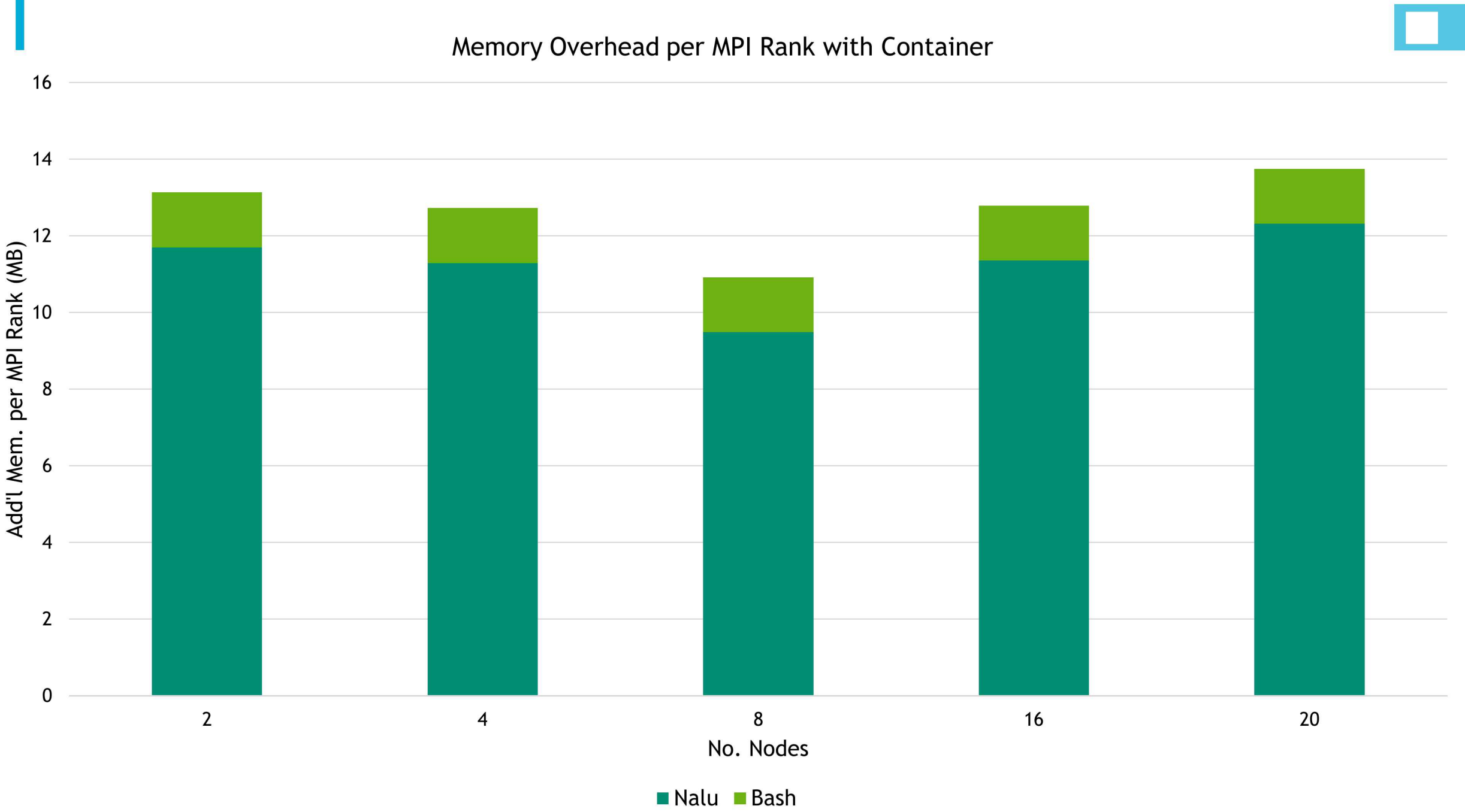
- zlib-1.2.11
- bzip2-1.0.6
- boost-1.65.0
- hdf5-1.8.19
- pnetcdf-1.8.1
- netcdf-4.4.1
- parmetis-4.0.3
- superlu_dist-5.2.2
- superlu-4.3
- suitesparse-5.1.0
- matio-1.5.9
- yaml-cpp-0.5.3
- Trilinos-develop-7c67b929
- Nalu-master-11899aff

- **Notes:**
  - Built with Docker, imported into Singularity
  - Container image available on Dockerhub

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



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


Memory Overhead per MPI Rank with Container
Add'l Mem. per MPI Rank (MB)
0
2
4
6
8
10
12
14
16
2
4
8
16
20
No. Nodes
Nalu
Bash

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

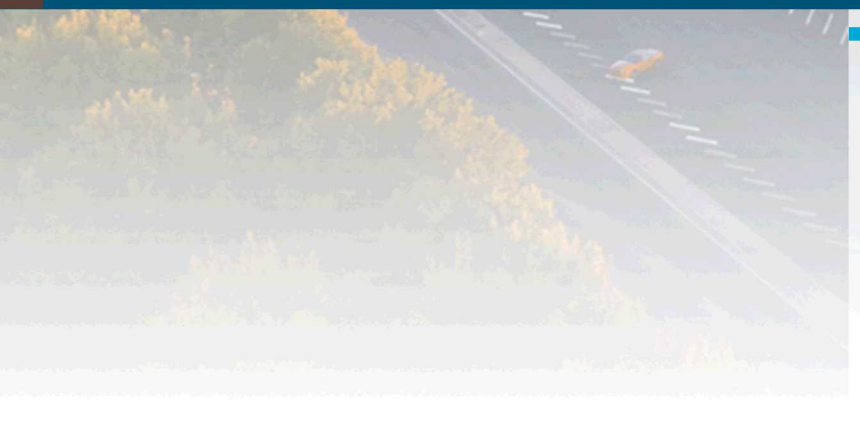

ASTRA
"Per aspera ad astra"
RM SUPERCOMPUTER
VANGUARD
ATSE
stack
2.3 PFLOPs peak
>5000 TX2 CPUs, ~150k cores
885 TB/s memory bandwidth peak
332 TB memory
1.2 MW

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
- 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.

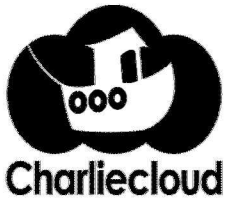
Containers will fully work with automated transfers for use in air gapped networks

Towards the DOE ECP Supercontainer Project



Supercontainers

- 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)

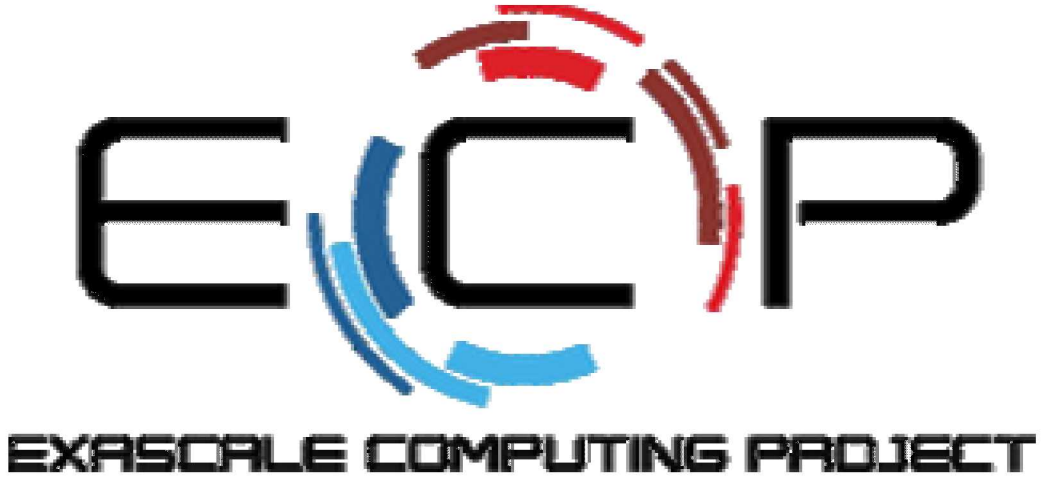


Q: What is the ECP?
A: **The Exascale Computing Project (ECP)** is focused on accelerating the delivery of a capable exascale computing ecosystem that delivers 50 times more computational science and data analytic application power than possible with DOE HPC systems such as Titan (ORNL) and Sequoia (LLNL). With the goal to launch a US Exascale ecosystem by 2021, the ECP will have profound effects on the American people and the world.

The ECP is a collaborative effort of two U.S. Department of Energy organizations – the **Office of Science (DOE-SC)** and the **National Nuclear Security Administration (NNSA)**.

ECP Supercontainers Project

- Join effort across Sandia, LANL, LBNL, LLNL, U. of Oregon
- Ensure container runtimes will be scalable, interoperable, and well integrated across the DOE
- Enable container deployments from laptops to Exascale
- Help ECP applications and facilities leverage containers most efficiently
- Three-fold approach to achieve success:
  - Scalable R&D activities
    - Will containers work @ Exascale
    - Performance studies
  - Collaboration with related ST and interested AD projects
    - Integration with Spack
    - Developer deep-dive sessions
    - CI/CD pipeline with DOE Gitlab
  - Training, Education, and Support
- Activities conducted in the context of interoperability
  - Portable solutions
    - Containerized ECP that runs on Astra, A21, El-Capitan, …
  - Work for multiple container implementations
    - Not picking a “winner” container runtime
  - Multiple DOE facilities at multiple scales

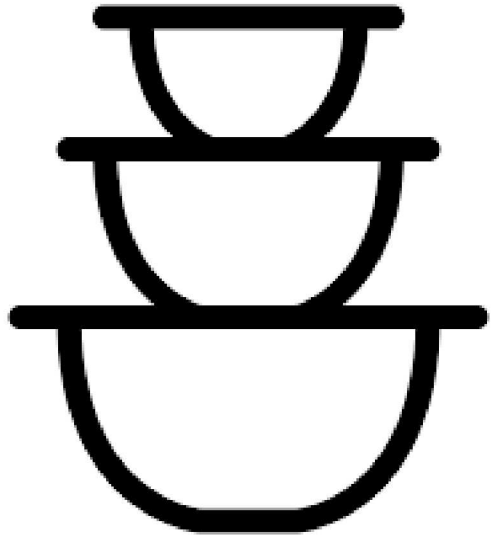


Conclusion

- Containers have taken hold in HPC
- Singularity runtime has demonstrated value in testbeds and production DOE/NNSA workloads
  - Performance is near-native
  - MPI ABI compatibility is key
  - Multiple architecture support
- ECP Supercontainers
  - Performance must be validated at scale
  - Interoperability is important
  - Exascale software stack

Future Directions in Containers

- Containers must work at Exascale
  - DOE ECP efforts are depending on it
  - Embrace architectural diversity
- Containerized CI/CD pipeline with Gitlab
- Build-time optimizations
  - Entire ECP software stack in container
  - Multi-stage builds
  - 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 and
- Reproducibility?



Thanks!

ajyoung@sandia.gov

14th Workshop on Virtualization in High-Performance Cloud Computing (VHPC'19) @ISC19 in Frankfurt, DE

Papers due April 19th - vhpc.org

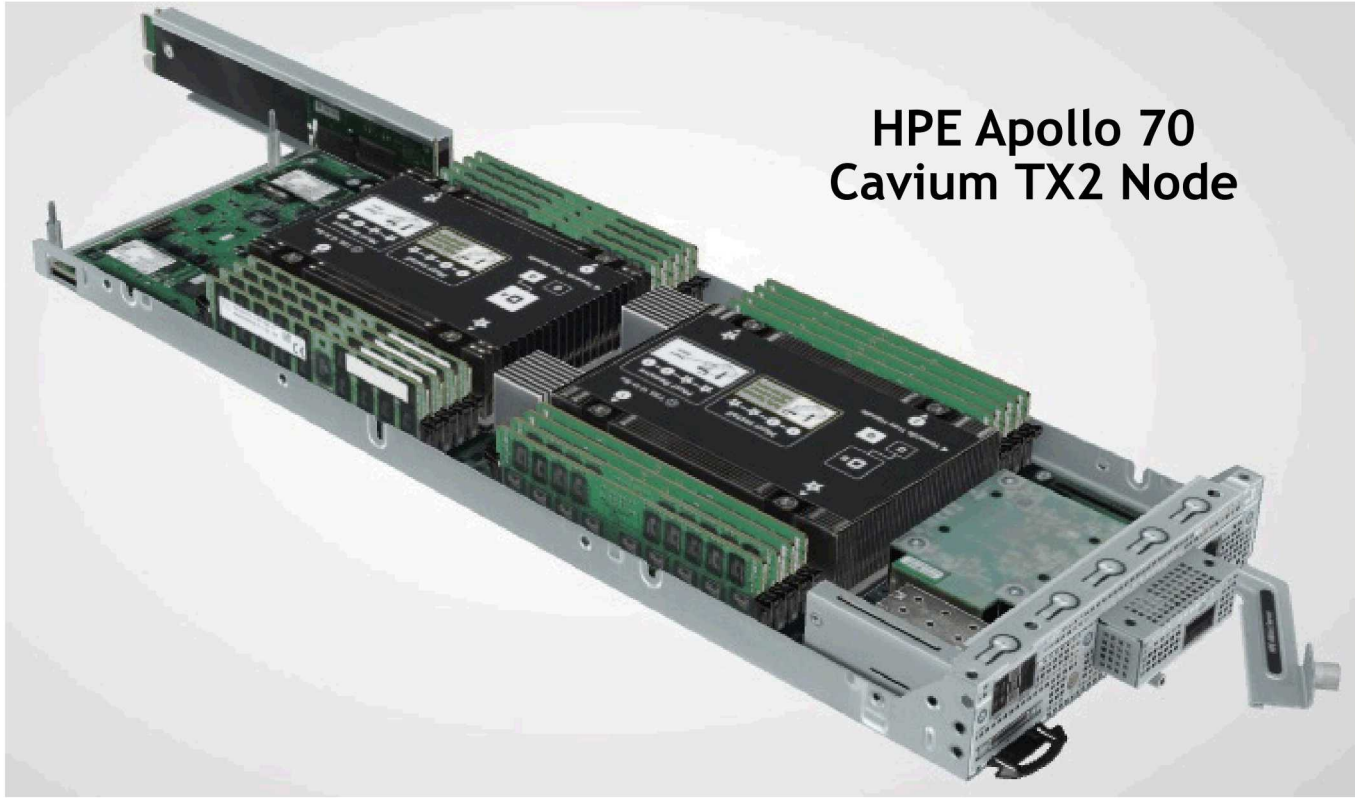
Students, Postdocs, Collaborators...

Backup Slides

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

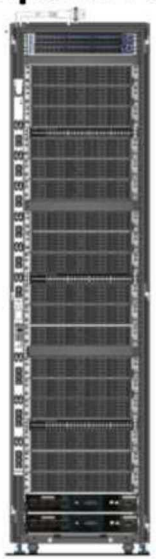
HPE Apollo 70 Chassis: 4 nodes

18 chassis/rack

72 nodes/rack

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

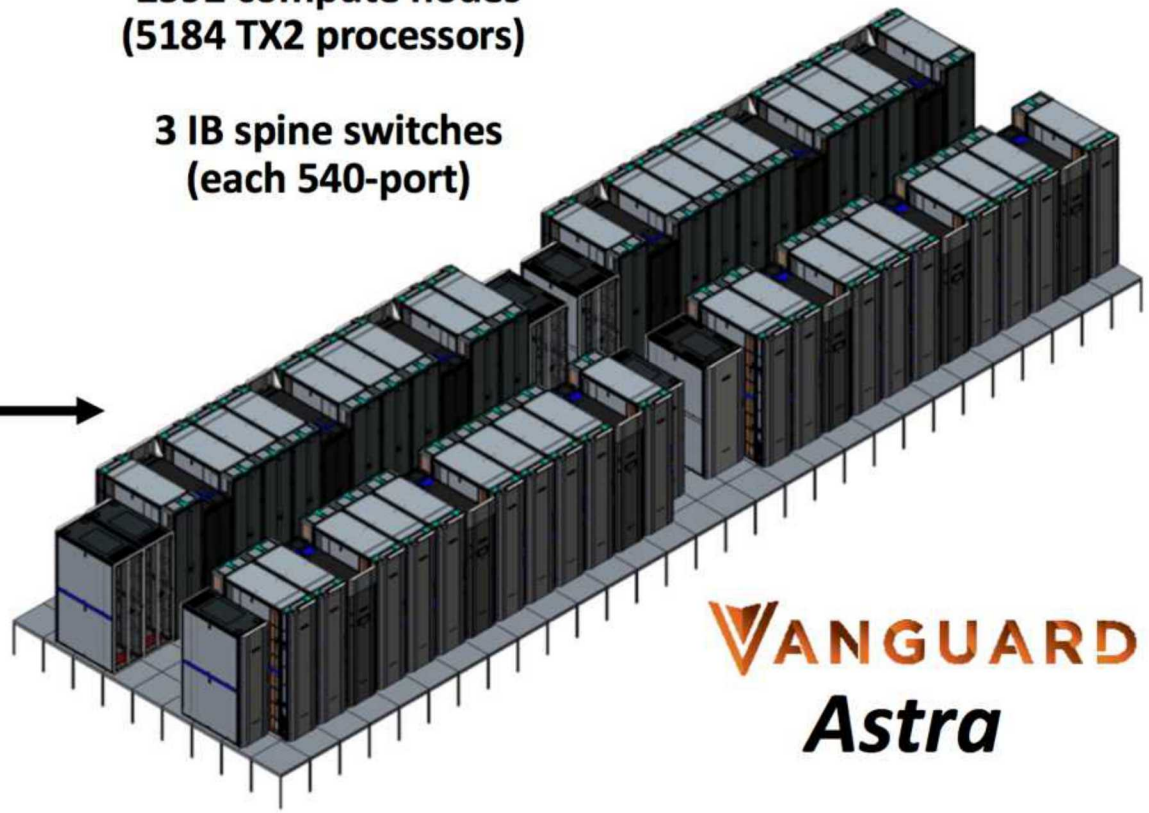
HPE Apollo 70 Rack



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

2592 compute nodes
(5184 TX2 processors)

3 IB spine switches
(each 540-port)



VANGUARD
Astra

Sandia has a history with Arm - NNSA/ASC testbeds

2014

Hewlett Packard Enterprise

ARM

Hammer

Applied Micro
X-Gene-1
47 nodes

ARM

Sullivan

Cavium ThunderX1
32 nodes

2017

Hewlett Packard Enterprise

ARM

Mayer

Pre-GA Cavium
ThunderX2
47 nodes

2018

Hewlett Packard Enterprise

ARM

Vanguard/Astra

HPE Apollo 70
Cavium ThunderX2
2592 nodes

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