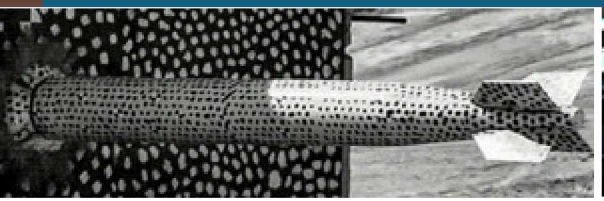
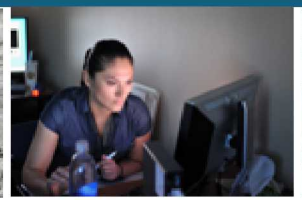
SAND2020-7555PE

Containers in HPC:
Testbeds, Production, and Towards Exascale





PRESENTED BY

Andrew J. Younge

Sandia National Laboratories

ajyoung@sandia.gov

Berkeley Lab CS Seminar

July 24th, 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.

Abstract:

Container computing has revolutionized how many industries and enterprises develop and deploy software and services. This model has recently gained traction in the High Performance Computing (HPC) community through the enablement of HPC-specific container runtimes. As the complexity of HPC applications expand, workload ensembles are becoming increasingly complex and, as a result, difficult to maintain when the target platforms' environments are increasingly diverse and continually changing. The advent of containers in HPC offer the ability for developers to differentiate software components and containerized environments makes novel software stacks readily attainable for application teams.

This talk provides an overview of containers in HPC, starting with initial testbed performance characterizations of containers and spanning into a multi-laboratory DOE effort, named SuperContainers, which looks to enable containers for Exascale workloads. This talk highlights the system software research efforts for ensuring efficient container utilization, including overhead evaluation, scalability considerations, deployment assessments on non-traditional architectures, and best practices for using containers in HPC. The talk will also provide an exemplar DevOps process which utilizes Podman to build containers and Singularity to deploy containers on an Arm-based computing ecosystem and the challenges associated therein. Finally, we will define new challenges in interoperability and investigate the ability to enable portable container solutions across DOE facilities, ranging from laptops to Exascale platforms

Bio:

Dr. Andrew J. Younge is a Computer Scientist in the Scalable System Software department at Sandia National Laboratories. His research interests include containers, high performance computing, system software, distributed systems, and energy efficient supercomputing. The cornerstone of his research focuses on improving the usability and efficiency of supercomputing system software. Andrew currently serves as the Principal Investigator for the Supercontainers effort under the DOE Exascale Computing Project and is a key contributor to the Vanguard Astra system, the world’s first Petascale ARM supercomputer. Prior to joining Sandia, Andrew held visiting positions at the MITRE Corporation, the University of Southern California’s Information Sciences Institute, and the University of Maryland, College Park. He received his PhD in Computer Science from Indiana University in 2016 and his BS and MS in Computer Science from the Rochester Institute of Technology in 2008 and 2010, respectively.

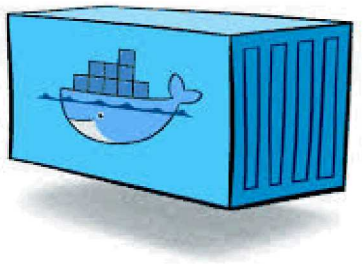
Outline

- What is a container and do they matter in HPC?
- Initial experimentation with containers in HPC
- Petascale & Production
- Containers at Exascale = Supercontainers
- Into the future?
- Tupperware

What is a Container?

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



HPC in the Cloud?

- 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
  - Inflexible for emerging apps and workflows

- What about Containers?
  - Can we support containers in HPC in the same way as clouds do?
  - Does this model fit for _both_ HPC and emerging workloads across DOE?
  - Can we adapt our programming environments into container images?

Initial look at Containers in HPC

HPC Container Vision

- Support HPC software development and testing on laptops/workstations
  - Create working container builds that can run on supercomputers
  - Minimize development time on supercomputers
- 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 requirements, etc
- Performance matters
  - Use mini-apps to “shake out” container implementations on HPC
  - Expand to mission workloads & applications
  - Envision features to support future workflows (ML/DL/in-situ analytics)

Containers in HPC

Wanted Features

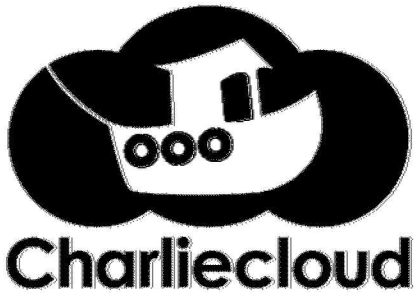
- **BYOE - Bring-Your-Own-Environment**
  - Developers define the operating environment and system libraries in which their application runs
- **Composability**
  - Developers have control over how their software environment is composed of modular components as container images
  - Enable reproducible 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

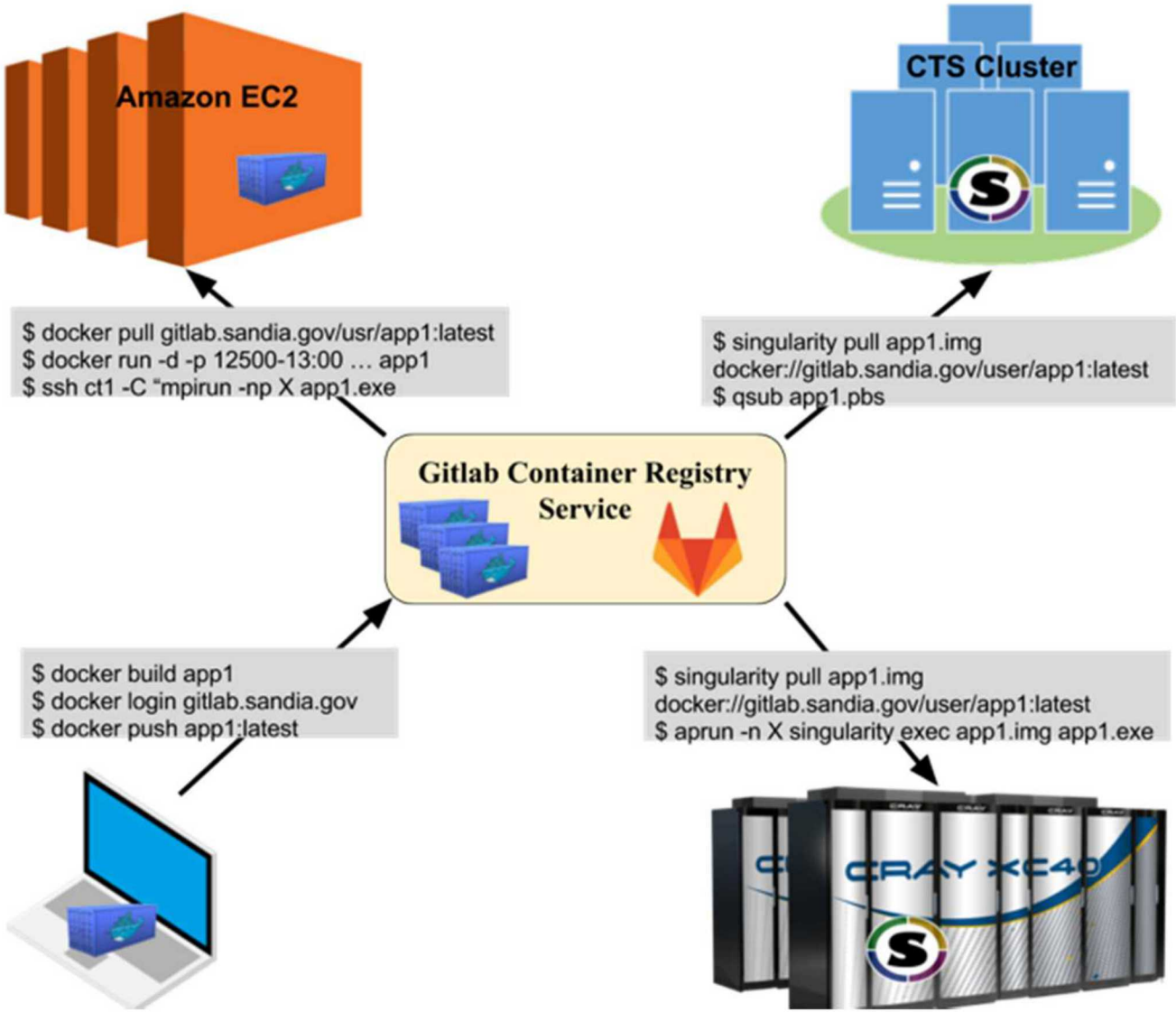
- Docker is not good fit for running HPC workloads
  - Building with Docker on my laptop is ok
  - Security issues, no HPC integration
- Several different container options in HPC



- 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

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) and local resource manager (SLURM)
  - Separate networks maintain separate registries



Singularity on a Cray Supercomputer (in 17')

- Cray supercomputers 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

- Now much easier – just install RPM from Sylabs

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

Singularity on a Cray Supercomputer (in 17')

- Supercomputers 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

- Now much easier – just install RPM from Sylabs

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

Tale of Two Systems – HPC vs Cloud

Volta

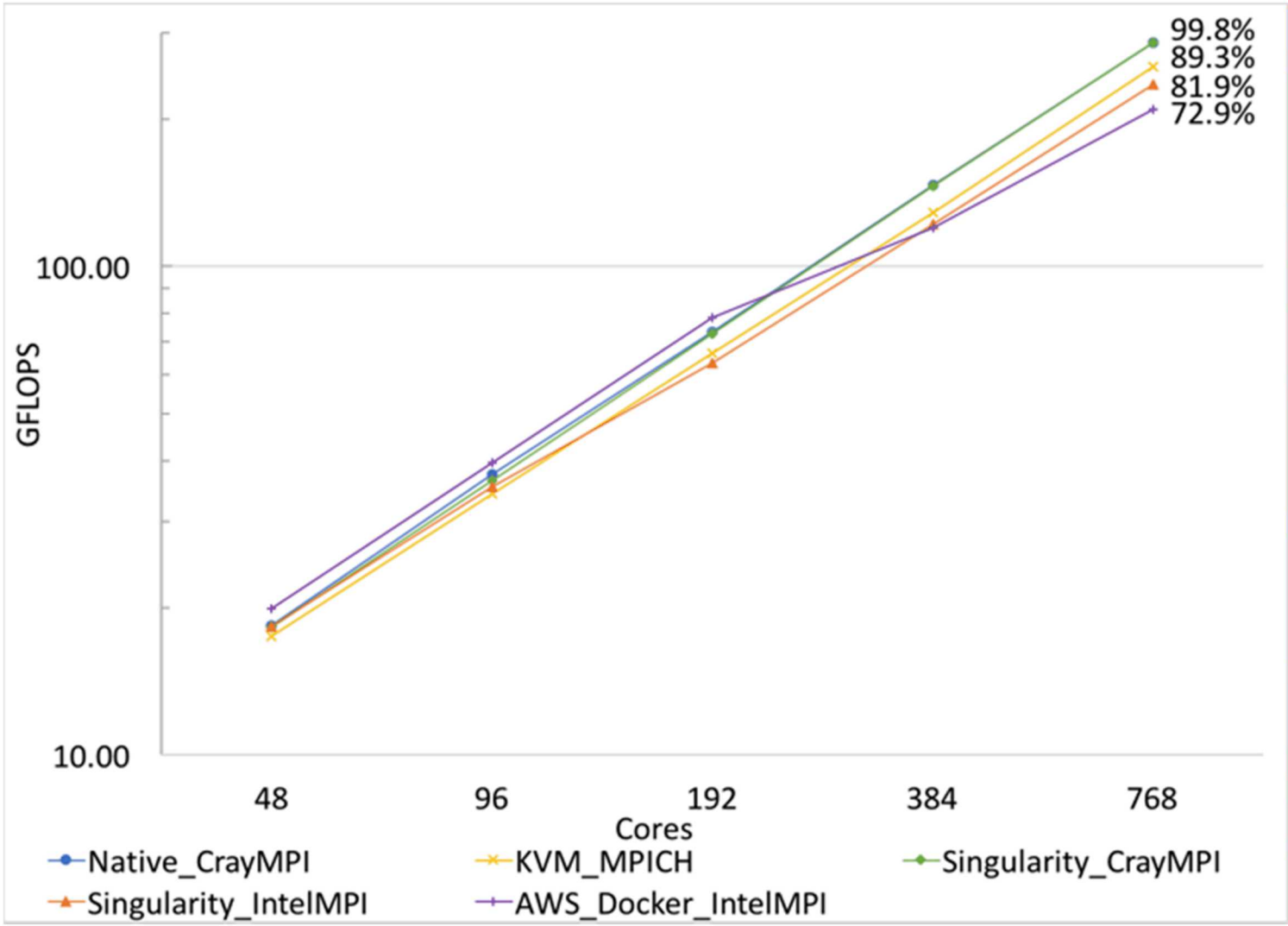
- Cray XC30 system
- NNSA ASC testbed at Sandia
- 56 nodes:
  - 2x Intel ”IvyBridge” E5-2695v2 CPUs
  - 24 cores total, 2.4Ghz
  - 64GB DDR3 RAM
- Cray Aries Interconnect
- No local storage, Shared DVS filesystem
- Singularity 2.X
- Cray CNL ver. 5.2.UP04
  - Based on SUSE 11
  - 3.0.101 kernel
- 32 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
  - Singularity with CrayMPI = native
  - KVM with MPICH = slower
  - AWS faster but does not scale



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

Petascale and Production

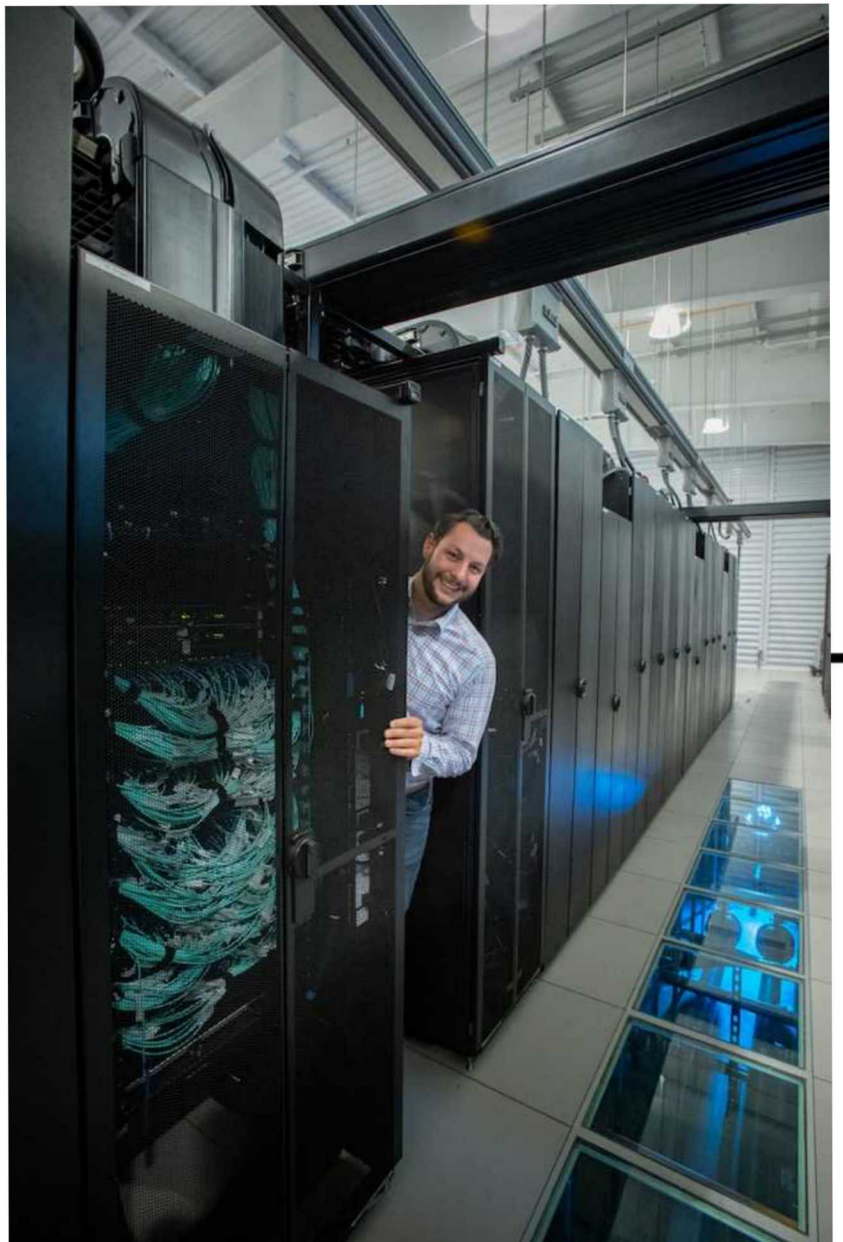

ASTRA
"Per aspera ad astra"
Hewlett Packard
Enterprise
RM SUPERCOMPUTER
VANGUARD
ATSE
stack
2.3 PFLOPs peak
>5000 TX2 ARM CPUs, ~150k cores
885 TB/s memory bandwidth peak
332 TB memory
1.2 MW

Vanguard Astra: At a Glance

Hewlett Packard Enterprise

- 2,592 HPE Apollo 70 compute nodes
  - 5,184 CPUs, 145,152 cores, 2.3 PFLOPs (peak)
- Marvell Thunder-X2 ARM SoC, 28 core, 2.0 GHz
- Memory per node: 128 GB
  - 16 x 8 GB DDR DIMMs
  - Aggregate capacity: 332 TB, 885 TB/s (peak)
    - 247 GB/s per node STREAM
- Mellanox IB EDR, ConnectX-5
  - 112 36-port leaf, 3 648-port spine switches
- ATSE software stack
  - TOSS Base Operating system
- HPE Apollo 4520 All-flash Lustre storage
  - Storage Capacity: 990 TB (usable)
  - **Upgrade to 3X memory**
  - Storage Bandwidth: 250 GB/s
    - 400 GB/s stunt mode, 432 GB/s peak

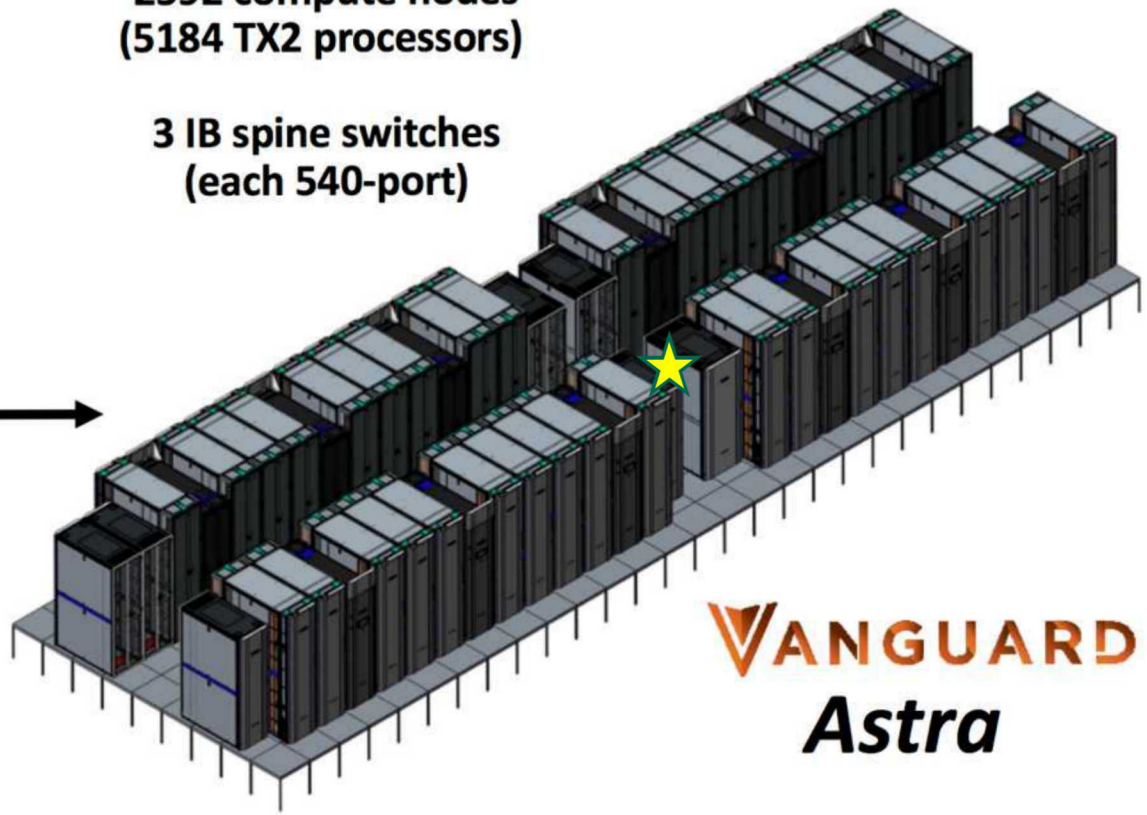
Astra System Architecture



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

2592 compute nodes
(5184 TX2 processors)

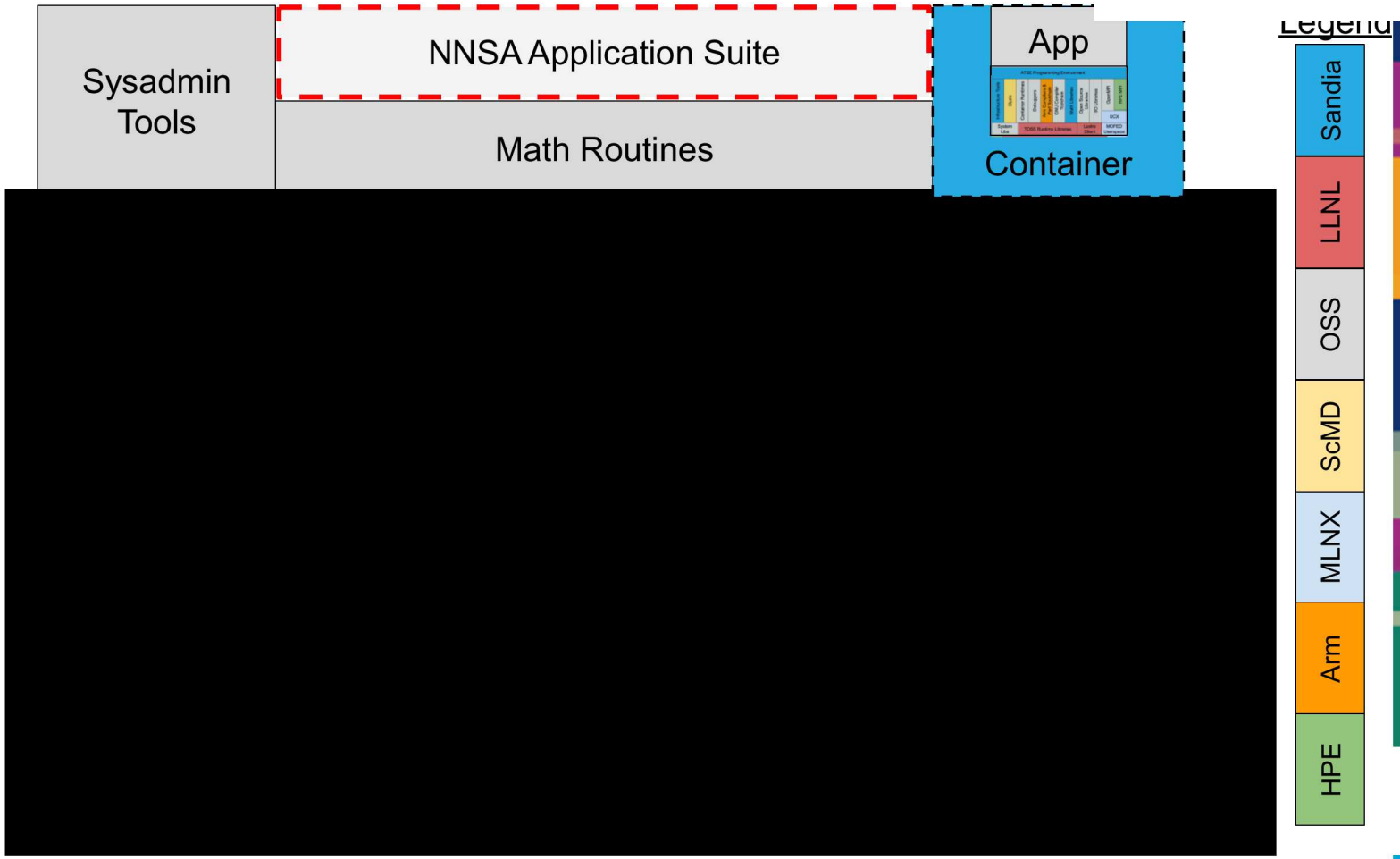
3 IB spine switches
(each 540-port)



VANGUARD
Astra

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



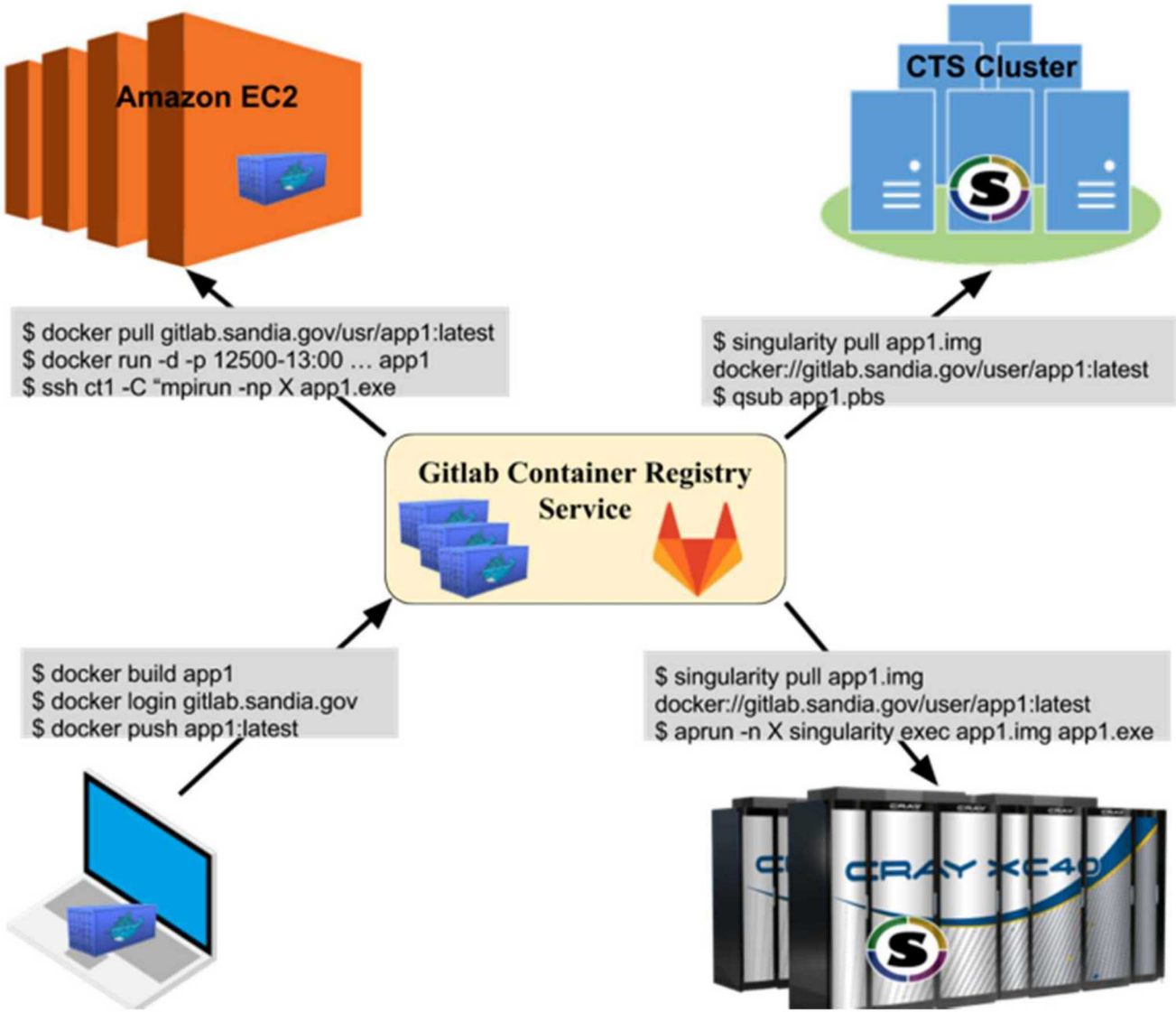
ARM and Containers – will it blend?

- Little understanding of container mechanisms on Arm
  - Especially for HPC
  - It should ‘just work’ - but does it?

- Action Plan:
  - Draw from existing containers & virtualization R&D at Sandia
  - Leverage Astra and various ARM testbeds
  - Develop initial container workflow model
    - Build ARM-based containers?
    - How to map such containers on Astra?
    - Can we deploy mission applications?
  - Will it ~~blend~~ scale?

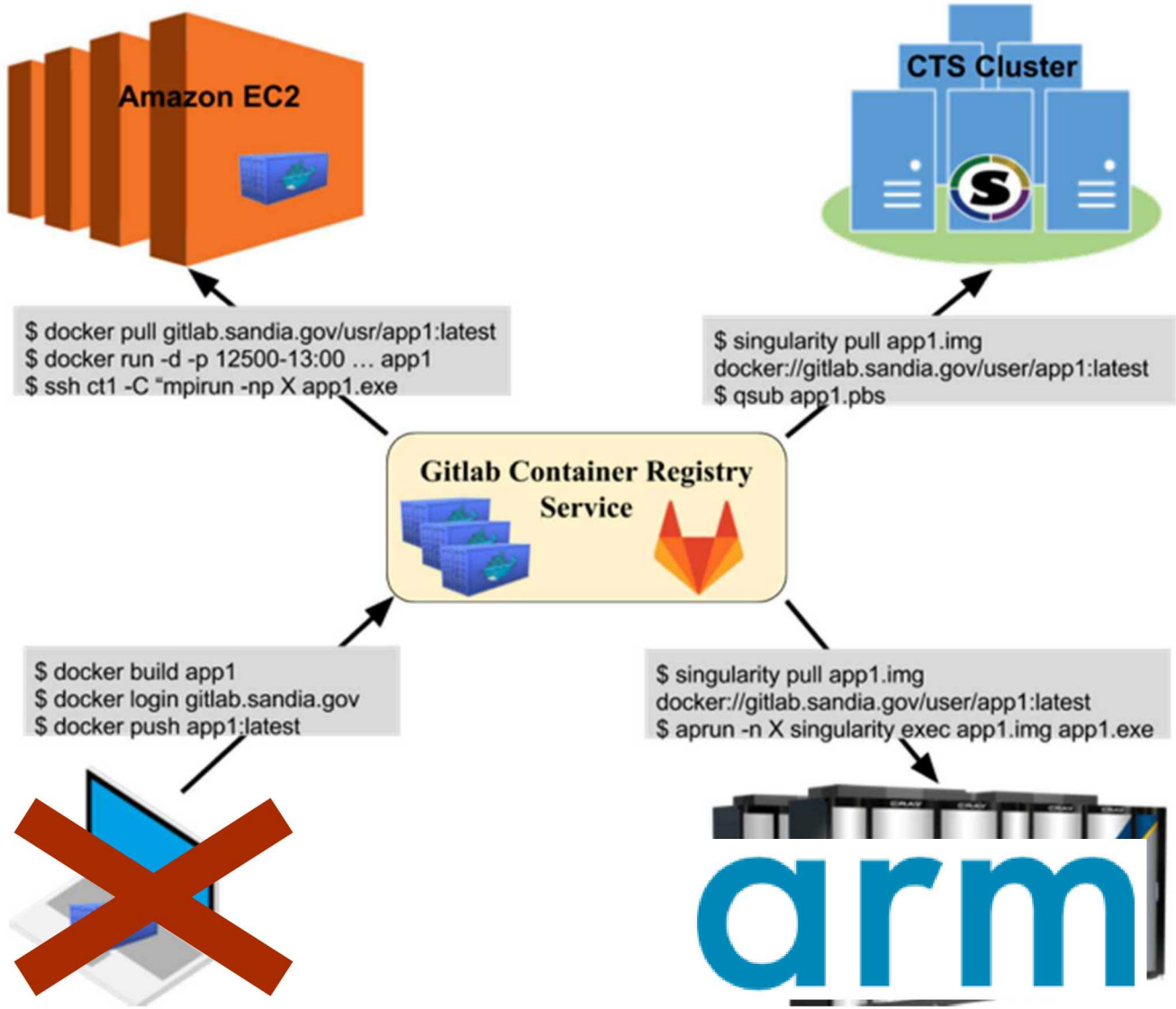
Container DevOps

- 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) and local resource manager (SLURM)
  - Separate networks maintain separate registries



Container DevOps **on ARM**

- 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) and local resource manager (SLURM)
  - Separate networks maintain separate registries



Tracks 1 & 2: Workstations & Remote Container Builder

1. Marvell TX2 build workstations to build ARM containers
   - Inflexible, clunky, isolated
   - Need many workstations

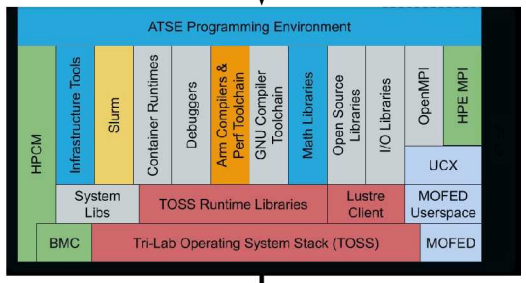
2. Codesigned a Remote Builder for containers with Sylabs
- Enables users to build for alternate architectures:
  - Ex. build AARCH64 container from AMD64 workstation
  - Can be used as part of CI/CD process (GitHub, etc.)
- Builds run natively on alternate architecture, giving great performance
- Centralized 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

Track 3: Podman for Un-privileged Container Builds

- Build containers directly on the supercomputer
  - Doing so requires root level privs
  - root in HPC is bad, Docker is root equivalent
- Leverage user namespaces for _building_ containers
- Podman and Buildah to provide container builds while maintaining user-level permissions
  - CLI equivalent to Docker
  - User namespaces
  - Set uid/gid mappers
  - TBD Overlay & FUSE for mount
- Ongoing Collaboration

  with RedHat

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




```
podman push gitlab.doe.gov/atse/astra:1.2.4
```

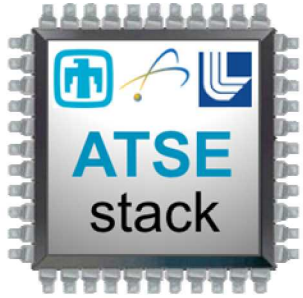
```
singularity build atse-astra-1.2.4.sif docker://gitlab.doe.gov/atse/astra:1.2.4
```


```
salloc -N 2048 && mpirun -np $NP singularity exec atse-astra-1.2.4.sif /app
```

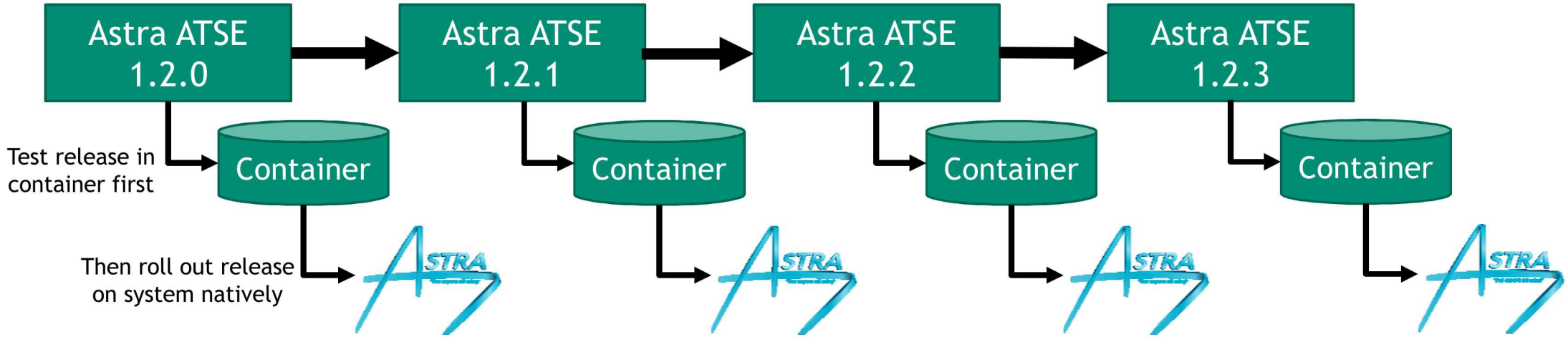
Pedretti, Younge, et. al, *Chronicles of Astra: Challenges and Lessons from the First Petascale Arm Supercomputer* , to appear in SC20

Production Usage for NNSA Mission

System Software Stack Testing & Debugging

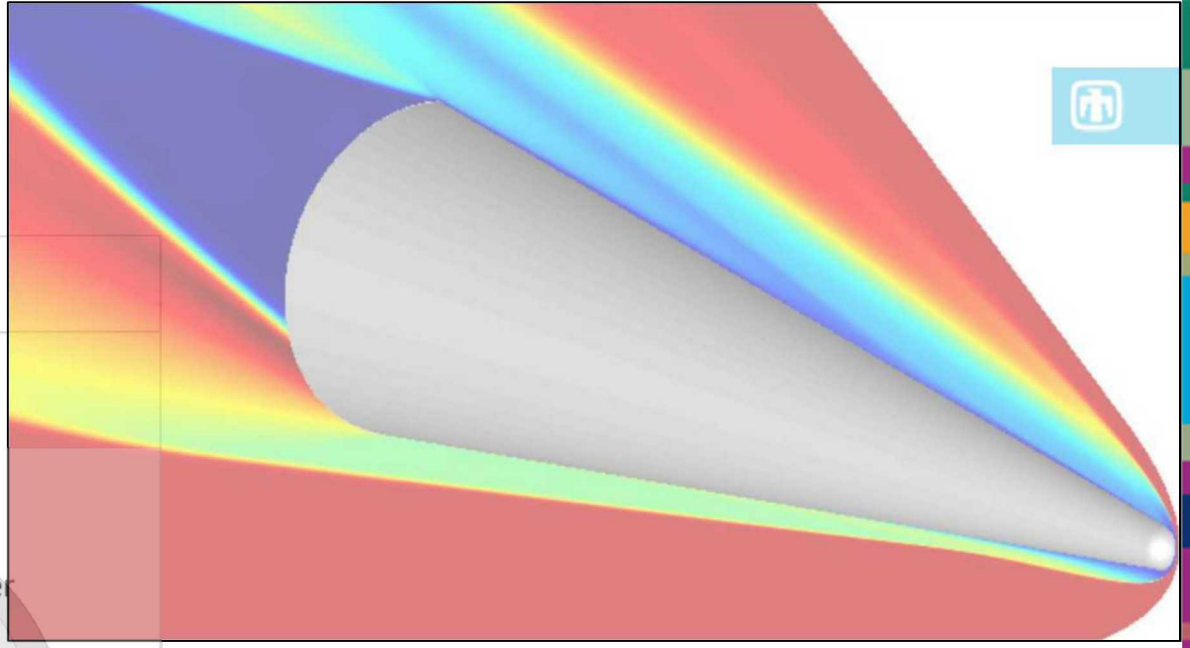


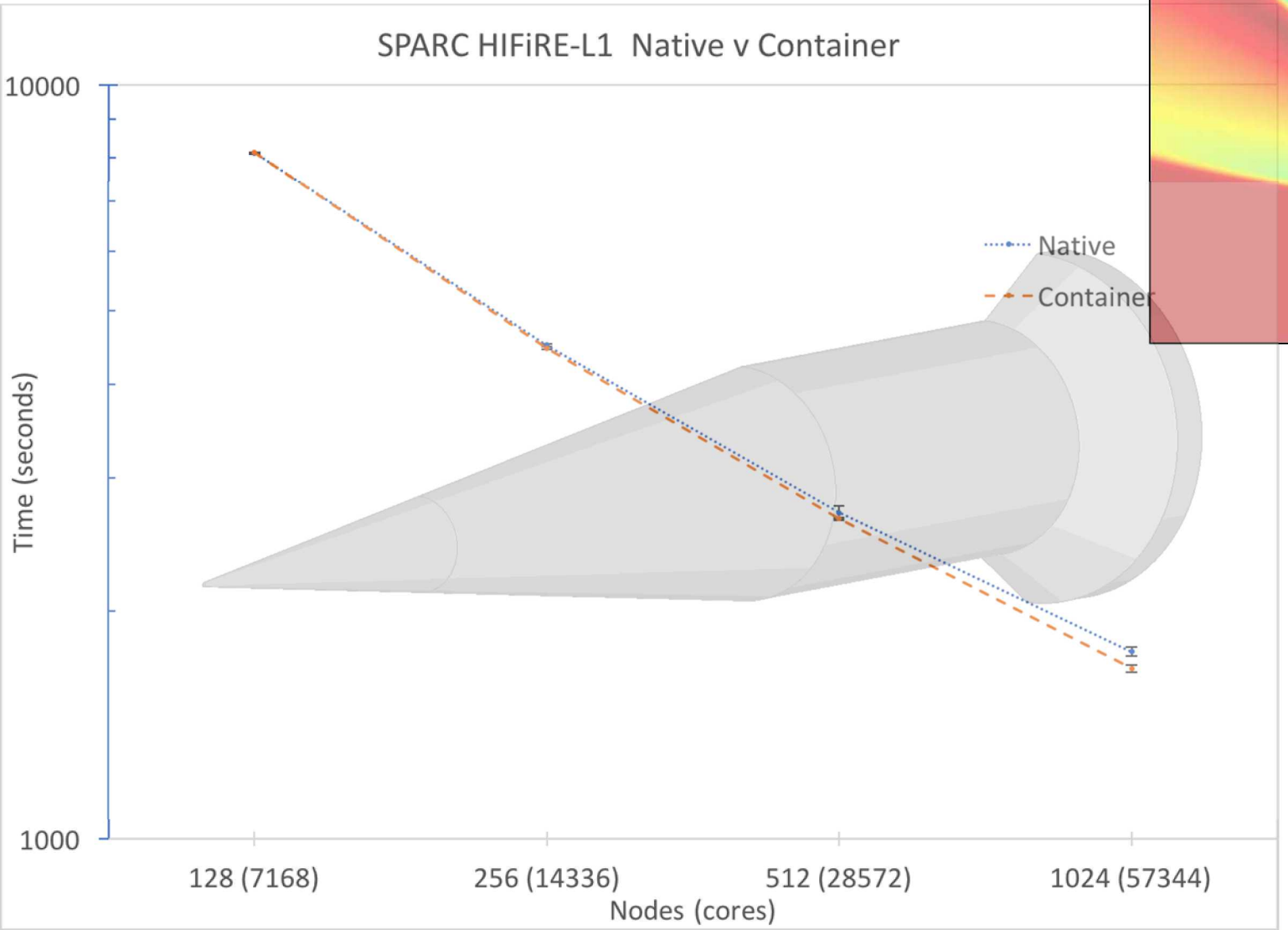
- Astra ATSE programming environment release consists of:
  - TOSS base operating system + Mellanox InfiniBand stack
  - {2 compilers} * {3 mpi implementations} * {~25 libraries} = 150 packages
  - Each release packaged as a container for testing and archival purposes



- 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 1: SNL ATDM App



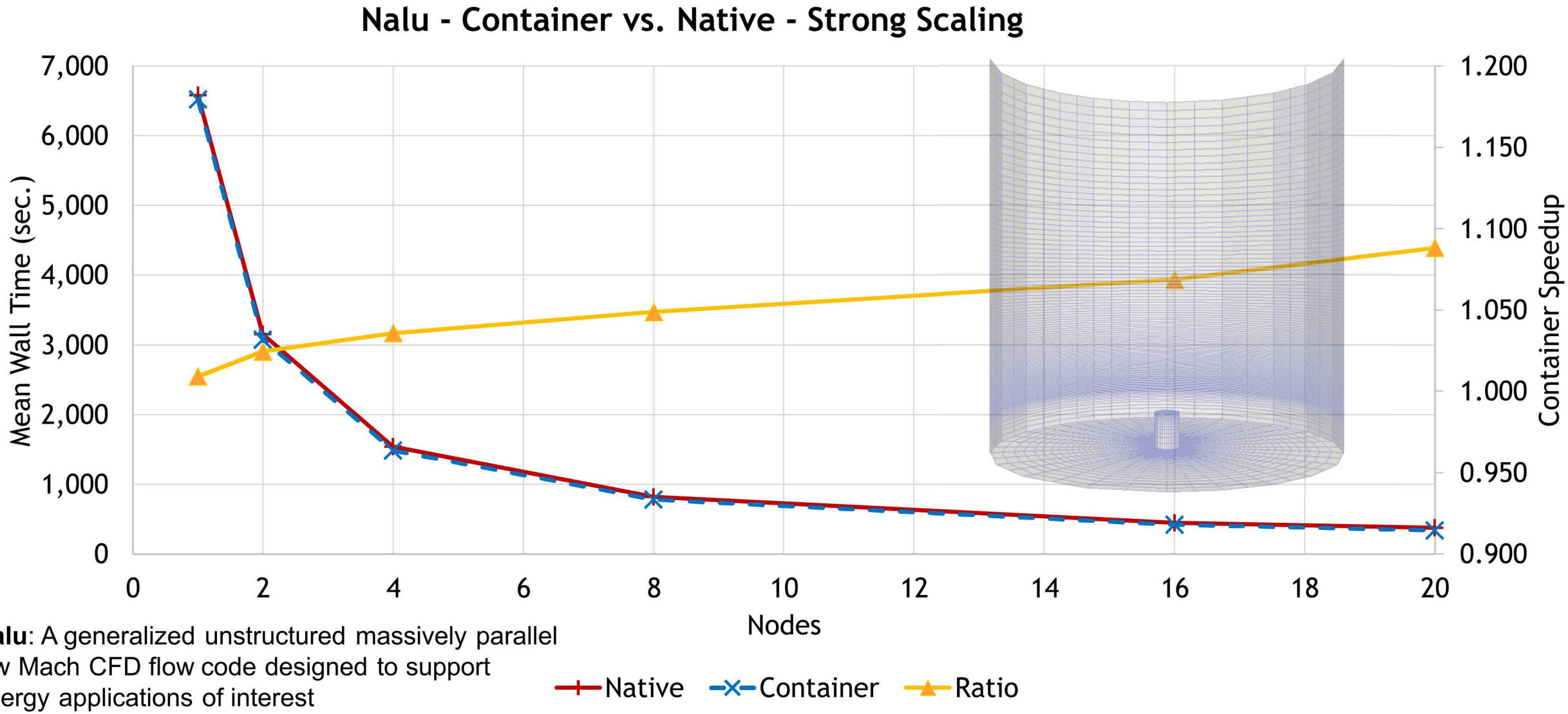


Points:

- 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

Case Study 2: Nalu CFD

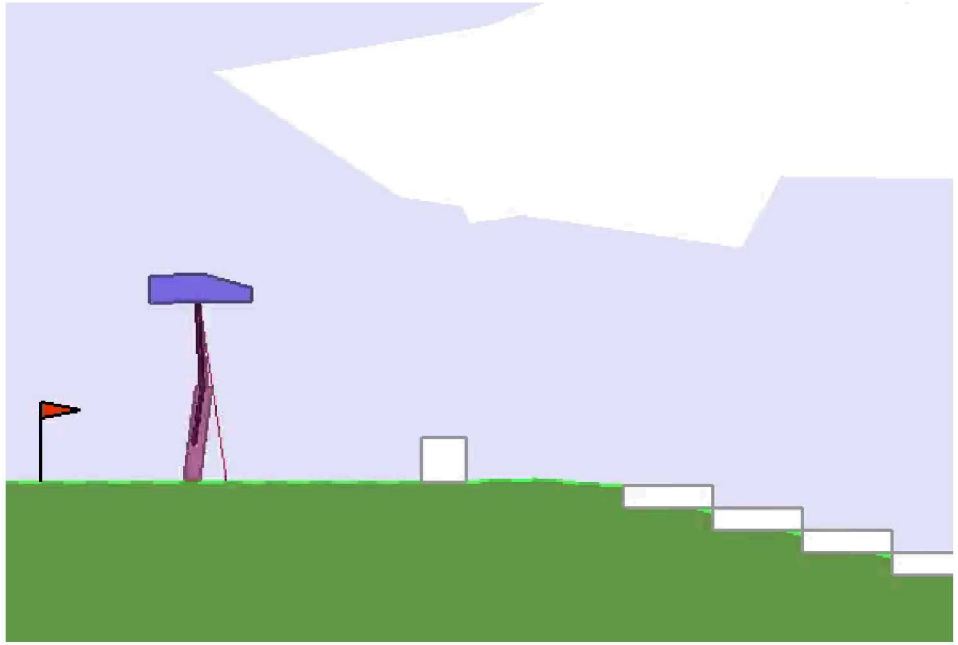


Nalu: A generalized unstructured massively parallel low Mach CFD flow code designed to support energy applications of interest

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/

Containers help support Emerging HPC workloads like Reinforcement Learning

ECP Supercontainers

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 to 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

Emerging workloads on HPC with Containers

- Extreme-scale Scientific Software Stack (**E4S**)
  - Container image contains everything and the kitchen-sink
    - Includes all ECP software activities
  - Lightweight base images now available
- 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
- Working with DOE app teams to deploy custom ML tools in containers
- Investigating scalability challenges and opportunities

Credit: Sameer Shende (U. Oregon)

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

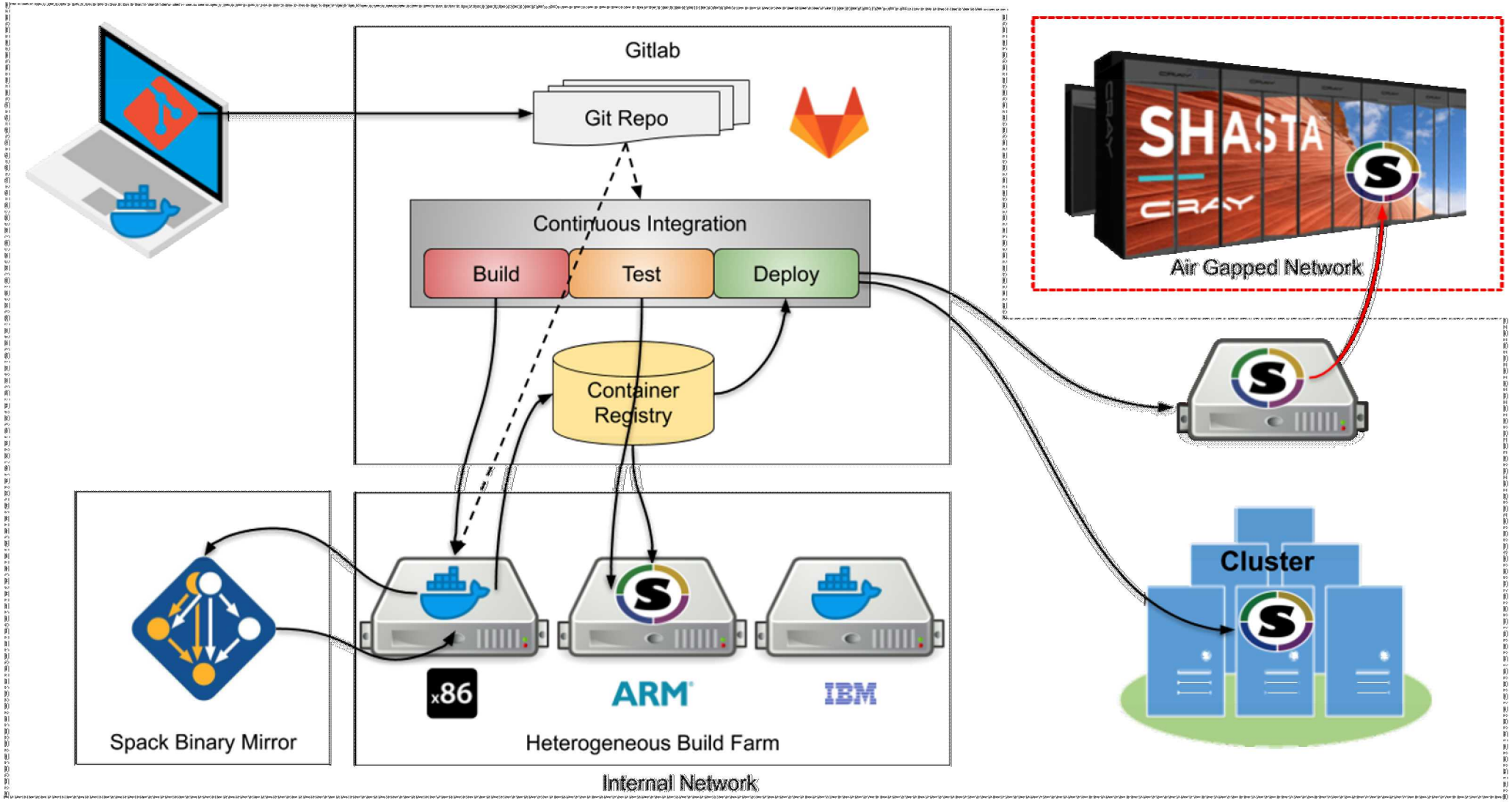
Build with **docker build**.

Run with Singularity
(or some other tool)

Credit: Todd Gamblin (LLNL)

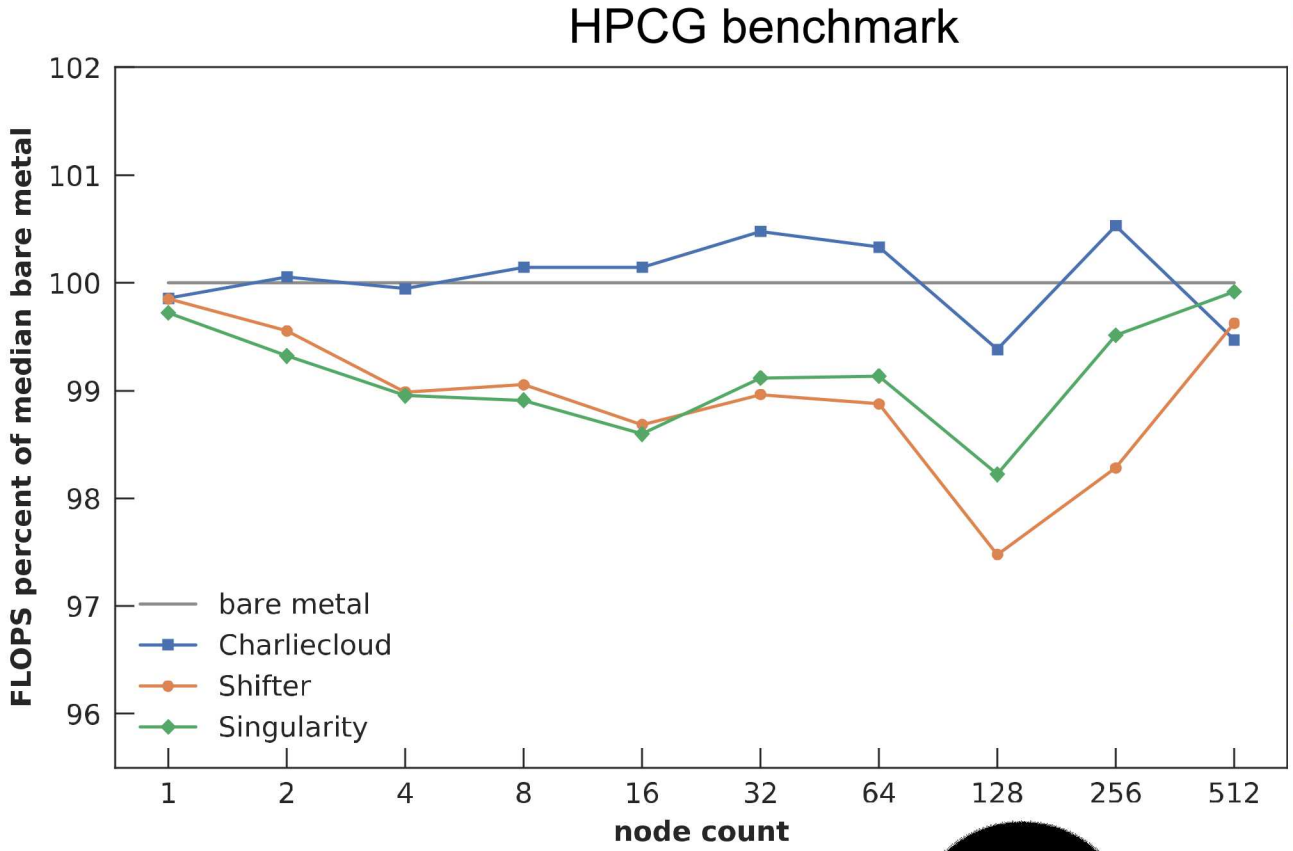
Future Containerized CI Pipeline

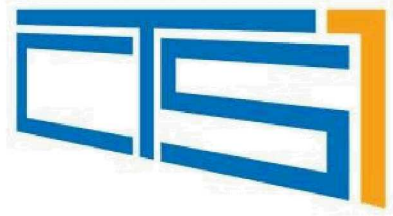
- As a *developer* I want to *generate container builds from code pull requests* so that *containers are used to test new code on target HPC machines*.

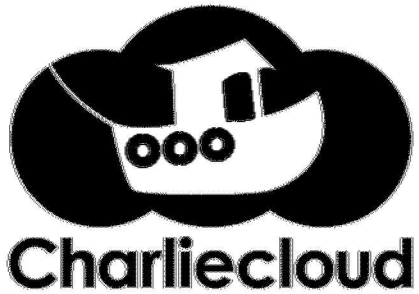


"HPC container runtimes have minimal or no performance impact"

- 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 scale to Exascale







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

Custom OCI Image Labels

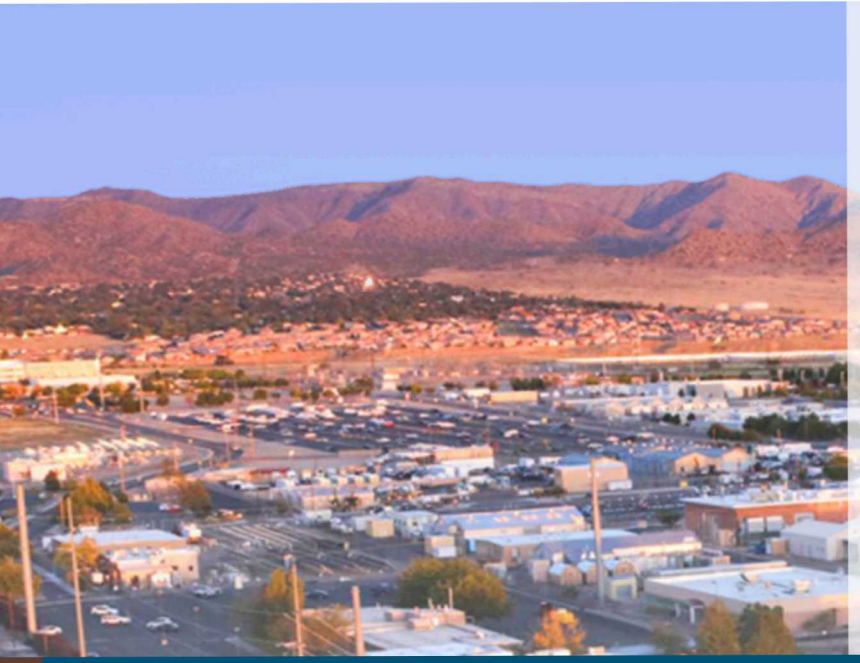
SHIFTER

- 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)

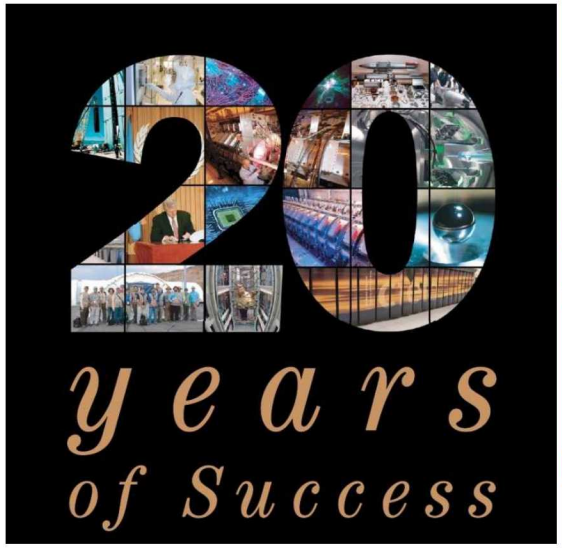


Future Views

[readily available for critique]

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



Container Performance Portability Continuum

How do we strike the right balance?

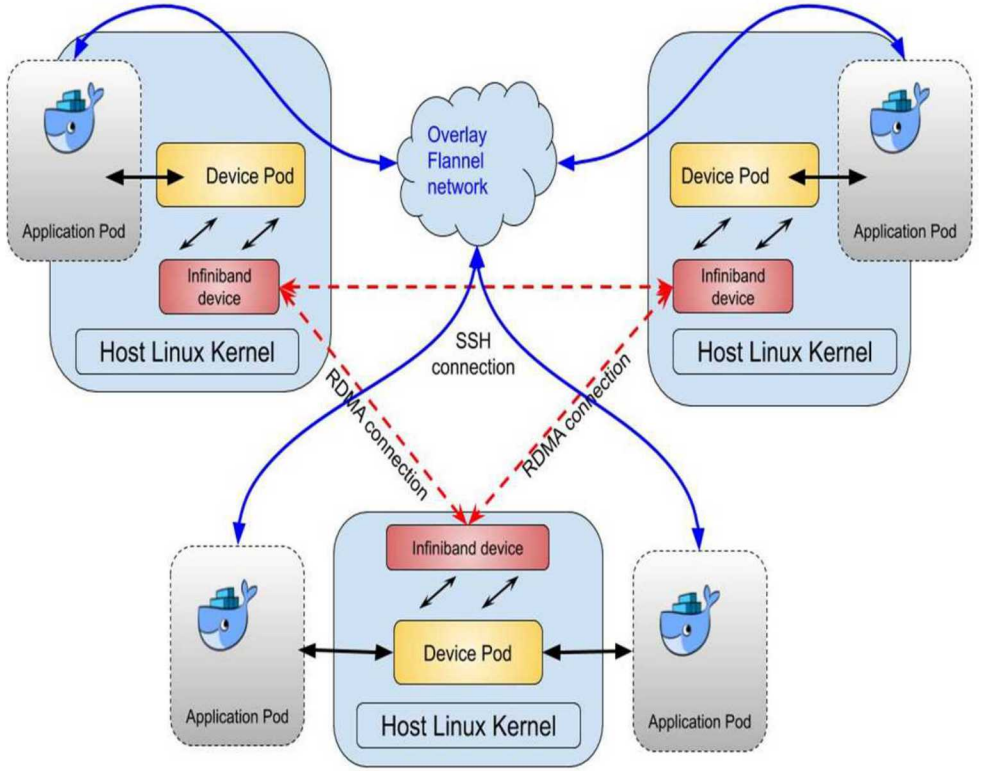
- Portable container images can be moved form one resource deployment to another with ease
- Reproducibility is possible
  - Everything (minus kernel) is self-contained
  - Traceability is possible via build manuscripts
  - No image modifications
- Performance can suffer – no optimizations
  - Can't 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
  - Match accelerator libs to host driver
- Not portable across multiple systems

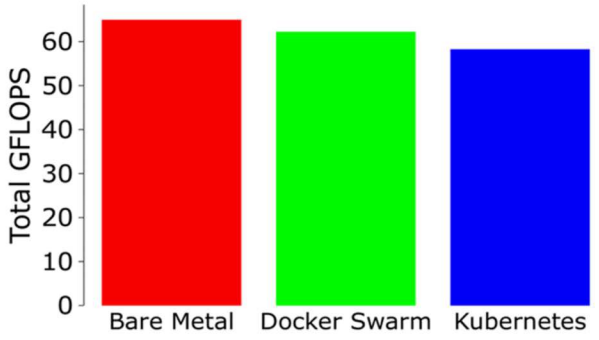
Kubernetes is coming…

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!



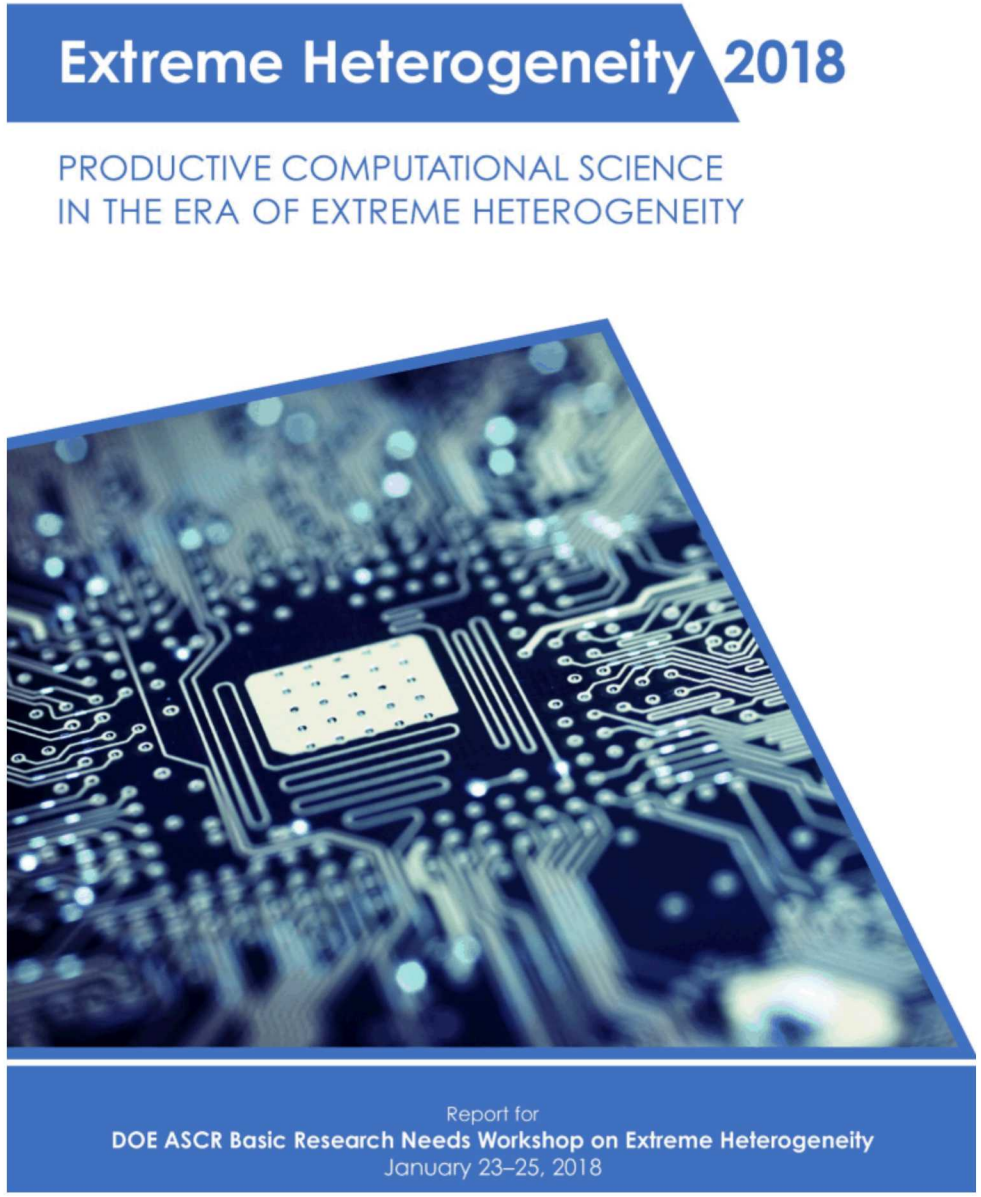
HPCG



Credit: Angel Beltre (Binghamton University)

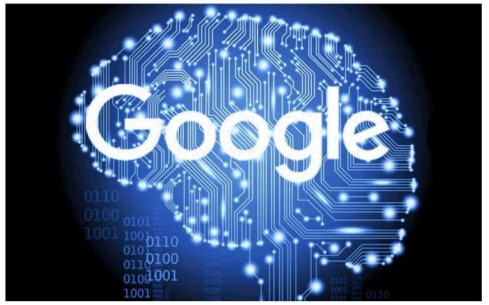
Position 1: Heterogeneity is the Future of HPC…

- 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

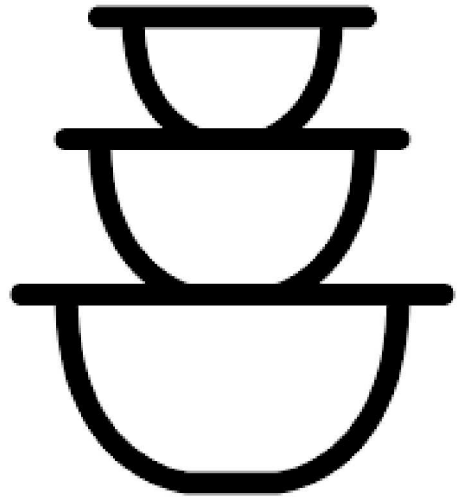
- The hyperscalers are finally paying attention to HPC
  - *"The physical network topology does affect performance; particularly important is the performance of MPI Allreduce, grouped by splitting the mesh by a subset of the dimensions, which can be very efficient [5] [6] if each such group is physically 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 mean 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

Tupperware

Container Takeaways (Tupperware?)



- Use ~~Docker~~ Podman to build manifests of full apps
  - Developers specify base OS, configuration, TPLs, compiler installs, etc
  - Use base or intermediate container images (eg: TOSS RPMs in a container)
  - Leverage container registry services for storing images
  - Import/flatten OCI images into Singularity & run on HPC resources
    - Also works for Charliecloud and Shifter
- Containers have demonstrated minimal overhead for HPC apps
- Enabling On-prem unprivileged containers builds now
  - More to come with Podman & Buildah for HPC
- HPC Container Advantages
  - Simplify deployment to analysts (just run this container image)
  - Simplify new developer uptake (just develop FROM my base container image)
  - Decouple development from dependency release cycles
  - Reproducibility has a new hope?
- HPC Container Caveats
  - ABI compatibility with MPI an ongoing issue
  - Still can’t build an ARM64 container image from my Mac laptop (yet)
  - Containers are currently an option in HPC, not a mandate

Conclusion

- Can containers play a role in HPC? Yes
  - HPC container runtimes work. Many options exist
- Demonstrated containers are viable in HPC
  - Deployments in testbeds to production Petascale
  - Modern DevOps approach with containers is useful
- Confirmed viability on Arm supercomputer
  - Scaled production apps to 2048 nodes on Astra
  - Podman helping expand the ecosystem
- Super*containers* for Exascale
  - Prepare to enable containers at Exascale
  - Simplify HPC application deployment via modern DevOps
  - Support next generation AI & ML apps
- Containers can increase software flexibility in HPC

Acknowledgements:
Kevin Pedretti (1422)
Si Hammond (1422)
Jim Laros (1422)
Anthony Agelastos (9326)
Stephen Olivier (1423)
Justin Lamb (9326)
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 (CTRL CMD)
Eduardo Arango (Red Hat)

Questions?

ajyounge.com - ajyoung@sandia.gov

CANOPIE-HPC Workshop!

- https://www.canopie-hpc.org
  - *2nd International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC) at SC20*
  - Principal venue for leaders in the field to stimulate research and interactions in relation to cutting-edge **container** technologies, **virtualization**, and **OS system software** as it relates to supporting **High Performance Computing** (HPC).
  - Encourage you to submit a paper!
    - Papers due: September 3, 2020
- HPC-Containers Slack channel: http://bit.ly/hpccslack