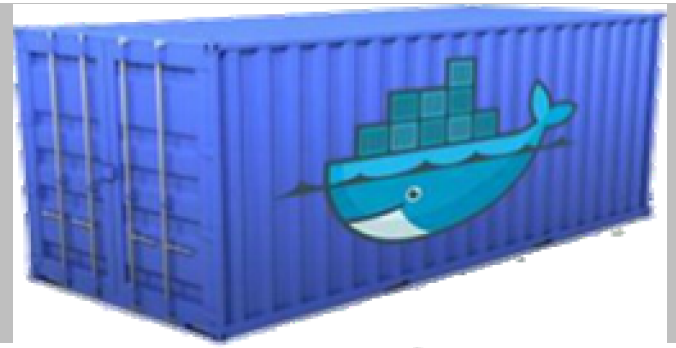
SAND2018-1864C



A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds

Andrew J. Younge, Kevin Pedretti, Ryan E. Grant, Ron Brightwell

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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-NA-0003525.

Outline

Sandia National Laboratories

- Introduction & Background
- Containers for HPC
- Deployment and DevOps
- Singularity on a Cray
- Results
  - Cray XC30 Supercomputer
  - Amazon EC2
- Conclusion

Introduction

- Containers are gaining popularity for software management of distributed systems
- Enable way for developers to specify software ecosystem
- DOE supercomputers need to support emerging software ecosystems
  - Applicable to DevOps problems seen with large HPC codes today
  - Also applicable to new frameworks & cloud platform services
- But HPC systems are very dissimilar from cloud infrastructure…

Background

- Abstracting hardware and software resources has had profound impact on computing
- Virtual Machines to cloud computing in the past decade
  - Early implementations limited by performance
  - HPC on clouds: FutureGrid, Magellan, Chameleon Cloud, Hobbes, etc
  - Some initial successes, but not always straightforward
- OS-level virtualization a bit different
  - User level code packaged in container, can then be transported
  - Single OS kernel shared across containers and provides isolation
  - Cgroups traditionally multiplexes hardware resources
  - Performance is good, but OS flexibility is limited

Containers in Industry

Sandia National Laboratories

- Containers are used to create large-scale loosely coupled services
- Each container runs just 1 user process – “micro-services”
  - 3 httpd containers, 2 DBs, 1 logger, etc
- Scaling achieved through load balancers and service provisioning
- Jam many containers on hosts for increased system utilization
- Helps with dev-ops issues
  - Same software environment for developing and deploying
  - Only images changes are pushed to production, not whole new image (CoW).
  - Develop on laptop, push to production servers
  - Interact with github similar to developer code bases
  - Upload images to ”hub” or “repository” whereby they can just be pulled and provisioned

Container features wanted in HPC

Sandia National Laboratories

- **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 re- sources.
- **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

Sandia National Laboratories

- **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
  - In 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 Options

- Docker:
  - Industry leading tool for packaging & deploying containers
  - Security concerns, root user operation, lack of HPC integration
- Shifter:
  - Developed for HPC, converts Docker images, runs on shared FS
  - Cray-specific deployment, portability across systems may be limited
- Charlie Cloud:
  - Also developed for HPC, small codebase
  - Limited portability or uptake in greater user community
- Singularity:
  - Designed for HPC, uses single image files, user mappings, shared FS support
  - Gaining popularity, simple deployment, supports own SingularityHub containers

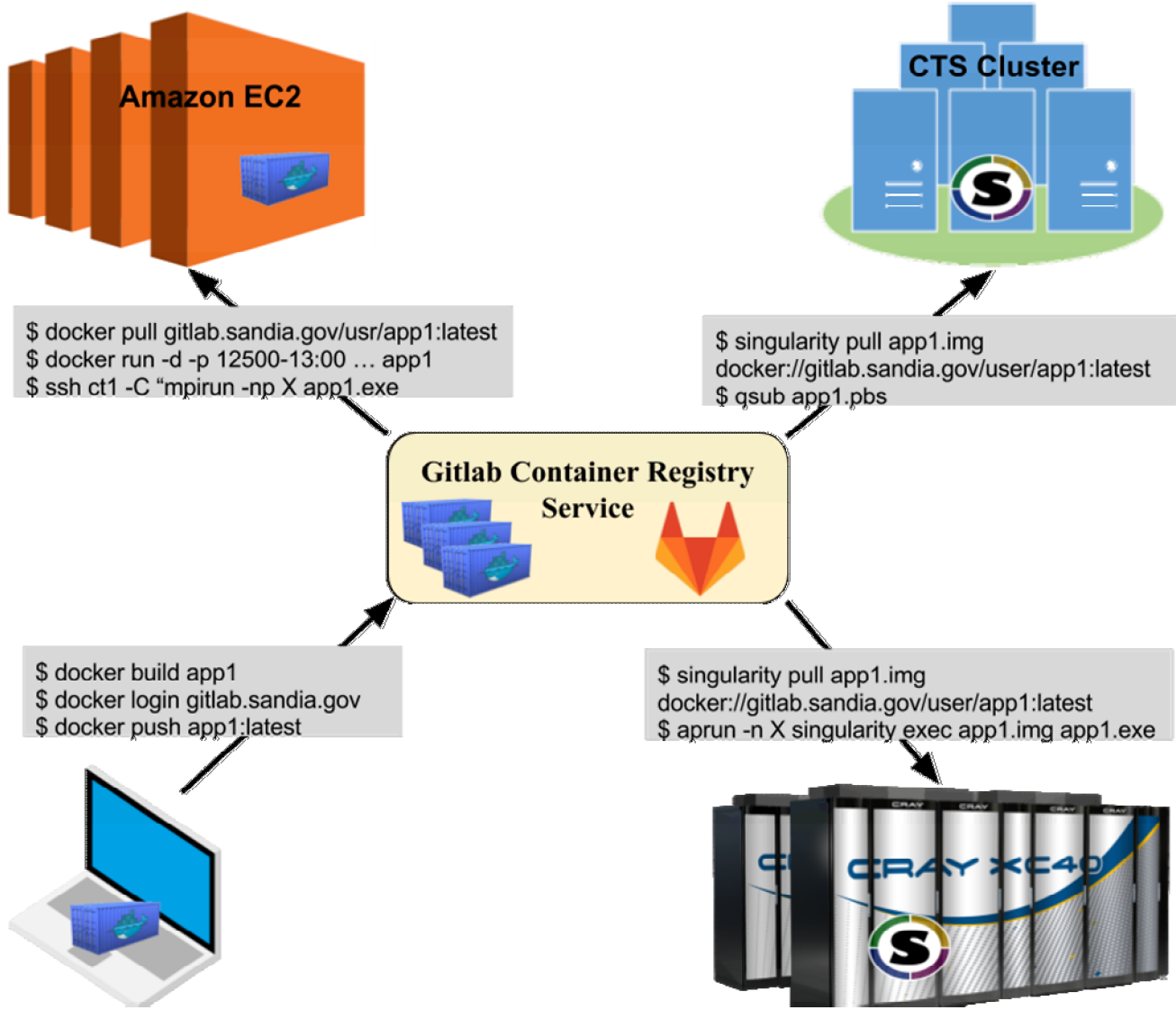
Container Vision @ Sandia

Sandia National Laboratories

- Support software dev and testing on laptops
  - Working builds then can run on supercomputers
  - 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
- Performance matters!
- Want to manage permutations of architectures and compilers
  - x86 & KNL, ARMv8, POWER9, etc
  - Intel, GCC, LLVM

Container DevOps @ Sandia

- Impractical for apps to use large-scale supercomputers for DevOps and/or 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 Gitlab registry services
    - Separate networks maintain separate registries
  - Import to target deployment
    - Leverage local resource manager



Container Builds

- Example Trilinos container build
  - Muelu Tutorial
  - Trilinos on version 12.8.1
- Uses ajyounge/dev-tpl as base container
  - Containers all 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

```
FROM ajyounge/dev-tpl
# Load MPICH from TPL image
RUN module load mpi
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 in Muelu tutorial
RUN ln -s /opt/trilinos/trilinos-build/pkgs/...\
/opt/muelu-tutorial WORKDIR /opt/muelu-tutorial
```

Singularity HPCG container on a Cray

Sandia National Laboratories

- Cray systems can represent pinnacle of HPC
  - 4 of the 10 fastest supercomputers today are Cray (Nov 17 Top500)
- Cray systems are _*very*_ 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

Evaluation

- Singularity containers on a Cray now possible
  - Need to explore the consequences of such a system
  - Evaluate runtime performance of applications
- Created containers with benchmarking tools
  - Intel IMB benchmark
    - Detailed network evaluation
    - Point-to-Point
    - Global AllReduce
  - HPCG
    - “Bookend” of HPC performance
    - Well balanced, and memory intensive
    - Relatable to real-world performance of multi-physics applications

Tale of Two Systems

Sandia National Laboratories

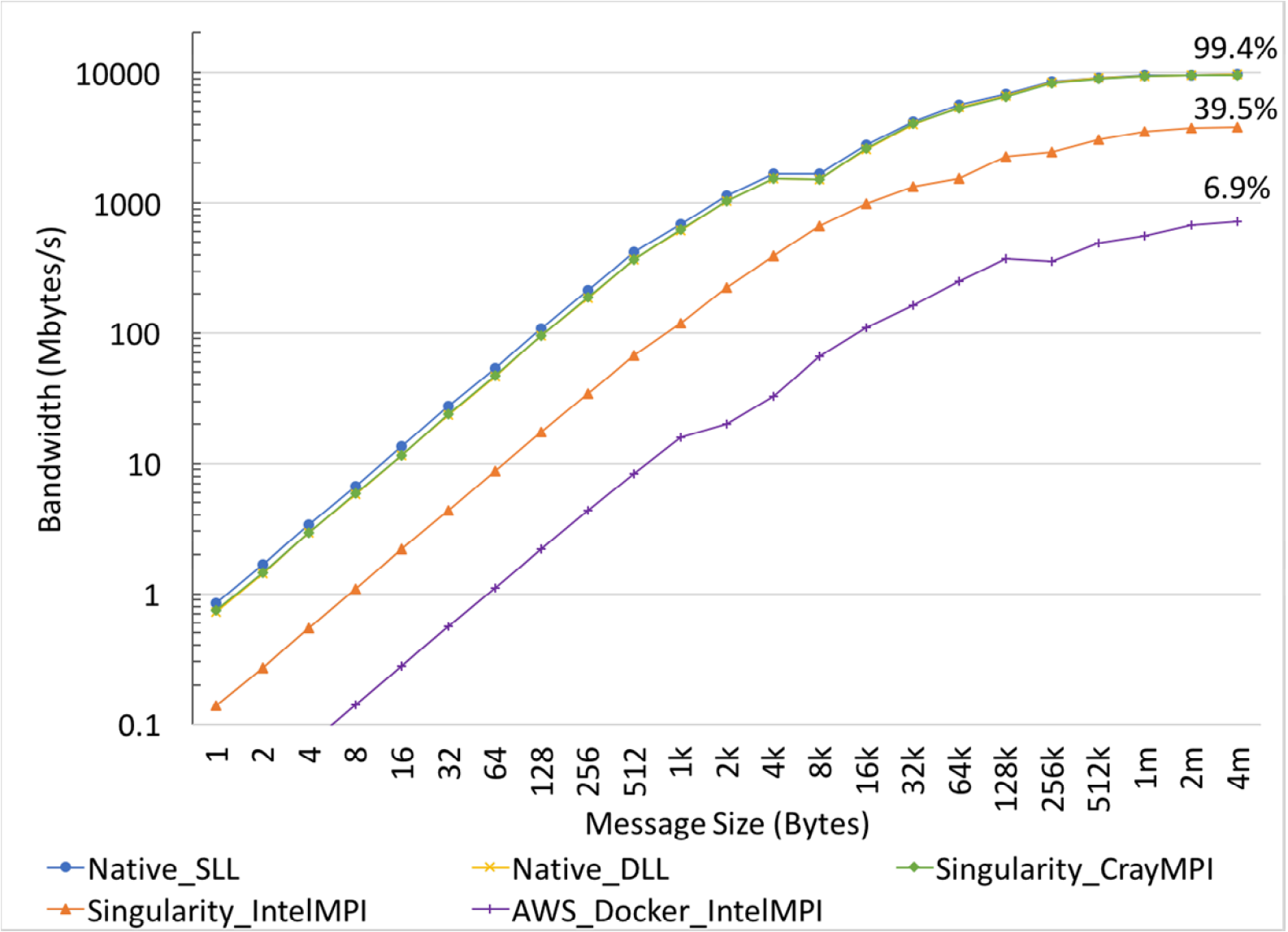
Volta

- 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
- Cray CNL ver. 5.2.UP04
  - Based on SUSE 11
  - 3.0.101 kernel
- 32 nodes used to keep equal core count

Amazon EC2

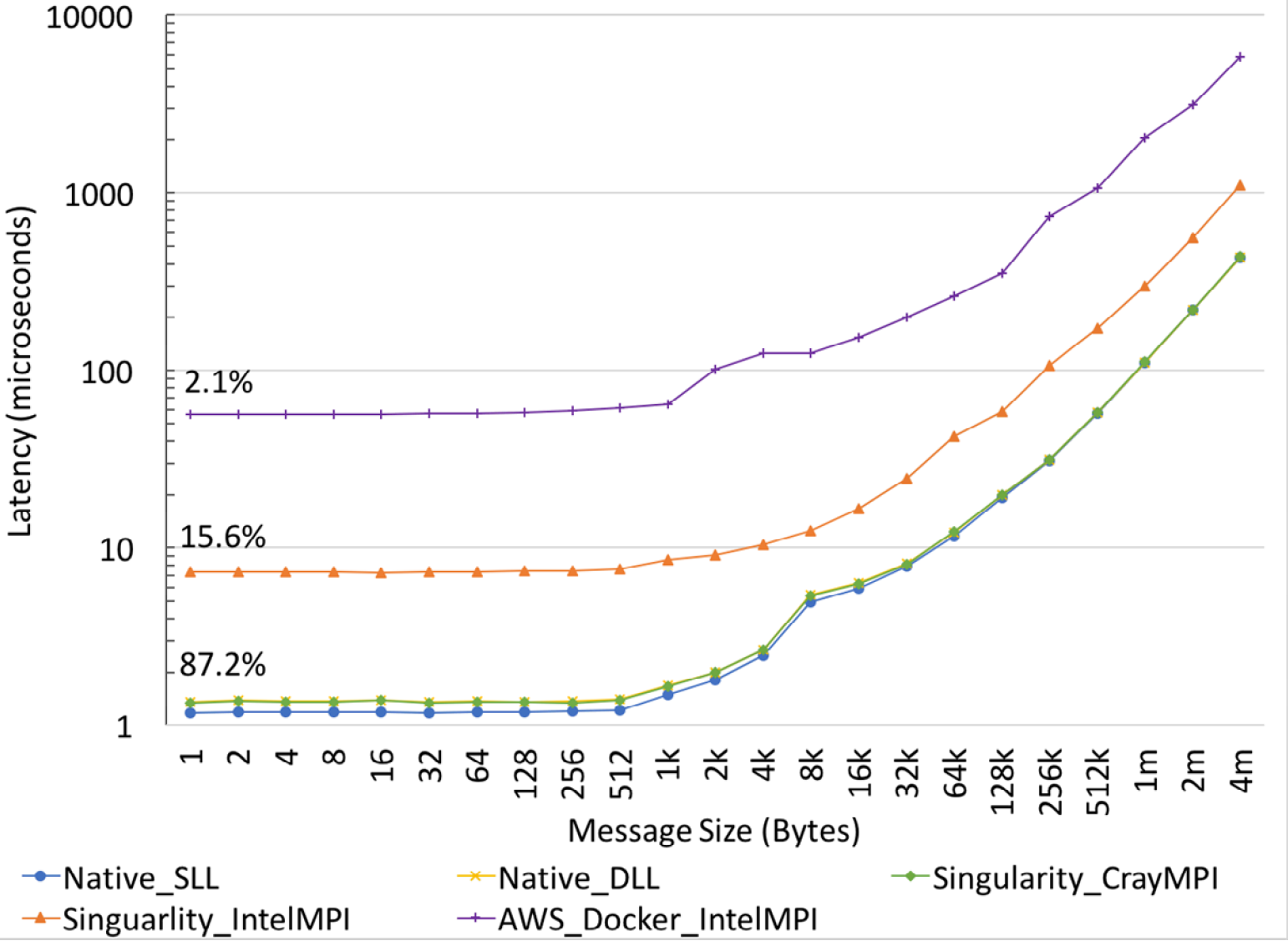
- Common public cloud service from AWS
- 48 c3.8xlarge instances:
  - 2x Intel “IvyBridge” E5-2680 CPUs
  - 16 cores total 32 vCPUs (HT), 2.8Ghz
  - 10 core chip, 2 cores reserved by AWS
  - 60 GB RAM
- 10 Gb Ethernet network w/ SR-IOV
- 2x320 SSD EBS storage per node
- RHEL7 compute image
  - Docker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

IMB PingPong Bandwidth (log scale)



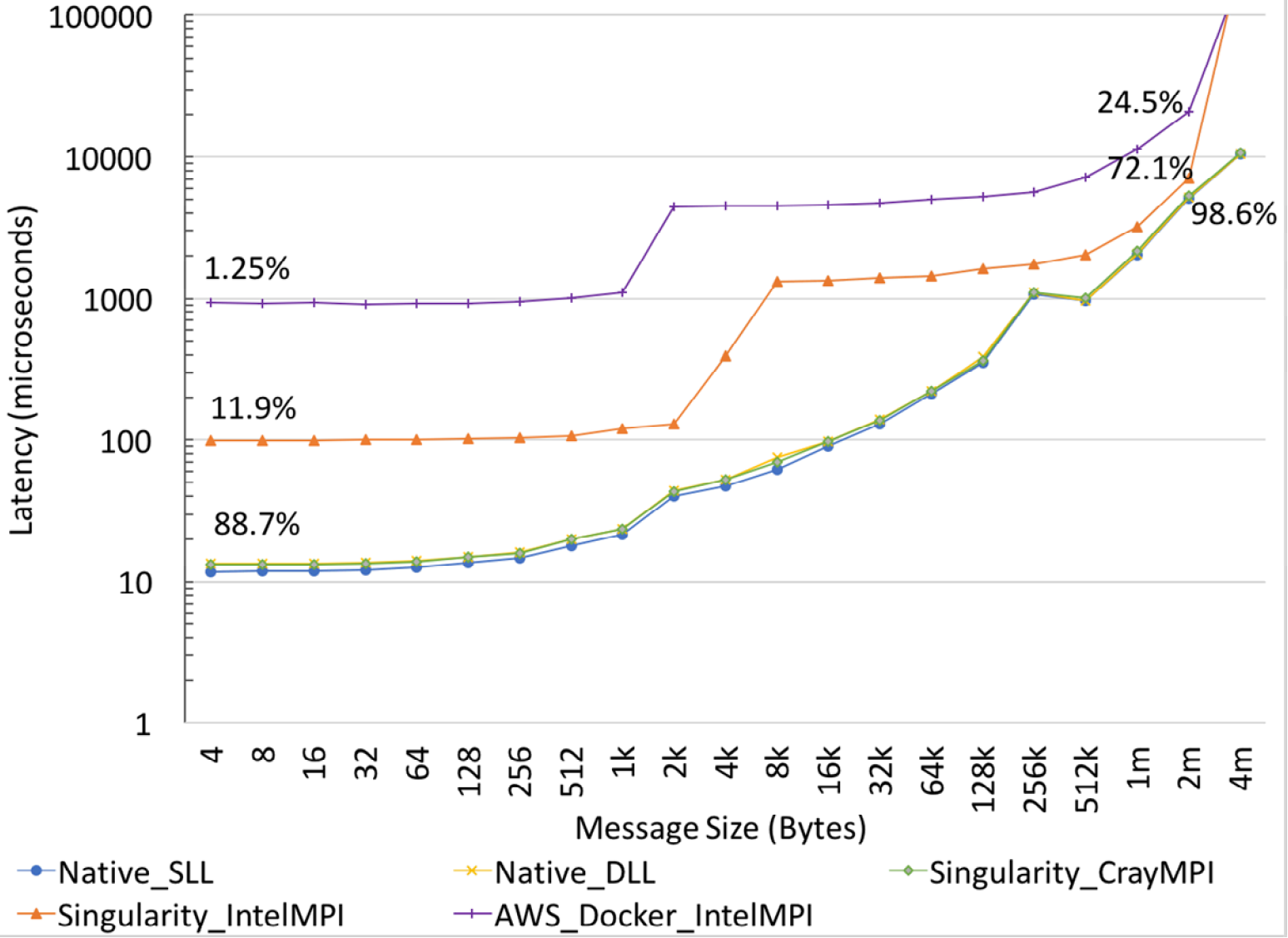
- Aries Interconnect provides peak 96 Gbps bandwidth
- Singularity w/ CrayMPI provides 99.4% of native efficiency
- Singularity w/ IntelMPI drops to only 39.5% of native efficiency
- Amazon EC2 provides peak 7 Gbps bandwidth
  - Just 6.9% of native Aries
- Illustrates massive difference between commodity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

IMB PingPong Latency (log scale)



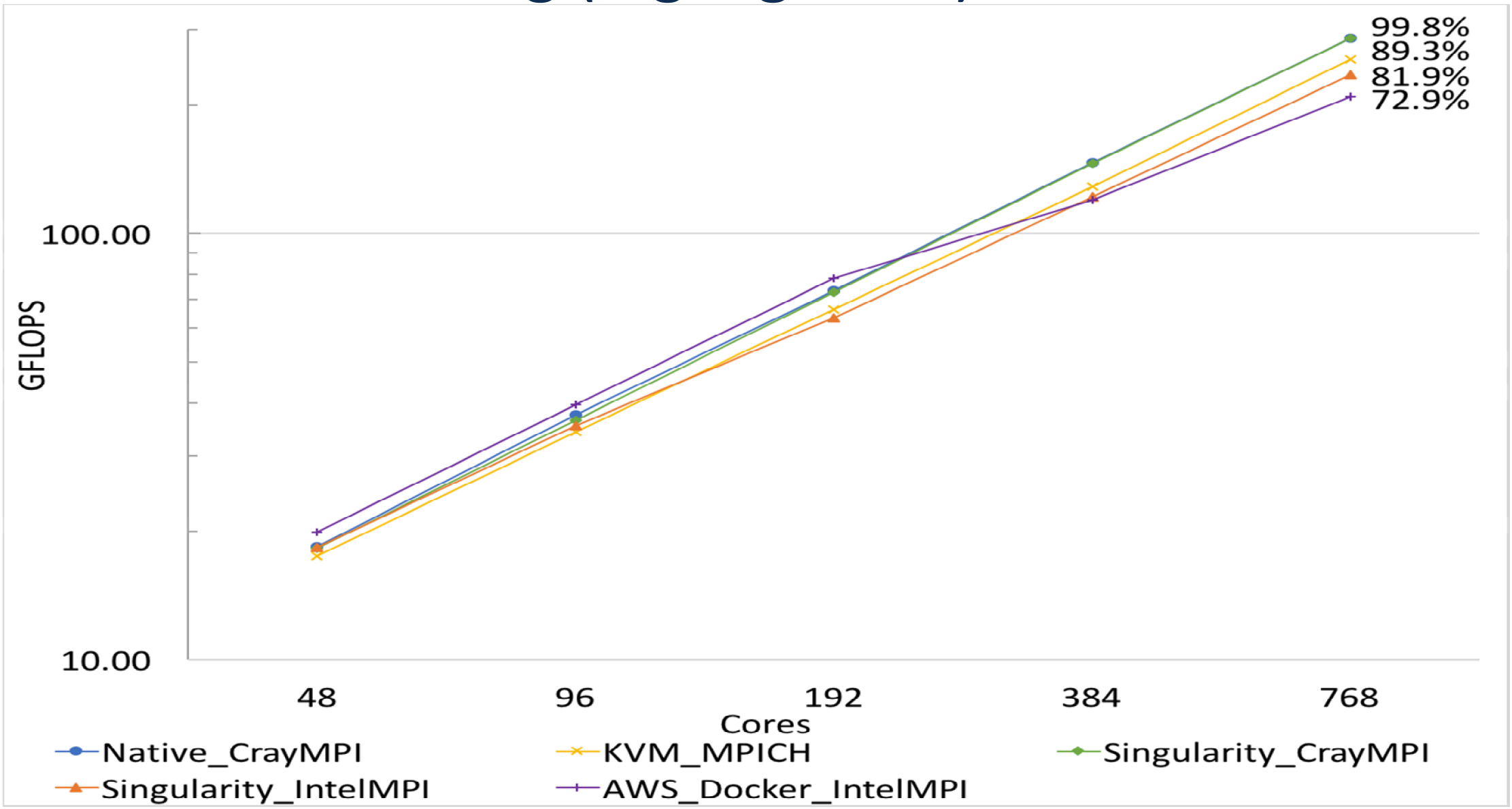
- Aries native latencies around 1.1 to 1.3 microseconds
  - ~200 ns = static linking
- Singularity w/ CrayMPI achieves 87.2% of native latency
  - Overhead = Native DLL
- Singularity w/ IntelMPI only achieves 15.6% efficiency
  - ~7 microseconds
- Amazon EC2 offers just 2.1% of native efficiency
  -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

HPCG Weak Scaling (log log scale)



Discussion

- Containers in HPC are different than containers in Industry
  - Running Docker alone is unacceptable
  - Need for HPC-centric containerization efforts
- Developing DevOps models for custom software ecosystems
- 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

Conclusion

- There is a growing need for containerization within HPC
- Evaluated the feasibility of extending HPC system usability through containerization
  - Trilinos-based containers
  - DevOps models for HPC apps
- 1st demonstration of Singularity on Cray supercomputer
  - Found near-native performance of Singularity when using CrayMPI
  - Dynamic linking is necessary
  - IntelMPI in containers not nearly as performant
- Amazon EC2 may be useful for small-scale development and testing, but beyond small-scale efficiency decreases significantly