Initial Experiences with Deploying Singularity on a Cray XC Supercomputer

Andrew J. Younge, Kevin Pedretti

{ajyoung,ktpedre}@sandia.gov
Outline

- Overview of Containers
- Containers in HPC
- Why Singularity?
- HPC containers @ Sandia
  - Trilinos & ATDM apps
  - HPCG
- Dev-ops Mechanisms
- Initial Benchmarking
- Conclusion
What are Containers?

- “An object that can be used to hold or transport something.”
- A way to package necessary components of running applications.
  - Libraries, software, files, environment settings, etc.
- OS-level virtualization
  - Relies on 1 OS kernel – aka “chroot on steroids”
  - cgroups for resource isolation, namespaces for process isolation, chroot for filesystem isolation.
- Different than Host Virtualization
  - Single OS Kernel that does all the hard work
Containers in Industry

- Containers are used to create large-scale loosely coupled services
- Each container runs 1 user process – “micro-services”
  - 3 httpd containers, 2 DBs, 1 logger, etc
- Scaling achieved through load balancers and 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

- Developers prescribe running software environment
  - “Bring-your-own-environment”
  - Not bound by vendor software delivery
  - Not bound by sysadmin support for additional libraries
  - Developers know best how to run, let users just specify it
- Easy definition of application compilation & runtime setup
- Integration with github or other dev environments
- Could enable better portability between architectures
Container features not wanted in HPC

- Overhead – cannot slow down advanced architecture supercomputers beyond reason
  - Posit: <5% maybe ok, any more is big problem
- Micro-services support and on-node resource partitioning
  - Don’t need cgroups to slice up individual compute nodes
  - Not running services, but real applications
- Running as root!
- Networking aspects can be left out
Container Vision @ Sandia

- Support software dev and testing on laptops which create working builds that can run on HPC machines
  - May also leverage VM/binary translation
- Let developers specify how to build the environment AND the application
  - Users just import container and run on target platform.
  - Many containers, but can have different code “branches” for arch, compilers, etc.
  - Not bound to vendor and sysadmin versions & release cycles
- Want all the performance
- Want to manage permutations of architectures and compilers
  - X86 & KNL, ARM, POWER9, etc.
  - Intel, GCC, LLVM
Why Singularity?

- Singularity is a simple container solution created by LBNL
- Based on singleton container images
  - Not layered AUFS images or dev-mapper insanity
  - Image sharing & management made easy
- Provides user namespaces
  - User ajyounge on HPC system maps to ajyounge in container
  - Running as root on HPC resources not allowed!
- Site filesystems can also be mounted
  - Bring in MPI libs or tuned libraries, etc
- Integration with existing scheduling systems
  - Make binaries available on compute nodes
- No Vendor lock-in. Want portable HPC container solution
  - Supported in OpenHPC via 1.3.1 release last week
Singularity on Cray XC-series

- Crays are special machines
  - Cray CNL is read-only image with tmpfs mounts
  - Lustre or NFS over Cray DVS filesystem
  - Specialized Linux kernel w/out standard feature sets
- Had to modify CNL to build in necessary kernel features
  - XC30 runs 3.0.101 kernel (old)
  - Rebuild Cray image with build-in features
    - Loopback device support and EXT3
- Provision new CNL to interactive nodes and compute nodes
  - Similar to KVM on Cray effort (Related Work)
  - “Enabling Diverse Software Stacks on Supercomputers using High Performance Virtual Clusters”
Container Build #1: Trilinos Muelu

- Trilinos provides math library packages for many applications of interest @ Sandia
- Trilinos itself depends on numerous 3rd party libraries
- Can condense complex compilation steps down to just a simple Dockerfile
  - Predictable & stable environment across deployments
  - Enables testing across multiple architectures and validation of TPL changes

```
FROM ajyounge/dev-tpl

WORKDIR /opt/trilinos
# Copy files to image
COPY do-configure /opt/trilinos/
# Download Trilinos source tarball
# Extract Trilinos source file & load mpi library
RUN tar xf /opt/trilinos/trilinos.tar.gz -C /opt/trilinos/
RUN rm -f /opt/trilinos/trilinos.tar.gz
RUN mv /opt/trilinos/trilinos-12.8.1-Source /opt/trilinos/trilinos
RUN mkdir /opt/trilinos/trilinos-build
RUN module load mpi

# Compile Trilinos
RUN /opt/trilinos/do-configure
RUN cd /opt/trilinos/trilinos-build && make -j 3
# Link in a tutorial directory, and then set the workdir
RUN ln -s /opt/trilinos/trilinos-build/packages/muelu/doc/Tutorial/src /opt/muelu-tutorial
WORKDIR /opt/muelu-tutorial
CMD ["/bin/bash"]
```
Container Build #2: HPCG

- Straight-forward container build for HPCG
- Use Centos7 image, install basic software
- Install Intel parallel studio 2017
  - Silent configuration
  - Pull in site license or use trial
  - Clean up install files (>8GB)
- Extract and build HPCG 3.0 with cxx = mpiicpc

FROM centos:7.2.1511
ARG intel_file=parallel_studio_xe_2017_update2
# Dependencies and MPICH
RUN yum update -y && yum groupinstall -y "Development Tools"
RUN yum install -y mpich-3.2 mpich-3.2-devel redhat-1sb

# Intel compiler install
COPY $intel_file.tgz /
RUN tar xvfz /$intel_file.tgz
RUN mkdir -p /opt/intel/licenses
COPY USE_SERVER.lic /opt/intel/licenses/
# Silent configuration installation
COPY silent.cfg /$intel_file/silent.cfg
RUN /$intel_file/install.sh --silent /$intel_file/silent.cfg
RUN echo "source /opt/intel/bin/compilervars.sh intel64" >> /etc/bashrc
RUN rm -rf /$intel_file && rm /$intel_file.tgz

#Build and HPCG
COPY hpcg-3.0.tar.gz /opt/
RUN tar xvfz /opt/hpcg-3.0.tar.gz -C /opt/
COPY Make/Linux_intel_mpich /opt/hpcg-3.0/setup/
RUN mkdir -p /opt/hpcg-3.0/Linux_intel_mpich/
WORKDIR /opt/hpcg-3.0/Linux_intel_mpich
RUN ../configure Linux_intel_mpich
RUN /bin/bash -c "source /opt/intel/bin/compilervars.sh intel64 && make"
CMD ["/bin/bash"]
Dev-ops Pathway

Gitlab Container Registry → Singularity Server → Cray Login Server

Cray CNL

Lustre / NFS
Dev-ops Pathway

Gitlab Container Registry → Singularity Server → Cray Login Server

Lustre / NFS

Cray CNL

lap$ docker login gitlab.sandia.gov
lap$ docker build .
lap$ docker tag 0e5574283393 ajyounge/hpcg-container
lap$ docker push ajyounge/hpcg-container:latest
Dev-ops Pathway

Gitlab Container Registry → Singularity Server → Cray Login Server

ss$ sudo singularity create --s 12G hpcg-container.img
ss$ sudo singularity import hpcg-container.img
docker://gitlab.sandia.gov/ajyounge/hpcg-container:latest

lap$ docker login gitlab.sandia.gov
lap$ docker build .
lap$ docker tag 0e5574283393 ajyounge/hpcg-container
lap$ docker push ajyounge/hpcg-container:latest
Dev-ops Pathway

ss$ sudo singularity create –s 12G hpcg-container.img
ss$ sudo singularity import hpcg-container.img
docker://gitlab.sandia.gov/ajyounge/hpcg-container:latest

lap$ docker login gitlab.sandia.gov
lap$ docker build .
lap$ docker tag 0e5574283393 ajyounge/hpcg-container
lap$ docker push ajyounge/hpcg-container:latest

cray$ scp ss:-/hpcg-container.img .
cray$ aprun –n 24 –L 62,63 singularity exec
hpcg-container.img ./xhpcg
**Dev-ops Pathway (new)**

```
lap$ docker login gitlab.sandia.gov
lap$ docker build .
lap$ docker tag 0e5574283393 ajyounge/hpcg-container
lap$ docker push ajyounge/hpcg-container:latest
```

```
cray$ singularity pull --name hpcg.container.img
docker://gitlab.sandia.gov/ajyoung/hpcg-container:latest
```
Singularity + Cray Interconnect

- Using Volta testbed – Cray XC30 IvyBridge
- Container using TCP/IP – no changes necessary
  - Use Cray’s IP-over-Aries Ethernet device (ipogif0)
  - Better than 10gb Ethernet performance (~32Gbs)
- Intel MPI not optimized for Aries network
- Bring Cray’s MPI implementation in container
  - Mount /opt/cray
  - Mount /var/opt/cray
  - Set LD_LIBRARY_PATH accordingly in container

```
cray$ aprun -n 24 -L63 singularity exec hpcg-container.img /bin/bash -c "export
LD_LIBRARY_PATH=/opt/cray/ugni/6.0-1.0502.10863.8.29.ari/lib64:/opt/cray/xpmem/0.1-
2.0502.64982.5.3.ari/lib64:/opt/cray/pmi/5.0.11/lib64:/opt/cray/udreg/2.3.2-
1.0502.10518.2.17.ari/lib64:/opt/cray/mpt/7.5.1/gni/mpich-intel-abi/16.0/lib:/opt/cray/alps/5.2.4-2.0502.9822.32.1.ari/lib64:/opt/cray/wlm_detect/1.0-
1.0502.64649.2.1.ari/lib64:/opt/intel/lib/intel64:$LD_LIBRARY_PATH & & /opt/hpcg-
3.0/Linux_intel_mpich/bin/xhpcg"
```
HPCG Efficiency

HPCG Single-node Performance (24 cores, 2 sockets)

- Native: 99.1%
- KVM: 96.4%
- Singularity: 99.1%
HPCG Performance Summary

- Singularity presents near-native runtime performance
  - KVM also good, but has a little more overhead (likely due to IntelMPI)
- Scaling results TBD, but expect the same
  - KVM scales 90% of native @ 786 cores, Singularity will be better
  - Using Cray MPI & Aries Interconnect is a key feature to getting near-native performance
  - Staying ABI compatible for MPI is mandatory
- Image deployment is most likely source of overhead
  - Scalability of mounting lookback images on Lustre/NFS?
  - Read-only helps, but may not solve all problems
Conclusion

- Singularity works on Cray XC series supercomputers
  - Modifications to CNL necessary
  - Performance is near-native
- Additional features needed for clean deployment
  - Site-specific ENV variables
  - OverlayFS
- Performance near-native with HPCG
  - Not surprising
  - Using Cray MPI and ABI compatibility
- Singularity is ideal for HPC interoperability
Future Considerations

- Container storage at scale
- How to use other tuned libraries and site-specific software
  - ABI compatibility?
- Can the HPC community agree on container interoperability?
  - Image formats, manifests, etc.
- Multi-architecture support
- Vendor support for laptop development?
Thanks!

ajyoung@sandia.gov
Backup Slides

Exceptional service
in the national interest

S

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.
Enabling Diverse Software Stacks on Supercomputers using High Performance Virtual Clusters

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

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

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-NA0003525.

Approved for public release; further dissemination unlimited.
Intra-node communication – Cray MPI vs Intel MPI (KVM)
XC30 HPCG KVM Scaling