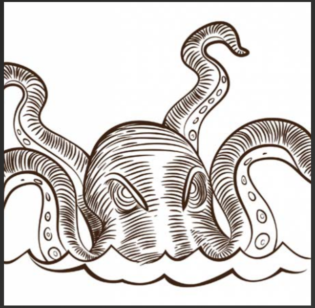
Los Alamos
NATIONAL LABORATORY
EST. 1943

NNSA
National Nuclear Security Administration



slurm
workload manager

docker

Charliecloud

Los Alamos
NATIONAL LABORATORY
EST. 1943

Delivering science and technology
to protect our nation
and promote world stability

LA-UR-18-29031

Slurm in a container only world

Are we crazy?

Paul Peltz and Lowell Wofford

9/25/2018

Why Containers?

- **Users are demanding it for Crossroads**
  - Reproducibility
  - Supportability
  - Consistency
- **System Support staff as well?**
  - Support burden shifts to the user

Containers Pros/Cons

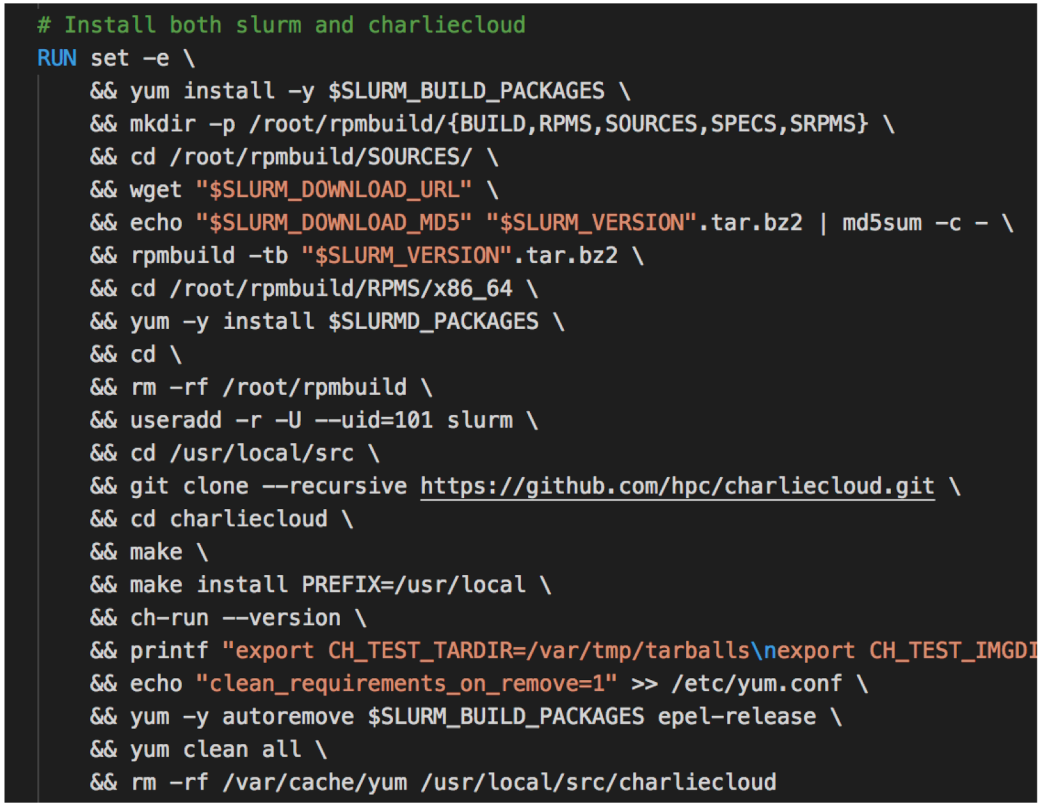
- **Pros**
  - Boot once, Run forever*
  - Rolling Upgrades!
- **Cons**
  - Multi-Service containers
  - Containers within containers can be tricky
- **Unknowns**
  - Containers as base OS never been attempted?
  - Memory usage?
  - Security Implications?

Multi-Service Containers

- **Interesting Use Case**
  - Systemd or not?
    - Systemd requires additional privileges to run
    - Probably can't do it with u-root's unshare
  - Supervisord
    - Service manager for containers
    - Many of the same controls for managing services
    - Unit-like file definitions
  - Slurmd

```
[program:slurmd]
command=/usr/sbin/slurmd -Dvvv
user=root
autostart=true
```

Dockerfile for layer1



```
# Install both slurm and charliecloud
RUN set -e \
    && yum install -y $SLURM_BUILD_PACKAGES \
    && mkdir -p /root/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS} \
    && cd /root/rpmbuild/SOURCES/ \
    && wget "$SLURM_DOWNLOAD_URL" \
    && echo "$SLURM_DOWNLOAD_MD5" "$SLURM_VERSION".tar.bz2 | md5sum -c - \
    && rpmbuild -tb "$SLURM_VERSION".tar.bz2 \
    && cd /root/rpmbuild/RPMS/x86_64 \
    && yum -y install $SLURMD_PACKAGES \
    && cd \
    && rm -rf /root/rpmbuild \
    && useradd -r -U --uid=101 slurm \
    && cd /usr/local/src \
    && git clone --recursive https://github.com/hpc/charliecloud.git \
    && cd charliecloud \
    && make \
    && make install PREFIX=/usr/local \
    && ch-run --version \
    && printf "export CH_TEST_TARDIR=/var/tmp/tarballs\nexport CH_TEST_IMGDI
    && echo "clean_requirements_on_remove=1" >> /etc/yum.conf \
    && yum -y autoremove $SLURM_BUILD_PACKAGES epel-release \
    && yum clean all \
    && rm -rf /var/cache/yum /usr/local/src/charliecloud
```

- CentOS 7 based
- Download and install slurm and charliecloud
- Start supervisord
  - slurmd
  - rsyslogd
  - sshd
  - munge
- Copy in things like ssh keys and munge keys
- Currently about 362MB

docker run -dt -p 6818:6818 -p 3222:22 -h c1 --mount type=bind,source="/home/peltz",target=/home/peltz --security-opt='seccomp=unconfined' --name layer1 kraken/layer1-supervisord:1.0

Slurm for deployment of Containers

- **How about sbcast?**
  - Doesn’t perform at scale to distribute container image
  - May also depend on what network your slurm communication happens over, e.g. 1Gb Ethernet vs HSN
- **Private Registry?**
  - N-1 problem
- **See next slide for testing results**
  - https://bit.ly/2xxoGYa

LA-UR-18-27358

Evaluating Container Image Distribution Methods for HPC Using Charliecloud

Steven Anaya
New Mexico Tech

Michael Cutshaw
Dakota State University

Shane Goff
New Mexico Highlands University

Mentors: Reid Priedhorsky, Tim Randles

Overview

Containers have many benefits such as portability and freedom from a provided software stack. We studied several container image distribution methods for use with Charliecloud at scale.

Methods

- Experiment factors: distribution method, application, cluster, and node count
- Five repetitions for each job configuration
- Charliecloud as primary tool for building, compressing, and uncompressing images
- Modular scripting scheme to automate job submission and monitoring

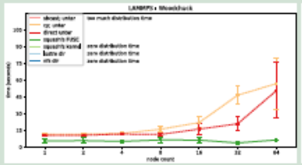
Woodchuck Specifications

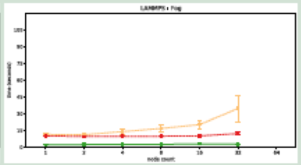
- 192 compute nodes
- 10Gbps Ethernet
- 16 cores, 2.4GHz, per node

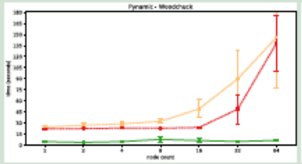
Fog Specifications

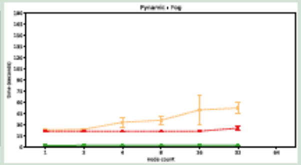
- 32 compute nodes
- 100Gbps Intel Omni-Path
- 36 cores, 2.1GHz, per node

Direct Untar, SquashFS Winners





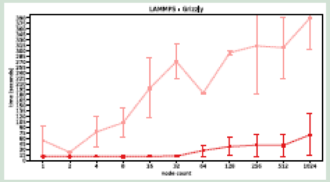




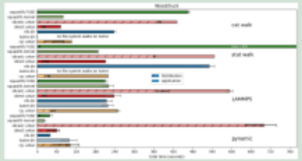
Poor Sbcast Performance at Scale

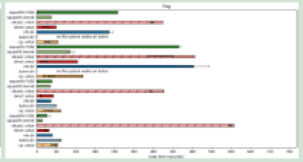
Grizzly Specifications:

- 1490 compute nodes
- 100Gbps Intel Omni-Path
- 36 cores, at 2.1GHz per node



Application Times





Conclusions

Question	Answer
Fastest network method?	Kernel-Mounted SquashFS
Fastest in-memory method?	Direct Untar
Network method impact on time?	Varies with application by I/O

Other Questions

Question	Answer
What is the runtime overhead of a FUSE-mounted SquashFS?	Minimal, unless I/O intensive application
Is Sbcast effective at a larger scale?	No. (but see limitations)

Limitations

- Cat walk and stat walk could not be run on Lustre due to concerns of overloading the metadata server
- Pynamic's benchmark time is produced by Pynamic itself instead of the 'time' command
- We tested only the Direct Untar and Sbcast methods at a large scale (128-1024 nodes)
- Distribution times can be affected by network congestion on Lustre and NFS
- LAMMPS does not scale as we expect for reasons we do not understand
- Sbcast may be running over the management network
- LAMMPS was unable to run at 128 nodes on Woodchuck

Future Work

Perform tests…

- with other scientific applications e.g. Vector Particle-In-Cell (VPIC)
- with larger node counts (512, 1024, 2048) across all applications
- on a Cray System
- with different file size profiles
- while monitoring various confounding factors e.g. Lustre, unpack speed, network
- on SquashFS methods in NFS
- that analyze other resource use (RAM, network bandwidth, etc.)

Generic slurm plugin for containers

- **Not necessary for some methods of container usage**
  - squashFS from PFS
- **Slurm running charliecloud**
  - Not difficult and no special plugins required

```
[peltz@c1 ~]$ cat slurm-1230.out
tarball: /home/peltz/mpihello.tar.gz
image: /var/tmp/mpihello replacing existing image /var/tmp/mpihello
/var/tmp/mpihello unpacked ok
container: mpirun (Open MPI) 2.1.5 0:
init ok c1, 1 ranks, userns 4026533176 0:
send/receive ok 0:
finalize ok
```

Future Work

- **What if slurmd was not the source of node state information?**
  - Slurmctld queries an API for Kraken or other system state managers
    - For example, Cray's system software wants to be the source of truth for system state
    - Historically the cray system and slurmctld do not agree
      - Must wait for timeouts
      - Or utilities such as slurmsmwd
- **Cut out docker as the container runtime for layer1**
  - Use unshare directly from u-root
- **Go get kraken**
  - https://github.com/hpc/kraken/

Thanks!