Hiding All the Details: Running Grid Jobs Inside Docker Containers on the OSG

Derek Weitzel, John Thiltges, Brian Bockelman
University of Nebraska - Lincoln
Hiding the Details

• Going back to the 1980’s, HTCondor strives to have the job runtime environment be run and defined by the submit host.

  • This is surprisingly difficult to do - look at the limitations (and hence popularity) of the standard universe.

• Why is this a good idea? Two examples:

  1. Enable OS updates independent of job environment
     • Sysadmins may want to run RHEL6
  2. Allow users to define their own execution environment
     • Special environments for applications
Large Hammer, Small Problem

• Great! Let’s use the VM universe!

• Virtual machines are hard to author - existing tools are poor and user-unfriendly.

• Virtual machine environments are large (in MB).

• Potentially significant overheads - especially in IO.

• Ouch!
Recent History

• Greg Thain gave a talk on isolating users.

• Used PID Namespaces & cgroups to isolate.

• chroots are used to provide a user environment distinct from the host.
Usage at Nebraska

• With sufficient effort, we used the built-in techniques to manage the transition from RHEL5 to RHEL6.

• Allowed us to transition to RHEL6 at our own rate, before all users were ready.

• Our chroot capability has slowly degraded over time.

• Why? These are hard to author — like VMs, no great tooling exists to manage ‘raw’ chroots.
A new approach: Docker

• Chroot, namespaces, and cgroups are all a part of Docker’s containerization solution.

• IMPORTANT: Docker provides very approachable way to compose and publish images.

• We don’t need to maintain a RHEL6 image, but only our local customizations on top.

• Decided to use HTCondor’s new Docker universe.

• Big picture: transform incoming grid jobs into Docker universe jobs.
Docker Performance

- Docker, in practice, is often faster than Virtual Machines.

Figure 2. Performance of (a) HEP-SPEC and (b) IPerf for the various machine configurations.

Note that neither of these plots start at zero so that the differences can be seen more clearly.

Then added as a volume to the container by using the "-v" flag on the command line. A single container was created which once again utilised all available resources.

The docker container used was an official Centos 6 container with the necessary grid software installed on top. An external volume called "/data" was used to save analysis output into, and the container was ran interactively. The command line options are listed below where "longr/centos6-chep" is the Centos 6 container with grid software installed

```
docker -D run -v /data:/data -v /afs:/afs -v /cvmfs/atlas.cern.ch:/cvmfs/atlas.cern.ch -i -t longr/centos6-chep /bin/bash
```

5. Analysis

We wanted to compare virtualisation and containerisation, and how both of these compare with baremetal performance. To do this we used standard tools designed to benchmark critical parts of a system, namely CPU, network and disk access. The tools used were:

- iperf-2.0.5
- bonnie++-1.96
- hep-spec06 (v1.1)

These tools were ran three times, on two different machines with each of the three setups.

5.1. HEP-SPEC

HEP-SPEC06[1] is the HEP-wide benchmark for measuring CPU performance and is based on the SPEC CPU2006 benchmark suite. The HEP-SPEC tests were run three times for each configuration, where more than one guest is running on the host, the scores are summed together to give a total HEPSPEC for the machine.

Figure 2(a) shows that the management overhead due to the hyper-visor is clear for the virtual machine where a significant drop in HEPSPEC score is seen. Containerisation shows a small drop of 1-2 HEPSPECS, but this is within the noise for Bare Metal.

Base Environment

• CentOS 7.2: This is our admin’s preferred OS.

• Docker v1.9.1: Default version of Docker for RHEL7.

• HTCondor 8.5.4: Contains a few useful bug fixes and new features over the current stable series.

• We’re focusing on enabling jobs from CMS and OSG: hence we’ll need to layer on a few quirky customizations.

• Not necessarily needed by others.
Default Container Setup

• Based off of CentOS 6
  • + OSG WN packages
  • + gcc, glibc-headers… for various system dependencies from CMS.
• https://hub.docker.com/r/unlhcc/osg-wn-el6/
FROM centos:centos6

RUN yum -y install http://repo.grid.iu.edu/osg/3.3/osg-3.3-el6-release-latest.rpm && \
    yum -y install epel-release && \
    yum -y install osg-wn-client osg-wn-client-glexec cvmfs && \
    yum -y install glibc-headers && \
    yum -y install gcc && \
    yum -y install redhat-1sb-core sssd-client && \
    yum clean all && \
    yum -y update

# Create condor user and group
RUN groupadd -r condor && \
    useradd -r -g condor -d /var/lib/condor -s /sbin/nologin condor

# Add lcmaps.db
COPY lcmaps.db /etc/lcmaps.db

That's it!
Docker Volumes

• There are a few important directories from the host that need to be available to the container - for example, the HDFS-based storage system.

• Docker refers to these as **volume mounts**. Currently, we bring in a total of 6 different directories.

• Most volumes are marked read only - no need for the jobs to write to these.

• Exception is SSSD: need to write to a Unix socket to lookup usernames.

  • Access control to HDFS is based on Unix username: hence, we need to leak this information to the container. May not be necessary for others!

```
DOCKER_VOLUME_DIR_CVMFS         = /cvmfs:/cvmfs:ro
DOCKER_VOLUME_DIR_ETC_CVMFS     = /etc/cvmfs:/etc/cvmfs:ro
DOCKER_VOLUME_DIR_HDFS          = /mnt/hadoop:/mnt/hadoop:ro
DOCKER_VOLUME_DIR_GRID_SECURITY = /etc/grid-security:/etc/grid-security:ro
DOCKER_VOLUME_DIR_SSSD          = /var/lib/sss/pipes/nss
DOCKER_VOLUME_DIR_NSSWITCH     = /etc/nsswitch.conf:/etc/nsswitch.conf:ro
DOCKER_MOUNT_VOLUMES = CVMFS, ETC_CVMFS, HDFS, GRID_SECURITY, SSSD, NSSWITCH
```
The HTCondor-CE uses the `condor_job_router` to provide sites with the ability to customize jobs.

1. GlideinWMS factories submit to the HTCondor-CE.
2. Job Router component transforms the CE job to use Docker universe.
   - Surprisingly, no new JobUniverse.
   - Sets DockerImage.
   - Changes the Cmd string.
Job Route Configuration

- Snippets from `condor_job_router` transform language

- Cmd needs to be prepended with `./`

```plaintext
copy_Cmd = "orig_Cmd"
eval_set_Cmd = ifThenElse(regexp("^/", orig_Cmd), orig_Cmd, strcat("./",orig_Cmd))
```

- Docker image needs to be set

```plaintext
copy_DockerImage = "orig_DockerImage"
eval_set_DockerImage = ifThenElse(isUndefined(orig_DockerImage), "unlhcc/osg-wn-el6", orig_DockerImage)
```
The Full Route

This is one of multiple possible routes jobs can match

```plaintext
[ \
  GridResource = "condor localhost localhost"; \
  eval_set_GridResource = strcat("condor " , $(FULL_HOSTNAME), $(FULL_HOSTNAME)); \n  TargetUniverse = 5; \n  MaxIdleJobs = 5; \n  name = "Local_Docker"; \n  set_Requirements = ( TARGET.Memory >= RequestMemory ) && ... (remainder truncated) \n  delete_PeriodicRemove = true; \n  /* Set Docker parameters */ \n  set_WantDocker = true; \n  /* If Cmd does not start with '/', prepend './' to include cwd */ \n  copy_Cmd = "orig_Cmd"; \n  eval_set_Cmd = ifThenElse(regexp("^/", orig_Cmd), orig_Cmd, strcat("./",orig_Cmd)); \n  /* Trying to directly test DockerImage failed, so we copy first */ \n  copy_DockerImage = "orig_DockerImage"; \n  eval_set_DockerImage = ifThenElse(isUndefined(orig_DockerImage), "unlhcc/osg-wn-el6", orig_DockerImage); \n  /* Do not match Andrea Sciaba's various DNs against this route (all DNs use the same email address) */ \n  requirements = target.x509UserProxyEmail != "User@example.com"; \n]
```

Note **MaxIdleJobs** prevents too many OSG jobs from using this route. Limit will be lifted as we become more comfortable with Docker.
View from the worker node
(Un)Trusted Images

• HTCondor treats all Docker images the same.

• We want to differentiate the images that come from the “good guys” (us) versus the “bad guys” (users).

• Still uncomfortable with the idea of allowing users to request arbitrary images.

• RHEL7.2 includes various sandboxing mechanisms: there’s no (publicly) known ways to break out, but the track record is relatively poor.
Status

• Running Production CMS and OSG jobs

• Currently ~10% of the Nebraska Tier 2 is Docker-enabled.

• Will be expanding to the entire cluster in the coming weeks: goal is to be done by the end-of-summer.

• Next step is to further explore how to (safely) expose this capability to OSG VOs and users.
Wishlist

- Things that would simplify our setup:

  - Pass resource accounting (CPU, memory usage) from Docker to HTCondor. Scheduled for 8.5.5.

  - Avoid prepending `. /` to the `Cmd`.

  - Make volume mounts conditional: we only want to expose HDFS and SSSD to CMS jobs.

- Ability to whitelist particular images - evaluated on worker node!

- Ability to mark jobs in “untrusted images” with the Linux “NO_NEW_PRIVS” flag (prevents setuid).
Questions?