Singularity and HTCondor

Brian Bockelman

European HTCondor Week 2017

What problems are we solving?

- **Isolation**: We launch arbitrary user code ("payload") that shouldn't have access to our wrapper scripts ("pilot"). Specifically:
  - *File isolation*: pilot determines what files the payloads can read and write.
  - *Process isolation*: payload can only interact with (see, signal, trace) its own processes.
    - These are ***simple*** kinds of isolation. Others (e.g., kernel isolation, network isolation) are less important!
- **glexec replacement**: Retire our particularly problematic current solution to isolation. Niche and expensive.
- **Homogeneous / portable OS environments**: Make user OS environment as minimal and identical as possible!

Old Adventures in Isolation and Traceability: **MUPJ and `glexec`**

- The WLCG experiments have *heavily* used the Multi-User Pilot Job (**MUPJ**) model:
    - A generic "pilot job," *owned by the experiment*, is submitted to the site batch system.
    - This pilot job launches one or more scientific *payload* jobs. This is where the "actual computing" is done!
        - Each payload job belongs to an individual user.
- We need **isolation** so user payloads cannot interact with each other or the pilot. (No credential stealing!!).
- We need **traceability** so sites can identify who uses a computing resource at any given time.
- Traditionally, isolation and traceability is provided by the batch system: launches each user's jobs as a separate Unix user.

Introducing: Singularity

- Singularity is a container solution tailored for the HPC use case.
    - It allows for a portable of OS runtime environments.
    - It can provide isolation needed by our users.
- Simple isolation: Singularity does not do resource management (i.e., limiting memory use), leaving that to the batch system.
- Operations: No daemons, no UID switching; **no edits to config file needed**. “Install RPM and done.”
- Goal: User has no additional privileges by being inside container. E.g., disables all `setuid` binaries inside the container.

http://singularity.lbl.gov

Yet Another Container Syndrome

- "But HTCondor already supports Docker! Why do we need Yet Another Container?"

  - Singularity support works even if HTCondor runs as non-root (i.e., glideinWMS).

  - Singularity does not require any additional system services / daemons. Tradeoff: requires `setuid`.

  - Works inside Docker — important for sites that already invest heavily in Docker (like mine!).

IMPORTANT:
Singularity provides a path to non-`setuid` isolation

And there was great rejoicing!

Why Docker?

- There remain a good number of reasons to use Docker universe:
    - Docker implements additional resource management and isolation mechanisms.
    - Built-in image distribution mechanism.
    - Wider acceptance / larger ecosystem / more mature.
- To each their own: pick the correct technology to fit your site.
- **Nebraska uses both**: Docker for site batch system, Singularity for pilots inside the batch system.

View From the Worker Node

Site Batch System

```
/usr/sbin/condor_master -f
 \_ condor_procd -A /var/run/condor/procd_pipe -L /var/log/condor/ProcdLog -R 1000000 -S 60 -C 554
 \_ condor_shared_port -f
 \_ condor_startd -f
     \_ condor_starter -f -a slot1_1 red-gw2.unl.edu
         \_ python /usr/local/libexec/condor-docker run --cpu-shares=560 --memory=250000m --hostname cmspr
             \_ /usr/bin/docker-current run --cpu-shares=560 --memory=250000m --name HTCJob406040_0_slot1_
```

Docker

```
/usr/bin/dockerd-current --add-runtime docker-runc=/usr/libexec/docker/docker-runc-current --default-runti
 \_ /usr/bin/docker-containerd-current -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --sh
     \_ /usr/bin/docker-containerd-shim-current 737770d03e6f22108ac9acb89def79655fffbafbfc4fe7082f43a3bb40
```

Pilot

```
\_ /bin/bash ./condor_exec.exe -v std -name v3_2 -entry CMS_T2_US_Nebraska_Red_gw2_whole -clientn
    \_ /bin/bash /var/lib/condor/execute/dir_729792/glide_McAkr7/main/condor_startup.sh glidein_c
        \_ /var/lib/condor/execute/dir_729792/glide_McAkr7/main/condor/sbin/condor_master -f -pid
            \_ condor_procd -A /var/lib/condor/execute/dir_729792/glide_McAkr7/log/procd_address
            \_ condor_startd -f
                \_ condor_starter -f -a slot1_1 vocms0311.cern.ch
```

Singularity

```
|   \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/condor
|       \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/co
```

Payload

```
|           \_ /bin/bash /srv/condor_exec.exe pdmvserv_task_EGM-PhaseISpring17wmL
|               \_ python2 Startup.py
|                   \_ /bin/bash /srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh  sl
|                       \_ cmsRun -j FrameworkJobReport.xml PSet.py
\_ condor_starter -f -a slot1_8 vocms0311.cern.ch
```

Singularity

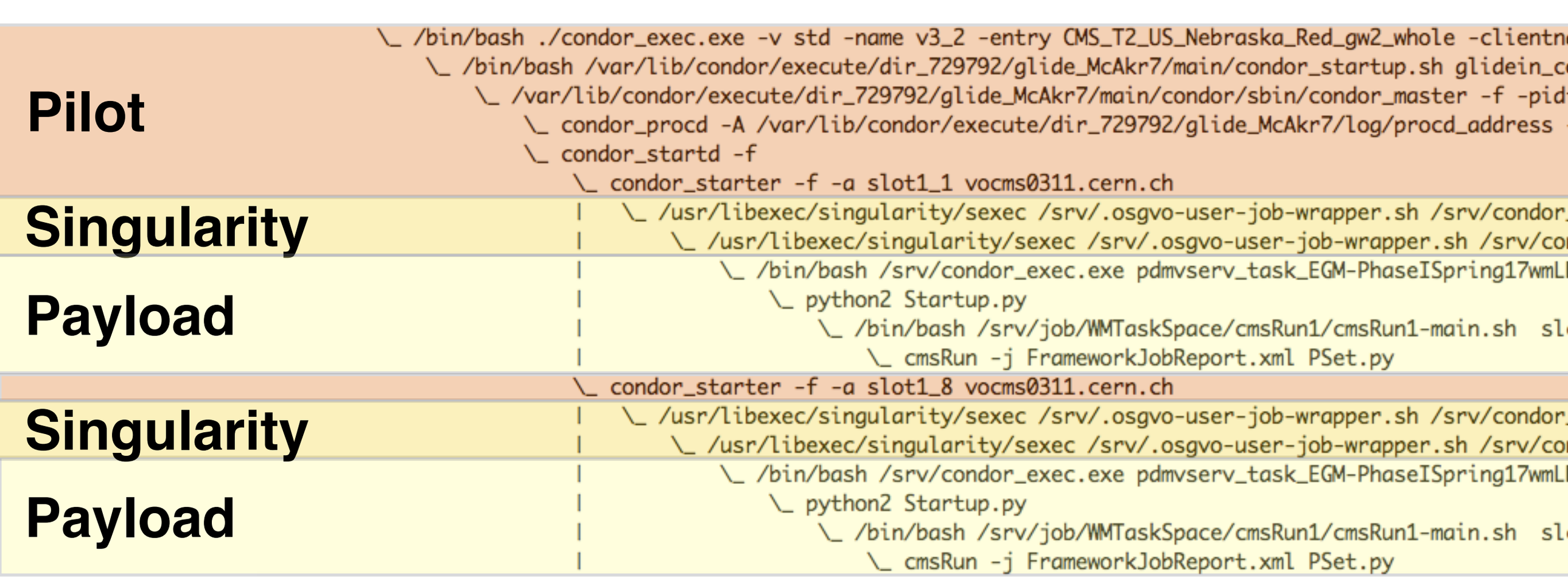
```
|   \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/condor
|       \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/co
```

Payload

```
|           \_ /bin/bash /srv/condor_exec.exe pdmvserv_task_EGM-PhaseISpring17wmL
|               \_ python2 Startup.py
|                   \_ /bin/bash /srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh  sl
|                       \_ cmsRun -j FrameworkJobReport.xml PSet.py
```

View From the Pilot

No visibility into the host OS!



View From the Payload

User jobs are isolated from each other,
but it's still a familiar OS environment

Payload

```
|          \_ /bin/bash /srv/condor_exec.exe pdmvserv_task_EGM-PhaseISpring17wmL
|              \_ python2 Startup.py
|                  \_ /bin/bash /srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh  sl
|                      \_ cmsRun -j FrameworkJobReport.xml PSet.py
```

OS Portability

- Containers provide OS portability - the ability to define your job's OS environment and have it identical everywhere.
  - Solves a very tough transition problem for CMS - we need something like containers to move our sites forward!

Tomorrow's CHTC Users

OS	%age of CHTC users
Require EL 7	Was 5%, going ↑
Either EL 6 or 7	Was 90%, going ?
Require EL 6	Was 5%, going ↓

CMS is here; old releases CANNOT use EL7!

On Image Distribution…

- Docker images are a list of *layers*, each a tarball.
    - DockerHub limit is 10GB. In practice, ranges of 500MB (minimal image, caring users) to 4GB (large scientific organization) are common.
- Singularity has three image formats:
    - Native format: raw filesystem image, loopback mounted. Large - 10GB.
    - SquashFS-based compressed image. Slightly smaller than Docker (stays compressed on disk).
    - Simple chroot directory.
- **How does one deliver these to thousands of worker nodes?**

Image Distribution

- Observed several strategies in the wild:
    - Drop raw image onto shared file system.
    - Copy image files to worker node.
    - Synchronize chroot directory to CVMFS.
- Tradeoffs to consider:
    - **How much freedom will you give to users**? Can they specify their own image? Are they restricted to a whitelist?
    - Use of cache (what is the working set size?). If user-specifies images, the working set size might be fairly unpredictable.
    - Scalability of distribution mechanism.
    - Does the full image get downloaded to the worker node?

Singularity around town

- Some of the heaviest users of Singularity are on the OSG:
  - Currently, CMS launches about 1.2M containers / week on OSG.
  - OSG VO has launched 17M containers since mid-February.
    - To see how OSG exposes this functionality to users, see: https://go.unl.edu/osg-singularity
- At several large NSF supercomputing sites: SDSC, TACC.
- Popular across a range of HPC sites (med centers, university computing centers, big labs), which was Singularity's original niche.

Integration with HTCondor

- Singularity availability and version advertised in ClassAd.
- HTCondor will launch jobs inside Singularity based on a few `condor_startd` configuration variables:
    - `SINGULARITY_JOB`: If true, then launch job inside Singularity.
    - `SINGULARITY_IMAGE_EXPR`: ClassAd expression; evaluated value is the path used for the Singularity image.
    - `SINGULARITY_TARGET_DIR`: Location inside Singularity container where HTCondor working directory is mapped.
- See https://htcondor-wiki.cs.wisc.edu/index.cgi/tktview?tn=5828 for details. **Examples follow**.
- The details are a bit hidden under the cover; **still experimenting with the best user interface**.
    - While base functionality is in 8.6.x, more UI work will occur in HTCondor 8.7.x.

Example:
All Jobs Into the Container

- All config is controlled by the `condor_startd`.
- Example config:

```
# Only set if Singularity is not in $PATH.
#SINGULARITY = /opt/singularity/bin/singularity
# Forces _all_ jobs to run inside singularity.
SINGULARITY_JOB = true
# Forces all jobs to use the CernVM-based image.
SINGULARITY_IMAGE_EXPR = "/cvmfs/cernvm-prod.cern.ch/cvm3"
# Maps $_CONDOR_SCRATCH_DIR on the host to /srv inside the image.
SINGULARITY_TARGET_DIR = /srv
# Writable scratch directories inside the image.  Auto-deleted afte
the job exits.
MOUNT_UNDER_SCRATCH = /tmp, /var/tmp
```

Example:
Only on User Request

- However, startd config variable can reference the user job using `TARGET`.

```
SINGULARITY_JOB = !isUndefined(TARGET.SingularityImage)
SINGULARITY_IMAGE_EXPR = TARGET.SingularityImage
```

- In this configuration, Singularity is only used if the user specifies an image in their submit file:

```
+SingularityImage = "/cvmfs/cernvm-prod.cern.ch/cvm3"
```

Example:
Image based on OS name

- Startd config snippet:

```
SINGULARITY_JOB = \
    (TARGET.DESIRED_OS isnt MY.OpSysAndVer) && \
        ((TARGET.DESIRED_OS is "CentOS6") || \
            (TARGET.DESIRED_OS is "CentOS7"))
SINGULARITY_IMAGE_EXPR = \
    (TARGET.DESIRED_OS is "CentOS6") ? \
      "/cvmfs/singularity.opensciencegrid.org/library/centos:centos6"
      "/cvmfs/singularity.opensciencegrid.org/library/centos:centos7"
```

- User adds this to the job:

```
+DESIRED_OS="CentOS6"
```

Conclusions

- Singularity is another container technology in our toolbox.
  - Different set of tradeoffs than Docker:
    - I.e., `setuid` binary but no system service.
    - Currently, most popular where HTCondor runs as non-root.
    - Interface will be a work-in-progress during 2017. **Currently, completely managed/implemented by sysadmin**.
- CMS and OSG utilize Singularity as a mechanism for *isolation* and *OS portability*.