New Adventures in Isolation and Traceability

Brian Bockelman
Spring HEPiX 2017
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.
glexec

- **Oversimplification**: glexec is a setuid binary, invoked by the pilot, which:
  
  - Takes a command and a set of X509 credentials,
  
  - Logs the credential’s identity through a secure channel (logfile on disk),
  
  - Maps the X509 credentials to a Unix identity using the LCMAPS framework, then
  
  - Executes the command under the mapped Unix identity.

- When used by the pilot, glexec provides isolation and traceability for MUPJ.
glexec is not popular :(

• glexec is a neat solution to the problems raised by the MUPJ model. But glexec is not popular :(

• **Why?** Personal opinions:

  • Our ecosystem is not built with user switching in mind:

    • Batch systems are confused when a job writes files or launch processes owned by different users.

    • Pilots must carefully manage file and process ownership when using glexec. Significant work to integrate into the framework.

  • glexec requires configuration. glexec can rely on external services.

  • Sites view it as a chore, not as a feature.
Let’s try something different
Isolation: 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 CMS.
- 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
IMPORTANT: Singularity provides a path to non-setuid isolation

And there was great rejoicing!
Singularity itself will be deployable in CVMFS
Who is in a container?

- Three options when using containers:
  - A: Batch system starts pilot inside a container.
  - B: Pilot starts each payload inside its own container.
  - C: Combine A and B.

- Option A does not meet our isolation needs.

- Nebraska uses Option C.
View From the Worker Node

Site Batch System

Docker

Site Batch System

Pilot

Site Batch System

Singularity

Payload

Singularity

Payload
View From the Pilot

No visibility into the host OS!

<table>
<thead>
<tr>
<th>Pilot</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/bin/bash ./condor_exec.exe -v std -name v3_2 -entry CMS_T2_US_Nebraska_Red_gw2_whole -clientname By scientists</code></td>
<td></td>
</tr>
<tr>
<td><code>/bin/bash /var/lib/condor/execute/dir_729792/glide_McAkr7/main/condor_startup.sh glidein_cms3</code></td>
<td></td>
</tr>
<tr>
<td><code>/var/lib/condor/execute/dir_729792/glide_McAkr7/main/sbin/condor_master -f -pidfile /var/lib/condor/execute/dir_729792/glide_McAkr7/log/procd_address</code></td>
<td></td>
</tr>
<tr>
<td><code>/condor_startd -f</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Singularity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/usr/libexec/singularity/seexec /srv/osgvo-user-job-wraper.sh /srv/condor/execute/dir_729792/glide_McAkr7/main/condor_startup.sh glidein_cms3</code></td>
<td></td>
</tr>
<tr>
<td><code>/bin/bash /var/lib/condor/execute/dir_729792/glide_McAkr7/main/sbin/condor_master -f</code></td>
<td></td>
</tr>
<tr>
<td><code>/condor_startd -f</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/bin/bash /srv/condor_exec.exe pdmserv_task_EGM-PhaseISpring17wMLTTT TTTask_A1 4</code></td>
<td></td>
</tr>
<tr>
<td><code>/python2 Startup.py</code></td>
<td></td>
</tr>
<tr>
<td><code>/bin/bash /srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh slurm</code></td>
<td></td>
</tr>
<tr>
<td><code>/cmsRun -j FrameworkJobReport.xml PSet.py</code></td>
<td></td>
</tr>
<tr>
<td><code>/condor_starter -f -a slot1_8 vocms0311.cern.ch</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Singularity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/usr/libexec/singularity/seexec /srv/osgvo-user-job-wraper.sh /srv/condor/execute/dir_729792/glide_McAkr7/main/condor_startup.sh glidein_cms3</code></td>
<td></td>
</tr>
<tr>
<td><code>/bin/bash /var/lib/condor/execute/dir_729792/glide_McAkr7/main/sbin/condor_master -f</code></td>
<td></td>
</tr>
<tr>
<td><code>/condor_startd -f</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/bin/bash /srv/condor_exec.exe pdmserv_task_EGM-PhaseISpring17wMLTTT TTTask_A1 4</code></td>
<td></td>
</tr>
<tr>
<td><code>/python2 Startup.py</code></td>
<td></td>
</tr>
<tr>
<td><code>/bin/bash /srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh slurm</code></td>
<td></td>
</tr>
<tr>
<td><code>/cmsRun -j FrameworkJobReport.xml PSet.py</code></td>
<td></td>
</tr>
<tr>
<td><code>/condor_starter -f -a slot1_8 vocms0311.cern.ch</code></td>
<td></td>
</tr>
</tbody>
</table>
View From the Payload

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

• How do we deliver an OS environment to CMS pilots?

• Singularity has its own image creation utilities or can convert Docker images.
  
  • Given the immense ecosystem of Docker images and tooling, we have chosen the latter approach.

• Traditionally, Singularity images are a single file. These get large: simple LIGO image might be about 4GB. Singularity can also just read from a directory.

• What tool would CMS use to distribute a directory of software across the global infrastructure? CVMFS

  • Every 15 minutes, we sync updates from DockerHub to CVMFS for a whitelisted set of images.

• CVMFS also provides per-file caching and file-level de-duplication. To launch python only requires downloading 3MB of data from a 3GB image. CVMFS also provides efficient cache management.

Take a look!

ls /cvmfs/singularity.opensciencegrid.org
Singularity provides Isolation

But it knows nothing of X509, pilots, or payloads. What about traceability?
• glexec keeps all traceability data on site. If you want to know who used worker node X at time Y, simply view your logs!

• **Observation:** glexec is a communication channel between the VO and site.

• By setting environment variables to point at an X509 proxy when invoking glexec, the VO is telling us the given user is associated with the executable.

• Since glexec is not widely used by VOs, in reality most sites will need to ask the VO to trace resource usage. **Not CMS at FNAL!**

• **FNAL request:** *Can we keep site-level traceability when using Singularity?*
Traceability with HTCondor-CE

• The HTCondor-CE provides a mechanism for running pilots to advertise current status to the CE.

  • GlideinWMS automatically sends pilot ads to the CE. Can see these with `condor_ce_status`.

• **IDEA**: Can we use this communication channel for tracebility?

  • **Yes**! CMS already sends payload user information to the CE.

• Submitted a patch to HTCondor to allow us to log the payload.

  • Waiting for next HTCondor release; subsequent HTCondor-CE release will support traceability. Hopefully, FNAL can then switch to Singularity.
So Where Are We?

• Singularity deployments are starting to occur at sites. RPM is installed at most US Tier-2 sites.

• OSG pilots have used Singularity since February; typically 40-70% of the opportunistic pool has Singularity enabled.

• CMS pilots have used Singularity since mid-March for volunteer sites; on by default in production since last week!

• Result? About 10 million containers launched since February.
Help Wanted!

- Deploy Singularity at your site!
- Upgrade your VO’s pilot to invoke Singularity.
- Join the WLCG Isolation and Traceability Working Group!
Help Wanted: Authz Overhaul! (?)

Move storage authentication from GSI/proxy to OAuth2-based.

Move authorization model from POSIX-like to VO-centric ("the ALICE model").
Questions?