Singularity and OSG

Brian Bockelman
WLCG Workshop, June 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 we already support Docker! Why do we need Yet Another Container?”
    - Singularity support works even if invoker 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

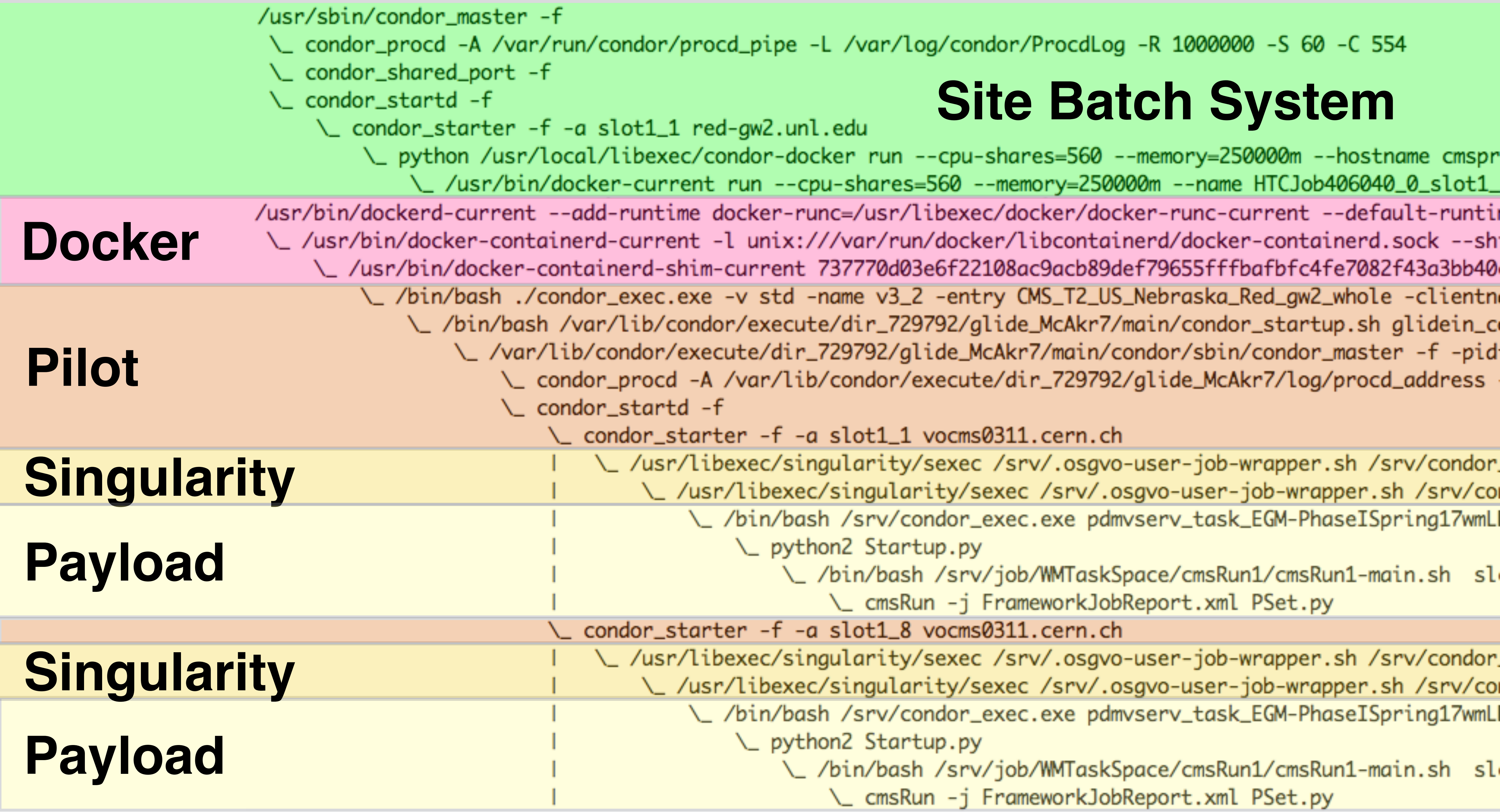
And there was great rejoicing!

🎉

Why Docker?

- There remain a good number of reasons to use Docker:
    - 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.
    - For example, both technologies are built-in to HTCondor.
- **Nebraska uses both**: Docker for site batch system, Singularity for pilots inside the batch system.

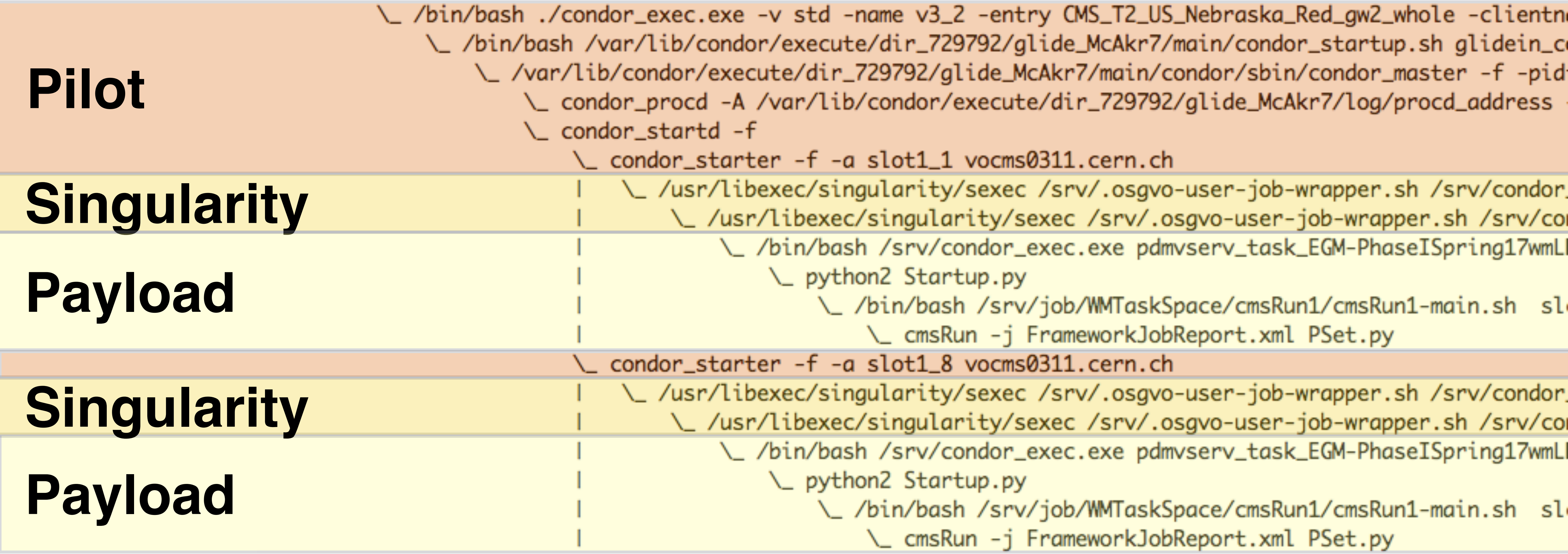
View From the Worker Node



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

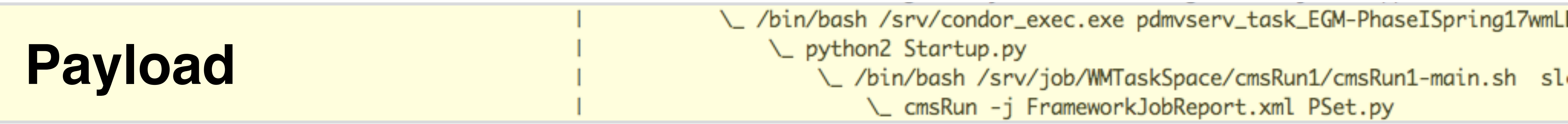
No visibility into the host OS!



```
\_ /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
            |   \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/condor
            |      \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/co
            |         \_ /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
            |   \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/condor
            |      \_ /usr/libexec/singularity/sexec /srv/.osgvo-user-job-wrapper.sh /srv/co
            |         \_ /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 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-PhaseISpring17wmLL
|              \_ python2 Startup.py
|                  \_ /bin/bash /srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh  sl
|                      \_ cmsRun -j FrameworkJobReport.xml PSet.py
```

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?**
  - On OSG, we do this by distributing the chroot directory via CVMFS.

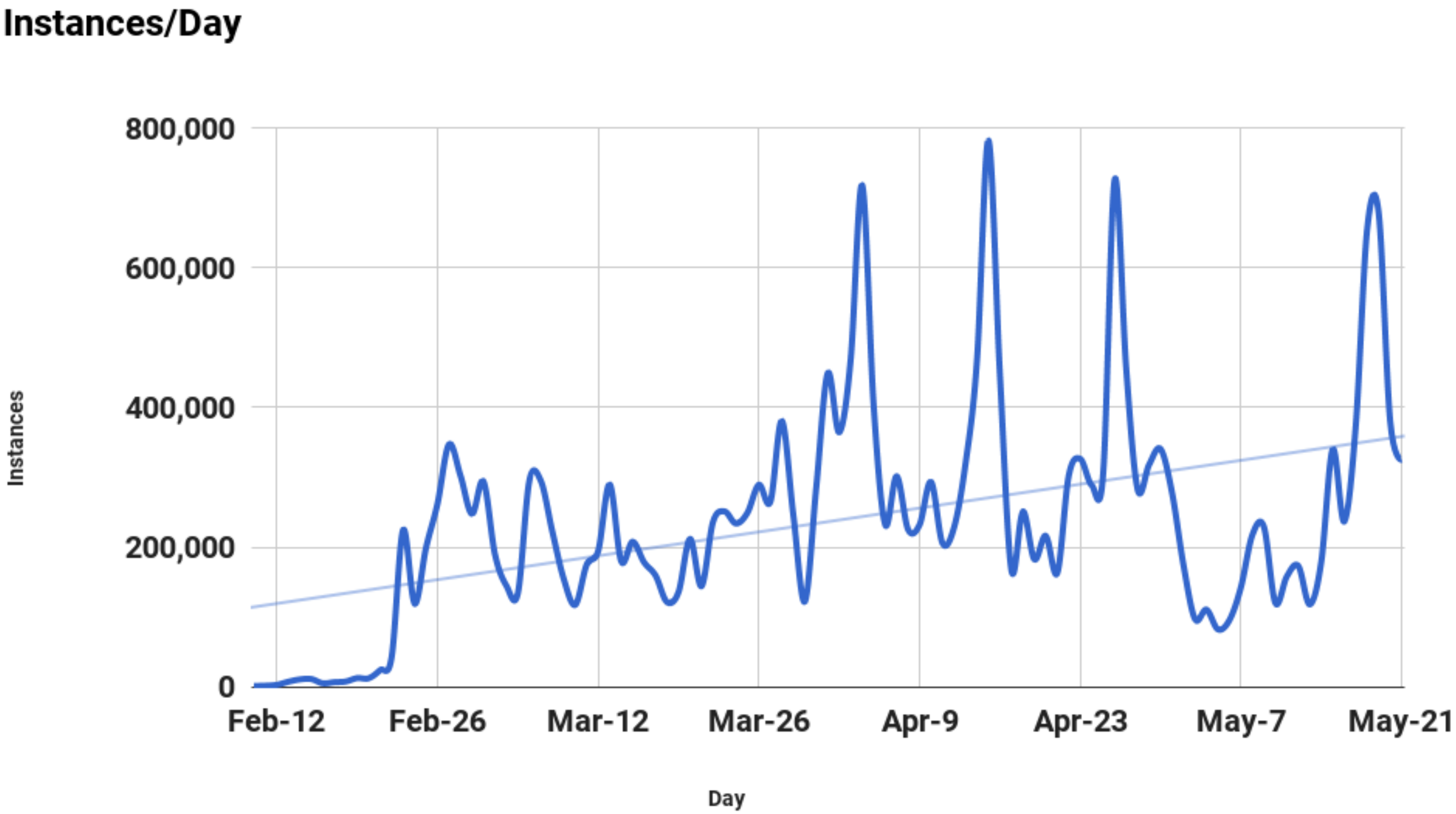
Integration with OSG Users

- OSG VOs can request a certain Docker image to be replicated to CVMFS by sending a pull request against the official repo:
  - https://github.com/opensciencegrid/cvmfs-singularity-sync
  - OSG Staff will verify this request originated from an OSG VO (basically, someone needs to sign the AUP).
  - After initial approval, subsequent image updates are auto-sync'd to CVMFS.
- OSG VO will automatically select an OS image if no container is selected; otherwise, user can specify that containers are required and which CVMFS image to use.
  - To see how OSG exposes this functionality to users, see: https://go.unl.edu/osg-singularity

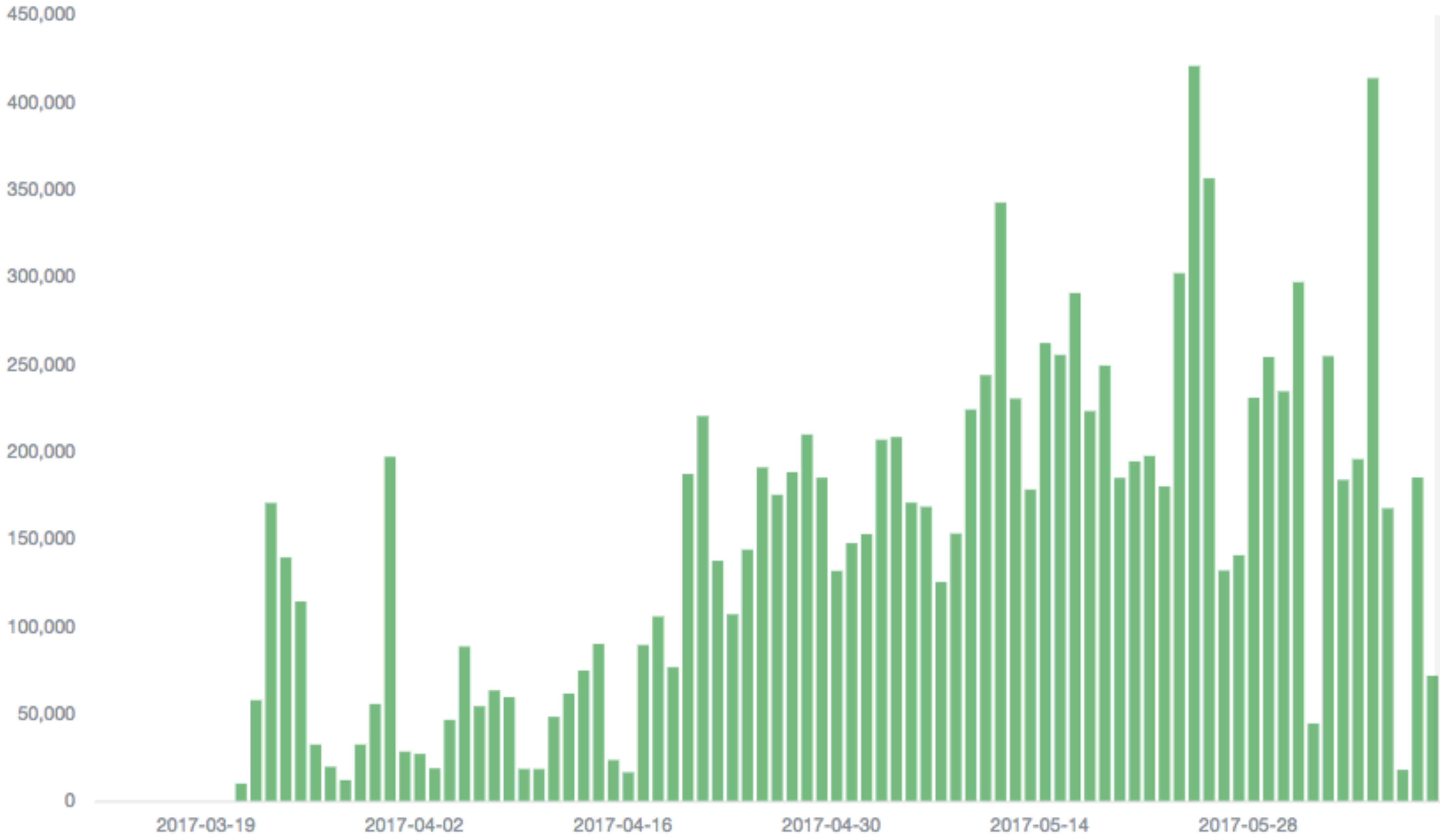
Singularity around town

- Some of the heaviest users of Singularity are on the OSG:
  - Currently, CMS launches about 1M containers / week.
  - OSG VO has launched 30M containers since mid-February.
- 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.

OSG Ramp-up


Instances/Day
800,000
600,000
400,000
200,000
0
Instances
Feb-12
Feb-26
Mar-12
Mar-26
Apr-9
Apr-23
May-7
May-21
Day

CMS Ramp-up


450,000
400,000
350,000
300,000
250,000
200,000
150,000
100,000
50,000
0
2017-03-19
2017-04-02
2017-04-16
2017-04-30
2017-05-14
2017-05-28

Whodunnit?

- 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.
- Current HTCondor release (8.6.3) allows us to log the payload.
  - Subsequent HTCondor-CE release (2.2.1?) 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 50-80% of the opportunistic pool has Singularity enabled.
  - CMS pilots have used Singularity since mid-March for volunteer sites; on by default in production since mid-April!
- OSG strongly recommends Singularity 2.2.1 from EPEL.

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*.