
Singularity and High Throughput Computing

Singularity User Group meeting 2019

Dave Dykstra <dwd@fnal.gov>
Fermi National Accelerator Laboratory

Many slides based on material from Mats Rynge, University of Chicago

High Throughput Computing

High Throughput Computing (HTC)

Sustained computing over long periods of time. Usually single-thread code, or low number of cores threaded within each node. “Embarrassingly parallel” applications. Uses commodity, relatively inexpensive hardware (much like most commercial cloud resources).

vs. High Performance Computing (HPC)

Great performance over relatively short periods of time. Large scale MPI. Each project typically gets small percentage of total time. Specialized low-latency networking hardware, high speed filesystem.

***Distributed* HTC**

No shared file system.

Users ship input files and some software their jobs.

Opportunistic Use

No allocations. Resource owners have priority; other jobs compete for available resources. Applications (esp. with long run times) can be *preempted* (or killed) by resource owner’s jobs. Applications should be relatively short or support being restarted.

Open Science Grid



A **framework** for large scale Distributed HTC resource sharing addressing the technology, policy, and social requirements of sharing computing resources in the U.S.

OSG is a **consortium** of software, service and resource providers and researchers, from universities, national laboratories and computing centers across the U.S., who together build and operate the OSG project. The project is funded by the NSF, and provides staff for managing various aspects of the OSG.

Integrates computing and storage resources from over **100 sites** in the U.S.



Worldwide LHC Computing Grid

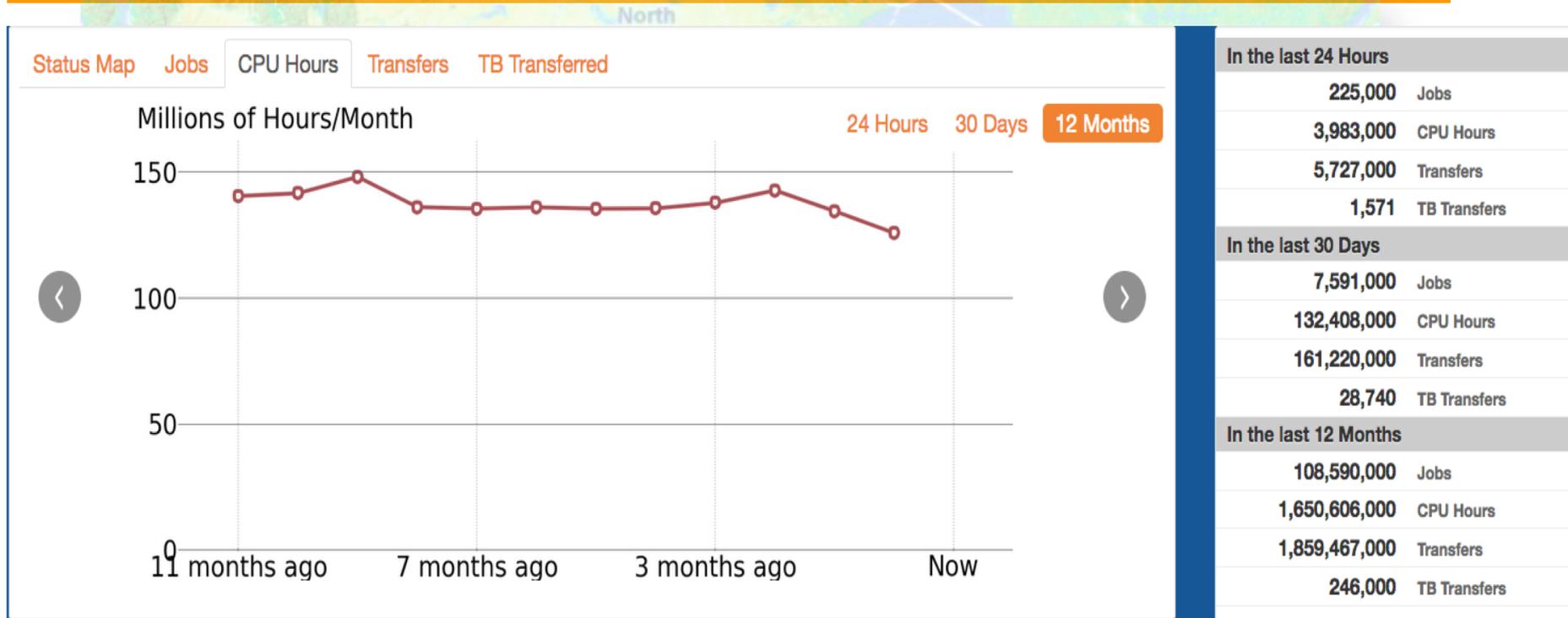
The WLCG is the HTC computing grid for the Large Hadron Collider at CERN. In addition to the OSG, it also includes a roughly similar amount of resources in EGI, the European Grid Initiative.

The WLCG is a collaboration of more than 170 sites in 42 countries. The mission of the WLCG project is to provide global computing resources to store, distribute and analyze the ~50-70 Petabytes of data expected every year of operations from the LHC. (Data expected to grow significantly.)

The OSG was initially created by the LHC High Energy Physics (HEP) experiments, and funding for them paid for the majority of the resources. They are still the largest users; CMS uses about half the core hours.

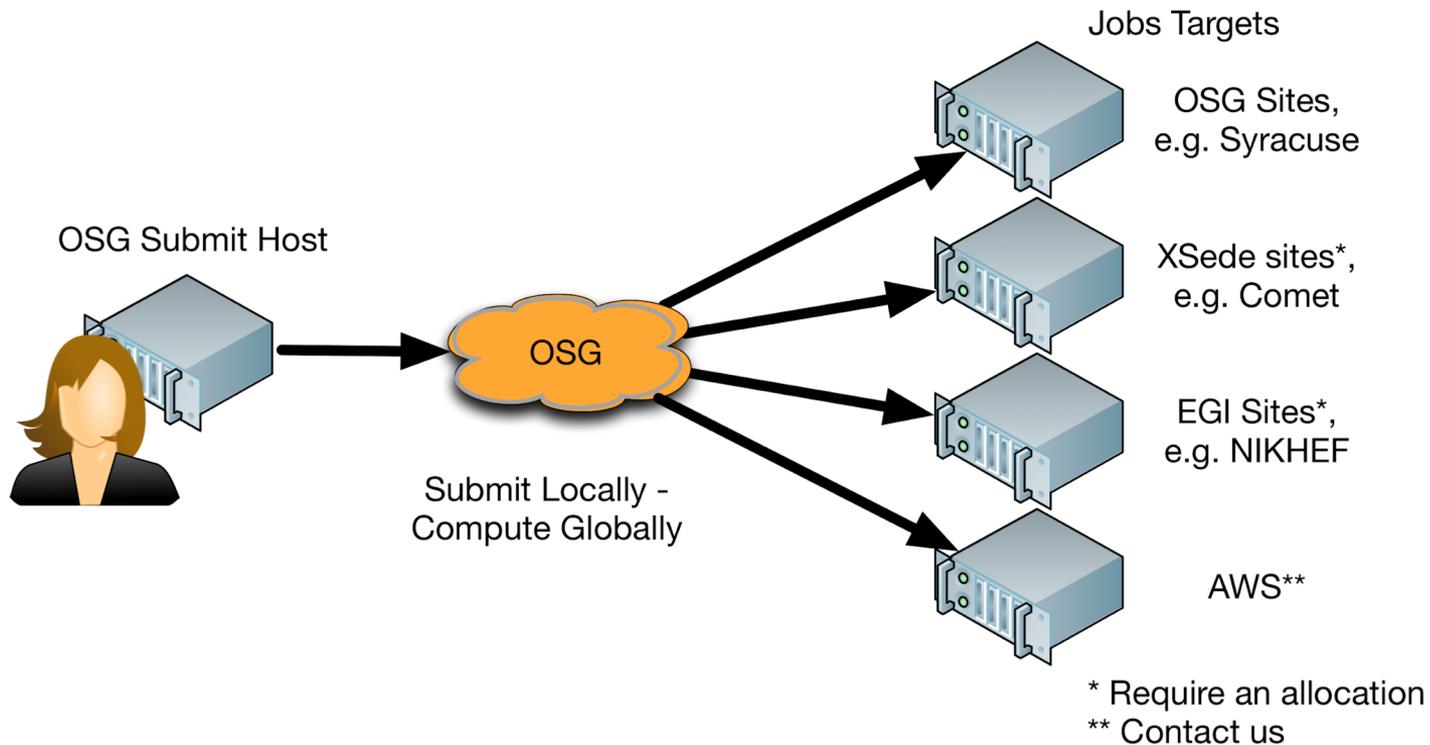


Open Science Grid scale



~ 4.5 million CPU hours delivered per day (~190K avg cores)

“Submit Locally, Run Globally”



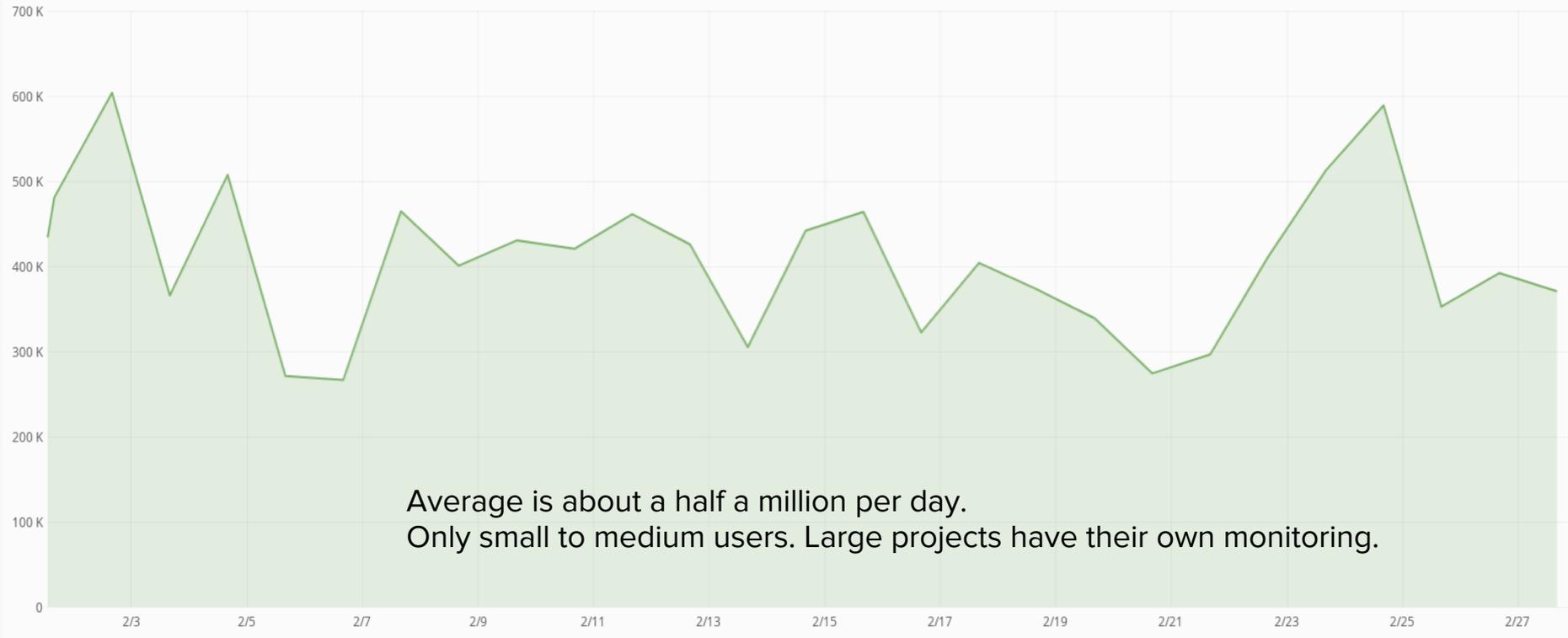
Motivations to use Singularity

- **Invoked by unprivileged user** - HTC uses unprivileged “pilot jobs” to bootstrap, and those each read from a per-Virtual Organization (VO) queue of jobs from different users.
- **Process isolation** - Isolates the job environment (`--ipc --pid`) so that a job can not affect other jobs.
- **File isolation** - Isolates the job file system (`--contain`), so that a job can not peek at other jobs’ data.
- **Consistent environment (default images)** - If a user does not specify a specific image, a default one is used by the job. The image contains a decent base line of software, and because the same image is used across all the sites, the user sees a more consistent environment than if the job landed in the environments provided by the individual sites.
- **Custom software environment (user defined images)** - Users can create and use their custom images, which is useful when having very specific software requirements or software stacks which can be tricky to bring with a job. For example: Python or R modules with dependencies, TensorFlow
- **Enables special environment such as GPUs** - Special software environments to go hand in hand with the special hardware.

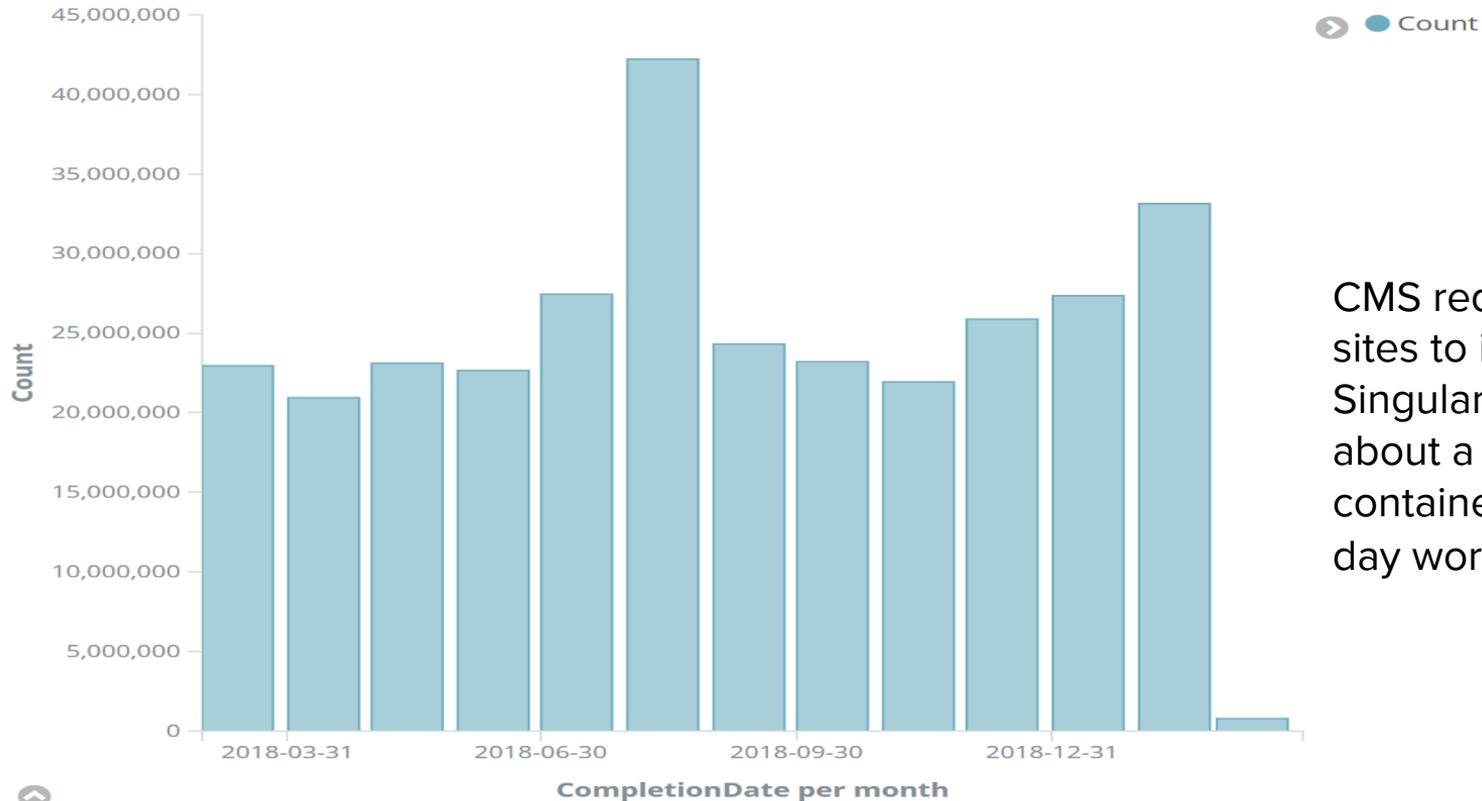
Container Lifecycle (Hint: ephemeral)

- Every job is encapsulated in a separate container instance
- Container instance dies when the job finishes
- Lots of container image reuse, as workloads generally use one or a small number of images for a large number of jobs
- Application software is mostly outside of the container

OSG Singularity instances per day



CMS Singularity instances per month

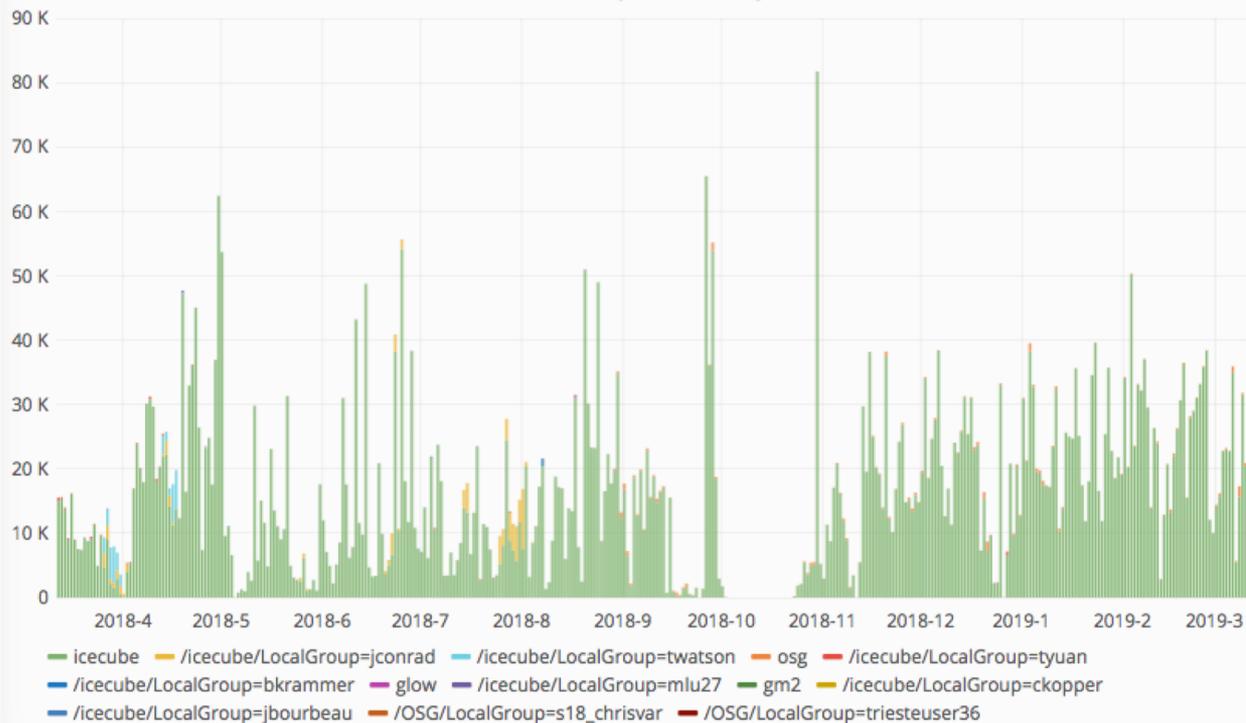


CMS requires all its sites to install Singularity. They do about a million container starts per day worldwide.



GPU Usage in OSG powered by Singularity

WallHoursSpentOnJobsByVO



Wall HoursByVO

	total
icecube	5.72 Mil
/icecube/LocalGroup=jconrad	71.0 K
/icecube/LocalGroup=twatson	37.8 K
osg	35.5 K
/icecube/LocalGroup=tyuan	7.11 K
/icecube/LocalGroup=bkrammer	1.613 K
glow	783
/icecube/LocalGroup=mlu27	486
gm2	3
/icecube/LocalGroup=ckopper	0.10
/icecube/LocalGroup=jbourbeau	0.10
/OSG/LocalGroup=s18_chrisvar	0.02
/OSG/LocalGroup=triesteuser36	0

Usage over the last year

Average about 1K cores

1.5 million containers x 5 GB (average size of container for estimate, CMS actually bigger) =

7.5 PB / day

We need an efficient way to distribute containers!

CVMFS - CERN Virtual Machine File System

“The CernVM File System provides a scalable, reliable and low-maintenance software distribution service. It was developed to assist High Energy Physics (HEP) collaborations to deploy software on the worldwide-distributed computing infrastructure used to run data processing applications. CernVM-FS is implemented as a **POSIX read-only file system** in user space (a FUSE module). Files and directories are hosted on standard web servers and mounted in the universal namespace `/cvmfs`.”

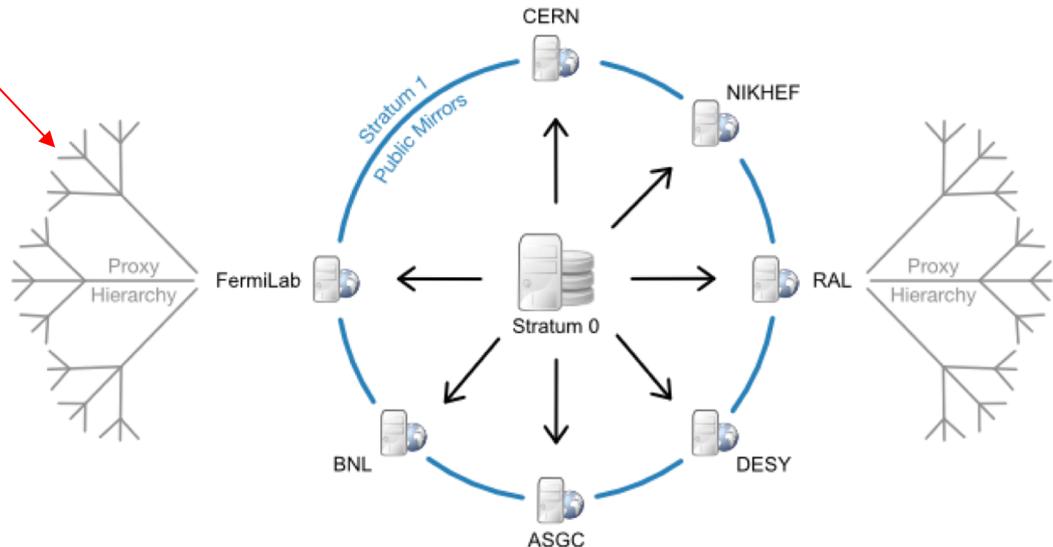
Your job is here!

Pre-existed use of containers

Used for software and some data

Heavily cached, read-only

Available across OSG, EGI, some XSEDE resources



CVMFS features

- **Files appear immediately present, but are only downloaded on demand**
- **Metadata operations are done on the worker node, on “catalogs” downloaded in chunks of about 100K files or less**
- **Files stored and transferred named by secure hash of contents of files, deduplicated and compressed**
- **Cryptographically verified with a digital signature on one small file**
- **Larger files broken up into ~8 MB chunks (by default) to smooth out load on servers**
- **Served from small number of worldwide “stratum 1” servers, cached in http proxy caching servers (squid) at each site, and cached on each worker node** - caches greatly reduce latency and bandwidth
- **Optional “external data server” mode** - for data (metadata uses standard path) of partially reused data files, using separate caching servers at geographically distributed high-capacity network sites

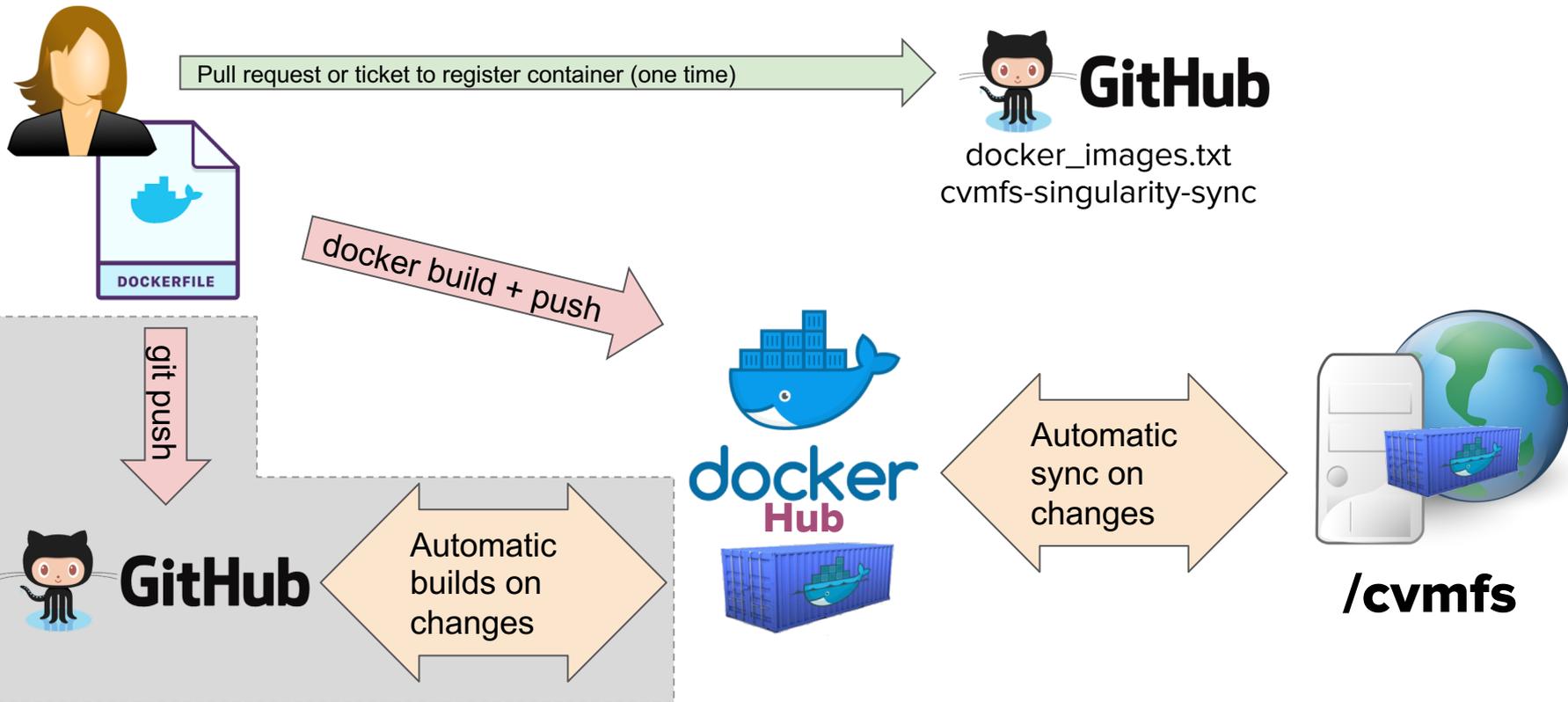
Example CVMFS Repositories

/cvmfs/
ams.cern.ch
atlas.cern.ch
cms.cern.ch
connect.opensciencegrid.org
icecube.opensciencegrid.org
fermilab.opensciencegrid.org
ligo.opensciencegrid.org
ligo-containers.opensciencegrid.org <- large project with their own containers
nova.opensciencegrid.org
oasis.opensciencegrid.org
singularity.opensciencegrid.org <- general containers (next few slides)
stash.osgstorage.org <- ~1PB of user published data

cvmfs-singularity-sync

- Containers are **defined using Docker**
 - Public Docker Hub
- ... and **executed with Singularity**
 - No direct access to the Singularity command line - that is controlled by the infrastructure
- <https://github.com/opensciencegrid/cvmfs-singularity-sync>

User-defined Container Workflow



Extracted Images

We store container images on CVMFS in extracted form (Singularity calls this “sandbox” mode). That is, we take the Docker image layers or the Singularity img/simg/sif files and export them onto CVMFS. For example, ls on one of the containers looks similar to ls / on any Linux machine:

```
$ ls /cvmfs/singularity.opensciencegrid.org/opensciencegrid/osgvo-el7:latest/  
cvmfs  host-libs  proc  sys  anaconda-post.log  lib64  
dev    media     root  tmp  bin                 sbin  
etc    mnt       run   usr  image-build-info.txt singularity  
home   opt       srv   var  lib
```

Result: Most container instances only use **a small part** of the container image **(50-150 MB)** and that part is **cached** in CVMFS! We don't care about docker layers because cvmfs deduplicates everything.

Mountable image files vs CVMFS images

- **When a high speed local mounted filesystem is available, singularity mounts the image file on the client as a loopback filesystem** – this moves the metadata operations to the client and only reads the pieces actually used – **CVMFS does too**
- **The difference is that files published in CVMFS are instantly available worldwide**
- **But, most HPC admins are suspicious of FUSE, so we build image files to use HPC allocations**
 - CMS and ATLAS application software is much larger than the OS code that they usually have in a container image; normally we bind mount /cvmfs into the container for application code
 - They find it very difficult to trim the size down, so the typical CMS and ATLAS HPC image size is around 200 GB and takes about 8 to 12 hours to build reading from CVMFS, then needs to be uploaded to each HPC filesystem
 - Typically these containers include the “pilot”, so we get no isolation between users and so have to run only “production” jobs all by the same user id, no user analysis jobs

Unprivileged, non-setuid Singularity

- **Because HTC does not need to use loopback mounts, and we can avoid overlays by using the underlay feature for adding bind mounts, so we can use the unprivileged namespace feature now standard in RHEL 7.6**
 - This is considered to be key for reducing vulnerability to risks of setuid-root. Unprivileged namespaces also get CVEs, but there are many more people examining the Linux kernel than Singularity.
 - Underlay works by first bind-mounting everything requested onto a scratch area, then bind-mounting everything else from the image onto the same scratch area, and using a read-only bind-mount of the scratch area as '/' for the container.
- **The ability to mount FUSE filesystems in unprivileged namespaces is now in the latest Linux kernels, which when it becomes available should provide even more highly useful functionality to unprivileged Singularity**

EPEL

- **WLCG has standardized on Red Hat Enterprise Linux and its derivatives, Scientific Linux and CentOS**
- **Extra Packages for Enterprise Linux (EPEL) is part of the Fedora project and adds packages for RHEL (currently RHEL6 and RHEL7)**
- **OSG has their own software support team and yum repositories, and EGI has their own yum repositories where they import rpms, and both depend on EPEL** – the original supporter of singularity in EPEL lost interest, so I volunteered to take over
- **Currently I have installed singularity-3.1.0 in the current Fedora releases, but I am waiting on at least one singularity fix (recently merged) and on more feedback from production users of EPEL singularity-2.6 before upgrading to singularity-3.x in EPEL** – not in epel-testing yet either in case there needs to be a security release 2.6.x, but it is in the osg-upcoming yum repository

Links

- <https://opensciencegrid.org>, <https://display.opensciencegrid.org>
- <http://wlcg.web.cern.ch>
- <https://cernvm.cern.ch/portal/filesystem>