# Distributing User Code with the CernVM FileSystem

Dave Dykstra

CHEP 2019

7 November 2019

Presenter: Ken Herner

Fermilab

CernVM
File system

# Distributing user code to grid jobs

- Experiment code is successfully being distributed to grid jobs with the CernVM FileSystem (CVMFS)
  - Relatively large amount of code
  - Managed by a few privileged people per experiment
  - Some delays in availability are acceptable
    - Typically delays around 30 minutes, sometimes much longer due to outages
- End user code (such as for physics analysis) has not been using CVMFS
  - Much smaller amount of code for each user
  - Authentication for so many users is a challenge
  - Significant delays are not acceptable
    - Should reliably be less than 10 minutes so job batches aren't significantly delayed

# Fermilab user code distribution story

- Many Fermilab-based experiments were distributing code via Network Attached Storage (NAS) mounted on worker nodes
  - Frequent overloads experienced
  - Couldn’t expand to grid
- NAS directories were unmounted from worker nodes
- Users asked to instead download code tarballs from high speed file server (dCache)
  - Many jobs downloading up to 3GB tarballs at the beginning of jobs from dCache overloaded individual disk servers
  - To mitigate, tarballs moved to “resilient” pool with copies on 20 disk servers each
    - Highly wasteful of disk space, and both LAN and WAN bandwidth

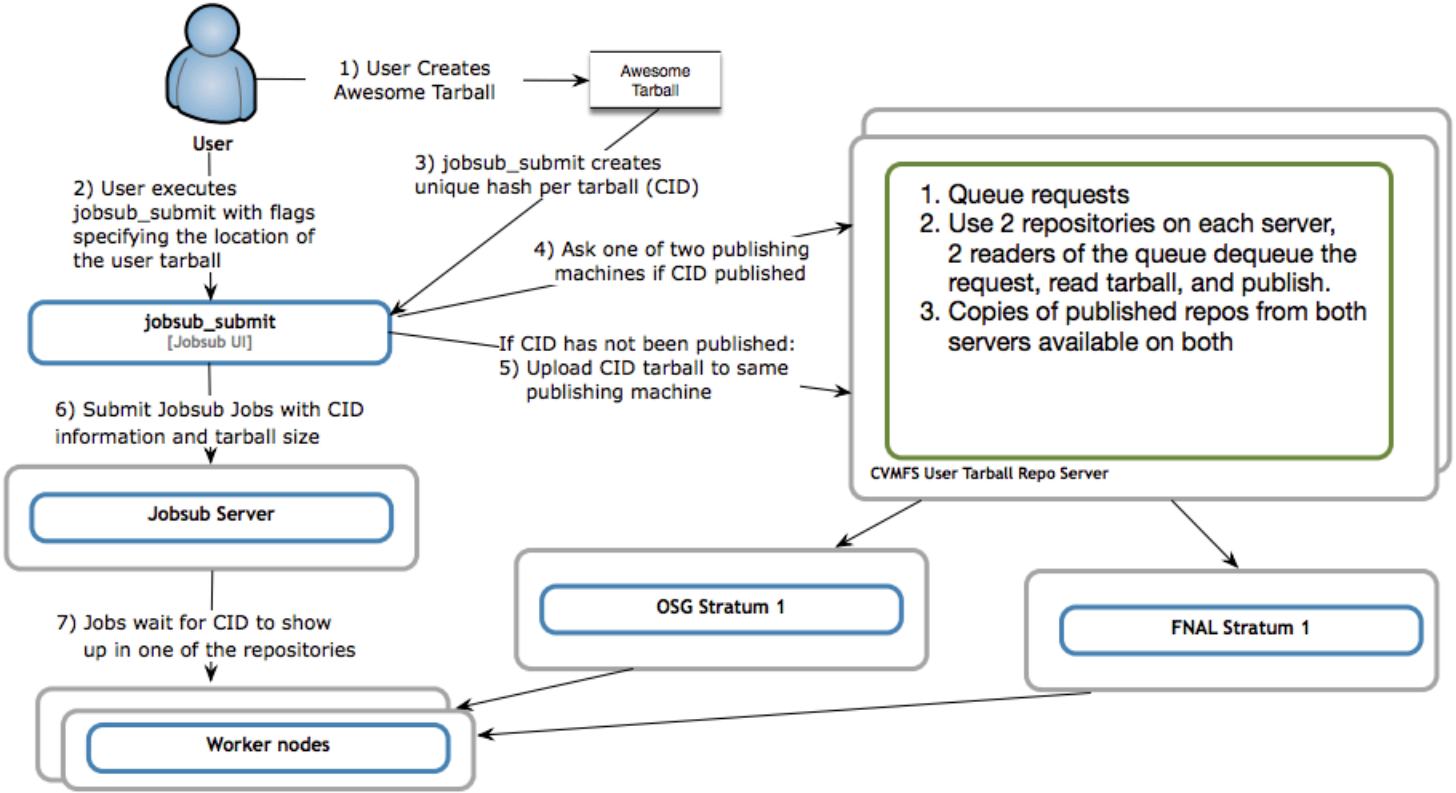
# Considering CVMFS for user code

- CVMFS is efficient for code distribution
  - Software tends to have many files in common with previous versions
    - Files reused thanks to cvmfs deduplication
  - Site and worker node caching works great for running many jobs
  - Only the subset of files that are actually used are downloaded
- The main challenge for user code on CVMFS is that standard publish + distribution delays are too long
  - A proof-of-concept test using existing tarballs from dCache showed publish rate could be reasonably handled by one server
  - Distribution timing parameters can be reduced
- Reliability is much more important than for standard code
  - Use two servers

# System design

- Two publishing servers for redundancy
- Each publishing server provides web api
- User code tarballs given unique Code ID (CID) based on hash of content
- Automatically clean out old tarballs
- Two repositories on each server so cleanups don't block publishing
  - When not cleaning up, publish in parallel
- Integrate with local job submission system (jobsub in Fermilab's case)
  - Clients directly upload tarballs to a publishing server
  - Authenticated by X.509 proxy
- Publishing servers unpack each tarball in a directory name based on its CID
- Minimize distribution delays to be less than 5 minutes
- Job wrapper script waits (with a timeout) for CID to appear in any of the four repositories and passes the directory path to the job

# Control flow



# Publishing server API

- /pubapi/publish?cid=XXX
  - Uses POST to upload tarball; easy to do with curl
  - Does queueing and publishing
  - Responds OK when queued or PRESENT if cid already existed
    - Also updates a timestamp if PRESENT, to defer cleanup
  - CIDs assigned by client; api accepts any CID and may include slashes to group into subdirectories
    - Fermilab's client puts them into subdirectories by VO
- /pubapi/exists?cid=XXX
  - Responds MISSING or PRESENT
- /pubapi/update?cid=XXX
  - Exactly like exists, except updates the timestamp if PRESENT
- /pubapi/config
  - Returns configuration, currently a list of configured repositories
- /pubapi/ping
  - Returns OK, for monitoring and load balancing purposes

# Repository cleanup

- CID directories are automatically removed after a configurable number of days (default 30) since the last time they were used
  - CVMFS is only used in this system for grid jobs, not as an archive
  - Users must keep track of their own code and can republish if they need the same code again later
- Timestamps for previously published CIDs are stored in any repository when a CID is reused
  - Because server and repository allocated for new publish might be different than the original one
- Cleanups happen in one repository per hour starting at a configurable hour of the night, followed by cvmfs garbage collection

# Publishing server software packaging

- Most of the publishing server software is packaged in a single rpm plus its dependencies
  - Designed to be able to be deployed by multiple organizations
  - Nothing Fermilab-specific
- Configuration is in a single simple file
  - Mainly just repository names needed
  - Some other standard system configuration needed such as grid-mapfile
- Creates repositories and replicas
- Provides https web api
- Does the automatic cleanup

# Minimizing distribution delays

- Distribution update time was minimized
  - Time-To-Live set to 15 seconds in repository configuration instead of usual 4 minutes
  - Cache delay set to usual 61 seconds in Apache
  - Stratum 1 set to check for updates to these repositories twice per minute instead of usual once every 5 minutes
    - Separate cron so doesn't need to wait for other repositories
  - cvmfs client kernel cache flush takes the usual 1 minute
- Total delay for small updates should be less than 3 minutes after publication

# Status

- Production publication system is in place, including expedited updates
- Updates to jobsub put into production yesterday
- VOs will begin transitioning to it soon
- Publishing server code available as open source
  - https://github.com/cvmfs-contrib/cvmfs-user-pub
- Thank you to other contributors: Shreyas Bhat, Dennis Box, Hyunwoo Kim, and Tanya Levshina