CernVM-FS with Apptainer/Singularity: A Great Combination

Dave Dykstra, dwd@fnal.gov

EasyBuild User Meeting

25 January 2022

Fermilab

CernVM File system

Apptainer

CernVM-FS (CVMFS) features

- Files appear immediately present but are only downloaded on demand
- Origin of the files can be from repositories on a “Stratum 0” hosted anywhere in the world
- Metadata operations are done on the client node, using “catalogs” downloaded in chunks of about 100K files or less
- Files are stored and transferred named by a secure hash of the *content*, deduplicated and compressed
- Served from small number of worldwide “Stratum 1” servers, cached in http proxy caching servers (squid) at each site, and cached on each client node
- Cryptographically verified with a digital signature on one small file in each repository
- Larger files broken up into ~8MB chunks by default to smooth out the load on servers
- 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

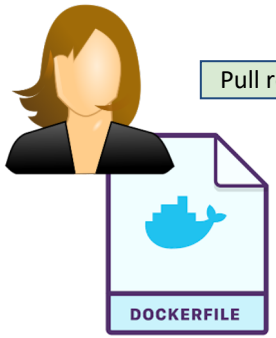
Distributing containers in CVMFS

- CVMFS is an ideal method for distributing containers with Apptainer/ Singularity “sandbox” style of unpacked files
  - Has all the advantages of SIF files that made Singularity popular on HPC systems:
    - All the metadata operations are moved to the client (huge performance win)
    - Only the portions that are used need to be read
    - Cryptographically verified
    - Stored compressed
  - In addition it has many other advantages:
    - Instant worldwide updates
    - No need for setuid privileges to mount SIF files (singularity and apptainer themselves are distributed in CVMFS and run using unprivileged user namespaces)
    - No need for local high speed file server (excellent for High Throughput Computing)
    - Scales to much larger application software base (CMS and ATLAS SIF files including many software releases were ~200GB and took half a day to generate)
    - Much more efficient storage due to file-level deduplication

Shared container repositories

- CERN and OSG host shared repositories for containers automatically updated from dockerhub (or other container registries)
  - /cvmfs/singularity.opensciencegrid.org
  - /cvmfs/unpacked.cern.ch
- Using shared repositories has the great advantage of deduplication of many common operating system files, plus convenience of updates
- Projects generally host containers only for the base operating system plus the packaged software that they need to add
  - Application software goes in project-specific separate repositories, e.g. /cvmfs/dune.opensciencegrid.org or /cvmfs/cms.cern.ch
  - Later in the talk I'll discuss plans for full-application end user containers
- The cvmfs project includes a tool to manage container repositories called cvmfs-ducc, and OSG has its own (which predated cvmfs-ducc) called cvmfs-singularity-sync

Workflow to update containers in shared repos



Pull request or ticket to register container (one time)

GitHub

docker build + push

git push

GitHub

Automatic builds on changes

docker Hub

Automatic sync on changes

/cvmfs

CVMFS on HPC

- CVMFS was designed for and is essential for High Throughput Computing, but is also very valuable for High Performance Computing
- Installing CVMFS on HPC sometimes poses some special challenges
  - CVMFS requires local cache, and some HPC nodes are diskless
    - Many alternatives, but best is usually for system admin to mount separate loopback filesystem per node for cache and for local job scratch workspace, which has advantage of moving metadata operations to the node
  - CVMFS is FUSE-based and some system admins are suspicious of FUSE
  - Not all HPC users need CVMFS, and system admins don't always want to reserve resources for it when it is not being used

CVMFS completely unprivileged

- The easiest and most robust way to install CVMFS is using a package installed as root by the system administrator
- When that is not possible, my cvmfsexec package makes it easy as possible to use cvmfs as an unprivileged user
  - There are 4 different ways to use it, all of which have caveats
  - The easiest method with the least caveats is the 3rd documented method, which depends on unprivileged user namespaces including unprivileged fuse mount, available in RHEL >= 7.8. I will focus on that.

cvmfsexec example usage

- Included makedist tool downloads cvmfs software and configuration that can be sent to jobs
  - creates default, osg, or egi cvmfs configuration
- Example with cvmfsexec method 3:

```
$ git clone https://github.com/cvmfs/cvmfsexec
$ cd cvmfsexec
$ makedist osg
$ cvmfsexec grid.cern.ch atlas.cern.ch -- ls /cvmfs
atlas.cern.ch config-osg.opensciencegrid.org grid.cern.ch
```

- Can also run singularity inside it:

```
$ cvmfsexec oasis.opensciencegrid.org singularity.opensciencegrid.org -- \
     /cvmfs/oasis.opensciencegrid.org/mis/apptainer/bin/apptainer run -C \
     /cvmfs/singularity.opensciencegrid.org/library/centos:centos7
```

Self-extracting distribution script

- After running makedist, use makedist -o to make self-extracting script including the cvmfs distribution

```
makedist –o /tmp/cvmfsexec
```

- Send /tmp/cvmfsexec to a job, and when it is executed it will extract the cvmfsexec and cvmfs distribution into a .cvmfsexec subdirectory and run from there

What about squids?

- cvmfs requires a local squid cache to work well at scale
- Between makedist and makedist -o you can edit cvmfs configuration to identify the squid's location
- Or, default configuration uses Worldwide LHC Grid (WLCG) Web Proxy Auto Discovery (WPAD) servers at CERN & Fermilab
  - following WLCG standard, first looks for local http://grid-wpad/wpad.dat or http://wpad/wpad.dat services
  - if those are not found, http://cernvm-wpad.cern.ch/wpad.dat or http://cernvm-wpad.fnal.gov/wpad.dat are consulted
    - if squids are known for the requesting GeoIP organization, they are returned
    - if no squids are known, connects DIRECT to openhtc.io Cloudflare aliases
    - if many requests from same org with no squid within 15 minutes, directs to monitored fallback squids at CERN or FNAL to identify where local squids are needed
- frontier-squid can auto-register itself with WLCG WPAD (via shoal)

Local cache considerations

- Cache has to be managed carefully with production use
  - by default, mountrepo (invoked by cvmfsexec to mount a repository) just allocates some space (4GB) under its dist subdirectory, shared between the repositories mounted
  - multiple jobs on the same machine can’t easily share the cache
    - works best if controlled by pilot jobs allocated with as large a portion of a node as possible
- Avoid putting the cache on shared filesystems because of high load on metadata servers
  - Loopback-mounted filesystems work better

Scaling up number of containers in CVMFS

- Manually registering containers for CVMFS distribution is designed for small numbers of containers per project
  - Intended for small group of people that maintain base software distributions for a project
- Many different considerations in order to scale this up to large numbers of researchers in large projects running individual analysis jobs
  - Time to publish containers can become prohibitive
  - Time to distribute to stratum 1s and clients needs to be minimal
  - Reliability more critical for fast turnaround in development cycle
  - Cleanup of software no longer used by jobs should be automated

New CVMFS feature

- There's a new feature in cvmfs-server-2.9.0 that addresses the container publish time problem
  - If a fully self-contained container is made up of many docker layers, presumably almost all the files will be in layers shared with many members of a large project
    - No project software would come from a separate CVMFS repository as is now the case
  - The new feature enables a quick copy of only "catalogs" from one path to another
    - Files would be shared in any case because of deduplication, but this avoids the very costly time to reprocess & republish all those shared files
  - cvmfs-ducc detects when layers are shared and invokes new catalog copy feature

Remaining issues addressed in existing package

- My cvmfs-user-pub package used only at Fermilab so far addresses the other issues
  - Publishes only user software, outside containers, from tarballs
  - Reliability through redundancy
    - 2 dedicated publishing servers, each with 2 CVMFS repositories so “garbage collection” never blocks publishing
  - CVMFS distribution time minimized
    - Stratum 1s update the repositories on a fast track every minute instead of (typically) every 5 minutes shared serially with all the other repositories
    - Time between clients checking for updates reduced from 4 minutes to 15 seconds
    - Result is updates should always be available on clients within a few minutes
  - Automatic deletion of publishes that haven’t been used in 30 days
- Plan is to extend cvmfs-user-pub to publish containers using cvmfs-ducc

Links

- https://cernvm.cern.ch/fs/
- https://apptainer.org
- https://cvmfs.readthedocs.io/en/stable/cpt-ducc.html
- https://github.com/opensciencegrid/cvmfs-singularity-sync
- https://github.com/cvmfs/cvmfsexec
- https://github.com/cvmfs-contrib/cvmfs-user-pub