CernVM-FS Status and Development Plans

Jakob Blomer

ALICE T1/T2 Workshop, Strasbourg
May 5th, 2017
Current Scale of Deployment

- > 475 million files under management
- > 75 repositories
- alice.cern.ch: 18 million files, 2.3 TB
- alice-ocdb.cern.ch: 1.2 million files, 280 GB
- alice-nightlies.cern.ch: 1 million files, 190 GB
Please remove ASCG from your worker node configuration

- cvmfs-config-default version 1.3
- check /etc/cvmfs/domain.d/cern.ch.{local,conf}
New features to be rolled out with release 2.4

- Instant access to snapshots and branching
- Client cache plugins and tiered cache
- Distributed release manager machines
- Reduction of content propagation delay: “5 Minutes CernVM-FS”
- Docker graph driver plugin (independently released)
Developments Under Construction
New features in 2.4 release

1. **Exposed Access to Snapshots.** A new virtual directory that provides access to snapshot access within a single mountpoint, like

   /cvmfs/alice-ocdb.cern.ch/.cvmfs/snapshots/tag-xyz

2. **Branching.** Support for hotfixes of historic execution environments

   ![Branching Diagram]

3. **Snapshot Diffs.** Show change set between any to snapshots

   ```
   $ cvmfs_server diff alice.cern.ch v1.0 v2.0
   M /changelog (File)
   M /latest (Link)
   A /v2.0 (Directory)
   R /externals/unused-lib (Directory)
   ```
Demo on alice-ocdb.cern.ch
CernVM-FS Cache Plugins

Motivation for cache plugins

- More **flexibility** for client deployment:
  - Diskless server farms
  - HPC “burst buffers”: utilize fast, possibly non-POSIX storage
- Opens the door to external contributions!

For standard deployment on the Grid nothing changes!
HPC Example: hot cache in memory, warm cache in cluster file system

```
CVMFS_WORKSPACE=/var/lib/cvmfs  # Named pipes, sockets, etc

CVMFS_CACHE_PRIMARY="hpc"

CVMFS_CACHE_hpc_TYPE=tiered
CVMFS_CACHE_hpc_UPPER="memory"
CVMFS_CACHE_hpc_LOWER="preloaded"

CVMFS_CACHE_memory_TYPE=external
CVMFS_CACHE_memory_CMDLINE=/usr/libexec/cvmfs/cache/cvmfs_cache_ram,
/etc/cvmfs/default.local
CVMFS_CACHE_memory_LOCATOR=unix=/var/lib/cvmfs/cvmfs-cache.socket

# Preloaded alien cache directory on GPFS
CVMFS_CACHE_preloaded_TYPE=posix
CVMFS_CACHE_preloaded_ALIEN=/gpfs/cvmfs_cache
CVMFS_CACHE_preloaded_SHARED=no
CVMFS_CACHE_preloaded_QUOTA_LIMIT=-1

# Plugin configuration
CVMFS_CACHE_PLUGIN_LOCATOR=unix=/var/lib/cvmfs/cvmfs-cache.socket
CVMFS_CACHE_PLUGIN_SIZE=1024
```
Distributed Release Manager Machines

User interface remains largely the same:

```
cvmfs_server transaction /ocdb/2018/run001
```

Most components functional, currently working on final catalog merging

Work by R Popescu
Reducing propagation delay from <33 min to <5 min

1. Triggered replication, part of publish operation: 15 min → 0
2. Apache object expiry configuration: 2 min → 30 s
3. CernVM-FS file catalog TTL: 15 min → 4 min
4. Improved Fuse kernel cache handling (RHEL≥7): 1 min → few seconds

Modifications 1–3 rolled out for alice-ocdb.cern.ch / CERN Stratum 1!
Docker Graphdriver Plugin
Docker Workflow

Layers are tarfiles, which need to be downloaded and locally extracted.
Docker Graph Driver Plugin

Work by N Hardi, expected H2/2017

Host machine

Docker client

Docker daemon

Graphdriver plugin

Graphdriver plugin API

S3 client

CVMFS Client

plugin API

Internet

S3

CVMFS

Docker registry

Graphdriver plugin

Regular image

Thin image

read-write layer

local read-only layer

thin image layer

read-only layer on CVMFS

jblomer@cern.ch

Containers
## A number of exciting features for the 2.4 release

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>./cvmfs/snapshots directory</td>
<td>implemented</td>
</tr>
<tr>
<td>Branching</td>
<td>implemented</td>
</tr>
<tr>
<td>Diff viewer</td>
<td>implemented</td>
</tr>
<tr>
<td>External cache plugins</td>
<td>implemented</td>
</tr>
<tr>
<td>Tiered cache</td>
<td>implemented</td>
</tr>
<tr>
<td>In-memory cache plugin</td>
<td>implemented</td>
</tr>
<tr>
<td>Distributed release manager machines</td>
<td>in progress</td>
</tr>
<tr>
<td>“5 minutes CernVM-FS”</td>
<td>rollout started (alice-ocdb)</td>
</tr>
<tr>
<td>Docker graph driver plugin</td>
<td>working prototype</td>
</tr>
</tbody>
</table>

- ALICE is a key driver for CernVM-FS developments!

Many of the 2.4 developments are triggered by ALICE use cases
Backup
Source code: https://github.com/cvmfs/cvmfs
https://github.com/cernvm

Downloads: https://cernvm.cern.ch/portal/filesystem/downloads
https://cernvm.cern.ch/portal/downloads

Documentation: https://cvmfs.readthedocs.org

Mailing list: cvmfs-talk@cern.ch
cernvm-talk@cern.ch

JIRA bug tracker: https://sft.its.cern.ch/jira/projects/CVM
# CernVM-FS In Containers

## Bind Mount

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>docker run -v /cvmfs:/cvmfs:shared ...</code> or</td>
<td>Cache shared by all containers on the same host</td>
</tr>
<tr>
<td><code>docker run -v /cvmfs/sft.cern.ch:/cvmfs/sft.cern.ch ...</code></td>
<td></td>
</tr>
</tbody>
</table>

## Docker Volume Driver

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://gitlab.cern.ch/cloud-infrastructure/docker-volume-cvmfs/">Link</a></td>
<td>Integrates with Kubernetes</td>
</tr>
<tr>
<td><code>docker run --volume-driver cvmfs -v</code></td>
<td></td>
</tr>
<tr>
<td><code>cms.cern.ch:/cvmfs/cms.cern.ch ...</code></td>
<td></td>
</tr>
</tbody>
</table>

## From Inside Container

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>docker run --privileged ...</code></td>
<td>Probably not very much used in practice</td>
</tr>
</tbody>
</table>
Callbacks to be implemented by plugin developer

// Reading data
int cvmcache_chrefcnt(struct hash object_id, int change_by);
int cvmcache_object_info(struct hash object_id, struct object_info *info);
int cvmcache_pread(struct hash object_id, int offset, int size, void *buffer);

// Transactional writing in fixed-sized parts
int cvmcache_start_txn(struct hash object_id, int txn_id, struct info object_info);
int cvmcache_write_txn(int txn_id, void *buffer, int size);
int cvmcache_abort_txn(int txn_id);
int cvmcache_commit_txn(int txn_id);

// Optional: quota management
int cvmcache_shrink(int shrink_to, int *used);
int cvmcache_listing_begin(...);
int cvmcache_listing_next(int listing_id, ...);
int cvmcache_listing_end(int listing_id);