Global Software Distribution with CernVM-FS

Jakob Blomer
CERN

2016 CCL Workshop on Scalable Computing
October 19th, 2016
The Anatomy of a Scientific Software Stack
(In High Energy Physics)

$ cmsRun DiPhoton_Analysis.py$
The Anatomy of a Scientific Software Stack
(In High Energy Physics)

$\text{cmsRun DiPhoton\_Analysis.py}$

My Analysis Code
< 10 Python Classes

CMS Software Framework
$O(1000)$ C++ Classes

Simulation and I/O Libraries
ROOT, Geant4, MC-XYZ

CentOS 6 and Utilities
$O(10)$ Libraries
The Anatomy of a Scientific Software Stack
(In High Energy Physics)

$\texttt{cmsRun DiPhoton\_Analysis.py}$

### How to install on...
- **my laptop:**
  - compile into `/opt`
  - $\sim 1$ week
- **my local cluster:**
  - ask sys-admin to install in `/nfs/software`
  - $> 1$ week
- **someone else’s cluster:** ?

### CentOS 6 and Utilities
- $O(10)$ Libraries

### Simulation and I/O Libraries
- ROOT, Geant4, MC-XYZ

### CMS Software Framework
- $O(1000)$ C++ Classes

### My Analysis Code
- $< 10$ Python Classes
The Anatomy of a Scientific Software Stack
(In High Energy Physics)

$\texttt{cmsRun DiPhoton\_Analysis.py}$

How to install (again) on...

- my laptop:
  compile into /opt
  \(\sim\) 1 week
- my local cluster:
  ask sys-admin to install in /nfs/software
  > 1 week
- someone else’s cluster: ?
Beyond the Local Cluster

**World Wide LHC Computing Grid**

- ~200 sites: from 100 to 100,000 cores
- Different countries, institutions, batch schedulers, OSs, …
- Augmented by clouds, supercomputers, LHC@Home
What about Docker?

**Example: R in Docker**

```bash
$ docker pull r-base
→ 1GB image
$ docker run -it r-base
$ ... (fitting tutorial)
→ only 30 MB used
```

**It’s hard to scale Docker:**

<table>
<thead>
<tr>
<th>iPhone App</th>
<th>Docker Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 MB</td>
<td>1 GB</td>
</tr>
<tr>
<td>changes every month</td>
<td>changes twice a week</td>
</tr>
<tr>
<td>phones update staggered</td>
<td>servers update synchronized</td>
</tr>
</tbody>
</table>

→ Your preferred cluster or supercomputer might not run Docker
A File System for Software Distribution

Pioneered by CCL’s GROW-FS for CDF at Tevatron

Refined in CernVM-FS, in production for CERN’s LHC and other experiments

1. Single point of publishing
2. HTTP transport, access and caching on demand
3. Important for scaling: bulk meta-data download (not shown)
Two independent issues

1. How to mount a file system (on someone else’s computer)?
2. How to distribute immutable, independent objects?
Content-Addressable Storage: Data Structures

Object Store
- Compressed files and chunks
- De-duplicated

File Catalog
- Directory structure, symlinks
- Content hashes of regular files
- Digitally signed → integrity, authenticity
- Time to live
- Partitioned / Merkle hashes (possibility of sub catalogs)

⇒ Immutable files, trivial to check for corruption, versioning

Repository

Object Store

File catalogs

SQLite

jblomer@cern.ch
Transactional Publish Interface

Publishing New Content

\[
\begin{align*}
[ \sim ] & \# \text{ cvmfs\_server transaction icecube.opensciencegrid.org} \\
[ \sim ] & \# \text{ make DESTDIR=~/cvmfs/opensciencgrid.org/amd64-gcc6.0/4.2.0 install} \\
[ \sim ] & \# \text{ cvmfs\_server publish icecube.opensciencegrid.org}
\end{align*}
\]

Uses `cvmfs\_server` tools and an Apache web server
Publishing New Content

[ ~ ]# cvmfs_server transaction icecube.opensciencegrid.org
[ ~ ]# make DESTDIR=/cvmfs/opensciencegrid.org/amd64-gcc6.0/4.2.0 install
[ ~ ]# cvmfs_server publish icecube.opensciencegrid.org

Uses cvmfs-server tools and an Apache web server
Content Distribution over the Web

Server side: stateless services

Data Center

Caching Proxy
\(O(100)\) nodes / server

Worker Nodes

HTTP \(\rightarrow\) Caching Proxy

Web Server
\(O(10)\) DCs / server

HTTP \(\rightarrow\) Web Server
Content Distribution over the Web

Server side: stateless services

Data Center

Load Balancing
$\mathcal{O}(100)$ nodes / server

Worker Nodes

Web Server
$\mathcal{O}(10)$ DCs / server

HTTP

jblomer@cern.ch

CernVM-FS

9 / 15
Content Distribution over the Web

Server side: stateless services

Data Center

Caching Proxies
\( \mathcal{O}(100) \) nodes / server

Worker Nodes

Failover

HTTP

Web Server

\( \mathcal{O}(10) \) DCs / server

HTTP
Content Distribution over the Web

Server side: stateless services

Data Center

Caching Proxies
$\mathcal{O}(100)$ nodes / server

Failover

HTTP

Worker Nodes

Mirror Servers
$\mathcal{O}(10)$ DCs / server

Geo-IP
Content Distribution over the Web

Server side: stateless services

Data Center

Caching Proxies
\( O(100) \) nodes / server

Mirror Servers
\( O(10) \) DCs / server

Worker Nodes

HTTP

Failover

jblomer@cern.ch

CernVM-FS
Server side: stateless services

Data Center
Worker Nodes
Prefetched Cache

Caching Proxies
$\mathcal{O}(100)$ nodes / server

Mirror Servers
$\mathcal{O}(10)$ DCs / server
Mounting the File System Client: Fuse

Available for RHEL, Ubuntu, OS X; Intel, ARM, Power
Works on most grids and virtual machines (cloud)

open(/ChangeLog)
glibc
VFS
inode cache
dentry cache
syscall
/dev/fuse
Fuse
: ext3
NFS
libfuse
CernVM-FS
file descr.
SHA1
inflated+verify
HTTP GET

jblomer@cern.ch
CernVM-FS
Mounting the File System Client: Parrot

Available for Linux / Intel

Works on supercomputers, opportunistic clusters, in containers

Parrot Sandbox

open(/ChangeLog) -> fd

glibc

syscall / Parrot

libcvmfs

file descr.

SHA1

inflated + verify

HTTP GET

VFS
inode cache
dentry cache

NFS

ext3

SQLite

user space
kernel space

jblomer@cern.ch

CernVM-FS

11 / 15
Scale of Deployment

- > 350 million files under management
- > 50 repositories
- Installation service by OSG and EGI
Docker Integration

Docker Daemon

pull & push containers

Docker Registry

Improved Docker Daemon

Funded Project

file-based transfer

CernVM File System

Under Construction!

jblomer@cern.ch
Client Cache Manager Plugins

**Draft C Interface**

```c
// Client Cache Manager Plugins

void cvmfs_add_refcount(struct hash object_id, int change_by);
void cvmfs_pread(struct hash object_id, int offset, int size, void *buffer);

// Transactional writing in fixed-sized chunks
void cvmfs_start_txn(struct hash object_id, int txn_id, struct info object_info);
void cvmfs_write_txn(int txn_id, void *buffer, int size);
void cvmfs_abort_txn(int txn_id);
void cvmfs_commit_txn(int txn_id);
```

Under Construction!

jblomer@cern.ch  CernVM-FS  14 / 15
Summary

CernVM-FS

- Global, HTTP-based file system for software distribution
- Works great with Parrot
- Optimized for small files, heavy meta-data workload
- Open source (BSD), used beyond high-energy physics

Use Cases

- Scientific software
- Distribution of static data e.g. conditions, calibration
- VM / container distribution cf. CernVM
- Building block for long-term data preservation

Source code: https://github.com/cvmfs/cvmfs
Downloads: https://cernvm.cern.ch/portal/filesystem/downloads
Documentation: https://cvmfs.readthedocs.org
Mailing list: cvmfs-talk@cern.ch
Backup Slides
### Fuse Module
- Normal namespace: `/cvmfs/<repository>`
  e.g. `/cvmfs/atlas.cern.ch`
- Private mount as a user possible
- One process per fuse module + watchdog process
- Cache on local disk
- Cache LRU managed
- NFS Export Mode
- Hotpatch functionality
  `cvmfs_config reload`

### Mount helpers
- Setup environment (number of file descriptors, access rights, ...)
- Used by `autofs` on `/cvmfs`
- Used by `/etc/fstab` or mount as root
  `mount -t cvmfs atlas.cern.ch /cvmfs/atlas.cern.ch`

### Diagnostics
- Nagios check available
- `cvmfs_config probe`
- `cvmfs_config chksetup`
- `cvmfs_fsck`
- `cvmfs_talk`, connect to running instance

### Parrot
- Built in by default
Statistics over 2 Years

Software Directory Tree

- atlas.cern.ch
  - repo
    - software
      - x86_64-gcc43
      - 17.1.0
      - 17.2.0
      - ...

Files

File System Entries $[\times 10^6]$
Experiment Software from a File System Viewpoint

Statistics over 2 Years

File System Entries [$\times 10^6$]

File Kernel

Duplicates

Software Directory Tree

atlas.cern.ch

repo

software

x86_64-gcc43

17.1.0

17.2.0

...
Between consecutive software versions: only $\approx 15\%$ new files
Experiment Software from a File System Viewpoint

Statistics over 2 Years

File System Entries [$\times 10^6$]

-folded into directories and symlinks

File Kernel

Software Directory Tree

- atlas.cern.ch
- repo
- software
- x86_64-gcc43
  - 17.1.0
  - 17.2.0
  - ...

Fine-grained software structure (Conway’s law)
Between consecutive software versions: only $\approx 15\%$ new files

jblomer@cern.ch
Directory Organization

Typical (non-LHC) software: majority of files in directory level ≤ 5
Cumulative File Size Distribution

cf. Tanenbaum et al. 2006 for “Unix” and “Webserver”

Good compression rates (factor 2–3)
## Runtime Behavior

### Working Set
- \( \approx 10\% \) of all available files are requested at runtime
- Median of file sizes: \(< 4\, \text{kB}\)

### Flash Crowd Effect
- Up to 500 kHz meta data request rate
- Up to 1 kHz file open request rate

---

Shared Software Area

\[ \text{dDoS} \]

---

jblomer@cern.ch
Software vs. Data

Based on ATLAS Figures 2012

<table>
<thead>
<tr>
<th>Software</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX Interface</td>
<td>put, get, seek, streaming</td>
</tr>
<tr>
<td>File dependencies</td>
<td>Independent files</td>
</tr>
<tr>
<td>$10^7$ objects</td>
<td>$10^8$ objects</td>
</tr>
<tr>
<td>$10^{12}$ B volume</td>
<td>$10^{16}$ B volume</td>
</tr>
<tr>
<td>Whole files</td>
<td>File chunks</td>
</tr>
<tr>
<td>Absolute paths</td>
<td>Any mountpoint</td>
</tr>
<tr>
<td>Open source</td>
<td>Confidential</td>
</tr>
</tbody>
</table>

WORM (‘‘write-once-read-many’’)
Versioned