Status and Future Perspectives of CernVM-FS
http://cernvm.cern.ch/portal/filesystem

Jakob Blomer, Predrag Buncic, Ioannis Charalampidis, Artem Harutyunyan, Dag Larsen, René Meusel

CERN    PH-SFT

CHEP 2012
1. CernVM-FS Overview

2. Providing the CernVM-FS Client in Heterogeneous Environments

3. Improving CernVM-FS on the Publisher’s End

4. Summary
CernVM File System – Overview

Caching HTTP file system, optimized for software delivery

Operating System & Applications

CernVM-FS

HTTP Content Distribution Network

OS Kernel

Fuse

File System Buffers

CernVM-FS Hard Disk Cache

CernVM-FS “Repository” (All Releases Available)

(Known) Users: ATLAS (+ Conditions Data), LHCb (+ Conditions Data), CMS, NA61, NA49, BOSS, Geant4, AMS, LHC@Home 2.0

CDN: Full replicas at CERN, RAL, BNL, ASGC, FermiLab

Site-local cache servers (Frontier Squids)

Avg. Load: Very modest,

\[ \approx 5 \text{ MB/s}, 20 \text{ requests per second on CERN Replica} \]

Volume: 75 million objects (2010: 30 million), 5 TB (2010: 1 TB)
CernVM File System – Overview

Caching HTTP file system, optimized for software delivery

ATLAS + AMS Volume
- ~300G / month
- ~3 Million objects / month

Installation of AMS software (no negative impact on operations)

Plot provided by Line Everaerts
1. CernVM-FS Overview

2. Providing the CernVM-FS Client in Heterogeneous Environments

3. Improving CernVM-FS on the Publisher’s End

4. Summary
CernVM-FS Client in Heterogenous Environments

In order to fully benefit from CernVM-FS, the file system has to be available on all relevant computing resources.

Range of Environments:

- Scientific Linux, Ubuntu, SuSE, OS X
- 1 core to 48+ cores
- 1 to 10 mounted repositories
- Possibly no Fuse, no local hard disk

Portability:

- Portable C++ / POSIX code
- Library interface, connector to Parrot (by Dan Bradley)

Scalability:

- Memory fragmentation
  - open hash collision resolution $\Rightarrow$ linear probing
  - path strings stored on the stack
- Cache thrashing
  - direct mapped cache $\Rightarrow$ LRU cache
- Concurrent file system access
  - Fine-grained locking
  - Asynchronous, parallel HTTP I/O
Benchmarks

Speed comparison between 2.0.11 and 2.1.X

Scalability – Independent Processes (Warm Caches)

- ATLAS Athena (2.0.11)
- ATLAS Athena (cvmfs 2.1.X)
- LHCb Brunel (cvmfs 2.0.11)
- LHCb Brunel (cvmfs 2.1.X)

1LHCb uses ≈300 file catalogs: fine-grained locking pays off
Benchmarks
Speed comparison between 2.0.11 and 2.1.X

Scalability – Multi-Threading (Compile Kernel)

- cold cache, cvmfs 2.0.11
- cold cache, cvmfs 2.1.X
- warm cache, cvmfs 2.0.11
- warm cache, cvmfs 2.1.X

Scalability of version 2.1.X limited by the scalability of the kernel build system
Benchmarks

Memory comparison between 2.0.11 and 2.1.X

Memory Consumption

| Number of Cores | Memory Consumption [MB] |
|-----------------|-------------------------|
| 1               | 20                      |
| 4               | 25                      |
| 8               | 30                      |

“ever-growing” libfuse cache

1 libfuse cache shrinks on high memory pressure
Memory Consumption – Worst Case (du /cvmfs/sft.cern.ch)

- Worst case: recursive listing of /cvmfs/sft.cern.ch (1.5 Million entries, up to 6000 entries per directory)
- Memory fragmentation with std::string becomes an issue
Issue: Enforce shared *quota*, coordinated bookkeeping required

Idea: Turn the cache manager thread into a shared process

- No extra service: automatically spawned by first `cvmfs` mount point, automatically terminated by last unmount
- Named pipe can be turned into a network socket: Foundation for distributed shared memory cache
1. CernVM-FS Overview

2. Providing the CernVM-FS Client in Heterogeneous Environments

3. Improving CernVM-FS on the Publisher’s End

4. Summary
Motivation

**Benchmark:** Distribute new ATLAS release
400,000 files and directories, 10 GB compressed, 20% new data

Necessary steps and delays with current version:

1. New release
2. Installation
3. Publishing 1.5 h
4. Replication 1 h to 2 h
5. Squid TTL 15 min
6. CernVM-FS TTL 1 h
Available

(Compared to “Grid Installation Jobs”: delay reduced from days to 4 h to 5 h)

**Challenges:**
- POSIX read/write interface required
- Bulk write of many small files

**Goal:** Overall delay less than 1 h
- AUFS part of Scientific Linux
- < 5% performance loss (untar)

Improvements compared to a separate read-write copy of the repository:

- Authoritative repository storage benefits from de-duplication
  Storage savings for ATLAS: from 22 Million to 1.8 Million objects
- Encapsulated change set in scratch area
- Snapshots provided by CernVM-FS
Scaling of the Publish Process

Roles: File System Interface, Worker Node, Job Manager, Master Storage (+ Stratum 0 Webserver, Signing Server)

Protocols: Chirp, HTTP

Storage Interface: Put, Get, Rename/Commit (on Stratum 0)
New ATLAS release, 400 000 files and directories:

Details by step:

2. Encapsulated scratch area allows for fast local storage / ramdisk
3. Distributed prototype reduces processing of the changeset to <20 min
4. Immediate replication at 4 MB/s: < 10 min
5. Can be reduced to 5 min
6. Can be reduced to 5 min to 15 min

⇒ Overall delay from 210 min to 270 min to 30 min to 50 min
1. CernVM-FS Overview

2. Providing the CernVM-FS Client in Heterogeneous Environments

3. Improving CernVM-FS on the Publisher’s End

4. Summary
CernVM-FS Client

- On the way to support all relevant HEP computing resources

Publisher’s End

- Persistent storage entirely in CernVM-FS format
- Time to publish a new release can be reduced to < 1h

CernVM-FS has the potential to be used as exclusive software distribution system with low maintenance on both reader’s and publisher’s end.

Source code: https://github.com/cvmfs/cvmfs
Nightly builds: http://ecsft.cern.ch/dist/cvmfs
Mailing lists: cvmfs-talk@cern.ch, cvmfs-devel@cern.ch
Next major release planned for August 2012
Do not forget to visit the

CERN PH/SFT Group Booth

in Kimmel Center (right in front of coffee table on 4th floor)

To learn more about the CERN Virtual Machine
Poster 134: Managing Virtual Machine Lifecycle in CernVM Project
Poster 135: Long-term preservation of analysis software environment

To learn more about CernVM Co-Pilot
“CernVM Co-Pilot: an Extensible Framework for Building Scalable Cloud Computing Infrastructures”
(by A Harutyunyan)
5 Backup Slides
New meta-data memory cache:

**Memory Cache in 2.0.X**
- inode $\mapsto \{\text{path, meta-data}\}$ cache by libfuse:
  - size controlled by memory pressure
- Hash map with chaining as collision resolution (vulnerable to memory fragmentation)
- path $\mapsto$ meta-data cache:
  - direct-mapped / 2-way-associative hybrid cache (vulnerable to cache thrashing)

**Memory Cache in 2.1.X**
- All caches: CernVM-FS least-recently-used (LRU) data structure
- LRU: $\mathcal{O}(1)$, hash map with linear resolving + list
- static memory pool pre-allocated
CernVM-FS Versions: 2.0.11, 2.1.0 preview (git-86806d060e5)  default installation
Machines: Intel Xeon E5345 (8 cores), 8 GB RAM, SLC5
         AMD Opteron 6164 HE (48 cores), 96 GB RAM, SLC6
RTT to Web Proxy: 100 µs to 200 µs
Repository Revisions: ATLAS – 526 (SHA-1 eb3d939bdc7af12882383d905e52571772a946ec)
                   LHCb – 2151 (SHA-1 85c4d9e7cc3bd7db5b60cb57fd50c0a567cdd)
                   SFT – 125 (SHA-1 a271e80c9947b2c21c9bc0f5850bc551600bb6)

Benchmark Scripts:

. /cvmfs/lhcb.cern.ch/etc/login.sh
. SetupProject.sh Brunel
gaudirun.py $BRUNELSYSROOT/tests/options/testBrunel.py

. $ATLAS_LOCAL_ROOT_BASE/user/atlasLocalSetup.sh
. $AtlasSetup/scripts/asetup.sh -cmtconfig=x86_64-slc5-gcc43-opt 17.4.0
  athena.py AthExHelloWorld/HelloWorldOptions.py

cd /cvmfs/sft.cern.ch/lcg/external/experimental/linux
./compileKernel.sh 2.6.32.57

du -ch -max-depth=3 /cvmfs/sft.cern.ch
Discarding the “Shadow Tree”

Current Backend

- Repository
  - Software
  - ChangeLog
  - Compression, SHA-1
  - 806fbb67373e9...

- Local File System / NFS
  - Chunk store
  - File catalogs

- SQLite

\[ \text{Repository}_{r+1} = f(\text{Repository}_r, \text{Shadow Tree, ChangeLog}) \]

- 2 data copies
- ChangeLog requires kernel module

New Backend

- CernVM-FS Read-Only
  - Read/Write Scratch Area
  - AUFS (Union File System)

- Repository

\[ g(\text{CernVM-FS}_r, \text{Scratch Area}) \]

- Standard components
- < 5% performance loss (untar)
- Snapshots provided by CernVM-FS

(Not shown: interface to storage)

Storage savings for ATLAS:

From 22 Million to 1.8 Million file system objects
Mac OS X support
From sources
Packaging (by Manuel Giffels) ready for testing

NFS Export
For immutable mount points ready for testing
Including automatic catalog reload under investigation

Shared local hard disk cache
ready for testing

Encrypted repository / ownership support
planned

Distributed storage backend
under development

Connector to Parrot (by Dan Bradley)
to be ported from 2.0 branch

Distributed Shared Memory Cache
Automatic peer discovery ready for testing
to be ported from 2.0 branch
Support for “μ file catalogs”
to be implemented
Support for file chunking
Remote cache access
to be implemented