Journal of Physics: Conference Series **608** (2015) 012031 doi:10.1088/1742-6596/608/1/012031

Recent Developments in the CernVM-File System Server Backend

R Meusel, J Blomer, P Buncic, G Ganis, S Heikkila
CERN PH-SFT
CERN, CH-1211 Geneva 23, Switzerland

E-mail: rene.meusel@cern.ch

Abstract. The CernVM File System (CernVM-FS) is a snapshotting read-only file system designed to deliver software to grid worker nodes over HTTP in a fast, scalable and reliable way. In recent years it became the de-facto standard method to distribute HEP experiment software in the WLCG and starts to be adopted by other grid computing communities outside HEP. This paper focusses on the recent developments of the CernVM-FS Server, the central publishing point of new file system snapshots. Using a union file system, the CernVM-FS Server allows for direct manipulation of a (normally read-only) CernVM-FS volume with copy-on-write semantics. Eventually the collected changeset is transformed into a new CernVM-FS snapshot, constituting a transactional feedback loop. The generated repository data is pushed into a content addressable storage requiring only a RESTful interface and gets distributed through a hierarchy of caches to individual grid worker nodes. Additonally we describe recent features, such as file chunking, repository garbage collection and file system history that enable CernVM-FS for a wider range of use cases.

1. Introduction

The CernVM File System (CernVM-FS) is a read-only file system [1] designed for accessing large centrally installed software repositories based on HTTP. The repository content is distributed through multiple layers of replication servers and caches to individual grid worker nodes where CernVM-FS repositories are usually mounted as a FUSE [2] file system. Both meta-data and file contents are downloaded on first access and cached locally. Nowadays CernVM-FS is a mission-critical tool for the distribution of HEP software [3, 4] in the World-wide LHC Computing Grid [5] and recently gains adoption in other fields [6]. The four LHC experiments sum up to some 110 million file system objects and 4 TB of file contents, which doubled in the last two years (Figure 1 shows the growth of the ATLAS software repository as an example).

Prior to distributing, the centrally installed software repository content is prepared by the CernVM-FS server tools. Each file is compressed and stored in a content-addressable storage based on a cryptographic hash of the file content. This allows for trivial consistency checks and yields de-duplication on file content level. Relevant file system meta information (i.e. path hierarchy, access flags, file names, ...) is stored in hierarchical file catalogs which are eventually put into the content-addressable storage along with the actual data.

2. Usage Statistics and New Use Cases

In the last two years the amount of LHC experiment software accessible through CernVM-FS has more than doubled with a steady growth rate. In August 2014 CernVM-FS repositories under the domain `cern.ch`[1] were providing on-demand access to more than 7 TB of software and conditions data stored in about 125 million file system objects (i.e. files, directories, symlinks). All repositories hosted at CERN have been successfully migrated to the CernVM-FS 2.1.x repository scheme as of the 15th of September 2014 which is a crucial precondition to benefit from any of the discussed features in this work.

Beyond that, there are many more CernVM-FS repositories hosted by other institutions including but not exclusively DESY, NIKHEF and Fermilab. CernVM-FS's popularity inspired some new use cases and hence new challenges in the future development of the system. In this section we present statistics figures and a summary for future challenges due to these new use case scenarios.

2.1. Statistics for LHC Experiment Software Repositories

The four LHC experiment software repositories are accounting for more than 4 TB of files and nearly 110 million file system objects. We have seen a steady growth for those repositories (see Figure 1) while the average file size increased only slightly. However, in Section 2.2 we take a look at other CernVM-FS repositories that exceed the average file size of the LHC experiment's repositories by several orders of magnitude (cf. Tables 1 and 2).

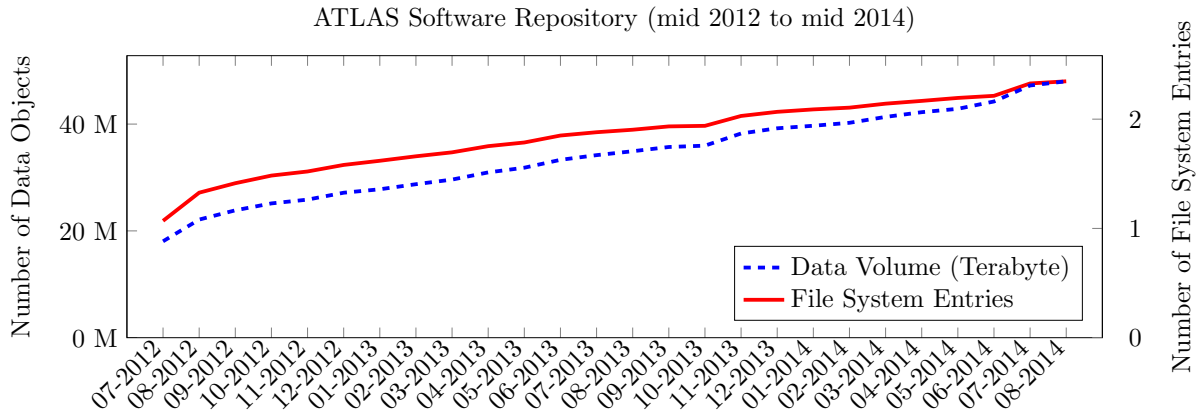

Figure 1. Growth of the ATLAS software repository (atlas.cern.ch) in the last 24 months. Showing a doubling in both file system entries (red) and available data volume (blue).

[1] alice-ocdb.cern.ch, alice.cern.ch, ams.cern.ch, atlas-condb.cern.ch, atlas.cern.ch, belle.cern.ch, boss.cern.ch, cernvm-prod.cern.ch, cms.cern.ch, geant4.cern.ch, grid.cern.ch, lhcb.cern.ch, na61.cern.ch, sft.cern.ch

Repository	# File Objects	# Data Objects	Volume	Avg File Size
atlas.cern.ch	48'000'000	3'700'000	2.2 TB	68 kB
cms.cern.ch	37'000'000	4'800'000	0.9 TB	33 kB
lhcb.cern.ch	15'800'000	4'600'000	0.5 TB	43 kB
alice.cern.ch	7'000'000	240'000	0.5 TB	94 kB
sum/average:	107'800'000	13'340'000	4.1 TB	60 kB

Table 1. File system statistics of conventional LHC experiment software repositories.
Effective: August 2014

2.2. New Use Cases of CernVM File System Impose New Challenges
CernVM-FS's scalability depends on the efficiency of its aggressive cache hierarchy. Therefore we assume that repository updates are infrequent (i.e. hours to weeks), individual distributed files are small (< 500 kiB) and the accessed working set of geographically collocated worker nodes is similar. These assumptions hold true for pure software repositories, hence the widespread and successful utilisation of CernVM-FS. Nevertheless, emerging use cases are relaxing some of these assumptions:

(i) Some LHC experiments began to distribute conditions databases[2] through CernVM-FS [3]. These files are accessed in very similar patterns as the experiment software itself, but the average *file size is orders of magnitude larger* (cf. Tables 1 and 2). CernVM-FS has to treat large files differently to avoid cache cluttering and performance degradation.

Repository	# File Objects	# Data Objects	Volume	∅ File Size
ams.cern.ch	3'700'000	1'900'000	2.0 TB	0.7 MB
alice-ocdb.cern.ch	700'000	700'000	0.1 TB	0.2 MB
atlas-condb.cern.ch	8'400	7'800	0.5 TB	60.7 MB
sum/average:	4'400'000	2'600'000	2.6 TB	20.5 MB

Table 2. File system statistics of repositories containing experiment conditions data.
Note: ams.cern.ch does contain software also
Effective: August 2014

(ii) CernVM-FS persistently stores all historic snapshots of a repository, making it a good fit for long term analysis software preservation [7]. However, it lacks a means to easily manage and access these preserved snapshots.

(iii) It is planned to use CernVM-FS to distribute nightly integration build results to simplify development and testing of experiment software. While such repositories naturally experience large and frequent updates (i.e. LHCb would publish about 1'000'000 files summing up to 50 GB per day), the particular build results need to stay available only for a short period of time. Hence, the long term persistency mentioned before *quickly fills up the repository's backend storage* with outdated revisions and requires garbage collection.

(iv) With an increasing number of institutions hosting and replicating CernVM-FS repositories the ecosystem is evolving from a centralised and controlled software distribution service into a more mesh-like structure with many stakeholders. In this new environment configuration and distribution of public keys becomes increasingly challenging for individual clients.

[2] Repositories: atlas-condb.cern.ch and alice-ocdb.cern.ch

3. New and Upcoming Features in the CernVM-FS Server

The features presented here do not represent an exhaustive list but are a selection of changes relevant to the challenges we present in Section 2.2. Note that some of the features are already available in the current stable release (i.e. CernVM-FS 2.1.19) while others are scheduled for a release in the next 3 to 6 months.

3.1. File Chunking for Large Files

CernVM-FS and its caching infrastructure were designed to serve many small files as efficiently as possible. Assuming that small files (f.e. binary executables or scripts) will be read entirely, CernVM-FS always downloads complete files on `open()` and stores them in a local cache. Larger files ($> \sim$ 50MiB) are much more likely to contain data resources (f.e. SQLite databases) that are usually read sparsely. Such access patterns in large files would unnecessarily clutter the cache and waste bandwidth.

Newer CernVM-FS Servers[3] therefore divide large files into smaller chunks, allowing for better cache exploitation and partial downloads. We use the XOR32 rolling checksum [8] for content aware cut mark detection to exploit de-duplication between similar large files in our content-addressable storage format. Hence, the chunk size is not static but varies in a configurable range (default: 4 MiB $<$ size $<$ 16 MiB).

This new feature is particularly interesting for repositories hosting large files like conditions databases. It is completely transparent for both the repository maintainer and the users of CernVM-FS and enabled by default.

3.2. Named CernVM File System Snapshots and Rollbacks

Keeping track of historic snapshots in a CernVM-FS repository was cumbersome as they were identified only by a SHA-1 hash comparable to a commit hash in Git. With named file system snapshots[4] we now provide a means to catalogue these snapshots along with an arbitrary name and a description. The benefit of this history index is two-fold: facilitated mounting of ancient repository revisions on the client side and on the other hand the possibility to rollback revisions on the server side.

Nevertheless, we do not intend to transform CernVM-FS into a fully featured version control system. Named snapshots are meant to simplify the handling of old file system snapshots only. A server side rollback therefore invalidates all named snapshots that succeed the snapshot targeted by the rollback. The server side rollback is meant as an *undo*-feature. Thus we prevent the occurrence of diverging branches in a CernVM-FS repository.

3.3. Garbage Collection on Revision Level

CernVM-FS is following an insert-only policy regarding its backend storage. Data is only added and references to it are updated, while preceding snapshots are never removed from the repository's backend storage. This yields a system where historic state remains reconstructible at the expense of an ever growing backend storage. In Section 3.2 we described how this can be utilised for long term preservation of software environments.

However, in use cases like the publishing of nightly integration build results this exhaustive history is not required or would even become prohibitively large in a short period of time.

In an experimental nightly build repository set up by LHCb, about one million new files per nightly build release summing up to some 50 GB were published. Due to de-duplication the backend storage (600 GB volume) was able to store the repository for about one month before

[3] Experimental file chunking as of version 2.1.7 (stable since 2.1.19)
[4] Named Snapshots are available since CernVM-FS 2.1.15

running full. Figure 2 plots the number of contained files (red) compared to referenced data objects (blue) and the stored data objects (brown) over the repository revisions.

Note that removing files from the CernVM-FS repository (see revisions 28 to 34 in Figure 2) leads to a decrease in the number of referenced data objects while the number of stored objects remains the same. The additionally stored data objects are referenced by historic snapshots but not in the latest revision of the repository. Since this use case does not require preservation of historic revisions, the yellow area describes the amount of garbage accumulated in the backend storage.

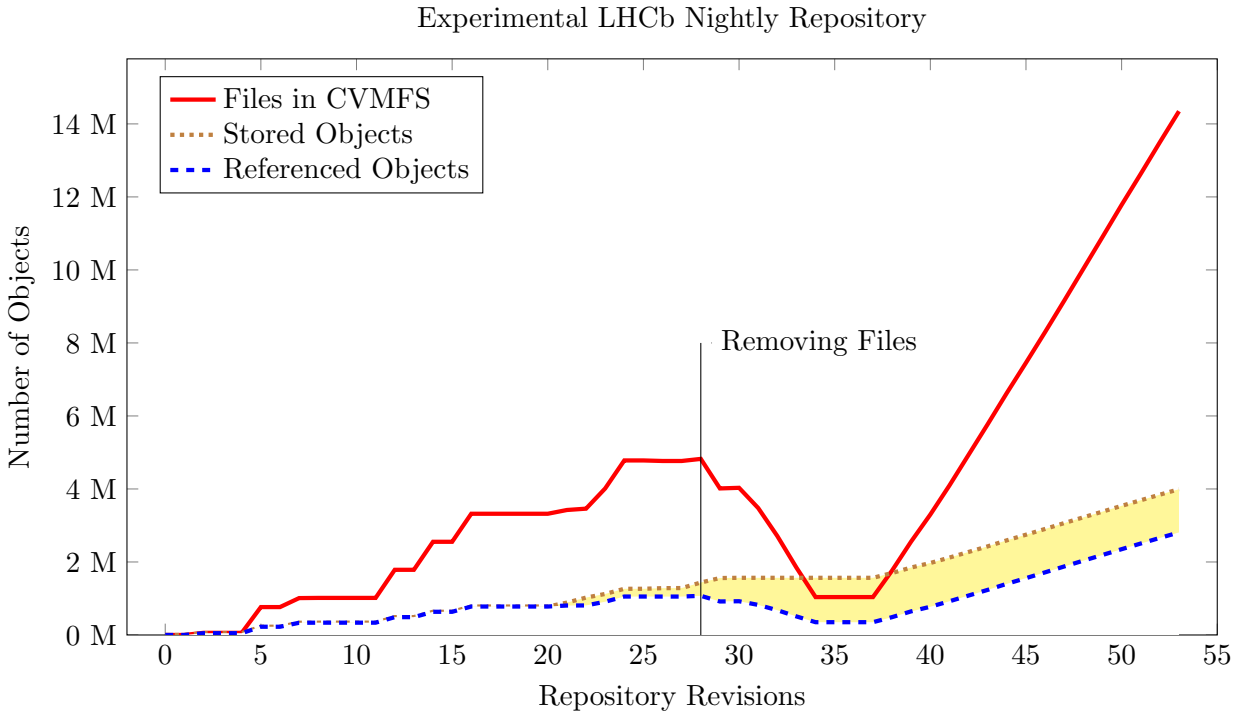

Figure 2. Statistics of an experimental LHCb nightly build repository comparing referenced files (red), referenced data objects (blue) and stored data objects (brown). Area in yellow depicts potential garbage in the repository's backend storage.

We are currently working on a garbage collector[5] for CernVM-FS on revision level that is striving to solve this problem. It will remove historic snapshots from a CernVM-FS repository including all data objects that are not referenced by any conserved snapshots. Due to the architecture of CernVM-FS's internal data structures (i.e. the file system catalogs) as a Merkle Tree [9] and a hash based content-addressable storage for all data objects; a simple mark-and-sweep garbage collector can efficiently detect such garbage objects. We already demonstrated the feasibility by a proof-of-concept implementation of such a garbage collector on the LHCb nightly repository.

3.4. Configuration Repository for Bootstrapping CernVM-FS Clients

With a quickly growing number of CernVM-FS repositories hosted and replicated by various institutions the configuration of clients on both worker nodes and interactive machines becomes

[5] Planned for a release with CernVM-FS 2.1.20

increasingly complex. Therefore we are adding support for a special configuration repository[6] that is envisioned to contain both default settings and public keys for various scenarios and production repositories. This is meant to simplify the distribution of (default) configuration for many repositories through a central entry point.

Site administrators would point their CernVM-FS clients to this configuration repository instead of setting up repository and replication URLs and public keys manually. Hence, mounting a repository becomes a two-stage process: First the configuration repository is mounted to retrieve the default settings for the actual production repository which is mounted as a second step. As a result, the configuration of CernVM-FS will be drastically simplified.

Summary

In this work we outlined use cases for CernVM-FS that have been emerging while the system became widely used along with the challenges they implied for the further development of CernVM-FS. In Section 3 we introduced new features that address these new requirements.

The new server components of CernVM-FS 2.1.x cope with a wider range of use cases while still keeping the focus on scalable software and conditions data distribution. In the next releases we will simplify the configuration of CernVM-FS clients as well as introduce a garbage collection feature allowing repository maintainers to prune unneeded historic repository snapshots.

References

[1] Jakob Blomer, Predrag Buncic, and Thomas Fuhrmann. Cernvm-fs: Delivering scientific software to globally distributed computing resources. In *Proceedings of the First International Workshop on Network-aware Data Management*, NDM '11, pages 49–56, New York, NY, USA, 2011. ACM.

[2] FUSE: Filesystem in Userspace.

[3] Jakob Blomer, Carlos Aguado-Snchez, Predrag Buncic, and Artem Harutyunyan. Distributing lhc application software and conditions databases using the cernvm file system. *Journal of Physics: Conference Series*, 331(4):042003, 2011.

[4] J Blomer, P Buncic, I Charalampidis, A Harutyunyan, D Larsen, , and R Meusel. Status and future perspectives of cernvm-fs. *Journal of Physics: Conference Series*, 396(5):052013, 2012.

[5] LHC Computing Grid Technical Design Report.

[6] C Condurache and I Collier. Cernvm-fs beyond lhc computing. *Journal of Physics: Conference Series*, 513(3):032020, 2014.

[7] Dag Toppe Larsen, Jakob Blomer, Predrag Buncic, Ioannis Charalampidis, and Artem Haratyunyan. Long-term preservation of analysis software environment. *Journal of Physics: Conference Series*, 396(3):032064, 2012.

[8] Kendy Kutzner. *The decentralized file system Igor-FS as an application for overlay-networks.* PhD thesis, Karlsruhe Institute of Technology, 2008.

[9] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO '87, pages 369–378, London, UK, UK, 1988. Springer-Verlag.

[6] `$CVMFS_CONFIG_REPOSITORY` is planned for CernVM-FS 2.1.20