20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013) IOP Publishing
Journal of Physics: Conference Series **513** (2014) 042034 doi:10.1088/1742-6596/513/4/042034

Prototyping a file sharing and synchronization service with Owncloud

Jakub T. Mościcki and Massimo Lamanna
CERN IT/DSS, Geneva 23, Switzerland
E-mail: jakub.moscicki@cern.ch, massimo.lamanna@cern.ch

Abstract. We present the summary of technical evaluation of the Owncloud product as a technology to implement the CERNBOX service - an open source alternative to Dropbox use at CERN. In this paper we highlight the strengths of Owncloud and we also identify core issues which needs to be addressed for large-scale deployment at CERN.

1. Introduction

Users at CERN are attracted by external cloud storage services, such as Dropbox or Google Drive, to share and manage their files. The end-user uptake of these services happens spontaneously and becomes important: on a single working day we have recently counted around 4500 distinct hosts (laptops or desktops) from `cern.ch` network connecting to `dropbox.com` domain[1]. In addition, there are currently ongoing discussions within physics collaborations to actually buy cloud storage from services like Dropbox.

This trend may lead to a data ownership and management problem for the IT: (sensitive) organization data stored on servers outside of corporate control, outside of established policies, outside of enforceable SLAs and in unknown geographical locations. Mitigating this risk also provides a good incentive to rethink how our storage services are delivered to end-users.

Our storage service offering is currently missing a file synchronization and sharing platform which would allow offline work, support mobile devices, seamlessly integrate with major desktop environments and the same functionality as the commercial competitors. As the market of open source projects capable of meeting such requirements begins maturing we started evaluating potential technologies as a part of the CERNBOX project.

The goal of CERNBOX is to provide a viable alternative to Dropbox for CERN users to address current user expectations and keep the data under control. The CERNBOX project will also provide a starting point to investigate scenarios for future evolution of storage interfaces such as integration of "Dropbox" capabilities with physics data storage, file synchronization as a way to providing home-directories in the virtualized IT infrastructure or support for scientific workflows in a system federating the global High Energy Physics community.

2. Open source Dropbox alternatives

To attract the users CERNBOX must provide similar level of platform integration as the commercial competitors. The major platforms in the scope of CERNBOX include Windows, Linux

[1] DNS resolution statistics.

(SLC and other popular flavors), MacOSX, Android and iOS. Setup and access to CERNBOX should be trivial. In our opinion the platform integration and convenience for end-users are the key factor to success.

A solution based on Open Source software is a major requirement for CERNBOX to avoid vendor lock-in and to manage evolution of the service towards future use-cases. The market of open source Dropbox alternatives is rapidly evolving and most of the products are not fully mature yet. Some products such as Syncany[2] still have an experimental status. Others have interesting design but lack in some areas. For example, Sparkleshare[3] uses git under the hood and therefore is not ideal for certain common file types (media, pictures, binary files). Seafile[4] is an young product and provides a good set of features, however, the user interface currently does not provide a similar look-and-feel as Dropbox. Integration of the sync client into desktop environment is in development. Pydio[5] (formerly known as AjaXplorer) was developed primarily as a web-based sharing platform. It provides integration with popular content management systems such as Drupal and provides mobile clients. The desktop sync client, a java-based application with rsync component, is in development.

Owncloud[6] has been developed as a drop-in Dropbox replacement solution for corporate environments with rich set of features and a modern interface. It provides a cross-platform web/mobile/desktop work environment and supports modern desktop-integration concepts such as drag-and-drop, system-tray integration and system notifications for sync clients. Access control is supported for sharing of files and folders[7]. Files may also be anonymously shared with external users by the means of hashed URLs with optional expiry date and password protection. File versioning and recycle bin functionality are supported core features. Owncloud also provides an open application-framework which may be used to extend functionality and it features the concept of application store. We found that the general vision of the product matches well what we need for CERNBOX.

Owncloud has recently been evaluated and deployed at several universities and research institutes in Europe (e.g. Technical University of Berlin [2], Max Planck Institute for Neurological Research). ETH Zurich runs a public beta service based on Owncloud Enterprise Edition and commercial filer storage. The system is currently used by 1900 users with a goal to be a sharing platform for students and staff.

3. Technical overview of Owncloud

Owncloud, currently at version 5, in spite of misleading numbering scheme is a young and rapidly evolving product. Owncloud 5.0.0 Community Edition was released in March 2013 and it was initially unstable and buggy. However, in the course of 2013 there was a succession of releases in the Owncloud 5 series and the quality of the software has been rapidly increasing.

Owncloud server framework is implemented in PHP as a part of a typical LAMP[8] architecture. Desktop sync clients are implemented in C++ with Qt[9] GUI and libneon as a WebDAV client[10]. The server and client source codes, including the Android client application, are hosted on

[2] `www.syncany.org`
[3] `www.sparkleshare.org`
[4] `www.seafile.com`
[5] `http://pyd.io`
[6] `owncloud.org`
[7] ACLs (Access Control Lists) are not propagated on the file-system level but are enforced when accessing files via the owncloud server.
[8] Open-source stack based on Linux, Apache, mysql and php.
[9] `qt-project.org`
[10] `www.webdav.org/neon`

github[11], with an exception of iOS client which is closed-source.

We deployed Owncloud server (Owncloud Community Edition) on the SLC6 platform with Apache web-server and mysql database backend. Configuring HTTP access over SSL was straightforward. Owncloud server also integrates smoothly with CERN's Active Directory LDAP server over SSL-encrypted connection. Integration with CERN Single Sign-On infrastructure requires Shibboleth authentication plugin which is part of Owncloud Enterprise Edition and was not tested.

3.1. File transport and synchronization protocol

The file transport is based on HTTP/WebDAV. Owncloud WebDAV server is based on a PHP SabreDAV[12] module. The server's compliance to WebDAV standard (RFC2413) is verified as a part of Owncloud release process.

Standard command-line clients, such as `curl` or `cadaver`, may be used to download and upload files. Popular desktop environments, such as MacOSX/Finder, allow to browse the content of the WebDAV server directly. It is possible to mount WebDAV as a filesystem, however, we found that fuse-mounted WebDAV access is not stable on all platforms. It is also possible to use Owncloud WebDAV server as an endpoint for 3rd party applications, such as Cyberduck[13] or Foldersync[14].

File upload size in unlimited because both Owncloud server and sync client support chunk upload via an Owncloud-specific attribute extension in the WebDAV request header. The default chunk size is 10 MB.

Desktop sync client watches local folders for changes and uploads files to the server as needed. Currently there are not attempts made to optimize the transfer by sending only the file differences. The client keeps track of the synced files in a local state database implemented as a hidden sqlite file in the local sync folder. File versions are identified by an ETag[15] property which is a randomly generated unique identifier. When a new version of a file on the server is created a new ETag is generated and stored in the server's database. If a client's ETag is not matching the server's ETag then the client downloads the file. Conflicts are left for manual resolution by the user (locally modified file is renamed to indicate a conflict).

Owncloud supports Unicode filenames but restricts the use of special characters, such as `[":;()%?*]`, to the least-common denominator allowed by supported platforms. Symbolic links, hard links and special files are not supported.

3.2. File storage layout

Files are stored on the server's filesystem (primary storage) in a transparent directory tree which reflects the client folder structure. We consider this simple storage layout to be an extremely important property of the system. It is not only easy to understand, but it also allows to access data directly on storage if needed.

The implementation of file uploads on the server is atomic: a file is first stored under a temporary name in the destination folder and then renamed (this assumes that the underlying rename operation on the server's filesystem is atomic). The file entry in the server's database is updated and a new ETag generated. SabreDav also allows to dynamically map a file path which is part of URL's logical namespace into a physical resource on the server. It is hence possible (and supported by Owncloud) to connect an external storage endpoints (secondary storage) such as SFTP, S3 or WebDAV server, or Dropbox account or yet another Owncloud instance.

[11] `http://github.com/owncloud`
[12] `code.google.com/p/sabredav`
[13] `cyberduck.ch`
[14] `www.tacit.dk/foldersync`
[15] ETag is a standard HTTP header attribute for cache control.

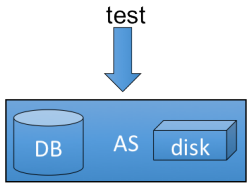

Figure 1. Baseline server configuration: local application server (Apache), local mysql database, local disk storage. Server hardware: 32-core AMD Opteron processors with 64 GB RAM, 1Gb Ethernet interface and standard SATA hard-drive.

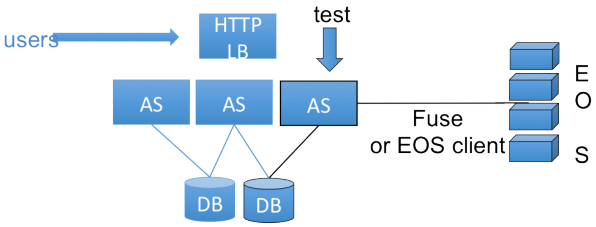

Figure 2. Intended final configuration for a beta-service at CERN: HTTP load-balancer (LB), a cluster of Apache application servers (AS), a mysql database cluster (DB) and backend storage based on EOS.

Secondary storage endpoints are visible as parts of user's directory tree and are accessible via the file path URL in a way similar to filesystem mountpoints.

3.3. Testing of Owncloud

For the tests we have used the latest available version of Owncloud Community Edition (5.0.11) without any special enterprise features. We deployed an automatic testing framework with 50 OpenStack VMs running sync clients continuously and a fast LSF batch queue with up to 1000 worker nodes to create activity bursts. As of the time of writing this report scalability- and stress tests are still work in progress.

Fig.1 shows the baseline server configuration used for testing with local filesystem and local database. Fig.2 shows intended final configuration with a storage backend based on EOS[1] - the low-latency physics data analysis storage system at CERN.

We have conducted a set of core logics tests to verify the consistency of behavior in difficult synchronization scenarios such as file conflicts or conflicting operations (create/move/delete). This includes testing of unusual directory layouts and various boundary conditions.

Additionally, in a series of field testing experiments, we have checked how system behaves in "realistic" situations, such as, a user continuously working on a small subset of files in his home directory. We have used randomly generated files with sizes corresponding to the size distribution found in CERN AFS home directories. All files have been checksummed and we continuously verify their integrity in an automated way.

3.4. Data integrity

In the field testing we have not observed corrupted files and the system proved to be reliable. However it was possible to construct a test-case triggering a race-condition which resulted in a corrupted file store on the server. The bug occurred during chunk-upload of large files in parallel.

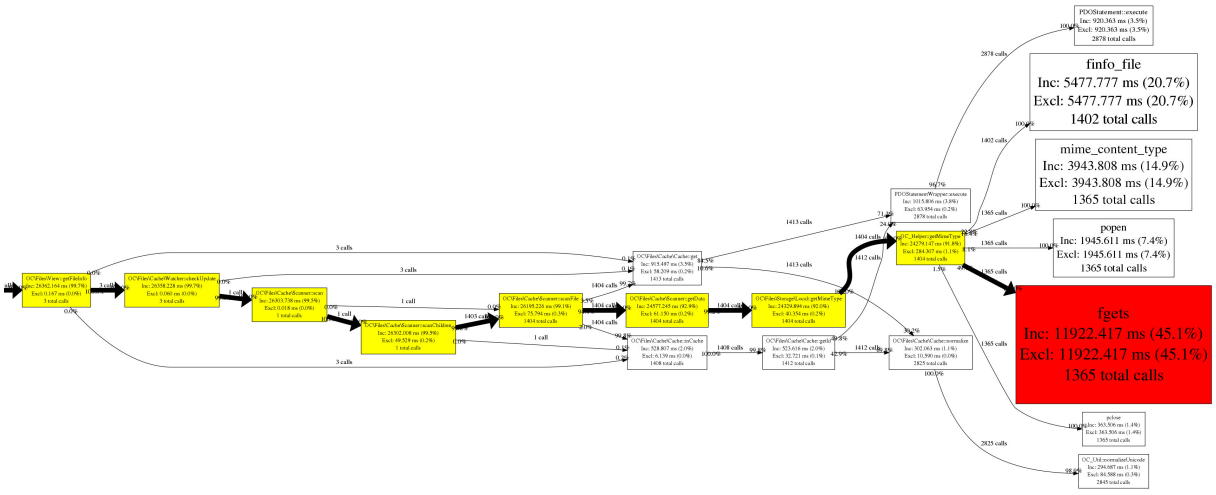

Figure 3. A critical path of the server-side call-graph of a metadata request (`PROPFIND`) issued by the sync client after the file has been uploaded and added to a directory containing about 1400 files. Scan of the entire directory is performed which results in a large number of metadata queries on the storage system. Aggressive MIME-type determination policy results in reading the headers of 1400 files (in this case the MIME-type could not be guessed from file's extension).

We also tested file sharing. Currently a problem with updating of ETags in the database cache on the server prevents the file changes in shared folders to be propagated between the sync clients unless the shared folder is at the top level in user's directory tree.

Occasionally spurious conflict files are created when for application with complex data writing patterns to a file. For example, this happens for VirtualBox[16] images stored in a sync folder. In the past it was also observed this for MS Office files however it was fixed in recent Owncloud releases.

3.5. Scalability

Owncloud optimizes detection of file changes on the server in the following way: if there are no changes on the server then the ETag of the top level directory remains unchanged. The sync client needs one request to the server to verify this condition. If a file is changed then new ETags are generated for all directories which are on the access path to that file. The sync client then queries only the parts of the directory tree which have changed and eventually fetches the changed file. During our tests this mechanism has been proven to work as designed. The effective cost of detecting the change is proportional to $O(logN)$ where N is the number of files in the directory tree.

However, current implementation does not scale well as the number of files in a single directory increases. The cost of adding new files is proportional to $O(N)$ where N is the number of files in the directory. This was identified as a bug which triggers full directory scan to determine the mime-types of the files. Fig. 3 shows an example of the PHP function call-graph involved on the server when a new file is uploaded to a directory.

Scalability of Owncloud server is impacted by the complexity of Owncloud framework for core operations: there may be up to 5000 PHP function calls per individual file-related request. Enabling accelerators such as APC speeds up execution by 20%. It is also recommended to run a modern PHP interpreter version but anyway the performance benefits are quite limited (we

[16] `www.virtualbox.org`

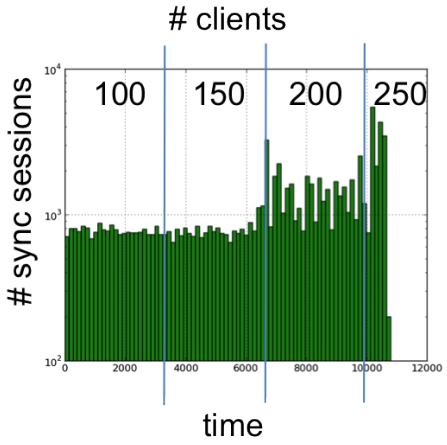

Figure 4. The number of sync sessions handled by a baseline server in a function of the number of clients. Binned data (120 second intervals).

observed 15% gain from PHP version 5.3.3 to 5.4.16). Further tuning of LAMP components is required and may include optimizing database access (enabling of row-locking database engine such as MySQL/Innodb, tuning indices on file cache table), using caching components such as memcached and web-server tuning.

Scalability of current implementation of Owncloud may be constrained because of the load generated on the database and storage resources by Owncloud application server: the number of database base operations and metadata queries to the storage should be substantially reduced.

Fig. 4 shows the number of sync sessions handled by Owncloud server in the baseline configuration. The test was performed for 100 test accounts and a variable number of clients syncing folders in parallel. Each client syncs 100 small files in a single directory. At 200 clients the server was reaching the CPU limit. The irregular number of sessions in 200-250 range comes from client requests being dropped by the client. This demonstrates that the server is correctly handling the overload situation by refusing service to excessive clients, however, the server also requires tuning to support a larger number of parallel clients with smaller CPU footprint.

3.6. Performance

Streaming performance for large files measured with HTTP GET/PUT methods is satisfactory (Table 1). However, the chunk upload by the sync clients scales poorly as the number of chunks increases (upload time is proportional to $O(N)$ where N is the number of chunks). This has been identified as a bug in the chunk-upload implementation on the server.

Table 1. Owncloud WebDAV streaming performance (HTTP PUT/GET) for a 400 MB file.

MB/s	upload	download
http	40	100
https	25	60

The average sync rates of files is 1-5 Hz and does not depend on the file sizes if below the chunk size limit. Low sync rate stem from large framework overheads. The effect is amplified as the client issues three WebDAV requests per one synced file. While this may be sufficient for classic use-case of low-traffic syncing of desktop documents it may become problematic for high-performance syncing of large data volumes. In our opinion there is room for substantial improvements in the server's implementation.

We also observed that the sync client tends to traverse directories out-of-order. This may result in inefficient handling of nested directories.

The interactions between various components and applications in the server framework need a careful review, since the activity generated from the web interface may occasionally invalidate ETags and cause re-download of all user files by the sync client.

4. Summary

Building a sharing and synchronization platform is a challenging task as it requires to support semantics of different operating- and file-systems. Owncloud is a young and rapidly evolving product with a very interesting vision and rich set of features. In addition the roadmap of Owncloud is extremely interesting. For example, Owncloud 6 comes with functionality for collaborative editing of documents similar to Google Docs.

To fully exploit the potential of Owncloud for large-scale deployments such as at CERN it is necessary to address the critical issues identified during our tests. These issues have been reported[17] and are currently being addressed by the development team. We acknowledge a good feedback from the Owncloud development team and rapid introduction of bugfixes. In our opinion it is critical to improve the robustness and efficiency of the core sync layer of Owncloud in the future releases.

We think that automated testing framework of the core functionality is a necessary tool for sites considering a large-scale deployment of Owncloud, and ultimately, may also be an interesting tool for the Owncloud development team. Therefore we plan to share our testing framework with the community[18].

At the present stage, we are convinced that CERNBOX could become an important component of our service offer in the area of home directories for CERN users. We plan to deploy a pilot service based on Owncloud for further evalutation and user feedback.

References

[1] Andreas J Peters and Lukasz Janyst, Exabyte Scale Storage at CERN, Journal of Physics: Conference Series, Volume 331, Number 5

[2] T.Hildmann, ownCloud at TUB (in German), `http://www.tubit.tu-berlin.de/fileadmin/a40000000/tubIT/veranstaltungen/ownCloud_TUB_Aug_2013.pdf`

[17] `http://github.com/owncloud`

[18] `http://github.com/opensmashbox`