ownCloud®

Your Cloud, Your Data, Your Way

The File Sync Algorithm of the ownCloud Desktop Clients

Workshop on Cloud Services for File Synchronisation and Sharing

November 17-18, 2014, CERN

Klaas Freitag

ownCloud Client Developer

freitag@owncloud.com

Klaas Freitag

- Open source developer by passion
- Started the ownCloud sync client in late 2011
- Changed from Linux distributor SUSE to the ownCloud company in February 2012

Topics

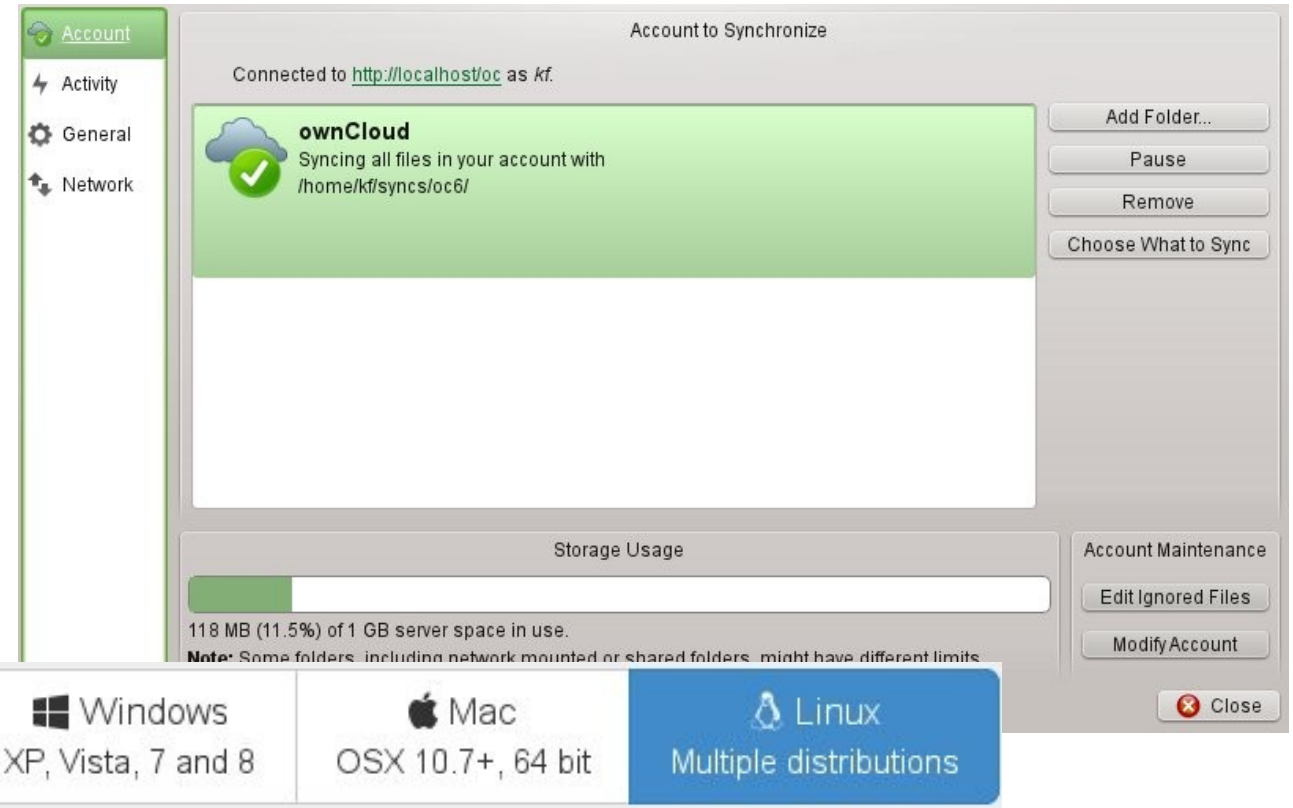
- Syncing – General Thoughts
- Three phases of ownCloud Sync Protocol
- Some Challenges and how ownCloud deals with it
- Possible Future Improvements
- Discussion

What is Syncing?

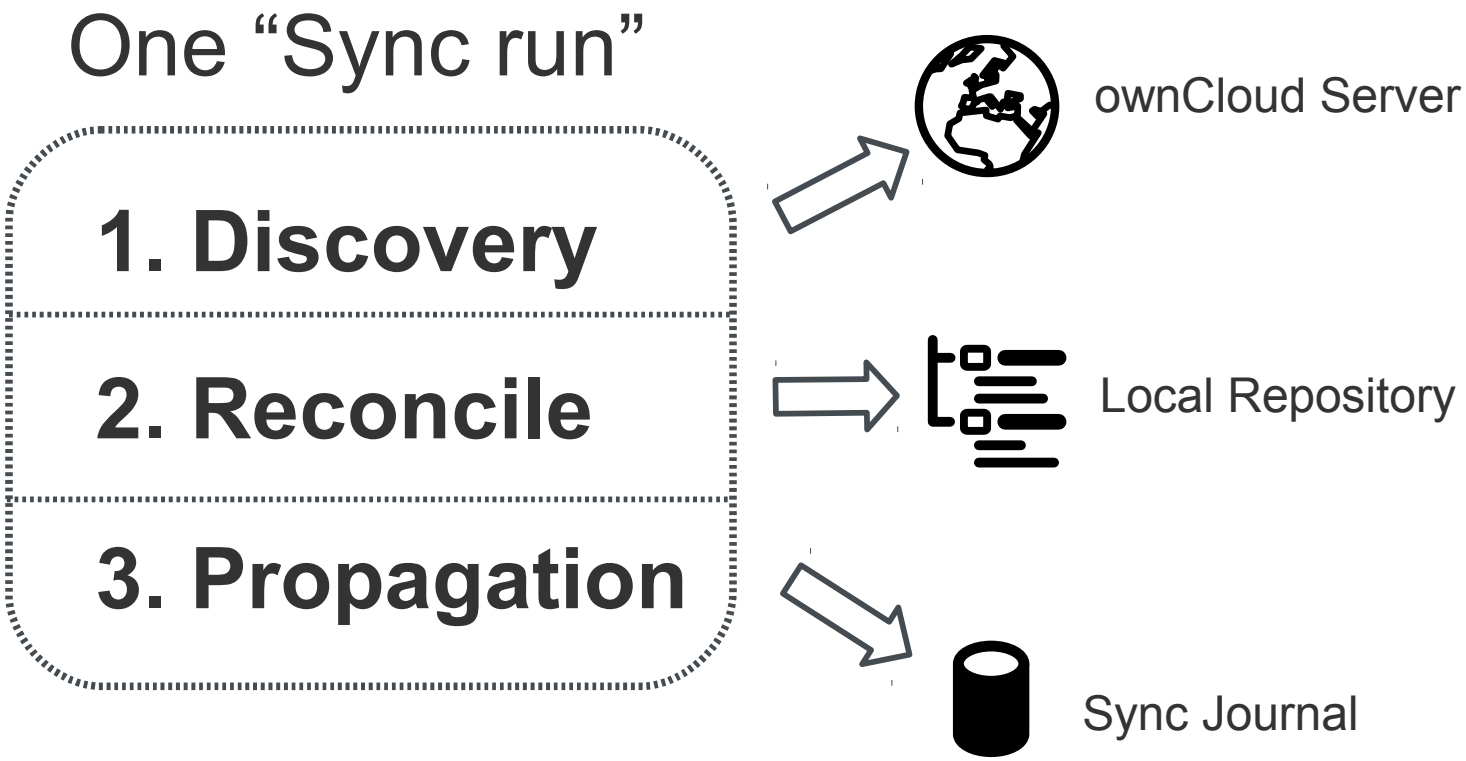
- two collections of data is kept in exactly the same state: data and metadata
- Changes are triggered equally by both sides
- No time constraints
- Correctness: Never data loss or corruption
- Ability to deal with big data sets
- No user interaction

OwnCloud Sync Clients

- Open source software licensed under GPL
- Based on open standards

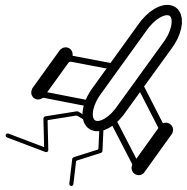


Three Phases of ownCloud Syncing



Three Phases of ownCloud Syncing

When is a sync run started:



At the program start

Regularly after a certain time

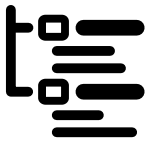
Local repository: On notification

… at some other events

Three Phases of ownCloud Syncing

Discovery phase: Collecting data

File metadata: Size, modification time, filename (utf-8), phash



Local

Inode

hardlink count

file mode

ownCloud

File id

Remote permissions

Sharing state

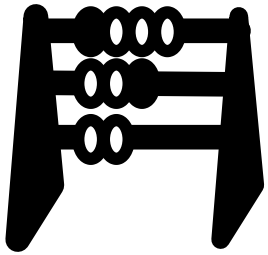
ETag

Result: Two trees of file meta data in memory.

Reconcile phase: Decision taking

Iterate over both file trees from discovery phase.

Decide what to do with each file based on the collected meta data.



Result: A working list for propagation.

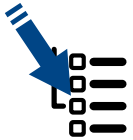
Reconcile phase: Examples

Local file changed:

Modification time **of local file is different to the one in the sync journal → upload if remote is unchanged**

Remote file changed:

Etag **is different as the one saved in the sync journal → download if local is unchanged**



Local file renamed:

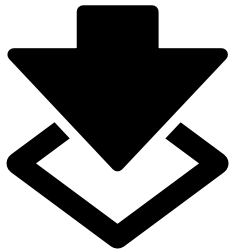
A local file is new. Another file with the same *inode* exists in the sync journal → remote rename

Remote file renamed:

A remote file is new. A file with different name but same file id is listed in sync journal → local rename.

Propagator phase: Execute Jobs

- Execute the action job list per directory.
- Local repository: Platform dependent file system operations
- Remote repository: HTTP/WebDAV operations like PUT, GET, MKCOL, MOVE, PROPSET
- Network operations run parallel HTTP requests
- Sync journal maintenance, error handling, progress...



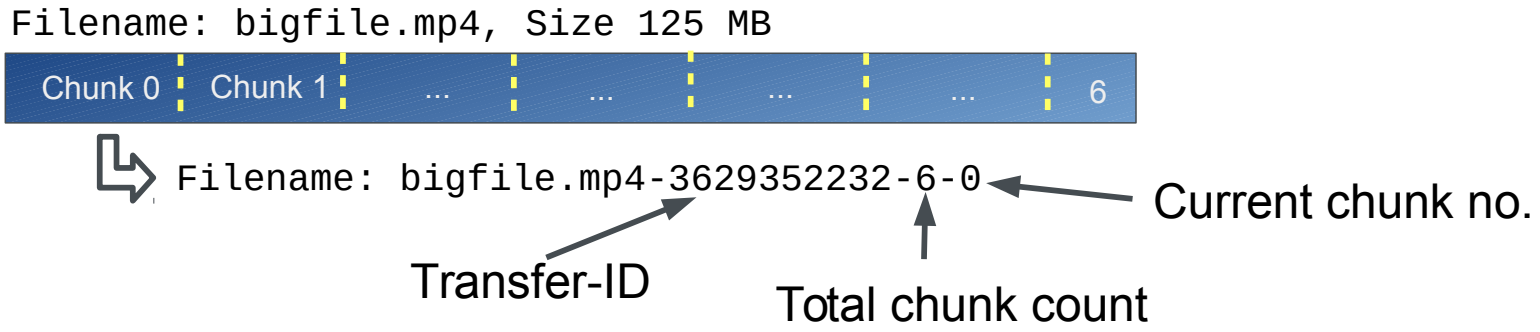
Propagator phase: Big file chunking

Big files can not be uploaded by PUT

- File is split in chunks of equal size (except the last)
- Upload to server with a special name pattern
- Custom header to indicate the chunking
- After all chunks were received, the server assembles the original file
- Allows upload resume

Three Phases of ownCloud Syncing

Propagator phase: Big file chunking



Example PUT request

```
PUT /oc/remote.php/webdav/bigfile.mp4-3629352232-6-0 HTTP/1.1
Content-Type: application/octet-stream
If-Match: "545b88ef8fa22"
OC-Chunk-Size: 20971520
OC-Chunked: 1
OC-Total-Length: 48482110
X-OC-Mtime: 1415965487
Authorization: Basic aXXXXXXXXX=
...
```

Challenges

WebDAV is perfect

- Fundamental part of ownCloud
- It's a very well known protocol
- Easy to extend
- Easy to route through firewalls

...but there are challenges:

- Performance: Everything is a single request
- No batch requests by default
- PUT requests: limited in size, no ranges
- Missing Functionality

ownCloud WebDAV Extensions

Extend functionality and efficiency

- *File ids* to detect remote renames
- *ETag* change propagation through the directory tree on server side
- Additional HTTP headers to transmit meta data to save requests
- Recursive Propfind

Asynchronous PUT

- After upload of all chunks, server returns a URL to poll if the assembling of the file has finished on the server.

Discovery Phase Improvements

- Keep the file tree in memory, only apply changes

Delta Sync – most wanted!

- User often over-estimate the effect
  - Compressed files do hardly delta sync
  - Hash sums needed
- Maybe mime-type based approach?
  - Just send the changed metadata
  - Decompress file contents and sync differences only

Thanks a lot!