# CernVM-FS Tarball Ingestion

Simone Mosciatti
11 / 06 / 2018
EP-SFT weekly meeting

# TOC

- Goals
- Preview
- Background
  - Docker images layers
  - Tarball
- General workflow
- Changes introduced in the codebase
- Status and future work

# Goals

- Speed
  - Avoid materializing the tar content on the union file system followed by reading it back
- Publish files that otherwise are hard to publish
  - Magic file of the union file system
  - Files that are not owned by repository owner
  - Specials files (pipe, sockets, etc…)
- Use cases
  - Container layer tarballs
  - HEP application software as tarballs
  - (Technical basis for portals)

# Background on CVMFS ingestion

- After `cvmfs_server transaction` we start recording and changes to the repo
- On `cvmfs_server publish` we traverse the repository and we write the changes into the backend storage & catalog
- Directories & links
  - recorded in the catalog
- Files
  - Recorded in the catalog
  - Upload on storage

# Add a container layer tarball to CVMFS

```
$ cvms_server transaction repo.cern.ch

$ tar xf foo.tar --owner=$(id -u) --group=$(id -u) --no-xattrs
--exclude="*dev/*" #more complex than just untar

$ cvmfs_server publish repo.cern.ch
```

1. Write the content of the tarball to disk
    a. Copy the content on /var/spool/cvmfs/...
2. Publish it on CVMFS
    a. Read it back in from /var/spool/cvmfs/…
    b. process it
    c. Write it to /srv/cvmfs/… (or to S3)

# New provided interface

1. Single command, no need of transaction + publish
2. No double write
3. Possible to read stream from STDIN

```
# Extract foo.tar inside repo.cern.ch/foo/

cvmfs_server ingest --tar_file foo.tar --base_dir foo/ repo.cern.ch
```

# New provided interface

1. Single command, no need of transaction + publish
2. No double write
3. Possible to read stream from STDIN

```
# Extract foo.tar inside repo.cern.ch/foo/

cvmfs_server ingest --tar_file foo.tar --base_dir foo/ repo.cern.ch

# Like above but reading from STDIN

gunzip -c foo.tar.gz | cvmfs_server ingest --tar_file - --base_dir foo/
repo.cern.ch
```

# New provided interface

1. Single command, no need of transaction + publish
2. No double write
3. Possible to read stream from STDIN

```
# Extract foo.tar inside repo.cern.ch/foo/

cvmfs_server ingest --tar_file foo.tar --base_dir foo/ repo.cern.ch

# Like above but reading from STDIN

gunzip -c foo.tar.gz | cvmfs_server ingest --tar_file - --base_dir foo/
repo.cern.ch

# Delete foo (a file or a directory) from repo.cern.ch

cvmfs_server ingest --delete foo repo.cern.ch
```

# Under the hood

# Background on Docker Images

- Docker layers are simple tarfiles
- docker “composes” the containers root filesystems from layers using an union filesystem
- Layers are read-only, images can share layers
- How we removes files from docker images?
  - On the top layer we overwrite the file with a whiteout one.

# Background on Tarball Structure

- Sequences of blocks (512 bytes)

# Background on Tarball Structure

- Sequences of blocks (512 bytes)
- Each entity is 1 block of header + `n` blocks of data

# Background on Tarball Structure

- Sequences of blocks (512 bytes)
- Each entity is 1 block of header + `n` blocks of data

Encoding of linux stat structure:

- type
- name
- size
- linkcount
- etc...

# Background on Tarball Structure

- Sequences of blocks (512 bytes)
- Each entity is 1 block of header + `n` blocks of data
- Folders, links, etc, are only header
  - Files contains several blocks

Encoding of linux stat structure:

- type
- name
- size
- linkcount
- etc...

# Background on Tarball Structure

- Sequences of blocks (512 bytes)
- Each entity is 1 block of header + `n` blocks of data
- Folders, links, etc, are only header
  - Files contains several blocks
- 2 empty blocks works as EOF

Encoding of linux stat structure:

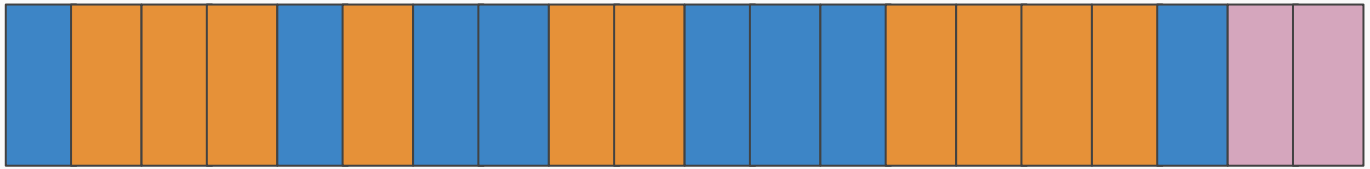
- type
- name
- size
- linkcount
- etc...

# Background on Tarball Structure

- Sequences of blocks (512 bytes)
- Each entity is 1 block of header + `n` blocks of data
- Folders, links, etc, are only header
  - Files contains several blocks
- 2 empty blocks works as EOF

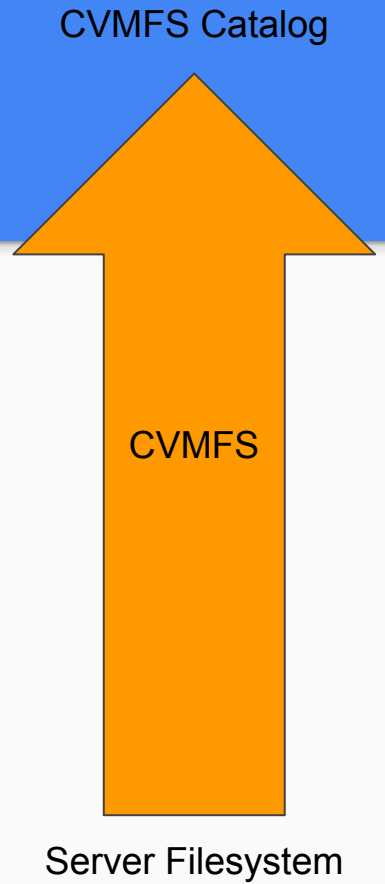
Encoding of linux stat structure:

- type
- name
- size
- linkcount
- etc...



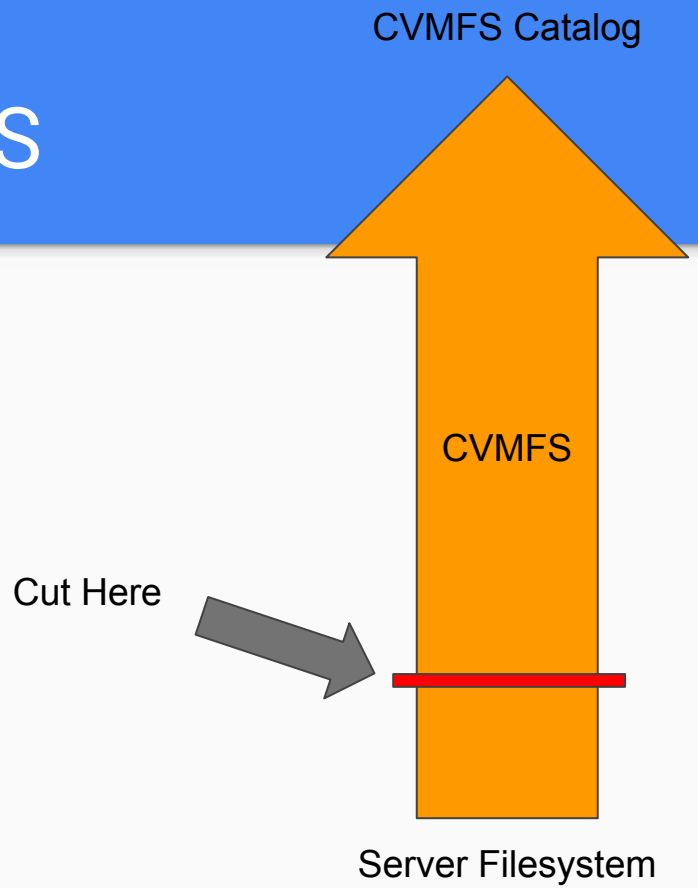
# Notable changes in CVMFS

- Abstraction for files to be published
  - Before we were just passing `std::string path`
  - Opening the path
  - Spooling the file by reading it
    - Send the file to the storage
    - Add the file to the catalog



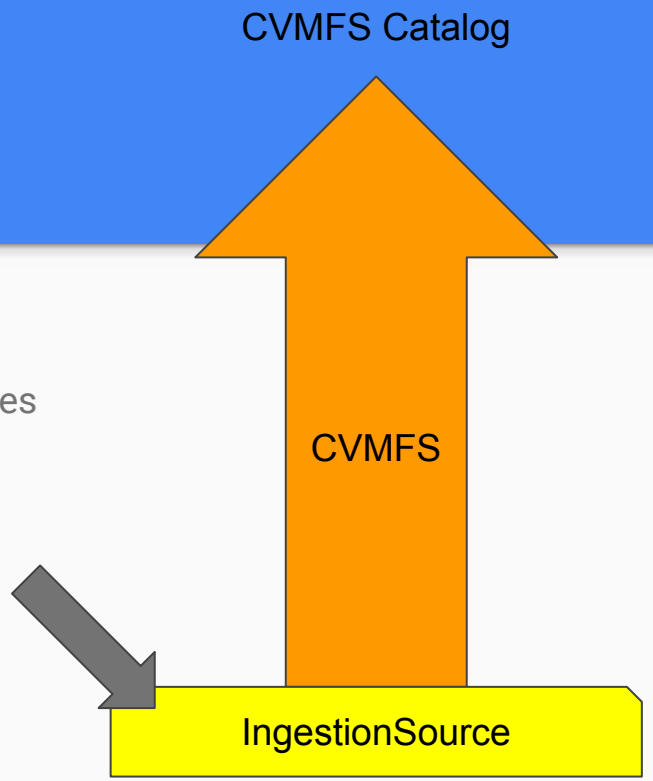
# Notable changes in CVMFS

- Abstraction for files to be published
  - Before we were just passing `std::string path`
  - Opening the path
  - Spooling the file by reading it
    - Send the file to the storage
    - Add the file to the catalog



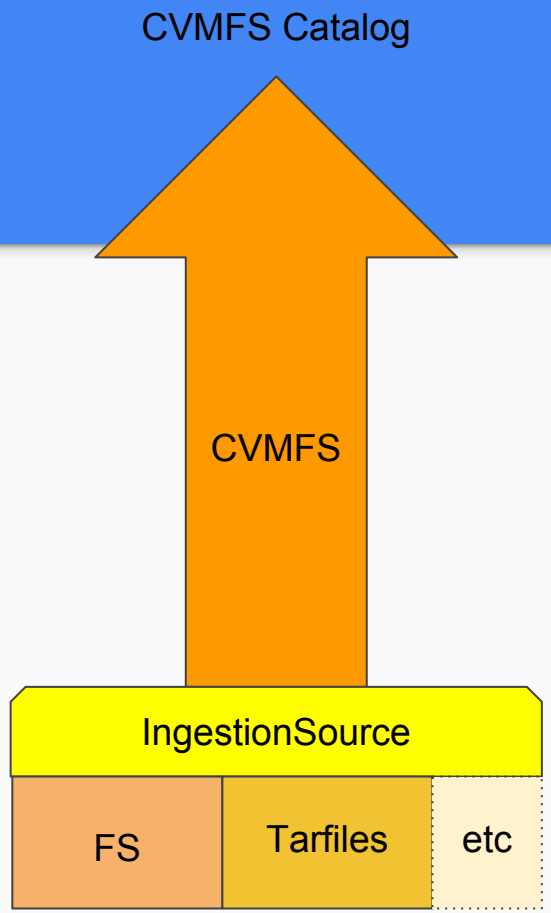
# Notable changes in CVMFS

- Introducing IngestionSource
  - std::string GetPath() // still needed for the catalog updates
  - bool Open()
  - ssize_t Read()
  - bool Close()
  - bool GetSize(uint64_t* size)



# Notable changes in CVMFS

- Implement IngestionSource
  - Filesystem
  - Tarfiles
  - Other possibilities
    - RPMs
    - Object packs (publishing through the new gateway)
    - Add file via network (?)



# Difficulties in managing tarfiles

- “Stream” the tarball from the beginning to the end
  - No “rewind” of the tarball
- Provide support for reading the files from STDIN
  - Can’t read it twice
- Tarball too big to fit in memory
- Out of order expansion of the filesystem tree
  - Not always from the root to the leaves
  - Possible to find first the file `` `/foo/bar.txt` `` than the directory `` `/foo` ``
- Complex to integrate into the multithreading CVMFS spooling framework

# Multithreading

- CVMFS use multithreading for spooling the files
- Files to be published are pushed into a queue
- A processing pipeline: pool of threads pop the files from the queue and processes them (chunk, compress, hash and upload)
- On the FS we can read multiple files at the same time
  - The OS will multiplex for us
- On tarballs this is impossible, we can read only one file at the time
  - Hold a lock waiting until all bytes are pushed into the pipeline
  - Release the lock and move to the next entry in the tarfile

# Status and timeline

- Already merged PR for tarballs containing only
  - Regular file
  - Directory
- Completing the PR for
  - Softlinks
  - Hardlinks
  - Character devices
  - Etc…
- At the same time I am completing the work on the docker2cvmfs plugin and conversion utility that is uncovering interesting corner cases

# Results

- We are able to ingest files that otherwise are hard to ingest
- Technical center piece for several follow up developments
  - Publishing of containers images
  - Foundations of portals (S3 endpoint into a repository)
  - Speed up of the gateway publishing
  - Possible to extend with new package formats (RPM, DEB)
- We record an improvement in performance
  - ~9% on fast, SSD equipped machines
    - ~4% Total time (userspace + kernel) on slow machines
    - ~25% Wall time on slow machines
      - Result skewed by CPU time allocated to untar in Openstack ~50%

# A small preview of the whole workflow!

Small preview video