Zstd:
A new compression algorithm for CVMFS

Laura Promberger

CernVM Workshop 2024

Performance Analysis: Finding Bottlenecks

"We found that, by parallelizing the data decompression, we can improve performance on multiple-processes / multiple-data scenarios" (Aug 22)

https://indico.cern.ch/event/1180962/contributions/4960898

What is zstd?

- Zstandard (zstd) developed by Facebook/Meta
- Exists since 2016
- BSD license

- Offers lots of compression levels to trade-off compression ratio vs speed vs memory footprint
  - Compression ratio as zlib default but significantly faster
  - Can achieve high compression ratios as lzma

What is zstd? II

Compressor name	Ratio	Compression		Decompress.	
zstd 1.5.6 -1	2.887	510 MB/s	~5x faster	1580 MB/s	~4x faster
zlib 1.2.11 -1	2.743	95 MB/s		400 MB/s	
brotli 1.0.9 -0	2.702	395 MB/s		430 MB/s	
zstd 1.5.6 --fast=1	2.437	545 MB/s		1890 MB/s	
zstd 1.5.6 --fast=3	2.239	650 MB/s		2000 MB/s	
quicklz 1.5.0 -1	2.238	525 MB/s		750 MB/s	
lzo1x 2.10 -1	2.106	650 MB/s		825 MB/s	
lz4 1.9.4	2.101	700 MB/s		4000 MB/s	
lzf 3.6 -1	2.077	420 MB/s		830 MB/s	
snappy 1.1.9	2.073	530 MB/s		1660 MB/s	

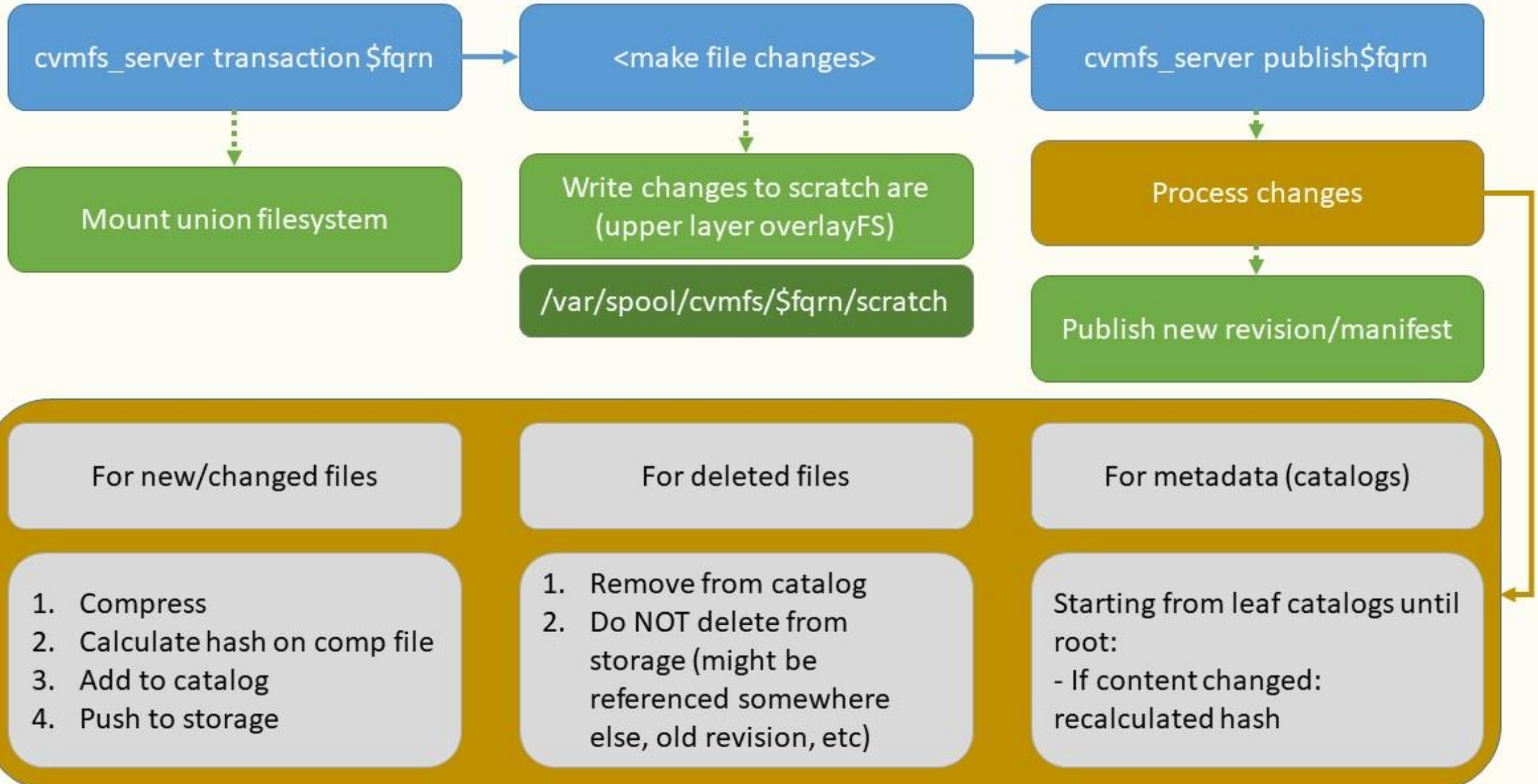
https://github.com/facebook/zstd#benchmarks (Sept 14, 2024)

Handling of files and metadata in CVMFS

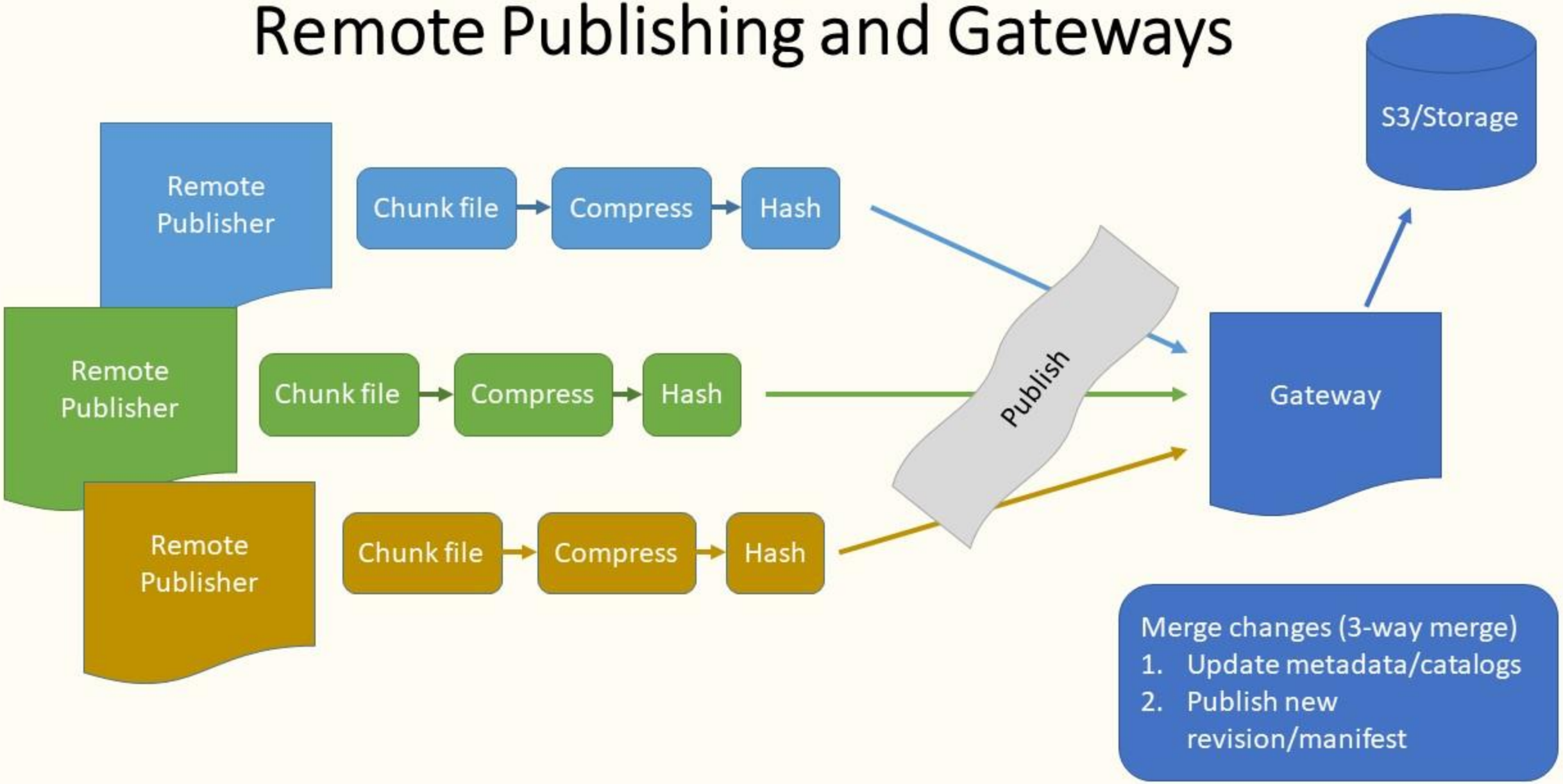
- Content-Addressable Storage (CAS) stored in Merkle Tree
  - Similar structure to git
- Root catalog as entry-point to discover and access the entire repository

- Files
  - Chunked
  - Compressed
  - Hashed --> CAS
- Metadata
  - “Catalogs”
  - Contains hashes to all reachable files
  - Stored in sqlite
- “Magic” extended attributes
  - Some stored, some calculated on the fly

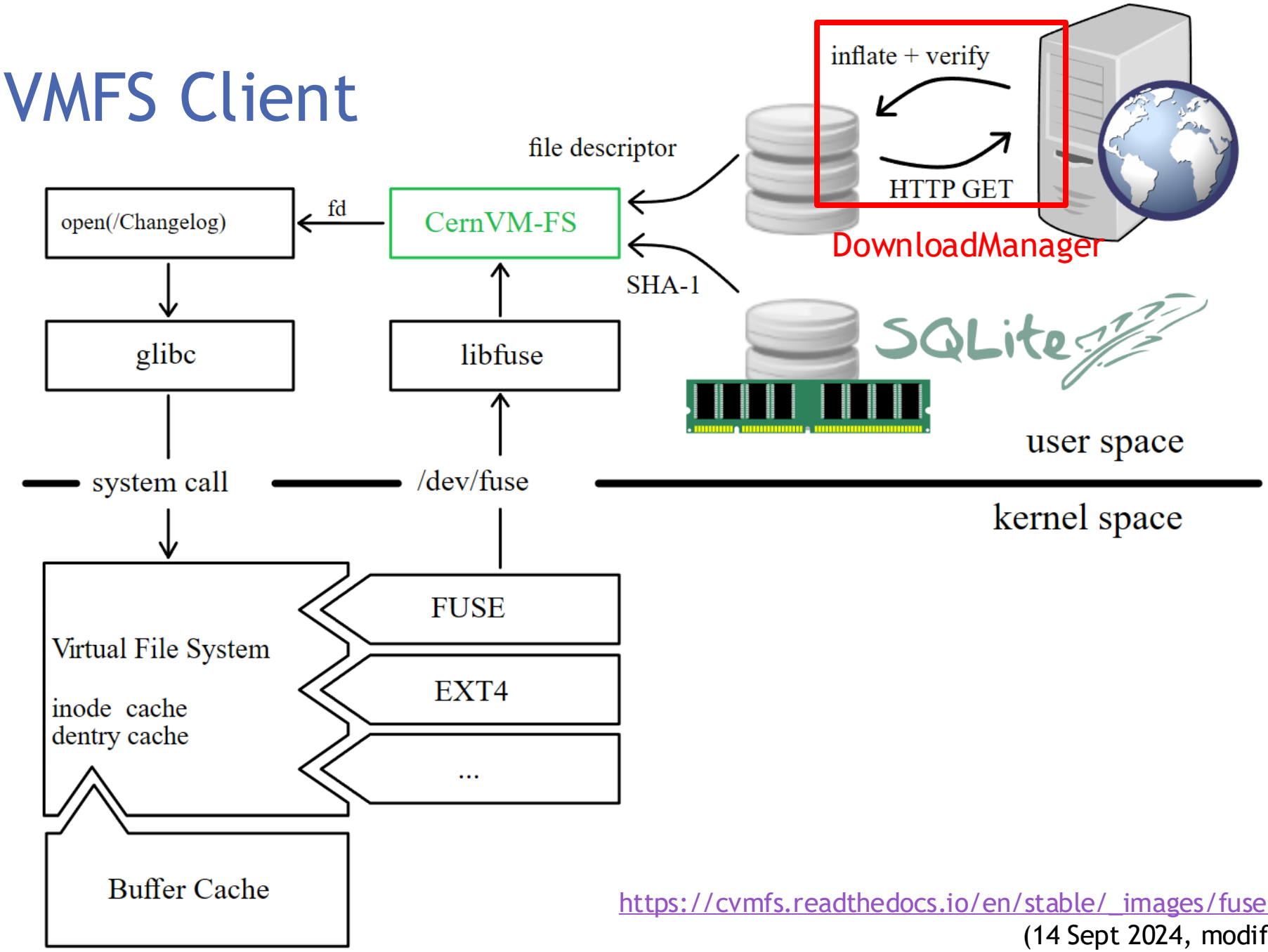
Publishing Process CVMFS



Remote Publishing and Gateways

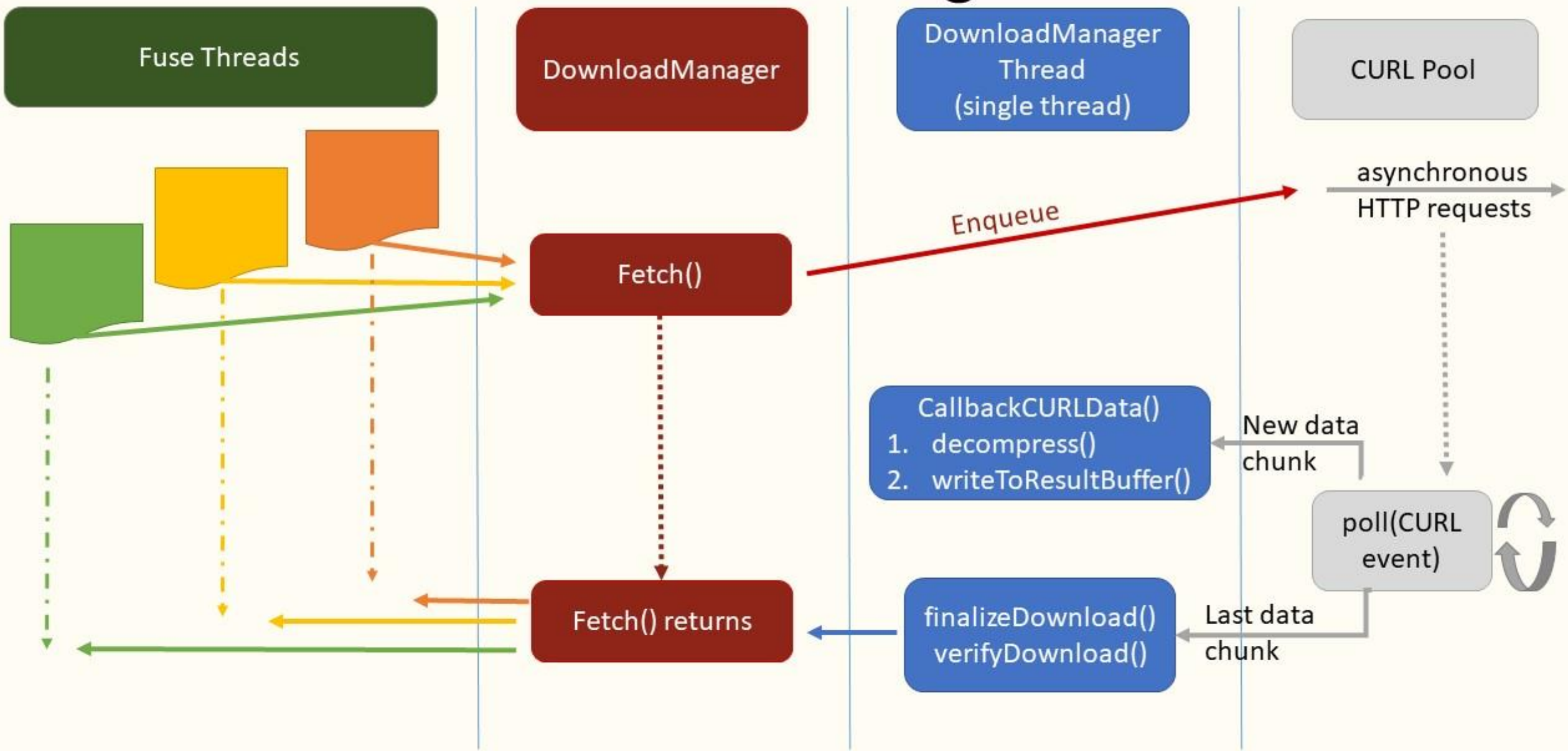


CVMFS Client



https://cvmfs.readthedocs.io/en/stable/_images/fuse.svg
(14 Sept 2024, modified)

DownloadManager



Was has happened since then….

1) Performance improvements for downloads

- “parallel decomp” for FUSE threads
  - Needs sophisticated communication via queues
  - Ongoing performance engineering
    - Many core machines with heavy download load significantly benefit
    - Still open to find good default config for small machines

Was has happened since then….

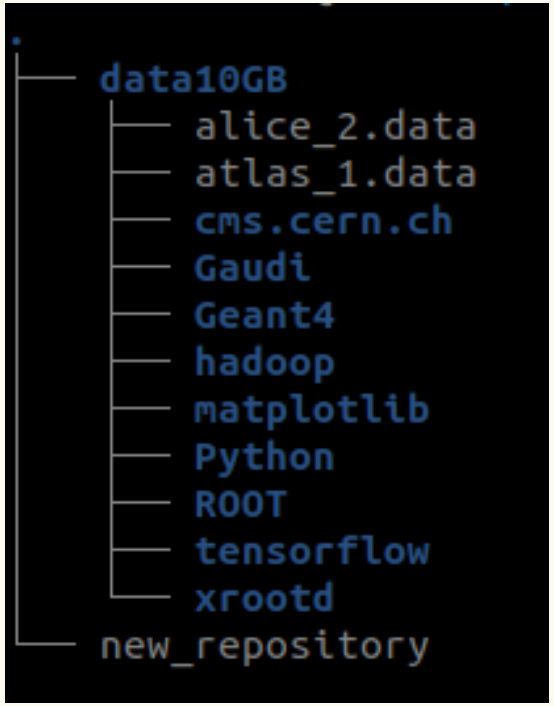
2) Zstd as new compression algorithm

- Introducing Compressor/Decompressor class
  - Significant refactoring
  - Works for existing algorithms: zlib, copy
- Pilot PR
  - Hard-coded replacement of all zlib with zstd

Pilot PR: Setup

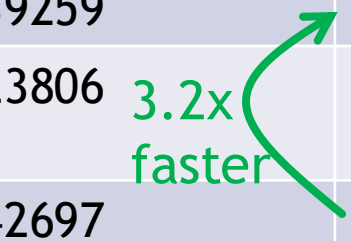
- Around 10 GB
  - Common HEP software
  - Single platform: el9
- Number of files: 58798
- 2 large files:
  - alice_2.data: 983M
  - atlas_1.data: 4.1G

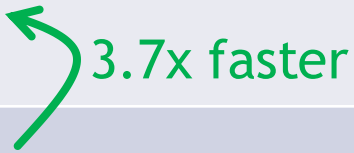
- Note: cms.cern.ch folder contains only:
  - cms.cern.ch/el9_amd64_gcc13/cms/cms/



Pilot PR: Publishing

	Files: Bytes uploaded	Compression Ratio	Catalog: Bytes uploaded	Average run time (sec)
devel (May 10)	5187828757	2.01	5539259	70
WIP - new (De)Compressor	5187828757	2.01	5523806	82
Zstd (level 3)	5060130589	2.06	4842697	22





14% smaller

Pilot PR: Client

Still lots to do…

… and we need your feedback…

… about forward-backward compatibility

Questions - Design choices

- **Mix & match of compression algorithms in a single respository is possible**
  - Old clients will fail on zstd-compressed files
  - As long as catalogs are zlib compressed old clients can read repo
- **Duo-publishing of zlib and zstd compressed files?**
  - Support for old and new clients for fastest performance
  - Manifest would have a root catalog for each compression algo
  - Increases storage usage for S0/S1 by max. 2x

Questions - Design choices II

- **As long as catalogs are zlib compressed, old S1 can work with new zstd files**
  - How likely will you quickly upgrade the S1 to the new client?

Questions - Services

- **Vendor and version locking: cvmfs_fschk**
  - Cvmfs_fschk checks the local client cache for data integrity. For this it recompresses the data and compares the hashes to the hashes found in the repo.
  - Requires bit-for-bit equivalence
  - Used for detection of bit rot in the local cache
  - Questions for site maintainers
    - Is this still needed as a must-have?
    - How much bit rot do you see?

Questions – Services II

- **New service needed? Movement service zlib → zstd**
  - Interest to update old files repos to use zstd?
  - “Normal publishing”
    - From legacy bulk chunk removal we know that locking the entire repo does not work for most cases → needs too much time
  - Use remote publishers and the lease mechanism?

Questions?