CVMFS: Pushing performance on highly parallel, many-core clients

CHEP 2024, Kraków, Poland

Laura Promberger[1], Valentin Völkl[1], Jakob Blomer[1], Matt Harvey[2], and Reza Naghibi[2]

October 24, 2024

[1]CERN, EP-SFT, Switzerland
[2]Jump Trading

What is CernVM-FS?

- Read-only, on-demand, distributed file system
  - Distributes software independent of the underlying platform
- Uses HTTP for web transfer of files
- When software is locally cached it can be as fast as local installation
- All software and data reachable via `/cvmfs/<repo>/...`

For 10+ years it belongs to the critical infrastructure to run HEP computing

CVMFS is used on...

An incomplete selection...

- All WLCG grid sites
- HPC sites in Europe
  - Alps, Switzerland (6)
  - Karolina, Czechia (135)
  - LUMI, Finland (5)
  - Vega, Solvenia (226)
  - MareNostrum5 (8), Spain
  - ...
- Digital Research Alliance of Canada

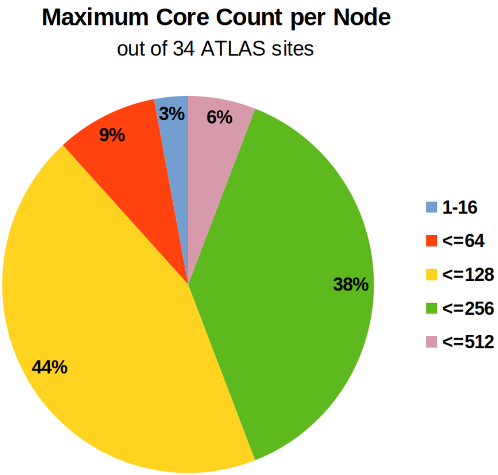
(Top500, June24)

- HPC sites in USA
  - ALCF Polaris (30)
  - ALCF Theta (134)
  - NERSC Perlmutter (14)
  - NERSC Cori (74)
  - NERSC Edison
  - OLCF Summit (9)
  - PSC Bridges-2
  - Purdue Anvil
  - SDSC Expanse (403)
  - TACC Frontera (33)
  - ...

CVMFS is used on... II

Typical CPU Specs of current HPC nodes

- AMD EPYC 7h12: 64 cores, 128 threads
- AMD EPYC 7763: 64 cores, 128 threads
- AMD EPYC 7742: 64 cores, 128 threads
- AMD EPYC 7452: 32 cores, 64 threads
- ARM A64FX: 48 cores, 48 threads
- Intel Xeon-SC 8628: 24 cores, 48 threads
- Intel Xeon Platinum 8480: 56 cores, 112 threads
- Intel Xeon 9480: 56 cores, 112 threads
- Intel Xeon 8460Y: 40 cores, 80 threads



Known issues on large many-core CVMFS clients

1. Crashing programs because out-of-file-descriptors
    - `https://github.com/cvmfs/cvmfs/issues/3067`
2. Bottleneck download: Decompression of downloaded chunks is sequential
    - `https://indico.cern.ch/event/1180962/contributions/4960898`

Benchmark setup

- Hardware
  - CVMFS client: 2x AMD EPYC 7702 64-Core (=256 virtual cores), 1 TB RAM, 2 TB NVMes
  - Private proxy: 1x Intel i7-7820X 8-Core, 64 GB RAM, 1 TB HDDs, 9 Gbps Ethernet, 0.3 ms latency
- Measurement modes
  - Cold cache: data only on proxy
  - Warm cache: data on local disk
  - Hot cache: data on local disk and kernel cache
- Relationship: 1 (virtual) thread = 1 process
  - 1 thread = 1 process of `command`
  - 256 threads = 256 processes of `command`
- 10 repetitions of each mode

Benchmark setup II

- Commands
  - Tensorflow (TF): Import numpy and tensorflow in python (LCG_103)
    - Each thread runs the same command
  - ROOT: Create 1D Histogram of 100 random values (LCG_103)
    - Each thread runs the same command
  - Different ROOT versions: 71 combinations of 12 ROOT (sub)versions and different compilers (dbg, opt, gcc, clang, ..) for EL9
    - Version selection: `thread_id` % 71
  - Random walk LCG: Read files given by file lists (LCG_106)
    - Each thread gets a different file lists
    - Each file lists should take around 40 sec runtime for single process, cold cache performance

LCG = Software stack: Over 800 external packages as well as HEP specific tools and generators.
See `https://ep-dep-sft.web.cern.ch/document/lcg-releases`

Known issues on large many-core CVMFS clients

1. **Crashing programs because out-of-file-descriptors**
   - `https://github.com/cvmfs/cvmfs/issues/3067`
   - Solution available since 2.11
2. Bottleneck download: Decompression of downloaded chunks is sequential
   - `https://indico.cern.ch/event/1180962/contributions/4960898`

Reference-counted cache manager

- Default cache manager
  - Each `open()` creates a new file descriptor even if the file is already used by some other process using CVMFS
  - Problem: On large many core machines it is easy to run out of file descriptors

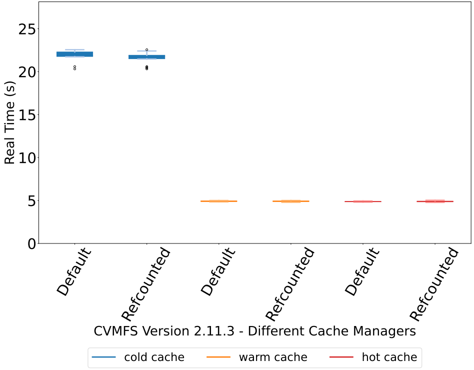
- Reference-counted cache manager
  - CVMFS deduplicates file descriptors when file is opened many times
  - Only one file descriptor per file
  - Available from version 2.11 on

- Side note: Default fd limit on EL9 is only 1024

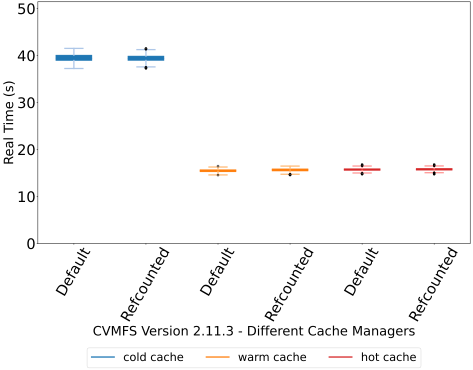
Reference-counted cache manager II - Comparison: Cache managers

Both cache managers, default and refcounted, have the **same performance**.

Default cache mgr: Even with max fd limit, TF only can run on 141 of 256 threads ϟ



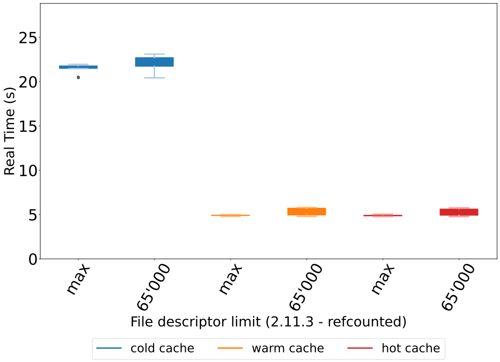
TF - 1 thread - max fd limit



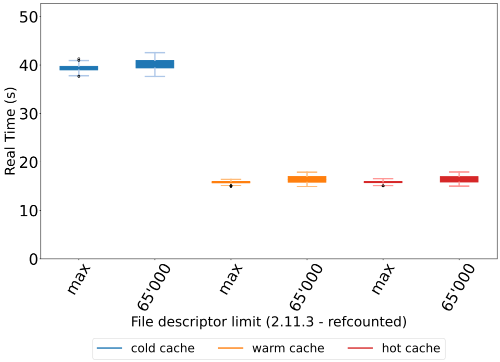
TF - 141 threads - max fd limit

Reference-counted cache manager III - Comparison: Fd limit

A lower fd limit seems to slightly decrease the overall CVMFS performance

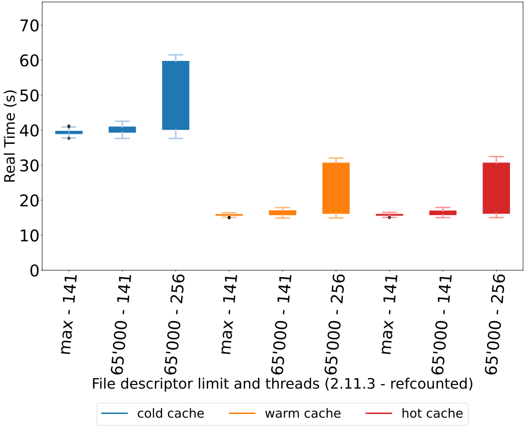


TF - 1 thread - refcounted



TF - 141 threads - refcounted

Reference-counted cache manager III - Comparison: Threads



TF - different threads - refcounted

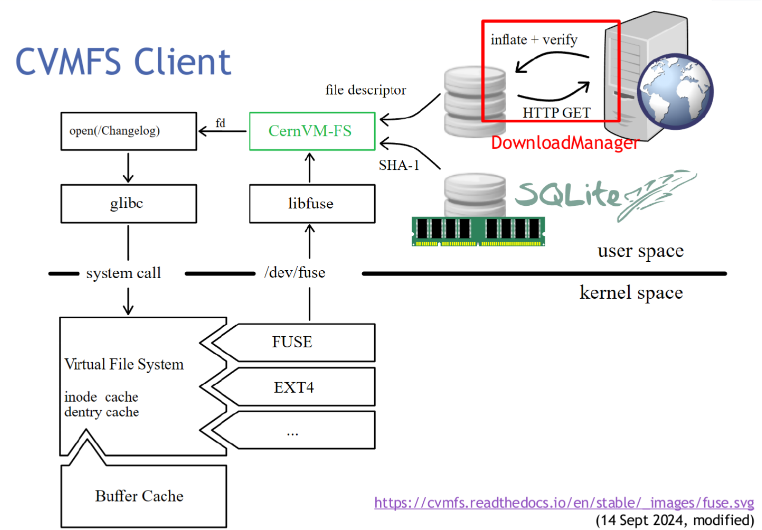
The **refcounted** cache manager allows us to **use the full machine** of 256 threads! ✔

Compared to 141 threads, 256 threads are on average 20-30% slower but 81% more work is performed

Known issues on large many-core CVMFS clients

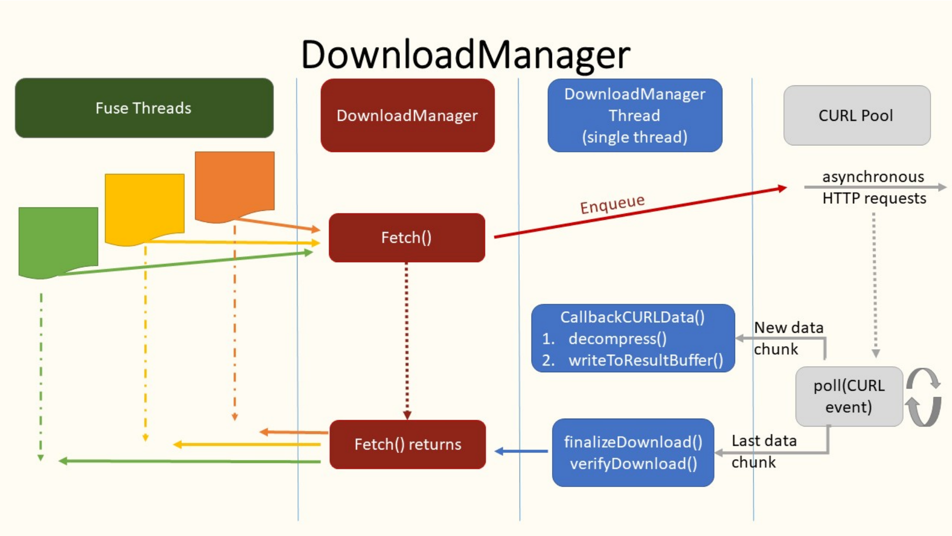
1. Crashing programs because out-of-file-descriptors
    - `https://github.com/cvmfs/cvmfs/issues/3067`
    - Solution available since 2.11
2. **Bottleneck download: Decompression of downloaded chunks is sequential**
    - `https://indico.cern.ch/event/1180962/contributions/4960898`
    - Currently only available as PR

CVMFS Client

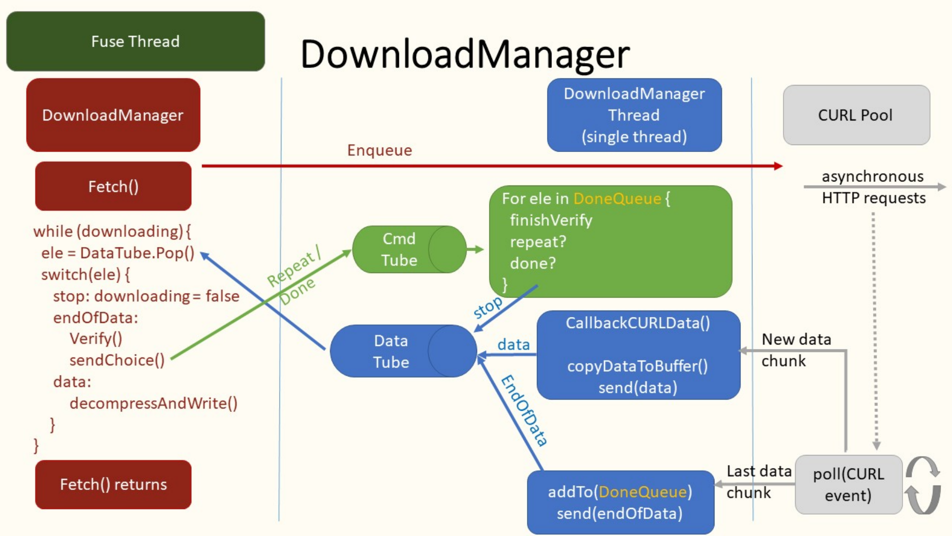


https://cvmfs.readthedocs.io/en/stable/_images/fuse.svg
(14 Sept 2024, modified)

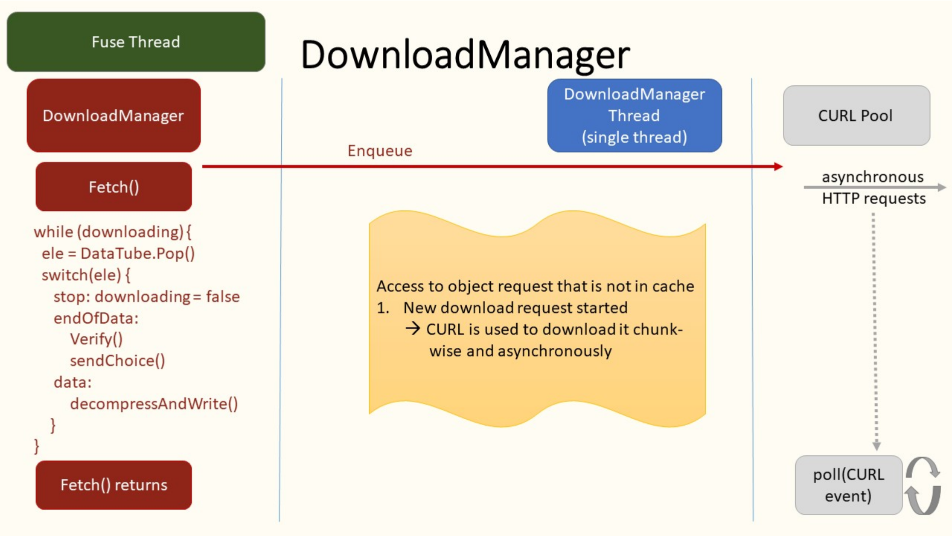
DownloadManager: Old implementation



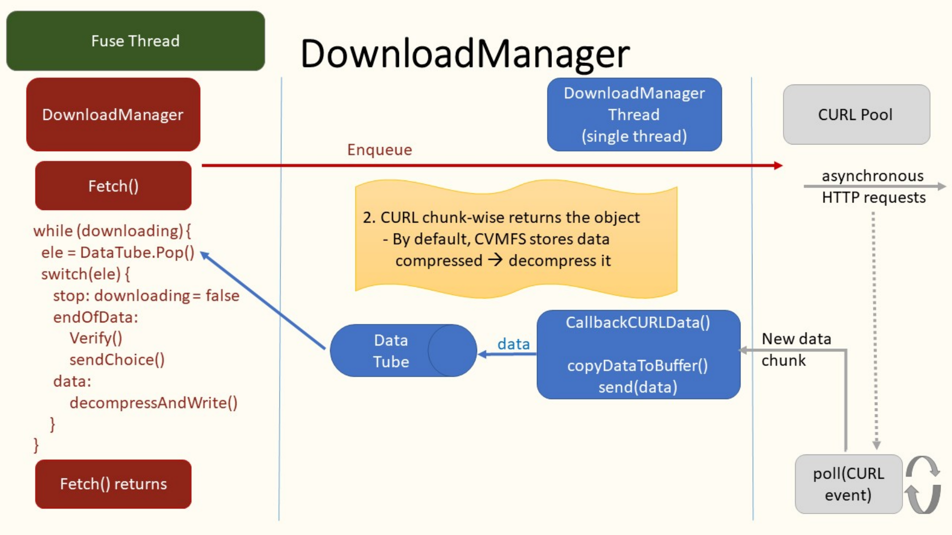
DownloadManager: New implementation



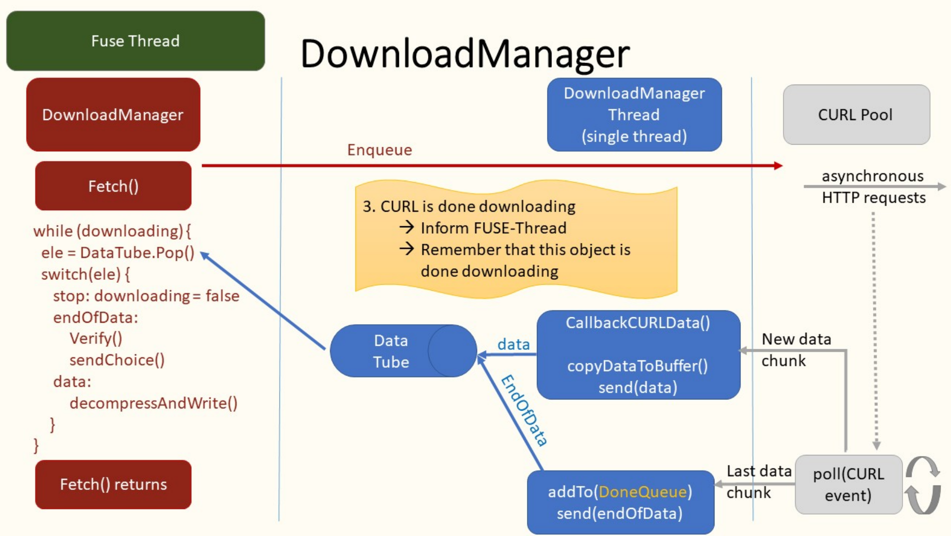
DownloadManager: New implementation



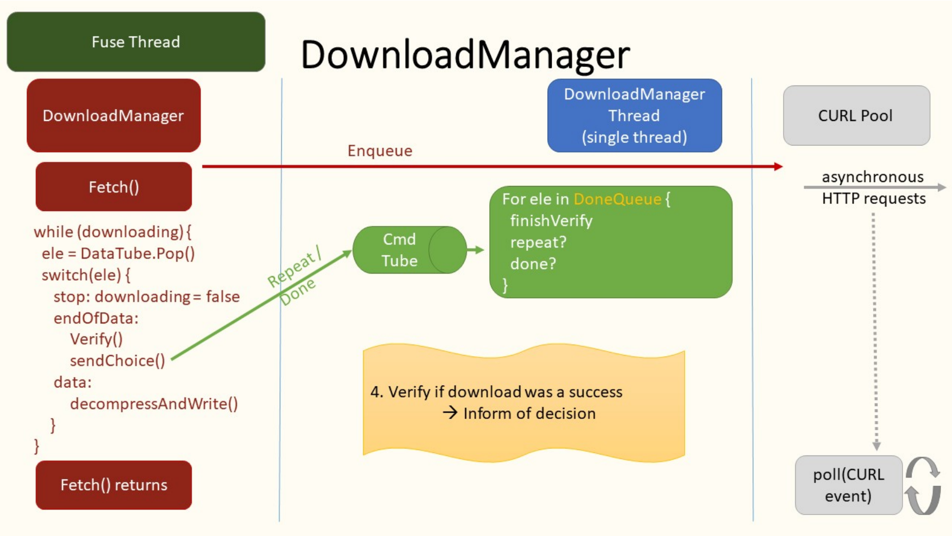
DownloadManager: New implementation



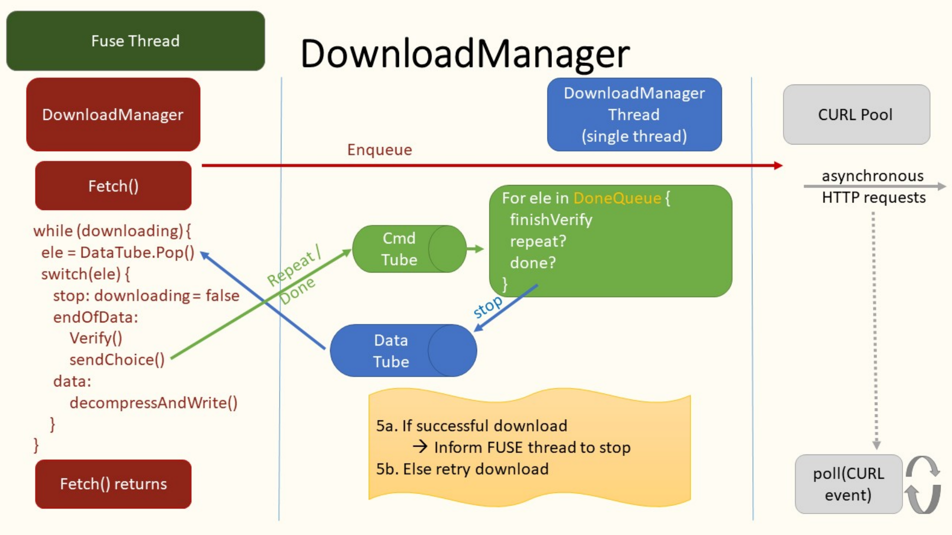
DownloadManager: New implementation



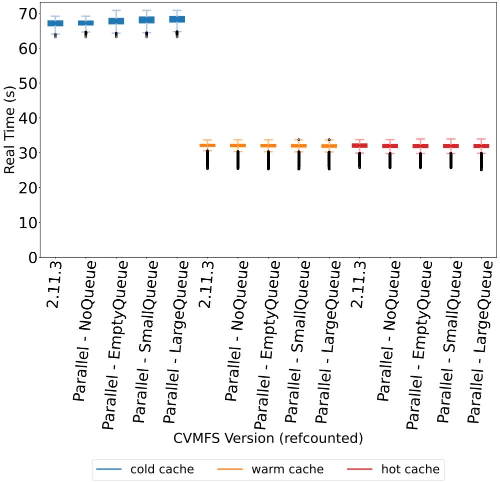
DownloadManager: New implementation



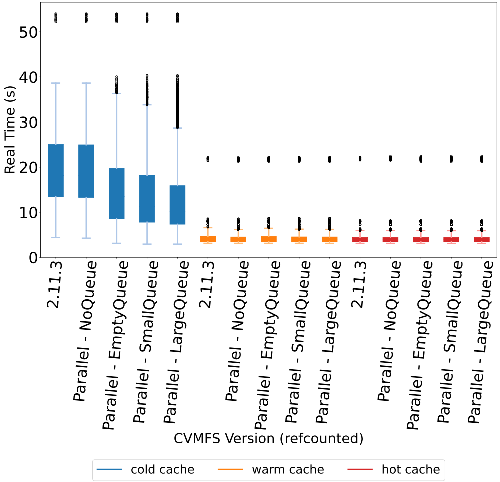
DownloadManager: New implementation



Parallel Decompression: Hot and warm cache unaffected

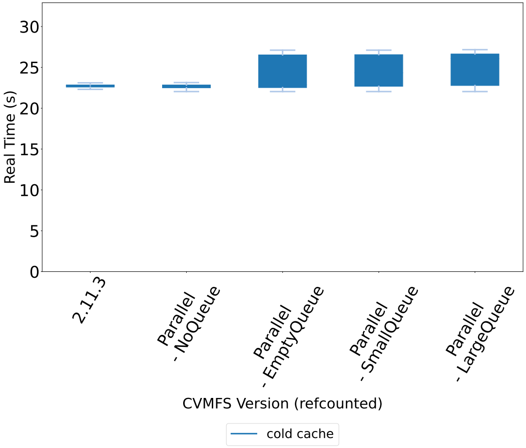


ROOT, 256 threads

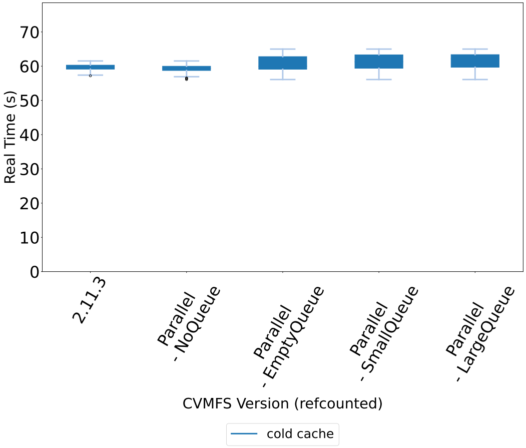


Random walk, 128 Threads

Parallel Decompression: All processes access same data

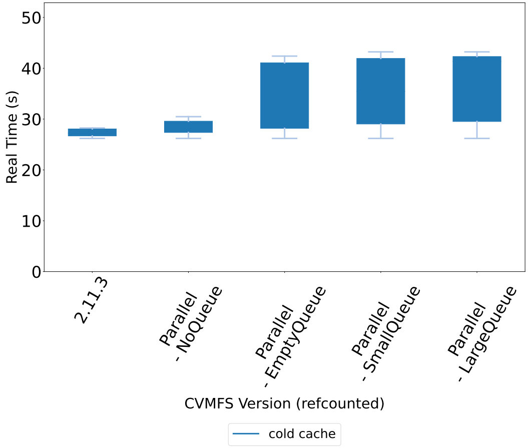


Tensorflow - 1 thread

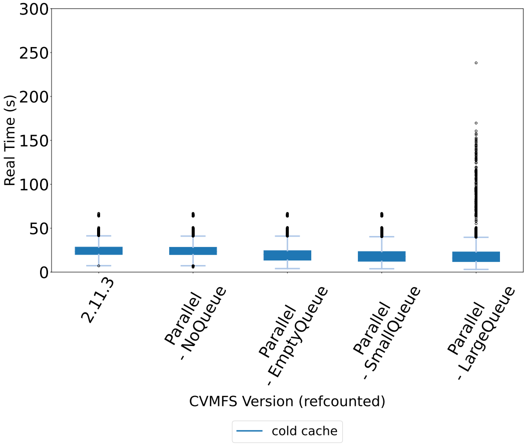


Tensorflow - 256 threads

Parallel Decompression: All processes access different data

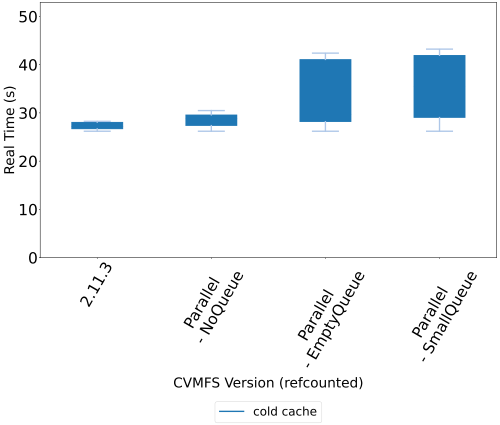


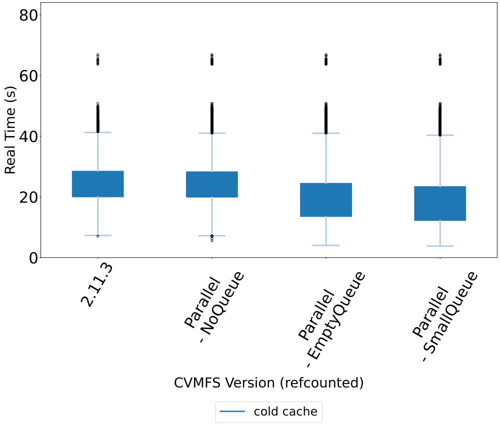
Random walk - 1 thread



Random walk - 256 threads

Parallel Decompression: All processes access different data - No LargeQueue





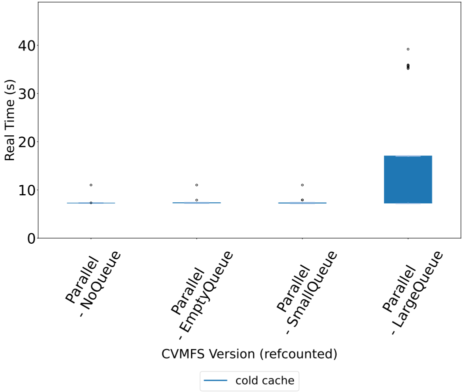
Random walk - 1 thread

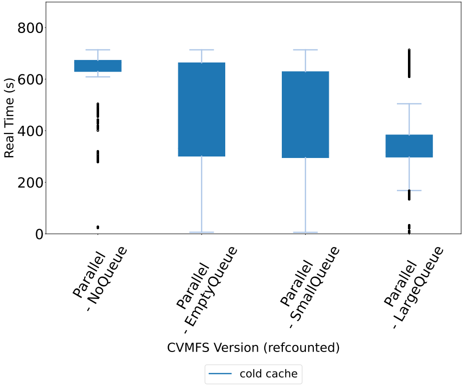
Parallel Decomp up to 29% slower

Random walk - 256 threads

Parallel Decomp up to 25% faster

Parallel Decompress.: 71 different combos of 12 ROOT versions × n compilers





Different ROOT versions - 1 thread

Parallel Decomp can be same speed

Different ROOT versions - 256 threads

Parallel Decomp 30 - 40% faster

Conclusion

Improved performance of the CVMFS client for large many-core machines

- **Reference-counted cache manager**
  - Allows to use the full performance of large many-core machines
  - Has a very similar performance to the default cache manager
  - From version 2.11 on available → **Use it! It only has advantages**

- **Parallel decompression of downloaded chunks**
  - Warm and hot cache unaffected by those changes
  - Characteristics of access patterns will help to find the most efficient configuration
    - Up to 30 - 40% faster for highly parallel file accesses on large many-core machines
    - Do not use parallel decompression if the access pattern is: sequential file access
  - **If uncertain about parallelism in download requests, use parallel decompression with an empty queue**
    - Max 10% slower (1 thread, sequential file access) but up to 30% faster

Add to your client config

Location of your CVMFS client config files: `/etc/cvmfs/`

Read more here: `https://cvmfs.readthedocs.io/en/stable/cpt-configure.html`

- **Reference-counted cache manager**
  - Minimum client version: 2.11
    `CVMFS_CACHE_REFCOUNT=yes`

- **Parallel decompression of downloaded chunks: Empty Queue**
  - Still a PR!, but syntax will be similar to:
    ```
    CVMFS_PARALLEL_DOWNLOAD_MIN_BUFFERS=0
    CVMFS_PARALLEL_DOWNLOAD_MAX_BUFFERS=0
    CVMFS_PARALLEL_DOWNLOAD_INFLIGHT_BUFFERS=1
    ```

Questions?

Find us at CHEP or write us!

Questions: `https://cernvm-forum.cern.ch`

Feature requests and bug reports: `https://github.com/cvmfs/cvmfs/issues/`

E-mail: laura.promberger@cern.ch