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

CHEP 2024, Kraków, Poland

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

October 24, 2024

[1]CERN, Experimental Physics Department, 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>/...`

Since 15+ 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

- Out of 34 ATLAS sites
  - 2 have up to 512 cores per node
  - 13 have up to 256 cores per node
  - 15 have up to 128 cores per node
  - 3 have up to 64 cores per node
  - 1 has only 1-16 nodes

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, 10 Gbps ethernet, 0.2 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`

2. Bottleneck download: Decompression of downloaded chunks is sequential
   - `https://indico.cern.ch/event/1180962/contributions/4960898`

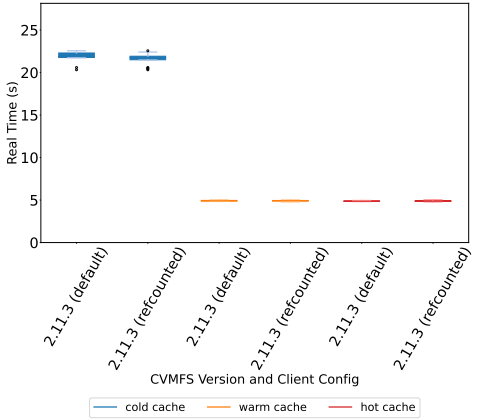
Reference-counted cache manager

- 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

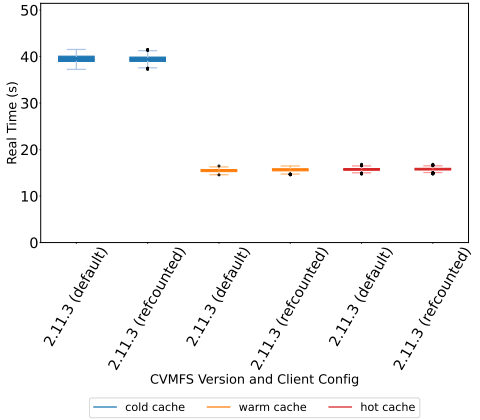
- Reference-counted cache manager
  - CVMFS deduplicates file descriptors when file is opened many times
  - Only one file descriptor per file

Reference-counted cache manager II - Comparison: Cache managers

Both cache managers, default and refcounted, have the same performance for version.
Even with a fd limit of 536 mio, TensorFlow only can run on 141 of 256 threads



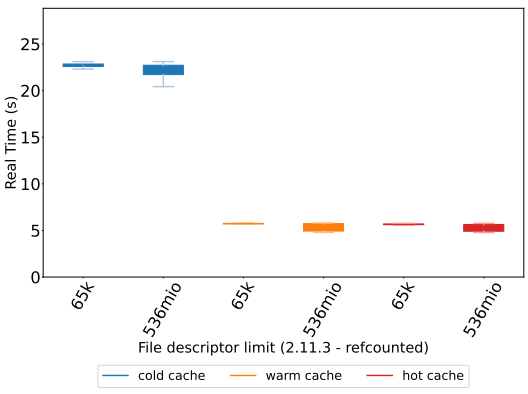
TF - 1 thread - fds 536'000'000



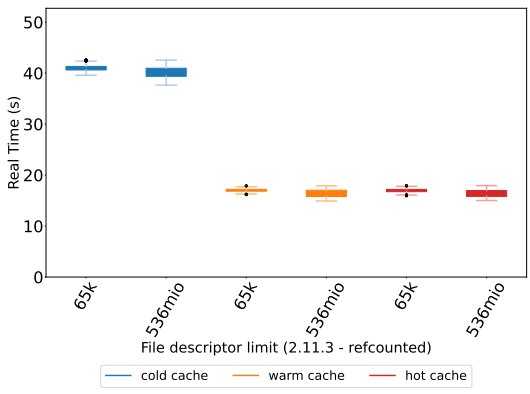
TF - 141 threads - fds 536'000'000

Reference-counted cache manager III - Comparison: FD limit

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

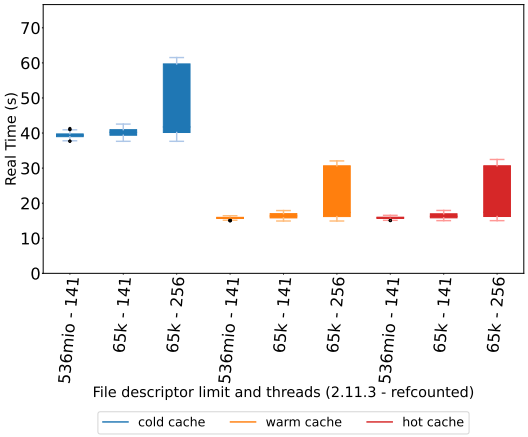


TF - 1 thread - refcounted



TF - 141 threads - refcounted

Reference-counted cache manager III - Comparison: Threads



TF - different threads - refcounted

Compared to 141 threads,
256 threads are on average:

- 20% slower for cold cache
- 32% slower for warm cache
- 31% slower for hot ache

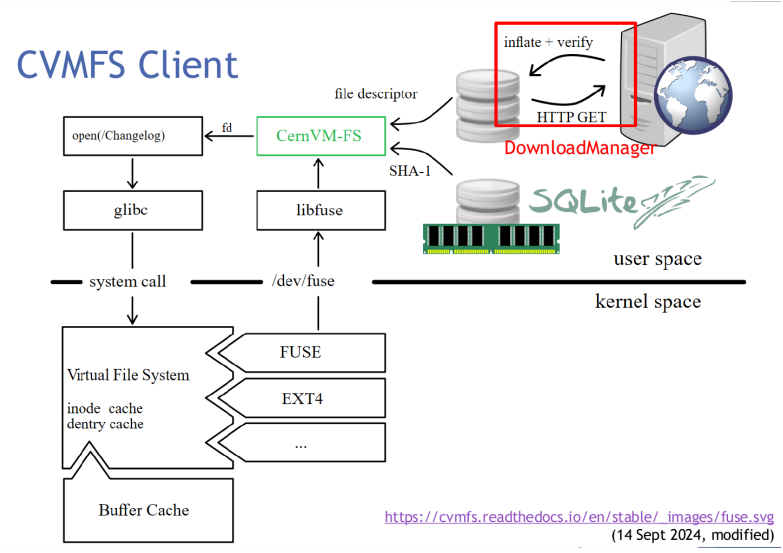
- But performed 81% more work

Known issues on large many-core CVMFS clients

1. Crashing programs because out-of-file-descriptors
   - https://github.com/cvmfs/cvmfs/issues/3067

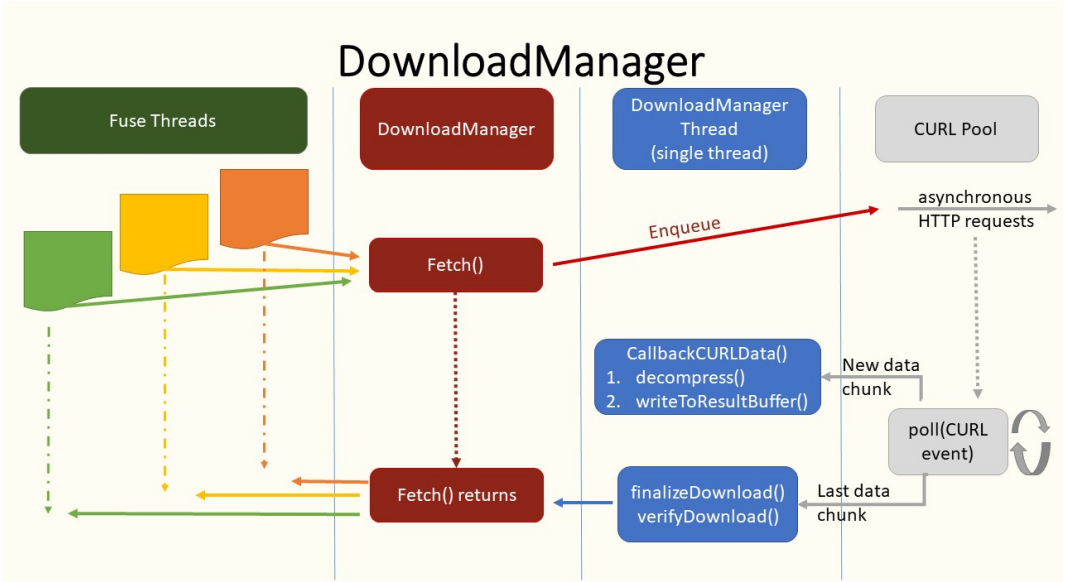
2. **Bottleneck download: Decompression of downloaded chunks is sequential**
   - https://indico.cern.ch/event/1180962/contributions/4960898

CVMFS Client

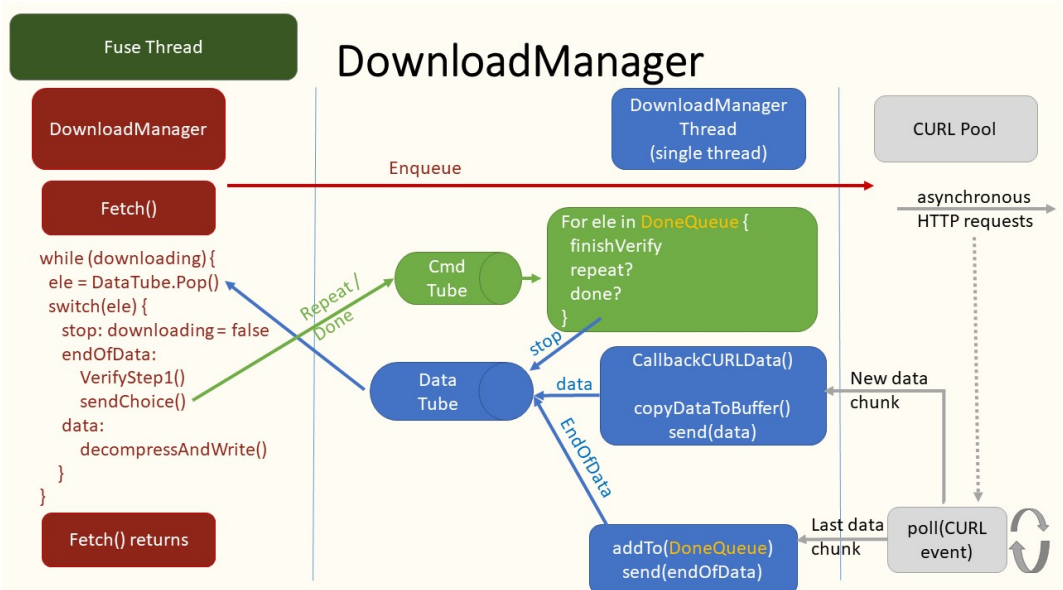


https://cvmfs.readthedocs.io/en/stable/_images/fuse.svg
(14 Sept 2024, modified)

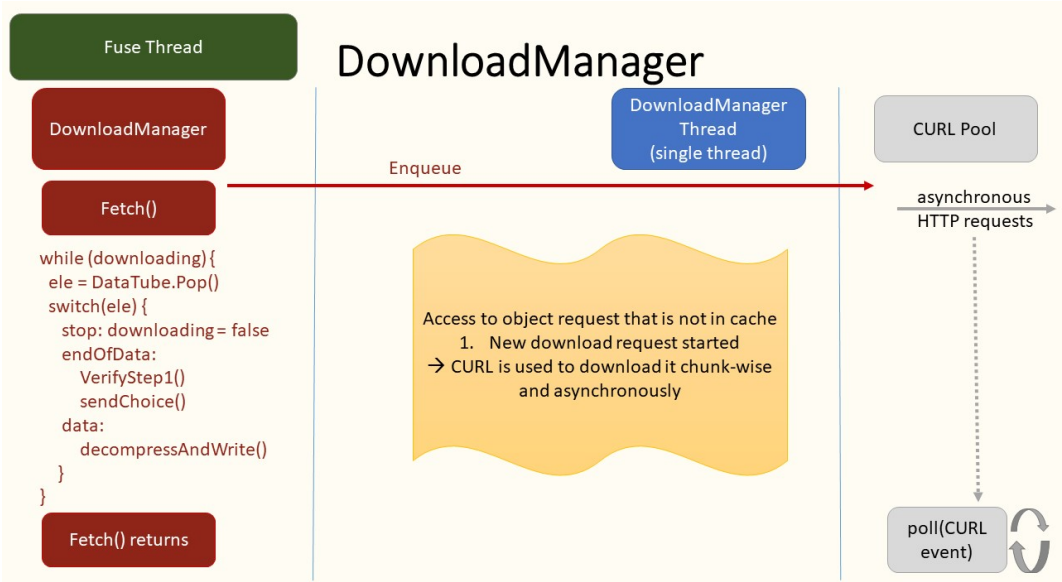
DownloadManager: Old implementation



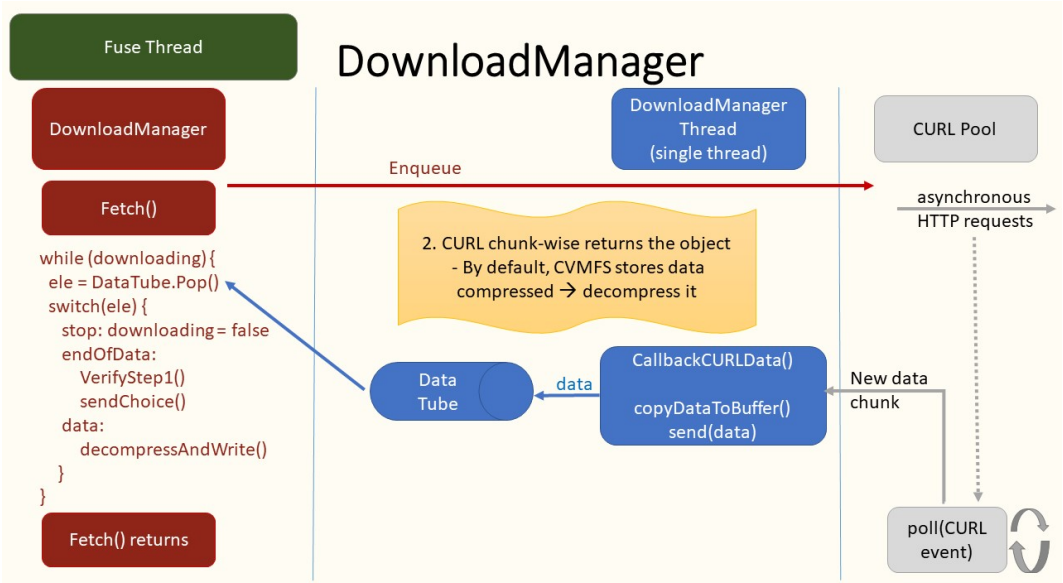
DownloadManager: New implementation



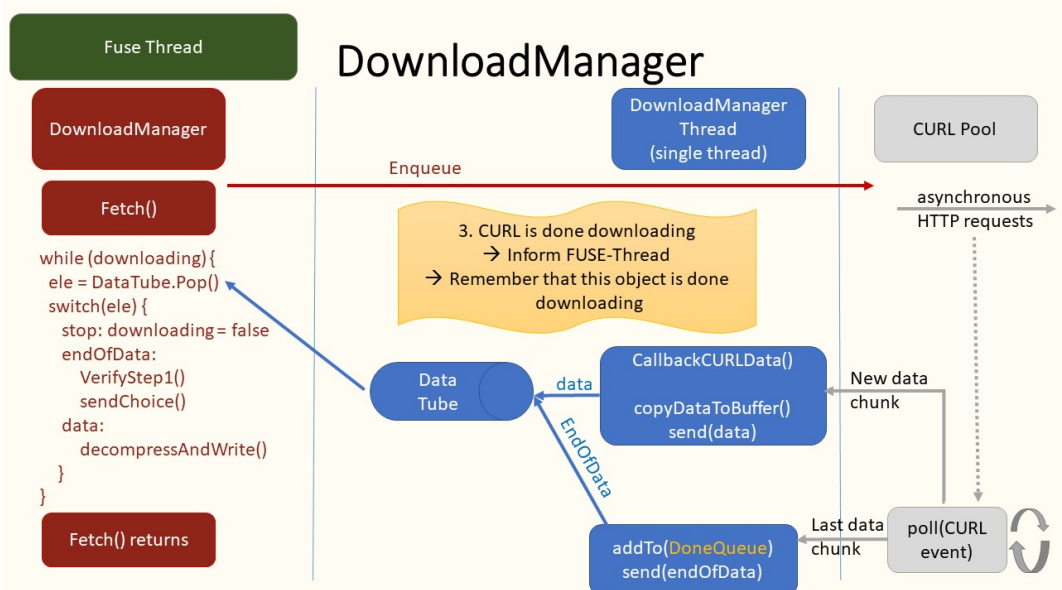
DownloadManager: New implementation



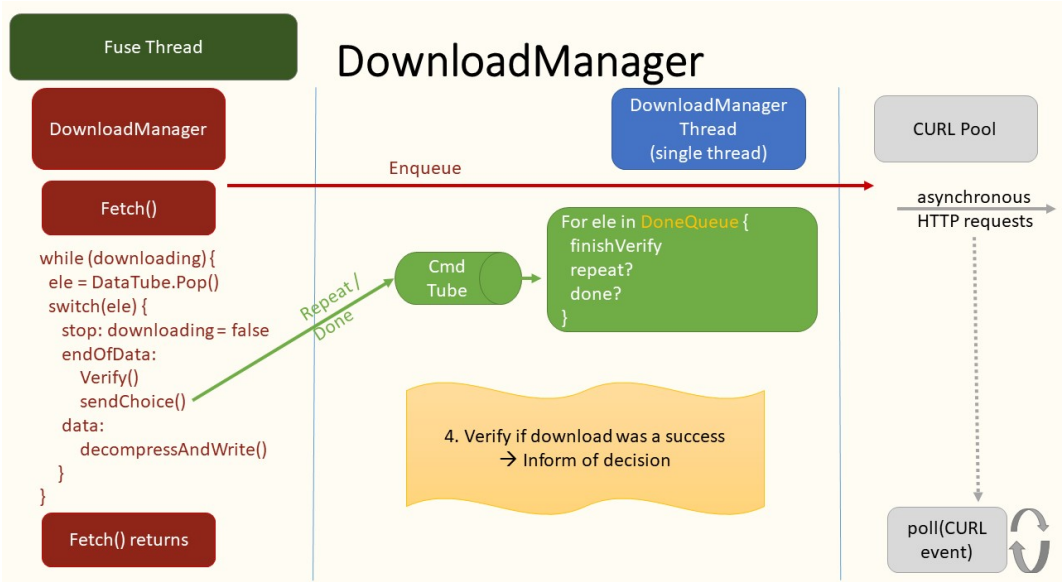
DownloadManager: New implementation



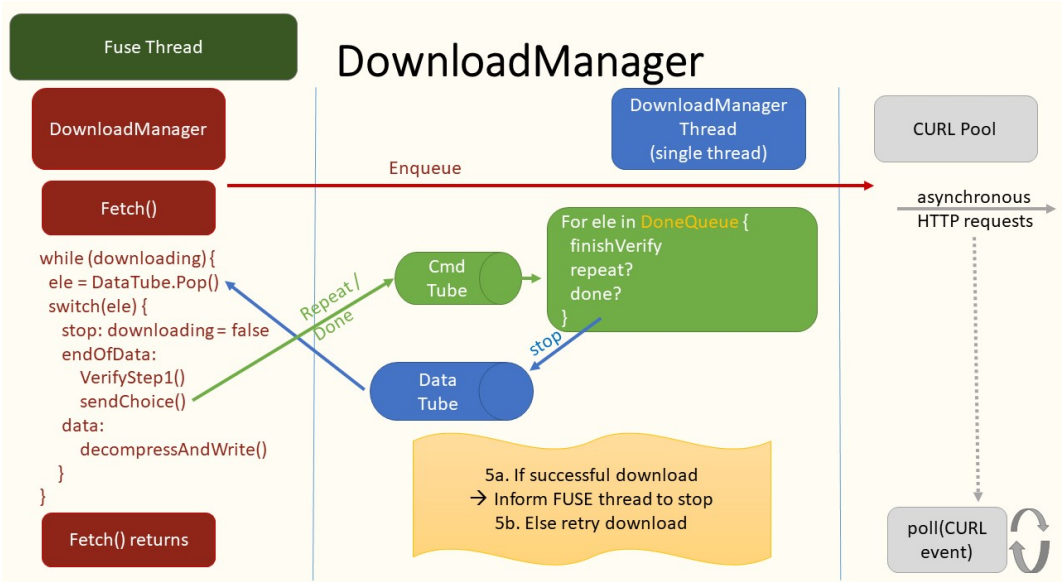
DownloadManager: New implementation



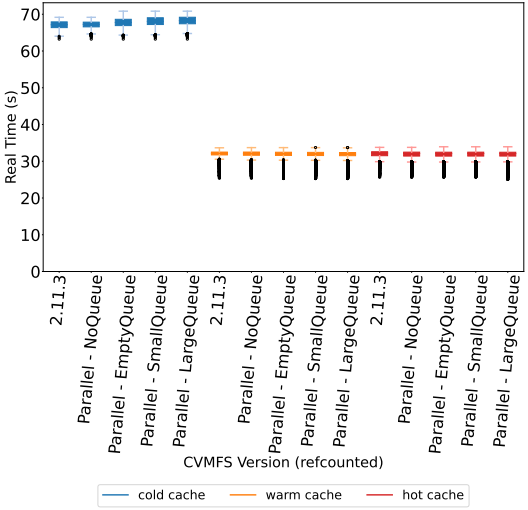
DownloadManager: New implementation



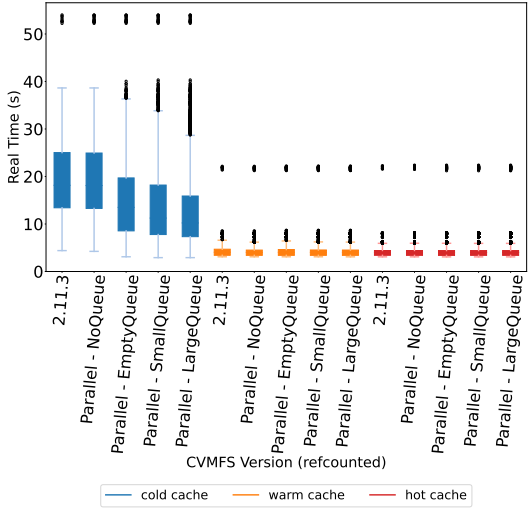
DownloadManager: New implementation



Parallel Decompression: Hot and warm cache unaffected

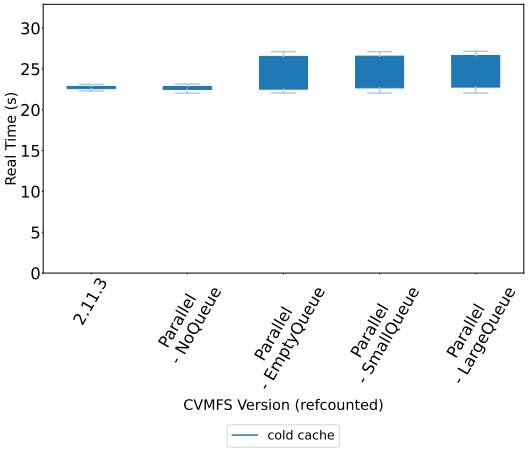


ROOT, 256 threads

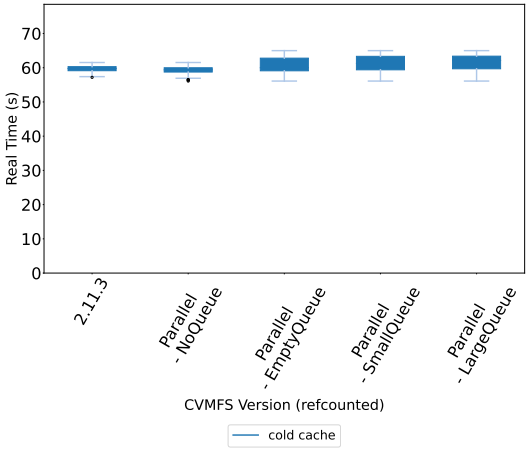


Random walk, 128 Threads

Parallel Decompression: All processes access same data

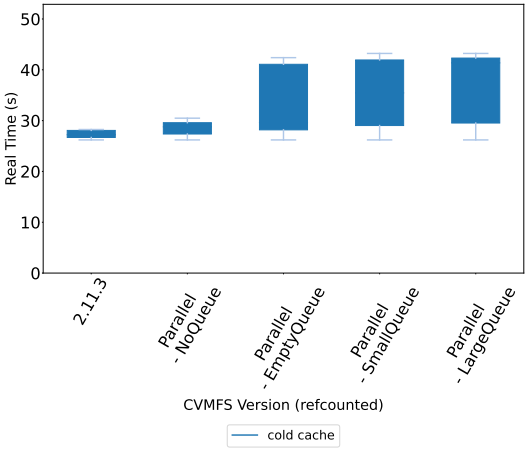


Tensorflow - 1 thread

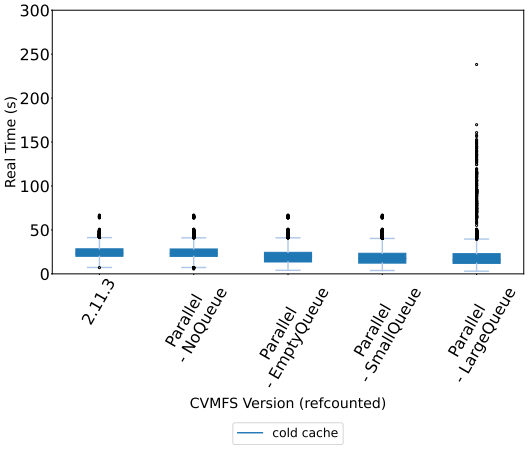


Tensorflow - 256 threads

Parallel Decompression: All processes access different data

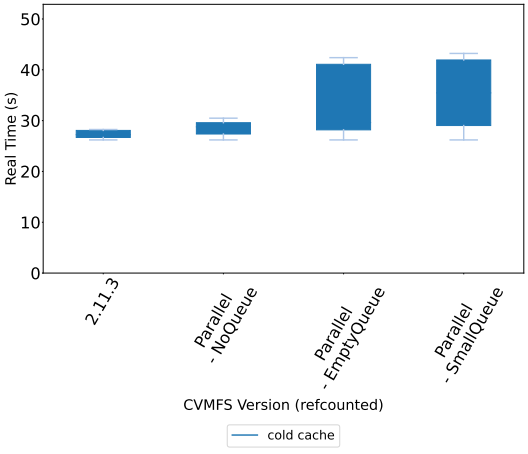


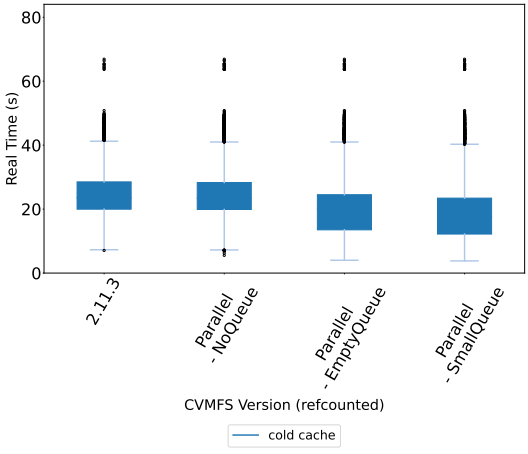
Random walk - 1 thread



Random walk - 256 threads

Parallel Decompression: All processes access different data - No LargeQueue





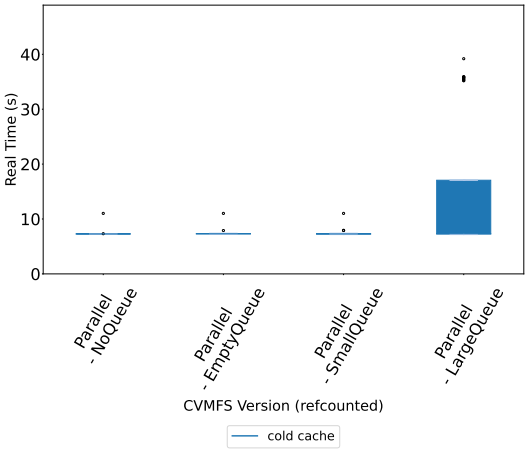
Random walk - 1 thread

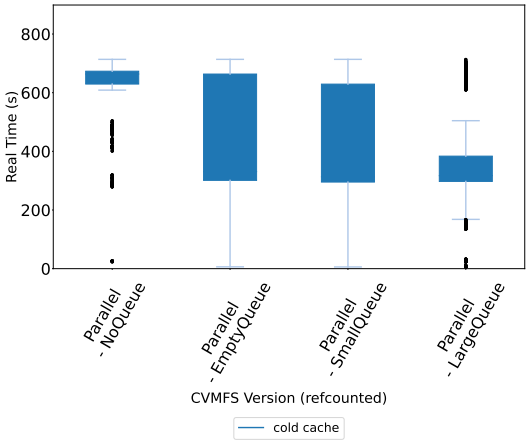
Parallel Decomp up to 29% slower

Random walk - 256 threads

Parallel Decomp up to 25% faster

Parallel Decompression: 71 different combinations of compilers and 12 ROOT versions





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

- **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

Questions?

Average Runtime: Baseline (2.11.3) vs XX

2.11.3.0_default_kernel_refcounted vs 2.12.0.0-parallelDecomp_default_kernel_refcounted							
		base time	cold cache	base time	warm cache	base time	hot cache
cvmfs-lcg-40sec-45-random-walk	1	27.305	104.48%	17.477	99.88%	17.526	99.91%
	128	19.00157031	99.53%	4.402296875	99.15%	4.353960938	99.90%
	256	25.01682031	99.09%	7.557464844	99.97%	7.46484375	100.00%
cvmfs-lcg-random-walk	1	5.394	101.71%	3.133	98.91%	3.11	99.68%
	128	13.81755469	98.93%	4.818109375	100.60%	4.8194375	100.14%
	256	21.05119922	99.87%	9.307738281	100.17%	9.194761719	100.70%
root	1	30.862	100.05%	5.38	100.08%	5.372	99.85%
	128	48.01150781	99.80%	16.78026563	100.29%	17.00715625	100.09%
	256	67.07679297	100.13%	31.79561328	99.95%	31.71991016	99.62%
tensorflow	1	22.743	99.57%	5.729	99.69%	5.656	100.70%
	128	39.59390625	99.64%	15.238875	100.88%	15.50726563	99.94%
	256	59.691125	99.42%	30.52238281	99.50%	30.48909766	100.01%

Average Runtime: Baseline (2.11.3) vs XX

2.11.3.0_default_kernel_refcounted vs 2.12.0.0-parallelDecomp_default_kernel_parallelEmpty_refcounted							
		base time	cold cache	base time	warm cache	base time	hot cache
cvmfs-lcg-40sec-45-random-walk	1	27.305	120.48%	17.477	98.91%	17.526	99.22%
	128	19.00157031	81.50%	4.402296875	99.45%	4.353960938	99.65%
	256	25.01682031	83.27%	7.557464844	99.79%	7.46484375	99.77%
cvmfs-lcg-random-walk	1	5.394	112.95%	3.133	98.01%	3.11	98.83%
	128	13.81755469	86.52%	4.818109375	100.34%	4.8194375	99.92%
	256	21.05119922	85.71%	9.307738281	100.00%	9.194761719	100.34%
root	1	30.862	101.49%	5.38	100.07%	5.372	100.07%
	128	48.01150781	100.74%	16.78026563	100.64%	17.00715625	100.26%
	256	67.07679297	101.10%	31.79561328	99.82%	31.71991016	99.62%
tensorflow	1	22.743	105.63%	5.729	99.86%	5.656	100.48%
	128	39.59390625	102.89%	15.238875	101.18%	15.50726563	99.84%
	256	59.691125	101.67%	30.52238281	99.61%	30.48909766	100.06%
cvmfs-root-all-versions	1	7.656	98.24%	562.8812578	99.71%	615.5923242	99.82%
	128	39.59390625	71.65%	28.27219531	102.99%	54.76466797	102.10%
	256	59.691125	72.78%	28.46540625	99.97%	54.67645703	100.23%

Average Runtime: Baseline (2.11.3) vs XX

2.11.3.0_default_kernel_refcounted vs 2.12.0.0-parallelDecomp_default_kernel_parallelSmall_refcounted

		base time	cold cache	base time	warm cache	base time	hot cache
cvmfs-lcg-40sec-45-random-walk	1	27.305	129.13%	17.477	99.02%	17.526	98.75%
	128	19.00157031	72.48%	4.402296875	99.35%	4.353960938	99.73%
	256	25.01682031	75.44%	7.557464844	99.46%	7.46484375	99.72%
cvmfs-lcg-random-walk	1	5.394	119.42%	3.133	97.45%	3.11	98.13%
	128	13.81755469	80.69%	4.818109375	100.13%	4.8194375	99.90%
	256	21.05119922	78.62%	9.307738281	100.25%	9.194761719	100.18%
root	1	30.862	102.12%	5.38	100.02%	5.372	100.17%
	128	48.01150781	101.19%	16.78026563	100.81%	17.00715625	100.31%
	256	67.07679297	101.47%	31.79561328	99.67%	31.71991016	99.62%
tensorflow	1	22.743	108.03%	5.729	99.86%	5.656	100.57%
	128	39.59390625	104.22%	15.238875	100.97%	15.50726563	99.91%
	256	59.691125	102.70%	30.52238281	99.47%	30.48909766	99.96%
cvmfs-root-all-versions	1	7.656	97.29%	562.8812578	99.40%	615.5923242	99.40%
	128	39.59390625	62.95%	28.27219531	104.06%	54.76466797	102.01%
	256	59.691125	63.64%	28.46540625	99.39%	54.67645703	98.77%

Average Runtime: Baseline (2.11.3) vs XX

2.11.3.0_default_kernel_refcounted vs 2.12.0.0-parallelDecomp_default_kernel_parallelLarge_refcounted							
		base time	cold cache	base time	warm cache	base time	hot cache
cvmfs-lcg-40sec-45-random-walk	1	27.305	134.24%	17.477	98.28%	17.526	97.66%
	128	19.00157031	67.02%	4.402296875	99.23%	4.353960938	99.75%
	256	25.01682031	77.61%	7.557464844	99.48%	7.46484375	99.79%
cvmfs-lcg-random-walk	1	5.394	122.90%	3.133	97.40%	3.11	98.08%
	128	13.81755469	77.37%	4.818109375	100.22%	4.8194375	99.97%
	256	21.05119922	74.41%	9.307738281	100.81%	9.194761719	100.43%
root	1	30.862	102.65%	5.38	100.15%	5.372	100.29%
	128	48.01150781	101.48%	16.78026563	100.99%	17.00715625	100.38%
	256	67.07679297	101.69%	31.79561328	99.55%	31.71991016	99.60%
tensorflow	1	22.743	109.89%	5.729	99.72%	5.656	100.55%
	128	39.59390625	104.73%	15.238875	100.86%	15.50726563	99.88%
	256	59.691125	103.16%	30.52238281	99.43%	30.48909766	99.89%
cvmfs-root-all-versions	1	7.656	190.37%	562.8812578	177.73%	615.5923242	177.56%
	128	39.59390625	59.88%	28.27219531	104.57%	54.76466797	103.29%
	256	59.691125	60.56%	28.46540625	101.14%	54.67645703	100.08%