UA Remote student internship report: Benchmarking modern Overlay FS features

Yuriy Belikov,
CERN-VM FS Team,
June 17th 2024

What is CernVM-FS?

The CernVM File System distributes LHC experiment software and conditions data to the world-wide LHC computing infrastructure.

- 5 billion files under management;
- 100 000 worker nodes with read-only clients installed
- Implemented as FUSE file system
- Uses a union file system: the read-only file system client is combined with a temporary scratch area to record a change set
- Relies on Overlay FS for repository updates (cvmfs_server)
- Uses content-addressable storage and Merkle trees in order to maintain file data and metadata

CVMFS main components

- FUSE-based client
- Command line toolchain for server-side (cvmfs_server utility is used for publishing repository updates)
- HTTP multilevel cache
- HTTP servers

Publishing with CVMFS

- Invokes cvmfs_server CLI utility
- Requires direct interaction with a release manager
- Writing cannot be done simultaneously by several writers
- Repository view is constructed via union mount (**Overlay FS**)
- Published files are compressed and hashed
- New data are stored in content addressable storage with their metadata preserved in SQLite file catalogs

Example:

```
[ ~ ] # cvmfs_server mkfs repo.name.org
[ ~ ] # cvmfs_server transaction repo.name.org
[ ~ ] # vim /cvmfs/repo.name.org/new_file.txt
[ ~ ] # cvmfs_server publish repo.name.org
```

Overlay FS general info

OverlayFS is a type of filesystem service in Linux that enables you to overlay one filesystem on top of another. This functionality allows multiple directories to be combined into a single unified view, making it particularly useful for systems where you want to maintain a read-only system base while making changes that appear to be writable.
A typical setup consists of:

- Lower Directory: This is the underlying filesystem which is typically read-only.
- Upper Directory: This directory is typically writeable. Modifications to the filesystem—such as adding, modifying, or deleting files—are performed in this directory.
- Merge (or Overlay) Directory: This is the unified view that users interact with. It appears to contain both the contents of the lower and upper directories. When a file from the lower directory is modified, the modified version is stored in the upper directory, while the original remains unchanged in the lower directory.
- Work Directory: This is used internally by OverlayFS to manage files while they are being modified. It needs to be an empty directory on the same filesystem as the upper directory.

OverlayFS general info

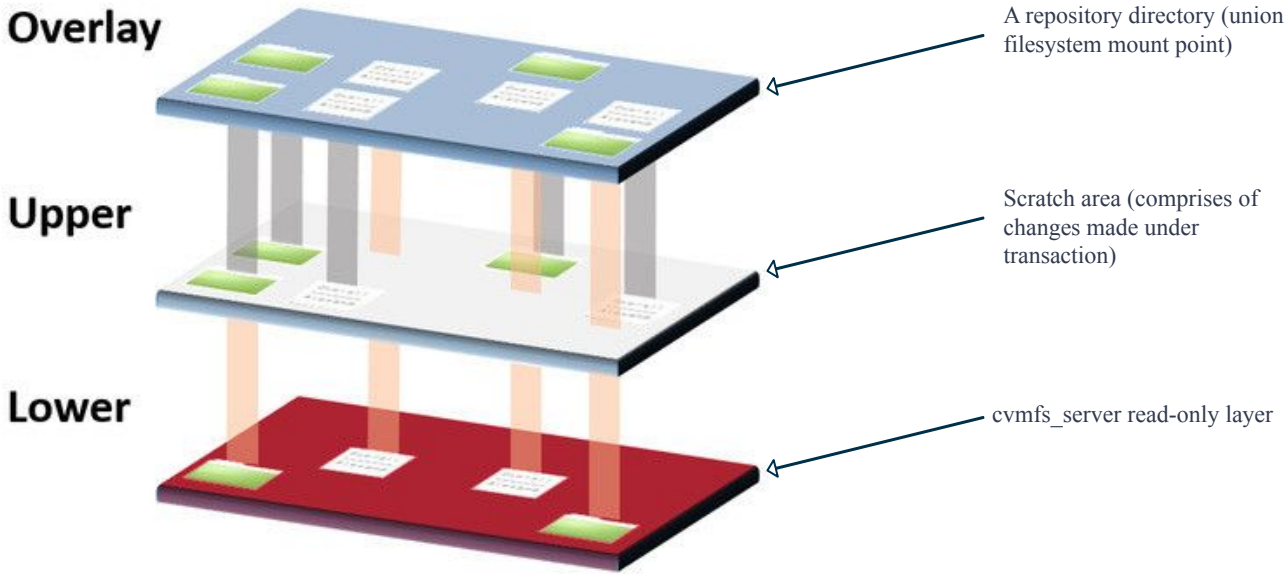


Image source:
https://commons.wikimedia.org/wiki/File:OverlayFS_Image.png

Short overview of the project

- On ***cvmfs_server publish*** the utility traverses scratch area and stores information about the contents of repository in file catalogs and the content itself in a compressed state.
- Since compression, hashing and updating file catalogs is performed for full copies of the modified files modern overlay FS features have a potential to improve performance of cvmfs_server transactions via avoiding the unnecessary deep copying to scratch area.

However, integrating zero-copy directory renames appeared not as trivial as was firstly expected:

- As initial logic of cvmfs transactions relies on the fact that scratch area contains full copies of file system objects zero-copy rename leads to wiping out old directory with all its contents
- Subdirectories removal creates a different footprint in the upper-layer directory: a whiteout appears instead of the removed subdirectory which is not the case for a usual setup

Project objectives

- Benchmark performance for a small set of file operations with OverlayFS setup on a local filesystem
- Compare time required to execute renaming and metadata-affecting operations on modified and regular mounts
- Spot nuances in how the modified setup works
- Start embracing modifications in ***cvmfs_server*** to avoid unnecessary copy-ups

What is the copy-up?

Let’s consider the following initial setup:

lower_dir	**upper_dir**
• a1.txt 5 MB • b1.txt 10 MB • c1.out 15 MB	• a.txt - 1 MB • b.txt - 2 MB • c.out - 3 MB

What is the copy-up?

Assuming that merge_dir is a new, empty directory created specifically for OverlayFS mount.

lower_dir	**upper_dir**	**merge_dir**
• a1.txt 5 MB • b1.txt 10 MB • c1.out 15 MB	• a.txt 1 MB • b.txt 2 MB • c.out 3 MB	• a.txt 1 MB • a1.txt 5 MB • b.txt 2 MB • b1.txt 10 MB • c.out 3 MB • c1.out 15 MB

Mounting example: mount -t overlay overlay -o\
lowerdir=/lower,upperdir=/upper,workdir=/work /merged

What is the copy-up?

Now, suppose a logged-in user launches any command that modifies the metadata of files (for instance, chmod) in the merge directory.

lower_dir	**upper_dir**	**merge_dir**
-- lower_dir -- a1.txt 5 MB -- b1.txt 10 MB -- c1.out 15 MB	-- upper_dir -- a.txt 1 MB -- b.txt 2 MB -- c.out 3 MB -- c1.out 15 MB (full copy)	-- merge_dir -- a.txt 1 MB -- a1.txt 5 MB -- b.txt 2 MB -- b1.txt 10 MB -- c.out 3 MB -- c1.out 15 MB

So, c1.out gets fully copied from lower_dir to upper_dir and that's what is called copy-up.

METADATA-ONLY COPYING

Operations that utilize metadata-modifying calls actually do not affect the file content itself. When a user performs such operations during ***cmvfs_server transaction*** they involve accumulating changes in a scratch area (an upper-layer). Hence it might be beneficial to avoid copying a full file to an upper-layer and copy only metadata info instead in terms of speed.

Kernel config option	CONFIG_OVERLAY_FS_METACOPY
Mount option	metacopy
Possible values	{on, off}

Mounting example: mount -t overlay overlay -o\
lowerdir=/lower,upperdir=/upper,workdir=/work,metacopy=on, redirect_dir=on /merged

METADATA-ONLY COPYING. IMPORTANT NOTES

According to OverlayFS documentation note: *redirect_dir={off|nofollow|follow[*]} and nfs_export=on mount options conflict with metacopy=on, and will result in an error.*

What is a filesystem object that gets created on the upper-layer on metacopy-modifying operation?

- Regular file that contains no data (a.k.a sparse file) with an xattr “trusted.overlay.metacopy” that is used as an indication that this upper file contains no data and that data copy-up should still be performed before the overlayfs file is opened for write

- If you read such a file it will contain all zeroes, but those zeroes are not actually stored on disk, so doing "du $METACOPY_FILE" should yield zero. Such objects could be created with the truncate(1) utility

Example: dh -h output from an upper-layer file with metacopy-only option set

0 $HOME/ovlfs_tuned_files/upper/seeded_file_1

ZERO-COPY RENAMING

Any operations that change directory name lead to copying the whole directory with its subtree to the upper layer. So, we are dealing with the same unnecessary copy-up problem that influence operations during ***cvmfs_server transaction***.

Kernel config option	OVERLAY_FS_REDIRECT_DIR
Mount option	redirect_dir
Possible values	{on, off, follow, no_follow}

A mount with redirect_dir=on creates a zero-sized filesystem object on the upper layer after renaming.

ZERO-COPY RENAMING. IMPORTANT NOTES

- Renaming is not allowed by default through rename(2)
- mv(1) works by default
- After renaming a whiteout with the old name is created in the upper-layer
- “Zero-copied” directory has extended attribute trusted.overlay.redirect that holds the old name of the renamed directory

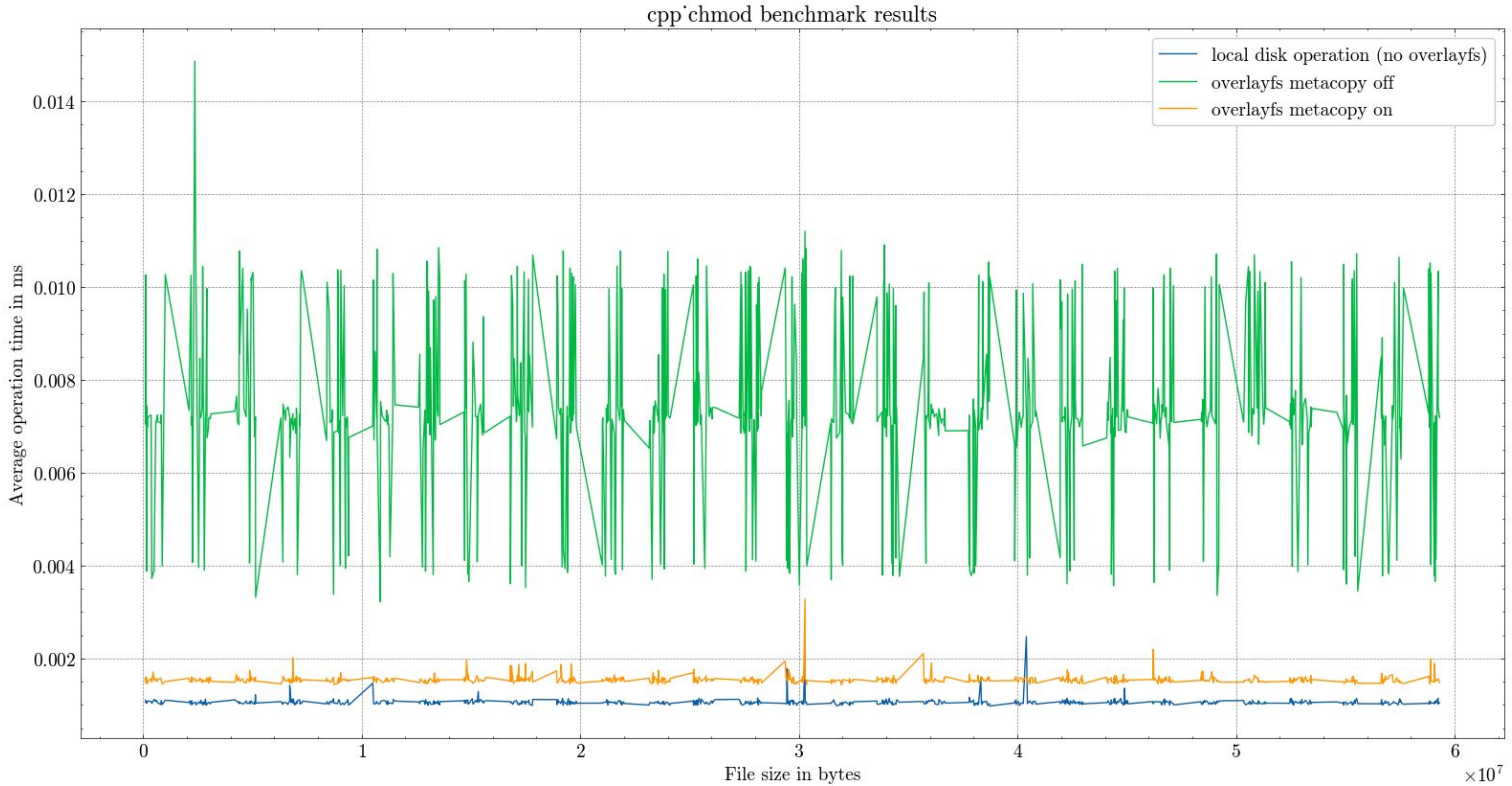
Benchmarking setup

Instance type	OS	Kernel	Hardware memory type:	RAM	Processor
Physical	Alma Linux 9.3 (Shamrock Pampas Cat)	5.14.0-362.18.1.el9_3.x86_64	SSD (~874 GB of storage available)	16 GB	Intel i7-10750H (12) @ 5.000GHz
Virtual Machine	Ubuntu 22.04	5.14.0-362.18.1.el9_3.x86_64	SSD (~120 GB of storage available)	8 GB	Intel i7-10750H (1) @ 2.592GHz
Virtual Machine	Alma Linux 8.9 (Midnight Oncilla)	4.18.0-513.5.1.el8_9.x86_64	SSD (~120 GB of storage available)	8 GB	Intel i7-10750H (1) @ 2.592GHz

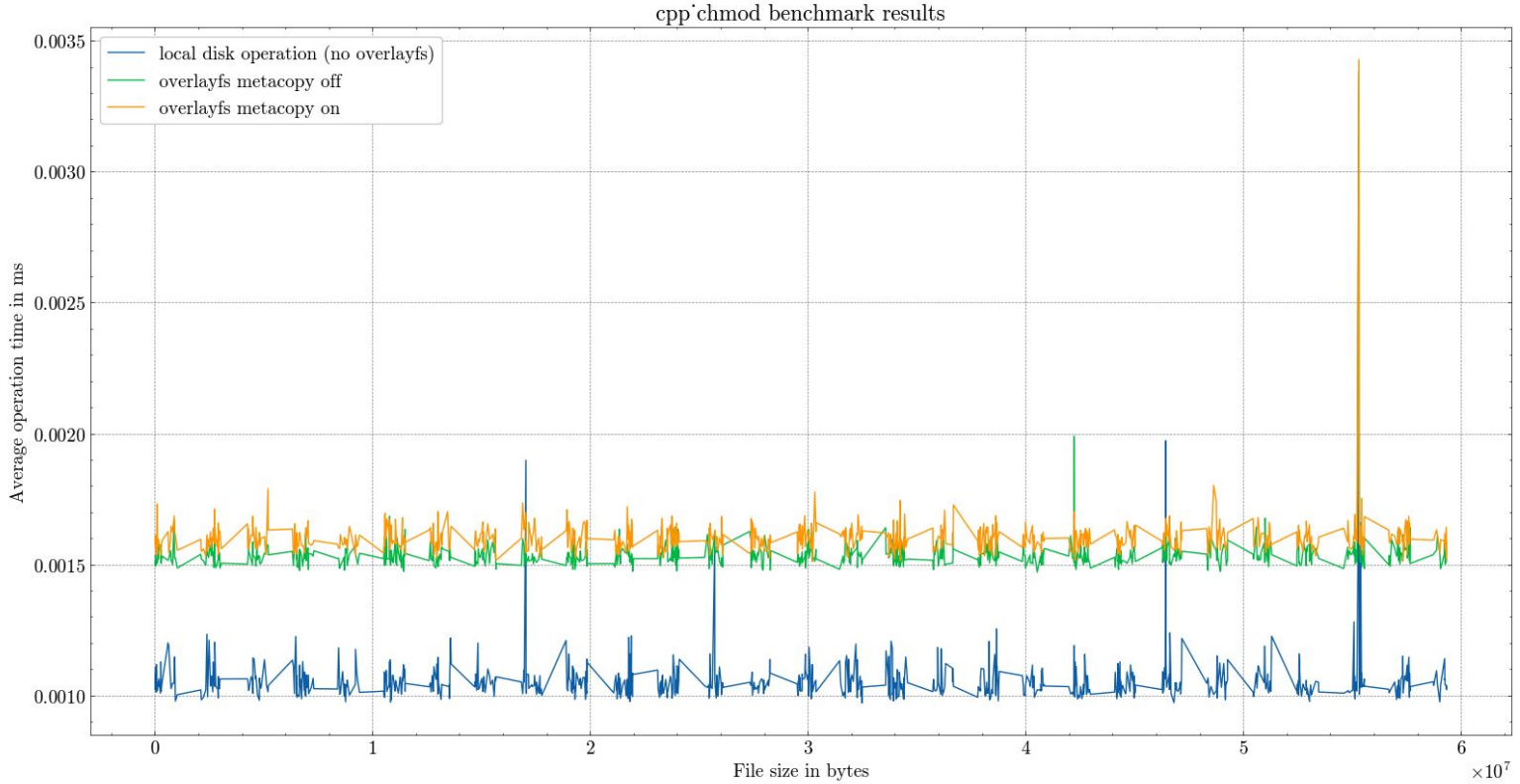
METACOPY BENCHMARKING SETUP

- Three directories were created in the local filesystem: lower directory for a tuned OverlayFS, the directory regular OverlayFS (without any additional configuration parameters), and the directory that doesn’t belong to any OverlayFS setup (a base directory)
- 800 files with sizes in range (0KB; 62MB)
- 1000 operations per each file in a directory (800 000 operations in a benchmark run)
- Measured with std::high_resolution_clock in milliseconds

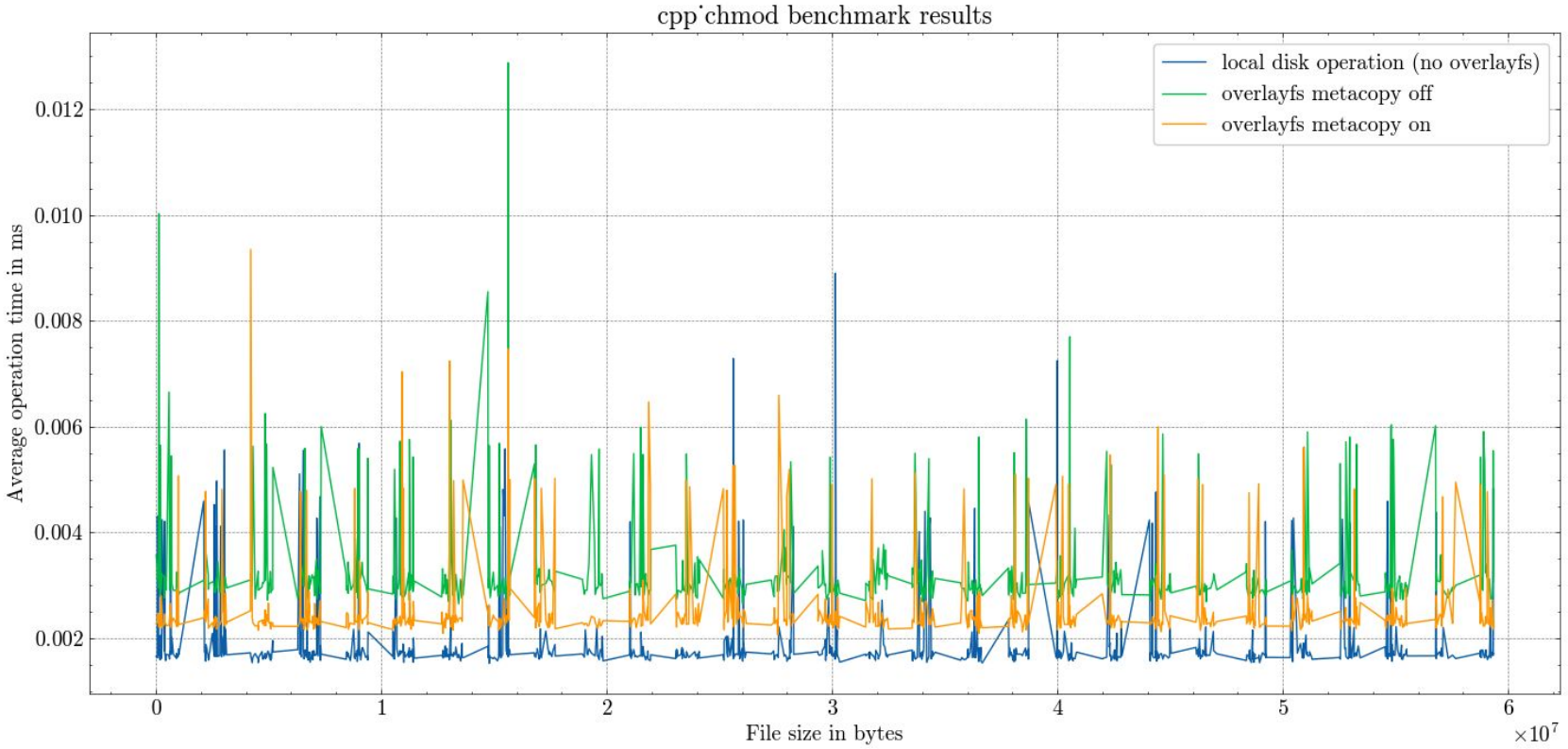
METACOPY BENCHMARKING PLOTS. ALMA LINUX 9



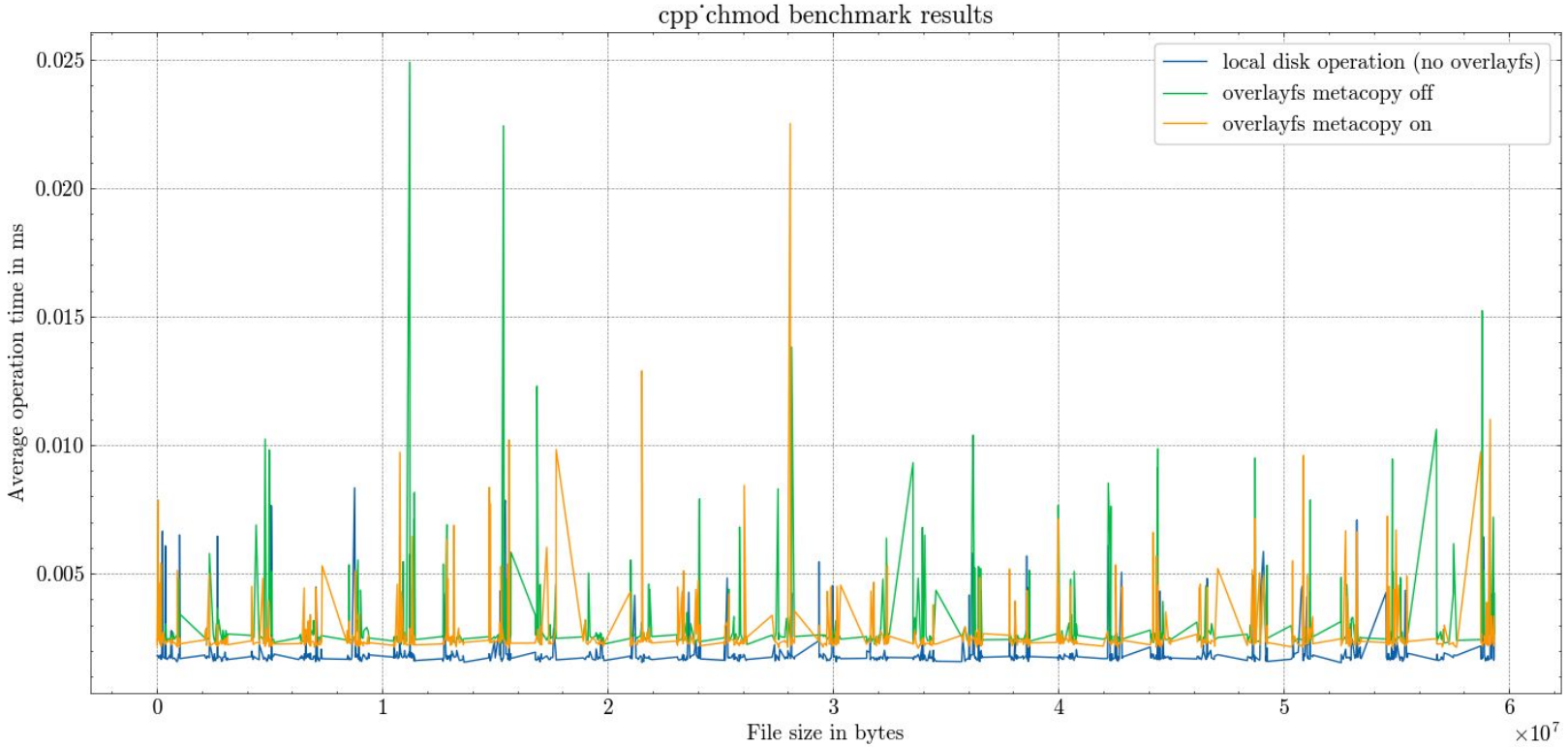
METACOPY BENCHMARKING PLOTS. ALMA LINUX 9



METACOPY BENCHMARKING PLOTS. ALMA LINUX 8



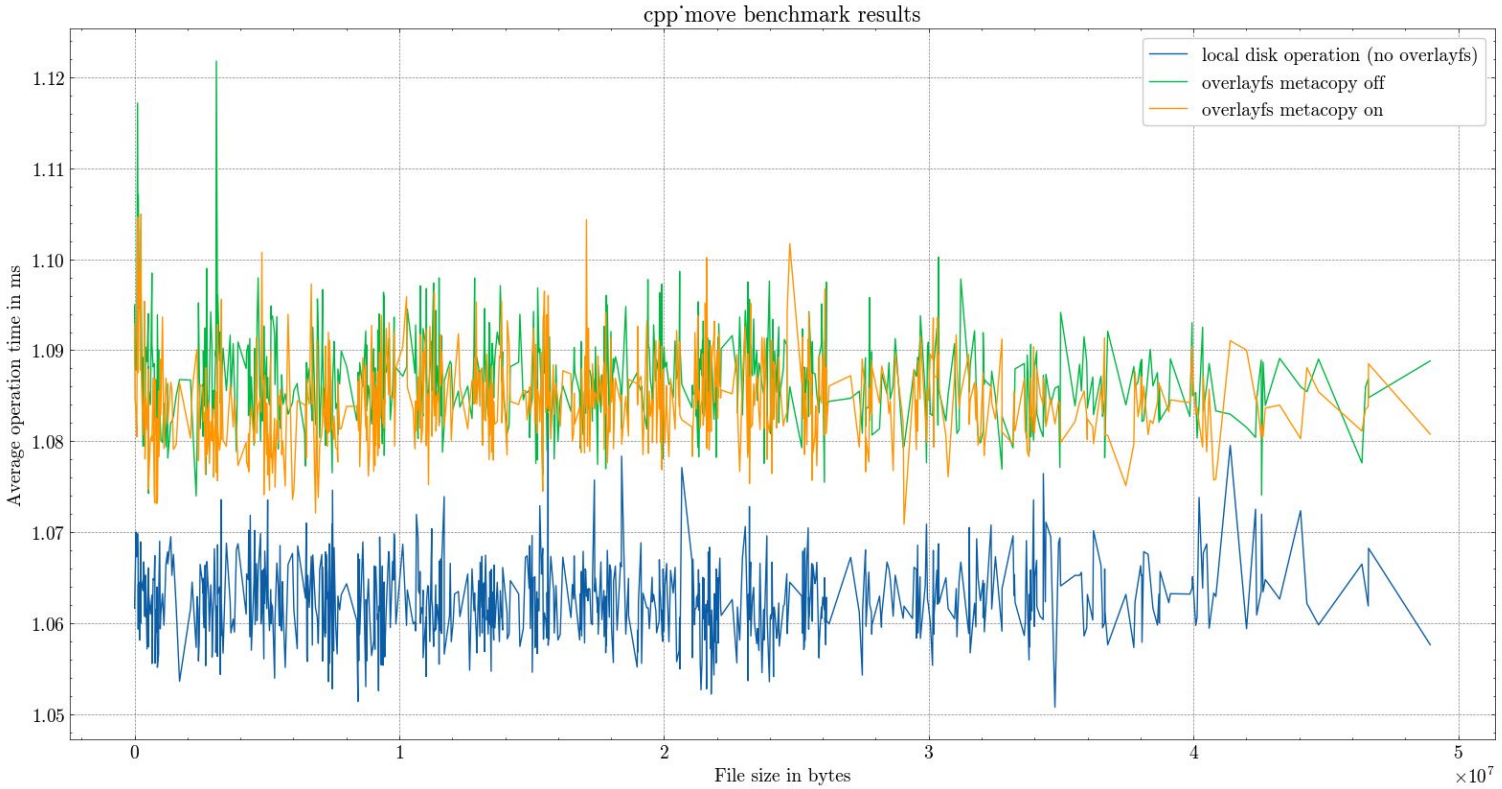
METACOPY BENCHMARKING PLOTS. ALMA LINUX 8



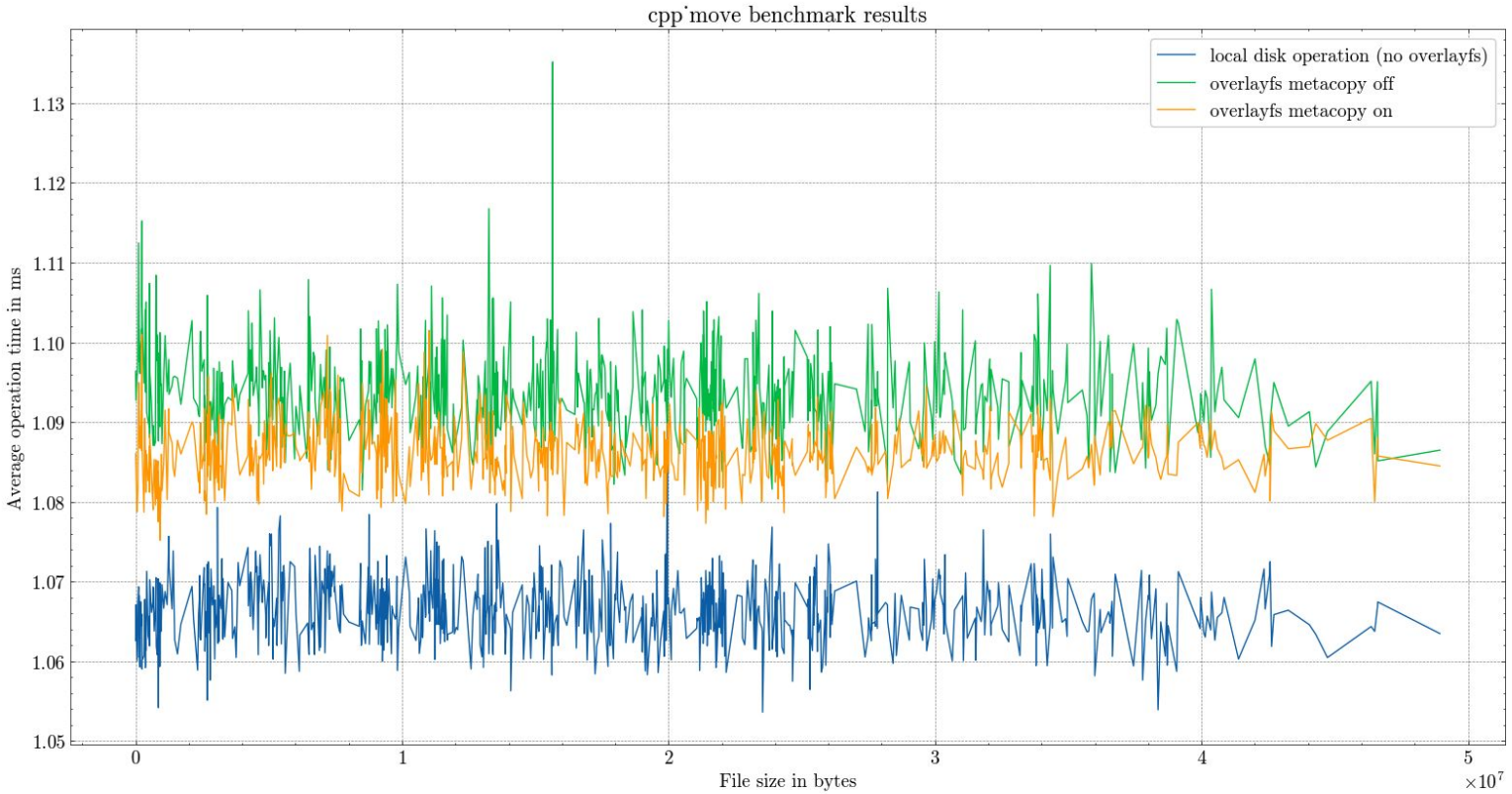
ZERO-COPY RENAME BENCHMARKING SETUP

- Three directories were created in the local filesystem: lower directory for a tuned OverlayFS, the directory regular OverlayFS (without any additional configuration parameters), and the directory that doesn't belong to any OverlayFS setup (a base directory)
- 800 directories with sizes in intervals:
  - (0KB; 50MB] for Alma Linux 9
  - (0KB; 60MB] for Ubuntu and Alma Linux 8
- 800 000 calls for mv(1) operation
- Time of each call was measured with std::high_resolution_clock in milliseconds

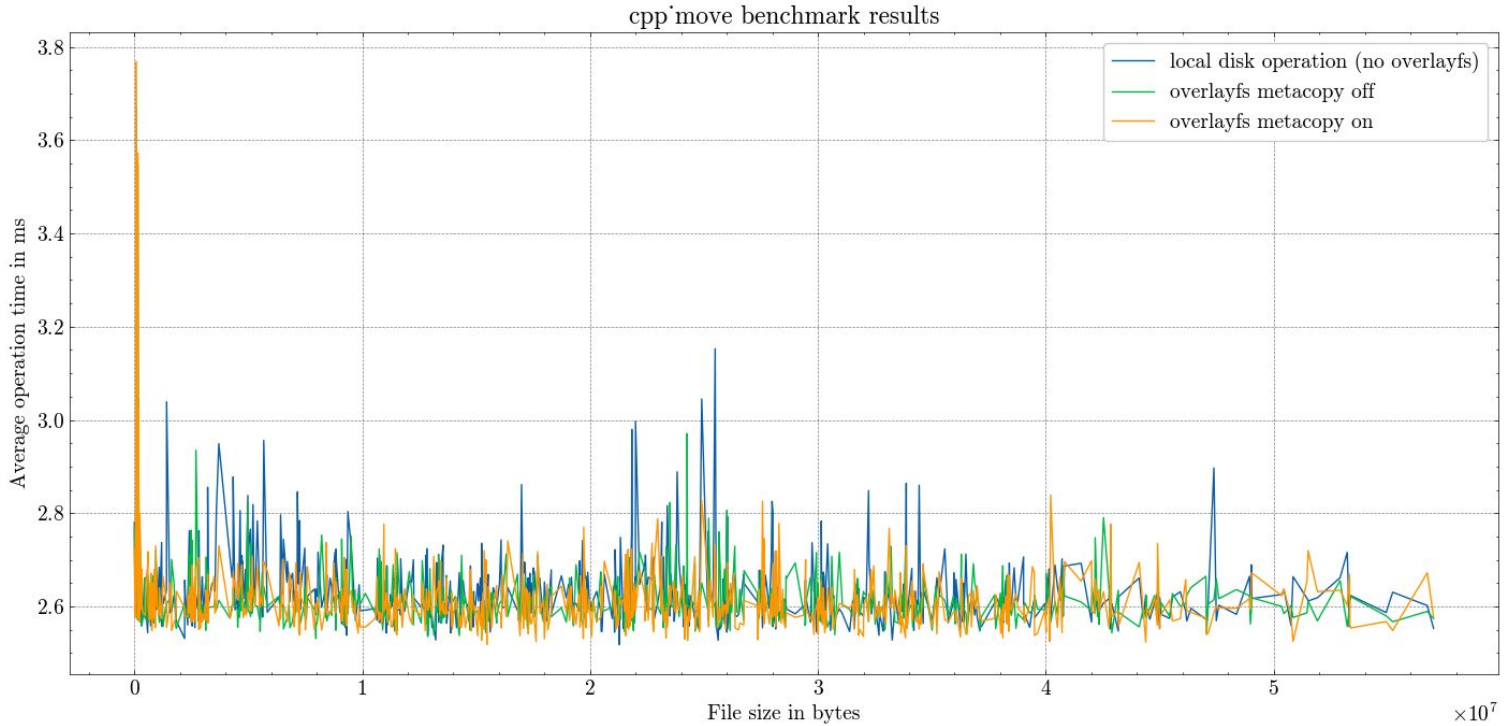
ZERO-COPY RENAMES BENCHMARKING PLOTS. ALMA LINUX 9



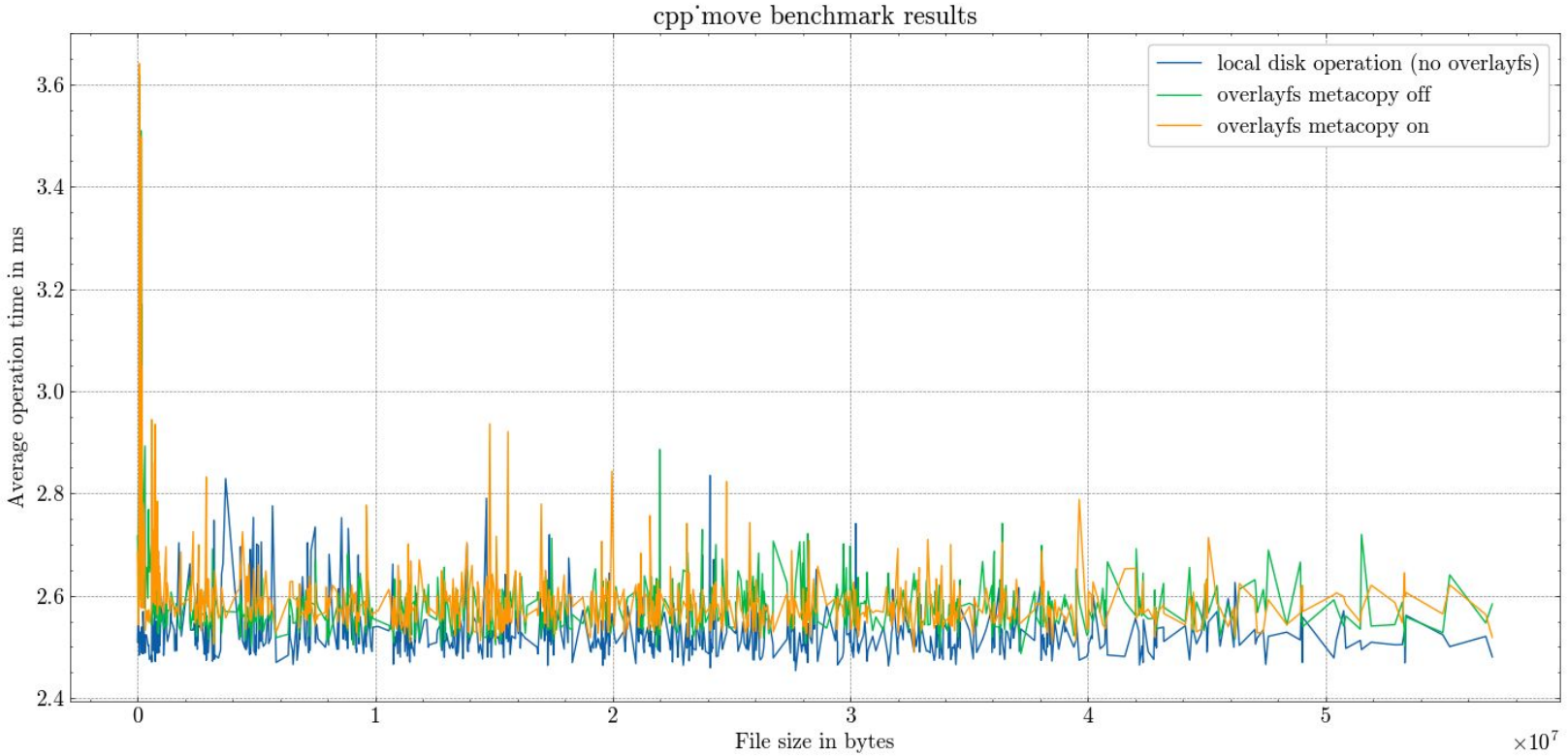
ZERO-COPY RENAMES BENCHMARKING PLOTS. ALMA LINUX 9



ZERO-COPY RENAMES BENCHMARKING PLOTS. ALMA LINUX 8



ZERO-COPY RENAMES BENCHMARKING PLOTS. ALMA LINUX 8



OverlayFS conclusions

- Only a marginal increase in the performance on both the physical device and virtual machine running Alma Linux
- From the note on OverlayFS official documentation: *Once the copy_up is complete, the overlay filesystem simply provides direct access to the newly created file in the upper filesystem - future operations on the file are barely noticed by the overlay filesystem (though an operation on the name of the file, such as rename or unlink, will, of course, be noticed and handled);* what possibly explains the difference between first and subsequent runs of the benchmark
- For the redirect_dir option only, there is no significant performance improvement on basic local tests, however, we expect the speed-up on real-world use-cases for ***cvmfs_server publish*** as this feature has the potential to reduce amount of data to be hashed, compressed and stored in CVMFS content-addressable storage
- The ongoing work is aimed at the full integration of both features in CVMFS and benchmarking with extended metrics on real (or close to real use-cases) data (https://github.com/cvmfs/cvmfs/pull/3547). Apparently, just setting these options on mount for CVMFS is not enough for the full integration and requires further tweaking the core logic of the cvmfs_server.

Google Summer of Code and FUSE-T. Motivation

- Currently CVMFS client for macOS fully relies upon MacFUSE module which is implemented as a kernel extension (kext)
- Apple explicitly stated that kexts are going to be deprecated in the near future
- kext requires reducing startup security level on the end-user's side (which could be done only in recovery mode) and several reboots

Google Summer of Code and FUSE-T. Overview

- FUSE-T is a user-space library (https://www.fuse-t.org/)
- **Claimed** to be a drop-in replacement for macFUSE
- Utilizes local NFSv4 Golang server that works through a connection with the implemented filesystem process and libfuse
- **Claimed to be more efficient than macFUSE**

Google Summer of Code and FUSE-T Progress

The progress could be tracked in my PR: https://github.com/cvmfs/cvmfs/pull/3587

What has been already done:

- CVMFS build update to overcome issues with dyld failing to find dynamic libraries
- Updated CMake build files use FUSE-T
- Updated FUSE-T installation check
- Achieved integration tests passing on a reduced tests set
- Extended GitHub Actions pipeline with updated macOS CI support

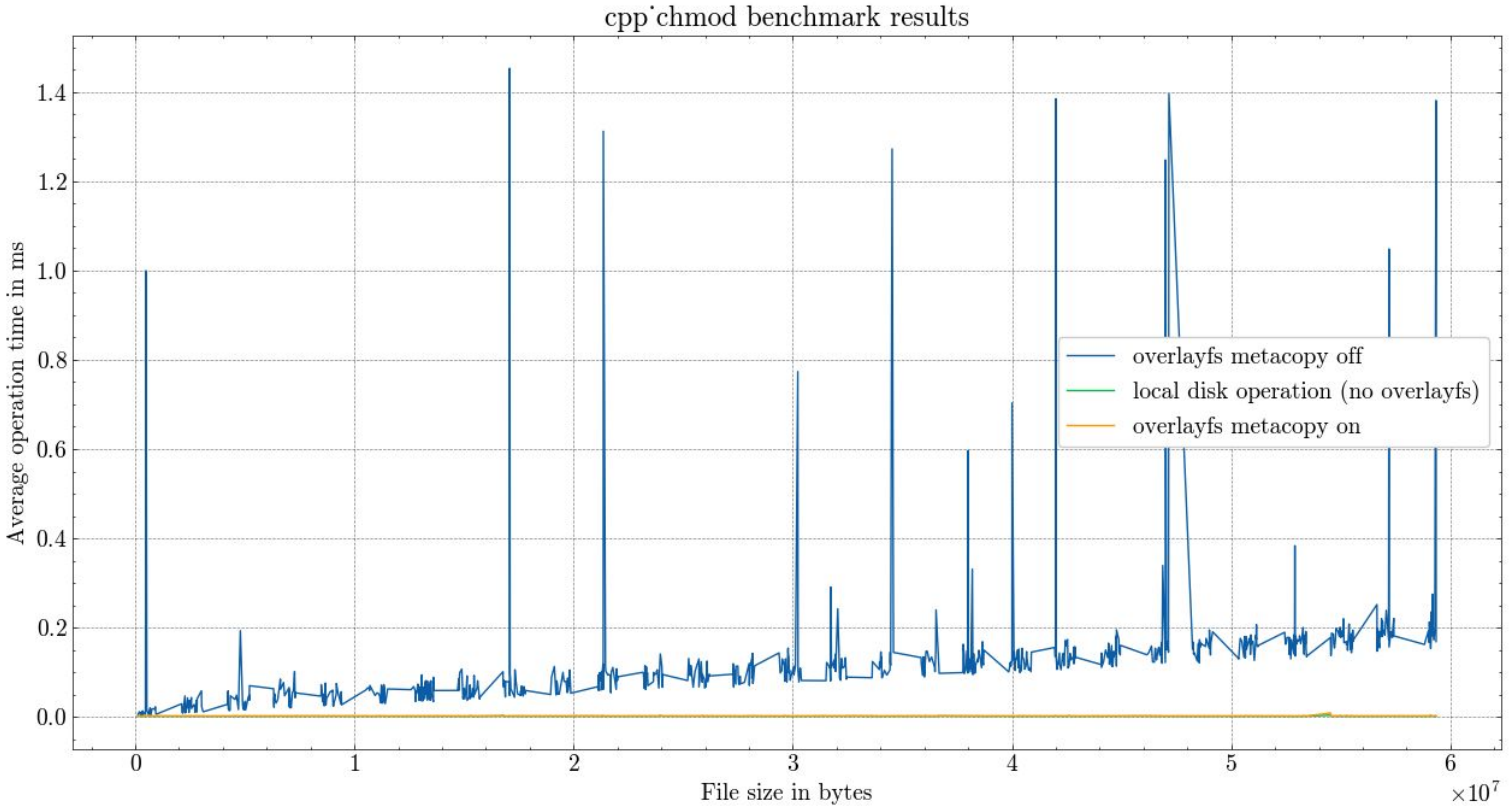
Google Summer of Code and FUSE-T. Issues

- Mounting takes a time window (usually a few seconds) long enough to fail immediate subsequent commands (such as calling ***ls <repo>*** right after mount)
- Extended attributes behave unexpectedly and oddly. Tests fail to retrieve xattrs while that could be clearly done in terminal
- Killing cvmfs processes lead to repositories unmounting
- Hidden attributes are not supported; not a big issue since they are utilized by cvmfs_server (which is not supported on macOS)

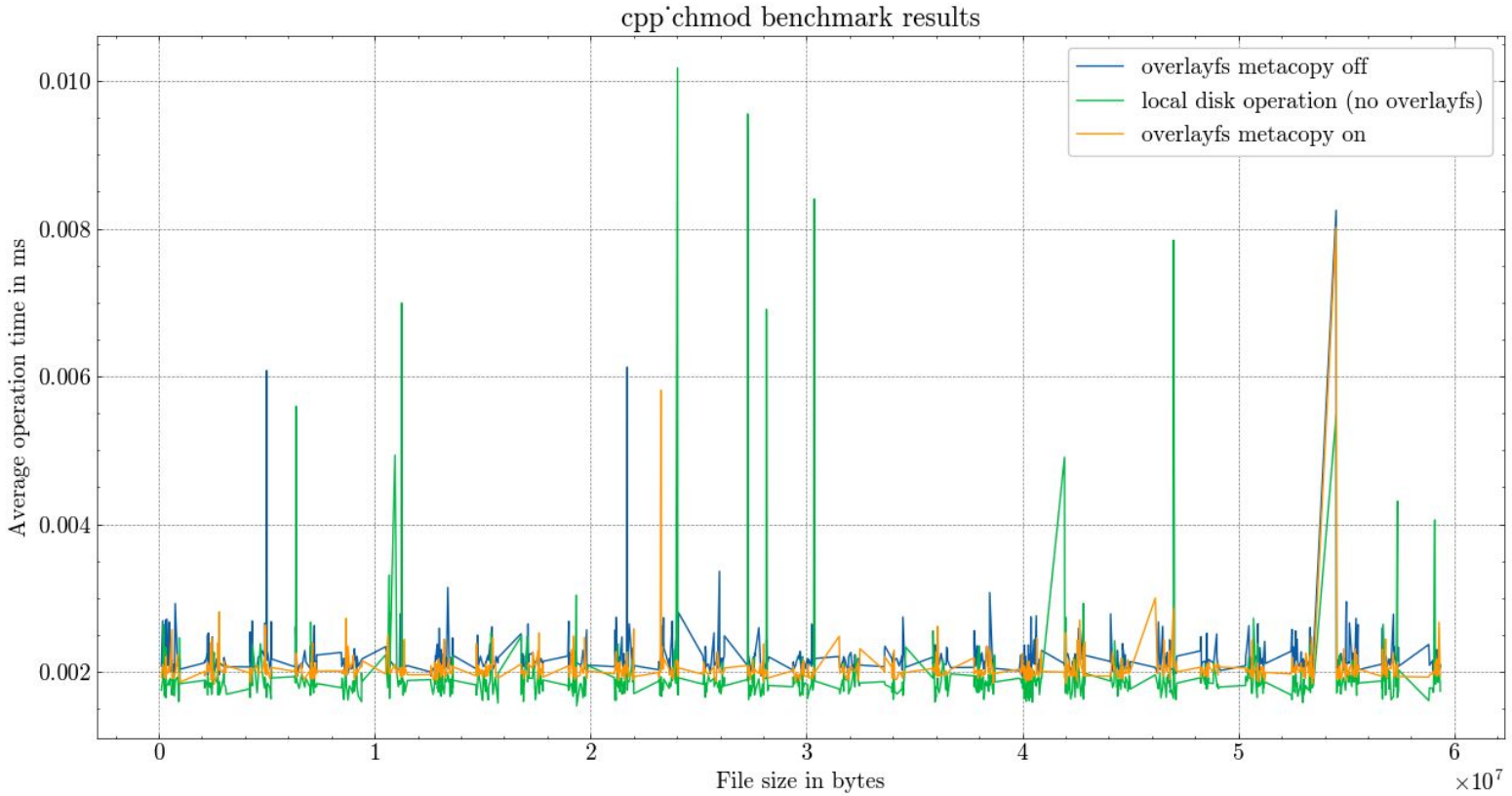
THANK YOU!

BACKUP SLIDES

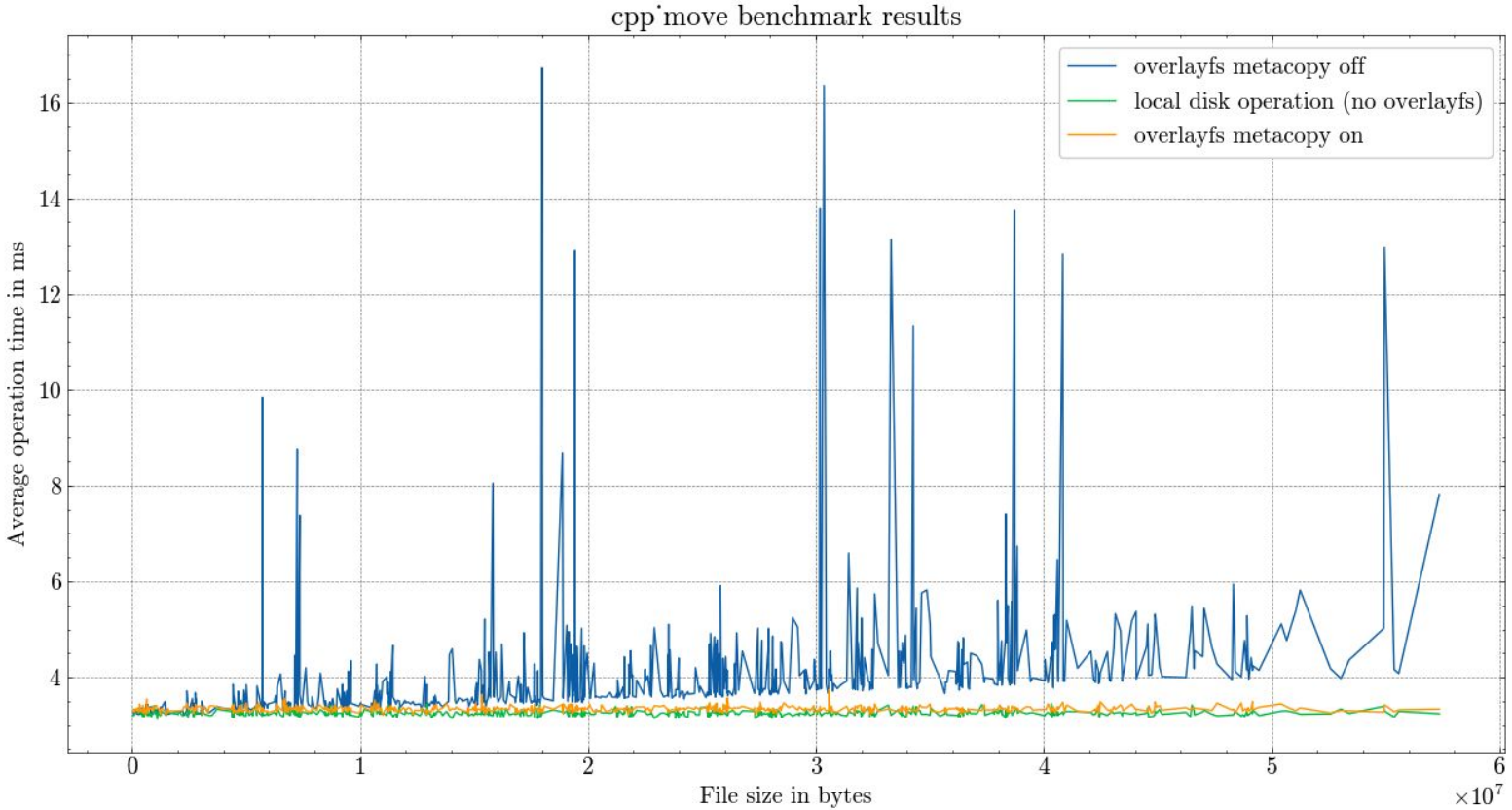
METACOPY BENCHMARKING PLOTS. UBUNTU 22.04



METACOPY BENCHMARKING PLOTS. UBUNTU 22.04



ZERO-CO BENCHMARKING PLOTS. UBUNTU 22.04



ZERO-CO BENCHMARKING PLOTS. UBUNTU 22.04