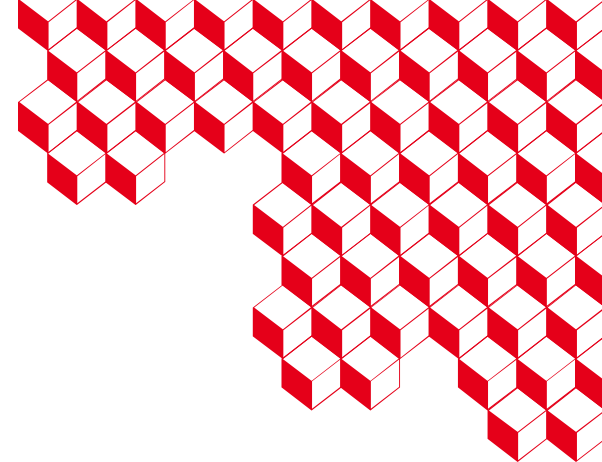




Migrating a Lustre-HSM filesystem (easily)

copy_tree

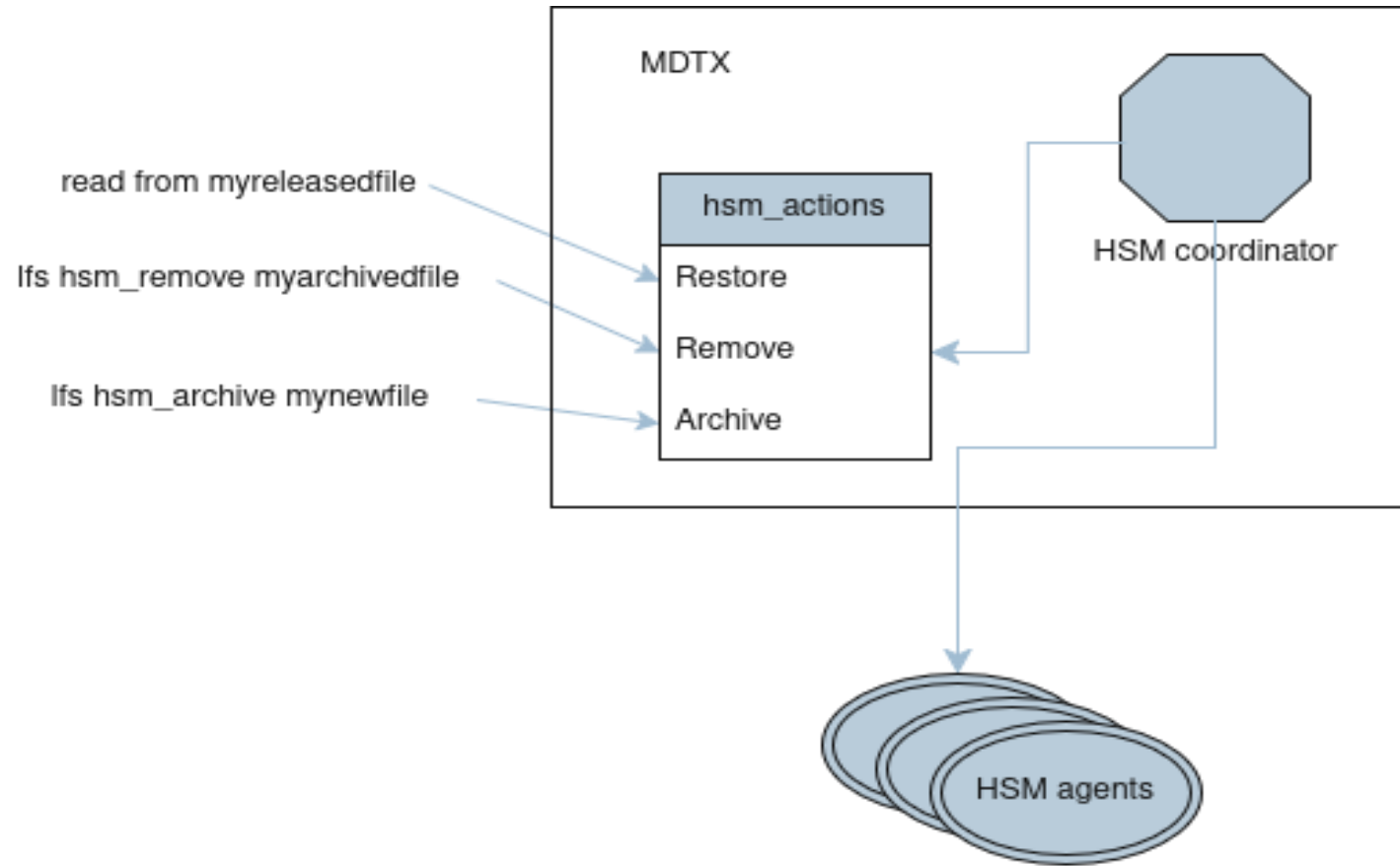


1. Goals

Goals

- **TGCC Store:** Hardware migration from a filesystem to another
 - The old hardware had to be removed by the end of January 2026
 - New filesystem was ready by the end of November 2025
- **The goals :**
 - Most of the files must be copied by the beginning of January
 - Have the smallest service interruption for users
 - Side effect: move from 2.12 to 2.15 (the last filesystem to migrate)
- **The challenge :** The filesystem uses HSM
 - Total volume stored 130PB (Lustre+archive)
 - 12PB in Lustre

What is Lustre HSM?



The tools

- Migrations cannot be done with *dsync* (MPIFileUtils) due to the use of HSM
 - The HSM state is not checked : this would trigger restores for released files
 - The files' HSM state is not copied properly into the target
- Two solutions :
 - Improve *dsync* to support HSM
 - Reuse existing internal tool *copy_tree* to do the migration

The strategy

- It was decided to stick to *copy_tree* given the time constraints
 - Some limitations of Lustre made the implementation in *dsync* tricky (more on that later)
- **Problem** : *copy_tree* does not support online copies while users have access to their files
 - It can only do a single full copy of the user's file tree, no updates after



2. Challenges with HSM

HSM state management (1)

- ***trusted.lhsm_fuid***:
 - Copytool specific xattrs which stores the file's identifier in the HSM backend
 - Without this, files cannot be restored
 - Contains *fsname:fid*
 - Since the FS name and FID change when migrating filesystems, this xattr ensures that files archived on the old system can still be restored on the new
 - Extra care must be taken to ensure that this xattr is copied properly between the filesystems
 - Using an xattr avoid contacting the HSM backend when we do these kinds of migrations

HSM state management (2)

■ Data version :

- All files in Lustre have a data version which changes when the file is modified
 - HSM stores the data version in *trusted.hsm* when the file is archived
 - This is called the archive version
 - If the data version and archive version do not match, Lustre will refuse to release a file
 - **Problem** : there were no API to update the archive version in 2.15
 - There is a custom patch in CEA's Lustre version to allow root to call *setxattr* on the HSM xattr
 - This makes *copy_tree* only work on CEA's Lustre version...

HSM state management (3)

■ Released files :

- These files must not be read, otherwise Lustre will try to restore them
 - Use of *llapi_hsm_import* function from the C API
 - Recreates a file in the released state with the same metadata as the source
 - If *trusted.lhsm_fuid* is set properly, the copytool will know which data is associated to this file



3 ■ *copy_tree* **improvements**

Online updates

- A few improvements were needed
 - **Check mtime and size :**
 - Heuristic to detect modifications between copies
 - Delete the destination and copy again if necessary
 - **Lease lock on the source inode :**
 - Files can be updated while *copy_tree* is working
 - Concurrent updates will break the lease lock, then *copy_tree* aborts the copy
 - **Metadata updates :**
 - Owner, group, permissions, xattrs, HSM state... can change between migrations
 - **Sparse copies :**
 - using SEEK_HOLE/SEEK_DATA backported in 2.12

Hardlinks management (1)



- Concurrent processes can work on the same inode from different links
 - **Problem** : two processes working on the same inode will create different inodes in the destination
 - some hardlinks may be lost in the migration process
 - **Solution** : list all the hardlinks first by reading *trusted.links*, then always start by creating the first one of the list
 - If two processes are working on the same inode, they will both try to create the same link, only one will succeed, the other one will stop
 - The one which wins the race will take an *flock* on the destination
 - Any process working from a different link will take the *flock* on the same link and fail

Hardlinks management (2)

- *trusted.links* may not contain all the hardlinks
 - **Problem** : there is a maximum size for xattrs since we didn't have the *large_ea* feature in 2.12
 - *trusted.links* may not contain all the hardlinks
 - **Solution** : check whether the link being processed belongs to the *trusted.links* list
 - If not, create the link as well.
 - There are two cases :
 1. The process has the *flock*, it creates all the links of *trusted.links* then itself
 2. Another process has the *flock*, it only creates this link

HSM state management

- **Updates** : files can become :
 1. **Dirty** : simply copy the new data, Lustre will mark the target file dirty automatically
 2. **Released** : simply release the file in the destination
 3. **HSM flags (noarchive, norelease...)** : simply add/remove the flags to/from the destination
 4. **Restored** : we don't want to trigger a restore here.
 - The data has already been loaded into Lustre (and the target filesystem is not yet linked to the HSM backend anyway)
 - In this case, delete the file in the target and redo the copy

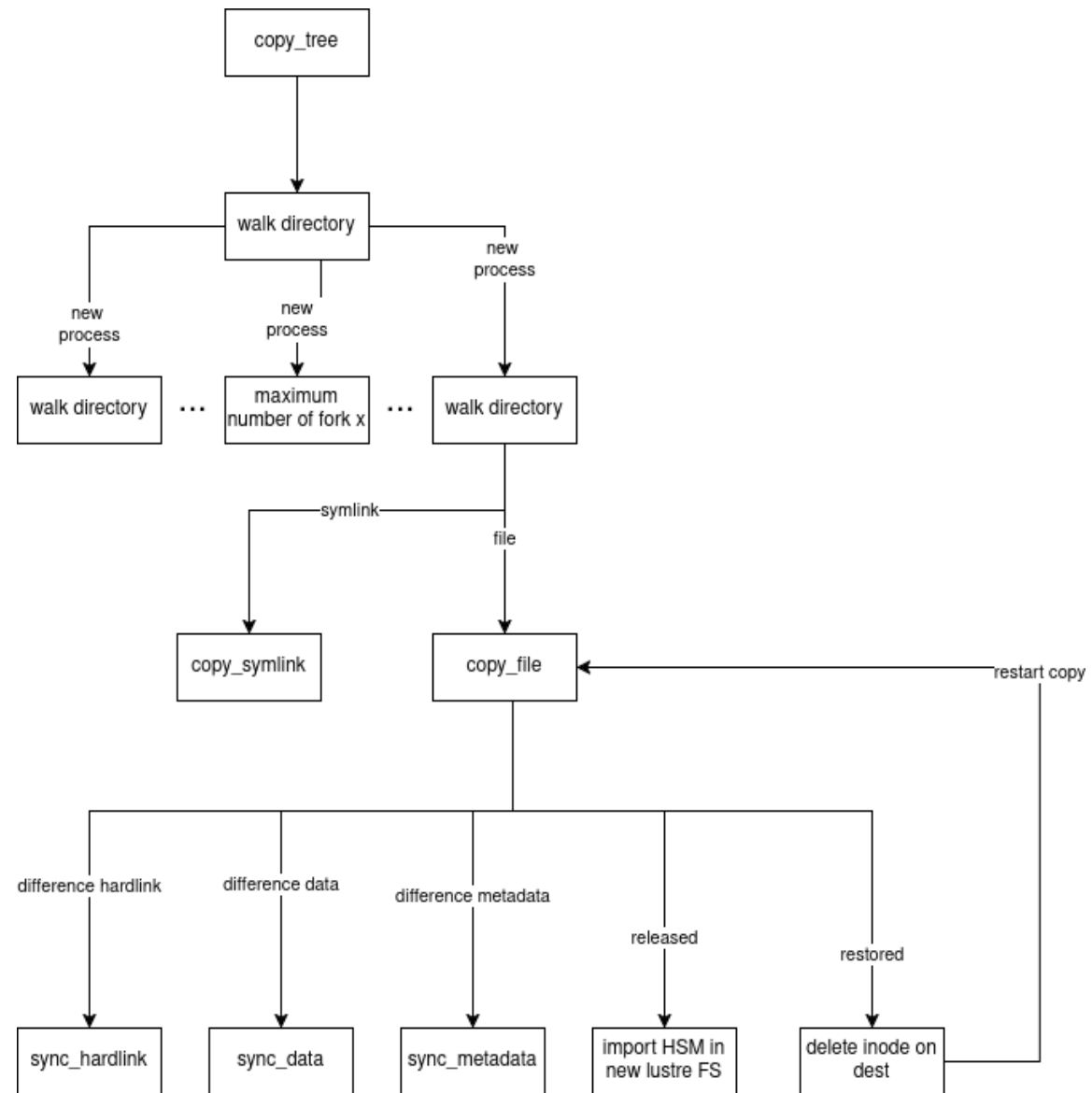
Manage removed files

- *copy_tree* only scans the source
 - It doesn't know which files are removed
 - We rely on *dcmp* to find these files and *drm* to delete

Parallelism

- *copy_tree* only scans inodes sequentially
 - It can copy big files asynchronously but small files are copied synchronously
- Multiple instances were run in parallel, distributed across user subdirectories
 - **Problem** : some directories where almost empty, others where very big
 - **Solution** : a very basic task scheduler was developped to distribute batches of directories to a set of nodes
 - As soon as a node finished, another batch was sent
 - Very big files still created parallelism issues but the load distribution was much better overall

Decision tree



The results

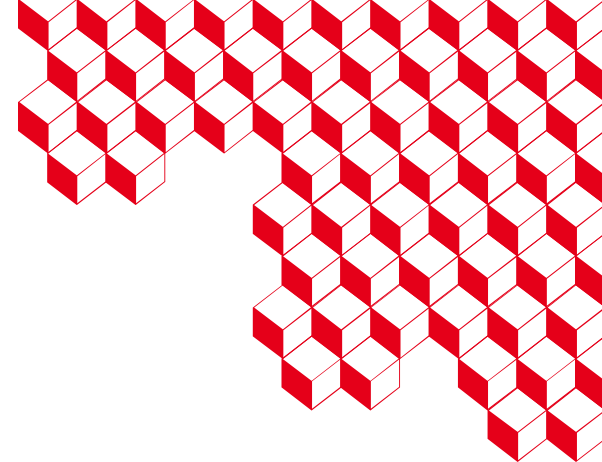
- *copy_tree* ran for about a month on 24 nodes for continuous resync
- Two 3-day interruptions to do a final offline copy and *dcmp+drm*
 - Each user only had a single 3-day interruption
- (almost) no files lost in the process



4. **What's next?**

What's next?

- Implement the HSM logic into MPIFileUtils
 - Lustre now has an API to update the archive version (LU-17863)
 - In principle, nothing prevents MPIFileUtils from implementing the same features as *copy_tree*
 - This would remove the need for the external task scheduler
 - We may also need to add parallel file copies for huge (+1TB) files which created a lot of bottlenecks



Any questions?

guillaume.courrier@cea.fr