



Lustre HSM

Nathan Rutman
SC09
Portland, OR



Goals

- Scalable HSM system
 - > No scanning
 - > No duplication of event data
 - > Parallel data transfer
- Interact easily with many HSMs
- Focus:
 - > Version 1 – primary goal deliver ASAP
 - > Version 2 – primary goal best feature set around
- bz15599

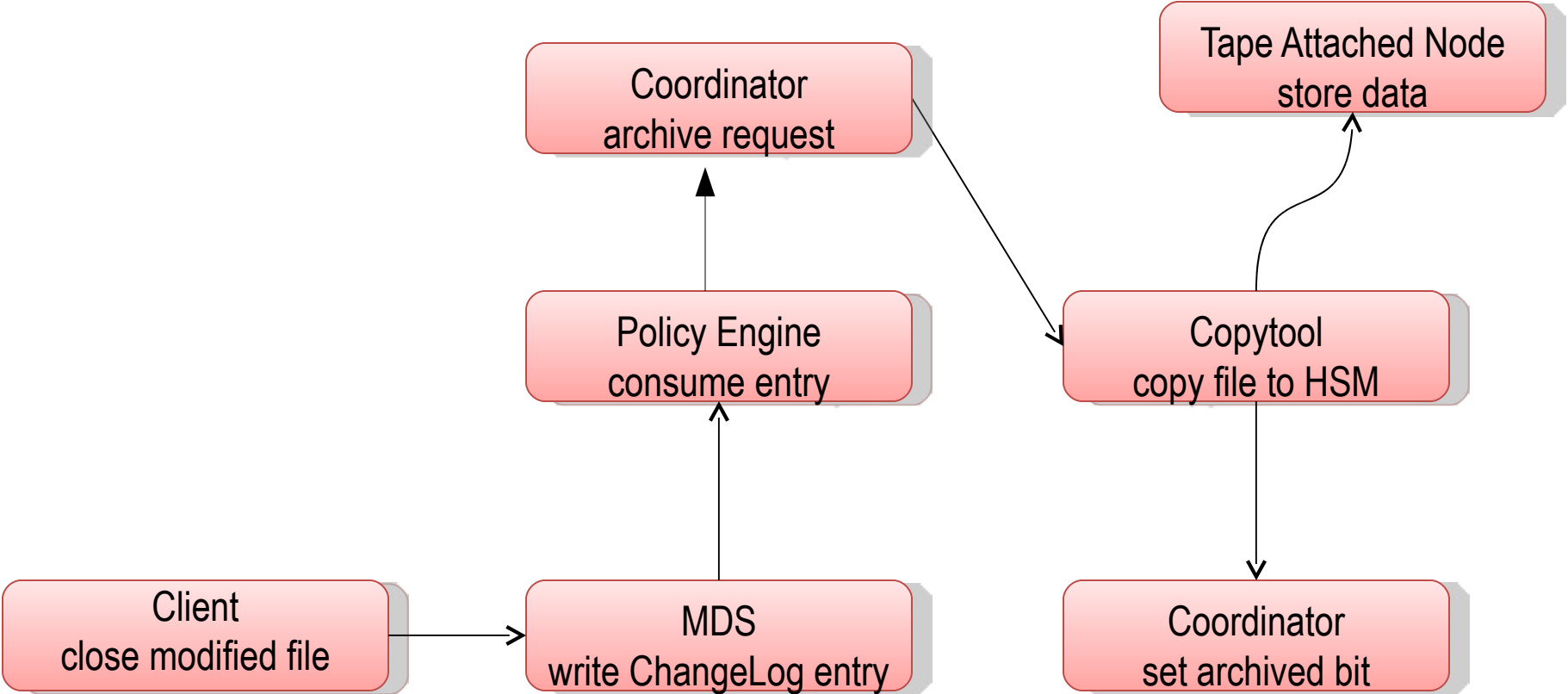
Features

- V1 Main Features:
 - > Whole file migration
 - > Copy data back when layout is requested
 - > External PolicyEngine decides what to archive, what to release from Lustre, and when
 - > File aggregation possible if PolicyEngine and Copytool support it.
- V2 Plans:
 - > Partial data released. Keep arbitrary extents on OSTs.
 - > Restore data only when needed (read, sub-stripe write)

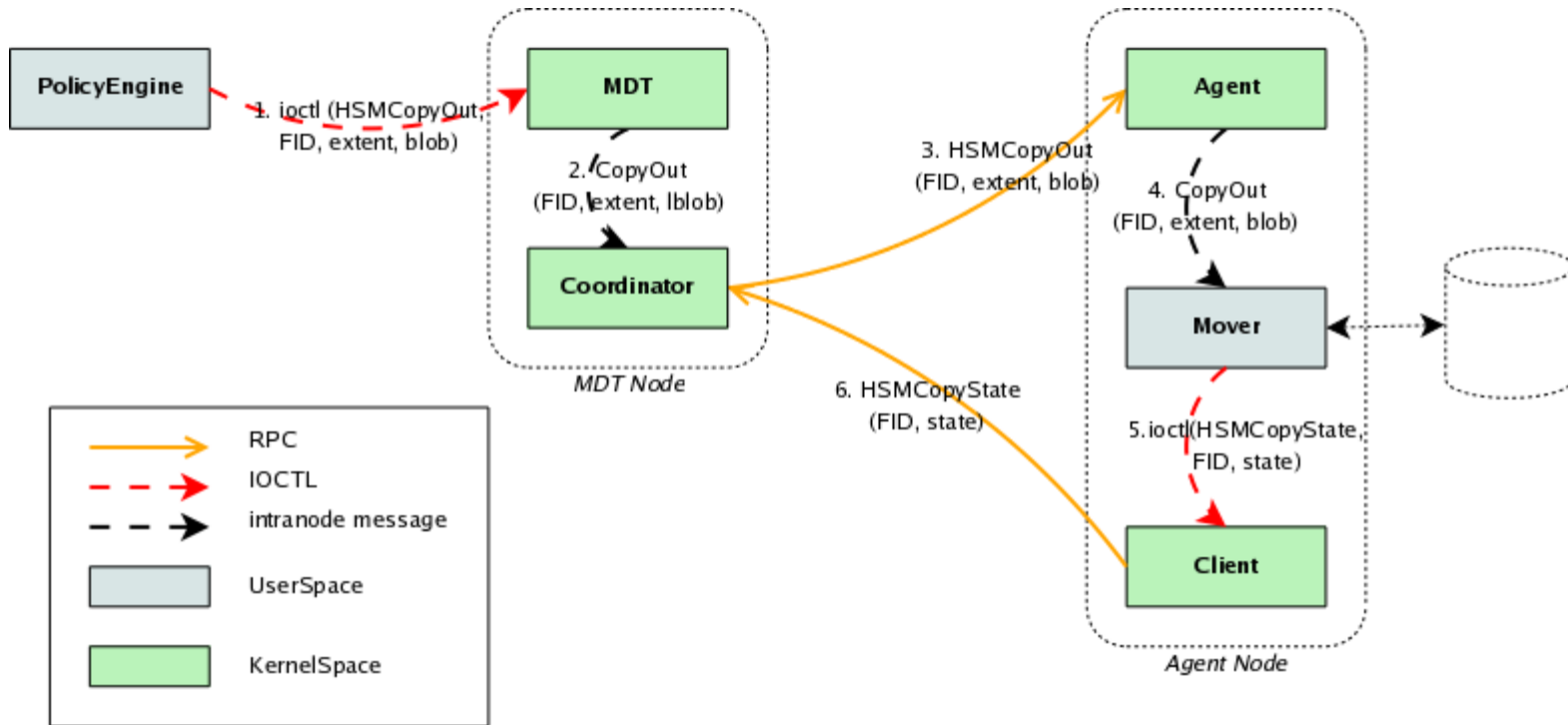
Components

- Policy Engine bz20693
- Coordinator bz20234
- MDT (LMA bz19669, layout lock bz13183)
- Copytool bz20334
- HSM backend

Archive modified files



Archive Modified Files



Policy Engine requests a copyout

PolicyEngine: Robinhood (bz20693)

- Robinhood is a filesystem monitoring, archive, and release tool.
- Project
 - > GPL-compatible, CEA development.
 - > Website: <http://robinhood.sf.net/>
- Main features
 - > Gather filesystem information from scans (POSIX scan) or event-based (Lustre 2.0).
 - > Trigger file releases depending on policy rules.
 - > Trigger file archiving depending on policy rules (for Lustre HSM).
 - > Policies based on file attributes and thresholds

Robinhood: Policy examples

- Based on file attributes like: path, name, time, size, ...
- Rules can be combined with boolean operators
- Files can be white-listed
- Simple example:

```

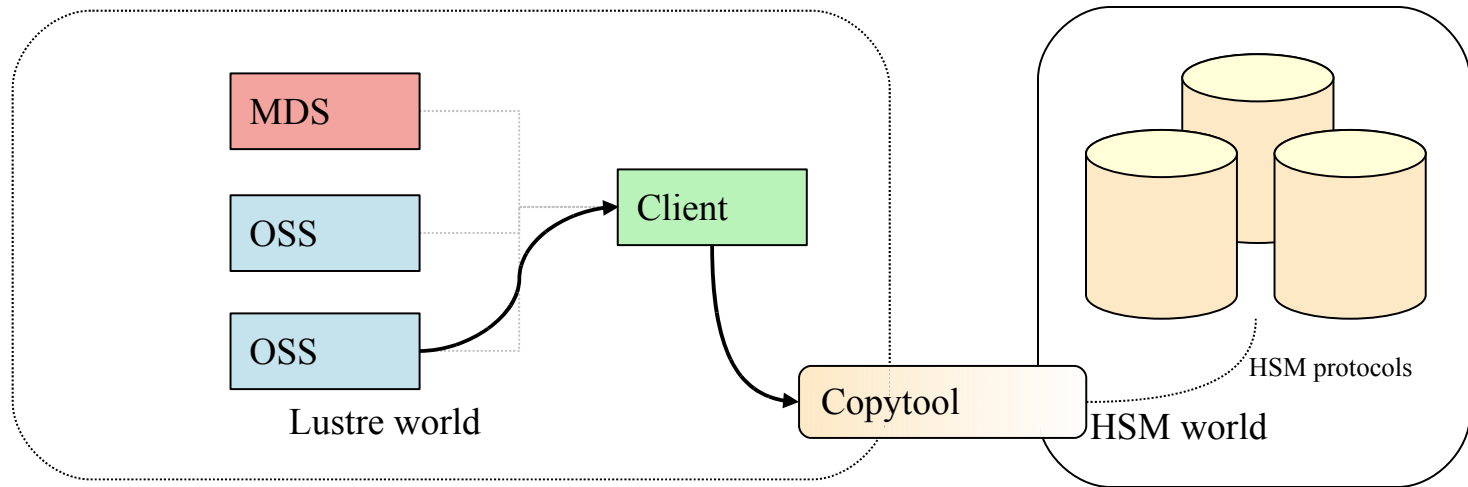
Purge_policy
{
    Trigger
    {
        trigger_on = OST_usage
        high_watermark_pct = 85%
        low_watermark_pct = 80%
    }
    "
    Whitelist {
        last_access < 1h
        or
        owner = root
    }
}

```

Copytool (bz20334)

- Userspace
- Interface Lustre and the HSM
- Know how to communicate with a specific HSM
 - > Map FID to HSM identifier
 - > Bulk data transfer
- llapi registration call informs coordinator

Architecture: Archive

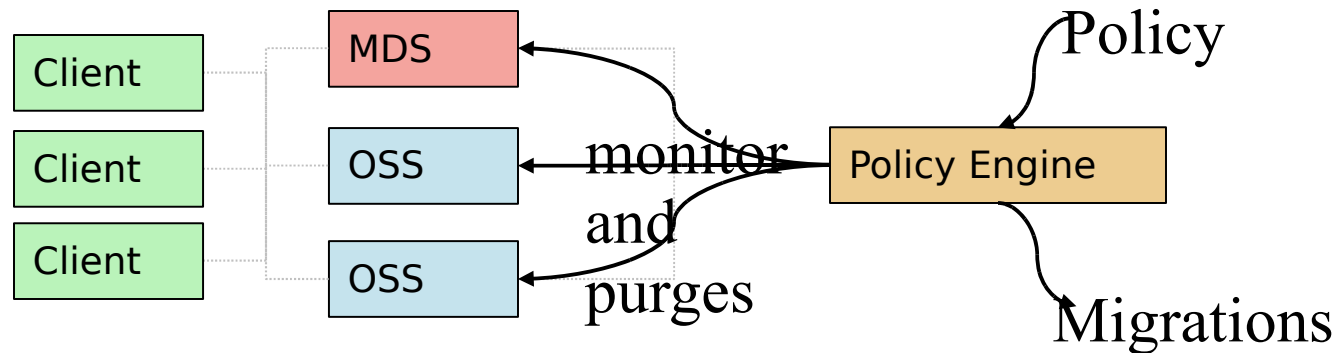


- Policy engine decides what and when to archive
- Coordinator assigns archive actions to copytools
- Copytools report progress
- Coordinator sets HSM bits in LMA

Archive Protocol

- Policy engine (or administrator) decides to copy a file to HSM, executes HSM_Archive ioctl on file
- ioctl caught by MDT, which passes request to coordinator
- Coordinator dispatches request to copytool. Request includes file extents
- Normal extents read lock is taken by copytool running on client
- Copytool sends "archive begin" status message to coordinator via ioctl on the file
- Coordinator/MDT sets "hsm_exists" bit and clears "hsm_dirty" bit. "hsm_exists" bit is never cleared, and indicates a copy (maybe partial/out of date) exists in the HSM
- Any writes to the file cause the MDT to set the "hsm_dirty" bit (may be lazy/delayed – SOM guarantees eventual correctness.)
- File writes need not cancel archiving, but may be desirable in some situations (V2)
- Copytool sends periodic status update to coordinator via ioctl calls on the file (e.g % complete)
- Copytool sends "archiving done" message to coordinator via ioctl
- Coordinator/MDT checks hsm_dirty bit.
- If not dirty, MDT sets "hsm_archived" bit.
- If dirty, coordinator dispatches another archive request; goto step 3
- MDT adds HSM_archived <fid> record to changelog
- Policy engine notes HSM_archived record from changelog

Architecture: Release



- Policy Engine
 - > Monitor filesystem disk space usage
 - > Pre-migrate aging data
 - > When Lustre needs more space, releases data from files already archived in the HSM
- MDT
 - > Removes file layout under layout lock

Release

- Release is possible if
 - > File is closed
 - > File is not dirty (HSM_dirty)
 - > File is archived in HSM (HSM_archived)
- Protocol
 - > MDT takes PW layout lock
 - > MDT takes full file extents lock, drops extents lock
 - > MDT verifies above conditions
 - > MDT erases layout, removes OST objects, sets hsm_released bit
- Metadata is retained by MDT; looks like unstriped file
- Size must be saved on MDT also

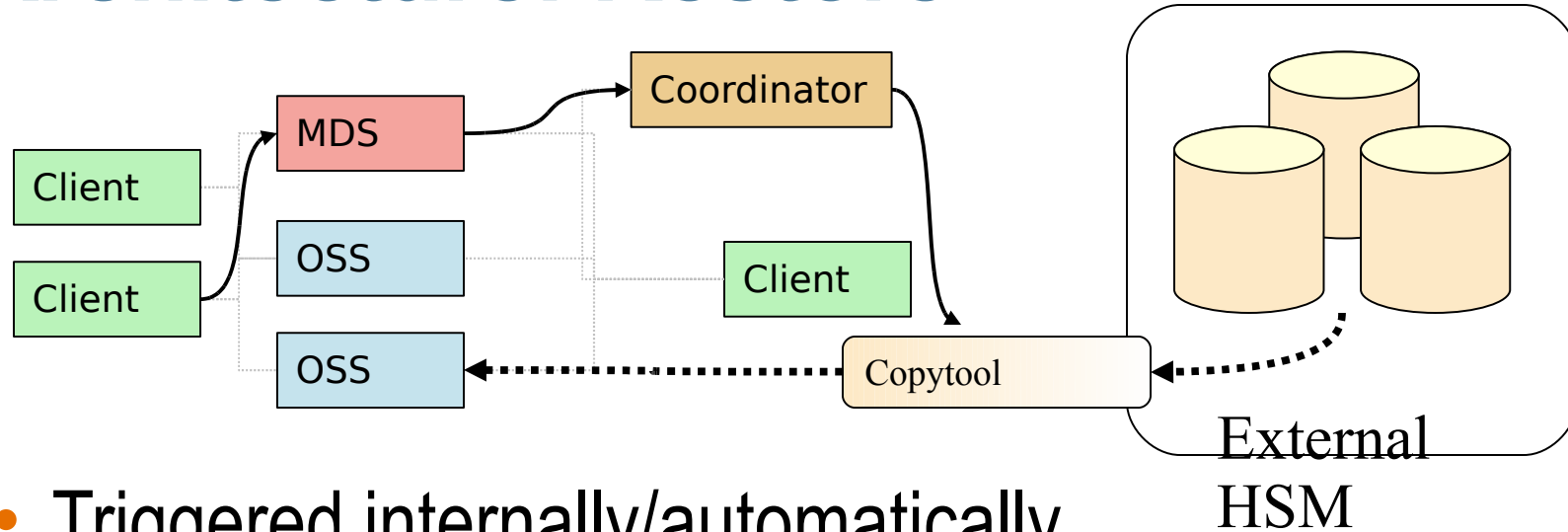
Layout Lock (bz13183)

- New MDT layout lock is created for every file
- Add MDS_INODELOCK_LAYOUT inodebit and IT_LAYOUT intent
- If MDT calls back inodelock bit, client marks layout invalid in `llu_inode_info`
- Client gets layout if needed (intent if it can, or explicitly:)
 - > Get the layout intent lock (cached)
 - > If the layout is invalid, get the layout
- Client takes ref on layout wherever layout info is used
 - > Extents (`llu_file_rwx`)
 - > Truncate (`llu_setattr_raw`, `ll_setattr_truncate`)
 - > Ost mtime (`llu_setattr_raw`, `ll_setattr_ost`)
 - > Mmap (`ll_file_mmap`)

Layout Generation (bz13183)

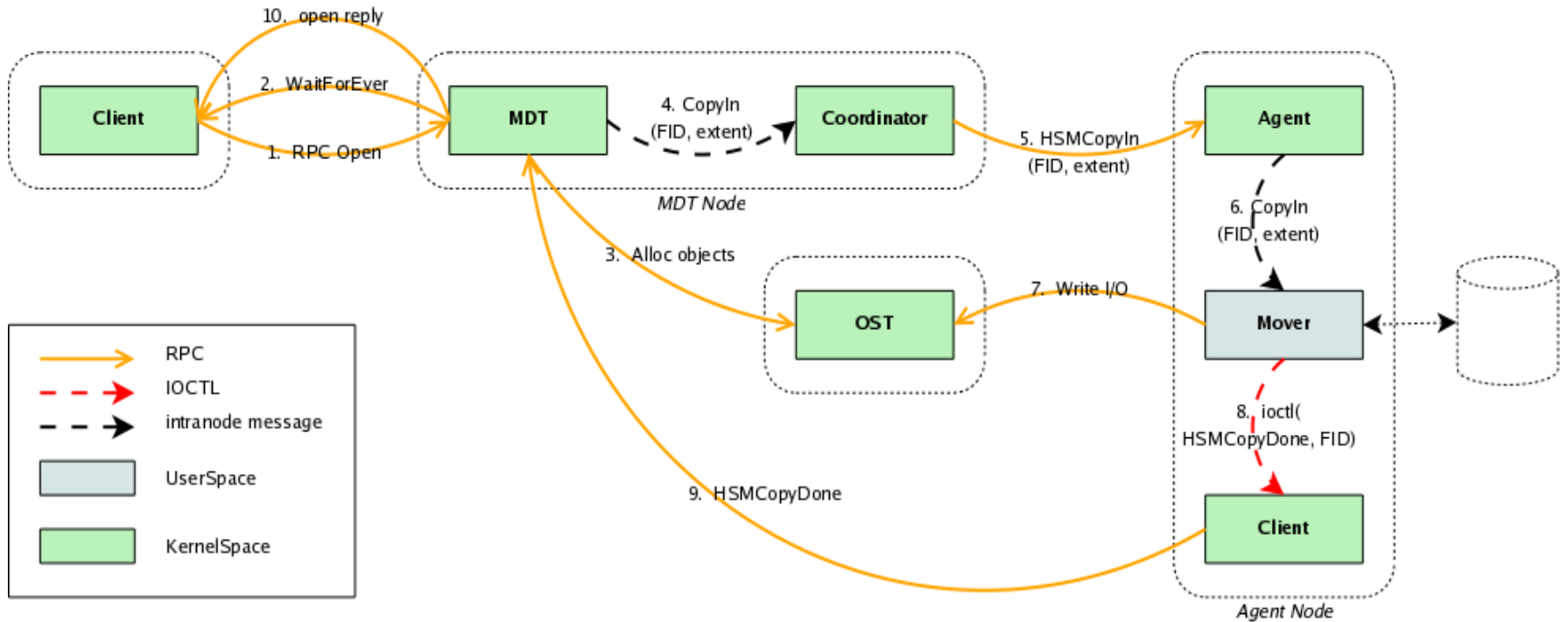
- Layout change increases layout gen
- Gen is passed with extent enqueue
- MDT takes full file extent locks on OSTs for layout change, propagating gen to OSTs
- OST stores greatest seen gen
- If OST sees request for an old layout gen (or deleted object), then the client has been partitioned from the MDT (didn't get layout update) and is denied extent locks ESTALE (not evicted from OST)
 - > Split lov_mds_md u32 stripe_count into u16 stripe_count and u16 layout_gen
 - > Add layout_gen to struct obdo and md_attr
 - > Lov increases gen in qos_prep_create

Architecture: Restore



- Triggered internally/automatically
- No policy engine interaction
- Transparent to client
- Coordinator requests restore from copytool
- Restore locking protocol is complicated

Restore RPCs



User triggers a copy-in (cache-miss)

Restore Locking Protocol (bz20631)

- Triggered by client layout intent lock request (O_NONBLOCK should not cause restore)
- MDT checks hsm_released bit; if released, the MDT takes PW lock on the layout
- MDT creates a new layout with a similar stripe pattern as the original, increasing the layout version, and allocating new objects on new OSTs with the new version.
- MDT (LOV) enqueues group write lock on extents 0-EOF (no timeout for group lock)
- MDT releases PW layout lock
- Client get CR layout, but must now wait for r/w file extent locks held by MDT
- MDT sends request to coordinator requesting restore of the file to .lustre/fid/<fidno> with group lock id and extents 0-EOF.
- Coordinator distributes that request to an appropriate copytool
- Copytool opens .lustre/fid/<fid> for write (gets current layout)
- Copytool uses group lock and copies data from HSM, reporting progress via ioctl
- When finished, copytool reports success and closes the file, releasing group extents lock
- MDT clears hsm_released bit
- MDT releases group extents lock. This sends a completion AST to the original client, who now receives his extents lock.
- MDT adds HSM_copyin_complete <fid> record to changelog (for policy engine)