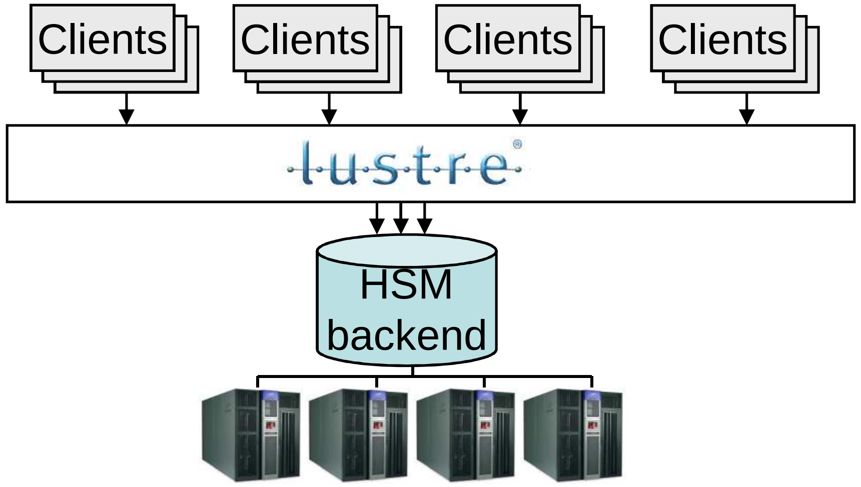
Lustre-HSM binding

Thomas LEIBOVICI
thomas.leibovici@cea.fr

Goals

- **HSM seamless integration**



- **Takes the best of each world:**
  - Lustre: high-performance disk cache in front of the HSM
    - parallel cluster filesystem
    - high I/O performance, POSIX access
  - HSM: long-term data storage
    - Manage large number of disks and tapes
    - Huge storage capacity

Back-ends

- **HSM independent design**
- **First supported storage back-ends:**
  - HPSS
  - POSIX
  - Sam/QFS
  - DMF
  - Enstore

Status and Availability

- **Status:**
  - Running prototype:
    - Main features implemented (archive, restore, release, state flags, import and recovery…)
    - Fixing tricky cases (MDS/client crash resilience)
  - End of implementation, starting integration tests
  - Patch review before landing

Collaboration

- **Oracle/Sun/CFS:**
  - Hua Huang: project manager
  - Nathan Rutman: Lustre expert, design, development
  - Andreas Dilger, Oleg Drokin: Lustre experts, design
- **CEA:**
  - J.C. Lafoucrière, A. Degrémont, Th. Leibovici: design, development
- **Other contributors:**
  - Tom Barron (ORNL)
  - Alex Kulyavtsev (Fermilab)
  - SGI (DMF binding)

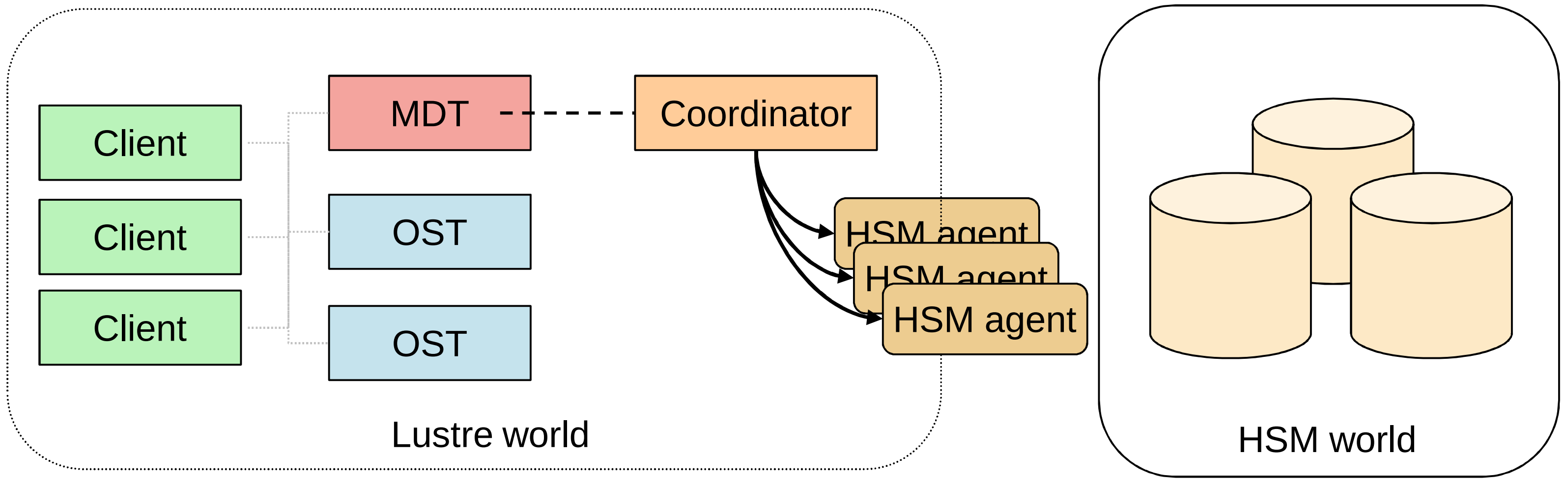
Design and features

Features

- **V1 Features:**
  - Migrate data to the HSM
  - Free disk space when needed
  - Bring back data on cache-miss
  - Policy management (migration, purge, soft rm,…)
  - Import from existing backend
  - Disaster recovery (restore Lustre filesystem from backend)

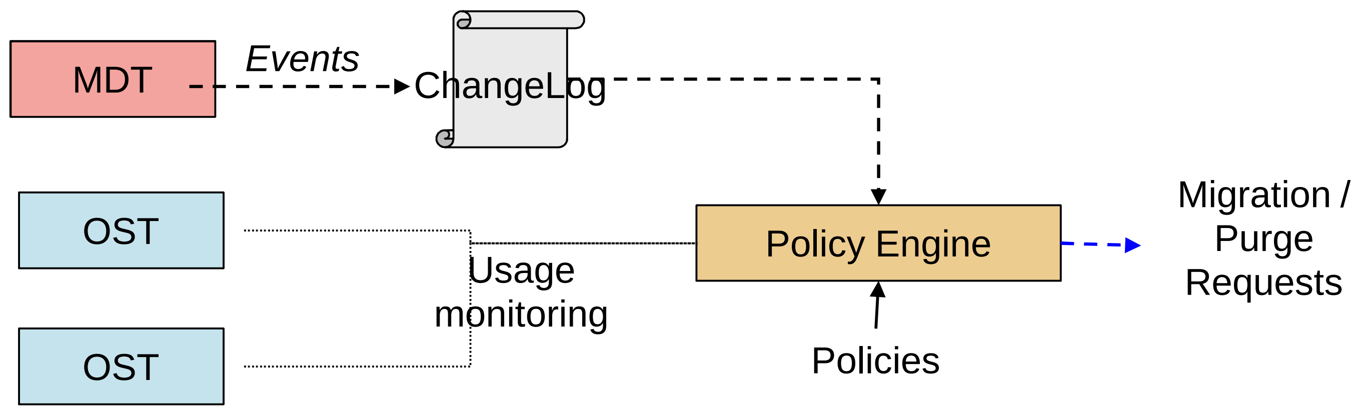
- **New components:**
  - Coordinator
  - HSM agents (backend specific daemon)
  - Policy Engine (user-space daemon)

Components (1/2)



- **Coordinator**
  - Manages file locking on data restoration
  - Centralizes copy requests
  - Dispatches requests on agents
- **HSM agents (user space)**
  - Move data, cancel copy and remove external storage files
  - Interface between Lustre and the HSM
  - Know how to communicate with a specific HSM

Components (2/2)



- **Policy Engine (user-space)**
  - Monitors filesystem disk space usage
  - Keeps track of files status and modification time
  - Pre-migrates not recently modified data
  - If free space is low, purges non recently accessed files (if already copied in the HSM)
  - Deferred rm in HSM (soft rm)

Robinhood: PolicyEngine for Lustre-HSM

- **PolicyEngine is the specification**
- **Robinhood is an implementation:**
  - CEA development
  - OpenSource (GPL-compatible)
  - http://robinhood.sf.net

- **Policies:**
  - File class definitions, associated to policies
  - Based on files attributes (path, size, owner, age, xattrs…)
  - Rules can be combined with boolean operators
  - LRU-based migr./purge policies
  - Entries can be white-listed

Robinhood: example of migration policy

- **Fileset definition:**

```
Filesets {
   FileClass small_files_A {
       definition { tree == "/mnt/lustre/project_A“ and size < 1MB }
       migration_hints = "cos=12" ;
   …
   }
}
```

- **Migration policy:**

```
Migration_Policies {
   ignore { size == 0 or xattr.user.no_copy == 1 }
   ignore { tree == “/mnt/lustre/logs” and name==“*.log” }

   policy migr_A_small {
        target_fileclass = small_files_A;
        condition { last_mod > 6h or last_copyout > 1d }
   }
   …
   policy default {
        condition { last_mod > 12h }
        migration_hints = “cos=42”;
   }
}
```

Robinhood: example of purge policy

- **Purge trigger:**

```
Purge_trigger {
   trigger_on = ost_usage;
   high_watermark_pct = 80%;
   low_watermark_pct = 70%;
}
```

- **Purge policy:**

```
Purge_Policies {
   ignore { size < 1KB }
   ignore { xattr.user.no_release = 1 or owner == “root” }

   policy purge_quickly {
        target_fileclass = classX;
        condition { last_access > 1min }
   }
   …

   policy default {
        condition { last_access > 1h }
   }
}
```

Incoming Features

- **v2 design process started at LUG 2010 (last week)**

- **Candidate features for Lustre-HSM v2:**
  - Partial file release/restore
  - Finest file locking on restore (v1: whole file)
  - Online metadata snapshot
  - Tape-friendly mass-restore operations
  - Enhanced multiple archive support (mirroring, policies…)
  - File writes may cancel archive operation
  - …

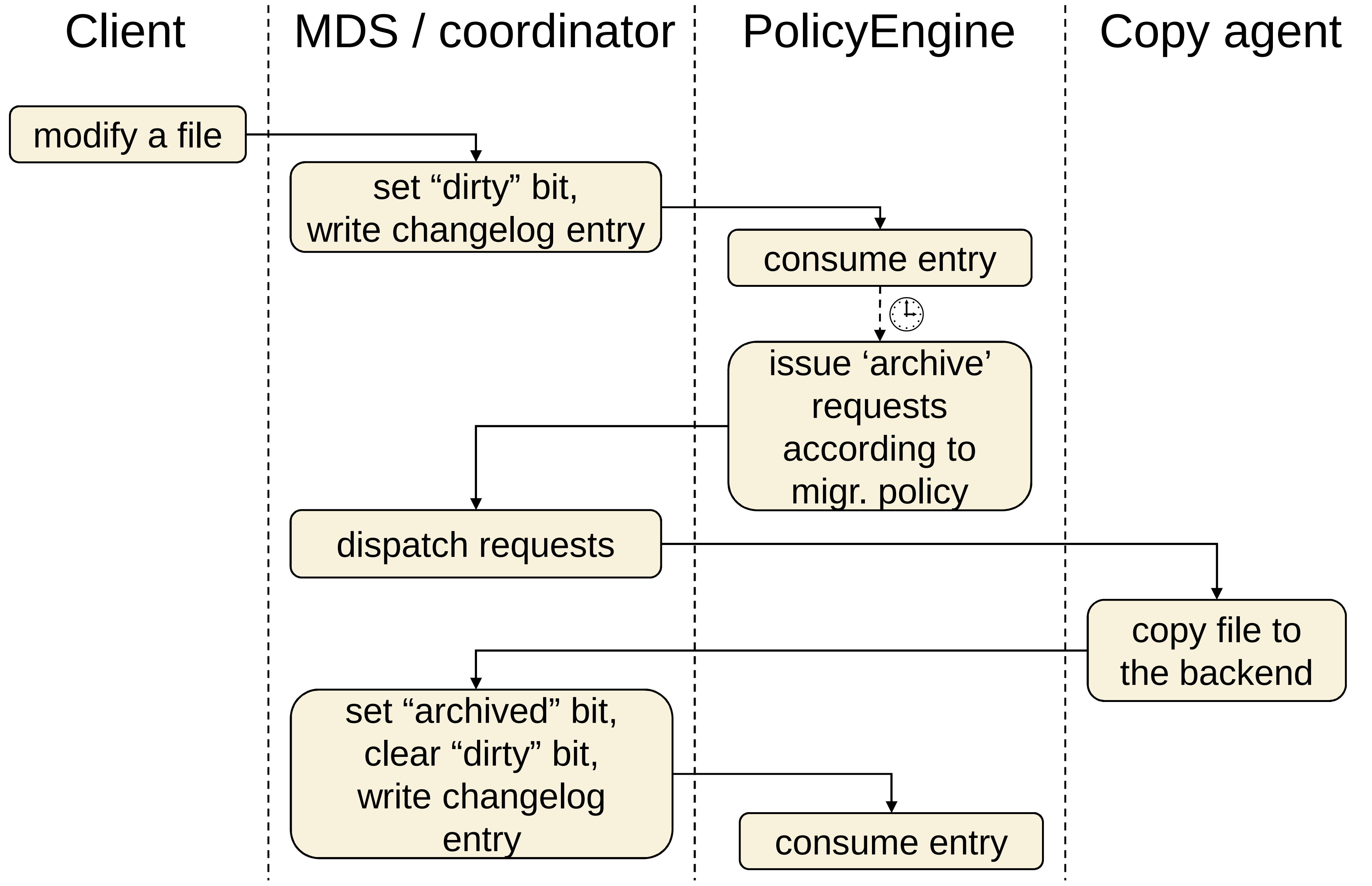
Demo

Lustre-HPSS binding

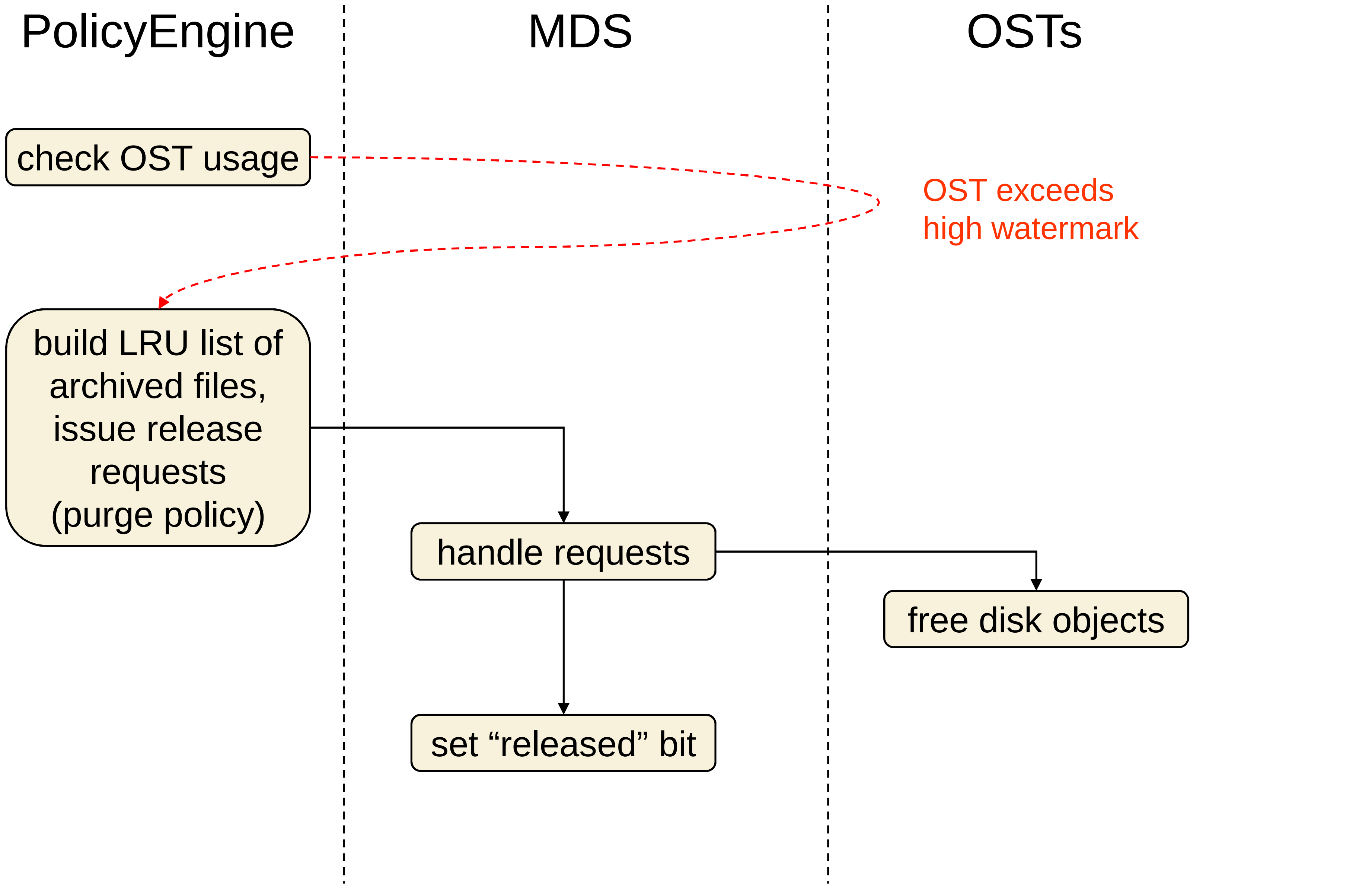
- **Video #1:** automatic file archiving and release
- **Video #2:** import of existing backend + basics
- **Video #3:** manual mass-archive/release actions

Questions

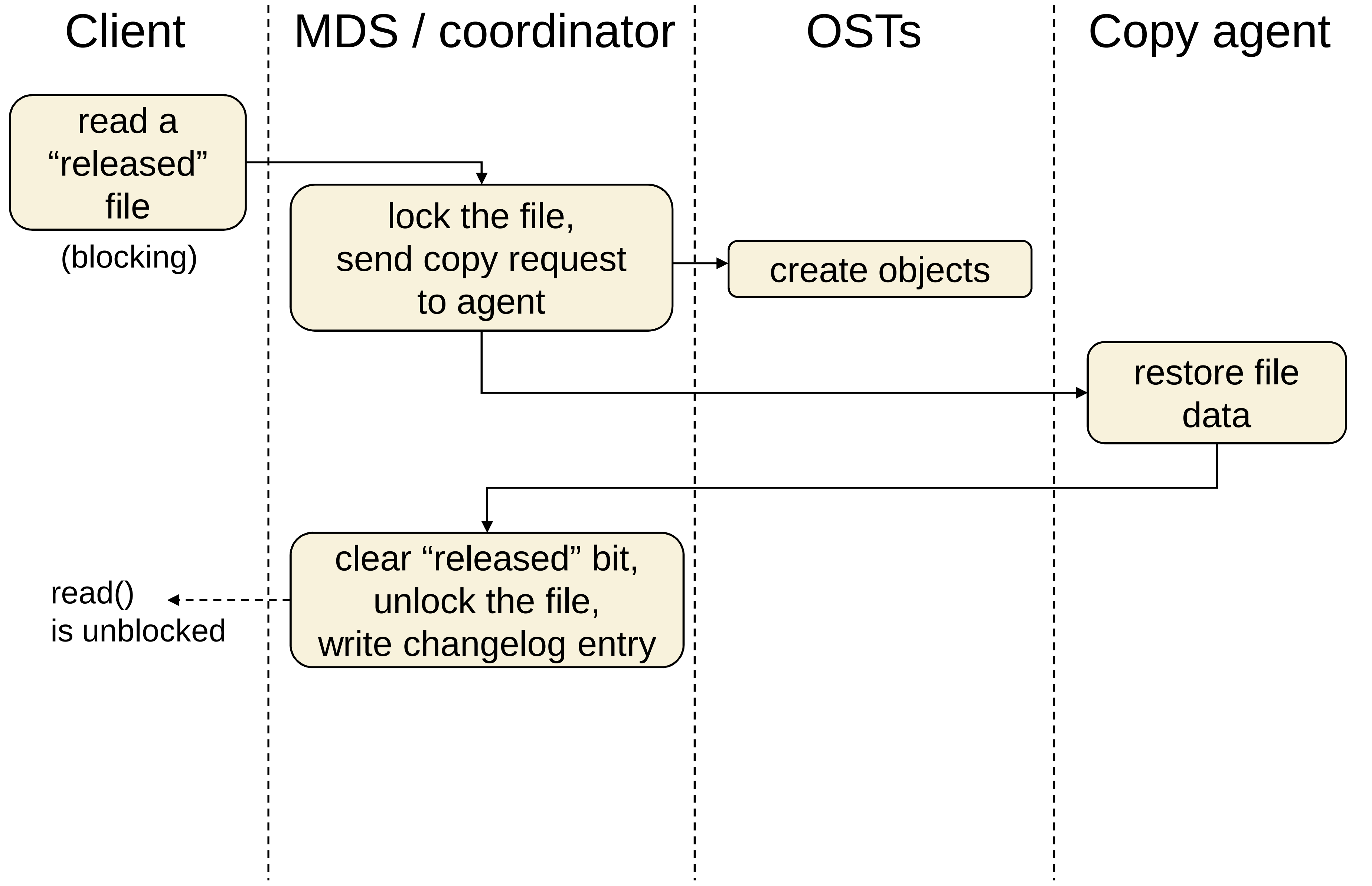
Use case (1): archive modified files


Client
MDS / coordinator
PolicyEngine
Copy agent
modify a file
set “dirty” bit, write changelog entry
consume entry
issue ‘archive’ requests according to migr. policy
dispatch requests
copy file to the backend
set “archived” bit, clear “dirty” bit, write changelog entry
consume entry

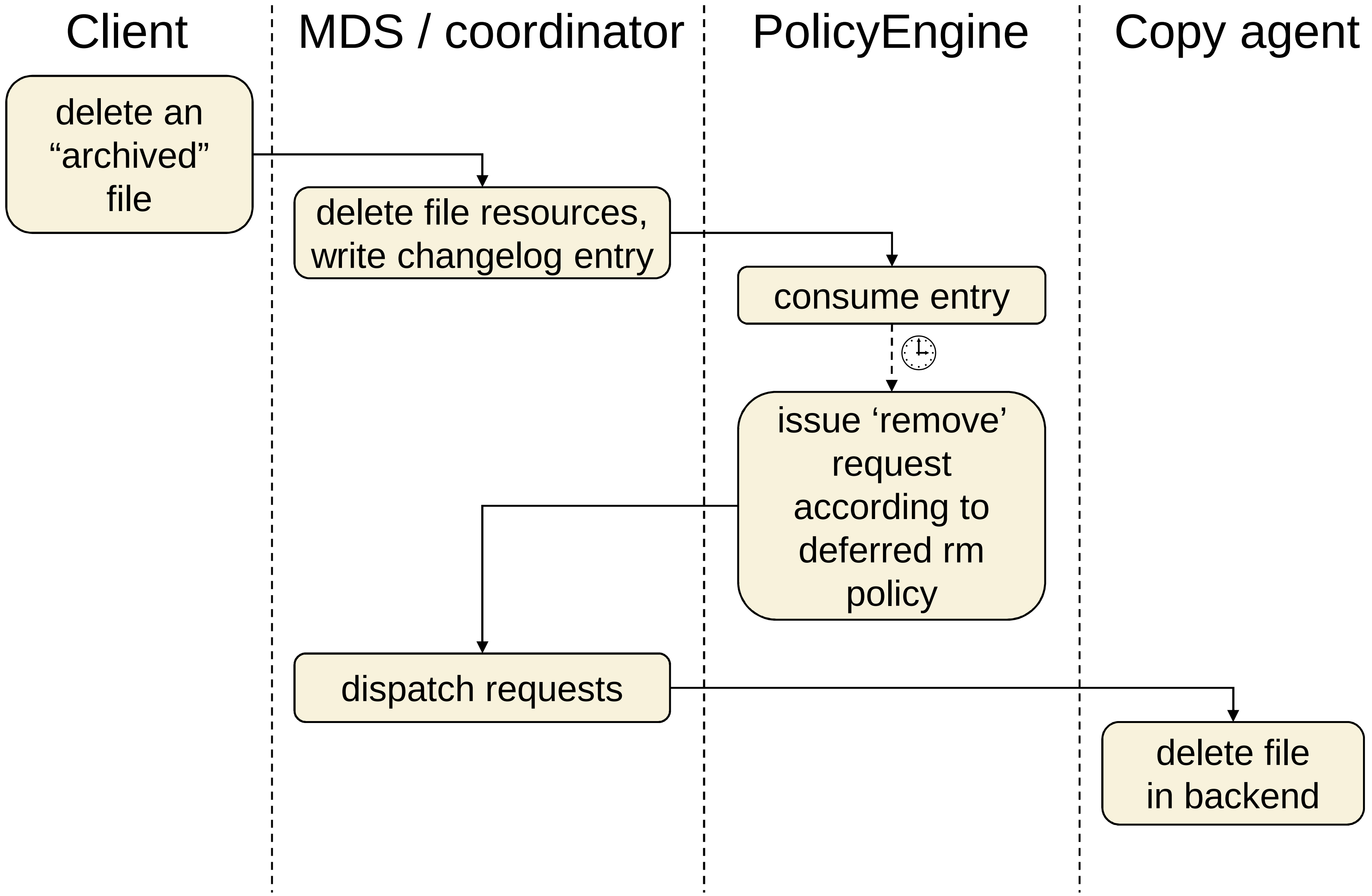
Use case (2): release disk space


PolicyEngine
MDS
OSTs
check OST usage
OST exceeds high watermark
build LRU list of archived files, issue release requests (purge policy)
handle requests
free disk objects
set “released” bit

Use case (3): restore released file



Use case (4): deferred removal (soft rm)


Client
MDS / coordinator
PolicyEngine
Copy agent
delete an “archived” file
delete file resources, write changelog entry
consume entry
issue ‘remove’ request according to deferred rm policy
dispatch requests
delete file in backend