Intel® Lustre* File Level Replication

Jinshan Xiong

Lustre Engineer

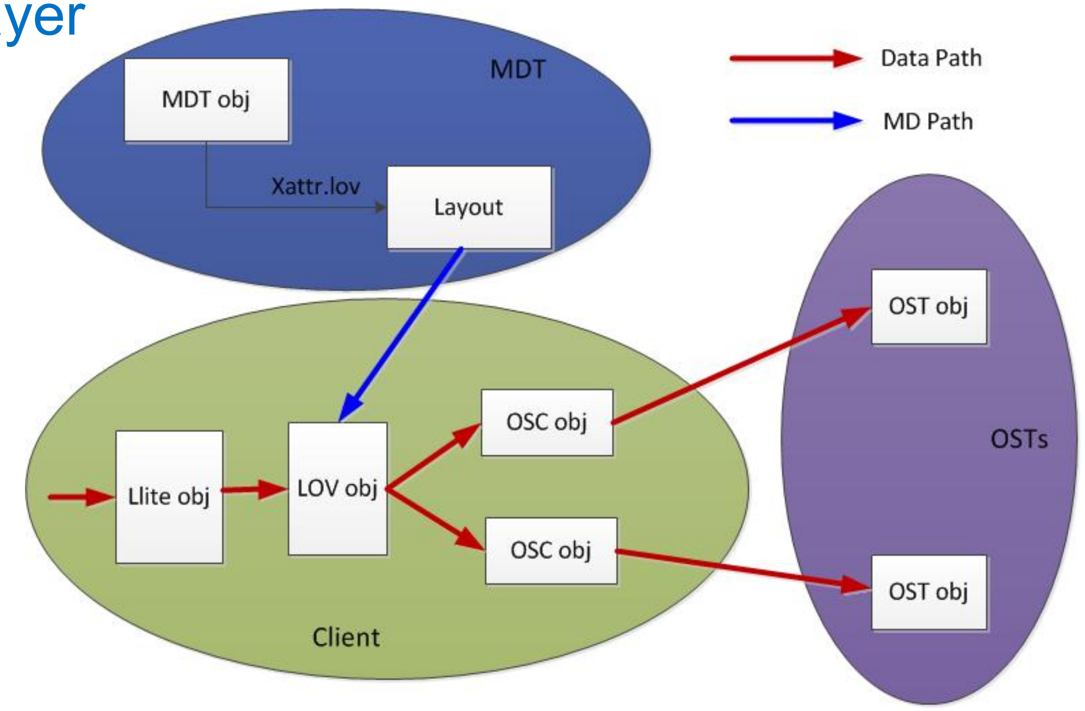
April 08, 2014

Agenda

- Data layout & Replication
- Replication overview
- I/O model for replication
- Current Status

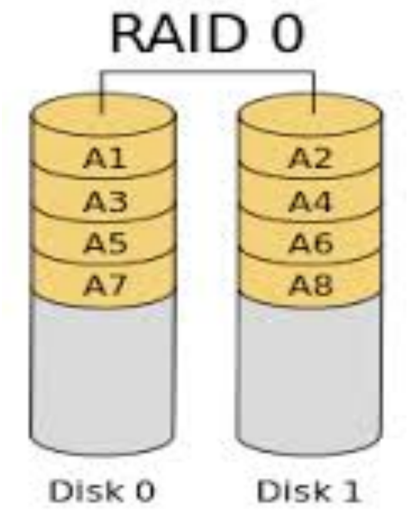
What is the data layout

- Data layout controls data placement to OSTs
- Stored as `trusted.lov` xattr on MDT
- Interpreted at the LOV layer
- Used to be immutable



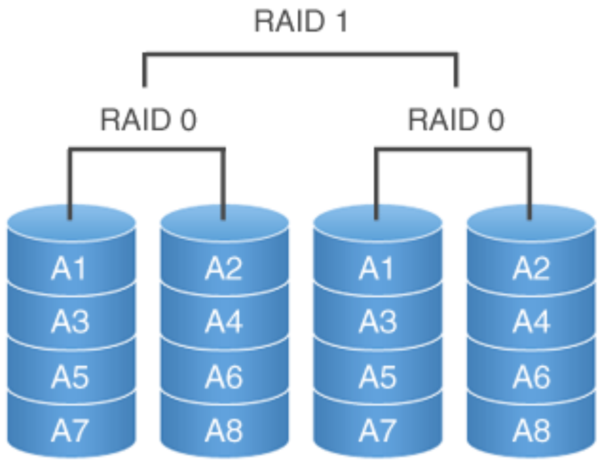
Current data model in Lustre*

- Lustre only supports RAID-0 - striping
  - File is not accessible if any OST died
  - Increasing stripe count decreases availability
  - RAID-0 data is unavailable/lost if any OST fails
  - Impossible to be deployed on commodity hardware
  - Not suitable for cloud computing environments



Layout for Replication

- RAID-0+1 will be supported
  - Store the same copy of data on multiple places
  - Each replica is represented by RAID-0
  - Easy to operate specific replicas
  - Reuse current code as much as possible



Replication Overview

- Improve read IO availability
  - If one replica is out of reach, IO engine on client can try another replica
  - No expensive data storage for OSTs
  - No HA configuration needed
- Improved read bandwidth
  - Different clients can read the same data from different replicas
  - Helpful for VM deployment, configuration files
- Good for read mostly and write rarely files
  - Write becomes expensive with multiple replicas
- Design funded by OpenSFS

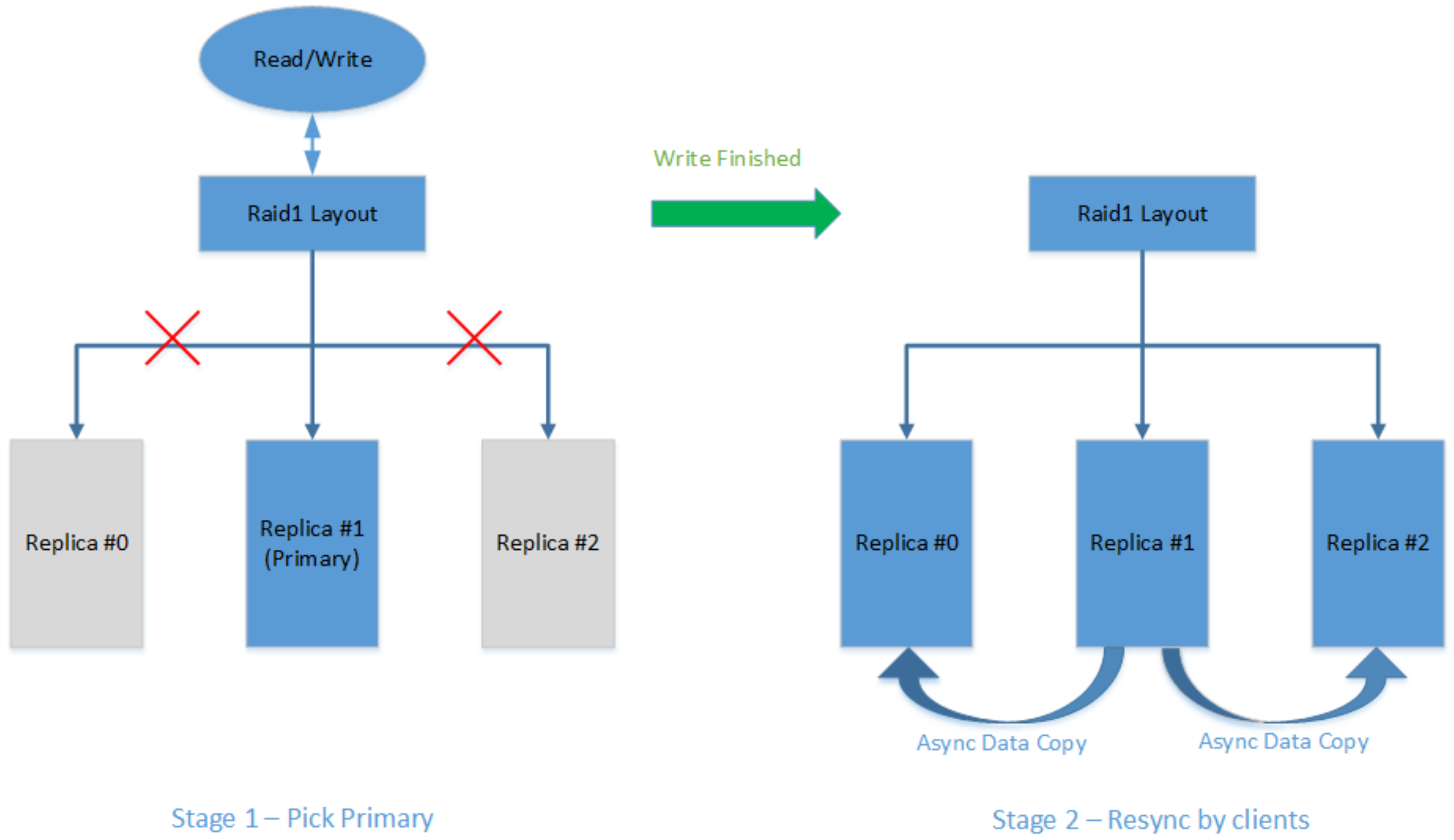
Read Replicated Files

- Policy to select replicas
  - OST type, preferred replica explicitly specified by user
  - OST failure domains can be isolated by OST pools
  - Runs on MDT and LOV on client
- Retry mechanism
  - IO doesn’t wait at the PTLRPC layer if OSTs are unreachable
  - LLITE retries the IO and LOV picks up new replica

Write to Replicated Files

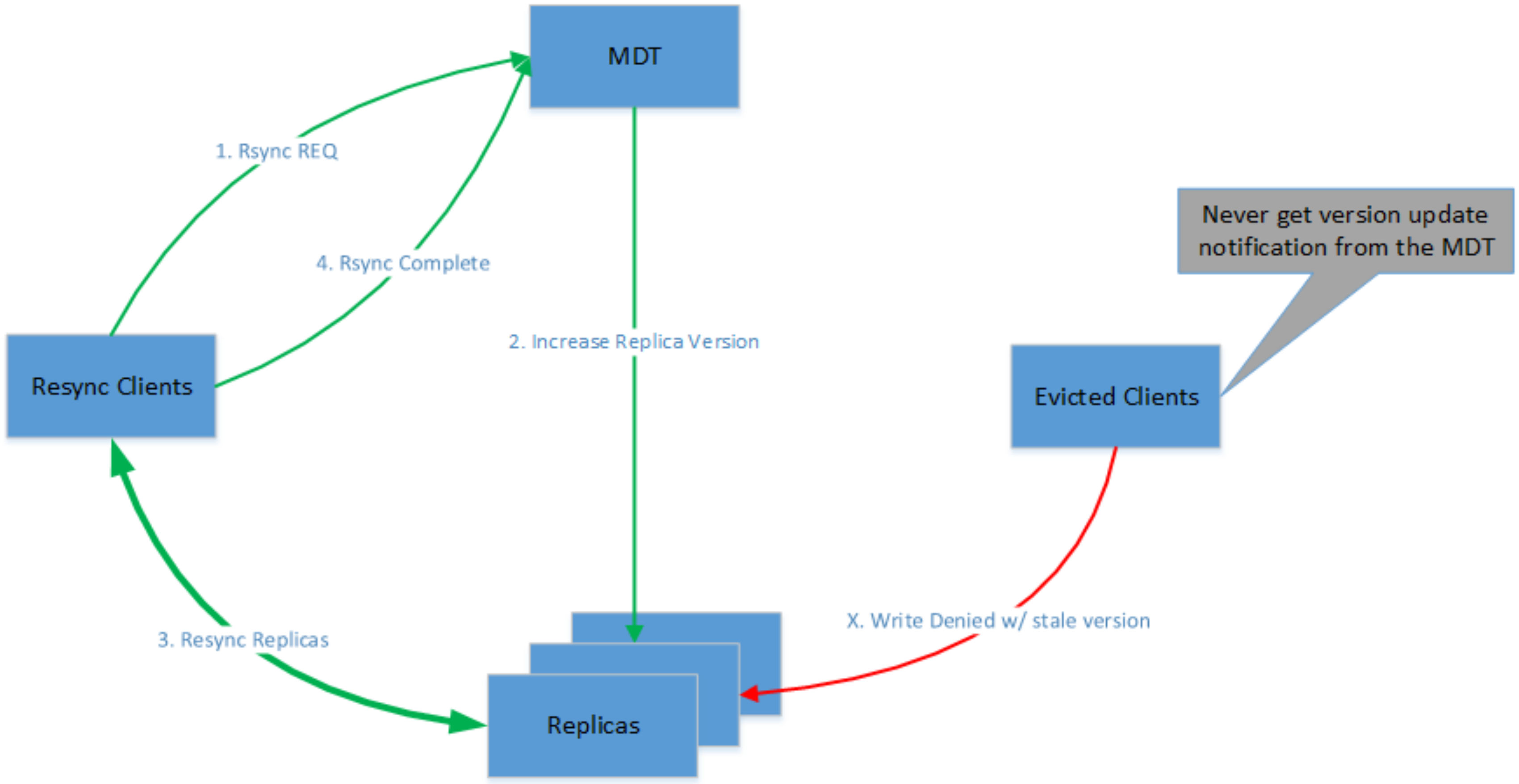
- Write is difficult
  - Data consistency is guaranteed for Lustre*
  - Node failure can happen any time during a write
  - Phantom writes from evicted clients make it harder
- Two stage write
  - Pick a replica as primary, and mark others as out of date
  - File is degenerated to a regular, unreplicated file
  - After write is done, out-of-date replicas will be resynced with primary

Two Stages Write



Versioned Replica

Purpose: stop writes from evicted clients


MDT
1. Rsync REQ
4. Rsync Complete
2. Increase Replica Version
Never get version update notification from the MDT
Resync Clients
Evicted Clients
3. Resync Replicas
X. Write Denied w/ stale version
Replicas

Current Status

- Two phase project
  - Phase 1 is to provide basic functionality
    - Replica reads, RAID-0+1 layout handling
    - Sync write on MDS to mark replicas stale
    - Offline resync tool, `lfs` interfaces
  - Phase 2 will improve write performance
    - Async writes to replicas from the client
    - No need for resync tool under normal operation
- HLD finished for Phase 1
  - HLD was funded by OpenSFS
  - Need funding for implementation