Whamcloud

Lustre Metadata Redundancy

Lai Siyao, Oleg Drokin

2026 Apr 27

DDN® STORAGE

Preface

Historically Lustre depended on special (expensive) hardware for HA

- Dual-ported HDDs/SSDs/…
- Smart storage controllers with coherent caches
- Or make-shift solutions (iscsi, drbd, …)

Now there's push for "software defined" solutions, so that already brought us:

- Mirroring
- Erasure coding (network raid) – in progress

But metadata is still not covered.

Goals

Whamcloud

Implement comprehensive redundancy for services and filesystem metadata:

- Filesystem configuration Service (MGS, hosted separately or combined with MDT0)
  - Servers and their current location
  - Nodemap configuration
- Special services:
  - Quota
  - POSIX file locks
  - Special “resources” (filesystem root, global rename lock and such)
  - FID location database
- Mirroring for File and directory inodes (across multiple MDTs)
  - What people usually think of as “metadata”:
  - Filenames, ACLs.

Fault-Tolerant MGS (Management Service)

Whamcloud

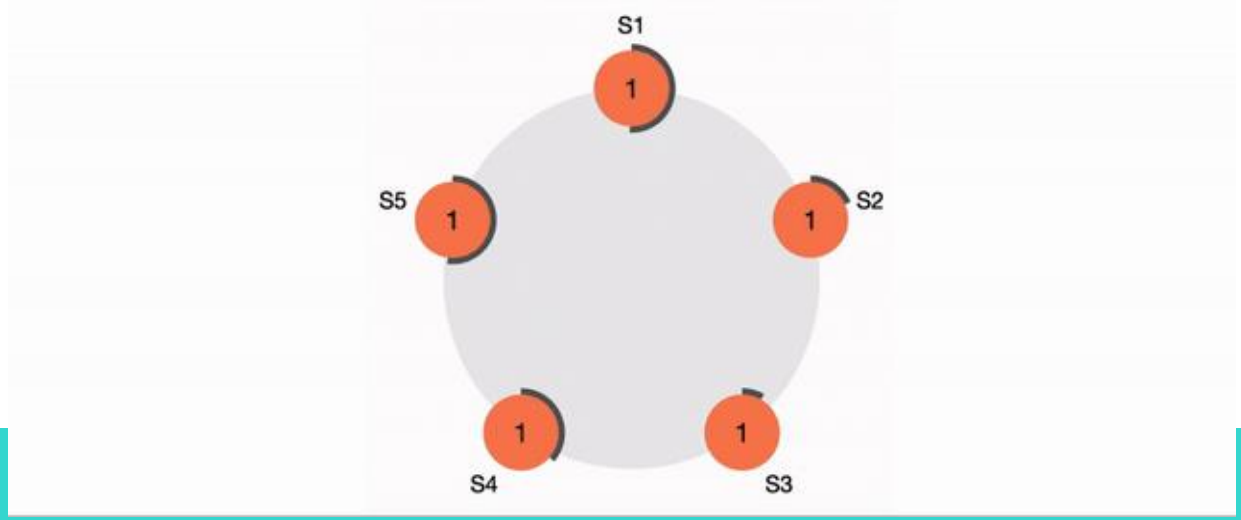
- It's hard to implement fully replicated MGS since there should be a single MGS to generate configuration files

1. All servers already have some cache for relevant MGS config data
   - Extend this to store all MGS information and add a mechanism to keep them in sync (LU-19915)
   - These could also be later used to rebuild new MGS
2. Now that we have the replicas, every server node would start an MGS instance (attached to one target)
   - Good enough for client mount purposes (LU-19916)
   - Client can mount using any MGS replica, so less load on primary MGS.
   - The clients need to know where the replicas are located.
3. Propagation of "Imperative recovery" information (LU-19917)
   - Target locations would need to be not only propagated to replicas, but to their clients as well

RAFT Consensus protocol (tiny overview)

Whamcloud

- Easy to understand
- Widely used
- Leader election from available candidates
- Replicated consistent logs distributed from the leader
- Leader pings all followers regularly
- If the leader dies the first to timeout follower to call for election
- If enough of others still alive nodes agree – it becomes the new leader.



Replicated FLD/Quota Service architecture

Whamcloud

- Lustre Raft Implementation (LU-19584, LU-19585)
  - Generic Raft protocol implementation from https://github.com/willemt/raft
  - Network (ptlrpc) and storage(LLOG) integration with Lustre
  - Raft initialization, handers, callback and cleanup
  - Each MDT can be regarded as a Raft Node
- Multiple Service support
  - Start FLDB/Quota service on all the node with MDT mounted as Raft node
  - The Updates for each service will be propagated to other Raft nodes by the Leader node
  - MGS could be added to the mix too
- Service configuration update by Raft
  - Replicated Service configuration changes by Raft FSM (finite state machine)

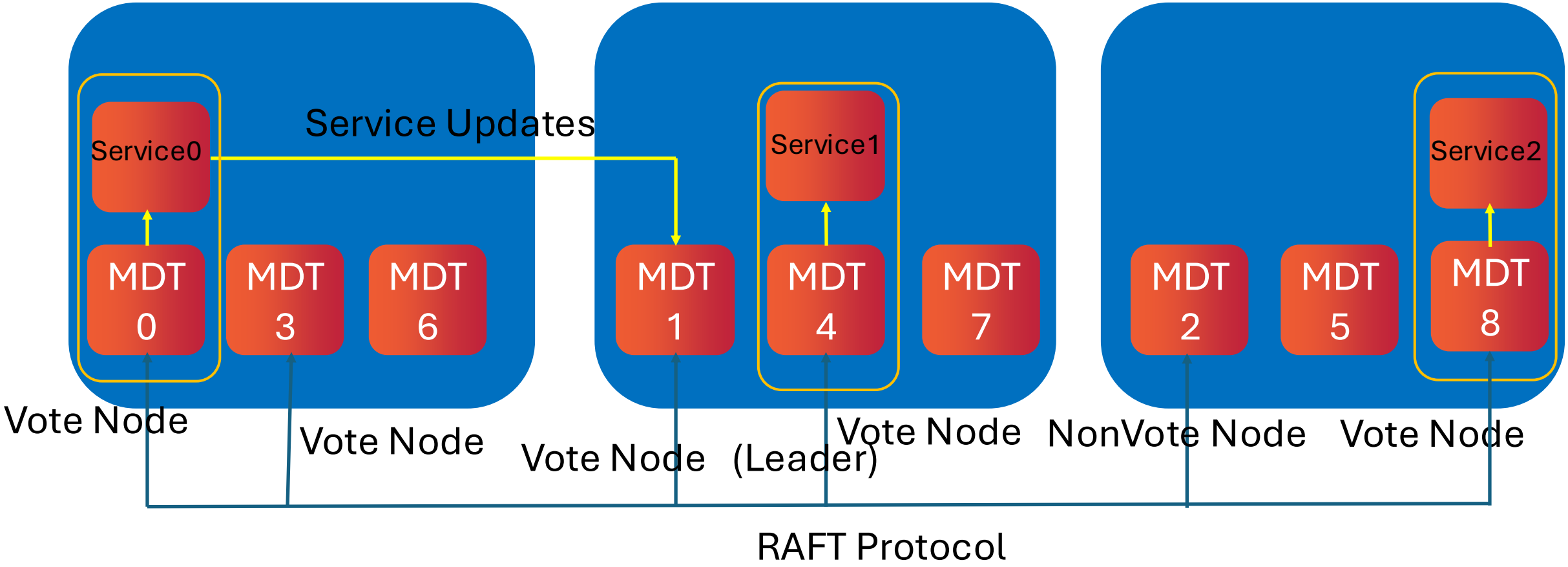
Raft Implementation

Whamcloud

- Core components
  - New kernel module: lraft
  - Leader election: Automatic selection of Leader node
  - Raft Log replication: Consistent state across nodes
  - Fault tolerance: Survives (N-1)/2 node failures

- Deployment
  - Mount option ‘-o raft’ for voting nodes, ‘-o raft_nonvote’ for non-voting nodes
  - MDT, OST will be mounted as “raft_novote” by default

Raft Implementation



Fault-tolerant Quota

Whamcloud

- Fault-tolerant Quota
  - The Quota setting will be synced between RAFT nodes with RAFT protocol
  - The Quota setting update request (by LCTL) can be sent to any MDT node
  - Improves reliability by avoiding loss/corruption of the filesystem configuration llog files
  - Improve availability by avoiding the dependence on the MDT0000

Quota - set

Whamcloud

- Quota Setting Update
  - The quota setting update request will be sent to any MDT by User
  - The request will be forwarded to the Leader RAFT Node
  - The request is processed in RAFT Leader Node
  - The quota setting is synced to other RAFT nodes by Leader Node after it is processed
  - Other MDTs will update its quota settings after receiving the update
- Quota Setting Retrieval
  - The quota setting on the MDT that receives the request will be used for the request
  - The quota usage will still be retrieved from the QSDs

Quota – grant update

Whamcloud

- Quota Grant Update
  - The quota grant is updated much more frequently than quota setting
  - Sync the quota grant between RAFT nodes could affect performance
  - The quota grant for specific Quota Id could be re-initialized if it is needed
- Quota Grant Sync
  - The quota grant could be synced in batch
  - Doesn’t sync the quota grant and re-generate it when new QMT is used

Quota – Load balancing

Whamcloud

- The Target QMT
  - The target QMT for different Quota ID could be different on the same QSD
  - The target QMT for the same Quota ID should be the same on different QSD
  - The change of the target QMT is managed by RAFT
  - whether the quota grant need to be synced between RAFT nodes
  - the FTQ without load balance will be implemented first

FLDB – Replication

Whamcloud

- FLDB – FID Location Database
  - Provides map of data/metadata location on services
  - needs strong consistency
  - RAFT protocol is suitable to provide these as well
    - though might be a bit of an overkill since the updates are infrequent.

Fault-tolerant POSIX file locks/special resource locks

Whamcloud

- Special resource locks use a reserved sequence (0x200000007) that normally lives on MDT0
- When MDT0 becomes unavailable (or removed), another MDT would be chosen to provide these services.
  - MGS will hold information about the new location

Inode Metadata Redundancy / replication

Whamcloud

- Really complicated problem.

- Several competing ideas at present
  - Dynamic mirroring on per-inode basis (LU-17818)
    - Transactions and metadata pointers increase proportional to desired mirroring level
    - Phased approach with top level directories replicated first, as most important and least changing
  - Fixed-size fs mirroring, where every inode is automatically replicated to pre-set number of MDTs using special sequences
    - Simplifies allocation/creation and search of replicas in exchange for reduced flexibility

Inode Metadata Redundancy / replication

Whamcloud

- Really complicated problem.

- Several competing ideas at present
  - Dynamic mirroring on per-inode basis (LU-17818)
    - Inode pointers would get multiple FIDs for replicas
    - Transactions and metadata pointers increase proportional to desired mirroring level
    - Phased approach with top level directories replicated first, as most important and least changing
    - Read only degraded operations
    - Then write to degraded directories

    - Eventually a complicated lfsck surgery

File Metadata Redundancy – fixed mirroring

Whamcloud

- FID Structure Enhancement
  - Reserve first 3 bits of FID sequence for replica ID
  - Support up to 8 replicas per file
  - Example FID structure for 3 replicas:
    R0: [0x200000401:0x1:0x0]
    R1: [0x20000020000401:0x1:0x0]
    R2: [0x40000020000401:0x1:0x0]
  - ROOT is no different from other files
  - Each replica directory tree operates independently, which makes LFSCK unchanged
  - Directory entry stores replica FID with the same replica ID
  - On MDT deletion – can just switch all FLDB entries of old MDT to other MDTs and populate there
  - On MDT reconstruction just replicate from other MDT entries.

Inode Metadata Redundancy / replication

Whamcloud

- Really complicated problem.

- Several competing ideas at present
  - Dynamic mirroring on per-inode basis (LU-17818)
    - Transactions and metadata pointers increase proportional to desired mirroring level
    - Phased approach with top level directories replicated first, as most important and least changing
  - Fixed-size fs mirroring, where every inode is automatically replicated to pre-set number of MDTs using special sequences
    - Simplifies allocation/creation and search of replicas in exchange for reduced flexibility
- Everything hinges on improved distribution transaction performance
  - And metadata is the last place we can afford slowdowns.
  - LU-17821

Questions?