Controlled Server Shutdown

Eric Mei

Lustre Group, Sun Microsystems

CSS Overview

- Server:
  - Initiated by every 'umount', before setting device read-only.
  - Send parallel notifications to all connected clients.
  - Wait a limited time for clients finish syncing data.
  - When disconnecting a client, keep its record in last_rcvd.
  - Proceed to rest of shutdown.
- Client (after got server notification):
  - Block all new forming RPCs.
  - Cancel all locks of corresponding namespace.
  - Wait for all inflight RPCs to drain.
  - Wait for replay-queue to drain.
  - Send a DISCONNECT to notify server.
  - Wait for recovery.

A little more details

- CSS is tentative only, we don't try too hard, don't wait for too long.
  - Server:
    - Ignore clients which is not in fully connected status.
    - Stop accepting new CONNECT once CSS started.
    - Do not evict clients who can't finish syncing.
  - Client:
    - If any RPC timeout happens, the client will simply abort syncing and enter recovery.
    - If can't finish syncing in time, abort syncing.
- If client detected server upgrading (to incompatible RPC wire format), and we have RPCs to replay / resend, do a self-evict (maybe not necessary?).

Client Details (1)

- Some locks can't be cancelled
  - Grouplock (Flock too??).
  - Cached data protected by the lock won't be flushed automatically.
  - For RPC compatibility, this is fine.
  - For Data safety, less optimal.
  - Solution: do an fdatasync to flush data once only?
  - Except for mmap, processes can be blocked before dirtying pages
- Small issue of draining RPC
  - Waiting for imp_inflight to be 0
  - PING itself will increase imp_inflight
  - Better to refcount threads beyond RPC blocking mutex
  - Once clients outside critical region all RPCs are finished sending
  - Flushing all pending RPCs and waiting for commit will clean client

Client Details (2)

- Cleanup everything
- Blocking new RPCs forming
  - Must be done before RPC is formed.
  - Ideally to block all obd_api / md_api to freeze new calls to the target being upgraded.
  - Problems:
    - But some of them may be used during syncing, so blocking must be selective (Need more thinking??)
    - In 1.8, some MDC functions are exported to llite directly. HEAD should have no such problem.
  - Result: the block checking is called in many places, hard to maintain.

Open Recovery (1)

- Currently Open Recovery
  - Open-create involves disk transaction on MDT.
  - Open-exist associated with a fake transno.
  - Open RPC remains in client ptlrpc-level replay-queue until closed.
  - Disadvantage:
    - Code is complex and continually broken (Andreas)
    - Evicted client may still think it has open files and continue doing I/O until got error in close.
    - Create problem regards to capability renewal.
    - Specifically for CSS: require open RPC conversion in case of MDS upgrade.

Open Recovery (2)

- Open reconstruction (proposed by Nico)
  - Client maintain a list of open file data (FID, mode, etc.)
  - Transno:
    - Open-exist don't owns a fake transno, thus won't go into replay queue.
    - Open-create RPC still enters replay-queue, but removed after create transaction committed.
    - In either case, proper info goes to open file data list.
  - In case of recovery:
    - RPC replay for uncommitted transactions (including create)
    - Recover open state by reconstruct open-exist RPCs based on open file data (open-by-FID).
    - RPC resend of unreplied transactions (including open-create)
    - VBR delays orphan recovery until recovery is finished

Open Recovery (3)

- Recovery order
  - Recover open at first, to avoid later replay of unlink removes the object.
  - Special treatment of unlink replay is needed – VBR does this already.
- Open recovery wrt other transaction
  - Create: when recover an open, the open-create transaction must have been committed, so the object must have been created already.
  - Unlink: if unlink transaction has been committed, the object should be in orphan list; otherwise still exist.
  - Setattr-permission: That's why MDT must bypass permission checking during open recovery.
- No RPC conversion would be needed
  - After a successful syncing, client replay queue will be empty.

Open Recovery (4)

- Alternative (proposed by Alex)
  - Implement open in terms of LDLM locks.
  - Single recovery mechanism:
    - On-disk transactions: RPC replay
    - In-core state: reconstruction / recover locks
  - Eliminate the special “open recovery” phase in recovery
  - We always need locks anyways for layout and attributes
  - Looks better, but perhaps requires more changes?
  - Can be implemented separately from CSS work

THANK YOU

Eric Mei
Lustre Group, Sun Microsystems