DNE Async Recovery

Wang Di & Andreas Dilger

September 24, 2014

Cross-MDT Normal Operation

Master MDT creates local (empty) transaction for RPC *operation*

- Master MDT gets DLM locks for local and remote objects
- Master MDD/LOD breaks operation into *updates* for local/remote targets
- OSP declare accumulates remote updates into transaction handle (thandle)

Transaction starts after all updates (local/remote) are declared

- Local OSD and remote OSP(s) write log record with **all** updates for recovery
- Recovery log is written exclusively by one Master MDD, not shared
- Update log on *slave* OSD includes master transno + *distribution ID*

Master MDD will stop the transaction first

Updates sent via Object Update Target (OUT) RPC to slave MDT/OSD

- Local update(s) and update log written atomically to disk

Cross-MDT Operation Log Cancel

Slave MDT will know if own updates have been committed

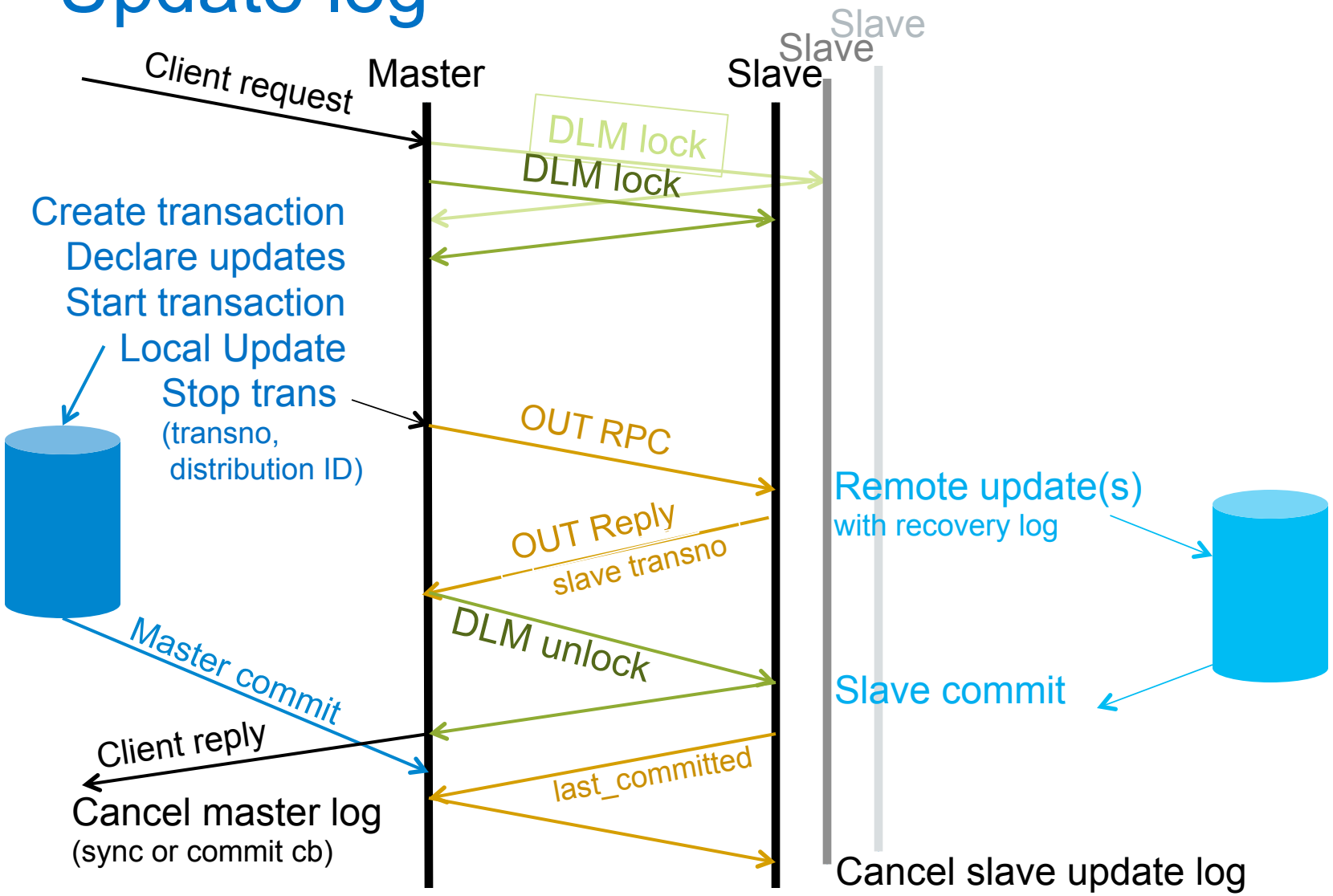
- Use local transno and commit callback from local OSD
- Use normal RPC reply to send pb_last_committed back to master

Slave MDT cancels log on Master MDT first, waits/commits

Slave MDT cancels local update log after Master commits

- Update log cancel on Master MDT must commit before Slave log cancel
- Slave MDT can know whether to replay updates by checking local update log
- Client request xid and master index stored in update log, replayed to last_rcvd

Update log


Client request
Master
Slave
Slave
Slave
DLM lock
DLM lock
Create transaction
Declare updates
Start transaction
Local Update
Stop trans
(transno,
distribution ID)
OUT RPC
Remote update(s)
with recovery log
OUT Reply
slave transno
DLM unlock
Master commit
Slave commit
Client reply
last_committed
Cancel master log
(sync or commit cb)
Cancel slave update log

Recovery process

Master MDT replays local/remote update log on recovery

- Aggregate update logs from all slave MDTs, remove duplicates
- Master and slave compare transno in update log to last_committed

Failover MDT will accept replay requests from other clients or MDTs.

- Replay will be handled by the order of transno, and t he recovery thread will choose the next replay request either from normal replay req(from other MDTs or client) or updates(got from other MDTs).
  - If the replay request(from client and other MDTs) are duplicate with the update, the update will be skipped.
  - 0_transno update can be handled in this process, because these updates will only exist on Master MDT, during recovery
    - If master MDT fails, it will resend these 0-transno update to the slave MDT, and slave MDT can know whether to redo these update by checking whether these update exist in the local log.
    - If Slave MDT fails, it will retrieve these 0-transno update from master MDT, and then check whether these updates exist in the local log.

Different failure cases

If master MDT fails

- If the update records have not been committed to disk, client will replay the request to master MDT, and master MDT will redo the request and sends update to slave MDTs
  - Each slave MDT will redo these updates if these updates do not exist in their local update log.
  - If the client crash at the same time, Master MDT(OSP) will retrieve the update from the slave MDT, and check whether the update needs to redo by comparing transno.
    - Master MDT will retrieve the update log from slave MDT by OSP, then call local OUT to redo these updates?
    - the master MDT might need to send the update to other slave MDTs as well. (for example rename), OSP might need upcall LOD to help?
- If the update records have been committed to disk, master MDT will retrieve the update log locally and re-send to the slave MDT.
  - Slave MDT will redo these updates if these updates do not exist in the local update log.

Different failure cases

If slave MDT fails,

- If updates have not been committed to disk on slave MDT, master MDT will resend updates to the slave MDT by normal ptlrpc replay.
- If updates have been committed to disk on slave MDT, it does not need to do anything, and the log cancel thread will cancel these records finally.

Different failure cases

If both master and slave MDTs fails,

- If the update has not been committed on master MDT, client will replay the request by normal ptlrpc replay.
- If client crash as well,
  - both master and slave MDT do not commit anything before fails, then the FS will keep consistency.
  - the update has only been committed on master MDT,
    - Master MDT retrieve the local master update records and send to the remote MDT, and remote MDT will redo it if updates do not exist in the update records.
  - The update has only been committed on slave MDT,
    - Master MDT(OSP) retrieve the update records from slave MDT, and redo locally if needed(by comparing the transno), and then it might also need to redistribute these updates to other slave MDTs.
  - If the update has been committed on both master and slave MDT, slave MDT will find the update log already exists when it gets update from master MDT and skips it.

Recovery example

mkdir

- The client sends the req to the MDT where the directory is.(different with 2.4/2.5, convenient for creating striped directory)
- If the master MDT restarts
  - If updates are not committed to disk, client will replay the the create req.
  - If updates have been committed to disk, the master MDT will resend the name insert to the slave MDT, and slave MDT will insert the name if the update does not exist in the local log.
- If the slave MDT restarts
  - If updates are not committed to disk, the master MDT will replay(normal ptlrpc replay) name insert to the slave MDT.
  - If updates have been committed to disk, it does not need to do anything.

Recovery example

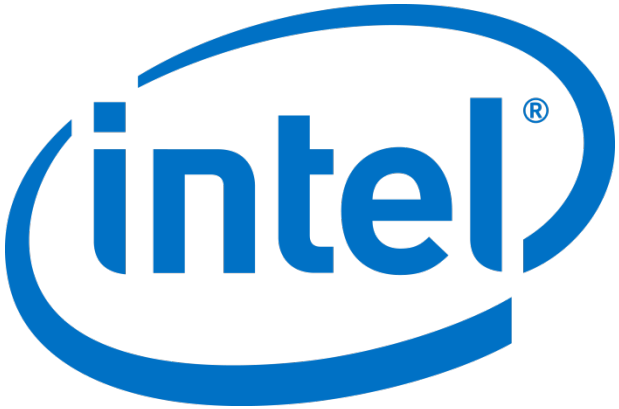
Mkdir

- If master and slave fails at the same time
  - If the update has been committed to master, master will retrieve update from local disk and resend to slave MDT. And slave MDT will redo it if the update does not exist in the local log.
  - If the update has been committed to slave, master will retrieve the update log from from slave, and redo it if needed(by comparing transno).
  - If the update has been committed both on master and slave, the slave will find out when master resend the update to the slave.

Recovery example

Rename (4 MDTs)

- If master MDT fails (or some slave MDTs fails at the same time)
  - If the update has not been committed to disk, client will replay the request.
  - If the update has not been committed to disk, and client crash.
    - Master MDT will retrieve the update record from the slave MDT, and know it needs to redo the update by checking the transno. Then send updates to other slave MDTs, and these slave MDTs will redo the updates check whether these updates exist in the local log
  - If the update has been committed to disk, master MDT will read the update log locally and resend the update to the slave MDT.
- If one of slave MDT fails or multiple slave MDT fails
  - If update has not been committed to disk, master MDT will replay the request by normal ptlrpc replay.
  - Otherwise, it does not need to do anything.


intel