LLOG (very) basics

Nathan Rutman

4/8/05

Log objects are used to maintain synchronized, persistent information on multiple systems. Generally, logs are written transactionally and cancelled when a commit on another system completes. Every log has an originator (the node that starts the transaction) and a replicator (the node that finishes/"cancels" the transaction). Individual log records are stored in log objects. Log objects are implemented as regular files (though journaled data) with file names of <inum>:<generation>.

The MDS writes some sort of transaction info into a record (usually an unlink record for a given object) along with the MDS-side operation in a single journal transaction. When the OST handles this operation it batches up "cancel" records and sends them back to the MDS to cancel the pending operation in the MDS llog. When the cancel commits then the MDS knows that the operation is completed on the OST side (the OST only sends the cancel after its own operations have committed). The "cancel" operation is merely sending a small record which says "operation N in llog <inum>:<generation> is done", and on the MDS side this only involves clearing a bit in the bitmap that marks the operation as still pending (llog_cancel_rec). If the MDS needs to do a reconnect to the OST (because of MDS or OST crash, disconnect) then it replays all of the pending operations in all of the logs (the llogs are per OSC btw) and the OST again sends cancel operations back to the MDS either when it completes the operation (if it was lost) or if the operation is already done (if the cancel was lost).