Metadata Refinements

Peter Braam

May 6, 2005

1 *Requirements from the Engineering Requirements Specification (formerly Architecture).*

In this document we describe a number of metadata handling improvements. The requirements of 1.1 and 1.2 are for detailed issues and do not need a separate HLD, although a good DLD is required.

1.1 Regular file creation

When a file is created, the MDS returns object id information. The objects will no longer be pre-created but only the object ids are managed by the MDS. The changes to be made are:

1. The MDS will not make pre-creation calls to the OST's to create objects. Objects are created upon first write.
2. The object ids handed out by the MDS are updated transactionally
3. objects beyond the last used object ids are removed from the OST's upon recovery, after client replay & resend completes.
4. The objects will store the owner, group in regular attributes and will store the MDS storeid (mds number, inode number) in an extended attribute.

1.2 Regular file unlink handling

The MDS will remove objects for files that are being unlinked. This is done asynchronously, but without delays. Clients will no longer make destroy calls to the OST's.

1.3 Attribute migration

The MDS will have up to date mtime, atime, ctime and size for inodes that are not open. This requires setting these attributes on the MDS after the file is closed.

1.4 OST - MDT unification (not required for cmd2)

The obdfilter and mdd code will be unified to provide a single device offering both api's simulteneously. In particular for objects that have no name in the MDS namespace will be accessed through the object namespace used on the OST.

1.5 Clustered MDS object handling (not required for cmd2)

When the MDS wishes to create objects on another MDS it will use object id's from a group assigned to this MDS without actually waiting for creations during an RPC. The objects will be placed in a directory structure as in 1.4. Directory entries referencing inodes on another MDS will only contain the object id.

1.6 Metadata transaction handling (not required for cmd2)

No RPC's will be made from within transactions.

2 Functional specification.

2.1 Attribute migration

3 Use cases.

1. A node configured for logging will write records
2. The granularity of the audit records can be changed or set at configure time
3. The log records can be read
4. The log records can be purged when processed

4 Logic specification.

The logic of the logging code itself is controlled by the SMFS plugin api.

5 State management.

6 Architectural alternatives (do not really belong in HLD)

7 Focus in inspection.