Auditing

Peter Braam

May 3, 2005

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

In this document we are describing how to generate llog entries for undo, replay and auditing. In addition we describe the other infrastructre required for the auditing component of Lustre.

1.1 Auditing

1.1.1 File set access

1. Audit logs will be stored as files in a Lustre file system. The root directory of this file system on the MDS is not ROOT but LUSTRE.
2. Mounting this LUSTRE fileset requires special permissions
3. LUSTRE contains a subdirectory called audit
4. audit contains subdirectories mds, oss, client
5. these directories contain one file for each mds, oss and client respectively

1.1.2 Audit logs

1. Audit logs are llog catalog objects
2. Audit logs can be generated for operations performed on client, OSS and MDS systems.
3. OSS and MDS audit logs are written locally and become the object of a file
   (a) OSS and MDS audit logs are updated transactionally, using the caches
4. Client audit logs are written stored on a round robin OSS
   (a) client audit log entries are made with synchronous RPCs

5. Files in Lustre with object that are llog catalogs or llog files
   (a) cannot be removed until the catalog is empty
   (b) allow a few ioctls to manipulate the catalog and its logs (such as canceling logs in the catalog, processing entries etc)
   (c) do not allow ordinary read-write operations

1.1.3 What is logged?

1. Granularity of auditing is per file system, directory (non-rec) or file
2. Audit logs can be limited to users in or not in a Unix group (e.g. don't audit executives, audit everyone else)
3. Clients only log operations that do not result in an RPC.
4. Log entries will contain fid and storage cookie of inodes to allow flexibility in the FID project
5. Filtering: audit logs can contain detailed logs of almost all access **including** operations that do not modify the file system. Examples:
   (a) log everything
   (b) failed authorization entries,
   (c) just of mount operations,
   (d) logging chdir is powerful (use intent on client).
   (e) Maybe a general bitmask can describe the granularity of the filter.

1.1.4 Special user

The special user is needed to reconstruct pathnames from fids, to access fids and objects directly and to perform other administrative tasks.

1. The special user has the capabilities to mount the LUSTRE fileset
   (a) check is in get_info handler during to get the fid of LUSTRE (instead of ROOT) during mount
2. The special user has the capability to run the fid->path operation on any file in the file syset
   (a) The storage cookie to path operation is based on searching for fids / inode numbers in directories - this operations requires the special user
3. The special user has the capability based on the GSS connection and its uid to access any object un-authorized by capabilities

   (a) The critical operation to allow is the one that extracts the EA from the objects which contains the MDS fid/storage cookie

4. The special user can cancel llog entries

1.1.5 Log processing

1. Audit entries are analyzed in user space and transferred to syslog by some daemon
2. An ioctl on the llog file allows entries in the llog or llogs in the catalog to be canceled (don't optimize this)
3. processed audit entries contain pathnames (not just fids)

1.1.6 General considerations

The three logs discussed are **very different** in nature and in processing requirements. For example, the audit log may contain read-only operations. The redo log only contains entries for entire file system transactions, while the undo log contains records for the component. During the DLD it is worth considering if a single format can be used, perhaps the records can have a bitmap indicating what fields are used in them.

The logs will be written as llog files, by an smfs thread. Hopefully the implementation of the redo log can provide guidelines for the other logs.

Some optmizations might be necessary, in particular a reasonable degree of concurrent access to the log is needed - intermezzo handled this with reserving enough space and then releasing the lock on the log.

The undo - redo operations must be absolutely and completely correct and testable separately.

A utility should be available to list logs and catalog elements, stating their name and purpose and other parameters, and to allow truncation of logs in a catalog and cancellation of records in an individual llog.

2 Functional specification.

2.1 Audit records

1. the audit records are created by an SMFS plugin. They are designed to be useful for ext3 when Lustre is not used.
   (a) the audit record that are written can be tuned to log
      i. all operations
      ii. all operations that incur permission failures
      iii. a sensible subset of operations, e.g. modifications
   (b) audit records can be written for

        i. the clients, MDS' and OST's or a subset of these
        ii. the entire file system or
        iii. for files in directories marked with a flag (non-recursively) or
        iv. for files marked with a flag. The flag is an EA (bit) - the EA's are propagated from the MDS to the OSS and clients when a file is accessed.

    (c) audit events recorded can be limited to operations performed by users in a unix group or NOT in a certain group
    (d) pathnames will be extracted from fids, no additional parent information is desirable in the audit log
    (e) the originating client / user of the logged command should be recorded in the log
    (f) an llog client api is created for writing a page full of records on the client and transferring it to the server where the llog is stored.
    (g) The audit llogs from clients are stored on OST nodes and the OST is allocated as #client-nid % #OST-count.
    (h) MDS inodes are placed in a directory /AUDIT/ on the MDS with a logical name referring to the client or target and the inodes use the llog objects to form files.
    (i) Normal Lustre mount with an additional fileset parameter in the mount command, mounts the lustre file system but with ROOT replaced by AUDIT. An audit analyzing client would mount hte file system and the AUDIT files.

```
mount -t lustre mtmnode:/fsname/AUDIT /audit/mt/pt
```

    (j) A special ioctl is possible on the files on the lustre client in this fileset to punch parts of the llog away.

2.2 Client auditing

1. Upon connect the server will open and possibly create a file in the AUDIT directory and assign an object to it on a round robin OST
    (a) Must use synchronous RPC's to the server node containing the log.
    (b) The server node may cache dirty client audit entries as part of a normal cached file system write
2. Only those operations are logged that use the cache, not those that result in immediate RPCs
3. Client auditing implies auditing of OSS and MDS

2.3 Pathname reconstruction

1. An extended attribute will contain a **single** parent store id (the *hardlink pathname reconstruction* problem is not addressed in this phase)
   (a) objects will contain the fid (see metadata roadmap file creation section on how to organize this)
   (b) A special MDS user will allow

2.4 Settings

1. Per file system settings are made in the disk structure of the obdfilter and mds.
2. Per file settings are in an EA
3. The settings contain:
   (a) what is audited and
   (b) for which group audits are skipped

3 Use cases.

1. A node configured for auditing 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.

5 State management.

6 Architectural alternatives (do not really belong in HLD)

7 Focus in inspection.