LFSCK1.5 Solution Architecture

1 Introduction

When a file is created on Lustre-2.x a FID (File IDentifier) is stored as part of the name entry in the parent directory. This is known as FID-in-dirent (File IDentifier in directory entry). With the FID-in-dirent, readdir on the MDT can fetch the FID from the directory page directly avoiding a costly lookup of the object LMA (Lustre Metadata Attributes) extended attribute stored on the inode. As a result, traversing the directory (such as "`ls`" and "`du`") with FID-in-dirent performs faster because FID lookups from the LMA are avoided. In addition, at file creation time, the FID of the parent directory and the name of the file are stored in the linkEA extended attribute on the inode. With the linkEA, any given FID can be parsed back to a full path from the root directory to the target file. This feature is valuable to ChangeLog based applications (i.e. "`lustre_rsync`") and when generating error messages or POSIX style pathname permission checks. Hard links to a regular file create the same FID-in-dirent and linkEA attributes and associated storage requirements.

Over the lifetime of an active filesystem, some FID-in-dirent and linkEA may become inconsistent or invalid as the result of on-disk corruption, after restoring from MDT file-level backup, or if the MDT filesystem was originally formatted under Lustre 1.8. Currently, if the MDT is upgraded from Lustre 1.8 (or following restoration from a MDT file-level backup,) the MDT will be missing the FID-in-dirent entries. This will reduce the performance of `readdir(3)` on the MDT. Additionally, for an MDT upgraded from Lustre 1.8 the linkEA is also unavailable and the 2.x "`lctl fid2path`" functionality will not be available for those files.

In LFSCK Phase 1.5 we will implement the functionality to verify and rebuild FID-in-dirent and linkEA on a single-MDT case. Additional operations will be added while the MDT is iterating over the objects table using functionality developed for OI Scrub. The FID-in-dirent name entry will be checked for consistency with the FID in the object LMA. Repair or rebuild of FID-in-dirent will be conducted as necessary. In addition, the name entry referenced by the object linkEA and the object linkEA pointer will be verified as valid. An unmatched or redundant object linkEA will be removed, and the missed object linkEA will be added. After an upgrade from Lustre 1.8, the inodes with IGIF FIDs (a special subset of FIDs that map directly to the underlying ldiskfs filesystem's inode and generation number) will store the IGIF FID in the LMA xattr on the inode and in the FID-in-dirent and linkEA.

2 Solution Requirements

2.1 Online FID-in-dirent and linkEA consistency check/repair

An online LFSCK must not render the system unavailable during LFSCK. The LFSCK for FID-in-dirent and linkEA consistency check/repair can be performed with clients concurrently accessing the filesystem. All normal operations (except for the cases that the target objects are not repaired or rebuilt yet, such fid2path against the non-repaired file), can be processed as without FID-in-dirent and linkEA consistency check/repair cases, and correctness must be assured.

LFSCK depends on the offline e2fsck to guarantee the on-disk data consistency for ldiskfs-based MDTs and OSTs. The LMA must be valid (trusted), and the object hardlink count (nlink) must match the count of the name entries which point to the target object.

2.2 LFSCK user space control

LFSCK user space control tools implemented in LFSCK Phase I will be extended to controlling the LFSCK for FID-in-dirent and linkEA consistency checking.

2.3 Support to resume from break point

As an online Lustre tool, the LFSCK may execute for prolonged periods in the background. To ensure a full system scan completes in a timely manner even if the MDT restarted due to a system crash LFSCK Phase 1.5 must support resuming from a checkpoint after a restart.

2.4 Rate control

The LFSCK for FID-in-dirent and linkEA consistency may affect the performance of some Lustre operations. The effect of the impact is unclear at this stage. A rate-control will be added to reduce the system load created by LFSCK 1.5. LFSCK phase I includes a rate-control mechanism and this will be extended to include FID-in-dirent and linkEA consistency checking.

3 Use Cases

3.1 Rebuild FID-in-dirent after MDT file-level backup/restore

Typically, the FID-in-dirent cannot be backed up by MDT file-level backup/restore because there is no POSIX API for extra data stored in the directory entry. As a result, after the MDT is restored from file-level backup, an administrator should run the LFSCK to rebuild the FID-in-dirent in addition to rebuilding the OI file.

3.2 Generate FID-in-dirent and linkEA for the MDT upgraded from 1.8-based device

The FID-in-dirent and linkEA are new features for Lustre-2.x. For the MDT upgraded from Lustre 1.8, neither FID-in-dirent nor linkEA is available. An administrator can run the LFSCK to store the IGIF FID in the inode's LMA xattr, and generate FID-in-dirent and linkEA for IGIF files.

3.3 Lustre FID-in-dirent and linkEA consistency routine checking

fsck is recommended as routine for local filesystems, the Lustre administrator should perform periodic Lustre consistency check identify and resolve inconsistencies. The FID-in-dirent and linkEA consistency check/repair is part of such routine check.

4 Solution Proposal

4.1 Identifying Inconsistency

The LFSCK for FID-in-dirent and linkEA consistency needs to scan the whole MDT. How to scan the MDT comprehensively and efficiently is technical issue to be resolved. Two choices present themselves:

- Namespace-based directory traversal
  The FID-in-dirent and linkEA are directory-based features. So traversing the Lustre ROOT directory recursively is the most intuitive method to scan the MDT device for the LFSCK for FID-in-dirent and linkEA consistency check/repair. However, relying exclusively on namespace-based directory traversal is not sufficient. As an online tool there may be rename operations from clients during the LFSCK that may cause the LFSCK to miss objects resulting in incomplete scan. In addition, namespace-based traversal can result in a large amount of random IO to read inodes and directories from disk that typically negatively effects performance.

- Object table-based iteration
  Object table-based iteration has been implemented during LFSCK Phase I. However, the Object table-based iteration executes without Lustre namespace knowledge. Check and repair FID-in-dirent and linkEA consistency without namespace information is not possible.
  For example, the LFSCK needs to know whether the given object is correctly referenced by some name entry, and whether its FID is stored in the parent directory name entry. So it needs to find its parent. The answer is linkEA. But linkEA is not always valid; particularly if upgrading an MDT from 1.8-based device where there is no linkEA for IGIF objects. So only the method of parsing parent from child is not enough, the LFSCK needs to parse child from parent, which requires directory traversal.

The solution is to combine the two methods above to perform the LFSCK for FID-in-dirent and linkEA consistency check/repair in a piecewise linear manner, to minimize random IO. The process will be as follows:

1. LFSCK is driven by object table-based iteration. The OSD layer object table-based iteration will scan and return each object in flat order.
2. If the object returned by the object table-based iteration is a directory, then the LFSCK will traverse the directory with namespace-based, non-recursive order. LFSCK ignores rename operations during the directory traversal because the object table-based iteration can guarantee that all the objects can be processed. Reading the directory blocks is a small fraction of the data needed for inodes they reference. In addition, child inodes are typically allocated following the parent directory inode, so for many directories the children inodes will have already been prefetched by the OI Scrub operation.
3. When the LFSCK traverses the directory, it checks every name entry in the directory. For each name entry, LFSCK finds related child object and compares the FID-in-dirent with the FID in the object LMA. Inconsistencies are repaired. For each child object, LFSCK also checks if there is valid linkEA that back points to the parent object, and repairs the inconsistency if present. This mechanism also verifies the `.` and `..` entries of each directory.
4. Continue to process next object by the object table-based interaction until all the objects have been processed.

The FID-in-dirent and linkEA are namespace-based features and will only affect the objects that are visible to clients. These objects are under the global filesystem `ROOT` directory, but not the local root of MDT backend filesystem. As a result, when LFSCK gets a directory object from low layer object table-base iteration it will first verify the directory is part of the client visible namespace by checking if it has an LMA FID. In the rare case of an IGIF directory object that has not been provided with an LMA FID, LFSCK will check if it is under the `ROOT` directory by `lookup("..")` recursively. If LFSCK finds a local object, it will be ignored. Following an upgrade and a completed LFSCK run all client namespace directories will have an LMA FID and will not need this extra verification.

System files/directories maybe optionally flagged in the inode (e.g. if looked up by `FID_SEQ_LOCAL_NAME` or `FID_SEQ_LOCAL_FILE`) to avoid being returned by otable iteration entirely.

4.2 Where to Fix the Inconsistency

The FID-in-dirent is OSD layer element that is invisible to layers above. Check and repair the FID-in-dirent must be processed inside the OSD.

However, linkEA is namespace related, and implemented in MDD layer. Check and repair the linkEA will be processed by LFSCK is in parallel to the MDD layer.

4.3 How to Fix the Inconsistency

Since the FID-in-dirent and linkEA belong to different code layers of the metadata stack, they will be repaired separately. This means that the FID-in-dirent and linkEA will be checked and repaired in different layers by different mechanism. It does not means repeat scanning or duplicate processing.

4.3.1 Check and repair FID-in-dirent inconsistency

The basic strategy is to verify the FID-in-dirent data during directory traversal while LFSCK is running. For each directory processed in the OSD, the d_ino is used to fetch the inode, and the dirent FID is compared with the FID in the object LMA. There are several cases and resolutions to consider:

1. The FID-in-dirent exists, but does not match the FID in LMA.
   Replace the FID-in-dirent with the LMA FID directly.
2. The FID-in-dirent does not exist, but the LMA FID exists (MDT restored from file-level backup or 1.8
   First, try to append the FID after its name entry in the parent directory directly. In the case where there is not sufficient space in its name entry to hold the FID remove the name entry from the parent directory first. Then, re-insert the name entry with the FID into the parent directory. During such remove-insert processing, an exclusive lock must be held against the directory entry to prevent concurrent access of related name entrys. This behaviour is similar to a MDD layer rename operation, but it is processed in the OSD layer.
3. The FID-in-dirent exists, but the FID in LMA does not exist.
   OI scrub will restore the LMA FID from the FID-in-dirent. This rare case is anticipated when LMA xattr on the device is corrupted, and/or removed by e2fsck. It may be possible that the system crashed after the LFSCK has already added FID-in-dirent for the 1.8-IGIF object, but before the OI scrub initialized the inode LMA and OI mappings.
4. The FID-in-dirent does not exist, and the LMA FID does not exist.
   This is for the case of MDT upgraded from 1.8-based device. This case is treated in the 'IGIF Objects Handling for FID-in-direcnt and linkEA' section below.

4.3.2 Check and repair linkEA consistency

The strategy is to compare the linkEA with the parent FID and the name entry in the parent directory. If an inconsistency is detected, the parent FID and name are trusted over entries in the linkEA, since e2fsck will ensure that the directory entry to inode mapping and nlink are consistent. The process is as follows:

1. If the LFSCK receives a directory object from a low layer object table-based iteration it traverses the directory in namespace-based order. For each name entry, the child object is identified the validity that linkEA points back to the parent directory object. If the child object does not contain a linkEA that back points to the parent FID and child name, then add a new linkEA with the parent FID and the child name.
2. If a non-directory object has multiple hard links (regardless of the number of linkEA entries), then all of the links need to be validated and the corresponding linkEA verified. As an optimization, these multiple linked objects can be pinned in RAM to avoid multiple repeated scans of the object. While the number of hard-linked files in Lustre is typically a small fraction of the total, if the number of such pinned objects grows too large, some objects can be dropped from the cache at the expense of requiring their links to be revalidated the next time the object is loaded.

4.3.3 IGIF Objects Handling for FID-in-dirent and linkEA

For lustre-2.x, objects with IGIF FIDs must be treated differently. They have no FID in LMA, no OI mappings in the OI files, and no FID-in-dirent nor linkEA. Additional logic on the MDT is required to process these IGIF objects. To unify the normal RPC processing, IGIF object is treated as a normal FID, so additional work is performed as following:

1. Add the IGIF FID to the object LMA during OI Scrub, as is the case for normal 2.x FIDs.
2. Add the IGIF FID to the OI file.
3. Add IGIF to the name entry in the parent directory as FID-in-dirent, during normal FID-in-dirent processing.
4. Add linkEA with the child name and the parent FID, during normal linkEA processing

With the above process, the IGIF object obtains most of the attributes of a normal object. Even after MDT file-level

backup/restore, the IGIF can be reserved as normal FID. The OI scrub will update/rebuild the OI files when needed to guarantee lookup-by-FID still works after MDT file-level backup/restore of IGIF files. So both the clients and the MDT can process the IGIF object as normal FID cases.

4.3.4 Build FID-in-dirent and linkEA through rename operation

During a rename operation Lustre-2.x the FID is added to the target object name entry in the target parent directory. And the target object name together with the target parent FID will be added to target object linkEA, and the old linkEA will be removed if it existed. So even through there was rename operations during the LFSCK for FID-in-dirent and linkEA consistency check/repair, and caused some objects missed the chances of checking/repairing FID-in-dirent and linkEA consistency under both the source parent directory and the target parent directory, the rename operation itself can guarantee that related FID-in-dirent and linkEA against the target objects are properly processed.

4.4 Resume from latest checkpoint

An OSD internal file will be introduced (named "`lfsck_namespace`") on the MDT to manage internal state for the LFSCK components. The LFSCK parameters, status, progress, statistics, and so on, will be recorded in the file. The file "`lfsck_namespace`" will be updated (in memory) for every object check/repair. For performance reasons, the updated "`lfsck_namespace`" will be synced to disk periodically. The default sync interval (or sync cycle) is 60 seconds. At each sync to disk, a new checkpoint is created that includes the LFSCK current position. If the MDS crashes during LFSCK, LFSCK will be resumed automatically or manually from the latest checkpoint recorded by the file "`lfsck_namespace`". At most one sync cycle work may be lost because of a crash.

4.5 Trigger strategy for FID-in-dirent and linkEA consistency check/repair

As part of the work for the LFSCK user space tools, `lctl` commands will be created or modified to control the kernel space LFSCK ("`lfsck_start`" and "`lfsck_stop`") from user space. An administrator can automate common tasks by scripting these commands, and trigger them with a timer (i.e. `crond` or `atd`). This will satisfy the recommendation of running LFSCK periodically.

In kernel space, during the normal RPC processing, it is simple to verify if the FID-in-dirent or linkEA is missed. Missing FID-in-dirent or linkEA typically correspond to an MDT having been restored from file-level backup or upgraded from 1.8-based device. Since they both userspace-triggered events, it is unnecessary include detecting these conditions in the kernel.

However, it is inconvenient for the RPC services threads to verify existing FID-in-dirent and linkEA are valid. Introducing additional logic will affect the original features benefit, such as verifying FID-in-dirent during a readdir will undermine the optimization achieved by FID-in-dirent feature. As a result, triggering the LFSCK for FID-in-dirent and linkEA consistency will not be performed automatically. FID-in-dirent and linkEA consistency will be processed immediately after a restore during normal OI Scrub and LFSCK processing, or during a manually triggered run of LFSCK.

4.6 Rate control

The existing LFSCK rate control framework applies to object table-based iteration. LFSCK for FID-in-dirent and linkEA consistency combines the namespace-based directory traversal and object table-based iteration. The objects scanned by any method must be counted so the LFSCK speed can be controlled.

5 Unit/Integration Test Plan

5.1 Start/stop FID-in-dirent and linkEA consistency check/repair through userspace commands

An administrator can perform FID-in-dirent and linkEA consistency check/repair by `lctl` command. This task can be stopped/paused through `lctl` command at the administrators will.

5.2 Monitor FID-in-dirent and linkEA consistency check/repair

An administrator can view the current LFSCK information for FID-in-dirent and linkEA consistency. Available information includes: status, speed, progress, statistics. Information is available through lproc.

5.3 The FID-in-dirent can be rebuilt after the MDT is restored from file-level backup

Verify the FID-in-dirent is available after the MDT restored from file-level backup once LFSCK is complete.

5.4 Build FID-in-dirent and linkEA for the MDT upgraded from 1.8-based device

Verify FID-in-dirent and linkEA are available for the MDT upgraded from 1.8-based device once LFSCK is complete, including files with multiple hard links.

5.5 Verify files with multiple hard links

Verify a filesystem with multiple hard links to each file.

5.6 Resume FID-in-dirent and linkEA consistency check/repair from the latest checkpoint

Demonstrate that FID-in-dirent and linkEA consistency check/repair begins from the latest checkpoint by default when it is restarted.

5.7 Rate control for FID-in-dirent and linkEA consistency check/repair

An administrator can specify the max speed for FID-in-dirent and linkEA consistency check/repair when starting LFSCK. Rate-limit can be viewed and adjusted during the LFSCK processing.

5.8 The Lustre system is available during LFSCK for FID-in-dirent and linkEA consistency check/repair

The LFSCK for FID-in-dirent and linkEA consistency executes concurrently with external services. Assuming the target has no on-disk corruption. Performance of concurrent services may be affected, but correctness is guaranteed.

6 Acceptance Criteria

The acceptance test will be performed against Lustre-Master or later. The LFSCK for FID-in-dirent and linkEA consistency will be accepted if meets the requirements:

1. All test scenarios are successfully demonstrated.