Final Report for the *FID-in-dirent and LinkEA Subproject* of the LFSCK project of the SFS-DEV-001 Contract Amendment #1

Revision History

Date	Revision	Author
06/05/13	Original	R. Henwood
06/20/13	Updated links, combined tables.	R. Henwood
06/21/13	Lustre trademark compliance.	R. Henwood
06/26/13	Disclaimer included	R. Henwood

1. Contents

2. Executive Summary

This document finalizes the activities undertaken during the Lustre File System Checker, Sub Project 3.1.5: FID-in-Dirent and LinkEA project within the OpenSFS Lustre* software development contract SFS-DEV-001 signed October 10th 2012.

Notable highlights of this project include:

- Demonstrated restoring from backup and a scanning rate of 80K objects per second for a directory of 8 million files.
- FID-in-Dirent and LinkEA was implemented with 6151 lines of code and was completed and landed on Lustre software master branch for inclusion into Lustre software release 2.4 on May 31st 2013.
- All assets from the project are attached to the public ticket LU-1866.

3. Statement of Work

LFSCK 1.5 FID-in-dirent and LinkEA implements the functionality to verify and rebuild FID-in-dirent (File IDentifier in directory entry) and LinkEA (Link Extended Attribute) entries. To achieve this, the functionality of the inode iterator (see LFSCK 1: OI Scrub) is enhanced. This enhancement will ensure the FID-in-dirent name entry is consistent with the FID in the inode Lustre file system Metadata Attributes (LMA) on Lustre file system version 2.x. If the name is found to be inconsistent, it is repaired or rebuilt as necessary.

LFSCK 1.5 also verifies the name entry for normal files (non-directory) is correctly referenced by the inode LinkEA and the inode LinkEA points to the valid name entry. An unmatched or redundant inode LinkEA is removed on discovery and a correct or missing inode LinkEA is added.

Finally, the FID-in-dirent scanning process adds IGIF FIDs to the dirent for upgraded 1.8 file systems. This additional step avoids the need to look up the inode during a readdir() operation.

4. Summary of Scope

1. In Scope

- Kernel-space FID-in-dirent and LinkEA check and repair of single MDT.
- User-space tools to control LFSCK for FID-in-Dirent and LinkEA consistency.
- Administrative documentation in the form of a man page and update to Intel Lustre software version 2.x manual.

2. Out of Scope

---

*Other names and brands may be the property of others.

- The LFSCK for FID-in-Dirent and LinkEA will be implemented against the inode iterator introduced in LFSCK phase I (Subproject 3.1: inode iterator and OI Scrub). Only ldiskfs OSDs will be implemented, tested and landed for this phase.
- LFSCK for FID-in-Dirent and LinkEA consistency will act only on a single MDT at this phase.
- Detection and resolution of internal file-system inconsistencies is not within LFSCK scope.

The complete scope statement is available at:

http://wiki.opensfs.org/images/a/a0/LFSCK_FID-in-dirent_LinkEA_ScopeStatement.pdf

5. Summary of Solution Architecture

When a file is created on Lustre file system version 2.x a FID is stored as part of the name entry in the parent directory. This is known as FID-in-dirent. With the FID-in-dirent feature, `readdir(2)` on the MDT can fetch the FID from the directory entry directly avoiding a costly lookup of the object LMA 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 file system was originally formatted with Lustre file system version 1.8. Currently, if the MDT is upgraded from Lustre 1.8, or following restoration from a MDT file-level backup, the MDT will lack the FID-in-dirent entries. This will reduce the performance of `readdir(3)` on the MDT. Additionally, an MDT upgraded from Lustre file system version 1.8 will lack the LinkEA 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 file system version 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.

1 Acceptance Criteria

FID-in-Dirent and LinkEA patch will be accepted as properly functioning if:

1. Start/stop FID-in-dirent and LinkEA consistency check/repair through userspace commands.
2. FID-in-dirent and LinkEA consistency check/repair progress can be monitored.
3. The FID-in-Dirent is rebuilt after the MDT is restored from file-level backup.
4. The FID-in-Dirent and LinkEA is rebuild after MDT is upgraded from 1.8-based device.
5. Files with multiple hard links are verified.
6. FID-in-Dirent and LinkEA check can be restarted from a checkpoint.
7. FID-in-Dirent and LinkEA rate of scanning is controllable.
8. The file system is available during a FID-in-Dirent and LinkEA consistency scan.

The complete Solution Architecture is available at:

http://wiki.opensfs.org/images/4/43/LFSCK_FID-in-dirent_LinkEA_SolutionArchitecture.pdf

6. Summary of High Level Design

FID-in-Dirent and LinkEA scanning (along with the other phases of LFSCK design) is a highly complex implementation task. The complete High Level Design is document weighing in at over 5000 words. This document describes in detail the following considerations:

- Scanning the MDT OSD device.
- Upgrading Lustre* file system version 1.8 objects.
- Filtering out objects not visible to the client.
- Finding a missing or inconsistent FID-in-Dirent and LinkEA.
- LFSCK tracing.
- LFSCK user space controls.
- Monitoring of LFSCK.
- LFSCK Rate control.
- Initial OI Scrub to verify local server objects.

---

*Other names and brands may be the property of others.

- Handling IGIF objects.
- Repair inconsistent FID-in-Dirent and LinkEA.
- Concurrent link, unlink, and rename operations can be performed during LFSCK.
- Race control between the LFSCK and other operations.
- Race control during FID-in-Dirent and LinkEA check/repair.
- Compatibility with other versions of Lustre software.
- API Changes.
- New parameter value in dt_it_ops::init.

The complete High Level Design is available at:

http://wiki.opensfs.org/images/4/4f/LFSCK_FID-in-dirent_LinkEA_HighLevelDesign.pdf

5. Summary of Implementation

LFSCK 1.5: FID-in-Dirent and LinkEA is implemented in the following patches:

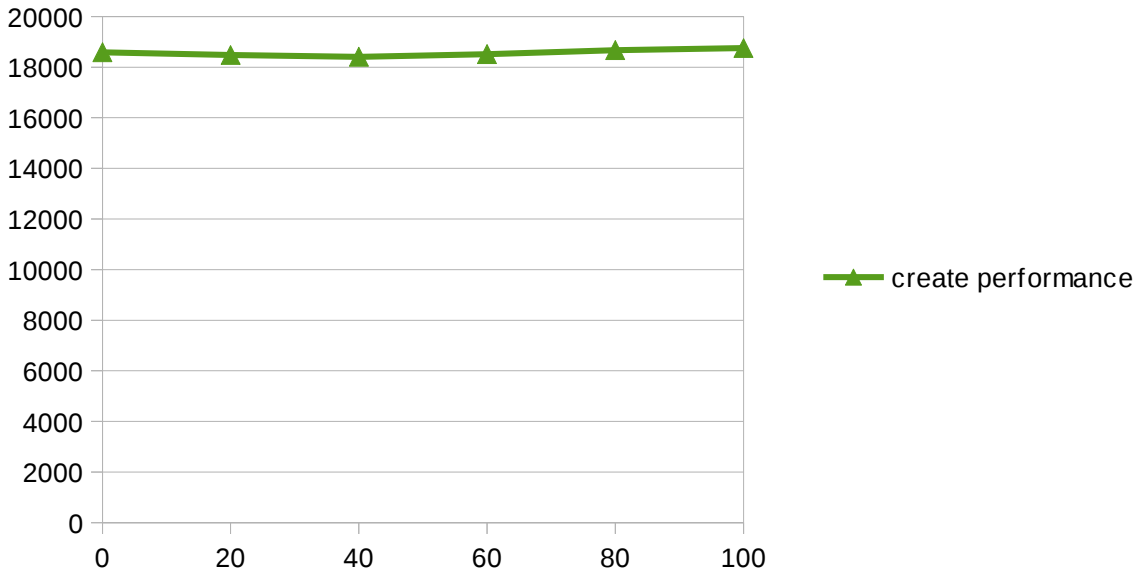
Commit	Patch title	Review
bad2b3e	LU-1866 lfsck: LFSCK for namespace consistency (3)	4914
c34e632	LU-1866 lfsck: LFSCK for namespace consistency (2)	4913
c5a411d	LU-1866 lfsck: FID-in-{dirent,LMA} check and repair	4912
21a9805	LU-1866 lfsck: LFSCK 1.5 user space control	4911
3d6a1b7	LU-1866 lfsck: LFSCK for namespace consistency (1)	4910
e5a6d30	LU-1866 lfsck: LFSCK main engine	4909
c961228	LU-1866 lfsck: general framework for LFSCK 1.5	4908
21b0820	LU-1866 lfsck: ancillary work for namespace LFSCK	4907
6e6357d	LU-1866 lfsck: enhance otable-based iteration	4906
4a88dc8	LU-1866 osd: FID-in-LMA and OI files	4904
073a1b0	LU-1866 scrub: initial OI scrub	4903
f4ea7b6	LU-1866 osd: ancillary work for initial OI scrub	4902
204c513	LU-1866 fid: cleanup object visibility definition and check	4901
f4547f0	LU-1866 misc: fix some issues found during LFSCK	5046

6. Summary of Demonstration

LFSCK 1.5: Successfully completed the following Demonstration criteria:

- Correctness:
- sanity-lfsck.sh

- sanity-scrub.sh
- sanity, sanityn, replay-single, conf-sanity, recovery-small, replay-ost-single, insanity, sanity-quota, sanity-sec, lustre-rsync-test, lnet-selftest, and mmp
- Performance of LFSCK 1.5 on a 2.4 MDT with routine consistency checking.
- Performance of LFSCK 1.5 on a 2.4 MDT restored from file-level backup with routine consistency checking.
- Performance of LFSCK 1.5 on a 2.4 MDT upgraded from 1.8.
- LFSCK performance under load.
- LFSCK impact on system performance:



As can be seen in the figure above, adjusting the speed control to fractions of full speed apparently has a negligible effect on the system under normal operation. This result may indicate that the CPU is sufficiently powerful to service both LFSCK namespace operations and normal workload operations simultaneously.

Complete results are available here:

http://wiki.opensfs.org/images/9/9c/LFSCK_FID-in-dirent_LinkEA_DemonstrationMilestone.pdf

7. Delivery

A complete list of code reviews and landings is included in section 5: Summary of Implementation. The work, landing and designs are recorded on the ticket LU-1866

https://jira.hpdd.intel.com/browse/LU-1866

7. Documentation

Documentation was completed as part of issue and landed on change http://review.whamcloud.com/5801 recorded against https://jira.hpdd.intel.com/browse/LUDOC-85.