LFSCK4 Landing milestone

Patches to enhance the performance of the previous LFSCK phases were landed over a period of approximately six months with the final outstanding patch landing on May 1st, 2015. The complete list of patches created by Intel and their commitment into the Lustre Master is recorded below.

9ce1fdd	LU-1452 scrub: OI scrub skips uninitialized groups
2c5b57c	LU-1453 scrub: auto trigger OI scrub more flexible
2d3d452	LU-1453 scrub: rename confused name full_scrub_speed
81be387	LU-5682 lfsck: optimize ldlm lock used by LFSCK
0f48753	LU-6177 lfsck: calculate the phase2 time correctly
ad40fea	LU-6350 lfsck: lock object based on prediction for bad linkEA
f3ea0ce	LU-6351 lfsck: check object existence before using it
5b3eef2	LU-6343 lfsck: locate object only when necessary
e6cb857	LU-6322 lfsck: show start/complete time directly
e9a6c24	LU-6317 lfsck: NOT count the objects repeatedly
d381825	LU-6316 lfsck: skip dot name entry

LFSCK4 Performance Demonstration Milestone

LFSCK 4: Performance has successfully completed both functional Acceptance and Performance tests. The performance results recorded herein illustrate performance expectations are exceeded during online operation and under load. In addition, LFSCK 4 has been shown to meet or exceed expectations running in a multiple MDT environment.

Demonstration on OpenSFS test cluster

This series of tests is intended to demonstrate the performance enhancements delivered as part of the LFSCK 4 project. No new functionality is included in this project and Correctness Test Coverage is included to illustrate no regressions are present. This demonstration plan compares LFSCK 3 and LFSCK 4 performance.

Correctness Test Coverage

The sanity-lfsck.sh and sanity-scrub.sh scripts are standard review tests that will be run and recorded. All previous LFSCK functionality is also tested with these scripts. For the specific case of MDT-MDT consistency, the test cases are:

- test 2e: The namespace LFSCK can detect inconsistent remote MDT-object's linkEA and repair it.
- test 6a: LFSCK can resume from the last checkpoint which is at the first-stage scanning.
- test 6b: LFSCK can resume from the last checkpoint which is at the second-stage scanning.
- test 7a: Non-stopped LFSCK can auto resume (the first-stage scanning) after the MDS restart.
- test 7b: Non-stopped LFSCK can auto resume (the second-stage scanning) after the MDS restart.
- test 9a: LFSCK speed is controllable during the first-stage scanning.
- test 9b: LFSCK speed is controllable during the second-stage scanning.
- test 10: During the LFSCK check/repair system inconsistency, the client can still access the system normally.
- test 22a: Repair unmatched name entry and MDT-object pairs (1). The parent_A references the child directory via some name entry, but the child directory back references another parent_B via its ".." name entry. The parent_B does not exist. Then the namespace LFSCK will repair the child directory's ".." name entry to reference the parent_A.
- test 22b: Repair unmatched name entry and MDT-object pairs (2). The parent_A references the child directory via the name entry_B, but the child directory back references another parent_C via its ".." name entry. The parent_C exists, but there is no the name entry_B under the parent_C. Then the namespace LFSCK will repair the child directory's ".." name entry and its linkEA to reference the parent_A.
- test 23a: Repair dangling name entry (1). The name entry is there, but the MDT-object for such name entry does not exist. The namespace LFSCK should find out and repair the inconsistency as required.
- test 23b: Repair dangling name entry (2). The object_A has multiple hard links, one of them corresponding to the name entry_B. But there is something wrong for the name entry_B and cause entry_B to references non-exist object_C. During the first-stage scanning, the LFSCK will think the entry_B as dangling, and re-create the lost object_C. When the LFSCK comes to the second-stage scanning, it will find that the former re-creating object_C is not proper, and will try to replace the object_C with the real object_A.
- test 23c: Repair dangling name entry (3). The object_A has multiple hard links, one of them corresponding to the name entry_B. But there is something wrong for the name entry_B and cause entry_B to references non-exist object_C. During the first-stage scanning, the LFSCK will think the entry_B as dangling, and re-create the lost object_C.

And then others modified the re-created object_C. When the LFSCK comes to the second-stage scanning, it will find that the former re-creating object_C maybe wrong and try to replace the object_C with the real object_A. But because object_C has been modified, so the LFSCK should NOT replace it to keep the data.

- test 24: Repair multiple-referenced name entry. Two MDT-objects back reference the same name entry via each own linkEA entry, but the name entry only references one MDT-object. The namespace LFSCK will remove the linkEA entry for the MDT-object that is not recognised. If such MDT-object has no other linkEA entry after the removing, then the LFSCK will add it as orphan under the .lustre/lost+found/MDTxxxx/.
- test 25: Repair invalid file type. The file type in the name entry does not match the file type claimed by the referenced object. The LFSCK will update the file type in the name entry.
- test 26a: Repair orphan MDT-object (1). The local name entry (back referenced by the MDT-object) is lost. The namespace LFSCK will add the missing local name entry back to the normal namespace.
- test 26b: Repair orphan MDT-object (2). The remote name entry (back referenced by the MDT-object) is lost. The namespace LFSCK will add the missing remote name entry back to the normal namespace.
- test 27a: Recreate the lost parent directory (1). The local parent (referenced by the MDT-object linkEA) is lost. The namespace LFSCK will re-create the lost parent as orphan.
- test 27b: Recreate the lost parent directory (2). The remote parent (referenced by the MDT-object linkEA) is lost. The namespace LFSCK will re-create the lost parent as orphan.
- test 29a: Repair invalid nlink count (1). The object's nlink attribute is larger than the object's known name entries count. The LFSCK will repair the object's nlink attribute to match the known name entries count.
- test 29b: Repair invalid nlink count (2). The object's nlink attribute is smaller than the object's known name entries count. The LFSCK will repair the object's nlink attribute to match the known name entries count.
- test 29c: Repair invalid nlink count (3). There are too many hard links to the object, and exceeds the object's linkEA limitation, as to NOT all the known name entries will be recorded in the linkEA. Under such case, the LFSCK should skip the nlink verification for this object.
- test 30: Recover the orphans from backend /lost+found. The namespace LFSCK will move the orphans from backend /lost+found directory (that is only valid for ldiskfs based backend) to normal client visible namespace or to the global visible ./lustre/lost+found/MDTxxxx/ directory.
- test 31a: Repair invalid name hash for striped directory (1). For the name entry under a striped directory, if the name hash does not match the shard (the case that some name entry should be inserted into other non-first shard, but inserted into the first shard by wrong), then the LFSCK will repair the bad name entry.
- test 31b: Repair invalid name hash for striped directory (2). For the name entry under a striped directory, if the name hash does not match the shard (the case that some name entry should be inserted into other non-second shard, but inserted into the second shard by wrong), then the LFSCK will repair the bad name entry.

- test 31c: Re-generate the lost master LMV EA for striped directory. For some reason, the master MDT-object of the striped directory may lost its master LMV EA. If nobody created files under the master directly after the master LMV EA lost, then the LFSCK should re-generate the master LMV EA.
- test 31d: Set broken striped directory (modified after broken) as read-only. For some reason, the master MDT-object of the striped directory may lost its master LMV EA. If somebody created files under the master directly after the master LMV EA lost, then the LFSCK should NOT re-generate the master LMV EA, instead, it should change the broken striped directory as read-only to prevent further damage.
- test 31e: Re-generate the lost slave LMV EA for striped directory (1). For some reason, the first slave MDT-object of the striped directory (that resides on the same MDT as the master MDT-object) lost its slave LMV EA. The LFSCK should re-generate the slave LMV EA.
- test 31f: Re-generate the lost slave LMV EA for striped directory (2). For some reason, the non-first slave MDT-object of the striped directory (that resides on different MDT as the master MDT-object) lost its slave LMV EA. The LFSCK should re-generate the slave LMV EA.
- test 31g: Repair the corrupted slave LMV EA. For some reason, the stripe index in the slave LMV EA is corrupted. The LFSCK should repair the slave LMV EA.
- test 31h: Repair the corrupted shard's name entry. For some reason, the shard's name entry in the striped directory may be corrupted. The LFSCK should repair the bad shard's name entry.

Performance Test Context

All tests require a populated filesystem in order to have something to scan. The filesystem will be created and populated with the following set of files: there are M MDTs, and for each MDT, there are N sub-roots, each sub-root contains 100K objects, including:

1. 78% (0-striped) regular files under the sub-root;
2. 3% local sub-dirs and each contains 5 (0-striped) regular files;
3. 0.4% 2-linked objects;
4. 0.3% remote sub-dirs and each contains 4 (0-striped) regular files;
5. 0.3% 2-striped sub-dirs (0.3% master objects plus 0.6% slave objects) and each contains 4 (0-striped) regular files with "all_char" stripe_hash.

Run namespace lfsck (with or without repairing) to scan the test unit without other system load. It will test the cases of M=2,4, 6, 8, and N=20,40, 60, 80, ...

Performance Test Coverage

The following scenarios will be tested:

1. Performance comparison between LFSCK 3 and LFSCK 4 running against multiple MDTs without inconsistencies.

This will provide a control benchmark for LFSCK 3 scanning. LFSCK 3 includes support for DNE striped and remote directories consistency checking (also known as MDT-MDT consistency checking). This test will ensure that the scanning rate across multiple MDTs within striped directories is consistent with expectations. The aggregate LFSCK scanning performance should scale as additional MDTs with allocated objects are added to the filesystem.

2. Performance comparison between LFSCK 3 and LFSCK 4 running against multiple MDTs with inconsistencies (lost linkEA).

This test will scan the full filesystem on all MDTs to look for inconsistencies in the filesystem namespace, including MDT-MDT inconsistencies, as is done in test #1. The intent of this test is to measure performance when the filesystem needs to repair a large number of inconsistencies and update each modified file on disk, in comparison to the "clean" case of test #1 which is a read-only test.

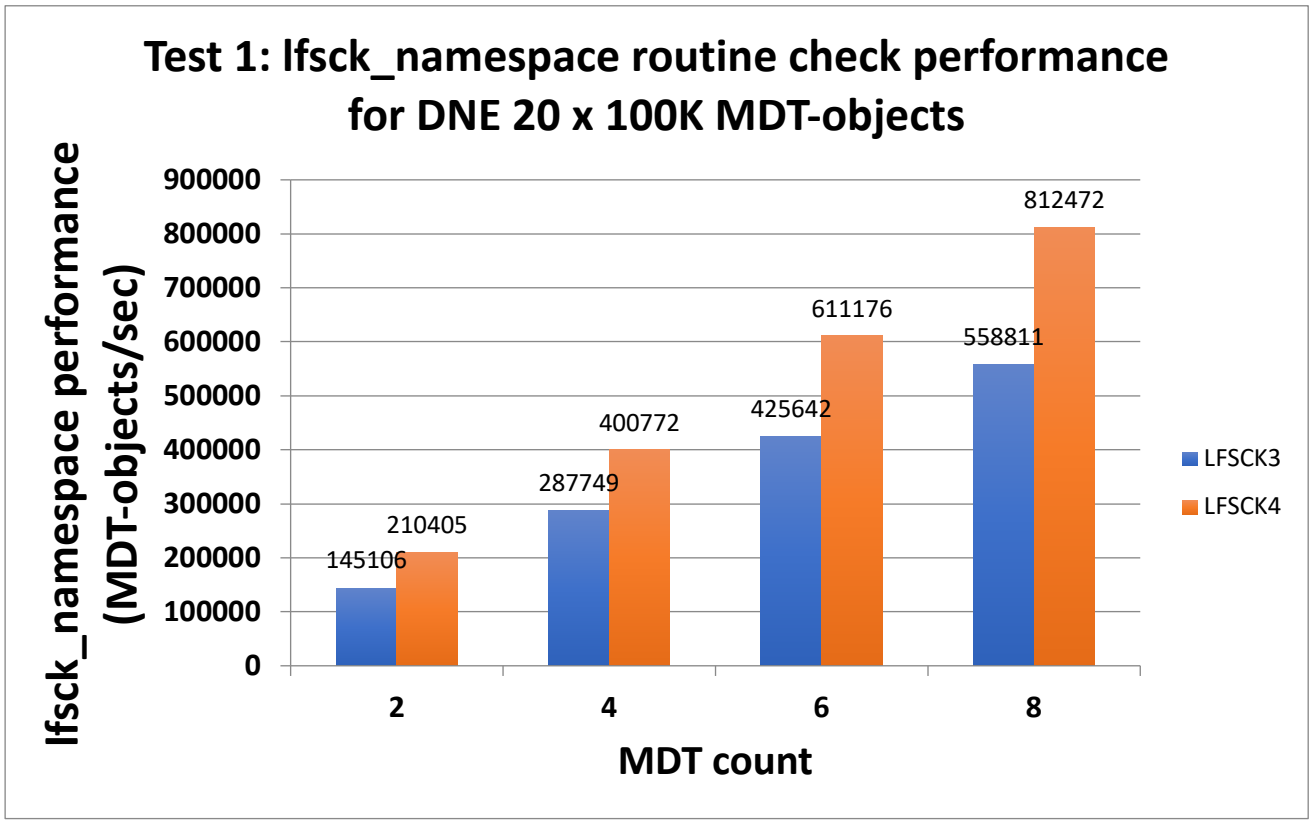
In this case, the filesystem has been intentionally corrupted during the filesystem population step by not storing the `link` xattr on each file in the filesystem using the `OBD_FAIL_LFSCK_NO_LINKEA` fault injection hook. This type of inconsistency could also happen for filesystems upgraded from Lustre 1.8. The `link` xattr stores the backpointer from each inode to the directory name entry/entries for each link to the file. During scanning, the LFSCK traversal will check for each name entry in each directory whether a corresponding name entry exists in the `link` xattr. When LFSCK finds that no entry is present in the `link` xattr for each directory entry, the `link` xattr is updated with a new `{ parent FID, filename }` entry for that directory entry. On files with multiple hard links there will be one entry in the `link` xattr for each link, subject to space availability in the `link` xattr.

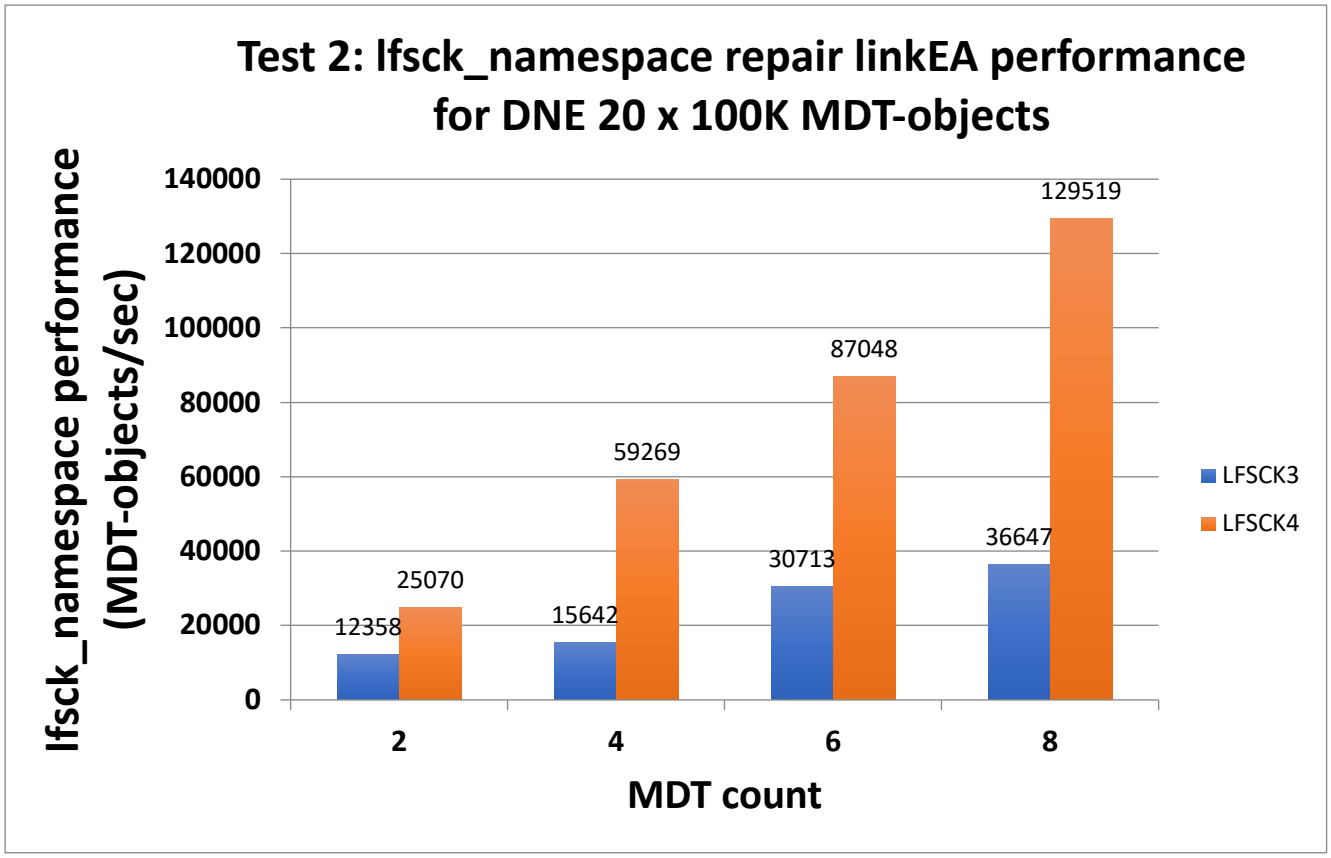
3. Performance comparison between LFSCK 3 and LFSCK 4 running with a simultaneous small file create workload on multiple MDTs without inconsistencies.

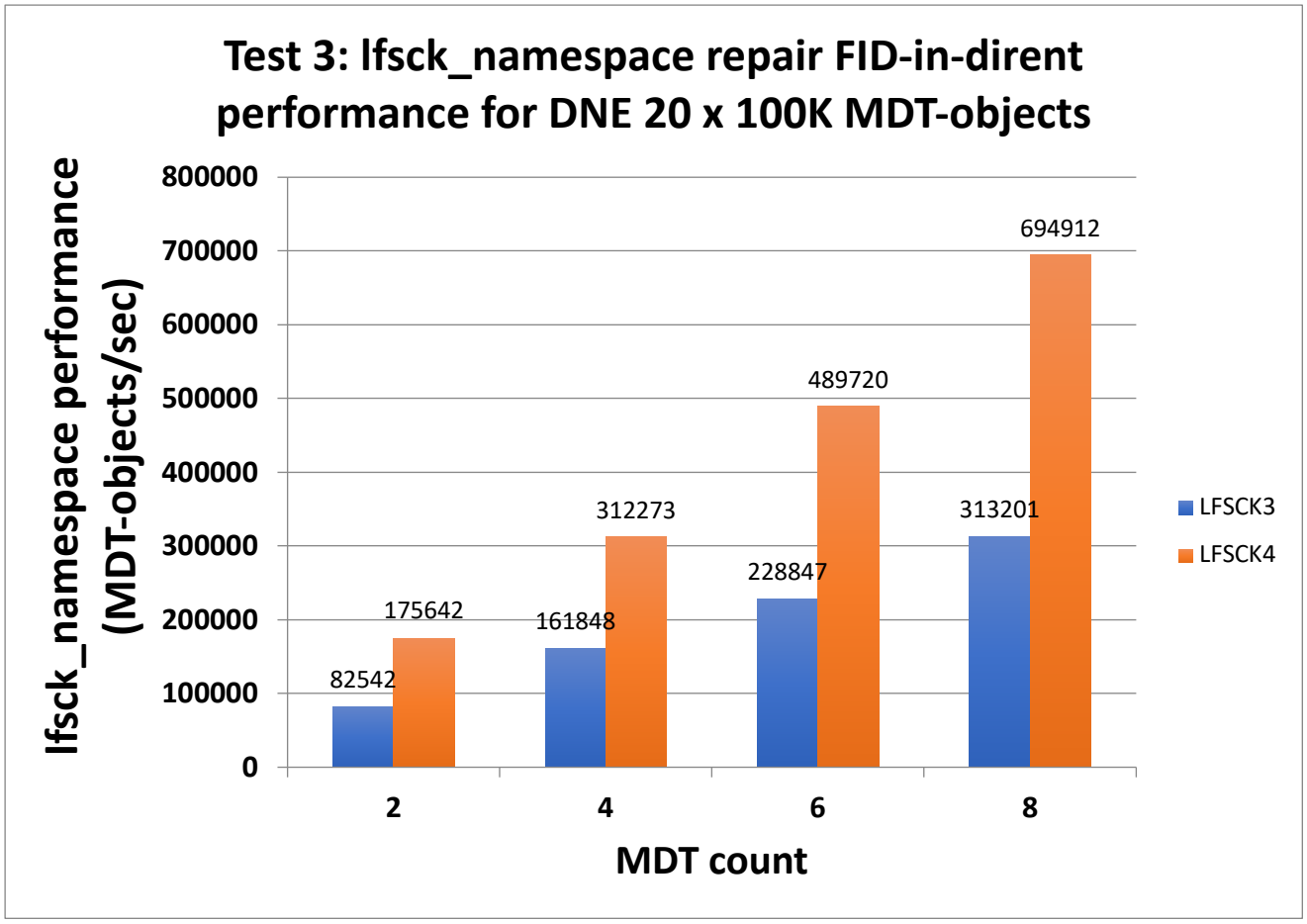
This test will measure the additional load online LFSCK 3 imposes on the MDS during a metadata-intensive workload. Online LFSCK includes a feature that allows the scanning rate to be limited. This feature is intended to enable an administrator to 'dial-back' the LFSCK scanning speed in a production environment to reduce or avoid impact on client metadata performance. This test provides a sweep of scanning rate measurements to give an administrator a feel for the performance change expected by choosing to reduce (or increase) the LFSCK scanning rate.

Results of the test

Each of the benchmark scenarios was run on the OpenSFS Test Cluster.




Test 2: lfsck_namespace repair linkEA performance for DNE 20 x 100K MDT-objects
lfsck_namespace performance (MDT-objects/sec)
140000
120000
100000
80000
60000
40000
20000
0
12358
25070
15642
59269
30713
87048
36647
129519
2
4
6
8
MDT count
LFSCK3
LFSCK4


Test 3: lfsck_namespace repair FID-in-dirent performance for DNE 20 x 100K MDT-objects
lfsck_namespace performance (MDT-objects/sec)
800000
700000
600000
500000
400000
300000
200000
100000
0
82542
175642
161848
312273
228847
489720
313201
694912
2
4
6
8
MDT count
LFSCK3
LFSCK4