Demonstration Milestone Completion for the LFSCK 4 (Performance) Sub-project 3.4 on the Lustre[1] File System FSCK Project of the SFS-DEV-001 contract.

Revision History

Date	Revision	Author
15/04/22	Original	Fan Yong, A. Dilger
15/04/23	Added patch link, more detail for fig 3	R. Henwood
15/04/24	Clarifying comments for tests	R. Henwood
15/05/07	Added results from non-zero-stripe	R. Henwood
15/05/07	Added results from different corruption type	R. Henwood
15/05/08	Explain results	R. Henwood
15/05/12	Explain file distribution choice	R. Henwood

1 Other names and brands maybe the property of others.

Table of Contents

Introduction

The following milestone completion document applies to Subproject 3.4 – LFSCK 4: Performance. This project is recorded in Amendment No. 1 on the OpenSFS Lustre Development contract SFS-DEV-001 signed July 30th, 2011.
The LFSCK 4: Performance code is functionally complete and recorded in the Implementation Milestone. The purpose of this Milestone is to verify the code performs acceptably in a production-like environment. In addition to completing all the Test Scenarios (demonstrated for the Implementation Milestone,) LFSCK 4: Performance has been measured as recorded below.
All tests were executed on the OpenSFS Functional Test Cluster. Details of the hardware are available in Appendix A. For all the tests with LFSCK 4, the patch series end with change 14332 was used. In order to deliver the best possible performance with more accurate measurement, additional work was completed beyond what was reported in the Implementation Milestone. The additional patches that were added into LFSCK 4 includes:

LU-6177	lfsck: calculate the phase2 time correctly
LU-6350	lfsck: lock object based on prediction for bad linkEA
LU-6351	lfsck: check object existence before using it
LU-6343	lfsck: locate object only when necessary
LU-6322	lfsck: show start/complete time directly
LU-6317	lfsck: NOT count the objects repeatedly
LU-6316	lfsck: skip dot name entry

This milestone is focused on illustrating the performance improvements that have been delivered by this project. Figures presented are chosen to emphasize comparative LFSCK3 vs LFSCK4 performance.

Correctness Test Coverage

1. `sanity-lfsck.sh`
   Test will be executed within Autotest and results automatically recorded in Maloo. Test will be automatically completed, triggered by a Gerrit check-in with commit message "`Test-Parameters: envdefinitions=ENABLE_QUOTA=yes mdtcount=2 testlist=sanity-lfsck`". All test cases must pass.
2. `sanity-scrub.sh`
   Test will be executed within Autotest and results automatically recorded in Maloo. Test will be automatically completed, triggered by a Gerrit check-in with commit message "`Test-Parameters: envdefinitions=ENABLE_QUOTA=yes testlist=sanity-scrub`". All test cases must pass.
3. Standard review tests
   The standard collection of review tests (currently including sanity, sanityn, replay-single, conf-sanity, recovery-small, replay-ost-single, insanity, sanity-quota, sanity-sec, lustre-rsync-test, lnet-selftest, sanity-lfsck, sanity-scrub, and mmp) will be executed within Autotest and results automatically recorded in Maloo. Tests will be automatically completed, triggered by a Gerrit check-in. All test cases should pass except for some known test failures unrelated to the LFSCK functionality.

The sanity-lfsck.sh and sanity-scrub.sh scripts are standard review tests that will be run and recorded. All previous LFSCK functionality (oi-scrub, FID-in-Dirent, MDT-OST consistency, MDT-MDT consistency) is also tested with these scripts.

MDT demonstration context

All tests require a populated directory on the file system. The directory will be created and populated with the following properties:

1. Create 'L' test root directories. 'L' is equal to the MDT count {2,4,6,8}. The directory in the root 'dir-X' is located MDT-X.
2. For each **MDT-X**, under its test root directory **dir-X**, create **M** sub-directories, where **M** = { 20, 40, 60, 80 }, for a maximum of 8 MDTs * 80 sub-directories per MDT = 640 directory trees.
3. Under each sub-directory, create 100K objects including:
    - 78% (0-striped) regular files under the sub-root
    - 3% local sub-dirs and each contains 5 (0-striped) regular files.
    - 0.4% 2-linked objects
    - 0.3% remote sub-dirs and each contains 4 (0-striped) regular files
    - 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.
4. This context was repeated with single striped files.

NOTES:

File distribution is chosen to attempt to represent a reasonable mix of files and directories under normal Lustre usage.

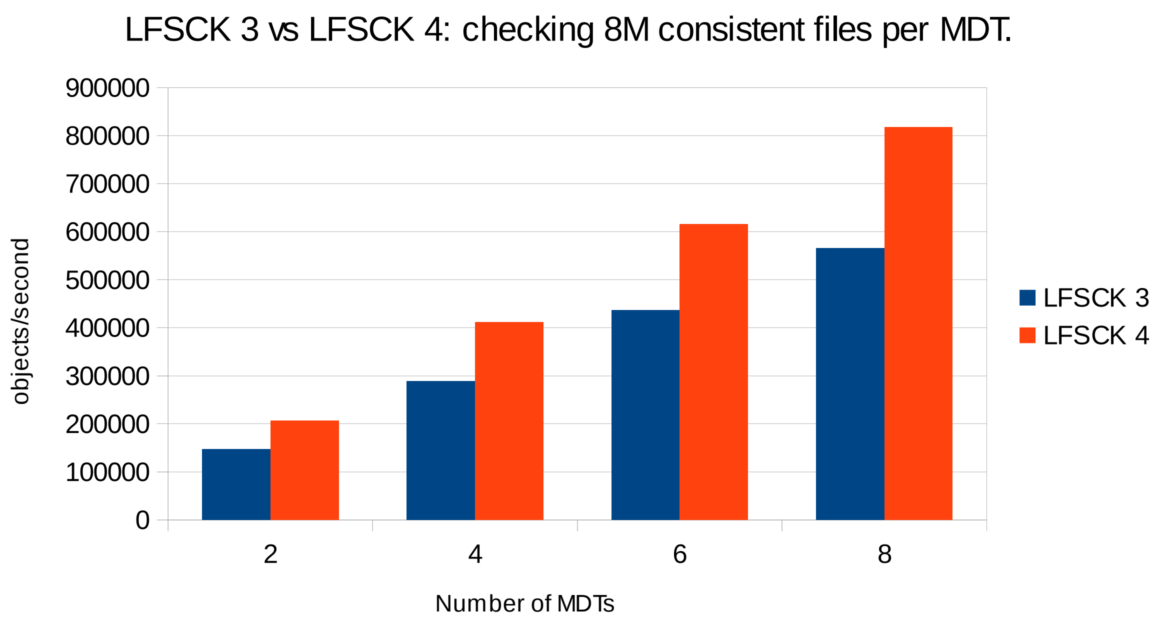
Zero-striped files are chosen for a number of reasons, including:

- Successive phases of LFSCK introduce and modify code from the previous phase. We have chosen to show performance improvements from the most recent functional improvement (LFSCK 3). LFSCK 3 is only concerned with MDT performance so file striping is redundant and can be ignored.
- Generating zero-striped files is significantly less time-consuming than striped files. This enables large numbers of files to be generated (and available for test) in a reasonable time.

Performance of LFSCK 3 vs LFSCK 4 against multiple, consistent MDTs.

This test provides a control benchmark for LFSCK scanning. LFSCK 3 and 4 include support for DNE striped and remote directories consistency checking (also known as MDT-MDT consistency checking). This test measures the scanning rate across multiple MDTs with striped directories with LFSCK 3 and LFSCK 4 compares the results. The aggregate LFSCK scanning performance is expected to scale linearly as additional MDTs with objects are added to the filesystem.

Result

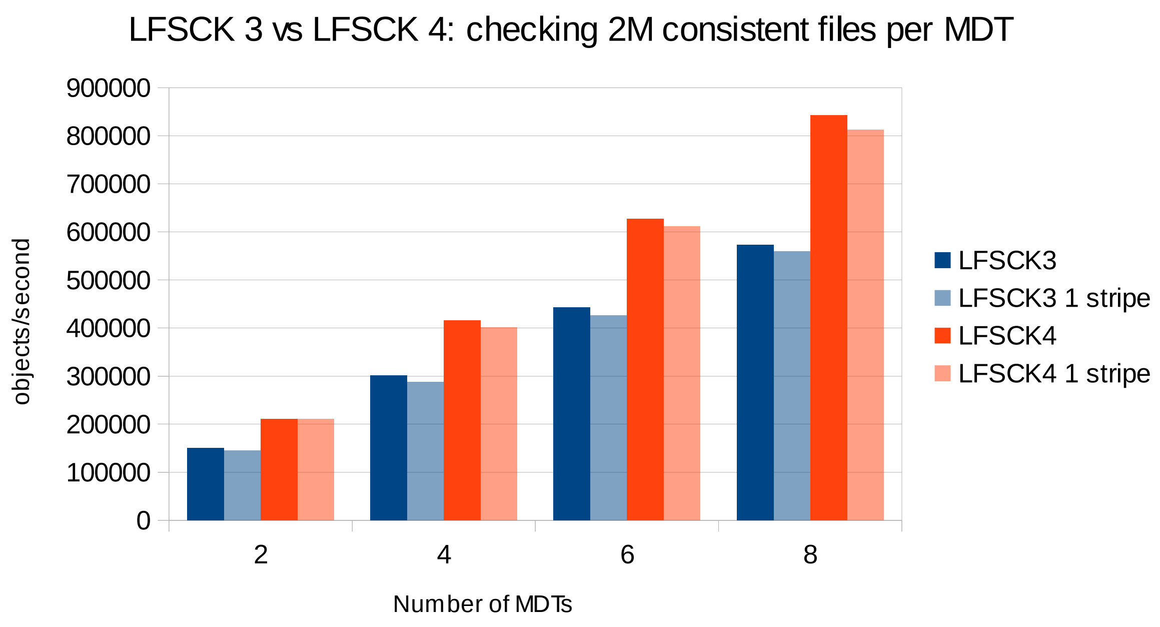


LFSCK 4 delivers at least 40% performance improvement over LFSCK 3 performance for the MDT counts we tested. In addition, expected scaling with the addition of MDTs is observed.

Performance of LFSCK 3 vs LFSCK 4 against multiple, consistent MDTs with single striped files.

An additional test was completed with single striped files. This provides a better picture of anticipated performance on real-world file systems. The results of this test are illustrated below:

Result



Both LFSCK 3 and LFSCK 4 show fractionally slower performance with a single striped file. This is expected because the addition of a stripe adds a small processing overhead for each object iteration. Overall performance is very encouraging with LFSCK 4 maintaining the large performance gains over LFSCK 3 for both zero and single striped files.

Performance of LFSCK 3 vs LFSCK 4 against multiple inconsistent MDTs.

This test scans the full filesystem on all MDTs to look for inconsistencies in the filesystem namespace, including MDT-MDT inconsistencies. The intent of this test is to measure performance when the filesystem needs to repair a large number of inconsistencies.

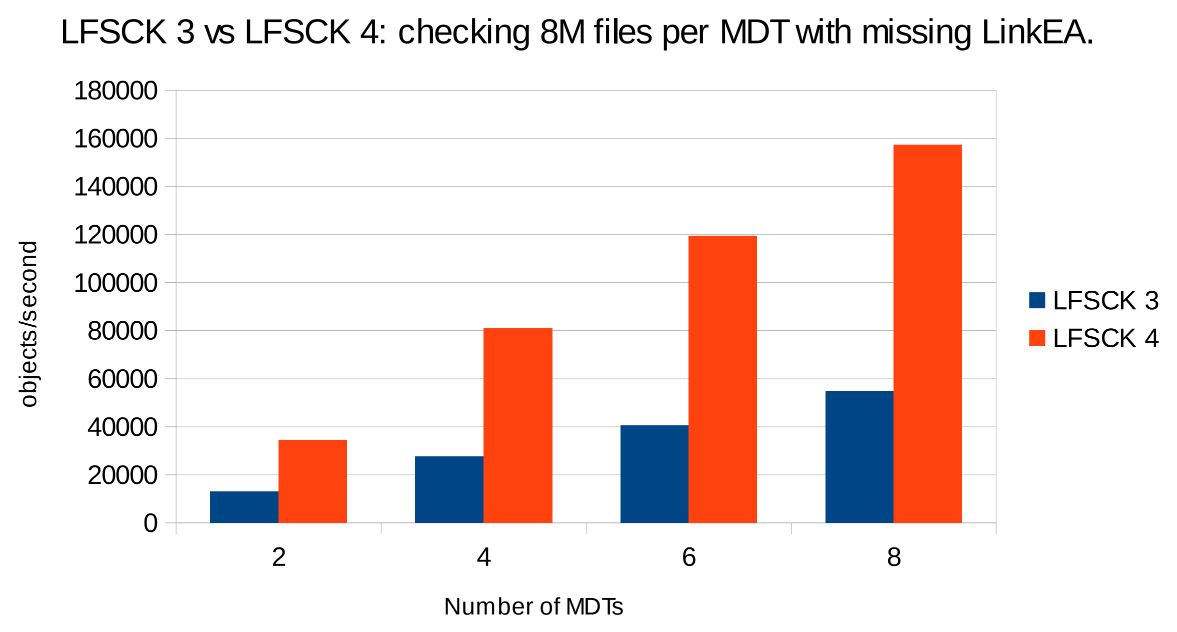
In for this test the filesystem has been intentionally corrupted during the filesystem population. The specific corruption is a missing `link` xattr on each file in the filesystem. Inconsistencies are created using the `OBD_FAIL_LFSCK_NO_LINKEA` fault injection hook. The `link` xattr stores the back-pointer from each inode to the directory name entry/entries for each link to the file. During scanning, the LFSCK traversal checks 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. Files with multiple hard links will contain one entry in the `link` xattr for each link, subject to space availability in the `link` xattr.

NOTES:
Only a single corruption type is used to exercise LFSCK during these tests. The reasons for choosing lost LinkEA as the single type include:

- Missing LinkEA corruption is simple to code into the test infrastructure by skipping LinkEA generation during file creation.
- Missing LinkEA corruption increases file creation performance so a larger test set can be created in a reasonable time.
- LFSCK 3 testing used missing LinkEA corruption and LFSCK 4 testing uses the same for comparison.
- For users upgrading from 1.8 file systems, the LinkEA will be missing and a large number of these repairs can be anticipated in such chases.

Result

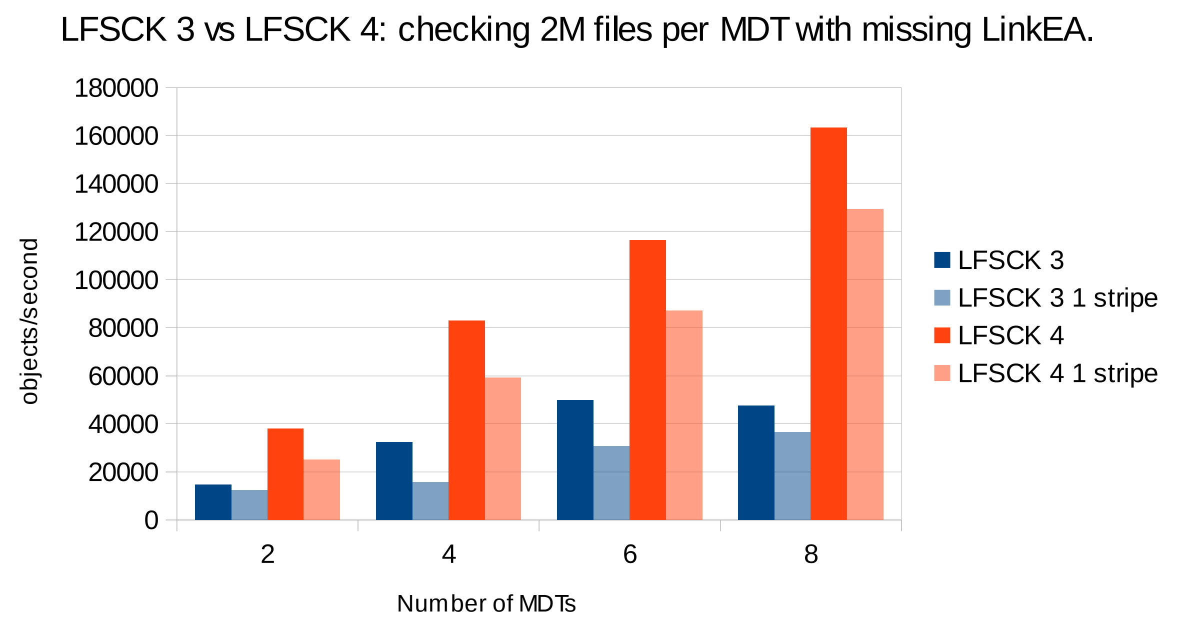
LFSCK 4 delivers over 160% performance improvement over LFSCK 3 performance for the 2 MDT counts we tested. This improvement grows to over 195% in the 4, 6, and 8 MDT counts . In addition, expected scaling with the addition of MDTs is observed.



Performance of LFSCK 3 vs LFSCK 4 against multiple inconsistent (missing LinkEA) MDTs with single striped files.

An additional test was completed with missing LinkEA and single striped files. This provides a better picture of anticipated performance on real-world file systems. The results of this test are illustrated below:

Result

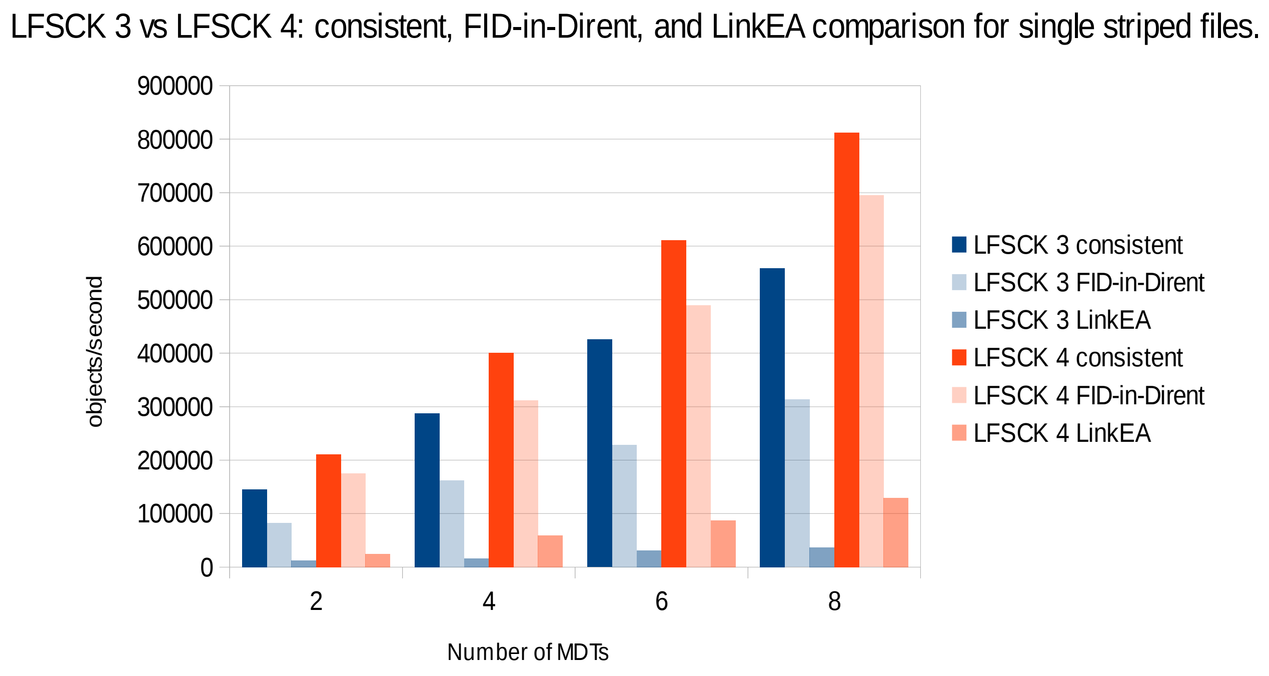


Both LFSCK 3 and LFSCK 4 show slower performance fixing a missing LinkEA entry on a single striped file. This is increased over consistent file results and expected because linkEA requires additional space on top of the stripe information. Larger storage requirements for a file entry increases the likelihood that performance will be lower because with the required information is now spread across multiple, possibly fragmented, blocks. Overall performance is very encouraging with LFSCK 4 maintaining the large performance gains over LFSCK 3.

Performance of LFSCK 3 vs LFSCK 4 against multiple inconsistent (incorrect FID-In-Dirent) MDTs with single striped files.

An additional test was completed with a incorrect FID-in-Dirent entry and single striped files. This provides good contrast with the missing LinkEA corruption and provides a better picture of anticipated performance on real-world file systems. The results of this test are illustrated below:

Result

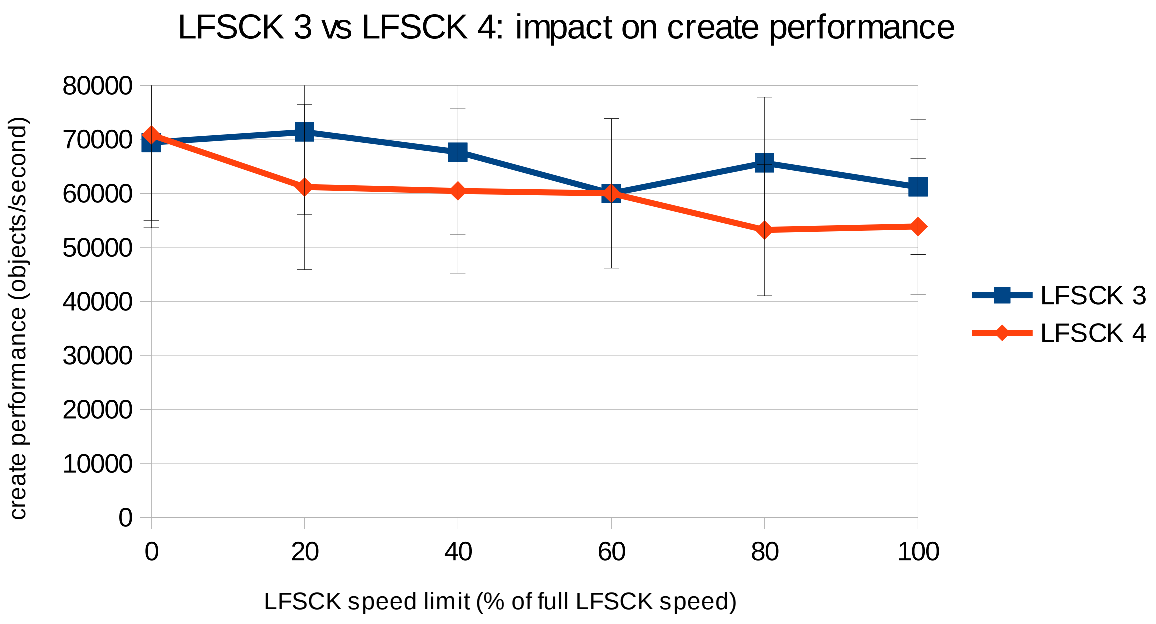


Both LFSCK 3 and LFSCK 4 show a significant performance difference between fixing FID-in-Dirent and missing LinkEA repair. This is expected because repairing a missing linkEA is more complex: foir example, repairing a missing LinkEA needs additional lookup and lock operations compared to repairing a corrupted FID-in-Dirent. Overall performance is very encouraging with LFSCK 4 maintaining the large performance gains over LFSCK 3 across all test.

Performance of LFSCK 4 on small file create performance on multiple MDT without inconsistencies

This test measures the additional load LFSCK 3 imposes on the MDS during a metadata-intensive application like 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. Two MDTs are used during this test.

Result



This figure show LFSCK 3 and LFSCK 4 performance with error included as the standard error of the mean. It is expected that the higher performance of LFSCK 4 will have a greater impact on create rates. However, this is not directly visible with our observations that include standard error. Across the spectrum of speeds there is only a modest impact, even with LFSCK running at full speed.

Conclusion

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.

Appendix A: OpenSFS Functional Test Cluster specification

client

- (2) Intel E5620 2.4GHz Westmere (Total 8 Cores)
- (1) 64GB DDRIII 1333MHz ECC/REG - (8x8GB Modules Installed) * (1) On Board Dual 10/100/1000T Ports
- (8) Hot Swap Drive Bays for SATA/SAS
- (6) PCi-e Slots 8X
- (3) QDR 40GB QSFP to QSFP iB Cables
- (3) Mellanox QDR 40GB QSFP Single Port

OSS server

- (1) Intel E5620 2.4GHz Westmere (Total 8 Cores)
- (1) 32GB DDRIII 1333MHz ECC/REG - (8x8GB Modules Installed) * (1) On Board Dual 10/100/1000T Ports
- (1) On Board VGA
- (1) On Board IPMI 2.0 Via 3rd. Lan
- (1) 500GB SATA Enterprises 24x7
- (1) 40GB SSD OCZ SATA
- (8) Hot Swap Drive Bays for SATA/SAS
- (6) PCi-e Slots 8X
- (3) QDR 40GB QSFP to QSFP iB Cables
- (3) Mellanox QDR 40GB QSFP Single Port

MDS server

- (1) Intel E5620 2.4GHz Westmere (Total 8 Cores)
- (1) 32GB DDRIII 1333MHz ECC/REG - (8x8GB Modules Installed) * (1) On Board Dual 10/100/1000T Ports
- (1) On Board VGA
- (1) On Board IPMI 2.0 Via 3rd. Lan
- (1) 500GB SATA Enterprises 24x7
- (1) 40GB SSD OCZ SATA
- (8) Hot Swap Drive Bays for SATA/SAS
- (6) PCi-e Slots 8X
- (3) QDR 40GB QSFP to QSFP iB Cables
- (3) Mellanox QDR 40GB QSFP Single Port