# Demonstration Milestone Completion for the LFSCK 1.5 Subproject 3.1.5 on the Lustre FSCK Project of the SFS-DEV-001 contract.

Revision History

| Date | Revision | Author |
|---|---|---|
| 04/13/13 | Original | R. Henwood |
| 04/25/13 | Clarifications of setup | R. Henwood |

## Table of Contents

## Introduction

The following milestone completion document applies to Subproject 3.1.5 – LFSCK 1.5: FID-in-Dirent and LinkEA of the Lustre FSCK within Amendment No. 1 on the OpenSFS Lustre Development contract SFS-DEV-001 agreed October 10, 2012.
The FID-in-Dirent and LinkEA code is functionally complete. 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,) FID-in-Dirent and LinkEA Performance will be measured as described below.
All tests were executed on the OpenSFS Functional Test Cluster. Details of the hardware are available in Appendix A. A Lustre build from March 30th 2013 was used.

## Correctness

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 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, 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.

### *Result*

This test has been completed successfully and the results are recorded in the Implementation milestone:
https://wiki.hpdd.intel.com/download/attachments/8553298/LFSCK15_Implementation_Milestone_Completion.pdf

## Unloaded Performance

All tests require a populated directory. The directory will be created and populated with the following characteristics:
1. A single test root directory.
2. N sub-directories under the test root directory.
3. Under each sub-directory, generate L sub-sub-directories, and use C threads to create 1M (1,024,000) files under these sub-sub-directories in parallel. Each thread creates (1M / C) files under its private directory.

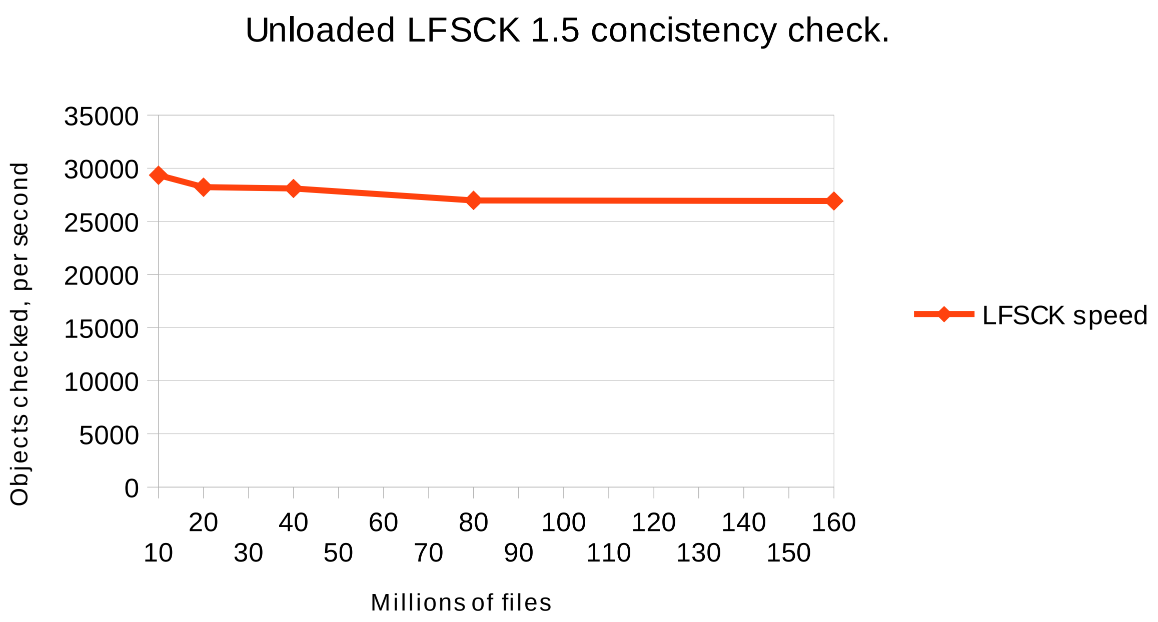
Once created, the populated directory structure with N sub-directories (described above) is known in the procedure below as PopD_n.

### *Measure performance of LFSCK 1.5 against vanilla Lustre 2.4 MDT device during routine consistency check.*

1. Install Lustre 2.4.
2. Run LFSCK on each PopD_n (with n={10, 20, 40, 80, 160}). LFSCK must run at full speed, without any Lustre derived system load.

```
lctl lfsck_start -M ${fsname}-MDT0000 -t namespace
```
3. Record the following:
   1. Start time, end time
   2. `lfsck_namespace` statistics before and after the run (`lctl get_param -n mdd.${fsname}-MDT0000.lfsck_namespace`)

### *Result*

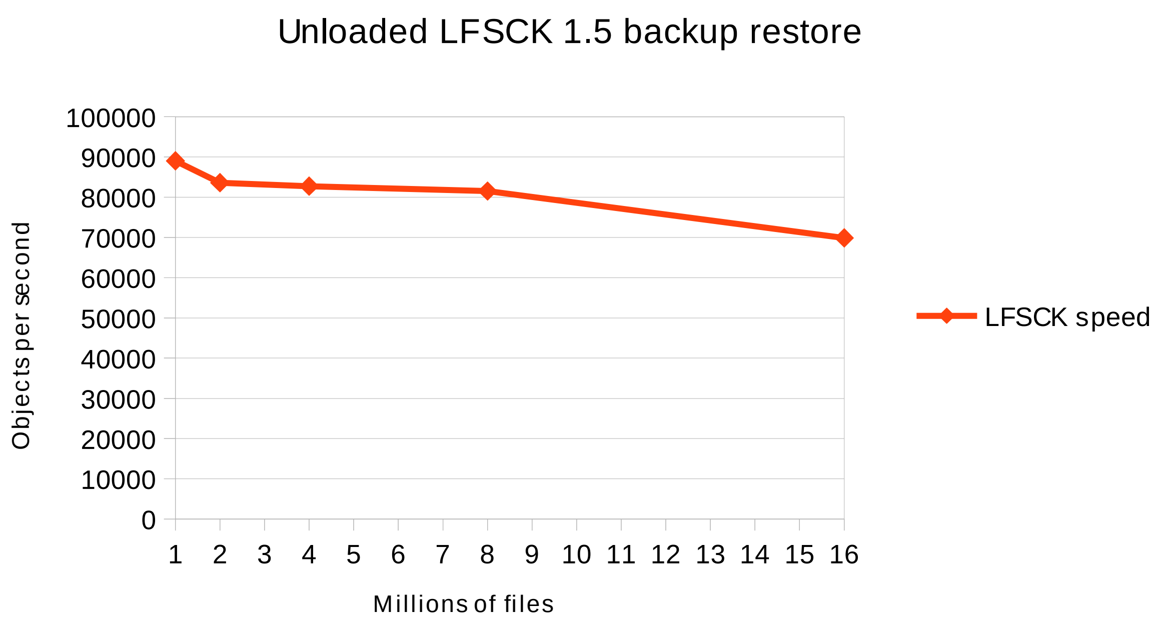


For unrestricted performance, LFSCK shows a small reduction in object through-put consistency checking as the number of files is increased from 10 to 160 million. The reduction is approximately 10% across a file count that increases more than one order of magnitude. This performance reduction is judged to be acceptable.

### *Measure performance of LFSCK 1.5 against vanilla Lustre 2.4 MDT device that is restored from a file-level backup.*

1. Install Lustre 2.4.
2. Create PopD_n (with n={1, 2, 4, 8, 16})
3. Perform MDT file-level backup.
4. Perform MDT file-level restore.
5. Run LFSCK on each PopD_n. LFSCK must run at full speed, without additional system load.
6. Record the following
   1. Start time, end time
   2. lfsck_namespace statistics before and after the run (`lctl get_param -n mdd.${fsname}-MDT0000.lfsck_namespace`)

### *Result*

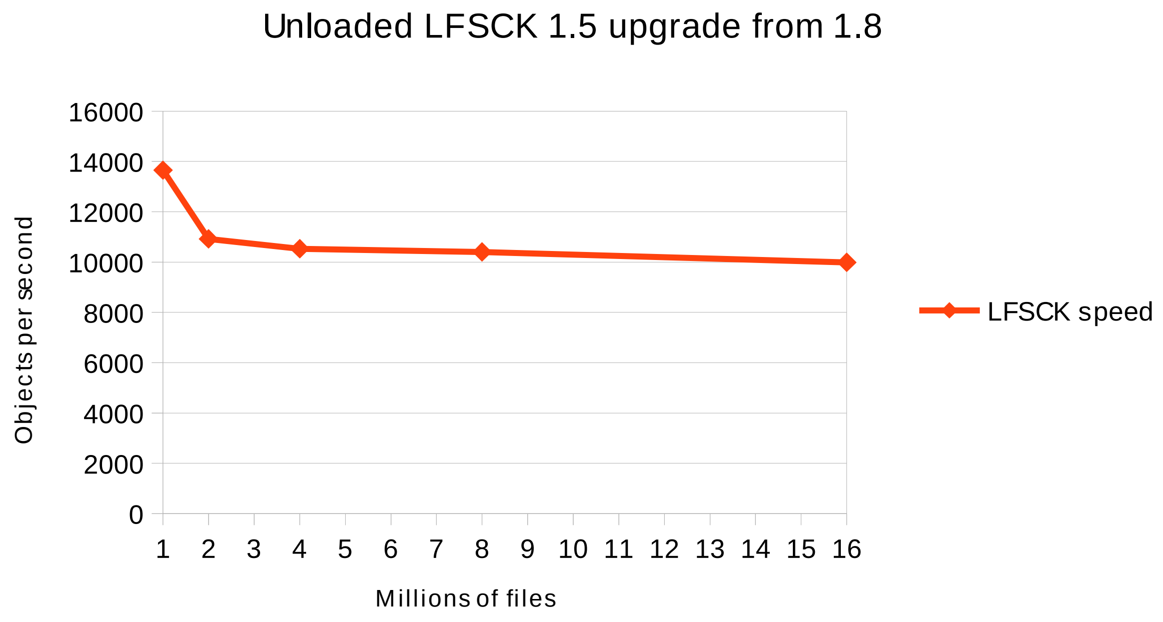


For unrestricted performance, LFSCK shows a small reduction in object through-put from a file-level backup as the number of files is increased from 1 to 16 million. The reduction is approximately 20% across a file count that increases more than one order of magnitude. This performance reduction is judged to be acceptable.

## *Measure performance of LFSCK 1.5 against Lustre 2.4 MDT device that has been upgraded from 1.8.*

1. Install Lustre 1.8
2. Create PopD_n (with n={1, 2, 4, 8, 16})
3. Update the system to 2.4. Use "`tunefs --dirdata`" to enable FID-in-dirent on MDT.
4. Install Lustre-2.4 and mount the old 1.8 MDT and OST devices with Lustre-2.4 modules.
5. Run LFSCK on each PopD_n. LFSCK must run at full speed, without additional system load.
6. Record the following:
    1. Start time, end time
    2. `lfsck_namespace` statistics before and after the run (`lctl get_param -n mdd.${fsname}-MDT0000.lfsck_namespace`)

### ***Result***



For unrestricted performance, LFSCK shows a large initial reduction in object through-put for an upgrade from 1.8 file system as the number of files is increased from 1 to 16 million. The reduction is approximately 25% across a file count that increases more than one order of magnitude. This performance reduction is judged to be acceptable for a task that many administrators may only every complete once.

## LFSCK Performance Under Load

Evaluate the affect running LFSCK online will have on the system. All tests require PopD_400 (described in section 2 Unloaded Performance).

### ***Measure performance impact of LFSCK at full speed.***

1. Install Lustre 2.4.
2. Create PopD_10.
3. Start LFSCK at full speed.
   `lctl lfsck_start -M ${fsname}-MDT0000 -t namespace`
4. Record the following:
   1. Start time, end time
   2. `lfsck_namespace` statistics before and after the run (`lctl get_param -n mdd.${fsname}-MDT0000.lfsck_namespace`)

### ***Result***

| | |
|---|---|
| Create objects per second without LFSCK | 18591 |
| LFSCK namespace full speed objects per second | 65015 |

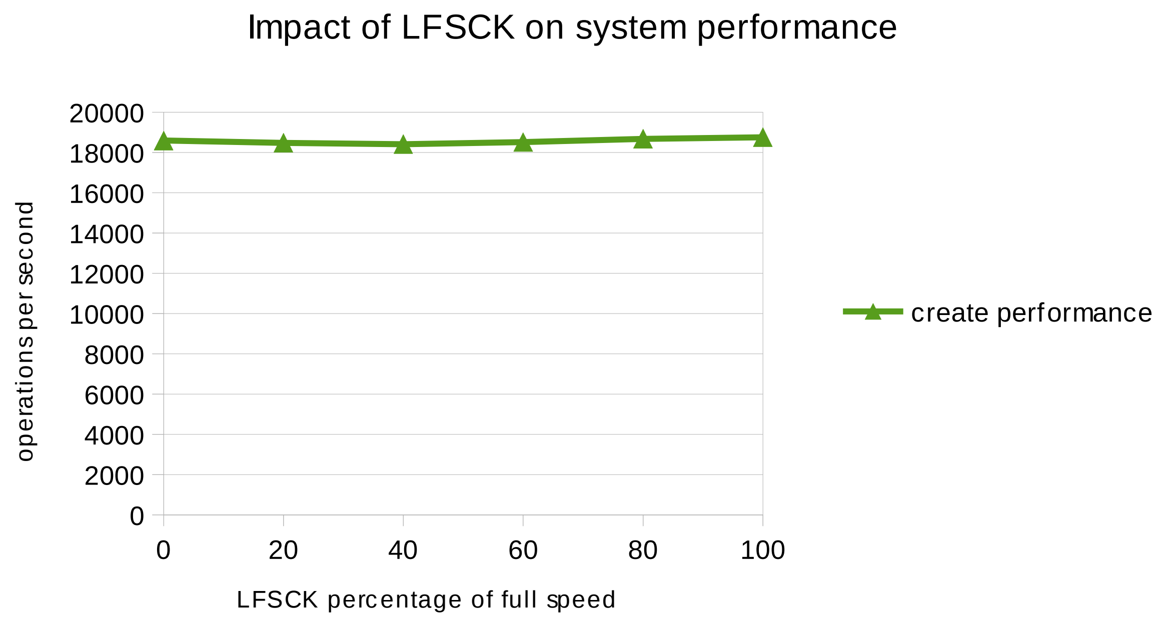
The create performance is provided as an average result of nine runs. The absolute measure of creates is low on the given hardware. Given that no changes were made to the hardware or configuration between runs, the comparative performance of Create and LFSCK object scanning are legitimate.

### ***Measure performance impact of LFSCK with variable speed limit.***

1. Begin the test as above.

2. Start LFSCK with speed limit = (1/5)S, (2/5)S, (3/5)S, (4/5)S.
   `lctl lfsck_start -M ${fsname}-MDT0000 -t namespace -s xxx`
3. Simultaneously C threads to create 2M files (`mds-survey`) in parallel. Each thread creates under its private directory, and create (2M / C) files.
4. Record the following:
   1. Start time, end time
   2. `lfsck_namespace` statistics before and after the run
      (`lctl get_param -n mdd.${fsname}-MDT0000.lfsck_namespace`)
   3. `mds-survey` performance creating 2M files.

### *Result*



Adjusting the speed control to fractions of full speed shows apparently has a negligible effect on the ability of the system to complete normal operations. This result may indicate that the CPU is sufficiently powerful to service both LFSCK namespace operations and normal workload operations simultaneously.

## Scale testing

### *Run LFSCK during SWL test on Hyperion.*

LFSCK must run successfully during a routine SWL stress test performed on Hyperion.

### *Result*

This test was completed with any issues on two occasions.

## Conclusion

LFSCK 1.5: FID-in-Dirent and LinkEA has successfully completed both functional Acceptance and Performance tests. The performance results recorded herein illustrate performance expectations are met or exceeded during online operation and under load.

## Appendix A: OpenSFS Functional Test Cluster specification

MDS server

- (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
- (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

OSS server

- (2) Intel E5620 2.4GHz Westmere (Total 8 Cores)
- (2) Copper Base CP0217 CPU Cooler 1U with Heat Pipe
- (1) 64GB 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