Self Extending Layout Test Plan

Title	Self Extending Layout Test Plan
Summary	Self-Extending Layout feature Test Plan
Owner	Vitaly Fertman
Task List	☐ Inspection Alexey Lyashkov ☐ Inspection Elena Gryaznova
Version	0.1
Last Update	30 May 2019
History of changes	

Table of contents

- Introduction
- Feature Installation and Set-UP
- Regression Tests
- New feature functional tests
- Performance
  - Large IO Performance
  - Small IO performance
  - MD performance
- New Feature Stress Tests
  - Racer
- Interoperability
- Feature interaction
  - HSM
  - LFSCK
  - Snapshot
  - Distributed Name Space (DNE)
  - DOM
- Documentation

Introduction

The document is intended to describe to the test plan for the Self Extended Layout (SEL) Lustre feature. This is an extension of the Partial File Layout (PFL) feature which allows to create a file layout with components followed by a new type uninitialised component - "extension space" components. When file access reaches this extension space, the system can check if there is still a sufficient amount of space available on the current OSTs and extend the length of the previous component, allowing the IO to continue on the same OSTs. If there is not enough space on the current OSTs, it will not extend the current component, and will instead modify the layout to switch to a new component on new OSTs. More details could be found here: HLD: Spillover Space - Self-Extending Layouts. The purpose of the feature is to mitigate the ENOSPC problem while using a small SSD OST pool to let IO automatically to spill over to a large HDD pool.

Feature Installation and Set-UP

The SEL feature is new to Lustre 2.13 and does not require any commands to enable the feature. The definition of the SEL layout for component files can be set for the whole file system, for a directory or for a specific file with the use of *lfs-setstripe* or using the new API *llapi_layout_extension_size_set* method.

Regression Tests

1. The acceptance-small test suites are to be run with a default layout specified for the file system to verify there is no functional regressions of existing testing.
2. A defined acceptance-small tests are also to be run with a set of SEL layouts specified for the file system to verify there is no

functional regressions while working with SEL-enabled files. The set of SEL layouts to be verified is:

- simple SEL layout: lfs setstripe -E -1 -z 64M ROOT_DIR
- SEL as not the last component: lfs setstripe -E 128M -z64M -E -1 -z 128M ROOT_DIR

Parameters are to be chosen so that the IO reaches the SEL component, i.e. IO must be larger than the first component size.

TBD: define the list of acceptance-small tests to be run.

New feature functional tests

There is a set of code paths which is checked by new tests

code path	action	test# covering it
A1. IO to a SEL component. The previous component is **not INIT'ed** (no flag "**init**" on the component). The OST assignment for the previous component fails due to OOS		
a. A next component exists	Remove both the SEL and the previous components, S pillover.	sanity-pfl 20b, 21b sanity-flr 204c
b. No next component	Error	sanity-pfl 20d
c. No next component, the previous component is a repeated one	Remove the previous (repeated) component, force the extension of the original previous component	sanity-pfl 22c, 22d
A2. The OST assignment for the previous **not INIT'ed** component succeeded but its extension fails due to low on space.		
a. A next component exists	Remove both the SEL and the previous components, S pillover.	sanity-pfl 20e 23e
b. No next component	Force the extension of the previous component	sanity-pfl 20d 23c
c. No next component, the previous component is a repeated one	Remove the previous (repeated) component, force the extension of the original previous component	sanity-pfl 22d sanity-flr 204f
A3. The extension of the previous **INIT'ed** component fails due to low on space		
a. A next component exists	Remove the SEL component, Spillover.	20a
b. No next component	Repeat the previous component.	sanity-pfl 22a 22b 22c 22d 23f sanity-flr 204e 204f
c. No next component, the previous component is a repeated one	<impossible>	
A4. The previous **INIT'ed** component gets OST assigned and extended, **SEL is left**	The previous component is extended, the SEL is shortened.	sanity-pfl 19a, 19b, 19c, 20a, 22a, 22b, 22c
A5. The previous **not INIT'ed** component gets OST assigned and extended, **SEL is left**	The previous component gets OST assigned and extended	sanity-pfl 19e, 20a, 20b, 21a, 21b, 22a, 22b, 22c sanity-flr 204a 204e
A6. The previous **INIT'ed** component gets OST assigned and extended, **SEL is not left**		
a. A next component exists	The previous component is extended, SEL is removed	sanity-pfl 19c, 19d
b. No next component	The previous component is extended to EOF, SEL is removed (a-la append)	sanity-pfl 23a
c. No next component, the previous component is a repeated one	<impossible>	
A7. The previous **not INIT'ed** component gets OST assigned and extended, **SEL is not left**		
a. A next component exists	The previous component gets OST assigned and extended, SEL is removed	sanity-pfl 23d

b. No next component	The previous component gets OST assigned and extended to EOF, SEL is removed (a-la append)	sanity-pfl 23b 23c
c. No next component, the previous component is a repeated one		sanity-pfl 23f

Also, the tests sanity-pfl 21* have DOM in the first component.

Performance

Large IO Performance

The large IO tests are to be done with the following settings

1. Default (non-SEL layout) to confirm there is no regression in the base performance.
2. SEL-layout with a large extension size, consider 100G / 1 stripe as an extreme case, i.e. if a component has 10 stripes, -z 1T is to be set. The test is supposed to show there is no performance penalty with large SEL space comparing with the similar layout PFL files.
3. SEL-layout with a small extension size, consider 64M / 1 stripe as the extreme case. The test is supposed to demonstrate the maximum performance penalty with small SEL space comparing with the similar layout PFL files.
4. SEL-layout with a real-life extension size, consider 16G or 32G / 1 stripe. The test is supposed to demonstrate an expected user performance penalty with SEL regions comparing with the similar layout PFL files.

The proposed SEL layout is lfs setstripe -E 1M -E -1 -z <ext_size> file.

The proposed PFL layout for comparison is lfs setstripe -E 1M -E -1 file.

The following set of large IO tests is to be done with the above settings: dd, IOR SSF and FPP sequential IO, test with 1 thread, with many threads, many clients.

Small IO performance

The small IO tests are to be done with the following settings

1. Write by 1k at offsets N*<ext_size> - it will result in 1 small write per each extension size, i.e. each write will initiate the SET extension procedure. This test checks the SEL extension procedure does not consume a significant amount of time and the performance is close to a base PFL layout.
2. The same as (1), but the OSTs are low on space. This test checks the attempt to repeat the component which fails and the following forced extension does not take extra time and the performance results are still the same as in (1).
3. The same as (1), but with different stripe count to show it does not change the performance much.

The proposed SEL layout is lfs setstripe -E -1 -z 100M <file>.

The proposed PFL layout is lfs setstripe -E -100M -E 200M -E 300M ... -E ${N}00M -E -1 <file>.

The OST's reserved_mb_low should be probably decreased so that the set of to be written data would not run to real ENOSPC on OSTs.

The following set of large IO tests is to be done with the above settings: [TBD]

MD performance

File creation/removal with SEL layout to confirm there is no regression against the base performance: mdsrate, mdtest.

New Feature Stress Tests

1. There are 2 pools of OSTs; IOR writes the set of data which exceeds the pool1 size. 2 runs:
    - The default SEL layout is: component1 is on pool1, component2 is SEL, component3 is on pool2. The IOR succeeds without ENOSPC.
    - A similar PFL layout is: component1 is on pool1, component2 is on pool2 ; The IOR fails with ENOSPC.
2. There are 2 pools of OSTs; IOR writes the set of data which fits well to pool1 size. however, there is a parallel job which is writing to pool1 until ENOSPC. 2 runs:
    - The default SEL layout is: component1 is on pool1, component2 is SEL, component3 is on pool2. The IOR succeeds without ENOSPC.
    - A similar PFL layout is: component1 is on pool1, component2 is on pool2 ; The IOR fails with ENOSPC.

Due to a high latency of informing MDS about the free space on OSS nodes, the parameters of these tests have to be accurately set up.

a. The OST's reserved_mb_low >= 2 * write_speed * update_interval

b. The OST's reserved_mb_low >= amount of IO threads * SEL extension_size

Where:

- update_internal is 5sec by default and cannot be controlled through procfs for now;

Racer

The script racer/file_create.sh has option to enable SEL layout: RACER_ENABLE_FLR [TBD]

Interoperability

Lustre clients prior to Lustre 2.13 will not be able to access existing SEL files. An attempt to access such a file will result with an error.

Due to lack or support in utilities, the old clients will not be able to create SEL files nor parse the existing SEL layouts.

Lustre 2.13 clients that try and create a SEL file on an older server will receive an error.

Feature interaction

HSM

SEL files must be archived, released and restored. Restored SEL files will have the same number of components and extent ranges as the original file.

LFSCK

The LFSCK tool shall correctly process files with a SEL layout, detect and correct filesystem corruption or implementation defects.

Q: some words about post-run and online parallel run.

Snapshot

A snapshot shall be taken of a file system with SEL files in it and where the file system has a default SEL file layout specified. The snapshot shall preserve the composite file layout if the snapshot is mounted and files are retrieved.

Distributed Name Space (DNE)

Stress tests with SEL default layout are to be created under remote directories and striped directories.

DOM

SEL files are to be created with DOM component at the beginning of the file.

Repeat the tests from the Regression Test sections with the following SEL layout:

- SEL layout with DOM: lfs setstripe -E 1M -L mdt -E 128M -z64M ROOT_DIR

More specific interaction of SEL and DOM is already covered by the "New Feature Regression Tests" section.

Documentation

Man pages for *lfs-setstripe*, *lfs-getstripe, lfs-find* were modified to include the additional functionality for the SEL feature. Also, man pages were added for the new API methods:

llapi_layout_extension_size_get, *llapi_layout_extension_size_set.*