DE LA RECHERCHE À L'INDUSTRIE

cea

www.cea.fr

Lecture on Parallel Filesystems

Lustre (2/4)

Jacques-Charles Lafoucriere

ENSIIE| 2018

How to set up a Lustre Cluster

6 Steps to configure a Lustre FS

1)Setup hardware

- Lustre servers/clients need Linux with a supported kernel

2)Configure storage devices

- Lustre servers require block devices

3)Setup network interfaces

- All Lustre machines must be able to exchange network packets

4)Install Lustre

- Lustre is available from sources or from pre-build RPM

5)Configure LNet

6)Configure Lustre

Lustre RPMS

Lustre RPMs

- lustre-modules-2.7.22.2-3.10.0_693.11.6.el7.x86_64.rpm
- lustre-2.7.22.2-3.10.0_693.11.6.el7.x86_64.x86_64.rpm
- lustre-tests-2.7.22.2-3.10.0_693.11.6.el7.x86_64.x86_64.rpm
- lustre-osd-ldiskfs-2.7.22.2-3.10.0_693.11.6.el7.x86_64.x86_64.rpm
- lustre-iokit-2.7.22.2-3.10.0_693.11.6.el7.x86_64.x86_64.rpm
- lustre-osd-ldiskfs-mount-2.7.22.2-3.10.0_693.11.6.el7.x86_64.rpm

E2fsprogs RPMs

- e2fsprogs-1.42.13.wc6-8.el7.centos.x86_64.rpm
- e2fsprogs-libs-1.42.13.wc6-8.el7.centos.x86_64.rpm

lustre-modules-2.7.22.2-3.10.0_693.11.6.el7.x86_64

```
/lib/modules/3.10.0-693.11.6.el7.x86_64/extra/kernel
/lib/modules/3.10.0-693.11.6.el7.x86_64/extra/kernel/fs
/lib/modules/3.10.0-693.11.6.el7.x86_64/extra/kernel/fs/lustre
fid.ko
fld.ko
lfsck.ko
llite_lloop.ko
lmv.ko
lod.ko
lov.ko
lquota.ko
lustre.ko
mdc.Ko mdd.ko mdt.ko
mgc.ko mgs.ko
obdclass.ko obdecho.ko
ofd.ko
osc.ko osp.ko ost.ko
ptlrpc.ko
ko2iblnd.ko ksocklnd.ko
libcfs.ko
lnet.ko
lnet_selftest.ko
```


lustre-2.7.22.2-3.10.0_693.11.6.el7.x86_64

```
/etc/ha.d/resource.d/Lustre
/etc/ha.d/resource.d/Lustre.ha_v2
/etc/ldev.conf
/etc/modprobe.d/ko2iblnd.conf
/etc/udev/rules.d/99-lustre.rules
/sbin/mount.lustre
/usr/bin/lfs
/usr/bin/lfs_migrate
/usr/bin/llobdstat
/usr/bin/llstat
/usr/bin/lustre_req_history
/usr/bin/plot-llstat
/usr/include/libcfs
/usr/include/libcfs/types.h
/usr/include/lustre
/usr/include/lustre/liblustreapi.h
/usr/include/lustre/ll_fiemap.h
/usr/include/lustre/lustre_user.h
/usr/include/lustre/lustreapi.h
```

```
/usr/lib64/libcfsutil.a
/usr/lib64/libiam.a
/usr/lib64/liblnetconfig.a
/usr/lib64/liblnetconfig.so
/usr/lib64/liblustreapi.a
/usr/lib64/liblustreapi.so
/usr/lib64/libptlctl.a
/usr/libexec/lustre/haconfig
/usr/libexec/lustre/lc_common
/usr/sbin/ko2iblnd-probe
/usr/sbin/l_getidentity
/usr/sbin/lc_cluman
/usr/sbin/lc_hb
/usr/sbin/lc_lvm
/usr/sbin/lc_md
/usr/sbin/lc_modprobe
/usr/sbin/lc_mon
/usr/sbin/lc_net
/usr/sbin/lc_servip
/usr/sbin/lctl
/usr/sbin/ldev
/usr/sbin/lhbadm
/usr/sbin/lhsmtool_posix
```

```
/usr/sbin/ll_decode_filter_fid
/usr/sbin/ll_recover_lost_found_objs
/usr/sbin/llog_reader
/usr/sbin/llverdev
/usr/sbin/llverfs
/usr/sbin/lnetctl
/usr/sbin/lr_reader
/usr/sbin/lst
/usr/sbin/ltrack_stats
/usr/sbin/lustre_rmmod
/usr/sbin/lustre_routes_config
/usr/sbin/lustre_routes_conversion
/usr/sbin/lustre_rsync
/usr/sbin/lustre_start
/usr/sbin/mkfs.lustre
/usr/sbin/routerstat
/usr/sbin/tunefs.lustre
/usr/share/lustre
/usr/share/lustre/mpich-1.2.6-
lustre.patch
/usr/share/man/……
```

ldiskfs

```
lustre-osd-ldiskfs-2.7.22.2-3.10.0_693.11.6.el7.x86_64
/lib/modules/3.10.0-693.11.6.el7.x86_64/extra/kernel/fs/lustre/ldiskfs.ko
/lib/modules/3.10.0-693.11.6.el7.x86_64/extra/kernel/fs/lustre/osd_ldiskfs.ko

lustre-osd-ldiskfs-mount-2.7.22.2-3.10.0_693.11.6.el7.x86_64
/usr/lib64/lustre/mount_osd_ldiskfs.so
```


Create Simple Lustre FS

Create a combine MGS/MDT

```
mkfs.lustre --fsname=TEST --mgs --mdt --index=0 /dev/blockdevice3
```

Start the combine MGS/MDT

```
mount -t lustre /dev/blockdevice3 /mnt/mdt
```

Create 2 OST

```
mkfs.lustre --fsname=TEST --mgsnode=MGS_NID --ost -index=0 /dev/blockdevice0
mkfs.lustre --fsname=TEST --mgsnode=MGS_NID --ost -index=1 /dev/blockdevice1
```

Start the 2 OST

```
mount -t lustre /dev/blockdevice0 /mnt/ost0
mount -t lustre /dev/blockdevice1 /mnt/ost1
```

Start Lustre Clients

On each client

```
mount -t lustre MGS_NID:/TEST /lustre
```

Lustre namespace is accessible under /lustre

File Layout

Lustre File Allocation

MDS allocates objects in OSTs

- Round-robin algorithm if OSTs are well balanced
- Weighted algorithm

MDS follows striping rules

- Explicit rule from file creation request
- Directory rule
- FS default

Why do we (not) need to stripe files

Pro

- To provide high bandwidth access
  - Over OST's
  - Over OSS's
  - Use case: single large shared file accessed from many compute nodes
- To provide more bandwidth than a single OST can do
- To use very large files

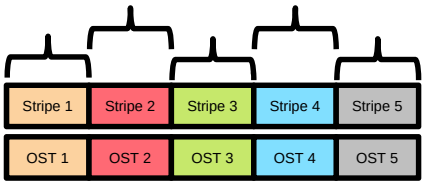
Cons

- Increase overhead
  - More locks, more network traffic for MD request (stat(), unlink())
- Increase contention
  - OST's will have more user sharing
- Increase availability risk

Choosing a Stripe Size

Stripe Size

- Size of data allocated on each OST



A balancing act

- A multiple of page size (i.e. N x 64 KB)
- Min of 512 KB (to fill network packet size which is 1 MB)
- Up to 4 GB (marge stripe size will be counterproductive if it does not match I/O pattern)
- Try to optimize for write boundaries to avoid cross object writes

MDT Allocation Algorithms

Round-Robin

- Used when OST's have the same amount of free space
  - Tunable
- Fastest algorithm
- The choice is made for the first OST
- Others are taken sequentially

	Stripe #	OST
FILE1	4	OST1 OST2 OST3 OST4
FILE2	3	OST5 OST6 OST7
FILE2	6	OST0 OST1 OST2 OST3 OST4 OST5
FILE4	3	OST6 OST7 OST0

MDT Allocation Algorithms (cont.)

Weighting

- Used when OST's are not identically filled
  - Tuneable
  - Try to allocate on the less filled OSTs
  - Try to balance over the many OST's
- The choice is made for all OSTs
- Longer time to get the allocation result

Layout Management

Command to manage file/directories rules

```
lfs setstripe
lfs getstripe
```

To create a file with 2 stripes of 2 MB

```
lfs setstripe -c 2 -S 2M spoo10
```

Layout Display

```
# lfs getstripe -v spoo10
spoo10
lmm_magic:          0x0BD10BD0
lmm_seq:            0x200000401
lmm_object_id:      0x7
lmm_fid:            [0x200000401:0x7:0x0]
lmm_stripe_count:   2
lmm_stripe_size:    2097152
lmm_pattern:        1
lmm_layout_gen:     0
lmm_stripe_offset: 0
        obdidx           objid           objid           group
             0               7             0x7               0
             1               7             0x7               0

#
```


Layout Policy: starting OST

```
# lfs setstripe -c 1 --ost 1 spoo99
# lfs setstripe -c 1 --ost 0 spoo100
# lfs getstripe spoo99 spoo100
spoo99
lmm_stripe_count:  1
lmm_stripe_size:   1048576
lmm_pattern:       1
lmm_layout_gen:    0
lmm_stripe_offset: 1
        obdidx           objid           objid           group
             1               8             0x8               0

spoo100
lmm_stripe_count:  1
lmm_stripe_size:   1048576
lmm_pattern:       1
lmm_layout_gen:    0
lmm_stripe_offset: 0
        obdidx           objid           objid           group
             0               8             0x8               0
```

Setting Layout for a Directory

A creation rule can be set on a directory

- Use of lfs setstripe
- Directory must be created before
- All files created in the directory without explicit rule will use the directory rule
- Can be seen with lfs getstripe
- Future subdirectory will inherit from the default

Directory Layout Examples

```
# mkdir dir0
lfs getstripe dir0
dir0
stripe_count:  1 stripe_size:   1048576 stripe_offset: -1
# mkdir dir1
# lfs setstripe -c 2 dir1
# cd dir1
# lfs getstripe .
.
stripe_count:  2 stripe_size:   1048576 stripe_offset: -1
# mkdir dir2
# lfs getstripe dir2
dir2
stripe_count:  2 stripe_size:   1048576 stripe_offset: -1
#
```

Changing File Layout

```
# lfs setstripe -c 2 spoo123 ; lfs getstripe spoo123 ; lfs migrate -c 1 spoo123 ; \
          lfs getstripe spoo123
spoo123
lmm_stripe_count:  2
lmm_stripe_size:   1048576
lmm_pattern:       1
lmm_layout_gen:    0
lmm_stripe_offset: 0
        obdidx           objid           objid           group
             0              12            0xc               0
             1              10            0xa               0

spoo123
lmm_stripe_count:  1
lmm_stripe_size:   1048576
lmm_pattern:       1
lmm_layout_gen:    1
lmm_stripe_offset: 0
        obdidx           objid           objid           group
             0              13            0xd               0
```

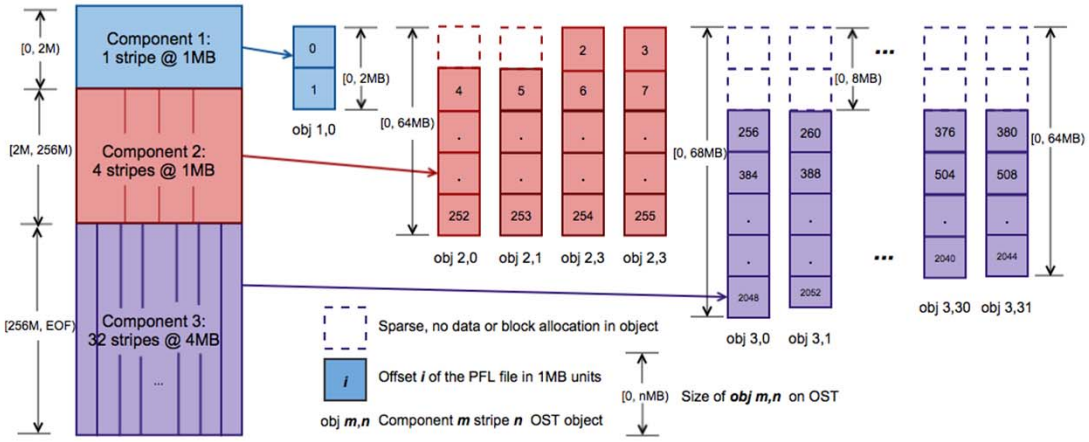

Progressive File Layout

Objective of PFL

Simplify layout management for user

- Layout can change during file growth
- Based on composite layout
  - An array of sub-layout components
  - Each sub-layout is a simple layout covering non overlapped extents of the file

Example of PFL

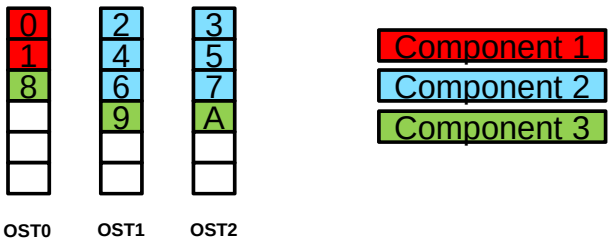


Mapping from 2055MB PFL file data blocks to OST objects of three components

PFL File Creation & Management

lfs setstripe, lfs getstripe, lfs migrate

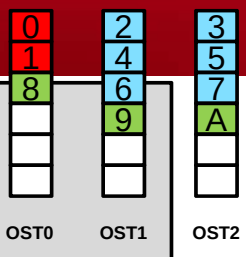
- Sub-layouts are defined by an ID
  - Allocated at layout creation
- Sub-layouts offset are set by their end
  - From end of previous to specified end
- Sub-layouts are set like simple layouts



Building a PFL

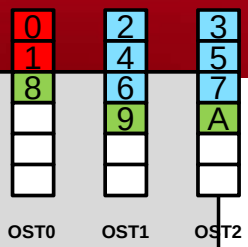
```
# lfs setstripe \
            -E 2M -c 1 -o 0 \
            -E 8M -c 2 -o 1,2 \
            -E -1 -c 3  pfl2
# lfs getstripe pfl2
pfl2
  lcm_layout_gen:  3
  lcm_entry_count: 3
    lcme_id:             1
    lcme_mirror_id:      0
    lcme_flags:          init
    lcme_extent.e_start: 0
    lcme_extent.e_end:   2097152
      lmm_stripe_count:  1
      lmm_stripe_size:   1048576
      lmm_pattern:       raid0
      lmm_layout_gen:    0
      lmm_stripe_offset: 0
      lmm_objects:
      - 0: { l_ost_idx: 0, l_fid:
[0x100000000:0x5:0x0] }
```

```
lcme_id:             2
lcme_mirror_id:      0
lcme_flags:          0
lcme_extent.e_start: 2097152
lcme_extent.e_end:   8388608
  lmm_stripe_count:  2
  lmm_stripe_size:   1048576
  lmm_pattern:       raid0
  lmm_layout_gen:    0
  lmm_stripe_offset: -1
lcme_id:             3
lcme_mirror_id:      0
lcme_flags:          0
lcme_extent.e_start: 8388608
lcme_extent.e_end:   EOF
  lmm_stripe_count:  3
  lmm_stripe_size:   1048576
  lmm_pattern:       raid0
  lmm_layout_gen:    0
  lmm_stripe_offset: -1
```



Filling a PFL File

```
# dd if=/dev/zero of=pfl2 bs=1M count=11
11+0 records in
11+0 records out
11534336 bytes (12 MB) copied, 0.0368475 s, 313 MB/s
# lfs getstripe pfl2
pfl2
  lcm_layout_gen:    5
  lcm_mirror_count:  1
  lcm_entry_count:   3
    lcme_id:             1
    lcme_mirror_id:      0
    lcme_flags:          init
    lcme_extent.e_start: 0
    lcme_extent.e_end:   2097152
      lmm_stripe_count:  1
      lmm_stripe_size:   1048576
      lmm_pattern:       raid0
      lmm_layout_gen:    0
      lmm_stripe_offset: 0
      lmm_objects:
      - 0: { l_ost_idx: 0, l_fid: [0x100000000:0xf:0x0] }
    lcme_id:             2
    lcme_mirror_id:      0
    lcme_flags:          init
    lcme_extent.e_start: 2097152
    lcme_extent.e_end:   8388608
      lmm_stripe_count:  2
      lmm_stripe_size:   1048576
      lmm_pattern:       raid0
      lmm_layout_gen:    0
      lmm_stripe_offset: 1
      lmm_objects:
      - 0: { l_ost_idx: 1, l_fid: [0x100010000:0xb:0x0] }
      - 1: { l_ost_idx: 2, l_fid: [0x100020000:0x2:0x0] }
    lcme_id:             3
    lcme_mirror_id:      0
    lcme_flags:          init
    lcme_extent.e_start: 8388608
    lcme_extent.e_end:   EOF
      lmm_stripe_count:  3
      lmm_stripe_size:   1048576
      lmm_pattern:       raid0
      lmm_layout_gen:    0
      lmm_stripe_offset: 1
      lmm_objects:
      - 0: { l_ost_idx: 2, l_fid: [0x100020000:0x3:0x0] }
      - 1: { l_ost_idx: 0, l_fid: [0x100000000:0x10:0x0] }
      - 2: { l_ost_idx: 1, l_fid: [0x100010000:0xc:0x0] }
```



Changing a PFL

Extending a PFL

- lfs setstripe –component-add –E -1 …

Deleting a component

- lfs setstripe –component-del
- Will remove associated data
- Must start from the end of file (hole is not allowed)

Changing a PFL

- lfs migrate

Find files

How to find files with Lustre specific criteria?

- lfs find
  - lfs find –component-count=3

OST Pools

Issue with Object Allocation

Homogenous OST

- Large dataset can be spread over many files
  - Lustre allocation algorithm will try to use globally as many OST as possible
  - This creates a dependency to all OSS/OST
- Same at user level
  - All user files are on all OST
  - This generate I/O collisions

Heterogeneous OST

- On large configuration OST's are not homogenous
- Would like to avoid multiple FS creation

OST Pools

Idea

- Associate OST's in a group named a Pool
- A Pool is an arbitrary subset of OSTs
- An OST can be a member of multiple pools
- No ordering of OSTs in a pool is defined or implied
- Stripe allocation in a pool follows the same rule as normal stripe allocator
- OST membership in a pool is flexible
- File created in a pool remain usable if pool disappears
  - Pools are used only at file allocation
  - Pools are advisory information (ignored if not found)
- Pool changes do not affect already created files

OST Pool Management

Pool creation

```
lctl pool_new FS.POOL
```

Pool extension/reduction

```
lctl pool_add FS.POOL OST_LIST/RANGE
lctl pool_remove FS.POOL OST_LIST/RANGE
```

Pool list

```
lctl pool_destroy FS.POOL
```


Pool Warning

- lctl command must be run on MGS
- lctl writeconf erases all pool information
  - It clears all Lustre conf
  - All targets/client need to re-register

OST Pool Usage

- Pool is “just” an option for layout management
  - lfs setstripe –p POOL
- Same use as usual but in a restricted sub-set of OSTs
- When displaying layout
  - The only difference is pool name field

OST Pool Usage

```
# lfs setstripe -c 2 -S 2M -p POOL1 spoo10
# lfs getstripe -v spoo10
spoo10
lmm_magic:          0x0BD30BD
lmm_seq:            0x200000401
lmm_object_id:      0x7
lmm_fid:            [0x200000401:0x7:0x0]
lmm_stripe_count:   2
lmm_stripe_size:    2097152
lmm_pattern:        raid0
lmm_layout_gen:     0
lmm_stripe_offset: 0
lmm_pool:           POOL1
        obdidx           objid           objid           group
             0               7            0x7               0
             1               7            0x7               0
```

Thank you for your attention

ENSIIE | 2018

Commissariat à l'énergie atomique et aux énergies alternatives
Centre DAM-Ile de France | 91297 Bruyères-le-Châtel Cedex
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 70 86
Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019

Direction des applications militaires
Département sciences de la simulation et de l'information
Service informatique scientifique et réseaux

Lustre Glossary

DNE - Distributed Namespace Environment - feature to aggregate multiple MDTs (possibly on many MDS's) into a single filesystem namespace

IDIF - OST object ID In FID - specific FID range reserved for compatibility with pre-DNE OST objects

IGIF - Inode and Generation In FID - specific FID range reserved for compatibility from Lustre 1.x MDT inode objects

FID - File IDentifier - unique 128-bit identifier for every object within a single filesystem.

LMV - Logical Metadata Volume - client software layer that handles client (llite) access to multiple MDTs

LOD - Logical Object Device - MDS software layer that handles access to multiple MDTs and multiple OSTs

LOV - Logical Object Volume - client software layer that handles client (llite) access to multiple OSTs

MDC - MetaData Client - client software layer that interfaces to the MDS

MDD - Metadata Device Driver - MDS software layer that understands POSIX semantics for file access

MDS - MetaData Server - software service that manages access to filesystem namespace (inodes, paths, permission) requests from the client.

MDT - MetaData Target - storage device that holds the filesystem metadata (attributes, inodes, directories, xattrs, etc)

MGS - Management Server - service that helps clients and servers with configuration

MGT - Management Target - storage device that holds the configuration logs

OFD - Object Filter Device - OSS software layer that handles file IO

OSC - Object Storage Client - client software layer that interfaces to the OST

OSD - Object Storage Device - server software layer that abstracts MDD and OFD access to underlying disk filesystems like ldiskfs and ZFS

OSP - Object Storage Proxy - server software layer that interfaces from one MDS to the OSD on another MDS or another OSS

OSS - Object Storage Server - software service that manages access to filesystem data (read, write, truncate, etc)

OST - Object Storage Target - storage device that holds the filesystem data (regular data files, not directories, xattrs, or other metadata)

SFP 2018 Parallel Filesystems / Lustre | PAGE 38