
whamcloud

whamcloud

European Lustre Workshop
Paris, France
September 2011

Hands on Lustre 2.x

- Johann Lombardi
  Principal Engineer
  Whamcloud, Inc.

Main Changes in Lustre 2.x

- MDS rewrite
- Client I/O rewrite
- New ptlrpc API called req_capsule
- Changelogs
- New File IDentifier (FID) and request format
  - incompatibility with 1.6/1.8 protocol

- More to come …
  - OSD restructuring

File Identifiers (FIDs)

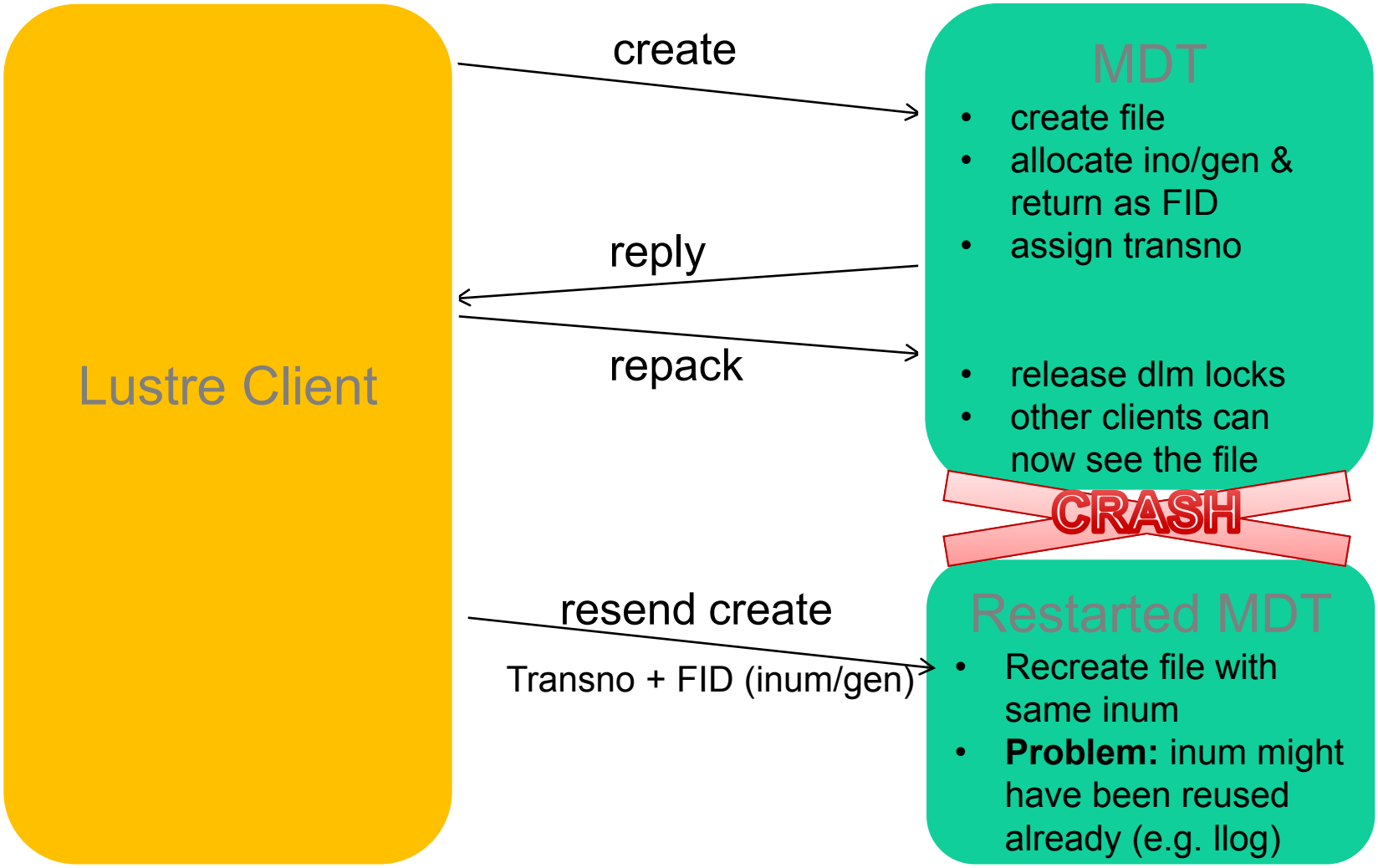
- All network filesystems rely on a file identifier
- Used to uniquely identify file/object in network request
- NFS uses a 64-bit file handle

FIDs in Lustre 1.8

- On the MDS, files are identified by:
  - 32-bit inode number allocated by underlying ldiskfs filesystem
  - 32-bit generation number also maintained by ldiskfs
- On the OSTs, objects are identified by:
  - 64-bit object id allocated sequentially starting from 1
  - 32-bit index which is the OST index in the LOV

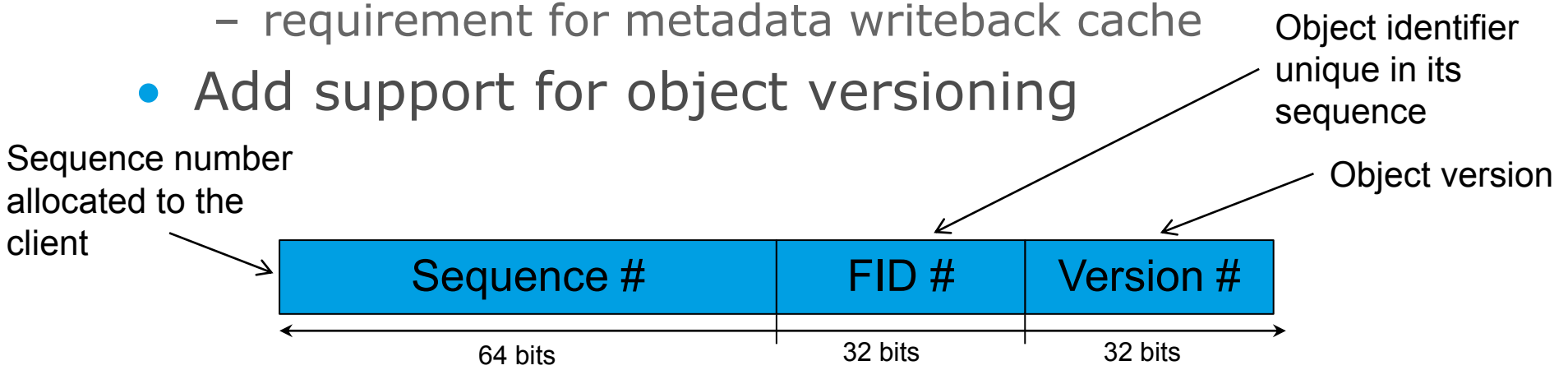
```
[client]# lfs getstripe foo
foo
lmm_stripe_count:   2
lmm_stripe_size:    1048576
lmm_stripe_offset:  0
      obdidx           objid           objid           group
           0               3             0x3               0
           1               3             0x3               0
```

Replay Issue



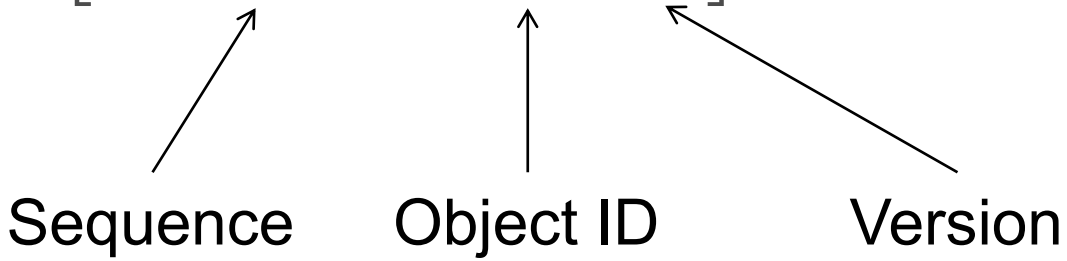
New FID Scheme in Lustre 2.x

- Independent of MDS backend filesystem
- Simplify recovery
  - e.g. no need to regenerate inode with specific inode number during replay
- Get rid of the infamous iopen patch
- Can be generated on the client
  - requirement for metadata writeback cache
- Add support for object versioning



FIDs in Practice

```
[client]# touch foo
[client]# lfs path2fid foo
[0x200000400:0x1:0x0]
```




```
[client]# ln foo bar
[client]# lfs fid2path /mnt/lustre [0x200000400:0x1:0x0]
/mnt/lustre/foo
/mnt/lustre/bar
```

Sequence

- Sequences are granted to clients by servers
- When a client connects, a new FID sequence is allocated
  - upon disconnect, new sequence is allocated to client when it comes back
- Each sequence has a limited number of objects which may be created in it
  - on exhaustion, a new sequence should be started
- Sequences are cluster-wide and prevent FID collision

Sequence in Practice

```
[client]# lctl get_param seq.cli-srv*.*
seq.cli-srv-xxxxx.fid=[0x200000400:0x1:0x0]
seq.cli-srv-xxxxx.server=lustre-MDT0000_UUID
seq.cli-srv-xxxxx.space=[0x200000401 - 0x200000401]:0:0
seq.cli-srv-xxxxxx.width=131072

[client]# touch foobar
[client]# lfs getstripe -v ./foobar
./foobar
lmm_magic:          0x0BD10BD0
lmm_seq:            0x200000400
lmm_object_id:      0x2
lmm_stripe_count:   1
lmm_stripe_size:    1048576
lmm_stripe_pattern: 1
lmm_stripe_offset:  1
      obdidx       objid       objid         group
           1           3         0x3             0
```

last allocated FID

unused sequence allocated to this client

number of FIDs that can be generated out of a sequence

Where are FIDs stored? (1/2)

- The underlying filesystem still operates on inodes
- An object index is stored on disk to handle FID/on-disk inode mapping
- For ldiskfs, the object index is an IAM lookup table (namely oi.16)

```
debugfs:  ls
    2(12)   .              2(12) ..             11(20) lost+found
   12(16)   CONFIGS    25001(16) OBJECTS       19(20) lov_objid
   22(16)   oi.16         23(12) fld           24(16) seq_srv
   25(16)   seq_ctl       26(20) capa_keys  25002(16) PENDING
25003(12)   ROOT          27(20) last_rcvd  25004(20) REM_OBJ_DIR
   31(3852)CATALOGS
```

Where are FIDs stored? (2/2)

- The FID is also stored:
  - in an extended attribute called LMA
    - stands for Lustre Metadata Attributes
    - also stores SOM/HSM states
    - see struct lustre_mdt_attrs for the format
  - in the directory entry, along with the filename
    - path->FID translation does not require accessing LMA XATTR
    - ext4 & e2fsprogs patch to support this feature

Dump/Restore with Lustre 2.x

- Object Index (OI) stores FID->ino translation
- FID also stored in LMA XATTR
- Dump/Restore of the MDT requires *either*:
  - restoring files with original inode number so that the OI is still valid
    - only block-level copy can grant this

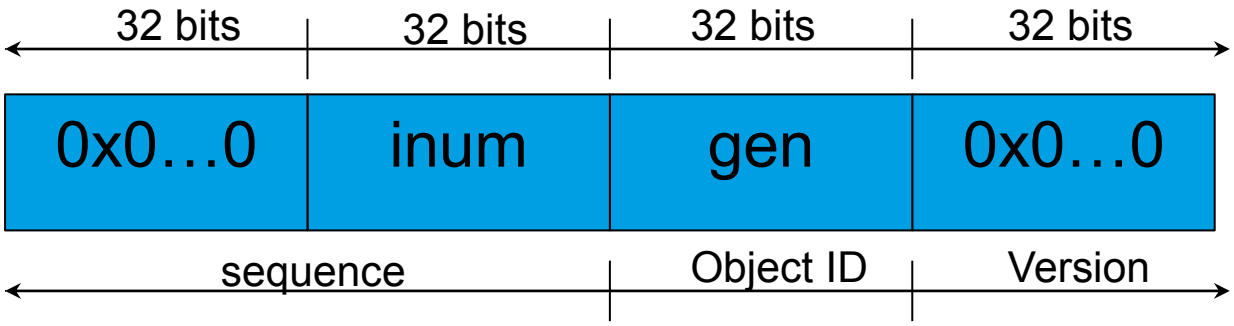
  OR
  - fixing the OI with new inode numbers
    - not possible currently
    - will be possible in the future with OI scrubbing (already funded by OpenSFS)

Link Extended Attribute

- New XATTR storing list of parent FIDs and names
- Useful for:
  - verifying directory hierarchy
  - FID to path translation
    - `lfs fid2path`
  - updating parent directory entries when migrating files
  - POSIX lookup-by-FID path permission checks

Compatibility Mode: IGIF

- Filesystems upgraded from 1.8 don't have fid stored in EA or in directory entry
- Name/fid mapping handled by IGIF
- IGIF allows to reversibly map inode/generation into FID
- Special sequence range reserved
  - up to ~4B of inodes



Interoperability Gotchas

- Upgrade from 1.8 to 2.x is supported
  - files created with 1.8 MDT use IGIF
  - new files use new FID scheme
- The 1.8 client understands the new FID format
  - 1.8 clients can talk to 2.x servers
- The 2.x client does not understand the old FID format
  - 2.x clients cannot talk to 1.8 servers
  - servers must be upgraded first
- Upgrade from 1.8 to 2.x with active clients **NOT** possible
  - all 1.8 clients are evicted during the upgrade

Configuration Parameter Glitches

- Procfs & binary paths of group upcall have changed
  - from `mdt.group_upcall=/path/to/l_getgroups` to `mdt.identity_upcall=/path/to/l_getidentity`
  - upgraded filesystems have the former in configuration log
  - warning message printed when trying to set `mdt.group_upcall` on 2.x, but mount still successful
  - upgraded filesystems use `NONE` by default and this can be fixed by running on the MGS:

```
lctl conf_param $FSNAME-MDT0000.mdt.identity_upcall=/path/to/l_getidentity
```

  - old param can be removed with lctl conf_param –d
- mdt.quota_type is now mdd.quota_type
  - patch available in LU-110

Ext4 Changes in 2.1

- Flexible block group (flex_bg) used by default
  - co-locate group bitmaps and inode tables to provide larger contiguous free spaces
  - avoid costly seeks for both data/metadata
- 128TB OST support
  - tested & validated
  - e2fsck fast thanks to flex_bg
    - Full e2fsck of a 128TB OST with 32k 4GB files takes 32 mins
- Support objects larger than 2TB
  - ext3 limits size of an individual object to 2TB
  - limit also hardcoded in lustre client
  - ext4 reports new limit (16TB) in superblock and lustre clients fetch this parameter at connect time

Merci

- Johann Lombardi

  Principal Engineer

  Whamcloud, Inc.