
whamcloud

whamcloud

Supercomputing 2011
Seattle, WA

Hands on Lustre 2.1

- Andreas Dilger
  Principal Engineer
  Whamcloud, Inc.

What's new in Lustre 2.1

- RHEL 6.1 server support (2.6.32 kernel)
- Increased OST size limit
- Improved mkfs and fsck performance
- Increased Lustre filesystem/file size limits
- Mkfs denser MDT inodes, fewer OST inodes
- Asynchronous journal commit on OST writes

New in Lustre 2.0

- ChangeLogs
- Commit-on-Share
- Limited MDT backup/restore options

Ext4 Changes in Lustre 2.1

- Flexible block group used by mkfs (*flex_bg*)
  - co-locates multiple block/inode bitmaps, inode tables
  - provides larger contiguous free spaces
  - avoids seeks for both data/metadata/e2fsck
- 128TB OST support (*64bit*)
  - tested & validated at this size (upper limit is higher)
  - e2fsck much faster, thanks to flex_bg and fewer inodes
    - Full e2fsck of 128TB OST with 32k 4GB files in 35 minutes
    - Full e2fsck of 128TB OST with 134M 0-byte files under 8 minutes
- Support OST objects up to 16TB (*huge_file*)
- Denser MDT inodes by default
  - 2048 bytes per MDT inode, important for flash storage
  - up to 1MB per OST inode, improves fsck time

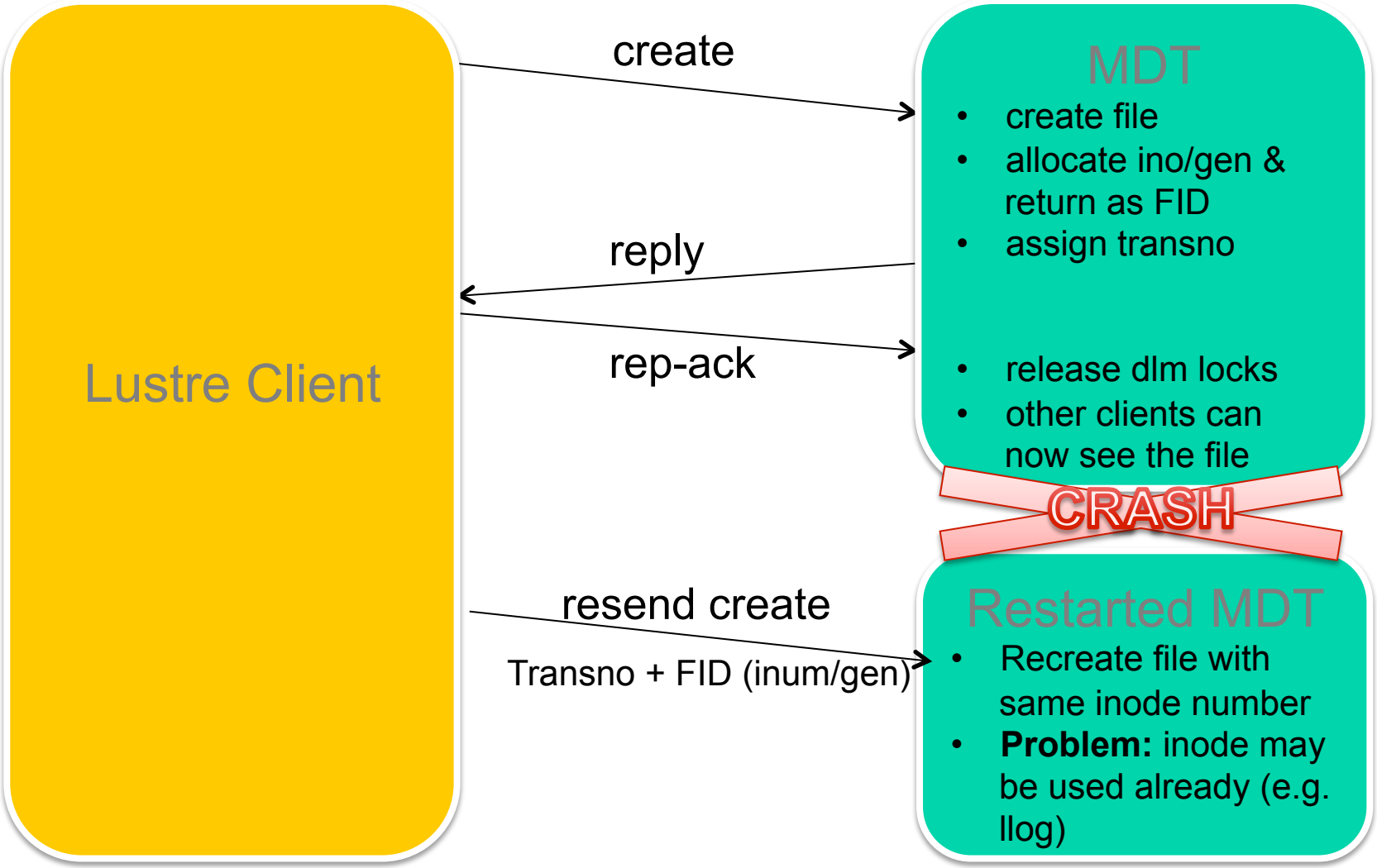
Main Code Changes Since Lustre 1.8

- MDS stack rewrite
  - Prepare for new back-end filesystems, distributed namespace
- Client I/O rewrite
- ChangeLogs
  - Record of changes to filesystem for HSM and other policy engines
- New ptlrpc API called req_capsule
- New File IDentifier (FID) and request format
  - Incompatible with Lustre 1.6/1.8 RPC protocol
  - 1.8 clients understand these changes
- More restructuring to come for 2.2
  - OSD API updated for new storage backend (e.g. ZFS)

File Identifiers in Lustre 1.8

- Uniquely identify file/object in RPCs
  - All network filesystems have them
- Lustre 1.8 uses two different identifiers
  - MDT inodes use 32-bit inode + 32-bit generation
    - MDT inode + generation allocated by ldiskfs
    - Only unique to a single MDT
    - Can change after backup/restore
    - Exposed to userspace, shouldn't change (NFS, tar)
  - OST objects use 64-bit object ID + 32-bit OSD number
    - OST object ID is allocated by Lustre
    - Objid is the same after backup/restore
    - Not exposed to userspace

1.8 Replay Issue



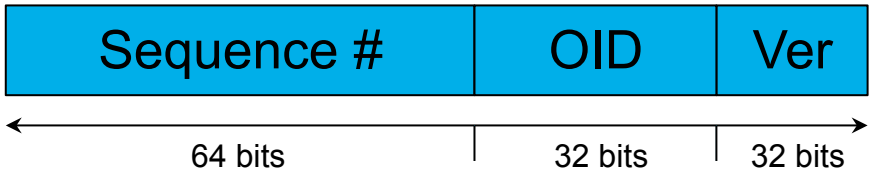
File Identifiers in Lustre 2.x

- Lustre 2.0+ MDT uses a 128-bit FID
  - 64-bit sequence number (new sequence for each client)
  - 32-bit object ID (sequential within each sequence)
  - 32-bit version (for snapshots, datasets; currently unused)
  - unique across entire Lustre filesystem
  - not generated by underlying disk filesystem
  - all files have 64-bit inode numbers (32-bit for 32-bit binary)
- Lustre 2.3+ OST will use same 128-bit FID
  - MDS (client) will be assigned a new sequence for each OST

New FID Scheme in Lustre 2.x

- Independent of backend filesystem
- Simplify recovery
  - no need to recreate specific inode number in replay
- FID can be generated on the client
  - requirement for metadata writeback cache
- Add support for object versioning
  - future use by snapshots, datasets

```
[client]# lfs path2fid foobar
[0x200000400:0x123:0x0]
```



FID Sequences

- Super Sequences granted to SEQ manager
  - SEQ managers (servers) may be MDT (2.0+) and OSTs (2.3+)
  - super sequence is very large (1B sequences), low traffic
- Sequences granted to clients by servers
  - new sequence allocated to client for each mount
  - clients can allocate 128k files, then get a new sequence
- Sequences cluster-unique, prevents FID collision

Where are FIDs Stored?

- Underlying ext4/ldiskfs still depends on inodes
- Object Index stores FID/inode mapping table
  - The ldiskfs object index is an IAM table (oi.16)
- In inode Lustre Metadata Attribute (*LMA*)
  - xattr also stores SOM/HSM states
  - see struct lustre_mdt_attrs for the format
- In inode *link* xattr (more on this later)
- In directory entry with filename, inode number
  - path->FID translation does not require accessing LMA xattr
  - ext4 & e2fsprogs patch to support this feature (*dir_data*)

link Extended Attribute

- xattr stores list of hard links to each inode
  - { parent FID, "filename" } for each hard link
  - normally limited in number, fits in inode
- LinkEA is very useful for:
  - verifying directory hierarchy
  - efficient FID to path translation via *lfs fid2path*
  - update parent directory entries when moving inodes
  - POSIX lookup-by-FID path permission checks
  - more easily generate pathnames for error messages

```
{[0x200000400:0x1:0x0], "foo" },
{[0x200000400:0x1:0x0], "bar" },
{[0x200000400:0x124:0x0], "baz" },
```

FIDs and link xattr in Practice

```
[client]# touch foobar
[client]# lfs path2fid foobar
[0x200000400:0x123:0x0]

[client]# lfs getstripe -v foobar
lmm_seq:            0x200000400
lmm_object_id:      0x123
  :                  :

[client]# ls -i foobar
1441152052557253 47 foobar
[client]# printf "%#x\n" $(stat -c %i /mnt/lustre/etc/hosts)
0x200000400000123

[client]# ln foo bar; mkdir tmp; ln foo tmp/baz
[client]# lfs fid2path /mnt/lustre [0x200000400:0x123:0x0]
/mnt/lustre/foo
/mnt/lustre/bar
/mnt/lustre/tmp/baz
```

Interoperability Constraints

- Upgrade from 1.8 to 2.x **IS** supported
  - old files created with 1.8 MDT use IGIF
  - new files use new FID scheme
  - no FID-in-directory until *dirdata* feature enabled
- 1.8.6+ client understands the new FID format
  - 1.8.6+ clients can talk to 2.x servers
- 2.x client does not understand old 1.8 format
  - 2.x clients **CANNOT** talk to 1.8 servers
  - servers must be upgraded before or with clients
- Live upgrade 1.8 to 2.x servers **NOT** possible
  - 1.8 clients evicted on upgrade (due to RPC format)
  - reconnect immediately with new RPC protocol

Backup/Restore with Lustre 2.x

- Object Index (OI) stores FID->inode mapping
- FID also stored in LMA xattr
- Backup/Restore of the MDT requires *either*:
  - restore files with original inode number
    - this doesn't happen with tar or rsync!
    - block-device copy (dd) does this (faster than tar!)
    - ensures that FID->inode mapping still valid

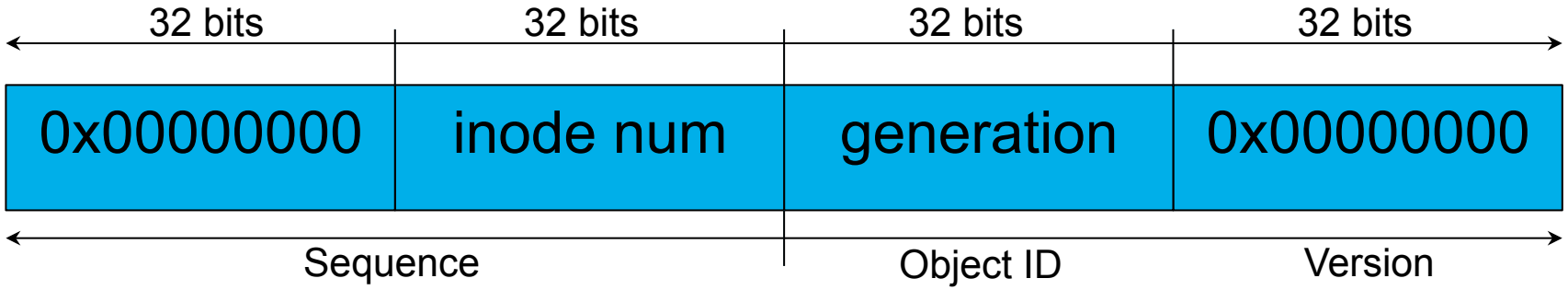
OR

  - fix OI/link with new inode numbers after restore (soon)
    - backup on 64-bit client (or may confuse tar/rsync)
    - online OI scrub fixes this (work in progress now)

Compatibility Mode: IGIF

- Upgraded 1.8 inodes don't have FIDs or link
- Allows reversible inode/generation to FID map
- First 4B sequence range reserved for IGIF

[client]# lfs path2fid oldfoo
[0x208020:0x1281aaf:0x0]



Thank You

- Andreas Dilger
  Principal Engineer
  Whamcloud, Inc.