LAD14 Reims France 2014

Intel Features and Future Development

Andreas Dilger

Principal Engineer

Overview

Features currently under development

- Dynamic LNET Configuration
- LFSCK MDT-MDT consistency checking
- Distributed Namespace Striped Directories
- Data on MDT (DOM)

Features in the design stage

- Layout enhancement
- Multiple metadata-modifying RPCs (multi-slot last_rcvd)

Dynamic LNET Config (2.7)

Allows configuring complex LNET networks at runtime

- New commands to configure networks and routes (like "route")

```
lnetctl net add --net {} --if {} [--peer_{credits,timeout} {}]
lnetctl net del --net {}
lnetctl net show [--net {}] [--verbose]

lnetctl route add --net {} --gateway {} [--hop {}]
lnetctl route del --net {}
lnetctl route show [--net {}] [--verbose]
lnetctl set {tiny,small,large}_buffers 8192
```

http://cdn.opensfs.org/wp-content/uploads/2014/04/D1_S13_DynamicLNETConfiguration.pdf

Dynamic LNET Config cont.

Config file for automatic configuration at startup/shutdown

- Will eventually replace lnet module parameters
- YAML format to be both human/machine readable
- Generate YAML config file from current settings on local node:

```
# lnetctl net show --verbose

net:
 - nid: 192.168.205.130@tcp1
   status: up
   interfaces:
     0: eth3
     1: eth4
   tunables:
     peer_timeout: 180
     peer_credits: 8
     peer_buffer_credits: 0
     credits: 256
```

```
# lnetctl route show --verbose

route:
 - net: tcp6
      gateway: 192.168.29.1@tcp
      hop: 4
      seq_no: 3
 - net: tcp7
      gateway: 192.168.28.1@tcp
      hop: 9
      seq_no: 4
```

LFSCK OST-MDT Checking (2.4-2.5)

Iterate MDT objects, check local consistency (Phase 1)

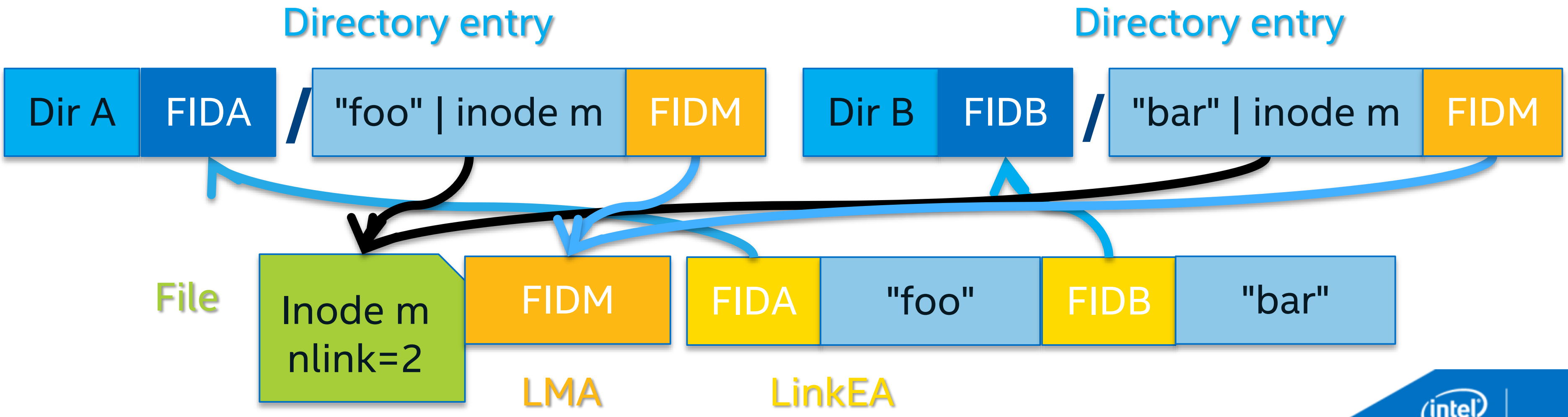
- Check/fix Object Index (OI Scrub) for FID->inode mapping

```
lctl lfsck_start -M fsname-MDT0000 [-t scrub] [--dry-run] [-s obj_sec]
lctl lfsck_stop -M fsname-MDT0000
```

Iterate names in each directory, check local namespace (Phase 1.5)

- Check/fix FID in dirent, "link" xattr for inode->parent dir backref

```
lctl lfsck_start  -M fsname-MDT0000 -t namespace ...
```



LFSCK OST-MDT Checking (2.6)

Iterate OST objects, check MDT-OST layout consistency (Phase 2)

- Verify OST local object directory entry against object ID
- Verify OST object->MDT inode back-reference ("fid" xattr)
- Generate in-memory bitmap of in-use FIDs on each OST

MDS iteration checks LOV layout on each inode

- Verify each OST object exists, optionally recreate missing objects
- Verify UID/GID on objects for quota
- As each object FID is verified, mark it in-use in bitmap
- Find missing or duplicate OST object references via bitmap

lctl lfsck_start ... [**-A**] -t **layout** [**--create-ostobjs**] ...

Added OSD object iterator for ZFS OSDs (OST, MDT)

LFSCK MDT-MDT Checking (2.7)

Check distributed consistency between MDTs (Phase 3)

- During Phase 1 MDT namespace iteration, verify and repair remote inodes
- Use Object Update Target (OUT) to access/modify remote MDT objects
- OUT between servers only, allows low-level object/index changes
- Check and repair remote LinkEA, nlinks, file type in dirent

Check and repair striped DNE directories

- Check master/slave directories, names hashed to correct slave

```
lctl lfsck_start ... -A -t all
lctl set_param debug=+lfsck
watch "lctl get_param mdd.*.lfsck_{layout,namespace}"
```

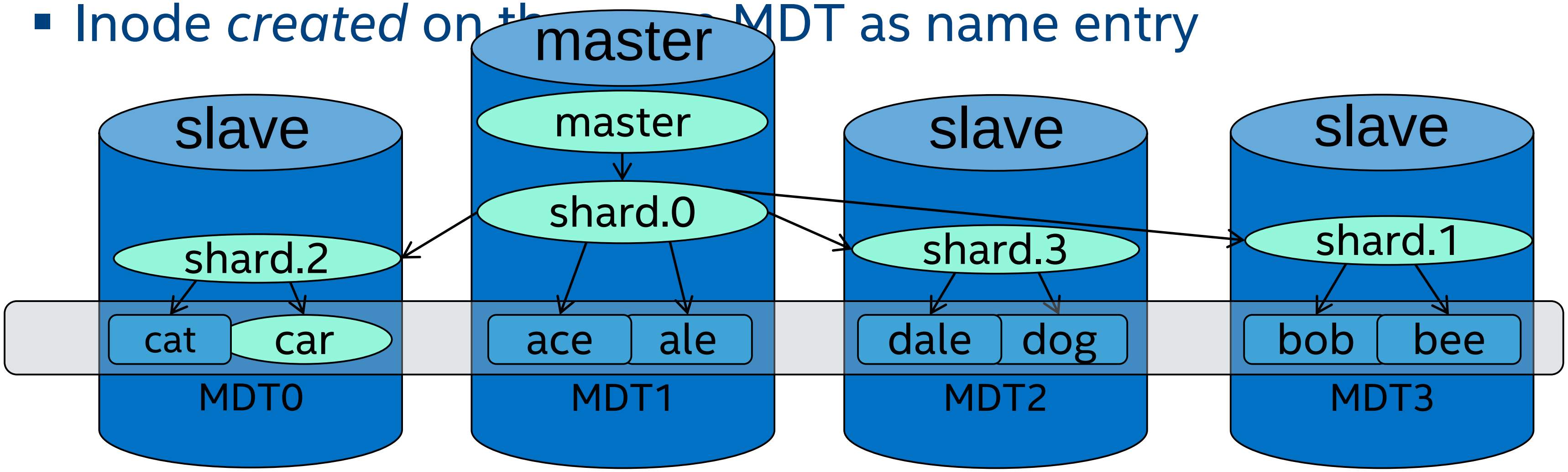
http://cdn.opensfs.org/wp-content/uploads/2013/04/Zhuravlev_LFSCK.pdf

http://wiki.opensfs.org/Contract_SFS-DEV-001

DNE Striped Directories (2.6/2.8)

Spread a single directory across MDTs

- Reduce contention, improve performance for large directories
- Directory layout + name hash locates slave MDT directory entry
- Directory shard on each MDT independent (lock, lookup, modify)
- Inode *created* on the same MDT as name entry



DNE Striped Directories (2.6/2.8)

MDT and directory layout statically selected at creation time

mkfs.lustre --mdt ... **--index mdt_idx** /dev/*mdtN*

lfs **mkdir -i mdt_idx** [-c {stripe_count}] *new_directory*

rmdir *empty_directory*

Migration tool to balance MDT usage

- Avoids data copy, moves objects to **new inode** on new MDT
  - Changes **FID** of the inode, not POSIX rename() compatible

lfs **mv -M *mdt_idx*** *file_or_directory*

https://wiki.hpdd.intel.com/display/PUB/Remote+Directories+Solution+Architecture

http://cdn.opensfs.org/wp-content/uploads/2013/04/LUG-2013_DNE.pdf

DNE Async Commit (2.8)

DNE remote/striped directory create currently synchronous (2.6)

- Use OSP to send *updates* to remote MDTs
- Need a sync on **both** slave and master to ensure consistency
- Files created within remote/striped directories NOT synchronous
- Rename and hard links not supported (return -EXDEV)

Async commit implements distributed DNE recovery

- Allow remote/striped operations to avoid sync updates
- Each target (master/slave) logs full copy of all updates
- Can replay update log if any target failed to commit updates

Data on MDT (2.x)

Efficiently store small files on the MDT

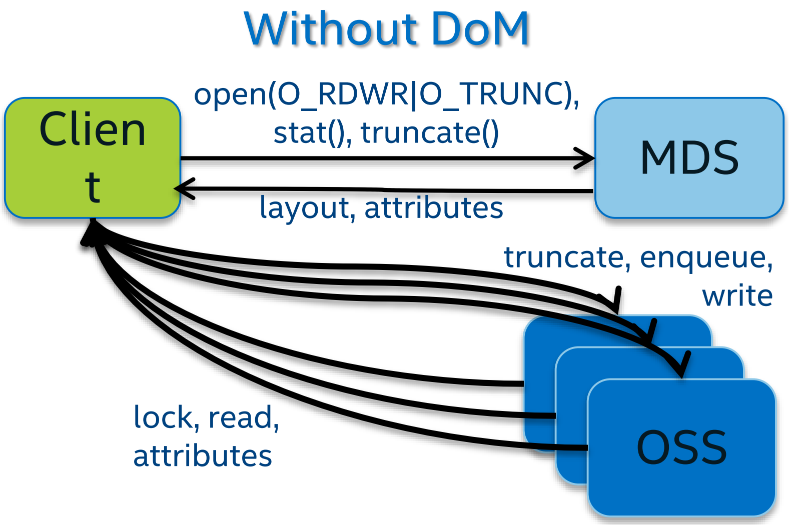
- Avoid OST object RPC + disk seek for each file access
- Avoid OST lock RPC for each file access
- Use small-file optimized MDT storage (RAID-10/SSD/NVRAM)
- Avoid RAID-5/6 read-modify-write for small writes

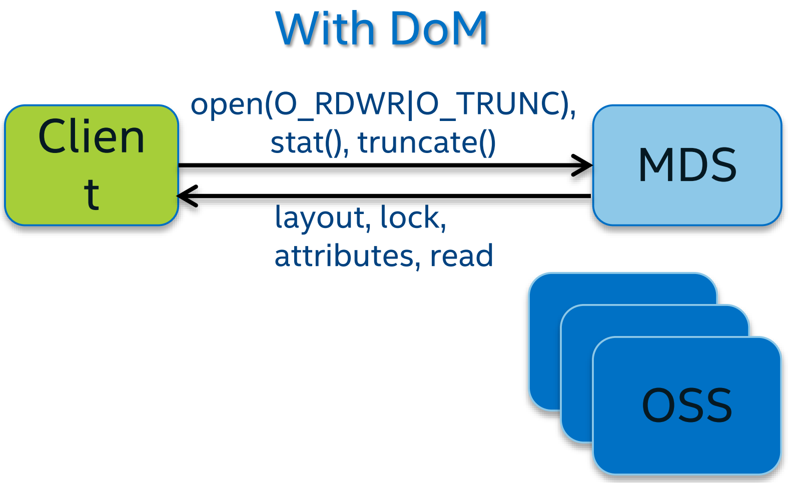
Space usage on MDT managed by quota

Small files are determined by the file layout

- Maximum MDT file size can be specified by min(user, admin)
- Typically expected to be <= 1MB, dependent on MDT space
  - Phase 1: Files larger than limit cannot be stored on the MDT (EFBIG)
  - Phase 2: Files larger than limit will be migrated to an OST

Data on MDT Implementation




With DoM
Clien t
open(O_RDWR|O_TRUNC), stat(), truncate()
MDS
layout, lock, attributes, read
OSS

DoM requested at file creation time like files on OSTs

- Can't do it after write because objects are allocated at open()
- Can set default DoM striping on subdirectories (phase 2)

lfs setstripe **--stripe-pattern=mdt** [--stripe-size=*size*] *new_file*

http://cdn.opensfs.org/wp-content/uploads/2014/04/D1_S10_LustreFeatureDetails_Pershin.pdf

http://wiki.opensfs.org/Contract_SFS-DEV-003

Multiple Metadata-Modifying RPCs (2.x)

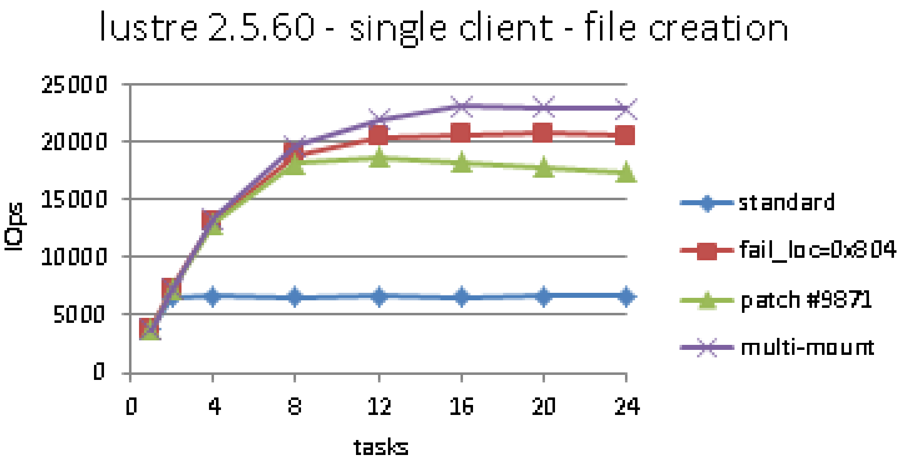
(aka Multi-slot **last_rcvd**)

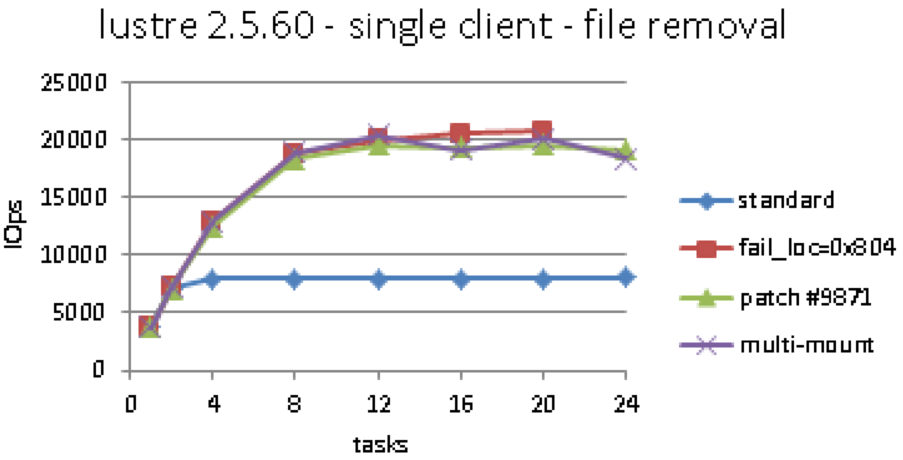
Currently limited to one RPC (+close) at client

- last_rcvd slot on MDT for each client to reconstruct reply
- Not a limit for many concurrent clients

Change to dynamic log on MDT to allow concurrent RPCs

- Allow multiple metadata-modifying RPCs in flight at one time
- Improve multi-threaded performance of one client


lustre 2.5.60 - single client - file creation


lustre 2.5.60 - single client - file removal

Layout Enhancement (2.y)

Allow compound layouts for regular files

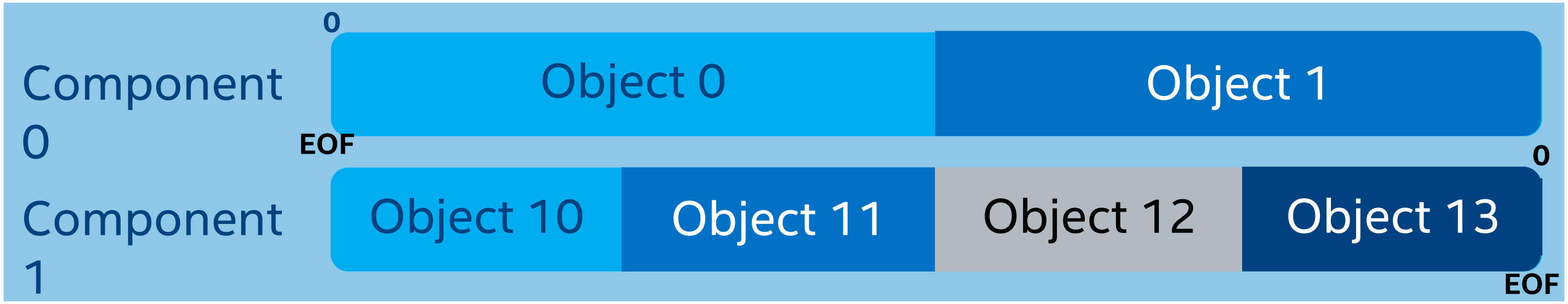
- Component layouts describe extent of file (some or all)
- Layout extents can be disjoint or overlapping
    - RAID-1 mirroring -> overlapping [0, EOF), [0, EOF)
    - Dynamic stripes -> disjoint [0, 32M), [32M, 1G), [1G, EOF)

```
struct lov_comp_md_v1 {
        __u32 lcm_magic;
        __u32 lcm_size;    /* total layout size */
        __u32 lcm_layout_gen;
        __u16 lcm_flags;  /* READ_ONLY, ... */
        __u16 lcm_entry_count;
        union {
                __u64 lcm_padding[2];
        } u;
        struct lov_comp_md_entry_v1 lcm_entries[0];
};
```

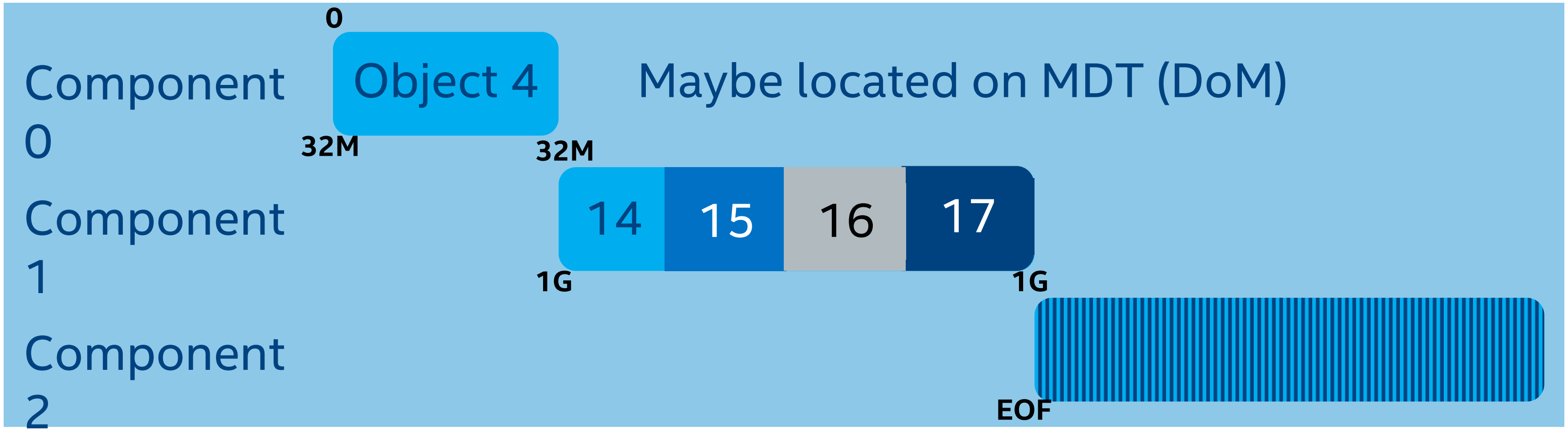
```
struct lov_comp_md_entry_v1 {
        __u32 lcme_id;      /* unique ID */
        __u32 lcme_flags; /* PRIMARY, STALE, ... */
        struct lu_extent lcme_extent;
        __u32 lcme_offset; /*  layout entry offset */
        __u32 lcme_size;    /* size of entry */
        __u64 lcme_padding;
};
```

Layout Enhancement Examples

RAID-0+1 mirrored file



Dynamic Striping



intel®