# Lustre 2.12 and Beyond

Andreas Dilger, Whamcloud

# Upcoming Release Feature Highlights

**Whamcloud**

▶ 2.12 is feature complete

- LNet Multi-Rail Network Health – improved fault tolerance
- Lazy Size on MDT (LSOM) – efficient MDT-only scanning
- File Level Redundancy (FLR) enhancements – usability and robustness
- T10 Data Integrity Field (DIF) – improved data integrity
- DNE directory restriping – better space balancing and DNE2 adoption

▶ 2.13 development and design underway

- Persistent Client Cache (PCC) – store data in client-local NVMe
- File Level Redundancy – Phase 2 Erasure Coding
- DNE auto remote directory/striping – improve load/space balance across MDTs

▶ 2.14 plans continued functional and performance improvements

- DNE directory auto-split – improve usability and performance of DNE2
- Client metadata Write Back Cache (WBC) – improve interactive performance, reduce latency

# LNet Network Health, UDSP (2.12/2.14)

▶ Detect network interface and router failures automatically

- Builds on LNet Multi-Rail in 2.10/2.11 (LU-9120 Intel/WC, HPE/SGI)
- **LNet fault tolerance** without lengthy Lustre recovery, optimize resend path
- Handle multi-interface router failures                    DONE

▶ User Defined Selection Policy (LU-9121 WC, HPE)          TODO

- Fine grained control of interface selection
  ○ TCP vs. IB networks, primary vs. backup
- Optimize RAM/CPU/PCI data transfers
- Useful for large NUMA machines
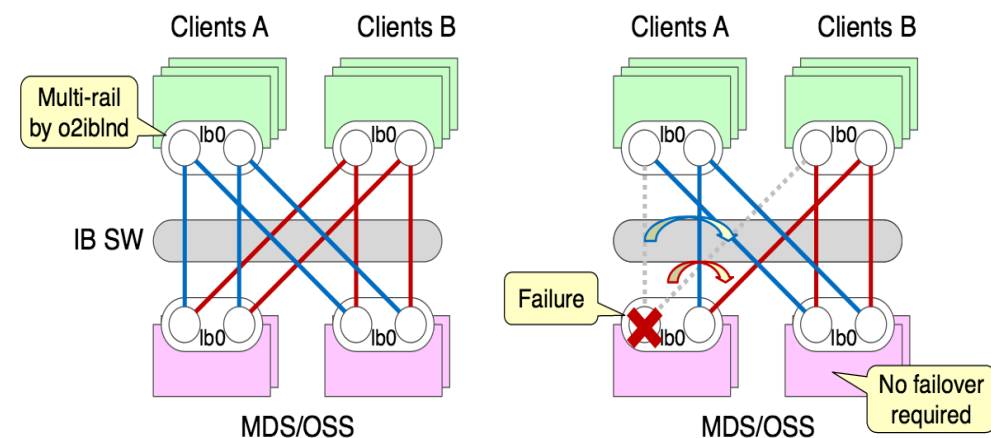
# Data-on-MDT Improvements  (LU-10176 WC 2.12+)

▶ Complementary with DNE 2 striped directories
  - Scale small file IOPS with multiple MDTs
▶ **Read-on-Open** fetches data (LU-10181)
  - Reduced RPCs for common workloads
▶ Improved locking for DoM files (LU-10175)
  - Drop layout lock bit without full IBITS lock cancellation
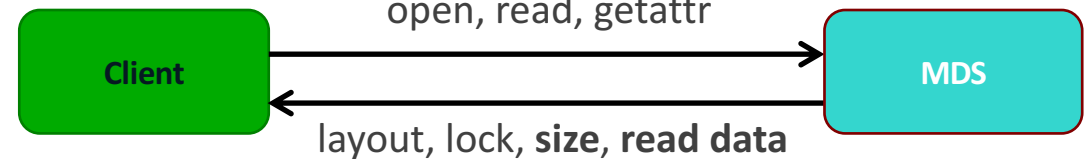  - Avoid cache flush and extra RPCs
  - Convert write locks to read locks
▶ Migrate file/component from MDT to OST (LU-10177)          DONE
▶ Migrate file/component from OST to MDT via FLR (LU-11421)          TODO
▶ Cross-file data prefetch via statahead (LU-10280)

```
                   open, read, getattr
   Client  ──────────────────────────────►  MDS
           ◄──────────────────────────────
                 layout, lock, size, read data
```

**Small file read directly from MDS**

# DNE Improvements                            (WC 2.12/2.13)

▶ **Directory restriping** from single-MDT to striped/sharded directories ([LU-4684](#))

 • Rebalance MDT space usage, improve large directory performance

▶ Automatically create new remote directory on "best" MDT with `mkdir()`

 • Simplifies use of multiple MDTs without striping all directories, similar to OST usage

 • In 2.11 in userspace for `lfs mkdir –i -1` ([LU-10277](#))                    DONE

▶ **Automatic MDT load balancing** at directory creation time ([LU-10784](#), [LU-11213](#))    TODO

 • Using standard `mkdir()` calls with parent `stripe_idx = -1`

▶ **Automatic directory restriping** as directory size grows ([LU-11025](#))

 • Create one-stripe directory for low overhead, scale shards/capacity/performance with size

 • Add extra directory shards when master directory grows large enough (e.g. 10k entries)

 • Move existing direntries to new directory shards

 • New direntries and inodes created on new MDTs

Master   +4 dir shards     +12 directory shards
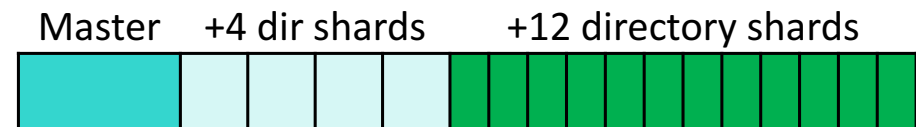
# ZFS Enhancements Related to Lustre (2.12+)

**Whamcloud**

▶ Lustre 2.12 `osd-zfs` updated to use ZFS 0.7.9
- Bugs in ZFS 0.7.7/0.7.10/0.7.11, not used by Lustre branch
- Builds with upstream ZFS pre-0.8.0 branch

▶ **Features in ZFS 0.8.x** release (target 2019Q1)
- Depends on final ZFS release date, to avoid disk format changes
- Sequential scrub/resilver (Nexenta)
- On-disk data encryption + QAT hardware acceleration (Datto)
- **Project quota** accounting (Intel)
- **Device removal** via VDEV remapping (Delphix)
- **Metadata Allocation Class** (Intel, Delphix)                LANDED
- Declustered Parity RAID (dRAID) (Intel)                IN PROGRESS

**OpenZFS**

# Miscellaneous Improvements

**Whamcloud**

▶ **UID/GID policy** for Token Bucket Filter (NRS-TBF) ([LU-9658](#) DDN)

▶ Improved JobStats allows admin-formatted JobID ([LU-10698](#) Intel)

    lctl set_param jobid_env=SLURM_JOB_ID jobid_name=cluster2.%j.%e.%p

▶ HSM infrastructure improvement & optimizations (Intel/WC, Cray)

- Improve Coordinator ([LU-10699](#)), POSIX Copytool ([LU-11379](#)), > 32 archives ([LU-10114](#)), …

▶ **Lazy Size-on-MDT** for local scan (purge, HSM, policy engine) ([LU-9358](#) DDN)

- LSOM is not guaranteed to be accurate, but good enough for many tools
- LSOM made available on client for apps *aware of limitations* (e.g. `lfs find`, `statx()`, …)

▶ **Lustre-integrated T10-PI** end-to-end data checksums ([LU-10472](#) DDN)

- Pass data checksums between client and OSS, avoid overhead, integrate with hardware      DONE

▶ Dump/restore of conf_params/set_param -P parameters ([LU-4939](#) Cray)      TODO

    lctl --device MGS llog_print testfs-client > testfs.cfg; lctl set_param -F testfs.cfg

# Improved Client Efficiency                                    (2.12+)

▶ **Disconnect idle clients** from OSS ([LU-7236](#) Intel)

  • Reduce memory usage on client and server for large systems

  • Reduce network pings and recovery times

▶ **Aggregate `statfs()`** RPCs on the MDS ([LU-10018](#) WC)

  • Reduce OSS overhead, avoid idle OST reconnection on client df

                                                                    DONE
_____

▶ Improved client read performance ([LU-8709](#) DDN)                 TODO

  • Improved readahead code, asynchronous IO submission

▶ Reduce wakeups and background tasks on idle clients ([LU-9660](#) Intel)

  • Synchronize wakeups between threads/clients (per jobid?) to minimize jitter

  • Still need to avoid DOS of server if all clients ping/reconnect at same time

# Performance Improvements for Flash (WC 2.12+)

Whamcloud

▶ **Reduce server CPU** overhead to improve small flash IOPS

- Performance is primarily CPU-limited for small read/write
- Any reduction in CPU usage directly translates to improved IOPS

▶ **Avoid page cache** on flash OSS (LU-11347)

- Avoids CPU overhead/lock contention for page eviction
- Streaming flash performance is often network limited
- ZFS drops pages from cache after use (LU-11282)          DONE

▶ Improved efficiency of ZFS IO pipeline          TODO

- Integrate with ABD in ZFS 0.8 to avoid `memcpy()` of data
- Further improvements with continued investigation/development

# FLR Enhancements                                 (WC 2.12+)

**Whamcloud**

► Continuation of FLR feature landed in Lustre 2.11 ([LU-9771](#))
► FLR-aware OST object allocator to avoid replicas on same OST/OSS ([LU-9007](#))
► Improve "`lfs mirror resync`" performance ([LU-10916](#))
  • Optimize multi-mirror resync (read data once, write multiple mirrors)
► **"`lfs mirror read`"** to dump specific mirror/version of file ([LU-11245](#))
► **"`lfs mirror write`"** for script-based resync ([LU-10258](#))
► **Mirror NOSYNC flag** + timestamp to allow file version/snapshot ([LU-11400](#))        DONE
► Improved replica selection at runtime ([LU-10158](#))                                    TODO
  • Select best write replica (PREFERRED, SSD vs. HDD , near to client), read (many mirror vs. few)
  • Allow specifying fault domains for OSTs (e.g. rack, PSU, network switch, etc.)
► Mount client directly on OSS for improved resync performance ([LU-10191](#))
► Support DoM components ([LU-10112](#))
► Pool or OST/MDT quotas ([LU-11023](#))
  • Track/restrict space usage on flash OSTs/MDTs
► **Erasure Coded** striped files ([LU-10911](#))

| Replica 0 | Object *j* (PRIMARY, PREFERRED) | |
| Replica 1 | Object *k* (STALE) | *delayed resync* |
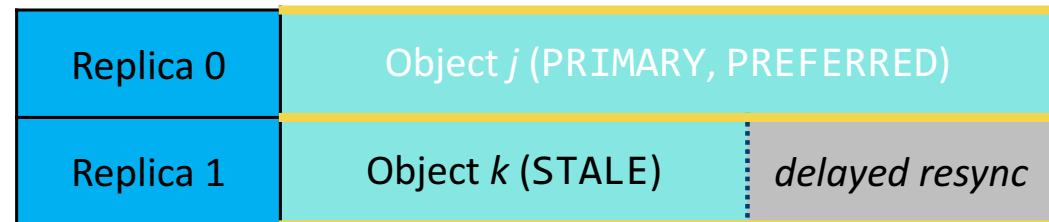
# Persistent Client Cache (PCC) (LU-10092 DDN, WC 2.13+)

*Whamcloud*

► Reduce latency, improve small/unaligned IOPS, reduce network traffic

► PCC integrates Lustre with persistent per-client local cache devices

- Each client has own cache (SSD/NVMe/NVRAM) as a local filesystem (e.g. ext4/ldiskfs)
- No global/visible namespace is provided by PCC, data is local to client only
- Existing files pulled into PCC by HSM copytool per user directive, job script, or policy
- New files created in PCC is *also* created on Lustre MDS

► Kernel uses local file if in cache or normal Lustre IO

- Further file read/write access "directly" to local data
- No data/IOPS/attributes leave client while file in PCC
- File migrated out of PCC via HSM upon remote access

► Separate functionality read vs. write file cache

► Could later integrate with DAX for NVRAM storage
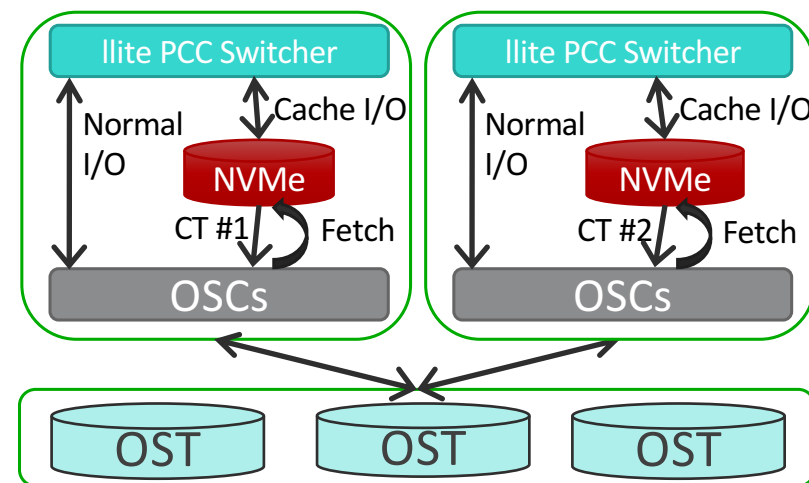
# Client Metadata Writeback Cache (WBC)    (LU-10983 2.14+)

Whamcloud

► Metadata WBC creates new files in RAM in *new directory*
  - Avoid RPC round-trips for each open/create/close
  - Lock directory exclusively, avoid other DLM locking
  - Cache file and directory data only in pagecache until flush
  - Flush tree incrementally to MDT/OST in background batches
► Could prefetch directory contents for existing directory
► Can integrate with PCC to avoid initial MDS create/write
► Early WBC prototype in progress
  - Discussions underway for how to productize it
  - Early results show 10-20x improvement for some workloads
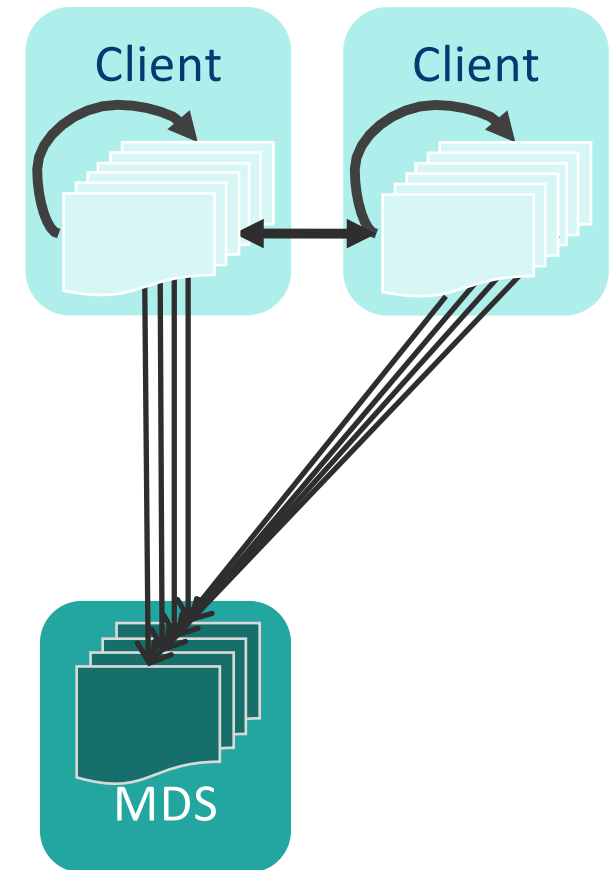
# Client Container Image (CCI)

(2.14+)

**Whamcloud**

▶ Filesystem images have had *ad hoc* with Lustre in the past
- Read-only cache of many small files manually mounted on clients
- Root filesystem images for diskless clients

▶ ***Container Image* is local ldiskfs image** mounted on client
- Holds a whole directory tree stored as a single Lustre file
- **Useful for workloads that are mostly self-contained**

▶ CCI integrates container handling with Lustre
- Mountpoint is registered with Lustre for automatic mount
- Image file read on demand from OST(s) and/or cached in PCC
- **Low I/O overhead, few file extent lock(s), high IOPS per client**
- Access, migrate, replicate image with large read/write to OST(s)

▶ MDS can mount and re-export image files for shared use

▶ CCI can hold archive of whole directory tree for HSM

▶ Unregister/delete image and all files therein with a few ops



Client    Client

MDS    OSS
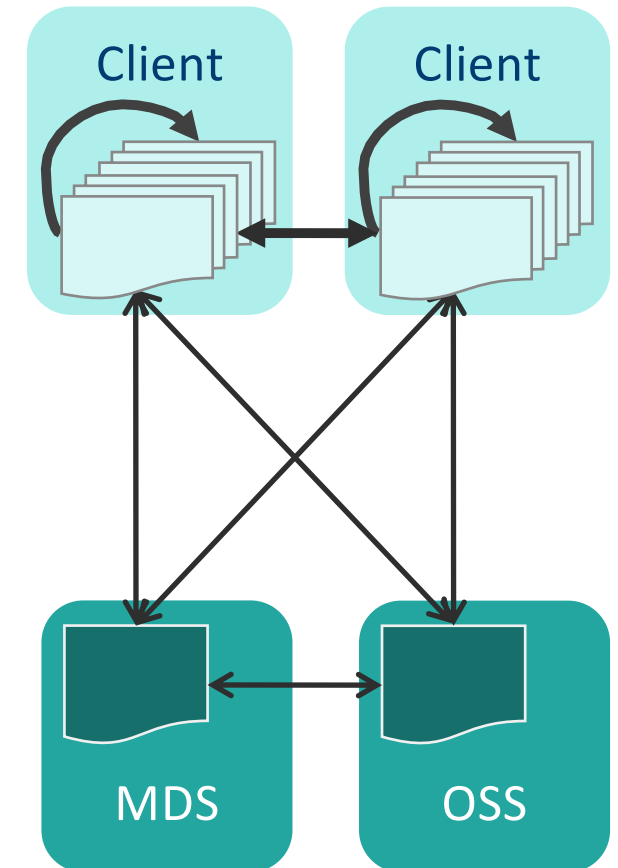
# Kernel Code Cleanup (ORNL, SuSE, Intel, WC, Cray)

**Whamcloud**

► Lustre client removed from kernel 4.17 ☹

- Work continuing on client cleanups *for* upstream at https://github.com/neilbrown/linux

► Lustre 2.12 updates for kernel 4.14/4.15 (LU-10560/ LU-10805)

► Improve kernel time handling (Y2038, jiffies) (LU-9019)

► Ongoing `/proc` -> `/sys` migration and cleanup (LU-8066)

- Handled transparently by `lctl`/`llapi_*` - please use them

► Cleanup of `wait_event`, `cfs_hash_*`, and many more internals

► Some builds/testing with ARM64/Power8 clients (LU-10157 little-endian!)

► Major `ldiskfs` features merged into upstream `ext4`/`e2fsprogs`

- Large xattr (`ea_inode`), directories > 10M entries (`large_dir`)
- `dirdata` feature not yet merged (needs test interface)

> Mmm, Lustre
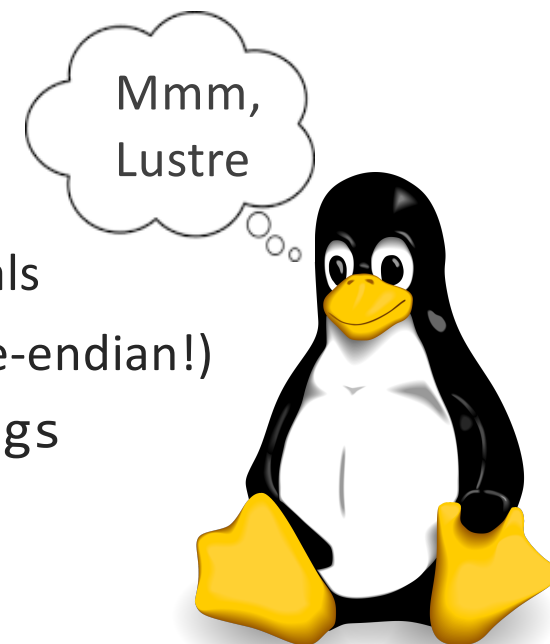
# Tiered Storage with FLR Layouts            (2.12/2.13) ☁️ *Whamcloud*

▶ Integration with job scheduler and workflow for file prestage/drain/archive

▶ Policy engine to manage migration between tiers, rebuild replicas, ChangeLogs

- Policies for pathname, user, extension, age, OST pool, mirror copies, ...

- FLR provides mechanism for safe migration of (potentially in-use) data

- RBH or LiPE are good starting points for this

▶ Needs userspace integration and Lustre hooks

- Integrated burst buffers a natural starting point
- Mirror to flash, mark PREFERRED for read/write
- Resync modified files off flash, release space

▶ Need OST/MDT/pool quotas to manage tiers
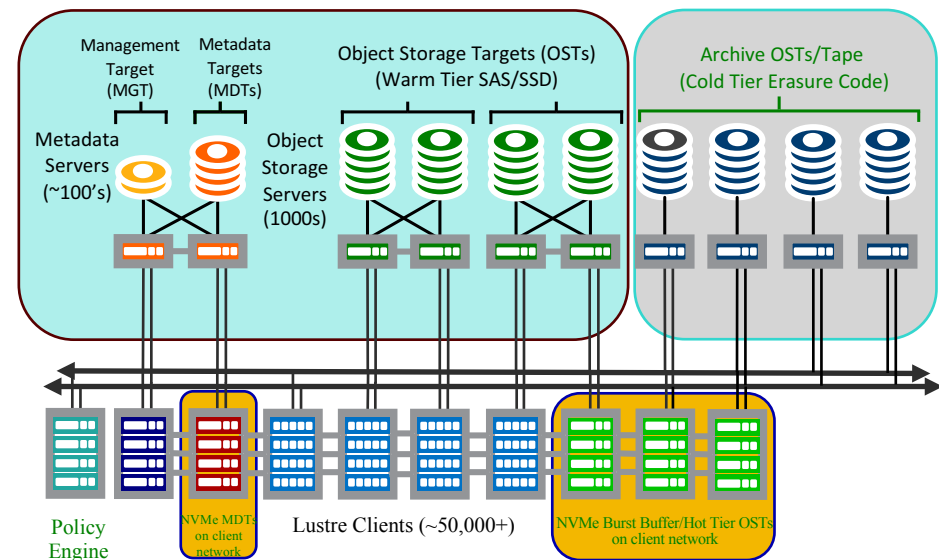
# FLR Erasure Coded Files      (LU-10911 WC 2.13+)

*Whamcloud*

► Erasure coding adds redundancy without 2x/3x overhead

► Add erasure coding to new/old striped files *after* write done

- Use delayed/immediate mirroring for files being actively modified

► For striped files - add N parity per M data stripes (e.g. 16d+3p)

- Parity declustering avoids IO bottlenecks, CPU overhead of too many parities
  - e.g. split 128-stripe file into 8x (16 data + 3 parity) with 24 parity stripes

| dat0 | dat1 | ... | dat15 | par0 | par1 | par2 | dat16 | dat17 | ... | dat31 | par3 | par4 | par5 | ... |
|------|------|-----|-------|------|------|------|-------|-------|-----|-------|------|------|------|-----|
| 0MB  | 1MB  | ... | 15M   | p0.0 | q0.0 | r0.0 | 16M   | 17M   | ... | 31M   | p1.0 | q1.0 | r1.0 | ... |
| 128  | 129  | ... | 143   | p0.1 | q0.1 | r0.1 | 144   | 145   | ... | 159   | p1.1 | q1.1 | r1.1 | ... |
| 256  | 257  | ... | 271   | p0.2 | q0.2 | r0.2 | 272   | 273   | ... | 287   | p1.2 | q1.2 | r1.2 | ... |