



Whamcloud

Lustre 2.14 and Beyond

Andreas Dilger, Whamcloud

Planned Feature Release Highlights

- ▶ **2.14** at feature freeze, with several important additions
 - DNE directory auto-split – improve usability and performance with multiple MDTs
 - OST Quota Pools – manage space on tiered storage targets with OST pools
 - Client-side *data* encryption – persistent encryption of data from client to disk
- ▶ **2.15** feature development already well underway
 - Client-side *directory* encryption – encrypt filenames on disk on MDT
 - File Level Redundancy - Erasure Coding (EC) – efficiently store striped file redundancy
 - LNet IPv6 addressing - allow over 32-bit addresses, more flexible server configuration
- ▶ **2.16** plans continued functional and performance improvements
 - Metadata Writeback Cache (WBC) – low latency file operations in client RAM
 - File Level Redundancy - Immediate Write – write to mirrors directly from client
 - Dynamic inode allocation for `ldiskfs` - improve flexibility for DoM and large OSTs

DNE Usability Improvements (WC)

(2.13+)



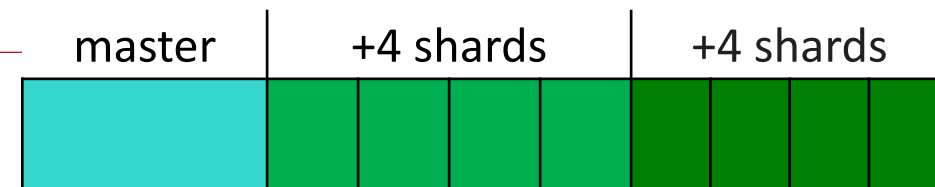
▶ **Space balance new directories** on "best" MDT based on available inodes/space

- Transparently select "best" MDT for normal `mkdir()` based on parent policy ([LU-10784](#))
- Set default policy on parent via `lfs setdirstripe -D -i -1 dir` ([LU-11213](#))
- Most useful for root and top-level user directories

2.13

2.14 ▶ **New crush directory hash type** ([LU-11025](#))

- Minimize number of directory entries migrated by restripe



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

- Create one-stripe directory for low overhead, increase shards/capacity/performance with size
- Add `mdt.*.dir_split_delta=4 shards` if shard over `mdt.*.dir_split_count=50000` entries
- Move fraction of existing **directory entries** to new directory shards (names only, not inodes)
- New directory entries and inodes created directly on new MDTs

2.15 ▶ **Improve MDT usage/space balancing** for new filesystems ([LU-13417](#))

▶ **Select closest network-local MDT(s)** for `mkdir` for tiered/distributed configs ([LU-12909](#))

OST Quota Pools ([LU-11023](#), Cray/HPE)

(2.14+)



▶ Account/limit space for OSTs in a specific pool

- Control usage of small flash OSTs in tiered config

▶ Use existing Lustre quota infrastructure

- OST already tracks space per UID/GID/ProjID
- Pool usage based on sum of current OSTs in pool

▶ Add quota pool limits per UID/GID/ProjID

- No extra accounting on the OSTs
- Only new aggregation/reporting by MDS

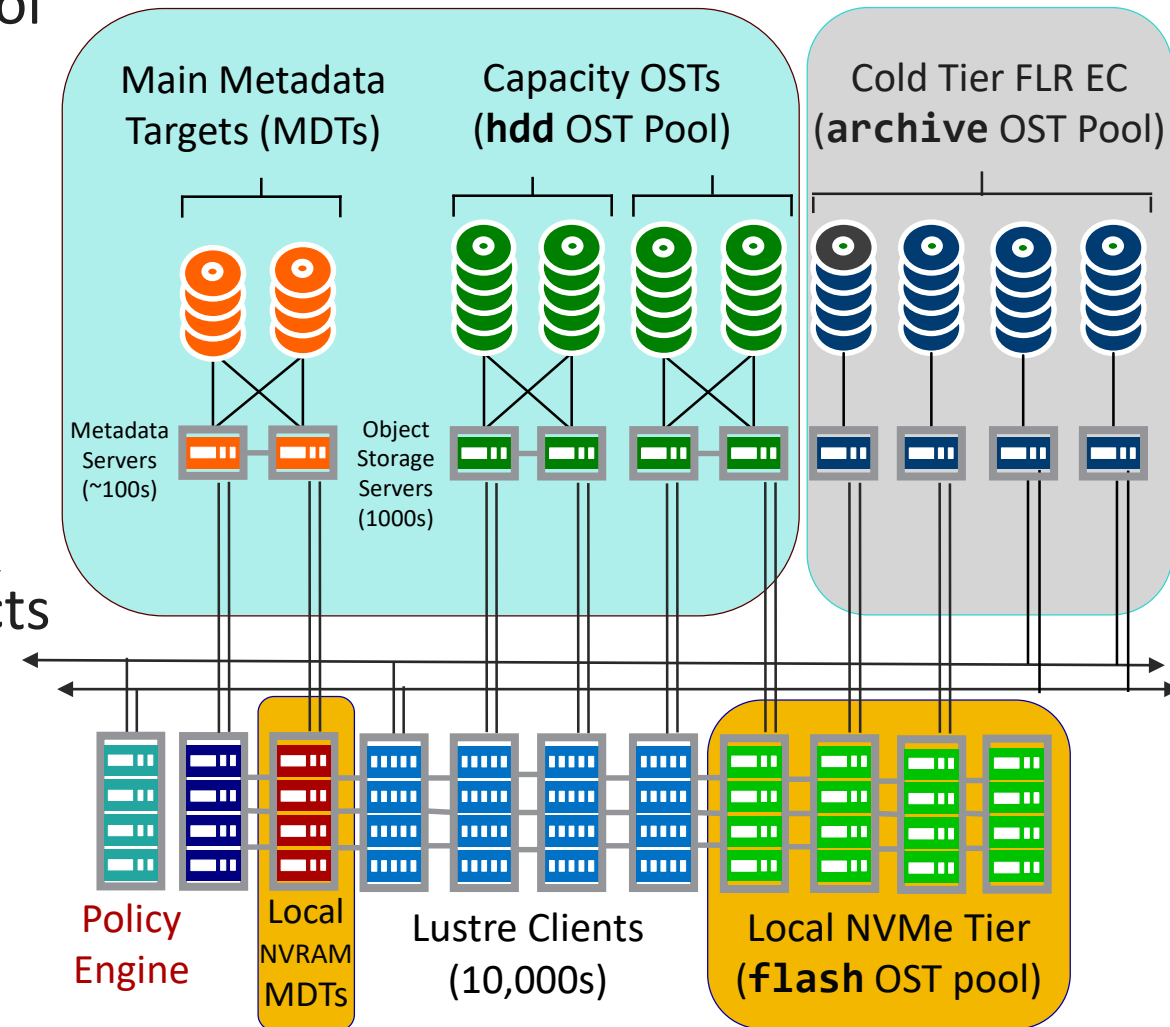
2.14

TBD ▶ Check quota limit when allocating OST objects

- Avoid OSTs with little/no quota available

▶ MDT pools to allow MDT tiering

- Manage/balance DoM MDT space usage
- Handle MDT storage classes (e.g. NVRAM vs. NAND)



Client *Data* Encryption to Disk ([LU-12755](#), WC) (2.14+)



▶ Protect from storage theft or loss, and network or malicious client snooping

▶ **Encryption on Lustre client** down to storage

- Securely store user crypto keys in client kernel keyring
- Data encrypted before sending to servers
- Data decrypted after receiving from servers
- Servers/storage only see encrypted data
- Transparent to backend filesystem/storage (ldiskfs/ZFS)
- Use larger client CPU capacity to encrypt/decrypt data

▶ **Use existing ext4/f2fs fsencrypt library/tools**

- Inventing your own encryption is a fool's errand
- **Per-directory tree** tunable encryption setting/user master key
- **Per-file encryption key**, itself encrypted by user master key
 - Fast and secure deletion of file once per-file key is erased
 - Decrypted data dropped from client cache when user master key removed



2.14

2.15

▶ **Filenames encrypted on client** for MDT directory entries

LNet Improvements

(2.14+)



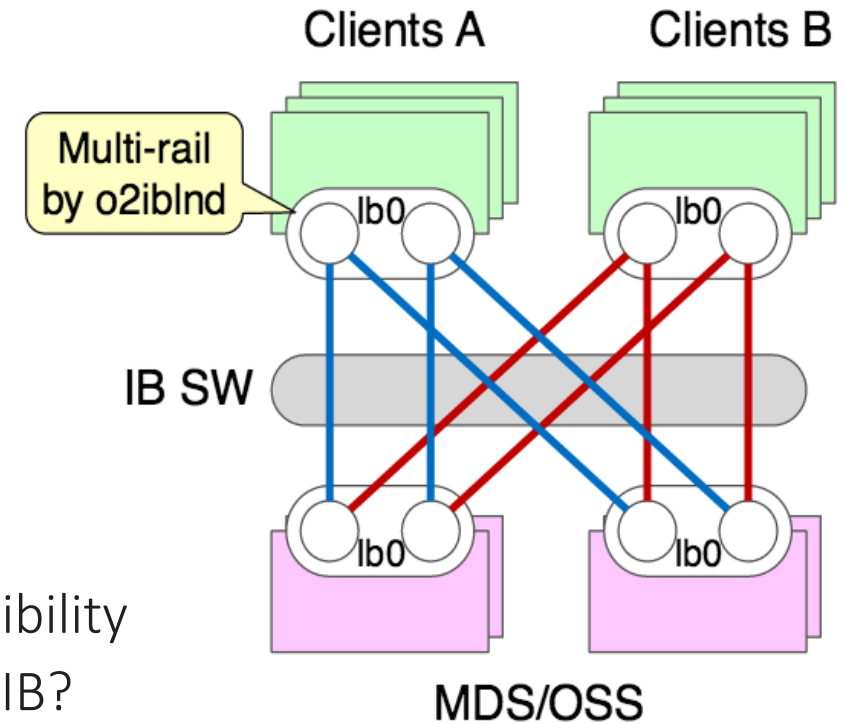
2.14 ► MR Router Health improvements ([LU-12941](#), [LU-13510](#), [LU-13025](#), ..., HPE, WC)

2.14 ► User Defined Selection Policy ([LU-9121](#), WC)

- Fine grained control of interface selection
 - TCP vs. IB networks, primary vs. backup, local vs. remote
- Optimize internal RAM/CPU/PCI data transfers
- Useful for large NUMA machines with multi-rail

2.15 ► IPv6 Node Addressing ([LU-10391](#), WC, SuSE)

- Allow NIDs larger than 32+32bits in TCP and IB
 - New sockv6lnd, o6iblnd nettypes for protocol compatibility
 - Allow direct IB GUID addressing, to avoid need for IPoIB?
- Use Imperative Recovery log for MDT/OST addressing on clients ([LU-10360](#), WC)
 - Allow OSTs and MDTs to mount on any server, no *requirement* for failover addresses
- NIDs no longer needed in Lustre config ([LU-13306](#), WC)



Data-on-MDT (DoM) Improvements (WC)

(2.14+)



▶ Shrink DoM component size if MDT free space running out too quickly ([LU-12785](#))

2.14 ▶ Early lock cancel for DoM, +28% IOPS on IO500 mdtest-easy-delete ([LU-12321](#))

2.15 ▶ Optimized DoM->OST component removal ([LU-13612](#))

- Avoid whole-file copy when freeing space from MDT

▶ Merge data write with MDS_CLOSE RPC ([LU-11428](#))

- Reduce RPC count by half for mdtest-hard-write

▶ Cross-file data prefetch via statahead ([LU-10280](#))



2.16 ▶ Store very small files (< 600 bytes) directly in Idiskfs inode (**inline_data**, [LU-5603](#))

▶ Dynamic inode allocation for Idiskfs ([LU-12099](#))

- Simplify initial MDT setup, less need for up-front decision about bytes-per-inode ratio
- Also improves flexibility for OSTs as they become larger

Miscellaneous Improvements

(2.14+)



▶ **Upstream kernel client cleanups** still in active development/merge (ORNL, SuSE, HPE)

- Lustre master <-> kernel client staying nearly in sync, hundreds! of patches landed
- Need IPv6 support in LNet before submitting upstream, per upstream request

▶ **Disable server page cache for large IOs** to avoid kernel overhead ([LU-12071](#), WC)

▶ **Stateless client-on-server mount** for data migration tasks ([LU-12722](#), WC)

▶ **statx()** for lightweight attribute fetching, file creation time ([LU-10934](#), WC)

2.14 ▶ **fa1locate()** for file preallocation (ldiskfs only) ([LU-3606](#), WC, End User)

2.15 ▶ **External HSM Coordinator** to simplify HSM optimization/improvement ([LU-10968](#), HPE)

▶ **fa1locate()** for ZFS, PUNCH_HOLE, ZERO_RANGE ([LU-11234](#), WC)

▶ **Pool Selection Policy** by filename extension, NID, UID/GID ([LU-11234](#), WC)

▶ **Dynamic openlock** on client for repeated opens ([LU-10948](#), WC)

▶ **O_TMPFILE** for creating temporary files outside namespace ([LU-9512](#))

Improved Client Efficiency for AI/ML

(2.14+)



- ▶ **Improve parallel client readahead** ([LU-12043](#), [LU-13386](#), [LU-13412](#), WC)
 - Parallel readahead for single user thread (e.g. "dd") from 1.9GB/s -> 4.0GB/s
 - ▶ **Improved strided readahead** (IO-500 ior-hard-read) ([LU-12518](#), [LU-12644](#), WC)
 - Detect and handle page-unaligned strided reads
 - Allow readahead to continue for slightly "imprecise" strides
 - ▶ **Asynchronous Direct IO (AIO/DIO)**, [LU-4198](#), WC, Uber)
 - Improved 4KB random IO via **libaio** (write 100k->266k IOPS; read 80k->610k IOPS)
 - ▶ **Bind service threads** to specific CPT cores ([LU-13258](#), WC, ORNL)
 - Readahead, pinger, export cleanup limited to run on specified cores
 - Avoid jitter in scheduling of other threads on node
-
- 2.14
- 2.15 ▶ **Optimized GPU data path** with RDMA
- ▶ **Local client mount on OST/MDT** for data mover/resync ([LU-10191](#), WC)
 - Beginning of optimization for server-local IO path to avoid RPC + data copy

Persistent Client Cache (PCC) ([LU-10092](#), WC) (2.13+)



▶ **Reduce latency**, improve small/unaligned IOPS, reduce network traffic

▶ PCC integrates Lustre with a persistent per-client **local cache storage**

- A local filesystem (e.g. ext4 or 1diskfs) is created on client device (SSD/NVMe/NVRAM)
- New files created in PCC are *also* created on Lustre MDS

2.13

2.15

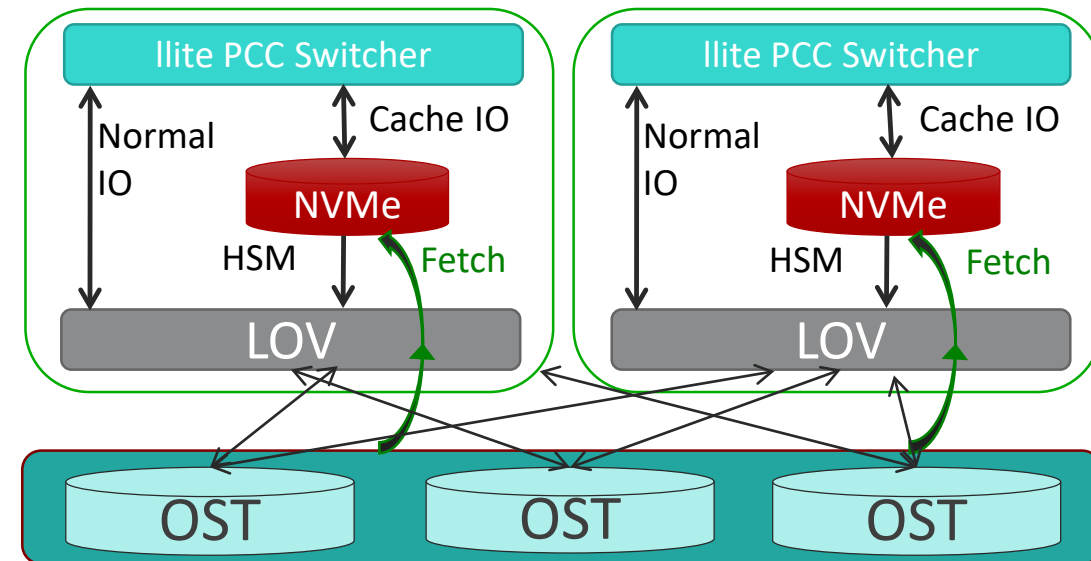
▶ Integrate PCC, HSM, FLR to manage layouts ([LU-13637](#))

- Simplify code, improve functionality

▶ Add **shared read vs. exclusive write** cache

▶ Integrate with DAX for NVRAM cache device

- Use dedicated NVRAM filesystem (e.g. NOVA) for speed



Ongoing `ldiskfs` Improvements

(2.14+)



- 2.14 ▶ **Fix huge OSTs mounting**, toward 1PiB `ldiskfs` OST ([LU-12988](#), [LU-13241](#), WC, HPE)

- 2.15 ▶ Existing features available that could be used by Lustre on `ldiskfs`
 - Efficient large block allocation for large OSTs (**bigalloc**, [LU-12967](#))
 - Files/dirs <600 bytes inside MDT inode, 3.7KB in 4KB inode (**inline_data**, [LU-5603](#))
 - Metadata integrity checksums persistently stored on disk (**metadata_csum**, [LU-13650](#))
- ▶ **Directory shrink** as files are deleted from old directories ([LU-12051](#))
 - Allow dynamic OST object directory allocation to improve performance ([LU-12988](#))
- ▶ **Merge `ldiskfs` `dirdata` feature to upstream `ext4/e2fsprogs`**

- 2.16 ▶ **Integrated `ldiskfs` filesystem snapshots** for MDTs and OSTs ([LU-13660](#), WC)
- ▶ **Dynamic `ext4` inode allocation** for MDTs and OSTs ([LU-12099](#))
 - Design discussions underway with upstream `ext4` maintainers
 - OSTs could allocate new inode tables when not enough free inodes for small files
 - MDTs could release unused inode tables for DoM when many free inodes

File Level Redundancy (FLR) Enhancements (WC) (2.15+)



▶ Erasure coding adds redundancy without 2x/3x mirror overhead ([LU-10911](#))

- Delayed erasure coding to new/existing striped files *after* normal write
- For striped files - add N parity per M data *stripes* (e.g. 16d+3p)
- Leverage CPU-optimized EC code ([Intel ISA-L](#)) for best performance
- Fixed RAID-4 parity layout *per file*, **declustered Parity** across files to avoid OST bottlenecks

2.15

2.15 ▶ HSM in composite layout xattr like FLR mirror ([LU-10606](#), WC)

- Allow multiple archives per file (POSIX, S3, tape, ...)
- Allow partial HSM file copy/restore to/from archive

2.16 ▶ Immediate file write mirroring ([LU-13643](#))

- Client writes both copies of mirror directly
 - Reduces available bandwidth on client
- Mirrors kept in sync unless client write fails
- Delayed resync if mirror goes stale, like before

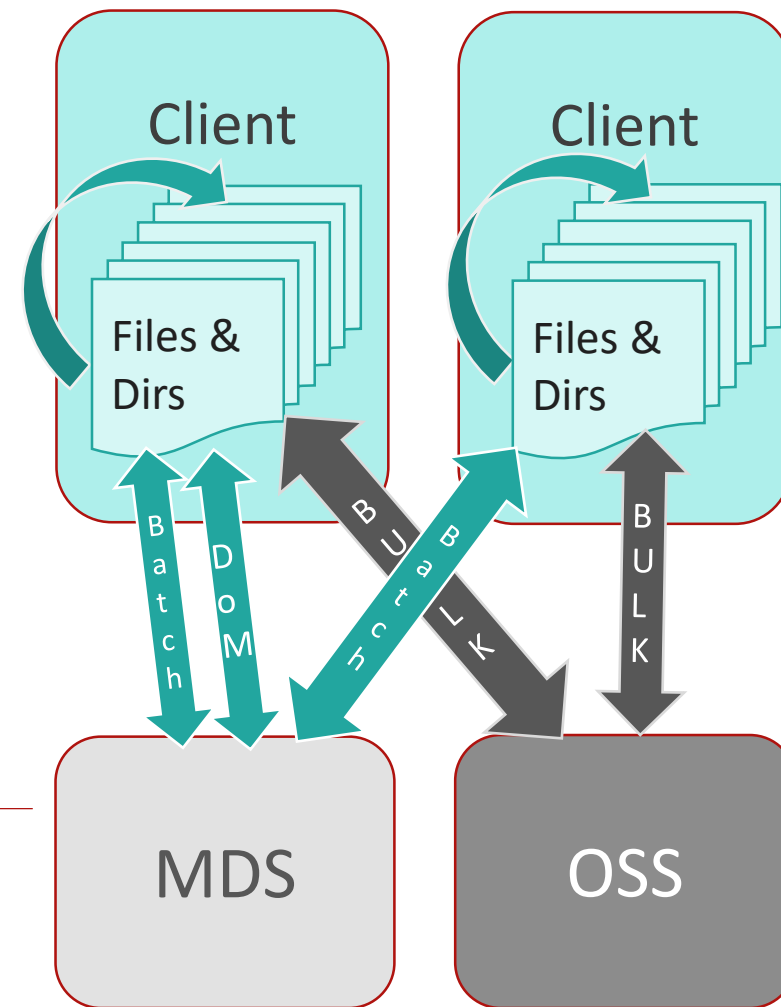
Replica 0	Flash Object <i>j</i> (PRIMARY, PREFERRED)
Replica 1	Flash Object <i>k</i> (PRIMARY, PREFERRED)
Replica 2	HSM S3 Archive <i>delayed sync</i>

Metadata Writeback Cache (WBC) ([LU-10983](#), WC) (2.16+)



- ▶ Create new dirs/files in client RAM without RPCs
 - Lock new directory exclusively at mkdir time
 - Cache new files/dirs/data in RAM until cache flush or remote access
- ▶ No RPC round-trips for file modifications in new directory
- ▶ Files globally visible on MDS flush, normal use afterward
 - Flush top-level entries, exclusively lock new subdirs, unlock parent
 - Repeat as needed for subdirectories being accessed remotely
 - Flush rest of tree in background to MDS/OSS by age or size limits
- ▶ WBC prototype developed to test concept
 - 10-20x *single-client* speedup in early testing (untar, make, ...)
- ▶ Productization of WBC code well underway
 - Complexity handling partially-cached directories, space usage

- 2.17 ▶ Aggregate operations to server to improve performance
 - Batch operations in one RCP to reduce network traffic/handling
 - Batch operations to disk filesystem to reduce disk IOPS?





Whamcloud

Thank You!
Questions?