



Whamcloud

Lustre 2.16 and Beyond

Andreas Dilger

Lustre Principal Architect



ddn

Planned Feature Release Highlights

- ▶ **2.16** currently landing new feature patches
 - **LNet IPv6 addressing** – allow 160-bit NIDs, more flexible server configuration (SuSE, ORNL)
 - **Optimized Directory Traversal (WBC1)** – cross-directory statahead (WC)
- ▶ **2.17** has some major features already lined up
 - **Client-side data compression** – use client to reduce network and storage (WC, UHamburg)
 - **Metadata Writeback Cache (WBC2)** – low latency file operations in client RAM (WC)
- ▶ **2.18** feature proposals in early discussion stages
 - **File Level Redundancy - Erasure Coding (EC)** – efficiently store file redundancy (ORNL?)
 - **Lustre Metadata Redundancy (LMR1)** – MDT0000 service redundancy

LNet Improvements

- ▶ Multiple TCP sockets for 100GigE+ performance ([LU-12815](#), WC)
 - Add `conns_per_peer=N` for `sock1nd` (4.1GB/s->**9.5GB/s** on 100GbE)
 - Auto-configure based on interface speed (e.g. 10Gbps=>N=2, 100Gbps=>N=4, ...)

- ▶ LNet Network Selection Policy (UDSP) ([LU-9121](#), WC)

- Allow policies for local/remote interface prioritization by NID
 - e.g. primary IB with TCP backup, select "best" router NID for client/server

2.15

2.16 ▶ IPv6 NID support ([LU-10391](#), SuSE, ORNL)

- Variable-sized NIDs (8-bit type, 8-bit size, 16-bit network, 128-bit+ address)
- Interoperable with existing current LNDs whenever possible

- ▶ Improved network discovery/peer health (HPE, WC)

- ▶ Simplified/dynamic server node addressing ([LU-14668](#), WC)

- Detect added/changed server interfaces automatically ([LU-10360](#))
- Reduce (and eventually eliminate) static NIDs in Lustre config logs
- Simplified handling for IPv6 NIDs by clients



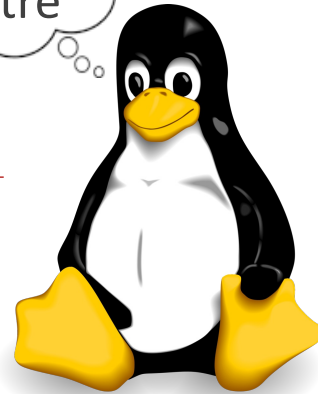
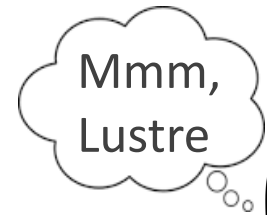
Client Improvements

▶ **GPU Direct RDMA** - directly into GPU, bypass CPU ([LU-14798](#), WC, NVIDIA, HPE)

- A100 2x200Gb IB **36GB/s** write, **39GB/s** read, **174GB/s** with 8x200Gb IB

▶ **Parallel large DIO** optimization ([LU-13798](#), [LU-13799](#), HPE, WC)

- Improve single-thread `read()/write()` (1.5GB/s->**15.8GB/s!**)
- Particular benefits for AIO/DIO and `io_uring` in client kernels 5.1 and later



2.15 ▶ Client-side filename encryption, migration/mirror ([LU-13717](#), [LU-14677](#), WC)

2.16 ▶ `o2ib1nd` cleanups for in-kernel OFED ([LU-8874](#), ORNL)

▶ Encrypted file backup/restore/HSM ([LU-14677](#), WC)

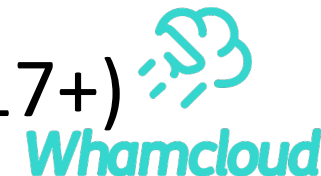
▶ Buffered/DIO/mmap performance/efficiency improvements ([LU-13805](#), WC)

▶ Ongoing code style/structure cleanup for upstream submission (ORNL)

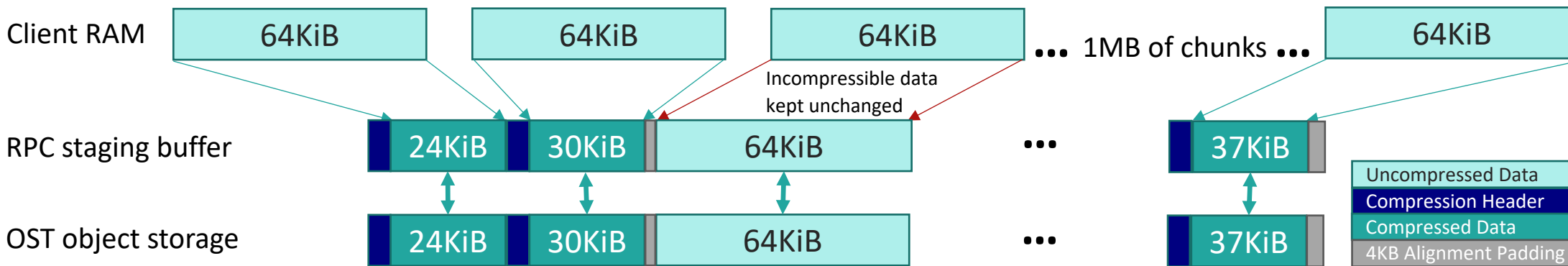
▶ Ongoing updates for newer kernels, usually within 1-2 minor releases (ORNL, SuSE)

2.17 ▶ Client-side Data Compression ([LU-10026](#), WC, UHamburg, Intel)

Client Data Compression ([LU-10026](#)) (WC, UHamburg 2.17+)



- ▶ Reduce space usage for all-flash OSTs to reduce storage cost
- ▶ Compress (lzo, lz4, gzip, ...) full RPC on client in chunks (64KiB-1MiB+)
 - Algorithm, level, chunk size can be selected per directory or file component (PFL, FLR)
 - Larger chunks improve compression, but higher read-modify-write overhead
 - Keep "uncompressed" chunks for incompressible data/file (.gz, .jpg, .mpg, ...)




- ▶ Client writes/reads whole chunk(s), (de-)compresses to/from RPC staging buffer
 - Independent/parallel compression for each chunk on separate core to reach GB/s speeds
- ▶ Optional write to uncompressed file mirror for random IO pattern
- ▶ Optional data (re-)compression during mirror/migrate (via transfer agent)

Backend OSD Improvements

- ▶ `fa1locate()` and `FALLOCATE_FL_PUNCH_HOLE` for ZFS ([LU-14157](#), AEON)
- ▶ Improved `ldiskfs` `mballoc` efficiency for large filesystems ([LU-14438](#), Google, WC, HPE)
 - O(1) lookup of power-of-two free blocks, bias allocations to start of HDDs
- ▶ OST object directory scalability for large OSTs ([LU-11912](#), WC)
 - Large OSTs (500-1000TB) have billions of objects, only 32 subdirs per MDT => 10M+ obj/dir!
 - Wider directory fanout not better, object create/remove accesses all dirs randomly
 - More OST FID Sequences (once per 32M vs. 4B objs), groups objs by age to improve efficiency
- 2.16
- 2.17 ▶ Parallel `e2fsck` for pass2/3 (directory entries, name linkage) ([LU-14679](#), WC)
 - Now slowest part of `e2fsck` (was 7% of total time, now **70%** after pass1/pass5 speedups)

MDT DNE Improvements

(WC 2.15+) 
Whamcloud

▶ **DNE MDT Space Balance** - load balancing with normal `mkdir` ([LU-13417](#), [LU-13440](#))

- Round-robin/balanced subdirs, prefer to stay on parent, limited layout inheritance depth
- Keep MDTs within free inodes/space (`mdt.*.mdt_qos_threshold_rr=5%`)

▶ **Single-dir migration** - "`lfs migrate -m -d <dir>`" ([LU-14975](#))

- Move only one directory level, instead recursing down full subdirectory tree

▶ **Balanced migration** - "`lfs migrate -m -1 <dir>`" ([LU-13076](#))

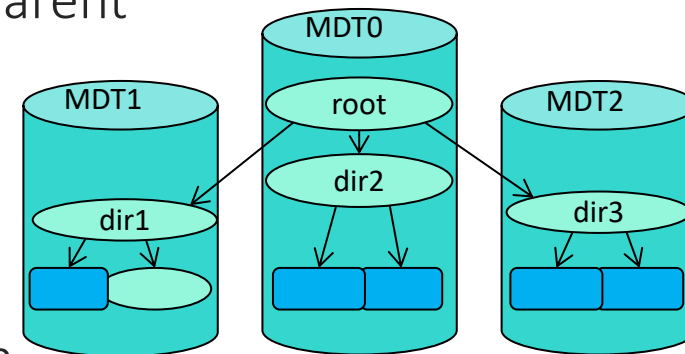
2.15 • Auto-select less-full MDTs for each directory, keep inodes local to parent

2.16 ▶ **DNE inode migration improvements** ([LU-14719](#))

- Pre-check target space, stop on error, improved CRUSH2 hash

▶ **DNE locking, remote RPC optimization** ([LU-15528](#))

- Better distributed transaction performance, reduce lock contention



Batched Cross-Directory Statahead (WBC1)

(WC 2.16)



▶ **Batched RPCs** for multi-update operations ([LU-13045](#))

- Allow multiple getattrs/updates packed into a single MDS RPC
- More efficient network and server-side request handling

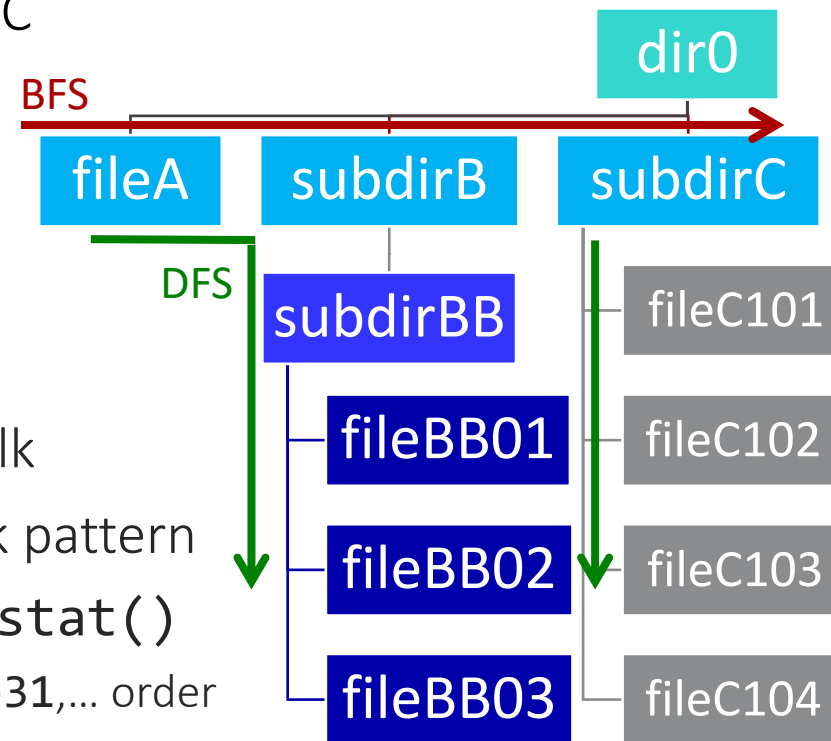
▶ **Batched statahead** for `ls -l`, `find`, etc. ([LU-14139](#))

- Aggregate getattr RPCs for existing statahead mechanism

▶ **Cross-Directory statahead** pattern matching ([LU-14380](#))

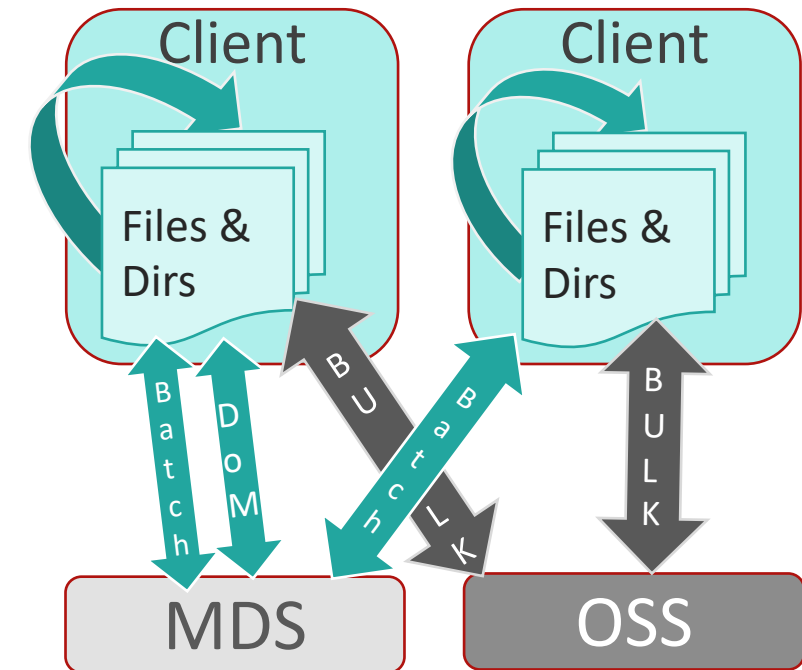
- Detect breadth-first (**BFS**) depth-first (**DFS**) directory tree walk
- Direct statahead to next file/subdirectory based on tree walk pattern
- Detect strided pattern for alphanumeric ordered traversal + `stat()`
 - e.g. `file00001,file001001,file002001...` or `file1,file17,file31,...` order

▶ IO500 `mdtest-{easy/hard}-stat` performance improved **77%/95%**



Metadata Writeback Cache (WBC2) ([LU-10983](#))

- ▶ 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
- ▶ Batch RPC for efficient directory fetch and cache flush
- ▶ **Files globally visible on remote client access**
 - 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
- ▶ Productization of WBC code well underway
 - Some complexity handling partially-cached directories
 - Need to integrate space usage with quota/grant
- ▶ WBC in pre-existing directories in later release
 - Read all directory entries before create to avoid duplicates



Lustre Metadata Redundancy ([LU-12310](#))

(2.18+)



In early discussion and architecture stages

▶ LMR1a: Replicate services to other MDTs

- Mirror FLDB, Quota, flock() across MDTs

▶ LMR1b: DNE transaction performance

- Transactions currently have excessive ordering/sync
- Need to improve if all mkdir are mirrored transactions
- Improves all DNE operation performance

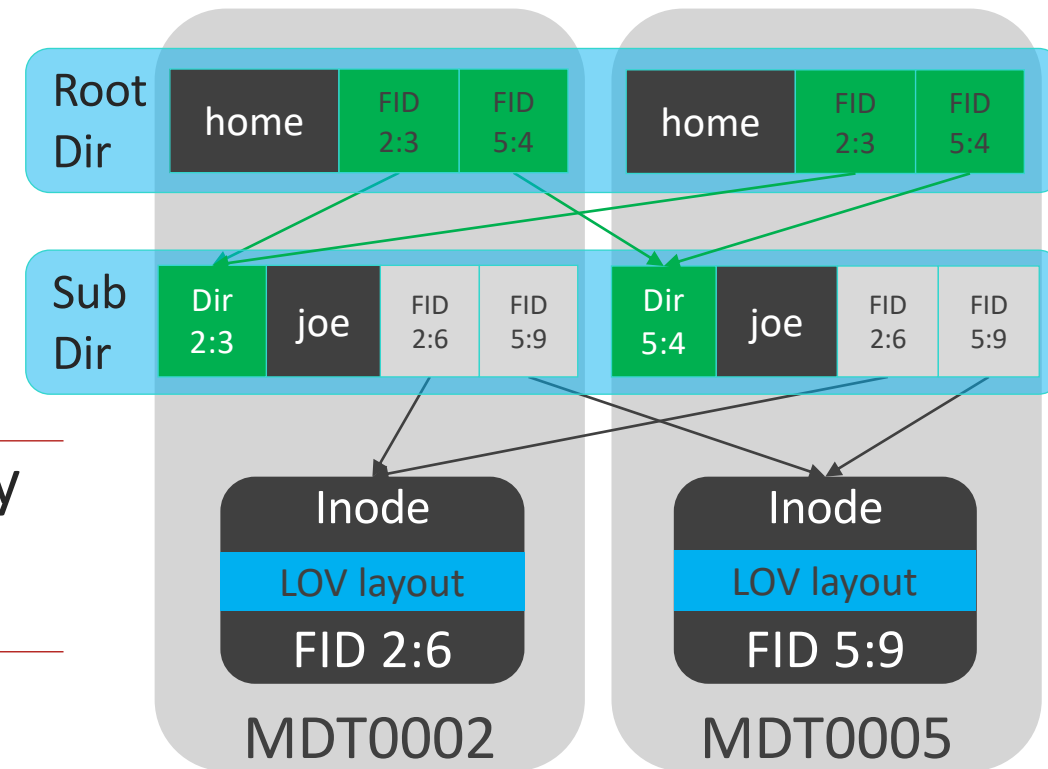
2.17

2.18 ▶ LMR1c: Replicate top-level dirs for availability

- ROOT/ dir (rarely changed) mirrored over MDTs
- No replication for regular file inodes initially

2.19+ ▶ LMR2/3 phases needed for full redundancy

- Full tree directory replication, file inode replication
- Configurable replication setting per directory/tree
- Recovery, LFSCK, rebuild replicated directories if full MDT loss





Whamcloud

Thank You!
Questions?