Whamcloud

Lustre 2.16 and Beyond (Redux)

Andreas Dilger

Lustre Principal Architect

Planned Feature Release Highlights

Whamcloud

- **2.16** nearly feature complete
  - **LNet IPv6 addressing** – must-have functionality for future deployments (SuSE, ORNL)
  - **Optimized Directory Traversal** (WBC1) – improve efficiency for accessing many files (WC)
- **2.17** has major features already well underway
  - **Client-side data compression** – reduce network and storage usage, costs (WC, UHamburg)
  - **Metadata Writeback Cache** (WBC2) – order of magnitude better metadata speed (WC)
- **2.18** feature proposals in early stages
  - **File Level Redundancy - Erasure Coding** (FLR-EC) – reduce cost, improve availability (ORNL)
  - **Lustre Metadata Redundancy** (LMR1) – initial stages of development for DNE mirroring
  - **Client Container Image** (CCI) – improved handling of aggregations of many small files

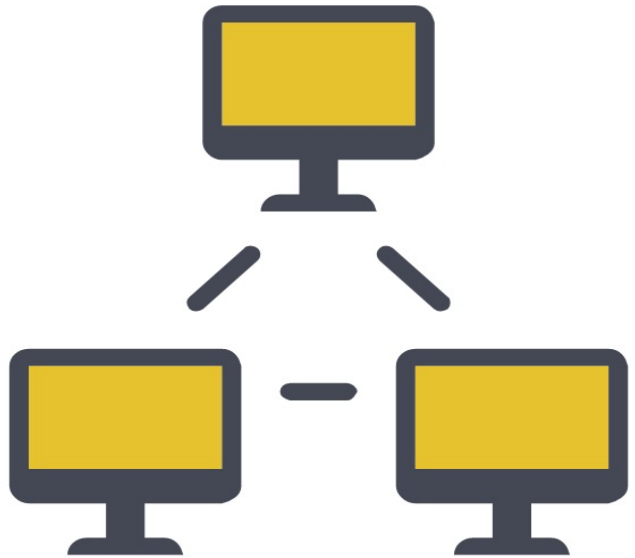
LNet Improvements

Whamcloud

Demand for IPv6 in new deployments as IPv4 is exhausted

- Relatively few external-facing Lustre systems means 10.x.y.z is still viable for most systems

- IPv6 large NID support (LU-10391 SuSE, ORNL)
  - Variable-sized NIDs (8-bit LND type, 8-bit address size, 16-bit network number, 16-byte+ address)
  - Interoperable with existing current LNDs whenever possible
  - Enhancements to LNet/`socklnd` for large NIDs mostly finished
  - Work ongoing to handle large NIDs in Lustre code
    - Mount, config logs, Imperative Recovery, Nodemaps, root squash, etc.
- Improved network discovery/peer health (HPE, WC)
- Simplified/dynamic server node addressing (LU-14668 WC)
  - Detect added/changed server interfaces automatically (LU-10360)
  - Reduce or eliminate static NIDs in Lustre config logs
  - Simplified handling for IPv6 NIDs by clients



Client-Side Usability and Performance Improvements

Whamcloud

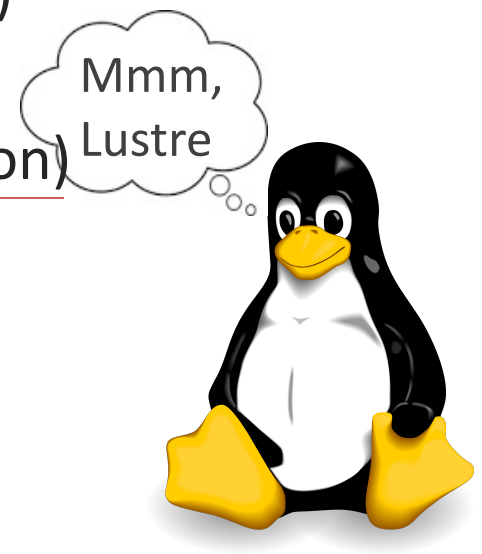
Ongoing ease-of-use and performance improvements for users and admins

- Parallel file/directory rename within a directory (LU-12125 WC)

2.15
- `lfs find -printf` formatted output of specific fields (LU-10378 ORNL)

2.16
- `lfs migrate` bandwidth limit, progress updates (LU-13482 Amazon)
- Ongoing code updates/cleanup for newer kernels (ORNL, HPE, SuSE)
- Remove 8192-device limit, for multiple large mounts (LU-8802 Amazon)

2.17
- Unaligned Direct IO (LU-13805 WC)
- Buffered/DIO performance/efficiency improvements (LU-14950 WC)
- Client-side Data Compression (LU-10026 WC, UHamburg, Intel)
- Client-side performance stats for targets (LU-7880 WC)
- Erasure Coded FLR files (LU-10911 ORNL)



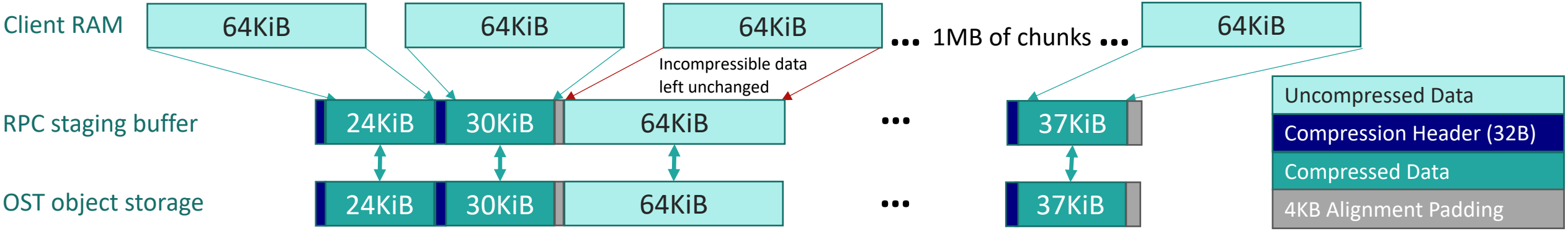
Client-Side Data Compression (WC, UHamburg 2.17+)

Whamcloud

Increased capacity and lower cost for all-flash OSTs

- Parallel compression of RPCs on client cores for GB/s speeds, **no server CPU overhead!**

- (De-)Compress (`lzo`, `lz4`, `gzip`,…) RPC on client in chunks (64KiB-1MiB+) (LU-10026)
  - **Per directory or file component** selection of algorithm, level, chunk size (PFL, FLR)
  - Keep "uncompressed" chunks as-is for incompressible data/file (`.gz`, `.jpg`, `.mpg`, …)



- Client writes/reads whole chunk(s), (de-)compresses to/from RPC staging buffer
  - Larger chunks improve compression, but higher decompress/read-modify-write overhead
- Optional write uncompressed to one FLR mirror for random IO pattern
- Optional data (re-)compression during mirror/migrate to slow tier (via data mover)

Server-side Capacity and Efficiency Improvements

Whamcloud

Ongoing performance and capacity scaling for next-gen hardware and systems

- OST object directory scalability for multi-PB OSTs
  - Reduced transaction size for many-striped files/dirs (LU-14918 WC)
  - Handling billions of objects on a single OST (LU-11912 WC)
  - Group objects into directories by age to optimize RAM/IOPS
- Read-only mount of OST and MDT devices (LU-15873 WC)
- `lljobstat` utility for easily monitoring "top/bad" jobs (LU-16228 WC)

2.16
  - Add IO size histograms to job_stats output, handle bad job names better

2.17
- Writeback cache for small, lockless, direct writes (`ior-hard-write`, LU-12916 WC)
  - Lower latency, write aggregation, no lock ping-ping
  - Use ext4 delayed allocation mechanism

Ongoing ldiskfs and e2fsprogs Improvements

Whamcloud

- More efficient ldiskfs mballoc for large filesystems (LU-14438 Google, IBM, WC, HPE)
  - Backport improved list-/tree-based group selection from upstream to el8.8
- `mkfs.lustre` to use `sparse_super2` feature for new filesystems (LU-15002 WC)
  - Allows larger group descriptor table for filesystems > 256TiB
  - Avoids `meta_bg` feature that splatters metadata across device (seek++)
- Hybrid ldiskfs LVM storage devices (NVMe+HDD) (LU-16750 WC)
  - Add `IOPS` flag to block groups on flash storage at start of device, use for all metadata
- Persistent `TRIMMED` flag on block groups during `fstrim` (LU-14712 WC)
  - Avoid useless `TRIM` commands on device after reformat and remount

2.16
- Fix `e2fsck` for shared blocks, large dirs/journal (LU-16171, LU-14710, LU-17117 WC)

2.17
- Parallel e2fsck for pass2/3 (directory entries, name linkage) (LU-14679 WC)

Improved Data Security and Containerization

Whamcloud

Growing dataset sizes and uses increases need to isolate users and their data

- Filenames encrypted on client in directory entries (LU-13717 WC)
- Migrate/mirror of encrypted files without key (LU-14667 WC)

2.15
- Encrypted file backup/restore/HSM without key (LU-16374 WC)

2.16
- Read-only mount enforced for nodemap clients (LU-15451 WC)
- Kerberos authentication improvements (LU-16630, LU-16646 WC, NVIDIA)
- Nodemap project quota mapping, squash all files to project (LU-14797 WC)
- Nodemap Role-Based Admin Controls (fscrypt, changelog, chown, quota) (LU-16524 WC)
- Cgroup/memcg memory usage limits for containers/jobs on clients (LU-16671 WC, HPE)

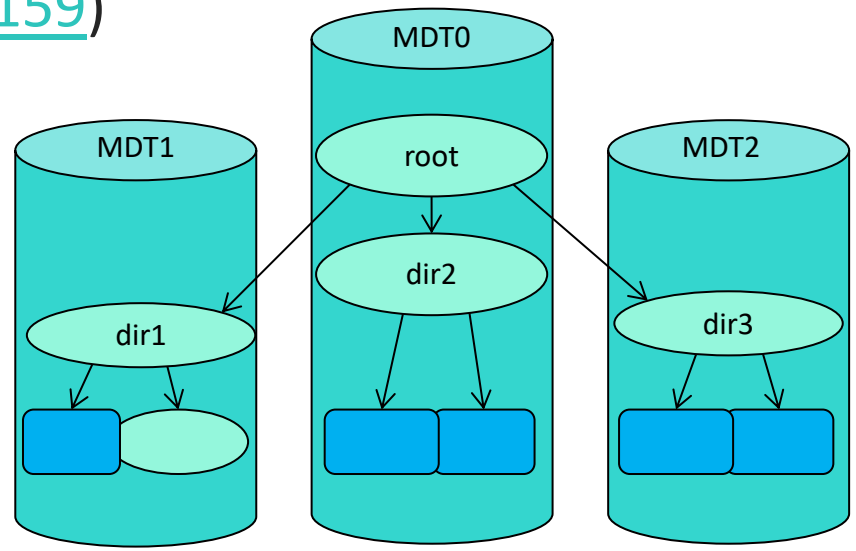
Metadata Server Improvements (WC 2.15+)

Improve usability and ease of DNE metadata horizontal performance/capacity scaling

2.15
- **DNE MDT Space Balance** - load balancing with normal `mkdir` (LU-13417, LU-13440)

2.16
- DNE inode migration improvements (LU-14719, LU-15720)
  - Pre-check target space, stop on error, improved CRUSH2 hash
- More robust DNE MDT llog recovery (LU-16203, LU-16159)
  - Handle errors and inconsistencies in recovery logs better
- Store jobid in "`user.job`" xattr for trace (LU-13031)

2.17
- DNE locking, remote RPC optimization (LU-15528)
  - Distributed transaction performance, reduce lock contention
- Lustre Metadata Robustness/Redundancy (LU-12310)
  - Phase 1 to distribute/mirror MDT0000 services to other MDTs

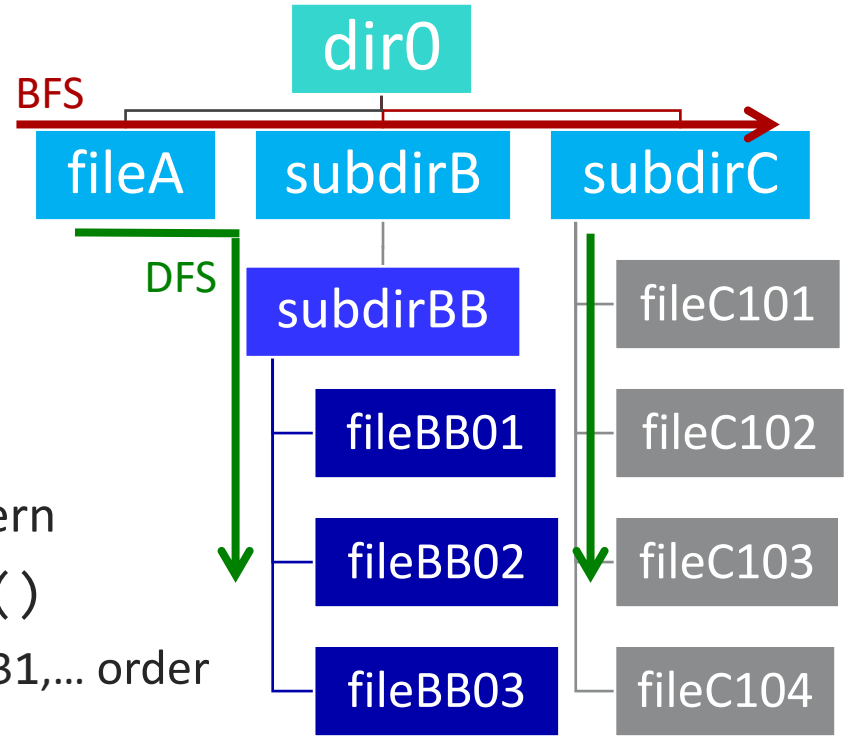


Batched Cross-Directory Statahead (WBC1) (WC 2.16)

Improved access speed and efficiency for large directories/trees

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

- **Batched RPC** infrastructure 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



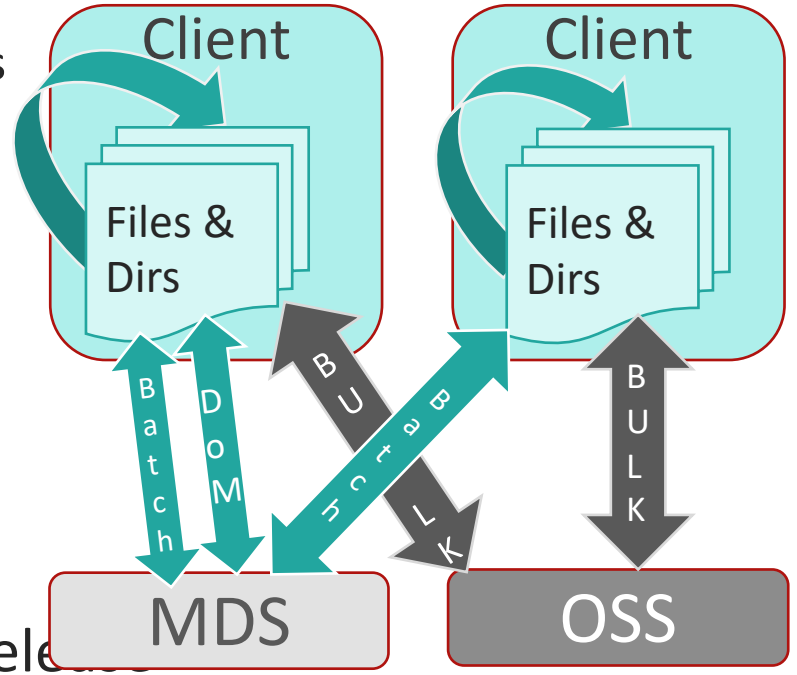
Metadata Writeback Cache (WBC2) (WC 2.17+)

Whamcloud

10-100x speedup for single-client file/dir create-intensive workloads

- Genome extraction/processing, untar/build, data ingest, producer/consumer

- Create new dirs/files **in client RAM without RPCs** (LU-10983)
  - Lock **new** directory exclusively at `mkdir` time
  - Cache new files/dirs/data in RAM until cache flush or remote access
- Batch RPC for efficient directory fetch and cache flush
- **Files globally visible** on remote client access, partial flush
  - Flush top-level entries, exclusively lock new subdirs, unlock parent
  - 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
  - Able to benchmark under intensive multi-client workloads
- WBC for pre-existing directories, PCC integration in later rele
  - Read all directory entries before create to avoid duplicate filenames



Thank You!
Questions?