Lustre 2.15 and Beyond

Andreas Dilger, Lustre Principal Architect

Planned Feature Release Highlights

Whamcloud

- **2.15** feature development complete
  - **LNet Network Selection Policy** (UDSP) – rules for selecting interface (WC)
  - **Client-side *filename* encryption** – persistent encryption filenames from client to disk (WC)
  - **MDT Auto Space Balance** - dynamic MDT selection for mkdir when free space uneven (WC)
- **2.16** plans continued functional and performance improvements
  - **LNet IPv6 addressing** – allow 160-bit NIDs, more flexible server configuration (SuSE)
  - **File Level Redundancy - Erasure Coding (EC)** – efficiently store file redundancy (WC, ORNL)
  - **Metadata Writeback Cache (WBC)** – low latency file operations in client RAM (WC)
- **2.17** feature proposals in early discussion stages
  - **File Level Redundancy - Immediate Write** – mirror directly from client
  - **Client-side data compression** – client CPU reduces net/disk usage

LNet Improvements (2.15/2.16)

Whamcloud

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

- 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
- IPv6 NID support (LU-10391, SuSE)
  - Variable-sized NIDs (8-bit type, 8-bit size, 16-bit network, 128-bit+ address)
  - Interoperable with existing current LNDs whenever possible

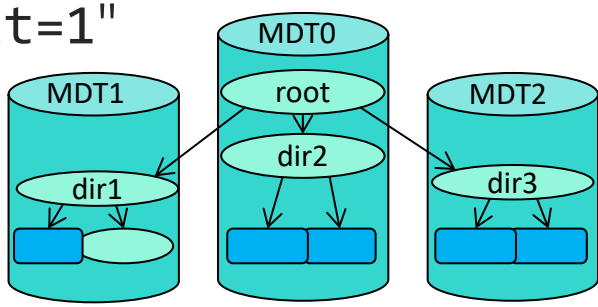
MDT DNE Usability Improvements (WC 2.15+)

Whamcloud

- **Parallel rename within a directory** of files/directories (LU-12125)
  - Avoid filesystem-wide *Big "FID" Lock* (BFL) for renames, only lock parent FID
- **DNE space balance `mkdir()`** - possible for every directory create (LU-13439, LU-13440)
  - `lfs setdirstripe -D -c 1 -i -1 [--max-inherit[-rr] <levels>] <dir>`
  - Round-robin/balanced subdirs, prefer to stay on parent, limited layout inheritance depth
- **Default DNE MDT space balance** - enable for all filesystems (LU-14792)
  - Keep MDTs within free inodes/space (`mdt.*.mdt_qos_threshold_rr=5%`)
  - Prefer remote dirs closer to root directory (larger threshold with depth)

2.15
  - Disable balance "`lfs setdirstripe -D -c 1 -i 0 --max-inherit=1`"

2.16
- Single-dir migration - "`lfs migrate -m -d <dir>`" (LU-14975)
  - Avoid full subdirectory migration to a single MDT
- Balanced migration - "`lfs migrate -m -i -1 <dir>`" (LU-13076)
  - Do not migrate to most-full MDT, keep migrated inodes local to parent directory



Client Encryption – Data at Rest (2.14/2.15, WC)

Whamcloud

- Protect from storage theft/loss, network/user snooping
- Use `fscrypt` library (ext4/f2fs/…) as basis (don't invent it!)
  - **Per directory tree** encryption with (optionally) multiple user key(s)

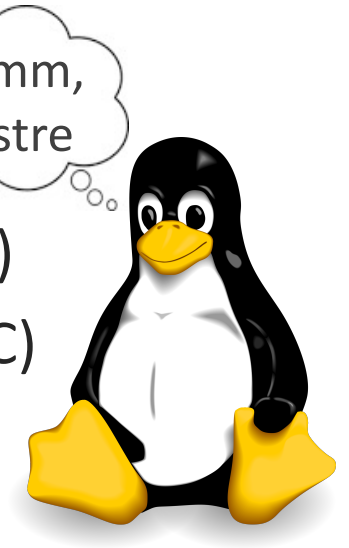
2.14
- **File data encrypted** on client and stored on OST

2.15
- **File names encrypted** in directory entries (LU-13717)
  - Filenames base64-ish encoded, can list directories without key
  - Can unlink files/dirs if needed without key
- Migrate/mirror of encrypted files without key (LU-14667)
  - Avoid pinning files to storage tier if encrypted

2.16
- Performance optimization of encryption (LU-15003)
  - Avoid 30% overhead from bounce page allocation

Client Usability Improvements (ORNL, SuSE, WC)

Whamcloud



- `fallocate()` for regular files on ldiskfs (LU-3606, AEON, WC)
- `SEEK_HOLE/SEEK_DATA` to efficiently handle sparse files (LU-10810, WC)

2.14
- `statfs()` on directory with projid returns project quota limits (LU-9555, WC)

2.15
- `statx()` allows fetching specific inode attributes, lazy file size (LU-10934, WC)
- Automatic open lock caching on client (LU-10948, WC, ORNL)
- Handle large ACLs up to 8k entries (LU-14430, WC)
- `fallocate(FALLOCATE_FL_PUNCH_HOLE)` to free space (LU-14160, AEON)
- `fallocate()` for DoM files (LU-14382, WC)
- `llstat/llobdstat` usability improvements (LU-13705, WC)

2.16
- Ongoing upstream kernel cleanups (ORNL, SUSE)
- `o2iblnd` cleanups for in-kernel OFED (LU-8874)

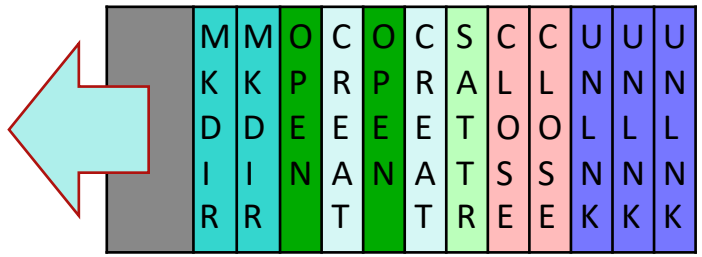
Changelog Usability Improvements (2.15+)

Whamcloud

- **Changelog Named Users** - de/register with name (LU-13055, WC)
  - Avoid unknown/stale Changelog users
    - e.g. a RobinHood user: "`lctl changelog_register --user rbh`" creates "`cl13-rbh`"
- **Changelog Per-user Mask** - only save records that will be used (LU-13338, WC)
  - Union of all registered user masks, use default `changelog_mask` if unspecified
  - Minimize Changelog overhead to the minimum number of records of interest
    - e.g. "`lctl changelog_register --user lamigo --mask=CLOSE,UNLNK`"
- Optimize Changelog Cleanup - free Changelogs by whole file (LU-14688, HPE)

2.15
  - Improve log cancel rate from ~150k/s to ~16M/s (100x speedup)

2.16
- Fix Idle User Deregistration (LU-14699, WC)
  - Avoid filling MDTs for common issue of forgotten Changelog user
  - Automatically delete idle users after time limits/number of records
  - Currently only done when llog is nearly full, be more proactive



Backend OSD Improvements

(2.14+)

Whamcloud

- **Parallel e2fsck** - speedup for pass1 (inodes) and pass5 (bitmaps) (LU-8465, WC)
- ZFS 2.0 support (LU-13946, LLNL, many others)

2.15

  - Persistent L2ARC, Special VDEVs (Metadata Allocation Class), …

2.16

- Parallel e2fsck for pass2 (directory scanning) (LU-14679, WC)
  - Now the slowest part of e2fsck (was 7%, now 70% of total time)
- ZFS 2.1 dRAID VDEVs - declustered parity and hot space (LLNL, HPE, Intel)
- Improved ldiskfs mballoc efficiency for large/full filesystems (LU-14438, Google, WC)
  - O(1) lookup of power-of-two free space, O(logN) lookup of other sizes
- Improved ldiskfs "`-o discard`" efficiency (LU-14712, Kuaishou, WC)
  - Allow real-time TRIM of flash storage to maintain peak performance
- OST object directory scalability (LU-11912, WC)
  - Group objects by age to limit dir size and improve efficiency

Improved Single Client Performance (2.14+)

Whamcloud

- **Improve parallel client readahead** (LU-12043, LU-13386, LU-13412, WC)
  - Parallel/unaligned readahead for single user thread (**dd**, 1.9GB/s->4.0GB/s)
- **Improved mmap readahead** chunk detection (LU-13669, WC)

2.14

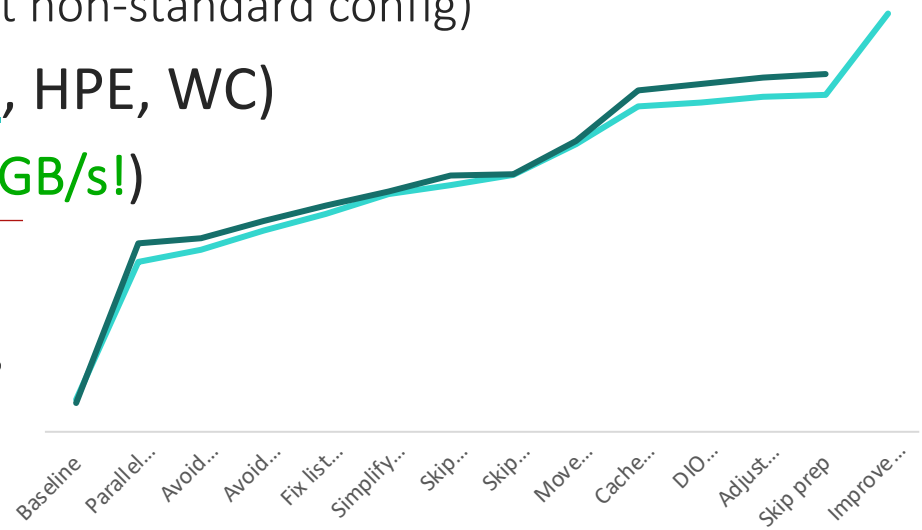
  - Reduced pagefault latency, avg 512usec->52usec, max 37msec->6msec

2.15

- **GPU Direct RDMA** - directly into GPU, bypass CPU (LU-14798, WC, NVIDIA)
  - Significant speedup for IO (A100 2x200Gb IB 25GB/s->36GB/s write, 23GB/s->39GB/s read @ 1MB)
  - Improve 110GB/s->174GB/s with 8x200Gb IB storage links (but non-standard config)
- **Parallel large DIO** optimization (LU-13798, LU-13799, HPE, WC)
  - Improve single-thread `read()`/`write()` (1.5GB/s->15.8GB/s!)

2.16

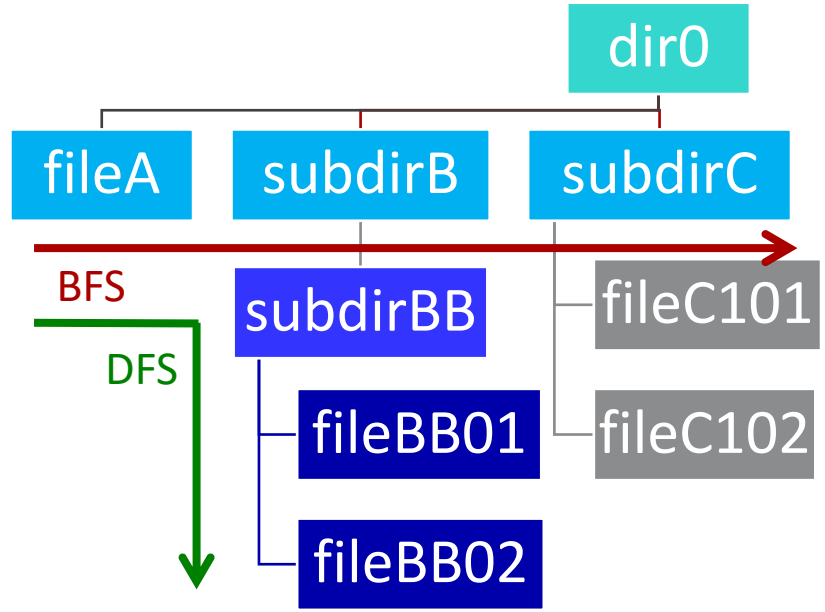
- Improved NID->CPT binding (LU-14676, WC)
  - Spread RPCs from a single client across server CPU cores



Batched Cross-Directory Statahead (WC 2.16)

Whamcloud

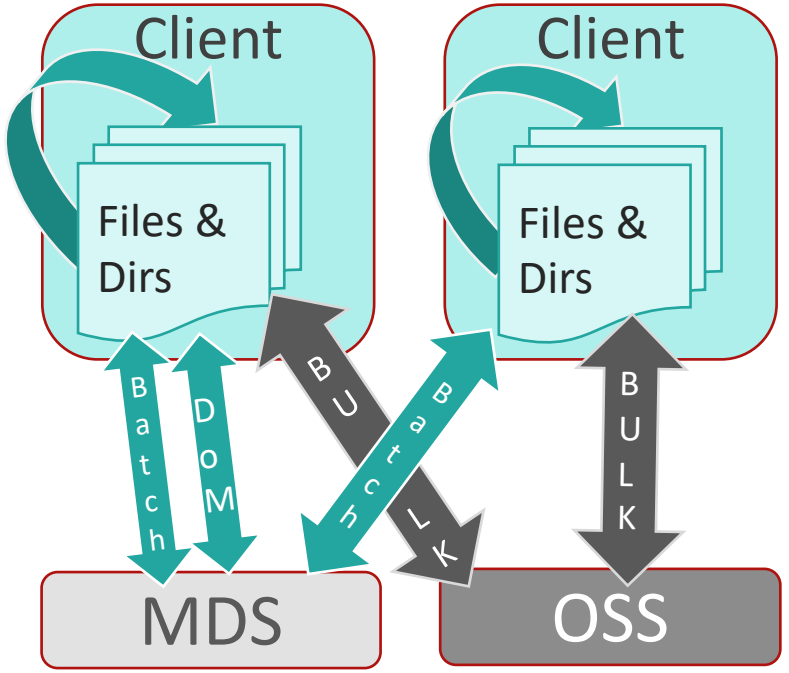
- **Batched RPCs** for multi-update operations (LU-13045)
  - Allow multiple getattrs/updates packed into a single MDS RPC
  - More efficient network and server-side request handling
- **Batched statahead** for `ls -l`, `find`, etc. (LU-14139)
  - Aggregate getattr RPCs for existing statahead mechanism
- **Cross-Directory statahead** pattern matching (LU-14380)
  - Existing statahead only detects `readdir()`-ordered `stat()`
  - Detect pattern for alphanumeric ordered traversal + `stat()`
  - Detect breadth-first (BFS) depth-first (DFS) directory traversal
  - Direct statahead to next file/subdirectory based on pattern



Metadata Writeback Cache (WBC) (LU-10983) (WC 2.16+)

Whamcloud

- 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 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
- WBC prototype developed to test concept
  - 10-20x *single-client* speedup in early testing (`untar`, `make`, …)
- Productization of WBC code well underway
  - Some complexity handling partially-cached directories
  - Need to integrate space usage with quota/grant



Thank You!
Questions?