Whamcloud

Lustre Recent and Upcoming Features

Andreas Dilger

Planned Feature Release Highlights

Whamcloud

- **2.14** nearing release, with several important additions
  - DNE directory auto-split – improve usability and performance with multiple MDTs
  - OST Pool Quotas – 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 *filename* encryption – persistent encryption filenames from client to disk
  - 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

Presentation Overview

Whamcloud

- See my LUG'20 presentation for discussion of other upcoming features
  - https://wiki.lustre.org/images/3/3a/LUG2020-Lustre_2.14_and_Beyond-Dilger.pdf
  - https://youtu.be/jWAp00J66KQ?t=4780
- This talk dives deeper into a few major changes of interest

- Improved Client Performance
- Very large ldiskfs OSTs
- File layout improvements

Improved Client Performance (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
- **Asynchronous Direct IO (DIO/AIO**, LU-4198, WC, Uber)
  - Improved 4KB random IO via **`libaio`** (write 100k->266k IOPS; read 80k->610k IOPS)

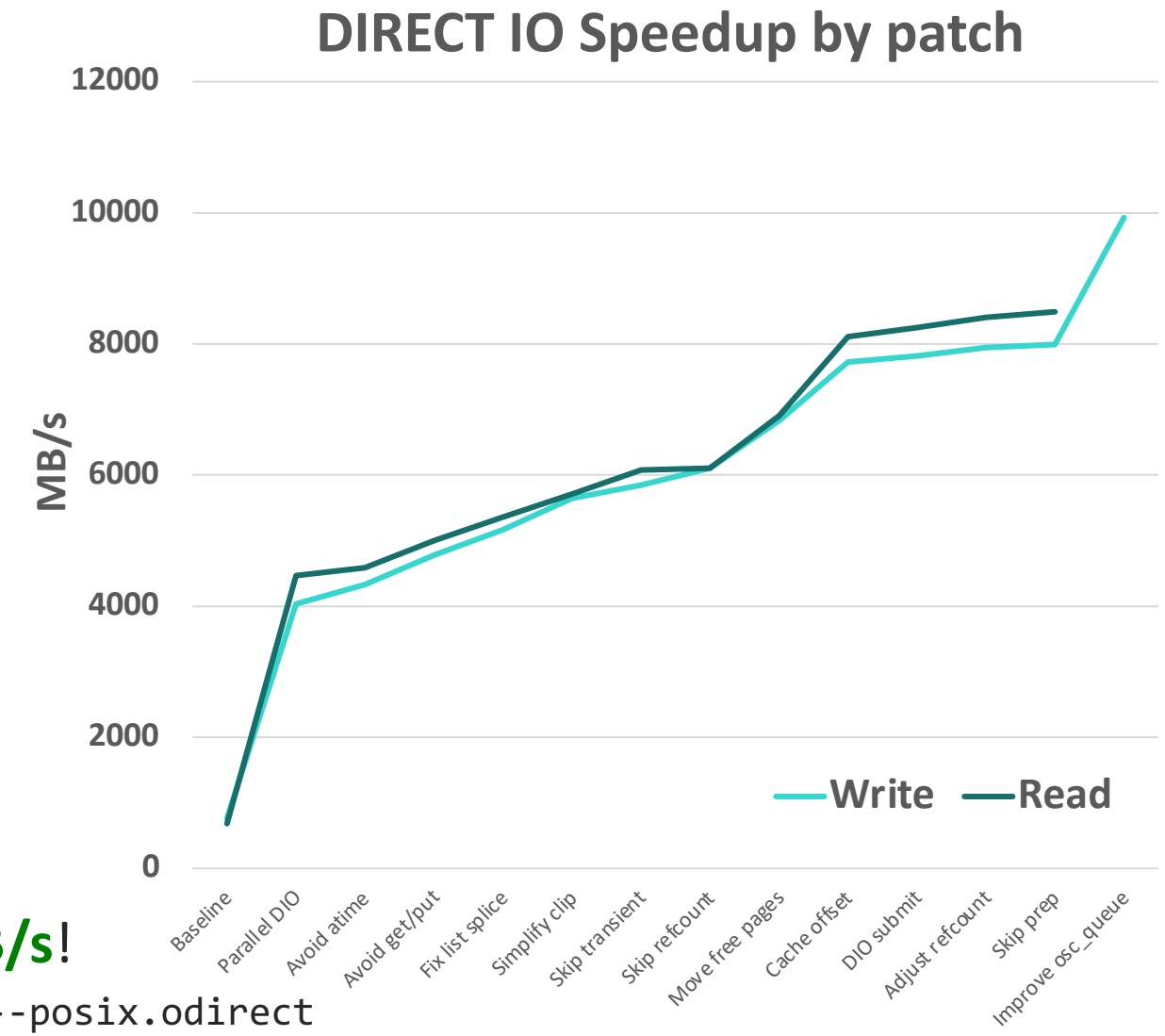
2.14
- **Working `io_uring`** with kernels 5.1 and later (LU-13801, WC)

2.15
- **Improved mmap readahead** chunk detection (LU-13669, WC)
  - Reduced pagefault latency, avg 512usec->52usec, max 37msec->6msec

Faster O_DIRECT (LU-13798, LU-13799, HPE, WC) (2.15)

Whamcloud

- Prototype patch series by Patrick Farrell
  https://review.whamcloud.com/39436...
- Parallel large single thread O_DIRECT IO
  - 64MB read/write ~700MB/s->**4GB/s**
- Reduce DIRECT IO submission overhead
  - No per-page inode timestamp updates
  - Skip unnecessary page/request refcount
  - Improve page list handling
  - Cache page offset calculation
  - Improve DIO space accounting
- Further DIO speedup from 4GB/s->**10GB/s**!



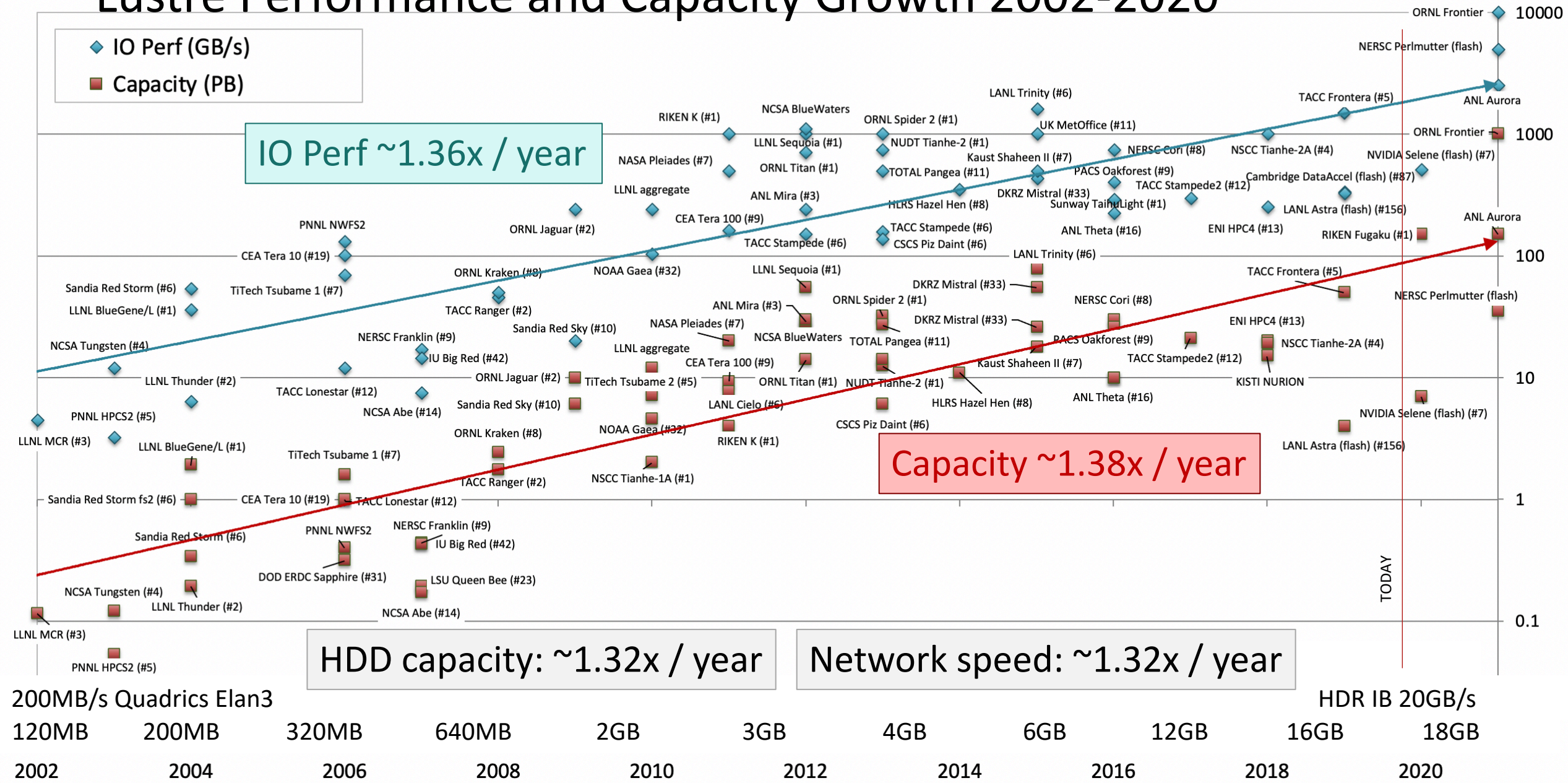

```
mpirun -np 1 IOR -wr -t64M -b64G -o iorfile --posix.odirect
```

Exceeding 1 PiB `ldiskfs` OSTs

Whamcloud

- OST size is growing exponentially
- HDD size continues to grow - 18TiB+ this year
- Declustered RAID benefits from more disks per LUN
  - Distributed hot spares reduce rebuild times with more disks
  - Often 50-100 HDD per OST instead of 8+2 in the past
- OST already at 600-700TiB, up to 1.2PiB in some cases
- Single OSTs larger than whole filesystems 15 years ago

Lustre Performance and Capacity Growth 2002-2020



Source: [Rock Hard Lustre](), Nathan Rutman, Cray (with updates for recent years); [Disk Drive Prices (1955-2019)](), John C. McCallum

Other `ldiskfs` Improvements (2.14+)

Whamcloud

- **Fix mount time for huge OSTs**, (LU-12988, LU-13241, WC, HPE)

2.14

- **Parallel e2fsck** speedups (LU-8465, WC)

2.15

- OST object directory scalability (LU-11912, WC)
  - Limit directory size and group objects by age for better insertion/removal efficiency
- **Directory shrink** as files are deleted from old directories (LU-12051)
- Existing features available that *could* benefit Lustre (under discussion)
  - Efficient large block allocation for large OSTs (**`bigalloc`**, LU-12967)
  - DoM files/directories <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)

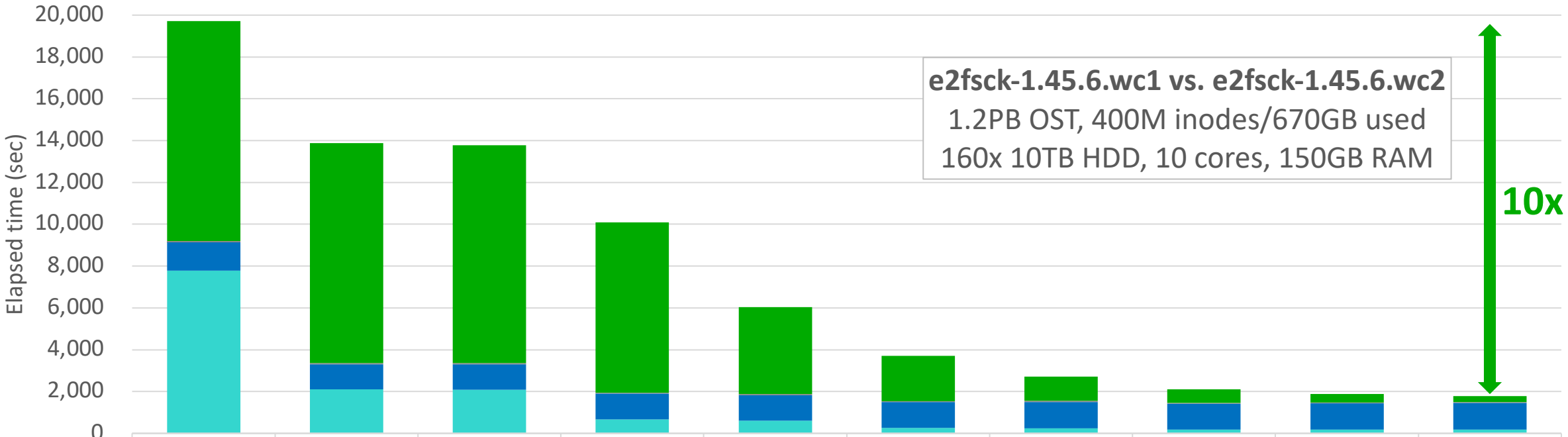
2.16

- **Dynamic ext4 inode allocation** for MDTs and OSTs (LU-12099)
  - Early discussions with upstream about design approach
- **Merge `ldiskfs dirdata`** feature to upstream `ext4/e2fsprogs`

Parallel e2fsck Performance at Scale (LU-8465)

Whamcloud

- Multi-threaded pass1 inode/block scan, and parallel pass5 bitmap loading



	original	thr=1	thr=2	thr=4	thr=8	thr=16	thr=32	thr=64	thr=128	thr=256	
pass5	10,523	10,524	10,427	8,133	4,166	2,168	1,169	654	399	286	36x
pass4	55	50	49	52	52	47	44	43	43	47	
pass3	0	0	0	0	0	0	0	0	0	0	
pass2	1,363	1,206	1,217	1,226	1,226	1,239	1,255	1,248	1,261	1,265	7%
pass1	7,777	2,101	2,092	669	597	254	250	171	181	191	40x

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

Whamcloud

- **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)
  - Parity can be added to existing striped file layouts
- **Integrate FLR with HSM/PCC/Foreign mirror** (LU-10606)
  - Multiple archives per file (POSIX, S3, tape, ...)
  - PCC with local cache and filesystem mirror
  - Partial HSM file copy/restore to/from archive

2.15

2.16

- **File Versions** using FLR mirrors (LU-12648)
- **Immediate file write** mirroring (LU-13643)
  - Client writes both copies of mirror directly

MDT Layout Improvements (WC) (2.14+)

Whamcloud

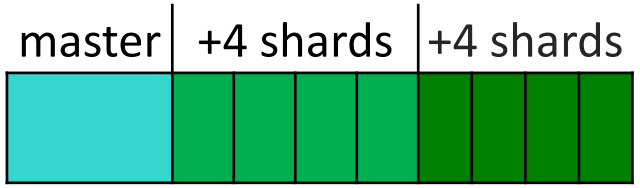
- **DNE space balance `mkdir()`** on "best" MDT by available inodes/space (LU-12624)

2.13

  - `lfs setdirstripe -D -c 1 -i -1 /path/to/directory`

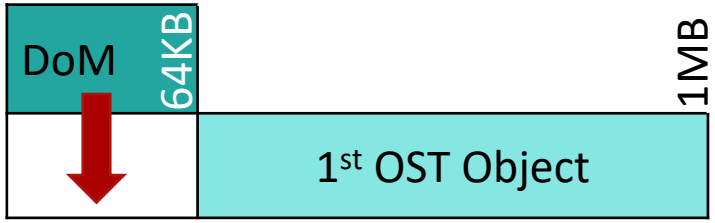
2.14

- **DNE `crush` directory hash** to improve migration (LU-11025)
- **DNE automatic directory restripe** as it grows (LU-11025)
  - Distribute large directories across multiple MDTs for space/performance
- **DoM component shrink** if MDT free space low (LU-12785)
  - Reduce or eliminate DoM component to avoid ENOSPC on MDT



2.15

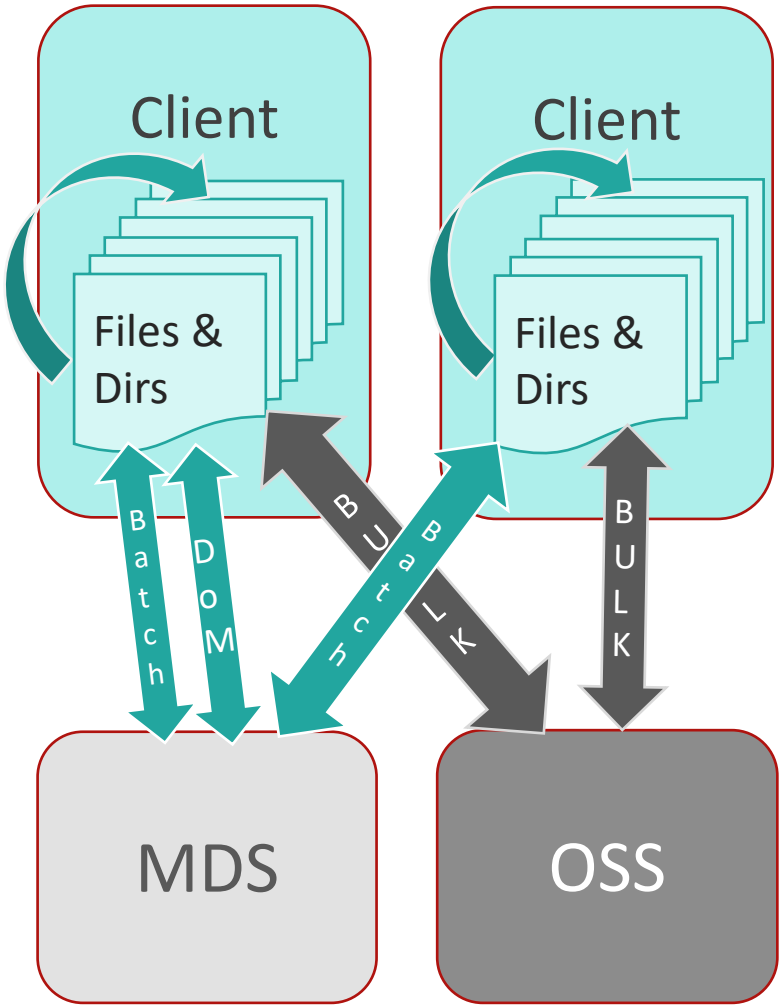
- DNE MDT usage/space balance for new filesystems (LU-13417)
- DoM->OST migration optimization (LU-13612)



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



Whamcloud

Thank You!
Questions?