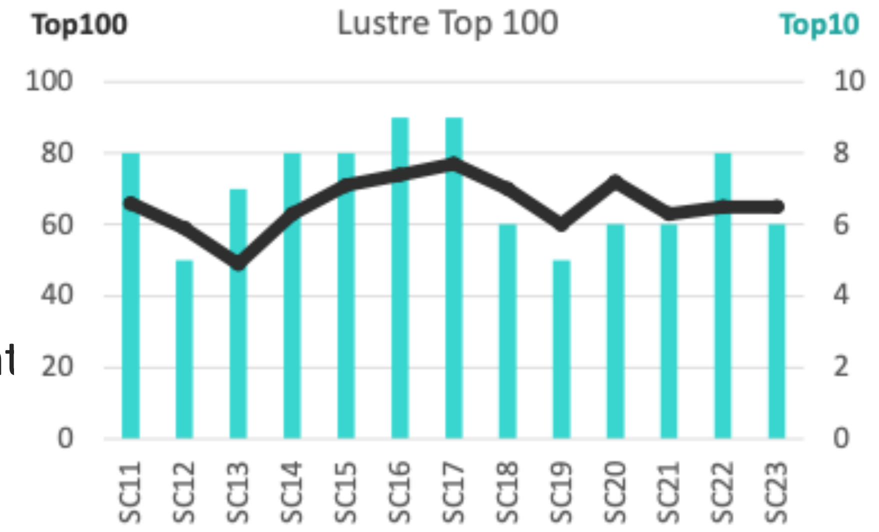
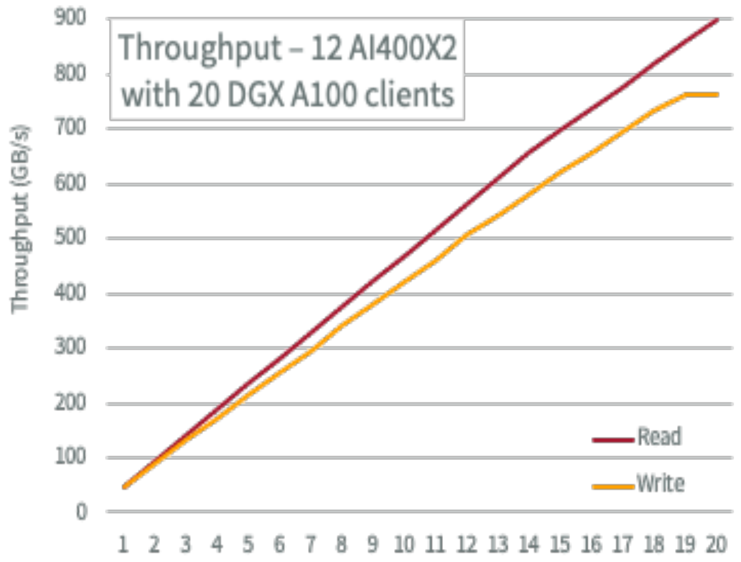
Lustre status and path forward

Sébastien Buisson
Senior Lustre Engineer

Big is Beautiful

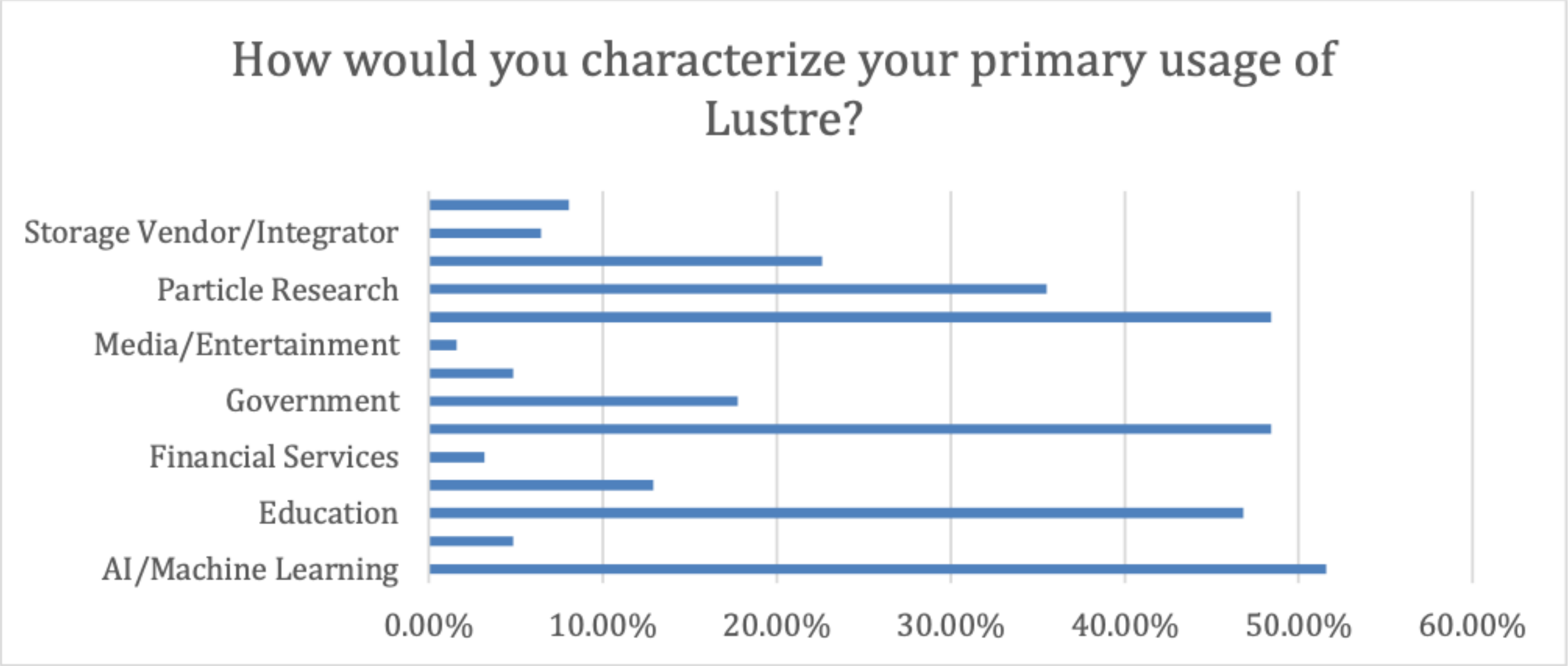
- Lustre is the preferred choice for Exascale systems
  - World's largest AI/ML systems (Eos, Selene, Cam1, Scaleway)…
  - …also 2RU servers available with 60W/80R GB/s and 3M read IOPS
- Proven scalability of servers and clients to meet system requirement
  - 100M+ IOPS, 10 M+ metadata op/sec, 100B+ files today, ongoing improvements
  - Capacity can grow almost without limits - 100's PB today, 1 EB+ in the near future
  - Bandwidth scales almost linearly, aggregate 10's of TB/s today
  - Fully support client capabilities - 100's cores, TB's RAM, multi-100Gbps NICs, GPU RDMA
- Ongoing improvements for large system deployments
  - Steady feature development to meet evolving system and application needs
- Improving ease-of-use and reliability
  - Demand for fast storage is everywhere





… but Size is not Everything

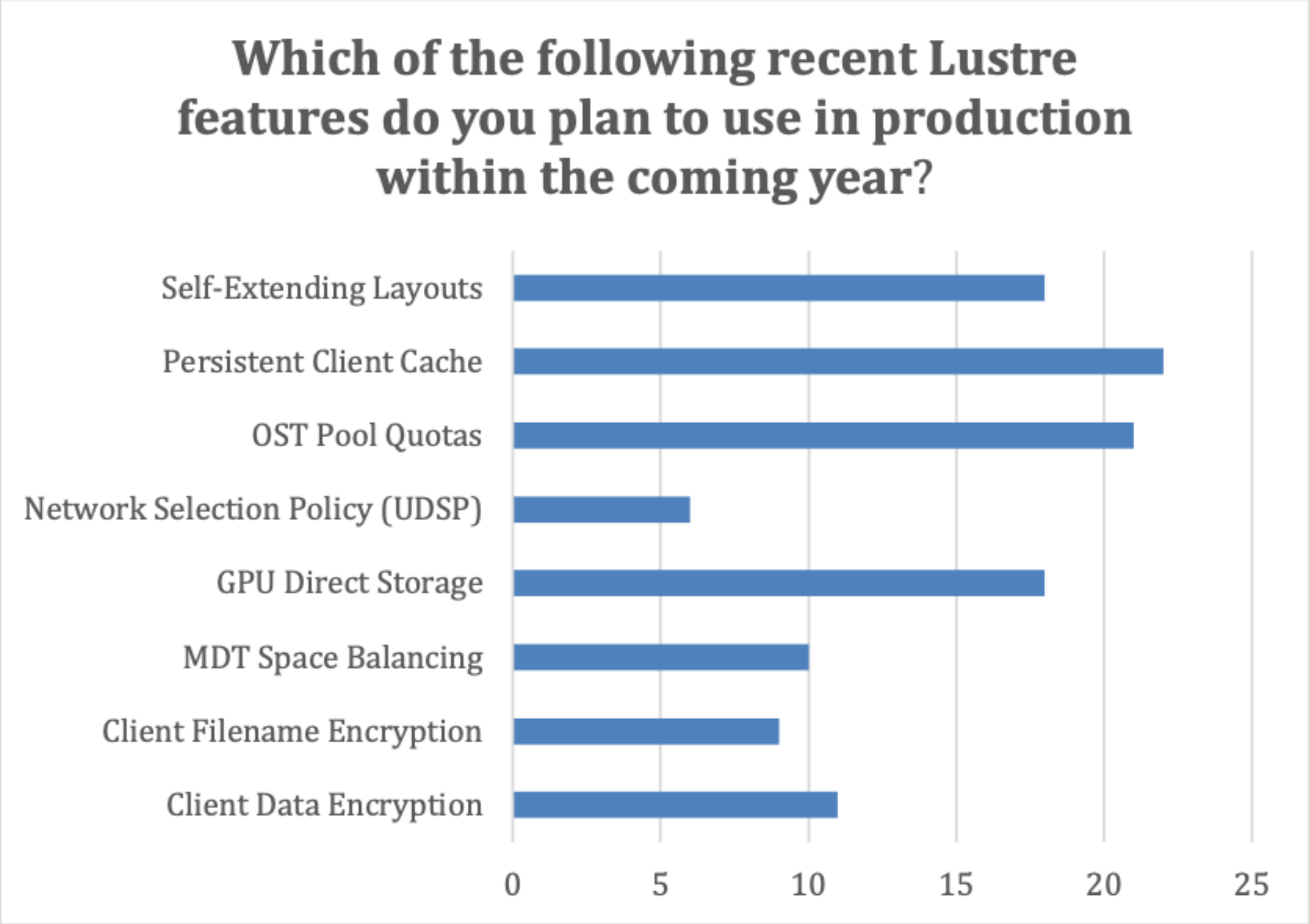
- Lustre at ease in a variety of market segments



https://wiki.opensfs.org/Lustre_Community_Survey

… but Size is really not Everything

- Ongoing improvements for smaller systems too
  - Ongoing improvements in ease of use, flexibility, monitoring, reliability
  - Configurable security, encryption, multi-tenant isolation, quotas, etc.

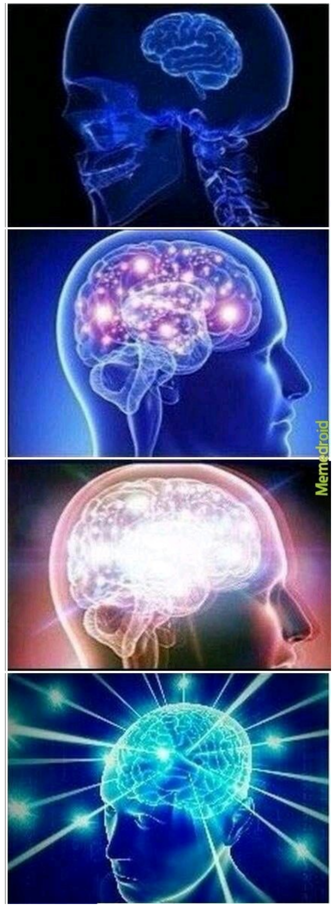


https://wiki.opensfs.org/Lustre_Community_Survey

Evolution of IO Interfaces

▶ POSIX has been the standard IO interface for decades

- Protects significant investment in application development
- Consistent behavior avoids chasing interface-of-the-month
- Data portability via protocol export (Lustre, NFS, SMB, S3, …)



▶ Keeping up with hardware speedups demands continual optimization

- Unaligned IO, cross-dir/file prefetch, WBC improves speed transparently
- Asynchronous meta/data ops via Linux io_uring, batched file create

▶ Opt-in API *extensions* for apps with special performance needs

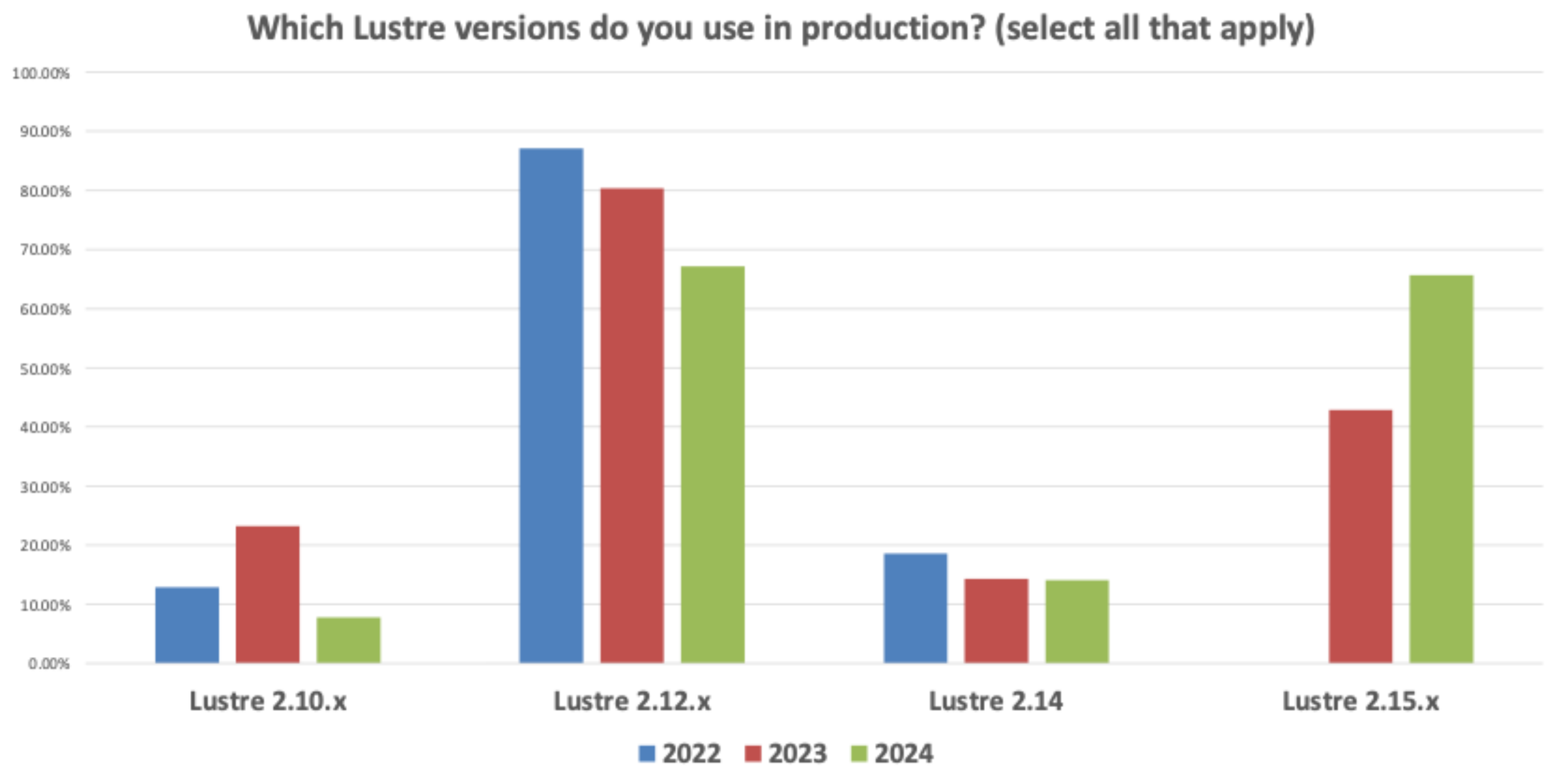
- Relaxed semantics/interfaces when/where applications need/understand it
- Data stored and continues to be accessible via standard APIs afterward

▶ POSIX will continue to be the common interface going forward

Lustre Community Release News

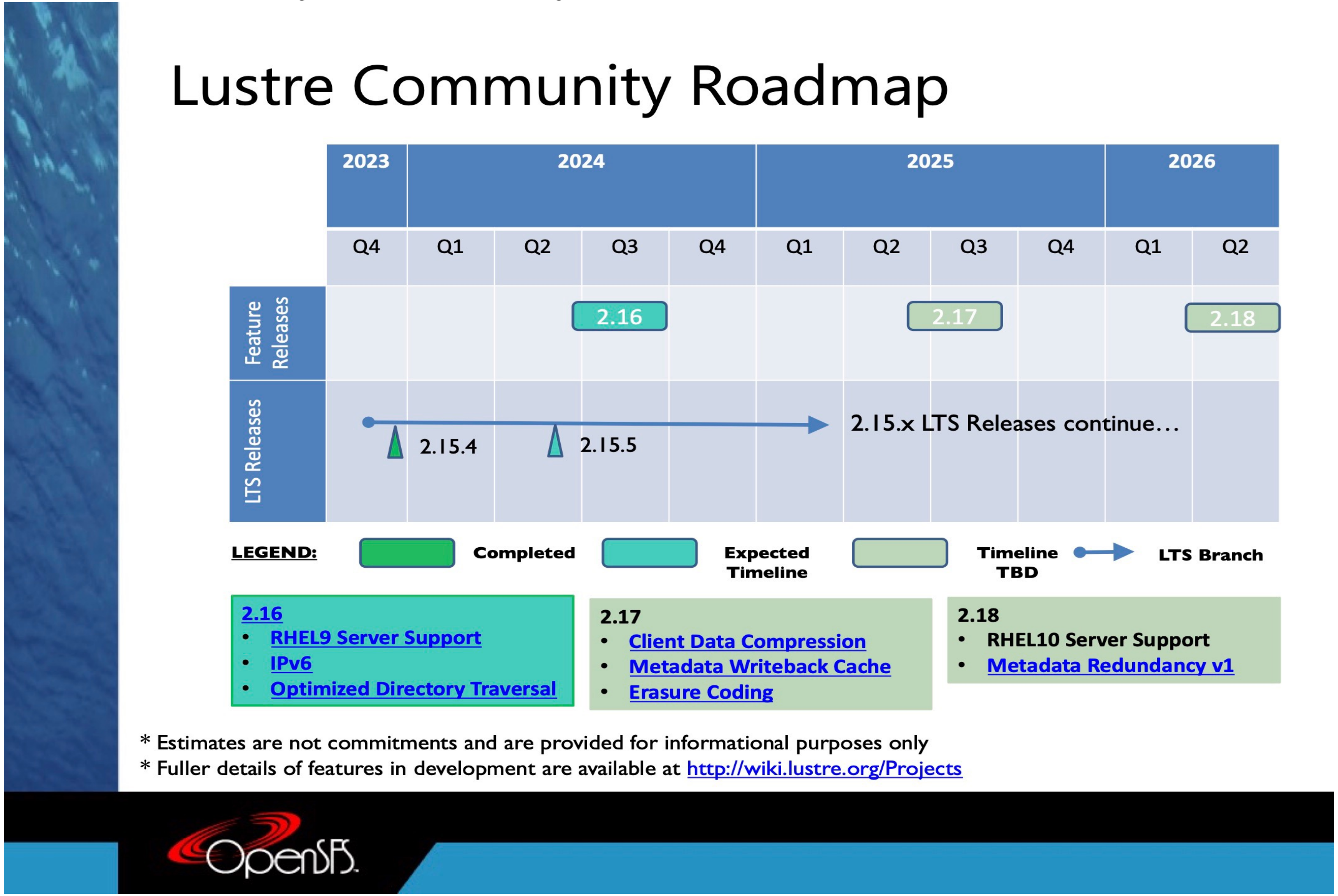
Lustre LTS Releases

- Lustre 2.15.0 GA June 2022
  - Replaced 2.12.x as LTS branch
- Lustre 2.15.5 coming soon
  - RHEL 8.10 servers/clients; RHEL 9.4 clients
  - https://wiki.lustre.org/Lustre_2.15.5_Changelog



https://wiki.opensfs.org/Lustre_Community_Survey

Lustre Community Roadmap



Planned Feature Release Highlights

Whamcloud

- **2.16** feature landings complete, now testing for release
  - **Optimized Directory Traversal** (WBC1) – improve efficiency for accessing many files (WC)
  - **LNet IPv6 addressing** – must-have functionality for future deployments (SuSE, ORNL)
  - **Unaligned Direct IO** – One-copy DIO to bypass client page cache (WC)
- **2.17** has major features already well underway
  - **Hybrid IO Optimizations** – Hybrid BIO/DIO and server writeback cache (WC, Oracle)
  - **Dynamic Nodemaps** – ephemeral/hierarchical configuration for subdirectory trees (WC)
  - **Client-side Data Compression** – reduce network and storage usage/cost (WC, UHamburg)
  - **Metadata Writeback Cache** (WBC2) – single-client metadata speedup (WC)
- **2.18** feature proposals in planning stage
  - **File Level Redundancy - Erasure Coding** (FLR-EC) – reduce cost, improve availability (ORNL)
  - **Lustre Metadata Redundancy** (LMR1) – improve availability for large DNE systems

LNet Improvements

Demand for IPv6 in cloud deployments as IPv4 addresses are exhausted

- IPv6 large NID support (LU-10391 SuSE, ORNL)
  - Variable-sized NIDs for future expandability
  - Interoperable with existing current LNDs whenever possible
  - Enhancements to LNet/`socklnd`/`o2iblnd` for large NIDs
- Handle large NIDs in Lustre configuration/mounting code
  - Mount, config logs, Imperative Recovery, Nodemaps, root squash, etc.
- Detect added/changed server interfaces automatically (LU-10360)
- Simplified configuration for IPv6 NIDs by clients (LU-14668)
- Testing is underway for 2.16.0 release

Single Client IO Performance Improvements

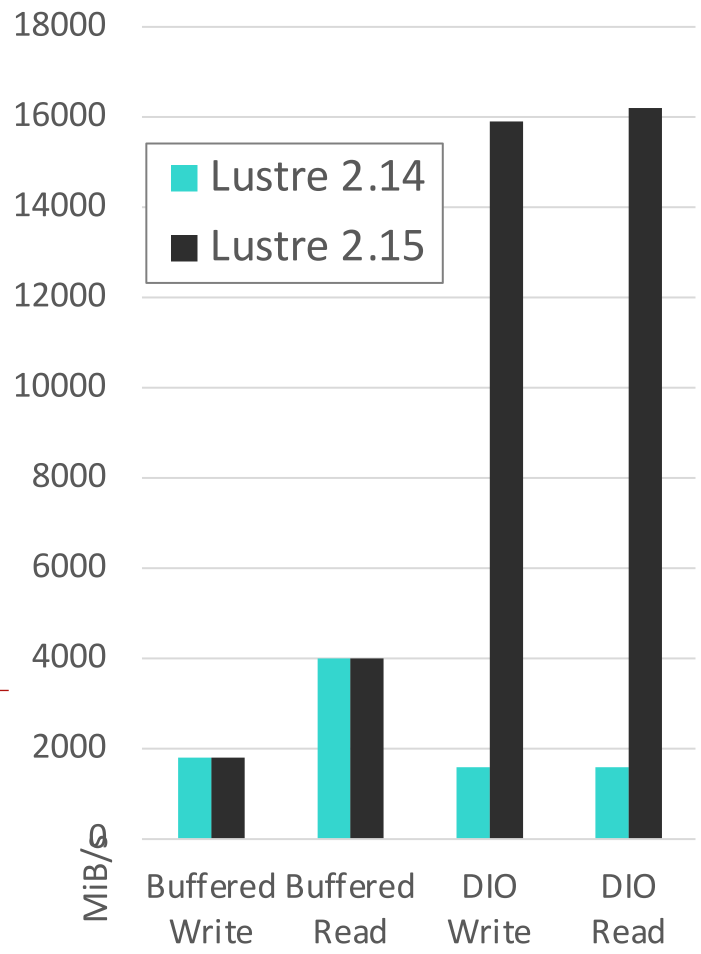
Client nodes are increasingly powerful (CPU, RAM, network) and data intensive

- Async Direct IO and `io_uring` on Linux 5.1+
  - Improved 4KB IOPS, write 100k->**266k IOPS**; read 80k->**610k IOPS**
  - Non-POSIX async interface for IO and (some) metadata operations
- Improved mmap readahead chunk detection
  - Reduced latency, avg 512usec->**52usec**, max 37msec->**6msec**
- Parallel/large Direct IO/Async IO performance
  - Improve large ***single-thread*** `read()`/`write()`, 700MB/s->**17GB/s**
- Unaligned Direct IO avoids page cache alignment/size
  - It turns out that data copy is not so bad after all

2.16

2.17

- Hybrid Buffered/Direct IO automatic switching
  - Dynamically switch IO to most efficient IO submission type
  - Transparent to userspace application with help of UDIO



Hybrid IO – IO path

- *What happens for each* `read()` *or* `write()` *?*
- Data transits between userspace, Lustre client, network and storage.
- POSIX gives two ways to do data I/O:
  - Buffered I/O
  - Direct I/O

Hybrid IO – Buffered IO

- Buffered IO means 'uses the kernel page cache'
  - Pages are created; inserted into cache; then data is copied to the page
    - This copy aligns data.
    - Storage and RDMA require aligned data for good performance
- Pros – Flexible:
  - Allows any I/O – no memory alignment requirements for userspace
  - Converts small application I/O to large I/O on disk
  - Allows hiding latency of slow devices (HDD)
- Cons – Not scalable:
  - Significant overhead for cache management

Hybrid IO – Direct IO

- Direct I/O means ‘direct from user memory, does not use the page cache’.

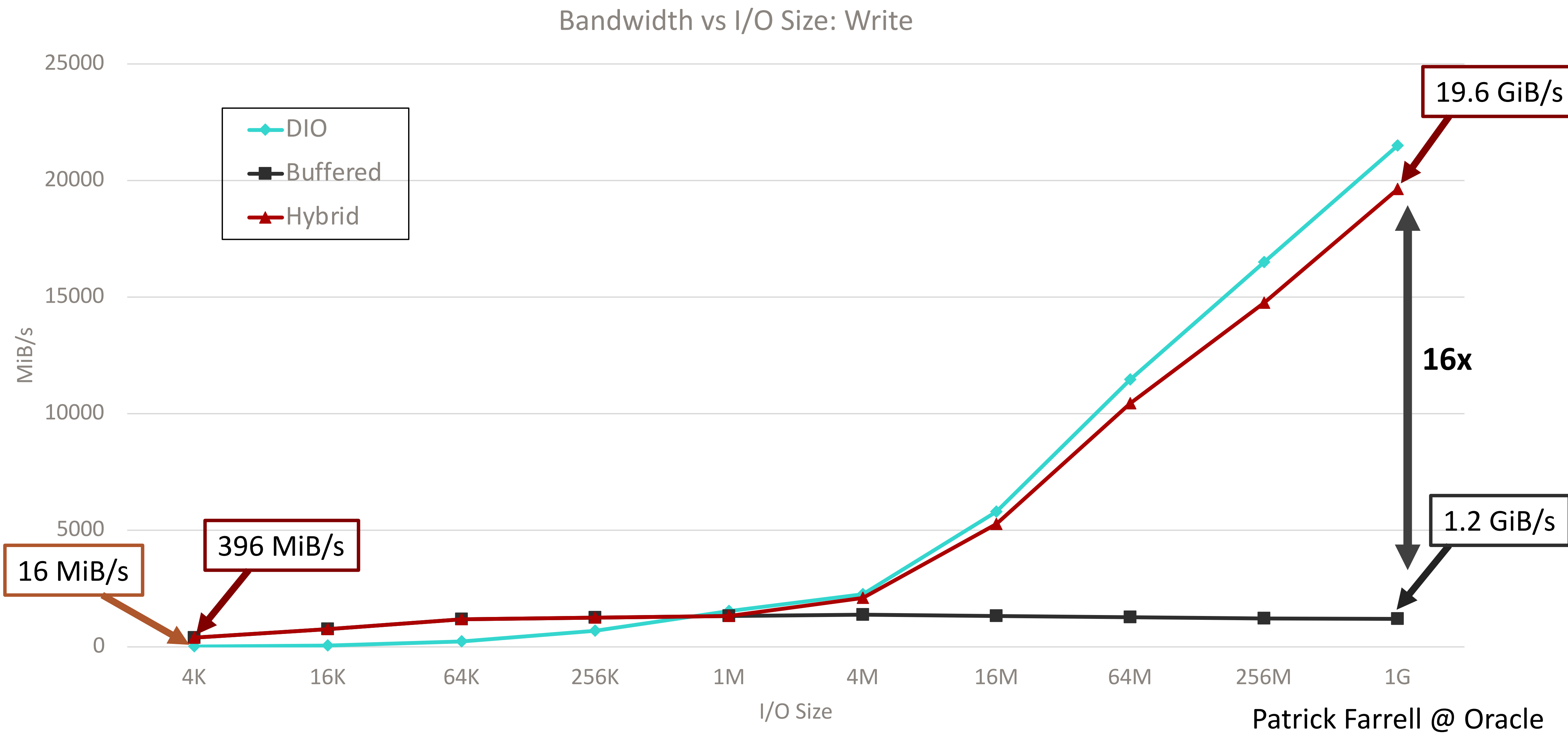
- Pros – Scalable:
  - Very high single stream performance with large I/O – 20+ GiB/s

- Cons – Inflexible:
  - Synchronous: I/O must go directly to disk
  - Exposes latency of slow devices
  - Alignment requirement

Hybrid IO

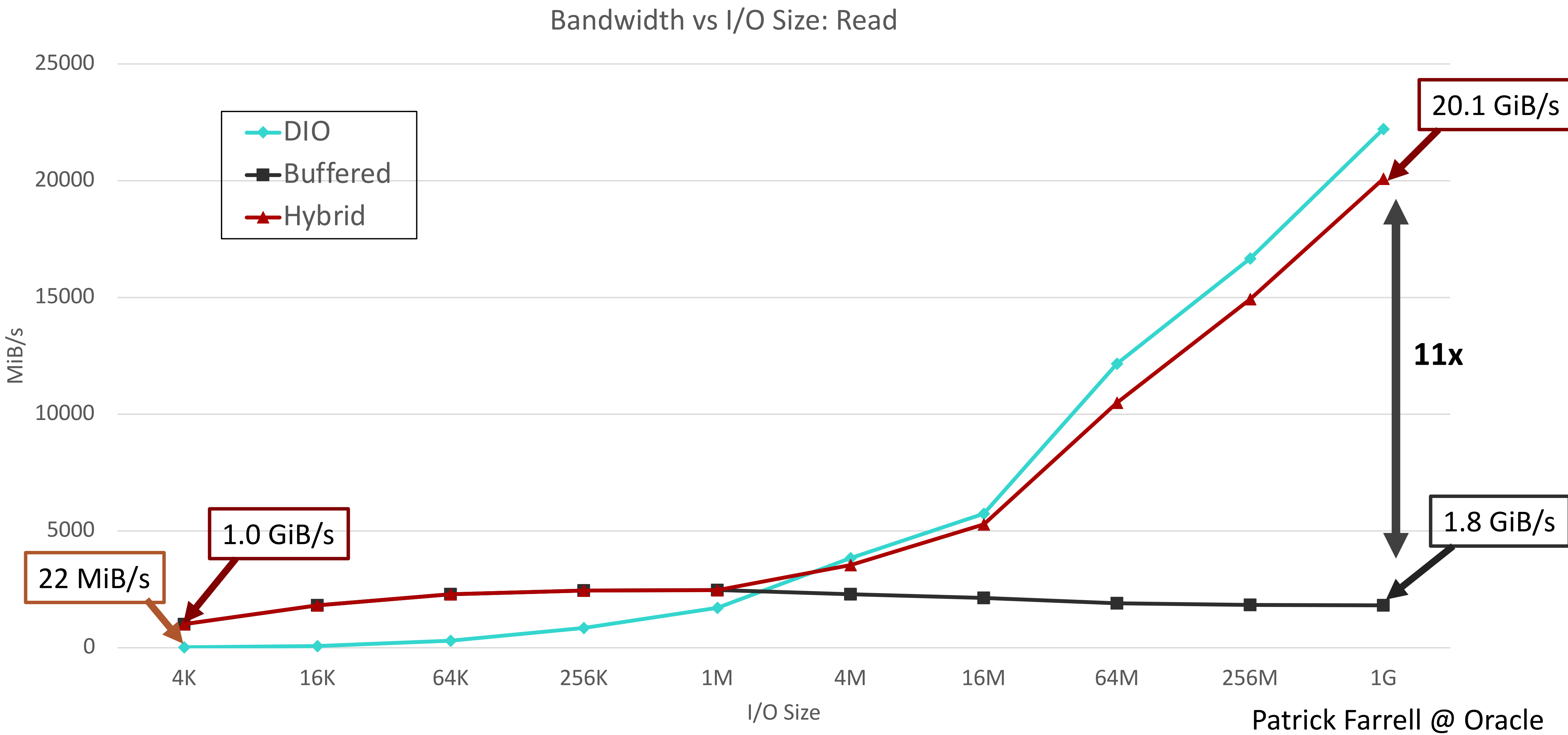
- Hybrid IO combines buffered and direct IO, getting the best of both
  - Use buffered I/O for small I/O
  - Use direct I/O for large I/O
  - Switch made internally, inside Lustre
    - transparent to applications
- Unaligned DIO
  - To get alignment:
    - Allocate an aligned buffer
    - Copy data to/from the buffer
    - Do direct I/O from the buffer

Hybrid IO: Write Performance



Patrick Farrell @ Oracle

Hybrid IO: Read Performance



Patrick Farrell @ Oracle

Client-Side Usability Improvements

Whamcloud

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

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

2.16 ▶ `lfs find -links` to find files with specific link count (LU-7495 LANL)

▶ `lfs migrate/mirror --bandwidth`, `--stats` updates (LU-13482 Amazon)
- For when Lustre client IO is ***too fast*** for the network/servers

▶ `lfs find -xattr` support (LU-15743 LANL)

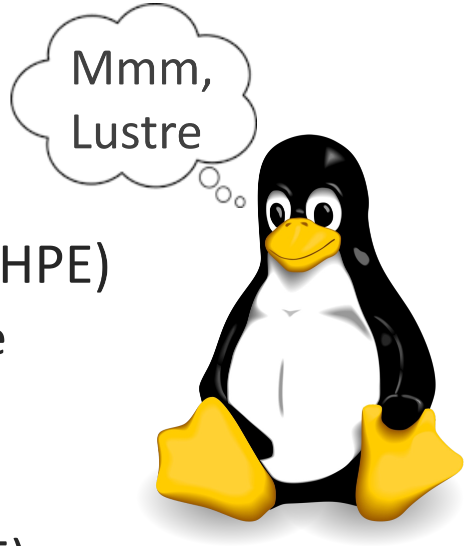
▶ Improved (over)stripe count handling (LU-13748 WC, LU-16623 WC, LU-16938 HPE)
- `lfs setstripe –E 32M –c -1` limits stripe count to 32 based on component size
- `lfs setstripe –C -N` to create N overstripes on all OSTs, N <= 32

▶ Remove 8192-device limit, for multiple large mounts (LU-8802 Amazon)

▶ Ongoing code updates/cleanup for upstream 6.x kernels (ORNL, HPE, WC, SuSE)

2.17 ▶ Client-side performance stats via statfs for each target (LU-7880 WC)

▶ Erasure Coded FLR files (LU-10911 ORNL)

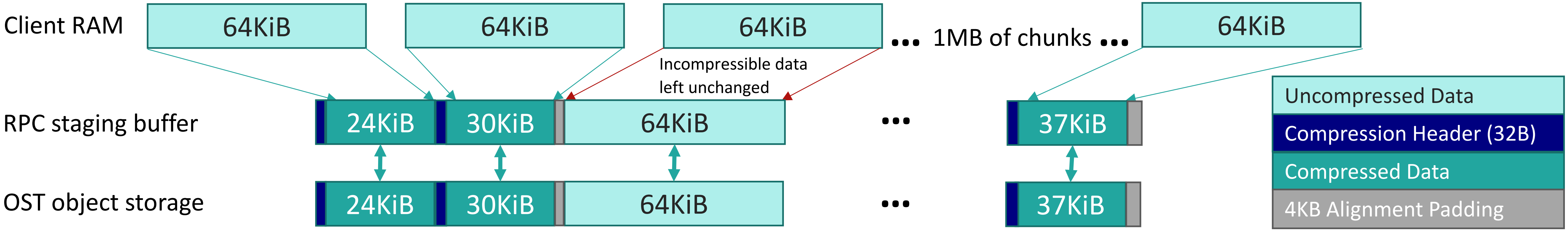


Client-Side Data Compression (2.17+ WC)

Whamcloud

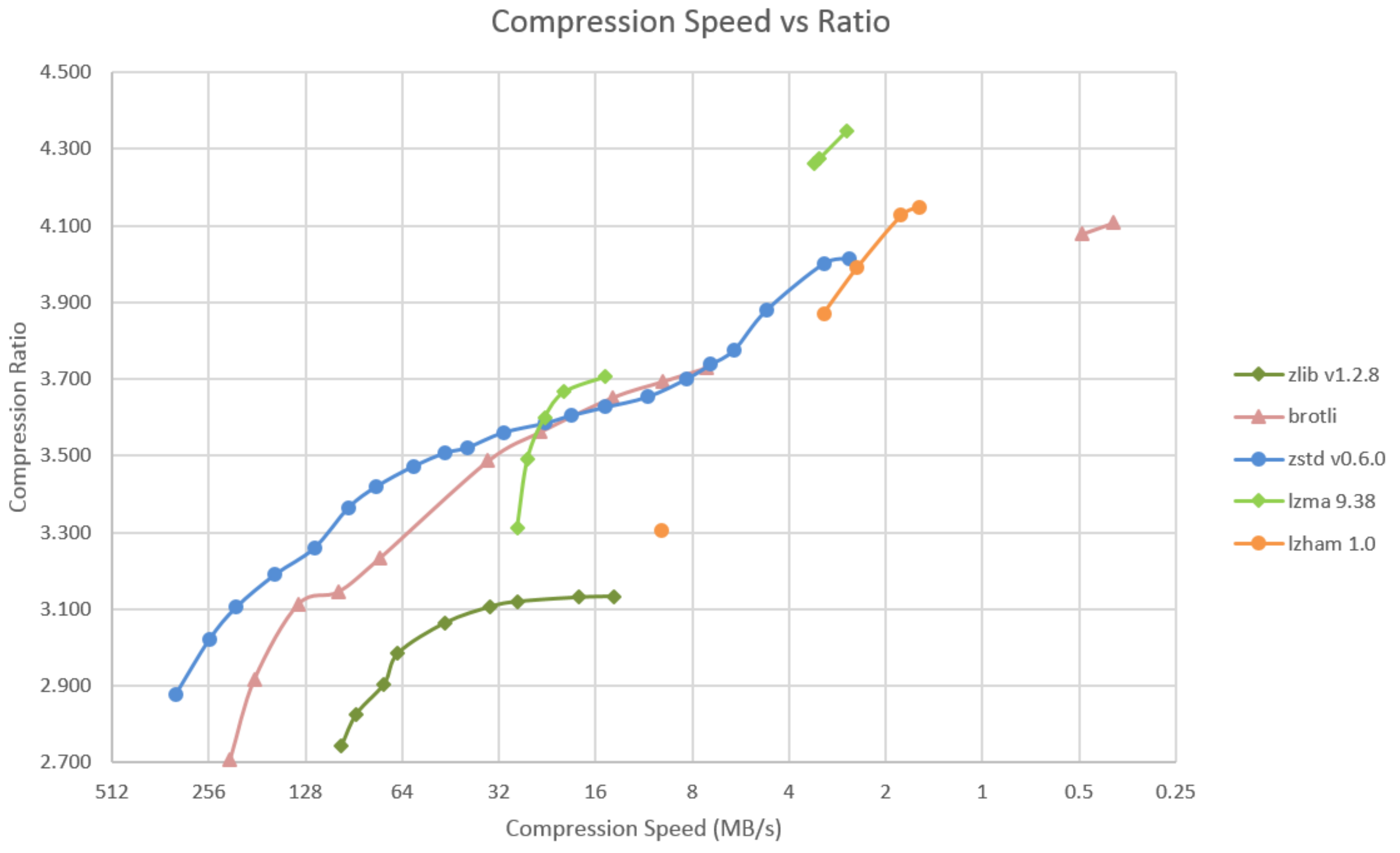
Increased capacity and lower cost per GB for all-flash OSTs

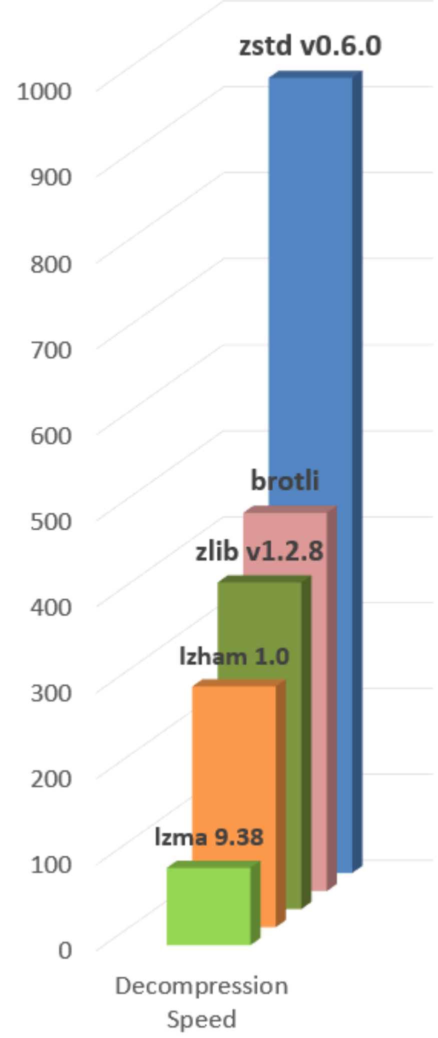
- Parallel (de-)compression of RPCs on client cores for GB/s speeds, **minimum server CPU usage**
- (De-)Compress (`lzo`, `lz4`, `zstd,`…) RPC on client in chunks (64KiB-4MiB+)
  - **Per directory or file component** selection of algorithm, level, chunk size (PFL, FLR)
  - Keep "uncompressed" chunks 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
  - Data (re-)compression during mirror/migrate to slow tier (via data mover)

Comparison of zstd vs. gzip Compression Speed vs. Ratio



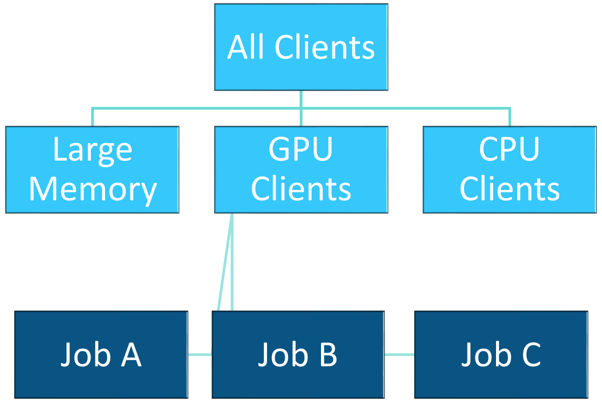


https://github.com/facebook/zstd/blob/dev/doc/images/DCspeed5.png

Improved Data Security and Containerization

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

- 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)
- 2.16 Configurable capabilities mask (LU-17410 WC)
- 2.17 Dynamic/hierarchical nodemap configuration (LU-17431 WC)
  - In-memory nodemap configuration for short-lived group (batch job)
  - Inherit parameters from static parent nodemap for most settings
- Encrypted file backup/restore/HSM without key (LU-16374 WC)



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)

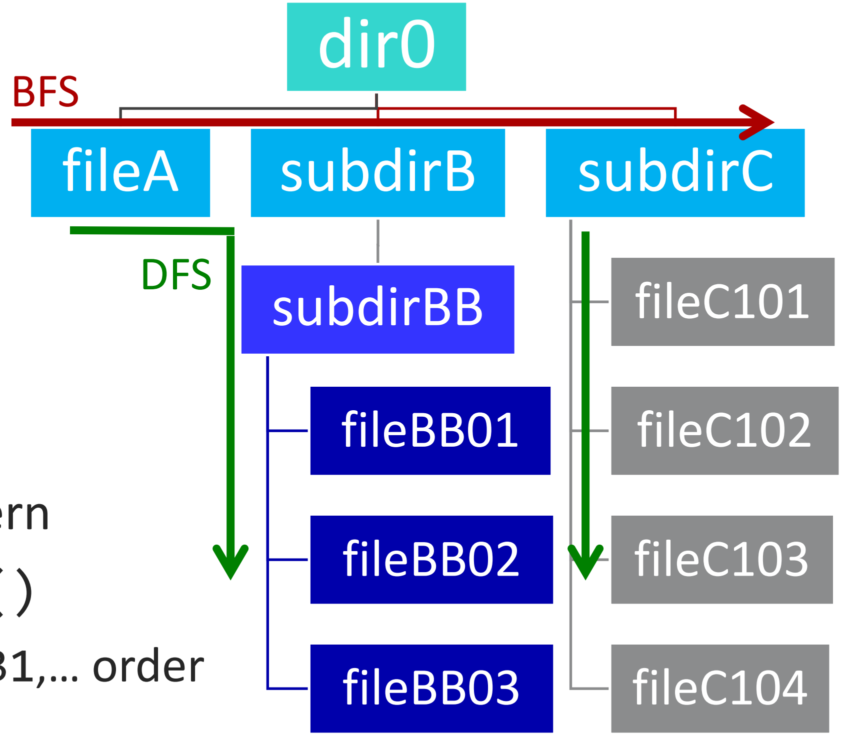
- Allow multiple getattrs/updates packed into a single MDS RPC
- More efficient network and server-side request handling

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

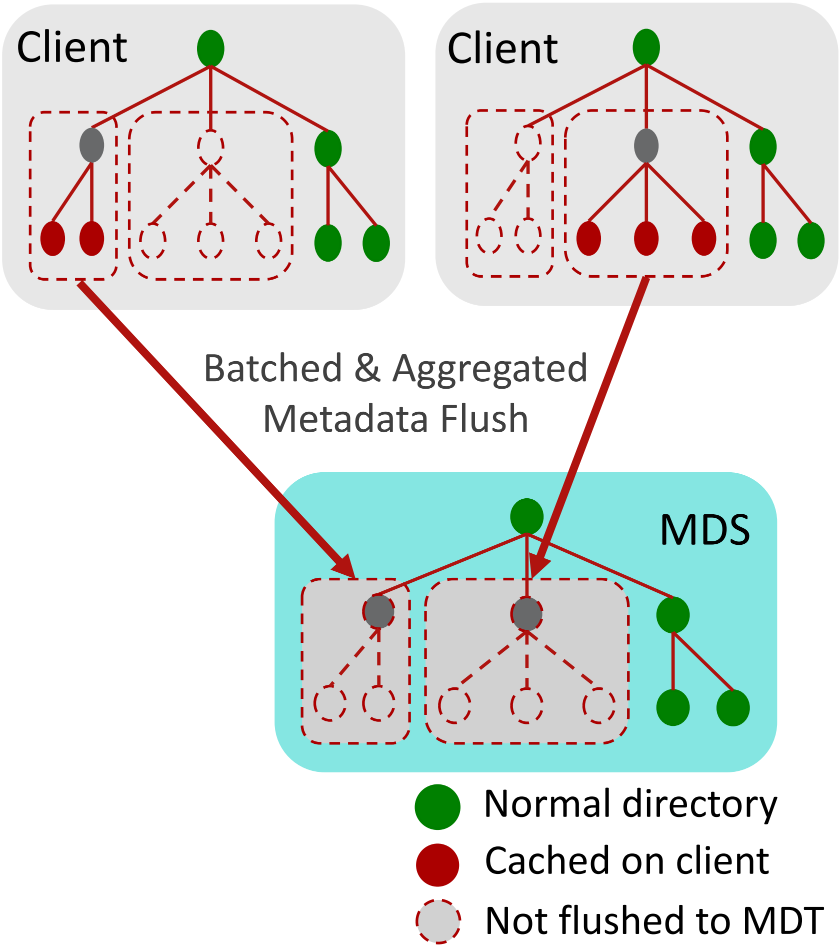
- Aggregate getattr RPCs for existing statahead mechanism

▶ **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+)



10-100x speedup for single-client create-intensive workloads

- Gene extraction/scanning, untar/build, data ingest, producer/consumer

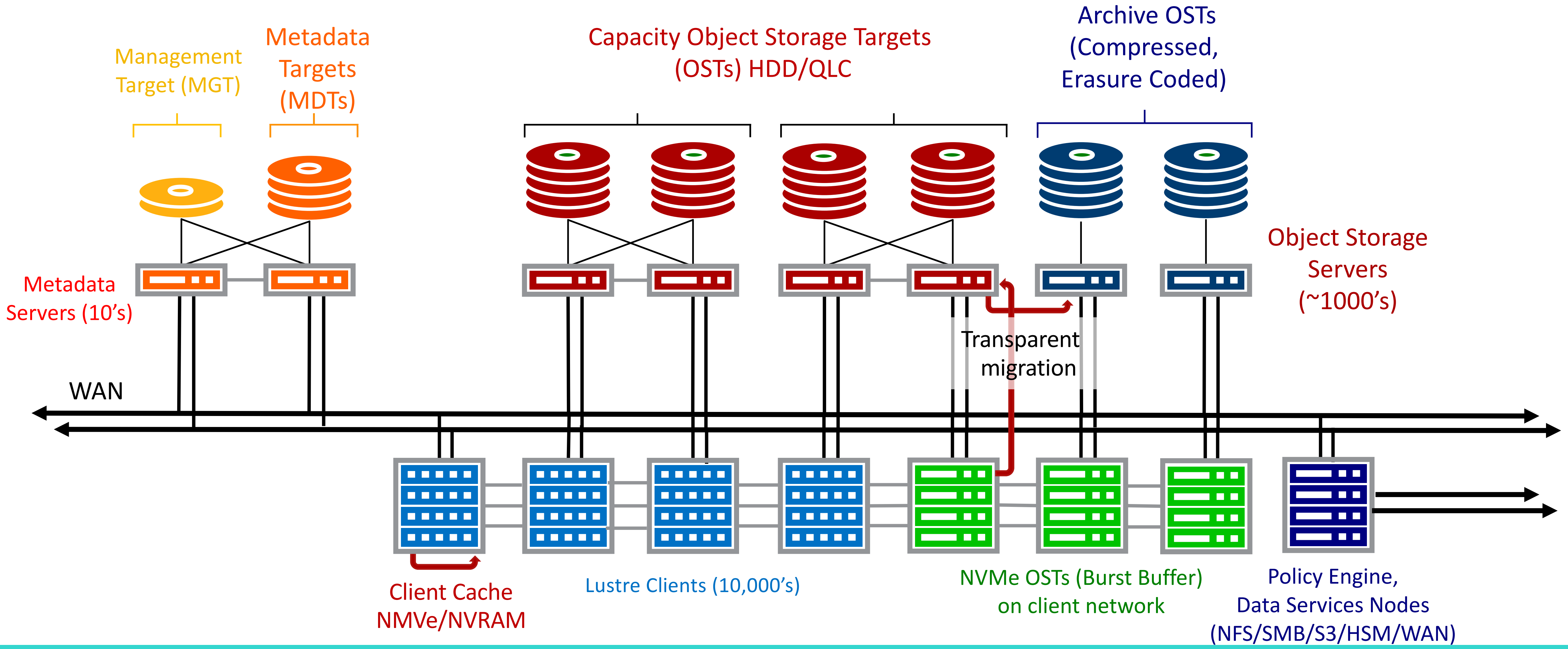
- Create new dirs/files **in client RAM without RPCs**
  - Lock new directory exclusively at `mkdir` time
  - Cache new files/dirs/data in RAM until cache flush or remote access
- **No RPC round-trips** for file modifications in new directory
- Batch RPC for efficient directory fetch and cache flush
- **Files globally visible on remote client access**
  - Flush top-level entries, exclusively lock new subdirs, unlock parent
  - 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

Whamcloud

Thank You!
Questions?

Tiered Storage and File Level Redundancy

Data locality, with *direct access* from clients to all storage tiers as needed



Server-side Usability Improvements

Whamcloud

Ongoing improvements to usability and robustness for ease of management

- OST Pool Spilling avoid out of space with hybrid OST tiers (LU-14825 WC)
- More robust MDT-MDT recovery llog handling (LU-several WC, CEA)
- Read-only mount of OST and MDT devices (LU-15873 WC)
- Hardening of online MDT/OST addition under load
  - MDT/OST "`-o no_create`" mount option to avoid new directory/object alloc (LU-12998 WC)
  - Delay/retry MDT/OST access when new target index found in layout (LU-17334 WC)
- `lljobstat` utility for easily monitoring "top/bad" jobs on MDT/OST (LU-16228 WC)
  - Add IO size histograms to `job_stats` output, handle bad job names better
- Store JobID into `user.job` xattr on inodes at create (LU-13031 LANL)
  - Semi-automatic provenance tracking for files/objects by users, admins (need JobID enabled)
  - Useful for post-mortem analysis of file creation issues (along with `crtime`)

2.16

2.17

- Enable default PFL layout on newly-formatted filesystems (LU-11918)
- Default NRS TBF rule(s) to keep "bad" jobs in check out of the box (LU-17296)

Server-side Capacity and Efficiency Improvements

Ongoing performance and capacity scaling for next-gen servers and storage

- Optimized locking for rename-intensive workloads (Spark, …)
  - Same-directory file/subdirectory rename optimization (LU-12125 WC)
  - MDS parallel cross-directory file rename optimization (LU-17426 WC)
  - Move rename to separate portal to avoid blocking other RPCs (LU-17441 WC)
  - Rename of regular file across projid directories without copy (LU-13176 WC)
- OST object directory scalability for multi-PB OSTs
  - Reduced transaction size for many-striped files/dirs (LU-14918 WC)
  - 2.16 Handling billions of objects on a single OST (LU-11912 WC)
- 2.17 ***Writeback*** cache for small, lockless, direct writes (LU-12916 WC)
  - Lower latency, small write aggregation, no lock ping-pong
  - Use ldiskfs delayed allocation (`delalloc`) until write is large enough, default 64KiB
  - Dynamic cache selection, complementary with client Hybrid Buffered/Direct IO

Ongoing ldiskfs and e2fsprogs Improvements

- Fix `e2fsck` for shared blocks, large dirs/journal (LU-16171, LU-14710, LU-17117 WC)
- `mkfs.lustre` to use `sparse_super2` feature for new filesystems (LU-15002 WC)
  - Allows larger group descriptor table for filesystems > 256TiB
  - 2.16 Avoids `meta_bg` feature that splatters metadata across device (seek avoidance)

- 2.17 More efficient ldiskfs mballoc for large filesystems (LU-14438 Google, IBM, WC, HPE)
  - Backport improved list-/tree-based group selection from upstream kernel
- Hybrid ldiskfs LVM storage devices (NVMe+HDD) (LU-16750 WC)
  - `IOPS` flag on block groups on NVMe at start of device, use for static/dynamic metadata
- Persistent `TRIMMED` flag on block groups during `fstrim` (LU-14712 WC)
  - Avoid useless `TRIM` commands on device after reformat and remount
- Enable ldiskfs delayed allocation for writeback cache (LU-12916 WC)
- Parallel e2fsck for pass2/3 (directory entries, name linkage) (LU-14679 WC)

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 at create (LU-13031, LANL)

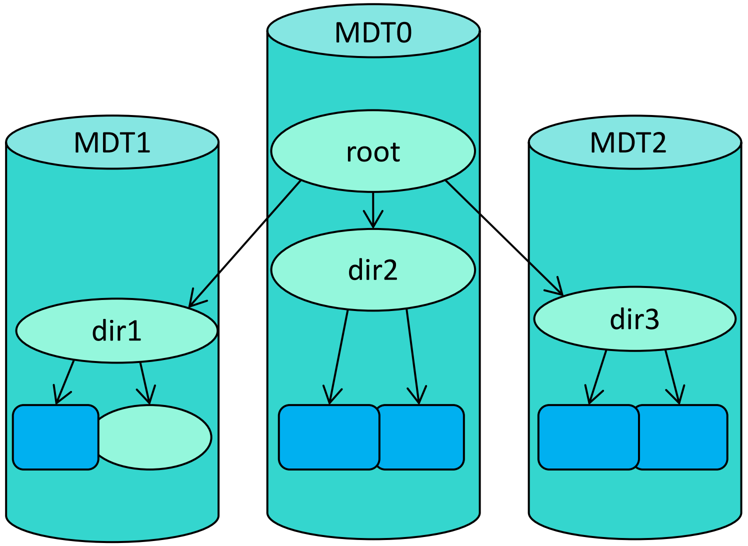
▶ Exclude list for pathnames from remote mkdir (LU-17334)

2.17 ▶ DNE locking, remote RPC optimization (LU-15528)

- Distributed transaction performance, reduce lock contention

2.18 ▶ Lustre Metadata Robustness/Redundancy (LU-12310)

- **Phase 1** to distribute/mirror MDT0000 services to other MDTs



Trends in High Performance Storage

AI/ChatGPT/LLM driving surge in new parallel flash storage users.
Compute models (weather, finance, …) grow resolution, inputs, historical data.
New computing paradigms continually drive new usage storage patterns.

Many needs met by all-flash storage, but not everyone has budget to scale.
Need more capacity, reduced costs, transparently on multiple storage types.

HDD, QLC plus compression trend to lowest $/TB, more accessible than tape.

Meta/data redundancy improves availability, reduces hardware complexity.

More security, multi-tenancy, data isolation (medical, privacy, IP, legal, …).

IO500 Performance History

Performance improvements go beyond what hardware upgrades have provided

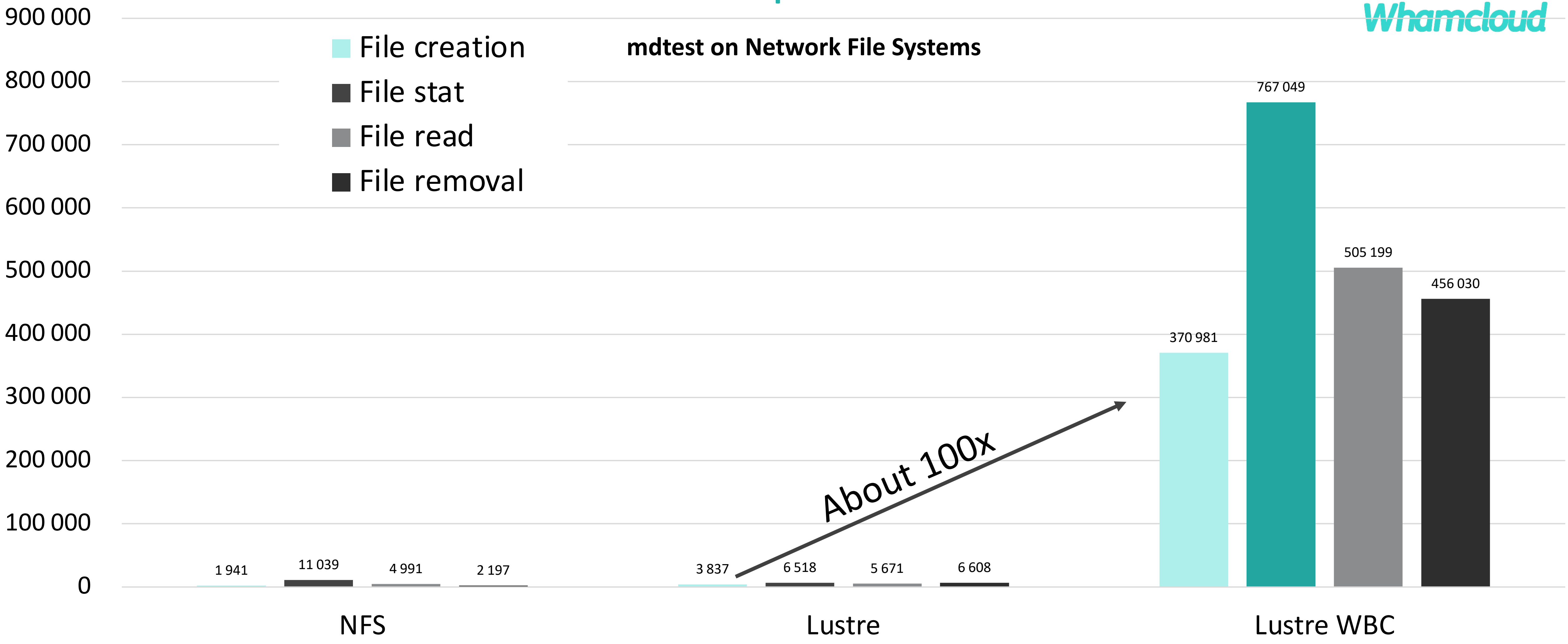
Whamcloud

- Hardware Configs
  - 4 x Lustre MDS+OSS
    - 12 x CPU core
    - 142GB RAM
    - 1 x HDR200 InfiniBand
    - 24 x NVMe (shared)
  - 10 x Lustre Client
    - 16 x CPU core
    - 96GB RAM
    - 1 x HDR100 InfiniBand

Storage Platform	1x ES400NV		1x ES400NVX	1x ES400NVX2			
	Pre-SC19	SC19	ISC20	ISC22	SC22	ISC23	ISC23/PreSC19
Lustre Version	Untuned	2.12.58+	2.13.53+		2.15.51+	2.15.55+	
ior-easy-write	25.8	28.62	37.56	55.95	58.07	57.88	2.2x
ior-easy-read	39.9	41.72	45.95	83.86	77.56	79.08	2.0x
ior-hard-write	2.7	2.96	2.77	5.02	5.27	5.38	2.0x
ior-hard-read	8.9	42.19	40.81	39.73	49.36	50.77	5.6x
find	1,735.4	810	1,698.00	6,248.55	12628.78	13,229.11	7.6x
mdtest-easy-write	143.8	152.84	157.22	270.04	312.9	344.70	2.3x
mdtest-easy-stat	455.0	451.97	453.51	740.01	1,278.50	1,276.31	2.8x
mdtest-easy-delete	88.5	132.76	135.09	223.61	272.64	311.16	3.5x
mdtest-hard-write	32.3	79.65	90.47	119.41	157.4	199.36	6.1x
mdtest hard-read	44.9	172.59	169	194.33	238.82	391.09	8.7x
mdtest Hard-stat	20.4	449.93	446.75	514.36	1,214.03	1,105.33	54.1x
mdtest Hard-delete	16.3	75.15	76.94	101.98	122.44	112.58	6.8x
Bandwdith	12.68	19.65	21.02	31.10	32.90	33.43	2.6x
IOPS	91.41	207.6	232.6	368.4	544.2	603.39	6.6x
Score	34.05	63.87	69.93	107.0	133.8	142.03	4.1x

https://io500.org/submissions/view/657

Metadata WBC Performance Improvements



Lustre: DDN AI400X Appliance (20 X SAMSUNG 3.84TB NVMe, 4X IB-HDR100)
Lustre clients: Intel Gold 5218 processor, 96 GB DDR4 RAM, CentOS 8.1
Local File System on SSD: Intel SSDSC2KB240G8