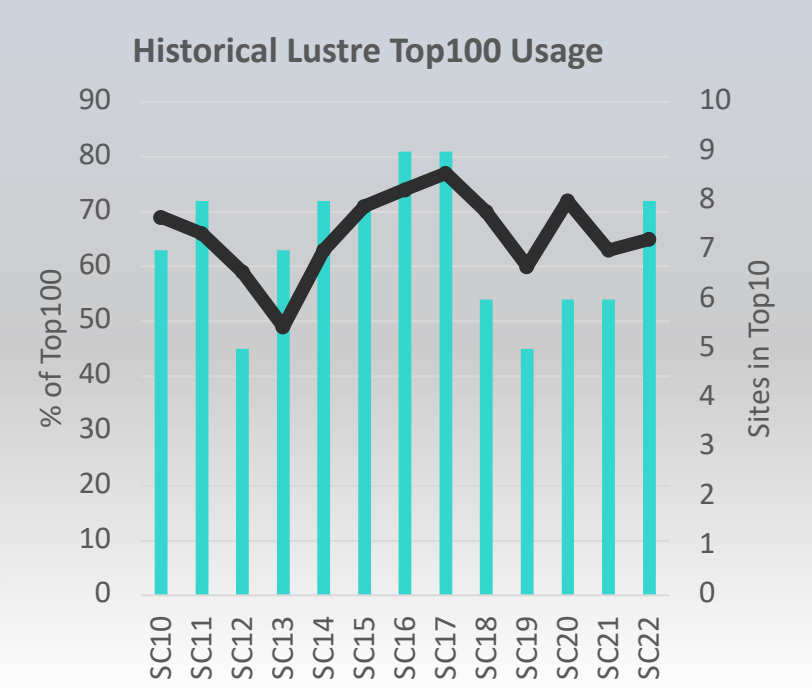
Whamcloud

将来のLustreロードマッププラン

井原修一

ddn

Lustre Committed to Exascale and the Future



- **Lustre is the preferred choice for Exascale systems**
  - 8 of top 10 HPC systems, largest AI/ML systems in the world

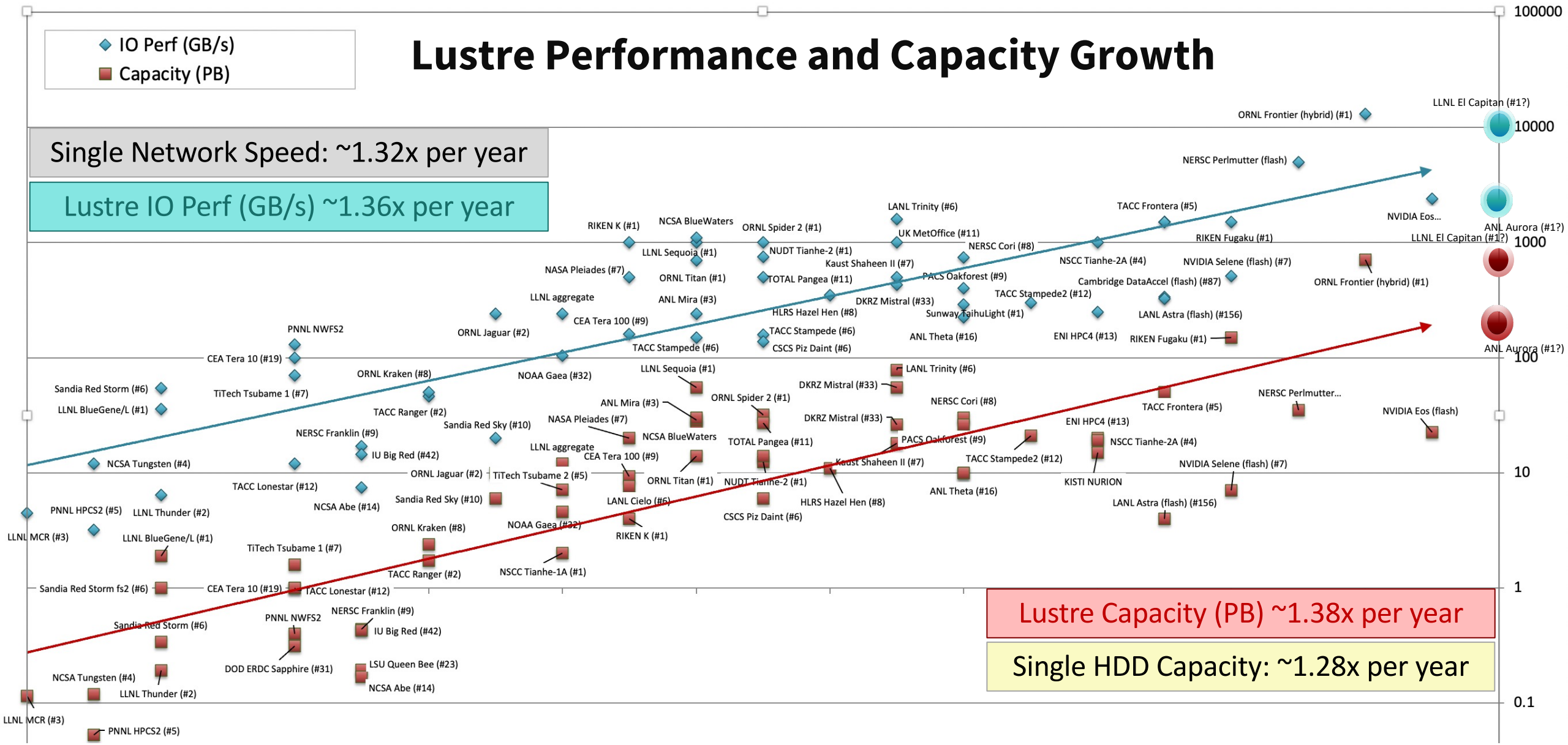
  … or 2RU server for workgroup with 16 GPU nodes

- **Scalability of servers and clients almost without limits**
  - 10 M/s+ metadata ops, 100B+ files, ongoing improvements
  - Capacity for any need - TB/s read/write, 100s of PB today and 1 EB+ in the near future
  - Fully support large clients - 100s of cores, TBs of RAM, multi-100Gbps NICs, GPU RDMA

- **Continued improvements for Top10 system deployments**
  - Steady feature development to meet evolving system and application needs
  - Working closely with multiple sites for next-generation systems

- **Improving ease-of-use and reliability**
  - Demand for fast storage is everywhere

Lustre Performance and Capacity Growth

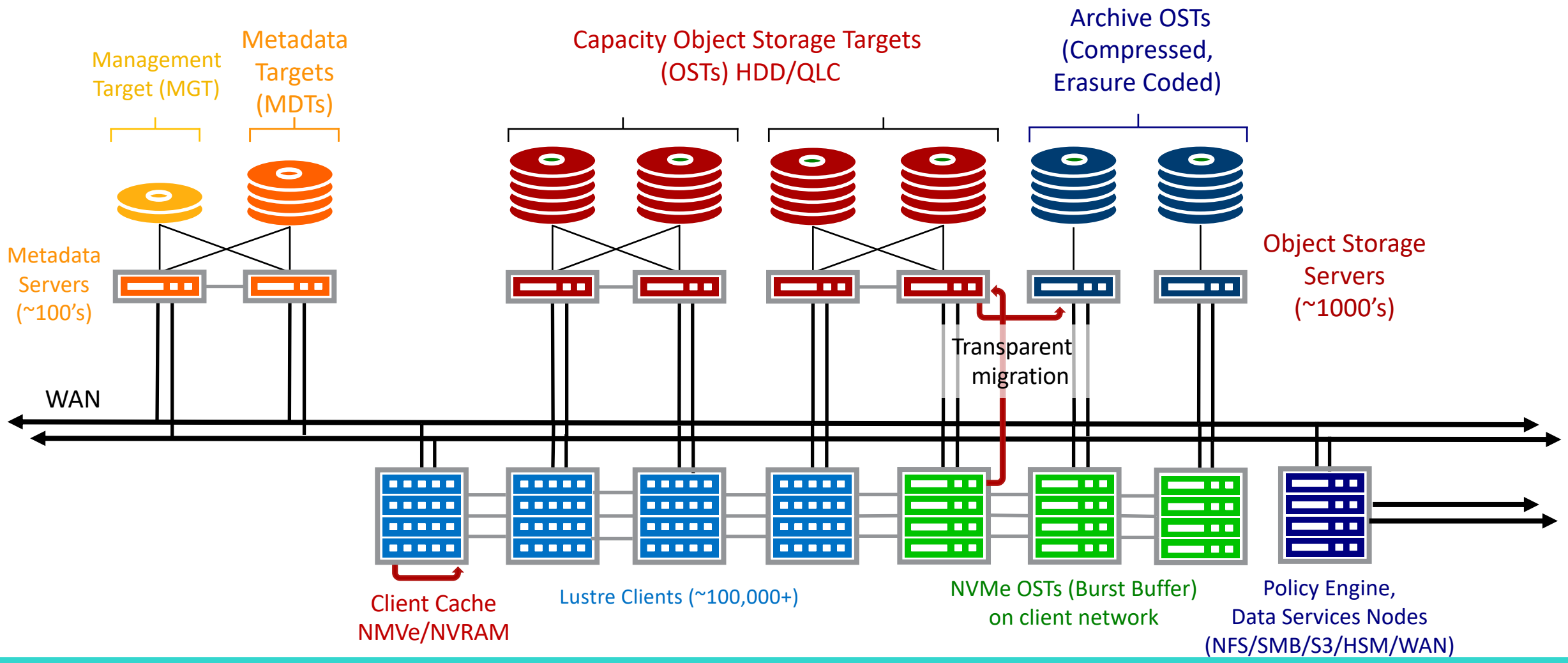


Source: [Rock Hard Lustre](), Nathan Rutman, Cray, (with updates since 2015); [Disk Drive Prices (1955-2020)](), John C. McCallum

Tiered Storage and File Level Redundancy

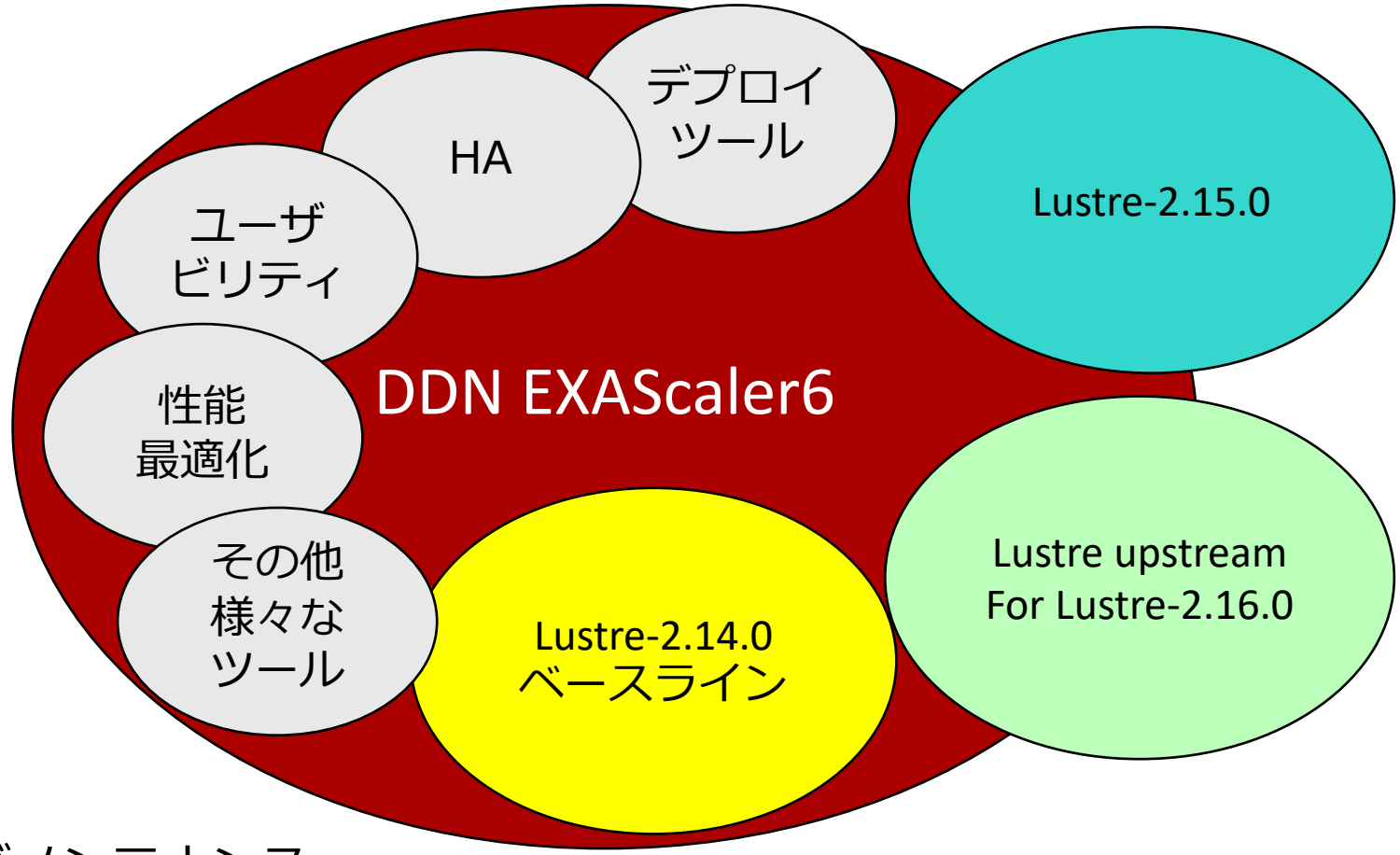
Data locality, with *direct access* from clients to all storage tiers as needed

Whamcloud



DDN EXAScaler と Lustre

DDN EXAScaler = DDNが提供するLustreディストリビューション



LustreはDDNがメンテナンス
ベースラインのLustreにBugfix及び新機能をバックポート、テストし提供する安定リリース版

Planned Feature Release Highlights

Whamcloud

- **2.16** feature landings complete
  - **Optimized Directory Traversal** (WBC1) – improve efficiency for accessing many files (WC)
  - **LNet IPv6 addressing** – must-have functionality for future deployments (SuSE, ORNL)
- **2.17** has major features already well underway
  - **Client-side data compression** – reduce network and storage usage/cost (WC, UHamburg)
  - **Metadata Writeback Cache** (WBC2) – single-client metadata speedup (WC)
  - **Unaligned IO Optimizations** – Hybrid BIO/DIO and server writeback cache (WC)
- **2.18** feature proposals in early stages
  - **File Level Redundancy - Erasure Coding** (FLR-EC) – reduce cost, improve availability (ORNL)
  - **Lustre Metadata Redundancy** (LMR1) – improve availability for large DNE systems

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

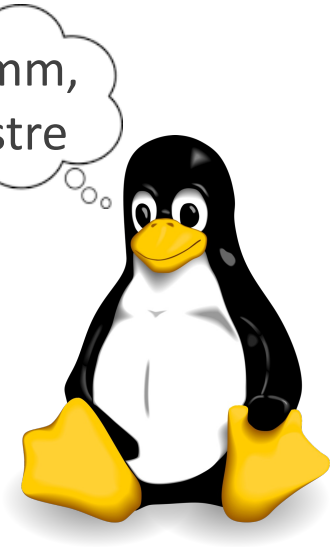
Ongoing ease-of-use and performance improvements for users and admins

- 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)
- Unaligned Direct IO (LU-13805 WC)

2.17
- Hybrid 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)



Improved Single Client Performance

Whamcloud

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

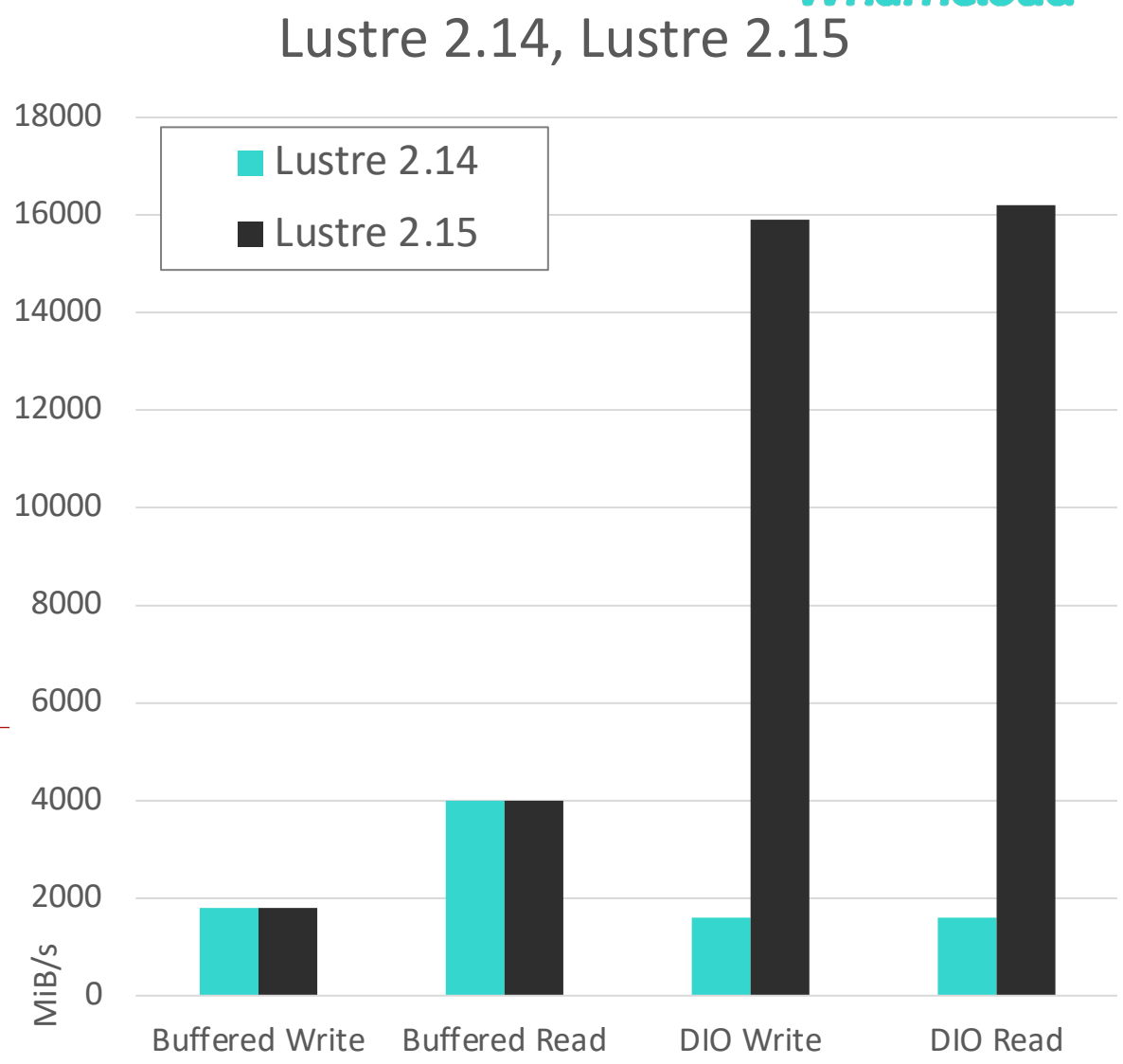
- **Improve parallel client readahead**
  - Parallel/unaligned readahead for single user thread (e.g. "**dd**") from 1.9GB/s->4.0GB/s
- **Async Direct IO** and **`io_uring`** on Linux 5.1+
  - Improved 4KB random IO via **`libaio`**, write 100k->266k IOPS; read 80k->610k IOPS
  - Non-POSIX async interface for IO and (evolving) metadata operations
- **Improved mmap readahead** chunk detection
  - Reduced pagefault latency, avg 512usec->52usec, max 37msec->6msec
- **Parallel/large Direct IO/Async IO performance**
  - Improve large *single-thread* `read()`/`write()`, reduced overhead, 700MB/s->17GB/s
- **Unaligned Large Buffered IO with Direct IO**
  - Avoid client page cache for IO > 1MiB, write 1.0GB/s->3.2GB/s, read 2.0->8.0GB/s
- **Improved client NUMA core binding**
  - Better distribution of RPCs from a single client/router across server CPU cores

Faster O_DIRECT and io_uring

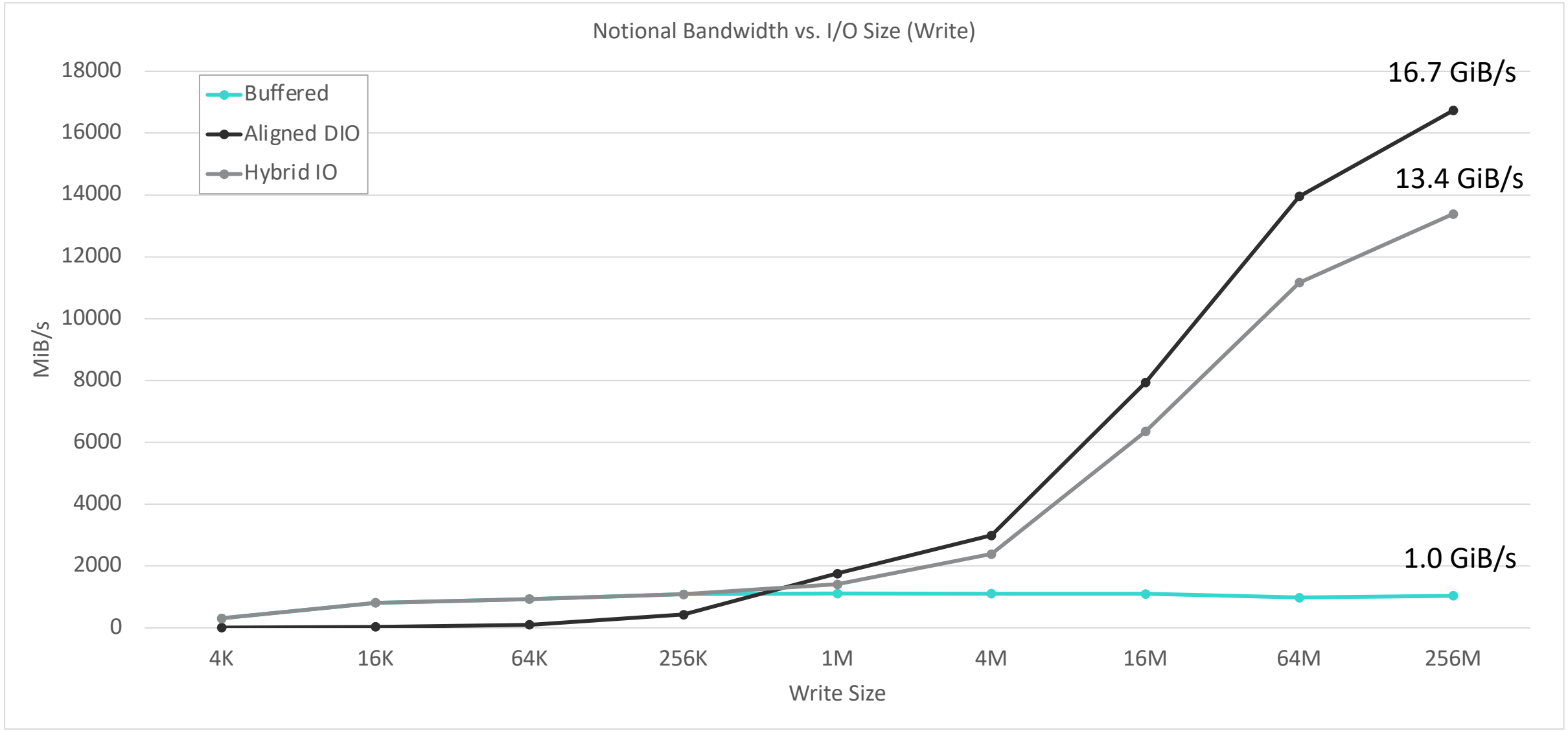
- Parallel large single thread O_DIRECT IO
  - 64MB read/write ~700MB/s->4GB/s
  - Able to leverage multiple IB interfaces
- Reduce O_DIRECT submission overhead
  - No per-page inode timestamp updates
  - Skip unnecessary page/request refcount
  - Improve page list handling, accounting, offsets
- Further DIO speedup from 4GB/s->**17GB/s+**

2.17

- Code under development has hit **50GB/s**!!!
  - With a single thread on a DGX client (8x IB)



Hybrid I/O: Where We're Headed (2.17)



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)

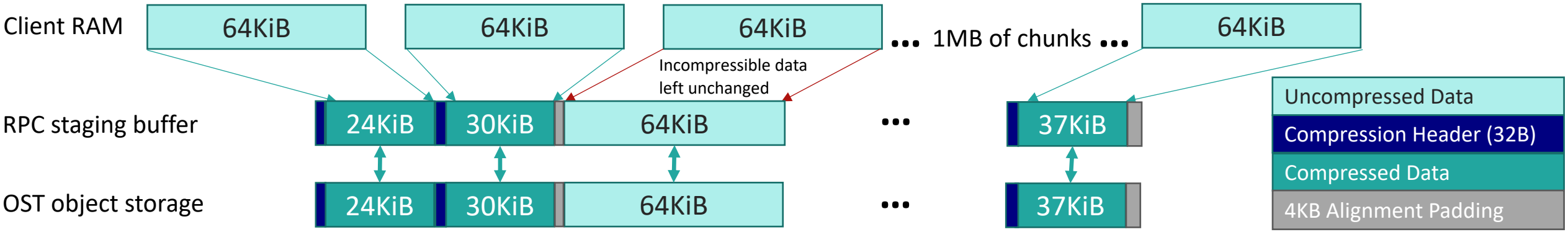
2.16
- Read-only mount enforced for nodemap clients (LU-15451 WC)
- Kerberos authentication improvements (LU-16630, LU-16646 WC, NVIDIA)
- Nodemap project quota mapping, squash all files to project (LU-14797 WC)
- Nodemap Role-Based Admin Controls (fscrypt, changelog, chown, quota) (LU-16524 WC)
- Cgroup/memcg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

Client-Side Data Compression (2.17+)

Whamcloud

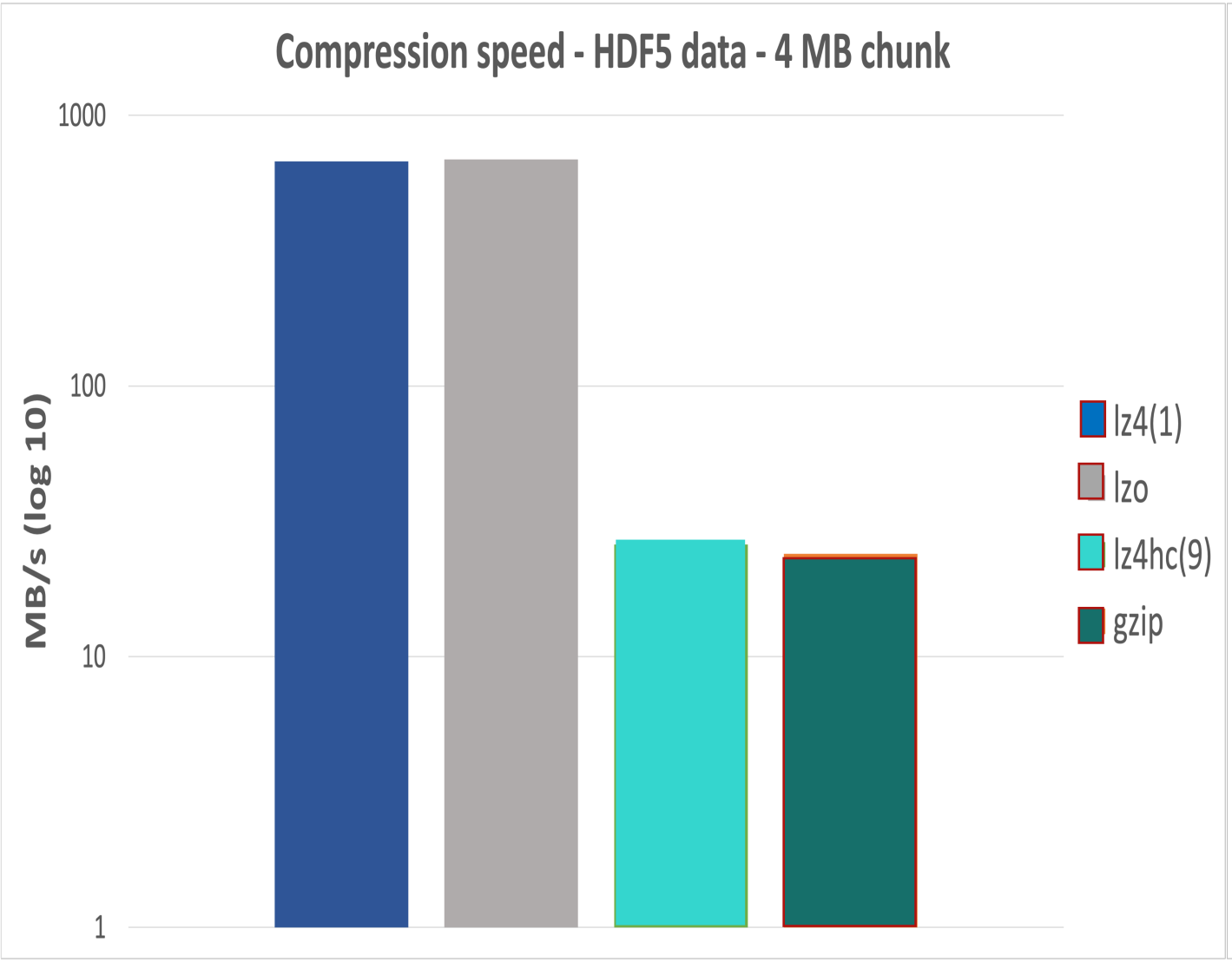
Increased capacity and lower cost for all-flash OSTs

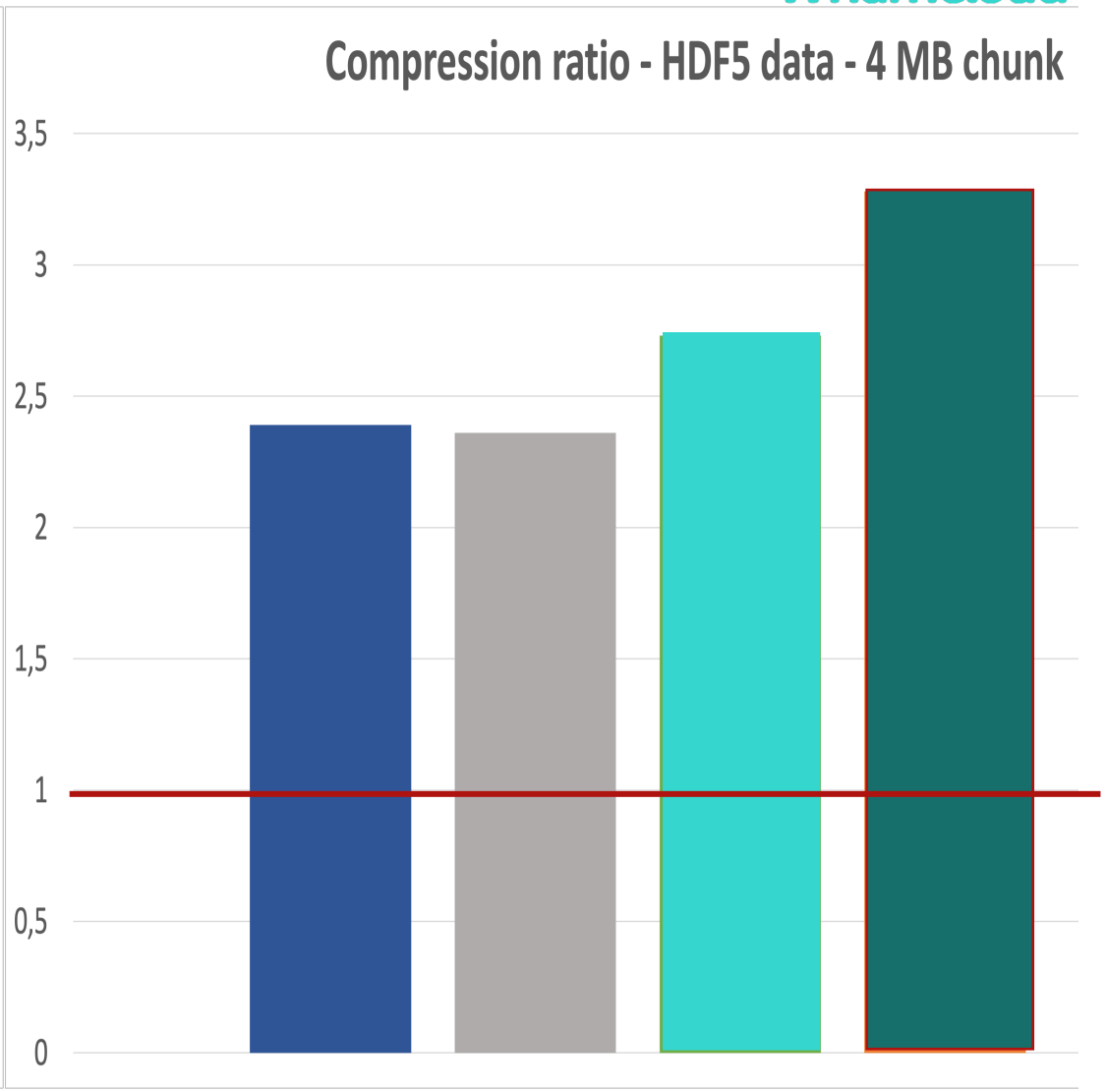
- Parallel compression of RPCs on client cores for GB/s speeds, **no server CPU overhead!**
- (De-)Compress (`lzo`, `lz4`, `gzip`,…) RPC on client in chunks (64KiB-1MiB+)
  - **Per directory or file component** selection of algorithm, level, chunk size (PFL, FLR)
  - Keep "uncompressed" chunks as-is 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)

Compression Algorithm Testing: HDF5 climate data





Server-side Capacity and Efficiency Improvements

Whamcloud

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

- 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 (LU-12916 WC)
  - Lower latency, write aggregation, no lock ping-ping
  - Use ext4 delayed allocation (delalloc) until write is large enough, default 64KiB
  - `ior-hard-write` improvement of **+50%** (+15% to overall Bandwidth score)

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)

Metadata Server Improvements (WC 2.15+)

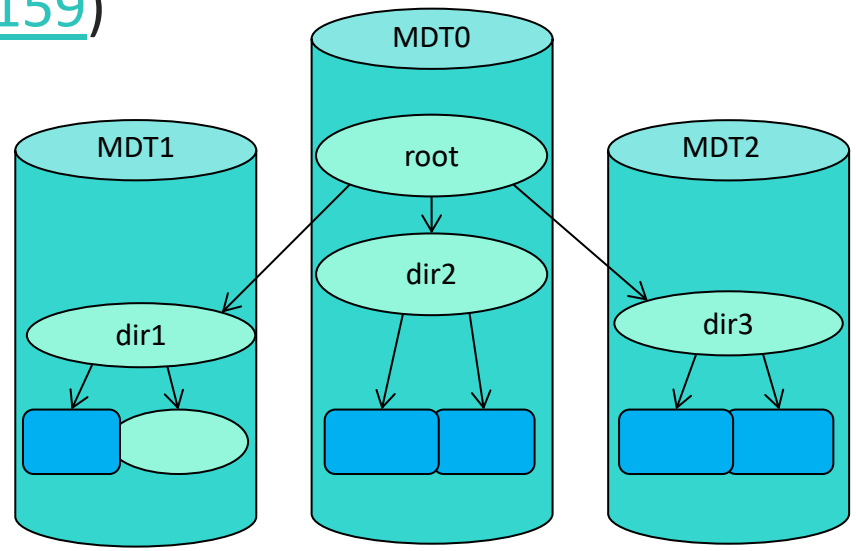
Whamcloud

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)

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



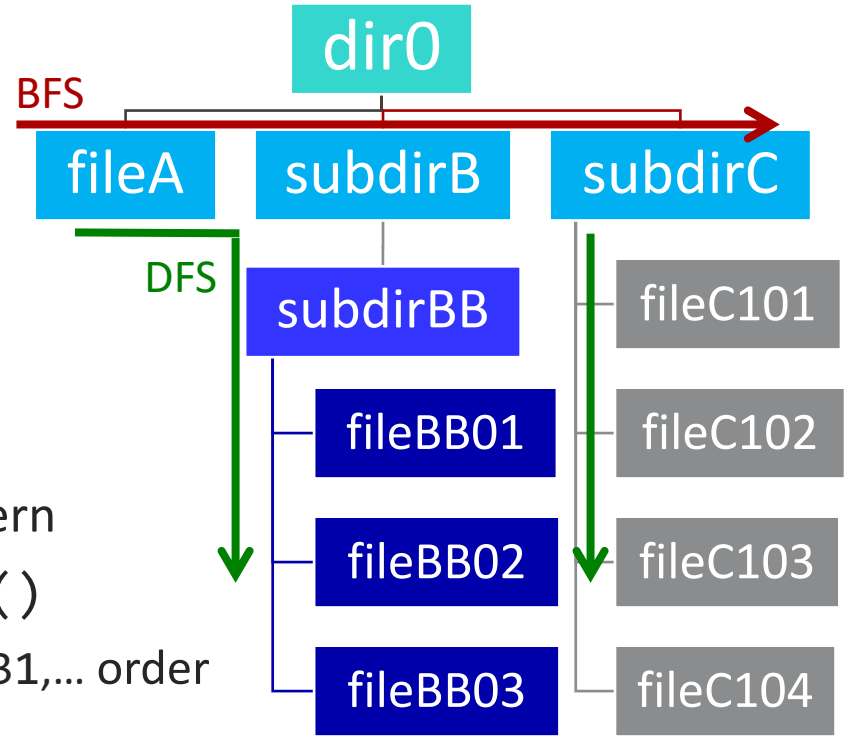
Batched Cross-Directory Statahead (WBC1) (WC 2.16)

Whamcloud

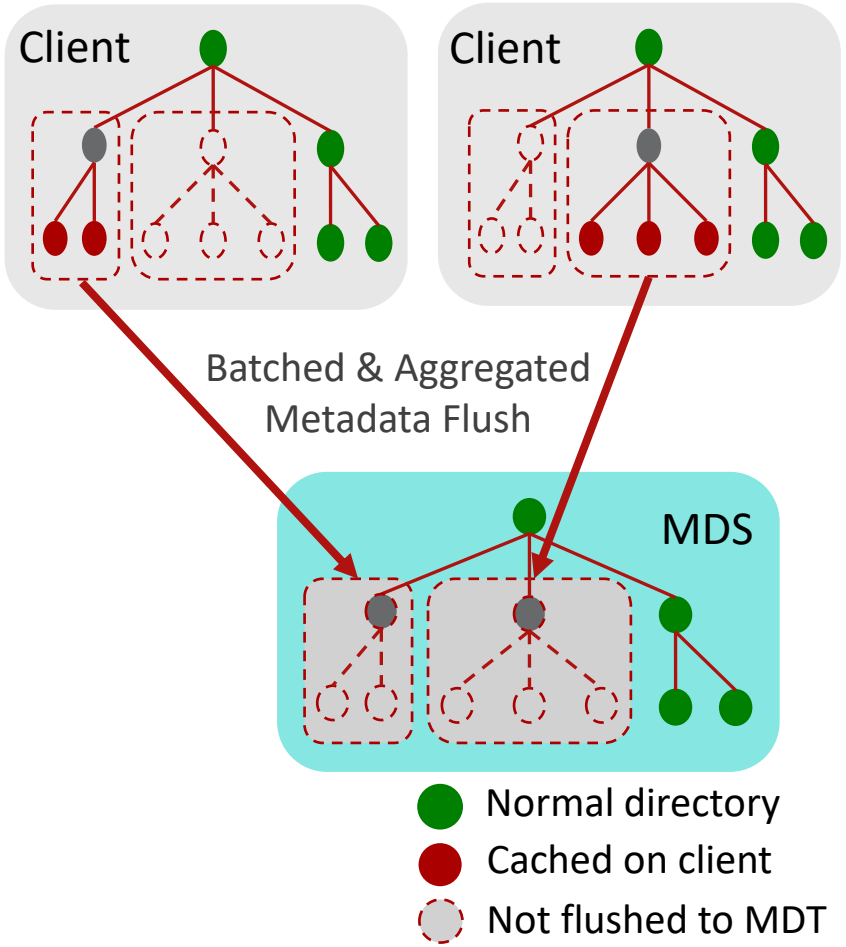
Improved access speed and efficiency for large directories/trees

- 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+)

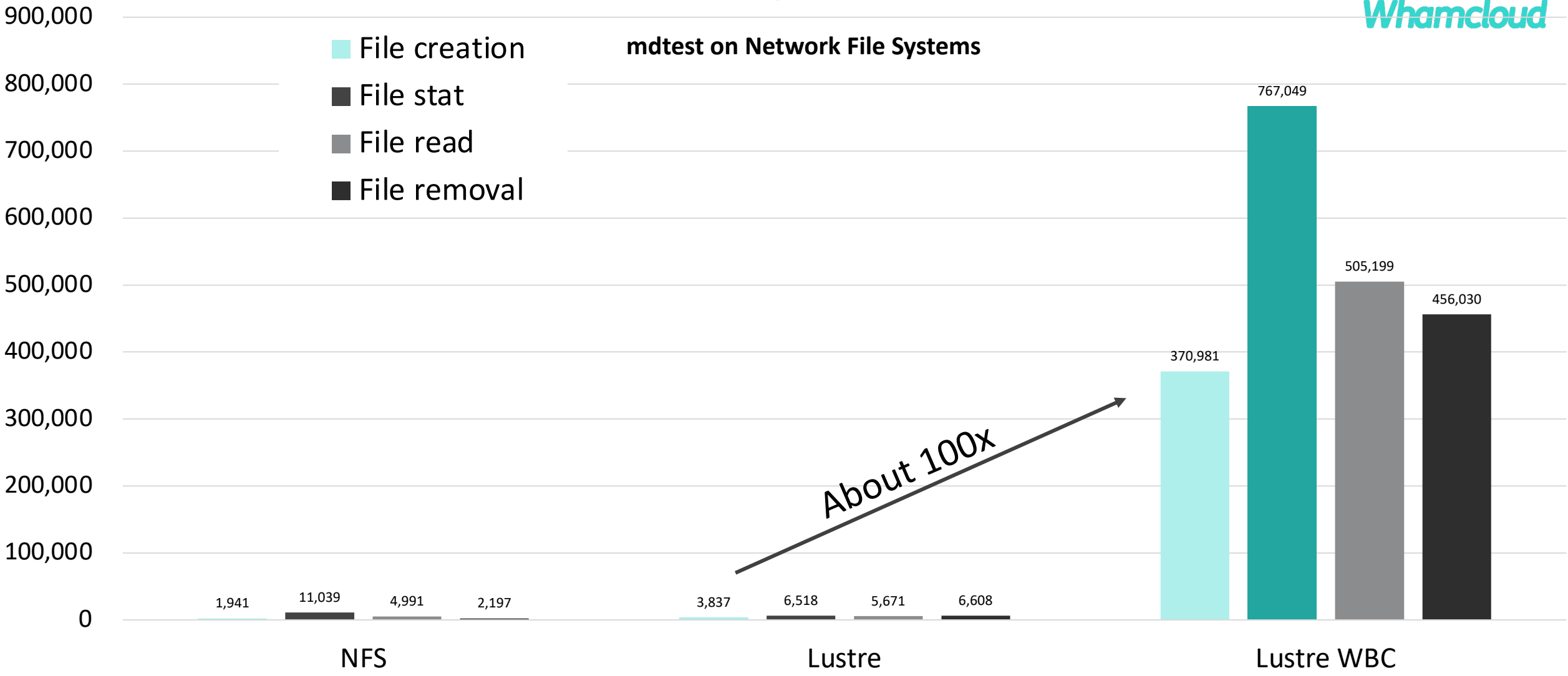


10-100x speedup for single-client create-intensive workloads

- 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

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

Client FLR Erasure Coded Files (2.18+, ORNL)

Whamcloud

- Erasure coding adds redundancy without 2x/3x mirror overhead
  - Improve data availability above hardware and network reliability
- Add erasure coding to new/old striped files ***after*** write done
  - Avoid overhead during initial application write
- For striped files - add N parity per M data *stripes* (e.g. 16d+3p)
  - Fixed **RAID-4** parity layout *per file*, declustered by file, CPU-optimized EC code (Intel ISA-L)
  - Parity declustering avoids IO bottlenecks, CPU overhead of too many parities
    - e.g. split 128-stripe file into 8x (16 data + 3 parity) with 24 total parity stripes

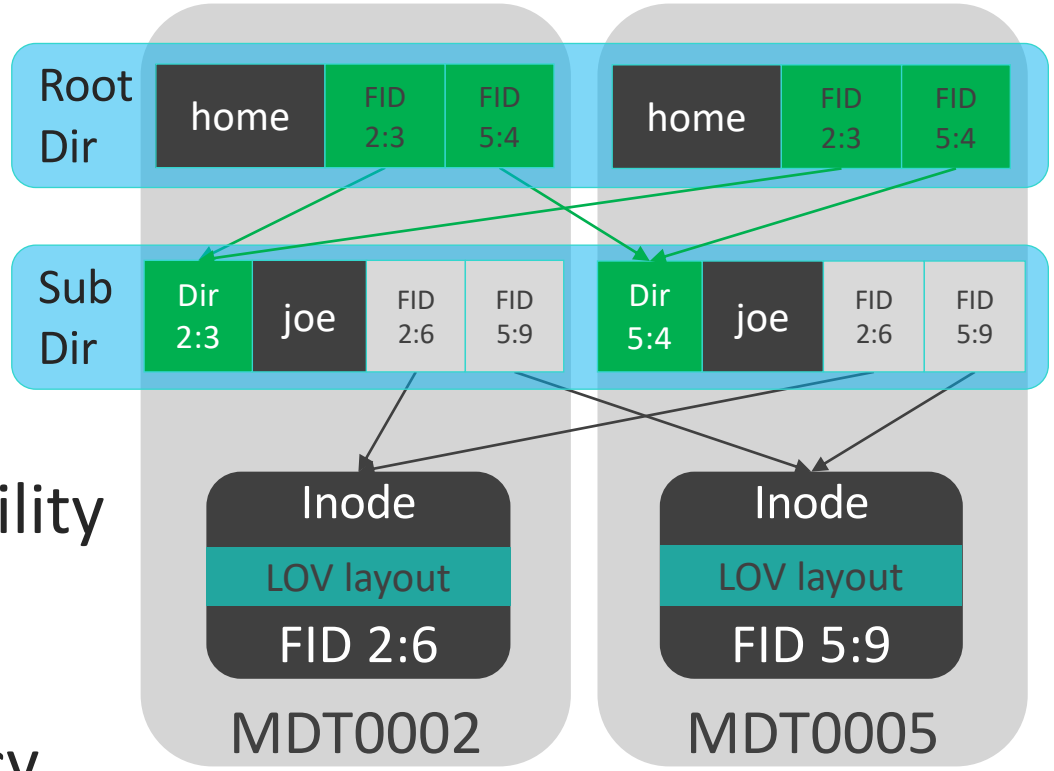
dat0	dat1	...	dat15	par0	par1	par2	dat16	dat17	...	dat31	par3	par4	par5	...
0MB	1MB	...	15M	p0.0	q0.0	r0.0	16M	17M	...	31M	p1.0	q1.0	r1.0	...
128	129	...	143	p0.1	q0.1	r0.1	144	145	...	159	p1.1	q1.1	r1.1	...
256	257	...	271	p0.2	q0.2	r0.2	272	273	...	287	p1.2	q1.2	r1.2	...

Lustre Metadata Redundancy (2.18+)

Whamcloud

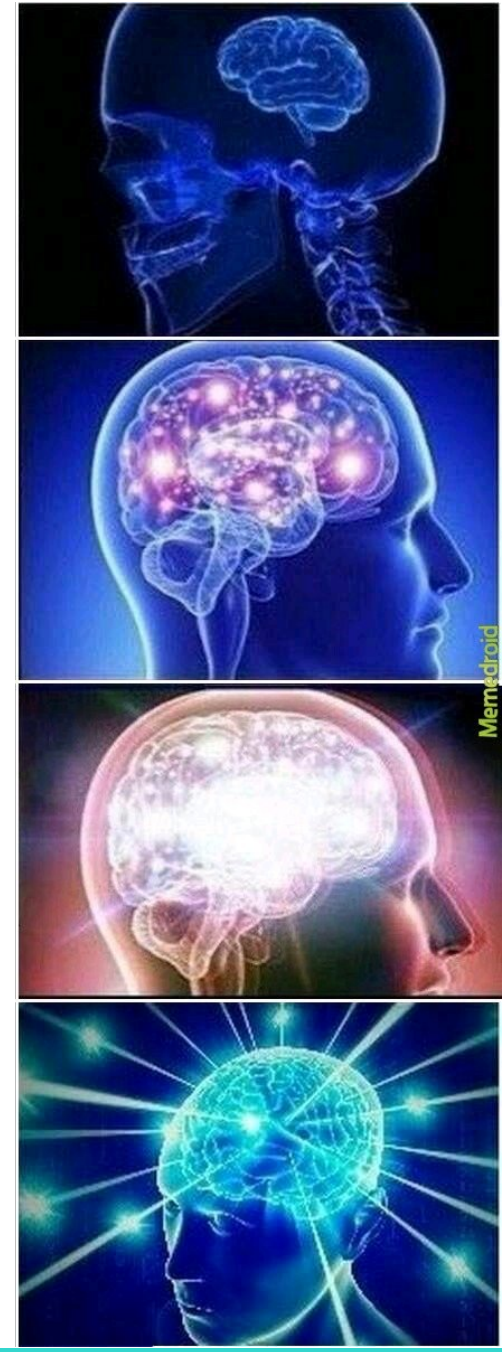
In early discussion and architecture stages

- LMR1a: Replicate services to other MDTs
  - Mirror FLDB, Quota, `flock()` scaling across MDTs
- LMR1b: DNE transaction performance
  - Remove excessive transactions ordering/sync
  - Improves **all** DNE operation performance
- LMR1c: Replicate top-level dirs for availability
  - **ROOT/** dir (rarely changed) mirrored over MDTs
  - No *per-file* metadata replication initially
- LMR2/3 phases needed for full redundancy
  - Full tree replication, inode replication, configurable per directory
  - Recovery, LFSCK, rebuild replicated directories after MDT loss



On the Evolution of IO Interfaces

- POSIX has been the standard IO interface for decades
  - Protects significant investment in developed applications and tools
  - Consistent behavior avoids need to chase interface-of-the-month
  - Data portability via protocol export (Lustre, NFS, SMB, S3, …)
- Optional API *extensions* for apps with special performance needs
  - Data continues to be accessible via standard APIs afterward
  - Specialized interfaces opt-in when/where applications need it
  - Applications can leverage extensions via common libraries or directly
- 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
- POSIX will continue to be the common interface going forward



Whamcloud

Thank You!

ddn