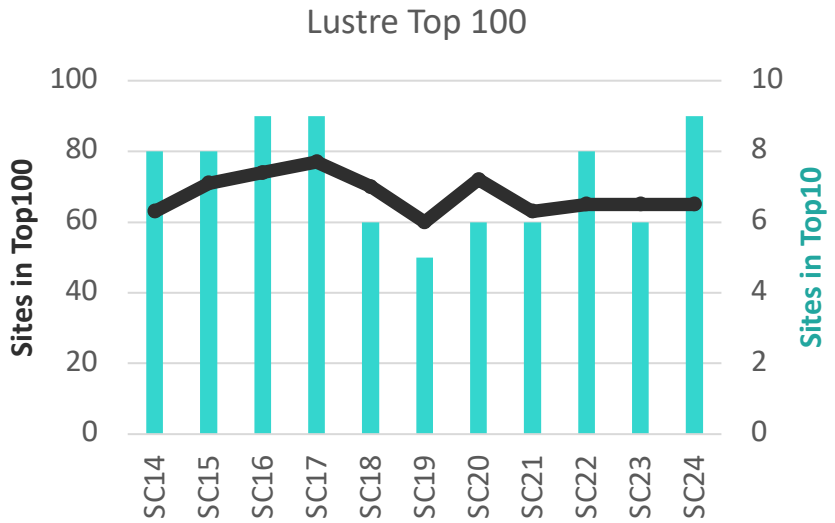
Whamcloud

Lustre 2.17 and Beyond

Andreas Dilger

Lustre Principal Architect

Lustre Committed to Exascale and the Future



- **The preferred choice for the world's largest systems**
  - Majority of Top10 and Top100 HPC systems use Lustre
  - World's largest AI/ML systems (Eos, Selene, X-AI, Scaleway)

  … or 2RU server for workgroup with 16 GPU nodes

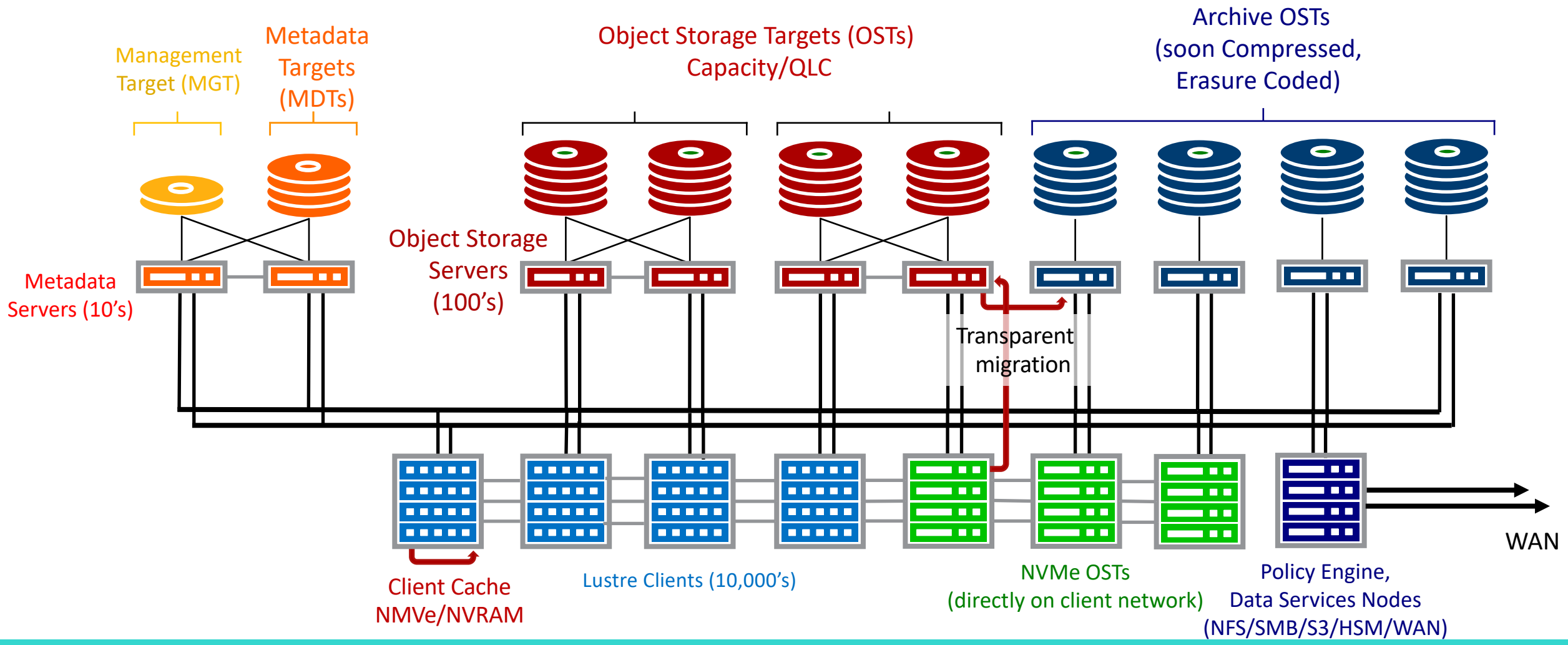
- **Scalability of servers and clients almost without limits**
  - 100M+ IOPS, 10 M+ metadata op/sec, 100B+ files
  - Capacity for any need – 10s TB/s read/write, 100s of PB today, 1 EB+ in the near future
  - Fully support large client nodes - 100s of cores, TBs of RAM, multi-400Gbps NICs, GPU RDMA

- **Continued improvements for large system deployments**
  - Steady feature development to meet evolving system and application needs
  - Virtualization of filesystem for multi-tenancy and data privacy

- **Improving ease-of-use, reliability, and efficiency**
  - Demand for fast and highly optimized storage is everywhere

Storage Management Across Entire Cluster

Data locality, with *direct client access* to all storage tiers



Planned Feature Release Highlights

Whamcloud

- **2.17** now open for major feature landings
  - **Hybrid IO Optimizations** – Hybrid BIO/DIO and server writeback cache (WC, Oracle)
  - **Dynamic Nodemaps** – ephemeral/hierarchical configuration for subdirectory trees (WC)
  - **File Level Redundancy - Erasure Coding** (FLR-EC) – M:N data redundancy (ORNL)
- **2.18** already has some features under development and detailed design
  - **Client-side Data Compression** – reduce network and storage usage/cost (WC, UHamburg)
  - **Fault Tolerant MGS** (LMR-FTM) – mirror MGS service and logs across MDS nodes
  - **Trash Can/Undelete** (TCU) – allow file recovery after accidental/malicious deletion
- **2.19** features under discussion for development
  - **Lustre Metadata Redundancy** (LMR1b) – MDT0000 services can run on other MDTs
  - **Metadata Writeback Cache** (WBC2) – single-client metadata speedup (WC)
  - **Lustre Metadata Redundancy** (LMR2a) – ROOT directory mirroring to other MDTs

LNet Improvements

Whamcloud

- IPv6 large NID support (LU-10391 SuSE, ORNL, HPE)
  - Demand for IPv6 in cloud deployments as IPv4 addresses are exhausted.

2.16

  - Also enables other large addresses (e.g. direct IB GUID instead of IPoIB)

2.17

- Mount without server NIDs in configuration logs (LU-10360, WC)
  - Servers register with MGS at mount, MGS IR Table sends server NIDs to clients
  - Clients can now (optionally) get server NIDs only from MGS IR Table, not config logs
  - Simplifies network setup, server migration/failover config, etc.
- Improve handling of MGS with many NIDs (LU-16738, WC)
  - Round-robin DNS records for multiple MGS NIDs at mount command-line
  - Display MGS hostname instead of NIDs for "`mount`" and "`df`" output
- Improve network transfer for sparse reads (LU-16897, WC)
  - Don't send holes/zero pages over network when no blocks allocated
- Remote LND configuration discovery for EFA LND (LU-18808, AWS)

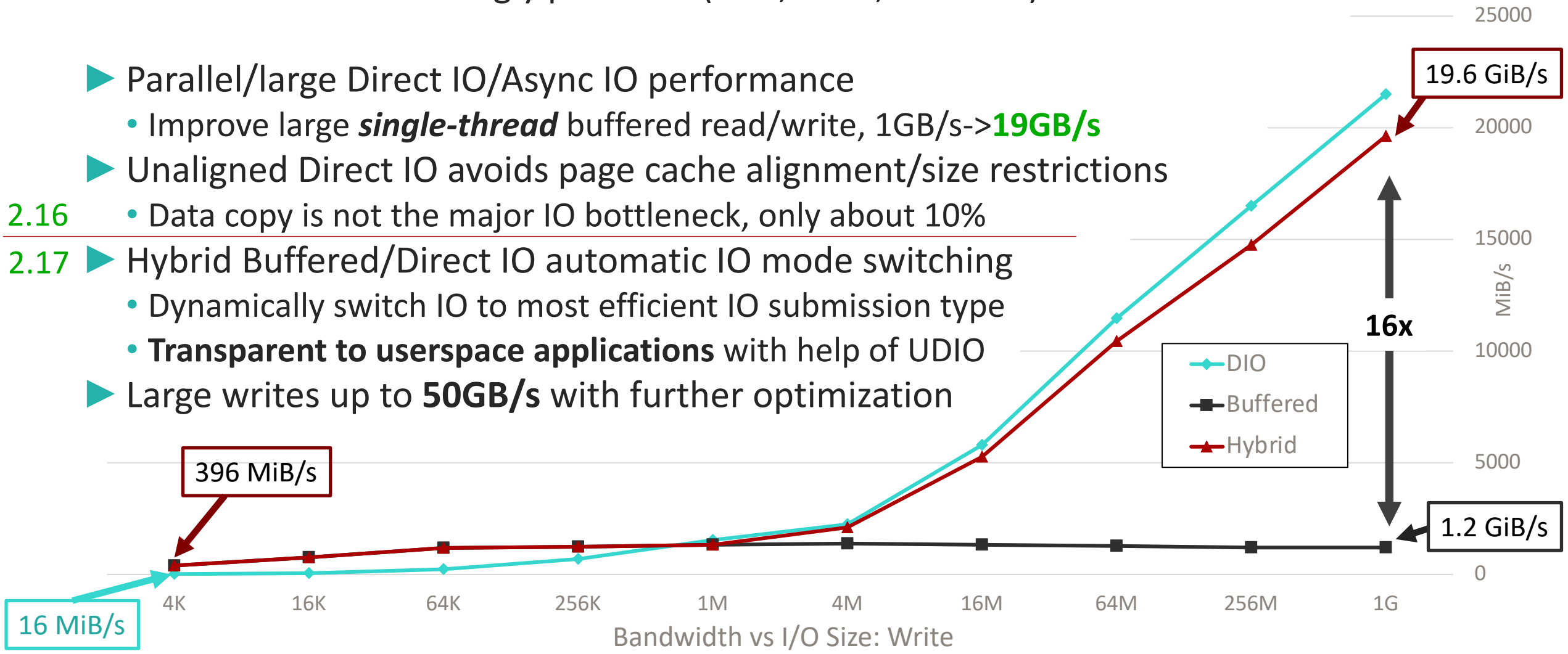
Hybrid IO: Improved Application IO Performance (2.17)

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

- Parallel/large Direct IO/Async IO performance
  - Improve large ***single-thread*** buffered read/write, 1GB/s->**19GB/s**
- Unaligned Direct IO avoids page cache alignment/size restrictions

2.16
  - Data copy is not the major IO bottleneck, only about 10%

2.17
- Hybrid Buffered/Direct IO automatic IO mode switching
  - Dynamically switch IO to most efficient IO submission type
  - **Transparent to userspace applications** with help of UDIO
- Large writes up to **50GB/s** with further optimization



Client-Side User Tools Improvements

Whamcloud

Sometimes it's the small things that make a big difference

- **`lfs df --mdt/--ost`** shows only MDT or OST devices (LU-17516 WC)
- **`lfs find -xattr/attr`** finds files with specific attributes (LU-15743 LU-16760 LANL)
- **`lfs find -printf/ls`** to better display found files (LU-7495 LU-15504 LANL, WC)
- **`lfs setstripe -C -N`** creates N (over)stripes per OST, up to 32x (LU-16938 HPE)

2.16

- **`lctl list_param --path`** prints full pathname for data scraping (LU-17343 WC)

2.17

- Numerous man-page improvements for `lfs` and `lctl` sub-commands (LU-4315 WC)
- Allow split **`lctl/lfs`** sub-commands (e.g. **`mirror_foo`** -> **`mirror foo`**) (LU-18114 WC)
- **`lctl get/list_param --merge`** aggregates similar output lines (LU-14590 WC)
- **`mount.lustre`** sets client-local parameters from `/etc/lustre` (LU-11077 WC)
- **`lfs find/project/quota`** allows named projects from `/etc/projects` (LU-13335 WC)
- **`lfs migrate/mirror`** allow filenames/FIDs from file (LU-18454 WC)

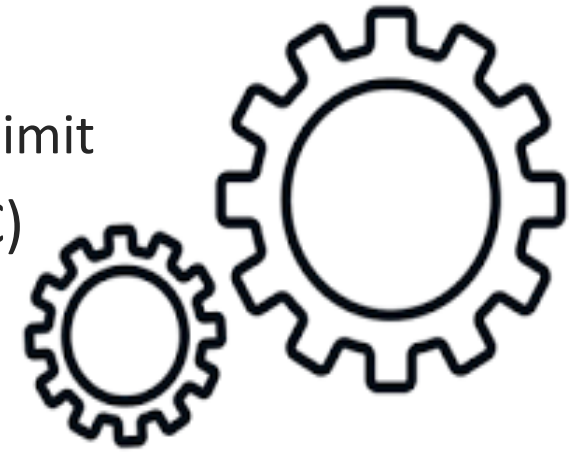
Client-Side Functionality Improvements

Whamcloud

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

- Ongoing code updates/cleanup for upstream 6.x kernels (ORNL, HPE, WC, SuSE)
- Allow specifying CPU cores to exclude from CPT list (LU-17501 WC)
  - `options libcfs cpu_pattern="C[0,1]"` skips cores on each NUMA node
- Obfuscate file/dir names in debug logs to limit PII release (LU-18810 WC)
- 2.17 Reduce RPC latency for client IO via CPU power states (LU-18446 NVIDIA)

- 2.18 Project quota aggregation/nesting (LU-18222 WC)
  - Like OST Pool Quotas, allows PROJIDs to be “nested” into larger PROJID limit
- Client-side performance stats via statfs for each target (LU-7880 WC)
- FLR Erasure Coded files with delayed resync (LU-10911 ORNL)
- Discussed upstreaming client into Linux kernel (ORNL, AWS)



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-15011 WC)
- `lljobstat` utility for easily monitoring "top/bad" jobs on MDT/OST (LU-16228 WC)

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

2.17
- Hidden and read-only attributes for files/dirs (LU-18615 AWS)
- Network Request Scheduler TBF (QOS) improvements
  - Rules by nodemap name (LU-17902 WC)
  - Rules by project ID (LU-17166 CEA, WC)

2.17
  - Rules with UID/GID/PROJID ranges (LU-18509 WC)

2.18
- Default NRS TBF rule(s) to keep “bad” jobs in check out of the box (LU-17296)
  - Help avoid “noisy neighbor” out of the box, and further tunable as needed
- Enable default PFL layout on newly-formatted filesystems (LU-11918)
  - Simplify usage and improve performance for most configurations

Server-side Capacity and Efficiency Improvements

Whamcloud

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

- More rename locking optimizations (Spark, …)
  - Reduce BFL lock hold time via speculative trylock (LU-17427 WC)
- Memory reduction for LDLM, JobStats (multiple)
- `flock` performance and scalability improvements (LU-17276 WC, SUSE)
- Read and return small directory contents on open (LU-18448 HPE)
- **RAM-based OSD backend** for testing/shared memory (LU-17995, LU-18813 AWS, WC)

2.17
  - Initially useful for API testing, code consolidation, benchmarking

2.18
  - Potentially useful for ephemeral workloads needing very large shared memory
- **OST 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 OST write is large enough, default 64KiB
  - Dynamic cache selection, complementary with client Hybrid Buffered/Direct IO

Ongoing ldiskfs and e2fsprogs Improvements

Whamcloud

- Persistent `TRIMMED` flag on block groups during `fstrim` (LU-14712 WC)
  - Avoid useless `TRIM` commands on device after reformat and remount
- More efficient ldiskfs mballoc for large filesystems (LU-14438 Google, IBM, WC)

2.17

  - Backport improved list-/tree-based group selection from upstream kernel

2.18

- Hybrid ldiskfs LVM storage devices (NVMe+HDD) (LU-16750 WC)
  - Allow storing metadata on NVMe at start of device, data on HDDs at end of device
- Enable ldiskfs delayed allocation for writeback cache (LU-12916 WC)
  - Allow aggregating small writes in server RAM instead of read-modify-write to client
- Parallel e2fsck for pass2/3 (directory entries, name linkage) (LU-14679 WC)

Improved Data Security and Multi-tenancy

Whamcloud

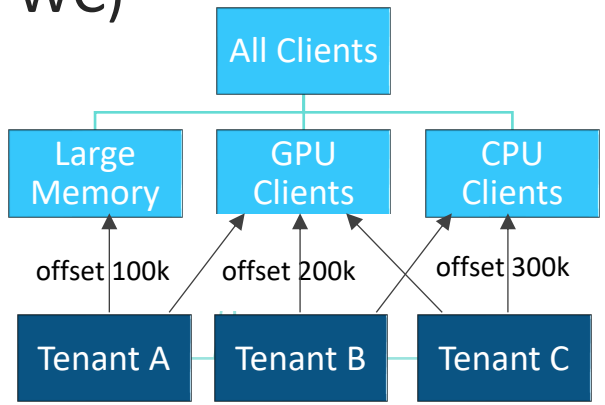
Increasing demand to isolate users and their data for legal/operational reasons

- Nodemap ID offset range for UID/GID/PROJID mapping (LU-18109 WC)
- Kerberos fixes and improvements (multiple NVIDIA, WC)
- Client-local `root` capability in nodemap (LU-18694 WC)
  - Allow root operations (chown, quota) within ID offset range
- Dynamic/hierarchical nodemap configuration (LU-17431 WC)
  - In-memory nodemap configuration for short-lived group (batch job)
- Multiple and read-only filesets per nodemap (LU-18357 WC)
- Configurable capabilities mask per nodemap (LU-17410 WC)
  - Defaults to all client capabilities disabled for security

2.17

2.18

- Encrypted fscrypt backup/restore without key (LU-16374 WC)



Client FLR Erasure Coded Files (2.17+, ORNL)

Whamcloud

- Erasure coding adds data redundancy without 2-3x mirror overhead
  - Improve data availability above hardware and network reliability
  - Initial target for large read-mostly files, adds redundancy/availability with low cost
- Add erasure coding to new/old striped files ***after*** write completed
  - Delayed redundancy avoids overhead during initial application write
- For large striped files - add N parity per M data *stripes* (e.g. 8d+2p or 16d+3p)
  - Fixed **RAID-4** parity layout *per file*, declustered by file, CPU-optimized EC code (Intel ISA-L)
  - EC RAID pattern independent of number of stripes in file, works with PFL layouts
  - Can survive full OST loss with minimal recovery delay, transparent to applications

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	…

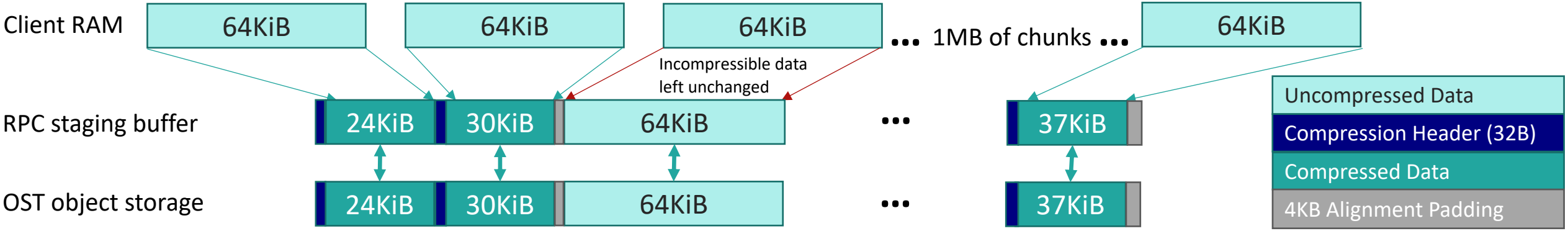
Existing stripes… Parity stripes added… Existing stripes… Parity stripes added…

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, **reduces server CPU load**
- (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
- Could write uncompressed to one FLR mirror for random IO pattern
  - Data (re-)compression during mirror/migrate to second mirror on slow tier (via data mover)

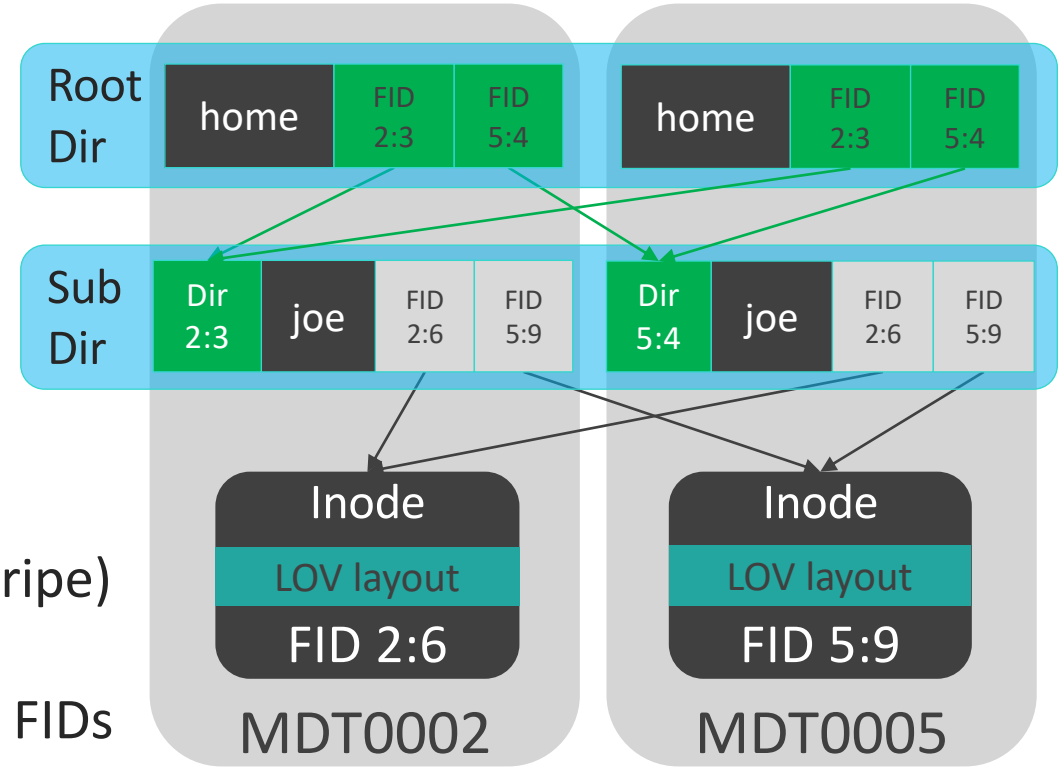
Trash Can/Undelete for Files and Directories (2.18 WC)

Whamcloud

- Allow files/directories to be undeleted after `rm`/`rmdir`
  - Rescue users from fat-finger mistakes or malicious scripts
  - Handle "`rm -r`" properly to allow whole-tree recovery
- Deleted files in trash are flagged and treated specially
  - Removed from user/group/project quota and `df` usage
  - Files cannot be read to avoid abuse, and apps know files are deleted
- Virtual `.Trash` directory visible in every directory
  - Can use normal tools to list and recover files or directories
  - `.Trash` is hidden from normal directory listing
- Users can view and recover their own files
  - Configurable expiry time before cleanup (e.g. max age = 7d)
  - Configurable filesystem fullness threshold (e.g. 80% full)
  - More sophisticated cleanup policy in userspace (e.g. by user, project, nodemap)

Lustre Metadata Redundancy (LMR) (2.18+)

- LMR1a: Fault Tolerant MGS (LU-17819)
- LMR1b: Replicate services to other MDTs
  - Mirror FLDB, Quota, `flock()` scaling over MDTs
- LMR1c: DNE performance (LU-7426)
  - Improve llog transaction ordering/sync
  - Improves **all** DNE operation performance
- LMR2: Replicate top-level dirs for availability
  - 2a: ROOT/ (rarely changed) mirrored to other MDTs
  - 2b: Subdirectories mirrored to 2+ MDTs (via setdirstripe)
  - 2c: Per-file metadata replication in a later stage
  - 2d: `e2fsck` to allow directory entries with multiple FIDs
- LMR3: Complex recovery handling
  - Write to directory while mirror offline, full MDT rebuild
- LMR4: LFSCK to repair/rebuild inconsistent mirrors

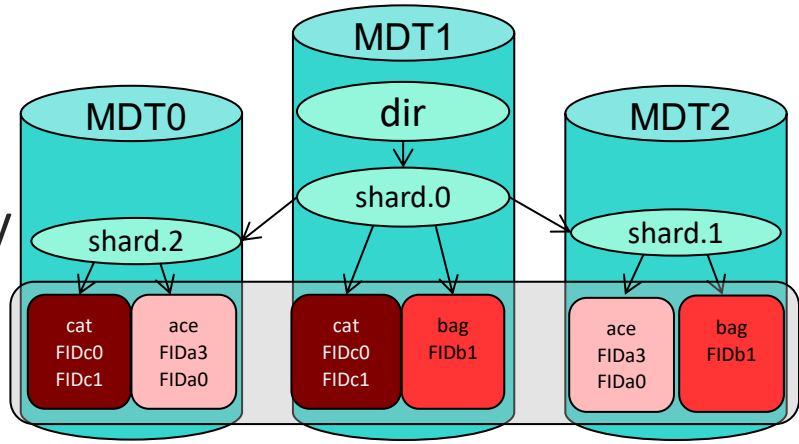


Fault Tolerant MGS (LMR-FTM) (2.18 WC)

Whamcloud

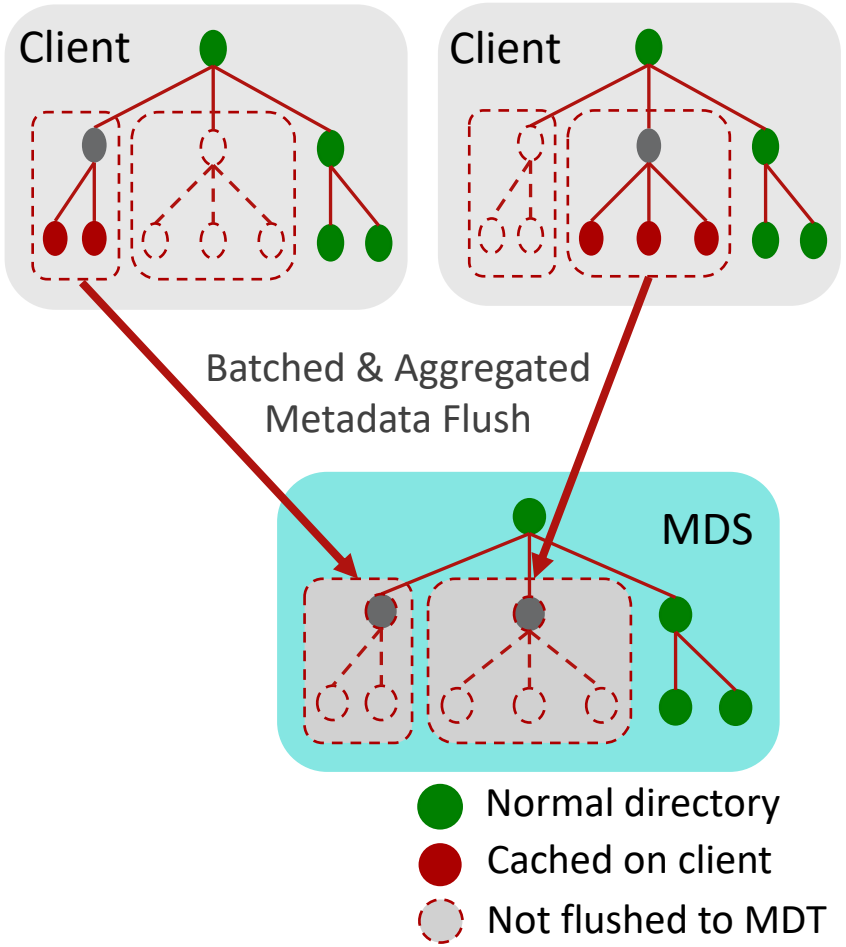
- Run MGS service on multiple MDS nodes for availability (LU-17819)
  - Allow clients to get config llogs from **any MGS node**
  - Reduces mount time/timeouts, distributes load in large clusters
  - MGS Imperative Recovery even if “primary” MGS node restarts
- Mirror MGS config logs to remote MDTs for redundancy
  - Use RAFT Consensus algorithm to coordinate MGS cluster
  - MGS Leader election, heartbeat, consistent log updates
  - Append-only logs, matches existing MGS config llog format

MDT0000 removal/replacement (LMR1b/2a) (2.18+)



- Remove critical service dependency on MDT0000
- Mirror FLDB to all servers (maps FID->MDT/OST)
  - Table is relatively small (few lines per target) and changes rarely
- Quota Manager migrated/mirrored to multiple servers
  - Hash distribution by UID/GID/PROJID, easily parallel operations
- LDLM `flock()` distribution and scaling over multiple MDS nodes
  - The `flock()` locking is not really related to MDT inodes, so can be anywhere
  - Need to handle deadlock detection, but is disjoint for process groups
- Mirror `ROOT/` directory to multiple MDTs
  - Rarely changes for most uses, some update overhead is acceptable
- Enough to allow removing/replacing MDT0000 on live system

Metadata Writeback Cache (WBC2) (WC 2.18+)

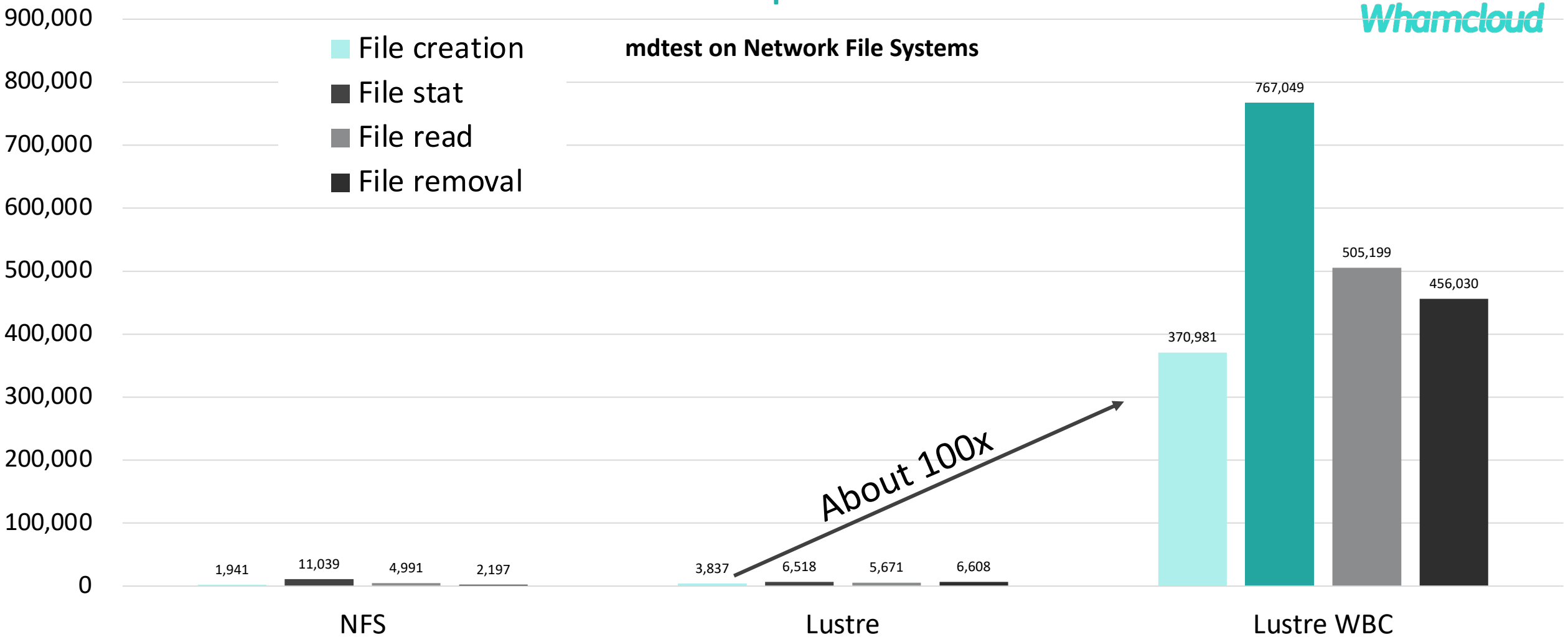


10-100x speedup for single-client create-intensive workloads

- 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

On the Evolution of IO Interfaces

- POSIX has been the standard IO interface for decades
  - Protects substantial investment in developed applications and tools
  - Avoids applications chasing interface-of-the-month and expensive rewrites
- ***Opt-in API extensions*** for apps with special performance needs
  - Relaxed semantics/interfaces when/where applications need/understand it
  - Avoids issues with changes in behavior - ***which* subset of POSIX is OK?**
  - Data stored and continues to be accessible via standard APIs afterward
  - Applications can leverage extensions via common libraries or directly
  - *Should we add an optimized native KV interface to Lustre clients?*
- Continuous underlying optimizations speed up existing applications
  - Unaligned/Hybrid IO, cross-dir/file prefetch, WBC for creates
  - Asynchronous meta/data ops via Linux `io_uring`, *batched file create API*
- POSIX will continue to be the most common interface going forward

Whamcloud

Thank You!

ddn