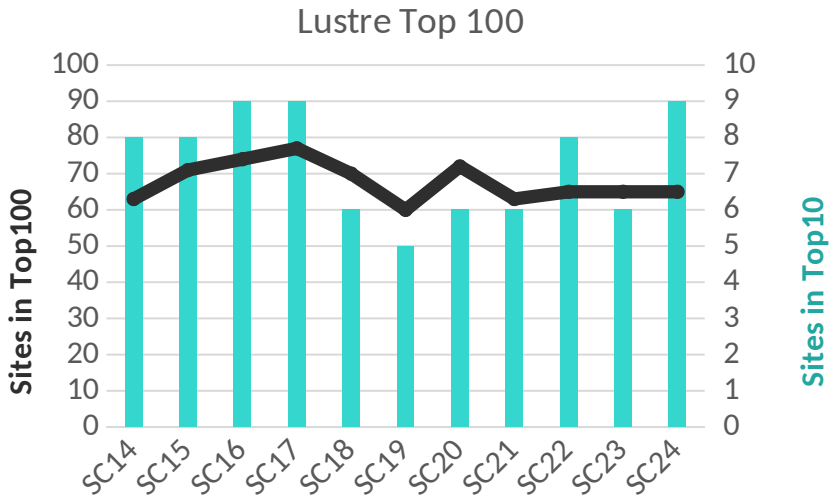
Whamcloud

Lustre 2.17 and Beyond

Andreas Dilger

Lustre Principal Architect

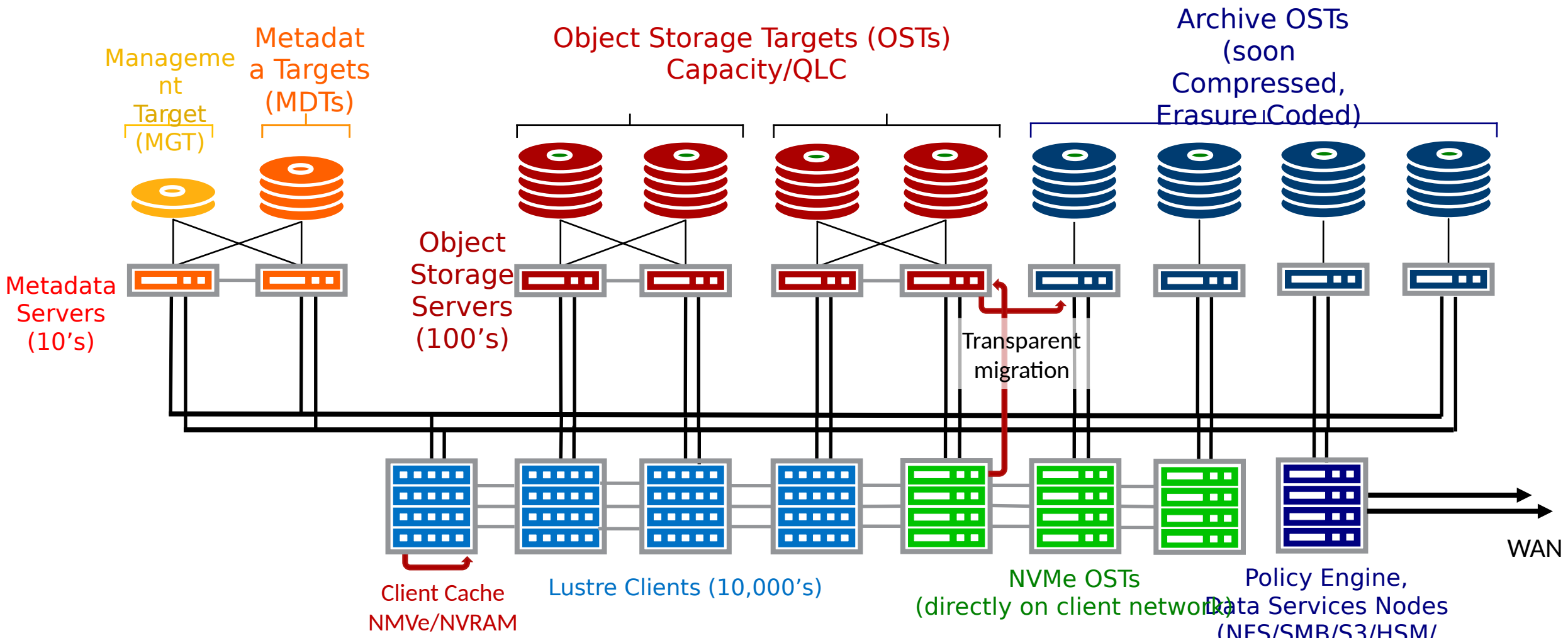
Lustre Committed to Exascale and the Future



- The preferred choice for the world's largest systems
  - Majority of Top10 and Top100 HPC systems use
  - World's largest AI/ML systems (xAI, NVIDIA, Mis
  - … or 2RU server for workgroup with 16 GPU
- Scalability of servers and clients almost without limits
  - 100M+ IOPS, 10 M+ metadata op/sec, 100B+ f
  - Capacity for any need – 10s TB/s read/write, 500 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 user isolation and data privacy
- Improving ease-of-use, reliability, and efficiency

Storage Management Across Entire Cluster

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



Planned Feature Release Highlights

Whamcloud

- **2.17** major feature landings complete, release stabilization underway
  - **Hybrid IO Optimizations** – Hybrid BIO/DIO and server writeback cache (WC, Oracle)
  - **Nodemap Improvements** – manage, virtualize, isolate filesystem tenants (WC)
  - **Dynamic LNet NID Configuration** – simplify and improve dynamic network config (WC)
- **2.18** already has several features well under development
  - **File Level Redundancy - Erasure Coding** (FLR-EC) – M:N data redundancy (WC, ORNL)
  - **Client-side Data Compression** – reduce network and storage usage/cost (WC)
  - **Fault Tolerant MGS** (LMR-FTM) – replicate MGS config management across MDS (WC)
  - **Trash Can/Undelete** (TCU) – allow file recovery after accidental/malicious deletion (WC)
- **2.19** features under discussion for development
  - **Lustre Metadata Redundancy** (LMR1b) – replicate quota, flock, … services to other MDS
  - **Metadata Writeback Cache** (WBC2) – single-client metadata speedup (WC)
  - **Lustre Metadata Redundancy** (LMR2a) – R00T/ directory mirroring to other MDTs

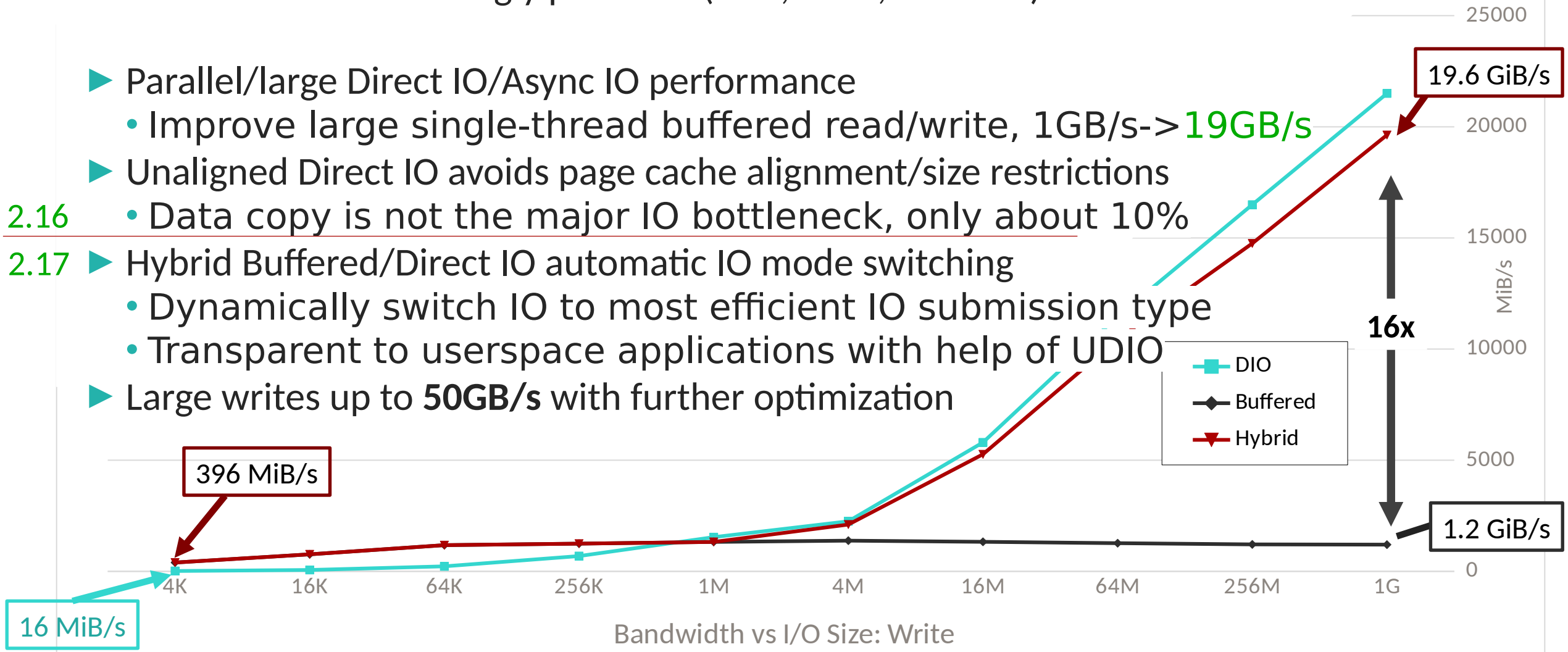
Hybrid IO: Improved Application IO Performance

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

2.16

2.17



Improved Data Security and User Isolation (2.17 WC)

Whamcloud

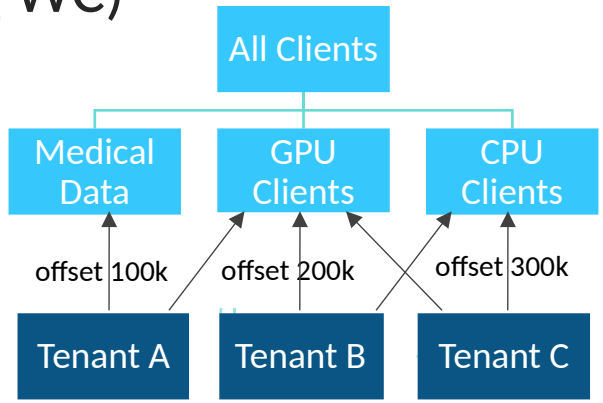
Increasing demand to isolate users and their data for legal/operational reasons

- Nodemap ID offset range for UID/GID/PROJID mapping (LU-18109 WC)
- Kerberos fixes and improvements (multiple NVIDIA, WC)
- Client-local `root` chown/quota within nodemap (LU-18694 WC)
- Dynamic/hierarchical nodemap configuration (LU-17431 WC)
- 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
- Ban clients from filesystem via nodemap (LU-19157 WC)

2.17

2.18

- Nodemap client identification by SSK/Kerberos Key (LU-18357 WC)
- Encrypted fscrypt backup/restore without key (LU-16374 WC)



Dynamic NID Configuration (2.17 WC)

- Servers can configure thousands of NIDs for tenant VLANs (LU-xxxx)
- Mount client without server NIDs in configuration logs (LU-10360, WC)
  - Servers register with MGS at mount time, MGS IR Table sends server N… clients
  - Clients can now (optionally) get server NIDs only from MGS IR Table, not config logs
  - Simplifies network setup, server migration/failover config, dynamic configuration
- Notify clients when server NIDs dynamically added/removed (LU-xxxx)
  - Filtered for clients that cannot access the new LNet due to VLAN

2.17

2.18

- Improve handling of MGS with many NIDs (LU-16738)
  - Display MGS hostname instead of NIDs in `/proc/mounts`
  - Hide ugly list of MGS NIDs for “mount” and “df” output
  - Round-robin DNS records for multiple MGS NIDs at mount command-line

LNet Improvements

- Finishing IPv6 NID support (LU-18417 ORNL, HPE, WC)
  - Integration of remaining IPv6 functionality with various features
- Allow server to configure hundreds of NIDs for isolated tenant VLANs (LU
- EFA optimized LND for Amazon FSX (LU-18808, AWS)
- Improve network transfer for sparse reads (LU-16897, WC)
  - Don’t send holes/zero pages over network when no blocks allocated

2.17

2.18

Client-Side User Tools Improvements

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

Whamcloud

- **`lfs`**, **`lctl`**, and **`llapi`** manual page improvements (LU-4315 … WC)
- **`lfs df`** new options to print specific fields, specific MDTs/OSTs (LU-18242 WC)
- **`lfs fid2path`** lookup for OST object FIDs (LU-13527 WC)
- **`lfs find -printf/-ls/-xattr`** options (LU-15743 LU-16760 LU-7495 LANL, WC)
- **`lfs find/project/quota`** allows named projects in `/etc/projid` (LU-13335 WC)
- **`lfs find`** multi-threaded parallel scanning (LU-17814 WC)
- **`lfs migrate/mirror`** allow passing filename/FID list from file (LU-18454 WC)
- **`lfs mkdir/setstripe –C -N`** creates N (over)stripes (LU-16938 LU-18244 HPE)
- **`lctl get/list_param`** show **`--path`**, **`--merge`** lines (LU-17343 LU-14590 WC)
- **`mount.lustre`** sets client-local parameters from `/etc/lustre` (LU-11077 WC)
- **`mount.lustre`** automatically loads SSK key from mountpoint (LU-11077 WC)

Client-Side Functionality Improvements

Whamcloud

Ease-of-use and performance improvements for end users and admins

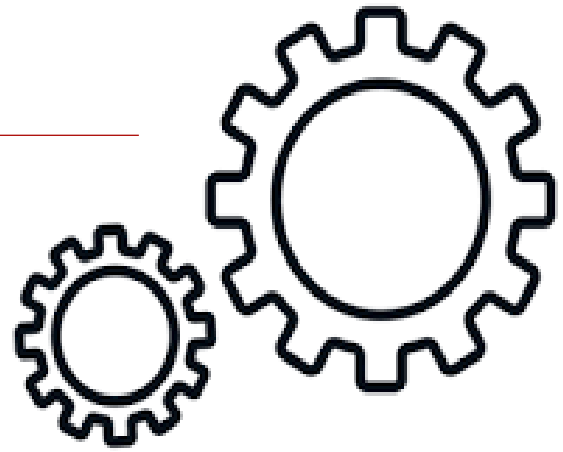
- Ongoing updates/cleanup for 6.x kernels and later upstreaming (ORNL, HPE, WC, SuSE)
  - Kernel 6.9 folios (page extents) allow efficient handling large IO/cache (LU-17916 HPE)
- Allow specifying CPU cores to exclude from CPT list (LU-17501 WC)
  - `options libcfs cpu_pattern=C[a,b-c]` skips NUMA cores, `"X[a,b-c]"` skips any cores
- Obfuscate file/dir names in debug logs to limit PII exposure (LU-18810 WC)
- Cache statfs() on project directory (LU-16771 WC)
- Reduce RPC latency for client IO via CPU power states (LU-18446 NVIDIA)

2.17

- Regexp to exclude mkdir from DNE3 MDT space balance (LU-19268 WC)

2.18

- Client-side performance stats via statfs for each target (LU-7880 WC)
- FLR Erasure Coded files with delayed resync (LU-10911 WC, ORNL)
- Lustre Quota Aggregation for ID ranges (LU-18222 WC)



Ongoing ldiskfs and e2fsprogs Improvements

Whamcloud

- More efficient ldiskfs mballoc for large filesystems (LU-14438 Google, IBM, WC)
  - Backport improved list-/tree-based group selection from upstream kernel
- Persistent `TRIMMED` flag on block groups avoids full fstrim on remount (LU-17980 WC)
- New "-o `fstrim`" feature for continual smart TRIM operations (LU-14712 WC)
- Export `fstrim` statistics per filesystem (LU-19158 WC)
- Patchless T10-PI integration between RPC and storage checksum (LU-10472 WC)
- Hybrid ldiskfs LVM storage devices (NVMe+HDD) (LU-16750 WC)

2.17

2.18

  - 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)

Server-side Capacity and Efficiency Improvements

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
- 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)
  - Initially useful for API testing, code consolidation, benchmarking
  - Potentially useful for ephemeral workloads needing very large shared memory

2.17

2.18

- **OST writeback cache** for small, lockless, direct writes (LU-12916 WC)
  - Lower latency, small write aggregation, no lock ping-pong
  - Use ldiskfs delayed allocation ("`-o delalloc`") until OST write is large enough, default 64KiB
  - Dynamic cache selection, complementary with client Hybrid Buffered/Direct

Server-side Usability Improvements

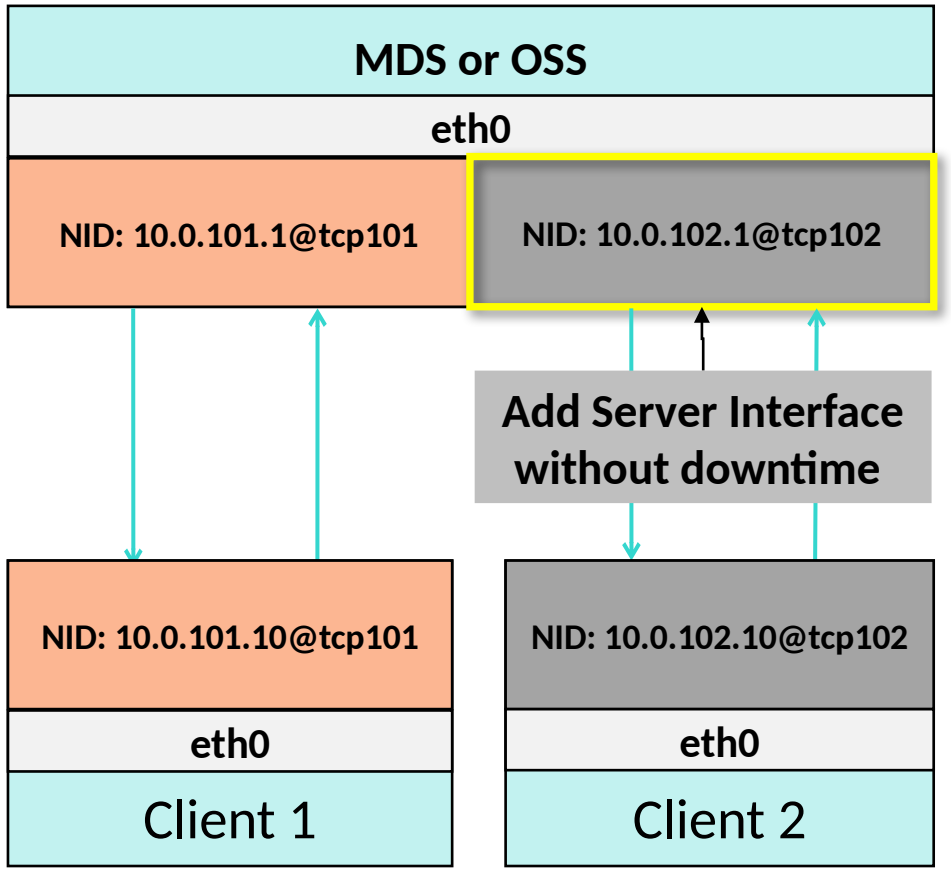
Ongoing improvements to usability and robustness for ease of management

- OST emergency read-only shutoff parameter (LU-12515 WC)
- Dynamic server NID configuration (LU-10360 WC)
- Network Request Scheduler TBF (QOS) improvements
  - 2.17 Store NRS TBF rules persistently via “lctl set_param –P” (LU-17920
  - 2.18 Compound rules for different TBF types (LU-18192 WC)
  - Rules by nodemap name, projid, ranges (LU-17902 LU-17166 LU-18
  - Default NRS TBF rule(s) to keep “noisy neighbor” in check (LU-1

- Lustre Quota Aggregation (LQA) to limit ranges of UID/GID/PROJID (LU-18222 WC)
  - Works with nodemaps to allow tenants to manage quota IDs while having per-tenant limits
- Enable default PFL layout on newly-formatted filesystems (LU-11918)
  - Simplify usage and improve performance for most configurations

Dynamic Server NIDs (LU-10360 2.17 WC)

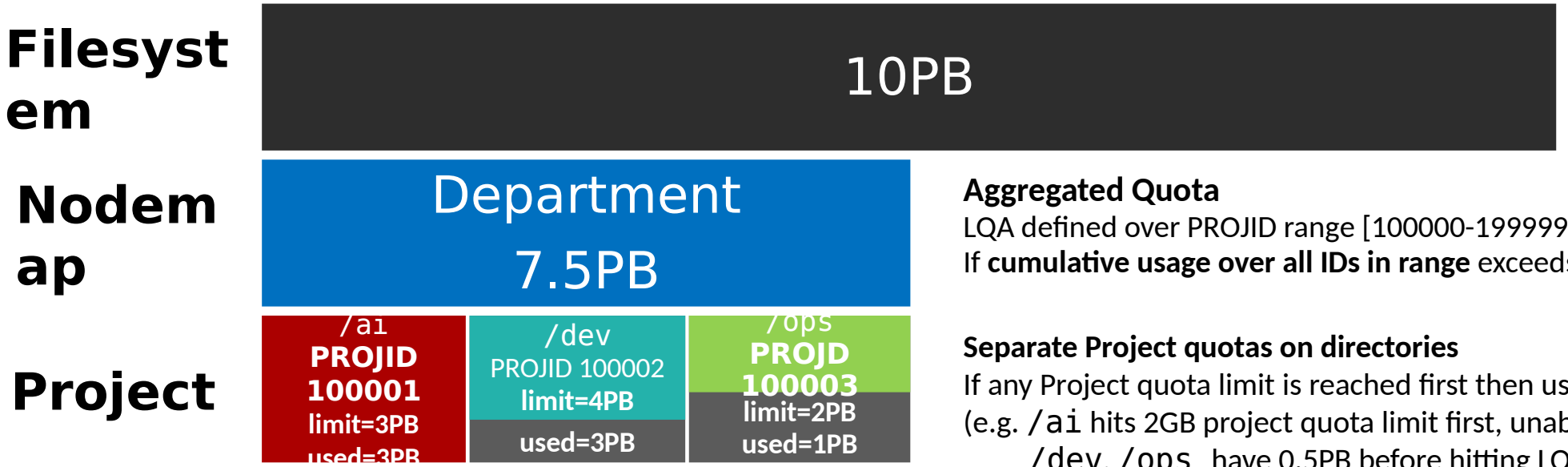
- New NID(s) can be dynamically added to servers online
  - Allow thousands of dynamic server NIDs to be configured
- No need to store dynamic NIDs in config log
  - Avoid writeconf outage during re-addressing/migration
  - More flexible for dynamic network environments
- Potentially useful for Cloud configurations
  - Move OSTs to new VM without target reconfiguration
- Dynamic NID configuration process
  - OSTs/MDTs advertise newly configured NIDs to MGS
  - MGS stores NIDs for each target in Imperative Recovery Log
  - Client uses MGS IR log to learn target NIDs if not in config log
    - Tunable to enable: `mdc.*.enable_dynamic_nids=1`



Lustre Quota Aggregation (LQA) (2.18 LU-18222 WC)

Whamcloud

- Aggregates quotas across a ranges of ProjIDs, UIDs and GIDs
  - Range quota is independent from quota on individual IDs
  - Lowest limit reached first is enforcing
- Facilitates multi-level quota for nodemaps
  - Delegate usage of Project, User, Group quota to Tenants



Aggregated Quota
LQA defined over PROJID range [100000-199999]
If **cumulative usage over all IDs in range** exceeds LQA limit then EDQUOT

Separate Project quotas on directories
If any Project quota limit is reached first then user receives EDQUOT
(e.g. `/ai` hits 2GB project quota limit first, unable to write more
`/dev`, `/ops` have 0.5PB before hitting LQA limit)

Client FLR Erasure Coded Files (LU-10911 2.18+ WC, ORNL)

- 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 Nd+Mp RAID Sets (e.g. 8d+2p or 16d+3p)
  - Fixed **RAID-4** RAID Set per file, declustered by file, CPU-optimized EC code (Intel ISA-L)

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	…

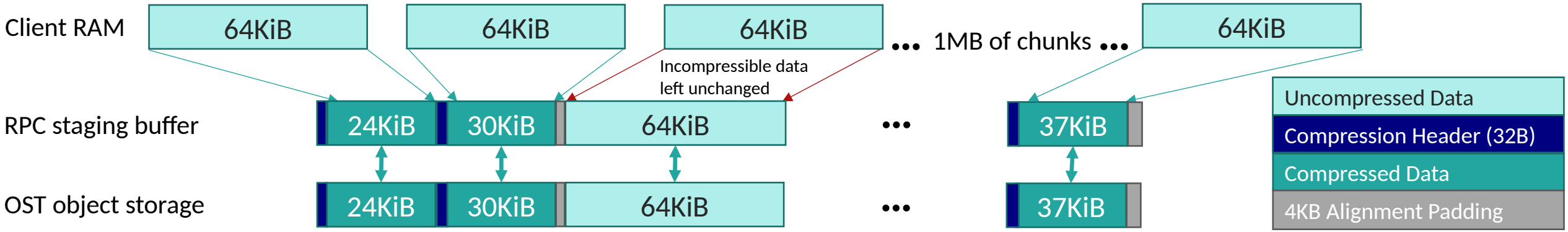
Existing stripes… Parity stripes added… Existing stripes… Parity stripes added…

Client-Side Data Compression (LU-10026 2.18 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)

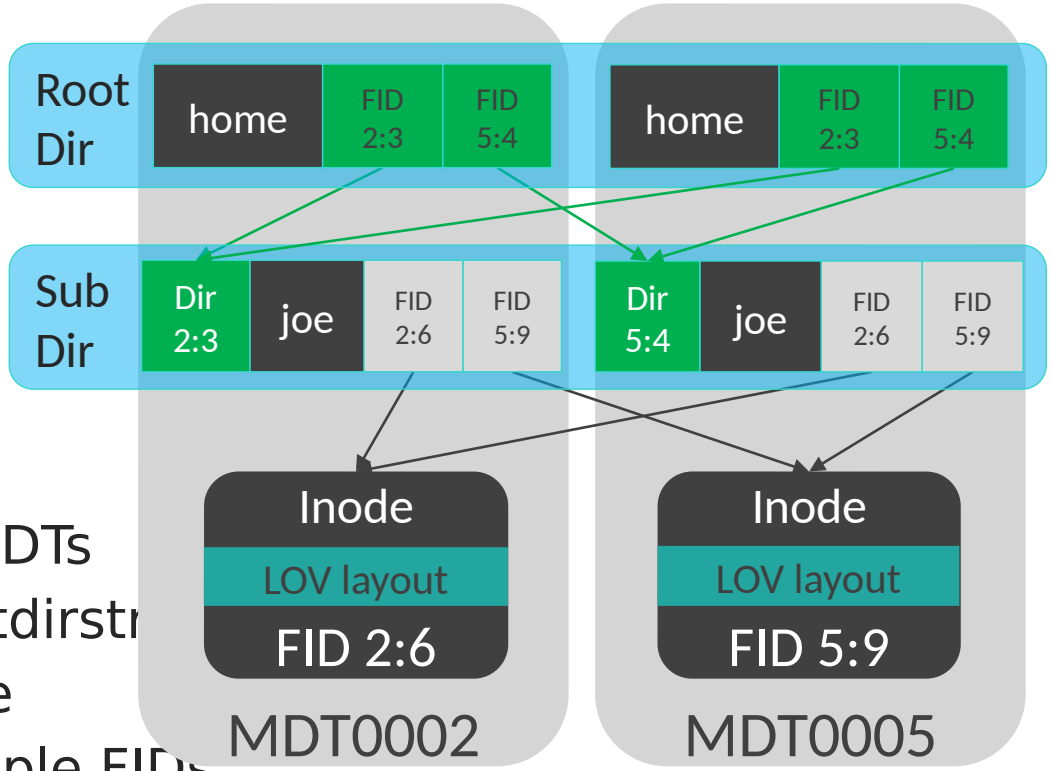
Trash Can/Undelete (LU-18465 2.18 WC)

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 delet
- 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: R00T/ (rarely changed) mirrored to other MDTs
  - 2b: Subdirectories mirrored to 2+ MDTs (via setdirstr
  - 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

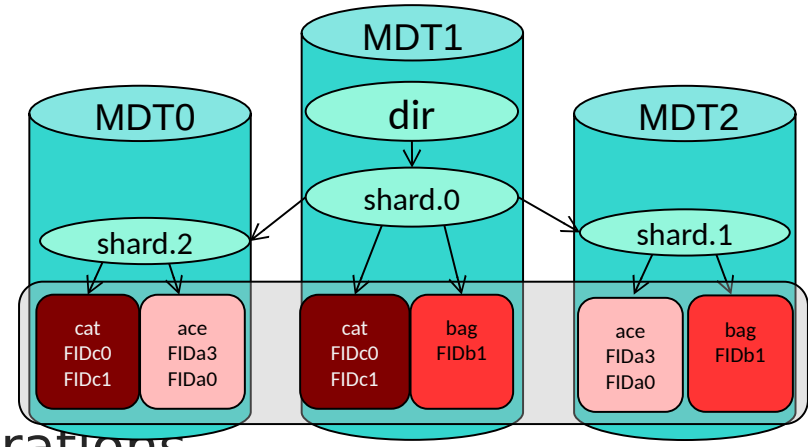


Fault Tolerant MGS (LMR-FTM) (2.18 WC)

Whamcloud

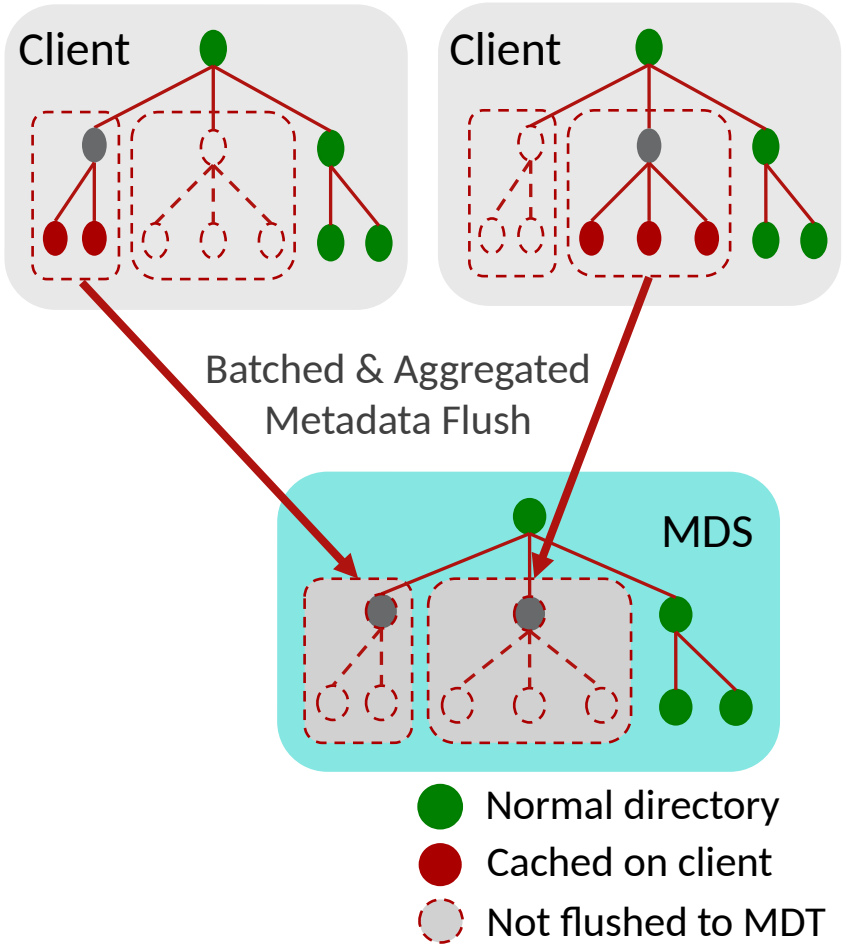
- Run MGS service on multiple MDS nodes for availability (LU-17819)
  - Allow clients to get config llogs from **any MDS node**
  - Reduces mount time/timeouts, distributes load in large clusters
  - MGS Imperative Recovery even if “primary” MGS node restarts
- Mirror MGS config logs to remote MDTs for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

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



- Remove critical service dependency on MDT0000
  - FLDB (maps FID->MDT/OST) redundancy via RAFT
  - Nodemap config redundancy via RAFT
  - Quota limit config redundancy via RAFT
- Quota Manager distributed to multiple servers
  - Hash distribution by UID/GID/PROJID, easily parallel operations
- LDLM `flock()` distribution and scaling over multiple MDS nodes
  - `flock()` locking is not really related to MDT inodes, so can run anywhere
  - Need to handle deadlock detection, but is disjoint for process groups, nodemaps
- Mirror `R00T/` 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+)

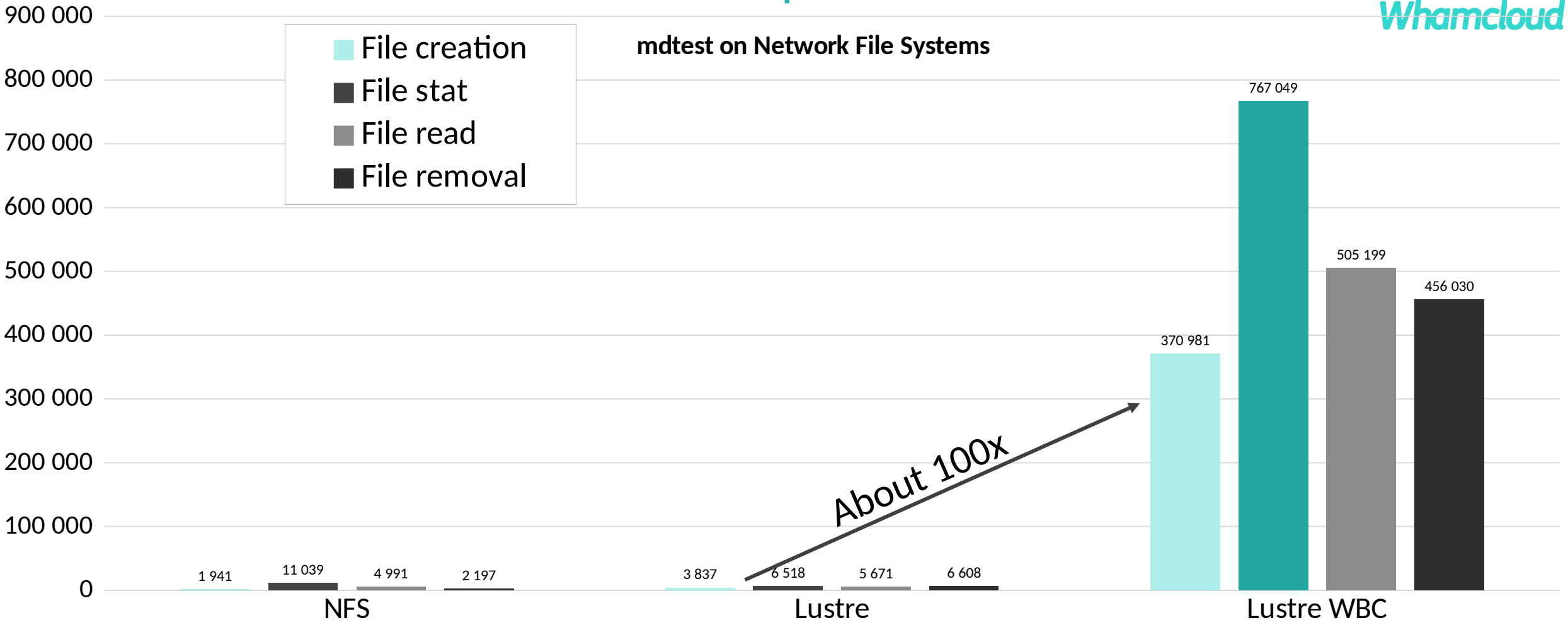


10-100x speedup for single-client create-intensive workloads

- 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 Filesystem on SSD: Intel SSDSC2KB240G8

Whamcloud

Thank You!

ddn