Whamcloud

Lustre 2.12 and Beyond

Andreas Dilger, Whamcloud

Upcoming Release Feature Highlights

2.12 is feature complete

- LNet Multi-Rail Network Health – improved fault tolerance
- Lazy Size on MDT (LSOM) – efficient MDT-only scanning
- File Level Redundancy (FLR) enhancements – usability and robustness
- T10 Data Integrity Field (DIF) – improved data integrity
- DNE directory restriping – better space balancing and DNE2 adoption

2.13 development and design underway

- Persistent Client Cache (PCC) – store data in client-local NVMe
- File Level Redundancy – Phase 2 Erasure Coding
- DNE auto remote directory/striping – improve load/space balance across MDTs

2.14 has continued functional/performance improvements

- DNE directory auto-split – improve usability and performance of DNE2
- Client metadata Write Back Cache (WBC)
  - Improve interactive performance, reduce latency

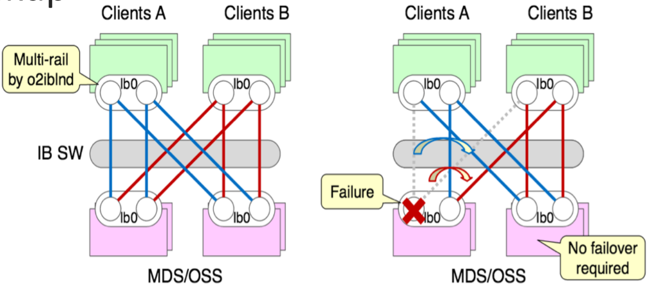
LNet Network Health, UDSP (2.12/2.14)

- Detect network interface and router failures automatically
  - Builds on LNet Multi-Rail in 2.10/2.11 (LU-9120 Intel/WC, HPE/SGI)
  - **LNet fault tolerance** without lengthy Lustre recovery
    - Optimize resend path to avoid faulty connections
  - Handle multi-interface router failures

DONE

TODO

- User Defined Selection Policy (LU-9121 WC, HPE)
  - Fine grained control of interface selection
    - TCP vs. IB networks, primary vs. backup
  - Optimize RAM/CPU/PCI transfers
  - Useful for large NUMA machines



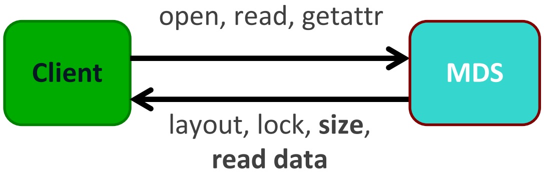
Data-on-MDT Improvement (WC 2.12+)

- ▶ Complementary with DNE 2 striped directories (LU-10176)
  - Scale small file IOPS with multiple MDTs
- ▶ **Read-on-Open** fetches data (LU-10181)
  - Reduced RPCs for common workloads
- ▶ Improved locking for DoM files (LU-10175)
  - Drop layout lock bit without full IBITS lock cancellation
  - Avoid cache flush and extra RPCs
  - Convert write locks to read locks
- ▶ Migrate file/component from MDT to OST (LU-10177)

DONE

TODO

- ▶ Migrate file/component from OST to MDT via FLR (LU-11421)
- ▶ Cross-file data prefetch via statahead (LU-10280)



Small file read directly from MDS

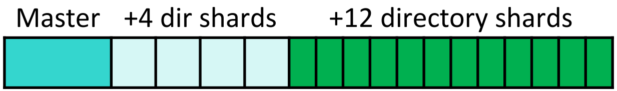
DNE Improvements (WC 2.12/2.13)

- **Directory restriping** from single-MDT to striped/sharded directories (LU-4684)
  - Rebalance MDT space usage, improve large directory performance
- Automatically create new remote directory on "best" MDT with `mkdir()`
  - Simplifies multiple MDTs without striping all directories, similar to OST usage
  - In 2.11 in userspace for `lfs mkdir -i -1` (LU-10277)

DONE

TODO

- **Automatic MDT load balancing** at directory creation (LU-10784, LU-11213)
  - Using standard `mkdir()` calls with parent `stripe_idx = -1`
- **Automatic directory restriping** as directory size grows (LU-11025)
  - Create one-stripe directory for low overhead, scale shards/capacity/performance with size
  - Add extra directory shards when master directory grows large (e.g. 10k files)
  - Move existing direntries to new directory shards
  - New direntries and inodes created on new MDTs



Master +4 dir shards +12 directory shards

ZFS Enhancements Related to Lustre (2.12+)

- Lustre 2.12 `osd-zfs` updated to use ZFS 0.7.9
  - Bugs in ZFS 0.7.7/0.7.10/0.7.11, not used by Lustre
  - Builds with upstream ZFS pre-0.8.0 branch
- **Features in ZFS 0.8.x** release (target 2019Q1)
  - Will move to 0.8 when officially released
  - Sequential scrub/resilver (Nexenta)
  - On-disk encryption + QAT acceleration (Datto)
  - **Project quota** accounting (Intel)
  - **Device removal** via VDEV remapping (Delphix)
  - **Metadata Allocation Class** (Intel, Delphix)

LANDED

IN PROGRESS

  - Declustered Parity RAID (dRAID) (Intel)

Miscellaneous Improvements (2.12)

- **UID/GID policy** for Token Bucket Filter (NRS-TBF) (LU-9658 DDN)
- Improved JobStats allows admin-formatted JobID (LU-10698 Intel)

```
lctl set_param jobid_env=SLURM_JOB_ID jobid_name=cluster2.%j.%e.%p
```

- HSM infrastructure improvement & optimizations (Intel/WC, Cray)
  - Coordinator (LU-10699), POSIX Copytool (LU-11379), > 32 archives (LU-10114), …
- **Lazy Size-on-MDT** for disk scan (purge, HSM, policy engine) (LU-9358DDN)
  - LSOM is not guaranteed to be accurate, but good enough for many tools
  - LSOM available on client to apps *aware of limitations* (`lfs find`, `statx()`, …)
- **Lustre-integrated T10-PI** end-to-end data checksums (LU-10472 DDN)
  - Checksums between client and OSS, low overhead, integrate with hardware

DONE

TODO

- Dump/restore of conf_params/set_param -P parameters (LU-4939 Cray)

```
mgs# lctl --device MGS llog_print testfs-client > testfs.cfg
mgs# lctl set_param -F testfs.cfg
```

Improved Client Efficiency (2.12+)

- **Disconnect idle clients** from OSS (LU-7236 Intel)
  - Reduce memory usage on client and server for large systems
  - Reduce network pings between clients and OSTs
  - Reduce OST recovery times due to fewer client connections
- **Aggregate `statfs()`** RPCs on the MDS (LU-10018 WC)
  - Reduce OSS overhead, avoid idle OST reconnection on client `df`

DONE

TODO

- Improved client read performance (LU-8709 DDN)
  - Improved readahead code (backward strided reads)
  - Asynchronous IO submission to avoid blocking user threads
- Reduce wakeups and background tasks on idle clients (LU-9660 Intel)
  - Synchronize wakeups between threads/clients (per jobid?) to minimize jitter
  - Still need to avoid DOS of server if all clients ping/reconnect at same time

Flash Performance Improvements (WC 2.12+)

- **Reduce server CPU** overhead to improve small flash IOPS
  - Performance is primarily CPU-limited for small read/write
  - Any reduction in CPU usage directly translates to improved IOPS
- **Avoid page cache** on flash OSS (LU-11347)
  - Avoids CPU overhead/lock contention for page eviction
  - Streaming flash performance is often network limited
  - ZFS drops pages from cache after use (LU-11282)

DONE

TODO

- Improved efficiency of ZFS IO pipeline
  - Integrate with ABD in ZFS 0.8 to avoid `memcpy()` of data
  - Further improvements with continued investigation/development

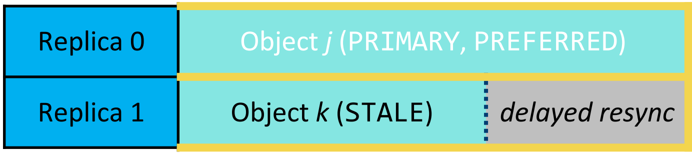
FLR Enhancements (WC 2.12+)

- Continuation of FLR feature landed in Lustre 2.11 (LU-9771)
- FLR-aware OST object allocator to avoid replicas on same OST/OSS (LU-9007)
- Improve "`lfs mirror resync`" performance (LU-10916)
  - Optimize multi-mirror resync (read data once, write multiple mirrors)
- **"`lfs mirror read`"** to dump specific mirror/version of file (LU-11245)
- **"`lfs mirror write`"** for script-based resync (LU-10258)
- **Mirror NOSYNC flag** + timestamp to allow file version/snapshot (LU-11400)

DONE

TODO

- Improved replica selection at runtime (LU-10158)
  - Select best write replica (PREFER, SSD vs. HDD , near to client), read (many mirror vs. few)
  - Allow specifying fault domains for OSTs (e.g. rack, PSU, network switch, etc.)
- Mount client directly on OSS for improved resync performance (LU-10191)
- Support DoM components (LU-10112)
- Pool or OST/MDT quotas (LU-11023)
  - Track/restrict flash OST/MDT space usage
- **Erasure Coded** striped files (LU-10911)

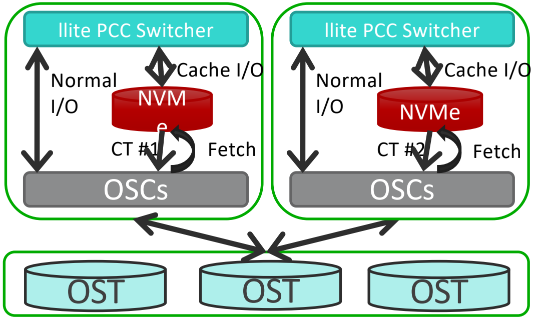


Persistent Client Cache (PCC) (DDN/WC 2.13+)

LU-10092

Whamcloud

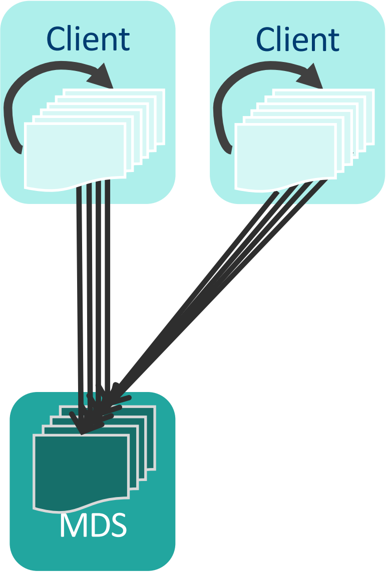
- Reduce latency, improve small/unaligned IOPS, reduce network traffic
- PCC integrates Lustre with persistent per-client local cache devices
  - Each client has own cache (SSD/NVMe/NVRAM) as a local filesystem (e.g. ext4/ldiskfs)
  - No global/visible namespace is provided by PCC, data is local to client only
  - Files pulled into PCC by HSM copytool by user request, job script, or policy
  - Each new file created in PCC is *also* created on Lustre MDS
- Kernel uses local file if in cache, or normal Lustre IO to OSTs
  - File read/write access “directly” to local data
  - No data/IOPS/attributes from client while in PCC
  - File moved out of PCC via HSM on remote access
- Separate read vs. write file cache features
- Later integrate with DAX for NVRAM PCC



Metadata Writeback Cache (WBC) (2.14+)

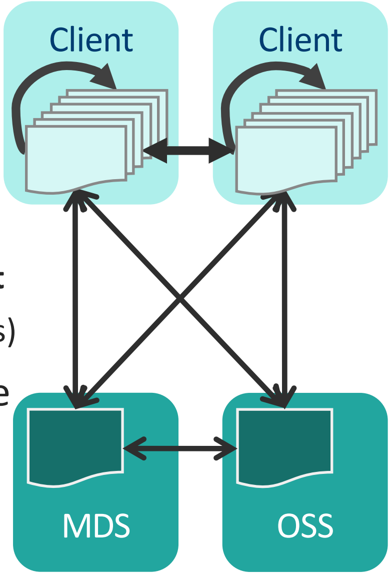
LU-10983

- Metadata WBC creates new files in RAM in *new directory*
  - Avoid RPC round-trips for each open/create/close
  - Lock directory exclusively, avoid other DLM locking
  - Cache file and directory only in pagecache
  - Flush tree in background to MDT/OST
- Could prefetch existing directory via readdir
- Can integrate with PCC to avoid MDS create/write
- Early WBC prototype in progress
  - Discussions underway for how to productize it
  - Early results show 10-20x speedup for some tests
    - Kernel untar, make, dbench, mdtest



Client Container Image (CCI) (2.14+)

- Filesystem images used *ad hoc* with Lustre in the past
  - Read-only cache of many small files manually mounted on clients
  - Root filesystem images for diskless clients
- ***Container Image* is local ldiskfs image** mount on client
  - Holds a whole directory tree stored as a single Lustre file
  - **Useful for workloads that are mostly self-contained**
- CCI integrates container handling with Lustre
  - Mountpoint is registered with Lustre for automatic mount
  - Image file read on demand from OST(s) and/or cached in PCC
  - **Low I/O overhead, few file extent lock(s), high IOPS per client**
  - Access, migrate, replicate image with large read/write to OST(s)
- MDS can mount/re-export image files for shared use
- CCI can hold whole directory tree for HSM archive
- Unregister/delete whole image with a few RPCs



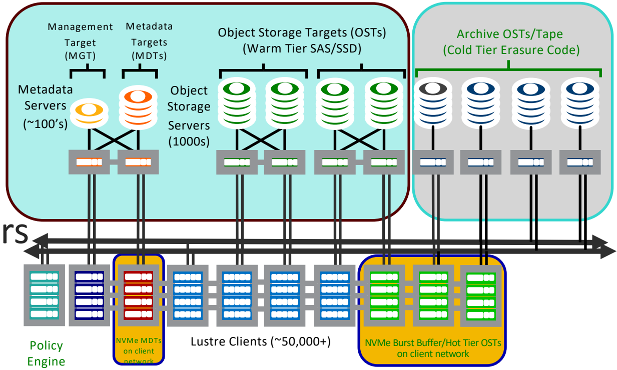
Whamcloud

Kernel Code Cleanup (ORNL, SuSE, WC, Cray)

- Lustre client removed from kernel 4.17 ☹
  - Work continuing on client cleanups *for* upstream at https://github.com/neilbrown/linux
- Lustre 2.12 updates for kernel 4.14/4.15 (LU-10560/ LU-10805)
- Improve kernel time handling (Y2038, jiffies) (LU-9019)
- Ongoing `/proc` -> `/sys` migration and cleanup (LU-8066)
  - Handled internally by `lctl` / `llapi_*` - please use them
- Cleanup of `wait_event`, `cfs_hash_*`, and many more internals
- Some build/test with ARM64/Power8 clients (LU-10157 little-endian!)
- Major `ldiskfs` features merged into upstream `ext4/e2fsprogs`
  - Large xattr (`ea_inode`), directories > 10M entries (`large_dir`)
  - `dirdata` feature not yet merged (needs test interface)

Tiered Storage with FLR Layouts (2.12/2.13)

- Integration with job scheduler and workflow for file prestage/drain/archive
- Policy engine manage migration over tiers, rebuild replicas, ChangeLogs
  - Policies for pathname, user, extension, age, OST pool, mirror copies, ...
  - FLR provides mechanism for safe migration of (potentially in-use) data
  - RBH or LiPE are good starting points for this
- Needs userspace integration and Lustre hooks
  - Integrated burst buffers a natural starting point
  - Mirror to flash for input/output files
    - Mark mirror PREFERRED for read/write
  - Resync modified files off flash
    - Release space for other users
- Need OST/MDT/pool quotas to manage tiers


Management Target (MGT)
Metadata Targets (MDTs)
Object Storage Targets (OSTs) (Warm Tier SAS/SSD)
Archive OSTs/Tape (Cold Tier Erasure Code)
Metadata Servers (~100's)
Object Storage Servers (1000s)
Policy Engine
NVMe MDTs on client network
Lustre Clients (~50,000+)
NVMe Burst Buffer/Hot Tier OSTs on client network

FLR Erasure Coded Files (LU-10911 WC 2.13+)

- Erasure coding adds redundancy without 2x/3x overhead
- Add erasure coding to new/old striped files *after* write done
  - Use delayed/immediate mirroring for files being actively modified
  - Can be added to existing striped files
- For striped files - add N parity per M data stripes (e.g. 16d+3p)
  - Parity declustering avoids IO bottlenecks, CPU high if many parities
    - e.g. split 128-stripe file into 8x (16 data + 3 parity) with 24 parity

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	...