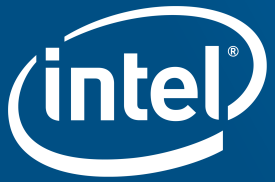


Lustre* 2.12 and Beyond

Andreas Dilger, Intel High Performance Data Division

LUG 2018

Upcoming Feature Highlights

2.12 landings ongoing with several features lined up

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

2.13/2.14 plans continued functional and performance improvements

- Persistent Client Cache (PCC) - store data in client-local NVMe
- DNE directory auto-split - improve usability and performance of DNE2
- File Level Redundancy - Phase 2 erasure coding

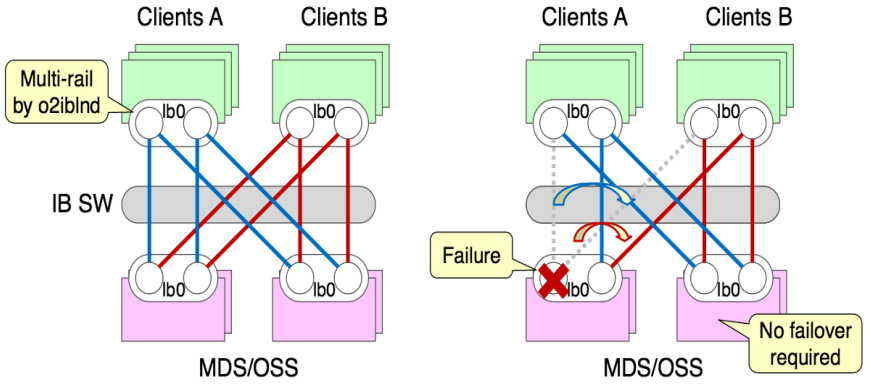
LNet Network Health, UDSP (2.12/2.13)

Builds on LNet Multi-Rail in 2.10/2.11 (LU-9120 Intel, HPE/SGI*)

- Detect network interface and router failures automatically
- Handle LNet fault w/o lengthy Lustre recovery, optimize resend path

User Defined Selection Policy (LU-9121 Intel, HPE*)

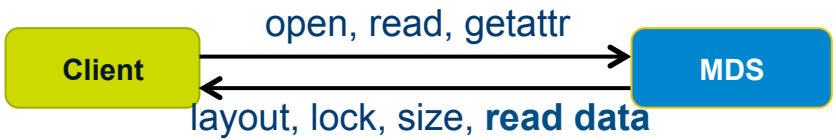
- Fine grained control of interface selection
- Optimize RAM/CPU/PCI data transfers
- Useful for large NUMA machines



Data-on-MDT Improvements (LU-10716 Intel 2.12)

Read-on-Open fetches data (LU-10181)

- Reduced RPCs for common workloads
- Allow cross-file readahead for small files



Small file read directly from MDS

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

Complementary with DNE 2 striped directories

- Scale small file IOPS with multiple MDTs

DNE Improvements (LU-4684 Intel 2.12/2.13)

Directory restriping from single-MDT to striped/sharded directories

- Rebalance MDT space usage, improve large directory performance

Automatically create new remote directory on "best" MDT with `mkdir()`

Automatic directory restriping to reduce/avoid need for explicit striping at create

- Start with single-stripe directory for low overhead in common use cases
- Add extra shards when master directory grows large enough (e.g. 10k entries)
- Move existing direntries to new directory shards, keep existing inodes in place
- New entries+inodes created in new shards to distribute load across MDTs
- Performance scales as directory grows

ZFS Enhancements Related to Lustre (2.12+)

Lustre 2.12 `osd-zfs` being updated to use ZFS 0.7.8

- Serious bug in ZFS 0.7.7, was never landed to Lustre branch

Features in ZFS 0.8.x release (target 2018Q4)

- Depends on final ZFS release date, to avoid disk format changes
- Sequential scrub/resilver (Nexenta)
- On-disk data encryption + QAT hardware acceleration (Datto)
- Project quota accounting (Intel)
- Device removal via VDEV remapping (Delphix) (not yet landed)
- Metadata Allocation Class (Intel, Delphix) (not yet landed)
- Declustered Parity RAID (dRAID) (Intel) (not yet landed)

Miscellaneous Improvements (2.12/2.13)

Token Bucket Filter (NRS-TBF) UID/GID policy (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
```

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

```
lctl --device MGS llog_print testfs-client > log; lctl set_param -F log
```

HSM infrastructure improvement & optimizations (LU-10383 Intel, Cray)

Lazy Size-on-MDT for local scan (purge, HSM policy engine) (LU-9358 DDN)

- LSOM is not guaranteed to be accurate, but good enough for many tools

Lustre-integrated T10-PI end-to-end data checksums (LU-10472 DDN)

- Pass data checksums between client and OSS, avoid overhead, integrate with hardware

Persistent Client Cache (PCC) for client-side NVMe/NVRAM data cache (LU-10092 DDN)

Upstream Kernel (ORNL, Intel, Cray, SuSE)

Kernel 4.14 updated to approximately Lustre 2.8, plus many fixes cross-ported

Lustre 2.12 updates for kernel 4.14/4.15 (LU-10560/ LU-10805)

Improve kernel time handling (Y2038, jiffies) (LU-9019)

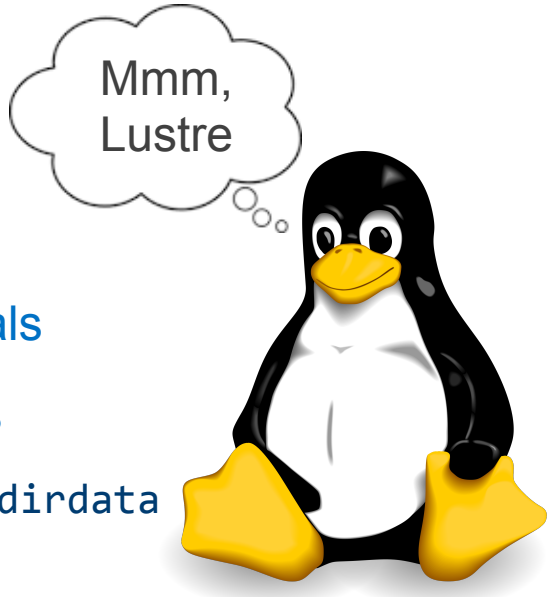
Ongoing `/proc` -> `/sys` migration and cleanup (LU-8066)

- Handled transparently by `lctl` and `llapi_*` - please use them

Cleanup of `wait_event`, `cfs_hash`, and many more internals

Major ldiskfs features merged into upstream ext4/e2fsprogs

- Large xattr (`ea_inode`), directories > 10M entries (`large_dir`), `dirdata`



FLR Enhancements (Intel 2.12)

Continuation of FLR feature landed in Lustre 2.11 (LU-9771)

Improved FLR-aware OST object allocator (LU-9007)

- Avoid replicas on same OST/OSS (easy) and same enclosure/rack/PSU (needs external input)

Improved replica selection at runtime (LU-10158)

- Decide which replica to modify at write time (`PREFERRED`, near to client, SSD vs. HDD)

Server local client for improved resync performance (LU-10191)

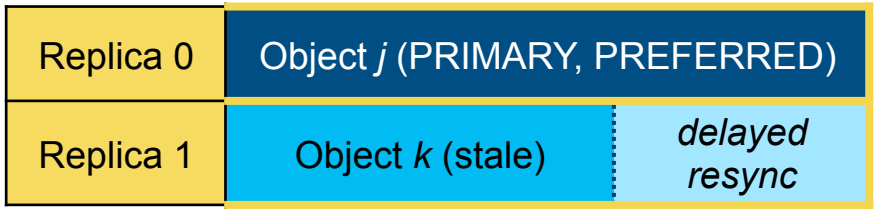
- Mount directly on OSS with disabled recovery and cache

Improve `lfs mirror resync` performance (LU-10916)

- Optimize multi-mirror resync (read once)

`lfs mirror open` command (LU-10258)

- External resync/migration tools can open mirror



FLR Erasure Coded Files (LU-10911 Intel 2.13)

Erasure coding gives redundancy without 100% or 200% mirror overhead

Add erasure coding to new or existing striped files *after* write is finished

- Use delayed/immediate mirroring for files being actively modified

Suitable for striped files - add N parity per M data stripes (e.g. 16d+3p)

- 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 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	...

Tiered Storage with FLR Layouts (2.12/2.13)

Integration with job scheduler and workflow for prestage/drain/archive

Policy engine to manage migration between tiers, rebuild replicas, ChangeLogs

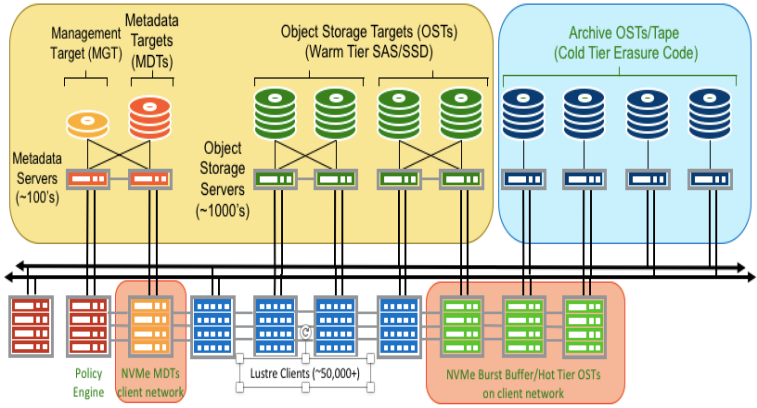
- Policies for pathname, user, extension, age, OST pool, mirror copies, ...
- FLR provides mechanisms for safe migration of (potentially in-use) data

Multiple policy/scanning engines shown at LUG'17

Multiple presentations on tiered storage at LAD'17

- Integrated burst buffers a natural starting point

Mostly userspace integration, with Lustre hooks



Improved Client Efficiency (2.12+)

Improved client read performance (LU-8964, Intel)

- Leverage kernel readahead code, run asynchronously

Disconnect idle clients from servers (LU-7236 Intel)

- Reduce memory usage on client and server for large systems
- Reduce network pings and recovery times
- Aggregate `statfs()` RPCs on the MDS (LU-10018)

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

Client-Side Data Compression (LU-10026 2.13)
University Hamburg

Piecewise compression

- Compressed in 32KB chunks
- Allows sub-block read/write

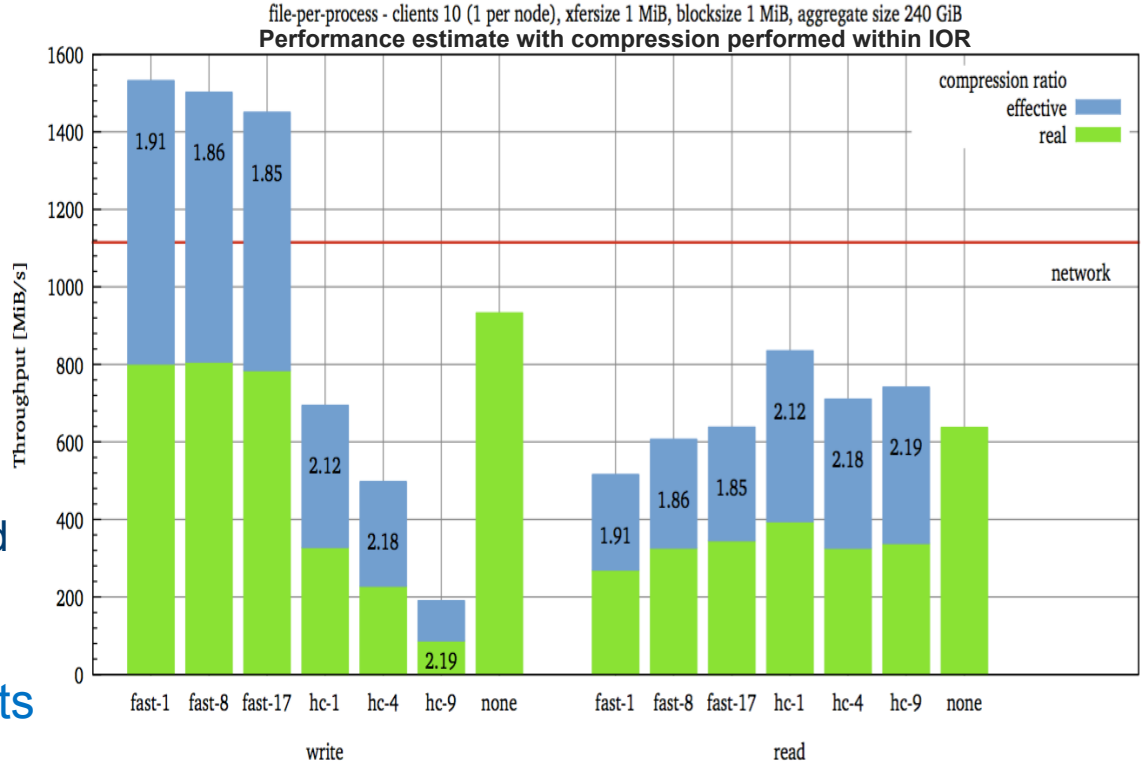
Integrated with ZFS data blocks

- Leverage per-block type/size
- Code/disk format changes needed

Avoid de-/re-compressing data

Good performance/space benefits

- Graph courtesy Uni Hamburg



Client-side Writeback Cache OR Fileset (2.14+)

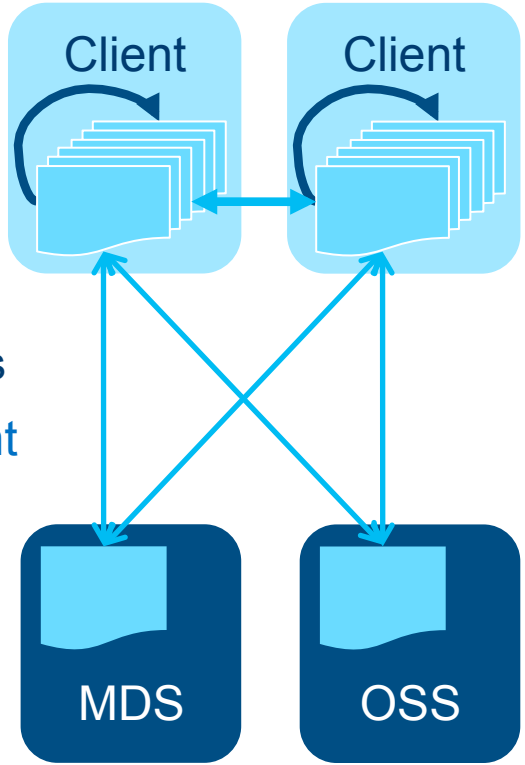
Client WBC creates files in RAM in new directory

- Could prefetch directory contents for existing directory
- Avoid RPC round-trips for each open/create/close
- Lock directory exclusively, avoid other DLM locking
- Cache file data only in pagecache until flush
- Flush tree incrementally to MDT/OST in background batches

Fileset is local ldiskfs image mounted transparently on client

- Image is file in local PCC or OST, holds whole directory tree
- Low overhead, only file extent lock(s), high IOPS/client
- Access, migrate, replicate with large reads/writes to OSTs
- MDS can mount and export image files for shared use

Early WBC prototype in progress, discussions underway



DNE Metadata Redundancy (2.14+)

New directory layout hash for mirrored directories, mirrored MDT inodes

- Each dirent copy holds multiple MDT FIDs for inodes
- Store dirent copy on each mirror directory shard
- Name lookup on any MDT can access via any FID
- Copies of mirrored inodes stored on different MDTs

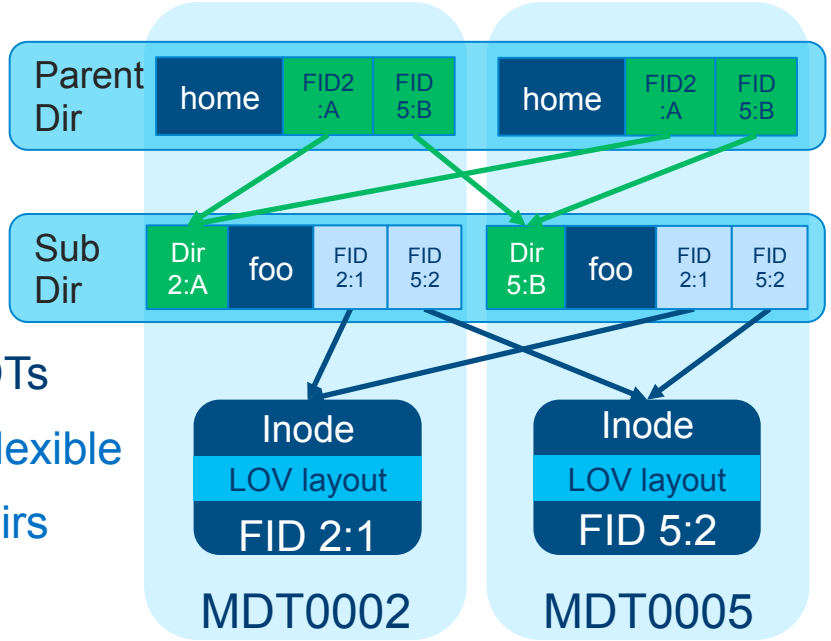
DNE2 distributed transaction for update recovery

- Ensure that copies stay in sync on multiple MDTs

Redundancy policy per-filesystem or subtree, is flexible

Flexible MDT space/load balancing with striped dirs

Early design work started, discussions ongoing



Legal Notices and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life-saving, life-sustaining, critical control or safety systems, or in nuclear facility applications.

Intel products may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document may contain information on products in the design phase of development. The information herein is subject to change without notice. Do not finalize a design with this information. Intel may make changes to dates, specifications, product descriptions, and plans referenced in this document at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Intel Corporation or its subsidiaries in the United States and other countries may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Performance estimates or simulated results based on internal Intel analysis or architecture simulation or modeling are provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

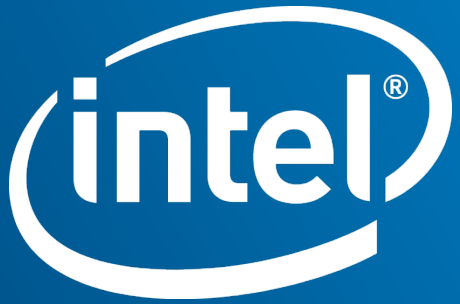
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel processors of the same SKU may vary in frequency or power as a result of natural variability in the production process.

Intel, the Intel logo, 3D-Xpoint, Optane, Xeon Phi, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.


intel®