Whamcloud

Lustre 2.13 and Beyond

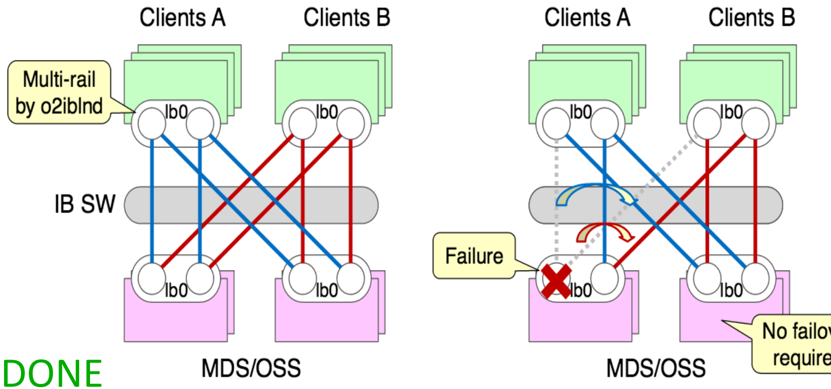
Andreas Dilger

Upcoming Release Feature Highlights

- **2.13** development and landing underway, ETA August, 2019
  - **Persistent Client Cache** (PCC) – store data in client-local NVMe
  - **DNE automatic remote directory** – improve load/space balance across MDTs
  - **LNet User Defined Selection Policy** – tune LNet Multi-Rail interface selection
- **2.14** already has a number of features underway
  - **File Level Redundancy** – Erasure Coding (EC) for striped files
  - **OST pool quotas** – manage space on heterogeneous storage targets
  - **DNE directory auto-split** – improve usability and performance of DNE2
- **2.15** plans continued functional and performance improvements
  - **Client-side data encryption** - end-to-end data privacy
  - **Client-side data compression** - data size reduction, improved network bandwidth
  - **LNet IPv6** - improved addressing, with simplified network configuration

LNet Network Selection Policy and More (WC 2.13+)

- User Defined Selection Policy (LU-9121)
  - Builds on LNet Multi-Rail in 2.10/2.11
  - Fine grained control of interface selection
    - TCP vs. IB networks, primary vs. backup
  - Optimize RAM/CPU/PCI data transfers
  - Useful for large NUMA machines



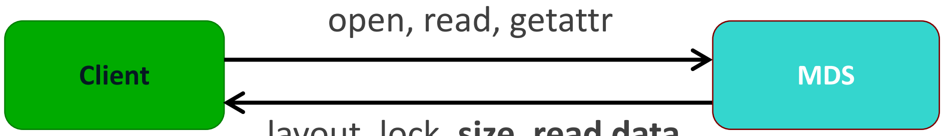
DONE

TODO

- LNet Unit Test Framework (LU-10973)
  - Improved test configuration/coverage for LNet
- IPv6 Investigation, Design, and Implementation (LU-10391)
  - Remove direct NID usage from Lustre config files
  - Simplify server address changes (DHCP even?)
  - Protocol change, with interoperability for existing releases

Data-on-MDT Improvements (LU-10176 WC 2.12+)

- ▶ Complementary with DNE 2 striped directories
  - 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)
  - Convert write locks to read locks



Small file read directly from MDS

- ▶ Migrate files with MDT component to OST-only layout (LU-10177)

DONE

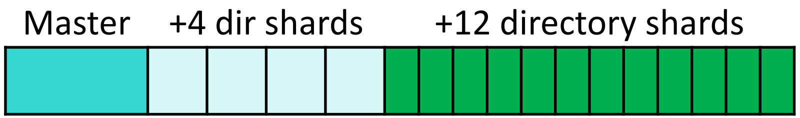
TODO

- ▶ Support DoM+FLR components (LU-10112)
- ▶ Migrate DoM component from OST to MDT via FLR (LU-11421)
- ▶ Cross-file data prefetch via statahead (LU-10280)
- ▶ Allow MDT-only filesystem (LU-10995)

DNE Improvement (WC 2.12+)

- **Directory restriping** from single-MDT to striped/sharded directories (LU-4684)
  - Rebalance MDT space usage, improve large directory performance
- Balance MDT usage by creating new directories on MDT with most free space
  - Simplifies use of multiple MDTs without striping all directories, similar to OST usage
  - In userspace with `lfs mkdir –i -1` (LU-10277)
  - At directory `mkdir()` time for "size hashed" directory layout (LU-10784, LU-11213)
- **Improved DNE file create performance** for clients (LU-11999 Uber) DONE

- **Automatic directory restriping** as directory size grows (LU-11025) TODO
  - Create one-stripe directory for low overhead, scale shards/capacity/performance with size
  - Add extra directory shards when master directory grows large enough (e.g. 10k entries)
  - Move existing dirents to new directory shards
  - New dirents and inodes created on new MDTs



ZFS Enhancements Related to Lustre (2.12+)

Whamcloud

- Lustre 2.12.1/2.13 `osd-zfs` updated to use ZFS 0.7.12
  - Bugs in ZFS 0.7.7/0.7.10/0.7.11, not used by Lustre branch
  - Builds with upstream ZFS 0.8.0-rc code for testing
- **Features in ZFS 0.8.x** release (target 2019Q2)
  - **Sequential scrub/resilver** to speed up disk repair/replace (Nexenta)
  - **On-disk data encryption** + QAT hardware acceleration (Datto)
  - **Project quota** accounting (Intel)
  - **Device removal** via VDEV remapping (Delphix)
  - **Metadata Allocation Class** (Intel, Delphix)
  - **TRIM/UNMAP** support for flash devices (Nexenta, LLNL) DONE
- **Features underway for ZFS 0.9** release IN PROGRESS
  - Declustered Parity RAID (dRAID) (Intel)

OpenZFS

Miscellaneous Improvements (2.12/2.13)

- HSM infrastructure improvement & optimizations (Intel/WC, Cray)
  - Improve Coordinator (LU-10699), POSIX Copytool (LU-11379), > 32 archives (LU-10114), …
- **Lazy Size-on-MDT** (LSOM) for local scan (purge, HSM, policy engine) (LU-9358 DDN)
  - LSOM is not 100% accurate, but good for apps *aware of limits* (e.g. `lfs find`, `statx()`, …)
- **Lustre-integrated T10-PI** end-to-end data checksums (LU-10472 DDN)
  - Pass data checksums between client and OSS, avoid overhead, integrate with hardware

2.12

2.13

- **Foreign Layout** file/directory in namespace (DAOS, CCI) (LU-11376 Intel)
- **LSOM** support for `lfs find` and other utilities (LU-11367 WC)
- **SELinux policy verification** to verify clients are using correct policy (LU-8955 WC)
- "**`lfs migrate -A`**" auto-stripe count when migrating files (LU-8207 NASA)

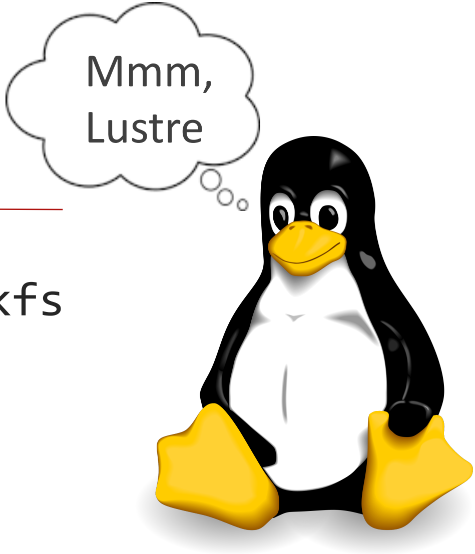
Kernel Code Improvements (ORNL, SUSE, WC, Cray)

- ▶ Lustre client removed from kernel 4.17 staging area ☹
  - Work ongoing on client for upstream resubmission at https://github.com/neilbrown/linux
- ▶ Build/test with **ARM64/Power8** clients (LU-10157)
- ▶ Major `ldiskfs` features merged into upstream `ext4/e2fsprogs`

DONE
  - Large xattrs (`ea_inode`), directories over 10M entries/2GB (`large_dir`)

TODO
  - Extended data in directory (`dirdata`)

- ▶ Existing `ext4` features available that could be leveraged by `ldiskfs`
  - Tiny files (1-800 bytes) stored directly in inode (`inline_data`)
  - Metadata integrity checksums (`metadata_csum`)
  - Efficient allocation for large OSTs (`bigalloc`)
- ▶ New `ext4` features currently under development
  - Verity – data checksums, directory shrink - reduce space as files deleted



File Layout Enhancements (WC 2.13+)

Whamcloud

- **Overstriping** allows multiple file stripe objects per OST (LU-9846 Cray, WC)
  - Extension of existing RAID-0 layout type, with fixes for large stripe counts
  - Useful for shared-file workloads or very large OSTs (e.g. declustered-parity RAID storage)
  - Improves storage utilization through higher concurrency (locking and threads)
- **PFL Self-Extending Layouts** (SEL) handles full OSTs during file write (LU-10070 Cray)
  - Last component is a template to instantiate EOF component if needed
- **Data Placement Policy** (DPP) improves workload-to-layout mapping (LU-11234 WC)
  - Allow selecting layout based on file extension, NID, UID, GID, JobID, ProjID, etc.
- Improved FLR replica selection at runtime (LU-10158)
  - PREFERRED, SSD vs. HDD , near to client, read (many mirror vs. few)
  - Allow specifying fault domains for OSTs (e.g. rack, PSU, network switch, etc.)
- **OST/MDT quotas** (LU-11023, Cray)
  - Track/restrict space usage on flash OSTs/MDTs

FLR Erasure Coded Files (LU-10911 WC 2.14)

Whamcloud

- Erasure coding adds redundancy without 2x/3x mirror overhead
- Add erasure coding to new/old striped files *after* write done
  - Use delayed/immediate mirroring for files being actively modified
  - Leverage CPU-optimized EC code (Intel ISA-L) for best performance
- For striped files - add N parity per M data *stripes* (e.g. 16d+3p)
  - Fixed RAID-4 parity layout *per file*, but declustered *across files*
  - 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	...

Improved Client Performance (2.13+)

- **Single thread create** performance on **DNE2** striped directories (LU-11999 Uber)
  - Reduce locking overhead/latency *for single-threaded* workloads (**780/sec -> 2044/sec**)
- **Improved client read** performance (LU-8709, LU-12043 WC)
  - Optimize *single-threaded readahead* using async prefetch threads with 128MB window
  - Improved single-threaded streaming read performance (**1.9GB/s -> 4.0GB/s**)
  - Multi-threaded IOR maintains performance (10817 MB/sec -> 11196 MB/s)
- **Async Direct IO** (AIO-DIO) improvement (LU-4198 Uber, WC)
  - Reduce overhead of submitting small DIO data writes
  - Enable async IO interface to avoid waiting on requests
  - Other related changes to improve efficiency (single client 4KB read **80k IOPS -> 600k IOPS**)

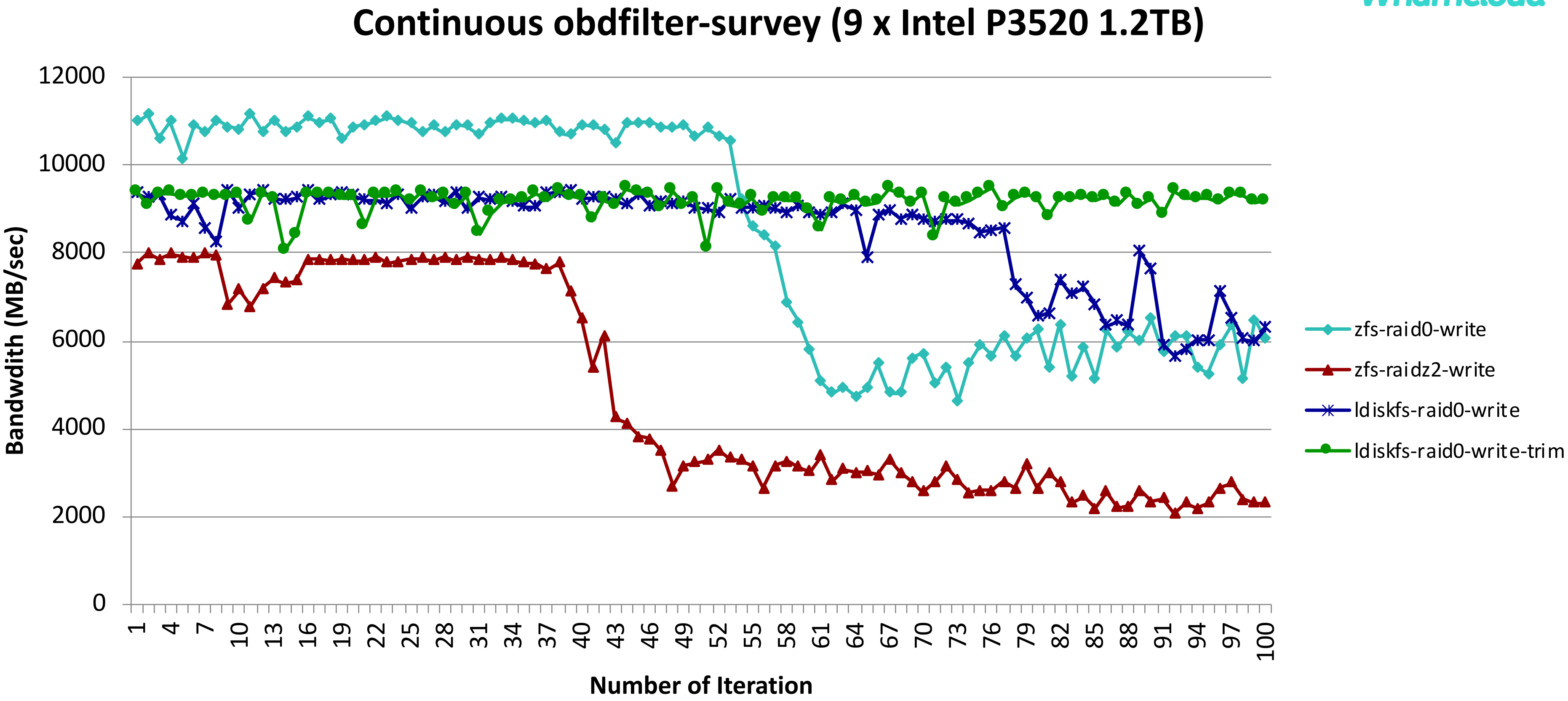
Improved Server Performance for Flash (WC 2.12+)

Whamcloud

- **Reduce server CPU** overhead to improve small flash IOPS (LU-11164)
  - Performance is primarily CPU-limited for small read/write
  - Any reduction in CPU usage directly translates to improved IOPS
- **Avoid page cache** on `ldiskfs` flash OSS (LU-11347)
  - Avoids CPU overhead/lock contention for page eviction
  - Streaming flash performance is often network limited
- **TRIM** of backend flash storage (LU-11355 WC)
  - Pass-through to backend ldiskfs filesystem `fstrim` functionality DONE
  - Pass-through to backend ZFS filesystem TRIM functionality TODO
- Improve performance of small, unaligned writes (journal?)
- Improve efficiency of ZFS IO pipeline
  - Integrate with ABD in ZFS 0.8 to avoid `memcpy()` of data

Flash FTL Garbage Collection Impact vs. TRIM

Whamcloud



Tiered Storage with FLR Layouts (2.12+)

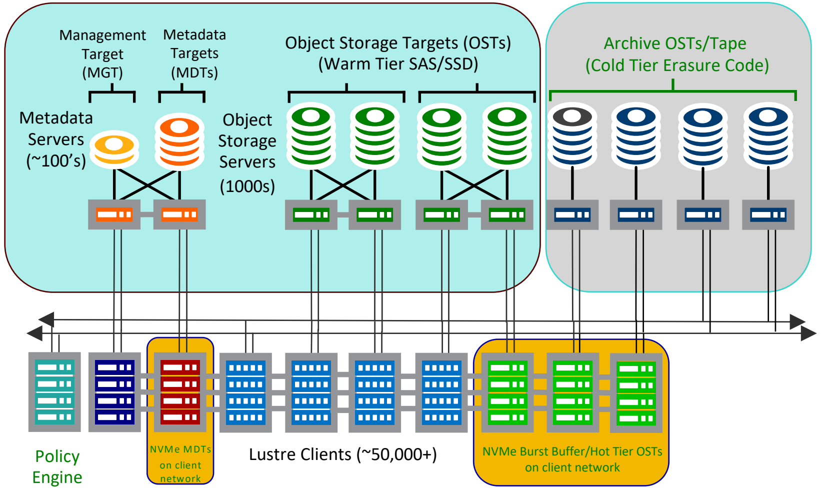
Whamcloud

- Integration with job scheduler and workflow for file 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 mechanism for safe migration of file data
  - RobinHood or Stratagem are good options for this
- Needs userspace integration and Lustre hooks
  - Integrated burst buffers a natural starting point
  - Mirror to flash, mark **PREFERRED** for read/write
  - Resync modified files off flash, release space

DONE

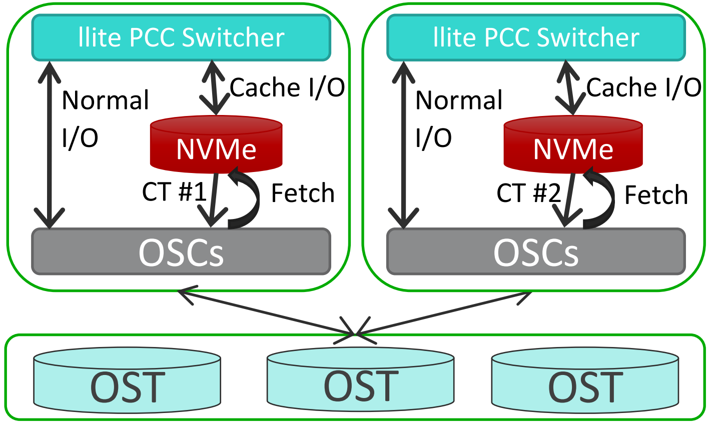
TODO

- Quota on flash OST/MDT (LU-11023 Cray)
  - Limit users from consuming all fast storage
- Integrate HSM into composite layout
  - Allow multiple archives per file (e.g. tape + S3)
  - Allow partial file restore from archive


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

Persistent Client Cache (PCC) (LU-10092 DDN, WC 2.13+)

- **Reduce latency**, improve small/unaligned IOPS, reduce network traffic
- PCC integrates Lustre with persistent per-client **local cache devices**
  - **A local filesystem** (e.g. ext4/ldiskfs) is created on client device (SSD/NVMe/NVRAM)
  - **Only data is local to client**, no global/visible namespace is provided by PCC,
  - Existing files pulled into PCC by HSM copytool per user directive, job script, or policy
  - New files created in PCC is *also* created on Lustre MDS
- Kernel uses local file if in cache or normal Lustre IO
  - Further file read/write access "directly" to local data
  - No data/IOPS/attributes leave client while file in PCC
  - File migrated out of PCC via HSM upon remote access
- Separate **shared read** vs. **exclusive write** cache
- Could later integrate with DAX for NVRAM storage



Encryption – Data at REST (WC 2.14+)

- Keep data safe against common attack vectors
  - Protect against storage loss/theft/return/mistakes/etc.
  - Protect against network/user/admin snooping
- Encryption at client RPC level, only client has keys
  - Data decrypted after RPC received from servers
  - Applications see clear text, in page cache on client
  - Data encrypted when building RPC to send to servers
  - Network/servers/storage/backups only see encrypted data
- Based on `fscrypt` library from `ext4` (for Android, don't invent own encryption!)
  - Can enable encryption with user key(s) per directory tree
  - Leverage client kernel crypto acceleration (CPU, dedicated hardware)
  - Per-file encryption key(s), itself encrypted by user key
    - Fast, secure deletion of file when key deleted with MDT inode, no overwrite needed
  - Filenames encrypted before sending to MDS, works with DNE

Whamcloud