ISC High Performance
The HPC Event.

Whamcloud

Lustre: The Next 20 Years

Andreas Dilger
Principal Lustre Architect
CTO Whamcloud

A Long Time Ago, In A Company Far, Far Away...

Stelias Computing

- Home
- About
- Projects
- News
- Press
- Relationships
- Contact

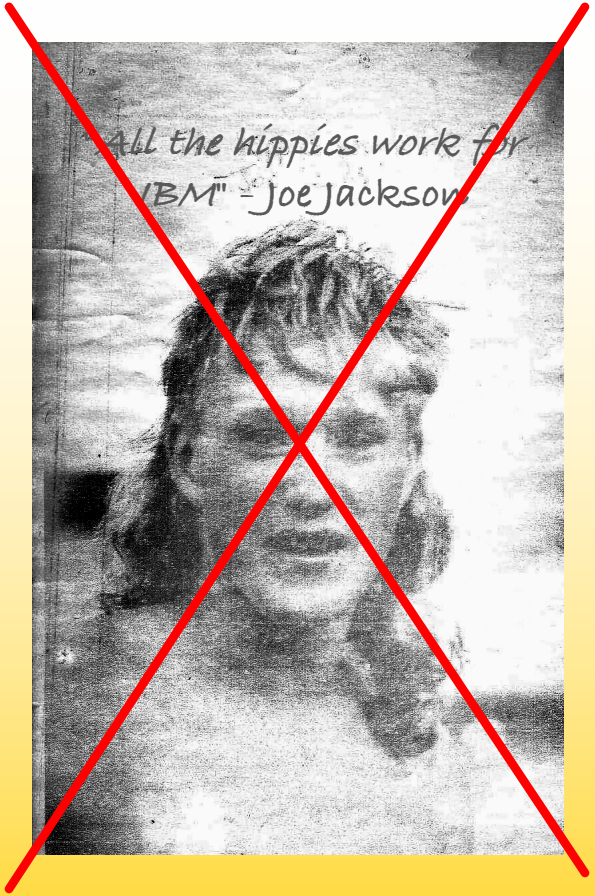
```
commit 139a736b8b98d38be9265b6e1fcf6b54ee4c0c71 (tag: refs/tags/0.0.0)
Author: pschwan <pschwan>
Date:   Thu Jun 3 01:34:05 1999 +0000

    foo
```

```
commit 906f38ee0bc23a1b153fde2e9bf1063ccb0f40c9 (0.0.0-22-g906f38e)
Author: adilger <adilger>
Date:   Wed Dec 1 18:45:16 1999 +0000

    Updated parameters for obdfs_writepage() to use struct *dentry instead of
```

```
commit 667e9cd3c1193c9e858512ced5ebccd26e0e6ab2 (0.0.0-108-g667e9cd)
Author:     braam <braam>
AuthorDate: Sun Mar 11 00:53:49 2001 +0000

    a working file system!
```


Source: http://www.myshared.ru/slide/137795/

Someone Had A Modest Goal...



Cluster File Systems, Inc

Someone Had A Modest Goal...

Lustre

The Inter-Galactic File System

Peter J. Braam

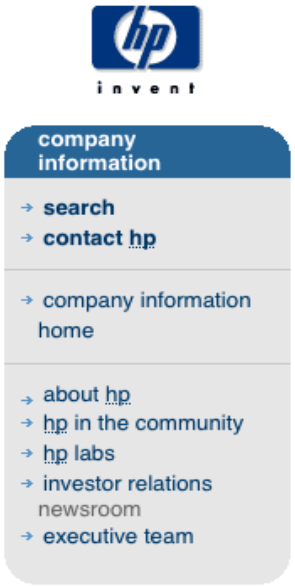
braam@clusterfs.com

http://www.clusterfs.com

Cluster File Systems, Inc

Source: *Lustre, The Inter-Galactic File System*, Peter Braam, 2002

And Brought It All Together...



U.S. Department of Energy Agency Selects HP to Co-develop Linux Software for Clustered Computing

File System Software Targeted for Use on Clusters of up to 10,000 Systems

PALO ALTO, Calif., Aug. 8, 2002

HP (NYSE:HPQ) today announced it has been chosen by the U.S. Department of Energy's (DOE) National Nuclear Security Administration (NNSA) to develop and deploy file system software for Linux clusters.

The joint research and development effort between HP and NNSA to develop the software, code-named Lustre, is a three-year project. HP is supplying program management, development and test engineering, hardware, services and support to the initiative in a cost-sharing arrangement with NNSA and the DOE labs, including Lawrence Livermore National Laboratory, Los Alamos National Laboratory and Sandia National Laboratories. HP is working in conjunction with Cluster File Systems, Inc., which is serving as a subcontractor on the Lustre project.

Lustre is a high-performance, highly scalable, Linux-based file system designed to work on large compute clusters that provide more than 100 teraflops with high demand for storage and input/output performance. Lustre will be made available initially to each of the DOE labs, including the Pacific Northwest National Laboratory, on HP's Linux-based high-performance computing cluster solutions.

***Stelias Computing* in Peter Braam's basement**

Object-based storage project for Seagate

- Ethernet-connected HDDs with embedded Linux
- OBDfs developed from 1999/06 to 2000/03
- `ext2` filesystem split in half with OBD API between
- Basic local-storage IO functionality demonstrated

Brief interlude at *TurboLinux* in Santa Fe, NM

CFS formed for ASCI Path Forward 2001/03

- One client+HDDs turned into distributed parallel fs
- US DOE required larger partners for project credibility
  - Enter HP+Intel for program mgmt., testing, etc. 2002/08

First production use on MCR (#3) at LLNL

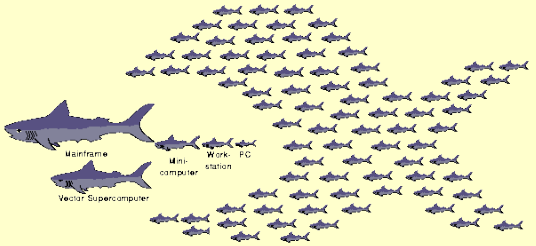
- First testing 2002/08, production 2002/11, 1.0 2003/12

Second install HPCS2 (#5) at PNNL in 2003/07

- Team lived & developed onsite for two weeks

 Source: *The Lustre Storage Architecture*, Phil Schwan, 2004

Top500 Cluster



NOW

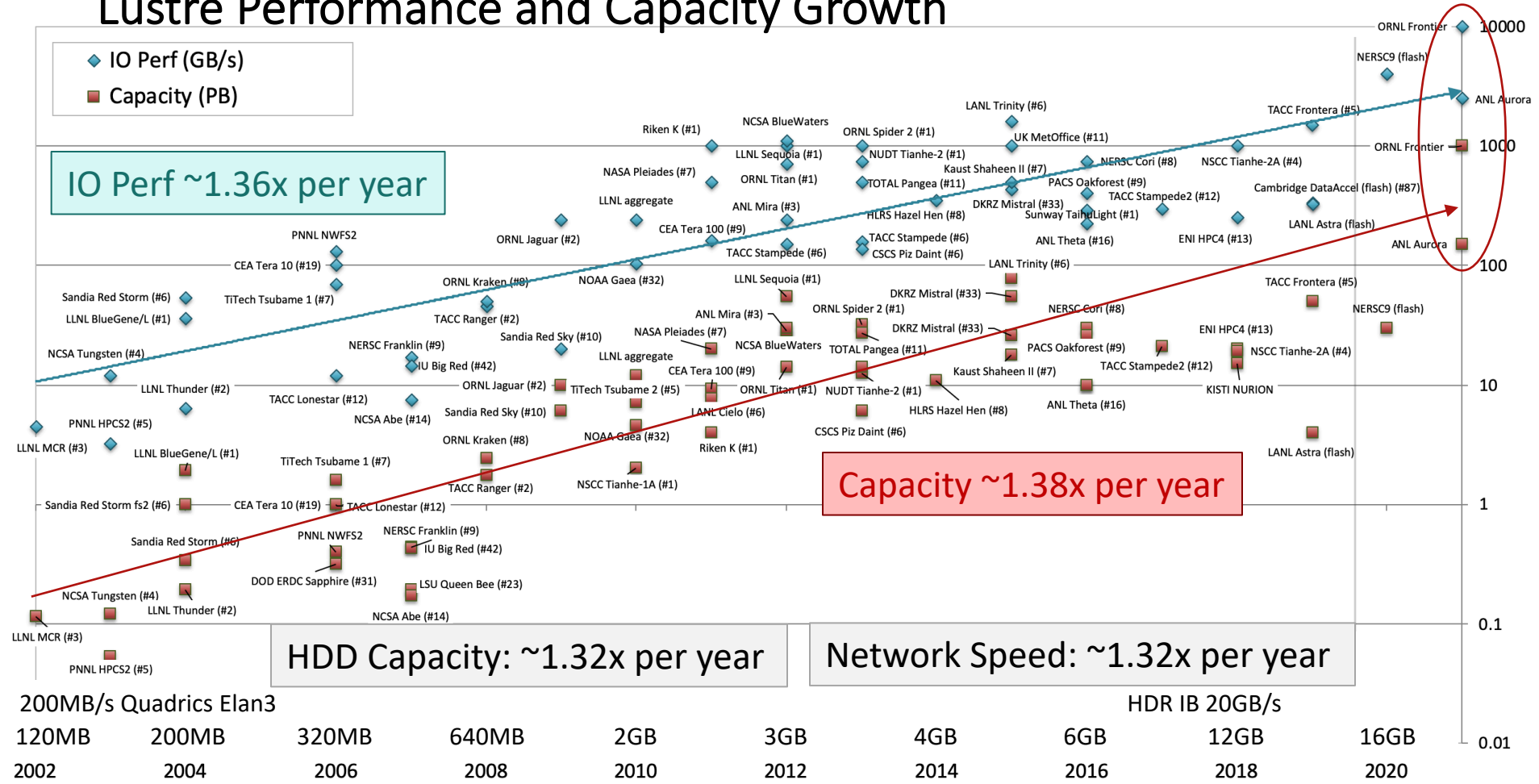
MCR LINUX CLUSTER
LLNL, LIVERMORE, CA
LINUX NETWORX/QUADRICS
R_{max}: 5.69 TFlops

- **11.2 Tflops Linux cluster**
- **4.6 TB of aggregate memory**
- **138.2 TB of aggregate local disk space**
- **1152 total nodes plus separate hot spare cluster and development cluster**
  - **4 GB of DDR SDRAM memory and 120 GB Disk Space per node**
- **2,304 Intel 2.4 GHz Xeon processors**
- **Cluster File Systems, Inc. supplied the Lustre Open Source cluster wide file system**
  - **115TB capacity, 4.48GB/s peak I/O speed**
- **Cluster interconnect: QsNet ELAN3 by Quadrics,**

A similar cluster with Myrinet connection announced for Los Alamos National Lab, planned for 2006

Source: *From the Earth Simulator to PC Clusters*, Desy, SC'02

Lustre Performance and Capacity Growth

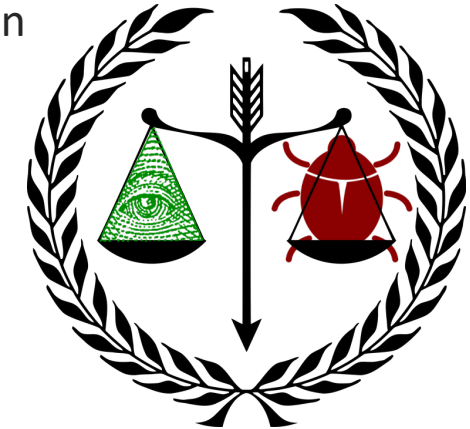


Source: [Rock Hard Lustre](), Nathan Rutman, Cray (with updates for recent years); [Disk Drive Prices (1955-2019)](), John C. McCallum

*You may ask yourself, "Well, how did I get here?"**

Whamcloud

- Good timing and some luck - there was a gap in storage at scale, Linux was new
- Innovative ideas providing strong architectural foundation
  - Robust protocol compatibility allows ***incremental*** and ***distributed*** feature development
  - Leverage existing disk filesystems/tools (ext4/e2fsprogs, later ZFS)
- Strong development team - great people, regardless of location
- **Balance** between funded NRE and production usage
  - Can't live in your own bubble too long, need to work with users
- **GPL** license helped contribution and avoided strangulation
  - We wouldn't be here without GPL keeping the code free
  - Major benefit from external contributors to Lustre, ext4, ZFS, Linux
- **Hard work**: bugzilla tickets: 24700, JIRA issues: 12460+
  - `Total files:  1861, Total lines of code: 1021026`
  - `Total LOC changed: 8049464, Commits: 19401, Authors: 351, Orgs: 42+`



* With apologies to David Byrne and the Talking Heads

Where Are We Going From Here?

Whamcloud

- New feature development and maintenance releases ongoing
  - Good base of features to build on, mature development and testing process
  - Feature release every ~7 months, new LTS every ~2 months or as needed
- Growing group of solid developers beyond those at Whamcloud
  - AWS, CEA, Cray, HPE, ORNL, SuSE, others contributed 25% of changes for 2.13.0
- Annual Lustre User Groups (USA, Paris, Tokyo, Beijing), BOFs (SC, ISC)
- Strong interest and usage from commercial cloud (AWS, GCP, Azure)
- Great showing in IO-500 this year

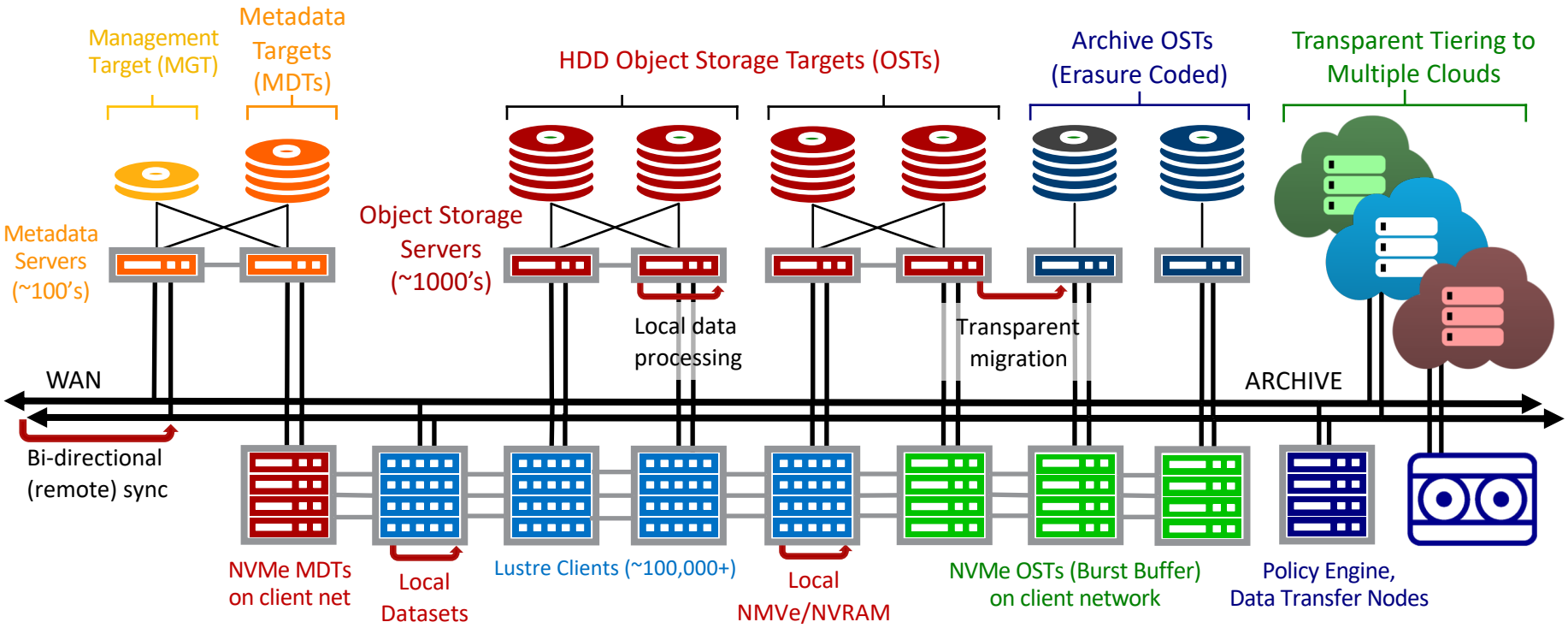
  #1 system @ Cambridge (3.9x speedup on same hardware over 2018-11 list)
- Continue to improve functionality/scalability as workloads/systems require

Source: IO-500 List 2019-06, Virtual Institute for I/O

Tiered Storage and File Level Redundancy

Data locality, with *direct access* from clients to all storage tiers as needed

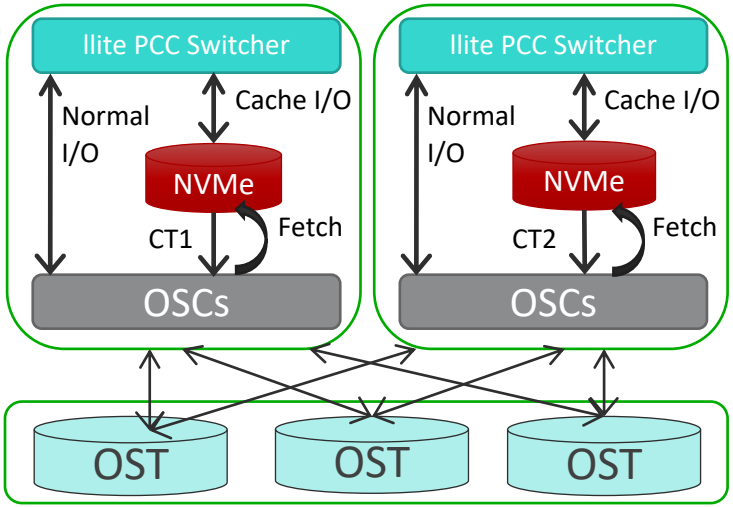
Whamcloud



Persistent Client Cache (PCC) (LU-10092 2.13+)

Whamcloud

- **Reduce latency**, improve small/unaligned IOPS, reduce network traffic
- PCC integrates Lustre namespace with client **local cache devices** (e.g. ext4 on NVMe)
  - Files pulled into PCC by HSM *copytool* per user directive, job script, policy
  - **Only file data is local to client**, global namespace is provided by Lustre
- Kernel uses PCC file, if present, or normal Lustre IO
  - Further file read/write access *directly* to local data
  - No data/IOPS/attributes to servers while file in PCC
  - Data migrated out of PCC via HSM on remote use/flush
- Separate **shared read** vs. **exclusive write** cache
- Future integration with **DAX** for **NVRAM** storage



FLR Erasure Coded Files (LU-10911 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	...

Client-side data processing

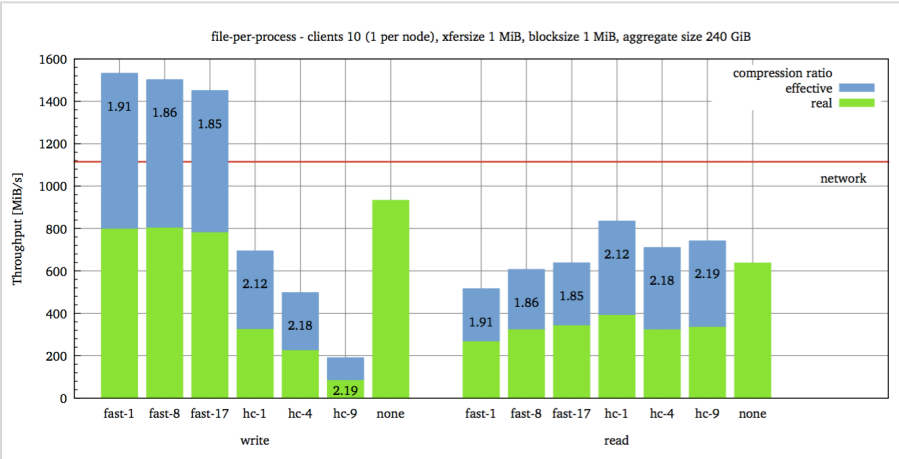
Whamcloud

Client-side encryption (LU-12275 2.15)

- End-to-end data protection
  - Data encrypted on wire and at rest
- Servers do not have keys, only users
- Per-user keys means data safety
  - Per-file derived key means safe deletion
- Based on `fscrypt` from Google/ext4
  - Don't invent own encryption!
- Accelerated encoding
  - AES CPU/QAT offload

Client-side compression (LU-10026 2.16)

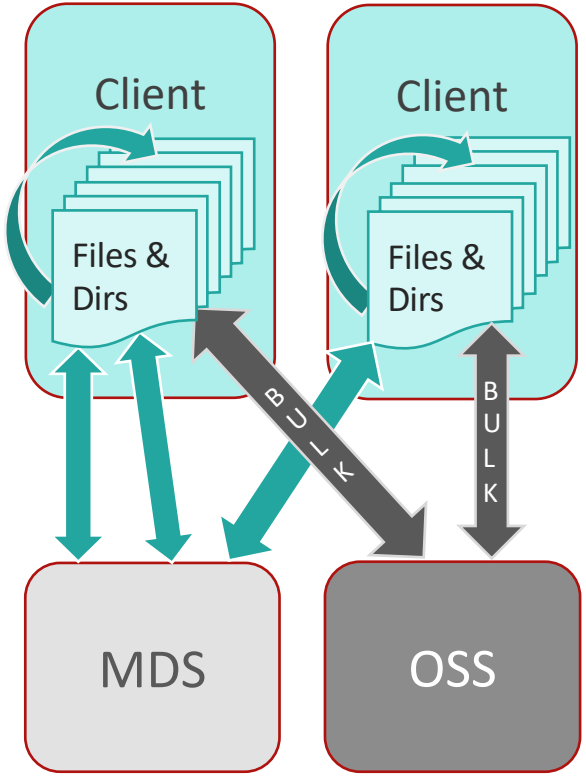
- Reduce data transfer to network/disk
  - Can read/write faster than network
- Data chunking to allow random IO
- Accelerated compression on client
  - Multiple cores or hardware offload



Source: *Enhanced Adaptive Compression in Lustre*, Anna Fuchs, LAD'16

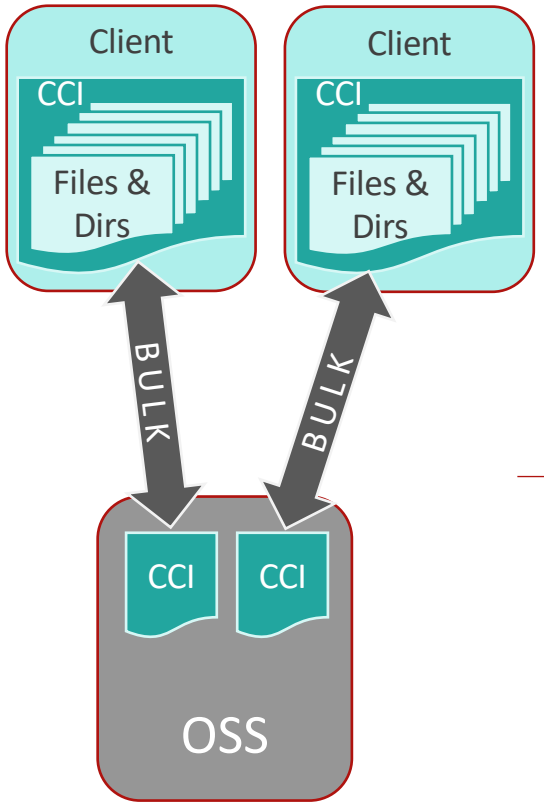
Client Metadata Writeback Cache (WBC) (LU-10983)

Whamcloud



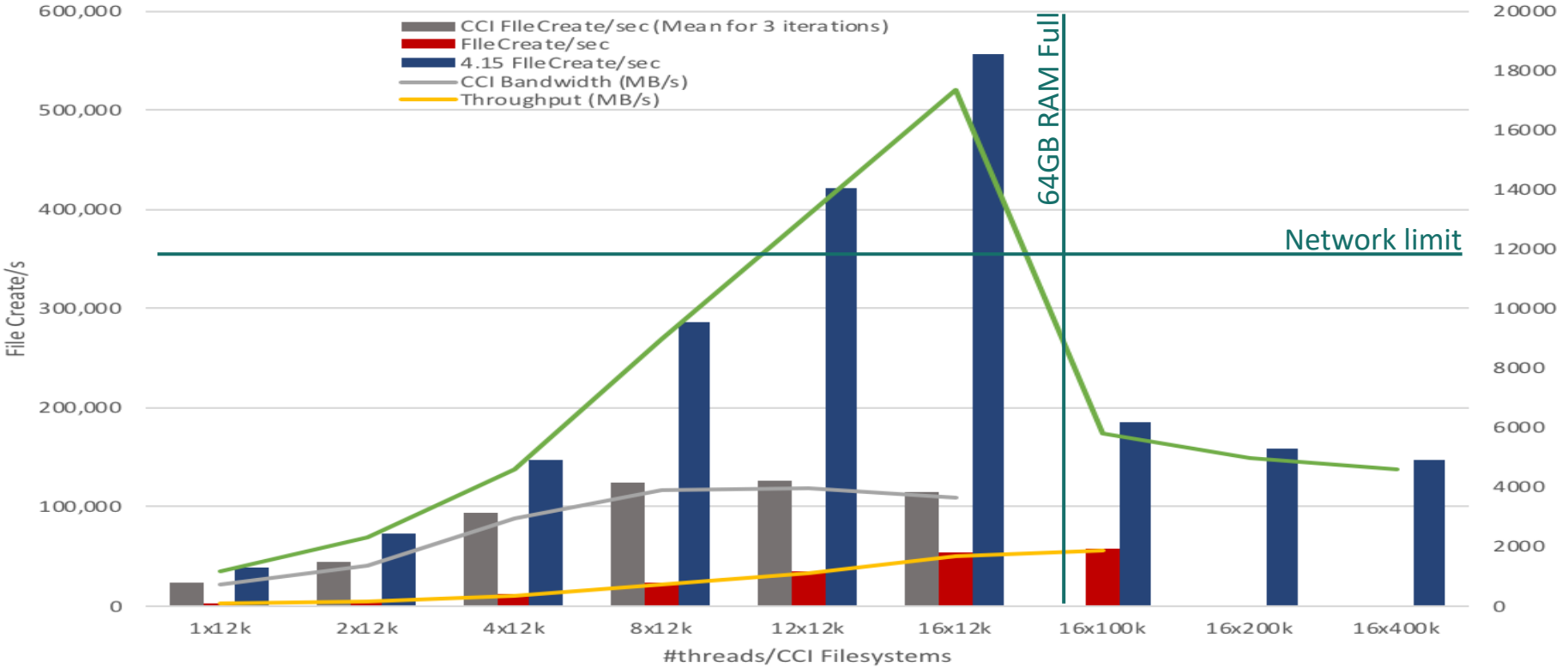
- WBC create/modify files in **RAM/local NVMe without MDS RPCs**
  - Exclusively lock new parent directory at `mkdir` time
    - No filenames in new dir or `readdir()` existing entries
  - Client can create new files and subdirectories therein with no MDS RPCs
  - Cache new files/directories only in RAM/local NVMe until cache flush
  - Flush files immediately to MDS due to other client access/lock conflict
    - Push top-level entries to MDS, lock new subdirs, repeat as needed
  - Flush rest of tree in background to MDS/OSS by age or memory pressure
- Changes globally visible on MDS flush to, ***normal access thereafter***
- Add aggregate operations to MDS to improve performance
  - Batch operations on network to reduce RPCs, on disk to reduce IOPS
- Basic WBC *prototype* developed to test concept
  - No cache/quota/space limits, no background flushing, no batching, ...
  - Early tests show **10-20x *single-client*** speedup tests (`untar`, `make`, …)

Client Container Image (CCI)



- Filesystem images are used *ad-hoc* with Lustre today
  - **Read-only cache** of many small files manually mounted on clients
  - **Root filesystem images** for diskless clients/VMs
- **CCI is ldiskfs/zfs image** file loopback mounted on client
  - Whole directory tree (maybe millions of files) stored in one CCI file
  - **Best for self-contained workloads** (AI, Genomics, ensemble runs) DONE
- CCI integrates container image handling with Lustre TODO
  - Image is registered to Lustre directory for control future access
  - **Transparently mounts** registered image at client on directory access
  - Image data blocks read on demand from OST(s) and/or client cache
- Leverage *foreign layout* support added for DAOS namespace
  - CCI can bind image junction type into namespace

Single Client 32KB File Create Performance (MDS vs. CCI)

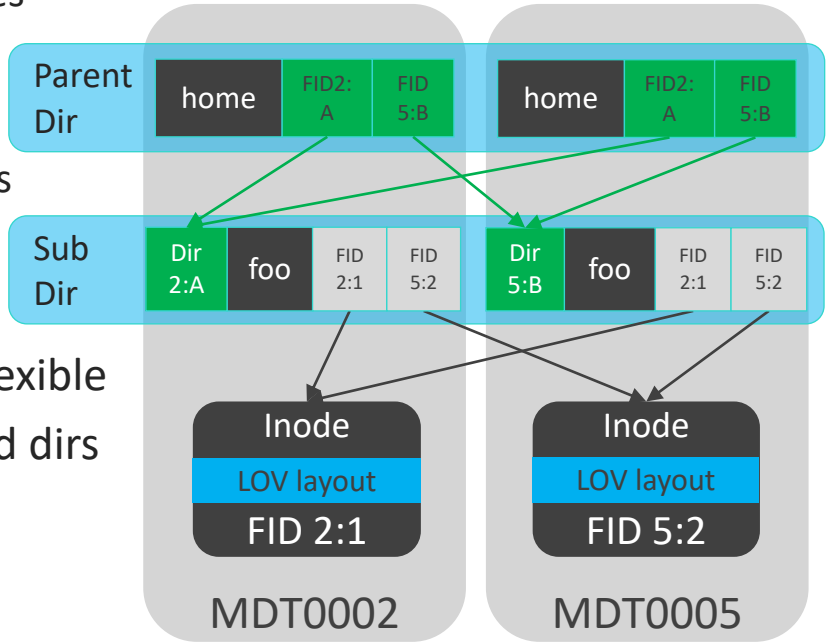


- 4.15 CCI improvement due to Ubuntu 4.15 kernel loopback driver
- Early testing of CCI prototype shows promise

DNE Metadata Redundancy

Whamcloud

- 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 update recovery
  - Ensure that copies stay in sync on multiple MDTs
- Redundancy policy per-filesystem/subtree is flexible
- Flexible MDT space/load balancing with striped dirs
- Early design work started, discussions ongoing



Leverage New Storage & Networking Technologies

- PCC will be able to expose in-client NVRAM directly to applications via DAX
  - NVRAM over the network is much less useful than on client
- NVRAM-local MDT OSD when economically viable at scale
  - NOVA is an example of local Linux NVRAM filesystem, may be suitable for MDT
  - Lower overhead from byte-granular storage updates, faster storage
- Track I/O through entire application lifecycle
  - Label I/O from application to server, in storage, account IOPS like CPU cycles
- Users will know even less about how their applications perform I/O
  - Concentrate knowledge and experience in application libraries
- Closer integration/interfaces between applications, libraries, and storage
  - Always-on I/O tracking of prior application runs automatically tunes later runs
  - POSIX will continue to be important, with (*ad-hoc*) extensions/bypass/layers

What do things look like further into the future?

Whamcloud

- No crystal ball is available, but there is always time before it arrives
  - A single OSS is now faster and larger than an entire Top-10 filesystem from 2002
  - PB-sized all-flash storage with millions of IOPS and TB/s wasn't conceivable in 1999
  - Lustre can run this today and continues to improve across workloads
- Storage will continue to improve in multiple dimensions
  - Leveraging those improvements does not happen by itself
  - Consider interactive performance of user interfaces vs. CPU speedup...
- Software R&D must continue to find new ways to leverage new hardware
  - Several year effort to interface Linux kernel (DAX) with persistent memory

Lustre Feature Roadmap

Lustre (Lite) 1.0 **(Linux 2.4 & 2.6)**	**Lustre 2.0** **(Linux 2.6)**	**Lustre 3.0**
2003	2004	2005
Failover MDS	Metadata cluster	Metadata cluster
Basic Unix security	Basic Unix security	Advanced Security
File I/O very fast (~100's OST's)	Collaborative read cache	Storage management
Intent based scalable metadata	Write back metadata	Load balanced MD
POSIX compliant	Parallel I/O	Global namespace

 Source: *The Lustre Storage Architecture*, Peter Braam, 2003

Whamcloud

Performance Improvements 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

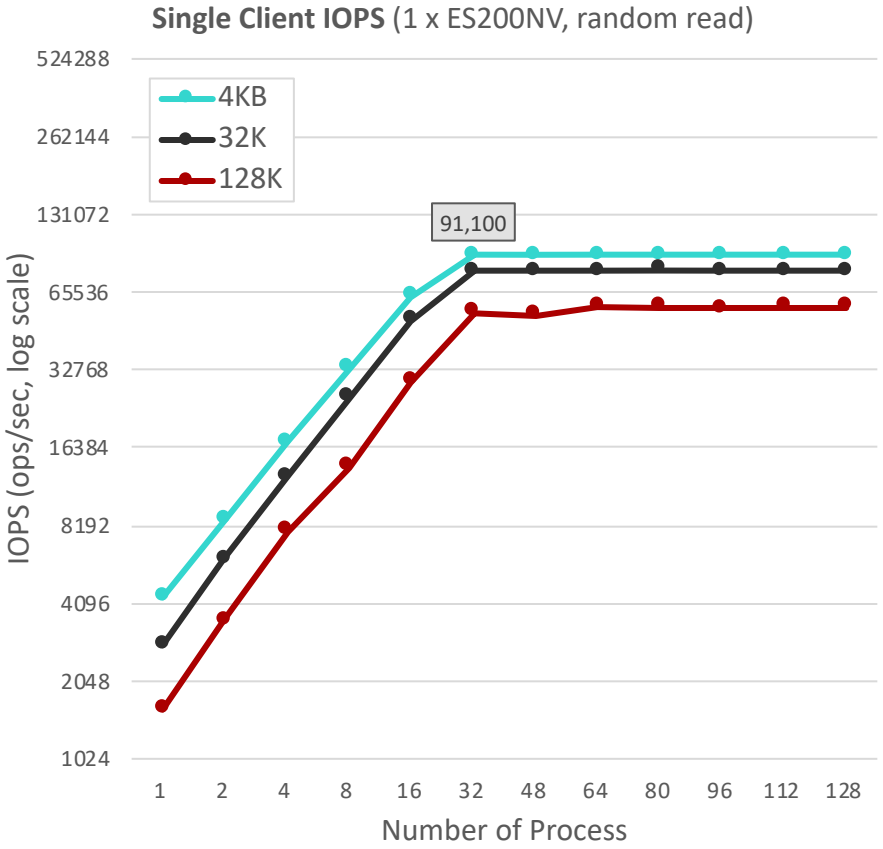
DONE

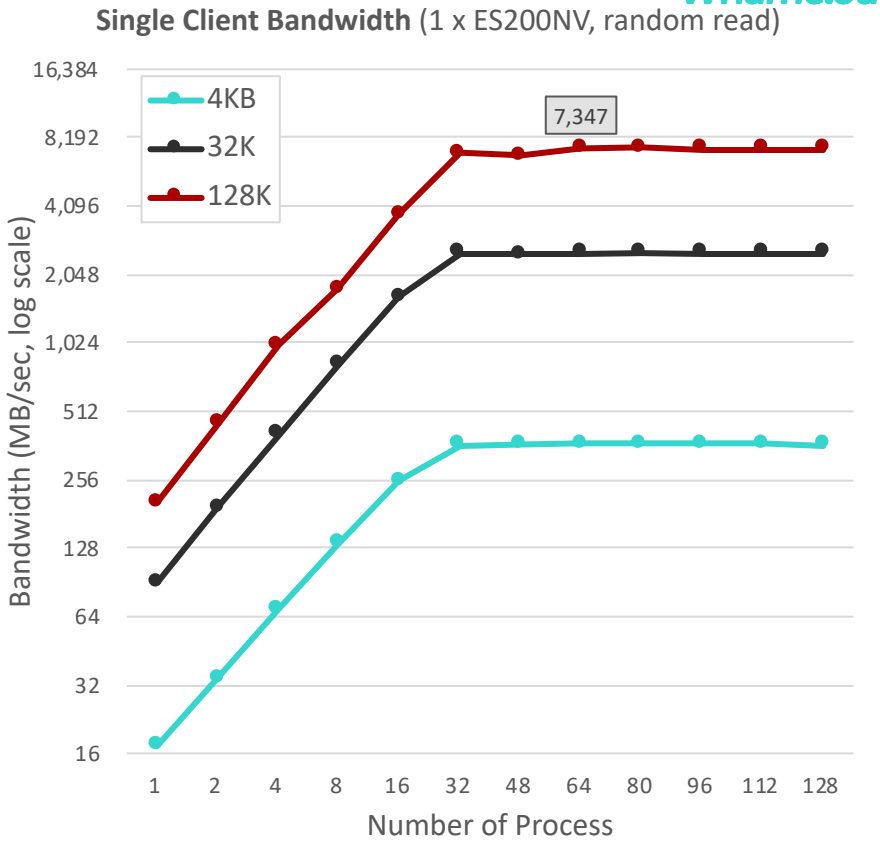
- TRIM of backend flash storage (LU-11355)
  - Access backend filesystem `fstrim` functionality
- Improve performance of small, unaligned writes
- Improved efficiency of ZFS IO pipeline
  - Integrate with ABD in ZFS 0.8 to avoid `memcpy()` of data

TODO

Single Client IOPS and Bandwidth (1 client, 1x ES200NV)

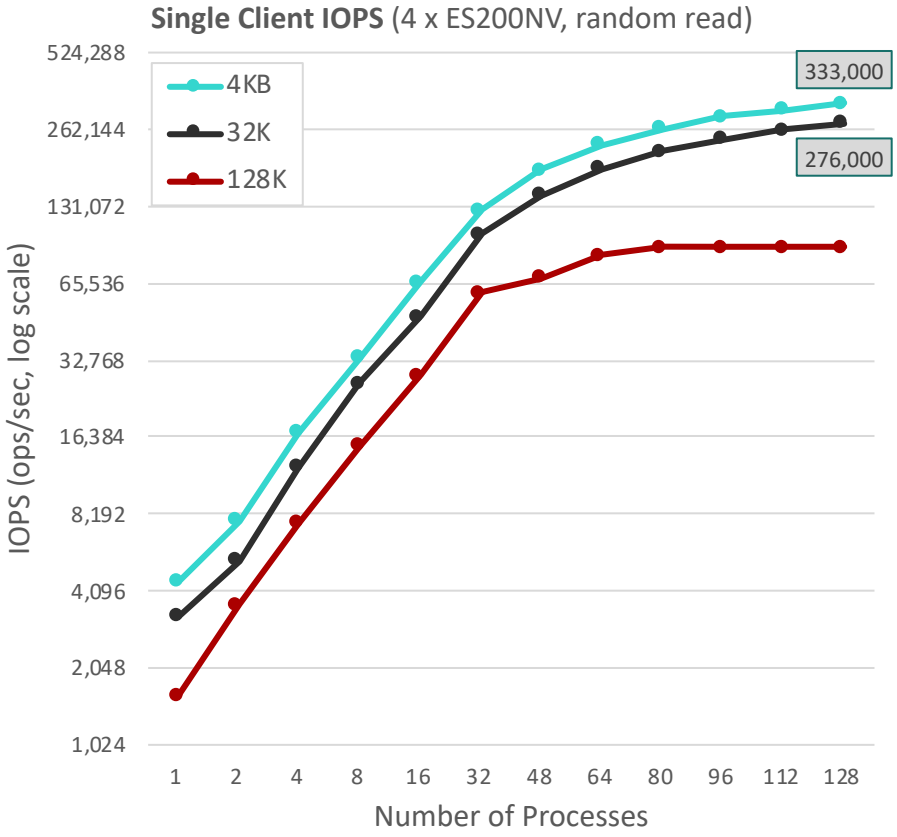
Whamcloud

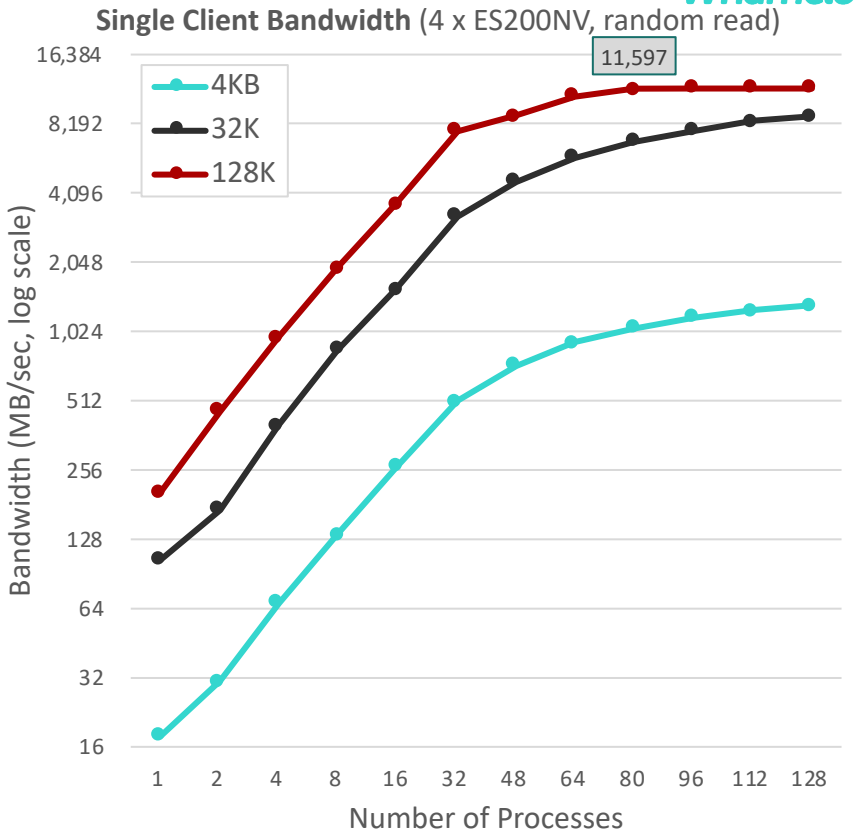




Single Client IOPS and Bandwidth (1 client, 4x ES200NV)

Whamcloud



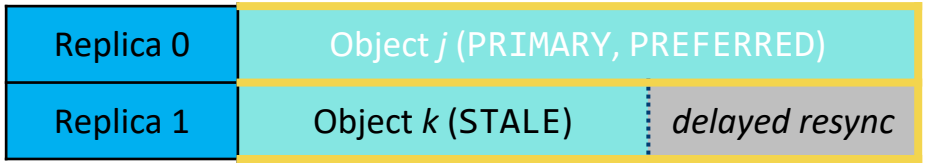


File Level Redundancy (FLR) Enhancements (WC 2.12+)

Whamcloud

- Allow Lustre-level mirroring for files, can be configured arbitrarily per file/directory
- 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

- Improved replica selection at runtime (LU-10158) TODO
  - Select best write replica (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.)
- Mount client directly on OSS for improved resync performance (LU-10191)
- Support DoM components (LU-10112)
- **OST/MDT quotas** (LU-11023, Cray)
  - Track/restrict space usage on flash OSTs/MDTs



Tiered Storage with FLR Layouts (2.12/2.13)

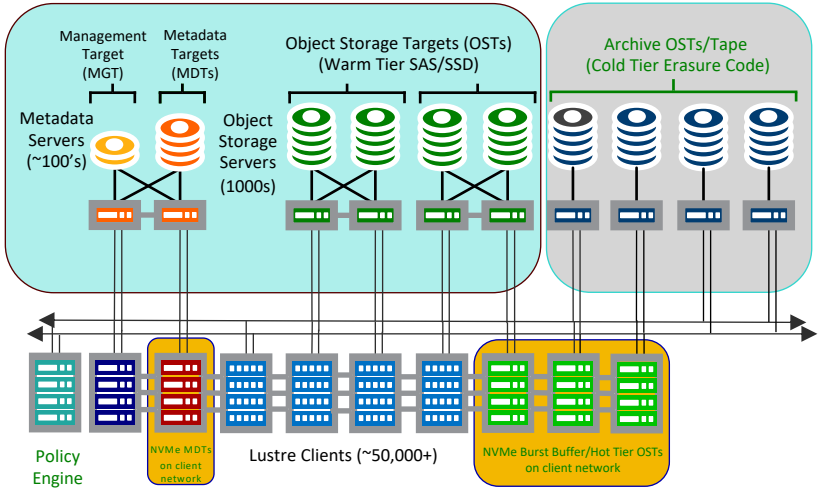
Whamcloud

- 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

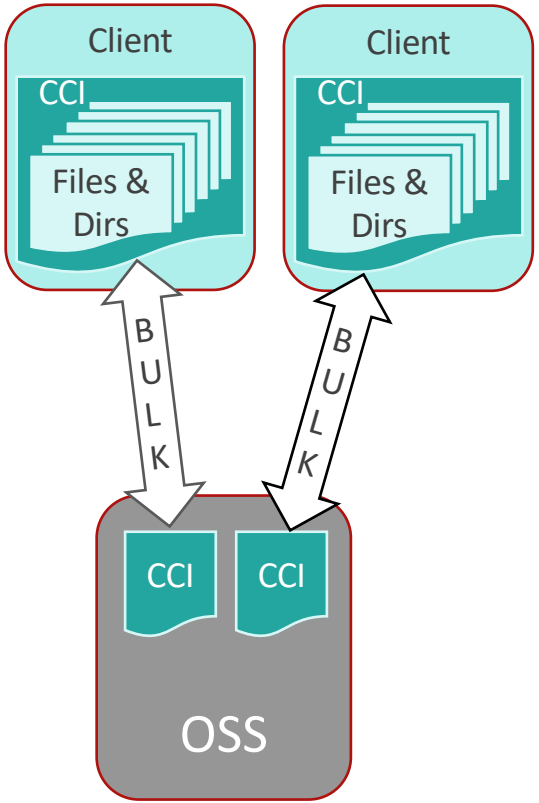
DONE

TODO

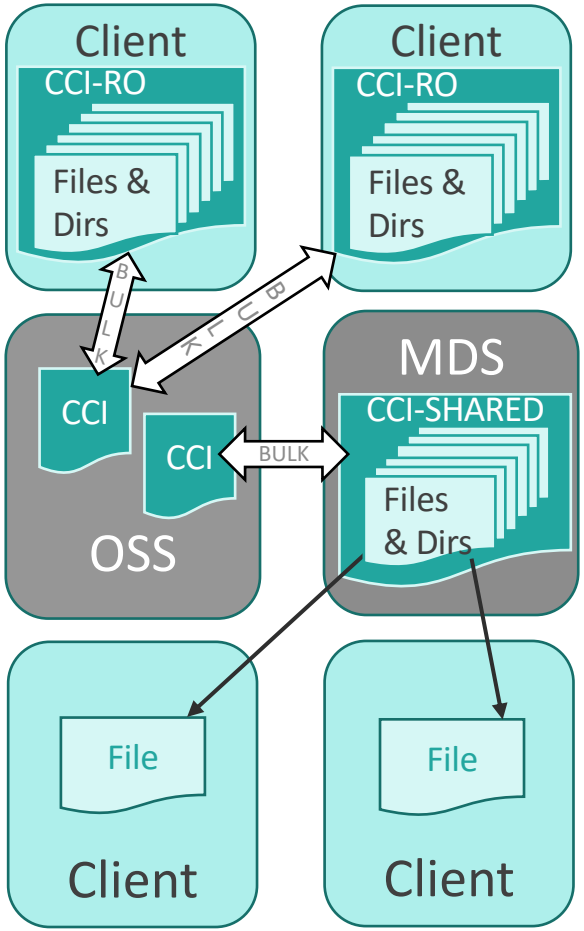
- Quota on flash OST/MDT (LU-11023 Cray)
  - Limit users from consuming all fast storage
- Integrate HSM into composite layout
  - Allow multiple archives per file (tape, S3)
  - Allow partial file restore from archive



CCI Performance Gains



- **Low I/O overhead, few file lock(s), high IOPS per client**
  - Readahead and write merging for data and metadata
  - Client-local in-RAM filesystem operations with very low latency
- Access, migrate, replicate image with large bulk OSS RPCs
  - Thousands of files aggregated with MB-sized network transfers
  - Leverage existing high throughput OSS bulk transfer rates
  - 1GB/s OSS read/write is about 30,000 32KB files/sec
- Unregister+delete CCI to remove all its files with few RPCs
  - Simplifies user data management, accounting, job cleanup
  - Avoid MDS overhead when dealing with groups of related files



CCI Access Models

Whamcloud

- Need to integrate image handling on Lustre client/MDS
  - Integrate CCI creation with job workflow is easiest
  - CCI layout type on parent directory creates CCI upon mkdir
  - Enhance ldiskfs online resize to manage image size
- Client ***exclusively*** mounts CCI(s) and modifies locally
  - For initial image creation/import from directory tree
  - For workloads that run independently per directory
- Multiple clients ***read-only*** mount single image
  - Shared input datasets (e.g. gene sequence, AI training)
- MDS exports ***shared read-write*** image to many clients
  - Internal mount at MDS attaches image to namespace
  - Use Data-on-MDT to transparently export image tree to clients
- Process whole tree of small files for HSM/tiering
  - Efficiently migrate tree to/from flash tier, to/from archive

Comparison and Summary of WBC vs. CCI

Whamcloud

Metadata Writeback Cache

- Keep normal namespace
- **Transparent to users**
- Very low latency metadata operations
- **Faster single client**
- Network **batch RPCs** improves other ops
- **Lower total overhead** due to fewer layers

Client Container Image

- **Segregated directory subtree**
- Needs input from user/job
- Not for all usage patterns
- **Faster total performance**
- Network **bulk IO** reduces MDS workload
- File aggregation simplifies dataset management (e.g. fast unlink)
- **Metadata tiering/HSM**

- Significant improvements for evolving HPC workloads
- Leverages substantial functionality that already exists