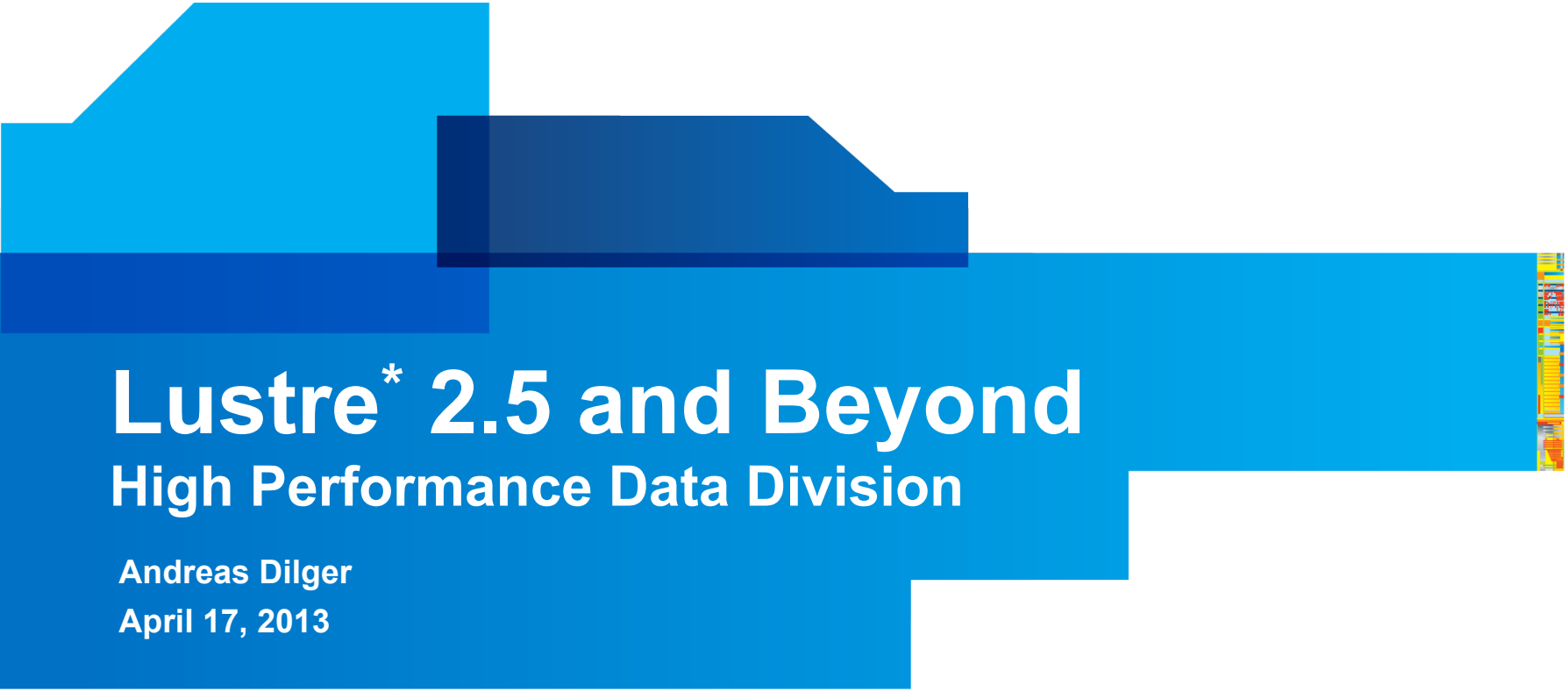


Lustre* 2.5 and Beyond

High Performance Data Division

Andreas Dilger

April 17, 2013

Feature Submission Process

- T minus 3 months: Features must be **LANDED**
  - Feature description and design in Jira ticket
  - Patch submission must be started well before cutoff date
  - Need to test, inspect, update, retest, integrate with other new features
- T minus 2 months: Documentation and test plan completed
  - Plan for new functionality/performance/load testing
  - Manual updates for user-facing features/tunables
  - Unix man pages for tools, APIs better if with patch itself
- T minus 1 month: only bug fixes landed after this point
- No features are guaranteed to be in any release
  - Train model only includes features that are ready by cutoff

http://wiki.opensfs.org/Lustre_Community_Development_in_Progress

Features Discussed In Other Presentations

- Distributed Namespace (DNE) – Intel
  - Asynchronous remote updates, rename, migration (2.5)
  - Stripe/Shard single directories over multiple MDTs (2.6)
- Lustre File System Check (LFSCK) – Intel
  - MDT/OST consistency check/repair (2.5)
  - MDT/MDT consistency check/repair (2.6)
- DAOS Exascale Infrastructure – Intel
  - Scalable broadcast network / Health Network (2.6/2.7)
- FEFS Features – Fujitsu
  - LNET Routing, big-endian servers, memory reduction, others (2.5/2.6/2.7)

Data Migration – CEA (2.4/2.5)

- Uses Layout Lock and object data version from HSM
- Move file data between OSTs safely
  - Restripe files after creation
  - Move files between storage tiers/pools for performance/space
  - Evacuate OST for service or removal
- Files can be open and in use
  - In case of conflict, IO optionally blocked or migration aborted
  - MDT inode stays the same, open file handles preserved
- `lfs migrate` used by `lfs_migrate` script if layout lock available
- Needs policy engine in order to be really useful

Hierarchical Storage Management – CEA (2.5)

- Copy Tools initially for HPSS and POSIX archives
- Uses CEA Robin Hood for policy engine
  - Leverages Lustre ChangeLog to avoid scanning
- Client-side changes largely landed in 2.4
  - Layout lock, object data version, copytool APIs, RPC protocol
- Server changes currently under development
  - MDS Coordinator, File Release, Copy Tools
- Infrastructure usable for migration, replication, …

Client Extended Attribute Cache – Xyratex (2.5)

- Fetch all xattrs from MDS with on inode lookup
  - Avoid RPC round-trip for *each* xattr access
  - Avoid RPC round-trip for xattrs that do not exist
- Cache xattrs on client under MDT IBITS lock
- Important for Samba/CIFS exporting performance
- Important for SELinux and similar labeling systems

Client Updates (2.5/2.6)

- Broad interest in community in this work
  - Contributions from EMC, ORNL, SuSE, LLNL, Intel , NRL
- Desire to include Lustre client in upstream Linux kernel
  - Easier for customer installations, reduce/eliminate lag for new kernels
- Need to clean up ten years of legacy code
  - Linux coding style is only a small part of this
  - Remove Solaris/WinNT/MacOS cfs_ API wrappers (*entire code?*)
  - Separate client and server functionality
  - Simplify abstraction layers in the client IO stack
  - Update code to use newer Linux VFS/VM interfaces
  - Remove dead code (via static code analysis)
  - Update build system (cleanup, DKMS packages)

Shared Key Crypto – Indiana University (2.6)

- Simplified node authentication and RPC encryption
  - WAN or other separate administrator domains
  - Uses existing Lustre GSSAPI/sptlrpc infrastructure from Kerberos
- Shared secret key is known by clients and servers
  - Key distribution external to Lustre (USB key, phone, (e)mail, pigeon)
  - Different keys for different client clusters
  - Servers can understand multiple keys per cluster
  - Rotate keys as needed, lifetimes can overlap
- Authenticate remote *nodes* instead of *users* like Kerberos
- Uses AES-128 encryption
  - Flexible to allow other encryption in the future

UID Mapping – Indiana University (2.6)

- Multiple clusters with different UID/GID maps
  - WAN or other separate administrator domains
  - Maps are maintained only on a cluster granularity
- Remote cluster nodes defined by client NID range
  - Optionally authenticated by shared-key authentication
  - *Cluster* can be one node or a whole campus
- Map remote UID/GID to MDS-local values on MDS
  - Does not need any changes to remote clients
  - Store remote UID/GID in MDS-local rage
- Map remote UID/GID on OSS for quota

File Replication – Intel proposal (2.5/2.6/2.7)

- Mirror files across multiple OSTs (RAID-1+0)
  - Redundancy in case of OST failure (disk, network, software)
  - Read performance of hot files
- Phase 1: read-only replicas
  - Copy file in userspace then merge copy onto original inode
  - Implement reads, layout, failure handling, no overhead for write
  - Different layouts for each copy (tiered storage, RAID-1+6?)
- Phase 2: synchronous replication
  - Send each write to multiple OSTs, wait for commit
  - Implement writes, immediate redundancy, write overhead
- Phase 3: asynchronous replication
  - Send each write to multiple OSTs, no waiting
  - Complex recovery model (needs DAOS features)

T10-PI Data Integrity – Xyratex (2.6)

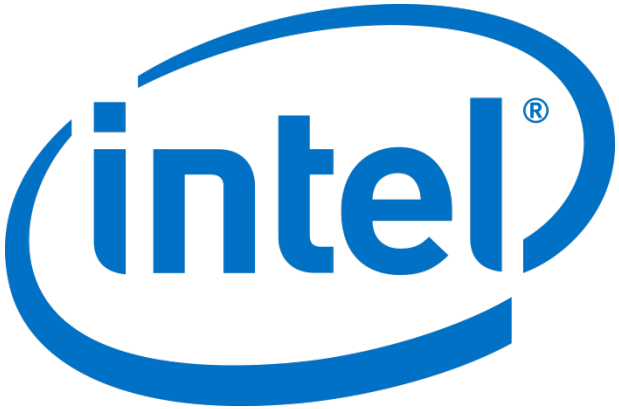
- SCSI Standard, implemented in some HBAs and disks
- Data integrity from syscall interface to disk for each sector
  - 16-bit Guard Tag (CRC or IP checksum)
  - 32-bit Reference Tag (low bits of sector address)
  - 16-bit App Tag (from application, if there was an API)
- Computed on client for each page (N sectors) on write
  - Kept with page in cache until RPC is generated
- Sent with each sector in RPC for read/write
  - Optionally in RPC request (increases request size), or with bulk data transfer
- Returned from server for each page (N sectors) on read
- Validated by peer, resend RPC on error

Small Files on MDT – Intel Proposal (2.6)

- Combine MDS and OSS into Unified Target
  - Remove duplication of common code
  - Allow client to read/write file data from MDT, index on OST
- Store small files on high-IOPS MDT storage
  - Reduce RPCs for small files (attrs for size, locks, RAID r-m-w)
  - Migrate file data to OST if it grows too large
  - Small-file workloads may only have MDTs?

Btrfs OSD – Intel Proposal (2.6)

- Desire to have advanced backing filesystem features
  - Snapshots, checksums, copy-on-write
  - ZFS is great and in 2.4, but has unfortunate license issues (real or not)
- Need to investigate and develop OSD layer for Btrfs
  - Lustre is ready for different OSD abstractions
  - Btrfs is slowly becoming defacto filesystem for desktops
  - Is Btrfs is ready for Lustre? (performance, reliability, code, tools)
  - People will want to use it anyway, and it will improve quickly
- More work also needed in Lustre to use advanced features
  - Distributed coherent snapshot mechanism
  - Access multiple datasets and object versions


intel