Whamcloud

In The Works

Sébastien Buisson

Principal Engineer – Lustre Architect

Whamcloud In The Works

- Multi-Tenancy for Lustre
- FLR Erasure Coded Files
- Fault Tolerant MGS
- Trash Can Undelete
- Client Hybrid IO
- Metadata Write-Back Cache
- Lustre Metadata Redundancy
- IPv6
- ldiskfs

Multi-Tenancy for Lustre

Whamcloud

- Support various populations (tenants) on a single file system
  - Isolate users and their data for legal/operational reasons
  - Cloud Providers

- Dynamic NIDs
- Lustre Quota Aggregation
- GSS identification
- Nodemap

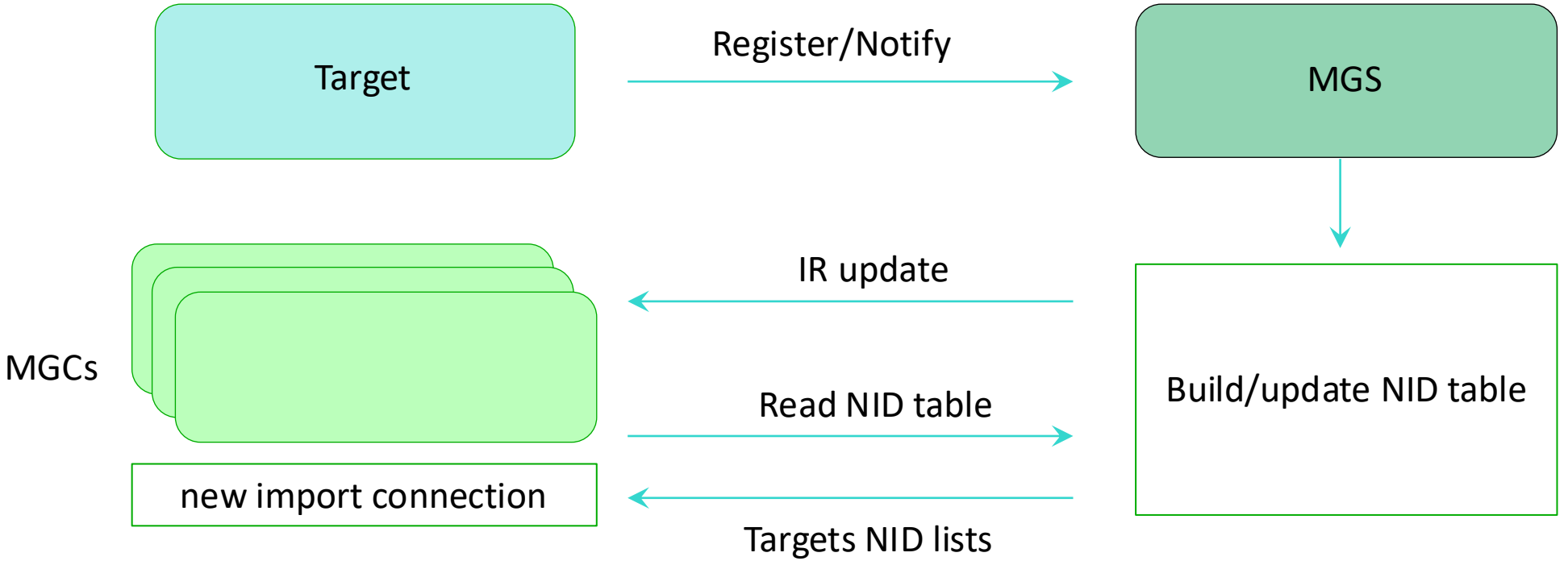
Dynamic NIDs

Whamcloud

- Multitenancy may require one LNet per tenant
  - All Lustre targets must have a NID on each LNet
- Configs cannot maintain so many NIDs
- Configs should not expose all LNets to all tenants

- Solution is to use Imperative Recovery for clients to get NIDs from MGS
  - NID tables on MGS should be reliable and consistent
  - NID tables access should be scalable

Dynamic NIDs



Lustre Quota Aggregation

Whamcloud

- In the context of multi-tenancy
  - Tenant admins might need to manage quotas inside the tenancy.
  - But global Lustre fs must enforce tenant quotas to avoid block/inode abuse.
- LQA enables aggregating and limiting usage across multiple quota IDs
  - Many distinct LQAs can be created in a single filesystem
  - LQA consists of one or more non-overlapping ID ranges
  - Quota limits are set per LQA for each quota type (usr/grp/prj)

GSS nodemap identification

Whamcloud

- In the context of multi-tenancy
  - Client-to-tenant distribution can be evolving and difficult to maintain.

- Nodemap traditionally relies on NIDs for client identification
- But GSS auth can be leveraged to provide an alternative approach
  - Simplifies nodemap isolation implementation
  - Provides more flexibility

GSS nodemap identification

Whamcloud

- New nodemap isolation implementation
  - Identify clients used by tenant
  - Install SSK key associated with tenant
  - Enable GSS identification on nodemap for tenant

⇒No need to update nodemap definitions when tenant clients are moved across nodes, only SSK key is needed

Other Nodemap improvements

Whamcloud

- Multiple filesets
  - Nodemap fileset restricts clients to a sub-directory, but one fileset may be too limiting
  - One primary default fileset + alternate filesets allow access to subdirectories (can be RO)
- Dynamic nodemaps
  - For ephemeral use cases that require frequent nodemap changes, e.g., in compute jobs
- OST object tagging with parent FID and UID/GID/PROJID
  - Allows scanning the OST and getting the MDT FID from objects, e.g., when migrating files off an OST
  - Check OST objects against nodemap ID/offset mapping
- Per-nodemap capabilities mask
  - Finer-grain capabilities mask compared to global enable_cap_mask parameter

FLR Erasure Coded Files

Whamcloud

- Erasure coding adds data redundancy without 2-3x mirror overhead
  - Improve data availability above hardware and network reliability
- Read-Only EC
  - “Read-Only” because mostly valuable in read-only workflows
  - After a file is written, an EC component can be added to it
    - If file is modified, EC needs to be recomputed.
  - If OSTs are unreachable, missing objects can be reconstructed from EC parity.
- Immediate Write Mirroring
  - Write plain mirrors (full object copies) as part of the write transaction initiated by the client
- Read-Write Erasure Coding
  - Write EC parity as part of the write transaction initiated by the client application.

Fault Tolerant MGS

Whamcloud

- The MGS is a central piece of a Lustre cluster
  - Required for new clients to mount
  - Used to speed up recovery with Imperative Recovery.

- Making MGS fault-tolerant provides:
  - Availability: using the service running on replicated nodes
  - Reliability: avoid configuration files loss/corruption
  - Performance: shortens mount time for large clusters

Fault Tolerant MGS

Whamcloud

- Run MGS service on multiple MDS nodes for availability
  - Allow clients to get config logs from **any MGS node**
  - MGS Imperative Recovery even if “primary” MGS node restarts

- Mirror MGS config logs to remote MDTs for redundancy
  - Use RAFT Consensus algorithm to coordinate MGS cluster
  - MGS Leader election, heartbeat, consistent log updates

Trash Can Undelete

Whamcloud

- Allow files/directories to be undeleted after `rm`/`rmdir`
  - Rescue users from fat-finger mistakes or malicious scripts
  - Increase tolerance to mistakes by providing a “second chance”
- Files are moved to Trash Can instead of being unlinked
  - Use the same “rm” command or unlink syscall, no need for specific command
  - Files removed from user/group/project quota and `df` usage
    - *Feels like* space is freed when moved to Trash Can
- Virtual `.Trash` directory accessible in every directory
  - Users can use normal tools to list and recover files or directories
  - Files cannot be read to avoid abuse
  - `.Trash` is hidden from normal directory listing

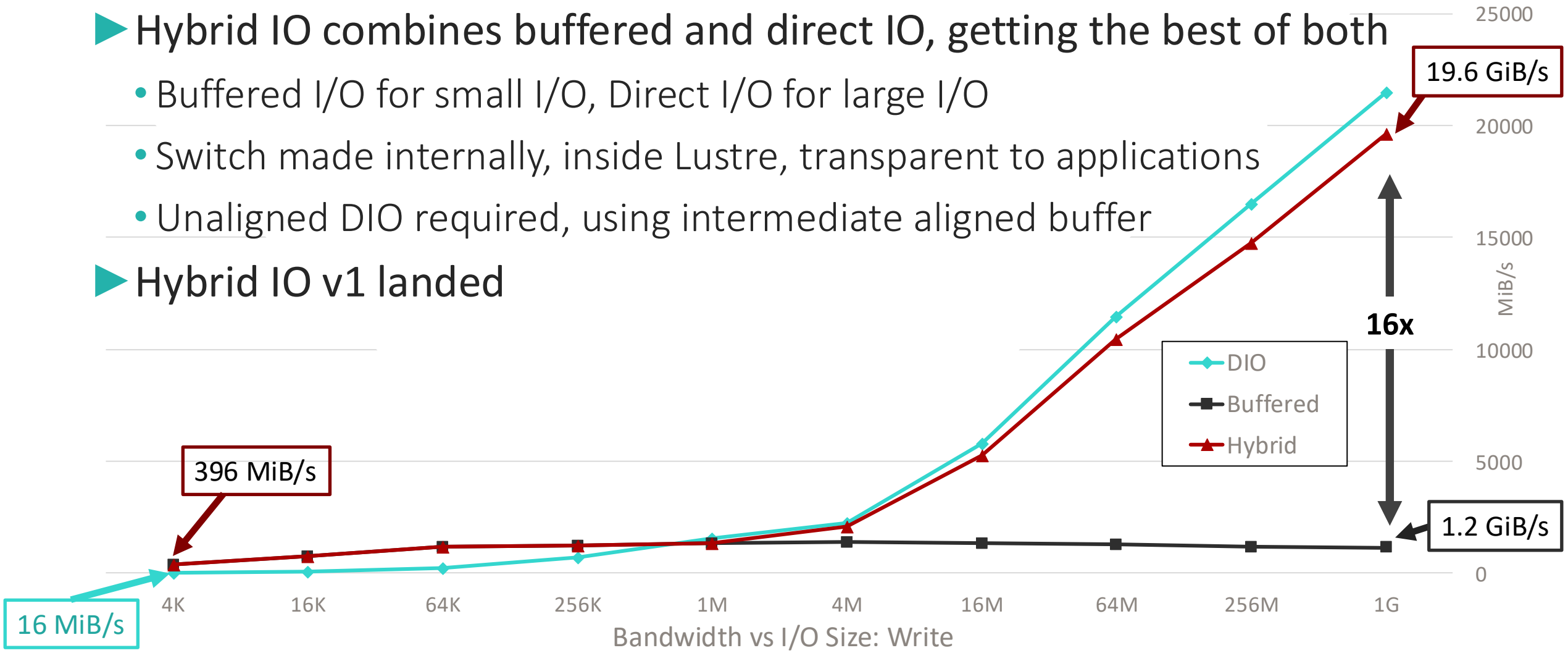
Trash Can Undelete – cont’

Whamcloud

- Trash Emptying
  - Files in TCU can be erased all together, e.g. to clean up disk space
- Trash Can space management: ltrash_purge tool
  - Configurable expiry time before cleanup (e.g. max age = 7d)
  - Configurable filesystem fullness threshold (e.g. 80% full)
  - More sophisticated cleanup policy in userspace (e.g. by user, project, nodemap)
- Enable TCU only when needed
  - Admin has system-wide control on enabling/disabling TCU
  - Users can disable/enable TCU on specific directories/files (TODO)

Client Hybrid IO

- Hybrid IO combines buffered and direct IO, getting the best of both
  - Buffered I/O for small I/O, Direct I/O for large I/O
  - Switch made internally, inside Lustre, transparent to applications
  - Unaligned DIO required, using intermediate aligned buffer
- Hybrid IO v1 landed

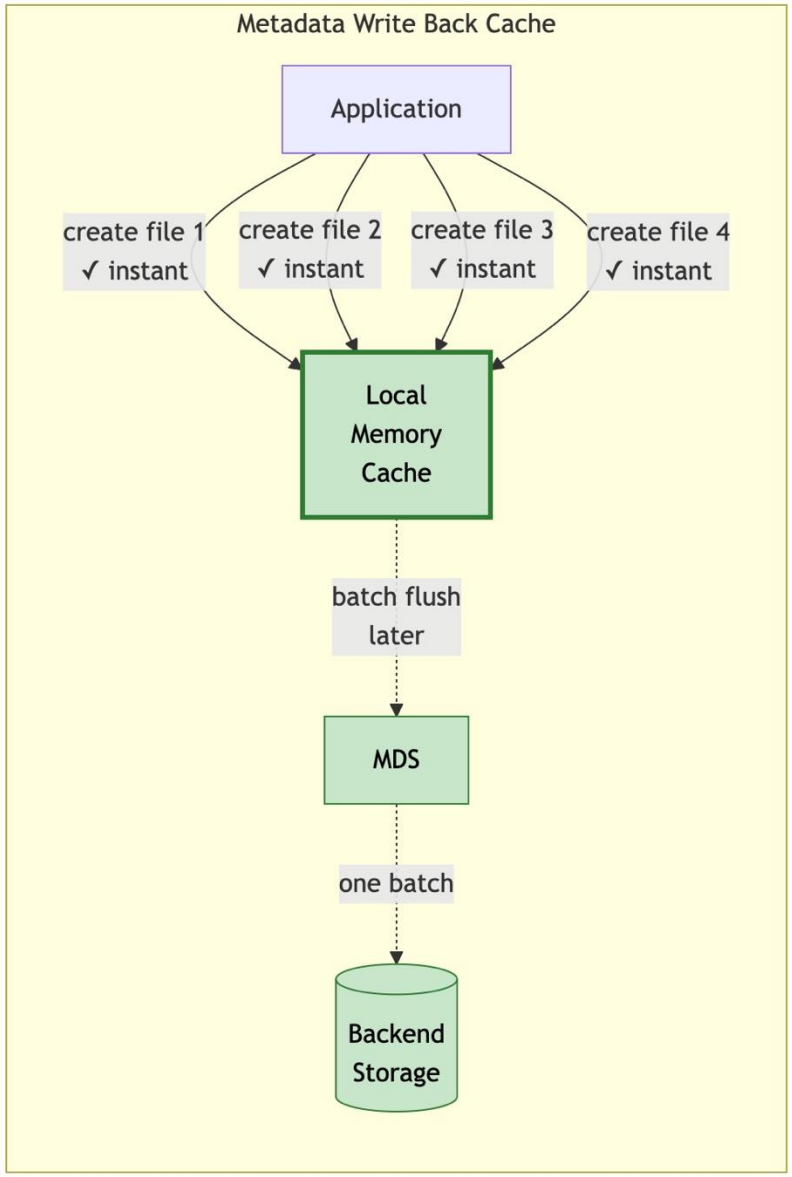


Metadata Writeback Cache

Whamcloud

- **Metadata intensive workloads can hit bottleneck due to:**
  - Namespace synchronization
  - Lock contention on directories
  - Transaction serialization
  - RPC overheads
- **Caching in current Lustre file system:**
  - Data I/O are fully cached on clients in face of no contention under the protection of page-granularity extent locks.
  - For metadata, only read (e.g. dentry, stat()) are cached under the protection of DLM locks.
    - All modifications or even open() need to communicate with MDS.

Metadata Writeback Cache



- 10-100x speedup for single-client create-intensive workloads
  - Gene extraction/scanning, untar/build, data ingest, producer/consumer
- Create new dirs/files **in client RAM without RPCs**
  - Lock new directory exclusively at `mkdir` time
  - Cache new files/dirs/data in RAM until cache flush or remote access
- **No RPC round-trips** for file modifications in new directory
- Batch RPC for efficient directory fetch and cache flush
- **Files globally visible on remote client access**
  - Flush top-level entries, exclusively lock new subdirs, unlock parent
  - Flush rest of tree in background to MDS/OSS by age or size limits

Lustre Metadata Redundancy

Whamcloud

- LMR1a: Fault Tolerant MGS
- LMR1b: Replicate services to other MDTs
- LMR1c: DNE performance (LU-7426)
- LMR2: Replicate top-level dirs for availability
  - 2a: ROOT/ (rarely changed) mirrored to other MDTs
  - 2b: Subdirectories mirrored to 2+ MDTs (via setdirstripe)
  - 2c: Per-file metadata replication in a later stage
  - 2d: e2fsck to allow directory entries with multiple FIDs
- LMR3: Complex recovery handling
- LMR4: LFSCK to repair/rebuild inconsistent mirrors

IPv6

Whamcloud

- Demand for IPv6 in cloud deployments as IPv4 addresses are exhausted
- Also enables other large addresses (e.g. direct IB GUID instead of IPoIB)
- Not only impacts LNet
  - Userspace tools
  - Nodemap
- IPv6 support really stable, with a few limitations
  - Configuration needs attention when mix IPv4/IPv6 is needed
  - Does not handle SLAAC
  - Only ksocklnd (Ethernet) support, not o2ib (Infiniband)
    - Work started, IPoIB with IPv6 addresses

ldiskfs

Whamcloud

- Hybrid ldiskfs LVM storage devices (NVMe+HDD)
  - Allow storing metadata on NVMe at start of device, data on HDDs at end of device
- Enable ldiskfs delayed allocation for writeback cache
  - Allow aggregating small writes in server RAM instead of read-modify-write to client
- Pack the GDT blocks together
  - Do not use meta_bg even for volumes > 256TB

	Original (s)	Patched (s)
mkfs, lazy_itable_init=1	374	22
mount	17	9
e2fsck -fn	427	226

- Parallel e2fsck for pass2/3 (directory entries, name linkage)

Whamcloud

Thank You!
Questions?