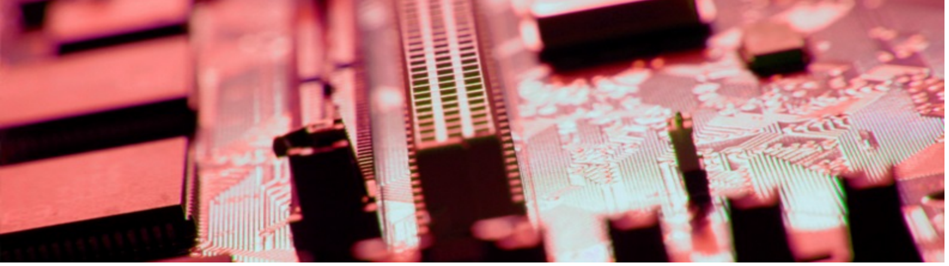


LCI workshop

on high performance clustered computing

Linux Clusters Institute: Lustre

Georgia Tech, August 15th – 18th 2017

J.D. Maloney | Storage Engineer
National Center for Supercomputing Applications (NCSA)
malone12@illinois.edu

Lustre Overview

- Open Source file system supported, developed by many companies and large institutions (Intel, Seagate, DDN, CEA, DOE)
- One of the two "prominent" file systems in used today by the world's largest supercomputers
- Known for its ability to scale sequential I/O performance as the storage system grows
- More complicated to administer, stricter operating environment (OFED stack, kernel, etc.)
- Can grow to greater numbers of clients

History of Lustre

- Long history in the HPC community
- Began in 1999 as a research project by Peter Braam at Carnegie Mellon University
- Braam started a company (Cluster File Systems Inc) and continued development of the file system
  - DOE backed project as part of the Advanced Strategic Computing Initiative
- Sun bought out Cluster File Systems Inc in 2007
  - Braam went to Xyratex (who acquired the hardware during the deal)
- Oracle took control of Lustre releases upon their acquisition of Sun Microsystems

History of Lustre

- Oracle announced it would cease development of the file system
  - Quickly an open source community sprang up to support the file system
- Many Lustre developers left Oracle founded Whamcloud that took a lot of Lustre development
- Whamcloud was acquired by Intel in 2012 and much of the talent in Whamcloud moved to Intel
- In 2017 Intel announced it was leaving the commercial Lustre space, community is continuing to support file system

Stand Out Lustre Features

- Able to handle extremely high client count 10’s to 100’s of thousands of clients can mount the file system
- File System throughput bandwidth scales very well
- Easily supports multiple networks simultaneously via LNET routing
- Open source, no licenses to run the file system
- Strong community behind the file system pushing it forward and contributing code

Stand Out Lustre Features

- Dynamic stripe capability allowing users to define the level of parallelization they're using on the file system (for objects on OSTs)
- Can run on top of software based RAID solutions
  - LDISKFS (ext4) (traditional)
  - ZFS (recent and up and coming)
- No fancy controllers are needed to maintain a reliable file system
- Uses HA/Failover to maintain high-uptime/reliability

Lustre Weaknesses

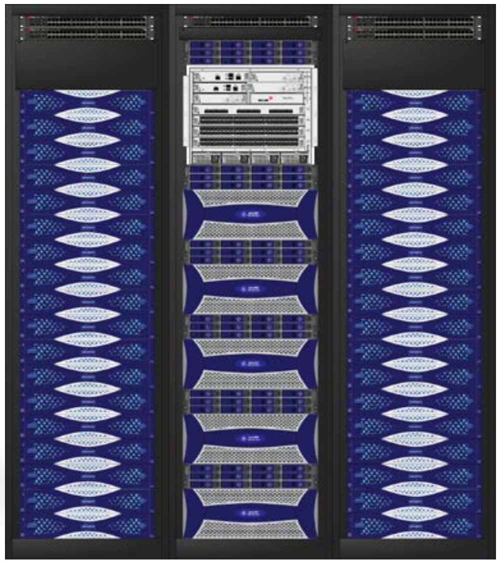
- Relies on Fencing/STONITH to maintain reliable system during node failure scenarios
- No built in policy engine to handle file movement, migration, and policies; relies on external policy engine
- HSM functionality while built in does not support tiers within the file system (something like storage pools in Spectrum Scale), only supports external connectors
- With Intel backing away from commercially supporting Lustre, support will have to come from vendor

Popular Lustre Appliances

- Can buy ready built appliances from many vendors, here are some:

Seagate

DDN


Warp Mechanics

LCI workshop
on high performance clustered computing

Lustre Hardware

- Many other vendors will sell you hardware pre-configured for Lustre file systems
- Find solution that hits your price point and you have confidence in your vendor to provide a solid product
- Can also build yourself on “white box”/commodity hardware
  - Or even in a virtual machine environment on your laptop to test
- Appliance or not depends on how much of a turn key solution you want to have or use

Lustre Concepts

Key Definitions

- MDS (Metadata Server) – Server(s) where information about the files are stored
  - eg. pointers, attributes, extended attributes
- MDT (Metadata Target) – Disk(s) attached to the MDS that provide the storage for the file system metadata
- OSS (Object Storage Server) – Servers that host the actual data itself (the actual files, but nothing about them)
- OST (Object Storage Target) – Disk(s) attached to the OSS that provide the storage for the file system data
- MGS (Management Server) – Stores Lustre file system information

Lustre High Level Architecture

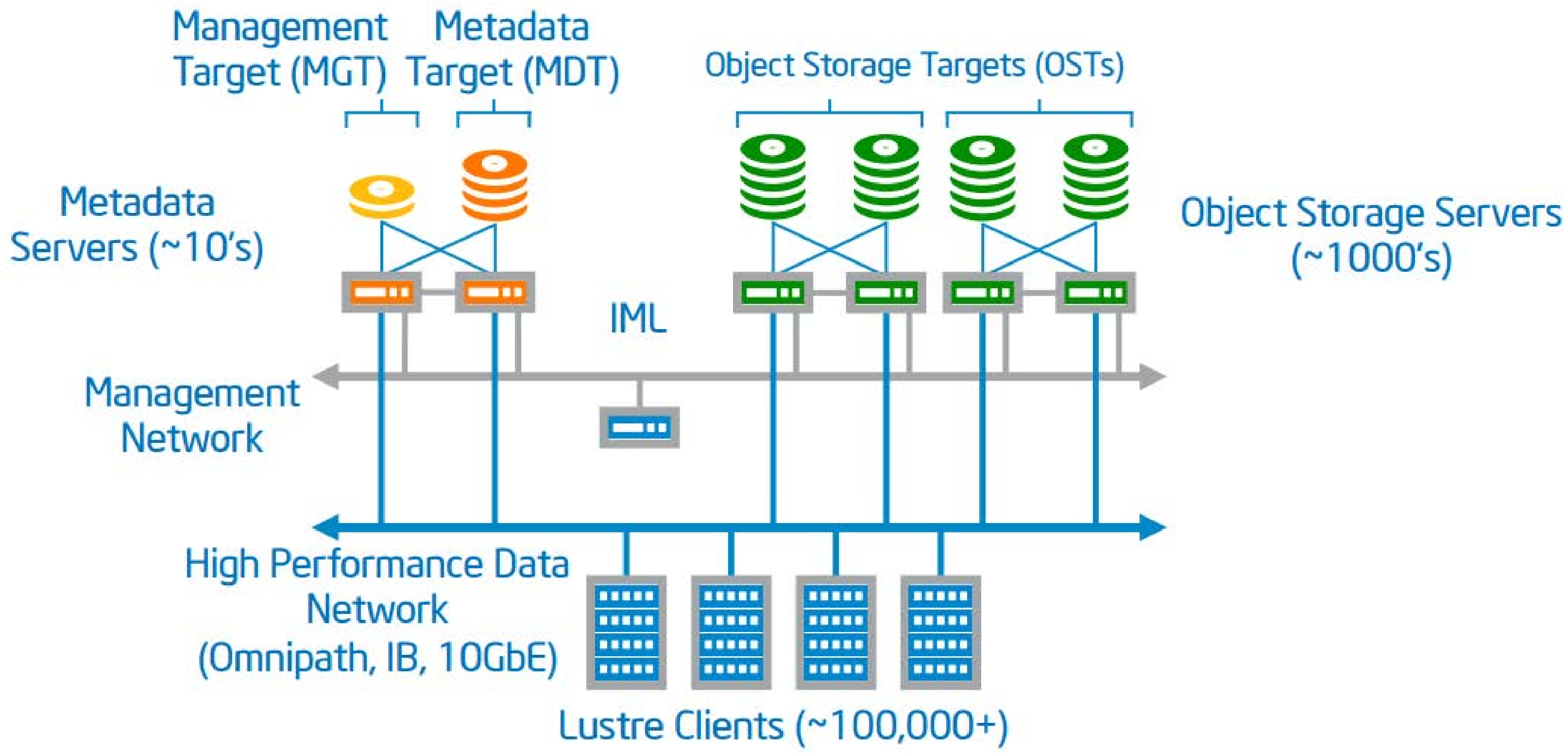
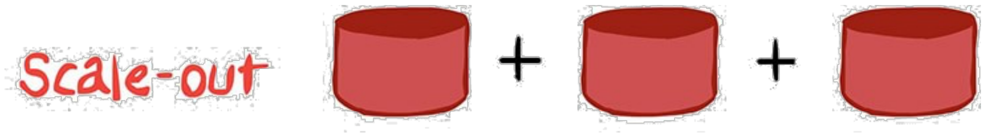

Management Target (MGT)
Metadata Target (MDT)
Object Storage Targets (OSTs)
Metadata Servers (~10's)
Object Storage Servers (~1000's)
IML
Management Network
High Performance Data Network (Omnipath, IB, 10GbE)
Lustre Clients (~100,000+)

Image Credit: nextplatorm.com

Scaling Out

- Both Metadata Servers and Object Storage Servers are increased to improve performance of the file system
- Metadata Scaling is newer to Lustre and has/is only becoming solid in the last couple versions
- Object Server Scaling has been a hallmark of Lustre
  - Why you see systems crossing over the 1TB/s mark on highly sequential benchmarks (think IOR's) and certain workloads (check pointing for example)
- MGS can handle multiple Lustre file systems, a new one is not necessary for every FS created



Examining a File I/O

- Following an I/O request helps us understand how the file system architecture responds to workloads
  - Seeing what the file system has to do for a given task makes it easier to see where bottlenecks form and why
  - Simplest unit of work, less distraction
- Notes for following slide:
  - Layout EA → Layout extended attribute, describes where the objects for a given file needs to land or where to find them (depending on the desired operation, read or write)
  - FID → File Identifier; a very important concept in Lustre, all files have a FID associated with them…this will come up later

Lustre File I/O Process

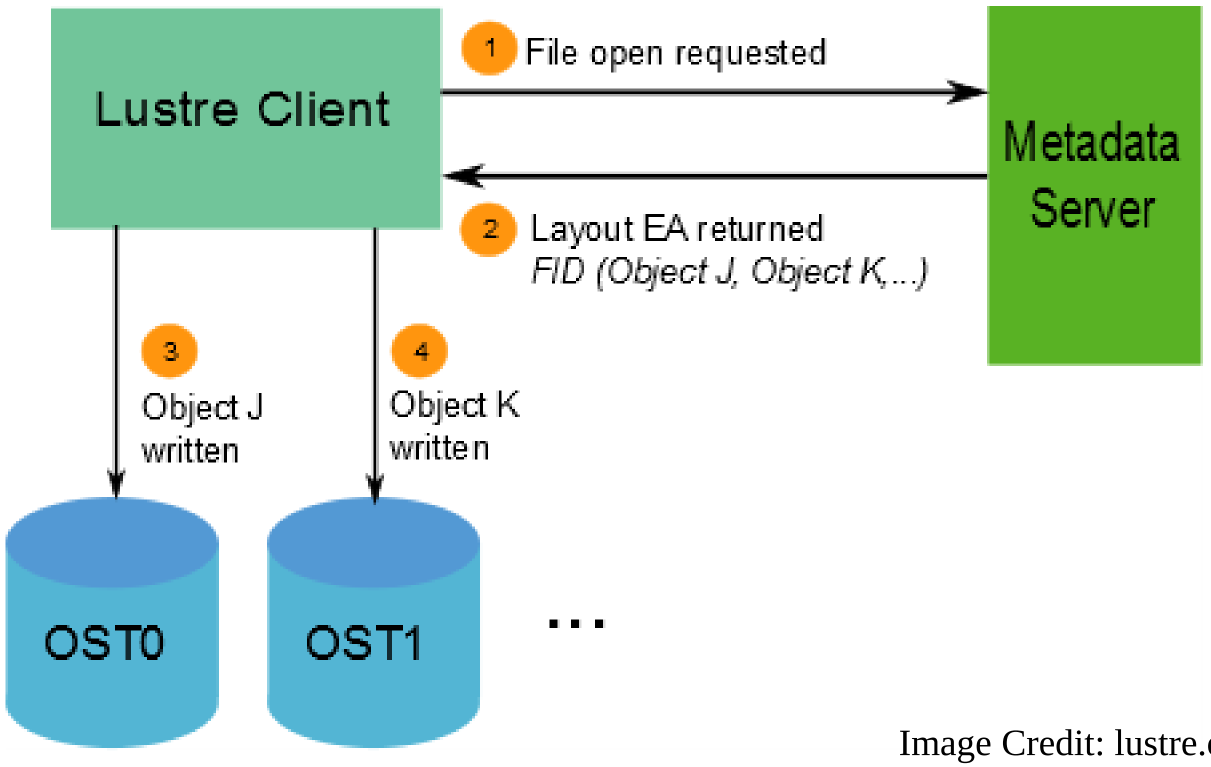


Image Credit: lustre.org

Lustre Metadata Architecture

- Now that we've walked through what a basic file I/O looks like, what can we infer about about our needs for metadata infrastructure?
  - A pretty powerful MDS for sure…or how about many MDS's
  - For clusters with many thousands of clients, that's a lot of beating to do on one MDS
- The MDS is critical to operations occurring on the file system in a timely fashion
- I/O from clients whether Read or Write need to go through the MDS to understand file layout and location
- Admin tools also tap into the MDS for information, increasing the load

Lustre Metadata Architecture

- DNE to the rescue
  - Distributed Namespace Environment
- Phase 1
  - Released in Lustre 2.4
  - Required more manual configuration
  - Each top level dir could be managed by different MDS to spread the load
  - Worked well if top level dirs had good load distribution
    - Not usually the case
- Phase 2
  - Release in Lustre 2.8
  - Each directory striped across multiple MDS servers
  - Easier to deploy, don’t have to worry about new top level dirs
  - Works regardless of load distribution on top level dirs

Lustre DNE1 Architecture

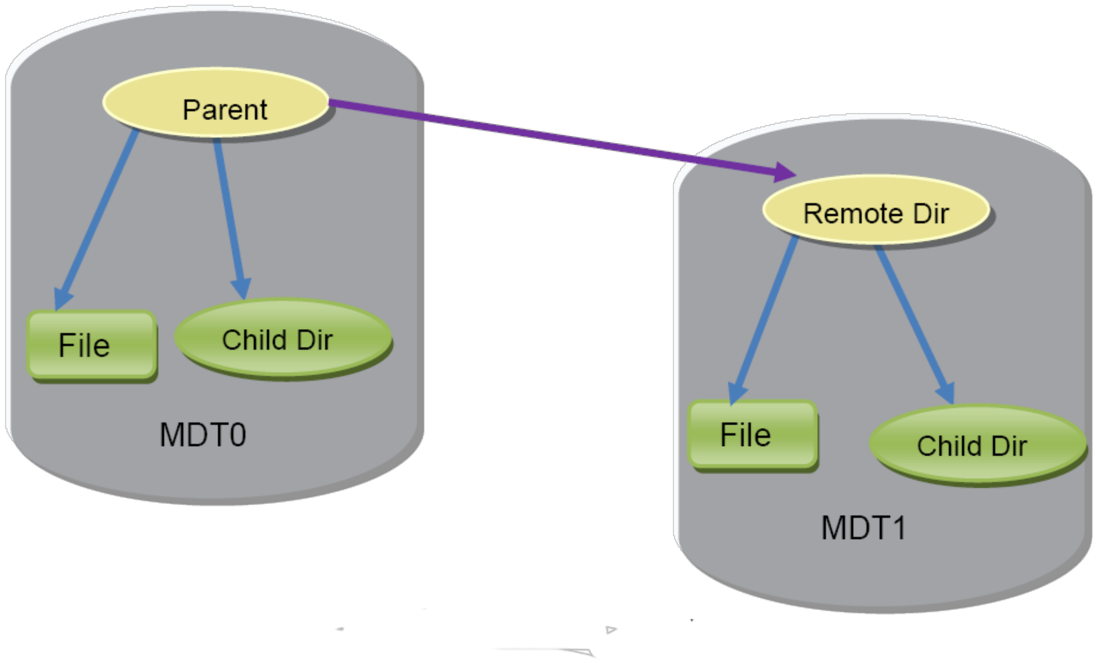


Image Credit: opensfs.org

Lustre DNE2 Architecture

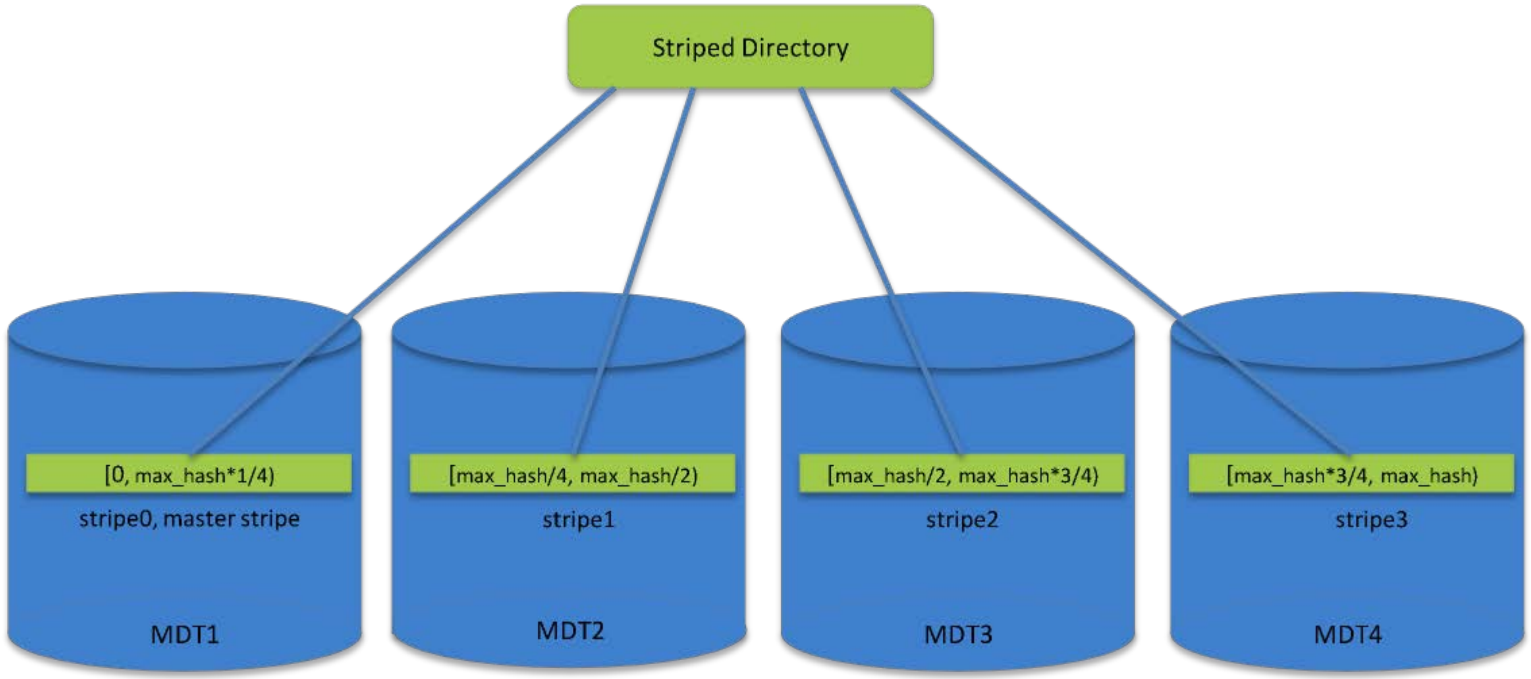


Image Credit: opensfs.org

Lustre Metadata Server

- Powerful Dual Socket CPU System
- Decent amount of memory, helpful to a degree
  - No more than 128GB per MDS
  - Use of ZFS or not has impact on memory configuration
- Fast disks for MDT are essential
  - Great candidate for this is NVME
  - Disks now come in U.2 form factor for easier access
  - RAID10 still best configuration for drives to build MDT
- Good network connectivity
  - Bandwidth does not need to be high as big data doesn't flow through MDS
  - Low latency connection is recommended
  - Usually low latency connections (IB, OPA, come with high bandwidth also)

Lustre Data Architecture

- From the I/O walk through activity what does that tell us about our needs for the Lustre data infrastructure?
  - Lots of high throughput OSS/OST's
  - Where all data actually lives, so we need capacity here and speed
- Actual data served to clients in and out of these machines
- Disks combined using either LDISKFS (ext4) or ZFS depending on the system
  - Meant to protect against drive failure
- Two OSS's can see each OST, HA Failover between those two OSS's allows one to adopt the other's OST's if the other OSS has an issue

Lustre Data Architecture

- Dynamic File Striping
  - Number of OSTs that the file is striped across when it is written out to disk
- The more OSTs that are striped across the higher the amount of sequential throughput there is on that file
- Useful on larger files not on small files
- Stripe width is set by the user for a file, not something that is configured at a system wide level
  - Changing the stripe width of an existing file requires a re-write of that file
- Depending on user usage and how the file system randomly picks OSTs, you need to watch for hot spots where some OSTs get more full than others

LCI workshop
on high performance clustered computing

Lustre Stripe Width

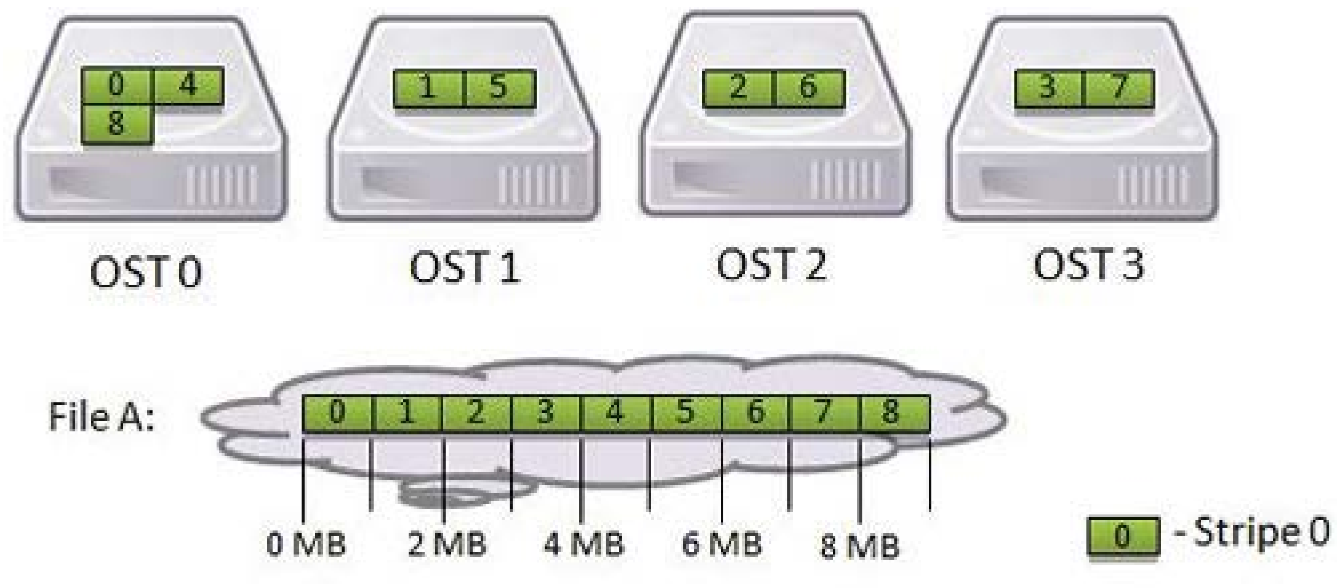


Image Credit: nics.tennessee.edu

Lustre Object Storage Server

- Powerful Dual Socket CPU System
- Decent amount of memory, helpful to a degree
  - No more than 128GB per OSS
  - Use of ZFS or not has impact on memory configuration
- Big disks for OSS are desired
  - Large current rotational drives are best
  - SAS connected HDD for multipath to two OSS hosts
  - Built with RAID6 or Erasure coded groups
- Good network connectivity
  - High bandwidth is good, want for streaming performance
  - Matching network capacity to disk performance if possible
  - High bandwidth fabrics (IB, OPA)

Lustre Quotas

Quotas

- Enabled/Disabled Globally for a filesystem
- Can be set for:
    - users
    - Groups
    - Projects (new in Lustre 2.10)
- Can be set on:
    - Inodes (MDT)
    - Blocks (OSTs)
- Limits
    - Hard (cannot go over)
    - Soft (can go over for the set grace period)

LCI workshop
on high performance clustered computing

Quota Distribution

- Quotas are distributed from the Quota Master Target (QMT)
  - Only one QMT is supported and only runs on the MDS
  - MDS maintains the Inode quotas
  - MDS and OSS track Block quotas
- All OSTs and MDTs setup a Quota Slave Device (QSD)
  - QSD connects to QMT to allocate/release quota space

Quota Allocation

- OSS maintains quota sub-allocations per user/group for local OSTs
  - As needed, OSS requests new allocation per OST before writing
- Quota space is initially allocated in very large chunks from QMT
  - Large requests when lots of quota is available to minimize quota requests
  - Minimum chunk size (qunit) for inodes an blocks
- Chunk requests will succeed until over the limit
  - Writes to OSTs with chunks available will still succeed
  - Can result in exceeding hard quota

Quota Grace

- Users/Groups enter “grace period” when soft limit is exceeded
  - Soft limit can be exceeded for the duration of the grace period
  - When grace period is exceeded, soft limit becomes the hard limit
  - Grace periods can only be modified globally
- Default: 1 week
- The difference between soft and hard limit should be limited to help keep overruns reasonable

Exceeding Quota

- Exceeding hard quota is possible
  - ”Granted cache” allows fast writes to succeed from clients
  - If the Lustre client has enough granted cache, it returns ‘success’ to users and arranges the writes to the OSTs.
  - Because the Lustre clients have already delivered a success to users, the OSTs cannot fail these writes.
- This overage can be mitigated, but not completely
  - Reducing the maximum amount of dirty data on the clients will limit this effect (min value is 1MB).

```
# lctl set_param osc.*.max_dirty_mb=8
```

Quota Configuration

- From Lustre versions 2.4 on, quota (accounting) is enabled by default
- Enabling/Disabling quota enforcement:
  - turn on user, group quotas for block only on filesystem lustre1, run on MGS

```
# lctl conf_param lustre1.quota.ost=ug
```

  - turn on group quota for inodes on filesystem lustre2, run on MGS

```
# lctl conf_param lustre2.quota.mdt=g
```

  - turn off all quotas for filesystem lustre3, run on MGS

```
# lctl conf_param lustre3.quota.ost=none
# lctl conf_param lustre3.quota.mdt=none
```

Quota Configuration

- Verify per target enforcement status

```
$ lctl get_param osd*.*.quota_slave.info
osd1.lustre-MDT0000.quota_slave.info=
target name: testfs-MDT0000
pool ID: 0 type: md
quota enabled: ug
conn to master: setup
user uptodate: glb[1],slv[1],reint[0]
group uptodate: glb[1],slv[1],reint[0]
```

Quota Configuration

- Quota limits are set using the command:

```
 lfs setquota
```

```
# lfs setquota –u user1 –b 307200 –B 309200 –i 10000 –I 11000
/mnt/lustre
```

```
# lfs setquota –g group1 –b 5120000 –B 5150000 –i 100000 –I
101000 /mnt/lustre
```

Quota Reporting

- The quota command displays the quota allocated and consumed by each Lustre target

```
$ lfs quota –u user1 -v /mnt/lustre
Disk quotas for user user1 (uid 6000):
Filesystem kbytes quota limit grace files quota limit grace
/mnt/lustre 0 30720 30920 - 0 10000 11000 –
lustre-MDT0000_UUID 0 - 8192 - 0 - 2560 –
lustre-OST0000_UUID 0 - 8192 - 0 - 0 –
lustre-OST0001_UUID 0 - 8192 - 0 - 0 –
Total allocated inode limit: 2560, total allocated block limit:
24576
```

Changelogs

LCI workshop
on high performance clustered computing

Changelogs

- Changelogs record events that change the file system namespace or file metadata.
- The change type, target and parent file identifiers (FIDs), the name of the target, and a timestamp are recorded.
- These can be used in a variety of ways:
    - Capture recent changes to feed into and archive system
    - Replicate changes in a filesystem mirror
    - Use the events to trigger actions based on specific criteria
    - Maintain a rough audit trail (no user information)

Changelog Record Types

- MARK – Internal record keeping
- CREAT – Regular file creation
- MKDIR – Directory creation
- HLINK – Hard link
- SLINK – Soft link
- OPEN – open file
- CLOSE – close file
- MKNOD – Other file creation
- UNLNK – Regular file removal
- RMDIR – Directory removal
- RNMFM – Rename, original
- RNMTO – Rename, final
- IOCTL – ioctl on file or directory
- TRUNC – Regular file truncated
- SATTR – Attribute change
- XATTR – Extended Attribute change
- HSM – HSM action
- UNKNW – Unkown operation

Changelog Usage

- Changelog records use space on the MDT
  - Sysadmin must register changelog users
  - Registered users specify which records they are ”done with” and the system purges up to that point
  - Changelog entries are not purged beyond a registered user’s set point.
- Register new changelog user

```
# lctl –-device lustre-MDT0000 changelog_register
lustre-MDT0000: Registered changelog userid 'cl1'
```

Changelog Usage

- To actually start the changelogs, the changelog_mask must be set
  - Changelog_mask specifies what record types are recorded

```
# lctl set_param mdd.lustre-MDT0000.changelog_mask=MARK CREAT
MKDIR HLINK SLINK MKNOD UNLNK RMDIR RNMFM RNMTO OPEN CLOSE
IOCTL TRUNC SATTR XATTR HSM

mdd.lustre-MDT0000.changelog_mask=

MARK CREAT MKDIR HLINK SLINK MKNOD UNLNK RMDIR RNMFM RNMTO OPEN
CLOSE IOCTL TRUNC SATTR XATTR HSM
```

- Note: This example sets almost all of the record types.In practice, on a busy system, some of these may need to be trimmed.

Changelog Usage

- Display changelogs

```
# lfs changelog fsname-MDTnumber [startrec [endrec]]
```

- Start and end records are optional
- Example changelog records:

```
$ lfs changelog lustre-MDT0000
1 00MARK 19:08:20.890432813 2010.03.24 0x0 t=[0x10001:0x0:0x0]
p=[0:0x0:0x0] mdd_obd-lustre-MDT0000-0
2 02MKDIR 19:10:21.509659173 2010.03.24 0x0
t=[0x200000420:0x3:0x0] p=[0x61b4:0xca2c7dde:0x0] mydir
3 14SATTR 19:10:27.329356533 2010.03.24 0x0
t=[0x200000420:0x3:0x0]
4 01CREAT 19:10:37.113847713 2010.03.24 0x0
t=[0x200000420:0x4:0x0] p=[0x20\ 0000420:0x3:0x0] hosts
```

Changelog Usage

- Use the changelog_clear command to clear old changelog records for a specific user.
  - This will most likely allow the MDT to free up disk space

```
# lctl changelog_clear mdt_name userid endrec
```

- Notify a device that user cl1 no longer needs records (up to and including 3):

```
$ lfs changelog_clear lustre-MDT0000 cl1 3
```

- Confirm operation was successful

```
$ lfs changelog lustre-MDT0000
```

```
4 01CREAT 19:10:37.113847713 2010.03.24 0x0
t=[0x200000420:0x4:0x0] p=[0x200000420:0x3:0x0] hosts
```

Changelog Usage

- To stop changelogs, changelog_mask should be set to MARK only

```
# lctl set_param mdd.lustre-MDT0000.changelog_mask=MARK
mdd.lustre-MDT0000.changelog_mask=MARK
```

- If disabling changelogs permanently, or for an extended period, clear any remaining backlog before deregistering changelog user using changelog_clear.

```
# lctl get_param mdd.lustre-MDT0000.changelog_users
mdd.lustre-MDT0000.changelog_users=
current index: 81680
ID      index
cl1     81500
```

- Backlog = [current_index] – [cl1 index]

```
81680 - 81500 = 180
```

Changelog Usage

- Clear remaining changelogs

```
$ lctl changelog_clear lustre-MDT0000 cl1 81680
```

- Confirm cleared changelog records

```
# lctl get_param mdd.lustre-MDT0000.changelog_users
mdd.lustre-MDT0000.changelog_users=
current index: 81680
ID     index
cl1    81600
```

- Deregister changelog user

```
# lctl --device lustre-MDT0000 changelog_deregister cl1
lustre-MDT0000: Deregistered changelog user 'cl1'
```

Consuming Changelogs

- The most common use of changelogs is in conjunction with Robinhood, an external filesystem reporting/policy engine.

- Robinhood uses a combination of scanning the filesystem and changelog consumption to create a replica/shadow of the filesystem metadata in an external database.

Robinhood

Robinhood

- External filesystem reporting/policy engine developed by CEA
- Runs on separate server from Lustre file system servers
- Utilizes a database backend (MySQL,MariaDB) to create a replica/shadow of the filesystem metadata
- Customizable policies with many triggering options
- Filesystem usage reports
- Tools: rbh-find, rbh-du

Robinhood Database

- Should have a dedicated Database server/disk
  - Highest available/affordable CPU frequency
  - As much memory as possible (cache as much DB as possible in memory)
  - Low Latency network
  - Fast disk for DB
  - DB server can mount Lustre and run robinhood locally, or the DB server can be completely dedicated and Robinhood is run on a separate server.
- Disk space ~1KB per entry (e.g. 100GB for 100 million entries)
  - SSD or NVME if possible

Robinhood Installation

- Tarball/RPMs from sourceforge:
  - https://sourceforge.net/projects/robinhood/
- Pull latest source from github and build rpms or build/install:
  - https://github.com/cea-hpc/robinhood
  - https://github.com/cea-hpc/robinhood/wiki/robinhood_v3_admin_doc#installation

Robinhood Configuration

- Robinhood provides a basic and extended examples in

`/usr/share/robinhood/templates`

- Basic configuration
    - Copy basic example

```
# cp /usr/share/robinhood/templates/basic.conf\
/etc/robinhood.d/newrbh.conf
```

- Edit DB details

```
ListManager {
    MySQL {
        server = your_db_server;
        db = your_db_name;
        user = robinhood;
        password_file = /etc/robinhood.d/.dbpassword;
    }
}
```

Robinhood Configuration

- Specify filesystem is Lustre and the mount point

```
General {
    fs_path = "/path/to/fs";
    # filesystem type, as displayed by 'mount' (e.g. ext4, xfs,
lustre, ...)
    fs_type = lustre;
}
```

- Also be sure to create the .dbpassword file with the database user password

```
password_file = /etc/robinhood.d/.dbpassword;
```

Configure Robinhood DB

- Setup and start MySQL
- Connect to MySQL as root and create Robinhood DB and Robinhood User

```
# mysql -p
mysql> create database rbh_lustre;
mysql> GRANT all on rbh_lustre.* to ‘rbh_user’@’lustre_client
host’ IDENTIFIED BY ‘rbh_user_password’;
mysql> FLUSH PRIVILEGES;
```

Feeding Robinhood

- Initial Scan
  - An initial full scan should be performed to fill the DB. While this is running it is also best to enable changelogs and start an instance of Robinhood doing this as well.
  - When you start Robinhood against an empty database it will create the appropriate schema before it starts doing it’s work

```
# robinhood --scan –-once –f newrbh –D
```

- Changelogs

```
# robinhood -r -f newrbh –D
```

Robinhood Policy

- Robinhood's policies can be used to perform many filesystem action either on a schedule or triggered by specific criteria. Some common policy definitions are included with the Robinhood install `/etc/robinhood.d/includes/`
- There are 4 elements that make up a Robinhood Policy
  - **Declaration**: specifies the name of the policy, status manager to manage status of entries, scope (static base set of entries to apply policy to), default action/sort (other basic policy actions).
  - **Parameters**: specific parameters to default actions, number of actions per run or rate limits, scheduling configuration to allow reordering or delayed actions

Robinhood Policy

- There are 4 elements that make up a Robinhood Policy (cont.)
  - **Rules**: excludes, conditionals for file classes, override default settings
  - **Triggers**: Conditions to start policy runs (set intervals, usage thresholds, etc.), optional set of resources to check (users, groups, OSTs), number of actions per run or rate limits.

Robinhood Policy Examples

- Cleanup/Purge Policy
- Add the tmpfs.inc include to the robinhood config

```
%include "includes/tmpfs.inc”
```

- Here is the content of tmpfs.inc:

```
# used to be rbh 2.5 "purge" policy in TMPFS mode
define_policy cleanup {
    scope { type != directory }
    status_manager = none;
    default_action = common.unlink;
    default_lru_sort_attr = last_access;
}
```

Robinhood Policy Examples

- File classes can/should be used to narrow the scope of entries to be processed

```
fileclass scratch_files {
      definition { type == file and tree == “/path/to/user/files” }
}
```

- Specify policy rules – clean(purge) files from the scratch_files fileclass that have access times older than 30 days

```
cleanup_rules {
     ignore { last_access < 7d }
     rule clean_scratch_files {
          target_fileclass = scratch_files;
          condition { last_access > 30d }
     }
     # default rule (optional): # apply to all other entries
     rule default { condition { last_access > 60d } }
}
```

Robinhood Policy Examples

- Define triggers

```
cleanup_trigger {
     trigger_on = global_usage;
     high_threshold_pct = 80%;
     low_threshold_pct = 75%;
     check_interval = 15min;
}
```

- The global_usage is based on the whole file system. This could also be triggered on “ost_usage” (if an OST hits a threshold), or user/group thresholds.

Robinhood Policy Examples

- Run the policy
  - By hand:

```
# robinhood --run=cleanup –d
```

  - Target only a specific user

```
# robinhood --run=cleanup --target=user:foo
```

  - Delete 1000 files from ost23

```
# robinhood --run=cleanup --target=ost:23 --max-count=1000
```

Robinhood Policy Examples

- Run the policy
- OR use the Robinhood service to run policies
  - Edit `/etc/sysconfig/robinhood` (or `/etc/sysconfig/robinhood.”fs` name” on RHEL7)
  - Append `--run=cleanup` to RBH_OPT (if the --run option is already present, add it to the list of run arguments, e.g. `--run=policy1,cleanup`)
  - Start (or restart) robinhood service:

RHEL6: `service robinhood restart`

RHEL7: `systemctl restart robinhood[@''fsname'']`

Robinhood Reports

```
# filesystem entries:
# rbh-report --fs-info
type , count, volume, avg_size
dir, 1780074, 8.02 GB, 4.72 KB
file, 21366275, 91.15 TB, 4.47 MB
symlink, 496142, 24.92 MB, 53

# user info, split by group
# rbh-report -u bar -S
user , group, type, count, spc_used, avg_size
bar , proj1, file, 4, 40.00 MB, 10.00 MB
bar , proj2, file, 3296, 947.80 MB, 273.30 KB
bar , proj3, file, 259781, 781.21 GB, 3.08 MB
```

Robinhood Reports

```
# file size profile for a given user
# rbh-report -u foo --szprof
user, type, count, volume, avg_size, 0, 1~31, 32~1K-, 1K~31K,
32K~1M-, 1M~31M, 32M~1G-, 1G~31G, 32G~1T-, +1T
foo , dir, 48, 1.48 MB, 31.67 KB, 0, 0, 0, 26, 22, 0, 0, 0, 0, 0
foo , file, 11055, 308.16 GB, 28.54 MB, 2, 0, 14, 23, 5276, 5712,
9, 17, 2, 0
# top disk space consumers
# rbh-report --top-users
rank, user , spc_used, count, avg_size
1, usr0021 , 11.14 TB, 116396, 100.34 MB
2, usr3562 , 5.54 TB, 575, 9.86 GB
3, usr2189 , 5.52 TB, 9888, 585.50 MB
4, usr2672 , 3.21 TB, 238016, 14.49 MB
5, usr7267 , 2.09 TB, 8230, 266.17 MB
```

Robinhood Tools

- rbh-find: Clone of the “find” command that searches the Robinhood DB instead of the filesystem to find objects matching the passed criteria

```
# rbh-find /mnt/lustre/dir -u root -size +32M -mtime +1h -ost 2
–ls
```

- rbh-du: Clone of the “du” command that uses the Robinhood DB instead of the filesystem to calculate disk usage

```
# rbh-du -H -u foo /mnt/lustre/dir.3 45.0G /mnt/lustre/dir.3
```

LNet Routing

LNet Routing Overview

- Configuring LNet is optional
  - LNet will use the first TCP/IP interface it discovers on a system if it is loaded using the `lctl network up` command.

- LNet configuration is required if:
  - Using Infiniband
  - Using multiple Ethernet interfaces

LNet Routing Overview

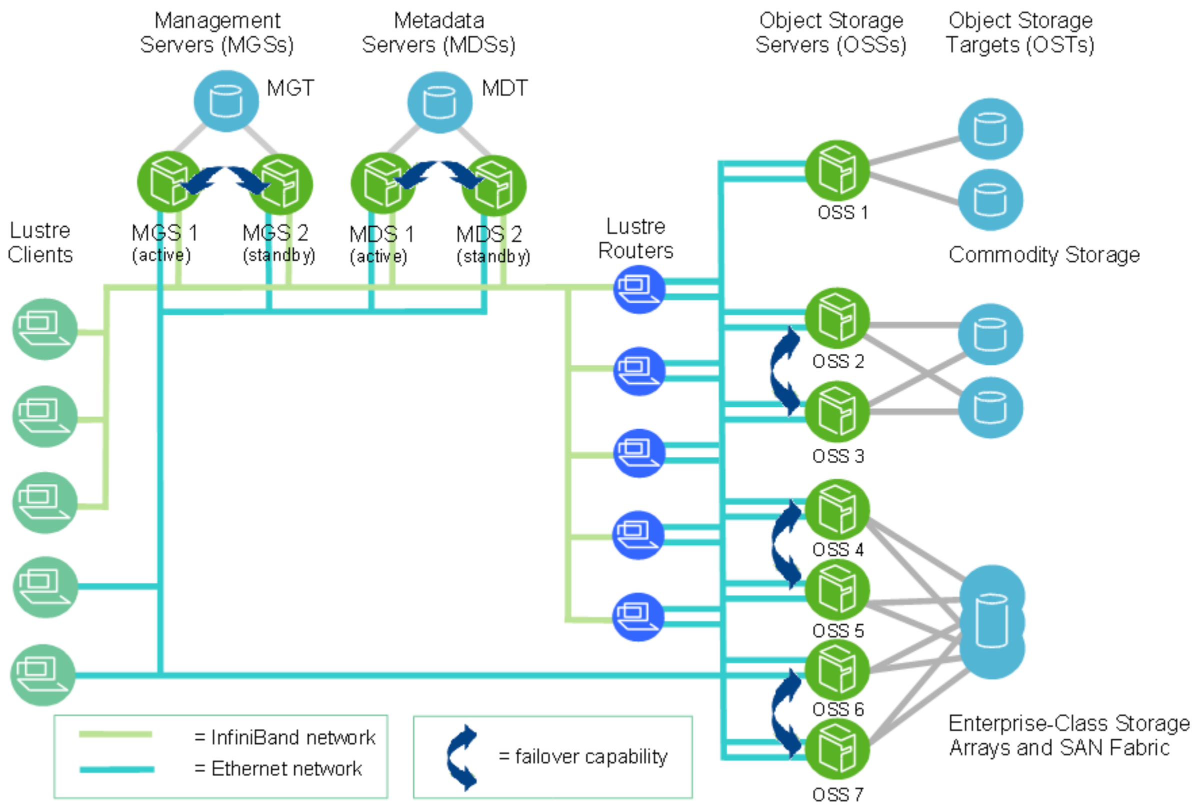


Image Credit: lustre.org

LNet Routing Overview

- Routes are typically set to connect to segregated subnetworks or to cross connect two different types of networks such as tcp and o2ib
- The LNet `routes` parameter specifies a colon-separated list of router definitions. Each route is defined as a network number, followed by a list of routers.
- Bidirectional routing example:

```
options lnet 'ip2nets="tcp0 10.1.0.*; \
   o2ib0(ib0) 10.11.0.[1-128]"' 'routes="tcp0 10.11.0.[1-8]@o2ib0; \
   o2ib0 10.1.0.[1-8]@tcp0"'
```

Configuring LNet

- LNet kernel module (lnet) parameters specify how LNet is to be configured to work with Lustre
  - `networks` - Specifies the networks to be used.

```
options lnet networks=tcp0(eth0),o2ib(ib0)
```

  - `ip2nets` - Lists globally-available networks, each with a range of IP addresses. LNet then identifies locally-available networks through address list-matching lookup.

```
options lnet 'ip2nets="tcp0(eth0) 10.1.0.[2,4]; tcp0
10.1.0.*; o2ib0 10.11.1.*"'
```

  - Use one or the other. Ip2nets is nice to allow one lnet.conf across a cluster.

LNet Configuration

- In Lustre 2.7 DLC – Dynamic LNet Configuration
- `lnetctl` utility can be used to configure LNet dynamically.
    - Add/Remove Networks
    - Add/Remove Routes
    - Enable/Disable Routing
    - Not everything is dynamically configurable

LNet Routing Sends/Receives

Receives

- Additional credit accounting when routers receive a message destined for another peer
- These credits account for resources used on the router node
- Peer Router Credit
  - Governs the number of concurrent receives from a single peer
- Router Buffer Credit
  - Router has limited number of three different sized buffers: tiny, small, large
  - Router buffer credits ensure we only receive if an appropriate buffer is available

Sends

- Every lnet_send() uses a peer credit and a network interface credit
  - except loopback interface NI: 0@lo

LNet Credits

- Max_rpcs_in_flight and max_pages_per_rpc, addressed later… effect router credits
- 4MB I/Os associate additional LNet memory descriptors with a bulk operation. LNet MTU is still 1MB
- The total number of messages (and thus credits) required to complete bulk reads and writes is lower for transfers > 1MB

LNet Credits

- lctl set_param osc.*.max_pages_per_rpc=1024

	256 pages/RPC	1024 pages/RPC
1MB Write	1 RPC, 2 Credits	1 RPC, 2 Credits
2MB Write	2 RPC, 4 Credits	1 RPC, 3 Credits
3MB Write	3 RPC, 6 Credits	1 RPC, 4 Credits
4MB Write	4 RPC, 8 Credits	1 RPC, 5 Credits

# of bulk write RPCs sent and peer credits taken for bulk transfer by a single client for different sized writes (read case is the same)

Lustre Client Tuning

Lustre Client Tuning

- Max_rpcs_in_flight
  - Max RPCS in flight between OSC and OST
  - 1-256 Default: 8

```
/proc/fs/lustre/osc/<OST name>/max_rpcs_in_flight
lctl set_param osc.*.max_rpcs_in_flight=256
```

- Max_pages_per_rpc
  - Max number of 4K pages per RPC
  - 256 = 1MB per RPC

```
/proc/fs/lustre/osc/<OST name>/max_pages_per_rpc
lctl get_param osc.*.max_pages_per_rpc
```

Lustre Client Tuning

- Max_dirty_mb
  - Maximum MBs of dirty data that can be written and queued on a client
  - Set per OST
  - Default: 32MB, MAX: 1024MB

```
/proc/fs/lustre/osc/<OST name>/max_dirty_mb
lctl get_param osc.*.max_dirty_mb
```

Resources

- http://lustre.org/getting-started-with-lustre/
- http://lustre.ornl.gov/lustre101-courses/
- http://opensfs.org/lug-2017/
- https://www.eofs.eu

LCI workshop on high performance clustered computing

Acknowledgements

- Members of the SET group at NCSA for slide creation and review
- Members of the steering committee for slide review

Questions