**Author:** Aaron Valoroso

**Date:** December 18th, 2017

**What is Lustre and how does it work?**

**Overview:** The Lustre file system is a type of parallel distributed file system that uses more than one server to create a storage solution for some of the fastest supercomputers in the world. Lustre has the ability to be scaled up or down with respect to the number of client nodes, disk storage and bandwidth, this can also be applied to the availability to each resource. The software can be tuned for any type of work environment whether it be fast or slow I/O speeds.

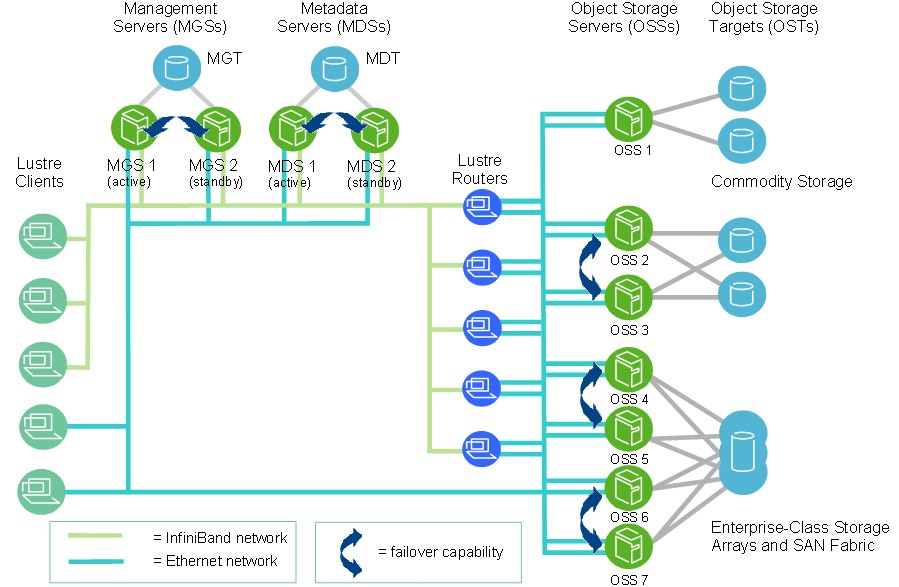
Here is a list of important features that Luster supports as well:

* **Security:** By default, TCP connections are only allowed from privileged ports. UNIX group membership is verified on the MDS.
* **Quotas:** User and group quotas are available for a Lustre file system.
* **Capacity growth:** The size of a Lustre file system and aggregate cluster bandwidth can be increased without interruption by adding new OSTs and MDTs to the Lustre cluster.
* **Controlled file layout:** The layout of files across OSTs can be configured on a per file, per directory, or per file system basis. This allows file I/O to be tuned to specific application requirements within a single file system. The Lustre file system uses RAID-0 striping and balances space usage across OSTs.
* **Network data integrity protection:** A checksum of all data sent from the client to the OSS protects against corruption during data transfer.
* **Disaster recovery tool:** The Lustre file system provides an online distributed file system check (LFSCK) that can restore consistency between storage components in case of a major file system error. A Lustre file system can operate even in the presence of file system inconsistencies, and LFSCK can run while the file system is in use, so LFSCK is not required to complete before returning the file system to production.
* **Performance monitoring:** The Lustre file system offers a variety of mechanisms to examine performance and tuning.

**Components:** An installation of Luster will include a management server (MGS), and one or more metadata server and object storage server. All these pieces are interconnect with Lustre networking (LNet).

* **Management Server (MGS):** This server stores configuration information for all of the Luster cluster components. The MGS will also communicate back and forth between each Luster component giving and receiving information.
* **Metadata Server (MDS):** This server creates the metadata that will be stored on the metadata targets. Each MDS manages the names and directories in the Lustre file system and provides network request handling for one or more local metadata target.
* **Object Storage Server (OSS):** This server provides file I/O service and network request handling for one or more local object storage targets. Typically, an object storage server serves between two or eight object storage targets, up to 16 TB each.
* **Metadata Target (MDT):** This is not a server but a hardware storage that stores metadata such as filenames, directories, permissions, and file layout. The metadata target connects to the metadata server.
* **Object Storage Target (OST):** This is not a server but a hardware storage that stores user file data.
* **Lustre Clients:**  Lustre clients are computational, visualization or desktop nodes that are running Lustre client software, allowing them to mount the Lustre file system.

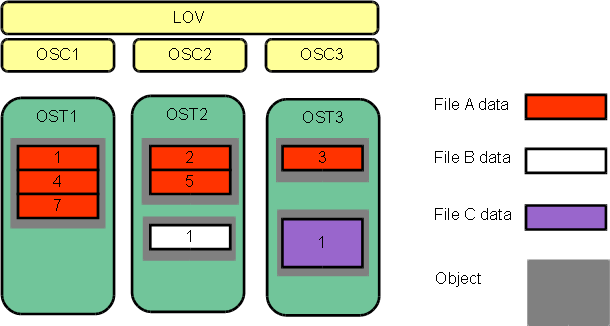
*The picture below shows how these components can be connected together, and use failover and different internet options. Also, how the Luster cluster can be at scale.*



**Striping:** One of the main factors leading to the high performance of the Lustre file system is the ability to stripe data across multiple OSTs in a round-robin fashion. The round-robin fashion is an algorithm employed by process and network schedulers that have no priority in the queue. Users can optionally configure for each file the number of stripes, stripe size, and OSTs that are used. The ability to stripe is also useful when a single OST does not have enough free space to hold an entire file.

Striping allows segments or ‘chunks’ of data in a file be stored on different OSTs. The number of objects in a single file is called the stripe\_count. Each object contains a chunk of data from the file. When the chunk of data being written to a particular object exceeds the stripe\_size, the next chunk of data in the file is stored on the next object.

*The picture below shows how striping works on the Lustre cluster.*

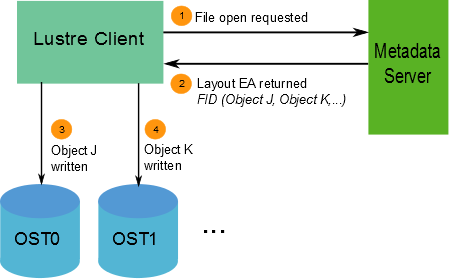
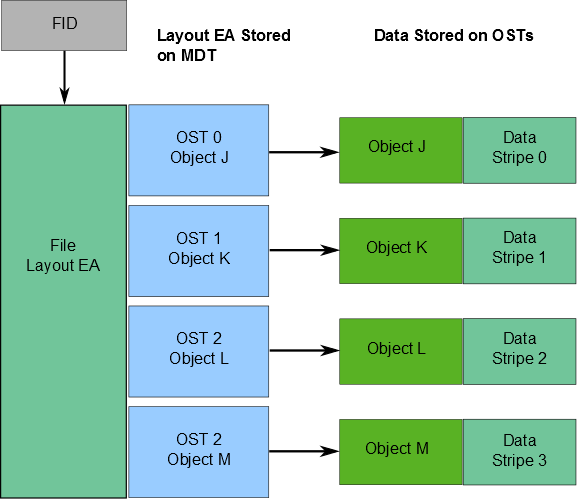


**Storage and I/O:** Luster uses file identifiers in its file system in order to keep track of all the files in the system. A Lustre file identifier (FID) is a 128-bit identifier that contains a unique 64-bit sequence number, a 32-bit object ID (OID), and a 32-bit version number. This sequence number is unique across all Lustre targets in the cluster.

Information about where file data is located on the OST(s) is stored as an extended attribute called “layout EA” in an MDT object identified by the FID for the file. An extended attribute are file system features that enable systems to associate computer files with metadata. If the file is a regular file, the MDT object points to 1 – to – N OST objects(s) that contain the file data. IF the MDT file points to one object, all the file data is stored in that object. If the MDT file points to more than one object, the file data is striped across the objects using RAID 0, and each object is stored on a different OST.

When a client wants to read from or wirte to a file, it first fetches the “File Layout EA” from the MDT object for the file. The client then uses this information to perform I/O on the file, directly interacting with the OSS nodes where the objects are stored. The MDS takes what file that the user wants to interact with and see what the object is point too. It then gathers the file data if it is striped and communicates with the OSS that grab the file data on the OST. This is all then returned back to the user.

*The picture below shows how accessing files works on a systematic level.*



Something to think about: You have to reformat the hard drive to have a smaller block size and then the user has to set the stripe size for the given hard drive?

**LNet:** A Lustre network is comprised of clients and servers running the Lustre software. The Lustre network is not confined to one LNet subnet but can span several networks provided routing is possible between the networks. In a similar manner, a single network can have multiple LNet subnets.

The Lustre network supports many commonly-used network types, such as InfiniBand and IP networks, and allows simultaneous availability across multiple network types with routing between them.

Here is a list of supported network types for LNet:

* + InfiniBand: OpenFabrics (OFFD) (o2ib)
  + TCP (Any network carrying TCP traffic, including GigE, 10GigE, and IPoIB)
  + RapidArray: ra
  + Quadrics: Elan

Remote direct memory access (RDMA) is permitted when supported by underlying networks using the appropriate Lustre network driver (LND). An LND is a pluggable driver that provides support for a particular network type. High availability and recovery features enable transparent recovery in conjunction with failover servers.

Here is a list of features for LNET:

* + RDMA, when supported by underlying networks.
  + Support for many commonly-used network types.
  + High availability and recovery.
  + Support of multiple network types simultaneously.
  + Routing among disparate networks.

LNet permits end-to-end read / write throughput at or near peak bandwidth rates on a variety of network interconnects.

**LNet Implementation:** The Lustre networking stack is comprised of two layers, the LNet code module and the LND. The LNet layer operates above the LND layer in a manner similar to the way the network layer operates above the data link layer. LNet layer is connectionless, asynchronous and does not verify that data has been transmitted like UDP, while the LND layer is connection oriented and typically does verify data transmission like TCP.

LNets are uniquely identified by a label comprised of a string corresponding to an LND and a number, such as tcp0, o2ib0, or o2ib1, that uniquely identifies each LNet. Each node on an LNet has at least one network identifier (NID). A NID is a combination of the address of the network interface and the LNET label in the form: address@LNet\_label.

Example:

[192.168.1.5@tcp0](mailto:192.168.1.5@tcp0)

[10.13.24.90@o2ib1](mailto:10.13.24.90@o2ib1)

In certain circumstances it might be desirable for Lustre file system traffic to pass between multiple LNets. This is possible using LNet routing. It is important to realize that LNet routing is not the same as network routing. Also configuring LNet is optional, and will use the first TCP/IP interface it discovers on a system (eth0) if it’s loaded using the lctl network up. If this network configuration is sufficient you do not need to configure LNet. LNet configuration is required if you are using InfiniBand or multiple Ethernet types. For more information look at chapter 9 in the Lustre user manual.

**What should be thought about before installing Lustre?**

**Pre-Thought:**

Some questions that should be asked before installing the Lustre software in your environment:

* + How big is the file system going to be?
  + How many clients will be connected to the system?
  + What kind of hardware will be available to use?

Another thing to keep in mind is scalability and future proofing. When determining the specific hardware requirements it’s a good thing to double or even triple the amount of resources that you would need for your Lustre cluster.

**Software / Hardware Requirements:**

In this section I took out the most important parts of the manual that matter, but you need more information about this section then go to chapter 5 in the Lustre manual. Also consult the chapter in the manual for getting the specific calculations for figuring out how to calculate the minimum requirements for storage and memory amount.

**General Information to think about:**

A Lustre file system can utilize any kind of block storage device such as single disks, software RAID, hardware RAID, or a logical volume manager.

Expensive switches are not needed because point-to-point connections between the servers and the storage arrays normally provide the simplest and best attachments.

For a production environment, it is preferable that the MGS have separate storage to allow future expansion to multiple file systems. However, it is possible to run the MDS and MGS on the same machine and have them share the same storage device.

Only servers running on 64-bit CPUs are tested and supported. 64-bit CPU clients are typically used for testing to match expected customer usage. The servers really only have been using SUSE or Redhat operating system to operate on.

The Lustre file system uses journaling file system technology on both the MDTs and OSTs. For a MDT, as much as a 20 percent performance gain can be obtained by placing the journal on a separate device.

**MGT:**

MGT storage requirements are small (less than 100MB even in the largest Lustre file systems), and the data on an MGT is only accessed on a server / client mount, so disk performance is not a consideration. However, this data is vital for file system access, so the MGT should be reliable storage, preferably mirrored RAID1.

**MDS:**

MDS storage is accessed in a database-like access pattern with many seeks and read-and-writes of small amounts of data.

**MDT:**

For maximum performance, the MDT should be configured as RAID1 with an internal journal and two disks from different controllers.

If you need a larger MDT, create multiple RAID1 devices from pairs of disks, and then make a RAID0 array of the RAID1 devices. This ensures maximum reliability because multiple disk failures only have a small chance of hitting both disks in the same RAID1 device.

The size of the MDT backing file system depends on the number of inodes needed in the total Lustre file system, while the aggregate OST space depends on the total amount of data stored on the file system.

It is recommended that the MDT have at least twice the minimum number of inodes to allow for future expansion and allow for an average file size smaller than expected.

If the MDT has too few inodes, this can cause the space on the OSTs to be inaccessible since no new files can be created.

**OSS:**

The data access pattern for the OSS storage is a streaming I/O pattern that is dependent on the access patterns of applications being used. Each OSS can manage multiple OSTs, one for each volume with I/O traffic load balanced between servers and targets. An OSS should be configured to have a balance between the network bandwidth and the attached storage bandwidth to prevent bottlenecks in the I/O path. Depending on the server hardware, an OSS typically servers between 2 and 8 targets, with each target between 24-48TB, but may be up to 256 terabytes (TBs) in size.

**Clients:**

A minimum of 2 GB RAM is recommended for clients.