**Lustre File System**

**Intro:**

The Lustre architecture is a storage architecture for clusters. The Lustre architecture is used for many different kinds of clusters. It is best known for powering many of the largest high-performance computing clusters worldwide. A computer cluster consists of a set of loosely or tightly connected computer that work together so that, in many respects, they can be viewed as a single system. The ability of a Lustre file system to scale capacity and performance for any need reduces the need to deploy many separate file systems, such as one for each compute cluster. The Lustre file system is best suited for uses that exceed the capacity that a single server can provide, though in some use cases, a Lustre file system can perform better with a single server than other file systems due to its strong locking and data coherency.

**Components:**

**Management Servers (MGS):** stores configuration information for all the system in a cluster and provides this information to other components. Each target contacts the MGS to provide information, and clients contact the MGS to retrieve information.

**Metadata Servers (MDS):** The MDS makes metadata stored in one or more MDTs available to the clients. Each MDS manages the names and directories in the file system and provides network request handling for one or more local MDTs.

**Metadata Targets (MDT):** Stores metadata (such as filenames, directories, permissions and file layout) on storage attached to an MDS. Each file system has one MDT. An MDT on a shared storage target can be available to multiple MDSs, although only one can access it at a time. IF an active MDS fails, a standby MDS can serve the MDT and make it available to clients. This is referred to as MDS failover.

**Object Storage Servers (OSS):** The OSS provides file I/O service and network request handling for one or more local OSTs. Typically, an OSS serves between two and eight OSTs, up to 16TB each. A typical configuration is an MDT on a dedicated node; two or more OSTs on each OSS node, and a client on each of a large number of compute nodes.

**Object Storage Targets (OST):** User file data is stored in one or more objects, each object on a separate OST in the system. The number of objects per file is configurable by the user and can be tuned to optimize performance for a given workload.

**Logical Object Volume (LOV):** Aggregates the OSCs to provide transparent access across all the OSTs. A client with the system mounted sees a single, coherent, synchronized namespace. Several clients can write to different parts of the same file simultaneously, while, at the same time, other clients can read from the file.

**Logical Metadata Volume (LMV):** Aggregates the MDSs to provide transparent access across all the MDTs in a similar manner as the LOV does for file access. This allows the client to see the directory tree on multiple MDTs as a single coherent namespace, and striped directories are merged on the clients to form a single visible directory to users and applications.

**LNet**

Supports many commonly-used network types, such as InfiniBand and IP networks, and allows simultaneous availability across multiple network types with routing between them. Remote directory 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, for example ksocklnd is the driver, which implements the TCP Socket LND that supports TCP networks. LNDs are loaded into the driver stack, with one LND for each network type in use.

**Some of the Key Features:**

* 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.

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 the data has been transmitted while the LND layer is connection oriented and typically does verify data transmission.

In certain circumstances it might be desirable for Lustre file system traffic to pass between multiple LNets. This is possible using LNet routing, but this is not the same as normal network routing.

**Supported Network Types:**

* InfiniBand: OpenFabrics OFED
* TCP (any network carrying TCP traffic)
* RapidArray
* Quadrics: Elan

**Failover:**

Availability is accomplished by replicating hardware and/or software so that when a primary server fails or is unavailable, a standby server can be switched into its place to run applications and associated resources. This process, called failover, is automatic in an HA system and, in most cases, completely application transparent.

**Capabilities:**

* **Resource Fencing:** Protects physical storage from simultaneous access by two nodes.
* **Resource Management:** Starts and stops the Lustre resources as a part of failover, maintains the cluster state, and carries out other resource management tasks.
* **Health Monitoring:** Verifies the availability of hardware and network resources and responds to health indications provided by the Lustre software.

HA software is responsible for detecting failure of the primary Lustre server node and controlling the failover. The Lustre software works with any HA software that includes resources (I/O) fencing. For proper resource fencing, the HA software must be able to completely power off the failed server or disconnect it from the shared storage device. IF two active nodes have access to the same storage device, data may be severely corrupted.

**Configurations:**

* Active/Passive Pair: In this configuration, the active node provides resources and serves data, while the passive node is usually standing by idle. If the active node fails, the passive node takes over and becomes active.
* Active/Active Pair: In this configuration, both nodes are active, each providing a subset of resources. In case of a failure, the second node takes over resources from the failed node.

Failover in a Lustre file system requires two nods be configured as a failover pair, which must share one or more storage devices. A Lustre file system can be configured to provide MDT or OST failover. For MDT failover, two MDSs can be configured to server the same MDT. Only the MDS node can serve an MDT at a time. For OST failover, multiple OSS nodes can be configured to be able to serve the same OST. However, only one OSS node can serve the OST at a time. An OST can be moved between OSS nodes that have access to the same storage device using umount/mount commands. Lustre software provides failover functionality only at the file system level.

**Performance:**

**I/O Tools:** There is a Lustre I/O tool kit that is used to benchmark the Lustre file system hardware and validate that it is working correctly. This should be done before install the Lustre Software. The tool kit can also be used to validate the performance of the various hardware and software layers in the cluster, and also to find and troubleshoot I/O issues. Typically, performance is measured starting with single raw devices and then proceeding to groups of devices. Once raw performance has been established, other software layers are then added incrementally and tested.

There are three tests, each of which tests a progressively higher layer in the Lustre software stack:

**sgpdd-survey** – Measure basic ‘bare metal’ performance of devices while by passing the kernel block device layers, buffer cache, and file system. The data gathered by this survey can help set expectations for the performance of a Lustre OST using this device. The script uses sgp\_dd to carry out raw sequential disk I/O. It runs with variable numbers of sgp\_dd threads to show how performance varies with different request queue depths.The script spawns variable numbers of sgp\_dd instances, each reading or writing a separate area of the disk to demonstrate performance variance within a number of concurrent strip files. For more information look at chapter 29.

**obdfilter-survey** – is a script that generates sequential I/O from varying numbers of threads and objects (files) to simulate the I/O patterns of a Lustre client. This script can be run directly on the OSS node to measure the OST storage performance without any intervening network, or it can be run remotely on a Lustre client to measure the OST performance including network overhead. This script can be potentially destructive and there is a small risk data may be lost. To reduce the risk, this script should not be run on devices that contain data that needs to be preserved. Lastly this script should only be ran on individual components and must be customizable depending on components that it will be running on. For more information look at chapter 29.

**ost-survey –** is a tool that is a shell script that uses lfs setstripe to perform I/O against a single OST. The script writes a file (currently using dd) to each OST in the Lustre file system, and compares read and write speeds. The tool is used to detect anomalies between other wise identical disk subsystems.For more information look at chapter 29.

Typically with these tests, a Lustre file system should deliver 85 – 90% of the raw device performance. There is a utility stats-collect in the tool kit that allows the user to collect application profiling information from Lustre clients and servers.

**Notes:**

* Scalability and performance are dependent on available disk and network bandwidth and the processing power of the servers in the system.
* There are user and group quotas on Lustre.
* A Lustre installation can be scaled up or down with respect to the number of client nodes, disk storage and bandwidth.
* The number of objects per file is configurable by the user and can be tuned to optimize performance for a given workload.
* The available bandwidth of the Lustre file system is as follows: the network bandwidth equals the aggregated bandwidth of the OSSs to the targets. The disk bandwidth equals the sum of the disk bandwidths of the storage targets (OSTs) up to the limit of the network bandwidth. The aggregate bandwidth equals the minimum of the disk bandwidth and the network bandwidth. The available file system space equals the sum of the available space of all the OSTs.
* One of the main factors leading to the high performance of the system is the ability to stripe data across multiple OSTs in a round-robin fashion. Users can optionally configure for each file the number of stripes, strip size, and OSTs that are used. Striping can be used to improve performance when the aggregate bandwidth to a single file exceeds the bandwidth of a single OST. The ability to stripe is also useful when a single OST does not have enough free space to hold an entire file. The number of objects in a single file is called the stripe\_count. The default value for stripe\_count is 1 stripe for file and the default value for stripe\_count is 1MB. The user may change these values on a per directory or per file basis. The maximum file size is not limited by the size of a single target. In the system, files can be striped across multiple objects (up to 2000), and each object can be up to 16TB in size with ldiskfs, or up to 256PB with ZFS. This leads to maximum file size of 31.25PB for ldiskfs or 8EB with ZFS.

**Basic Specifications / Information:**

* The size of a Lustre file system and aggregate cluster bandwidth can be increased without interruption by adding new OSTs and MDTs to the cluster.
* Uses RAID-0 striping and balances space usage across OSTs.
* The layout of files across OSTs can be configured on a per file, per directory, or per file system basis.
* The Lustre file system uses an improved version of the ext4 journaling file system to store data and metadata. This version, called ldiskfs, has been enhanced to improve performance.
* No single point of failure, and has multiple mount protection (MMP) provides integrated protection from errors in highly available systems that would otherwise cause file system corruptions.
* Provides an online distributed file system check (LFSCK) that can restore consistency between storage components in case of a major file system error.

**Tuning a Lustre File System:**

There are a ton of parameters that Lustre uses to configure with and use with other commands to adjust such configurations. Many options in the Lustre software are set by means of kernel module parameters. These parameters are contained in the /etc/modprobe.d/lustre.conf file. A lot more of this information can be found in chapter 30.

**Optimizing the Number of Service Threads:**

An OSS can have a minimum of two service threads and a maximum of 512 service threads. The number of service threads is a function of how much RAM and how many CPUs are on each OSS node (1 thread / 128MB \* num\_cpus). For example, if the load on the OSS node is high, new service threads will be started in order to process more requests concurrently, up to 4x the initial number of threads. One could potentially increase and decrease the size of the thread pool count to optimize the OSS.

Increase the number of I/O threads allows the kernel and storage to aggregate many writes together for more efficient disk I/O. It is very important to consider memory consumption when increasing the thread pool size. Drives are only able to sustain a certain amount of parallel I/O activity before performance is degraded, due to the high number of seeks and the OST threads just waiting for I/O.

Determining the optimum number of OSS threads is a process of trial and error, and varies for each particular configuration. Variables include the number of OSTs on each OSS, number and speed of disks, RAID configuration, and available RAM. Note: might want to start with a number of OST threads equal to the number of actual disk spindles on the node.

**Specifying the OSS Service Thread Count:**

The oss\_num\_threads parameter enables the number of OST service threads to be specified at module load time on the OSS nodes. After startup, the minimum and max

**Commands:**

**Note** – All the commands that are given below take a variety of options / parameters, I just provided a basic description on where to get your search started. You can find almost all these in chapter 40 or on Google.

**e2scan** – a utility that is an ext2 file system-modified inode scan program. This command uses the libext2fs to find inodes with ctime or mtime newer than a given time and prints out their pathname. Use this command to efficiently generate lists of files that have been modified.

**ior-survey** – a utility that is a script used to run the IOR benchmark.

**lctl** – a utility that is used for root control and configuration. With lctl you can directly control Lustre via an ioctl interface, allowing various configuration, maintenance and debugging features to be accessed.

**llobdstat** – a utility that displays OSTs statistics for the given ost\_name. This command should be ran directly on the OSS node.

**llog\_reader** – a utility that translates a Lustre configuration log into human-readable form.

**llstat** – a utility that displays Lustre statistics.

**llverdev** – a utility that verifies a block device is functioning properly over its full size. The command also verifies that the test pattern across the entire device to ensure that data is accessible after it was written, and that data written to one part of the disk is not overwriting data on another part of the disk.

**ll\_decode\_filter\_fid** – a utility displays the Lustre object ID and MDT parent FID. Basically the utility decodes and prints the Lustre OST object ID.

**ll\_recover\_lost\_found\_objs** – a utility helps recover Lustre OST objects (file data) from a lost and found directory and return them to their correct locations.

**lr\_reader** – a utility that translates the content of the last\_rcvd and reply\_data files into human readable form.

**lshowmount** – a utility that shows Lustre exports. It also shows the hosts that have Lustre mounted to a Server. This utility looks for exports from the MGS, MDS, and obdfilter.

**lst** – a utility that starts the LNet self-test. This test helps site administrators confirm that Lustre Networking has been properly installed and configured. The self-test also confirms that LNet and the network software and hardware underlying it are performing as expected.

**lustre\_rmmod.sh** – is a utility that removes all Lustre and LNet modules (assuming no Lustre services are running. This command does not work if Lustre modules are being used or if you have manually run the lctl network up command.

**lustre\_rsync** – a utility that synchronizes (replicates) a Lustre file system to a target file system.

**l\_getidentity** – a utility that handles Lustre user / group cache upcall. The l\_getidentity files are located at: /proc/fs/lustre/mdt/${FSNAME}-MDT{xxxx}/identity\_upcall.

**mkfs.lustre** – a utility that formats a disk for Lustre Service. After formatting, a disk can be mounted to start the Lustre service defined by this command.

**mount.lustre** – a utility that starts a Lustre client or target service.

**obdfilter-survey** – a utility that is a shell script that test performance of isolated OSTs, the network via echo clients, and an end to end test.

**ost-survey** – a utility that is an OST performance survey that tests client-to-disk performance of the individual OSTs in a Lustre File System.

**plot-llstat** – a utility that plots Lustre statistics. This command generates a CSV file and instruction files for gnuplot from the output of llstat.

**routerstat** – is a utility that prints Lustre router statistics.

**sgpdd-survey** – a utility that test ‘bare-metal’ performance, bypassing as much of the kernel as possible.

**stats-collect** – a utility that contains scripts used to collect application profiling information from Lustre clients and servers.

**tunefs.lustre** – is a utility that modifies configuration information on a Lustre target disk. This does not reformat the disk or erase the target information, but modifying the configuration information can result in an unusable file system.

**Maintenance:**

**Notes:**

* Multiple Remote Direct Memory Access networks can be bridged using Lustre routing for maximum performance. The Lustre software also includes integrated network diagnostics.
* Lustre can have a possible configuration that has an active / active failover of multiple MDT’s. This allows scaling the metadata performance of Lustre file systems with the addition of MDT storage devices and MDS nodes.
* Offers a variety of mechanisms to examine performance and tuning, let’s find out what these are!
* Since Lustre software release 2.8, DNE also allows the file system to distribute files of a single directory over multiple MDT nodes. A directory distributed across multiple MDTs is known as a striped directory. The question that I have to ask is that with our systems having 2.5, are we losing out on performance and efficiency here? Looking at the chart I can assume that the MDS and MDT can be bottlenecks for the supercomputer if the file structure ever gets to big.
* The backing OST storage should be RAID 5 or, preferably, RAID 6 storage. MDT storage should be RAID 1 or RAID 10.
* To be able to find the current version of Lustre that is in use on the client can be found using the command lctl get\_param version.

**Links:**

Community / Main website: <http://lustre.org>

Support / Training: http://www.hpdd.intel.com

Latest Documentation: https://wiki.hpdd.intel.com/display/PUB/Documentation

**Vocab:**

**Data-Coherency:** is data that is the same across the network, in other words data on the server and all the clients is synchronized.

**Metadata:** a set of data that describes and gives information about other data.

**Lustre Acronyms:**

ACL: Access Control List

DNE: Distributed Namespace Environment

FIDs: Lustre File Identifiers

HA: High Availability

LDLM: Lustre Distributed Lock Manager

LFSCK: Lustre File System Check

LMV: Logical Metadata Volume

LND: Lustre Network Driver

LNET: Lustre Networking

LOV: Logical Object Volume

MC: Management Client

MDC: Metadata Client

MDS: Metadata Server

MDT: Metadata Target

MGS: Management Server

MGT: Management Target

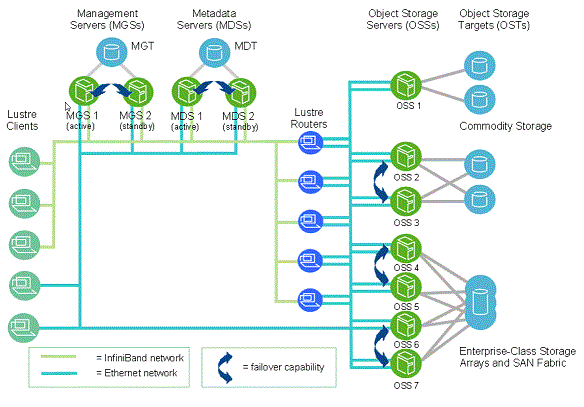
OID: Object ID

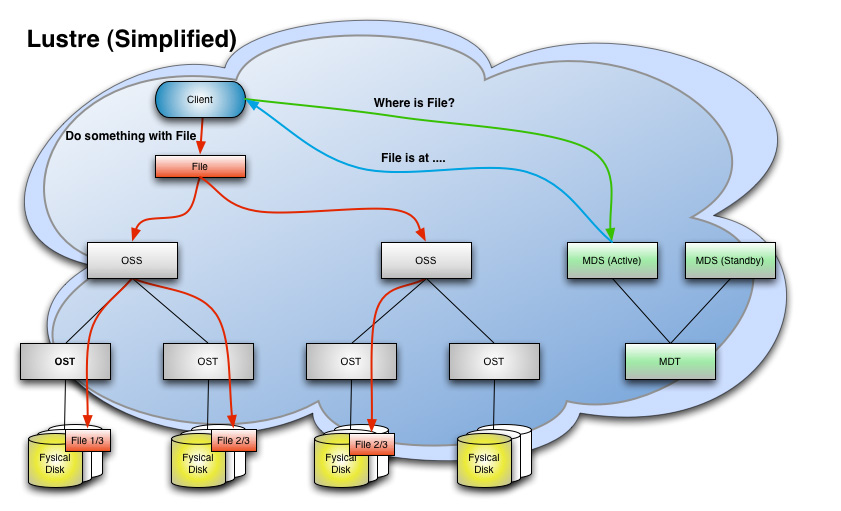
OSS: Object Storage Servers

OST: Object Storage Target

OSC: Object Storage Clients

**Lustre File System Layout**



****