Lustre (File System)

Subjects: Computer Science, Software Engineering
Contributor: HandWiki Huang

Lustre is a type of parallel distributed file system, generally used for large-scale cluster computing. The name Lustre is a portmanteau word derived from Linux and cluster. Lustre file system software is available under the GNU General Public License (version 2 only) and provides high performance file systems for computer clusters ranging in size from small workgroup clusters to large-scale, multi-site systems. Since June 2005, Lustre has consistently been used by at least half of the top ten, and more than 60 of the top 100 fastest supercomputers in the world, including the world's No. 1 ranked TOP500 supercomputer in June 2020, Fugaku, as well as previous top supercomputers such as Titan and Sequoia. Lustre file systems are scalable and can be part of multiple computer clusters with tens of thousands of client nodes, tens of petabytes (PB) of storage on hundreds of servers, and more than a terabyte per second (TB/s) of aggregate I/O throughput. This makes Lustre file systems a popular choice for businesses with large data centers, including those in industries such as meteorology, simulation, oil and gas, life science, rich media, and finance. The I/O performance of Lustre has widespread impact on these applications and has attracted broad attention.

Keywords: simulation ; meteorology ; multi-site

1. History

The Lustre file system architecture was started as a research project in 1999 by Peter J. Braam, who was a staff of Carnegie Mellon University (CMU) at the time. Braam went on to found his own company Cluster File Systems in 2001,[1] starting from work on the InterMezzo file system in the Coda project at CMU.[2] Lustre was developed under the Accelerated Strategic Computing Initiative Path Forward project funded by the United States Department of Energy, which included Hewlett-Packard and Intel.[3] In September 2007, Sun Microsystems acquired the assets of Cluster File Systems Inc. including its ”intellectual property“.[4][5] Sun included Lustre with its high-performance computing hardware offerings, with the intent to bring Lustre technologies to Sun's ZFS file system and the Solaris operating system. In November 2008, Braam left Sun Microsystems, and Eric Barton and Andreas Dilger took control of the project. In 2010 Oracle Corporation, by way of its acquisition of Sun, began to manage and release Lustre.

In December 2010, Oracle announced that they would cease Lustre 2.x development and place Lustre 1.8 into maintenance-only support, creating uncertainty around the future development of the file system.[6] Following this announcement, several new organizations sprang up to provide support and development in an open community development model, including Whamcloud,[7] Open Scalable File Systems, Inc. (OpenSFS), EUROPEAN Open File Systems (EOFS) and others. By the end of 2010, most Lustre developers had left Oracle. Braam and several associates joined the hardware-oriented Xyratex when it acquired the assets of ClusterStor,[8][9] while Barton, Dilger, and others formed software startup Whamcloud, where they continued to work on Lustre.[10]

In August 2011, OpenSFS awarded a contract for Lustre feature development to Whamcloud.[11] This contract covered the completion of features, including improved Single Server Metadata Performance scaling, which allows Lustre to better take advantage of many-core metadata server; online Lustre distributed filesystem checking (LFSCK), which allows verification of the distributed filesystem state between data and metadata servers while the filesystem is mounted and in use; and Distributed Namespace Environment (DNE), formerly Clustered Metadata (CMD), which allows the Lustre metadata to be distributed across multiple servers. Development also continued on ZFS-based back-end object storage at Lawrence Livermore National Laboratory.[12] These features were in the Lustre 2.2 through 2.4 community release roadmap.[13] In November 2011, a separate contract was awarded to Whamcloud for the maintenance of the Lustre 2.x source code to ensure that the Lustre code would receive sufficient testing and bug fixing while new features were being developed.[14]

In July 2012 Whamcloud was acquired by Intel,[15][16] after Whamcloud won the FastForward DOE contract to extend Lustre for exascale computing systems in the 2012 timeframe.[17] OpenSFS then transitioned contracts for Lustre development to Intel.

In February 2013, Xyratex Ltd., announced it acquired the original Lustre trademark, logo, website and associated intellectual property from Oracle.[8] In June 2013, Intel began expanding Lustre usage beyond traditional HPC, such as within Hadoop.[18] For 2013 as a whole, OpenSFS announced request for proposals (RFP) to cover Lustre feature development, parallel file system tools, addressing Lustre technical debt, and parallel file system incubators.[19] OpenSFS also established the Lustre Community Portal, a technical site that provides a collection of information and documentation in one area for reference and guidance to support the Lustre open source community. On April 8, 2014, Ken Claffey announced that Xyratex/Seagate was donating the lustre.org domain back to the user community,[20] and this was completed in March, 2015.

In June 2018, the Lustre team and assets were acquired from Intel by DDN. DDN organized the new acquisition as an independent division, reviving the Whamcloud name for the new division.[21]

In November 2019, OpenSFS and EOFS announced at the SC19 Lustre BOF that the Lustre trademark had been transferred to them jointly from Seagate.[22]

2. Release History

Lustre file system was first installed for production use in March 2003 on the MCR Linux Cluster at the Lawrence Livermore National Laboratory,[23] one of the largest supercomputers at the time.[24]

Lustre 1.0.0 was released in December 2003,[25] and provided basic Lustre filesystem functionality, including server failover and recovery.

Lustre 1.2.0, released in March 2004, worked on Linux kernel 2.6, and had a "size glimpse" feature to avoid lock revocation on files undergoing write, and client side data write-back cache accounting (grant).

Lustre 1.4.0, released in November 2004, provided protocol compatibility between versions, could use InfiniBand networks, and could exploit extents/mballoc in the ldiskfs on-disk filesystem.

Lustre 1.6.0, released in April 2007, allowed mount configuration (“mountconf”) allowing servers to be configured with "mkfs" and "mount", allowed dynamic addition of *object storage targets* (OSTs), enabled Lustre distributed lock manager (LDLM) scalability on symmetric multiprocessing (SMP) servers, and provided free space management for object allocations.

Lustre 1.8.0, released in May 2009, provided OSS Read Cache, improved recovery in the face of multiple failures, added basic heterogeneous storage management via OST Pools, adaptive network timeouts, and version-based recovery. It was a transition release, being interoperable with both Lustre 1.6 and Lustre 2.0.[26]

Lustre 2.0, released in August 2010, was based on significant internally restructured code to prepare for major architectural advancements. Lustre 2.x *clients* cannot interoperate with 1.8 or earlier *servers*. However, Lustre 1.8.6 and later clients can interoperate with Lustre 2.0 and later servers. The Metadata Target (MDT) and OST on-disk format from 1.8 can be upgraded to 2.0 and later without the need to reformat the filesystem.

Lustre 2.1, released in September 2011, was a community-wide initiative in response to Oracle suspending development on Lustre 2.x releases.[27] It added the ability to run servers on Red Hat Linux 6 and increased the maximum ext4-based OST size from 24 TB to 128 TB,[28] as well as a number of performance and stability improvements. Lustre 2.1 servers remained inter-operable with 1.8.6 and later clients.

Lustre 2.2, released in March 2012, focused on providing metadata performance improvements and new features.[29] It added parallel directory operations allowing multiple clients to traverse and modify a single large directory concurrently, faster recovery from server failures, increased stripe counts for a single file (across up to 2000 OSTs), and improved single-client directory traversal performance.

Lustre 2.3, released in October 2012, continued to improve the metadata server code to remove internal locking bottlenecks on nodes with many CPU cores (over 16). The object store added a preliminary ability to use ZFS as the backing file system. The Lustre File System CheckK (LFSCK) feature can verify and repair the MDS Object Index (OI) while the file system is in use, after a file-level backup/restore or in case of MDS corruption. The server-side IO statistics were enhanced to allow integration with batch job schedulers such as SLURM to track per-job statistics. Client-side software was updated to work with Linux kernels up to version 3.0.

Lustre 2.4, released in May 2013, added a considerable number of major features, many funded directly through OpenSFS. Distributed Namespace Environment (DNE) allows horizontal metadata capacity and performance scaling for 2.4 clients, by allowing subdirectory trees of a single namespace to be located on separate MDTs. ZFS can now be used as the backing filesystem for both MDT and OST storage. The LFSCK feature added the ability to scan and verify the internal consistency of the MDT FID and LinkEA attributes. The Network Request Scheduler[30] [31] (NRS) adds policies to optimize client request processing for disk ordering or fairness. Clients can optionally send bulk RPCs up to 4 MB in size. Client-side software was updated to work with Linux kernels up to version 3.6, and is still interoperable with 1.8 clients.

Lustre 2.5, released in October 2013, added the highly anticipated feature, Hierarchical Storage Management (HSM). A core requirement in enterprise environments, HSM allows customers to easily implement tiered storage solutions in their operational environment. This release is the current OpenSFS-designated Maintenance Release branch of Lustre.[32][33] [34][35] The most recent maintenance version is 2.5.3 and was released in September 2014.[36]

Lustre 2.6, released in July 2014,[37] was a more modest release feature wise, adding LFSCK functionality to do local consistency checks on the OST as well as consistency checks between MDT and OST objects. The NRS Token Bucket Filter[38] (TBF) policy was added. Single-client IO performance was improved over the previous releases.[39] This release also added a preview of DNE striped directories, allowing single large directories to be stored on multiple MDTs to improve performance and scalability.

Lustre 2.7, released in March 2015,[40] added LFSCK functionality to verify DNE consistency of remote and striped directories between multiple MDTs. Dynamic LNet Config adds the ability to configure and modify LNet network interfaces, routes, and routers at runtime. A new evaluation feature was added for UID/GID mapping for clients with different administrative domains, along with improvements to the DNE striped directory functionality.

Lustre 2.8, released in March 2016,[41] finished the DNE striped directory feature, including support for migrating directories between MDTs, and cross-MDT hard link and rename. As well, it included improved support for Security-Enhanced Linux (SELinux) on the client, Kerberos authentication and RPC encryption over the network, and performance improvements for LFSCK.

Lustre 2.9 was released in December 2016[42] and included a number of features related to security and performance. The Shared Secret Key security flavour uses the same GSSAPI mechanism as Kerberos to provide client and server node authentication, and RPC message integrity and security (encryption). The Nodemap feature allows categorizing client nodes into groups and then mapping the UID/GID for those clients, allowing remotely administered clients to transparently use a shared filesystem without having a single set of UID/GIDs for all client nodes. The subdirectory mount feature allows clients to mount a subset of the filesystem namespace from the MDS. This release also added support for up to 16MiB RPCs for more efficient I/O submission to disk, and added the `ladvise` interface to allow clients to provide I/O hints to the servers to prefetch file data into server cache or flush file data from server cache. There was improved support for specifying filesystem-wide default OST pools, and improved inheritance of OST pools in conjunction with other file layout parameters.

Lustre 2.10 was released in July 2017[43] and has a number of significant improvements. The LNet Multi-Rail (LMR) feature allows bonding multiple network interfaces (InfiniBand, Omni-Path, and/or Ethernet) on a client and server to increase aggregate I/O bandwidth. Individual files can use composite file layouts that are constructed of multiple components, which are file regions based on the file offset, that allow different layout parameters such as stripe count, OST pool/storage type, etc. Progressive File Layout (PFL) is the first feature to use composite layouts, but the implementation is flexible for use with other file layouts such as mirroring and erasure coding. The NRS Token Bucket Filter (TBF) server-side scheduler has implemented new rule types, including RPC-type scheduling and the ability to specify multiple parameters such as JobID and NID for rule matching. Tools for managing ZFS snapshots of Lustre filesystems have been added, to simplify the creation, mounting, and management of MDT and OST ZFS snapshots as separate Lustre mountpoints.

Lustre 2.11 was released in April 2018[44] and contains two significant new features, and several smaller features. The File Level Redundancy (FLR) feature expands on the 2.10 PFL implementation, adding the ability to specify mirrored file layouts for improved availability in case of storage or server failure and/or improved performance with highly concurrent reads. The Data-on-MDT (DoM) feature allows small (few MiB) files to be stored on the MDT to leverage typical flash-based RAID-10 storage for lower latency and reduced IO contention, instead of the typical HDD RAID-6 storage used on OSTs. As well, the LNet Dynamic Discovery feature allows auto-configuration of LNet Multi-Rail between peers that share

an LNet network. The LDLM Lock Ahead feature allows appropriately modified applications and libraries to pre-fetch DLM extent locks from the OSTs for files, if the application knows (or predicts) that this file extent will be modified in the near future, which can reduce lock contention for multiple clients writing to the same file.

Lustre 2.12 was released on December 21, 2018[45] and focused on improving Lustre usability and stability, with improvements the performance and functionality of the FLR and DoM features added in Lustre 2.11, as well as smaller changes to NRS TBF, HSM, and JobStats. It added LNet Network Health to allow the LNet Multi-Rail feature from Lustre 2.10 to better handle network faults when a node has multiple network interfaces. The Lazy Size on MDT[46] (LSOM) feature allows storing an estimate of the file size on the MDT for use by policy engines, filesystem scanners, and other management tools that can more efficiently make decisions about files without a fully accurate file sizes or blocks count without having to query the OSTs for this information. This release also added the ability to manually *restripe* an existing directory across multiple MDTs, to allow migration of directories with large numbers of files to use the capacity and performance of several MDS nodes. The Lustre RPC data checksum added SCSI T10-PI integrated data checksums[47] from the client to the kernel block layer, SCSI host adapter, and T10-enabled hard drives.

Lustre 2.13 was released on December 5, 2019[48] and added a new performance-related features Persistent Client Cache[49] (PCC), which allows direct use of NVMe and NVRAM storage on the client nodes while keeping the files part of the global filesystem namespace, and OST Overstriping[50] which allows files to store multiple stripes on a single OST to better utilize fast OSS hardware. As well, the LNet Multi-Rail Network Health functionality was improved to work with LNet RDMA router nodes. The PFL functionality was enhanced with Self-Extending Layouts[51] (SEL) to allow file components to be dynamically sized, to better deal with flash OSTs that may be much smaller than disk OSTs within the same filesystem. The release also included a number of smaller improvements, such as balancing DNE remote directory creation across MDTs, using Lazy-size-on-MDT to reduce the overhead of "lfs find", directories with 10M files per shard for ldiskfs, and bulk RPC sizes up to 64MB.[52]

Lustre 2.14 was released on February 19, 2021[53] and includes three main features. Client Data Encryption implements fscrypt to allow file data to be encrypted on the client before network transfer and persistent storage on the OST and MDT. OST Pool Quotas extends the quota framework to allow the assignment and enforcement of quotas on the basis of OST storage pools. DNE Auto Restriping can now adjust how many MDTs a large directory is striped over based on size thresholds defined by the administrator, similar to Progressive File Layouts for directories.

Lustre 2.15 was released on June 16, 2022[54] and includes three main features. Client Directory Encryption[55] expands on the fscrypt data encryption in the 2.14 release to also allow file and directory names to be encrypted on the client before network transfer and persistent storage on the MDT. DNE MDT space balancing automatically balances new directory creation across MDTs in the filesystem in round-robin and/or based on available inodes and space, which in turn helps distribute client metadata workload over MDTs more evenly. For applications using the NVIDIA GPU Direct Storage interface (GDS), the Lustre client can do zero-copy RDMA read and write from the storage server directly into the GPU memory to avoid an extra data copy from CPU memory and extra processing overhead.[56] User Defined Selection Policy (UDSP) allows setting interface selection policies for nodes with multiple network interfaces.

3. Architecture

A Lustre file system has three major functional units:

- One or more *metadata servers (MDS)* nodes that have one or more *metadata target (MDT)* devices per Lustre filesystem that stores namespace metadata, such as filenames, directories, access permissions, and file layout. The MDT data is stored in a local disk filesystem. However, unlike block-based distributed filesystems, such as GPFS and PanFS, where the metadata server controls all of the block allocation, the Lustre metadata server is only involved in pathname and permission checks, and is not involved in any file I/O operations, avoiding I/O scalability bottlenecks on the metadata server. The ability to have multiple MDTs in a single filesystem is a new feature in Lustre 2.4, and allows directory subtrees to reside on the secondary MDTs, while 2.7 and later allow large single directories to be distributed across multiple MDTs as well.
- One or more *object storage server (OSS)* nodes that store file data on one or more *object storage target (OST)* devices. Depending on the server's hardware, an OSS typically serves between two and eight OSTs, with each OST managing a single local disk filesystem. The capacity of a Lustre file system is the sum of the capacities provided by the OSTs.
- *Client(s)* that access and use the data. Lustre presents all clients with a unified namespace for all of the files and data in the filesystem, using standard POSIX semantics, and allows concurrent and coherent read and write access to the files in the filesystem.

The MDT, OST, and client may be on the same node (usually for testing purposes), but in typical production installations these devices are on separate nodes communicating over a network. Each MDT and OST may be part of only a single filesystem, though it is possible to have multiple MDTs or OSTs on a single node that are part of different filesystems. The *Lustre Network (LNet)* layer can use several types of network interconnects, including native InfiniBand verbs, Omni-Path, RoCE, and iWARP via OFED, TCP/IP on Ethernet, and other proprietary network technologies such as the Cray Gemini interconnect. In Lustre 2.3 and earlier, Myrinet, Quadrics, Cray SeaStar and RapidArray networks were also supported, but these network drivers were deprecated when these networks were no longer commercially available, and support was removed completely in Lustre 2.8. Lustre will take advantage of remote direct memory access (RDMA) transfers, when available, to improve throughput and reduce CPU usage.

The storage used for the MDT and OST backing filesystems is normally provided by hardware RAID devices, though will work with any block devices. Since Lustre 2.4, the MDT and OST can also use ZFS for the backing filesystem in addition to ext4, allowing them to effectively use JBOD storage instead of hardware RAID devices. The Lustre OSS and MDS servers read, write, and modify data in the format imposed by the backing filesystem and return this data to the clients. This allows Lustre to take advantage of improvements and features in the underlying filesystem, such as compression and data checksums in ZFS. Clients do not have any direct access to the underlying storage, which ensures that a malfunctioning or malicious client cannot corrupt the filesystem structure.

An OST is a dedicated filesystem that exports an interface to byte ranges of file objects for read/write operations, with extent locks to protect data consistency. An MDT is a dedicated filesystem that stores inodes, directories, POSIX and extended file attributes, controls file access permissions/ACLs, and tells clients the layout of the object(s) that make up each regular file. MDTs and OSTs currently use either an enhanced version of ext4 called *ldiskfs*, or ZFS/DMU for back-end data storage to store files/objects[57] using the open source ZFS-on-Linux port.[58]

The client mounts the Lustre filesystem locally with a VFS driver for the Linux kernel that connects the client to the server(s). Upon initial mount, the client is provided a File Identifier (FID) for the root directory of the mountpoint. When the client accesses a file, it performs a filename lookup on the MDS. When the MDS filename lookup is complete and the user and client have permission to access and/or create the file, either the layout of an existing file is returned to the client or a new file is created on behalf of the client, if requested. For read or write operations, the client then interprets the file layout in the *logical object volume (LOV)* layer, which maps the file logical offset and size to one or more objects. The client then locks the file range being operated on and executes one or more parallel read or write operations directly to the OSS nodes that hold the data objects. With this approach, bottlenecks for client-to-OSS communications are eliminated, so the total bandwidth available for the clients to read and write data scales almost linearly with the number of OSTs in the filesystem.

After the initial lookup of the file layout, the MDS is not normally involved in file IO operations since all block allocation and data IO is managed internally by the OST. Clients do not directly modify the objects or data on the OST filesystems, but instead delegate this task to OSS nodes. This approach ensures scalability for large-scale clusters and supercomputers, as well as improved security and reliability. In contrast, shared block-based filesystems such as GPFS and OCFS allow direct access to the underlying storage by all of the clients in the filesystem, which requires a large back-end SAN attached to all clients, and increases the risk of filesystem corruption from misbehaving/defective clients.

4. Implementation

In a typical Lustre installation on a Linux client, a Lustre filesystem driver module is loaded into the kernel and the filesystem is mounted like any other local or network filesystem. Client applications see a single, unified filesystem even though it may be composed of tens to thousands of individual servers and MDT/OST filesystems.

On some massively parallel processor (MPP) installations, computational processors can access a Lustre file system by redirecting their I/O requests to a dedicated I/O node configured as a Lustre client. This approach is used in the Blue Gene installation[59] at Lawrence Livermore National Laboratory.

Another approach used in the early years of Lustre is the *liblustre* library on the Cray XT3 using the Catamount operating system on systems such as Sandia Red Storm,[60] which provided userspace applications with direct filesystem access. Liblustre was a user-level library that allows computational processors to mount and use the Lustre file system as a client. Using liblustre, the computational processors could access a Lustre file system even if the service node on which the job was launched is not a Linux client. Liblustre allowed data movement directly between application space and the Lustre

OSSs without requiring an intervening data copy through the kernel, thus providing access from computational processors to the Lustre file system directly in a constrained operating environment. The liblustre functionality was deleted from Lustre 2.7.0 after having been disabled since Lustre 2.6.0, and was untested since Lustre 2.3.0.

In Linux Kernel version 4.18, the incomplete port of the Lustre client was removed from the kernel staging area in order to speed up development and porting to newer kernels.[61] The out-of-tree Lustre client and server is still available for RHEL, SLES, and Ubuntu distro kernels, as well as vanilla kernels.

5. Data Objects and File Striping

In a traditional Unix disk file system, an inode data structure contains basic information about each file, such as where the data contained in the file is stored. The Lustre file system also uses inodes, but inodes on MDTs point to one or more OST objects associated with the file rather than to data blocks. These objects are implemented as files on the OSTs. When a client opens a file, the file open operation transfers a set of object identifiers and their layout from the MDS to the client, so that the client can directly interact with the OSS node where the object is stored. This allows the client to perform I/O in parallel across all of the OST objects in the file without further communication with the MDS.

If only one OST object is associated with an MDT inode, that object contains all the data in the Lustre file. When more than one object is associated with a file, data in the file is "striped" in chunks in a round-robin manner across the OST objects similar to RAID 0 in chunks typically 1MB or larger. Striping a file over multiple OST objects provides significant performance benefits if there is a need for high bandwidth access to a single large file. When striping is used, the maximum file size is not limited by the size of a single target. Capacity and aggregate I/O bandwidth scale with the number of OSTs a file is striped over. Also, since the locking of each object is managed independently for each OST, adding more stripes (one per OST) scales the file I/O locking capacity of the file proportionately. Each file created in the filesystem may specify different layout parameters, such as the stripe count (number of OST objects making up that file), stripe size (unit of data stored on each OST before moving to the next), and OST selection, so that performance and capacity can be tuned optimally for each file. When many application threads are reading or writing to separate files in parallel, it is optimal to have a single stripe per file, since the application is providing its own parallelism. When there are many threads reading or writing a single large file concurrently, then it is optimal to have one stripe on each OST to maximize the performance and capacity of that file.

In the Lustre 2.10 release, the ability to specify *composite layouts* was added to allow files to have different layout parameters for different regions of the file. The **Progressive File Layout** (PFL) feature uses composite layouts to improve file IO performance over a wider range of workloads, as well as simplify usage and administration. For example, a small PFL file can have a single stripe on flash for low access overhead, while larger files can have many stripes for high aggregate bandwidth and better OST load balancing. The composite layouts are further enhanced in the 2.11 release with the **File Level Redundancy** (FLR) feature, which allows a file to have multiple overlapping layouts for a file, providing RAID 0+1 redundancy for these files as well as improved read performance. The Lustre 2.11 release also added the **Data-on-Metadata** (DoM) feature, which allows the *first* component of a PFL file to be stored directly on the MDT with the inode. This reduces overhead for accessing small files, both in terms of space usage (no OST object is needed) as well as network usage (fewer RPCs needed to access the data). DoM also improves performance for small files if the MDT is SSD-based, while the OSTs are disk-based. In Lustre 2.13 the **OST Overstriping** feature allows a single component to have multiple stripes on one OST, while the **Self-Extending Layout** feature allows the component size to be dynamic during write so that it can cope with (flash) OSTs running out of space before the whole filesystem is out of space.

6. Metadata Objects and DNE Remote or Striped Directories

When a client initially mounts a filesystem, it is provided the 128-bit Lustre File Identifier (FID, composed of the 64-bit Sequence number, 32-bit Object ID, and 32-bit Version) of the root directory for the mountpoint. When doing a filename lookup, the client performs a lookup of each pathname component by mapping the parent directory FID Sequence number to a specific MDT via the FID Location Database (FLDB), and then does a lookup on the MDS managing this MDT using the parent FID and filename. The MDS will return the FID for the requested pathname component along with a DLM lock. Once the MDT of the last parent directory is determined, further directory operations (for non-striped directories) take place exclusively on that MDT, avoiding contention between MDTs. For DNE striped directories, the per-directory layout stored on the parent directory provides a hash function and a list of MDT directory FIDs across which the directory is distributed. The *Logical Metadata Volume* (LMV) on the client hashes the filename and maps it to a specific MDT directory shard, which will handle further operations on that file in an identical manner to a non-striped directory. For **readdir()**

operations, the entries from each directory shard are returned to the client sorted in the local MDT directory hash order, and the client performs a merge sort to interleave the filenames in hash order so that a single 64-bit cookie can be used to determine the current offset within the directory.

7. Locking

The Lustre distributed lock manager (LDLM), implemented in the OpenVMS style, protects the integrity of each file's data and metadata. Access and modification of a Lustre file is completely cache coherent among all of the clients. Metadata locks are managed by the MDT that stores the inode for the file, using FID as the resource name. The metadata locks are split into separate bits that protect the lookup of the file (file owner and group, permission and mode, and access control list (ACL)), the state of the inode (directory size, directory contents, link count, timestamps), layout (file striping, since Lustre 2.4), and extended attributes (xattrs, since Lustre 2.5). A client can fetch multiple metadata lock bits for a single inode with a single RPC request, but currently they are only ever granted a read lock for the inode. The MDS manages all modifications to the inode in order to avoid lock resource contention and is currently the only node that gets write locks on inodes.

File data locks are managed by the OST on which each object of the file is striped, using byte-range extent locks. Clients can be granted overlapping read extent locks for part or all of the file, allowing multiple concurrent readers of the same file, and/or non-overlapping write extent locks for independent regions of the file. This allows many Lustre clients to access a single file concurrently for both read and write, avoiding bottlenecks during file I/O. In practice, because Linux clients manage their data cache in units of pages, the clients will request locks that are always an integer multiple of the page size (4096 bytes on most clients). When a client is requesting an extent lock the OST may grant a lock for a larger extent than originally requested, in order to reduce the number of lock requests that the client makes. The actual size of the granted lock depends on several factors, including the number of currently granted locks on that object, whether there are conflicting write locks for the requested lock extent, and the number of pending lock requests on that object. The granted lock is never smaller than the originally requested extent. OST extent locks use the Lustre FID of the object as the resource name for the lock. Since the number of extent lock servers scales with the number of OSTs in the filesystem, this also scales the aggregate locking performance of the filesystem, and of a single file if it is striped over multiple OSTs.

8. Networking

The communication between the Lustre clients and servers is implemented using Lustre Networking (LNet), which was originally based on the Sandia Portals network programming application programming interface. Disk storage is connected to the Lustre MDS and OSS server nodes using direct attached storage (SAS, FC, iSCSI) or traditional storage area network (SAN) technologies, which is independent of the client-to-server network.

LNet can use many commonly used network types, such as InfiniBand and TCP (commonly Ethernet) networks, and allows simultaneous availability across multiple network types with routing between them. Remote Direct Memory Access (RDMA) is used for data and metadata transfer between nodes when provided by the underlying networks, such as InfiniBand, RoCE, iWARP, and Omni-Path, as well as proprietary high-speed networks such as Cray Aries and Gemini, and Atos BXI. High availability and recovery features enable transparent recovery in conjunction with failover servers.

Since Lustre 2.10 the LNet **Multi-Rail** (MR) feature[62] allows link aggregation of two or more network interfaces between a client and server to improve bandwidth. The LNet interface types do not need to be the same network type. In 2.12 Multi-Rail was enhanced to improve fault tolerance if multiple network interfaces are available between peers.

LNet provides end-to-end throughput over Gigabit Ethernet networks in excess of 100 MB/s,[63] throughput up to 11 GB/s using InfiniBand enhanced data rate (EDR) links, and throughput over 11 GB/s across 100 Gigabit Ethernet interfaces.[64]

9. High Availability

Lustre file system high availability features include a robust failover and recovery mechanism, making server failures and reboots transparent. Version interoperability between successive minor versions of the Lustre software enables a server to be upgraded by taking it offline (or failing it over to a standby server), performing the upgrade, and restarting it, while all active jobs continue to run, experiencing a delay while the backup server takes over the storage.

Lustre MDSes are configured as an active/passive pair exporting a single MDT, or one or more active/active MDS pairs with DNE exporting two or more separate MDTs, while OSSes are typically deployed in an active/active configuration exporting separate OSTs to provide redundancy without extra system overhead. In single-MDT filesystems, the standby

MDS for one filesystem is the MGS and/or monitoring node, or the active MDS for another file system, so no nodes are idle in the cluster.

10. HSM (Hierarchical Storage Management)

Lustre provides the capability to have multiple storage tiers within a single filesystem namespace. It allows traditional HSM functionality to copy (archive) files off the primary filesystem to a secondary archive storage tier. The archive tier is typically a tape-based system, that is often fronted by a disk cache. Once a file is archived, it can be released from the main filesystem, leaving only a stub that references the archive copy. If a released file is opened, the Coordinator blocks the open, sends a restore request to a copytool, and then completes the open once the copytool has completed restoring the file.

In addition to external storage tiering, it is possible to have multiple storage tiers within a single filesystem namespace. OSTs of different types (e.g. HDD and SSD) can be declared in named storage pools. The OST pools can be selected when specifying file layouts, and different pools can be used within a single PFL file layout. Files can be migrated between storage tiers either manually or under control of the Policy Engine. Since Lustre 2.11, it is also possible to mirror a file to different OST pools with a FLR file layout, for example to pre-stage files into flash for a computing job.

HSM includes some additional Lustre components to manage the interface between the primary filesystem and the archive:

- Coordinator: receives archive and restore requests and dispatches them to agent nodes.
- Agent: runs a copytool to copy data from primary storage to the archive and vice versa.
- Copytool: handles data motion and metadata updates. There are different copytools to interface with different archive systems. A generic POSIX copytool is available for archives that provide a POSIX-like front-end interface. Copytools are also available for the High Performance Storage System[65] (HPSS), Tivoli Storage Manager[66] (TSM), Amazon S3, [67] and Google Drive.[68]
- Policy Engine: watches filesystem Changelogs for new files to archive, applies policies to release files based on age or space usage, and communicates with MDT and Coordinator. The Policy Engine can also trigger actions like migration between, purge, and removal. The most commonly used policy engine is RobinHood, but other policy engines can also be used.

HSM also defines new states for files including: [69]

- Exist: Some copy, possibly incomplete exists in a HSM.
- Archive: A full copy exists on the archive side of the HSM.
- Dirty: The primary copy of the file has been modified and differs from the archived copy.
- Released: A stub inode exists on an MDT, but the data objects have been removed and the only copy exists in the archive.
- Lost: the archive copy of the file has been lost and cannot be restored
- No Release: the file should not be released from the filesystem
- No Archive: the file should not be archived

11. Deployments

Lustre is used by many of the TOP500 supercomputers and large multi-cluster sites. Six of the top 10 and more than 60 of the top 100 supercomputers use Lustre file systems. These include the 700TB 5GB/s Orion filesystem for the Frontier supercomputer at Oak Ridge National Laboratory (ORNL),[70] Fugaku and K Computer[12] at the RIKEN Advanced Institute for Computational Science, Tianhe-1A at the National Supercomputing Center in Tianjin, China, LUMI at CSC, Jaguar and Titan at ORNL, Blue Waters at the University of Illinois, and Sequoia and Blue Gene/L at Lawrence Livermore National Laboratory (LLNL).

There are also large Lustre filesystems at the National Energy Research Scientific Computing Center, Pacific Northwest National Laboratory, Texas Advanced Computing Center, Brazilian National Laboratory of Scientific Computing,[71] and NASA[72] in North America, in Asia at Tokyo Institute of Technology,[73] in Europe at CEA,[74][75] and many others.

12. Commercial Technical Support

Commercial technical support for Lustre is often bundled along with the computing system or storage hardware sold by the vendor. Some vendors include Hewlett-Packard (as the HP StorageWorks Scalable File Share, circa 2004 through 2008),[76] ATOS, Fujitsu.[77] Vendors selling storage hardware with bundled Lustre support include Hitachi Data Systems (2012),[78] DataDirect Networks (DDN),[79] Aeon Computing, and others. It is also possible to get software-only support for Lustre file systems from some vendors, including Whamcloud.[80]

Amazon Web Services offers Amazon FSx for Lustre,[81] a fully managed service, making it easy to launch and run high-performance file systems cost effectively in their cloud.

References

1. "Company". old web site. Cluster File Systems, Inc.. http://www.clusterfs.com/company.html.
2. Peter J. Braam (August 4, 2002). "Lustre, The Inter-Galactic File System". Presentation slides. Lawrence Livermore National Laboratory. https://asc.llnl.gov/computing_resources/bluegenel/talks/braam.pdf.
3. R. Kent Koeninger (June 2003). "The Ultra-Scalable HPTC Lustre Filesystem". Slides for presentation at Cluster World 2003. http://www.linuxclustersinstitute.org/conferences/archive/2003/PDF/C04-Koeninger_K.pdf.
4. Britta Wülfing (September 13, 2007). "Sun Assimilates Lustre Filesystem". Linux Magazine. http://www.linux-magazine.com/online/news/sun_assimilates_lustre_filesystem?category=13402.
5. "Sun Microsystems Expands High Performance Computing Portfolio with Definitive Agreement to Acquire Assets of Cluster File Systems, Including the Lustre File System". Press release (Sun Microsystems). September 12, 2007. http://www.sun.com/aboutsun/pr/2007-09/sunflash.20070912.2.xml.
6. "Oracle has Kicked Lustre to the Curb". Inside HPC. 2011-01-10. http://insidehpc.com/2011/01/10/inside-track-oracle-has-kicked-lustre-to-the-curb/.
7. J. Leidel (August 20, 2010). "Whamcloud aims to make sure Lustre has a future in HPC". Inside HPC. http://insidehpc.com/2010/08/20/whamcloud-aims-to-make-sure-lustre-has-a-future-in-hpc/.
8. "Xyratex Advances Lustre® Initiative, Assumes Ownership of Related Assets". Press release (Xyratex). February 19, 2013. http://www.xyratex.com/news/press-releases/xyratex-advances-lustre%C2%AE-initiative-assumes-ownership-related-assets.
9. Rich Brueckner (November 9, 2010). "Bojanic & Braam Getting Lustre Band Back Together at Xyratex". Inside HPC. http://insidehpc.com/2010/11/09/bojanic-braam-getting-lustre-band-back-together-at-xyratex/.
10. Rich Brueckner (January 4, 2011). "Whamcloud Staffs up for Brighter Lustre". Inside HPC. http://insidehpc.com/2011/01/04/whamcloud-staffs-up-for-brighter-lustre/.
11. "Whamcloud Signs Multi-Year Lustre Development Contract With OpenSFS". Press release (HPC Wire). August 16, 2011. http://www.hpcwire.com/hpcwire/2011-08-16/whamcloud_signs_multi-year_lustre_development_contract_with_opensfs.html.
12. Brian Behlendorf. "ZFS on Linux for Lustre". Lawrence Livermore National Laboratory. http://wiki.lustre.org/images/b/b0/LUG-2011-Brian_Benhlendorf-ZFS_on_Linux.pdf.
13. Galen Shipman (November 18, 2011). "OpenSFS Update". Slides for Supercomputing 2011 presentation. Open Scalable File Systems. http://www.opensfs.org/wp-content/uploads/2011/11/SC11-OpenSFS-Update.pdf.
14. Whamcloud (November 15, 2011). "OpenSFS and Whamcloud Sign Lustre Community Tree Development Agreement". Press release. https://www.reuters.com/article/2011/11/15/idUS215861+15-Nov-2011+MW20111115.
15. Joab Jackson (2012-07-16). "Intel Purchases Lustre Purveyor Whamcloud". PC World. http://www.pcworld.com/article/259328/intel_purchases_lustre_purveyor_whamcloud.html.
16. Timothy Prickett Morgan (2012-07-16). "Intel gobbles Lustre file system expert Whamcloud". The Register. https://www.theregister.co.uk/2012/07/16/intel_buys_whamcloud/.
17. Timothy Prickett Morgan (2012-07-11). "DOE doles out cash to AMD, Whamcloud for exascale research". The Register. https://www.theregister.co.uk/2012/07/11/doe_fastforward_amd_whamcloud/.
18. Nicole Hemsoth (June 12, 2013). "Intel Carves Mainstream Highway for Lustre". HPC Wire. http://www.hpcwire.com/2013/06/12/intel_builds_mainstream_highways_for_lustre/.

19. Brueckner, Rich (21 February 2013). "With New RFP, OpenSFS to Invest in Critical Open Source Technologies for HPC". insideHPC. http://insidehpc.com/2013/02/21/with-new-rfp-opensfs-to-invest-in-critical-open-source-technologies-for-hpc/.
20. "Seagate Donates Lustre.org Back to the User Community". http://insidehpc.com/2014/04/seagate-donates-lustre-org-user-community/.
21. Daniel Robinson (June 27, 2018). "DDN Breathes New Life Into Lustre File System". https://www.nextplatform.com/2018/06/27/ddn-breathes-new-life-into-lustre-file-system/.
22. "Lustre Trademark Released to User Community". InsideHPC. November 24, 2019. https://insidehpc.com/2019/11/lustre-trademark-released-to-user-community/.
23. "Lustre Helps Power Third Fastest Supercomputer". DSStar. http://www.taborcommunications.com/dsstar/03/1125/107031.html.
24. "MCR Linux Cluster Xeon 2.4 GHz – Quadrics". Top500.Org. http://www.top500.org/system/6085.
25. Corbet, Jonathon (December 17, 2003). "Lustre 1.0 released". LWN.net. https://lwn.net/Articles/63536/.
26. Peter Bojanic (June 15, 2008). "Lustre Roadmap and Future Plans". Presentation to Sun HPC Consortium. Sun Microsystems. http://www.hpcuserforum.com/presentations/Tucson/SUN%20%20Lustre_Update-080615.pdf.
27. "OpenSFS Announces Collaborative Effort to Support Lustre 2.1 Community Distribution". Open Scalable File Systems. February 8, 2011. http://www.whamcloud.com/news-and-events/opensfs-announces-collaborative-effort-to-support-lustre-2-1-community-distribution/.
28. "Lustre 2.1 Released". http://www.marketwire.com/press-release/lustre-21-released-1567596.htm.
29. "Lustre 2.2 Released". Yahoo! Finance. https://finance.yahoo.com/news/lustre-2-2-released-165800113.html.
30. "A Novel Network Request Scheduler for a Large Scale Storage System". OpenSFS. June 2009. http://wiki.lustre.org/images/2/22/A_Novel_Network_Request_Scheduler_for_a_Large_Scale_Storage_System.pdf.
31. "A Novel Network Request Scheduler for a Large Scale Storage System". OpenSFS. June 2009. https://www.researchgate.net/publication/220232795.
32. Prickett Morgan, Timothy (5 November 2013). "OpenSFS Announces Availability of Lustre 2.5". EnterpriseTech. http://www.enterprisetech.com/2013/11/05/opensfs-announces-availability-lustre-2-5/.
33. Brueckner, Rich. "Video: New Lustre 2.5 Release Offers HSM Capabilities". Inside Big Data. http://inside-bigdata.com/2013/11/05/video-new-lustre-2-5-release-offers-hsm-capabilities/.
34. Hemsoth, Nicole. "Lustre Gets Business Class Upgrade with HSM". HPCwire. http://archive.hpcwire.com/hpcwire/2013-11-06/lustre_scores_business_class_upgrade_with_hsm.html.
35. "Lustre 2.5". Scientific Computing World. 12 November 2013. http://www.scientific-computing.com/products/product_details.php?product_id=1727.
36. Jones, Peter (September 9, 2014). "Lustre 2.5.3 released". https://lists.01.org/pipermail/hpdd-discuss/2014-September/001211.html. Morrone, Chris (Dec 7, 2015). "Retired Release Terminology". http://wiki.lustre.org/Retired_Release_Terminology.
37. "Lustre 2.6.0 released". July 30, 2014. https://lists.01.org/pipermail/hpdd-discuss/2014-July/001153.html.
38. Ihara, Shuichi (2014-10-14). "Lustre QoS Based on NRS Policy of Token Bucket Filter". http://cdn.opensfs.org/wp-content/uploads/2014/10/7-DDN_LiXi_lustre_QoS.pdf.
39. Uselton, Andrew. "Demonstrating the Improvement in the Performance of a Single Lustre Client from Version 1.8 to Version 2.6". http://opensfs.org/wp-content/uploads/2014/04/D1_S6_LustreClientIOPerformanceImprovements.pdf.
40. Jones, Peter (March 13, 2015). "Lustre 2.7.0 released". https://lists.01.org/pipermail/hpdd-discuss/2015-March/001829.html.
41. Jones, Peter (March 16, 2016). "Lustre 2.8.0 released". OpenSFS. http://lists.lustre.org/pipermail/lustre-announce-lustre.org/2016/000137.html.
42. "Lustre 2.9.0 Changelog". OpenSFS. December 7, 2016. http://wiki.lustre.org/Lustre_2.9.0_Changelog.
43. "Lustre 2.10.0 Changelog". OpenSFS. July 13, 2017. http://wiki.lustre.org/Lustre_2.10.0_Changelog.
44. "Release 2.11.0". OpenSFS. April 3, 2018. http://wiki.lustre.org/Release_2.11.0.
45. "Release 2.12.0". OpenSFS. December 21, 2018. http://wiki.lustre.org/Release_2.12.0.

46. Li Xi, DDN (June 2018). "Lazy Size on MDS". Lustre Wiki. http://wiki.lustre.org/images/4/4f/LUG2018-Lazy_Size_on_MDS-Xi.pdf.
47. Shuichi Ihara, DDN (June 2018). "T10PI End-to-End Data Integrity Protection for Lustre". Lustre Wiki. http://wiki.lustre.org/images/d/d1/LUG2018-T10PI_EndToEnd_Data_Integrity-Ihara.pdf.
48. "Release 2.13.0". OpenSFS. December 5, 2019. http://wiki.lustre.org/Release_2.13.0.
49. Li Xi, Whamcloud (June 2018). "Lustre Persistent Client Cache". http://wiki.lustre.org/images/0/04/LUG2018-Lustre_Persistent_Client_Cache-Xi.pdf.
50. Patrick Farrell, Whamcloud (April 2019). "Overstriping: Extracting Maximum Shared File Performance". http://wiki.lustre.org/images/b/b3/LUG2019-Lustre_Overstriping_Shared_Write_Performance-Farrell.pdf.
51. Patrick Farrell, Cray (March 15, 2019). "Spillover Space: Self-Extending Layouts HLD". https://jira.whamcloud.com/secure/attachment/30498/LUSTRE-98540817-120718-1412-90.pdf.
52. "Lustre 2.13.0 Changelog". December 5, 2019. http://wiki.lustre.org/Lustre_2.13.0_Changelog.
53. "Release 2.14.0". OpenSFS. February 19, 2021. http://wiki.lustre.org/Release_2.14.0.
54. "Release 2.15.0". OpenSFS. June 16, 2022. http://wiki.lustre.org/Release_2.15.0.
55. Sébastien Buisson (May 11, 2022). "Client Data Encryption". OpenSFS. https://wiki.lustre.org/images/0/01/LUG2022-Lustre_Client_Encryption-Buisson.pdf.
56. "NVIDIA Magnum IO GPUDirect Storage Overview Guide". NVIDIA. https://docs.nvidia.com/gpudirect-storage/overview-guide/index.html.
57. "Lustre to run on ZFS". Government Computer News. 2008-10-26. http://gcn.com/Articles/2008/03/26/Lustre-to-run-on-ZFS.aspx?p=1.
58. "ZFS on Lustre". 2011-05-10. http://zfsonlinux.org/lustre.html.
59. "DataDirect Selected As Storage Tech Powering BlueGene/L". HPC Wire. October 15, 2004. http://www.tgc.com/hpcwire/hpcwireWWW/04/1015/108577.html.
60. Suzanne M. Kelly (2006). "Catamount Software Architecture with Dual Core Extensions". http://www.sandia.gov/~smkelly/SAND2006-2561C-CUG2006-CatamountDualCore.pdf.
61. "Linux Kernel 4.18rc1 release notes". http://lkml.iu.edu/hypermail/linux/kernel/1806.2/00125.html.
62. Shehata, Amir. "Multi-Rail LNet for Lustre". Lustre User Group, April 2016. http://wiki.lustre.org/images/7/7c/LUG2016D2_Multi-Rail-LNet-for-Lustre_Shehata_Weber_v2.pdf.
63. Lafoucrière, Jacques-Charles. "Lustre Experience at CEA/DIF". HEPiX Forum, April 2007. http://hepix.caspur.it/afs/hepix.org/project/strack/hep_pdf/2007/Spring/Lustre-CEA-hepix2007.pdf.
64. Caldwell, Blane (March 9, 2016). "Lustre Networking Technologies: Ethernet vs. Infiniband". https://lustre.ornl.gov/ecosystem-2016/documents/topics/Caldwell-ORNL-EthernetVsInfiniband.pdf.
65. Aurélien Degrémont (September 17, 2013). "LUSTRE/HSM BINDING IS THERE!". https://www.eofs.eu/_media/events/lad13/10_aurelien_degremont_lustre_hsm_lad13.pdf.
66. Thomas Stibor (September 20, 2016). "TSM Copytool for Lustre HSM". https://www.eofs.eu/_media/events/lad16/08_tsm_copytool_for_lustre_stibor.pdf.
67. Robert Read (March 24, 2015). "Lustre HSM in the Cloud". http://wiki.lustre.org/images/e/ea/Lustre-HSM-in-the-Cloud_Read.pdf.
68. Stéphane Thiell (11 April 2021). "Lustre/HSM Google Drive copytool". https://github.com/stanford-rc/ct_gdrive/blob/master/README.md.
69. "Lustre HSM Project—Lustre User Advanced Seminars". April 16, 2009. http://wiki.lustre.org/images/4/4d/Lustre_hsm_seminar_lug10.pdf.
70. "Frontier supercomputer debuts as world's fastest, breaking exascale barrier". Oak Ridge National Laboratory. May 30, 2022. https://www.ornl.gov/news/frontier-supercomputer-debuts-worlds-fastest-breaking-exascale-barrier.
71. "LNCC – Laboratório Nacional de Computação Científica". Lncc.br. http://www.lncc.br/frame.html.
72. "Pleiades Supercomputer". www.nas.nasa.gov. 2008-08-18. http://www.nas.nasa.gov/Resources/Systems/pleiades.html.
73. "TOP500 List – November 2006". TOP500.Org. http://www.top500.org/system/8216.
74. "TOP500 List – June 2006". TOP500.Org. http://www.top500.org/system/8237.

75. "French Atomic Energy Group Expands HPC File System to 11 Petabytes". HPCwire.com. 2012-06-15. http://www.hpcwire.com/hpcwire/2012-01-25/french_atomic_energy_group_expands_hpc_file_system_to_11_petabytes.html.
76. "HP StorageWorks Scalable File Share". Hewlett-Packard. http://h20311.www2.hp.com/HPC/cache/276636-0-0-0-121.html.
77. "Fujitsu Releases World's Highest-Performance File System – FEFS scalable file system software for advanced x86 HPC cluster systems". 2015-06-13. http://www.fujitsu.com/global/about/resources/news/press-releases/2011/1017-01.html.
78. "High Throughput Storage Solutions with Lustre". 2015-04-14. http://www.hds.com/solutions/industries/oil-and-gas/exploration-and-production/high-throughput-storage-solutions.html.
79. "Exascaler: Massively Scalable, High Performance, Lustre File System Appliance". 2015-04-14. http://www.ddn.com/products/lustre-file-system-exascaler/.
80. "Lustre Support". 2018-11-27. http://whamcloud.com/support/.
81. "Amazon FSx for Lustre". 2019-06-11. https://aws.amazon.com/fsx/lustre/.
