.l.u.s.t.r.e.

Lustre: A Scalable, High-Performance File System

Cluster File Systems, Inc.

Abstract:

Today's network-oriented computing environments require high-performance, network-aware file systems that can satisfy both the data storage requirements of individual systems and the data sharing requirements of workgroups and clusters of cooperative systems. The Lustre File System, an open source, high-performance file system from Cluster File Systems, Inc., is a distributed file system that eliminates the performance, availability, and scalability problems that are present in many traditional distributed file systems. Lustre is a highly modular next generation storage architecture that combines established, open standards, the Linux operating system, and innovative protocols into a reliable, network-neutral data storage and retrieval solution. Lustre provides high I/O throughput in clusters and shared-data environments and also provides independence from the location of data on the physical storage, protection from single points of failure, and fast recovery from cluster reconfiguration and server or network outages.

1. Overview

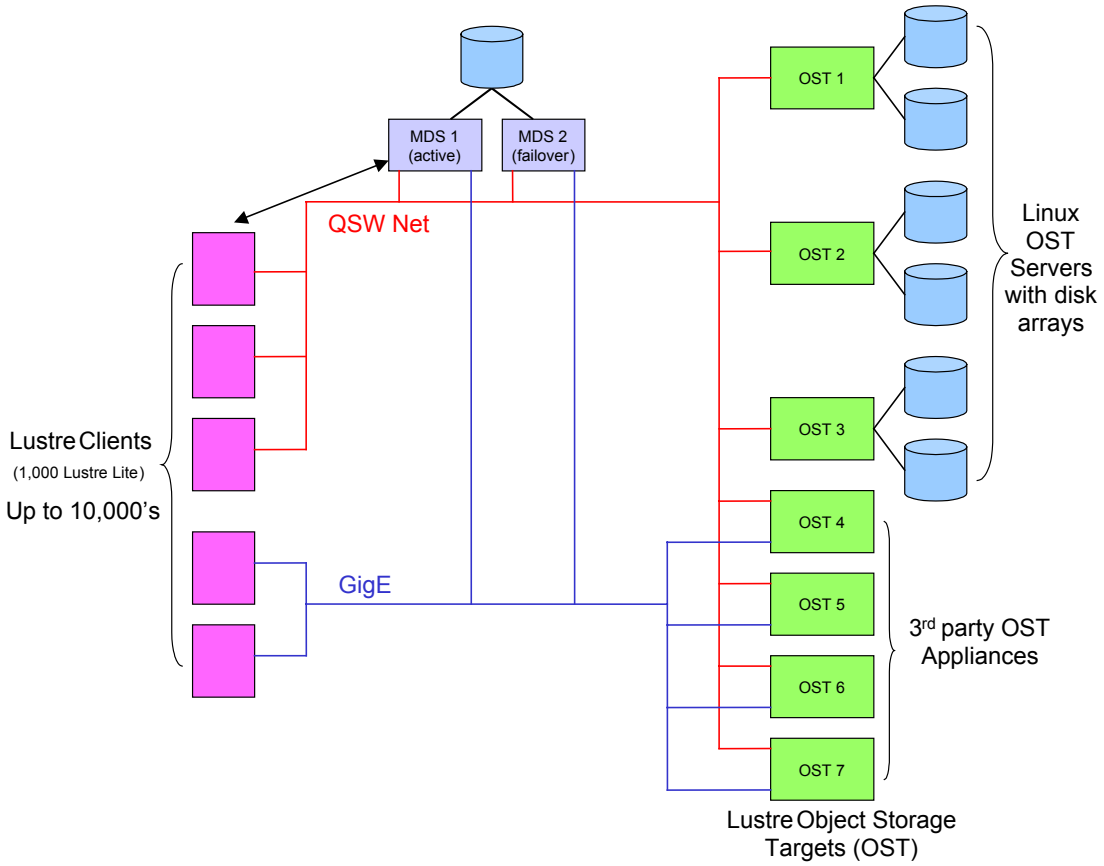
Network-centric computing environments demand reliable, high-performance storage systems that properly authenticated clients can depend on for data storage and delivery. Simple cooperative computing environments such as enterprise networks typically satisfy these requirements using distributed file systems based on a standard client/server model. Distributed file systems such as NFS and AFS have been successful in a variety of enterprise scenarios but do not satisfy the requirements of today's high-performance computing environments. The Lustre distributed file system provides significant performance and scalability advantages over existing distributed file systems. Lustre leverages the power and flexibility of the Open Source Linux operating system to provide a truly modern POSIX compliant file system that satisfies the requirements of large clusters today, while providing a clear design and extension path for even larger environments tomorrow. The name "Lustre" is an amalgam of the terms "Linux" and "Clusters".

Distributed file systems have well-known advantages. They decouple computational and storage resources, enabling desktop systems to focus on user and application requests while file servers focus on reading, delivering, and writing data. Centralizing storage on file servers facilitates centralized system administration, simplifying operational tasks such as backups, storage expansion, and general storage reconfiguration without requiring desktop downtime or other interruptions in service.

Beyond the standard features required by definition in a distributed file system, a more advanced distributed file system such as AFS simplifies data access and usability by providing a consistent view of the distributed file system from all client systems. It also supports redundancy, which means that failover services in conjunction with redundant storage devices provide multiple, synchronized copies of critical resources eliminating single points of failure. In the event of the failure of any critical resource, the file system automatically provides a replica of the failed entity that can therefore provide uninterrupted service. This eliminates single points of failure in the distributed file system environment.

Lustre provides significant advantages over the aging distributed file systems that preceded it. These advantages will be discussed in more detail throughout this paper, but are highlighted here for convenience. Most importantly, Lustre runs on commodity hardware and uses object based disks for storage and metadata servers for storing file system metadata. This design provides a substantially more efficient division of labor between computing and storage resources. Replicated, failover metadata Servers (MDSs) maintain a transactional record of high-level file and file system changes. Distributed Object Storage Targets (OSTs) are responsible for actual file system I/O and for interfacing with storage devices, which will be explained in more detail in the next section. This division of labor and responsibility leads to a truly scalable file system andd more reliable recoverability from failure conditions by providing a unique combination of the advantages of journaling and distributed file systems. Lustre supports strong file and metadata locking semantics to maintain total coherency of the file systems even in the presence of concurrent access. File locking is distributed across the storage targets (OSTs) that constitute the file system, with each OST handling locks for the objects that it stores.

Figure 1: Lustre Big Picture



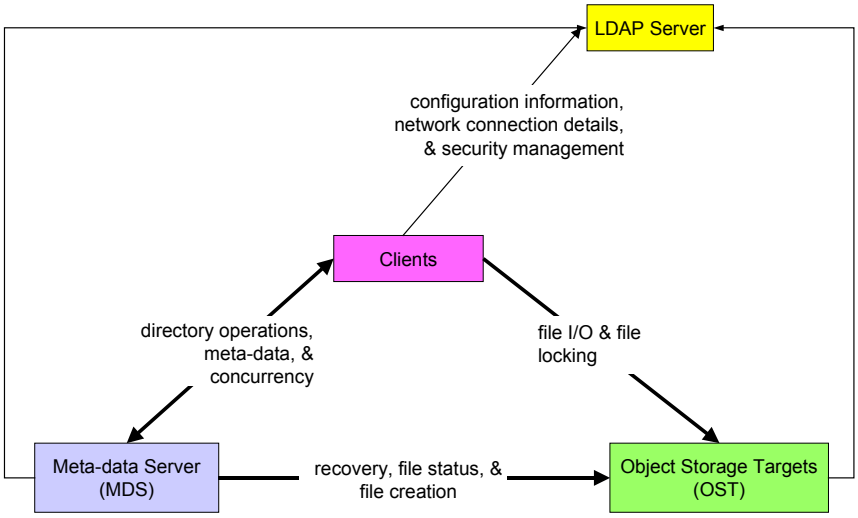
Lustre uses an open networking API, the Portals API, made available by Sandia. At the top of the stack is a very sophisticated request processing layer provided by Lustre, resting on top of the Portals protocol stack. At the bottom is a network abstraction layer (NAL) that provides out-of-the-box support for multiple types of networks. Like Lustre's use of Linux, Lustre's use of open, flexible standards makes it easy to integrate new and emerging network and storage technologies. Lustre provides security in the form of authentication, authorization and privacy by leveraging existing security systems. This makes it easy to incorporate Lustre into existing enterprise security environments without requiring changes in Lustre itself. Similarly, Lustre leverages the underlying journaling file systems provided by Linux to enable persistent state recovery, enabling resiliency and recoverability from failed OSTs. Finally, Lustre's configuration and state information is recorded and managed using open standards such as XML and LDAP, making it easy to integrate Lustre management and administration into existing environments and sets of third-party tools.

The remainder of this white paper provides a more detailed analysis of various aspects of the design and implementation of Lustre along with a roadmap for planned Lustre enhancements. The Lustre web site at http://www.lustre.org provides additional documentation on Lustre along with the source code. For more information about Lustre, contact Cluster File Systems, Inc. via email at info@clusterfs.com.

2. Lustre Functionality

The Lustre file system provides several abstractions designed to improve both performance and scalability. At the file system level, Lustre treats files as objects that are located through metadata Servers (MDSs). Metadata Servers support all file system namespace operations, such as file lookups, file creation, and file and directory attribute manipulation, directing actual file I/O requests to Object Storage Targets (OSTs), which manage the storage that is physically located on underlying Object-Based Disks (OBDs). Metadata servers keep a transactional record of file system metadata changes and cluster status, and support failover so that the hardware and network outages that affect one metadata Server do not affect the operation of the file system itself.

Figure 2: Interactions Between Lustre Subsystems



Like other file systems, the Lustre file system has a unique inode for every regular file, directory, symbolic link, and special file. The regular file inodes hold references to objects on OSTs that store the file data instead of references to the actual file data itself. In existing file systems, creating a new file causes the file system to allocate an inode and set some of its basic attributes. In Lustre, creating a new file causes the client to contact a metadata server, which creates an inode for the file and then contacts the OSTs to create objects that will actually hold file data. Metadata for the objects is held in the inode as extended attributes for the file. The objects allocated on OSTs hold the data associated with the file and can be striped across several OSTs in a RAID pattern. Within the OST, data is actually read and written to underlying storage known as Object-Based Disks (OBDs). Subsequent I/O to the newly created file is done directly between the client and the OST, which interacts with the underlying OBDs to read and write data. The metadata server is only updated when additional namespace changes associated with the new file are required.

Object Storage Targets handle all of the interaction between client data requests and the underlying physical storage. This storage is generally referred to as Object-Based Disks (OBDs), but is not actually limited to disks because the interaction between the OST and the actual storage device is done through a device driver. The characteristics and capabilities of the device driver mask the specific identity of the underlying storage that is being used. This enables Lustre to leverage existing Linux file systems and storage devices for its underlying storage while providing the flexibility required to integrate new technologies such as smart disks and new types of file systems. For example, Lustre currently provides OBD device drivers that support Lustre data storage within journaling Linux file systems such as ext3, JFS, ReiserFS and XFS. This further increases the reliability and recoverability of Lustre by leveraging the journaling mechanisms already provided in such file systems. Lustre can also be used with specialized 3rd party object storage targets like those provided by BlueArc.

Lustre's division of actual storage and allocation into OSTs and underlying OBDs facilitates hardware development that can provide additional performance improvements in the guise of a new generation of smart disk drives which provide object-oriented allocation and data management facilities in hardware. Cluster File Systems is actively working with several storage manufacturers to develop integrated OBD support in disk drive hardware. This is analogous to the way in which the SCSI standard pioneered smart devices that offloaded much of the direct hardware interaction into the device's interface and drive controller hardware. Such smart, OBD-aware hardware can provide instant performance improvements to existing Lustre installations and will continue the modern computing trend of offloading device-specific processing to the device itself.

Beyond the storage abstraction that they provide, OSTs also provide a flexible model for adding new storage to an existing Lustre file system. New OSTs can easily be brought online and added to the pool of OSTs that a cluster's metadata servers can use for storage. Similarly, new OBDs can easily be added to the pool of underlying storage associated with any OST. Lustre provides a powerful and unique recovery mechanism used when any communication or storage failure occurs. If a server or network interconnect fails, the client incurs a timeout trying

to access data. It can then query a LDAP server to obtain information about a replacement server and immediately direct subsequent requests to that server. An 'epoch' number on every storage controller, an incarnation number on the metadata server/cluster, and a generation number associated with connections between clients and other systems form the infrastructure for Lustre recovery, enabling clients and servers to detect restarts and select appropriate and equivalent servers.

When failover OSTs are not available Lustre will automatically adapt. If an OST fails - except for raising administrative alarms - it will only generate errors when data cannot be accessed. New file creation operations will automatically avoid a malfunctioning OST.

Figure 3: Lustre Client Software Modules

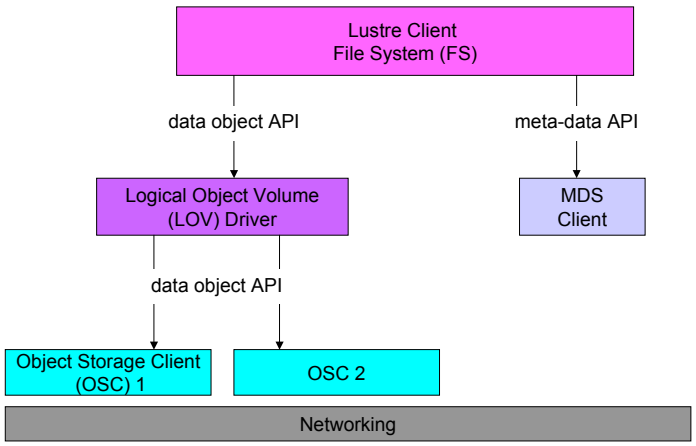


Figure 4 a,b: Lustre Automatically Avoids Malfunctioning OST

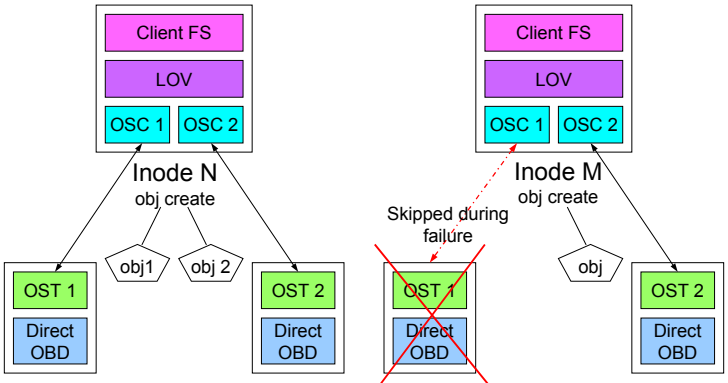
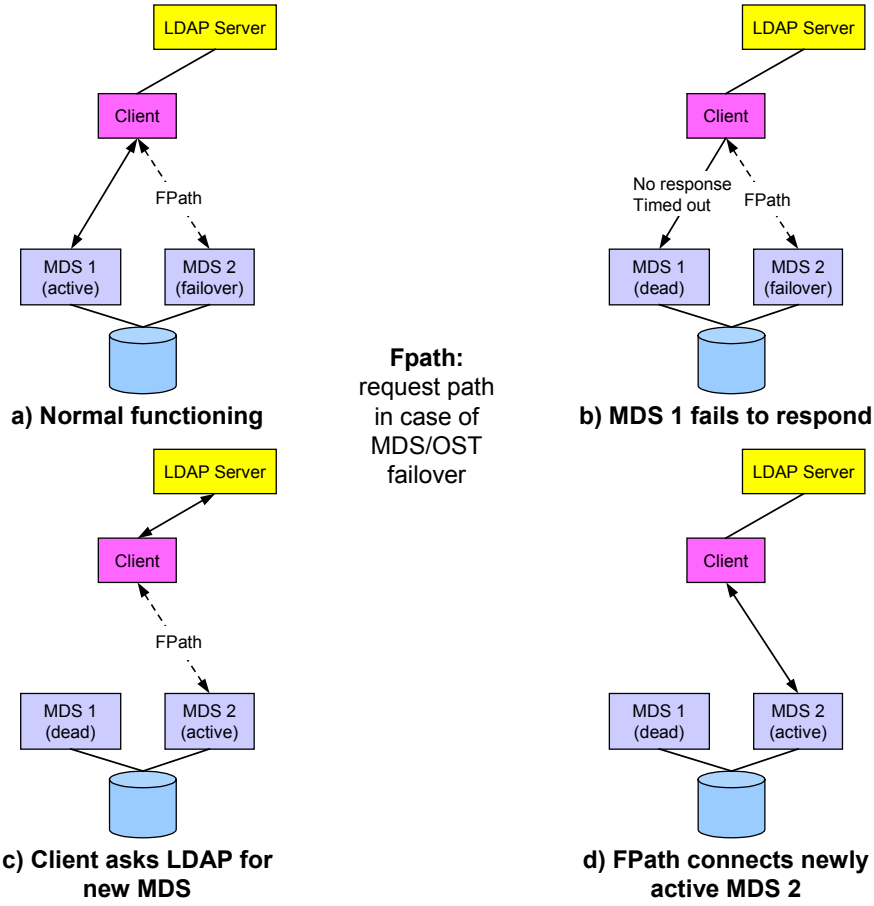


Figure 5 a-d: Lustre Failover Mechanism



3. File System Metadata and Metadata Servers

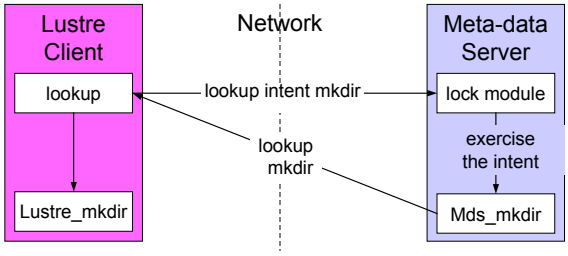
File system metadata is "information about information", which essentially means that metadata is information about the files and directories that make up a file system. This information can simply be information about local files, directories, and associated status information, but can also be information about mount points for other file systems within the current file system, information about symbolic links, and so on. Many modern file systems use metadata journaling to maximize file system consistency. The file system keeps a journal of all changes to file system metadata, and asynchronously updates the file system based on completed changes that have been written to its journal. If a system outage occurs, file system consistency can be quickly restored simply by replaying completed transactions from the metadata journal.

In Lustre, file system metadata is stored on a metadata server (MDS) and file data is stored in objects on the OSTs. This design divides file system updates into two distinct types of operations: file system metadata updates on the MDS and actual file data updates on the OSTs. File System namespace operations are done on the MDS so that they do not impact the performance of operations that only manipulate actual object (file) data. Once the MDS identifies the storage location of a file, all subsequent file I/O is done between the client and the OSTs. Using metadata servers to manage the file system namespace provides a variety of immediate opportunities for performance optimization. For example, metadata servers can maintain a cache of pre-allocated objects on various OSTs, expediting file creation operations. The scalability of metadata operations on Lustre is further improved through the use of an intent based locking scheme. For example, when a client wishes to create a file, it requests a lock from an MDS to enable a lookup operation on the parent directory, and also tags this request with the intended operation, namely file creation. If the lock request is granted, the MDS then uses the intention specified in the lock request to modify the directory, creating the requested file and returning a lock on the new file instead of the directory.

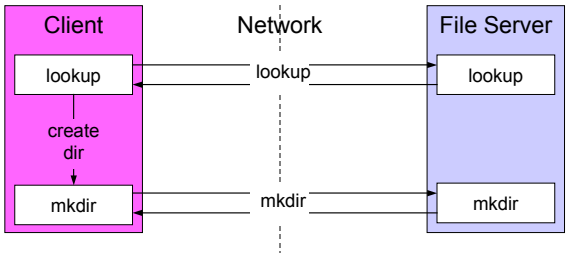
Divorcing file system metadata operations from actual file data operations improves immediate performance, but also improves long-term aspects of the file system such as recoverability and availability. Actual file I/O is done directly between Object Storage Targets and client systems, eliminating intermediaries. General file system availability is improved by providing a single failover metadata server and by using distributed Object Storage Targets, eliminating any one MDS or OST as a single point of failure. In the event of wide-spread hardware or network outages, the transactional nature of the metadata stored on the metadata servers significantly reduces the time it takes to restore file system consistency by minimizing the chance of losing file system control information such as object storage locations and actual object attribute information.

File System availability and reliability are critical to any computer system, but become even more significant when the number of clients and the amount of managed storage increases. Local file system outages only affect the usability of a single workstation, but central resource outages such as a distributed file system have the potential to affect the usability of hundreds or thousands of client systems that need to access that storage. Lustre's flexibility, reliable and highly-available design, and inherent scalability make Lustre well-suited for use as a cluster file system today, when cluster clients number in the hundreds or low thousands, and tomorrow, when the number of clients depending on distributed file system resources will only continue to grow.

Figure 6: Lookup Intents



a) Lustre mkdir



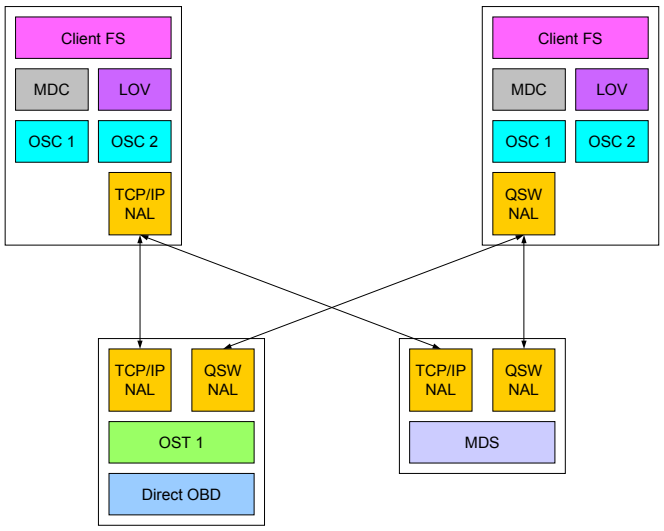
b) Conventional mkdir

4. Network Independence in Lustre

As mentioned earlier in this paper, Lustre can be used over a wide variety of networks due to its use of an open Network Abstraction Layer. Lustre is currently in use over TCP and Quadrics (QSWNet) networks. Myrinet, Fibre Channel, Stargen and InfiniBand support are under development. Lustre's network-neutrality enables Lustre to instantly take advantage of performance improvements provided by network hardware and protocol improvements offered by new systems.

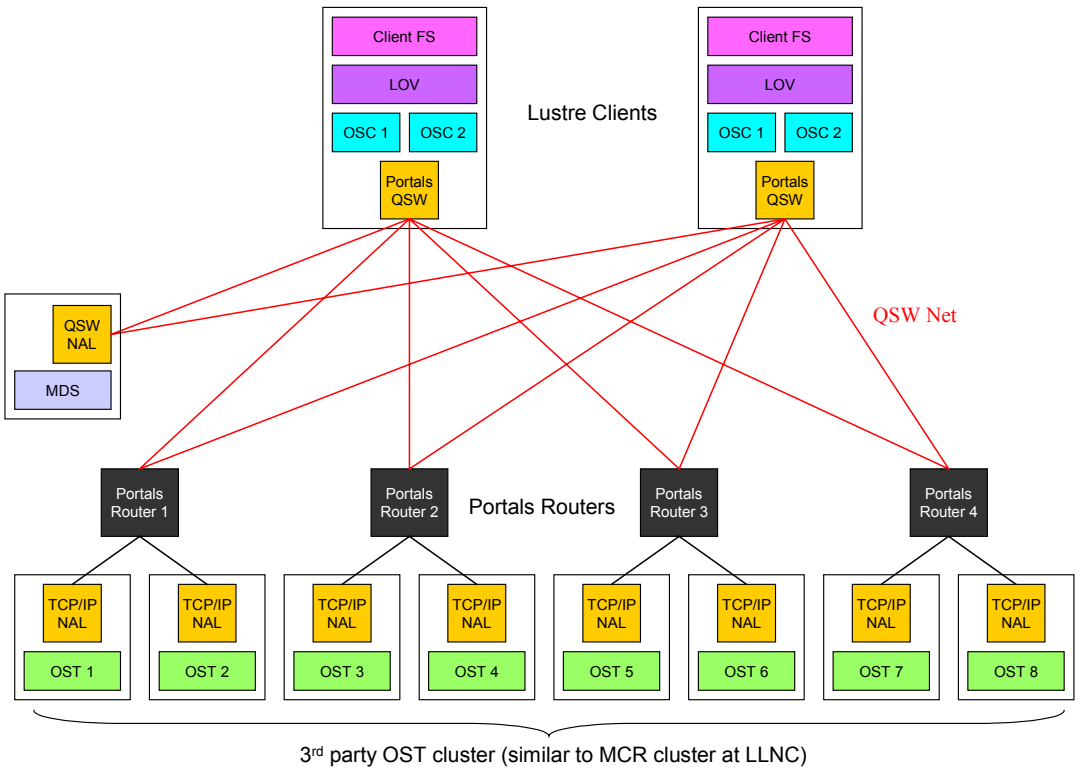
Lustre provides unique support for heterogeneous networks. For example, it is possible to connect some clients over an Ethernet to the MDS and OST servers, and others over a QSW network, all in a single installation.

Figure 7: Support for Heterogeneous Networks



Lustre also provides routers that can route the Lustre protocol between different kinds of supported networks. This is useful to connect to 3rd party OSTs that may not support all specialized networks available on generic hardware.

Figure 8: Lustre Network Routing



Lustre provides a very sophisticated request processing layer on top of the Portals protocol stack, originally developed by Sandia but now available to the Open Source community. Below this is the network data movement layer responsible for moving data vectors from one system to another. Beneath this layer, the Portals message passing layer sits on top of the network abstraction layer, which finally defines the interactions between underlying network devices. The Portals stack provides support for high-performance network data transfer, such as Remote

Direct Memory Access (RDMA), direct OS-bypass I/O, and scatter gather I/O (memory and disk) for more efficient bulk data movement.

5. Lustre Administration Overview

Lustre's commitment to using open standards such as Linux, the Portals Network Abstraction Layer, and existing Linux journaling file systems such as ext3 for journaling metadata storage is reflected in its commitment to creating open administrative and configuration information. Lustre's configuration information is stored in eXtensible Markup Language (XML) files that conform to a simple Document Type Definition (DTD), which is published in the open Lustre documentation. Maintaining configuration information in standard text files means that it can easily be manipulated using simple tools such as text editors. Maintaining it in a consistent fashion with a published DTD makes it easy to integrate Lustre configuration into third-party and open source administration utilities.

These configuration files can be generated and updated using the lmc (Lustre make configuration) configuration utility. The lmc utility quickly generates initial configuration files, even for very large clusters complex clusters involving 100's of OSTs, routers and clients.

Lustre is integrated with open network data resources and administrative mechanisms such as the Light-Weight Directory Access Protocol (LDAP) and the Simple Network Management Protocol (SNMP). The LMC utility can convert LDAP based configuration to and from XML based configuration information. The LDAP infrastructure provides redundancy and assists with cluster recovery.

To provide enterprise wide monitoring, Lustre exports status and configuration information into the SNMP agent, offering a Lustre MIB to the management stations.

Lustre provides several basic, command-line oriented utilities for initial configuration and administration. The lctl (Lustre control) utility can be used to perform low level Lustre network and device configuration tasks, as well as batch driven tests to check the sanity of a cluster.

The lconf (Lustre configuration) utility enables administrators to configure Lustre on specific nodes using user-specified configuration files. The Lustre documentation provides extensive examples of using these commands to configure, start, reconfigure, and stop or restart Lustre services.

6. Future Directions for Lustre

Lustre's distributed design and use of metadata servers and Object Storage Targets as intermediaries between client requests and actual data access leads to a very scalable file system. The next few sections highlight several issues on the Lustre roadmap that will help to further improve the performance, scalability, and security of Lustre.

6.1 The Lustre Global Namespace

As mentioned earlier in this paper, distributed file systems provide a number of administrative advantages. From the end-user perspective, the primary advantages of distributed file systems are that they provide access to substantially more storage than could be physically attached to a single system, and that this storage can be accessed from any authorized workstation. However, accessing shared storage from different systems can be confusing if there is no uniform way of referring to and accessing that storage.

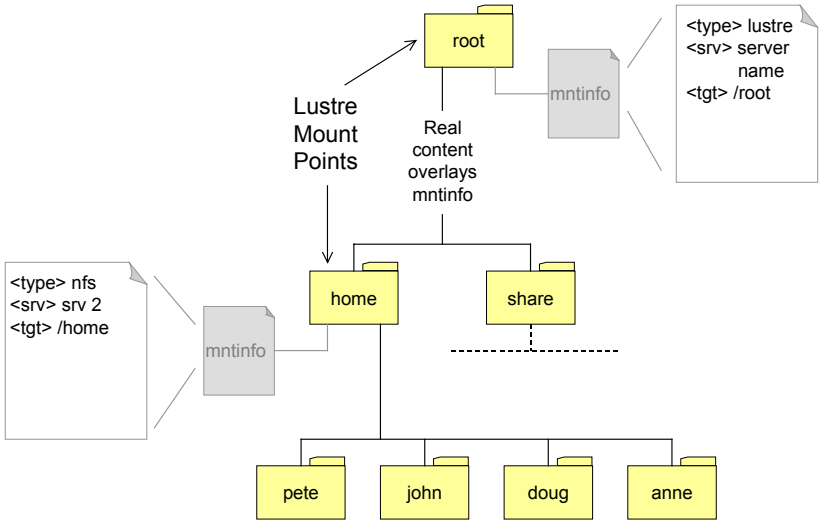
The classic way of providing a single way of referring to and accessing the files in a distributed filesystem is by providing a "global namespace". A global namespace is typically a single directory on which an entire distributed filesystem is made available to users. This is known as mounting the distributed filesystem on that directory. In the AFS distributed file system, the global namespace is the directory /afs, which provides hierarchical access to filesets on various servers that are mounted as subdirectories somewhere under the /afs directory. When traversing fileset mountpoints, AFS does not store configuration data on the client to find the target fileset, but instead contacts a fileset location server to determine the server on which the fileset is physically stored. In AFS,

mountpoint objects are represented as symbolic links that point to a fileset name/identifier. This requires that AFS mount objects must be translated from symbolic links to specific directories and filesets whenever you mount a fileset. Unfortunately, existing file systems like AFS contain hardwired references to mountpoints for the file systems. These file systems must therefore always be found at those locations, and can only be found at those locations.

Unlike existing distributed filesystems, Lustre intends to provide the best of both worlds by providing a global namespace that can be easily grafted onto any directory in an existing Linux filesystem. Once a Lustre filesystem is mounted, any authenticated client can access files within it using the same path and filename, but the initial mount point for the Lustre filesystem is not pre-defined and need not be the same on every client system. If desired, a uniform mountpoint for the Lustre filesystem can be enforced administratively by simply mounting the Lustre filesystem on the same directory on every client, but this is not mandatory.

Lustre intends to simplify mounting remote storage by setting special bits on directories that are to be used as mount points, and then storing the mount information in a special file in each such directory. This is completely compatible with every existing Linux file system, eliminates the extra overhead required in obtaining the mount information from a symbolic link, and makes it possible to identify mountpoints without actually traversing them unless you actually need information about the remote file system.

Figure 9: Lustre Global Namespace Filter



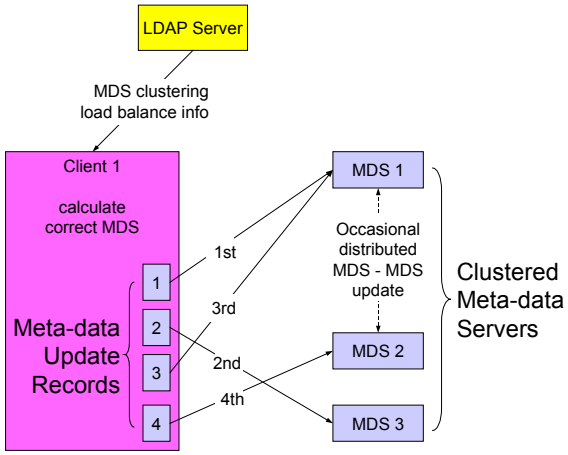
An easily overlooked benefit of the Lustre mount mechanism is that it provides greater flexibility than existing Linux mount mechanisms. Standard Linux client systems use the file /etc/fstab to maintain information about all of the file systems that should be mounted on that client. The Lustre mount mechanism transfers the responsibility for maintaining mount information from a single, per-client file into the client file system. The Lustre mount mechanism also makes it easy to mount other file systems within a Lustre file system, without requiring that each client be aware of all file systems mounted within Lustre. Lustre is therefore not only a powerful distributed file system in its own right, but also serves as a powerful integration mechanism for other existing distributed file systems.

6.2. Metadata and File I/O Performance Improvements

One of the first optimizations to be introduced is the use of a writeback cache for file writes to provide higher overall performance. Currently, Lustre writes are write-though, which means that write requests are not complete until the data is actually flushed to the target OSTs. In busy clusters, this can impose a considerable delay on every file write operation. The use of a writeback cache, where file writes are journaled and committed asynchronously, provides a great deal of promise for substantially higher-performance in file write requests.

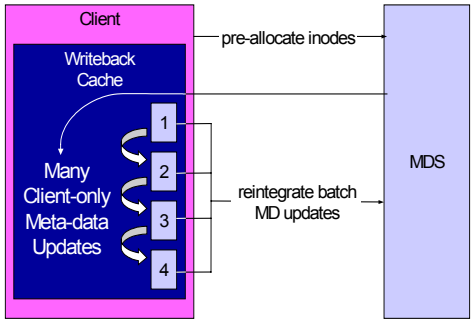
The Lustre file system currently stores and updates all file metadata (except allocation data, which is held on the OST) through a single (failover) metadata server. While simple, accurate, and already very scalable, depending upon a single metadata server can reduce the performance of metadata operations in Lustre. Metadata performance can be greatly improved by implementing clustered metadata servers. Distributing metadata information across the cluster will also result in distributing the metadata processing load across the cluster, improving the overall throughput of metadata operations.

Figure 10: Meta-data Clustering



A writeback cache for Lustre metadata servers is also being considered. If a writeback cache for metadata is present, metadata updates would be first written to this cache and would subsequently be flushed to persistent storage on the metadata servers at a later time. This will dramatically improve the latency of updates as seen by client systems. It also enables batch metadata updates, which could reduce communications and increase parallelism.

Figure 11: Lustre Meta-data Writeback Cache

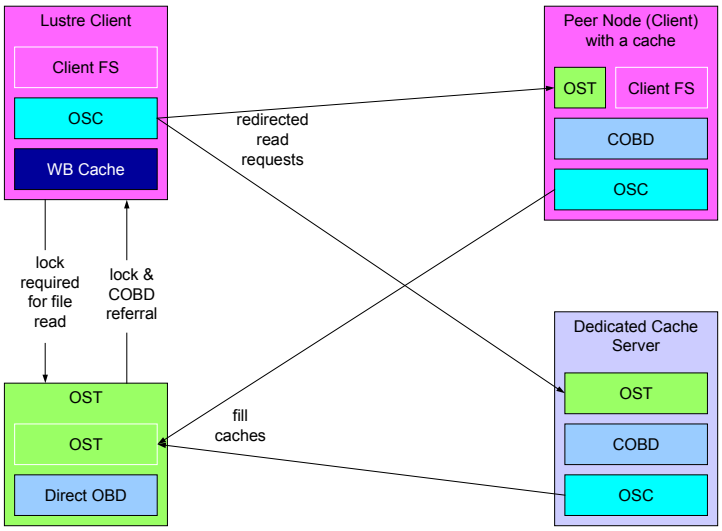


Read scalability is achieved in a very different manner. Good examples of potential bottlenecks in a clustered environment are system files or centralized binaries that are stored in Lustre but which are required by multiple clients throughout the cluster at boot-time. If multiple clients were rebooted at the same time, they would all need to access the same system files simultaneously. However, if every client has to read the data from the same server, the network load at that server could be extremely high and the available network bandwidth at that server could pose a serious bottleneck for general file system performance. Using a collaborative cache, where frequently requested files could be cached across multiple servers, would help distribute the read load across multiple nodes, reducing the bandwidth requirements at each.

A second example arises in situations like video servers where a large amount of data is read and written out to network attached devices. Lustre will provide QOS guarantees to meet these situations with high rate I/O.

A collaborative cache for Lustre will be added which enables multiple OSTs to cache data that is frequently used by multiple clients. In Lustre, read requests for a file are serviced in two phases: a lock request precedes the actual read request, and while the OST is providing the read lock, it can asses where in the cluster the data has already been cached to include a referral to that node for reading. The Lustre collaborative cache is globally coherent.

Figure 12: Collaborative Read Cache for Lustre



6.3. Advanced Security

File System security is a very important aspect of a distributed file system. The standard aspects of security are authentication, authorization, and encryption. While SANs are largely unprotected, Lustre provides the OSTs with a secure network attached disk (NASD) features.

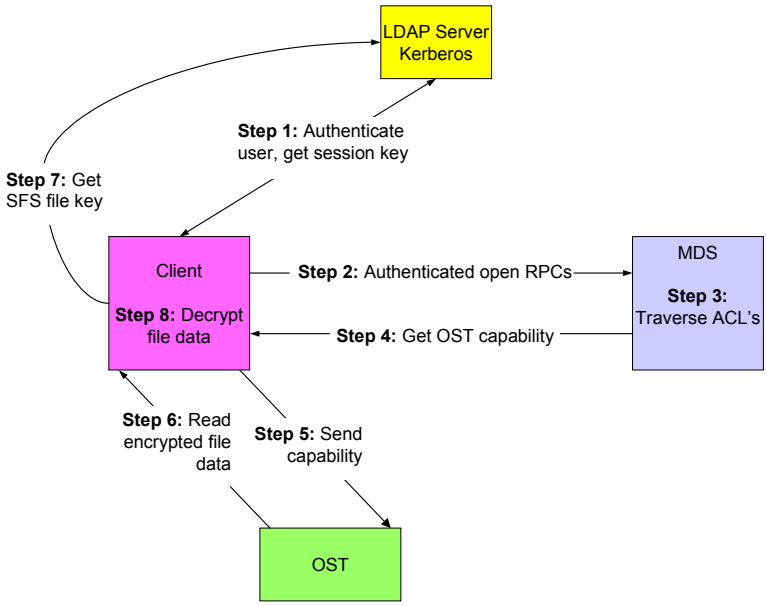
Rather than selecting and integrating a specific authentication service, Lustre can easily be integrated with existing authentication mechanisms using the Generic Security Service Application Programming Interface (GSS-API), an open standard that provides secure session communications supporting authentication, data integrity, and data confidentiality. Lustre authentication will support Kerberos 5 and PKI mechanisms as a backend for authentication.

Lustre intends to provide authorization using access control lists that follow the POSIX ACL semantics. The flexibility and additional capabilities provided by ACLs are especially important in clusters that may support thousands of nodes and user accounts.

Data privacy is expected to be ensured using an encryption mechanism such as that provided by the StorageTek/University of Minnesota SFS file system, where data can actually be automatically encrypted and decrypted on the client based on a shared key protocol which makes file sharing by project a natural operation.

The OSTs are protected with a very efficient capability-based security mechanism which provides very significant optimizations over the original NASD protocol.

Figure 13: Lustre Read File Security



6.4 Further Features

Sharing existing file systems in a cluster file system to provide redundancy and load balancing to existing operations is the ultimate dream for small scale clusters. Lustre's careful separation of protocols and code modules makes this a relatively simple target.

Lustre will provide file system sharing with full coherency by providing support for SAN networking, together with a combined MDS/OST which exports both the data and metadata API's from a single file system.

File system snapshots are a cornerstone of enterprise storage management. Lustre will provide fully featured snapshots, including rollback, old file directories, and copy on write semantics. This will be implemented as a combination of snapshot infrastructure on the client, OSTs, and metadata servers, each requiring only a small addition to its infrastructure.

7. Summary

Lustre is an advanced storage architecture and distributed file system that provides significant performance, scalability, and flexibility to computing clusters, enterprise networks, and shared-data in network-oriented computing environments. Lustre uses an object storage model for file I/O, and storage management to provide a substantially more efficient division of labor between computing and storage resources. Replicated, failover metadata Servers (MDSs) maintain a transactional record of high-level file and file system changes. Distributed Object Storage Targets (OSTs) are responsible for actual file system I/O and for interfacing with local or networked storage devices known as Object-Based Disks (OBDs).

Lustre leverages open standards such as Linux, XML, LDAP, SNMP, readily available open source libraries, and existing file systems to provide a powerful, scalable, reliable distributed file system. Lustre uses sophisticated, cutting-edge failover, replication, and recovery techniques to eliminate downtime and to maximize file system availability, thereby maximizing performance and productivity. Cluster File Systems, Inc., creators of Lustre, are actively working with hardware manufacturers to help develop the next generation of intelligent storage devices, where hardware improvements can further offload data processing from the software components of a distributed file system to the storage devices themselves.

Lustre is open source software licensed under the GPL. Cluster File Systems provides customization, contract development, training, and service for Lustre. In addition to service, our partners can also provide packaged solutions under their own licensing terms.

Contact Cluster File Systems, Inc. at info@clusterfs.com. You can obtain additional information about Lustre, including the current documentation and source code, from the Lustre Web site at http://www.lustre.org.

Lustre Whitepaper Version 1.0: November 11th, 2002