SEAGATE

Inside The Lustre File System

Technology Paper

An introduction to the inner workings of the world’s most scalable and popular open source HPC file system

Torben Kling Petersen, PhD

The Lustre High Performance Parallel File System

Introduction

Ever since the precursor to Lustre® (known as the Object-Based Filesystem, or ODBFS) was developed at Carnegie Mellon University in 1999, Lustre has been at the heart of high performance computing, providing the necessary throughput and scalability to many of the fastest supercomputers in the world. Lustre has experienced a number of changes and, despite the code being open source, the ownership has changed hands a number of times. From the original company started by Dr. Peter Braam (Cluster File Systems, or CFS), which was acquired by Sun Microsystems in 2008—which was in turn acquired by Oracle in 2010—to the acquisition of the Lustre assets by Xyratex in 2013, the open source community has supported the proliferation and acceptance of Lustre.

In 2011, industry trade groups like OpenSFS[1], together with its European sister organization, EOFS[2], took a leading role in the continued development of Lustre, using member fees and donations to drive the evolution of specific projects, along with those sponsored by users[3] such as Oak Ridge National Laboratory, Lawrence Livermore National Laboratory and the French Atomic Energy Commission (CEA), to mention a few.

Today, in 2014, the Lustre community is stronger than ever, and seven of the top 10 high performance computing (HPC) systems on the international Top 500[4] list (as well as 75+ of the top 100) are running the Lustre high performance parallel file system.

It is important to note that this paper is not intended as a training or operations manual, or even as a technical deep dive into Lustre internals, but rather as an introduction to the inner workings of the file system.

[1] Open Scalable File Systems – http://opensfs.org

[2] European Open File Systems – http://www.eofs.org

[3] http://top500.org

[4] This is a summary of characteristics for the largest supercomputer site. For more information see http://top500.org

Lustre Nomenclature

Lustre contains a large number of moving parts, and reviewing all of them is not possible within the confines of this paper. However, some of the main components include:

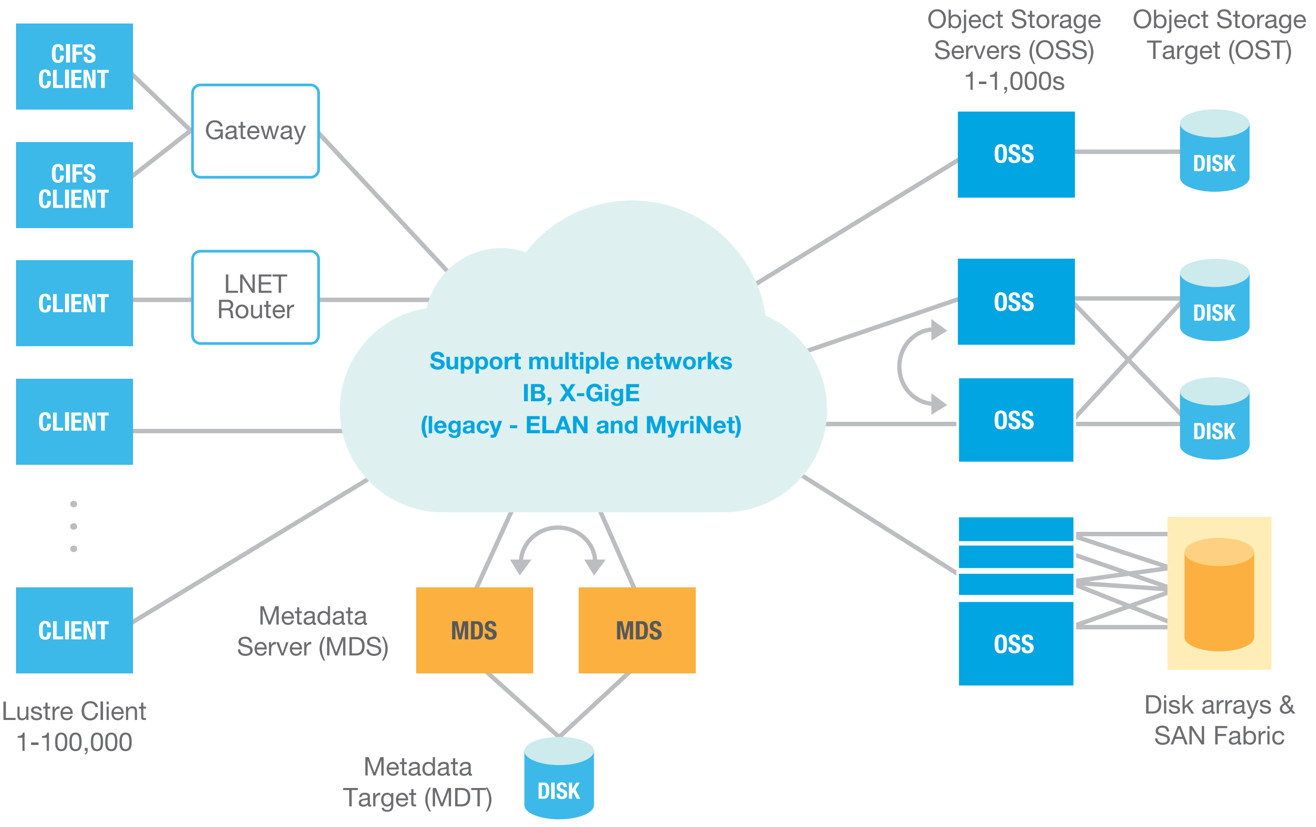
MDS (METADATA SERVER)	Responsible for managing all the metadata operations of the entire file system. Usually set up as a single pair of nodes in an active/passive failover mode with shared storage. For Lustre 2.4 and beyond, this can be extended to multiple pairs of active/active metadata servers (distributed namespace servers, or DNE).
MDT (METADATA TARGET)	The storage component where all metadata, for all files, is stored. Usually a single RAID10 array for maximum redundancy.
OST (OBJECT STORAGE TARGET)	Usually a RAID 5 or RAID 6 array used to store the actual data. The OSTs are a LUN specifically formatted for storing Lustre object data.
OSS (OBJECT STORAGE SERVER)	Nodes responsible for handling I/O data transfer for the OSTs. A single OSS can manage multiple OSTs and is usually configured on active/active failover pairs with shared drives for dynamic redundancy
LNET	The Lustre network layer is responsible for abstracting drives and physical components connecting the different parts of the cluster to the file system (see below).
LUSTRE CLIENT	The Lustre client is usually a compute node with a Lustre client installed allowing it to communicate with the file system. The client can be directly attached to the file system, or it can work through a LNET router.
LNET ROUTER	A node used to do network fabric or address range translation between directly attached clients and remote, network-connected client compute and workstation resources. Often used to let several different compute clusters talk to a single shared file system.

Lustre Features

Today, Lustre is based entirely on Linux and is using kernel-based server modules to deliver the expected performance. Lustre can support many types of clients and runs on almost any modern hardware. Scalability is one of the most important features of Lustre and can be used to create a single namespace of what appears to be almost limitless capacity.

However, while the general Lustre community is still far from reaching maximum configurations, the gap is rapidly decreasing for Lustre version 1.x compared to some of the largest supercomputer wins just recently announced. For example, the German Climate Computing Centre in Hamburg, called DKRZ[5], will begin installation of one of the world's largest Lustre installations beginning in 2015, and this site requires the use of over 4B files, which is above the current theoretical limit of Lustre version 1.x. Even though Lustre version 1.x may seem limitless, actual deployment practice and customer demand in the field are significantly outpacing aspects of the Lustre version 1.x use model, as will be described further below.

Lustre Features	Theoretical Limits	In Production June 2014[4]
File system size	512 PB	55 PB
Number of files	4 billion per MDT*	Approx. 2 billion
Single file size	2.5 PB	100 TB
Aggregate performance	7 TB/s	1.1 TB/s
No. of clients	>100,000	Approx. 50,000



[5] PRNewswire, June 26, 2014, Bull and Xyratex, a Seagate Company, Announce Reseller Partner Agreement and Major Design Win at DKRZ, http://www.prnewswire.com/news-releases/bull-and-xyratex-a-seagate-company-announce-reseller-partner-agreement-and-major-design-win-at-dkrz-264694861.html"

The Big Idea

While separating metadata and content on different systems is not unique, the design Lustre uses to do this has proven highly efficient and reliable.

The separation of metadata operations from actual data I/O operations allows the fundamental Lustre file architecture to grow to almost limitless theoretical capacity and performance. In the near term, however, practical constraints due to implementation are usually seen in resulting metadata performance limits, such as file creates/deletes and stats.

The original design with a single active/passive failover capable metadata server has been criticized by some as a weak point of Lustre—a feature that will limit its future usefulness. However, this is addressed in Lustre 2.4 and beyond (see below). By using discrete arrays as object storage targets (OSTs), Lustre is also capable of expanding the file system capacity and performance by adding more OSTs to an existing solution. Furthermore, this can be done online, and when done correctly as per industry best practice, will not incur any downtime. In addition, Lustre supports a number of extensions such as LNET routers and Lustre-compatible applications for sharing the file system over NFS or CIFS.

How Lustre Works

The concept of Lustre is actually quite simple. When a client requests to write a file to the file system, it contacts the MDS with a write request. The MDS checks the user authentication and the intended location of the file. Depending on the directory settings or file system settings, the MDS sends back a list of OSTs that the client can use to write the file. Once that reply is sent, the client interacts exclusively with the assigned OSTs without having to communicate with the MDS. This is true for any file regardless of size, whether it's a few bytes or a few terabytes. And as this communication (if using InfiniBand) will be done exclusively over Remote Direct Memory Access (RDMA), the performance is exceptional and the latency is minimal (see below).

The actual distribution is defined by the specific striping settings of a file, directory or file system. If the specific file write command does not have a pre-defined set of stripe settings, it will inherit the settings of the directory or file system to which it is written.

A file, a directory or the entire file system can be set to handle distribution using several parameters:

- **Stripe size –** The specific size of an object (a file usually consists of a number of stripes). The stripe size is usually set to 1 MB as this corresponds to the default RPC size in Lustre.
- **Stripe count –** Determines how many OSTs are to be used for a single file. The default is 1, but it can be set arbitrarily.
- **Stripe index –** Where to put the initial object of a file. This is usually set to MDS discretion. This allows the MDS to place files on OSTs with more capacity than others to maintain a more balanced system.

While Lustre is fully POSIX compliant (with the sole exception of updates to atime), it handles all transactions atomically. This means that all I/O requests are executed in sequence without interruption to prevent conflicts. No data is being cached outside the clients, and a file read or write acknowledgement is required to release the file lock.

To achieve parallelism, Lustre uses a distributed lock manager to handle the thousands of supported clients trying to access the file system. To clarify, each component in the Lustre file system runs an instance of the Lustre Distributed Lock Manager (LDLM). The LDLM provides a means to ensure that data is updated in a consistent fashion across multiple OSS and OST nodes.

Data Flow

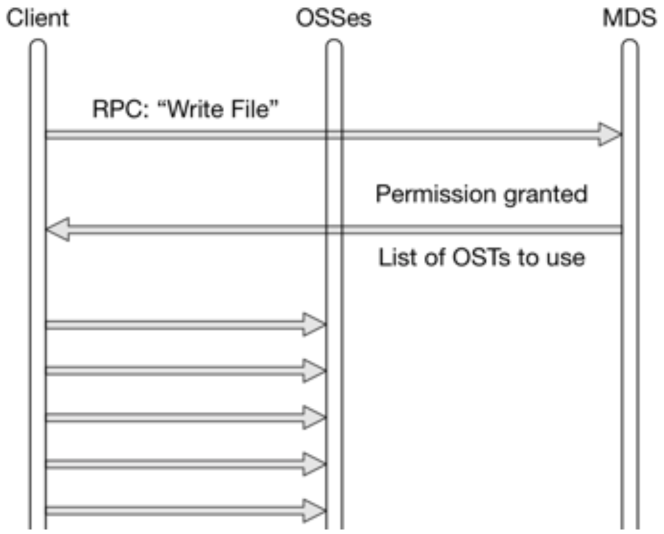
The mechanism employed by Lustre to manage a write or read operation can be simplified using the following examples (note that RDMA assumes InfiniBand-based networks):

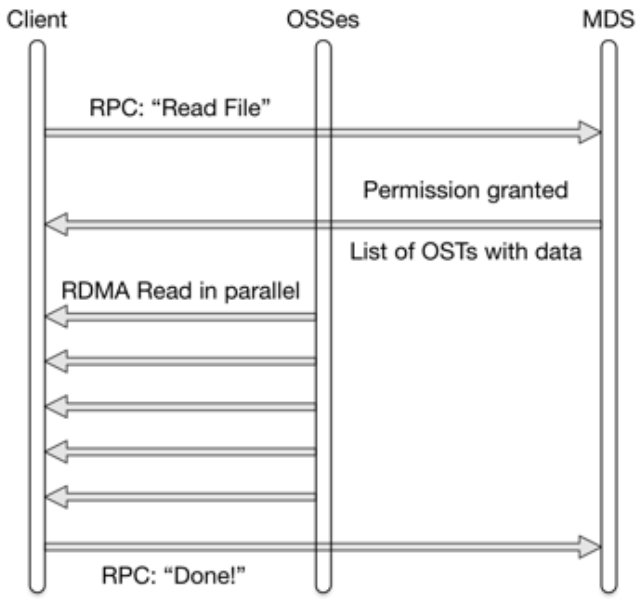
Write:

1. Client "asks" the MDS for permission to write a file.
2. MDS checks access rights, file properties, etc., and returns a list of OSTs to use.
3. The clients communicate directly with each OST, which writes the data in parallel until done (this communication continues until the entire file is written regardless of size, from KBs to TBs), and does not involve the MDS further.

Read:

1. Client "asks" the MDS for permission to read a file.
2. MDS checks access rights and file location, and returns a list of OSTs where the different stripes of the file are located.
3. The clients communicate directly with each OST, which reads the data in parallel until done reading the parts of the file the clients need.
4. Once the client is finished, it sends a single "done" to the metadata server to make the file accessible to other clients.





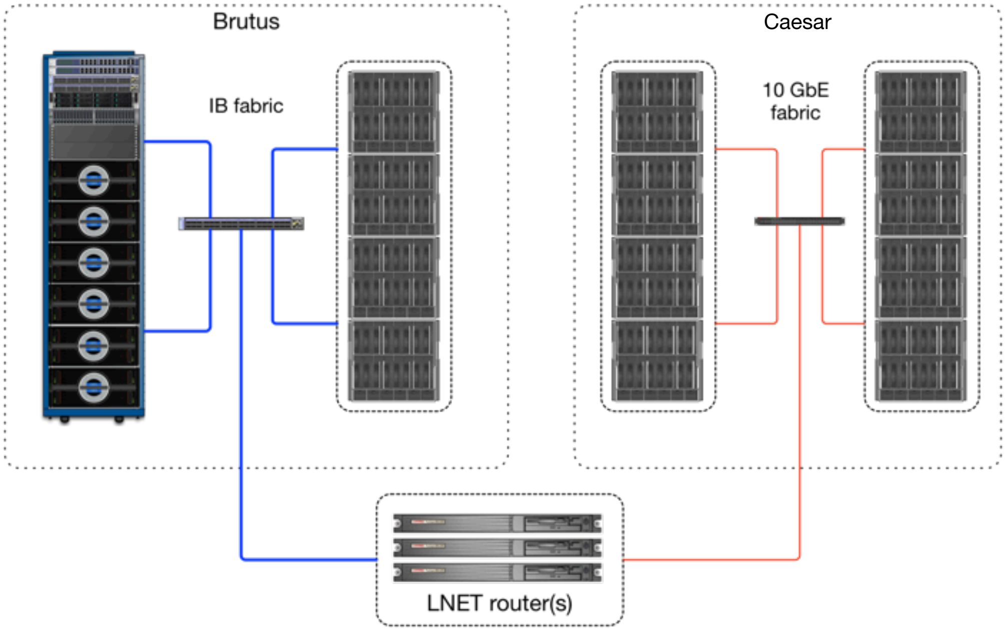
Networking Concepts

One of the unique features of Lustre is the abstraction of the network layer, which is done using a feature called LNET. In addition to the obvious network entities such as Ethernet (1, 10, 40 GbE and beyond) and InfiniBand (SDR, DDR, QDR, FDR and beyond), it is also capable of supporting legacy fabrics such as Quadrics (ELAN) and MyriNet. In addition, it has been enhanced to handle specific compute fabrics such as Cray Gemini, Aries and Cascade.

LNET is part of the Linux kernel space and allows for full RDMA (in the case of InfiniBand) throughput and zero copy communications. This means that for large streaming I/O, Lustre can initiate a multi OST read or write using a single Remote Procedure Call (RPC) that allows the client(s) to access data using pure RDMA, regardless of the amount of data being transmitted. This allows for extreme low-latency communication and extreme throughput.

LNET Routing

The unique features of LNET also provide the means for some advanced networking features. One such feature is LNET routing. Many HPC sites have several different compute systems as well as auxiliary systems and workstations for visualization and data analysis. Some of these network-connected resources use a separate file system for each, which leads users to manually or semi-automatically copy data between them. This not only introduces delays in moving large amounts of data, but also creates significant administrative overhead in file version control and managing duplicate files on multiple distributed systems. To counter this administrative challenge, some users prefer to have a single, large file system as a repository for all data in addition to the scratch properties usually indicating a Lustre file system in the first place; however, this requires all contributing resources at the site to connect as Lustre clients.



As you would want to separate the data-intensive Message Passing Interface (MPI) traffic between different compute solutions, a single, large interconnect linking all significant resources at a site is not ideal. Lustre solves this problem by having compute nodes sitting between the different interconnect fabrics, essentially acting as a client to each fabric. The LNET routing capabilities then provide an efficient (near wire speed) protocol to permit bridging between the networks, allowing a remote set of non-Lustre clients to mount the Lustre file system. In addition to network connectivity, file access and underlying lock management, the LNET network abstraction layer also allows the communication to undergo fabric translation, such as from Ethernet to InfiniBand.

In addition to linking Ethernet and InfiniBand-based interconnect fabrics, this methodology is also used by supercomputer systems such as the Cray XC series, in which the internal Aries interconnect needs service nodes (in effect, Lustre routers) to connect to an external IB-based storage fabric. An excellent example of this is the UK National High Performance Computing Facility system called "Archer," which was recently installed at the HPCC in Edinburgh[6].

[6] http://www.archer.ac.uk/training/courses/craytools/pdf/architecture-overview.pdf

Lustre Version 2 and Beyond

Although Lustre version 2 has been available since 2010, a majority of the systems currently running Lustre are still on the 1.8.x branch. The reasons for this are many, and vary from user to user. One of the main reasons, however, is a legacy perception of the stability of the Lustre code, which focused primarily on performance. Most development efforts up to Lustre version 1.8 were focused on stability to attain maximum performance, whereas the majority of the projects since then have focused on adding increased stability and robustness to attain even higher performance and data-capacity scale along with more enterprise features.

While most of today's research and academic users may not need, nor use, any of the new Lustre version 2.x features, leaders in government and the commercial technical computing industry are demanding better data management tools, reliability and serviceability. All of this means significantly improved solution robustness, flexibility and bottom-line usefulness or user productivity, not just the limited scope of "Lustre code" stability primarily for performance.

Overall, with proven supercomputing industry success and an expanded universe of new users in oil and gas, life sciences, genomics, pharmacology, finance and Big Data analytics, Lustre must be absolutely stable, robust and highly scalable in performance and capacity, as well as support a rich set of improved tools and capabilities. The best of both worlds is coming together. Today, Lustre 2.x systems are being deployed in many data centers belonging to energy companies and climate and weather services, as well as within financial institutions.

The following are brief explanations of some of these new Lustre 2.x features.

Lustre Changelogs

Monitoring changes in a Lustre 1.6/1.8 file system is highly resource-intensive and requires frequent disk scanning. As this is not feasible on a large, active system, a new mechanism to monitor changes was needed.

Lustre 2.0 and beyond (Lustre 2.6 is the most recent version as of this writing) contains a new kernel ring buffer called Lustre changelogs. The changelogs feature records events that change the file system namespace or file metadata. Depending on the intended use (Hierarchical Storage Management [HSM], file pruning, system rebalancing, quotas, etc.), specific changes such as file creation, deletion, renaming and attribute changes are recorded with the target and parent file identifiers (FIDs), the name of the target and a timestamp.

The data can then be used for a variety of purposes:

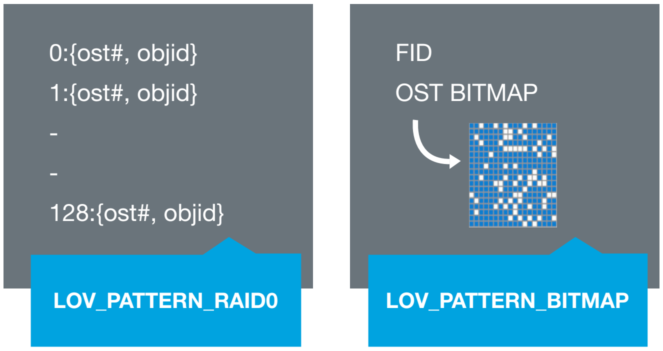
- Capture recent changes to feed into the policy engine of an archiving system or HSM backend
- Use changelog entries to exactly replicate changes in a file system mirror
- Set up "watch scripts" that take action on certain events or directories
- Maintain a rough audit trail (file/directory changes with timestamps but no user information)

To manage such events, several tools are currently available. Perhaps the prime tool currently available is RobinHood (developed by CEA as the policy engine for HSM), which extracts desired data and stores it in one or more external MySQL databases. The databases can then be used to generate the necessary or desired actions on files and data.

Data Layout Policies

While the use of extended attributes allowed Lustre to distribute a single file over up to 160 OSTs to provide a single shared file with maximum bandwidth, this has been shown to be an ineffective approach as requirements increase. Moving to an FID bitmap, and changing the distribution coding, a single file can now be distributed over up to 2,000 OSTs. This is usually referred to as very wide striping.

That said, most users keep the file distribution (i.e., striping) to a minimum and prefer to achieve parallelism by letting many different applications or write threads do I/O to a single OST and allow the MDS to distribute the load to all available OSTs.

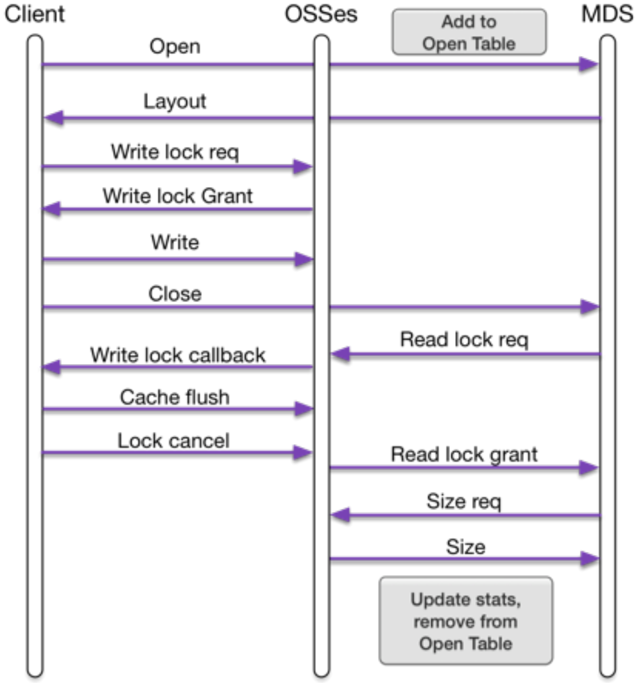


Size on MDS

One issue that emerges with a parallel file system capable of delivering a single namespace of 100+ PBs is the number of files and directories it needs to handle. Using tools such as “ls –l” and “df –h” incurs a significant performance impact on large systems, as each single object needs to be scanned. In Lustre versions earlier than 2.0, listing the files in a directory, getting the size of files and directories, and determining how much of the file system is being used required doing a “glimpse lookup”—in other words, finding the last object of a file and calculating the approximate size of each file based on the object size and extent of the objects. This requires the use of Lustre-specific pre-fix “lfs” that implements the above-mentioned shortcuts.

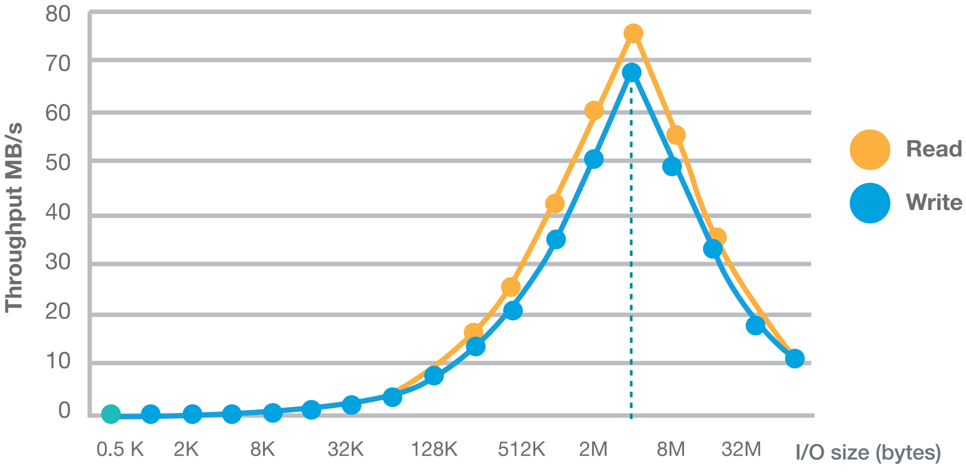
Obviously, this does not scale well, and best practices have been recommended (by still using the “lfs” pre-fix) to read sizes, etc., from the MDS rather than scan all the OSTs.

The Size on MDS (SOM) feature keeps a persistent database of Open Files that are indexed by FIDs. Clients request Size of file from MDS not by communicating to every OSS, but rather by single RPC to the MDS.



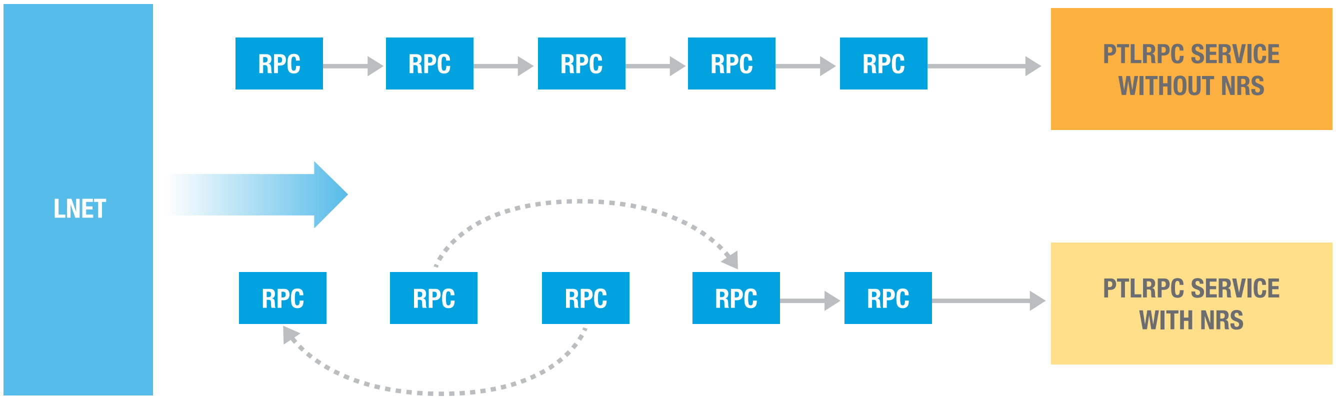
4 MB I/O

While the initial design of Lustre was very forward thinking, a few early decisions have had a significant impact on performance today. One such decision was to hardcode Lustre network transfers to 1 MB RPCs and cap network buffers to the same level. While this made perfect sense with the disk drives of the time, modern drives are different, and I/O throughput peaks at around 4 MB for both reads and writes using the original settings. The solution to this problem is to allow multiple 1 MB chunks to travel from client to server as part of a single, client-based read or write RPC. Rather than implementing fundamental changes to enable I/O buffers larger than 1 MB, Lustre can conduct multiple transfers as part of the RPC, before the OST backend initiates I/O. Because the OSS read cache feature provides read-only caching of data on every OSS, the size of the network transfer from server to client, for read operations, is less critical.



Network Request Scheduler (NRS)

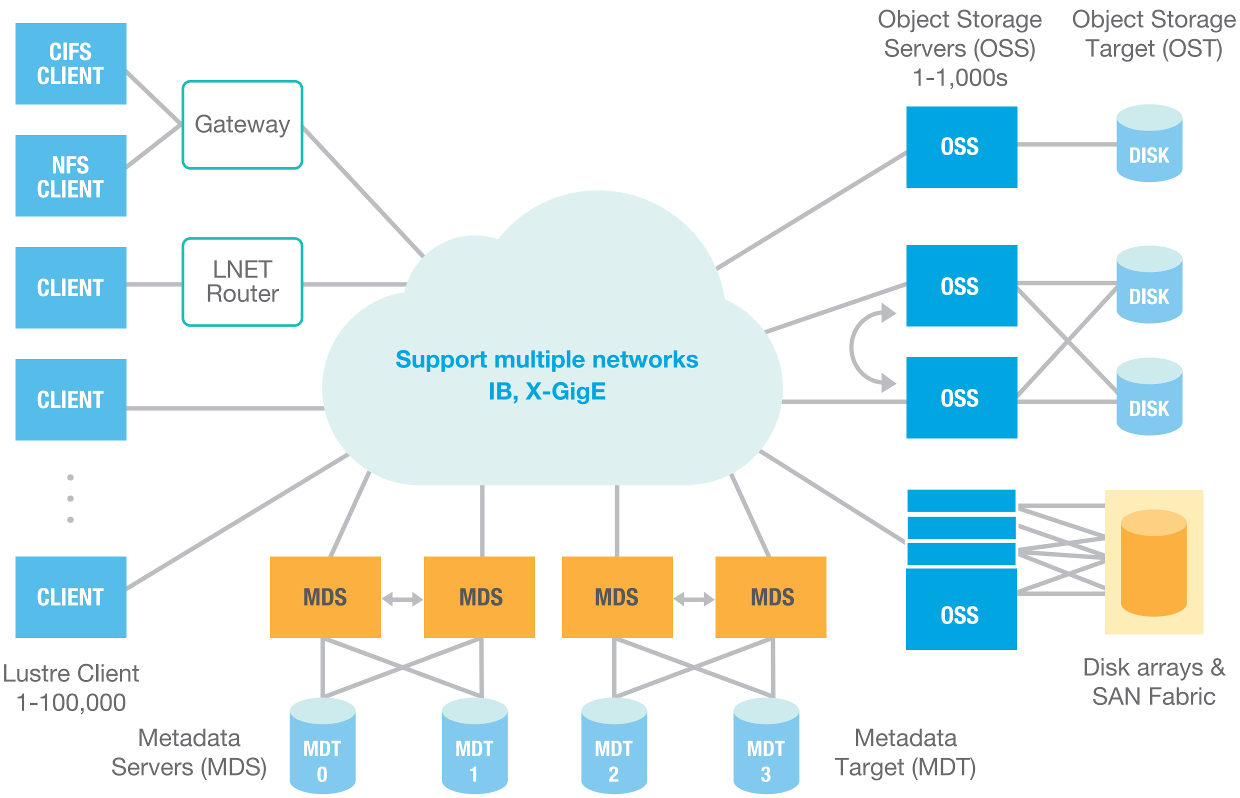
Lustre was originally created for a specific purpose: large streaming I/Os typical of supercomputers in U.S. national labs at the turn of the century. However, today's systems often require a large number of small files to be written and read as part of their execution. When studying individual client I/O behavior in large systems, taken individually, I/O often looks quite random. For a Lustre file system, random I/O is very detrimental with regards to performance, and steps are often taken to minimize random access I/O. However, when studying a larger system as a whole, these random I/O calls can form patterns reminiscent of streaming I/O behavior. The purpose of the network request scheduler is to re-order collective I/O (from many clients writing to a single file) into an effective, sequential stream (through the Portal RPC, or PTLRPC, service) while maintaining consistency with read-write ordering semantics and locking.



The NRS optimizes throughput by implementing several basic policies, such as fair client I/O scheduling and prioritized clients.

The NRS feature was introduced in Lustre 2.4 and works transparently by monitoring the I/O of the file system.

Distributed Namespace (DNE) Servers



As mentioned earlier, the perceived lack of metadata performance is often considered a major drawback of Lustre. To solve this problem, Lustre 2.4 shipped with enhanced metadata functionality where it is not possible to add multiple metadata namespace servers in an active/active design. Not long ago, the concept of "clustered metadata servers" was intended to offer more features than the current implementation; however, this has proven to be too difficult to develop, and that project has evolved into the current concept of Distributed Namespace (DNE) servers.

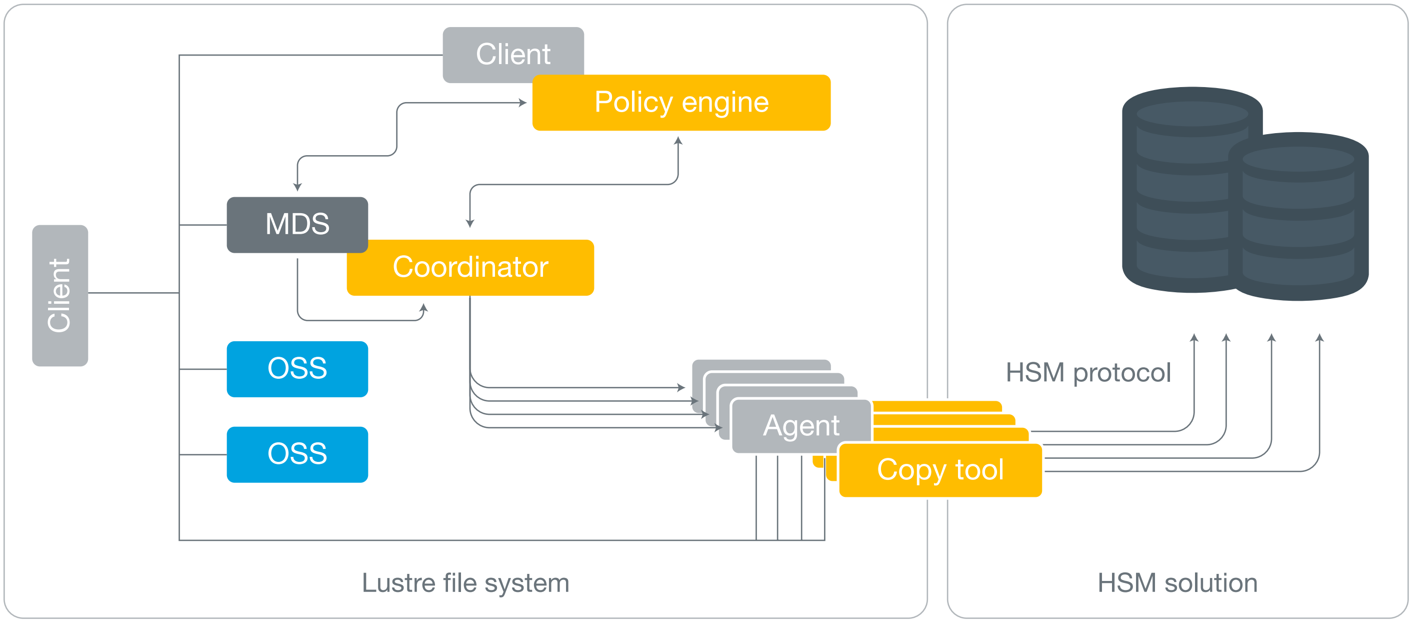
Each DNE server "owns" part of the logical file tree, with the MDT0 handling the root level and each DNE handling a piece of the entire namespace—for example, /mnt/lustre/a and /mnt/lustre/b, where a and b are physically separate parts of the file system. While the most obvious benefit of this approach is the ability to scale I/O to the system supporting in excess of 100,000 file creates, there are other, similarly important use cases, such as separating different applications' I/O loads to allow a well-behaved I/O to be unaffected by another application's disruptive I/O pattern.

Hierarchical Storage Management (HSM)

Lustre doesn't perform backups; currently, backups are performed either manually or using a brute force approach with one or more data movers bridging the area between Lustre and a secondary file system. By using Lustre-aware tools such as Lustre rsync or Mutil[7], it is possible to keep two or more directories in sync, thereby creating an effective backup copy. However, this approach lacks automation and flexibility, and more importantly, does not offer any intelligent data management features.

This has long been recognized by the French Atomic Energy Commission (CEA), which developed a full HSM solution for internal use that is now part of the Lustre 2.5 source tree.

[7] http://sourceforge.net/projects/mutil/files/



The HSM solution consists of three main functional units:

- **The Policy Engine** – Open source tool called RobinHood[8]. RobinHood uses the changelogs to create a database with the parameters used to define a policy, such as last touched, directory location, file size, owner, etc.
- **Coordinator** – Effectively adds a bit to the metadata to define it as "moved" so that a read request is aware that the file must be restored to the storage before the process can begin. The coordinator is also responsible for managing the objects that can be removed from the OSTs once they've been fully migrated.
- **CopyTool** – The system can support one or more copytools (each running on its own data mover client) that are responsible for copying out or restoring the object to an HSM backend.

The HSM backend is usually a complete storage system in its own right. It can be either disk-based or tape-based, or a combination. Today, there are several options from which to choose. The original design uses an open source tape-based backend called High Performance Storage System[9] (HPSS) originally developed by IBM.

In addition to the HPSS-based CopyTool, a POSIX copytool allows connectivity to several other types of HSM backend, such as:

- TSM – Tivoli Storage Manager
- SAM/QFS – Linux port of the old Sun Microsystem HSM solution
- TAS – Tiered Adaptive Storage – Cray HSM solution based on the Versity implementation of SAM-QFS[10]
- DMF – SGI's storage tier virtualization solution[11]

[8] https://github.com/cea-hpc/robinhood/wiki

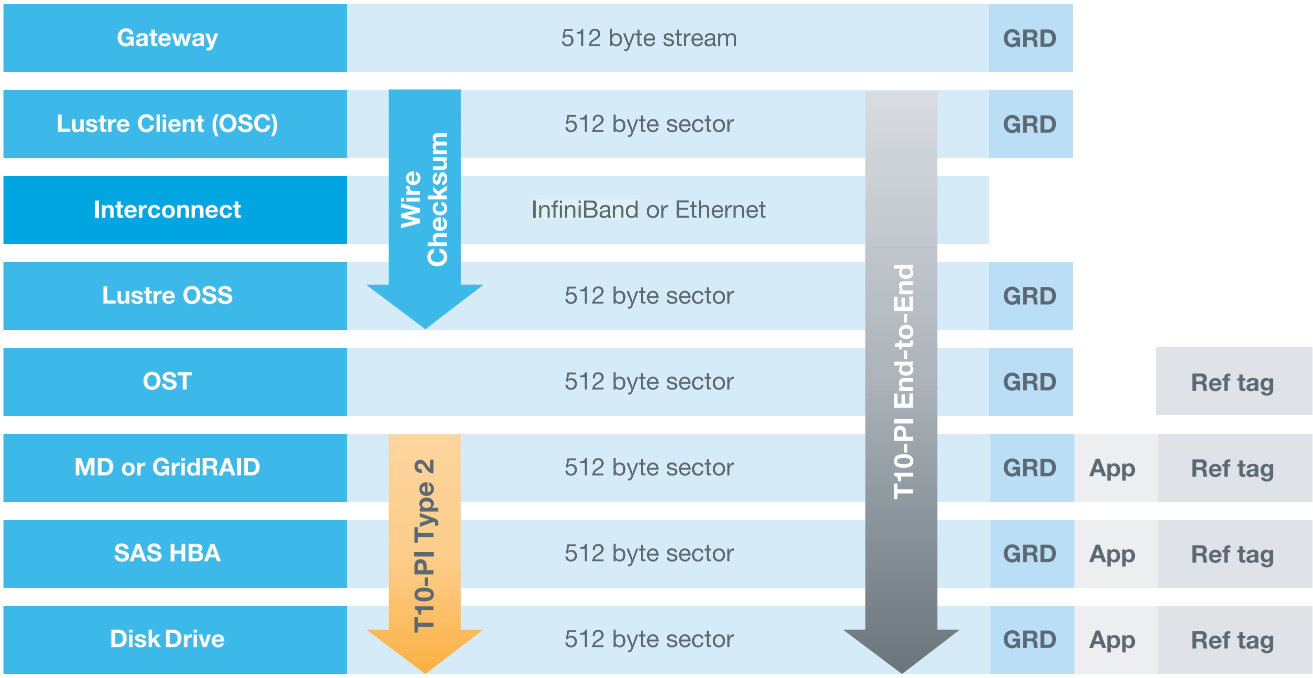
[9] http://www.hpss-collaboration.org

[10] http://www.versity.com / http://www.cray.com/Products/Storage/Tiered-Adaptive-Storage.aspx

[11] https://www.sgi.com/products/storage/idm/dmf.html

Data Integrity

Silent data corruption and bit rot have long been major issues with large file systems. While Lustre does employ check summing when sending data over the wire, the data path within each OSS is not protected in a similar way. There are currently several projects under way to add end-to-end data integrity to Lustre. One of the more promising projects is T10-PI (Protection Information, aka T10-DIF)—which, when combined with modern compute hardware, can perform the necessary calculations in hardware and in essence perform the computations at wire speed. While not fully available today, at least one major vendor is developing what's called T10-PI type 2, which entails check summing from the OSS to the disk drive. There is currently a project on the Lustre roadmap to push this all the way to the client.



Gateway	512 byte stream	GRD		
Lustre Client (OSC)	512 byte sector	GRD		
Interconnect	InfiniBand or Ethernet			
Lustre OSS	512 byte sector	GRD		
OST	512 byte sector	GRD		Ref tag
MD or GridRAID	512 byte sector	GRD	App	Ref tag
SAS HBA	512 byte sector	GRD	App	Ref tag
Disk Drive	512 byte sector	GRD	App	Ref tag

Lustre Futures

As mentioned above, Lustre is one of the most successful open source projects; development of new features is carried out by close to 70 developers representing some 20 companies and organizations. However, the complexity of maintaining a large body of source code and ensuring support for newer kernels and Linux distributions while still adding new functionality without introducing disruptive code regression artifacts is a tall order indeed. That said, the list of proposed feature developments is long and covers a number of very interesting features, including enhanced support for small files (including "Data on MDS," or storing small data directly on the MDT when this can be accomplished within a single RPC), enhancements in DNE (such as "striped directories" and "async commits"), modernizing Kerberos support (for wide area Lustre implementations, among others) and improved Lustre file systems check (LFSCK), to mention a few.

Summary and Discussion

Lustre has been a mature and stable file system for a number of years, but its requirements and associated use model are changing. Simplicity of management is now an often stated requirement in Request for Proposals (RFPs), and customer requirement discussions are occurring specifically around the need for a fully functional and scalable management graphical-user interface (in addition to the ubiquitous command line interface). Another trend in Lustre adoption and proliferation is that Lustre is no longer only used for the original purpose of constituting a scratch file system for HPC computations. Today, Lustre is increasingly used as a file system for home directories as well as for project space. This means the requirements for reliability and data integrity increase every year, along with expansion of the compatibility matrix for network- and gateway-connected compute resources. One might think that these requirements are opposite to extreme performance, but "best of both world" solutions are under way that work for both types of requirements.

Today, among leading data-intensive commercial entities and institutions that previously depended on enterprise file systems, there is a clear trend toward moving to Lustre. This includes not only weather and climatology institutions such as DED and DKRZ in Germany, ECMWF and Met Office in the UK and the South African Weather Service (SAWS), but also companies in the financial and energy sectors. The proven stability at sites such as NCSA Blue Waters in the U.S. and CEA in France has done much to prove the reputation that Lustre serves much more than just performance.

It is interesting to see storage companies such as Seagate, EMC and NetApp participating in the development efforts through OpenSFS[12], in addition to compute vendors such as Cray, Fujitsu and SGI augmenting the national labs and other high-profile end users. The continued support (both financially and through in-house development) is not only critical to the future of Lustre, but also forms the basis of the proven success of an open source, high performance parallel file system. The future is bright indeed.

Take the Next Step

Find out more about the Seagate® ClusterStor™ line of HPC storage systems by calling 1.800.SEAGATE or visiting www.seagate.com/hpc

[12] http://opensfs.org/participants/

seagate.com

AMERICAS Seagate Technology LLC 10200 South De Anza Boulevard, Cupertino, California 95014, United States, 408-658-1000
ASIA/PACIFIC Seagate Singapore International Headquarters Pte. Ltd. 7000 Ang Mo Kio Avenue 5, Singapore 569877, 65-6485-3888
EUROPE, MIDDLE EAST AND AFRICA Seagate Technology SAS 16–18, rue du Dôme, 92100 Boulogne-Billancourt, France, 33 1-4186 10 00