An Evaluation of Lustre as a General Purpose File System for CERN

Arne Wiebalck, Olof Bärring,
Tim Bell, Andras Horvath,
Peter Kelemen, Miguel Santos

IT FIO CERN

February 26, 2010

This report summarizes the efforts of an evaluation of the Lustre file system as a candidate for storage consolidation at CERN. Since the general performance and scalability characteristics of Lustre are undisputed and have been shown in numerous benchmarks before, performance benchmarks were limited to the special case of small files. The main focus of this evaluation however was on Lustre's operational qualities.
From the use cases relevant for CERN, the evaluation process derived a list of required criteria for a general purpose file system solution. Using several test instances, the Lustre file system has been examined using this list. Despite the support for some of the desired features, the lack of support for some of the essential requirements, such as strong authentication or user-transparent data migration, it was found that Lustre in its current state is not a valid solution to consolidate file system and storage solutions. For the analysis use case, it could still be considered but this would mean expanding the service catalogue.

Contents

1 Introduction and Methodology

This survey was initiated in early 2009 to evaluate Lustre as a candidate to consolidate storage solutions at CERN. The main aim – besides the benefits of running a scalable high-performance distributed file system – was to increase the overlap between the service teams and reduce the need for special knowledge and expertise for the various solutions.

The survey was structured into following phases:

1. Analyze the use cases and derive a list of requirements;
2. Gather information about Lustre from official sources, such as Lustre trainings or the Lustre User Group Meeting as well as from an exchange of experience with other sites running Lustre;
3. Get hands-on experience in order to familiarize ourselves with certain features and functionalities;
4. Document the findings.

During this evaluation emphasis has been put on operational aspects rather than on performance (with the exception of Lustre's performance for small files). The reason for this approach is that Lustre's performance and scalability features are undisputed and have been shown in many benchmarks by various sites, including the HEPiX File System and Storage Working Group [1]. There was no gain expected by repeating and confirming this assessment. However, the typical use case of Lustre as a fast scalable scratch space in closely controlled environments with small homogeneous user groups is different from what it would be used for at CERN. For this reason – and since such an evaluation is missing so far – manageability in a 24/7 non-closed environment was the primary focus.
This survey created a list of desired features that can be used for future evaluations of similar systems, see Appendix B.

1.1 The Use Cases and Their Requirements

The use cases relevant for CERN (and hence this evaluation) are

- home directories;
- project space;
- analysis space;
- hierarchical storage management system (HSM).

In the following we will describe them, along with their requirements and their current solution at CERN.
AFS[2] is used to store the **home directories** of the 25'000 CERN users. Home directories

are accessed by the users from the interactive cluster (lxplus), from the batch cluster (lxbatch) and from the users' personal PCs and notebooks, at CERN and all over the world. AFS also stores the so-called **project spaces**, which are areas assigned to and managed by certain groups at CERN. These groups include the LHC experiments that use their project space to develop, build or distribute their software, but also other projects or services at CERN, such as the CVS/SVN service and their repositories, the conference management system Indico or the TWiki service, to name a few examples. Organized in around 50'000 volumes and distributed over around 50 servers, the total amount of used AFS space is 25 TiB in approximately 400 million files. The number of accesses seen by the AFS servers is around 1.5 billion per day, whereof around 200 million are reads and writes, i.e. they hit the underlying storage devices. Approximately 1% of the access come from outside the CERN network.
Home directories and project spaces are similar when it comes to the requirements they impose on an underlying storage system. These requirements include:

- strong authentication to ensure data privacy;
- the capability to backup the data (at least once per day);
- reasonable performance for small files ($\ll$1 MiB);
- POSIX compliant file system access;
- support for ACLs & quotas;
- 24/7 availability;
- potentially: wide area access;
- potentially: replication to avoid hot spots and to increase availability.

Analysis space is foreseen for the analysis of data produced by the LHC experiments. A CERN working group has identified the requirements of this use case:

- low-latency (disk-based) access;
- a sustained rate of O(1000) metadata operations per second;
- several GiB/s aggregate bandwidth;
- POSIX compliant file system access;
- the possibility of XrootD[3] access;
- the capability to backup the data.

The current **HSM** used at CERN is the CERN Advanced Storage Manager (CASTOR)[4]. An alternative solution should have

- a flexible interface to allow for interfacing with various system, such as CASTOR or the Tivoli Storage Manager (TSM)[5]. A DMAPI-based[6] interface may allow for easier integration with such systems;
- the ability to scale well the with number of concurrent requests;
- no preference for a request type, e.g. write (migrate).

These lists of requirements of the use cases plus the manageability aspects derived from the experience with the current solutions defined the areas of interest which formed the guidelines to examine Lustre:

- Strong Authentication;
- Backup and Recovery;
- Small File Performance;
- Fault Tolerance;
- HSM Functionality;
- Life Cycle Management;
- Documentation;

In the following chapter these areas are examined for the Lustre file system.

2 Areas of Interest

2.1 Strong Authentication

At the time of this writing, the official Lustre releases do not support strong authentication (the version used for this evaluation was 1.8.0.1). There is a preview (a so-called early adoption) available from the 2.0 series, which provides a "usable, but incomplete"[7], untested and unsupported Kerberos implementation. The expected performance penalty is "of the order of a few percent"[7].
The Kerberos implementation of version 2.0 has been successfully used by the Pittsburgh Supercomputing Center[8], even in a setup over a WAN with cross realm authentication. The current implementation of Kerberos has not been tested at CERN (because of the fact that it is not complete).
The current plans of the Lustre team predict the availability of a fully implemented and supported version of Kerberos for late 2010 or beginning of 2011.[1]

2.2 Backup and Recovery

There are two aspects to consider when backing up a Lustre system: the backup of user data and the backup of metadata. The operations manual [9] proposes several options to backup both types of data. Most of them however require either to halt the Lustre system (file-level backups) or introduce the problem of synchronizing the operation between the server nodes (device-level backups, LVM backups of OSTs and MDTs). Hence, a backup based on LVM snapshots for the MDT and backups of the user data with common backup tools such as TSM seemed to be the only viable options for use at CERN.

2.2.1 User Data Backup and Recovery

The easiest way to provide a consistent and safe backup of the user data is to simply backup the Lustre file system from a client (or multiple clients) using common backup tools. This allows for an easy standardized recovery as the backup happens along with all required file metadata information (access rights, access times, etc.) and allows to decouple the backup of the Lustre MDS from the backup of the OSTs.
One difficulty is to determine the number of files that should be handed over to the backup system as the number of files could be some hundred millions and the usual `stat(2)` call would take way too long to ensure a backup at a reasonable frequency, e.g. once per day. For this, Lustre provides the tool `e2scan` which inspects the inodes on the MDT. Since the number of inodes is fixed at setup time, the time needed to perform a scan is independent of the number of files in the system and grows linearly with the size of the MDT. The time to compile the list of files to backup for a 2TB unloaded MDT was around 20 minutes[2]. The

[1] Please see also section 3.3
[2] Single Intel(R) Xeon(TM) CPU 3.40GHz, 2GB RAM, 300GB SATA disks in a hardware RAID-1 configuration.

list of files needs to be transferred to the Lustre client(s) that perform the actual backup. These clients obviously need sufficient rights to access all the data.
The version 2.0 feature of Changelogs (a mechanism which is supposed to track meta data operations on the Lustre file system and to which a client can subscribe) has been looked at as well, since this would render e2scan obsolete and create the list of files for backup on the fly. While this new mechanism works for some events, such as file creation or deletion, it does not (yet) inform subscribers about file changes, which would obviously be essential for backup. There are tags foreseen for this, but they are not yet implemented and will only come when the currently developed HSM extensions need them[10].

2.2.2 Metadata Backup and Recovery

For the backup of the metadata, LVM-based snapshots are recommended: the MDT is served from an LVM device from which snapshots are taken at regular intervals. This snapshotting is required in order to force Lustre to bring the MDT into a consistent state for backup. Once the snapshot is created, it is mounted as type lustrefs, the metadata files and the extended attributes are extracted, tar'ed and sent to the backup system, just as described in the operations manual. Afterwards, the file system is umounted and the snapshot deleted.

2.2.3 Discussion

Since the MDT backup and the backup of the OSTs are decoupled, consistency problems may arise: in case of a metadata recovery, files that have been created/deleted after the MDT snapshot has been taken are not/still available in the metadata and hence cause confusion on the clients. The former will create so-called *ghost files*, i.e. files that according to the MDT should exist, but which the OSTs don't know anymore. The latter will create so-called *orphan files*, i.e. objects on the OSTs for which no metadata entry exists.
Upon mount, Lustre will delete objects for which no metadata exists, i.e. it brings itself into a consistent state. There is no metadata information stored along with the objects on the OSTs and hence it is not possible to resynchronize an MDT from the OSSs.[3] Obsolete metadata entries have to be cleaned up manually by a (elaborate) file system check.
Backup for user data and metadata has been tested and set up on the PPS instance with TSM[5]. For user data backup, the list of files has been passed to TSM's command line interface. A further optimization could be to directly communicate with the backup API provided by the TSM client.

2.3 Small Files

2.3.1 Performance

For home directories and project space, the performance of small files, i.e. files up to 1 MiB in size, becomes particularly important. As Lustre is typically used – and hence tested –

[3] This is not completely true as files without names can be created from OST objects. The identification of these files may be difficult, however.

with large files, i.e. files of 1 GiB and above, this survey includes some simple performance measurements of Lustre's small file performance.
The benchmark chosen to assess Lustre's small file performance is PostMark[11] version 1.51, as it has been designed as a benchmark for applications that use small files, such as mail services or web servers. In addition, PostMark's able to read its configuration from a file which facilitates an easy reproducibility. Rather than an in-depth performance measurement (with all potential tuning), the benchmark results as presented in the following should be regarded as a first indicator of how Lustre behaves when dealing with small files.
The configuration of PostMark was for file sizes between 32 B and 1 MiB to create 100'000 files and 250'000 transactions (number of read/write/create/delete operations):

```
Current configuration is:
The base number of files is 100000
Transactions: 250000
Files are 64 bytes in size
Working directory:
/lustre/pps/striped (weight=1)
Block sizes are: read=512 bytes, write=512 bytes
Biases are: read/append=5, create/delete=5
Using Unix buffered file I/O
Random number generator seed is 42
Report format is verbose.
```

All measurements show results from a single client. The results of these measurements are displayed in figures 1 and 2. The bumps in the read/write rate for striped files are due to filling up the slab cache (which is a known bug in the version used) and to the manually enforced flush of that cache.
It should be noted that the Operations Manual in its 'Lustre Monitoring and Torubleshooting' section provides tuning parameters to improve performance for small files.
The main reason why Lustre's performance degrades with smaller files is supposedly the double-lookup with the metadata server: for every file (or better object) the client needs to talk to the metadata server (to find out the location) and connect in a second step to the object storage target. AFS, for instance, avoids this problem by introducing 'volumes' (basically containers for files and directories), which reduce the communication with the central servers significantly. The reason that writes are faster than reads can be explained by the usage of buffering (which has not been switched of when running PostMark) or by the fact that objects are created in bunches, rather than on a one-by-one basis.
The reason why small files in striped directories show a bad performance is that the objects on the OSTs are created on file creation, i.e. when the file size is not yet known. Hence, small files should be kept in non-striped directories. This forbids mixing small and big files (where we would like to benefit from the performance advantage of striping). For completeness it should be noted that the stripe count can be set on a per file basis, but since this needs to be done *before* the file is created, this is impractical in real life. The stripe count of a file

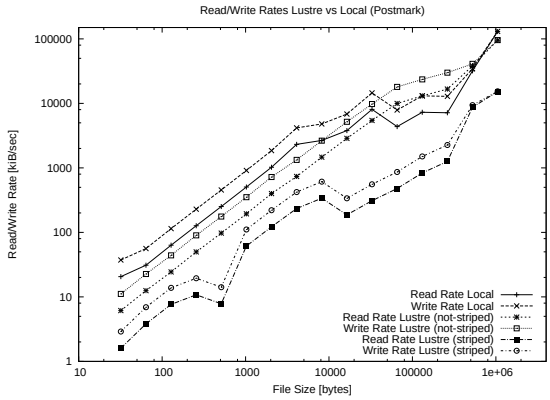

Figure 1: Lustre's read/write performance as measured with PostMark.

cannot be changed after the file has been created.

2.3.2 Consistency

The fsx file system exerciser[12] was used to check the file system integrity. One data error was observed:

```
READ BAD DATA: offset = 0x1cc, size = 0x6c4f
OFFSET  GOOD    BAD     RANGE
0x001cc 0xcf4e  000000    0x179
operation# (mod 256) for the bad dataunknown, check HOLE and EXTEND ops
```

The error was not reproducible, however. No other test performed has shown problems with Lustre's consistency.

2.4 Fault Tolerance

Lustre supports failover for the metadata (MDS) and the user data (OSS). It relies on external packages, such as heartbeat[13], pacemaker[14], or OpenAIS[15]. The operations manual[9] contains a (supposedly outdated) section on Lustre's failover capabilities, along with instructions on how to set it up. The authors have done this setup during a Lustre course, where it was sucessfully tested. For this reason and since several sites use and report this mechanism to work, it has not been studied in more detail during this survey.
Lustre's capability of failover between OSSs however can be used (and is used by some sites) to actually replace OSSs (not OSTs).
The MDS of the iSCSI-based PPS instance that has been used during this survey (see

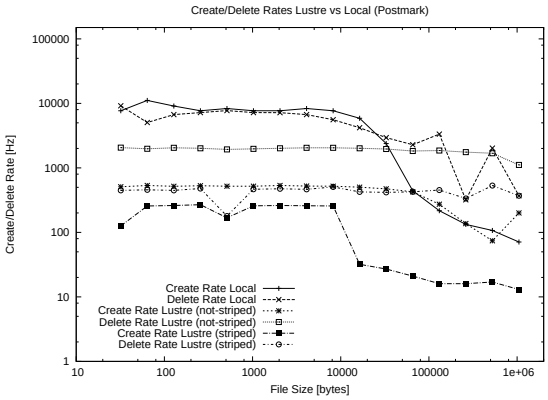

Figure 2: Lustre's create/delete performance as measured with PostMark.

Appendix A for a description) used a setup completely resilient against any single component failure. Although not directly Lustre related, this setup has proven to be stable during all the testing performed, including for instance the PostMark benchmarks for the small files use case.

2.5 HSM Functionality

At the time of this writing, Lustre does not provide support for an HSM system. There is, however, a joint (and very active) project of CEA[16] and the Lustre team to add such an extension. One of the design goals is to define the HSM interface in a way that Lustre could be hooked up to an arbitrary HSM systems. The Lustre team has decided not to use the DMAPI interface.
Initial implementations of this interface are foreseen for HPSS[17] and SAM/QFS[18]. A first version of this (along with an HPSS implementation) is expected to be available in 2010.

2.6 Life Cycle Managment (LCM)

There exist few tools for Lustre life cycle management. An overview on available tools for setup, monitoring and management is available from [19].

2.6.1 Installation and Setup

The installation of Lustre is quite straightforward and well documented in the operations manual[9]. The manual also provides information on how to patch and compile a Lustre kernel in case the precompiled packages from the SUN website cannot be used for some reason. This was necessary to facilitate the use of iSCSI and the instructions in the manual were

found to be correct. As CERN uses the Quattor framework[20] to install machines, Quattor templates and profiles for Lustre servers have been created. Along with a small script to set up the devices, this allowed for an semi-automatic installation of Lustre instances.
Upgrades, both for Lustre and the underlying system (kernel), cannot be easily be done in a user–transparent way, mainly since the the data can not be moved transparently between OSTs.
The size of the MDT determines the number of files that can be stored in a Lustre file system. Changing the (size of the) MDT later on seems to be possible, but difficult. Good planning is required.

2.6.2 Monitoring

As also described in the Lustre operations manual, there are several options to monitor a Lustre system: the built-in SNMP module, `collectl`[21] and LMT2[22]. The SNMP module even has a dedicated section in [9] and was not tested during this evaluation.
`collectl` provides a iostat-like view of a the status of the local system. One of the susbystems that can be monitored is a Lustre filesystem. `collectl` can monitor Lustre servers (MDS, OSS) as well as clients (and it detects by itself which component of a Lustre system it is run on). By default, `collectl` will display performance data, such as the number of reads and write requests or the amount of data being read/written, but it can also provide more in-depth statistics, such as the RPC metabuffer stats or per OST data. [23] gives an overview on what data is available from `collectl` for a Lustre system. `collectl` is not a distributed system, it only reports the local statistics, but it is none the less quite a useful tool to quickly see what is going on on a particular Lustre component, and the information could potentially be used to be fed into a distributed system. The installation of `collectl` is very easy and no configuration is needed to use it.
LMT2[22] is the **L**ustre **M**onitoring **T**ool developed at LLNL. It is based on the cerebro[24] collection of cluster monitoring tools for the distributed data collection. cerebro uses so-called metrics (pluggable sensors) to collect data for various system components, such as the load or the memory usage. For Lustre, LMT2 provides cerebro metrics to collect the data on the various Lustre components and stores them into a MySQL database. For data display, LMT2 provides Java-based clients. The installation of LMT2 turned out to be quite time-consuming, as several components have to be compiled, installed and configured (cerebro, MySQL and LMT2 itself). The author was not able to get the whole LMT2 chain running with reasonable efforts.
From this experience, writing sensors (potentially using the data provided by `collectl`) for any monitoring system (Ganglia[25], Nagios[26] or Lemon[27]) seems to be a valid alternative.

2.6.3 Draining and Retirement: Data Migration

Draining a server (OSS) or an device (OST), i.e. moving all data to another location, for maintenance (e.g. kernel upgrades, firmware updates) or in order to retire a machines (e.g.

end of warranty) is a day-to-day operation for data management systems. The operations manual[9] includes a section on how to remove an OST (and hence an OSS) temporarily or permanently from the Lustre system. The procedure basically consists of the following steps:

1. Make the OST read-only, so that new no objects can be created here (access to existing objects still work, however);
2. Identify all files that have objects residing on the OST;
3. Copy all these files (this will create new objects somewhere else);
4. Move the new files to their original names (this will change the metadata information to point to the new objects);
5. 'Remove' the OST permanently (it cannot really be removed, only deactivated).

There is no server-to-server migration of data available: the copy step goes through a client, it's a normal `cp(1)` operation on file system level. In addition, this migration of data is not user–transparent: processes with open file handles on files that are migrated with the procedure just described will be impacted at some point as the file that used to write to does not exist anymore.

2.6.4 Quotas

Lustre supports distributed quotas for users and groups, and there is a whole chapter in the Operations Manual that covers their configuration. Since the underlying mechanism to implement quotas is planned to be changed in the near future, quotas have not been examined in detail during this evaluation.

2.7 Documentation

Lustre comes with an almost 600 pages strong operations manual which is available free of charge from[28]. This manual basically covers all available features and gives procedures for installing, operating, monitoring, tuning and debugging a Lustre installation. The manual was found to be a good resource of information.
For further details and advanced questions, the Lustre mailing list archives [29] and the list itself [30] should be consulted. Response time and quality of feedback were found to be to very good.

3 Additional Thoughts and Findings

3.1 Client–Server Coupling

Lustre is typically used in 'closed' environments, where 'closed' refers to the fact that the clients are integral part of the Lustre system (unlike AFS where the system basically consists

of the servers while the clients can be distributed all over the world and are not at all under the control of the persons administering an AFS cell). For these closed environments, a tight coupling of clients and servers is not so much of a problem. Some of the use cases relevant for CERN it may be, however.
One example of the client–server coupling is the recovery case: when a Lustre server goes down and comes up again[4], it goes into the so-called **recovery** mode, where it recontacts the known clients *in serial order* and asks them to replay their requests. If clients are not recoverable, due to a shutdown for instance, the server waits for a timout to complete before evicting the client and aborting recovery, i.e. droppping all the requests that may still be there on **other** clients. This behaviour may result in data loss. Aditionally, while in recovery mode, *all* new connections from clients are rejected. In more open environments, this strict recovery behaviour, which has been established in order to ensure on-disk consistency of Lustre data, may block access to an OSS because of a meanwhile shut down client, and may lead to data loss (since clients with higher transaction number that actually could replay their requests are never asked to do so, since recovery was aborted before it was their turn). With version 1.8 version-based recovery (VBR) and delayed recovery (as a preview, switched off by default) have been introduced in order to alleviate the situation. VBR tracks changes in inode version numbers in order to allow more clients (potentially working on independent objects) to be re-integrated into the cluster. Hence, the risk of data loss is reduced and the coupling is less tight. Delayed recovery allows clients to be re-integrated even when the server has already finished the recovery. This is also a step into loosening the connection between clients and servers in a Lustre system. It is questionable, however, if this is sufficient for Lustre access from users' notebooks, for instance.

3.2 Users' Freedom

In day-to-day operations, it is desirable for the file system admin to control *how* users can use the system. For instance, control over the amount of space or bandwidth, or over the physical placement of data should be in the admin's hands. Lustre supports user and group quotas. At least two points, however, have been identified where Lustre users have too much control over the system: OST pools and striping. Pools are similar to service classes: pools are sets of OSTs that are intended to be used for a specific purpose, e.g. high-end redundant OSTs for user home directories, or huge and cheap disk arrays for scratch. Pool assignment of directories is only advisory, however: users can store their data wherever they want. Similarly, admins cannot enforce striping policies on certain directories, e.g. no striping on home directories with small files, in order to ensure decent performance or availablity to users. With Lustre, users need to adhere to certain policies in order to get the best out of the system and it should be possible for the administrator of the system to enforce these policies to a certain degree.

[4]This is only one example. There are more cases when recovery mode is entered, see the Operations Manual for more details.

3.3 Roadmap

Although a disclaimer on the roadmap states that the roadmap presents no comittment, some features have become 'moving targets', while others have been dropped completely. The support for strong authentication is an example for the first case, the development of replication an example for the latter. It is understood that priorities for certain features can change, development processes can be rearranged, the complexity of certain problems can be underestimated or that the 'market pressure' enforces publishing aggressive roadmaps. From a user's point of view, however, predictability is an important criteria when decisions whether or not to use certain technologies have to be taken or roll-outs have to be planned.

3.4 Lustre's Future

Due to the pending acquisition of SUN by Oracle, the future support, development and direction of Lustre is not completely clear at the moment. On the other hand, Lustre is open source, it has a large community and even if Oracle decides to drop it, it is not unlikely that another company (like ClusterStor, a company recently founded by Lustre's former main architect Peter Braam) picks it and runs it in a similar way Cluster File Systems did before. Another possibility would be that Lustre becomes an open-source project supported by its user community, just like it is the case for OpenAFS. Since Lustre's user community includes some of the larger research labs in the U.S. and since there is currently no other file system available that could compete with Lustre's performance and scalability features, it is very unlikely that even if Oracle decides to not continue development and support, the efforts around the Lustre filesystem would stop.

4 Wish List

This section discusses some potential extensions of Lustre that would improve Lustre's applicability for the above–mentioned use cases.

4.1 Completion of Strong Authentication

The completion of the support for Kerberos authentication is mandatory for the home directory and project space use cases. As discussed in section 2.1, this requirement will be fulfilled with one of the 2.X releases due in late 2010, early 2011.

4.2 User-transparent Data Migration

As discussed in section 2.6.3, Lustre is lacking the ability to migrate data in a transparent way. In an environment where the filesystem has to be available 24/7 and no downtimes are acceptable, this ability is mandatory.

4.3 File Replication

File replication would be a desirable feature as it may provide a solution for the lack of user-transparent data migration, see section 4.2, but would also be an opportunity to combine availability and performance. If replicas of a file would allow for different striping policies, there could be striped replicas for performance and not-striped replicas for availability. Managing replicas on the filesystem level, rather than on the underlying storage level, seems the better approach from the manageability point of view.

4.4 More System Control

Section 3.1 stated that some of the system settings, such as striping policies or pools, have only an advisory character. It would be, however, desirable that users cannot bypass these settings, but are forced to follow the settings that the admin decided on (unless the admin gives the user the freedom to change them, of course). Especially in the case of pools, i.e. the assignment of certain directories to certain OSTs, a user should not be able to change the settings as chosen by the system administrator.

4.5 Outsourced Privileges

When dealing with a larger user community that may be structured into groups, it would be desirable to have the ability to outsource certain functionality to privileged power users. This would offload the core admin team and redirect certain administrative tasks to the groups (e.g. the granting of access to certain directories).

5 Summary

Table 1 summarizes Lustre's operational qualities for version 1.8.0.1 as found by this evaluation. Regarding the use cases at CERN, Lustre is an interesting option for the analysis use case, where it fulfills all the criteria. Due to the poor support for day–to–day operations in an open 24/7 environment however – especially the lack of support for user–transparent data migration –, Lustre in its current state is not a valid option for the other use cases and hence not for storage consolidation at CERN.
However, since there are efforts to add support for some of the features, such as the HSM extension, and others are almost finished and will become available with one of the next releases, e.g. the support for strong authentication, this assessment may change, and hence the developments around the Lustre file system should also be followed in the future.

Feature	Available	Comment	Home	Project	Analysis	HSM
Strong Authentication	no	Kerberos implementation not complete, production version expected in late 2010 or early 2011.	x	x	x*	x
Backup and Recovery	yes	LVM snapshots of MDT plus standard backup mechanisms for user files.	x	x		
Small Files Performance	partially	For non-striped directories, as objects are created at file creation time on all OSTs of a strip. No mixing of small and huge files in the same directory.	x	x		
Fault Tolerance	yes	Works for both metadata and user data. Relies on external tools. Some support built-in.	x	x		
HSM Functionality	no	Under active development. First protoype expected in 2010.			x	x
File Replication	no	Dropped from roadmap.				
WAN Access	no	May become possible with Kerberos and cross-realm setups.	x*	x*		
Monitoring	partially	No built-in support. External tools available, but difficult to set up.	x*	x*	x*	x*
Installation	partially	No built-in support, but not really required as setup easy. Some additional tools exist (mostly simple scripts).				
Data Migration	no	Only copy and move via a client. Not user-transparent. Makes draining/retirement of OST impractical.	x	x	x*	
Quotas	yes	Per user and per group, underlying mechanims about to change.	x	x	x	
Strong control for admins	no	Examples: advisory pools and striping.	x*	x*	x*	
Privilege outsourcing	no		x*	x*	x*	
Documentation	yes	Manual and excellent community support.				
POSIX compliance	yes	Exceptions: atime updates and flock/lockf calls.	x	x	x	

Table 1: Lustre's operational characteristics along with the requirements for the indvidual use cases, marked by x. Additionally, features that would be nice to have are marked by x*.

A The PPS instance

A pre-production (PPS) instance was set up to acquire hands-on experience with Lustre beyond a testbed environment. The hardware used for the PPS instance included

- MDS: 2x Dell PowerEdge M600 Blade Servers with 2x QuadCore Intel(R) Xeon(R) CPU E5410 @ 2.33GHz and 16GB RAM;
- MDT: 2x Infortrend iSCSI arrays with 16x 500GB SATA disks in a RAID-10;
- OSS/OST: 4x Pyramid servers with 1x QuadCore Intel(R) Xeon(R) CPU L5420 @ 2.50GHz and 16GB RAM and 24x directly attached 1TB SATA disks;
- Clients: Dell PowerEdge M600 Blade Servers with 2x QuadCore Intel(R) Xeon(R) CPU E5410 @ 2.33GHz and 16GB RAM;
- Network: GigaBit Ethernet (shared iSCSI MDT with a seconds interface on a private LAN).

The iSCSI arrays are connected to the 2 MDSs in a multipath setup, i.e. each server sees each of the two arrays via two paths. Via multipathd and MD (mirroring) the iSCSI arrays are accessible to the servers. This way, a fully redundant setup has been created that can tolerate the loss of any component in the system. All Lustre servers run an Scientific Linux CERN (SLC5) version of Linux with a Lustre patch kernel version 2.6.18-128.7.1.el5. This kernel is self-compiled in order to enable iSCSI support (the version of the IB package as delivered from SUN was not working for us). The OSSs have 10 OSTs each. The Lustre versions used during this evaluation are '1.8.0.1' and '2.0alpha2'. No tuning had taken place, all parameters were set to their default values.

B Distributed Filesystem Checklist

During this evaluation the following requirements for distributed file systems have been identified:

- Support for strong authentication (e.g. Kerberos) and ACLs
- Support for life cycle management, this includes
  - adding/removing space (servers/partitions) to/from the system shall be done in a way that is completely transparent to the user; no system downtimes shall be required for this; in order to drain a server or the associated storage device, data shall be moveable between servers rather than copying data through a client;
  - the file system should come with tools for setup, configuration, debugging, management, and monitoring;
- Support for fault tolerance, this includes

  - the system shall have mechanisms for fail-over for data and meta-data in order to increase availabilty;
  - the system shall not have a single point of failure or it shall at least be possible to set the system up in a way that avoids a single point of failure;

- Strong system control for administrators, this includes
  - the system shall provide quotas for users and/or groups;
  - the system shall allow the administrator to control where data is stored, i.e. physically on which servers/partitions;
  - if the system supports striping the system shall allow the administrator to set striping policies;
  - if the system provides mechanisms such as pools, service classes, striping or quotas the user shall not have a possibility to override the settings of the administrator;

- Support for backup, in particular
  - to be able to deal with hundreds of millions files that need to be backed up once per day without a significant impact on the usability of the system;

- Support for an HSM interface, i.e.
  - the ability to (transparently) migrate/recall files from and to a tertiary (tape-based) storage system, such as TSM or HPSS;

- Support for file replication on file system level, which can be used for
  - load sharing: readers (and writers) may be redirected to the various replicas in order to distribute the load evenly over the servers;
  - availability: several copies of the data increase the availibity in case one of the servers/partitions becomes unavailable; clients failover to another copy;
  - data migration: increasing the number of replicas will create new copies on other servers/partitions, the original ones can be removed and replica count decreased;

- Support for privilege outsourcing: for installation with multiple user groups and/or project the system should support the possibility for the administrators to outsource certain privileges to project administrators in order to reduce the load on the core admin team;

- Support for WAN access: the system should allow for secure access over a WAN.

References

[1] http://hepix.caspur.it/storage

[2] http://www.openafs.org

[3] http://xrootd.slac.stanford.edu

[4] http://cern.ch/castor

[5] http://www-01.ibm.com/software/tivoli/products/storage-mgr

[6] The Data Storage Management (XDSM) API specification is available at http://www.opengroup.org/onlinepubs/9657099/

[7] Andreas Dilger, Sr. Lustre Engineer, during the Lustre User Group Meeting 2009.

[8] Josephine Palencia, Pittsburgh Computing Center, Private Communication.

[9] Lustre 1.8 Operations Manual, available from http://www.lustre.org.

[10] See the Changelog bug at https://bugzilla.lustre.org/show_bug.cgi?id=15699

[11] J. Katcher: "PostMark: A New File System Benchmark", available from http://communities.netapp.com/servlet/JiveServlet/download/2609-1551/Katcher97-postmark-netapp-tr3022.pdf.

[12] http://www.codemonkey.org.uk/projects/fsx

[13] http://www.linux-ha.org

[14] http://www.clusterlabs.org/wiki/Pacemaker

[15] http://www.openais.org

[16] http://www.cea.fr

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

[18] SAM/QFS on OpenSolaris: http://hub.opensolaris.org/bin/view/Project+samqfs/WebHome

[19] http://wiki.lustre.org/images/b/b5/Wednesday_Tools.pdf

[20] http://quattor.web.cern.ch

[21] `collectl` is available from sourceforge http://collectl.sourceforge.net

[22] LMT2 is available from sourceforge http://lmt.sourceforge.net

[23] Details on `collectl` for Lustre http://collectl.sourceforge.net/Lustre.html

[24] https://computing.llnl.gov/linux/cerebro.html

[25] Ganglia website: http://ganglia.sourceforge.net

[26] Nagios website: http://www.nagios.org

[27] Lemon website: http://lemon.web.cern.ch

[28] Lustre website: http://www.lustre.org

[29] The Lustre Discuss mailing list archives are available at http://groups.google.com/group/lustre-discuss-list

[30] The Lustre Discuss mailing list is available at lustre-discuss@lists.lustre.org