Charlie Carroll
Branislav Radovanovic
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120
USA
Main Phone: 651-605-9000
Fax: 651-605-9001

Abstract:

Lustre[1] is an open source, high-performance, distributed file system from Cluster File Systems, Inc. designed to address performance and scalability for storage and I/O within Cray supercomputers. A considerable number of high-performance parallel Lustre file systems for supercomputers and commodity compute clusters are available today. This paper will discuss Cray's history with Lustre, the current state of Lustre, and Cray's Lustre plans for the future. We will also provide recent performance results on a Cray XT3 supercomputer, showing both scalability and achieved performance.

Keywords:
XT3, Lustre, RAID

1 Introduction

There are a number of ways to classify parallel data access and distributed file systems. However, distributed file system is a generic term for a client/server file system architecture where the data resides on a remote host (device). Generally, distributed file systems could be classified into three main categories: client-server-based network file systems (like NFS and AFS), Storage Area Network (SAN) file systems, and parallel file systems.

NFS is the most common distributed file systems today. It is easy to deploy and manage, works across a most operating systems, and mid-range to high-end NFS storage generally has advanced data management capabilities, including snapshots. However, NFS is best used in creating an external pool of storage used by Cray machines and other servers for shared, managed data.

Shared-disk file systems usually rely on Fibre Channel Storage Area Networks to provide access to a shared disk or Logical Unit Number (LUN). The file systems that could fall into this category are the Red Hat Global File System (GFS), IBM’s General Parallel File System (GPFS), SGI’s CxFS, and Terrascale’s Terragrid software.

Parallel file systems rely on I/O nodes to provide disk access for both data and metadata to the clients from the rest of MPP or cluster system. The main advantage of parallel file

systems is that they provide excellent scalability, the ability to distribute large files across a number of I/O nodes, and a global name space. Generally, for metadata operations parallel file systems utilize dedicated Meta-Data Servers (MDS). Actual data transfers, once file information and locks are obtained from the MDS, occur directly between clients and I/O nodes (in case of Lustre, Object Storage Servers – OSTs). The file systems that could fall into this category are Lustre, Argonne's Parallel Virtual File System (PVFS), IBRIX Fusion, and others.

Lustre is an open source file system developed and maintained by Cluster File Systems, Inc. The largest Lustre installation is on Red Storm system at Sandia National Laboratories (SNL). Cray XT3 system [2] is the commercialized version of the Red Storm system. This paper discusses Lustre file system performance and methods used to measure I/O performance on a Cray XT3 supercomputer.

1.1. I/O Tests Used

As the popularity of commodity compute clusters and MPP supercomputers has grown in recent years many new I/O benchmarking tools have been developed and adapted to measure various aspects of parallel I/O file performance. There are many different aspects of a file system that could be evaluated. However, we concentrated on collective I/O since it is a common I/O pattern in which IO is performed by all nodes taking part in a job. This is common when a single file is used by the entire job. The Lustre architecture will utilize the aggregate bandwidth available. In well-tuned installations this should be close to the sum of the bandwidth offered by all I/O servers (OSTs). The bandwidth available to individual clients is usually limited not by the I/O servers, but by the network and attainable client file system performance. To minimize the impact of the back-end RAID (Redundant Array of Independent/Inexpensive Disks) system latencies (such as disk drive mechanical delays) and to evaluate true Lustre file system performance, lmdd [3] was chosen. Also, we looked into maximum write and read performance for the large number of sequential I/O streams using the well-known IOR benchmark [4], which should be demanding on the backend RAID subsystem.

The lmdd test is a simple I/O benchmark based on Unix dd command that measures sequential and random I/O. It does not use the Message Passing Interface (MPI), which makes it suitable to be run on both Catamount nodes and Linux (login) nodes. Catamount [5] is lightweight microkernel running on compute nodes.

The IOR test is a parallel file system benchmark developed by the Scalable I/O Project (SIOP) at Lawrence Livermore National Laboratory (LLNL) [6]. This program performs parallel writes and reads to and from a file using MPI-IO therefore, it could be run only on Catamount nodes since login nodes do not support MPI. Generally, IOR is used to generate three main file access patterns: file per process, segmented access, and strided access pattern. In our tests we used segmented access where each process writes (reads) to its own segment of a large shared file.

1.2. Lustre configuration on Cray XT3 supercomputer

Users are free to deploy Lustre (open source) on the hardware and storage of their choice. The Cray XT3 architecture consists of data RAIDs connected directly to I/O blades running Object Storage Targets (OSTs). I/O blades reside on the high-speed Cray interconnect. The Lustre file system manages the striping of file operations across OSTs (RAIDs). This scalable I/O architecture enables customers to configure the Cray XT3 with desired bandwidth by selecting the appropriate number of OSTs and RAIDs. It gives users and applications access to a filesystem with filenames that are uniform and global across all compute nodes.

Most of the storage solutions work well with small numbers of clients however, very few work well once the client count increases. A stable performance characteristic is typical issue of concern when many clients create files in a single directory. The ability of the file system to handle a flood of I/O or metadata requests that can be initiated by all clients

simultaneously is very important. The Lustre library (liblustre.a) is linked directly into applications running on the compute Catamount node. Data moves directly between applications space and the Lustre servers (OSTs) minimizing the need for additional data copying.

The Cray XT3 combines the scalability of Catamount Lustre clients with the I/O performance scaling to the aggregate bandwidth available from all OSTs. This storage solution allows for growth through additional OSTs and resizing the file systems. Lustre file system does not introduce limitations on file sizes or file system sizes. The limit is available physical space or up to 2^{64} blocks for both, file and file system size, whichever is less.

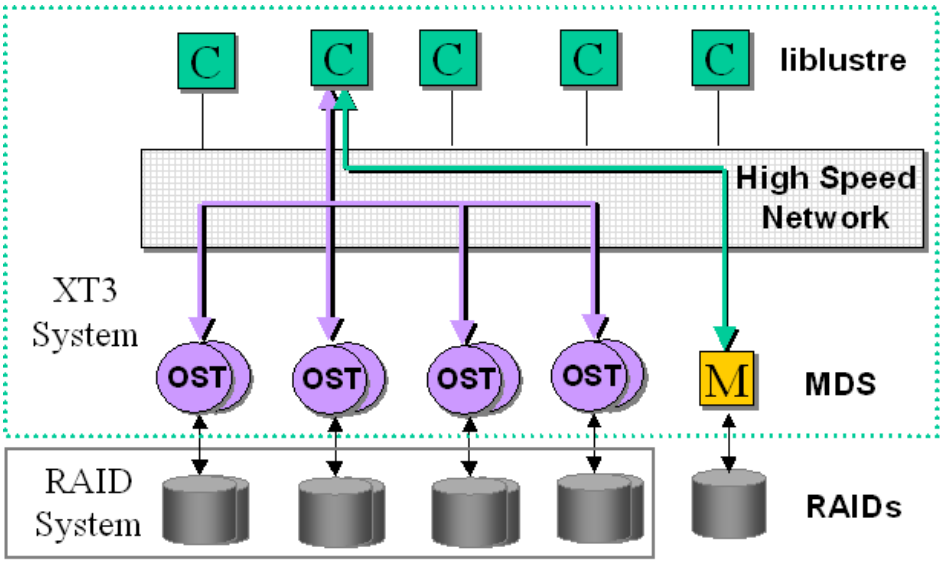

Figure 1.1

Figure 1.1 illustrates the XT3 system Lustre I/O architecture. File data is stored as storage objects on the OSTs while metadata is stored on a metadata server (MDS). MDS and OSTs currently run 2.4 Linux kernel. Catamount compute nodes use liblustre to access both metadata server and OSTs over the high-speed Cray network using portals. Back-end RAID systems are connected to the MDS and OSTs using point-to-point Fibre Channel links.

2 Related Work

A lot of work has been done in the area of Lustre performance troubleshooting and testing. However, Lustre implementation on Cray XT3 systems is slightly different than on the Linux platform. The Cray XT3 supercomputer is optimized for CPU intensive workloads utilizing Catamount (lightweight microkernel). Since, Catamount supports a reduced number of system calls, Lustre client is implemented in liblustre library, which is statically linked in at compile-time. Liblustre passes Lustre I/O calls via portals to

appropriate OSTs utilizing Lustre RPC and Remote Direct Memory Access (RDMA) facilities. This specific implementation does not allow I/O buffering on the Lustre client (Catamount) node. Therefore, I/O requests are sequentially executed. Average I/O delay (protocol overhead) associated with Lustre is approximately 1 millisecond for read and about 3 milliseconds for write requests. However, the Lustre client implementation is going to change significantly with Linux on compute node in future v.1.4.xx software releases, which is implemented using 2.6 Linux kernel. It is expected to have latencies significantly reduced for writes due to I/O buffering on the client (compute) node.

If an I/O operation, on a XT3 system, is striped across multiple OSTs, each I/O request is segmented by liblustre to 1MB (which is Lustre default RPC transfer size) and all the requests are sent to the appropriate OSTs simultaneously. Therefore, for large size I/O requests to a big shared file, it would be advantageous to configure the Lustre file system with as large number of OSTs as possible. However, if a large number of smaller files are accessed by an application running on number of Catamount nodes, then the stripe count could be set to 1 or 2. We observed no I/O buffering on the OST side either. All I/O requests are sent to the back-end RAID. Hence, to improve write performance write-back cache should be enabled although the command complete status is returned to the client before the data is actually written to the media.

We traced and analyzed I/O timings and bandwidth from the user space on a Catamount node all the way to the back-end RAID controller. How much I/O bandwidth an application running on a Catamount node is able to utilize depends on the size and the type of the I/O and Lustre file system configuration.

3 Lustre File System Performance

The main goal was to test Lustre file system performance minimizing the influence of the back-end storage (RAID).

3.1 Lustre and back-end RAID system

A RAID system is an electro-mechanical system used to store (retrieve) data to (from) the permanent storage media. Due to the mechanical nature of the disk drives at the back-end of the RAID controller, I/O timings could be very unpredictable. Generally, an average I/O performance could be considered but even that performance could deteriorate over time due to a number of various factors such as file fragmentation, disk drive grown defects, etc.

With this is mind among the other available tests, lmdd, was chosen and test parameters set to measure Lustre file system performance, as objectively as possible, minimizing the impact of the disk drives. Ideally, to test file system performance we would like to send file I/O to a solid-state disk drive, which was not available to us. Ultimately, file system performance does not depend on the LBAs (Logical Block Address) where the data are read from or written to considering the liblustre overhead associated with each transaction (1 msec for reads and 3 msec for writes). Thus, writing and reading

sequentially from a single file should mask mechanical and other latencies of the back-end RAID controller.

As mentioned previously, Lustre file system performance depends on the Lustre configuration. We run I/O benchmarking tests for a range of I/O request sizes (4kB, … 16MB), number of clients (1, … 256 clients), and number of OSTs (1, … 11 OSTs). I/O performance depends on whether the application uses single large shared file or large number of smaller files. For optimal performance, Lustre file system should be configured differently changing stripe count parameter.

For accessing large number of smaller files the working directory should be set to stripe count of one or two while for accessing small number of large shared files the stripe count should be as large as possible. In either case stripe size should be set to 1MB since this is the largest Lustre RPC transfer size on XT3 system. The lfs commands in the following example are used to set or view file/directory properties:

lfs setstripe /lus/directory-name/file-name 1048576 0 1
lfs find -v /lus/ directory-name/file-name

3.2 Overview: RAID 3 vs. RAID 5

RAID Level 3 also known as RAID 3 (Figure 3.1) utilizes byte-level striping across date disks with dedicated parity drive. Hardware accelerator may be used in some high-end systems to improve performance (calculate parity for writes and reads). Exact number of bytes sent in each stripe depends on the particular implementation. On some RAID 3 systems stripe size could be set to match specific I/O access pattern. Any single disk failure in the array can be tolerated data could be recalculated using parity information.

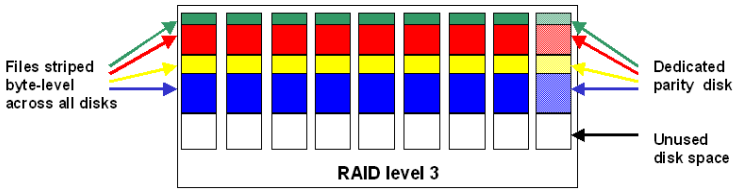

Figure 3.1

RAID Level 5 or simply RAID 5 (Figure 3.2) is the most common RAID level used today. It uses block-level striping with distributed parity. The stripe size could be set to a desired value, usually between 64kB and 256kB. High-end systems use hardware accelerator to calculate parity for writes and sometimes for reads.

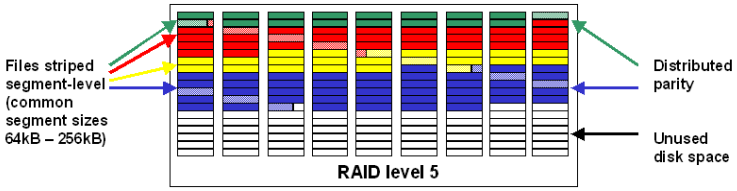

Figure 3.2

Distributed parity algorithm is used to write data and parity blocks across all the drives in the array eliminating the bottleneck associated with dedicated parity drive. RAID 5 architecture tolerates any single disk failure in the array; data is recalculated using parity information.

4 Lustre performance test results

In this chapter we will discuss the Lustre peak performance for sequential I/O, multiple sequential streams I/O access patterns, Lustre scaling, and achieved I/O bandwidth using RAID 3 and RAID 5 system.

4.1 Single OST (RAID 3) Lustre sequential I/O performance

Figures 4.1 and 4.2 show Lustre read and write performance to a single OST from one to 256 Catamount clients while varying I/O request sizes from 4kB to 16MB. Indeed, read performance graph (Figure 4.1) shows almost ideal behavior.

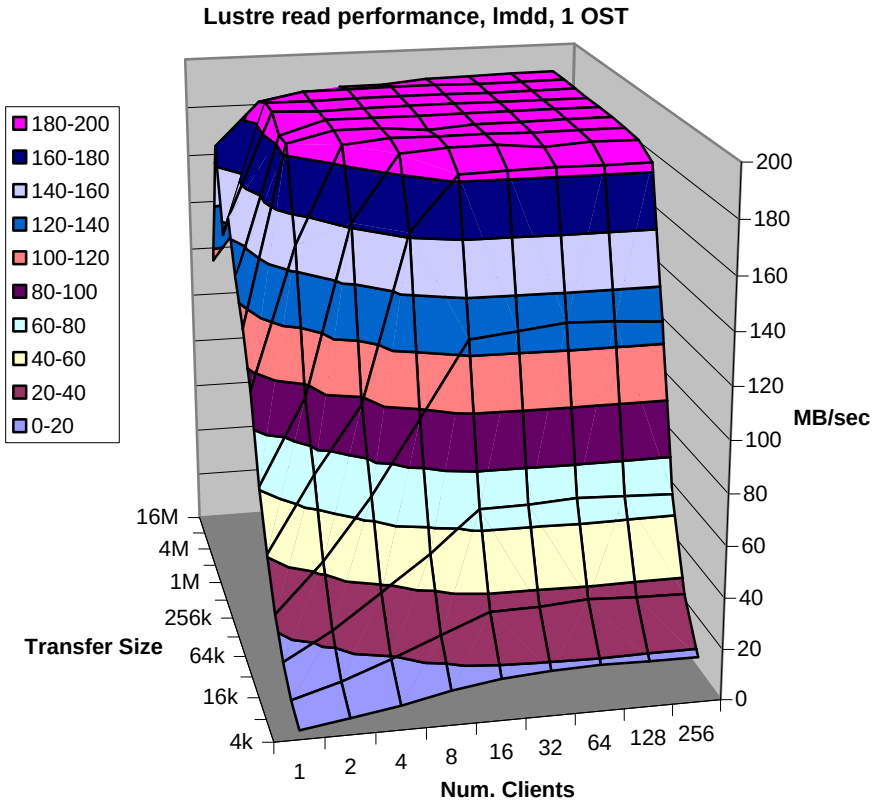

Figure 4.1

Back-end RAID 3 system is lightly loaded, receiving I/O requests from only one out of eight available Fibre Channel ports. It is needed as few as two Catamount clients to saturate the back-end Fibre Channel link.

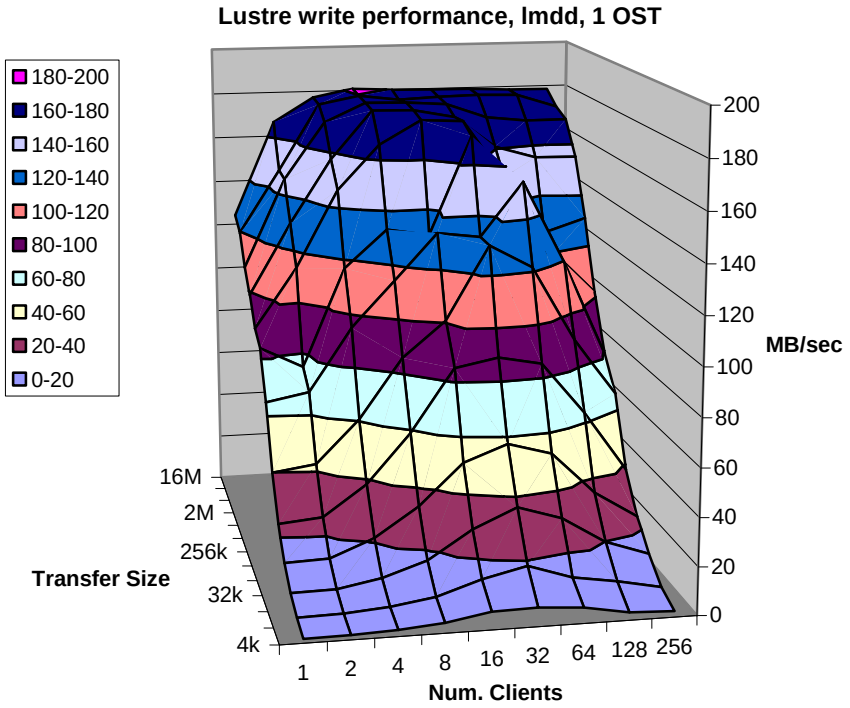

Figure 4.2

The theoretical bandwidth to a single OST is 200 MB/sec however, we achieved 188 MB/sec for reads and 180 MB/sec for writes. Two to four Catamount clients could easily saturate an OST and Fibre Channel link.

4.2 Eight OSTs (RAID 3) Lustre sequential I/O performance

Figures 4.3 and 4.4 demonstrate Lustre read and write bandwidth to eight OSTs using the same configuration and I/O sizes as in the previous tests (Figures 4.1 and 4.2). The theoretical bandwidth is 1600 MB/sec. Yet, I/O throughput for reads achieved over 93% of the theoretically available bandwidth mostly due to cache hits on the back-end RAID controller. All clients read the same file therefore, only the first read request goes to the disk and every subsequent read is the RAID cache hit. This is perfectly acceptable since the file system on the Catamount or OST nodes does not do any caching. However, that might change once transition to Linux 2.6 kernel is made. Write performance achieved slightly above 1300 MB/sec, which is very reasonable. We have to remember that overhead associated with write requests is three times higher than that for read requests. Also, every write request is preceded and followed by the ext3 file system journaling information. All the clients write to the same file overwriting the same file many times. Therefore, the disk seeks are kept at bare minimum even though every write request goes to the back-end disk. Since, every client's write request is accompanied by two journaling write requests, that becomes significant load for large number of clients (more than 64). The performance is noticeably affected for the write request sizes between 64kB and 1MB.

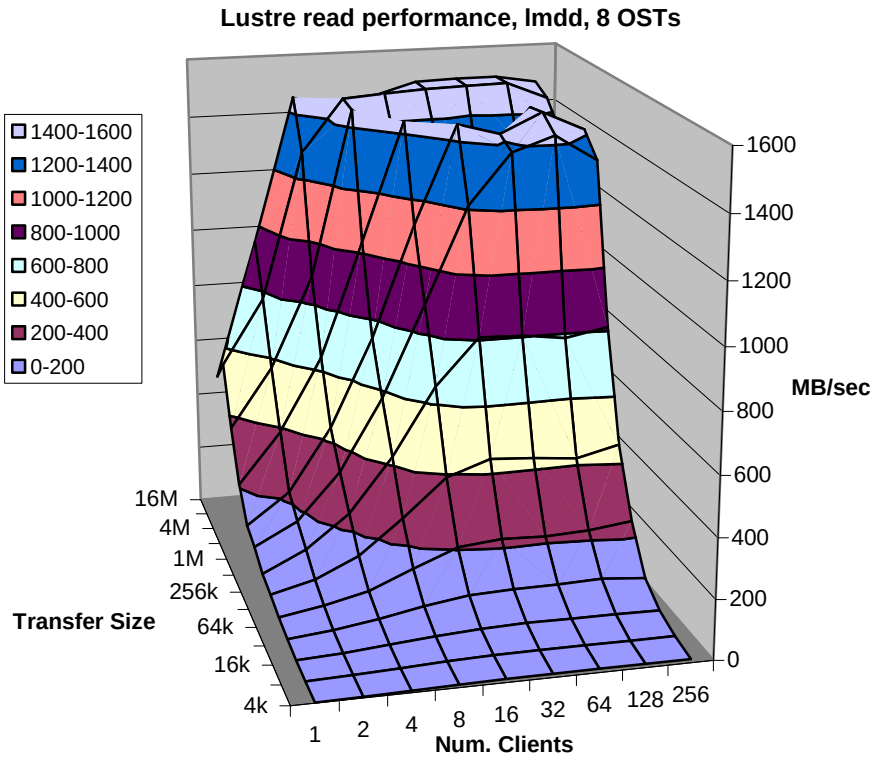
Lustre read performance, lmdd, 8 OSTs
1400-1600
1200-1400
1000-1200
800-1000
600-800
400-600
200-400
0-200
1600
1400
1200
1000
800
600
400
200
0
MB/sec
16M
4M
1M
256k
64k
16k
4k
Transfer Size
1
2
4
8
16
32
64
128
256
Num. Clients

Figure 4.3

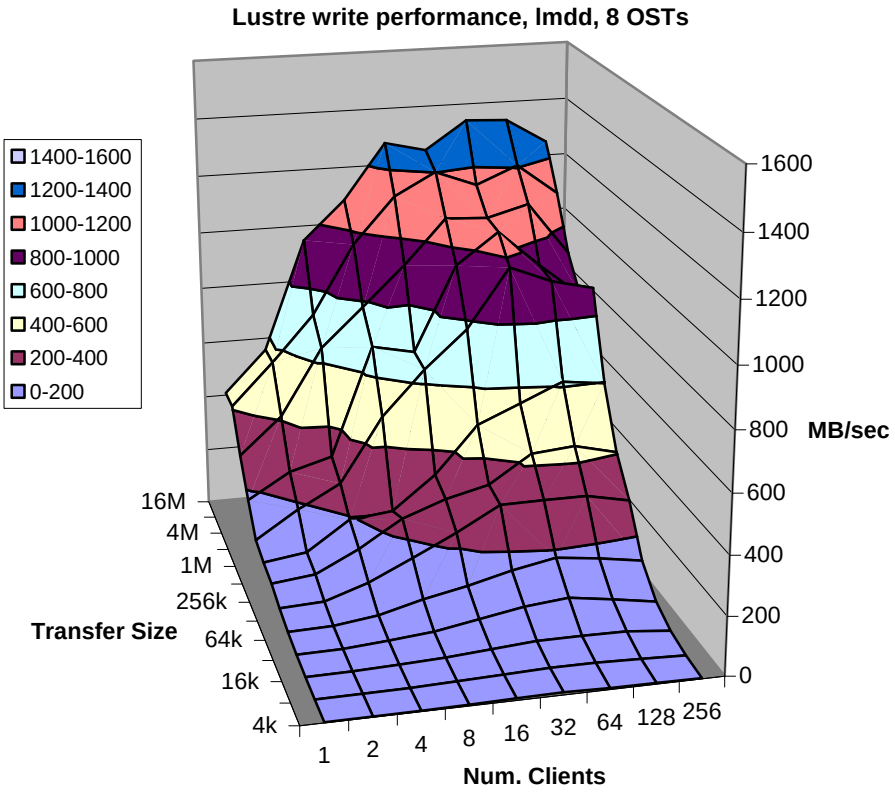

Figure 4.4

4.3 Eleven OSTs (RAID 3) Lustre sequential I/O performance

Figures 4.5 and 4.6 display Lustre read and write throughput for 11 OSTs (stripe count). The same test setup was used in this case as well. The theoretical throughput in this case is 1600 MB/sec. However, we slightly exceeded 1500 MB/sec, which is better than 90%. Write performance results were consistent with the previous write test for 8 OSTs results (Figure 4.4); we achieved more than 1300 MB/sec.

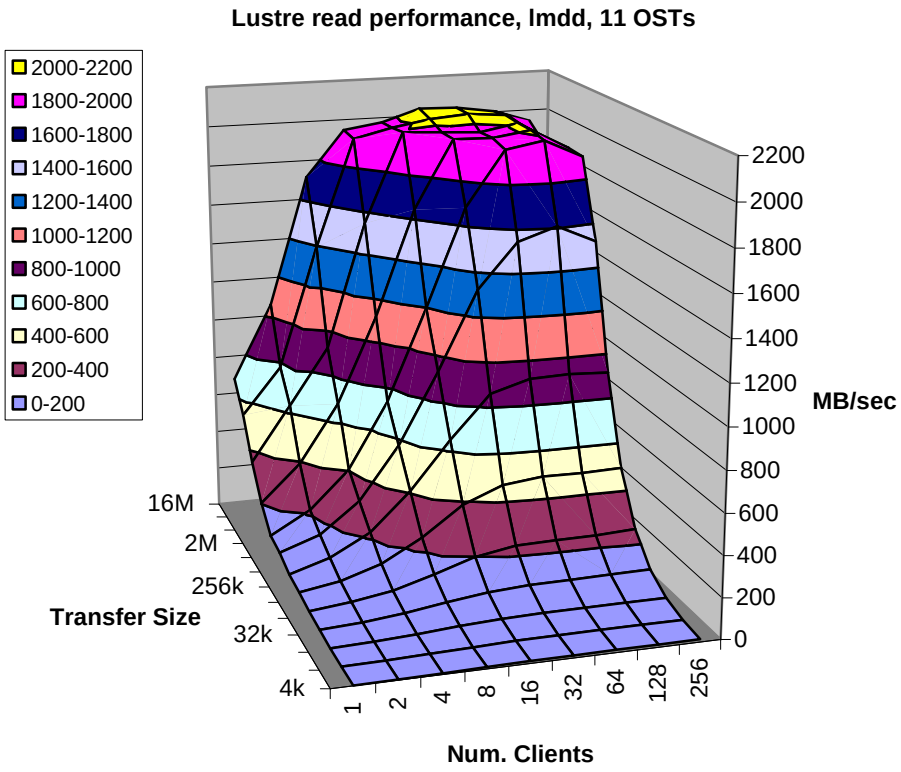

Figure 4.5

In general, it could be said that the Lustre file system performs satisfactory and scales well with the number of OSTs. We consistently exceeded 90% of the theoretical bandwidth for the reads while the writes were about 10% slower than the reads. As we mentioned before the 10% performance degradation could be attributed to the ext3 file system overhead associated with writing the ext3 journaling information and segmenting every write further to 128kB segments before sending it to the back-end RAID. We changed the Fibre Channel driver parameter to allow greater write sizes. However, we managed to increase write sizes only to 256kB. It appears that further Fibre Channel request size increase could not be done easily. Nevertheless, it has to be mentioned that all the tests were done on XT3 OS version 1.3.xx and we expect to see improvement in I/O performance once 2.6 Linux kernel based release 1.4.xx becomes available.

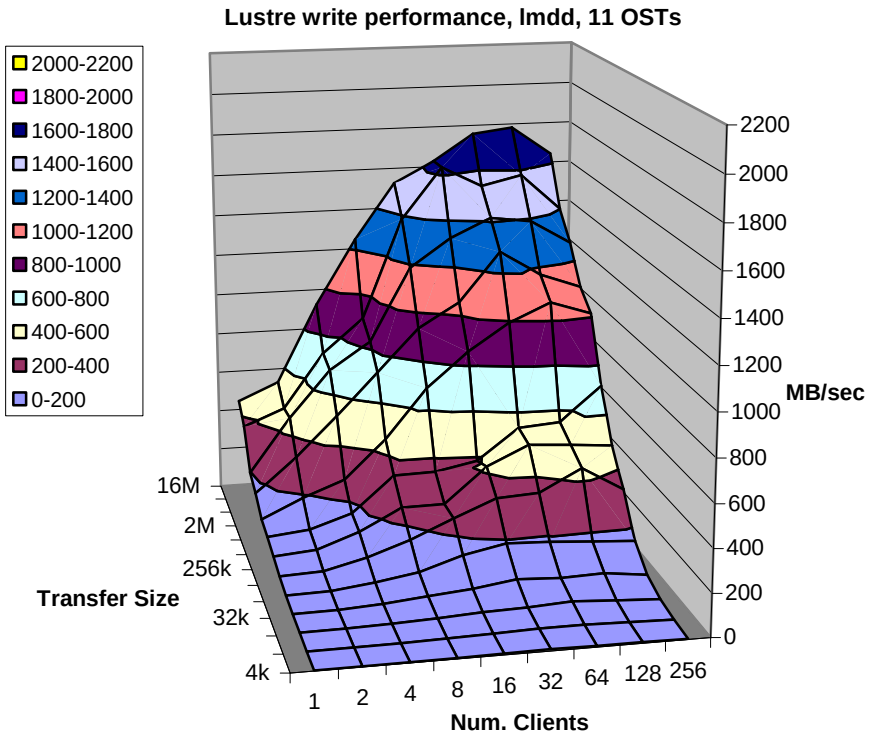

Figure 4.6

4.4 Eleven OSTs (RAID 3) Lustre random I/O performance

So far all the tests were executed using back-end storage system configured for RAID 3. This should maximize I/O performance for the large sequential I/O requests. However, it would be interesting to see the Lustre performance and the influence of RAID 3 back-end storage system for large number of sequential I/O streams. This induces great deal of stress to the back-end RAID controller.

Figures 4.7 and 4.8 present Lustre file system I/O performance for 11 OSTs. However, this time we were testing write and read performance using IOR test suite that generates multiple streams of sequential read and write requests. Actually, in this test, every Catamount client reads and writes to its own part of the shared file. This form the RAID point of view amounts to a quasi-random I/O traffic, when the large number of clients generates the I/O requests. It is interesting that while in the pervious, lmdd, test case we were exceeding 2 GB/sec for reads and got close to 1.8 GB/sec for writes, the peak performance for pseudo-random writes was slightly below 1.6 GB/sec. Apparently, even though the write-back cache mostly compensated for the “randomness” of the write requests we still see some performance impact due to mechanical latencies associated with the back-end disk drives.

However, quasi-random reads have even more dramatic impact on the overall I/O performance. While we were reading mostly from the cache in the previous example (Figure 4.5), this time (Figure 4.8) if we read by more than 16 clients the peak

performance drops from slightly above 1.4 GB/sec to 600 - 800 MB/sec on average. This test is done using the factory default read prefetch setting of x8. This dramatic drop in performance could be attributed to read cache misses (trashing). The actual data have to be read from the disk before it could be sent to the Lustre client. Read look-ahead algorithm could not compensate for the “randomness” of the read requests. Thus, it could be concluded that the aggregate bandwidth for the back-end storage system with its current configuration is about 800 MB/sec. This is about 50% lower than the theoretical bandwidth for that RAID. It is well know fact that RAID 3 has good but not excellent random read performance and this test exposed its weakness.

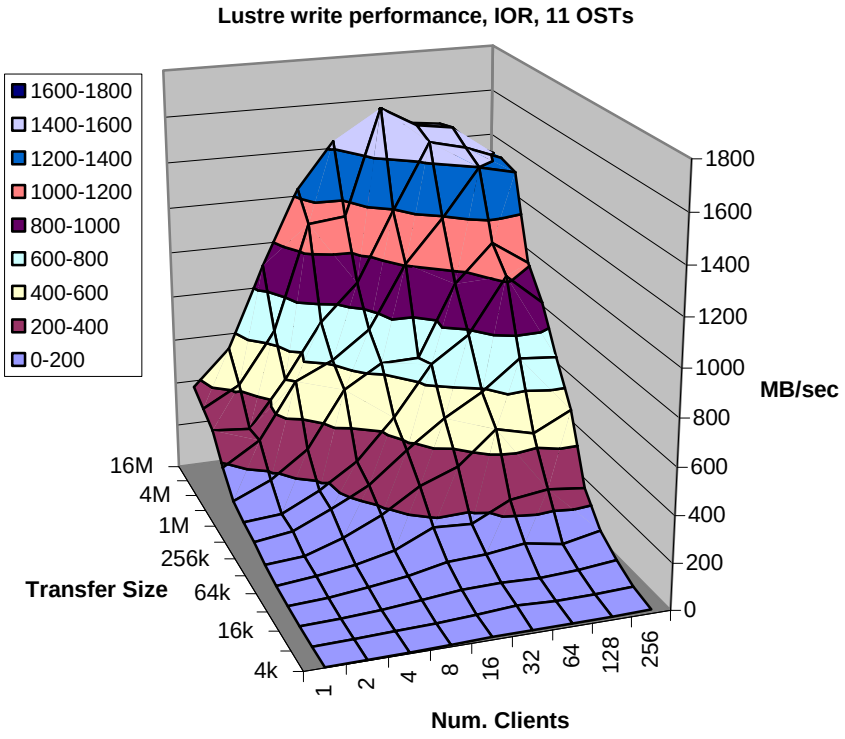

Figure 4.7

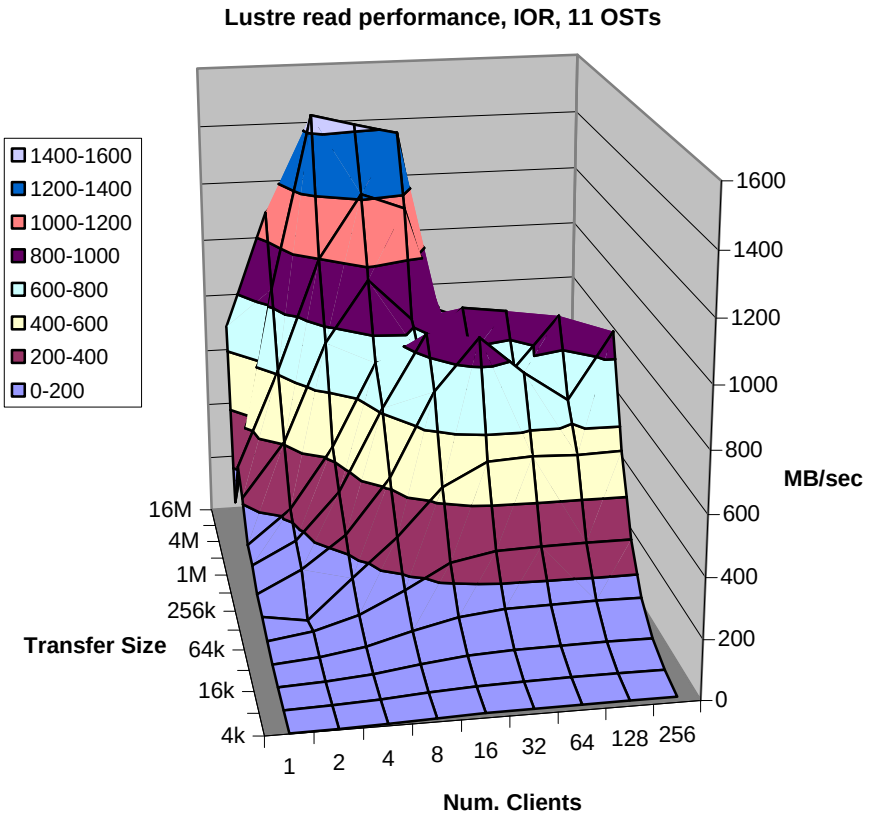

Figure 4.8

4.5 Tuning RAID 5 performance

Initial testing revealed that factory default settings on RAID 5 box were not adequate for the XT3 Lustre file system. Even though, I/O performance for read requests (Figure 4.9) performed satisfactorily, writes (Figure 4.10) were more than 50% slower than the reads. Therefore we had “tweak” the controller settings to improve write performance.

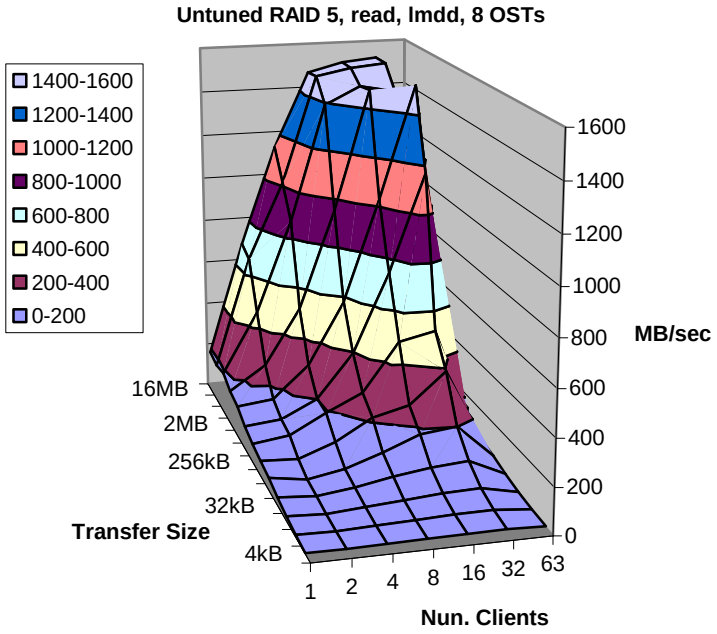

Figure 4.9

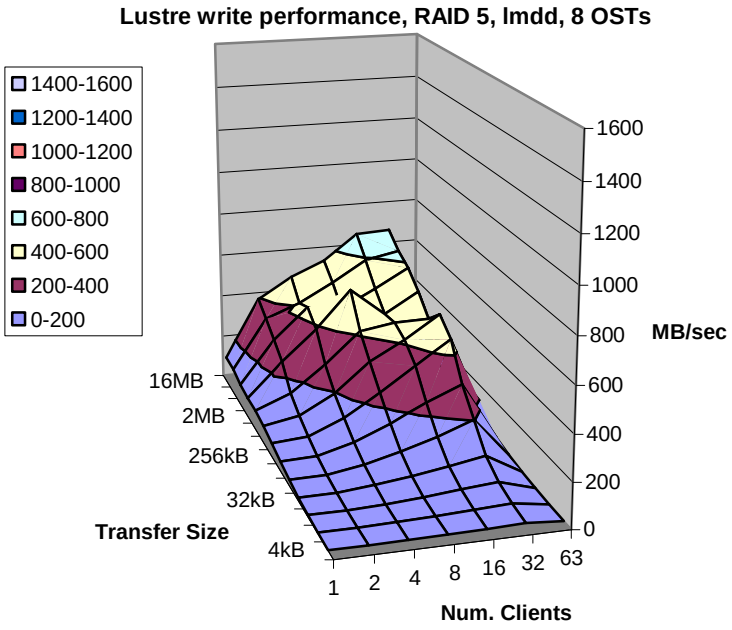

Figure 4.10

We found that the performance issue was due to write-back cache utilization was too low (watermarks were set too low) and the data layout on the disk drive had match the cache segment alignment. Also there were some other settings that less impacted the write performance. Carefully, tuning the controller we managed to boost write performance from 683 MB/sec to more than 1.4 GB/sec (see Figures 4.10 and 4.11). At the same time read performance (Figures 4.9 and 4.12) improved too, although not significantly.

4.6 Eight OSTs (RAID 5) Lustre sequential I/O performance

However, it would be interesting to compare the Lustre performance for RAID 5 system to the previously tested RAID 3 box. Since, the RAID 5 was attached to a smaller system we could not obtain the performance numbers for 128 and 256 Catamount clients. However, we still could compare the results using the previously obtained test results by looking only up to 64 Catamount nodes.

Figures 4.11 and 4.12 illustrate Lustre write and read performance, respectively, for 8 OSTs. Although, write bandwidth slightly exceeded 1.4 GB/sec (out of 1.6 GB/sec of the theoretical bandwidth for the RAID 5 system) the read performance was excellent exceeding 1.55 GB/sec, which is better than 96% of the theoretical bandwidth.

Therefore, similar Lustre performance could be achieved using both RAID 3 and RAID 5 storage systems. However, RAID 5 systems tend to be less expensive to the comparable RAID 3 systems. During preliminary testing for random I/O, RAID 5 outperformed significantly RAID 3 system. However, further testing is needed to get complete and accurate picture about Lustre performance using RAID 5 system.

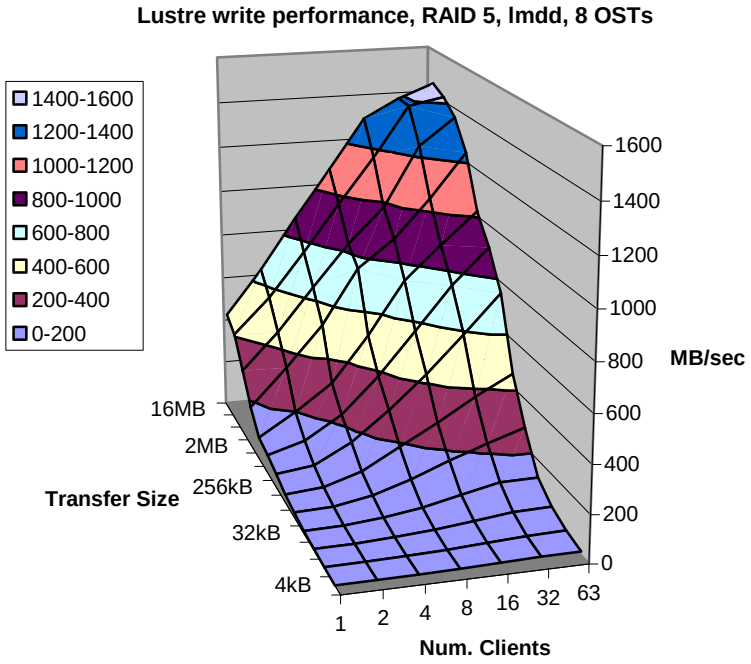

Figure 4.11

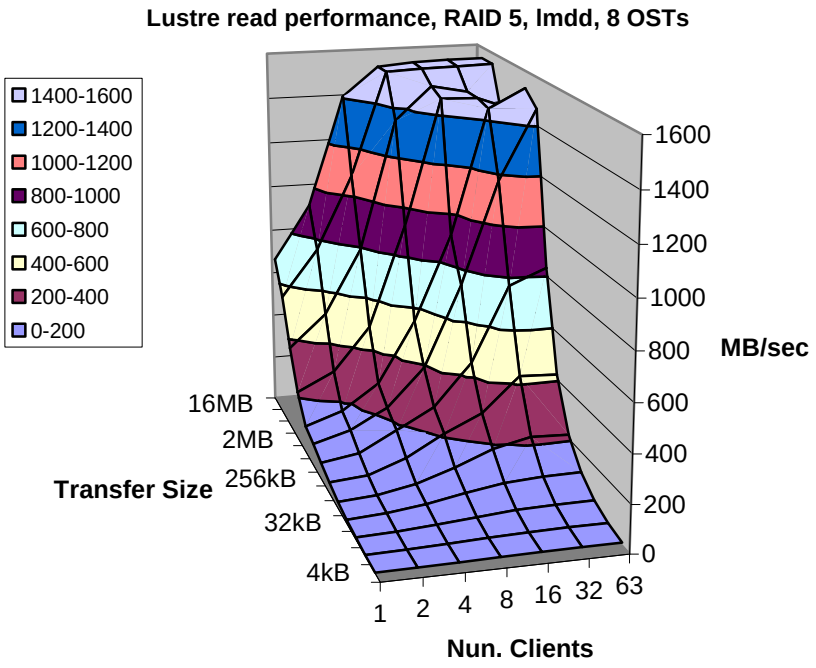

Figure 4.12

4.7 Lustre scaling

Figures 4.13 and 4.14 show how the Lustre performance scales as the number of OSTs increases from one to 11 OSTs. In this test we attempted to minimize the impact of the back-end RAID thus, maximizing the cache utilization for both reads and writes. To achieve that we used lmdd with the same settings as in section 4.1. All clients overwrite the same LBAs many times to minimize the disk drive mechanical latencies. Also, all clients read the same file therefore, only the first read request goes to the disk and every subsequent read is the controller cache hit.

The obtained test results are depicted on the figures 4.13 and 4.14. As it could be observed, the Lustre file system scales well indeed. However, further research should be done to investigate various I/O access patterns to different Lustre configurations.

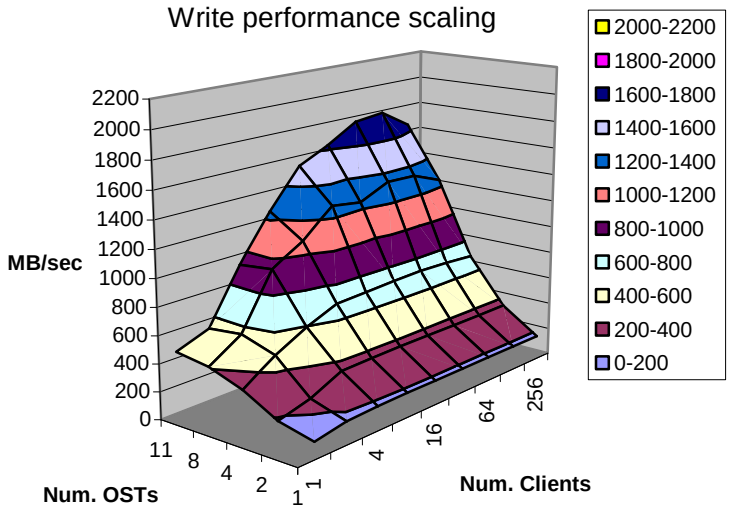

Figure 4.13

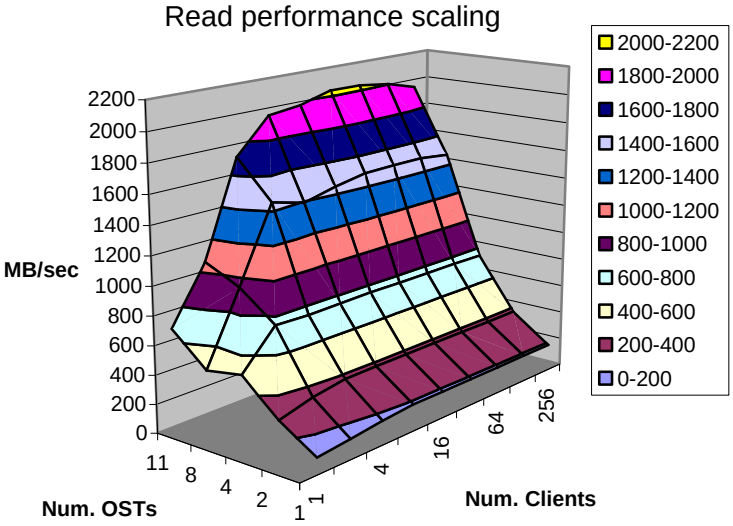

Figure 4.14

5 **Conclusion**

The test results presented in this paper reveal that the Lustre file system implementation on the Cray XT3 system performs generally well. Lustre file system performance should not depend on the type of I/O traffic since software is many times faster compared to the back-end RAID mechanical latencies. Whether it is sequential, quasi-random, or random, the overall I/O performance may be significantly different depending on the I/O traffic access patterns. Thus, for the random reads, the overall I/O throughput is heavily (negatively) influenced by the back-end RAID 3 performance. Therefore, further research

needs to be done to explore ways to improve (tune) RAID (especially RAID 3) system performance mainly for the random I/O requests.

Further testing could be done using Solid State Disk drive (SSD) for the back-end storage. This approach would eliminate delays associated with the disk drive mechanical latencies and emphasize the real Lustre file system performance. However, we attempted to achieve that by selecting the I/O test and choosing the parameters that would minimize the disk drive seek and rotational latencies. Another approach would be to configure RAMDISK on the OSTs thus, eliminating completely Fibre Channel and RAID influence on the file system performance.

Nevertheless, to minimize the latencies associated with writing the ext3 file system journaling information, couple of avenues could be explored, form writing the ext3 journaling information onto a remote node to creating a RAMDISK and mirroring it to the associated LUN. However, those options might have their own issues too. Undeniably, further research needs to be done to further improve Lustre I/O performance and address some of the previously mentioned issues.

References

[1] http://www.lustre.org/

[2] CRAY SUPERCOMPUTING COMPANY CRAY XT3
http://www.it.lut.fi/kurssit/05-06/Ti5317000/seminars/cray%20xt3%20doc.pdf

[3] lmbench: Portable Tools for Performance Analysis
http://www.usenix.org/publications/library/proceedings/sd96/full_papers/mcvoy.pdf

[4] Parallel File System Testing for the Lunatic Fringe:
the care and feeding of restless I/O Power Users*
http://storageconference.org/2005/papers/01_Hedgesr_lunatic.pdf

[5] Sandia's Red Storm Detailed Architecture
http://www.primidi.com/2003/10/29.html

[6] Lawrence Livermore National Laboratory
http://www.llnl.gov/