File Access Optimization with the Lustre Filesystem at Florida CMS T2

This content has been downloaded from IOPscience. Please scroll down to see the full text.
2015 J. Phys.: Conf. Ser. 664 042028
(http://iopscience.iop.org/1742-6596/664/4/042028)

View the table of contents for this issue, or go to the journal homepage for more

Download details:

IP Address: 137.138.93.202
This content was downloaded on 09/03/2016 at 08:40

Please note that terms and conditions apply.
File Access Optimization with the Lustre Filesystem at Florida CMS T2

P. Avery¹, D. Bourilkov¹, Y. Fu¹, B. Kim¹
¹University of Florida, Gainesville, FL, U.S.A.

E-mail: bockjoo@phys.ufl.edu

Abstract. The Florida CMS Tier2 center, one of the CMS Tier2 centers, has been using the Lustre filesystem for its data storage backend system since 2004. Recently, the data access pattern at our site has changed greatly due to various new access methods that include file transfers through the GridFTP servers, read access from the worker nodes, and the remote read access through the xrootd servers. In order to optimize the file access performance, we have to consider all the possible access patterns and each pattern needs to be studied separately. In this presentation, we report on our work to optimize file access with the Lustre filesystem at the Florida CMS T2 using an approach based on analyzing these access patterns.

1. Introduction
The Florida CMS[1] Tier2 center is one of the CMS T2 centers in the USA. The facility has been using the Lustre[2] filesystem for its data storage backend system since 2004.

In the beginning, the data access pattern was very simple. Namely, the necessary datasets for analyses are transferred to the site and then the analysis jobs are executed through local storage access. However, the simple data access pattern has changed greatly due to various new access methods. As before, data are transferred and the data analysis is performed either through the datasets stored in the local storage system or the xrootd server where the remote jobs access the datasets only through the xrootd servers. At the same time, local users submit jobs through the local batch system and access the local storage. The new data access pattern is illustrated in Figure 1.

Although not all clients access the filesystem simultaneously, it would be best if we can identify possible IO optimization techniques and optimize the IO from the different clients. Since the Florida CMS Tier2 center is one of the few sites where the high performance and the high throughput storage system is deployed, it is desirable to find the best Lustre storage filesystem utilization in the CMS computing environment.

The collected information during the access optimization study can be very useful in the monitoring of the storage system performance and can allow the site to decide when the hardware upgrade or the software reconfiguration are necessary. The collected information will be used for comparisons with new benchmarking. The previous presentations on the Lustre filesystem that are related with the current work can be found in References [3] and [4], where we have optimized the
filesystem at the stripe, TCP settings and remote procedure calls level. In this paper we concentrate on global optimizations.

2. The Florida CMS Tier2 Storage System

The Florida CMS Tier2 storage consists of the SRM[5] (storage resource manager) server, the GridFtp[6] servers, the Xrootd[7] servers, and the backend Lustre filesystem. The BestMan implementation is used for the SRM with a single server. Eight GridFtp servers are the Grid file transfer servers behind the BestMan SRM, each provides 10Gbps of network capability. The eight GridFtp servers also serve as Xrootd servers.

There are 19 backend Lustre filesystem servers that are also called the object storage server (OSS). Each server provides six or 9 object storage targets (OSTs). The deployed Lustre server version on the OSS is 1.8.7. OSTs are configured as RAID5 or RAID6 arrays and the actual files are stored in RAID arrays. The total usable space of the backend Lustre filesystem is 2.3 PB.

The entire system is connected through 55 Gbps infiniband (IB) interconnection. The storage system and the worker nodes where the analysis or the production jobs are executed are connected through 40 Gpbs IB-IP bridges. The storage system is also connected through the WAN that is capable of 100 Gbps. Figure 2 show a picture of the Florida T2 storage system.

![Figure 2 Picture of the Florida T2 Storage System](image-url)
3. Performance Checks with the Storage System

In order to check the performance of the Lustre filesystem, the following methods are considered:

- Simple Copy Test
- GridFtp and Xrootd File Transfer Test
- CMSSW IO Test

What we need to do eventually is to find the optimal way to avoid high load created by any of the data access methods.

3.1. Simple Copy Test

The simple copy test provides the sequential read and write-rate of the Lustre filesystem. The rate is measured as a function of the Lustre OST that is the easily testable smallest unit of the Lustre filesystem. In the CMS computing environment, the input files are typically ROOT[1] files. They are compressed files and the executables are usually accessing the files randomly. So, this simple copy test is not proportional to the CMS software I/O rate.

However, this is the basic measurement of the Lustre I/O and tells us the baseline performance of the filesystem and the hardware. The test is performed between the Client 1 and the Lustre filesystem in Figure 1.

In the simple copy test, the Linux cp command is used to write files to and read files from the Lustre filesystem, and the time is measured with the time command. The rate is calculated from the measured time as the file used has fixed size of 4GB. Figure 3 and 4 show the Lustre file read and write rate from the simple copy test, respectively. The left plot of the Figures is the rate as a function of the OST number and the right plot of the Figures (similar for later rate Figures) is the rate as a function of different time slices during the run in which a single OST rate is measured. Each dot of the right plot corresponds to the rate measurement of a single OST in that run. The right plot is included to show that usually there is very little time-dependence of the OST rate. The copy read rate is 650MB/s for the new OSTs and 200MB/s for the older OSTs. The write rate is worse than the read rate but is similar in the OST dependence. The hard drives in the system are uniform except for one OSS where a different brand drive is used on one of the 19 OSSes.

3.2. File Transfer Test

The types of file transfers include the file transfers between sites or GridFtp servers behind the SRM servers, the Xrootd remote file read, and the analysis jobs that transfer the produced files from worker modes to the storage element at the user’s home institution or the designated Tier2.

The file transfer test in this subsection only considers the file transfers through the GridFtp servers and the Xrootd servers. The file transfers from the user analysis jobs are usually limited by the network capability of the worker nodes. We assume this type of the storage access does not affect the storage system as much as the other types.

![Figure 3](image-url) Simple Copy Read Rate

![Figure 4](image-url) Simple Copy Write Rate
In the file transfer test, we have measured the single stream file transfer rate within the Florida T2 using the command tools globus-url-copy and xrdcp. This is to confirm the simple copy test results and the local network. The network topology involved in the file transfer test is illustrated in Figure 5. This checks the transfer rate only within the local network and corresponds to the “Local File Transfer Test” section in Figure 5.

![Network topology of the file transfer from the Florida T2 and outside](image)

Figure 5  Network topology of the file transfer from the Florida T2 and outside

The test is executed by transferring files from the Lustre storage using either the globus-url-copy or the xrdcp to the memory. Figure 6 shows the Lustre read rate using the globus-url-copy from Lustre to the memory. The rate is very similar to the result from the simple copy test shown in Figure 3 as expected. The xrdcp from Lustre to the memory is also measured using a similar method. The result is shown in Figure 7. The right plot of the Figure 7 is the time dependence of the measurement. During this measurement, we had to retire two OSS servers due to their age and the OSTs from 19 to 30 are missing in this plot. We can see Figure 7 have similar IO rates as the simple copy rates shown in Figure 3.

The single stream transfer rate is also measured between other sites and Florida T2 using the globus-url-copy or FTS. But this rate is mostly dependent on the other sites SE IO rate and the network. The average single stream rate ranges between 10 and 120 MB/s depending on the network status and the site status.

**3.3. CMSSW IO Test**

The CMS software typically writes compressed ROOT files. In order to see if there is any discernable OST dependence, we have used the CMSSW produced ROOT files to check the analysis read rate by utilizing the analysis framework used by the U. of Florida Higgs group’s ROOT tree creating analysis used for the Higgs to 4 lepton analysis. Figure 8 shows the CMSSW analysis read rate. As is shown, the average read rate is only around 17MB/s and the read rate is much lower than the other read rate tests performed.

This is due to the fact that the ROOT files are highly compressed and most of the read time is spent during the file decompression. Also the CMSSW analysis is typically very CPU-intensive. In order to confirm the result is due to the CPU-intensiveness, the ROOT files were put in the memory, ran the CMSSW IO test, and we found the result is similar to the average read rate as in the Figure 8.

The identical inputs and the software were run at other sites by staging in the input files at other sites. This was to check if there is any advantage in reading files from the Lustre over another filesystem - HPFS. The result shows there is no significant difference among different storage where the input files reside. This also confirms the CPU-intensiveness of the CMSSW IO.
The CPU dependence of the CMSSW read rate is checked at four different US Tier2 sites and this is shown in Figure 9. The purpose of the exercise is to show the CPU dependence of the CMSSSW analysis rate and to explain there could be a significant variation in the CMSSW analysis rate at different sites depending on the CPUs used in the worker nodes.

4. Optimizing the IO Activities
With the different possible Lustre access tests performed separately and to find the optimal way to control the IO activities, we have come up with a simple representation for all client activities that were emulated by running the following jobs simultaneously:

- CMSSW analysis jobs at the Lustre site
- CMSSW analysis jobs at a remote site with the direct xrootd access
- Jobs that transfer files from a remote host to the Lustre site storage element
- Simple copy jobs

The CMSSW analysis jobs at the Lustre site emulates the regular analysis jobs, the CMSSW analysis jobs at a remote site with the xrootd access emulates the more ubiquitous analysis jobs these days by reading the input files from the xrootd directly, the transfer jobs emulates the CMS PhEDEx transfers by selecting various source sites randomly. These are the control background jobs for the simple copy jobs that are the jobs from which we want to measure any IO variation.
In order to find the control background job that causes the most significant IO variation, number of jobs in one of the three control background jobs is varied while the number of jobs in the other two control background jobs is fixed.

From the test, we find that no particular background job impacts the overall performance of the Lustre system. Rather we found that the performance depends on the number of files accessed in an OST. When the files on the same OST were accessed simultaneously, the load on the OSS with which the OST is associated rose high and the read rate declined significantly. In a normal operational mode, the load on the individual OSS ranges between 0 and 20 in a single OSS. However, 60 files residing on a single OST in an OSS created a load of 120 for an OSS with eight cores. Figure 10 illustrates this and shows the xrootd read rate same as Figure 7 but a separate read rate is measured for OSTs 1 to 18 by reading 60 files simultaneously on the OST 1 that resides on OSS1 (OST1 – OST6). A factor of ~3 reduction in read rate is seen. From this observation, we have optimized the Lustre system by constantly reshuffling the files around different OSTs and this reduced the impact from such an access.
5. Conclusions
We have performed IO tests with three different file access mechanisms and no critical issue was found. The methodology is not limited to Lustre but can be applied to other storage systems. We find various IO patterns can cause trouble in some IO servers, e.g., high load on the GridFtp servers but these IO troubles are network related and they were either non-standard workflows due to network issue or very IO intensive workflows.

In a very quiet condition, we were able to achieve very high sequential read and write rate and the Lustre filesystem may be used for the very high IO intensive workflows that are currently running at the CMS Tier1 centers only.

We find no particular file access method impacts the Lustre filesystem, when used for data files of gigabyte sizes. We also find the uniformity of the system is the most important and files need to be distributed among different OSTs uniformly to avoid the performance reduction. The uniform distribution of files across different OST is the most important optimization. This has been implemented and run regularly to ensure there are no hot spots among OSTs. The situation is not different with other filesystems.

The studies reported here have been good learning practices and we cautiously project the work can be extended to find more efficient ways of accessing files on the Lustre filesystem.

References
[1] CMS Collaboration 2008 “The CMS experiment at the CERN LHC”, JINST 3 (2008), no. 08, S08004
[2] Lustre Home Page, http://lustre.org/, Retrieved May 4, 15
[3] Y. Wu, B. Kim, P. Avery, Y. Fu, D. Bourilkov, C.Taylor, C.Prescott and J.Rodriguez, “Lustre filesystem for CMS storage element (SE)”, J. Phys. Conf. Ser. 331, 052034 (2011)
[4] ExTENCI Collaboration, D. Bourilkov, et. al., “Secure Wide Area Network Access to CMS Analysis Data Using the Lustre Filesystem”, J. Phys. Conf. Ser. 396, 032014 (2012)
[5] Donno, et. al., 2008 Storage Resource Manager version 2.2: Design, implementation, and testing experience, J. Phys.: Conf. Ser. 119 062028
[6] Allcock W, et. al., 2005 ACM/IEEE Supercomputing pp.54, 54, 12-18
[7] Xroot Home Page, http://www.xrootd.org/, Retrieved May 4, 15
[8] Brun R and Rademakers F 1996 “ROOT - An Object Oriented Data Analysis Framework”, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. And Meth. in Phys. Res. A 389 (1997) 81-86