HPC File Systems in Wide Area Networks: Understanding the Performance of Lustre over WAN

Alvaro Aguilera, Michael Kluge, Thomas William, and Wolfgang E. Nagel

Technische Universität Dresden, Dresden, Germany
{alvaro.aguilera,
michael.kluge,thomas.william,wolfgang.nagel}@tu-dresden.de

Abstract. Using the first commercially available 100 Gbps Ethernet technology with a link of varying length, we have evaluated the performance of the Lustre file system and its networking layer under different latency scenarios. The results led us to a better understanding of the impact that the network latency has on Lustre's performance. In particular spanning Lustre's networking layer, striped small I/O, and the parallel creation of files inside a common directory. The main contribution of this work is the derivation of useful rules of thumbs to help users and system administrators predict the variation in Lustre's performance produced as a result of changes in the latency of the I/O network.

1 Introduction

Scientific instruments create an enormous amount of data every day. For example, the NASA Earth Observation System (EOSDIS) created about 2.9 TiB data in average each day in 2010 [1]. To share this data with collaborating scientists, WAN file systems have already proven their value. The European DEISA project [2] utilizes a series of dedicated 10 Gbps links to serve a distributed GPFS file system to different HPC centers. Since about 2006, the parallel file system Luste has gained some attention while being used in WAN environments [3,4,5]. These evaluations and the consecutive use of Lustre as a production file system in the DataCapacitor project at Indiana University have demonstrated that parallel file systems can be used efficiently on networks with latencies of more than 100 milliseconds.

However, up to now, the relevant publications only describe different use cases, experiences, and tuning efforts, but none focuses on the interplay between the network latency, the Lustre tunables and the resulting performance of the file system. Advancing this understanding will certainly ease Lustre's tuning effort as well as give some hints about how to use the file system in production, e.g. how files should be striped. Our aim in this paper is to make a first step in this direction by analyzing our observations on the 100 Gpbs Ethernet testbed.

The paper is structured as follows. The next section (Section 2) introduces the testbed system itself. In Section 3, previous work on performance models,

C. Kaklamanis et al. (Eds.): Euro-Par 2012, LNCS 7484, pp. 65–76, 2012.

especially for parallel file systems, is reviewed. Within Section 4, the Lustre networking layer is evaluated and a simple performance model for this software layer is presented. After this, Section 5 deals with the performance obtained when different I/O calls are issued from a single client while Section 6 extends this work to multiple clients. Section 7 gives a conclusion and sketches future work.

2 100 GbE Testbed between Dresden and Freiberg

The 100 Gbps testbed, provided by Alcatel-Lucent, T-Systems, HP, and DDN, provided a unique resource to, on the one hand test this new technology, and on the other hand to extend our knowledge about different network services. The testbed spans the distance between the cities of Dresden and Freiberg in Saxony, Germany with a geographical distance of about 37 km and a optical cable length of about 60 km. During the test, additional boxes with optical cables have been used to extend the testbed from 60 up to 400 km. This allows us to conduct experiments using different latency configurations with a reliability not found in software-based latency injection methods. An Alcatel-Lucent 1830 photonic service switch connects both sides and can transmit 100 Gbps on a single carrier. The 7750 SR-12 service router links the optical layer and the network adapter. Both service routers (Freiberg and Dresden) have one media dependent adapter (MDA) with 100 Gbps, two adapters with 5x10 Gbps, five adapters with 2x10 Gbps, and 20x1 Gbps adapters.

HP provided 34 DL160G servers, 17 on each location, which are stocked with a ServerEngines-based 10 Gbps card, each connected to one of the 10 Gbps interfaces of the Ethernet switch. All servers are equipped with one six core Intel Westmere (Xeon 5650, 2.67 GHz) processor and 24 GiB of RAM.

Several sub-projects were scheduled on this testbed. Initial TCP tests provide the subsequent projects with a reliable base in terms of the available bandwidth and the network behavior in general. Three different parallel file systems: GPFS, Lustre, and FhGFS are installed on the HP servers to study the impact of the

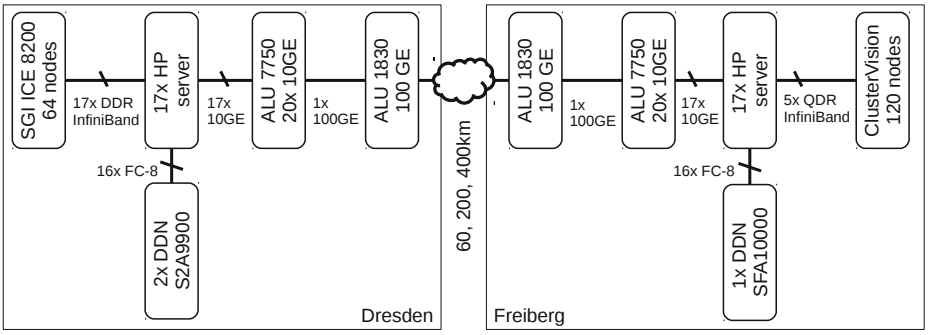

Fig. 1. 100 Gbps testbed equipment with connection cables

latency on the file system performance. An overview of the testbed setup is shown in Figure 1.

3 Related Work

The performance analysis of network file systems accessed over high latency networks such as WANs has been the focus of several studies during the last decade, and has been gaining importance as the technological trends make this use case more and more practical. The first studies describing the viability of a WAN file system in HPC context were conducted by the researchers working on the TeraGrid project, and can be found in [6]. The first published experiences using a minimally tuned Lustre file system in a WAN environment are detailed in [3,4,5,7]. More recent publications concentrate not on analyzing Lustre's raw performance over WAN, but more on its suitability for concrete use cases. An analysis of the use cases that would profit from an HPC WAN file system the most is presented in [8]. In [9], Cai et al. evaluated the suitability of a Lustre over WAN solution to sustain database operations. In [10], Rodriguez et al. describe their experience using Lustre over WAN for the CMS experiment at the LHC. Even though the modeling and simulation of storage systems have been a subject of study for at least two decades, most of the publications concentrate on modeling the individual components and not the file systems. To the best of our knowledge, there is only one publication explicitly dealing with the modeling of Lustre's performance, namely [11], in which Zhao et al. applied the idea of relative modeling to predict the performance of a Lustre file system.

4 Lustre's Networking Layer

In the first part of this paper we will discuss the performance of Lustre at its networking layer without considering the storage hardware and software components acting on lower layers. Understanding the performance of Lustre's networking layer is a first logical step in order to gain an understanding about how this parallel file system behaves when its major tunable parameters and network conditions are changed.

4.1 The LNET Protocol

Lustre Networking (LNET) is a custom networking API that leverages on the native transport protocol of the I/O network to interconnect the Metadata Server (MDS), the Object Storage Servers (OSSs), and the client systems of a Lustre cluster. It offers support for most of the network technologies used in HPC through a set of Lustre Network Drivers (LNDs) that are both available as individual kernel modules, and user space libraries. Internally, LNET uses a stateful protocol based on remote procedure calls that was derived from Sandia Portals. The bandwidths achievable by LNET during the file system

operation are determined by a combination of its own performance parameters, and the characteristics and configuration of the underlying I/O network. In low-latency environments, the relevance of the former group of parameters is not apparent, since Lustre achieves its maximal performance without much tuning. Their importance, however, is promptly made clear as soon as the network latency increases.

LNET fragments all data transfers in units called Lustre RPCs, whose sizes are always aligned to the system page size, and range from a single page (in most cases 4096 bytes) up to one megabyte. The maximal size a Lustre RPC may have, can be modified on a per client basis as long as the new size satisfies the conditions stated in [12]. In order to minimize the transmission and processing overhead associated with small RPCs, Lustre tries to merge adjacent RPCs to form RPCs of maximal size. The individual size of an RPC being transmitted is ultimately determined by a combination of the maximal RPC size, the size of the buffers being read or written by the application, and on whether or not the operation is immediately committed to disk (call to `fsync()`, operation in `O_DIREC` mode, etc.). LNET is a stateful protocol in which every RPC being sent must be acknowledged by the receiver. Similarly to TCP's sliding window protocol, LNET may send more than one RPC before waiting for an acknowledgment response. These unacknowledged RPCs are normally referred to as the RPCs in flight. Like in the previous case, the maximal number of RPCs in flight sent by LNET during any operation is a parameter that can be defined on a per client basis.

In the following section we explore the interplay between the size and count of Lustre RPCs in flight, and how they affect, together with the network latency, the LNET effective bandwidth.

4.2 Model Constraints

The performance of LNET is heavily dependent on the underlying transport protocol it relies upon, and especially on its congestion avoidance mechanisms. In order to keep our model as simple, and as general as possible, we will constrain it to data transmissions that are unthrottled by the transport protocol of the I/O network. The behavior of the LNET bandwidth as a function of its in-flight data is best exemplified in Figure 2. Our focus will lie in the unthrottled state (a) in which changes in the RPC size or count, as well as on the network latency yield a full effect on the LNET bandwidth. Even though this constrain is certainly undesirable, doing otherwise would incur in excessive complexity, while simultaneously tying our model to a particular transport protocol (implementation).

4.3 Proposed Model

It is well understood that for any given network, the relation between the bandwidth-delay product and the amount of in-flight data that is actually present in a network segment at a given time is one of the key factors determining the network throughput [13]. The maximal amount of in-flight data D_{max} that fits

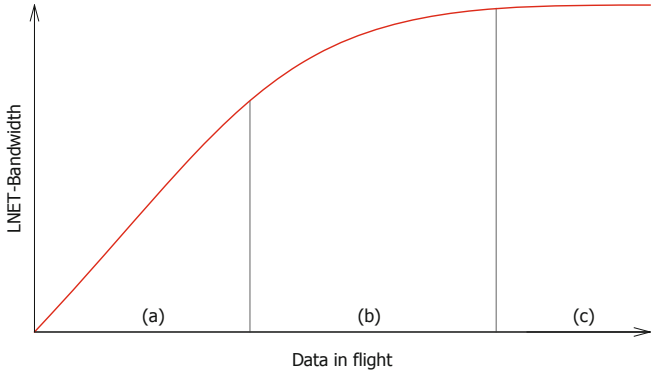

Fig. 2. Behavior of the LNET bandwidth as a function of its in-flight data. (a) unthrottled communication, (b) throttled by the transport protocol, and (c) maximal utilization reached.

inside a network path is determined by the bandwidth-delay product, defined as the round-trip time l of the network, multiplied by the network bandwidth b, plus the buffer space m of the network devices along the way (Equation 1). It is trivial to see that (omitting the buffer space m) any change in the network latency from l_0 to l_1 will immediately affect D_{max} by a factor of l_1/l_0.

$$D_{max} = l * b + m \tag{1}$$

On the other hand, the amount of data D that LNET may put down the wire before waiting for an acknowledgment is mainly defined by the size s and number c of RPCs in flight (Equation 2). In a similar way, a change in the size and number of RPCs in flight should affect D by a factor of $(s_1c_1)/(s_0c_0)$.

$$D = s * c \tag{2}$$

Our first modeling hypothesis will be that the LNET bandwidth will vary by the same factor, and in direct proportion to D, and by the same factor but in inverse proportion to D_{max}. This means that the expected variation in a known LNET bandwidth b_0 resulting from a change in the network latency, or in the size and number of RPCs in flight during an unthrottled communication can be calculated using Equation 3.

$$b = (\frac{s_1}{s_0}\frac{c_1}{c_0}\frac{l_0}{l_1})b_0 \tag{3}$$

4.4 Measurement and Comparison with the Model

The benchmarking of the LNET performance was conducted on the testbed system previously introduced using the LNET-Selftest tool distributed with Lustre.

This tool allows the generation of intense LNET traffic between groups of nodes without requiring any physical I/O. The generated workload is also supposed to be similar to that produced by Lustre during real I/O operation.

Using (3) we were able to obtain a good approximation of the experimental data for changes in the network latency and RPC count. However, the bandwidth changes resulting from increasing the RPC size up from its minimum value were roughly half as big as those predicted. This difference indicates that an increase in the RPC size doesn't translate 1:1 to an increase in the LNET in-flight data. In spite of this, the model can be adapted by introducing a factor k to account for this overhead, as shown in Equation 4 .

$$b = \frac{1}{k}\left(\frac{s_1}{s_0}\frac{c_1}{c_0}\frac{l_0}{l_1}\right)b_0 \tag{4}$$

It is expected for the value of k to vary depending on each particular network configuration. For the testbed system, a value of $k = 2.27$ yielded the best results with an overall error of less than 15%. Figure 3 and 4 compare some of the predictions against the experimental data. The predictions of the model were done using the measured performance obtained with one 4 KiB RPC in flight and a network latency of 2.17 ms to extrapolate all other points.

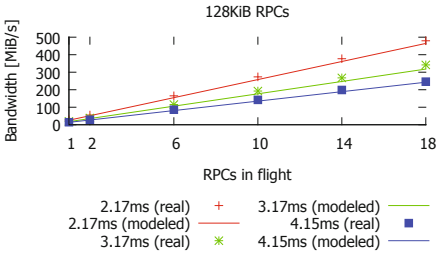

Fig. 3. Predictions of the LNET model for 128 KiB RPCs using different latency settings and RPC counts.

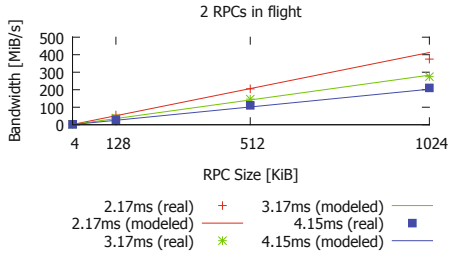

Fig. 4. Predictions of the LNET model for 2 RPCs in flight using different latency settings and RPC sizes.

5 Single Client Performance Observations

The aim of this section is to describe the impact of the latency for small file accesses and a single Lustre file system client. This data is advantageous for users that use WAN file systems similiar to home file systems, for example for compiling source code or for editing input files. The main parameters that have an impact on the performance and that can be influenced by the user are mainly the file size, the access size and for Lustre, the way the striping is done. For the striping it is worth mentioning, that the stripe size is fixed to 1 MiB as this is the native stripe size of the DDN devices. Thus, only the number of used stripes can be adjusted.

5.1 Setup and Measurements

For these tests we performed initial measurements with IOzone and a fiber length of 200 km and compared the results with results gathered at 400 km. Comparing this data with performance data collected locally (at 0 km) would have not made sense in this context as the communication locally was done via InfiniBand and any comparison would not only include the latency difference but also the difference between the protocols.

5.2 Observations and Findings

Fig. 5 shows the difference between the 200 km and the 400 km data for different I/O functions, for different file sizes and one block size. The figure shows the performance in KiB/s for the 400 km case in percent, using the performance for 200 km as 100 %. It shows that there is a noticeable performance impact only on the initial write of a file. All other functions, which reuse existing files, show only a small performance impact. This impact is due to the fact that the creation of a file needs at least one RTT. With increasing file sizes, this additional RTT has less influence on the total time of the operation.

Fig. 6 shows a more detailed performance study of the influence of the file size and the stripe count on the performance. The figure shows the differences in the latencies for the initial creation of files with different sizes and different stripe counts. Here, we subtracted the numbers gathered at 200 km from the numbers gathered at 400 km. As the next step, we normalized this time difference to the difference in the RTTs (4.14 ms − 2.17 ms = 1.97 ms) between the two distances. This allows to characterize the impact of the additional distance on the performance.

Up to 1 MiB file size, all data is written to a single stripe and the number of stripes that the file can use does not influence the performance. The '1' in the figure in these cases just means that at 400 km it takes 1.97 ms longer to create a file and to write the content than it takes at 200 km. For each additional stripe used there is a penalty that is added as soon as a new stripe is used. This is due to the fact that the Lustre file system creates the objects on the storage servers for the first stripes in a sequential fashion.

This can create a large impact on file systems with a large number of stripes used by default, as the time to write the first N MiB will always be (N+1)∗RTT. The +1 has to be added for the initial file creation on the metadata server, the stripes are created in an extra step. The problem with this finding is that in these cases the bandwidth is determined by the RTT, and not by the capabilities of the link.

Fig. 7 shows that there is no significant performance impact by the addition of 200 km to the distance when the same file is accessed with different block sizes or with different I/O functions. As most file I/O for these cases is rather small, this implies that the clients cache most I/O operations efficiently.

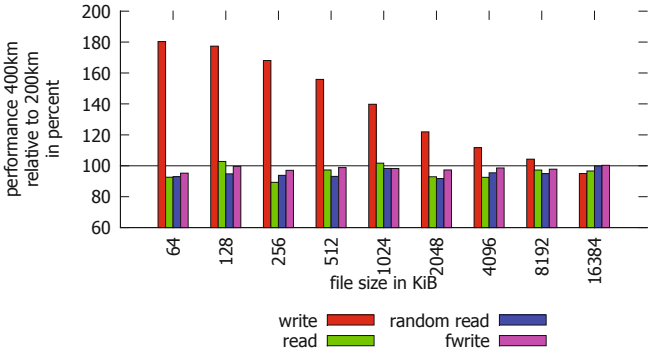

Fig. 5. Comparison of different I/O functions for 200 km and 400 km for different file sizes. A block size of 4 KiB and a stripe count of 1 was used. The performance for 400 km is given in percent relative to the performance for 200 km.

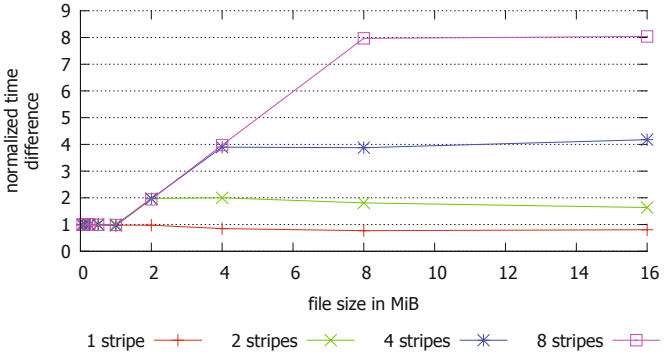

Fig. 6. Time differences for the initial file creations between the 400 km and the 200 km setup. The time difference is normalized to the difference of the RTTs for both distances.

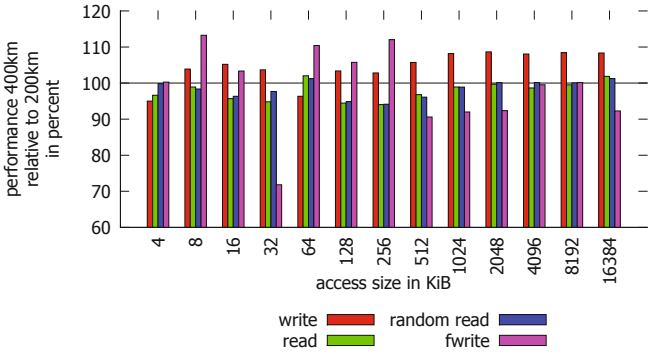

Fig. 7. Comparison of different I/O functions for 200 km and 400 km for different block sizes. A file size of 16 MiB and a stripe count of 1 was used. The performance for 400 km is given in percent relative to the performance for 200 km.

6 Performance Observations for Multiple Clients

The goal of the experiment was to determine the latency of the I/O traffic for multiple clients working in parallel, and to analyze the impact of the extension of the testbed fiber length on different file sizes. We therefore used the Lustre file system described above to generate unidirectional I/O traffic from Dresden to Freiberg. Up to 16 clients were used to measure latency and bandwidth using 8 B files for latency and 2 GiB files for bandwidth. The benchmarking tool used to generate the workload was IOR [14].

6.1 Setup and Measurements

Each process wrote its own file using POSIX I/O in `O_DIRECT` mode to ensure that the I/O operation is immediately committed to disk (see Fig. 8). The value *blockSize* in Fig. 9 denotes the amount of data that is written to each file per process, while *transferSize* represents the size of the payload being sent with each I/O request. This is restricted by IOR to a minimum of 8 bytes and a maximum of 2 GiB. IOR uses MPI to synchronize processes, therefore switching on *intraTestBarriers* adds a `MPI_Barrier(all)` between each test to ensure that there is no traffic left from previous I/O operations.

```
api=POSIX
filePerProc=1
useO_DIRECT=1
intraTestBarriers=1
repetitions=10
writeFile=1
readFile=0
```

Fig. 8. IOR configuration header

```
RUN
 blockSize=8
 transferSize=8
 numTasks=16
RUN
 blockSize=2147483648
 ...
```

Fig. 9. I/O test with 16 processes each sending 8 B and 2 GB

The 1 GbE management links were used for the MPI communication not to disrupt the Lustre traffic. Also, each test was repeated 10 times for higher accuracy. The benchmarks were executed on the Dresden HP nodes on a mount point pointing to the Lustre in Freiberg using the SFA10K DDN storage and the 16 Freiberg HP nodes as OSTs.

In the experiments, IOR was run on one Server using one process writing to only one OST at first. Then additional servers doing the same file I/O were added up to the point where 16 processes were writing 16 files in parallel each to its own OST. This setup generated data for:

- different optical line length (60 km, 200 km, and 400 km)
- different number of parallel writes (1-16)
- different file sizes (8 B up to 2 GiB).

6.2 Observations and Findings

IOR can separately log the times for the functions involved in writing a file to disk (`open()`, `write()`, `close()`). We normalized the values to the round-trip time that was measured on the TCP layer (60 km → 0.72 ms, 200 km → 2.17 ms, 400 km → 4.14 ms) to see how many RTTs it takes to complete each of the functions and whether this depends on the number of parallel clients or not.

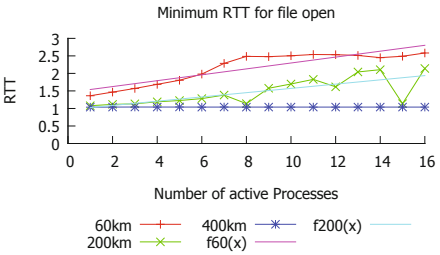

Fig. 10. Minimum RTT for file open of 8 Byte files

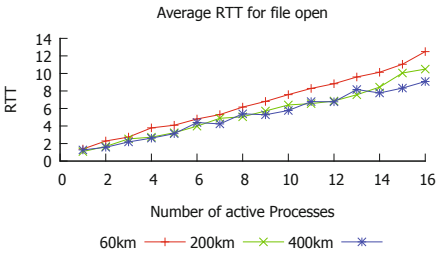

Fig. 11. Average RTT for file open of 8 Byte files

We first look at the time needed to open a file at the different line lengths and for different numbers of parallel clients. The numbers shown are for 8 Byte files but are essentially the same for 2 GiB files as well. The minimum numbers in Fig. 10 show the anticipated time of 1 RTT only for the 400 km distance. The latency of the computer hardware has a larger influence at 60 km due to the small transfer time compared to the time needed for processing. The time needed for the `open()` call is the RTT plus an overhead which in turn is a function depending on the number of parallel processes. Using a linear least squares fit we can determine the slope being 0.08 for 60 km and 0.06 for 200 km. This means that each additional process adds an overhead of 8 % of the RTT for 60 km and 6 % for 200 km to the minimum time needed to complete a `open()` call for the first process in the group that issues the open call. Additionally, the average time rises with the number of parallel processes (all processes involved and 10 repetitions) as seen in Fig. 11. This is due to the locking of the directory in which all files reside.

For the `close()` call the minimum times are nearly the same across all process counts at 1 RTT as there is no additional workload on the MDS. Again, 2 GiB files do not differ from 8 bytes files. Fig. 13 shows the transfer times for the `write()` call with minimum and maximum values as error bars. For large files the distance has no significant impact on the transfer time. As each of the processes was writing to its own OST, the number of parallel streams has no noticeable influence on the individual performance either.

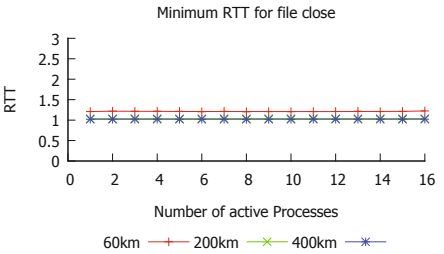

Fig. 12. Minimum RTT to close files of 8 bytes

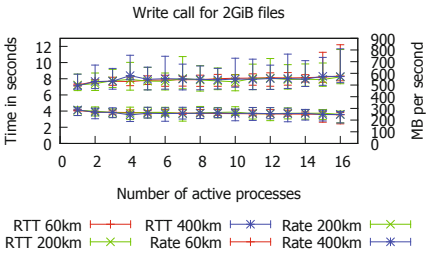

Fig. 13. Write RTT and data rate for 2 GiB files

7 Conclusion

In this paper we explored some aspects of the performance variation exhibited by the Lustre file system when subjected to changes in the network latency. Furthermore, we used the empiric results obtained using a testbed network between the cities of Dresden and Freiberg to derive basic rules explaining the observed interaction between different performance parameters. Our findings describe the interplay between the bandwidth-delay product of the network, and the size and count of Lustre RPCs in flight for unthrottled communications, the penalty introduced by the stripe count during file creation, and the overhead encountered when concurrently opening files from multiple nodes.

There are several ways in which this work could be further improved. The first one would be to investigate whether the results are still valid for other deployments of Lustre or not. Among the other things deserving a deeper look are the relation between the RPC size and its induced overhead (expressed with the constant k), and how the congestion avoidance mechanisms acting at the transport layer of the network affect the predictions of the LNET-model.

Acknowledgments. We would like to conclude this paper by expressing our gratitude to all the sponsors and our networking group for the support they provided.

References

1. NASA. Key Science System Metrics (2010), `http://earthdata.nasa.gov/about-eosdis/performance`
2. Gentzsch, W.: DEISA, The Distributed European Infrastructure for Supercomputing Applications. In: Gentzsch, W., Grandinetti, L., Joubert, G.R. (eds.) High Performance Computing Workshop. Advances in Parallel Computing, vol. 18, pp. 141–156. IOS Press (2008)
3. Simms, S.C., Davy, M., Hammond, B., Link, M., Stewart, C., Bramley, R., Plale, B., Gannon, D., Baik, M.-H., Teige, S., Huffman, J., McMullen, R., Balog, D., Pike, G.: All in a Day's Work: Advancing Data-Intensive Research with the Data Capacitor. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, SC 2006. ACM, New York (2006)

4. Simms, S.C., Pike, G.G., Balog, D.: Wide Area Filesystem Performance Using Lustre on the TeraGrid. In: Proceedings of the TeraGrid 2007 Conference (2007)
5. Simms, S.C., Pike, G.G., Teige, S., Hammond, B., Ma, Y., Simms, L.L., Westneat, C., Balog, D.A.: Empowering Distributed Workflow with the Data Capacitor: Maximizing Lustre Performance Across the Wide Area Network. In: Proceedings of the 2007 Workshop on Service-Oriented Computing Performance: Aspects, Issues, and Approaches, SOCP 2007, pp. 53–58. ACM, New York (2007)
6. Andrews, P., Kovatch, P., Jordan, C.: Massive High-Performance Global File Systems for Grid Computing. In: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, SC 2005, p. 53. IEEE Computer Society, Washington, DC (2005)
7. Filizetti, J.: Lustre Performance over the InfiniBand WAN. In: Proceedings of the 2010 Lustre User Group (2010)
8. Michael, S., Simms, S., Breckenridge III, W.B., Smith, R., Link, M.: A Compelling Case for a Centralized Filesystem on the TeraGrid: Enhancing an Astrophysical Workflow with the Data Capacitor WAN as a Test Case. In: Proceedings of the 2010 TeraGrid Conference, TG 2010, pp. 13:1–13:7. ACM, New York (2010)
9. Cai, R., Curnutt, J., Gomez, E., Kaymaz, G., Kleffel, T., Schubert, K., Tafas, J.: A Scalable Distributed Datastore for BioImaging (2008), `http://www.r2labs.org/pubs/BioinformaticsDatabase.ps`
10. Rodriguez, J.L., Avery, P., Brody, T., Bourilkov, D., Fu, Y., Kim, B., Prescott, C., Wu, Y.: Wide Area Network Access to CMS Data Using the Lustre Filesystem. Journal of Physics: Conference Series 219(7), 072049 (2010)
11. Zhao, T., March, V., Dong, S., See, S.: Evaluation of a Performance Model of Lustre File System. In: 2010 Fifth Annual ChinaGrid Conference (ChinaGrid), pp. 191–196 (July 2010)
12. Oracle, Inc. Lustre 2.0 Operations Manual (2010), `http://wiki.lustre.org/images/3/35/821-2076-10.pdf`
13. Lakshman, T.V., Madhow, U.: The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. IEEE/ACM Trans. Netw. 5, 336–350 (1997)
14. Shan, H., Shalf, J.: Using IOR to Analyze the I/O Performance for HPC Platforms. In: Cray User Group Conference (2007)