Enterprise HPC storage systems

Advanced tuning of Lustre clients

Torben Kling Petersen, PhD
HPC Storage Solutions
Seagate Technology
Langstone Road, Havant, Hampshire PO9 1SA, UK
Torben_Kling_Petersen@seagate.com

John Fragalla, MSc
HPC Storage Solutions
Seagate Technology
46831 Lakeview Blvd, Fremont, California 94538, USA
John_Fragalla@seagate.com

Abstract — **High performance compute systems place an increasing demand on the their primary connected storage systems. Modern filesystem implementations are no longer just for "scratch" only, data availability and data integrity are considered just as important as performance. In today's market place as well as in advanced research, the demands on a storage system increase along a number of fronts including capacity, data integrity, system reliability, and system manageability, in addition to an ever-increasing need for performance. Understanding the tuning parameters available, the enhancements in RAID reliability as well as possible tradeoffs between slightly opposing tuning models becomes a critical skill. The Lustre filesystem has for many years been the most popular distributed filesystem for HPC. While the Lustre community to date has been partial to older Lustre server and client releases, a number of the new features desired by many users, requires moving to more modern versions. The major dilemma historically has been that client versions such as 1.8.x perform faster than the 2.x releases, but support for modern Linux kernels is only available with recent Lustre server releases, and newly-implemented feature sets requires moving to newer client versions.**
This paper examines the client and server tuning models, security features and the advent of new RAID schemes along with their implications for performance. Specifically, when using benchmarking tools such as IOR, new testing models and parameter sets requires storage I/O benchmarking to change and more closely mimic contemporary application I/O workloads.

Keywords — HPC storage, I/O performance, PD-RAID, benchmarking.

I. Introduction

Performance of HPC storage systems has always been the main selection criteria when adding a new file system to a new or existing compute system. And as the amounts of data required for many application suites is increasing exponentially, the need for data intensive compute solution is ever increasing. Today, this applies to academic, research and commercial HPC users and the boundaries between different types of users are disappearing whether we are talking about CFD calculations for the next airliner, weather simulations or energy centric exploration.

However, a clear trend today, mirrored in many of the RFPs being issued, is the requirement of enterprise features such as HSM and backups, better management tools and requirements of enterprise level reliability, availability and serviceability (RAS) that one normally associate with enterprise storage systems from HDS or EMC.

II. Lustre Parallel File system

To achieve the required throughput of modern HPC systems, most solutions are based on the concept of a parallel file system. While there are a number of parallel file systems used in HPC solutions today, ranging from commercial and proprietary solutions such as IBM GPFS, to new concepts such as BeeGFS (aka Frauenhofer GFS) and Ceph, the open source driven Lustre currently dominates with 7 of the top 10 systems and more that 70% of the Top100 [1] on the Top500 list of supercomputers.

In addition to being the most popular parallel file system in use today, it also benefits from having a large community [2, 3] of both end users (commercial and academic) and industry partners involved in developing Lustre. With a 6 month average release cycle the current maintenance release (Lustre 2.5 released in the fall of 2013) and the latest feature release (Lustre 2.6 released in spring of 2014), an ever increasing portfolio of features and functionalities are being made available. As mentioned above, many are still performance related but a majority of the enhancements or new features are for improved data management and scalability. To benefit from new features, enhanced clients are also released, mainly for newer Linux kernel releases, but adoption to those is currently slow. One of the main reasons for this is a community wide belief that newer clients are significantly slower than the current mainstay (Lustre client 1.8.9) and the benefit of new features does not make the performance decrease worthwhile. In addition, a common practice in many datacenters today is to turn off one of the data integrity features in Lustre systems, namely in memory and over the wire checksums in order to improve performance. With current systems using ECC protected memory, this might not be a big issue but for data transmitted over the wire, turning of checksums might lead to data corruption that can occur from the network.

In addition to continuing the work presented at the IEEE HPEC conference 2013 [4], the team wanted to compare the impact of modern clients and the tuning needed to make these clients perform at peak efficiency as well as examine the impact of checksums over the wire.

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

III. File and storage system architectures

While a well-designed storage system can deliver aggregated throughputs in excess of 1 TB per second [5] the maximum performance is still limited by the slowest component in the system. Examining a solution from the application to the disk blocks reveal a number of potential areas where the ultimate performance of a solution are being limited. Factors that affect the storage performance ranges from the choice of RAID system (hardware or software based such as MD-RAID), disk protocols such as S-ATA and SAS, disk type (even within a range of say 4 TB NL-SAS drives from one of the 3 major vendors, each type can have very different I/O characteristics), firmware levels of all hardware, the components of the software stack, tuning parameters (important even on the client side) and type of interconnect to mention a few. In addition to many of the above-mentioned components, increasing disk capacities are also starting to worry many users. As disk drives now begin to approach 6 TB in capacity, traditional RAID models are starting to become a problem. While modern disk drives, despite the increase in capacity, still seem to fail with the same frequency as earlier generations, a replaced drive in a RAID-6 array now takes days rather than hours to rebuild. The solution stated by many experts is to move to a de-clustered parity RAID design

Similar to several other storage vendors, Xyratex has developed a version of parity de-clustered RAID (PD-RAID) called GridRAID. While PD-RAID is by no means a new concept [6], technology has just recently allowed developers the compute speed and capacity to perform the more complex calculations needed. While this paper is not intended to fully describe the methodology, GridRAID uses a semi-random block allocation including distributed spare blocks for faster re-builds. Simplistically, GridRAID uses an 8+2+2 algorithm (8 data, 2 parity and 2 spare blocks) per integral stripe in a given tile enabling a solution to fully utilize 41 drives within a single array. This enables parity rebuilds to complete up to 4 times faster than regular RAID-6 and significantly improves a systems mean time to repair (MTTR). While there are many different offerings of PD-RAID, few implementations use the same paradigm so comparing between the different vendors might be difficult. However, the general consensus among customers seem to be that while PD-RAID implementations provide much faster rebuilds as a result of a failed drive, there is a performance penalty to pay.

IV. Aims

As it is impossible to examine all combinations of hardware, software, interconnects, disk types etc., we have selected a few areas that we believe needed closer examination.

A. Lustre wire checksums

Lustre currently uses two software based checksum algorithms, Adler32 and CRC32, where the default algorithm in Lustre is Adler32. Different generations of Lustre clients implement this feature slightly different and so the impact on performance will differ. As a part of the current work, we examined the impact of using wire checksums with different client versions.

B. Lustre clients

As Lustre 2.x clients are considered slower than the more common Lustre 1.8.9 client, we wanted to examine the difference in performance and what the optimal tuning parameters for each client is. For users interested in utilizing features released in later versions of Lustre, moving to newer clients is necessary. In addition, users moving to more modern Linux releases (such as SuSE 11 or RH6.5) tend to use new Lustre versions with new kernels, due to improved performance in the Lustre 2.x client code, as this paper will discuss and to take advantage of the new features available in recent releases of Lustre.

C. RAID layout

GridRAID changes the total layout of a Lustre file system by using many more drives per object store target (OST) thereby reducing the total number of targets by 4x. That said, the total number of spinning drives per file system is the same and any performance difference will be a result of the software RAID functionality. The current study uses the Xyratex GridRAID model for all the benchmark activities. While this study did not specifically compare GridRAID with MD-RAID, the current results can be compared to the data presented at the IEEE HPEC 2013 conference [4].

V. System architecture

A. Storage Subsystem

This study utilized a standard Xyratex ClusterStor 6000 Lustre based storage appliance. The MDS/MGS component consisted of a pair of active-passive failover servers with a shared storage (24 disk array) acting as the systems MDT/MGT. A 5 RU enclosure, known as a Scalable Storage Unit (SSU) hosting 84 3,5' disk drive slots and two embedded OSS server modules in the back of the enclosure was used for the HA object store server pair [7]. The following storage components were used:

- 82 Western Digital Verdi near line SAS drives (3,5' form factor) with a capacity of 4 TB.
- 2 Hitachi Ralston Peak SSDs (2.5' form factor in a 3.5' carrier) with a capacity of 100 GB.

Checksums Impact (Max_RPCs_in_Flight/Max_Dirty_MB)									
Client version	Average Difference	Reads 32/128	Write 32/128	Read 64/256	Write 65/256	Read 128/512	Write 128/512	Read 256/1024	Write 256/1024
1.8.9	MB/s	203	398	3	434	71	523	31	502
	%	2%	10%	-1%	10%	0%	13%	0%	12%
2.1.6	MB/s	542	270	446	196	446	189	453	180
	%	22%	12%	19%	9%	18%	8%	9%	8%
2.4.3	MB/s	333	95	277	77	380	194	327	316
	%	7%	3%	5%	2%	7%	4%	5%	6%
2.5.1	MB/s	486	161	245	155	366	257	347	423
	%	12%	5%	5%	4%	6%	6%	6%	9%

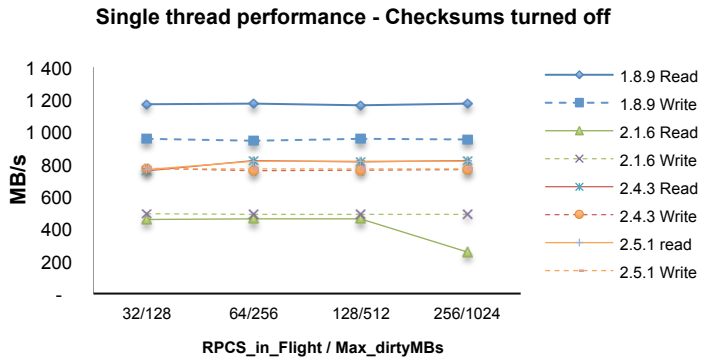

Fig 1 Results of single thread performance with checksums turned off.

The disk drives (41 per OSS) were configured as two GridRAID (8+2+2) volumes formatted with ldiskfs rendering them into 2 object store targets (OST). The 2 SSDs (formatted with RAID 1) were exclusively used for write intent bitmaps (WIBs) and for Linux journals.

The embedded server modules (2 per enclosure) are based on a single socket Intel Xeon CPU, 32 GB RAM and an onboard Mellanox CX3 FDR capable HCA. The servers used Xyratex Lustre 2.1.0.x4-74. The solution benchmarked consists of a total of 1 SSU, 2 OSS servers, and 2 OSTs. The appliance was running ClusterStor OS 1.4.0-r18794.

The storage subsystem is configured with the following Lustre parameters: write through cache enabled, read cache enabled and the read cache max filesize = 1M. The MDT, MGS and OSTs leverage software RAID and no data caching. All data is stored directly to disk and committed before sending an ACK back to the client.

B. *Clients and interconnect*

The test clients were made up of 8 x86 compute nodes with the following configuration:

- 2x Intel Westmere CPUs with 48 GB RAM
- QDR interface (MLX FW 2.9.1000)
- 2.6.32-358.el6.x86_64, SL6.5, openmpi-1.5.4-2, IOR 2.10.3, stock OFED .

All clients as well as the ClusterStor appliance were connected to a Mellanox based FDR capable core switch fabric. The following Lustre client versions were used: 1.8.9, 2.1.6, 2.4.3 and 2.5.1

VI. TEST METHODOLOGY

To define the optimal benchmark parameters, a number of tests with different parameters were used. This paper will focus on the following subset of these tests establishing both the base line as well as optimal configurations.

A. *I/O mode*

IOR supports a number of I/O modes such as Buffered I/O and Direct I/O. In addition, the benchmark suite also supports simulation of several application characteristics such as file per process (FPP) and single shared file (SSF). While we tested both buffered and direct I/O using the file per process methodology, the results reported were done using Direct I/O using FPP. Transfer size was set to 64 MB per task; blocksize was set to 1024GB for the single thread runs and 512GB for multiple tasks. Finally, we used the stonewall option, write for 4 minutes, and read for 2 minutes. For the file system, we used the default Lustre stripe size of 1 megabyte and Lustre stripe count of 1. The IOR "machinefile" was configured using 'hostname' slots=4 max_slots=12 for all tests.

B. *Run mode*

All benchmark runs were done using a fully scripted procedure for each separate test and client version. The file system was un-mounted and remounted between each run to clear out any cached information (including Linux pages) and to reset the different tuning parameters. Care was taken to not use the same specific filename twice. While the actual benchmark files were erased between each run, re-using filenames has been shown to negatively affect performance of a Lustre file system.

C. *Variable tuning parameters*

As the aim of this paper is to study the impact of different tuning parameters on the different Lustre client versions, the following parameters were changed during each specific client run:

- Max RPCs in Flight [32; 64; 128; 256]
- Max dirty MBs [128; 256; 512; 1024]
- Checksums [on; off]
- Disabling LRUs [off]

VII. RESULTS

As this paper is limited in size, we cannot report all iterations performed to find the optimal performance of the ClusterStor solution. We have therefore chosen to discuss the parameters that produce the best results.

A. *Single thread performance*

While single thread performance does not produce enough I/O to saturate a storage solution, running single thread tests does reveal the capabilities of a specific client.

The results of the single thread benchmarks (fig 1.) do highlight several interesting facts. For this particular test, changing the RPCs in flight and max dirty MBs did not have any effect at all. Secondly, the results also clearly shows that the 1.8.9 client is superior in both reads and writes while the 2.1.6 client is inferior to all other clients. The 2.4.3 and 2.5.1 clients performed identical in these tests but are still ~25% lower than the 1.8.9 client.

B. *Effects of checksums*

Enabling Adler32 wire checksums has an effect as predicted on both reads and writes (table 1) when comparing the peak results of each client. The fact that most clients are affected by between 5 and 10% match the experiences in the field. In all cases when looking at multi-client and multi-thread IOR results with checksums enabled, read had little to no impact when checksums were enabled, but performance impact to writes across all clients varied between 5% to 10%,

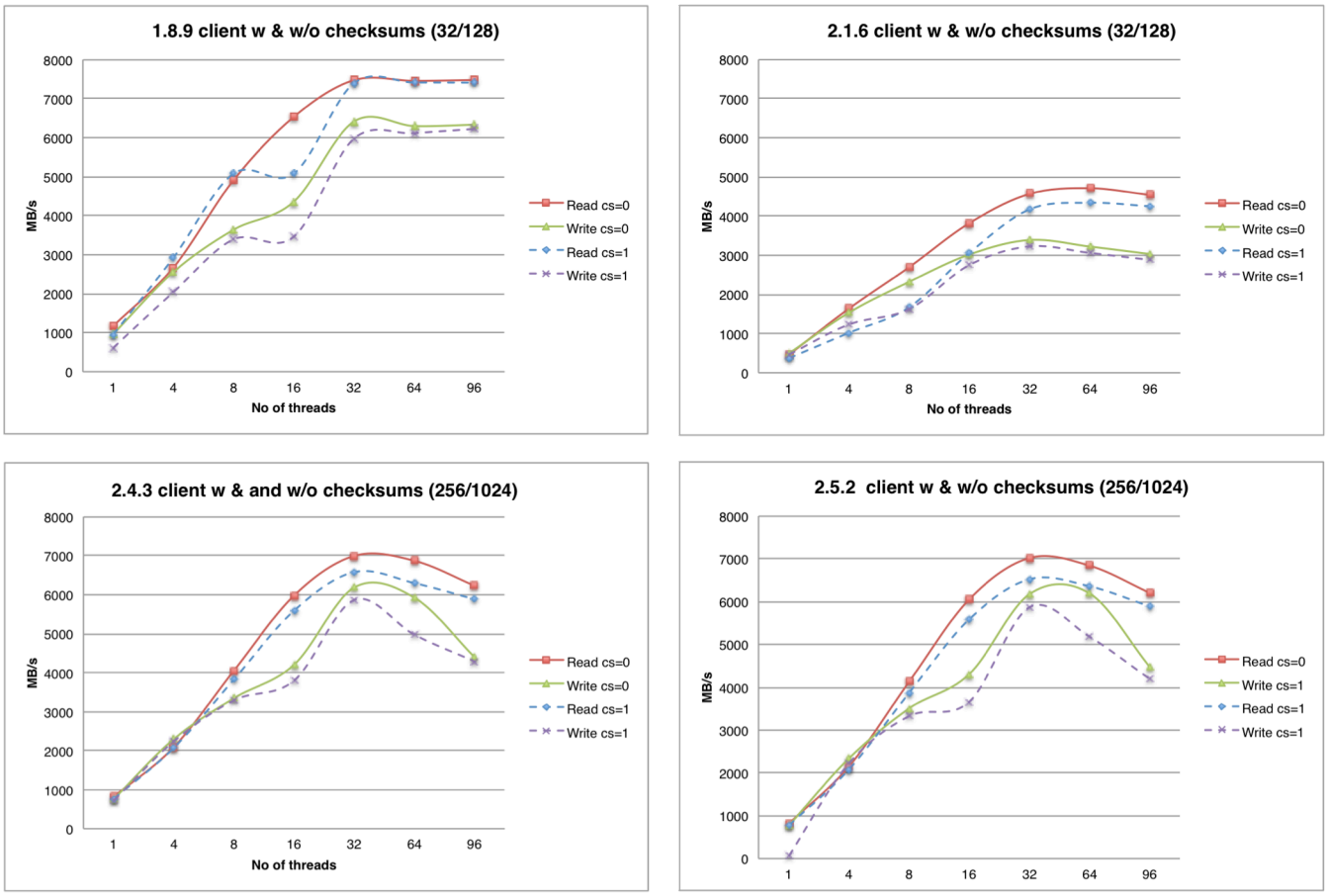

Figure 2 Effect of wire checksums on the read and write performance of different Lustre clients at the tuning parameters producing peak performance.

depending on the Lustre turning and the number of tasks across the 8 clients that were used.

C. *Client comparisons*

Comparing the four different clients again shows the same basic pattern with the 1.8.9 client being superior to all of the 2.x clients (fig. 3). That said, the peak performance of the 2.4.3 and the 2.5.1 clients were almost in line with the 1.8.9 client using 128 RPCs in flight but both clients fell off once peak was reached whereas the 1.8.9 client kept performing close to peak efficiency. Looking at the data, 2.4.3 and 2.5.1 peak performance was optimal when the number of tasks was less than the total number of CPU cores available, while 1.8.9 did not have the same performance drop off when using all CPU resources available. Also, as seen before the 2.1.6 client was significantly slower than the other clients.

When comparing all client 1.8.9 runs using different parameters for "RPCs in flight" and "Max DirtyMBs", once above 32 and 128 respectively, the 1.8.9 client version had no additional effect on increased performance, but it did have a significant performance impact when increasing the Max RPCs in Flight on 2.x Clients, especially 2.4.3 and 2.5.1 client, as seen in Figure 3. Max Dirty MB, for our test case with Direct IO operation with IOR had no impact on performance for reads.

D. *MD-RAID vs GridRAID*

Comparing the impact of using GridRAID layout with the more traditional MD-RAID format reveals that GridRAID enhances overall performance, especially on reads for Direct I/O operations. While the results are based on 2 separate benchmark runs with slightly different client versions and client types, the storage hardware is very similar. One interesting result is the number of clients required for peak performance is lower for GridRAID based systems and also worth noting that the MD-RAID peak performance required writes to use buffered I/O with reads needing direct I/O whereas Grid-RAID reach peak performance on both read and write using direct I/O [4].

VIII. Discussion

As this paper covers a lot of different areas, we will focus on the main findings and the subsequent recommendations. Furthermore, the results obtained using buffered read and write I/O is not presented, nor discussed in this paper.

Single thread performance

While HPC storage solutions are not normally geared towards single threaded I/O, a large number of existing codes do. It is therefore of great importance to a number of customers that these applications can perform well.

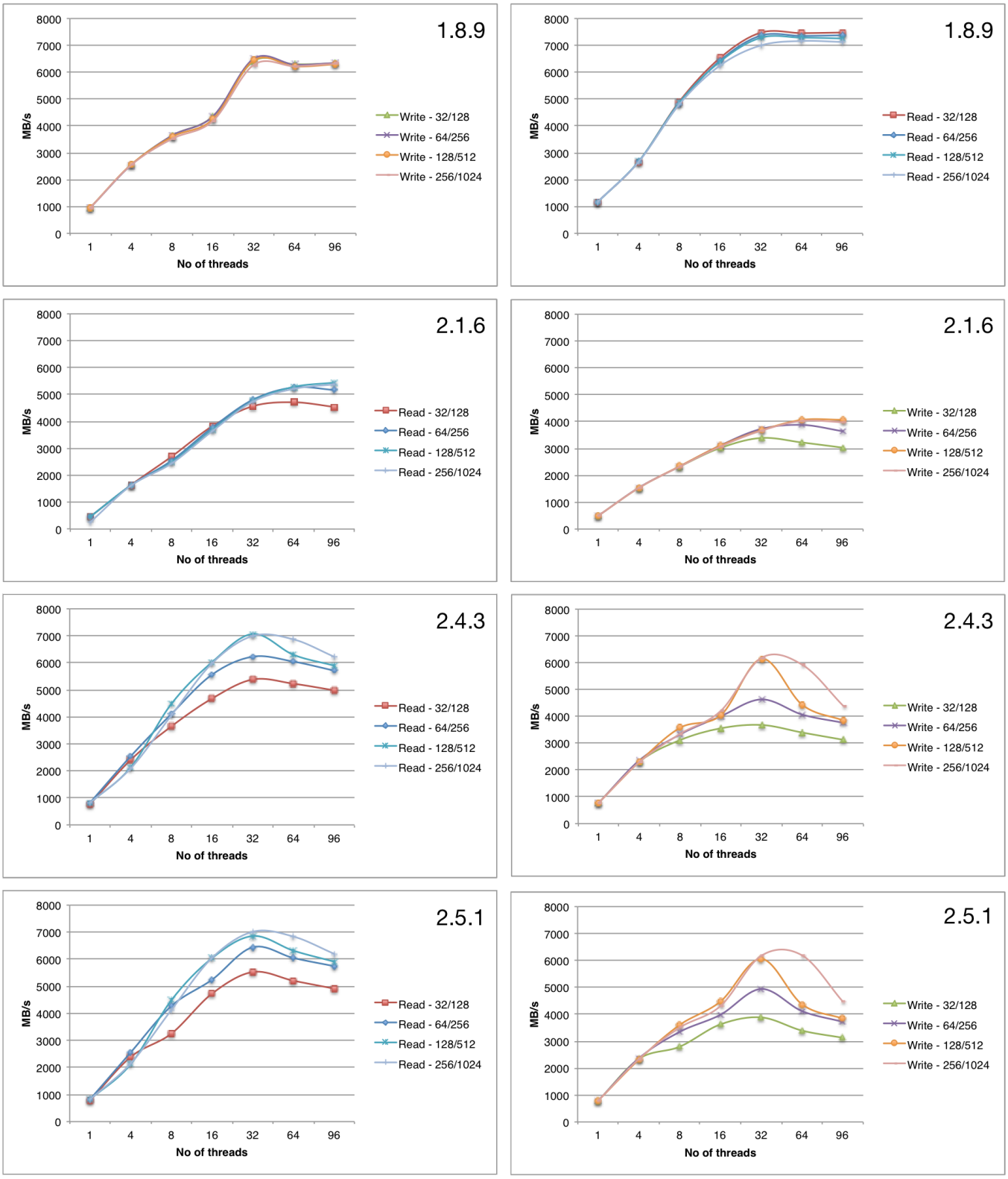

Figure 3 - Read and write performance of different clients as a result of changing the “RPCs_in_flight” and “Max_dirtyMBs” parameters

As seen with the benchmarks done, tuning parameters have little or no effect on single thread performance. More over, in this specific area, the 1.8.9 client still perform significantly better than any of the newer iterations. For reads the 2.4.3 and 2.5.1 clients is only capable of reaching about 70% of the 1.8.9 client while writes are slightly better at around 81%. This means that a user with a significant amount of single threaded I/O must make a hard choice between using features of newer clients versus performance of the specific codes.

Wire checksums

To our knowledge, very few if any studies have been done that looked specifically at the impact of wire checksums on performance. We assumed that the tribal knowledge in the community was based on observations and tests and not on assumptions. The fact that most benchmarks and acceptance tests are done with checksums turned off and then subsequently turned on for production work, means that deployments are often done on false premises. Admittedly, the impact on the 2.5.1 client nearing 6%, with the exception of one case having a 10% impact, is not insubstantial but for the 1.8.9 client the effects were just north of 10% for writes and negligible for reads. The fact that most clients are affected with between 5 and 10% at peak performance (table 1) match the experiences in the field. One could argue that on infiniband based interconnects with its peer to peer architecture, checksums are less important than on ethernet based interconnects but this does not preclude the risk of data corruption. A better approach would be to apply modern CPU technology, which allows for hardware accelerated checksum calculations such as crc32c and others but this would require modification of both the server and client code.

That said, there are project currently done within the open source community of employing hardware accelerated T10-PI checksums from the client all the way down to the disk drive. Modern HDDs already support T10-PI but again the clients and the server architecture requires modification. The projects are subject to OpenSFS roadmap and approval.

GridRAID

Many vendors today offer systems using parity de-clustered RAID (PD-RAID) in one form or another. However, while little or no actual data have been published on the performance of systems using PD-RAID, many users have experienced small to significant performance degradation using these implementations. The Xyratex implementation of PD-RAID (ie. GridRAID), however, seems to improve performance slightly, especially on reads. The fact that good performance can be achieved for both reads and writes using the same basic parameters as well as IOR direct I/O mode is a slight benefit. That said, we believe it is important to note that this pertains to benchmarking activities. For actual production systems, traditional MD-RAID implementations might be faster in some cases due to 4x the OSTs in the system, or vice versa, where GridRAID implementation might be faster than MD-RAID, it depends on application I/O. But one thing is for certain, GridRAID has up to 4x better MTTR than MD-RAID, which in itself a huge benefit for file system performance and data integrity.

The motivation for using GridRAID or not should never be for performance reasons but done if long rebuild times of a storage array is the main concern.

Clients

The main goal of this paper was to compare different client versions and find tuning parameters enabling the 2.x clients to perform similar to the 1.8.9 client. While this was indeed possible for the wider IOR runs where at least peak performance of the 2.4.3 and 2.5.1 clients was very close to the peak shown with 1.8.9. That said, the pattern of higher loads of the 2.x clients did had a tendency to taper off while the 1.8.9 maintained performance and it is obvious that this behavior requires further study.

While the results support a general move to contemporary client versions, the significant differences in performance with checksums and single threaded applications does raises the question as to whether users with mixed workloads should indeed upgrade. As it is very easy to compile the 1.8.9 client for modern Linux kernels, staying on this version is definitely an option at this point in time.

Summary

Specific recommendations on which client to use and how to tune them on a given system is difficult at best. One fact that seem clear is that the need to push client systems with more threads and larger, per thread load, is important to push modern storage solutions to peak performance. So should a user upgrade to the latest version of Lustre clients? This cannot be easily determined as every user and every site has different requirements and workloads. For users that want or need to move to more modern clients, this work does indicate that it is possible to get very similar performance from a 2.5.x client as from the tried and tested 1.8.9 client. For sites requiring the use of some of the latest features in Lustre such as HSM, distributed namespace servers (DNE), imperative recovery, layout locks to name a few, a 2.5.x client or later is required, which means additional tunings such as increasing the per client load.

ACKNOWLEDGMENT

The following people are acknowledged for their contribution to this paper: Eugene Birkine, Miro Lehocky, Meghan McClelland, Karl Merritts, Nathan Rutman, Rex Tanakit, and Chris Walker.

REFERENCES

[1] Top500. "Top500," http://www.top500.org.
[2] OpenSFS. "OpenSFS," http://opensfs.org.
[3] EOFS. "EOFS," eofs.org.
[4] T. Kling Petersen, and J. Fragalla, “Optimizing Performance of HPC Storage Systems,” in 2013 IEEE High Performance Extreme Computing Conference, Waltham, MA USA, 2013.
[5] B. Bolding. "NCSA BW," http://blog.cray.com/?p=5661.
[6] M. Holland, and G. A. Gibson, *Parity declustering for continuous operation in redundant disk arrays*, Pittsburgh, Pa.: School of Computer Science, Carnegie Mellon University, 1992.
[7] K. Claffey, A. Poston, and T. Kling Petersen, "Xyratex ClusterStor - World Record Performance at Massive Scale," http://www.xyratex.com/sites/default/files/files/field_inline_files/Xyratex_white_paper_ClusterStor_The_Future_of_HPC_Storage_1-0_0.pdf, 2012].