Empirical Analysis of a Large-Scale Hierarchical Storage System

Weikuan Yu, H. Sarp Oral, R. Shane Canon, Jeffrey S. Vetter, and Ramanan Sankaran

Oak Ridge National Laboratory
Oak Ridge, TN 37831
{wyu,oralhs,canonrs,vetter,sankaranr}@ornl.gov

Abstract. To prepare for future peta- or exa-scale computing, it is important to gain a good understanding on what impacts a hierarchical storage system would have on the performance of data-intensive applications, and accordingly, how to leverage its strengths and mitigate possible risks. To this aim, this paper adopts a user-level perspective to empirically reveal the implications of storage organization to parallel programs running on Jaguar at the Oak Ridge National Laboratory. We first describe the hierarchical configuration of Jaguar's storage system. Then we evaluate the performance of individual storage components. In addition, we examine the scalability of metadata- and data-intensive benchmarks over Jaguar. We have discovered that the file distribution pattern can impact the aggregated I/O bandwidth. Based on our analysis, we have demonstrated that it is possible to improve the scalability of a representative application S3D by as much as 15%.

1 Introduction

High Performance Computing (HPC) is quickly moving into the peta-scale computing era and beyond [2, 3]. In such large-scale environments, the composition and organization of the I/O system (and its storage hardware) is as complicated as the other system components, such as processor/memory and interconnects. The I/O system often encompasses a combination of software and hardware components such as parallel I/O programming middleware, parallel file systems, storage area networks, and the hardware storage devices. Such complex system presents a steep learning curve for application scientists to realize, appreciate and leverage the performance characteristics of storage systems, and as a result, causing a performance gap between the system deliverable I/O rate and the application achievable bandwidth.

This paper adopts a perspective from user-level programs and applications, and presents a set of experiments that analyze the performance characteristics of the hierarchical storage system on Jaguar [11]. Several earlier studies have measured the performance of different Cray XT platforms using micro-benchmarks [9, 12]. They have included relevant I/O performance information such as peak system throughput and the impact of Lustre file striping patterns. But little has been revealed on the performance implications of the storage hierarchies. Especially that little is pursued on leveraging such information for the performance benefits of scientific applications. In this paper, we focus on the evaluation, tuning and optimization of parallel I/O programs from user applications' point of view, assuming no privileges on re-configuring any system component. We first characterize the hierarchical nature of Jaguar's storage system. Then, we use micro-benchmarks to evaluate the performance of individual storage components, and the scalability of storage system for metadata- and I/O-intensive programs. Moreover, we examine the performance impacts of file distribution over Jaguar's hierarchical storage system. In doing so, we use an open-source MPI-IO library: OPAL (**OP**portunistic and **A**daptive MPI-IO Library over **L**ustre) [15], which allows dynamically-controlled file distribution by a parallel program using the MPI-IO interface [13, 14]. Furthermore, we demonstrate that leveraging the understanding of storage characteristics can directly lead to the I/O performance improvement in a combustion application, S3D [7].

The rest of the paper is organized as follows. In the next section, we provide an overview of Jaguar [11] and the organization of its storage system. Following that, we discuss our experimental analysis of parallel I/O on Jaguar in Sections 3, 4 and 5. In Section 6, we demonstrate the optimization of S3D. Section 7 concludes the paper.

2 Hierarchical Organization of Jaguar's Storage System

Jaguar's storage system is equipped with the Silicon Storage Appliance (*S2A*) from DataDirect Network (DDN), Inc. The S2A family of products has a number of key features for offering reliable, sustaining bandwidth from a large storage pool without going through a switch-based, contention-prone storage area network [5]. The current storage system deployed on Jaguar is DDN S2A 9550, a revision of S2A 9500. Each S2A consists of two Parity Processing Engines (PPEs). These PPEs horizontally parallelize the data streams to its array of Disk Controller Engines (DCE). DCEs in turn commit I/O to tiers of vertically attached SATA disks. Due to page limit, please refer to [5] for further details on the internal hierarchies of S2A.

2.1 Organization of Storage Devices and Lustre File Systems on Jaguar

Jaguar is equipped with 18 S2A storage targets, providing I/O services through Lustre file system. Each S2A target has a capacity of 32TB. We also refer to an S2A target as a ***rack*** as it forms an entire rack in the machine room. A separate rack is assigned as the storage device for metadata over Jaguar. Each S2A is equipped with 16 tiers of disks. These 16 tiers are organized into 16 LUNs (Logical Unit Number). Each LUN spans two tiers through vertical striping and disks within a single tier are allocated into two LUNs. Each LUN has a capacity of 2TBs and a 4KB block size. The cache is set to 1MB on each DCE.

Lustre [4] is a POSIX-compliant, object-based parallel file system. Lustre separates essential file system activities into three components, often referred to as clients, Meta-Data Servers (MDSes) that manage Lustre metadata, Object Storage Servers (OSSes) that serve clients' object requests, and Object Storage Targets (OSTs) that serve backend storage as objects, respectively. Three Lustre file systems on Jaguar provide the temporary scratch space for experimental data, namely, scr144, scr72a and scr72b. 72 I/O service PEs are configured as OSSes for these file systems. In addition, each file system has its own dedicated MDS service node. Every LUN from S2A 9550 is configured as a single OST for one of three Lustre file systems. For this reason, in the rest of paper we use ***OST*** and ***LUN*** interchangeable. We also use the term ***rack*** to represent all the 8 LUNs in a S2A for a Lustre file system. DDN 9550s are connected to Jaguar service nodes via direct 4Gbps fibre channel (FC) links. Fig. 1 shows the assignment of LUNs to the services nodes and Lustre file systems. The biggest file system, scr144, is equipped with 144 OSTs, i.e. 8 LUNs from every S2A 9550 storage devices, with a storage capacity of 288TBs. The other two, scr72a and scr72b, have 72 OSTs. More information on the network configuration of Jaguar can be found in [6, 11].

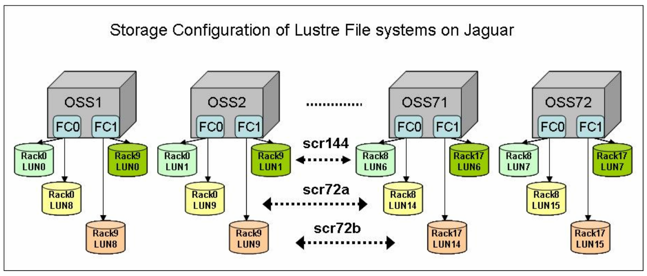


Fig. 1. Storage Organization of Lustre File systems on Jaguar

3 Bandwidth Characterization of Different Hierarchies

We start by examining the characteristics of individual components. The IOR benchmark [1] from Lawrence Livermore National Laboratory is used for these bandwidth measurements. Our experiments were conducted while other jobs were also running on Jaguar. We instructed IOR to run three iterations and reported the peak measurement. Then the same experiment was repeated 10 times. The final result was the average of 10 peak measurements.

Different Mapping of Storage Accesses – As discussed earlier, the storage system on Jaguar is presented to the parallel applications as a hierarchy of components, from individual LUNs, racks, to the entire storage subsystem. When parallel processes

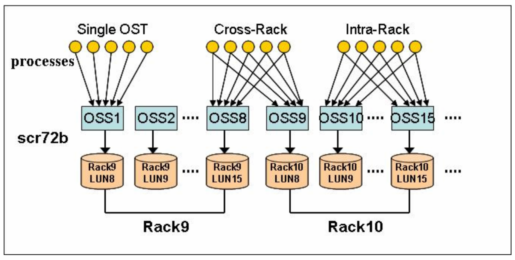
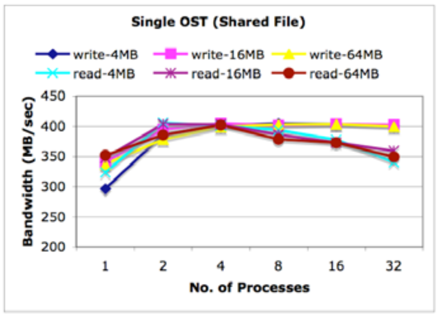


Fig. 2. Mapping Scenarios among Processes, Lustre OSSes and DDN 9550s

access files stored on the storage devices, their I/O requests are mapped to these storage components. We have categorized the scenarios as *single-OST mapping*, *intra-rack mapping* and *cross-rack mapping*. Fig. 2 illustrates these mapping scenarios on the *scr72b* Lustre file system of Jaguar. A file that stripes across all LUNs in the *scr144* file system is an extreme case of cross-rack mapping.

3.1 Single-OST

In the single-OST experiments, a varying number of parallel processes are concurrently reading from or writing to a single OST. The transfer size per client is set to vary between 4MB and 64 MB. Fig. 3(L) shows the results with a shared file. The maximum read or write bandwidths are 406 MB/s, which comes close to the peak of a single OST, with 4Gbps Fibre Channel adaptors. Comparing the read and write performance, it is evident that the performance of writes scales more gracefully under the hot-spot pressure from many processes. As also shown in the figure, the performance of parallel writes is not as sensitive to the changing transfer sizes. In addition, we have measured the performance of single-OST mapping using different Lustre stripe sizes – the unit size of objects in which Lustre stripes data to different OSTs. Not surprisingly, the stripe size does not affect the I/O performance when a file is located within a single OST.



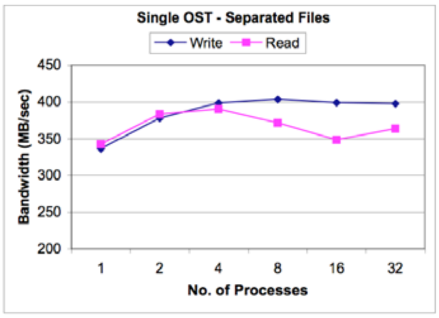


Fig. 3. Performance of Single OST: (L) Shared File; (R) One File per Process

Based on the results of Figure 3(L) (as well as those of Figure 4), we measured the performance of a single OST with separated files using only 16MB and 64MB, using the stripe size of 4MB. Fig. 3(R) shows our measurement results. Again, the performance of writes scales more gracefully to the hot-spot pressure from many processes. In these tests, both reads and writes can achieve the peak performance.

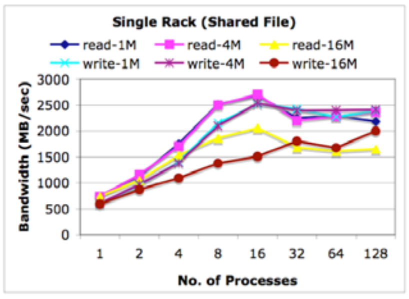
3.2 Single Rack

Single-Rack Bandwidth – We measured the peak performance of a single rack. All files are striped across 8 LUNs, i.e., all 16 tiers in an S2A target. Based on the earlier results, we have chosen the transfer sizes to be 64 MB. Fig. 4(L) shows the performance of a single rack with different Lustre stripe sizes (1MB, 4MB, 16MB and

64MB). Parallel reads provide better peak bandwidth compared to parallel writes. As also shown in the figure, the stripe sizes 1MB and 4MB are able to deliver the best performance, while 4MB is slightly better overall. The aggregated performance of a single rack increases initially with the increasing number of processes. The performance for both reads and writes reaches the peak with 16 processes. However, neither of the two can sustain the same peak performance with a strong scaling on the number of processes. As seen earlier with single OST, reads see more performance degradation due to the increasing number of processes.

We also measured the performance of a single rack with a separated file for each process, using the stripe size of 4MB. Fig. 4(R) shows our measurement results. In contrast to the single OST results, when the number of processes is less than 64, the performance of reads is significantly better than that of writes. However, as the number of processes increases, both reach a plateau at around 2500MB/sec. As shown in the figure, the peak performance is much lower than the peak physical bandwidth (8 LUNs of ~406MB/sec each). This phenomenon suggests that when all the tiers of a rack are mobilized, it does not sustain linear scaling due to potential contentions on shared data paths, such as P3E and DCE.

Intra-Rack Scaling – In this test, the targeted files are striped across multiple LUNs, starting all from the first LUN. Figures 5 and 6 show the performance of intra-rack



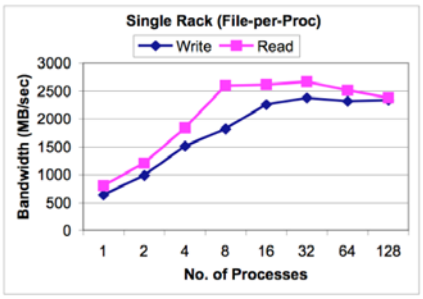
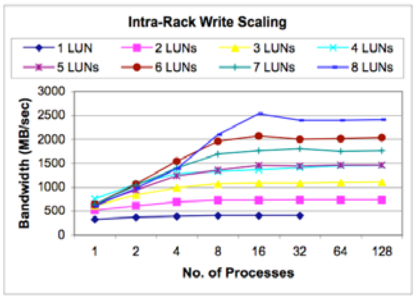


Fig. 4. Performance of a Single Rack: (L) Shared File; (R) One File-Per-Process



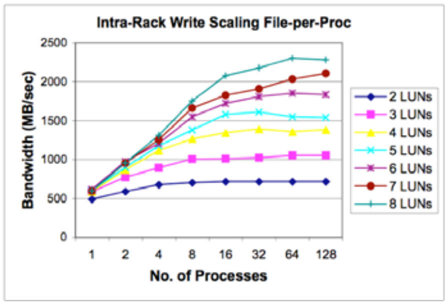


Fig. 5. Performance of Intra-Rack Scaling with Shared File

Fig. 6. Performance of Intra-Rack Scaling with one File per Process

scaling with a shared file or one file-per-processes, respectively. As shown in Figure 5, the performance of 3 LUNs is about the same with that of 4 LUNs, while the performance of 7 LUNs is rather lower than that of 6 LUNs! In contrast, these tests with one file per processes provide a different trend. The aggregated bandwidth is in general increasing along with the number of the LUNs being used. Due to page limit, only the write performance is shown. These results suggest that: (a) contentions among tiers can vary dependent on their physical locations; and (b) one file-per-process mode would help smoothen out the achievable data accesses among LUNs. Our purpose for this performance analysis is to examine the performance trend of Jaguar's storage system with various testing scenarios and shed light on what possible strategies to exploit benefits or avoid impacts. The root causes of these problems need further examination of the internal processing of storage and file system. It is not intended in this application-oriented empirical I/O performance analysis.

3.3 Cross-Rack

In view of the results with intra-rack scaling, we consider that it is possible to improve the aggregated bandwidth with 8 LUNs by distributing LUNs across different racks. Without changing the configuration of the existing configuration of Jaguar storage system, it is possible to study this by dividing the stripes of a file into two contiguous racks. We have measured the performance when different LUNs across two racks are used for storage. For example, a file with 2 stripes will utilize one LUN from each rack; a file with 3 stripes will use one LUN from a rack and two from the other, and so on. We used the OPAL library for the cross-rack experiments because it provides the convenience of arbitrary specification of striping pattern on a per-file basis. Figures 7 and 8 show the performance of cross-rack scaling (write only again) with a shared file or one file-per-process, respectively. Interestingly, the earlier situation of reverse scaling in aggregated bandwidth is no longer present. One might argue that the performance difference is solely because of the utilization of more P3E engines from multiple racks. Our further study on the impact of file distribution suggests that this is not the case, because a significant performance improvement is also possible when the same racks are employed for storage, except that the first LUN starts from a different location.

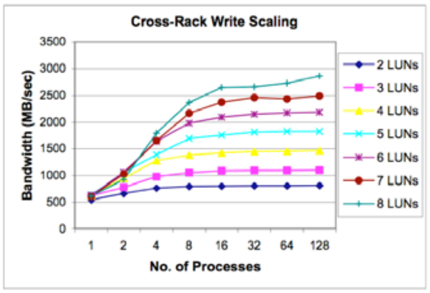


Fig. 7. Performance of Cross-rack Scaling with Shared File

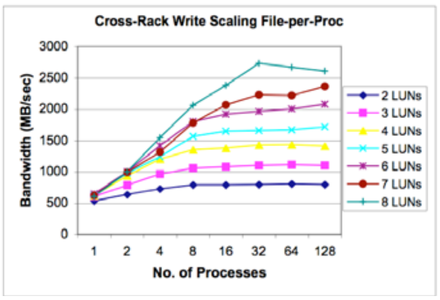


Fig. 8. Performance of Cross-rack Scaling with one File per Process

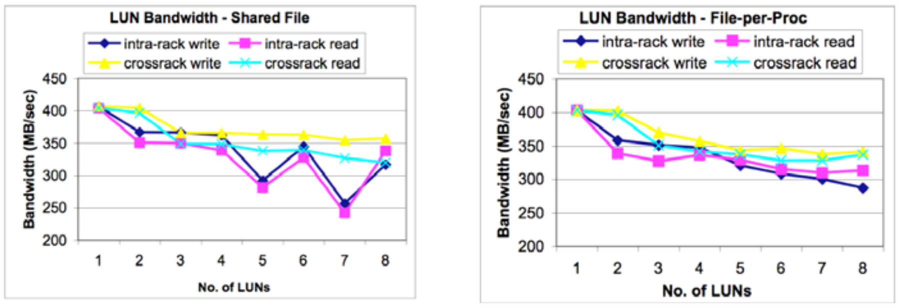


Fig. 9. Bandwidth per LUN for Intra-Rack and Cross-Rack

Achieved I/O Bandwidth per LUN – We also note the aggregated performance with cross-rack mapping rises much faster. So we take the peak bandwidth when different numbers of LUNs are used for intra-rack and cross-rack testing. Fig. 9 shows the achieved bandwidth per LUN under different mapping scenarios. It is evident that cross-rack mapping improves the performance per LUN for both the shared file and one file-per-process cases.

3.4 System Scalability

We have also measured the scalability trends with more racks of storage devices, using the largest file system scr144. Fig. 10 shows the aggregated bandwidth with a shared file. The files are created to stripe across an increasing number of racks. As shown in the figure, the performance of reads seems to scale better compared to that of writes. When files are striped across many racks, the achieved peak bandwidth per LUN is around 300MB/sec for reads, and 250MB/sec for writes. Note that this measurement is done with 1024 processes, base on the fact that too many processes to a fixed number of storage targets can lead to performance drops – a phenomenon we showed in earlier figures and also observed at other sites with similar Cray XT platforms (c.f. Figure 1 and [9]).

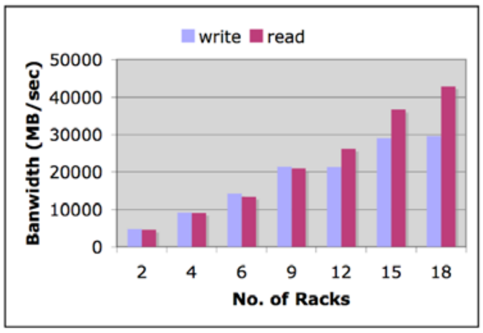


Fig. 10. Bandwidth Scalability across a Varying Number of Racks

4 Parallel File Open

Besides the need for scalable bandwidth, there is also an increasing need on the meta-data scalability of Lustre over Jaguar's storage system. In particular, the problem of creating and opening files in parallel across tens of thousands of processes has been a significant scalable challenge [10]. There is also a team from the U.S. Department of Energy laboratories working on extending the current POSIX I/O interface with routines to facilitate scalable implementation of parallel open [8]. Using the scr72b file system over Jaguar, we have measured the scalability of creating/opening files in parallel across an increasing number of racks. Fig. 11 shows the time taken for opening a shared global file across all processes and the same for opening one file per process. In our tests, all files are striped with its first OST corresponds to the first LUN of the first rack. As shown in the figure, for either mode the time increases dramatically with the number of processes, as well as with the increasing racks of storage devices. Note that, besides the drawback of bandwidth scalability shown in Section 3, the one file-per-process mode is also an order of magnitude more costly than the shared-file mode.

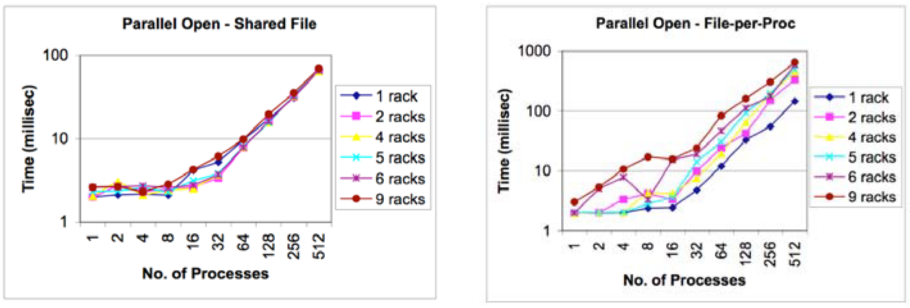
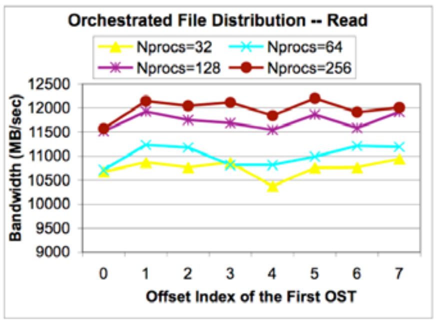


Fig. 11. Scalability of Parallel Open with an Increasing Number of Racks

5 Optimizing File Distribution

The cross-rack results suggest that the I/O performance is dependent on the physical locations of OSTs within a rack. Without changing the configuration of Jaguar's storage system in production, it is possible to examine potential benefits by adjusting the distribution of the files, especially for the one file-per-process mode. To this aim, we have used the OPAL library to create many files, one file per process. To demonstrate the possible performance difference caused by the locations within a rack, files need to be striped across less than 8 LUNs. We chose 4-stripe files in this experiment. All files are evenly distributed across all racks. However, in each test, the *offset index*–the index of a file's first OST inside a DDN rack–is globally controlled. Fig. 12 shows the performance of aggregated bandwidth with different offset indices for the first OST/LUN. Among the eight possible indices, 0 and 4 lead to the lowest performance, while all other indices can improve the aggregated bandwidth to a varying degree between 4% and 10%. The maximum process count in these tests is 256. Also note



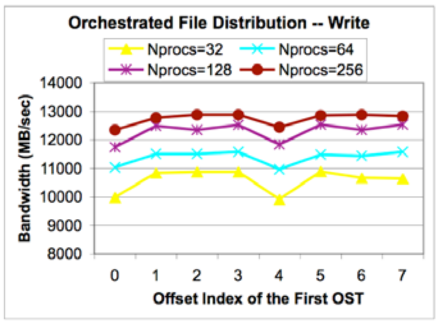


Fig. 12. Benefits of Optimized File Distribution

that in these tests all racks have the same number of LUNs being used. The only difference is on the offset indices. These results suggest that it could be beneficial for the improvement of overall bandwidth if a hierarhical storage system can be configured in a way that LUNs from different racks are interleaved with each other. This will increase the possibility of cross-rack mapping for files with small striping widths.

6 Case Study: Turbulent Combustion Simulation, S3D

Combustion simulation requires an unprecedented computational power to achieve deep understanding and fine-granularity control. S3D is a prominent application for turbulent combustion simulation. It is based on a high-order accurate, non-dissipative numerical scheme and solves the full compressible Navier-Stokes, total energy, species and mass continuity equations coupled with detailed chemistry. S3D is parallelized using a three-dimensional domain decomposition and MPI communication. Each MPI process is in charge of a piece of the three dimensional domain. All MPI processes have the same number of grid points and the same computational load. A complete course of S3D simulation can take millions of computation hours to finish. To save its intermediate simulation results, the original S3D code writes out its output from each process into individual files, periodically at every simulation step, which creates an enormous amount of data for the storage system.

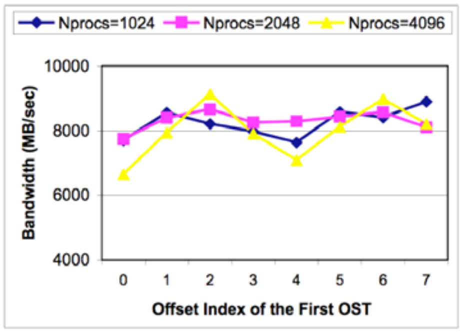


Fig. 13. Benefits of Optimized File Distribution for S3D

We exploited the benefits of optimized file distribution for S3D. With the large number of test cases, we used a smaller file system (scr72b) for this experiment. By globally orchestrating the offset index of the first OST for all files, we have measured the application-level I/O bandwidth for S3D. Fig. 13 shows that the maximum bandwidth can be achieved when the offset index is selected as 2 or 6, for S3D with 4096 processes. The performance difference can be more than 35% compared to the choices of the first OST as 0 or 4. A random choice of the first OST would be closed to the average of the eight different choices, to which the optimized choice of the first OST at 2 (or 4) can still improve an I/O bandwidth by 15%. This case study demonstrates that optimizing file distribution pattern can bring significant returns on I/O performance, especially for applications that have to maintain the one file-per-process pattern.

7 Conclusions

In this paper, we characterized the hierarchical organization of the storage system on Jaguar [11]. With an application-oriented perspective, we empirically analyzed the I/O performance of the different levels of storage components. We also examined the scalability of Jaguar storage system for metadata- and data-intensive parallel programs. We revealed that the file distribution pattern in a large-scale application can impact its aggregated I/O bandwidth. Based on our analysis, we have demonstrated its benefits to the combustion application S3D. It is also possible to optimize file distribution to gain 15% I/O performance improvement compared to the default implementation of S3D. The case study with S3D offers an example for similar optimizations to applications with similar I/O access patterns. Much of our analysis is limited with tuning and optimizations within parallel I/O library and/or user-level programs. A number of questions need to be further addressed by adjusting and reconfiguring the low-level storage hierarchies. We plan to pursue such studies with a similar, yet smaller DDN storage platform. Our empirical study presented herein could lend insights to other system practitioners on how to optimize the configuration of their storage systems.

Acknowledgments

This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725.

References

[1] IOR Benchmark, `http://www.llnl.gov/asci/purple/benchmarks/limited/ior`
[2] TOP 500 Supercomputers, `http://www.top500.org/`

[3] Simulation and Modeling at the Exascale for Energy, Ecological Sustainability and Global Security (E3SGS), Town Hall Meetings (2007), `http://hpcrd.lbl.gov/E3SGS/main.html`

[4] Cluster File System, Lustre: A Scalable, High Performance File System

[5] DataDirect Network, S2A Nearline SATA Solution Overview (2005)

[6] Fahey, M., Larkin, J., Adams, J.: I/O Performance on a Massively Parallel Cray XT3/XT4. In: 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS 2008), Miami, FL (2008)

[7] Hawkes, E.R., Sankaran, R., Sutherland, J.C., Chen, J.H.: Direct Numerical Simiulation of Turbulent Combustion: Fundamental Insights towards predictive Models. Journal of Physics: Conference Series, 65–79 (2005)

[8] High End Computing Extenstions Working Group (HECEWG), Manpage - openg (group open), `http://www.opengroup.org/platform/hecewg/uploads/40/10899/openg.pdf`

[9] Laros, J., Ward, L., Klundt, R., Kelly, S., Tomkins, J., Kellogg, B.: Red Storm IO Performance Analysis, in Cluster, Austin, TX (2007)

[10] Latham, R., Ross, R., Thakur, R.: The Impact of File Systems on MPI-IO Scalability. In: Proceedings of the 11th European PVM/MPI User Group Meeting (Euro PVM/MPI 2004), pp. 87–96 (2004)

[11] National Center for Computational Sciences, `http://nccs.gov/computing-resouces/jaguar/`

[12] Shan, H., Shalf, J.: Using IOR to Analyze the I/O Performance of XT3, in Cray User Group (CUG), Seattle, WA (2007)

[13] Thakur, R., Gropp, W., Lusk, E.: An Abstract-Device Interface for Implementing Portable Parallel-I/O Interfaces. In: Proceedings of Frontiers 1996: The Sixth Symposium on the Frontiers of Massively Parallel Computation (1996)

[14] Thakur, R., Gropp, W., Lusk, E.: On Implementing MPI-IO Portably and with High Performance. In: Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems, pp. 23–32 (1999)

[15] Yu, W., Vetter, J.S., Canon, R.S.: OPAL: An Open-Source MPI-IO Library over Cray XT. In: International Workshop on Storage Network Architecture and Parallel I/O (SNAPI 2007), San Diego, CA (2007)