A Revolutionary Approach to I/O and Storage for Exascale Scientific Computing

Jay Lofstead[*], Carlos Maltzahn[†], Scott Klasky[‡], Hasan Abbasi[‡], Qing Liu[‡]
Matthew Curry[*], Mark Ainsworth[¶], Manish Parashar[§]
[*] Sandia National Laboratories gflofst@sandia.gov
[†] University of California, Santa Cruz carlosm@soe.ucsc.edu
[‡] Oak Ridge National Laboratory
[§] Rutgers University
[¶] Brown University

ABSTRACT

The longest existing parallel I/O APIs, HDF5, PnetCDF, and NetCDF, are focused on treating the entire output from a paralell application as a single unit. These libraries work to gather the data from all processes and arrange it such that it is organized much like as if it were written from a single process. This approach, in particular the two-phase collective I/O with data sieving portion, has proven difficult to scale for 3-D domain decompositions at scale. ADIOS shifted the focus from treating the entire application output as a single entity to treating the output from each writer as a self-contained entity. This eliminated the need for the problematic two-phase collective I/O and has demonstrated good scalability well into the petascale era. Another shift is required to address the needs of exascale systems. New system I/O bandwidth is not keeping pace with the compute acceleration leading to accute bottlenecks. While "burst buffer" technology will help address the performance gap, it does not address the problem completely. This paper describes a new approach to not just an I/O library, but also the underlying storage infrastructure to address the needs of exascacle applications cognizant of projected exascale platform characteristics and in the context of current industry trends.

1. INTRODUCTION

The rest of the paper is organized as follows. A brief overview of related work is presented first in Section 2. Section 3 discusses the programmatic interface end users will see when interacting with the storage array. Section 4 discusses the challenges and opportunities for deploying scalable storage infrastructure. Section 5 discusses the quality of service features proposed for this project. Next is an exploration of the metadata challenges required to support such a system in Section 6. An initial prototype demonstration of the functionality is presented in Section 7. The paper is concluded in Section 8 with a summary of the broad issues covered in the paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
PDSW15 November 16, 2015, Austin, TX, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2505-9/13/11...$15.00.
http://dx.doi.org/xxxx.xxx/xxxxxxxxxx

2. RELATED WORK

Ceph [8] is a distributed object store and file system. It offers both a POSIX and object interface including features typically found in parallel file systems. Ceph's unique striping approach uses pseudo-random numbers with a known seed eliminating the need for the metadata service to track where each stripe in a parallel file is placed. PVFS [2] offers optimizations to reduce metadata server load, such as a single process opening a file and sharing the handle. It has been commercialized in recent years as OrangeFS. Lustre [1] has become the de facto standard on most major clusters offering scalable performance and fine-grained end-user and programmatic control over how data is placed in the storage system. GPFS [6] offers a hands-off approach for providing good performance for scaling parallel IO tasks and is used extensively by its owner, IBM. Panasas [4] seeks to offer a dynamically adaptive striping system that detects the need for additional stripes for performance and adjusts the file layout as necessary.

Other file systems, like GoogleFS [3] and HDFS [7], address distributed rather than parallel computing and cannot be compared directly. The primary difference between distributed and parallel file systems is the ability of the file system to store and retrieve data simultaneously from multiple clients, in parallel, and treat the resulting collection of pieces as a single object. Distributed file systems rely on a single client creating a file, but distributing the set of files across a wide array of storage devices. The other, popular distributed file system of note is NFS [5] that has been used for decades for enterprise file systems. These other file systems are mainly of interest in the context of the ACG features of FFSIO and will be discussed more in Section **??**.

3. END-USER API LAYER

4. STORAGE CHALLENGES AND OPPORTUNITIES

5. QUALITY OF SERVICE FEATURES

6. METADATA CHALLENGES

7. DEMONSTRATION

This stack has an early prototype implementation intended to test concepts rather than performance and scalability. It has focused on examining the interaction of the different APIs for each layer to flesh out any detailed requirements or concerns that may have been missed in the conceptualization of this IO stack. To demonstrate the viability of the IO stack described in this paper, we show some very early performance results from the untuned prototype.

8. CONCLUSIONS

9. ACKNOWLEDGEMENTS

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

10. REFERENCES

[1] P. J. Braam. The lustre storage architecture. Cluster File Systems Inc. Architecture, design, and manual for Lustre, Nov. 2002. http://www.lustre.org/docs/lustre.pdf.

[2] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, Oct. 2000. USENIX Association.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *Proceedings of the NineteenthACM Symposium on Operating Systems Principles*, pages 96–108, Bolton Landing, NY, Oct. 2003. ACM Press.

[4] Object-based storage architecture: Defining a new generation of storage systems built on distributed, intelligent storage devices. Panasas Inc. white paper, version 1.0, Oct. 2003. http://www.panasas.com/docs/.

[5] B. Powlowski, C. Juszczak, P. Staubach, C. Smith, D. Lebel, and D. Hitz. NFS version 3: Design and implementations. pages 137–152, 1994.

[6] F. Schmuck and R. Haskin. GPFS: A shared-disk file system for large computing clusters. In *Proceedings of the USENIX FAST '02Conference on File and Storage Technologies*, pages 231–244, Monterey, CA, Jan. 2002. USENIX Association.

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[8] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 2006Symposium on Operating Systems Design and Implementation*, pages 307–320. University of California, Santa Cruz, 2006.