Sunil Template

Contents

Chapter 1

The Convergence of Enterprise, Internet Scale, and High Performance Computing Storage Infrastructures

Jay Lofstead

Sandia National Laboratories

Eric Barton

Intel

Matthew Curry

Sandia National Laboratories

Carlos Maltzahn

University of California, Santa Cruz

Robert Ross

Argonne National Laboratory

Craig Ulmer

Sandia National Laboratories

Abstract

Large scale storage infrastructures have been significantly impacted by the growth in data analytics applications. High Performance Computing storage infrastructures, once the extreme end of the storage scale spectrum, must now adapt to technologies optimized for large scale data analytics applications. Hardware changes, such as storage class memory, are also affecting how the exascale storage stack will be constructed. We examine use cases, trends, convergent technologies, and new opportunities generated by this technology blending.

1.1 Introduction

HPC infrastructures have grown around the requirement to handle large, decomposed data structures for parallel computation. Single data objects may be 100s of TB spread across the entire machine. Paralell storage systems have grown trying to address performance and storage requirements while maintaining backwards compatibility with the standard POSIX interface. Unfortunately, this is proving increasingly difficult. Big data application, on the other hand, focus on searching through immense volumes of tiny items looking for patterns or correlations that may lead to insights. Some science applications, such as genomics, have a workload pattern similar to these big data applications. For these applications, parallel file systems are not well suited because of the broad, relatively continuous read demand of independent items does not benefit from the overheads of parallel file systems. Instead, distributing storage across the compute infrastructure makes more sense. With pressures to effectively leverage a single platform for these disparate workloads and the shifting storage market, new considerations for how to design storage resources for extreme scale compute systems must be made.

Traditionally, the HPC market has focused on supporting coherent and consistent output methods from parallel sources to parallel targets. This requirement is driven from validating that the output of a single item is complete and correct. Largely, the workload is write-intensive during the expensive, at scale computation process with a read-intensive phase lasting months on cheaper machines or at small scale with low priority. File systems like the dominant Lustre [2] and GPFS [5] systems have been carefully optimized to address these workloads.

The big data market has opposite priorities. The big computation phase requires reading in large data quantities for processing at scale. The output from this process can be handled at much smaller scale later and is orders

of magnitude smaller. Given the small item focus, the overhead inherent in coherent and consistent storage for write intensive workloads is both unnecessary and a heavy cost. Instead, systems like HDFS [7] dominate. These work by storing files that will be read for processing throughout the compute area including assumption that failures are common prompting replication.

The output from intitial stream or file processing for the big data workloads use distributed object-based storage technology. It offers independent, uncoordinated data access with a simplistic key search space for subsequent analysis. The profit potential for this market has caused an explosion in specialized products aimed at accelerating this processing style. For small enough data sets, in memory object stores like memcached and Radius dominate. For larger data sets, approaches like Google's Big Table are accomplishing the same function. There are also hardware products targeting this market segment, such as Kinetic [6], offering a native object interface for the devices connected directly to a network.

Adding complexity to this storage environment is the relentless performance improvements and cost reductions for solid state storage, like NAND-based flash memory. These devices have already rendered 15,000 RPM disk drives obsolete. The 10,000 RPM disk drives will not survive for more than a few more years. New disk technology like shingled drives [9] offer a path for disks to survive longer. The enormous capacities for read intensive, write infrequently workloads is very attractive for many communities. For example, storing images created sequentially for later read-intensive processing can yield a better cost/performance balance.

This chapter investigates these new technologies and how they affect extreme scale computing. We evaluate how the HPC environment can and must adapt to this new storage envionment to address future computing needs and to take advantage of the different kinds of technology being developed. We will also consider the planned reintegration of large scale computing from the split of big data applications from simulation-based computing with both the necessary and forced integration of these large, expensive platforms for multi-use.

1.2 Existing File Systems

HDFS developed to support the Hadoop implementation of the MapReduce system. It offers a distributed, replicated file store optimized to support the MapReduce processing configuration. Each file is small enough to fit on a single device. Ceph also addresses this distributed computing infrastructure, but with a different emphasis. It seeks to offer scale out performance for objects across a storage infrastructure. With the rise of cloud systems using object-based storage, such as Amazon's S3, the interaction style offered by

Ceph has been adopted for similar workloads. Ceph has features to handle storage devices failing and new ones joining a running system without interruption. Ceph offers a complete file system including metadata management support as well as an object system for users. GlusterFS focuses on providing a network attached storage interface to storage distributed throughout a cluster. Rather than providing metadata services, GlusterFS relies on the underlying storage file system for most basic capabilties, such as security.

Parallel file systems are optimized to support large files that must be spread across mutliple devices. For example, a 100 TB file cannot fit on any current storage device and cannot be stored with any performance. Parallel file systems solve this problem by using multiple devices spread across multiple servers together as if they are a single device. Data is striped across devices, all of which can be written to or read from simultaneously. This parallel access offers aggregate performance enabling manipulating very large files with reasonable performance. Because of these characteristics, parallel file systems are deployed on most simulation intensive large scale compute systems to handle the large single object output characteristic of these applications. Lustre is arguably the most popular parallel file system appearing on a majority of the Top500 machines. IBM's GPFS is increasing in popularity as optimizations focusing on addressing big data workloads are incorporated. PVFS offers a rethinking of some of Lustre's earlier limitations to give better scalability characteristics.

Discussion
The different optimizations distributed vs. parallel file systems offer are at the cost of supporting the other kind of workload. As was mentioned above, parallel file systems aim to support very large objects and aggregate simultaneous writing and reading for a single object across the array. For workloads consisting of entirely small objects, this functionality and overhead is a cost. Similarly, the inability of the distributed file systems to handle arbitrarily large objects and massive parallel simultaneous access to a single object make them unsuitable for simulation workloads.

For both workloads, a more flexible object-based interface are being considered. This is discussed more below.

1.3 Object-Based Stores

The earliest object store is probably the Wisconsin Storage System [3] published in 1985. It offered a general storage infrastructure for both databases and file systems. Many current systems were built using a similar infrastructure, such as Lustre [2], GPFS [5], and Panasas [8]. The general idea is to offer s a standardized way to address an arbitrary storage space with a key-based

access. These object storage systems assume that some sort of structure is imposed on top to track what objects correspond to which user items.

While object storage may have been used behind the scenes for years, raw object storage exposed to the end-user programmer did not into vogue until the big data era arrived. System developers for big data processing realized that the overhead imposed by the object management forced serialized or at least coordinated access. By shifting the mapping load to the end-user programmer level and using the object storage layer directly, greater perceived performance can be achieved. In many cases, by stripping down the requirements to the absolute minimum required semantics for a particular application, actual performance gains are achieved. The explosion of specialized storage systems like HDFS and GFS represent this model. Key for this model achieving performance is the ability of each process to work independently without any required consistency or coherence with neighbor processes potentially working on part of the same data set. The dominant object stores are systems like Memcached [4]. This stands in stark contrast to how supercomputing applications generally operate.

The supercomputing domain maintained the consistency requirements due to the bulk synchronous parallel processing. Instead, parallel, that is coordinated, file systems were embraced. The challenge today is that parallel file systems are having difficulty effectively scaling to handle the IO demands.

Here we want to talk about how object-based key-value stores are used for big data applications summarizing the specific features that identify this market segment.

1.3.1 HPC Oriented Object Stores

Parallel file systems inherently have an object-like layer beneath the surface. The requirement to spread a single file across multiple devices for capacity reasons alone prompts this approach. The actual implementation may vary, such as using individual files within a local file system, each representing part of parallel file. Popular examples include Lustre [2] and GPFS [5].

In some cases, directly using a key-value store for HPC applications is being considered [10]. The next generation Lustre project is also considering a key-value infrastructure [1] to address performance challenges.

The real challenge with key-value stores for HPC applications is the metadata management. All of these projects have taken a similar step to the big data application is that the applications are required to manage the object list to determine what data is stored in which object.

1.4 Next Generation HPC Storage Systems

Here we want to talk about, at a proposal sort of level, what we think needs to be done.

Talk about the disconnect between metadata and storage and the complications it introduces and some ideas on what we plan to do about it.

Talk about the major efforts

1.4.1 Lustre/DAOS

The FFSIO phase II sort of info to start. Keep it acceptable. This seems to me to be just a pure object store.

1.4.2 Kelpie/Data Warehousing

Native multi-level key-value store

1.4.3 Hybrid Models

SSIO project from ORNL/Sandia

1.5 Conclusions

This is our overall view on things

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.

Bibliography

[1] Eric Barton. Lustre*-fast forward to exascale. *Lustre User Group Summit*, 2013.

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

[3] H-T. Chou, David J. Dewitt, Randy H. Katz, and Anthony C. Klug. Design and implementation of the wisconsin storage system. *Software: Practice and Experience*, 15(10):943–962, 1985.

[4] Brad Fitzpatrick. Distributed caching with memcached. *Linux journal*, 2004(124):5, 2004.

[5] Frank Schmuck and Roger 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, January 2002. USENIX Association.

[6] Segate. The segate kinetic open storage vision. http://www.seagate.com/tech-insights/kinetic-vision-how-seagate-new-developer-tools-meets-the-needs-of-cloud-storage-platforms-master-ti/, 2014.

[7] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert 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] Brent Welch, Marc Unangst, Zainul Abbasi, Garth A. Gibson, Brian Mueller, Jason Small, Jim Zelenka, and Bin Zhou. Scalable performance of the panasas parallel file system. In Mary Baker and Erik Riedel, editors, *Proceedings of the USENIX FAST'08Conference on File and Storage Technologies*, pages 17–33. USENIX, February 2008.

[9] Roger Wood, Mason Williams, Aleksandar Kavcic, and Jim Miles. The feasibility of magnetic recording at 10 terabits per square inch on conventional media. *Magnetics, IEEE Transactions on*, 45(2):917–923, 2009.

[10] Yanlong Yin, Antonios Kougkas, Kun Feng, Hassan Eslami, Yin Lu, Xian-He Sun, Rajeev Thakur, and William Gropp. Rethinking key-value store for parallel i/o optimization. In *Data Intensive Scalable Computing Systems (DISCS), 2014 International Workshop on*, pages 33–40. IEEE, 2014.