Storage Support for Data-Intensive Scientific Applications on the Cloud

Dongfang Zhao and Ioan Raicu

Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

dzhao8@iit.edu, iraicu@cs.iit.edu

ABSTRACT

This paper presents a new storage architecture to address the I/O bottleneck of conventional HPC architecture. We describe how to design and implement a real system of such architecture, namely FusionFS, as well as the unique features that are not supported in conventional storage systems. We then explain the need for FusionFS to be ported to the Cloud, and point out the directions and steps for the FusionFS migration from HPC systems to the Cloud.

1. INTRODUCTION

State-of-the-art high-performance computing (HPC) storage subsystems for scientific computing are mainly comprised of the parallel filesystems (for example, GPFS [1]) deployed on remote storage servers. Many believe that this design would not meet the I/O requirement of the emerging exascale computing because of the segregation of compute and storage resources. Indeed, our simulation [2] predicts, quantitatively, that the system availability would go towards zero at exascale.

The authors have proposed a storage architecture with node-local disks for HPC systems [3]. Although co-locating compute and storage is not a new idea in general, it has not been widely adopted in scientific computing. We have built a node-local filesystem, FusionFS, with two major principles: maximal metadata concurrency and optimal file write, both of which are crucial to scientific applications. FusionFS has been deployed on an IBM Blue Gene/P supercomputer at 16K nodes. FusionFS's performance is orders of magnitude faster than other major file systems such as GPFS, PVFS, HDFS, and S3.

As many scientific applications are moving from the conventional HPC systems to the Cloud platforms, we believe it is critical to provide a Cloud counterpart of FusionFS. Although FusionFS has been extremely successful in HPC systems, it is unclear whether it would provide a superior performance than the state-of-the-art, such as HDFS [4]. Therefore, we propose to customize and evaluate FusionFS on the Cloud so that scientific applications deployed on the Cloud will also benefit from FusionFS's I/O performance.

2. FUSION DISTRIBUTED FILE SYSTEM

We design and implement a filesystem prototype, the Fusion distributed filesystem (FusionFS), to justify the superiority of node-local distributed filesystems over remote parallel filesystems. FusionFS is built from the ground up with the following two assumptions: (1) it is efficient for small- and medium-sized files, i.e., metadata-intensive operations, and (2) file write should be optimized. Neither of above assumptions fits in the scope of Cloud Computing, where files are assumed to be large in size and file read is typically more frequent than file-write (write-once-read-many).

We achieve the first goal by distributing file metadata (via a distributed hashtable [5]) to all compute nodes. Experimental results show that FusionFS metadata rate outperforms GPFS by more than one order of magnitude [6]. The second goal, i.e., write optimization, is achieved by writing data locally (if possible), where we design multiple file transfer protocols. In terms of performance, we deploy FusionFS on a 16K-node IBM Blue Gene supercomputer, and observe 2.5 TB/s aggregate throughput [7].

2.1 Hybrid and Cooperative Caching

When the node-local storage capacity is limited, remote parallel filesystems should coexist with FusionFS to store large-sized data. In some sense, FusionFS is regarded as a caching middleware between the main memory and remote parallel filesystems. We are interested in what placement policies (i.e., caching strategies) are beneficial to HPC workloads. We have tried both a user-level caching middlewere on every compute node (HyCache [8]) and a cooperative caching mechanism across all the compute nodes called HyCache+ [9].

2.2 Accesses to Compressed Data

Conventional data compression embedded in filesystems naively applies the compressor to either the entire file or every block of the file. Both methods have limitations on either inefficient data accesses or degraded compression ratio. We introduce a new concept called virtual chunks, which enable efficient random accesses to the compressed files while retaining high compression ratio. The key idea [10] is to append additional references to the compressed files so that a decompression request could start at an arbitrary position.

2.3 Space-Efficient Data Reliability

The reliability of distributed filesystems is typically achieved through data replication. That is, a primary copy serves most requests, and there are a number of backup copies (replicas) that would become the primary copy upon a failure. As a more efficient manner, we integrated GPU-accelerated erasure coding to FusionFS and report the performance in [11].

2.4 Distributed Data Provenance

The traditional approach to track application's provenance

is through a centralized database. To address this performance bottleneck on large-scale systems, in [12] we propose a lightweight database on every compute node. This allows every participating node to maintain its own data provenance, and results in highly scalable aggregate I/O throughput. We also explore the feasibility of tracking data provenance in a completely distributed manner in [13].

3. PROPOSED WORK

We plan to customize and evaluate FusionFS on the Cloud in three main directions: integration with data management, extreme-scale simulation, and the ability of FusionFS to adjust itself.

3.1 Integration with data management

We will integrate popular data management frameworks, such as data-aware scheduling [14, 15], into FusionFS. The goal is to have a full software stack for scientific computing. The state-of-the-art solution on the Cloud, namely MapReduce-HDFS, is originally crafted for the workloads that are quite different than scientific applications.

3.2 Extreme-scale simulation

We will deploy FusionFS on the Cloud at the largest possible scale and evaluate its performance with a variety of benchmarks. The traces we gathered from these benchmarks will be used to tune the parameters of a FusionFS-based simulator. We will then simulate real-world workloads on FusionFS in order to study its viability at the scales forthcoming in the next decade.

3.3 Self-adjustment

We plan to employ machine learning techniques (for example, incremental algorithms [16–18] to better understand and hopefully predict the I/O workloads so that a data-aware caching mechanism autonomously adjust the file placement. To the best of our knowledge, such an intelligent file system has not existed. This will also open the door of many interdisciplinary collaborations between system researchers, statisticians, and domain scientists.

Acknowledgement

The FusionFS project was supported in part by the National Science Foundation under awards OCI-1054974 (NSF CAREER).

References

[1] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST)*, 2002.

[2] D. Zhao, D. Zhang, K. Wang, and I. Raicu, "Exploring reliability of exascale systems through simulations," in *Proceedings of the 21st ACM/SCS High Performance Computing Symposium (HPC)*, 2013.

[3] I. Raicu, I. T. Foster, and P. Beckman, "Making a case for distributed file systems at exascale," in *Proceedings of the third international workshop on Large-scale system and application performance*, 2011.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of IEEE Symposium on Mass Storage Systems and Technologies*, 2010.

[5] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "ZHT: A lightweight reliable persistent dynamic scalable zero-hop distributed hash table," in *Proceedings of IEEE International Symposium on Parallel and Distributed Processing*, 2013.

[6] D. Zhao and I. Raicu, "Distributed file systems for exascale computing," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC '12), doctoral showcase*, 2012.

[7] D. Zhao, Z. Zhang, X. Zhou, T. Li, K. Wang, D. Kimpe, P. Carns, R. Ross, and I. Raicu, "FusionFS: Toward supporting data-intensive scientific applications on extreme-scale distributed systems," in *Proceedings of IEEE International Conference on Big Data*, 2014.

[8] D. Zhao and I. Raicu, "HyCache: A user-level caching middleware for distributed file systems," in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, 2013.

[9] D. Zhao, K. Qiao, and I. Raicu, "Hycache+: Towards scalable high-performance caching middleware for parallel file systems," in *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2014.

[10] D. Zhao, J. Yin, and I. Raicu, "Improving the i/o throughput for data-intensive scientific applications with efficient compression mechanisms," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC '13), poster session*, 2013.

[11] D. Zhao, K. Burlingame, C. Debains, P. Alvarez-Tabio, and I. Raicu, "Towards high-performance and cost-effective distributed storage systems with information dispersal algorithms," in *Cluster Computing, IEEE International Conference on*, 2013.

[12] C. Shou, D. Zhao, T. Malik, and I. Raicu, "Towards a provenance-aware distributed filesystem," in *5th Workshop on the Theory and Practice of Provenance (TaPP)*, 2013.

[13] D. Zhao, C. Shou, T. Malik, and I. Raicu, "Distributed data provenance for large-scale data-intensive computing," in *Cluster Computing, IEEE International Conference on*, 2013.

[14] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang, and I. Raicu, "Optimizing load balancing and data-locality with data-aware scheduling," in *Proceedings of IEEE International Conference on Big Data*, 2014.

[15] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde, "Swift: Fast, reliable, loosely coupled parallel computation," in *Proceedings of the 2007 IEEE Congress on Services*, 2007.

[16] D. Zhao and L. Yang, "Incremental isometric embedding of high-dimensional data using connected neighborhood graphs," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 31, no. 1, Jan. 2009.

[17] R. Lohfert, J. Lu, and D. Zhao, "Solving sql constraints by incremental translation to sat," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2008.

[18] D. Zhao and L. Yang, "Incremental construction of neighborhood graphs for nonlinear dimensionality reduction," in *Proceedings of International Conference on Pattern Recognition*, 2006.