Storage Support for Data-Intensive Applications on Extreme-Scale HPC Systems

Dongfang Zhao and Ioan Raicu

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

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

1. INTRODUCTION

Many believe that current high-performance computing (HPC) storage systems would not meet the I/O requirement of the emerging exascale computing because of the segregation of compute and storage resources. Indeed, our simulation predicts, quantitatively, that the system availability would go towards zero at exascale. This work proposes a storage architecture with node-local disks for HPC systems. Although collocating compute and storage is not a new idea in general, it has not been adopted in HPC systems. We build a node-local filesystem, which we called FusionFS, with two major design principles: maximal metadata concurrency and optimal file write, both of which are crucial to HPC applications. We also discuss FusionFS's integral features such as hybrid and cooperative caching, efficient accesses to compressed files, space-efficient data redundancy, and distributed provenance tracking. We have evaluated FusionFS on petascale supercomputers with 64K-cores and compared its performance with major storage systems such as GPFS, PVFS, HDFS, and S3.

1.1 Limitations of Conventional Architecture

In order to understand HPC storage performance in the emerging exascale computing, we design and implement a simulator [1] validated by real application traces. We scale simulations of synthetic workloads and IBM Blue Gene logs to 2-million nodes, and find that conventional parallel filesysetms on remote storage nodes would result in zero system availability and efficiency. Nevertheless, result shows that a node-local distributed filesystem is promising to achieve highly scalable I/O throughput and efficient checkpointing.

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 [2]) to all compute nodes. Experimental results show that FusionFS metadata rate outperforms GPFS by more than one order of magnitude [3]. 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 [4].

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.

Our first attempt is a user-level caching middlewere on every compute node, assuming a memory-class device (for example, SSD) is accessible along with a conventional spinning hard drive. That is, each compute node is able to manipulate data on hybrid storage systems. The middleware, named HyCache [5], speeds up HDFS by up to 28%.

Our second attempt is a cooperative caching mechanism across all the compute nodes, called HyCache+ [6]. HyCache+ extends HyCache in terms of network storage support, higher data reliability, and improved scalability. In particular, a two-stage scheduling mechanism called 2-Layer Scheduling (2LS) is devised to explore the data locality of cached data on multiple nodes. HyCache+ delivers two orders of magnitude higher throughput than the remote parallel filesystems, and 2LS outperforms conventional LRU caching by more than one order of magnitude.

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 [7] is to append additional references to the compressed files so that a decompression request could start at an arbitrary position. Current system prototype [8] assumes the references are equidistant, and experiments show that virtual chunks improve random accesses by 2X speedup.

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.

One concern with the conventional approach is its space efficiency; for example, two replicas imply poor 33% space efficiency. On the other hand, erasure coding has been proposed to improve the space efficiency; unfortunately it is criticized on its computation overhead. We integrated GPU-accelerated erasure coding to FusionFS and report the performance in [9]. Results showed that erasure coding could improve FusionFS performance by up to 1.82X.

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 [10] 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. Admittedly, an obvious drawback of this approach is on the interaction among multiple physical databases: the provenance overhead becomes unacceptable when there is heavy traffic among peers.

To address the above drawback, we explore the feasibility of tracking data provenance in a completely distributed manner in [11]. We replace the database component by a graph-like hashtable data structure, and integrate it into the FusionFS filesystem. With a hybrid granularity of provenance information on both block- and file-level, FusionFS achieves over 86% system efficiency on 1,024 nodes. A query interface is also implemented with small performance overhead as low as 5.4% on 1,024 nodes.

3. CONCLUSION AND FUTURE WORK

We present a high-level introduction to FusionFS, the Fusion distributed file system. We report its I/O performance on up to 16K compute nodes of a leadership-class supercomputer; its excellent scalability shows its potential for the emerging exascale systems. The uniqueness of FusionFS lies in its highly scalable metadata and file-write performance. We also discuss its integral features such as cooperative caching, efficient accesses to compressed data, space-efficient data reliability, and distributed data provenance.

There are three main directions of FusionFS's future work. First, we are integrating popular data management frameworks, such as data-aware scheduling [12], into FusionFS. Second, we will simulate real-world workloads with the CODES framework [13] in order to study the viability of FusionFS at exascales. Third, we plan to employ machine learning techniques (for example, incremental algorithms [14–16] to better understand and hopefully predict the I/O workloads so that a data-aware caching mechanism autonomously adjust the file placement.

Acknowledgment. This work was supported in part by the National Science Foundation under awards OCI-1054974.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] D. Zhao, J. Yin, K. Qiao, and I. Raicu, "Virtual chunks: On supporting random accesses to scientific data in compressible storage systems," in *Proceedings of IEEE International Conference on Big Data*, 2014.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] J. Cope, N. Liu, S. Lang, P. Carns, C. Carothersy, and R. B. Ross, "CODES: Enabling co-design of multi-layer exascale storage architectures," in *Workshop on Emerging Supercomputing Technologies*, 2011.

[14] 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.

[15] 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.

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