High-Performance Storage Support for Scientific Applications on the Cloud

Dongfang Zhao*, Xu Yang*, Iman Sadooghi*, Gabriele Garzoglio†, Steven Timm†, Ioan Raicu*‡

*Illinois Institute of Technology †Fermi National Accelerator Lab ‡Argonne National Lab

{dzhao8,xyang56,isadooghi}@iit.edu, {garzogli,timm}@fnal.gov, iraicu@cs.iit.edu

ABSTRACT

Although cloud computing has become one of the most popular paradigms for executing data-intensive applications (for example, Hadoop), the storage subsystem is not optimized for scientific applications. We believe that when executing scientific applications in the cloud, a node-local distributed storage architecture is a key approach to overcome the challenges from the conventional shared/parallel storage systems. We analyze and evaluate four representative file systems (S3FS, HDFS, Ceph, and FusionFS) on three platforms (Kodiak cluster, Amazon EC2 and FermiCloud) with a variety of benchmarks to explore how well these storage systems can handle metadata intensive, write intensive, and read intensive workloads.

1. INTRODUCTION

While cloud computing has become one of the most prevailing paradigms for data-intensive applications, many legacy scientific applications are still struggling to leverage this new paradigm. One issue for scientific applications to be deployed on the cloud lies in the storage subsystem. For instance, HDFS [1] and its variations [2] are popular file systems in cloud platforms for many workloads in data centers built on commodity hardware; yet, many scientific applications deal with a large number of small files [3] that are not well supported by the data parallelism of HDFS. The root cause of the storage discrepancy between scientific applications and cloud computing stems from their assumptions: Scientific applications assume their data to be stored in remote parallel file systems, while cloud platforms provide node-local storage available on the virtual machines.

This paper shares our view on how to design storage systems for scientific applications on the cloud. Based on the literature and our own experience on cloud computing and high-performance computing (HPC) in the last decade, we believe that cloud storage would need to provide the following three essential services for scientific applications:

1. Scalable metadata accesses. Conventional centralized mechanisms for managing metadata on cloud computing, such as GFS [2] and HDFS [1], would not suffice for the extensive metadata accesses of scientific applications.

2. Optimized data write. Due to the nature of scientific applications, checkpointing is the main approach to achieve fault tolerance. This implies that the underlying storage system is expected to be efficient on data write as checkpointing per se involves highly frequent data write.

3. Localized file read. After a failure occurs, some virtual machines (VM) need to restart. Instead of transferring VM images from remote file systems, it would be better to keep a local copy of the image and load it from the local disk when needed.

In order to justify the above arguments, we analyze four representative file systems. Two of them are originated from cloud computing (S3FS [4], HDFS [1]). S3FS is built on top of the S3 storage offer by Amazon EC2 cloud as a remote shared storage with the added POSIX support from FUSE [5]. HDFS is an open-source clone of Google File System (GFS [2]) without POSIX support. The other two file systems were initially designed for high-performance computing (Ceph [6], FusionFS [7]). Ceph employs distributed metadata management and the CRUSH [8] algorithm to balance the load. FusionFS is first introduced in [9] and supports several unique features such as erasure coding [10], provenance [11], caching [12, 13], and compression [14, 15]. This study involves three test beds: a conventional cluster Kodiak [16], a public cloud Amazon EC2 [17], and a private cloud FermiCloud [18].

2. DISTRIBUTED METADATA ACCESS

State-of-the-art distributed file systems on cloud computing, such as HDFS [1], still embrace the decade-old design of a centralized metadata server. The reason of such a design is due to the workload characteristic in data centers. More specifically, a large portion of workloads in data centers involve mostly large files. For instance, HDFS has a default 64 MB chunk size (typically 128 MB though), which implicitly implies that the target workload has many files larger than 64 MB; HDFS is not optimized for files smaller than 64 MB. Because many large files are expected, the metadata accesses are not intensive and one single metadata server in many cases is sufficient. In other words, a centralized metadata server in the conventional workloads of cloud computing is not a performance bottleneck.

The centralized design of metadata service, unfortunately, would not meet the requirement of many HPC applications that deal with a larger number of concurrent metadata ac-

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 ACM 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.
ScienceCloud'15, June 16, 2015, Portland, Oregon, USA.
Copyright © 2015 ACM 978-1-4503-3570-6/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2755644.2755648.

cesses. HPC applications are, in nature, highly different than those conventionally deployed on cloud platforms. One of the key differences is file sizes. For instance, Welch and Noer [19] report that 25% – 90% of all the 600 million files from 65 Panasas [20] installations are 64 KB or smaller. Such a huge number of small files pose a significantly higher pressure to the metadata server than the cloud applications. A single metadata server would easily become the bottleneck in these metadata-intensive workloads.

A distributed approach to manage metadata seems to be the natural choice for scientific applications on the cloud. Fortunately, several systems (for example, [6, 7]) have employed this design principle. In the remainder of this section, we pick FusionFS and HDFS as two representative file systems to illustrate the importance of a distributed metadata service under intensive metadata accesses. Before discussing the experiment details, we provide a brief introduction of the metadata management of both systems.

HDFS, as a clone of the Google File System [2], has a logically[1] single metadata server (i.e., namenode). The replication of the namenode is for fault tolerance rather than balancing the I/O pressure. That is, all the metadata requests are directed to the single namenode—a simple, yet effective design decision for the cloud workloads. FusionFS is designed to support extremely high concurrency of metadata accesses. It achieves this goal by dispersing metadata to as many nodes as possible. This might be overkill for small- to medium-scale applications, but is essential for those metadata-intensive workloads that are common in scientific applications.

Fig. 1 shows the metadata performance of HDFS and FusionFS on Kodiak. The workload in this experiment is to let each node create empty files to the file system at different scales from 4 to 512 nodes. The reported numbers are aggregate metadata throughput.

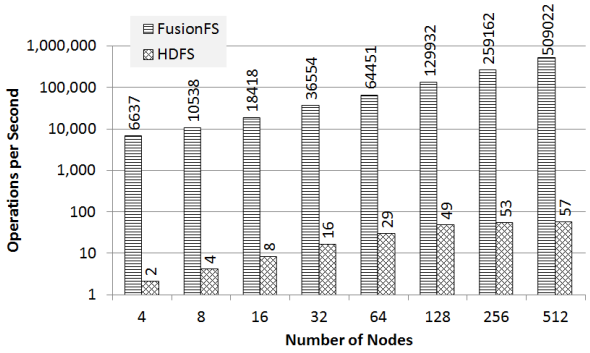

Figure 1: Metadata performance of FusionFS and HDFS are compared.

There are three important observations worth more discussion. First, the baseline (i.e., 4 nodes) shows that HDFS can only create 2 files per second, while FusionFS delivers more than 6K operations per second. This drastic gap is mainly due to that HDFS is a Java implementation with large overhead when communicating with the name server and FusionFS's metadata is highly optimized with C/C++. Second, HDFS's metadata performance starts to taper off beyond 128 nodes. This can be best explained by that the single name server gets saturated by the aforementioned workloads from 128 nodes. Third, the metadata throughput of FusionFS shows almost linear scalability from 4 nodes all the way to 512 nodes. This trend justifies the effectiveness of employing a distributed metadata management for intensive metadata accesses.

[1]because it gets replicated on multiple nodes, physically

3. OPTIMIZED DATA WRITE

Data write is one of the most common I/O workloads in scientific applications due to their de facto mechanism to achieve fault tolerance—checkpointing. Essentially, checkpointing asks the system to periodically persist its memory states to the disks, which involves a larger number of data writes. The persisted data only need to be loaded (i.e., read) after a failure occurs in a completely nondeterministic manner. As the system is becoming increasingly larger, the time interval between consecutive checkpoints is predicted to be dramatically smaller in future systems. [21] From storage's perspective, cloud platform will have to provide highly efficient data write throughput for scientific applications.

Unfortunately, HDFS could hardly provide optimized data write due to the metadata limitation discussed in Section 2. Fig. 2 shows the write throughput of FusionFS and HDFS on Kodiak. Similarly to the metadata trend, the write throughput of HDFS also suffers poor scalability beyond 128 nodes.

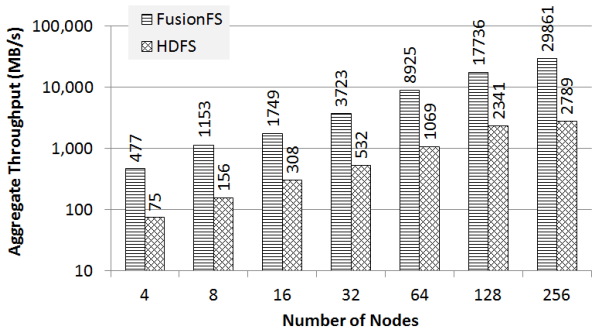

Figure 2: Write throughput of FusionFS and HDFS are compared.

Another option in cloud platforms is the remote shared storage. It usually provides a unified interface and scalable I/O performance for applications. One example is the S3 storage on Amazon EC2 cloud. S3 does not only provide a set of API but also leverages FUSE [5] to serve as a fully POSIX-compliant file system named S3FS. Therefore S3FS is becoming a popular replacement of the conventional remote shared file systems [22, 23] in HPC. Nevertheless, we will show that S3FS is hardly competitive to a node-local file system.

Fig. 3 shows the aggregate throughput of both FusionFS and S3FS on Amazon EC2 m3.medium instances. Obviously, S3FS is orders of magnitude slower than FusionFS. The reason is that every file operation on S3FS invokes a sequence of remote data transfer while FusionFS tries to deal with data on the local disk. FusionFS has shown highly strong scalability on HPC machines on up to 16K nodes resulting in an aggregate 2.5 TB/s throughput; we are now seeing a similar success on the cloud.

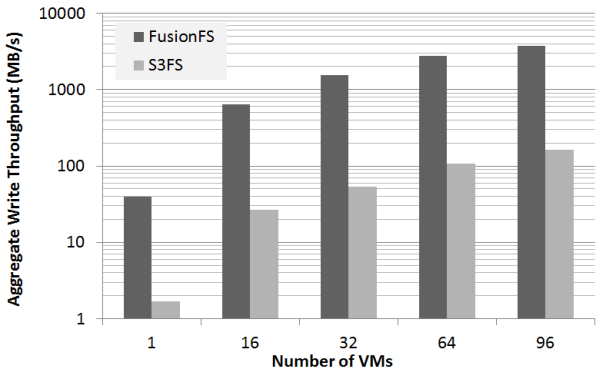

Figure 3: Write throughput of FusionFS and S3FS are compared.

According to FusionFS's performance, we envision the node-local storage is essential to achieve optimal write throughput. To further justify that, we conduct similar experiments with another node-local distributed file system Ceph [6] deployed on FermiCloud [18]. Fig. 4 shows that the write throughput of Ceph is almost linearly scalable and the efficiency is always higher than 90%. The baseline number is relatively low (6.66 MB/s) because we only allow five writers in the experiment.

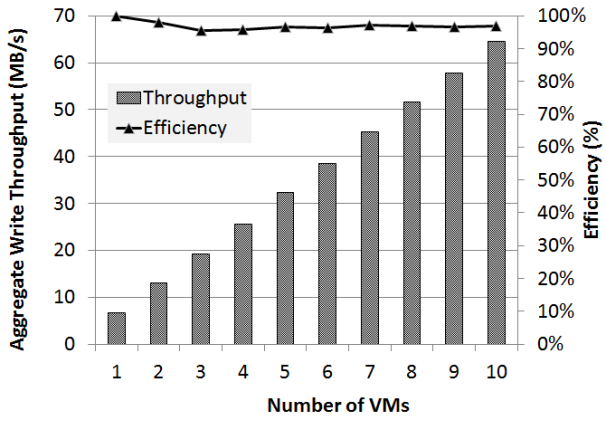

Figure 4: Scalable Write Throughput of Ceph.

The scalable throughput of Ceph is attributed by its distributed metadata and data movement. The metadata in Ceph is organized in such a way that high availability and scalability are guaranteed. An extra ceph metadata server can be standby, ready to take over the duties of any failed metadata server that was active, thus eliminating the single point of failure on metadata. Moreover, Ceph can be configured with multiple metadata servers that split the directory tree into subtrees (and shards of a single busy directory), effectively balancing the load amongst all the active servers. Ceph use Object Storage Device (OSD) Daemons to handle the read/write operations on the storage disks. Unlike traditional architectures, where clients talk to a centralized component (for example, a gateway, a broker), Ceph allows clients to interact with Ceph OSD Daemons directly [8]. This design prevents the single point of failure and improves the performance and scalability of Ceph's I/O throughput.

Nevertheless, a key difference exists between Ceph and FusionFS. In Ceph, a complicated algorithm (CRUSH) migrates data across multiple physical nodes to achieve load balance in nearly real-time; FusionFS always writes to the local storage and asynchronously calls a background process to balance the data. In other words, FusionFS trades real-time load balance for higher I/O performance, which is desirable in many scenarios.

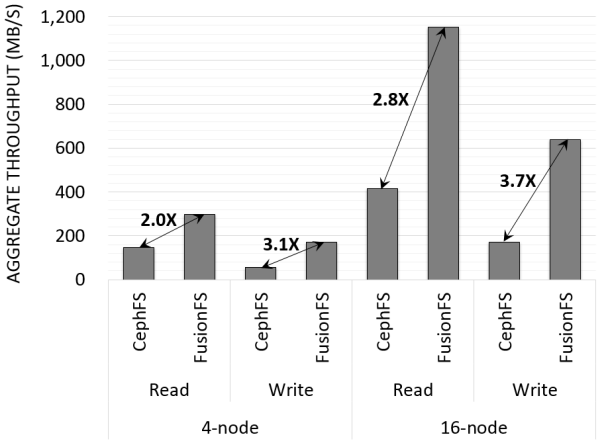

Figure 5: Ceph and FusionFS on Amazon EC2.

Fig. 5 shows the aggregate throughput of Ceph and FusionFS on Amazon EC2 cloud, both of which are deployed with FUSE and measured when writing/reading a distinct file of 4 GB on each node. The setup of both filesystems are the same: each node behaves as a data server, a metadata server, and a client. We observe both file systems have good scalability, but FusionFS is stronger: from 4 to 16 nodes the gap between Ceph and FusionFS is increased from $(2.0+3.1)/2 = 2.5\times$ to $(2.8+3.7)/2 = 3.3\times$. Also note that the throughput of FusionFS is reaching the hardware limit: on 4 nodes it delivers aggregate 170 MB/s indicating 42.5 MB/s per node, and the raw bandwidth is 44.9 MB/s. That is, FusionFS achieves $42.5/44.9 = 94.7\%$ efficiency.

4. LOCALIZED FILE READ

File read throughput is an important metric and is often underestimated since a lot of effort is put on data write as discussed in Section 3. When a VM is booted or restarted, the image needs to be loaded into the memory and this is becoming a challenging problem in many cloud platforms [24, 25]. Therefore a scalable read throughput is highly desirable for the cloud storage, which urges us to revisit the conventional architecture where files are typically read from remote shared file systems. In HPC this means the remote parallel file system such as GPFS and Lustre, and in cloud platforms such as Amazon EC2 it implies the remote S3 storage, or the S3FS file system.

Ideally, local file reads would be preferable since it avoids the costly remote data transfer. To demonstrate this, we compare FusionFS and S3FS on Amazon EC2 in Fig. 6. FusionFS, which leverage local disks for file reads, shows orders of magnitude higher throughput than S3FS.

On FermiCloud, Ceph, another local-read-enabled file systems also achieves a scalable read throughput and high efficiency as shown in Fig. 7 (the relatively low baseline num-

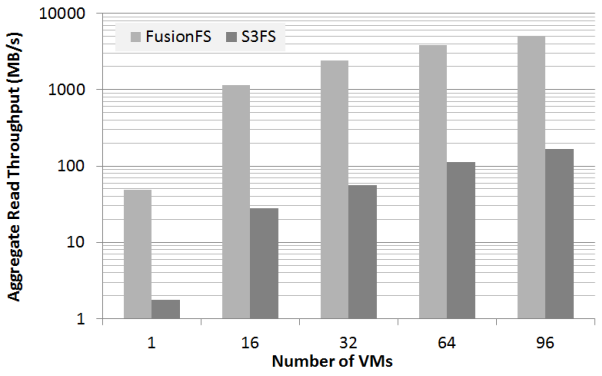

Figure 6: Read throughput of FusionFS and S3FS are compared.

ber is due to the same reason in write throughput; only five readers are enabled). We believe this is essential for read-intensive applications on the cloud, such as VM image loading. While strictly local reading might not always be an option in all cases because some applications poses unique workflows and are hard to determine before run time, more "static" workloads such as VM loading should be supported by local file read if at all possible.

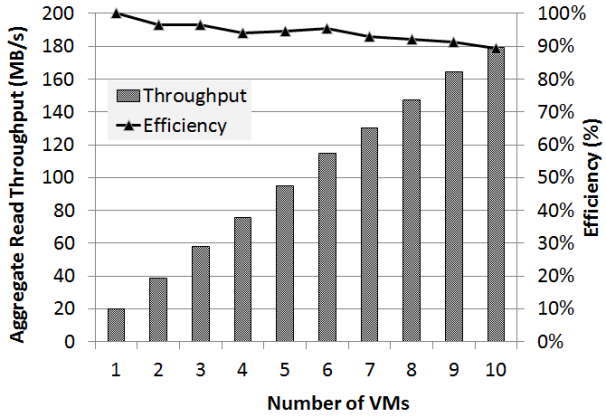

Figure 7: Scalable Read Throughput of Ceph.

5. RELATED WORK

Conventional storage in HPC systems for scientific applications are mainly remote to compute resources. Popular systems include GPFS [22], Lustre [23], PVFS [26]. All these systems are typically deployed on a distinct cluster from compute nodes. Cloud computing, on the other hand, is built on the commodity hardware where local storage is typically available for virtual computing machines. The de facto node-local file system (Google File System [2], HDFS [1]), however, can be hardly leveraged by scientific applications out of the box due to the concerns on small file accesses, POSIX interface, and so forth. Another category of storage in the cloud is similar to the conventional HPC solution—a remote shared storage such as Amazon S3. A POSIX-compliant file system built on S3 is also available named S3FS [4]. Unfortunately its throughput performance usually becomes a bottleneck of the applications and thus limits its use in practice.

6. CONCLUSION

This position paper envisions the characteristics of future cloud storage systems for scientific applications that used to be running on the HPC systems. Based on literature and our own FusionFS experience, we believe the key designs of future storage system comprise the fusion of compute and storage resources as well as completely distributed data manipulation (both metadata and files), in order to make cloud computing more adaptable to scientific applications, namely (1) distributed metadata accesses, (2) optimized data write, and (3) localized file read.

Acknowledgments. This work was supported in part by the National Science Foundation under awards OCI-1054974 (CAREER), by the US Department of Energy under contract number DE-AC02-07CH11359, and by KISTI under a joint Cooperative Research and Development Agreement CRADA-FRA 2014-0002/KISTI-C14014.

7. REFERENCES

[1] K. Shvachko, et al. The hadoop distributed file system. In *MSST*, 2010.
[2] S. Ghemawat, et al. The Google file system. In *SOSP*, 2003.
[3] P. Carns, et al. Small-file access in parallel file systems. In *IPDPS*, 2009.
[4] S3FS. https://code.google.com/p/s3fs/.
[5] FUSE. `http://fuse.sourceforge.net`.
[6] S. Weil, et al. Ceph: A scalable, high-performance distributed file system. In *OSDI*, 2006.
[7] D. Zhao, et al. FusionFS: Toward supporting data-intensive scientific applications on extreme-scale distributed systems. In *Big Data Conference*, 2014.
[8] S. Weil, et al. Crush: Controlled, scalable, decentralized placement of replicated data. In *SC*, 2006.
[9] D. Zhao, et al. Distributed file systems for exascale computing. In *SC*, 2012.
[10] D. Zhao, et al. Towards high-performance and cost-effective distributed storage systems with information dispersal algorithms. In *CLUSTER*, 2013.
[11] D. Zhao, et al. Distributed data provenance for large-scale data-intensive computing. In *CLUSTER*, 2013.
[12] D. Zhao, et al. Hycache+: Towards scalable high-performance caching middleware for parallel file systems. In *CCGrid*, 2014.
[13] D. Zhao, et al. HyCache: A user-level caching middleware for distributed file systems. In *IPDPSW*, 2013.
[14] D. Zhao, et al. Virtual chunks: On supporting random accesses to scientific data in compressible storage systems. In *Big Data Conference*, 2014.
[15] D. Zhao, et al. Improving the i/o throughput for data-intensive scientific applications with efficient compression mechanisms. In *SC*, 2013.
[16] Kodiak. https://www.nmc-probe.org/wiki/machines:kodiak.
[17] Amazon EC2. http://aws.amazon.com/ec2.
[18] FermiCloud. http://fclweb.fnal.gov/.
[19] B. Welch, et al. Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions. In *MSST*, 2013.
[20] D. Nagle, et al. The Panasas activescale storage cluster: Delivering scalable high bandwidth storage. In *SC*, 2004.
[21] D. Zhao, et al. Exploring reliability of exascale systems through simulations. In *HPC*, 2013.
[22] F. Schmuck, et al. GPFS: A shared-disk file system for large computing clusters. In *FAST*, 2002.
[23] P. Schwan. Lustre: Building a file system for 1,000-node clusters. In *Linux symposium*, 2003.
[24] H. Wu, et al. A reference model for virtual machine launching overhead. *IEEE Transactions on Cloud Computing*, 2014.
[25] H. Wu, et al. Modeling the virtual machine launching overhead under fermicloud. In *CCGrid*, 2014.
[26] P. Carns, et al. PVFS: A parallel file system for linux clusters. In *Annual Linux Showcase and Conference*, 2000.