几个常见分布式文件系统特征分析和性能对比

熊 文 喻之斌 须成忠

（ 中国科学院深圳先进技术研究院 深圳 518055 ）

摘 要 近年来随着云计算市场规模不断增长，作为云计算平台基础设施的云存储系统也随之显得越来越重要。数以万计的互联网应用已经运行于云计算环境，同时大量不同的应用也即将从传统运行环境转移到云计算平台。不同的互联网应用的存储需求可能不一样。例如：应用中涉及的单个文件大小，文件数量，IO访问模式，读写比率等，都对底层存储系统提出了不同的要求。这说明在云计算环境中，单个文件系统可能无法满足全部应用的存储需求， 本文尝试通过在单一云计算平台中部署多个不同分布式文件系统来优化存储系统的总体性能。

为了优化混合式文件系统的性能，首先需要分析不同文件系统的性能特征。本文通过量化方法分析了云计算环境下几个常用的分布式文件系统，这些文件系统分别是ceph，moosefs，glusterfs和hdfs。实验结果显示：即使针对同一文件的相同读写操作，不同分布式文件系统之间的性能也差异显著，当单个文件的大小小于256MB时，moosefs的平均写性能比其它几个文件系统高22.3%；当单个文件大小大于256KB时，glusterfs的平均读性能比其它几个文件系统高21.0%。这些结果为设计和实现一个基于以上几个分布式文件系统的混合式文件系统提供了基础。

关键词 分布式文件系统；性能测量；基准测试程序

A Characterization and Analysis of Distributed File Systems

XIONG Wen YU Zhi-bin XU Cheng-zhong

(*Center for Cloud Computing, Shenzhen Institutes of Advanced Technology, Shenzhen* 518055, *China*)

Abstract Recently, there has been an explosive growth in cloud computing, greatly increasing the importance of storage in such systems. A wide range of applications have been running in cloud and more and more variant applications are rushing into this platform. Different applications may have different requirements for storages such as file size, the number of files, and I/O performance. This indicates only a unified file system in cloud would keep the overall system performance suboptimal or even cannot satisfy the need of all applications in a cloud. However, it is unclear that whether it is beneficial to optimize the overall I/O performance by employing variant file systems in a single cloud computing platform.

In this paper, we address the above problem by characterizing several popular distributed files systems used in cloud computing. These file systems are ceph, moosefs, glusterfs and hdfs. Through the characterization, we find that the performance of the same operation such as read or write may be dramatically different for different file systems. When the file size is less than 256 MB, moosefs has the best writing performance. On average, its writing performance outperforms others by 22.3%. As for reading performance, glusterfs is the best when the file size is larger than 256KB. Its reading performance is 21.0% higher than other file systems. These findings lead us to design a hybrid file system for cloud computing platform, attempting significantly improve the overall performance.

Keywords distributed file system; performance measurement; benchmarks

作者简介：熊文，博士研究生，研究方向为大数据存储技术；喻之斌，副研究员，研究方向为计算机体系结构和性能评估，在体系结构和性能评估领域的顶级会议上发表了论文多篇，是多个顶级期刊如TPDS的审稿人，也是多个知名国际会议如HPCA的审稿人；须成忠，研究员，现任中国科学院深圳先进技术研究院研究员、首席科学家、云计算研究中心主任，美国韦恩州立大学电子与计算机工程系终身教授、云计算与互联网实验室主任，须成忠教授是IEEE和IEEE计算机协会资深会员，美国华裔教授学社理事，他也是中国自然科学基金会2010“海外学者合作研究基金”（“海外杰青”）获得者。2011年入选“广东省领军人才”计划和国家“千人计划”，主要研究方向为并行与分布式系统、互联网与云计算、高性能计算、移动嵌入式系统。

1 Introduction

In recent years, there is an explosive growth in cloud computing, greatly increasing the importance of storage in such systems. According to a new Forrester report called “Sizing the Cloud” [1], which is published by an independent research institute -Forrester Research- expects the global cloud computing market to reach $241 billion in 2020 compared to $40.7 in 2010. At the same time, Cloud Storage has also been increased in popularity recently. As one of the three types of basic resources in Cloud Computing platform [8], storage does not only meet the storage requirements of various applications in cloud platform, but also provides the capability for other basic infrastructures to store and to retrieve data. Furthermore, there are many popular applications, such as dropbox, icloud and ubuntu one, directly constructed on cloud storage systems [2].

On the other hand, industry has already shift gears to run applications on cloud. Taking the top two cloud computing platforms as example, there are a few hundreds of popular applications already deployed in the Amazon EC2 [3]. Meanwhile, there are dozens of typical applications running in the windows Azure platform [4].

In addition to the larger number of applications running on cloud, the types of applications are also dramatically different. For example, the various applications in Amazon EC2 have been classified into nine categories, including application hosting, backup and storage, content and delivery, e-commence, high performance computing, media hosting, on demand workforce, search engines and web hosting[8].

Different applications may have different requirements for storage. For example, CampusLIVE uses CloudBerry Lab solutions on Amazon Simple Storage Service and Amazon CloudFront to serve millions of static images[3]. Soundtrckr is the first geosocial Internet radio, with 8 million songs available to users to create radio stations and easily share them on social media applications[3]. And Marcellus provides video platform, which delivers high quality video access on its clients’ Websites. The file size of those applications distributes between dozens of KBs of image, a few MBs of song and a few GBs of high definition video [3].

Only one file system will keep the overall performance of cloud system suboptimal or even cannot satisfy the need of all applications in a cloud. Naturally, making multiple file systems co-exist in the same cloud may be feasible. However, it is unclear that whether it is beneficial to optimize the overall I/O performance by employing variant file systems in a single cloud computing platform.

To address this problem, in this paper, we characterize several popular distributed files systems used in cloud computing. These file systems are ceph[8], moosefs[9], glusterfs[10] and hdfs[11]. Through the characterization, we find that the performance of the same operation such as read or write may be dramatically different for different file systems. When the file size is less than 256 MB, moosefs has the best writing performance. On average, its writing performance outperforms others by 22.3%. As for reading performance, glusterfs is the best when the file size is larger than 256KB. Its reading performance is 21.0% higher than other file systems.

In particular, the main contributions of this paper are as follows:

- We characterize distributed file systems from several different aspects, including architecture of distributed file system, algorithm of metadata indexing and data locating and file system interface. We have run a series of experiments to evaluate the performance of the four different distributed file systems.
- We propose an approach to optimize overall I/O performance for applications involved files with different size. The key idea is to store the file with a fixed size to the best suitable distributed file system.

The rest of the paper is organized as follows. Section 2 describes the four distributed file systems. Section 3 depicts the experimental methodology. Section 4 shows the results and analysis and section 5 concludes the paper.

2 Distributed file systems

In this section, we describe four different distributed file systems respectively. These distributed file systems including: glusterfs, hdfs, ceph and moosefs.

2.1 Ceph

Ceph is a distributed object store and file system designed to provide excellent performance、reliability and scalability. Ceph provides a traditional file system interface with POSIX semantics and provides object storage and block device interfaces [8]. Ceph has four components which are monitor, object storage daemon, client, and metadata servers.

Monitor provides authentication for members in the storage cluster, and monitors the state of all members in the storage cluster.

Object storage daemon is a smart storage node interacting with other Object storage daemons, and provides the capability of self-managing.

The client accesses object storage system or distributed file system by librados or librbd and get data by interacting with the Object storage daemons directly.

The metadata server cluster provides a service that maps the directories and file names of the file system to objects stored within RADOS clusters.

2.2 Hdfs

Hdfs is the default file system in hadoop ecosystem. It provides native support for mapreduce computing framework. It also provides proprietary APIs and POXIS like interface by fuse-dfs component [11].

Hdfs adopt master-slave architecture. An hdfs cluster consists of a single namenode and a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of datanodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. hdfs exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of data-nodes.

2.3 Glusterfs

Glusterfs provides an interface with POXIS semantics and NFS/CIFS interface. It is a scale-out NAS file system and has three different components including client, storage node and NFS/Samba storage gateway. Storage nodes are typically deployed as storage bricks[10].Glusterfs provides customers the capability to build RAID-like storage system.

Glusterfs is based on peer to peer architecture, without metadata server, clients take more responsibilities including volume management, I/O scheduling, file locating and data caching.

2.4 Moosefs

Moosefs provides interface with POXIS semantics and it is available on every Operating System with a working FUSE implementation [9].

Moosefs consists of four components including chunkserver, metalogger server, client and metadata server.

Chunkservers storing files data and synchronizing it among themselves.

Metadata server is a single machine managing the whole file system and storing metadata for every file.

Metalogger servers are responsible for storing metadata changelogs and downloading main metadata file periodically; so as to promote these servers to the role of the metadata server when the primary master stops working.

Client use as daemon process named mfsmount to communicate with the metadata server and chunkservers.

2.5 Design Decisions of Distributed File Systems

The design decisions including: architecture of distributed file system, the algorithm of metadata indexing and data locating, the file system interfaces, data replicate mechanism, data migration mechanism, disaster recovery mechanism and the snapshot technology, the detail information as table 1 described.

MDS is metadata server in moosefs and ceph, while MDS is namenode in hdfs.

All of the four different distributed file system provides the capability to storing data between different fault domains.

3 Experimental Methodology

In this section, we evaluate the four different distributed file systems using a file server workload (based on filebench).

3.1 Experimental Platform

The configurations of all the machines are configured with, Intel(R) Xeon(R) CPU E5620 @2.40GHz 2CPU 8 cores processor, 16GB of memory, three 2000G 7200 rpm disks, and a 1000Mbs full-duplex Ethernet connection to switch, and all the member of each cluster are connected

Table 1. Design decitions of four distributed file system

Distributed file system	Moosefs	Ceph	Glusterfs	Hdfs
Metadata server	one MDS, single poin t of failure	MDS cluster	No MDS	One MDS, single point of failure
Metadata indexing	metadata is stored In the memory of metadataserver	metadata is stored in the memory of metadata server cluster	Metadata is cached inthe memory of client	metadata is stored in the memory of metadata server
Data locating	lookup the directory tree in MDS	consistent hash algorithm	consistent hash algorithm	lookup the directory tree in MDS
interfaces	POSIX	POSIX, REST API, block device interface, librbd and librados	POSIX，NFS/CIFS	POSIX like, libhdfs
data distribution	File data is divided into chunks with a fixed size, each chunk be stored in chunkservers	File is divided into object, each object be storedin object storage devices	it provides customer the capability to buildR AID-like storage system	File data is divided into data blocks with a fixed size, each block be stored in data nodes
availability	Multiple replicas in different fault domains	Multiple replicas in different fault domains	Data mirroring in different fault domains	Multiple replicas in different fault domains
disaster recover	MDS failture, mds-recovering manually, chunkserver failture, data migrating automaticly	MDS and object storage device failture, data migrating automaticly	Storagenode failture, recovering manually	MDS failture, recovering manually, datanode failture, data migrating automaticly
scalability	MDS is the perform ance Bottleneck	Linear nearly scale-out	Linear nearly scale-out	MDS is the performance Bottleneck

to one switch.

The system software information about the measuring environment is as follows: OS, SUSE Enterprise Edition 11 sp1 x86_64; OS kernel, 2.6.32.12-0.7; glusterfs, version3.2.1; moosefs, version 1.6.25; ceph, v0.34; hdfs, version 1.0.3; filebench version 1.4.9.1.

The ZCAV effect was taken into consideration, each physical disk was divided into two partitions with fixed size, each partition was formatted to ext3 as the default local file system, and just the second partition be used in the measuring procedure[12].

We run a series of experiments to evaluate the performance of different distributed file systems respectively, information of the four storage cluster as table 2 described.

In hdfs cluster, both the MDS and the second namenode are in the same physical machine. Other node is the metalogger server in moosefs cluster, is the monitor in ceph cluster and is the second namenode in hdfs cluster.

Table 2. Member of storage cluster

member\DFS	moosefs	ceph	glusterfs	hdfs
storage node	3	3	3	3
MDS	1	1	0	1
other node	1	1	0	1
client	1	1	1	1

3.2 Test Methodology

Filebench takes a file size distribution, a read/write ratio and the number of subdirectories. For each workload, filebench creates the specified number of subdirectories and creates predefined file size distribution within those subdirectories. After the configuration was build, transactions including a series of read or write operations, are performed against it. We record the number of files written/read per second, the total size of the file set and the time to write or read the entire file set.

In the reading throughput test, two threads read simultaneously from the distributed file system, each

thread reads a sequentially selected file from the predefined file hierarchy, the size of the I/O operations be specified with 1MB.

In the writing throughput test, two threads write simultaneously to the distributed file system, each thread writes to a file according to the predefined file hierarchy, the size of the I/O operations be specified with 1MB.

In between each test, we unmounted the tested distributed file system, and remounted it, this ensured that we started each test on a cold cache for that distributed file system. For each test, we took 10 measurements and averaged them.

4 Results and Analysis

4.1 Reading Performance

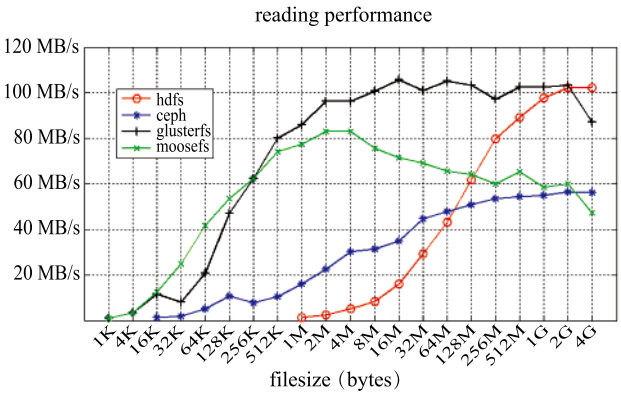

Figure 1. Reading performance of hdfs, ceph, glusterfs and moosefs with difference file sizes

When the file size less then 256KB, the distributed file system with best reading performance is moosefs, meanwhile, the file size is larger than 256KB, the one with best reading performance is glusterfs.

The average speed calculated by the total size of the file set and the total time the benchmark consuming.

Table 3. Average reading speed

file system	Ceph	Glusterfs	Moosefs	Hdfs
Average speed(MB/s)	55.5	94.18	60.53	92.54

4.2 Writing Performance

When the file size less then 256MB, the distributed file system with best writing performance is moosefs, meanwhile, the file size is larger than 256MB, the one with best writing performance is hdfs.

The average speed calculated by the total size of the file set and the total time the benchmark consuming.

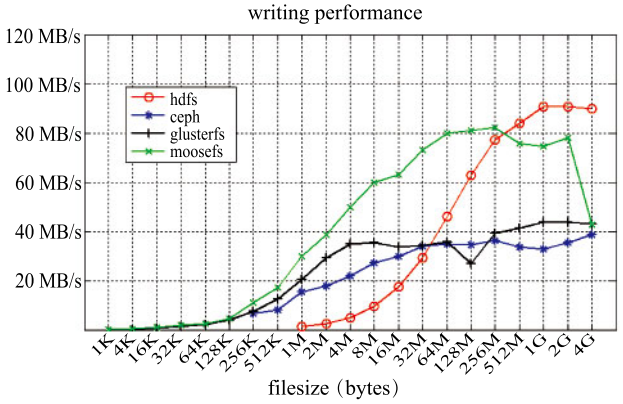

Figure 2. Writing performance of hdfs, ceph, glusterfs, and moosefs with different file sizes

Table 4. Average writing speed

file system	Ceph	Glusterfs	Moosefs	Hdfs
Average speed(MB/s)	36.43	42.49	76.77	84.14

4.3 Different Replicas Level in Hdfs

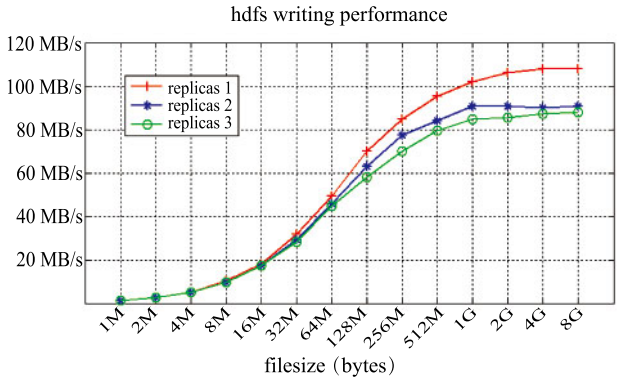

Figure 3. Impact of various replicas levels on writing performance of hdfs

On average, the configuration with one replica is 25% faster than the one with three replicas on writing performance.

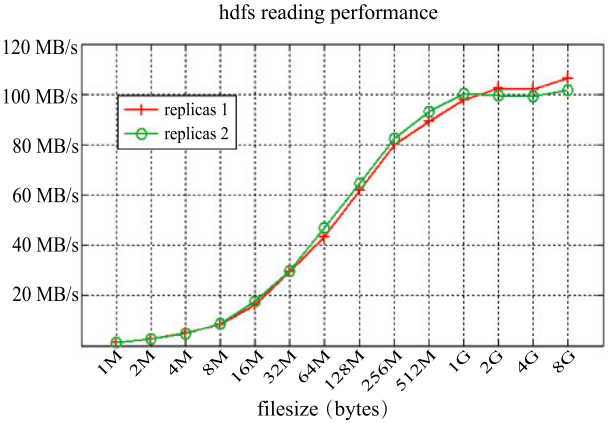

Figure 4. Impact of various replicas levels on reading performance of hdfs

There is no obvious performance variance between the two different replicas level on reading performance.

5 Conclusions

Given a series of different file size classes, the performance of operations such as read or write are dramatically different on the four distributed file systems. For a specific application involved a huge number of files with different size, this approach, by storing the file with size less than 256MB to moosefs and storing the file with size larger than 256MB to HDFS, can greatly enhance the overall write performance. by storing the file with size less than 256KB to moosefs and storing the file with size larger than 256KB to glusterfs, can greatly enhance the overall read performance.

6 Future Work

Based on our characterization, we can imagine that a hybrid file system can provide customers the capability of employing the best suitable file system to store files with different size and consequently greatly enhance the overall performance of a distributed storage system in cloud platform. We, therefore, will implement a cloud file system with different file systems but with a unified interface.

参 考 文 献

[1] Forrester [EB/OL]. http://www.forrester.com/

[2] Drago I, Mellia M, Munafò M M, et al. Inside dropbox understanding personal cloud storage services [C] // Internet Measurement Conference, Boston, USA, 2012.

[3] http://aws.amazon.com/solutions/case-studies/.

[4] http://www.windowsazure.com/.

[5] Ghemawat S, Gobioff H, Leung S T. The Google file system [C] // In Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, 2003.

[6] Lenk A, Klems M, Nimis J, et al. IEEE Cloud [C] // Proceedings of the International Conference on Software Engineering Challenges of Cloud Computing, 2009: 23-31.

[7] Liu X H, Han J Z, Zhong Y Q, et al. Implementing WebGIS on Hadoop: A Case Study of Improving Small File I/O Performance on HDFS [C] // Cluster Computing and Workshops, 2009.

[8] Weil S, Brandt S A, Miller E L, et al. Ceph: a scalable, high-performance distributed file system [C] // Proceedings of the 7th Conference on Operating Systems Design and Implementation, 2006.

[9] Moosefs homepage [EB/OL].http://www.moosefs.org/

[10] GlusterFS homepage [EB/OL]. http://www.gluster.org

[11] Hadoop homepage [EB/OL]. http://hadoop.apache.org/

[12] Traeger A, Zadok E, Joukov N, et al. A nine year study of file system and storage benchmarking [J]. ACM Transactions. 2008, 4(2): 56.