Chapter 68
Cloud Storage Architecture with Meta-Data Service Layer in Cloud Computing

Kai Fan, Libin Zhao, Hui Li, and Yintang Yang

Abstract With the rise of cloud computing, cloud storage has become a challenging issue. It is a huge challenge to design a distributed file architecture to meet the requirements of cloud storage. In this paper, in order to improve the system reliability and performance, we propose a cloud storage architecture with a meta-data service layer. The proposed architecture is a distributed file storage system based on the master-slave architecture, which uses multiple proxy servers of the meta-data server to establish a peer-to-peer meta-data service layer. Each meta-data server and the proxy server, could be an access to service for clients rather than only one fixed access as usual, which can improve the parallel processing performance of a meta-data service layer greatly. Some P2P techniques are used between proxy servers to solve the disadvantages of the master-slave architecture efficiently. Analysis and evaluation are performed to demonstrate that the proposed architecture improves the system reliability and performance greatly.

Keywords Cloud storage • Architecture • Meta-data service layer • Reliability • Performance

68.1 Introduction

The master-slave and the peer-to-peer (P2P) are the two most basic architectures. The former has the advantages of simpleness, easy operation and maintenance. On the other hand, the latter has the advantages of reliability and stability. GFS [1], the representative of the master-slave storage architecture, has the bottlenecks of reliability and performance [2, 3], which leads to the inefficient concurrent access and the single point failure. In order to improve the parallel processing performance, the parallel optimization BlobSeer [3] storage layer is used to substitute the

K. Fan (✉) • L. Zhao • H. Li
State Key Laboratory of Integrated Service Networks, Xidian University, Xian, China
e-mail: kfan@mail.xidian.edu.cn

Y. Yang
Key Lab. of the Minist. of Educ. for Wide Band-Gap Semiconductor Materials and Devices, Xidian University, Xian, China

J. Mu et al. (eds.), *The Proceedings of the Third International Conference on Communications, Signal Processing, and Systems*, Lecture Notes in Electrical Engineering 322, DOI 10.1007/978-3-319-08991-1_68

distributed file storage system HDFS [4] in Hadoop cloud computing system. Although the BlobSeer improves the concurrency, it cannot address the single point reliability efficiently. The P2P based data sharing network [5, 6] is proposed to eliminate the single point failure and ensure the expansion and reliability. In addition, Amazon presents the decentralized P2P cloud storage architecture Dynamo, which is the top-down P2P structure. Although Dynamo ensures the balance of data distribution and node load, the update and search of large-scale P2P network will be delayed greatly. Furthermore, Dynamo improves the reliability, but it delays some data processing in consistency.

Although the consistency, reliability and availability cannot be achieved simultaneously, they can be balanced to some extent. We propose a cloud storage architecture with a meta-data service layer. The proposed architecture is a distributed file storage system based on the master-slave architecture, which uses multiple proxy servers of the meta-data server to establish a P2P meta-data service layer. The P2P server structure is only used in the meta-data service layer. All the meta-data is stored with one copy in each different meta-data servers, which can ensure the fault tolerance, and reduce the complexity of the file lock service in the case of one file being written by many clients. In the proposed architecture, each meta-data server or each proxy server could be an access to service for clients rather than only one fixed access as usual, which can improve the parallel processing performance of the meta-data service layer greatly. In addition, proxy servers share part of services of the meta-data server in GFS and HDFS to improve the performance. In addition, some P2P techniques, such as configuration, service scheduling, and searching, are used between the proxy servers to solve the disadvantages of the master-slave architecture efficiently. That can improve the reliability and the access concurrency, in which the meta-data can be used as the system cache in each proxy server. Analysis and evaluation are performed to demonstrate that the proposed architecture improves the system reliability and performance greatly.

The remainder of this paper is organized as follows: in Sect. 68.2, the cloud storage architecture with a meta-data service layer is proposed; in Sect. 68.3, the system service interaction protocol of the proposed architecture is proposed; and the analysis and evaluation of the proposed architecture is performed in Sect. 68.4; finally, concluding remarks are provided in the last section.

68.2 Cloud Storage Architecture with a Meta-Data Service Layer

The cloud storage architecture with a meta-data service layer is shown in Fig. 68.1. The block-data service layer, the meta-data service layer and the client application layer are defined from the inside out respectively.

The meta-data service layer is the key of the architecture. Compared with GFS system, the meta-data service layer is a P2P distributed service network using multiple proxy servers in the proposed architecture, which is like the meta-data server Master in GFS.

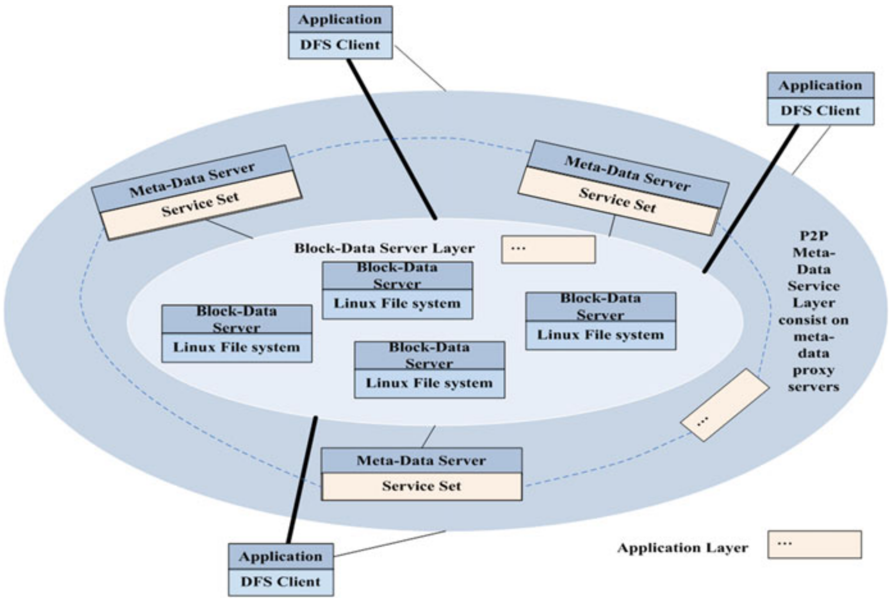

Fig. 68.1 The cloud storage architecture with a meta-data service layer

The definition of the function of the meta-data service layer is as follows.

1. Request listening: that is to listen on the request events of the client and the block-data server. The client events are the name space to be created and deleted, files to be created, wrote, read, deleted and renamed, list of files information to be accessed, resource lock to be obtained and released, and so on. The events of the block-data server are the heartbeat information, the file block information, the error information, and so on.
2. Request processing: that is to process the request listening events and return the result.
3. Meta-data management: the meta-data is mainly the name space, the mapping from file to the file block and the mapping from the file block to the block-data server.
4. Name space management: the name space is managed by using the directory tree structure.
5. File management: that is some basic operations to the file. For example, creating, writing, deleting and renaming.
6. File block management: that is to create or copy new files, delete the invalid file blocks and recover the orphaned file blocks.
7. Load balance of the block-data server: due to the uneven distribution of file blocks in different block-data servers, which caused by a large number of file blocks writing and deleting the file block load balance, should be performed.

8. Lease (session) management: that is to manage the lease of client. That includes the obtaining and releasing lease, and recovering lease, in which the lease is expired.
9. Heartbeat detection: the block-data server will report its load condition to the meta-data server though sending heartbeat information regularly.

The function of the block-data service layer is similar as the basic master-slave distributed file system. The definition of the function is as follows.

1. Data block information management: there may be tens of thousands of data blocks in the block-data servers, and the block-data server may operate any data block in anytime.
2. Data block writing and reading: client could write data blocks to or read data blocks from the block-data servers frequently. When the data block is being written, it also should be backed up.
3. Data block transmission: the data block transmission between block-data servers is frequent. When the writing operation and backup operation are performed, the data block transmission will be established between block-data servers.
4. Sending heartbeat information to the meta-data server: the meta-data server judges whether the block-data server is working normally based on the heartbeat information.
5. Processing the command information of the meta-data server: when the system is running, the meta-data server will tell the block-data server to back up, delete and move file blocks.
6. Reporting the file block information to the meta-data server: because file blocks will change in the block-data server, the block-data server should report the file block information to the meta-data server regularly. Then the newest block information will be in the meta-data server.
7. Processing client requests: the system interaction is the interaction between client and the block-data servers, such as the writing and reading of data blocks. Therefore, the interactive interface should be established between client and the block-data servers.

The client application layer is the service access layer, in which the compliable interface is provided as GFS. The storage interface is provided to client using the programming language in the proposed architecture, which can provide the function access using the command-line mode, and reduce the difficulty of the operation and maintenance. The definition of the function is as follows.

1. Directory management: that is to create, rename and delete directories.
2. File management: that is some basic operations to the file. For example, uploading, downloading, deleting and renaming.
3. Data flow operation: the output flow to file system will be created, when a client wants to upload a file to the system. Otherwise, the input flow reading from the system will be created when a client wants to download a file in the system.
4. The operation of resource lock: that is to obtain the resource lock and release it.

68.3 System Service Interaction Protocol

We can summarize the interaction in distributed file storage system as writing and reading operation. The data updating can be regarded as a special writing process, which is the same as the ordinary writing operation process. The processes of writing and reading are shown in Fig. 68.2.

The communication in the architecture consists of the control flow and the data flow, which are shown in the thin lines and thick lines. We define the main process in different layers.

68.3.1 The Writing Operation Step

W1: request writing, in addition, sending file information.
W2: returning the information of main storage block, including the mapping of file name, the size of the block, and so on.
W3: client writes files into the block-data server based on the result of W2.
W4: the block-data server backups files based on the result of W2.
W5: the backup node returns the backup result to main storage server.
W6: the result of block storage is sent to the meta-data server to generate the meta-data.
W7: confirming the meta-data.
W8: sending the result of client. If it fails, returning events, which will be completed by using control command and state recovering.

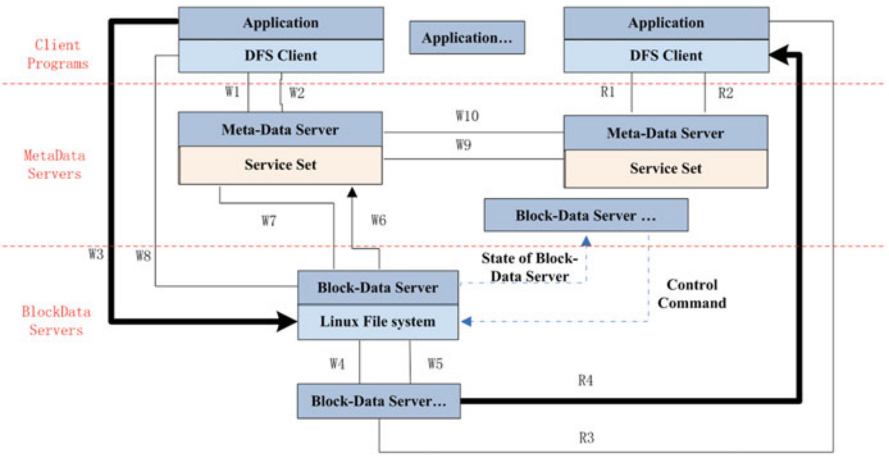

Fig. 68.2 The system service interaction process

W9: updating the local meta-data and the corresponding vector clock, and this operation information is transmitted to other main servers, meta-data proxy servers, by IP multicasting. Other servers choose to update their meta-data set or not based on the vector clock.

W10: checking the results returned from other main servers. The fault-tolerant processing is performed if there is a return warning with failure.

68.3.2 The Reading Operation Step

R1: request reading data.

R2: the main server returns the information of file blocks.

R3: client sends block information to the block-data server.

R4: the block-data server sends data blocks.

The copy process of the meta-data in servers is the key issue of the system performance improvement, such as W9 and W10 in Fig. 68.2.

68.4 Analysis and Evaluation

The architecture evaluation mechanism [3] we used is to determine the properties of the architecture by modeling or simulating one or many aspects in the system. We use theoretical modeling to qualitative analyze the proposed architecture. In addition, we code and implement the meta-data service layer model of meta-data proxy servers.

The analysis of reliability in Sect. 68.4 is valid based on following assumptions:

- life of each node follows an *exp* distribution.
- the life of all nodes is I.D.D.

68.4.1 The Analysis of System Scale

There are 40 servers in each cabinet in the condition of HDFS typical configuration. Every cluster can support about 1,000 nodes mostly in GFS. In the proposed architecture, the larger node scale can be supported. That is only the advantage of the proxy server model.

The proxy server in the meta-data service layer is a group of the P2P meta-data server. The scale should not be too large. One reason is to reduce the complexity of consistency maintenance. The other reason is the P2P structure is used in this layer, in which when P2P network scale is too large, network changing and data updating would cause the problem of jitter and delay, and the higher searching complexity.

So the meta-data service layer based on small scale not only can reduce the design complexity of the large-scale P2P network, but also can support much larger block server node scale, except the simplified consistency. We can estimate as follow.

The number of the meta-data server node, the proxy server node, in the meta-data service layer is at least 2 and at most 256 in the proposed architecture. That is the number of preliminary estimating. The node scale of the block-data server can be expanded of at least 200 times in GFS and HDFS theoretically. In addition, because every proxy server is an access for client, the concurrent processing capability of the meta-data service can be enhanced at least 200 times. When the number of the proxy server is less than 256, this is already a considerable enhancement. In addition, there is no the single point failure. The performance enhancement will be quantitatively analyzed in the availability analysis.

68.4.2 The Analysis of Service Performance

There are higher concurrent responses in the proposed architecture, because the proxy server shares the work of the Master node in GFS. The enhancement of system performance will be analyzed in the two aspects of writing and reading, and the simulation is also performed.

The average writing time of concurrent file request with different number of proxy servers is shown in Fig. 68.3. We have performed eight concurrent processes

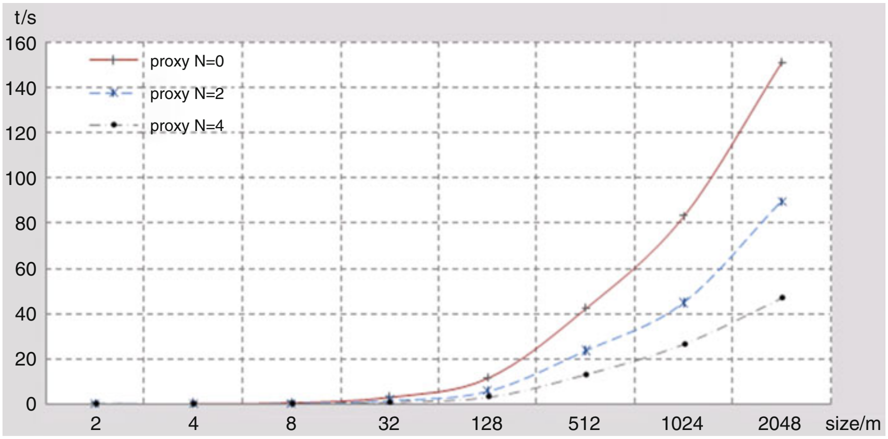

Fig. 68.3 The average writing time of concurrent file request with different number of the proxy servers

with 4,000 connections in each process. The size of files in each concurrent request is from 2 to 2,048 MByte.

From Fig. 68.3, the larger of the number of the proxy server and the larger of the file writing concurrent request, the less time cost in writing and the higher performance of the proposed architecture.

68.4.3 *The Analysis of System Reliability*

The reliability of the GFS and the proposed architecture with different server nodes is shown in Fig. 68.4.

From Fig. 68.4, the larger the system scale is, the higher reliability the proposed architecture achieves. The solution of the single point failure in GFS is the multiple nodes backed up. When the number of nodes is much larger, the reliability will not be improved obviously with the change of the number of nodes. Otherwise, the proposed architecture uses multiple proxy servers in service at the same time. Although the service performance will be decreased when many nodes cannot work, the availability will not be affected. Therefore, the proposed architecture has higher reliability with more nodes.

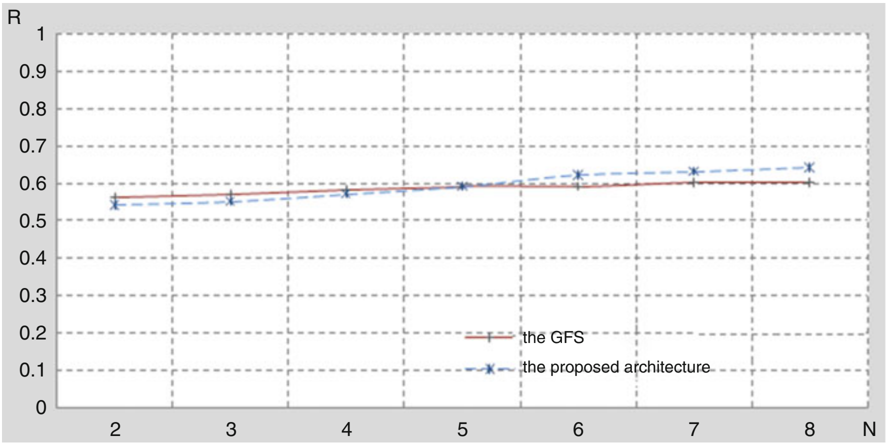

Fig. 68.4 The reliability of the GFS and the proposed architecture

Conclusions
In this paper, a cloud storage architecture with a meta-data service layer has been proposed. Compared with other existing cloud storage architectures, the proposed architecture achieves higher reliability and performance. For our future work, we will further explore other challenging issues, such as, file blocking storage method in cloud computing environment.

Acknowledgements This work has been financially supported by the National Natural Science Foundation of China (No. 61303216 and No. 61373172), the China Postdoctoral Science Foundation funded project (No.2013M542328), the Xidian-Ningbo Information Technology Institute Seed Foundation funded project, National 111 Program of China B08038, and the Xian Science and Technology Plan funded project (CXY1352WL30).

References

1. Zhan Y, Sun Y (2009) Cloud storage management technology. In: ICS, pp 301–311
2. Ye W (2009) SaaS architecture design. Publishing house of electronics industry, pp 56–78
3. Nicolae B, Moise D, Antoniu G (2010) BlobSeer: Bringing high throughput under heavy concurrency to Hadoop map-reduce applications. In: Parallel and Distributed Processing, pp 1–11
4. Shvachko K, Kuang H, Radia S, Chansler R (2010) The Hadoop distributed file system. In: Mass Storage Systems and Technologies, pp 1–10
5. Xu K, Song M, Zhang X, Song J (2009) A cloud computing platform based on P2P. In: IT in Medicine and Education, pp 427–432
6. Rivero M, Rubino G (2010) Priority-based scheme for file distribution in peer-to-peer networks. In: Communications, pp 1–6