An Efficient Big Data Storage Service Architecture

Yan Wang[1,3*], Xiang Li[1], Peixiang Bai[1], Haibo Wang[2,3], Jianmin Dong[2,3]
[1]College of Computer Science, Inner Mongolia University,Hohhot, 010021, China
[2]Inner Mongolia Discipline Inspection Information Center, Hohhot, 010021, China
[3]Inner Mongolia Big Data Laboratory for Discipline Inspection and Supervision, Hohhot, 010021, China

2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 978-1-6654-0527-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSCWD54268.2022.9776303

Abstract—**Due to the limitations of the distributed file storage system, when data storage is surging, the depth and width of file directory will continue to increase, which results in reduced I/O efficiency of metadata management. Rapidly growing data can also cause system congestion or data loss. In today's environment where information and data are linked and collaborative in all walks of life, this phenomenon is absolutely unacceptable. Object storage is an efficient way to store massive data through separating metadata. However, the original data storage system cluster has been configured with non-object-oriented storage data types. If the underlying storage system is replaced by object storage, it will not be compatible with the upper file parallel processing model. To solve this problem, this paper proposes a big data storage service architecture based on object storage. In the architecture, a transparent access method from file system to object storage cluster is designed to support efficient access to object storage system in the form of file. For improving the reliability of data storage service, a high fault tolerance mechanism based on multi-gateway is presented. Experimental results show that the architecture can effectively solve the problem of data access compatibility from file system to object storage system and significantly improve storage efficiency.**

Keywords—Distributed file system, Object storage, Metadata management; Transparent access; Multi-gateway fault tolerance

I. INTRODUCTION

With the advent of the era of big data, the traditional data warehouse has been unable to meet the growing demand for data storage. The ever-increasing storage requirements will inevitably lead to performance bottlenecks and scalability problems in storage systems. Studying big data storage methods and building an efficient big data storage model are the premises for analyzing and applying big data. Cloud computing becomes an ideal choice for big data management because of its scalability and flexibility. The cloud provides not only a suitable framework for the cluster of big data, but also an efficient distributed data storage model for the placement of big data. It processes data aggregation and management through some appropriate web services (such as RESTful API), that's, storage-as-a-service (STaaS) [1].

Cloud storage is a storage system for massive data based on clusters, which can efficiently store, access and process data and provide data migration, backup, disaster recovery and other services [2]. Over the past few years, the demands for cloud storage have grown exponentially. Since the distributed file system is famous for its high capacity, high stability and high scalability, cluster servers generally adopt file system. Currently, scalable parallel file systems are often used with large storage clusters. Especially, Hadoop can be deployed on a cheap cluster and has become the mainstream platform for big data storage by virtue of its advantages of open source, strong scalability and good fault tolerance. However, the data collected by various digital technologies include not only two-dimensional data, but also video, photos, blogs, emails and other forms of data. For example, tens of billions of small files are created by web applications every day, including photos, videos, music files; a billion pieces of content are added to the social network every day; hundreds of billions of e-mails are sent by people every day. Most of the contents are unstructured data. The storage of big data must efficiently store unstructured data. In general, a single file system can support at most millions of files under optimal performance. Therefore, the file storage is not suitable for managing such huge unstructured big data. When dealing with high concurrent data access, the shortcomings of the file system will show up, mainly including the following two aspects:

1)With the increase of the depth and bandwidth of directory tree structure, the efficiency of data management and the speed of I/O operation will drop significantly.

2)When tightly coupled with the kernel, storing or accessing data is inefficient due to the limitations of data block size, data consistency, caching strategy and other aspects.

The emergence of object storage decouples the metadata through adopting completely random distributed mechanism to separately store metadata and object data. Its flat data structure allows it to manage tens of billions of storage objects, supporting objects of any size ranging from bytes to trillions of bytes. In this way, the object storage capacity can be scaled from terabytes to exabytes, eliminating the scalability bottleneck caused by the complex iNode mechanism of the file system. Therefore, object storage system becomes an effective storage solution. Distributed object storage architecture has become the practical standard of high-performance storage in big data, cloud computing and high-performance computing [4].

However, due to the development limitations of core technology, intermediate platform and business plane, many existing cloud storage systems generally adopt file-oriented Hadoop architectures, using NAS devices to store all kinds of unstructured data in the form of files. In the face of the growing demands for unstructured data storage, NAS storage has faced bottlenecks in access performance, storage capacity, and data manageability. In order to effectively solve these problem while minimizing the cost of retrofitting the storage architecture, our paper proposes a mechanism for merging files with object data. This mechanism takes the object storage as the back-end storage of parallel analysis framework, designs a transparent access method of object-oriented storage, which solves the compatibility with the original file access system. At

* Corresponding author

the same time, it can realize the high availability of storage service for big data by multi-gateway fault tolerant mechanism.

In the following, Section 2 introduces related work. Section 3 focuses on the architecture and its key technologies. Section 4 describes the test experiments for the architecture, and analyzes the experimental results. In the end, we summarize the main contributions of our paper.

II. RELATED WORKS

Cloud storage is a cloud computing system with data storage and management as its core. The system integrates a large number of different types of storage devices in the network to provide the functions of data storage and access. High I/O performance, high reliability and low storage cost are all desirable features of cloud storage systems [5]. To maximize these capabilities, researchers have done a lot of work in recent years. In order to improve the efficiency of cloud storage and reduce the cost of access, a new storage solution ASSER was proposed to provide stable and efficient I/O performance at a lower storage cost in [6], which was an ASS mapping chain composed of coding and application. He et al. constructed a balanced data structure SGMHT (extension of Merkle Hash Tree based on scapegoat Tree) and implemented a new scheme to support simultaneous update of multiple data blocks [7]. In [8],a new cloud storage model based on a distributed deduplication with the fingerprint index management (DDFI) model is proposed.In [9], Mansouri et al. considered the problem of data placement in the new generation tiered cloud storage services offering hot and cool tiers that were characterized by differentiated quality of service (i.e, access latency, availability and throughput) and the corresponding storage and access costs. In order to improve the query performance of cloud storage services, a column-oriented storage system based on distributed file system and computing framework, Haery, was proposed in [10]. The system used a cumulative high-dimensional data model, a complex linearization algorithm and an efficient query algorithm to solve how to apply a predefined, well-partitioned data model to flexible, time-varying key-value data. In [11], the architecture of data storage and analytic based on Spark is proposed in the big data lake to handle the increasingly dynamic data that need flexibility. Munir et al. in [12] considered different data fragmentation strategies (i.e., horizontal, vertical or hybrid) behave better or worse according to the access patterns of the subsequent operations. They presented a cost-based approach which helps deciding the most appropriate storage format in every situation.

In addition, when data owners subscribe to the cloud storage service, they will lose the control of outsourcing data. Therefore, it is also a key concern in the cloud storage field to check privacy and integrity of the data on the cloud. In [13], Reyes-Anastacio et al. designed and implemented an efficient service based on a provable data ownership encryption model that enabled organizations to verify the integrity of data on demand without retrieving files from the cloud. In [14]， an information leakage aware storage system in multicloud (StoreSim) was presented to solve an important information leakage problem caused by unplanned data distribution in multicloud storage services. In order to ensure the security of client reduplication, Youn et al. presented an efficient hash based random security proof scheme to prove the ownership of the file [15]. When the encrypted data stored in the cloud becomes large, the retrieval of encrypted data becomes an urgent problem to be solved. For this, Hoang et al. introduced a novel dynamic searchable symmetric encryption framework, called (IM)-DSSE, which can achieve a high level of privacy, efficient search/update, and low client storage with actual deployments on real cloud settings [16]. In order to verify the integrity of dynamic data in a big data platform, the blockchain technique instead of a Third Party Auditor was utilized to provide a better auditing service in [17].

However, with the large-scale adoption and increasing popularity of digital technology, a large amount of unstructured data is generated, and the scale of data also grows exponentially. The shortcomings of cloud storage system characterized by distributed files are increasingly exposed. Object storage has emerged as an example of the widespread adoption of storing massive data, particularly unstructured data, on the Internet [18]. For example, Amazon proposed Amazon Simple Storage Service [19], which divides traditional files, database tables, images, video and other information into fixed size data blocks, called "objects". These data objects are stored on the object storage device according to the data allocation algorithm. Examples for public cloud offerings, which belong to this kind of services, are the Google Cloud Storage (GCS) and Microsoft Azure Blob Storage [20]. Examples for service solutions are Eucalyptus Walrus, Ceph , Fake S3, Minio and OpenStack Swift, which provide similar functionality through implement API of the object storage service [20]. In order to overcome I/O bottleneck in managing large amounts of metadata and transaction logs, Yoon et al. proposed an object-based cloud storage coordination storage system (HMS) that combines non-volatile storage with traditional storage [21]. The rapid growth in the number of object storage services also increases the complexity of the infrastructure behind them. In order to properly operate and manage a complex object storage infrastructure, some data management strategies are needed to be provided to solve the challenges for managing the data efficiently in object-based storage systems. In [22], Wadhwa et al. presented a resource contention aware load balancing tool (iez) for large scale distributed object-based storage systems, and then extend iez to support Progressive File Layout for scientific applications and workflows.

It can be seen from the above publications that most of the existing research on big data storage focus on the performance improvement of large-scale cluster system. On the one hand, the traditional file storage system is constantly optimized; on the other hand, a new object storage service architecture is proposed to provide reliable, fast and extensible storage services through the object client at any time. However, the most popular cluster storage services for big data are file-oriented storage systems, which have been configured with non-object-oriented data types. If the file-based storage service is directly switched to object-based storage service, it is not compatible with the organization form of upper data, and has some compatibility problems with parallel analysis framework. To solve this problem, we take the object storage system as the background storage of Hadoop, and proposes a high availability object storage method to support the big data file access service.

III. Big data storage service architecture and its key technologies

A. Overall design of the big data storage service architecture

The architecture designed in our paper is to solve the problem of transparent access from the file system to the object storage cluster, so that the data storage platform based on the file system can support efficient storage and processing services of massive data, which makes up for the disadvantages of the file system in processing a large amount of unstructured data. The basic idea of the architecture design is roughly described as follows. Firstly, a distributed object storage is used as the backend storage of traditional file system. Secondly, a data protocol for transparent access from file system to object storage is designed. Thirdly, a multi-gateway fault tolerant management mechanism is presented to ensure high availability of object storage services. Among them, the most critical and basic step is to choose an appropriate distributed system to support object storage.

Currently, some storage systems adopt Ceph as the only back-end storage for Openstack or Hadoop, which is a model for data processing through running Hadoop based on Ceph distributed storage systems. Ceph is an open-source distributed storage system with high scalability, high availability and high performance [25], which is mainly composed of Monitor, OSD, and MDS. It uniquely provides three interfaces in the system: file-based interface, block-based interface and object-based interface, which are compatible with the API of object-interface service including Swift and S3. RADOS, the lowest core of Ceph, provides basic storage services for user data, which guarantees high reliability, high scalability, high performance and other characteristics of the system. Among them, a dynamic library based on RADOS layer, LIBRADOS, is the original and unified interface for clients to access Ceph cluster, which realizes common functions such as Crush algorithm and network communication. When Ceph distributed storage is used to replace HDFS in Hadoop, the distributed expansion capacity of Ceph system can be combined with the high-concurrency data processing capacity of Hadoop to improve the efficiency of data access and the overall performance of the system.

The basic architecture of big data storage service based on Ceph is shown in Fig. 1. It is divided into object storage client, file system client, RGW functional layer, Libcephfs interface layer, RGW interface layer and unified storage pool.

When managing user data, Hadoop or Fuse clients reach the gateway through the HTTP protocol, transform S3 protocol into an interface that directly calls the storage system and perform the function of accessing objects. When using S3 clients, the underlying S3 storage can be called using the file system, of which each layer logic is represented by a directory of files. When receiving the corresponding command sent by the file system client, the system checks the permissions of the command. After passing the check, parse the command and judge the configuration file, select the appropriate processing path, and return the result. When the client receives the result, it can directly access the object uploaded by the object storage service through the file system. By the way, the system can support for accessing S3 storage system.

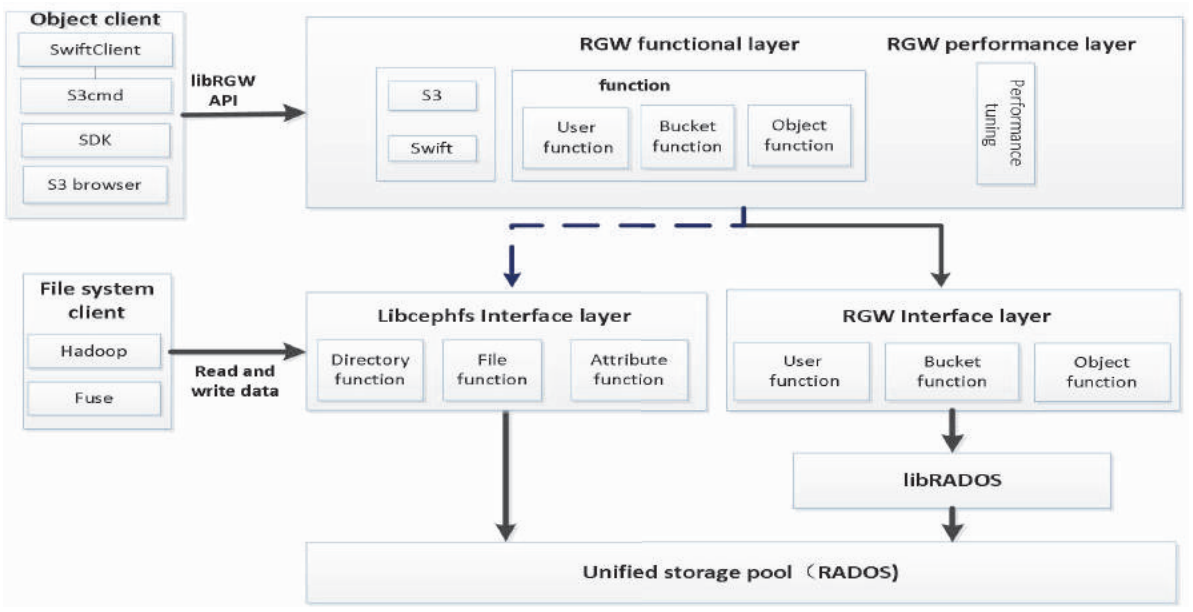

Fig.1 Big data storage service architecture based on object storage

Due to client limitations, only one *radosgw* gateway can be configured per client. When the *radosgw* gateway fails, the client will not be able to access the storage system properly. Therefore, in order to make the business run normally, in our system, multi-object storage gateway based on CTDB is used to transfer virtual IP in the gateway to the normal business node and improve the fault tolerance of the system.

CTDB is a lightweight cluster database that can manage node members, performs recovery/failover and IP relocation, and monitors the status of administrative services. Therefore, the high availability of application cluster can be realized based on CTDB. Fig.2 shows the high-availability architecture based on multi-gateway object storage of our system. When each node starts CTDB service, it creates and listens for the IP address of the local node. When the *radosgw* of a node fails, the nodes communicate with each other and their corresponding virtual IP is reallocated under the CTDB mechanism. For example, when the *radosgw* corresponding to the virtual IP node (100.10.1.2) fails, it will be migrated to the first node. Below, we will describe the key technologies of the architecture in detail.

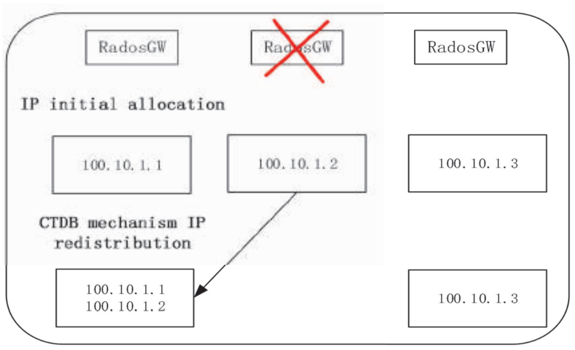

Fig.2 Object storage system based on multi-gateways

B. *Transparent access to the object-oriented storage service*

To facilitate the description of the problem, the concepts involved in transparent access are presented below.

Definition 1 Object Path (S) The path link information of the object in the standard protocol, that is, the URL of the object.

Definition 2 Access Permission (P) The parameters of permission verification for accessing to the server.

Definition 3 Access Aging (T) The valid time range of permission verification set by the server.

Definition 4 Object Gateway Service (M) The service used to identify storage gateway information of the object, such as host number and port number.

The basic steps of transparent access to object-oriented storage services proposed in this paper is shown in Fig.3. The description is as follows:

Step 1 Modify the attribute values of cluster configuration file *core-site.xml*, *mapred-site.xml*, *yarn-site.xml* and *hbase-site.xml* so that Ceph cluster is used as the back-end storage of the file system.

Step 2 Send an instruction from the file system to request data stored in the object storage system.

Step 3 Convert the instruction into the command format of standard object protocol.

Step 4 Take the host number and port number in the protocol as the basis of body identification to obtain the corresponding object gateway service M.

Step 5 Judge whether the target data can be obtained in this request according to the values of P and T in the request protocol.

Step 6 Return the target data in the object storage system to the end user.

C. *High availability mechanism based on object storage*

This paper realizes high availability of the architecture through CTDB cluster database. First, start the *radosgw* process of the object storage gateway on all monitoring nodes of the Ceph cluster. Second, install and configure the CTDB cluster on all

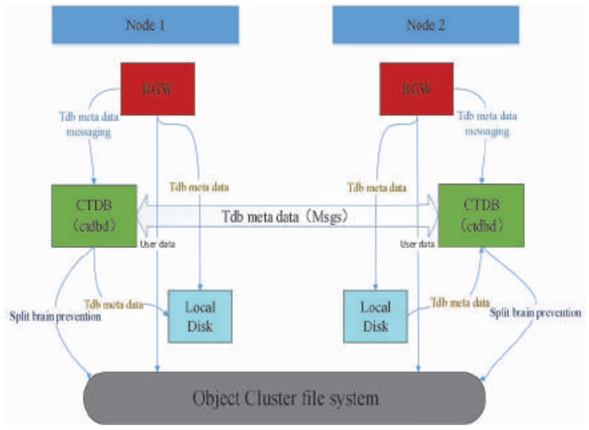

Fig. 3. Transparent access flowcharts

MON nodes. Third, utilize the CTDB to manage the object storage gateway for high availability of the object storage gateway. The relationship between the CTDB and the object storage service cluster is shown in Fig.4. In the figure, a CTDB daemon, *ctdbd*, is running on each cluster node. When the object gateway service (M) receives a command of storing data, it does not write data directly to its database, but interacts with its *daemon-ctdbd*. The *daemon* can interact with the metadata in the database over the network. During this process, the *ctdb*_daemon process handles events through epoll and checks the status of each node through running the monitoring script regularly. When the state of RGW gateway is abnormal, the corresponding process is performed. In addition, the ctdb recovered process also periodically queries the ctdb daemon for system status. When the RGW gateway needs to be repaired, the election process is initiated to select the repair node. After the election, start the repair process on the selected repair node, and inform all nodes to enter the repair state, repair Tdb(Trivial Database), redistribute public IP and trigger TCP reconnection.

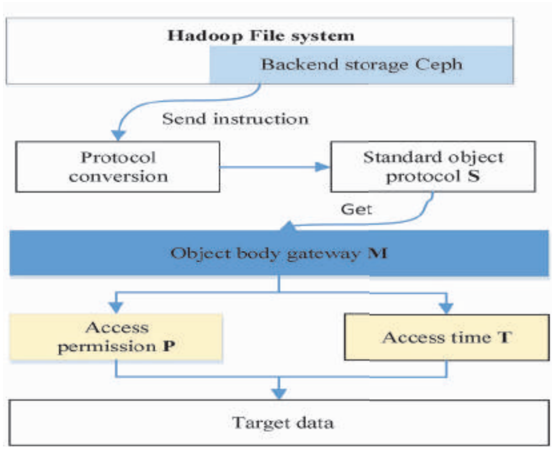

Fig.4 CTDB-based cluster graph of object storage services

IV. Experiment and Result Analysis

In order to verify the effectiveness of the architecture proposed in our paper, we have conducted the following experiments to verify the performance of the system.

In this experiment, we deployed 3 servers, each of which used 2 SAS disks as system disks, 1 SSD disk as MDS storage disk, 35 SATA disks as data storage disks, and configured Ceph system with two monitor nodes and two MDS Physical networks deployed in clusters fall into two categories: storage data network and storage business networks. The former is used for data transmission between storage nodes, and its IP range is 192.168.2.1~192.168.2.3. The latter provides external service interaction of clusters, and its IP range is 192.168.3.1~192.168.3.3. The hardware configuration for each server is shown in table 1.

We applied TestDFSIO to test the I/O performance for the transparent access method, that is, throughput or average I/O rate. TestDFSIO is a cluster performance testing tool for Hadoop platform. In the test, the number of files per read or write was 100-400, and the size of a single file was 1024-5120 Mb. And in the process of testing, we turn off the IP of the running object gateway so as to trigger a multi-gateway failover of the system.

TABLE 1 SERVER ENVIRONMENT CONFIGURATION

Ceph-Node: Ceph V10.2.0 Jewel Hadoop-Node: Hadoop-2.7.2
CPU: Intel(R) Xeon(R) CPU E5-2680 v3 @2.50GHz RAM: 192GB/node SAS Disk: 2/node, Raid1 SSD Disk: 1/node,800G。 SATA Disk: 35/node,4TB Switch: Ruijie RG-6220-48XS

The throughput for writing data of our big data storage architecture under different workloads is shown in Fig. 5. As can be seen from Fig. 5, when the file size is small, the system is far from reaching its throughput bottleneck. With the gradual increase of file capacity, the throughput of the system also increases, which indicates that the transparent access method can process data very quickly and up to 717Mb/s. When the file size is fixed, the throughput of the system grows slowly with the increase of the number of files. Finally, the throughput gradually stabilizes. Compared to writing data directly on the original HDFS system, the performance of our system is better.

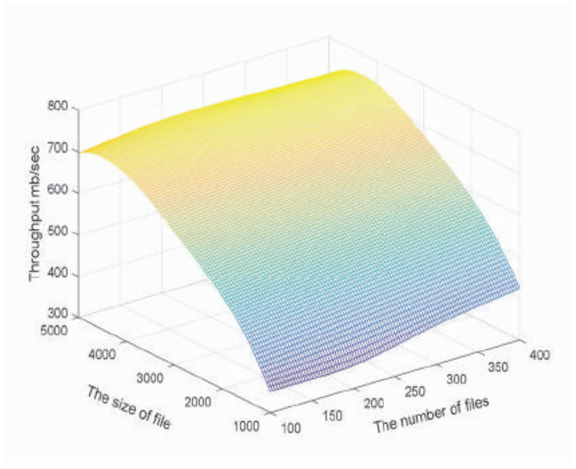

Fig.5 Throughput for writing data under different workloads

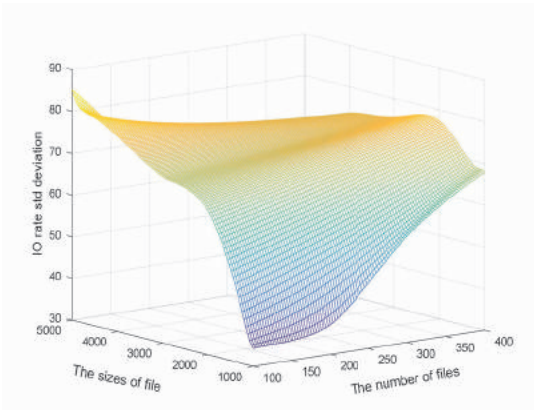

Fig.6 The deviation of I/O rate for writing data under different workloads

The deviation of I/O rate for writing data of the architecture under different workloads is shown in Fig. 6. As can be seen from Fig. 6, the deviation of I/O rate is very low when a small number of small data blocks are written to the storage. As the size and number of files increase, the deviation of I/O rate gradually increases. However, when the size of file is larger than 3500 M and the number of files is larger than 300, the deviation of I/O rate tends to be stable and stays at an acceptable level. It is not significantly different from the original HDFS. So, it proves that the transparent access method proposed in our paper does not significantly affect the basic performance of the original file system.

In general, it can be seen from the experimental results that the transparent access can write data blocks more stably.

In the following experiment, the number of files(N) is set to 100. The throughput for reading data of our big data storage service architecture under different workloads is shown in Fig. 7. As can be seen from Fig. 7, when the number of files is fixed, the throughput and average read rate of the system increase with the increasing of the size of file, but when the size of file reaches 5120 MB, these two indicators tend to be stable. In general, the transparent access method can read data stably. The reading performance of our system is in line with the original HDFS system.

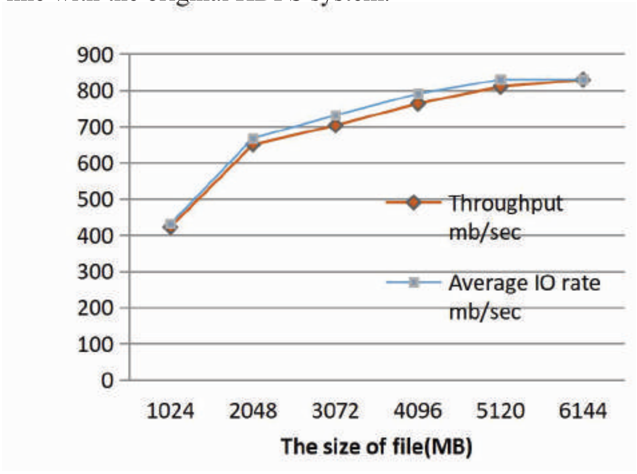

Fig.7 The reading performance of the system under N=100

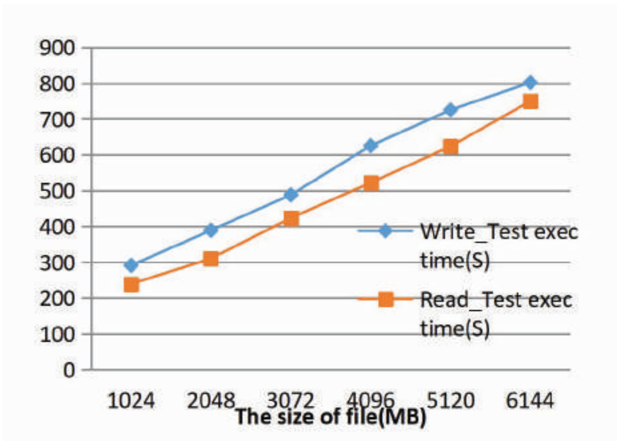

Fig.8 The time consumption of I/O under N=100

Fig. 8 shows the time consumption for I/O operations at different sizes of file when N=100. As the size of file increases, the time spent on I/O operations increases linearly. The overall performance of the system is no less than HDFS storage system, which can effectively solve the compatibility problem of data access from file system to object system. Meanwhile, the multi-gateway fault tolerance mechanism based on CTDB also ensures the high availability of the system.

V. CONCLUSION

In the age of big data, the storage service of unstructured data is used more and more frequently. For the problem that the big data parallel processing framework of deployed distributed file storage system is not compatible with object storage service, a new storage system architecture is imperative. Here are the advantages of the architecture:

1) It's able to support mining and analysis of massive data stored by object type on Hadoop platform with MapReduce.

2) A transparent access service interface from file system to object storage system is implemented, which supports access to object storage systems in the form of file.

3) An object storage method based on multi-gateway management is implemented to ensure the health of the system when the object storage gateway fails or the node fails.

By completing the above work, we can directly take the object storage system as the backend of the existing file system-based big data parallel processing architecture, and fully combine the advantages of object storage structure and file parallel processing model to complete efficient big data analysis services.

ACKNOWLEDGMENTS

This work was supported in part by Natural Science Foundation of China under Grants 62162047, Natural Science Foundation of Inner Mongolia under Grants 2019MS06029, Inner Mongolia Science and Technology Plan Project under Grant 2019GG372, 2021GG0155, the Open-topic of Inner Mongolia Big Data Laboratory for Discipline Inspection and Supervision (IMDBD2020012, IMDBD2021014), the Self-topic of Engineering Research Center of Ecological Big Data, Ministry of Education, Inner Mongolia Engineering Laboratory for Cloud Computing and Service Software, Inner Mongolia Engineering Lab of Big Data Analysis Technology.

REFERENCES

[1] Sravan Kumar R, Saxena A. Data integrity proofs in cloud storage[C]// Third International Conference on Communication Systems & Networks (2011).

[2] Aujla G S , Kumar N , Zomaya A Y , et al. Optimal Decision Making for Big Data Pro-cessing at Edge-Cloud Environment: An SDN Perspective[J]. IEEE Transactions on Indus-trial Informatic:1-1(2017).

[3] Abadi D, Agrawal R, Ailamaki A, Balazinska M, Bernstein PA, Carey MJ, Chaudhuri S, Chaudhuri S, Dean J, Doan A: The beckman report on database research. Commun ACM, 59(2):92–99 (2016).

[4] Carullo G , Mauro M D , Galderisi M , et al. Object Storage in Cloud Computing Envi-ronments: An Availability Analysis[J] (2017).

[5] Hill L , Levy F , Kundra V , et al. Data-Driven Innovation for Growth and Well-being[J]. (2015).

[6] Yin J , Tang Y , Deng S , et al. ASSER: An Efficient, Reliable, and Cost-Effective Storage Scheme for Object-Based Cloud Storage Systems[J]. IEEE Transactions on Computers, 66(8):1326-1340(2017).

[7] He J , Zhang Z , Li M , et al. Provable Data Integrity of Cloud Storage Service with En-hanced Security in Internet of Things[J]. IEEE Access, PP(99):1-1(2018).

[8] Saraswathi S S , Malarvizhi N . Distributed deduplication with fingerprint index management model for big data storage in the cloud[J]. Evolutionary Intelligence, 2020(11).

[9] Mansouri Y , Erradi A . Cost Optimization Algorithms for Hot and Cool Tiers Cloud Stor-age Services[C]// 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE Computer Society (2018).

[10] Song J , He H , Thomas R , et al. Haery: a Hadoop based Query System on Accumulative and High-dimensional Data Model for Big Data[J]. IEEE Transactions on Knowledge and Data Engineering, PP(99):1-1(2019).

[11] Yang C T , Chen T Y , Kristiani E , et al. The implementation of data storage and analytics platform for big data lake of electricity usage with spark[J]. The Journal of Supercomputing, 2020.

[12] Munir R F , Abelló, Alberto, Romero O , et al. A Cost-based Storage Format Selector for Materialization in Big Data Frameworks[J]. DISTRIBUTED AND PARALLEL DATABASES, 2020, 38(2):335-364.

[13] Reyes-Anastacio H G , Gonzalez-Compean J L , Morales-Sandoval M , et al. A data integ-rity verification service for cloud storage based on building blocks[C]// 2018 8th International Conference on Computer Science and Information Technology (CSIT). IEEE Com-puter Society(2018).

[14] Zhuang H , Rahman R , Hui P , et al. Optimizing Information Leakage in Multicloud Storage Services[J]. IEEE Transactions on Cloud Computing:1-1(2018).

[15] Youn T Y , Chang K Y . Bi-directional and concurrent proof of ownership for stronger storage services with de-duplication[J]. Science China Information Sciences, 61(3):032107 (2018).

[16] Hoang T , Yavuz A A , Merchan J G. A Secure Searchable Encryption Framework for Pri-vacy-Critical Cloud Storage Services[J]. IEEE Transactions on Services Computing, PP(99):1-1(2019).

[17] A J L , A J W , B G J , et al. Blockchain-based public auditing for big data in cloud storage-ScienceDirect[J]. Information Processing & Management, 2020, 57(6).

[18] Tavakoli N , Dai D , Chen Y . Client-side straggler-aware I/O scheduler for object-based parallel file systems[J]. Parallel Computing (2018).

[19] Mesnier M, Ganger G R, Riedel E.Object-basedStorage[J].IEEE Communications Maga-zine, 41 (8) :8 (2003).

[20] Baun C , Cocos H N , Spanou R M . OSSperf – a lightweight solution for the performance evaluation of object-based cloud storage services[J]. Journal of Cloud Computing, 6(1):24 (2017).

[21] Yoon S K , Youn Y S , Son M H , et al. Harmonized memory system for object-based cloud storage[J]. Cluster Computing, (2):1-14(2017).

[22] B Wadhwa. Scalable Data Management for Object-based Storage Systems[D]. Virginia Polytechnic Institute and State University,2020.