FSTOR: A Distributed Storage System that Supports Chinese Software and Hardware

Yuheng Lin[2], Zhiqiang Wang[1(✉)], Jinyang Zhao[3], Ying Chen[2], and Yaping Chi[1]

[1] Cyberspace Security Department, Beijing Electronic Science and Technology Institute, Beijing, China
wangzq@besti.edu.cn

[2] Department of Cryptography and Technology, Beijing Electronic Science and Technology Institute, Beijing, China

[3] Beijing Baidu T2Cloud Technology Co. Ltd., 15A#-2nd Floor, En ji xi yuan, Haidian District, Beijing, China

Abstract. In order to develop a distributed storage system that adapts to Chinese software and hardware, build a cloud computing platform that is independently usable, safe and reliable, data utilization is more concentrated and intelligent, and service integration is more unified and efficient. This paper designed and implemented a distributed storage system that supports Chinese software and hardware, which is compatible with Chinese mainstream CPU, operating system, database, middleware and other software and hardware environments. After a lot of experiments and tests, it is confirmed that the system has high availability and high reliability.

Keywords: Cloud computing platform · Distributed storage system · Localization

1 Introduction

The distributed storage system is a data storage technology that distributes data on multiple independent devices, and provides storage services as a whole externally1,2. It has the characteristics of scalability, high reliability, availability, high performance, high resource utilization, fault tolerance and low energy consumption3. Its development process can be roughly divided into three stages. One is the traditional network file system, which is typically represented by Network File System (NFS), etc., the second is the general cluster file system, such as Galley, Shared File System (GPFS), etc., and the third is the object-oriented transit distributed file system, such as Google File System (GFS), Hadoop Distributed File System (HDFS), etc. NFS4,5 is a UNIX presentation layer protocol developed by SUN; GPFS6,7 is IBM's first shared file system. GFS8 is a dedicated file system designed by Google to store massive search data. The above-mentioned typical distributed storage systems are all developed by foreign companies, and all have incompatibility with Chinese software and hardware.

Z. Qian et al. (Eds.): WCNA 2021, LNEE 942, pp. 342–350, 2022.
https://doi.org/10.1007/978-981-19-2456-9_36

In response to the above problems, this paper designed and implemented a localized distributed software-defined storage system named FSTOR, which is based on B/S architecture, has standard interfaces and supports various localized operating systems and virtualization systems, and both servers and databases are localized facility. The system implements distributed cloud storage block storage services, snapshot management, full-user mode intelligent cache engine, cluster dynamic expansion, pooled storage function, fault self-check and self-healing functions.

The organization structure of this article is as follows: The first part introduces the relevant research background of the system; the second part introduces the system architecture; the third part describes the functional architecture of the system; the fourth part tests the system and analyzes the test results; the fifth part summarizes full text.

2 System Structure

The detailed system architecture is shown in Fig. 1. The overall technology and software system can run normally on the Chinese CPU. The Chinese x86 architecture Zhaoxin, the ARM architecture Feiteng and the Alpha Shenwei can be used, and the operating system Kylin or CentOS can be used. The system can use automated operation and maintenance technology to ensure daily operation and maintenance management, including but not limited to data recovery, network replacement, disk replacement, host name replacement, capacity expansion, inspection, failure warning, capacity warning, etc.

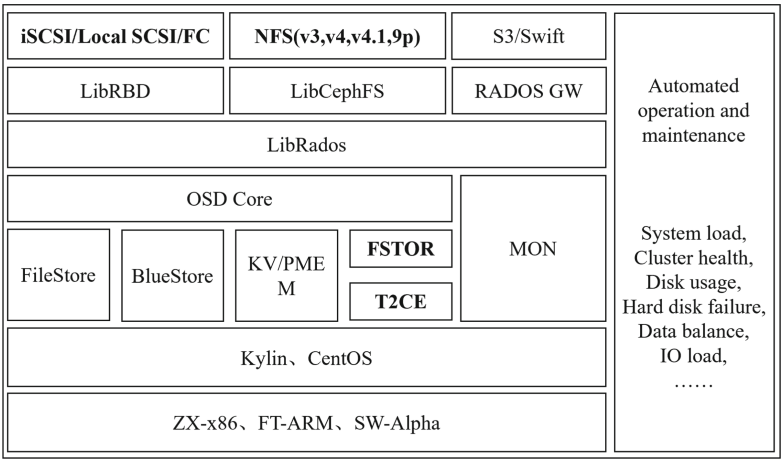

Fig. 1. System architecture diagram

(1) **LibRBD**
A module that supports localized block storage, abstracts the underlying storage, and provides external interfaces in the form of block storage. LibRBD supports the localized virtualization technology to be mounted to the localized operating system through the RBD protocol, and is provided to some localized databases.

(2) **Libcephfs**
A module that supports localized Posix file storage, supports the Kylin and the CentOS operating system to mount the file system locally to the Chinese operating system through the mount command and provide it for use.
(3) **RADOS GW**
In order to support a gateway module for localized object storage, two different object storage access protocols, S3 and Swift, are provided. Localized software can use these two protocols to access the object storage services provided by the system.
(4) **Librados**
A module supporting blocks, files, and object protocols is responsible for interacting with the core layer of the Chinese storage system. It is a technical module of the interface layer.
(5) **MON**
The brain of the system. The management of the storage system cluster is handed over to MON.
(6) **OSD Core**
Responsible for taking over the management of a physical storage medium.
(7) **FileStore**
An abstract module that manipulates the file system. The system accesses business data through the Poxis standard vfs interface. The space management of the physical disk is handed over to the open source xfs file system to manage.
(8) **BlueStore**
A small Chinese file system. It can replace the xfs file system to manage the physical disk space, reducing some performance problems caused by the xfs file system being too heavy.
(9) **T2CE**

A Chinese smart cache module. The system can make full use of physical hardware resources to improve storage performance. Its intelligent caching engine can perceive data characteristics and frequency, and store data that meets a predetermined strategy on high-speed devices, and store data that does not meet the predetermined strategy on slow devices. Under the premise of not significantly increasing hardware costs, use high-speed equipment to drive low-speed equipment to ensure business performance requirements.

The intelligent cache engine revolves around the close cooperation between multiple core modules such as IO feature perception, intelligent aggregation, disk space allocation and defragmentation, and maximizes the combination of high-speed and low-speed devices between performance and capacity to achieve a perfect balance. The smart cache uses a large number of efficient programming models and algorithms to maximize the performance of high-speed devices.

3 Function Architecture

The system function framework is shown as in Fig. 2. The system includes a hardware abstraction layer, a unified storage layer, a storage service layer, an interface protocol layer and an application layer. The unified storage layer includes multiple copies,

pooling, tiered storage, linear expansion, fault medical examination, data recovery QoS, erasure coding, strong data consistency, intelligent caching, dynamic capacity expansion, fault domain and fault self-healing. The storage service layer includes snapshot cloning, data link HA, data stream QoS, encryption compression, quota control, thin provisioning, multipart upload, permission control, version control, multi-tenancy, data tiering, and write protection. The interface protocol layer includes block storage interface, object interface and file storage interface. The application layer includes virtualization, unstructured data and structured data.

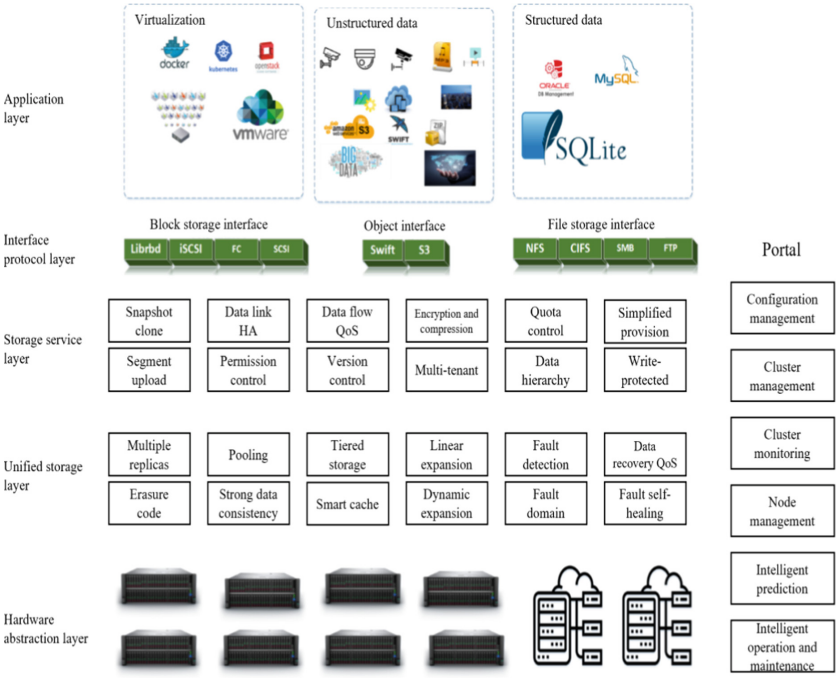

Fig. 2. Functional architecture diagram

(1) **Object Storage Segmented Upload**
Segmented upload is the core technology of breakpoint continuingly functions. When the fault is restored, avoid re-uploading the content of the uploaded file and cause unnecessary waste of resources. Users can also implement user-side QoS functions based on the multipart upload function. The multipart upload function will verify the content of the uploaded file, and the parts that fail the verification will be re-uploaded.

(2) **Dynamic Capacity Expansion and Reduction Without Perception**
The system supports dynamic capacity expansion and contraction without perception, and can respond to changes in application requirements in a timely manner

without perception of the application, ensuring the continuous operation of the business. In addition, the performance also increases linearly with the increase of the number of nodes, giving full play to the performance of all hardware.

(3) **Data Redundancy Protection Mechanism**
The system provides two different pool data redundancy protection mechanisms: replica and erasure code to ensure data reliability.

Replica mode is a data redundancy realized by data mirroring, with space for reality. Each replica keeps complete data, and users can pool 1–3 replicas according to specific business requirements to maintain strong consistency. The greater the number of replicas, the higher the fault tolerance allowed, and the consumed capacity increases proportionally.

Erasure code mode is an economical redundancy scheme, which can provide higher disk utilization. Users can choose K + m combination according to the specific business requirements. K represents to store the original data in K blocks, and M represents to generate M pieces of coded data. The size of each piece of coded data is the same as that of the block. The K pieces of block data and M pieces of coded data are stored separately to achieve data redundancy. According to any k pieces of data in K + m, the original data can be reconstructed.

(4) **Troubleshooting**

The system supports a variety of different levels of fault domain design, the smallest fault is the tiered disk, and the largest fault tier can be the data center. It is common to use the cabinet as the fault level, and the user can divide it according to the actual situation. The fault domain can ensure the failure level of data redundancy. Whether it is a failure of a disk, a rack, or a data center, the reliability of the data can be guaranteed. At the same time, the system also supports intelligent fault detection and fault self-healing and alarms to avoid manual intervention, and supports intelligent data consistency verification to avoid data loss due to silent errors.

4 System Test

4.1 Test Environment

The test environment topology is shown in Fig. 3. Four node servers and a notebook are used. The server and notebook are connected to the switch. FIO 2.2.10 (cstc10184742) is used as the test tool.

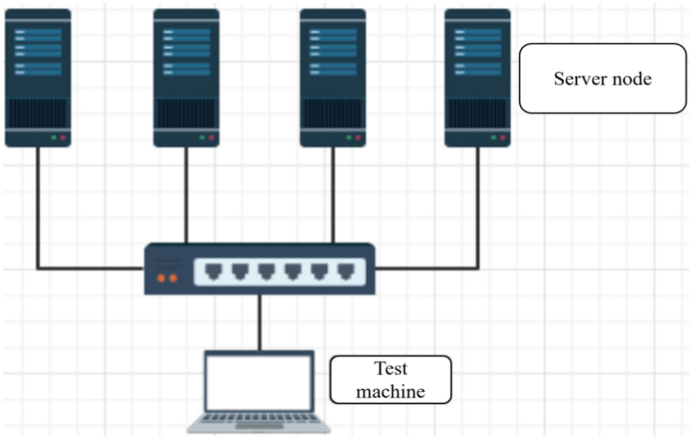

Fig. 3. System test network topology

The model and configuration of server and client are shown in Table 1. In the test, the model and configuration of the four node servers are the same, all of them are Kylin system, and the CPU is FT1500a@16c CPU.The notebook is the ultimate version of Windows 7 system, the model is ThinkPad T420, and the notebook is equipped with Fio.

Table 1. Environment configuration

Equipment name	Model and configuration	Operating system	Software configuration
Node server (4)	CPU: FT1500a@16c CPU 1.5 GHz RAM: 64 GB hard disk: 1.8TB	Kylin V4.0	FSTOR distributed storage system MariaDB V10.3 RabbitMQ V3.6.5
notebook (1)(CSTC10124326)	model: Thinkpad T420 CPU: Intel Core i5-2450M 2.50 GHz RAM: 4 GB hard disk: 500GB	Windows 7 Ultimate	Google Chrome 52.0.2743.116 Fio 2.2.10

4.2 Test Content

The content of system test is shown in Table 2. IOPs (input/output operations per second) is the input/output volume (or read/write times) per second, used for computer storage device performance test. The test results show that the system realizes the functions designed in all functional architectures.

Table 2. Test Content

Technical index	Test results
Block storage service	The block storage volume can be successfully created and the storage volume can be mapped to the virtual machine File system can be created for storage volume
Snapshot management	Supports the snapshot function of storage volumes, and clones new storage volumes through snapshots You can perform a rollback operation on the storage volume that has been snapshotted
Smart cache engine	The smart cache engine storage pool can be successfully created
Cluster dynamic expansion	A new storage server or hard disk can be added to the storage cluster
Pool storage function	Can create storage pools with different performance
Fault self-checking and self-healing	Delete an object storage device and kick it out of the cluster, and cluster business will not be interrupted
Web storage mount	Web storage can be mounted via NFS protocol
4k random write	4k random write without cache IOPS: 1694 4k random write IOPS with cache: 5149
4k random read	4k random read without cache IOPS: 2474 4k random read IOPS with cache: 6507
4k mixed random read and write	4k mixed random read without cache IOPS: 1944 4k mixed random read with cache IOPS: 4863 4k mixed random write without cache IOPS: 648 4k mixed random write buffered IOPS: 1621

4.3 Test Results

(1) **System Structure**
The system is based on B/S architecture, the server adopts Kylin v4.0 operating system, the database adopts MariaDB V10.3, the middleware adopts RabbitMQ v3.6.5, and the bandwidth is 1000Mbps. The client operating system is the ultimate version of Windows 7, and the browser adopts Google Chrome 52.0.2743.116.

(2) **Performance Efficiency**

The system performance is as follows: 4K random write without cache IOPs: 1694; 4K random write buffer IOPs: 5149; No IOPs: 4K random read cache; 4K random read buffer IOPs: 6507; 4K mixed random read without cache IOPs: 1944; 4K mixed random read buffer IOPs: 4863; 4K mixed random write without cache IOPs: 648.

5 Conclusions

Aiming at the problem that the distributed storage system needs localization and supports Chinese software and hardware, this paper designed and implemented a distributed storage system namedFSTOR, which runs on the Chinese operating system and CPU, and each module supports localization. The system ensures the daily operation and maintenance management by realizing automatic operation and maintenance, and ensures the reliability of data through two pool data redundancy protection mechanisms and fault or division methods: copy and erasure code. After a large number of tests, the system runs stably, realizes complete functions, and achieves high reliability and high availability.

Acknowledgments. This research was financially supported by National Key R&D Program of China (2018YFB1004100), China Postdoctoral Science Foundation funded project (2019M650606) and First-class Discipline Construction Project of Beijing Electronic Science and Technology Institute (3201012).

References

1. Zhu, Y., Fan, Y., Yubin, W., et al.: An architecture design integrating distributed storage. Henan Sci. Technol. **40**(36), 22–24 (2021)
2. Lin, C.: Research and Implementation of Replica Management in Large-scale Distributed Storage System. University of Electronic Science and Technology of China (2011)
3. Li, G., Yang, S.: The analysis of the research and application of distributed storage system. Network Secur. Technol. Appl. **2014**(09), 73+75 (2014)
4. Sandberg, R.: The sun network filesystem: design, implementation and experience. In: Proceedings of USENIX Summer Conference, pp. 300–313. University of California Press (1987)
5. Huang, Y.: Docker data persistence and cross host sharing based on NFS. North University of China, pp. 22–24 (2021)
6. Schmuck, F., Haskin, R.: GPFS: A shared-disk file system for large computing clusters. In: Proceedings of the Conference Oil File and Storage Technologies (FAST 2002), 28–30 January 2002, Monterey, CA, pp. 231–244 (2002)
7. Zhang, X.-N., Wang, B.: Installation configuration and maintenance of GPFS. Comput. Technol. Dev. **28**(05), 174–178 (2018)
8. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. ACM SIGOPS Operat. Syst. Rev. **37**(5), 29–43 (2003)