Hindawi
Journal of Sensors
Volume 2021, Article ID 1500829, 16 pages
https://doi.org/10.1155/2021/1500829

Research Article

Optimized Design of Multilines Center of Subway AFC System via Distributed File System and Bayesian Network Model

Hui Fang,[1,2] **Jiandi Jiang**,[2] **Feng Lin**,[3] **and Wei Zhang**[1]

[1]*Zhejiang University, Hangzhou, Zhejiang 310027, China*
[2]*Mechanical and Electrical Department, Ningbo Rail Transportation Group Co., Ltd., Ningbo, Zhejiang 315040, China*
[3]*Digital Information Division, UniTTEC Co., Ltd., Hangzhou, Zhejiang 310051, China*

Correspondence should be addressed to Wei Zhang; cstzhangwei@zju.edu.cn

Received 7 September 2021; Revised 26 October 2021; Accepted 9 November 2021; Published 20 December 2021

Academic Editor: Mu Zhou

Automatic fare collection system (AFCS) is a modern, automatic, networked toll collection system for rail transit ticket sales, collection, billing, charging, statistics, sorting, and management. To realize the subway transit networking operation, this paper designs the subway AFCS based on a distributed file system (DFS), namely, Gluster File System (GlusterFS). Firstly, the multiline center (MLC) in the subway AFCS is designed to analyze the status and current situation of distributed file processing in subway MLC system; secondly, the relevant technical theories are summarized, the Bayesian Network (BN) theoretical model and DFS are explored, and the principles of four DFS are comparatively analyzed; thirdly, the architecture and cluster mode of GlusterFS is expounded, and then based on GlusterFS, the architecture of subway AFCS is discussed. This paper presents several innovation points: first, the subway AFCS is designed based on GlusterFS by analyzing and aiming at the functional requirements, performance requirements, and safety requirements of the MLC subway system; second, the safety risk analysis (SRA) of AFCS is conducted from six security requirements, and a Web scanning system is designed to ensure the system data security. Finally, the design scheme is used to analyze the subway passenger flow and power consumption. The results demonstrate that the design scheme can competently adapt to three different application scenarios. Through comparison of two deployment modes of the Web scanning system, the data security Web scanning system can ensure the safe operation of the AFCS. Furthermore, the statistical analysis of subway passenger flow and power supply data shows that the proposed scheme can support the smooth operation of the subway system, which has significant practical value.

1. Introduction

As early as the 1970s, some foreign countries have already adopted the automatic fare collection system (AFCS) in urban rail transit (URT) [1]. The current social development and technological advancement have seen the formation of a more intelligent and reliable AFCS [2]. Multiple lines under the same operating entity share a multilines center (MLC) to realize rail transit networking operation through the same MLC and the AFC clearing center (ACC) interface; thus, MLC is the core of the multiline AFCS [3]. The newly built MLC system usually adopts the microservice architecture and runs on the private cloud or container cloud to ensure high availability, scalability, and operability [4]. The system itself or its internal services generally invoke each other through Remote Procedure Call Protocol (RPC), while such core data as transaction and reconciliation in the subway system is usually transmitted through the File Transfer Protocol (FTP) service in the form of files. The FTP server is essentially a long-connected and stateful service based on Transmission Control Protocol (TCP), which is usually single [5]. Additionally, new features, such as microservices, virtualization, and containerization also require the infrastructure layer to provide the function of sharing data volumes and files [6]. In the subway MLC system, there are mainly four kinds of data files [7]: (1) data interface between MLC and subordinate systems, which is stipulated by the urban subway network specification and contains messages and data files, among which the transaction data files and MLC parameter files are usually transmitted over FTP; (2)

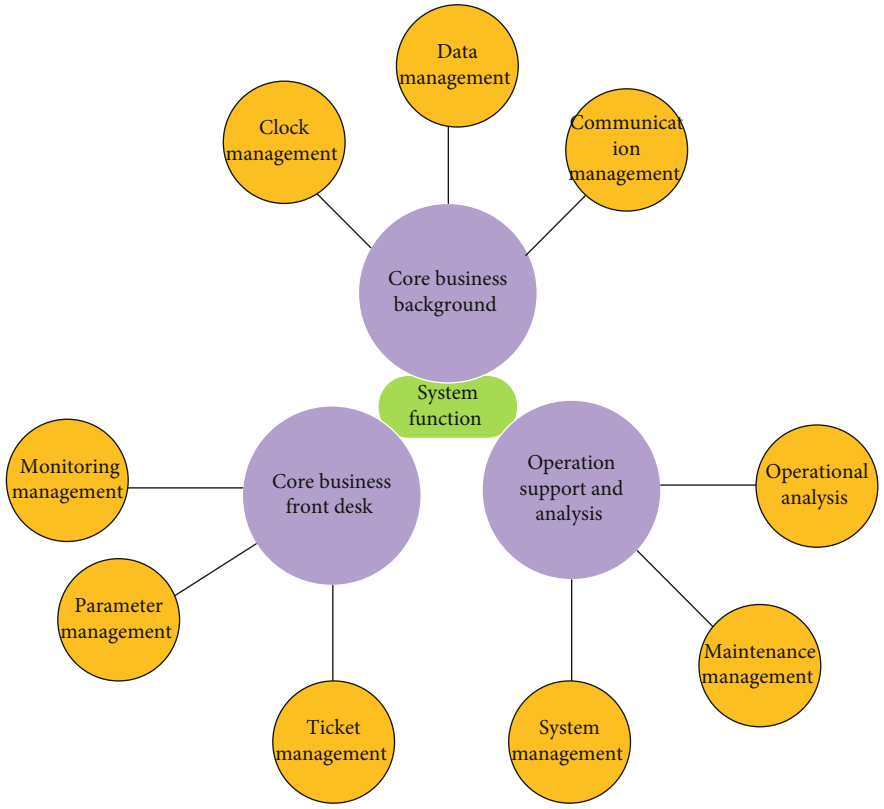

Figure 1: Functions of each systematic module.

data interface between MLC and the subway external system [8], which is jointly negotiated by the MLC system and the external system and can be presented invarious forms, such as messages, message queues, and files. The reconciliation data files are usually transmitted over FTP; (3) intertransmitting data files among subsystems within MLC. For example, the management workbench passes the generated parameter file to the communication service, and the communication service sends the file to the station over FTP [9]; and (4) backup files among primary-spare centers of MLC. Backup data of the master center and the disaster recovery center consists of database and data files [10].

To store data files, MLC usually mounts a block device of Storage Area Networking (SAN) on a computing node and then establishes a local file system (LFS) on the block device. However, the computing node that mounts a SAN volume is a single point, and SAN has some shortcomings. Typically, as a kind of dedicated device, SAN usually designates specific manufacturers for maintenance or expansion in the life cycle of the project. Therefore, the distributed file system (DFS) is taken as a solution scheme [11, 12].

Based on the above description, this paper tries to analyze and meet the functional requirements, performance requirements, and safety requirements of the MLC subway system design. First, four data cluster modes of the DFS software Gluster File System (GlusterFS) are analyzed, and the requirements of subway AFCS are discussed. Then, a subway MLC system is designed based on GlusterFS. Further, the six major risks in the MLC system are explored. Finally, the DFS-based subway security AFCS is constructed to ensure the security of system data, and the design scheme is applied to the concrete scenario. The designed MLC in AFCS cannot only meet the common functions of the subway system, such as ticket selling, checking, billing, charging, statistics, sorting, and management, but also can analyze the safety risks of the subway system and meet the six major safety requirements of the subway system.

2. Overview of Related Technologies

2.1. System Requirement Analysis (RA)

2.1.1. Functional RA. The fundamental functions of the AFCS for subway rail transit in China should include ticket management, ticket price management, data control, and urban rail transit card interface.

According to the overall planning of the system, the system can be roughly divided into three functional modules: core business background, core business foreground, and operation support and analysis. Figure 1 expatiates the functions of each module.

2.1.2. Performance RA. The performance requirements of the subway rail transit system are as follows.

(1) *Elementary Performance Indicator*. System capacity should be able to meet the line access and processing requirements of the maximum passenger flow within the jurisdiction. The system should accept and process the ticket price list and related system

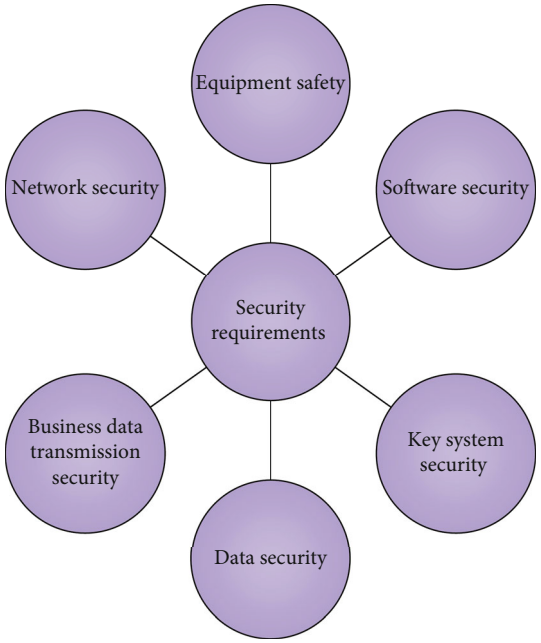

Figure 2: System security requirements.

parameters of the sorting center. In the case of normal network communication conditions, it must have the ability to receive various parameters set by ACC in real-time and issue various commands to SC devices in time

(2) *Transaction Data Processing Performance Indicator*. The system also needs to be able to handle multiple lines simultaneously

(3) *Report Generation Performance Indicator*. All operating reports in the system need to be completed within the specified time

(4) *Data Backup and Recovery Performance Indicator*. The regional center stores and manages the data collected by the station equipment and backs them up in time, so that different operators can sort, reconcile, and restore the damaged or lost data

2.1.3. Security RA. The AFCS is the center of subway rail transit, and security is crucial. Figure 2 demonstrates the security requirements.

Here is the expatiation of Figure 2.

(1) Equipment security

The AFCS needs management equipment to record the information in detail; the system data should be updated in real-time, including the system blacklist and configuration parameters.

(2) Software security

Software security in AFCS primarily refers to user identification, access restrictions, security protection, and monitoring management. Managers of the AFCS need to set the

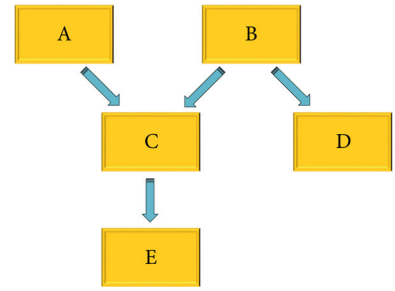

Figure 3: Directed acyclic graph.

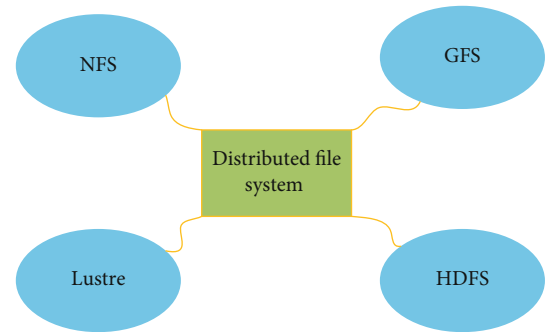

Figure 4: Classification of distributed file system.

corresponding user access accounts and passwords. Only through legal applications can access permissions be obtained. Additionally, the system must have a robust automatic data backup and recovery ability against misoperations.

(3) Key and system security

The key means the main key and sensitive data stored in the SAM card related to all offline transactions. The main key needs to set different versions and indicators to prevent them from losing.

(4) Data security

A considerable amount of transaction data will be generated in the AFCS, which are at risk of leakage, theft, tampering, and damage. Therefore, a complete AFCS must strengthen the management of data security from both hardware and software environments.

2.2. Bayesian Network (BN) Theory. BN is usually regarded as the fusion of the association graph (or directed graph) and Bayes theorem [13, 14]. The BN, or the belief network, mainly indicates the conditional probability relationship and causal relationship between variables. The conditional probability refers to the probability of an event occurring under the occurrence of another event [15], which is most commonly modeled using a directed acyclic graph (DAG). In a DAG, each node represents a variable, and the nodes are associated with each other. Generally, the relationship between nodes is represented by an arrow, illustrating the probability relationship between variables [16, 17]. Figure 3 reveals a simple DAG.

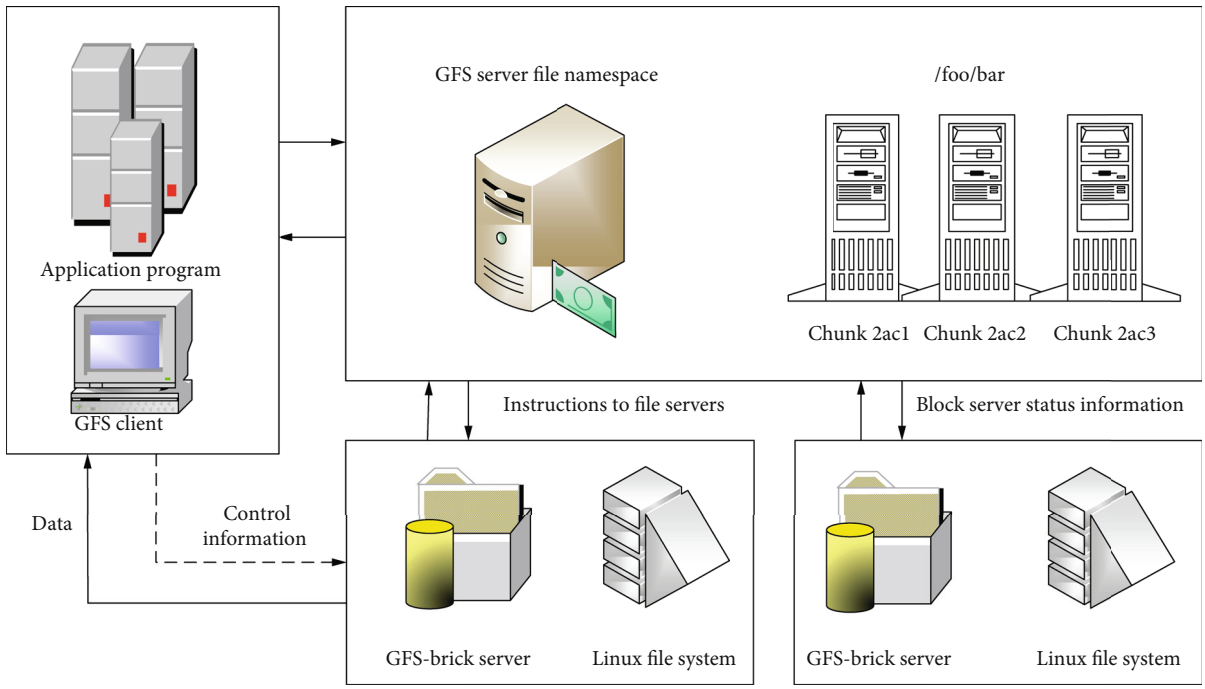

Figure 5: Google File System.

As displayed in Figure 3, the starting and ending nodes of each side are usually defined as the parent node and the child node, respectively; A is the parent node of C, and C is the child node of A; B is also a parent node, A and B are both the parent nodes of C, and C is a common child node of A and B. The node after another node is usually defined as a child node, and the node before another node is usually defined as its parent node [18]. For example, C and E are both the child node of A and B, and E is also a child node of C. In terms of node C, it can be used as either the child node of A and B or the parent node of E. In contrast, D is the unique child node of B. Obviously, the BN is an acyclic network [19, 20].

2.3. DFS

2.3.1. Brief Introduction to DFS. DFS is a storage system that can store data or information in multiple different devices [21, 22]. Its essence lies in the unified management (storage and reading) of computer network files and computer system local files by creating a large file system with a unified name. The DFS is compatible with other file systems with an ordinary file disk interface [23].

2.3.2. Typical DFS. Figure 4 illustrates the categories of DFS.

As shown in Figure 2, DFS contains the Network File System (NFS), Google File System (GFS), Lustre file system, and Hadoop Distributed File System (HDFS) [24].

(1) NFS

NFS provides users with easy and real-time access to network files anywhere through three versions: NFSv2, NFSv3, and NFSv4 by sharing data on remote servers with the local file. The NFS adopts the communication mode of RPC. When users access the network remote file, the NFS client will make a request to the server through the network. After receiving the request, the PRC finds the file access port and then passes the file to the NFS client, so that the file can be accessed normally.

(2) GFS

GFS is a DFS developed by Google, composed of a master node and multiple block servers. As shown in Figure 5, these system nodes are based on Linux machines. The block servers and clients can run simultaneously on a machine equipped with servers with better performance [25].

(3) Lustre file system

The Lustre file system is an open-source file system with high requirements for computer performance. The top ten computers in the world are equipped with the Lustre file system. In a file cluster, the Lustre file system is usually outfitted with tens of thousands of clients, which can handle tens of megabytes of storage. The architecture of the Lustre file system usually includes three parts, namely, metadata server (MDS), object storage server (OSS), and management server (MGS). MDS is mainly used for metadata storage. MDS can process 3 billion files according to statistics. OSS is primarily responsible for the actual exclusive data storage, which can provide multiple network request services for the unit storing the user file data. MGT is responsible for the registration and configuration information of the file system [26]. Figure 6 illustrates the architecture of the Lustre file system.

(4) HDFS

HDFS has strong data processing ability, able to process large data sets across computer clusters. Besides, HDFS has

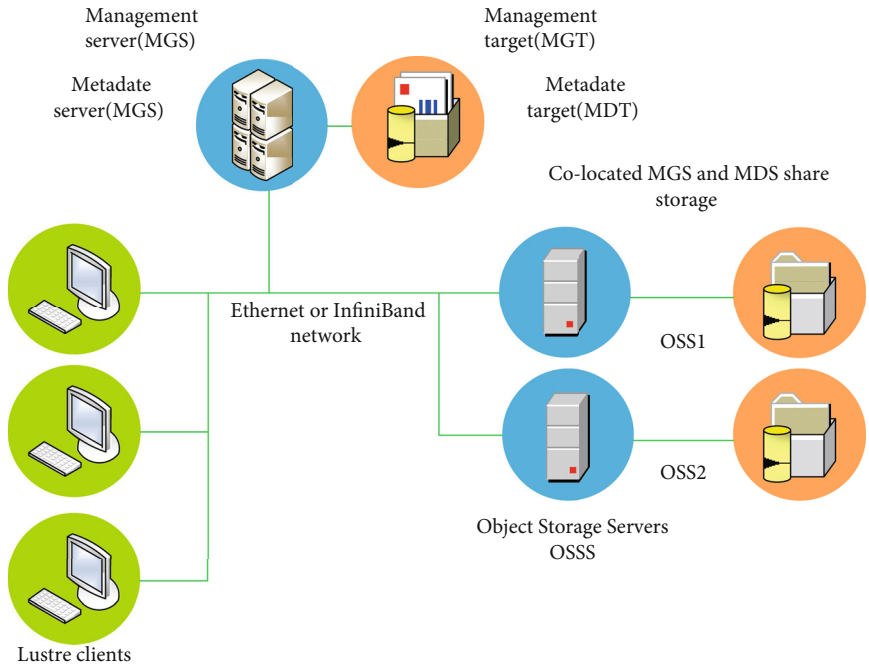

Figure 6: Architecture of the Lustre file system.

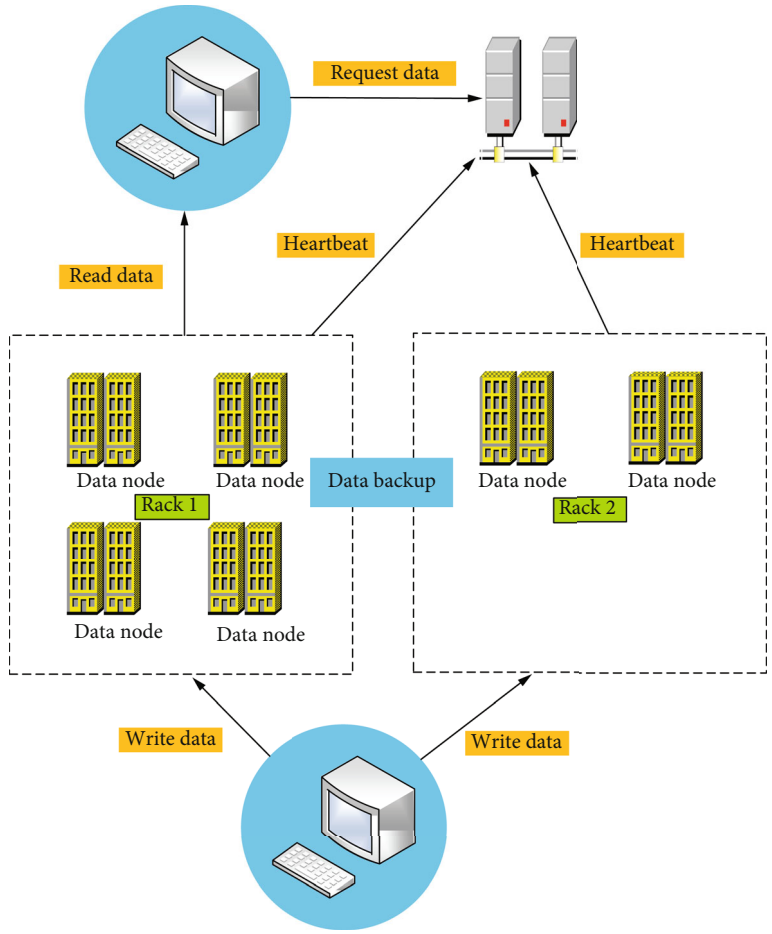

Figure 7: Architecture of Hadoop distributed file system.

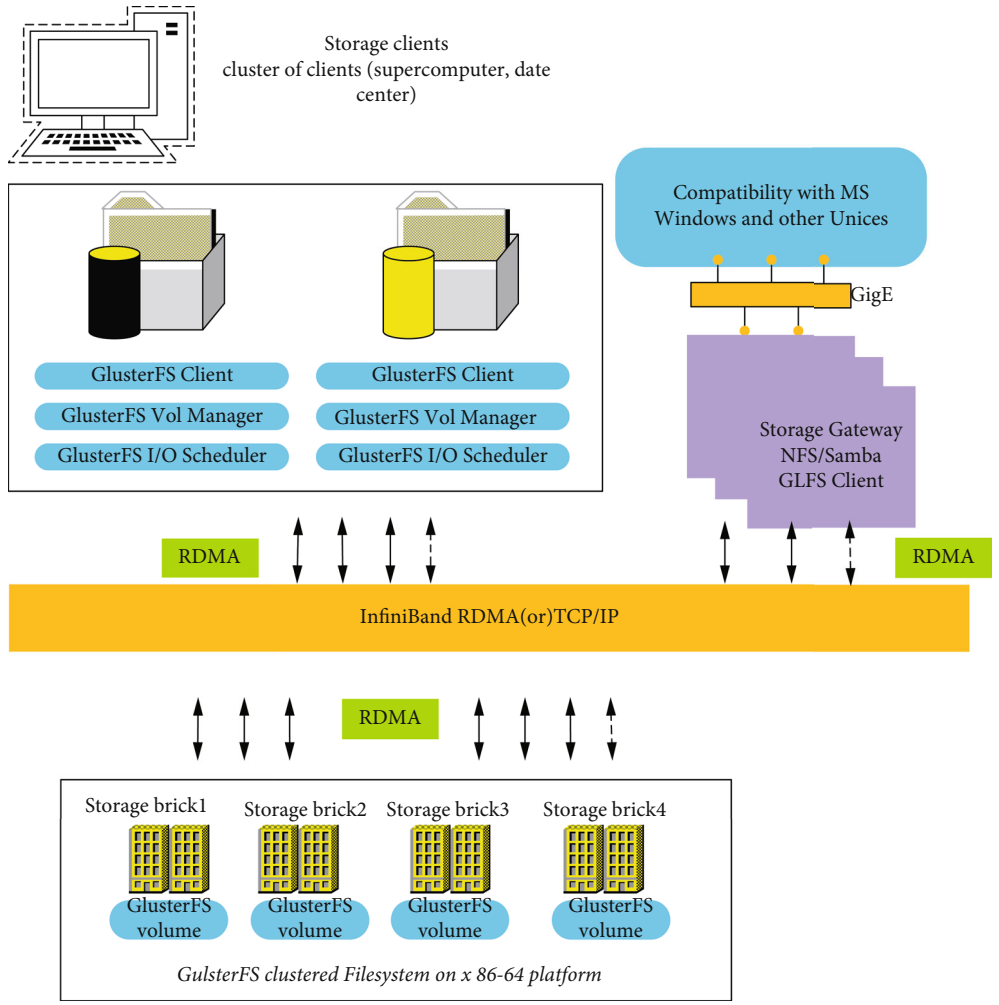

Figure 8: Architecture of Gluster File System.

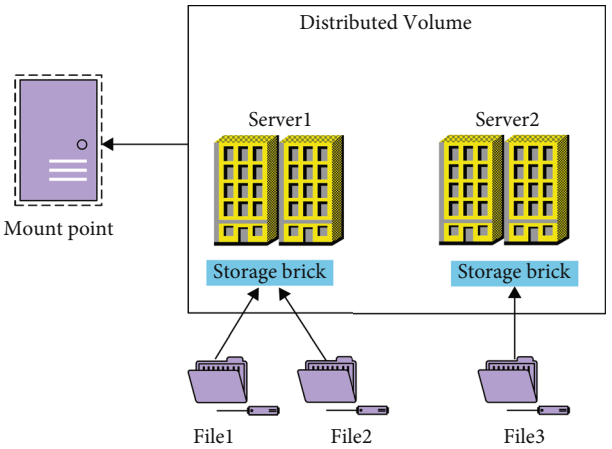

Figure 9: Framework of distributed Gluster File System volume.

low requirements for hardware, so it can run on ordinary and inexpensive hardware. HDFS is divided into metadata of the storage file system and application data. As presented in Figure 7, HDFS can store all files in the system as a series of blocks of the same size [27, 28].

2.3.3. Metadata Management. There are two main ways of metadata management. (1) Centralized metadata management. Centralized metadata only uses one metadata server, which is outstanding in data management. However, it has two inevitable problems of performance bottleneck and single point failure. (2) Distributed metadata management. The distributed metadata management disperses data in multiple servers. Each server is independent with the capability to provide independent data services. This method solves the two inevitable problems of centralized metadata management but also makes the system complicated with a performance overhead and consistency problems [29, 30].

2.4. GlusterFS

2.4.1. Architecture of GlusterFS. GlusterFS is an open-source DFS developed by Red Hat Inc. It has strong horizontal scalability and can integrate data from different servers. Figure 8 provides the architecture of GlusterFS.

Figure 8 shows that the server and client make up the GlusterFS; Remote Direct Memory Access (RDMA) or TCP is used to link the cluster server; the server can contain one or multiple nodes, and each node may contain multiple

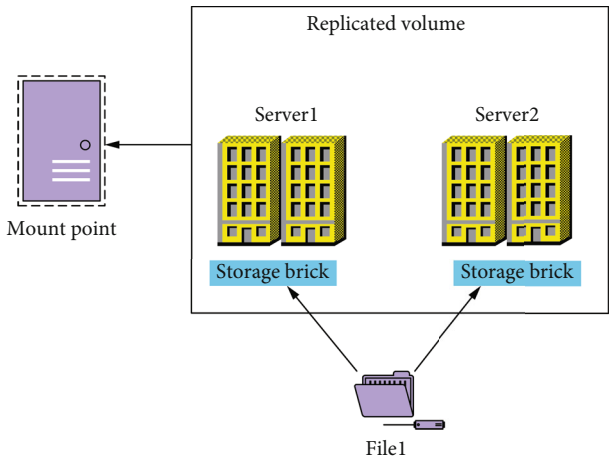

Figure 10: Framework of replicated Gluster File System volume.

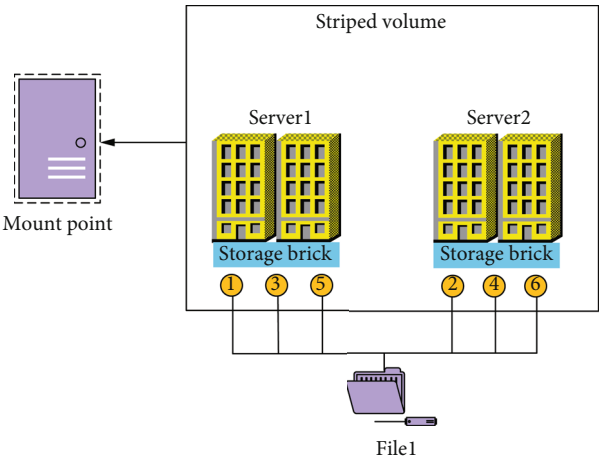

Figure 11: Framework of striped Gluster File System volume.

blocks; additionally, each block can be any directory in the LFS, and then, multiple blocks are combined to form various types of volume.

2.4.2. Cluster Modes of GlusterFS

(1) Distributed GlusterFS Volume (DGV). DGV assigns data files to block through the default algorithm and adjusts the volume capacity at a low cost. However, it cannot guarantee the data fault tolerance. The loss or damage of single block data will lead to the loss or damage of the entire file data. Figure 9 signifies the framework of two-block DGV, which utilizes the default algorithm to determine which block File 1, File 2, and File 3 belong to.

(2) Replicated GlusterFS Volume (RGV). RGV can reduce data damage or loss in DGV through the data backup mechanism of each block. The damaged data can be retrieved from the backup information of other blocks, or the RGV mechanism can be used to repair and restore data, thus greatly improving the fault tolerance. Figure 10 shows the framework of the two-block RGV.

As shown in Figure 10, the information of File 1 is stored in every block, so that the damaged File 1 information can be recovered through other blocks.

(3) Striped GlusterFS Volume (SGV). SGV divides the file into multiple blocks and then puts the file into a separate block in the way of "block packaging," thus achieving low redundancy and fault tolerance. However, SGV can improve the data reading ability. Figure 11 displays a two-block SGV.

(4) Dispersive GlusterFS Volume (DIGV). DIGV is realized based on erasure codes. The reliability can be adjusted by specifying the redundancy when creating a dispersive volume, and the redundancy determines the maximum number of deletable blocks in the volume under normal data access. Figure 12 indicates the six-block DIGV.

In Figure 12, there are six available spaces of DIGV, and the information of File 1 is stored in all six blocks. As long as any block data are normal, File 1 can be read normally. Therefore, DIGV has greater fault tolerance than RGV and SGV.

2.5. Brief Introduction to the AFCS. The AFCS is an automated system for the whole process of rail transit, including ticketing, ticket checking, billing, charging, statistics, sorting, and management, based on computer, communication, network, automatic control, etc. The AFCS of a line often includes three levels, called three subsystems, i.e., the line center system, the station system, and the station terminal equipment. In urban transit, the ACC system is often employed to share information and clear ticket between the lines. Figure 13 provides the basic structure of the AFCS in a subway rail transit project.

3. Design of the Subway MLC System

3.1. Design of Shared MLC of AFCS. First, multiple computing nodes form a DFS cluster, and each computing node mounts and manages its own disk. Meanwhile, the DFS cluster provides remote access interfaces to the external and maps all disks into one or more data volumes. Moreover, a DFS driver is set up on the business server, which connects to the DFS server, mounts remote data volumes as local volumes, and provides standard POSIX interfaces for upper applications. The upper application employs the POSIX interface to read and write local files or read and write remote files, so it needs no change here. Figure 14 represents the logical view of applying DFS in the MLC system.

As illustrated in Figure 14, DFS is the core of all systematic intertransmission files. An application on a server writes files to a remote volume, which can be read in real-time by any other server with the same data volume. Like the relational database, NoSQL, object storage, or other storage middleware, DFS, as a common mechanism for storing and exchanging data files, is the infrastructure provided by the MLC system for business applications. Therefore, MLC should deploy a separate DFS cluster to provide differentiated services to individual business systems.

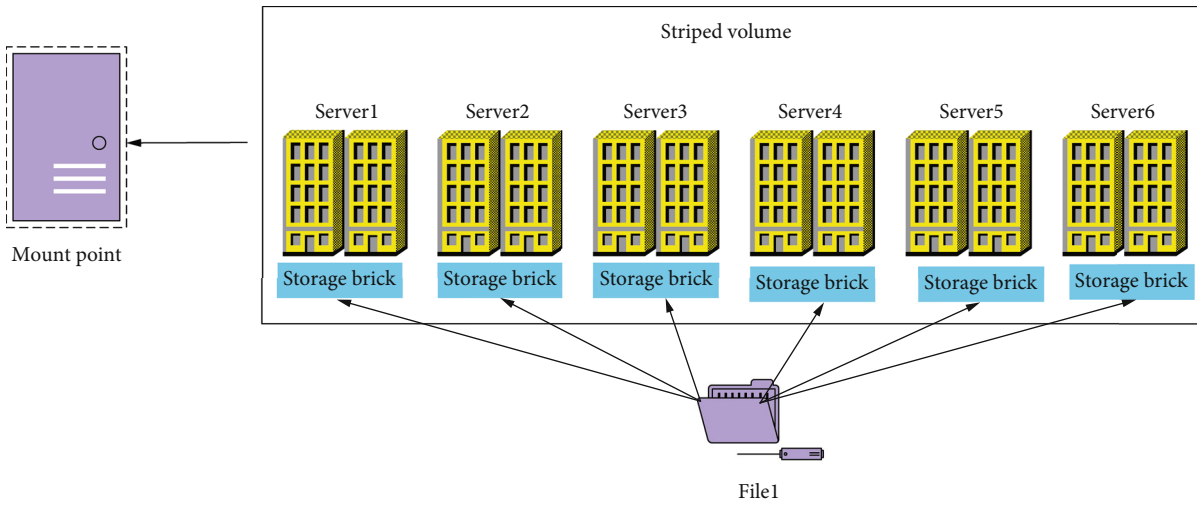

Figure 12: Framework of dispersive Gluster File System volume.

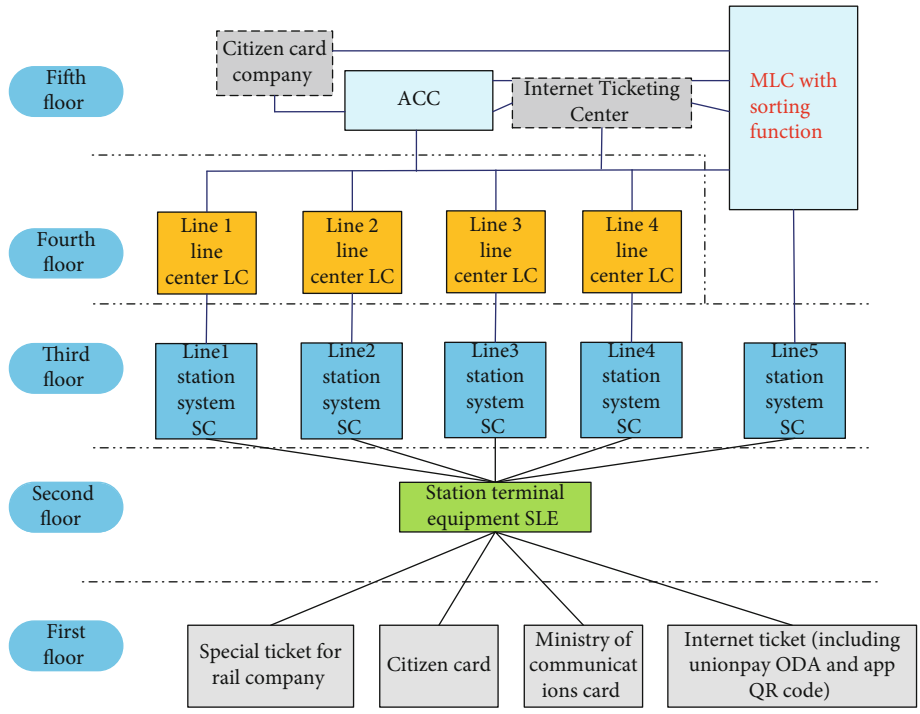

Figure 13: Systematic framework of the AFCS in a subway rail transit project.

Typically, DSF nodes mount local disks directly. DFS servers communicate with each other and automatically synchronize and backup data among nodes to ensure data security and high availability. However, according to the hardware resources of the project, DFS can also mount SAN equipment. At this time, DFS parameters can be adjusted to reduce the number of copies, and the data security is more dependent on SAN. DFS itself only ensures the high availability of computing nodes.

DFS is substantially a kind of technology with several implementation methods. After comparing CIFS, NFS, HDFS, Fast DFS, and other similar protocols or systems, the MLC system chooses GlusterFS or Ceph as the implementation of DFS. GlusterFS is relatively simple with low deployment and maintenance costs, but it is poor in low reading and writing small files. By contrast, Ceph has comprehensive functions and supports three access interfaces, namely, block device, file system, and object storage. Its access interfaces of the block device have been extensively tested in OpenStack. However, Ceph is complicated and requires high maintenance costs. In short, both Ceph and GlusterFS are mature open-source software, not subject to any technical, and Ceph has been transplanted to the domestic ARM server.

3.2. Security Design of Subway AFCS Based on GlusterFS. Figure 15 illustrates the security network architecture of the AFCS.

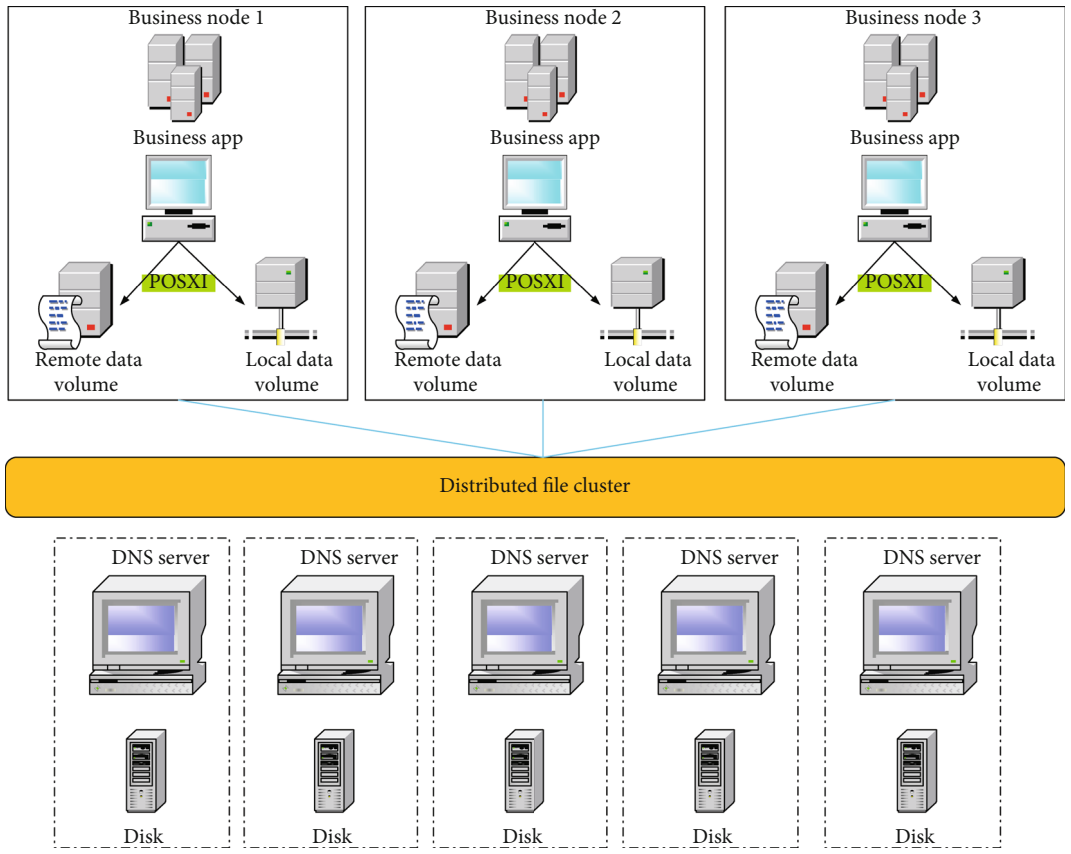

Figure 14: Logical view of applying DFS in the MLC system.

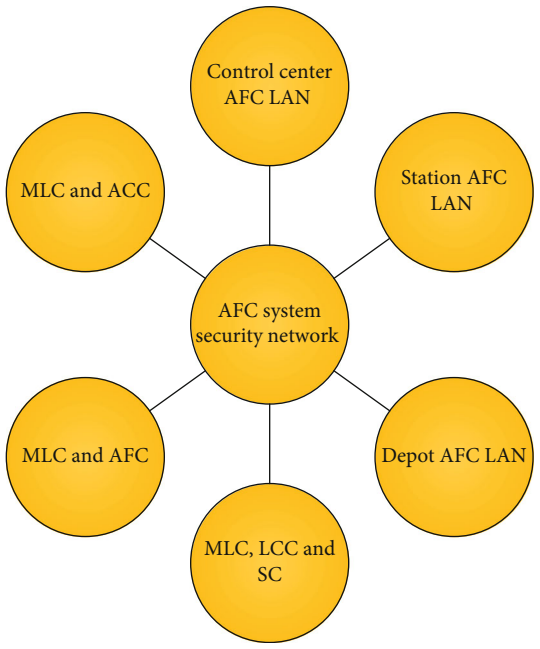

Figure 15: Security network architecture of the AFCS.

In Figure 15, the data transmission channels between the control center MLC and the depot and between MLC and the station are provided by the communication system. Additionally, the data transmission channels among the control center, depot, and the local area network (LAN) of the station AFCS are provided by the AFCS. The control center LAN is the Ethernet of 1,000 Mbps. The whole network adopts a switching-based network structure and considers necessary redundancy design. The station AFC LAN is the Ethernet with an industrial ring structure of 100 Mbps, and the recovery time of the switch ring network breakpoint is not more than 20 milliseconds.

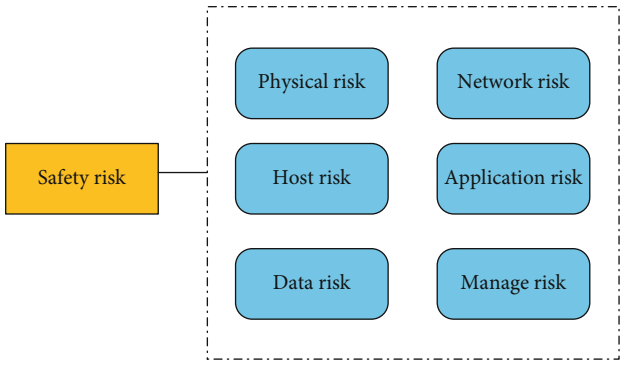

Figure 16: Security risks of the AFCS.

3.2.1. Security Risk Analysis (SRA) of the AFCS. The security risks of rail transit AFCS and network are generally reflected in the following six aspects, as shown in Figure 16.

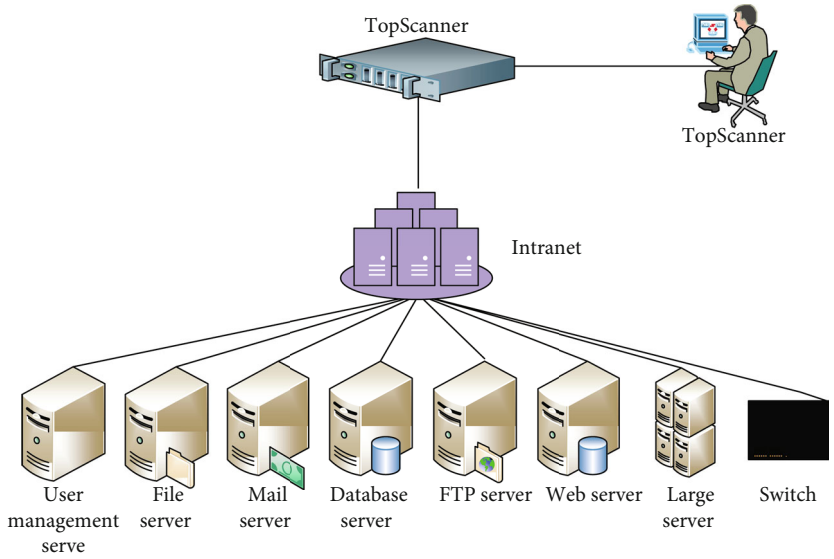

Figure 17: Independent deployment of the Web scanning system.

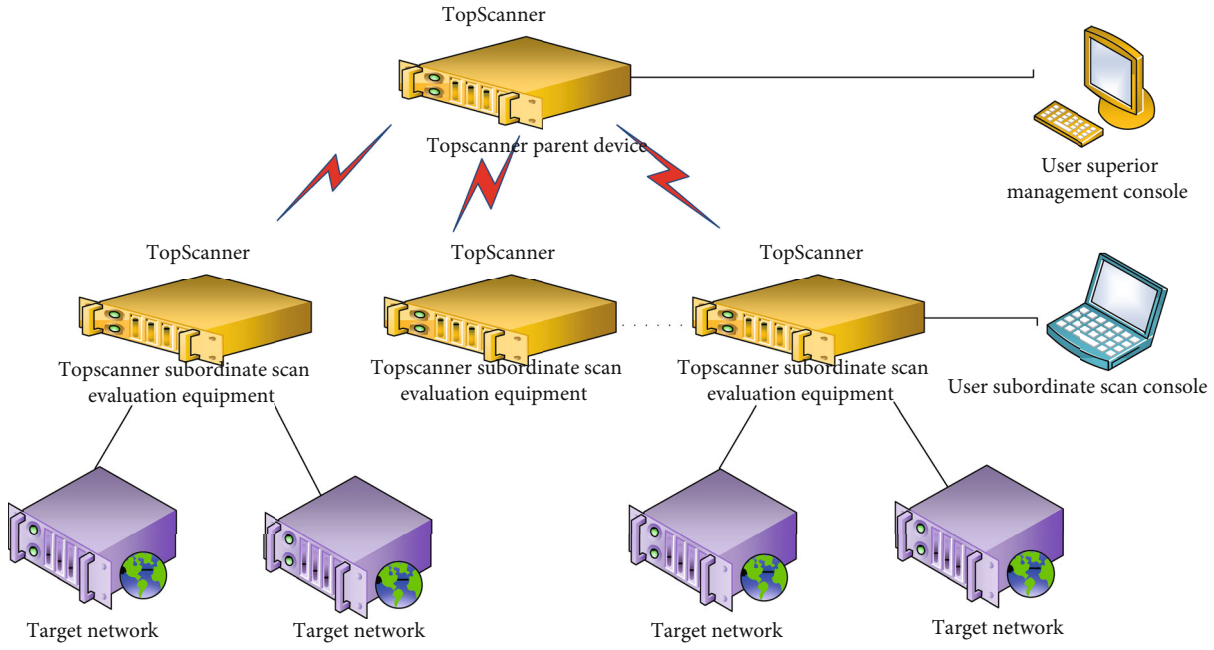

Figure 18: Distributed deployment.

Figure 16 demonstrates that there are six security risks for the AFCS. In this section, a Web scanning system is designed for data risks.

3.2.2. Web Scanning System for the Data Security of the Subway AFCS. A Web scanning system is designed to detect database vulnerabilities. The purpose is to check and evaluate the overall security of the database system, including the settings, the vulnerabilities, and the integrity of the database system. There are two kinds of deployment of the Web scanning system for application scenarios, and Figure 17 reveals an independent deployment.

Figure 17 implies that under independent deployment, only one TopScanner device is deployed in the network to connect to the network with the correct configuration, and then, it can be used normally. Its working range usually includes the entire network address of the user enterprise. The user can log in to the TopScanner system given any address and issue scanning inspection tasks. The address of the inspection task must be within the authorization range of the product and the assigned user. Figure 18 reveals the distributed deployment.

According to Figure 18, the distributed deployment employs several TopScanner systems to work jointly in the large-scale network, which can share and aggregate data among systems, facilitating users' unified management of distributed networks. TopScanner supports the two-level or multilevel distributed or layered deployment by users.

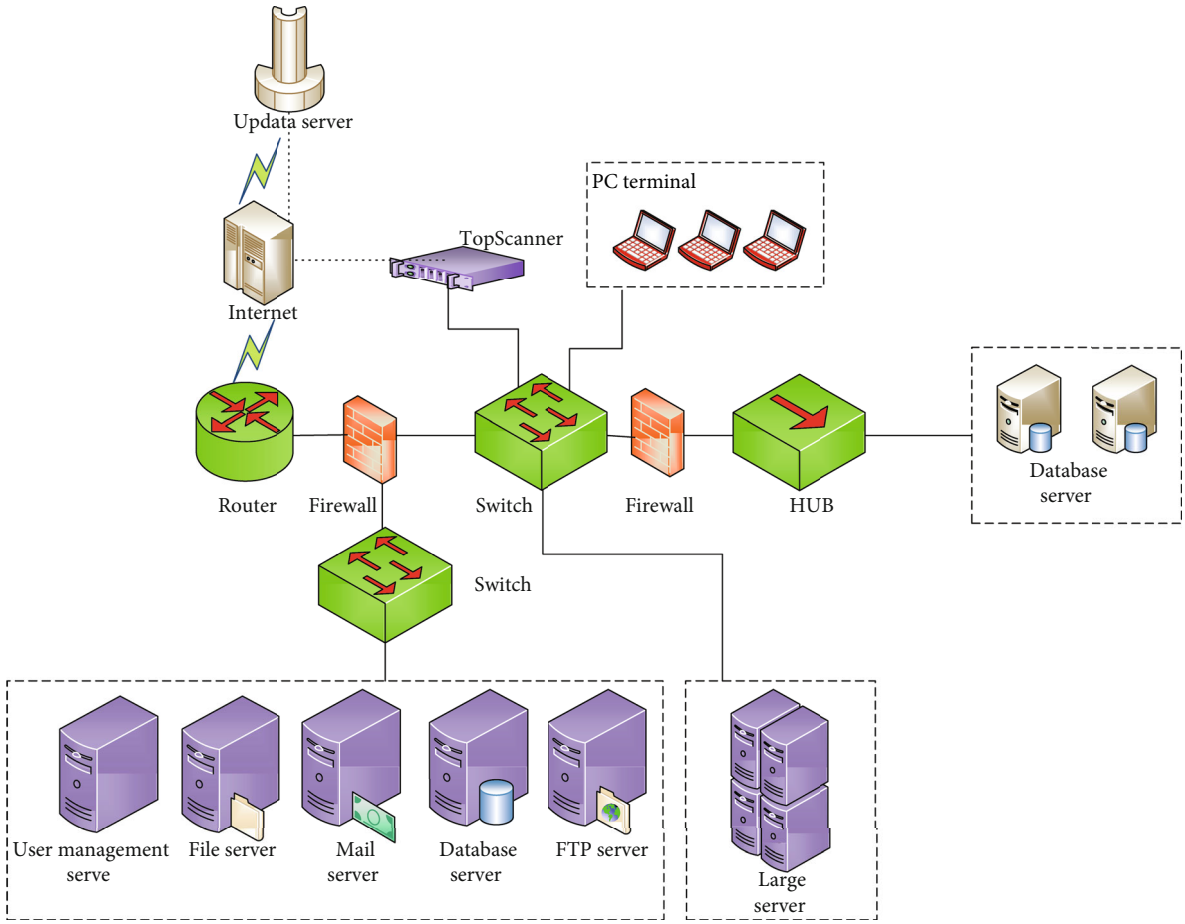

Figure 19: Web database scanning system.

Figure 19 provides the Web database scanning system designed based on the above deployment mode.

On the one hand, the Web scanning system can be deployed on the core switch for safety protection, and the system port addresses are distributed according to different networks. On the other hand, a distributed vulnerability scanning system can be configured alone in different networks to regularly detect the host in multiple different segments of the network and provide corresponding solutions. These solutions have effective protection effects.

3.3. Analysis of System Application Scenarios. The designed DFS is applied to three application scenarios of MLC, including the external data fire interface scenario, container cloud scenario, and primary-spare center data synchronization scenario.

3.3.1. External Data File Interface. Figure 20 is the logic diagram of the external data file interface scenario of MLC.

From Figure 20, the business of the external data file interface of MLC is as follows.

Business 1: MLC builds the FTP server, and the lines and stations upload transaction data, equipment audit data, and other information to the FTP server. MLC monitors the data files in the FTP server in real-time through DFS for analysis. At the same time, MLC writes the operating parameter files to the data volume mounted on the FTP server through DFS, and the station downloads the files through FTP.

Business 2: MLC builds the FTP server, and MLC regularly packs the transit card transaction as a file and stores it in the FTP server through DFS. The railway transit card company downloads the transaction file from FTP and returns the reconciliation file. MLC regularly scans the data volume mounted on FTP and obtains the reconciliation file.

In this scenario, the FTP server and MLC business server mount the same volume of DFS. The external system reads and writes volume files through the FTP interface, and the MLC system reads and writes volume files through the POSIX interface. DFS ensures data security and high availability. Moreover, the FTP server deploys multiple nodes, and multiple service nodes share the same Variable Information Processing (VIP). The FTP service nodes with VIP provide services to the outside world, and other nodes are in a standby state to ensure the high availability of the FTP server.

3.3.2. Persistent Volume (PV) of the K8S Cluster. Figure 21 displays the application scenarios of MLC in the container cloud.

MLC can also be deployed in the container cloud, which is generally managed by the K8S cluster. In the K8S cluster, each server is running on a Pod. After the Pod starts, it can be destroyed and reconstructed at any time. The

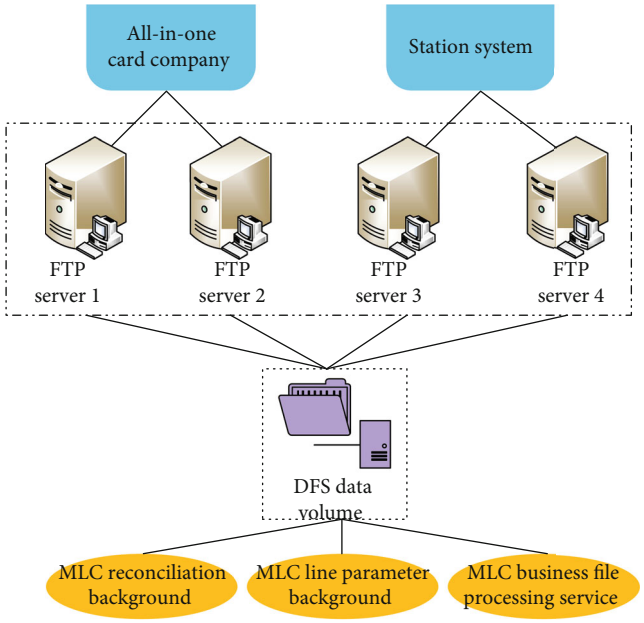

Figure 20: Architecture of the external data file interface of MLC.

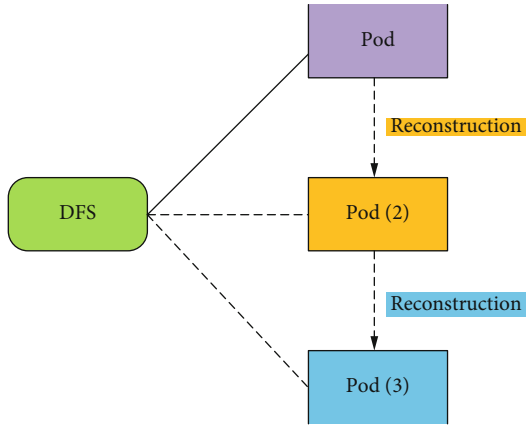

Figure 21: Application scenarios of MLC in the container cloud.

reconstructed Pod is completely new and does not save any state. However, the MLC system is developed from the existing system. In such a case, some servers are stateful, and the state is saved in files. Some middleware is also saved in local files. It is necessary to configure the PV in the K8S cluster through the shared volume service provided by DFS, to support this stateful server.

As shown in Figure 21, firstly, several volumes are allocated in advance in the DFS, and each volume has attributes, such as size. Then, all volumes and their attributes are declared in the form of PV in the K8S cluster. After each Pod is created, it applies for PV to the K8S cluster, and the K8S cluster assigns PV that meets the conditions to the Pod. The server in the Pod writes the state data into the volume corresponding to PV. Besides, the K8S cluster ensures that the allocated PV remains unchanged after the Pod is reconstructed so that the reconstructed Pod can read the previously saved data.

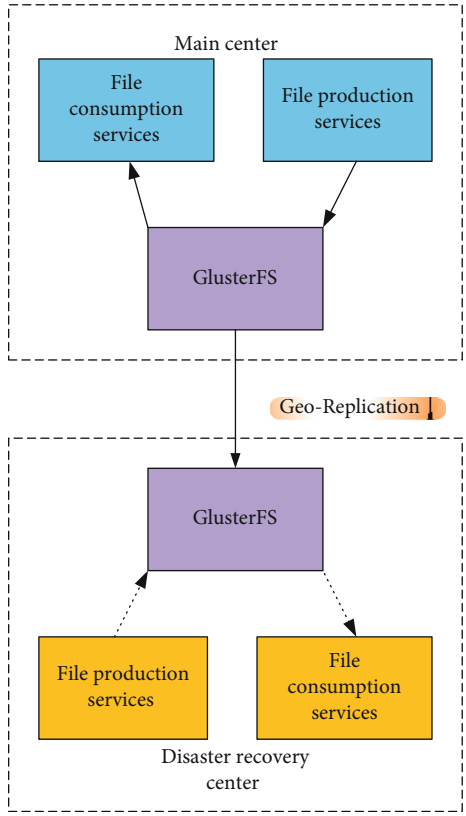

Figure 22: Geo-Replication function.

3.3.3. Primary-Spare Center Data Synchronization. The business files uploaded by the station are parsed by the MLC business processing module and moved to the local backup directory. Meanwhile, these business files are synchronized to the disaster recovery center in time to ensure that the data is not lost in extreme cases. After referring to DFS, the main center and disaster recovery center deploy their own DFS clusters. There are two methods to realize file synchronization between two centers. Figures 22 and 23 describe the main backup center data synchronization application scenarios.

Method 1: synchronizes by the file remote mirroring function of DFS, such as the Geo-Replication function of GlusterFS, as shown in Figure 22.

Method 2: the data volumes of two DFS clusters of the primary-spare center and the disaster recovery center are simultaneously mounted on the backup node of the disaster recovery center, and the folders are synchronized through the local file interface, as shown in Figure 23.

According to the above analysis, this MLC system has strong adaptability in different application scenarios, which proves that the designed scheme has high practical value.

3.3.4. Analysis of the Passenger Flow and Power Consumption of the System. This section verifies the data analysis ability of the proposed DFS-based subway AFCS from passenger flow and electric energy. Figure 24 indicates the statistical results of passenger flow.

According to statistical data in Figure 24, passenger flow increases from 8 : 00 to 20 : 00, decreases slightly at noon and afternoon, and peaks in the morning and evening. The

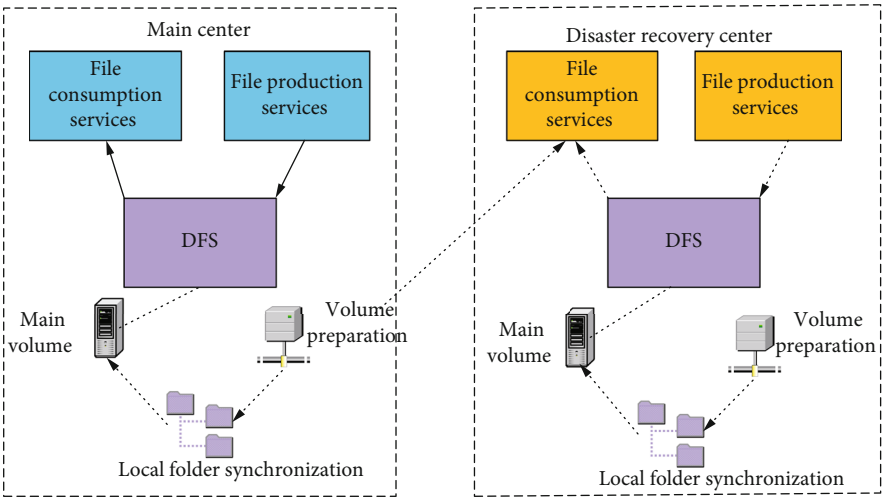

Figure 23: Data volumes in the DFS cluster.

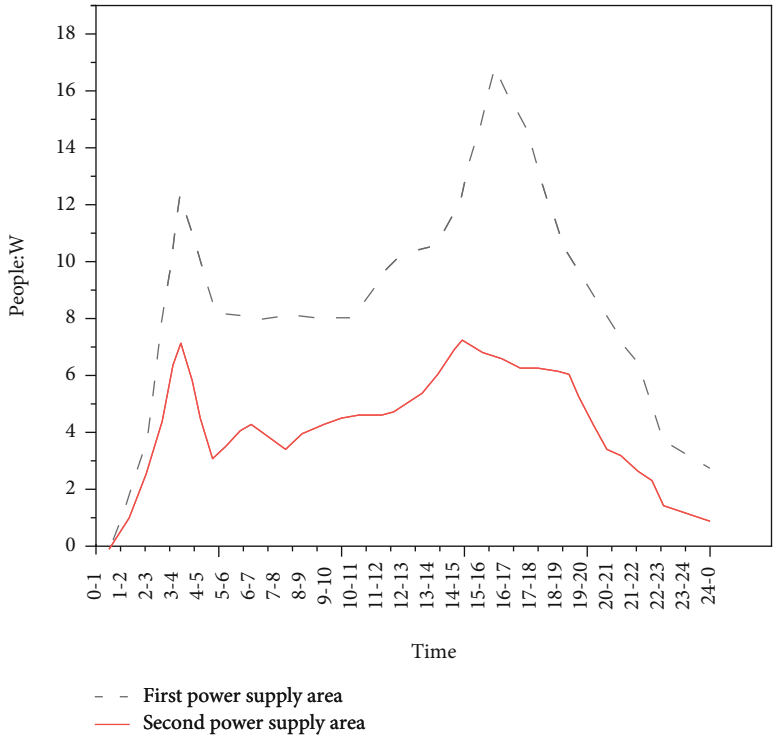

Figure 24: Statistical results of passenger flow.

passenger flow statistics results indicate that the passenger flow statistics of the proposed AFCS are in line with the actual situation, and the proposed AFCS can accurately predict the passenger flow of the first power supply area and the second power supply area. Figure 25 reveals the statistical results of power consumption.

As presented in Figure 25, the power consumption in the first power supply area is higher than that in the second power supply area. Moreover, the trend of consumption is consistent with the trend of passenger flow, and the power consumption increases sharply in the evening peak period. The comparative analysis of Figures 24 and 25 shows that the power consumption of the subway system is positively correlated with the passenger flow; besides, in the same power supply area, the time-variation of the power consumption and passenger flow of the subway system are consistent, which proves that the surge of power consumption of the subway system is caused by the change of passenger flow. The result proves that this AFCS can statistically analyze the subway data and provide accurate results, showing excellent feasibility and practicability. Table 1 displays the comparison between the actual and predicted passenger flow of the proposed subway AFCS during peak hours.

Table 1 demonstrates that the analysis forecasted errors of passenger flow in the first power supply area are 3.9%, 1.4%, and 3.4%, respectively, during three different periods.

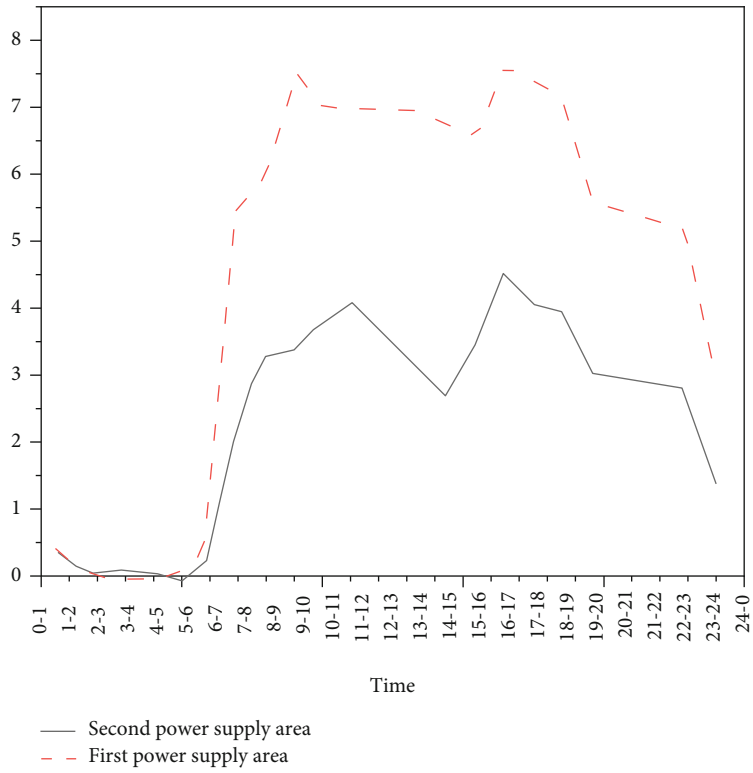

Figure 25: Statistical results of power consumption.

Table 1: Comparison between actual and predicted passenger flow of subway AFCS.

		8 : 00 AM -9 : 00 AM passenger flow (ten thousand)	11 : 00 AM-12 : 00 AM passenger flow (ten thousand)	14 : 00-15 : 00 passenger flow (ten thousand)
First power supply area	Analysis forecasted value	10.70	13.28	14.59
	Actual value	10.29	13.09	14.10
Second power supply area	Analysis forecasted value	4.61	5.38	6.91
	Actual value	4.21	5.08	6.61

The passenger flow analysis predicted errors of the second power supply area are 9.5%, 5.9%, and 4.5%, respectively, during three different periods, and the average error of the two power supply areas is 4.78%, indicating that the data analysis ability of the proposed subway AFCS is strong. Thus, the proposed AFCS can be used for subway information prediction and analysis.

4. Conclusion

Based on DFS, an MLC for subway AFCS is constructed, and a Web scanning system is designed for the data security of the subway AFCS. The conclusions are as follows.

(1) The subway AFCS based on GlusterFS can realize information management, data management, operation control, and ticket management within the jurisdiction of the subway

(2) The security system based on GlusterFS can maximally protest the whole system against attacks by the external network and prevent system data damage, which greatly improves the antirisk ability of the system

(3) The analysis results of subway passenger flow and power supply confirm the practical value of the system reported here

Data Availability

The labeled dataset used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no competing interests.

Authors' Contributions

Hui Fang designed technical route, performed the analyses, and wrote initial draft. Jiandi Jiang collected and managed the data and performed the analysis. Feng Lin reviewed and revised the manuscript. Wei Zhang provided direction selection, designed technical route, and edited manuscript.

Acknowledgments

This work was supported by the national key research and development program of China (2018yfb2101300).

References

[1] A. Salem, "Design, implementation and control of a SiC-based T5MLC induction drive system," *IEEE Access*, vol. 8, no. 99, pp. 82769–82780, 2020.

[2] P. Kurčík, M. Blatnický, J. Dižo, A. Pavlík, and J. Harušinec, "Design of a technical solution for a metro door system," *Transportation Research Procedia*, vol. 40, pp. 767–773, 2019.

[3] A. P. Rodrigues, R. Fernandes, and P. Vijaya, "Performance study on indexing and accessing of small file in Hadoop distributed file system," *Journal of Information & Knowledge Management*, vol. 20, no. 4, pp. 31–46, 2021.

[4] L. Tong, N. Liu, and S. Hu, "Study on key design parameters of subway source heat pump system with capillary exchanger," *Renewable Energy*, vol. 164, pp. 183–193, 2021.

[5] Q. Li, J. Loy-Benitez, S. K. Heo, S. Lee, H. Liu, and C. K. Yoo, "Flexible real-time ventilation design in a subway station accommodating the various outdoor PM_{10} air quality from climate change variation," *Building and Environment*, vol. 153, no. 4, pp. 77–90, 2019.

[6] J. Jiang, H. Nggar, and C. Xu, "Effect of ground motion characteristics on seismic fragility of the subway station," *Soil Dynamics and Earthquake Engineering*, vol. 143, no. 5, pp. 1169–1174, 2021.

[7] Q. J. Chen, T. Y. Zhang, N. Hong, and B. Huang, "Seismic performance of a subway station-tunnel junction structure: a shaking table investigation and numerical analysis," *KSCE Journal of Civil Engineering*, vol. 25, no. 5, pp. 1653–1669, 2021.

[8] P. A. Song, H. W. Wang, and X. X. Zhang, "An investigation on energy consumption of air conditioning system in Beijing subway stations," *Energy Procedia*, vol. 142, pp. 2568–2573, 2019.

[9] M. Méndez, "Operación Araña: reflections on how a performative intervention in Buenos Aires's subway system can help rethink feminist activism," *Estudos históricos*, vol. 33, no. 70, pp. 280–297, 2020.

[10] D. Gong and G. Li, "Research on multi-objective optimized target speed curve of subway operation based on ATO system," *International Core Journal of Engineering*, vol. 6, no. 2, pp. 133–137, 2020.

[11] A. S. Yadav, S. Agrawal, and D. S. Kushwaha, "Distributed ledger technology-based land transaction system with trusted nodes consensus mechanism," *Journal of King Saud University - Computer and Information Sciences*, 2021.

[12] H. E. Ciritoglu, T. Saber, and T. S. Buda, "Hadoop distributed file system," pp. 104–111, 2018.

[13] Y. Zhai, J. Tchaye-Kondi, and K. J. Lin, "Hadoop Perfect File: a fast and memory-efficient metadata access archive file to face small files problem in HDFS," *Computing*, vol. 156, pp. 119–130, 2021.

[14] J. Zou, A. Iyengar, and C. Jermaine, "Architecture of a distributed storage that combines file system, memory, and computation in a single layer," *The VLDB Journal*, vol. 29, no. 5, pp. 1049–1073, 2020.

[15] C. Chen, X. M. Liu, and Q. Li, "A rear-end collision risk evaluation and control scheme using a Bayesian network model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 264–284, 2019.

[16] C. Stokes, G. Masselink, M. Revie, T. Scott, D. Purves, and T. Walters, "Application of multiple linear regression and Bayesian belief network approaches to model life risk to beach users in the UK," *Ocean & Coastal Management*, vol. 139, no. 4, pp. 12–23, 2017.

[17] Y. Zhao, S. Hua, and X. Ren, "Relevance research of threat/error and undesired states in air traffic management based on Bayesian network model," *Journal of Air Transport Management*, vol. 60, no. 5, pp. 45–48, 2017.

[18] S. Gheisari and M. R. Meybodi, "A new reasoning and learning model for cognitive wireless sensor networks based on Bayesian networks and learning automata cooperation," *Computer Networks*, vol. 124, no. 12, pp. 11–26, 2017.

[19] L. Wu, "Student model construction of intelligent teaching system based on Bayesian network," *Personal and Ubiquitous Computing*, vol. 24, no. 3, pp. 419–428, 2020.

[20] A. Kab, B. Sl, and A. Ksp, "Detection of lung cancer by modified irregular tree structure Bayesian network model-based image segmentation," *Materials Today: Proceedings*, vol. 11, no. 3, pp. 1130–1138, 2021.

[21] V. S. Marichamy and V. Natarajan, "Big data performance analysis on a Hadoop Distributed File System based on geometric data perturbation technique," *Procedia Computer Science*, vol. 165, pp. 415–420, 2019.

[22] M. Alhowaidi, B. Ramamurthy, B. Bockelman, and D. Swanson, "Enhancing the SDTMA-NDN architecture for transferring the scientific data software using named data networking," *Computer Networks*, vol. 166, pp. 106954.1–106954.9, 2020.

[23] J. Liao, L. Li, H. Chen, and X. Liu, "Adaptive replica synchronization for distributed file systems," *IEEE Systems Journal*, vol. 9, no. 3, pp. 865–877, 2015.

[24] S. Lee, S. J. Hyun, H. Y. Kim, and Y. K. Kim, "Fair bandwidth allocating and strip-aware prefetching for concurrent read streams and striped RAIDs in distributed file systems," *Journal of Supercomputing*, vol. 74, no. 8, pp. 3904–3932, 2018.

[25] M. H. Cha, D. O. Kim, and H. Y. Kim, "Adaptive metadata rebalance in the exascale file system," *Journal of Supercomputing*, vol. 73, no. 4, pp. 1337–1359, 2020.

[26] X. Liu, Y. T. Lu, J. Yu, P. F. Wang, J. T. Wu, and Y. Lu, "ONFS: a hierarchical hybrid file system based on memory, SSD, and HDD for high performance computers," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 12, pp. 1940–1971, 2017.

[27] P. Bartus and E. Arzuaga, "GDedup: distributed file system level deduplication for genomic big data," in *2018 IEEE International Congress on Big Data (BigData Congress)*, vol. 7no. 10, pp. 120–127, San Francisco, CA, USA, 2020.

[28] Y. Chen, Z. Yi, and S. Taneja, "AHDFS: an erasure-coded data archival system for Hadoop clusters," *IEEE Transactions on*

Parallel & Distributed Systems, vol. 28, no. 11, pp. 3060–3073, 2019.

[29] M. W. U. Rahman, N. S. Islam, X. Lu, D. Shankar, and D. K. Panda, "MR-Advisor: a comprehensive tuning, profiling, and prediction tool for MapReduce execution frameworks on HPC clusters," *Journal of Parallel and Distributed Computing*, vol. 120, no. 10, pp. 237–250, 2018.

[30] L. Chen, S. Du, and C. Huang, "Design and implementation of teaching cloud platform based on the distributed file system," *Wireless Internet Technology*, vol. 5, no. 10, pp. 1–1, 2019.