A Method of Data Integrity Check and Repair in Big Data Storage Platform

Jiaxin Li[1(✉)], Yun Liu[1], Zhenjiang Zhang[1], and Han-Chieh Chao[2]

[1] School of Electronics and Information Engineering,
Beijing Jiaotong University, Beijing 100044, China
{17120080,liuyun,zhangzhenjiang}@bjtu.edu.cn
[2] Department of Electrical Engineering, National Dong Hwa University, Hualien 974, Taiwan
hcc@mail.ndhu.edu.tw

Abstract. In the big data storage platform, in order to ensure the security of user data, it is necessary to perform cyclic verification on the stored data and repair the damaged data in time. Considering the problems of low verification efficiency, low check frequency and low calibration accuracy of HDFS data integrity check, this paper proposed a new HDFS storage platform security check and repair scheme. The process can effectively reduce the amount of calculation and communication overhead, and can support the dynamic operation of the data. Experiments show that this method has certain advantages in security, scalability and flexibility.

Keywords: HDFS · Data integrity checking · Data recovery · Hash function

1 Introduction

With the rapid development of information technology and the increasing popularity of the network, the amount of data and information has grown rapidly, the world has entered the era of big data. In recent years, the Internet of things industry in China has developed rapidly. The product categories and total products of the Internet of things industry are increasing rapidly under the expansion of market demand, so the data that the Internet of things needs to process is also very large. In a big data environment, large amounts of data are stored remotely on the big data platform and distributed across different storage nodes [1]. In recent years, data corruption or loss on the storage platform caused by uncertainties such as software and hardware failures of large data platforms and user operations, has caused users to face great threats, causing frequent loss of events to users.

In order to overcome the possible damage and loss of data on big data platform, this paper designs a secure and reliable data integrity verification and repair method based on HDFS (Hadoop Distributed File System) distributed storage system, which provides an effective guarantee for the integrity of user data stored on big data platforms.

Y. Chen et al. (Eds.): BICT 2020, LNICST 329, pp. 183–188, 2020.
https://doi.org/10.1007/978-3-030-57115-3_15

2 Related Work

2.1 Data Integrity Verification

Data integrity is a basic requirement for data storage security. In the IoT cloud storage environment, users do not keep any copies of data locally, which makes it impossible to verify data in the cloud with traditional integrity verification methods. The traditional method of checking data integrity is access-based [2]. For example, a user calculates a hash value and signs it before storing it, and checks the integrity of the file by recalculating the hash value and verifying the signature when accessing the file. So, how to verify the integrity of the data in the cloud in real time and efficiently, and to take the necessary measures to recover the data when it is detected that the data is lost or damaged, is a very challenging problem.

At present, many data integrity verification algorithms are proposed. Liu et al. [3] reviewed the research work on cloud data integrity verification schemes, and summarized and compared representative cloud data integrity verification schemes. The existing data integrity verification algorithms still have some shortcomings: the verification information generally involves more complicated data structures and operations, and does not support dynamic update operations well; for the data recovery strategy, the amount of data that needs to be downloaded is too large, which increases the network communication overhead; the scheme of supporting fast location of faulty nodes can only achieve a limited number of integrity detections, and there is no scheme that can support the fast location of faulty nodes and achieve unlimited integrity detection.

2.2 HDFS-Based Cloud Storage System

Hadoop is a software framework for distributed processing of large amounts of data. Its core is Hadoop Distributed File System (HDFS) [4], which is the basis of data storage management in distributed computing. The HDFS cluster adopts a master/slave structure [5] model and consists of a Name Node and several DataNodes. It is deployed on inexpensive machines with high fault tolerance and high scalability.

At present, HDFS uses two verification methods [6] to ensure data integrity: checksum verification when data is read and written, and DataBlockScanner, a background process file block detection program run by DataNode, to periodically verify all file blocks stored on this data node. The CRC32 (A cyclic redundancy check 32) checksum of the data is calculated and written by the client when reading and writing data to the HDFS. HDFS calculates the checksum every fixed length, and the checksum is saved with the file block. For a large file, it takes a long time to divide it into small file blocks and then verify it. The calculation is inefficient and wastes storage resources. Since CRC32 is not considered for data security, it cannot detect whether the file block has been tampered with. The attacker can generate a fake file with the same check code through the original CRC32 check code. At this time, the file block is replaced but the CRC32 value is still the same and cannot be detected, and the data integrity and reliability check cannot be achieved. HDFS scans every 504 h. If file block corruption occurs during this time period, the system cannot detect it and the data integrity in the storage system cannot be guaranteed.

3 System Design and Implementation

3.1 System Architecture

In the HDFS distributed storage system, in order to implement security checksum repair for stored file blocks, this paper designs query module, verification module and repair module. The query module is configured to query storage information of file blocks in the system, and process user operations on data nodes and query requests of data nodes. The verification module is configured to periodically check the file blocks stored on the node to detect whether the file block is damaged. The repair module is used to process corrupted file blocks on the data nodes to implement file block repair.

The interaction between the modules is shown in Fig. 1. When the verification module finds an abnormal file block, the corresponding file block information is sent to the repair module, so that it starts the repair of the abnormal file block. At the same time, the file block information is also sent to the query module to query the distribution of the file blocks in the system, and the data is repaired by backing up the file blocks. After the repair module obtains the backup information of the abnormal file block through the query module, it communicates with the data node that stores the backup file block and the file block is complete, downloads the complete file block from the data node, and completes the repair of the file block.

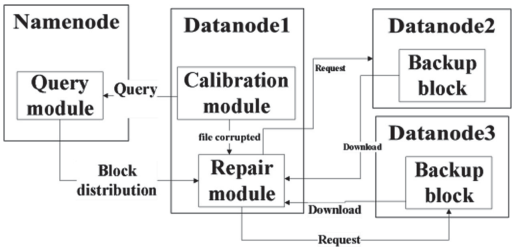

Fig. 1. Interaction diagram of system modules

3.2 Data Block Loop Check

In order to ensure that the data stored in the cloud is complete, the user must preprocess the data before uploading the file and generate verification metadata such as hash values, signature values, etc. that are used to check the integrity of the data in the later period The calculation of data is usually added to the location information of the data in the file. When the user performs a dynamic update operation, the change of location will cause the user to recalculate the verification metadata, which brings a lot of communication overhead and calculation cost. It is essential to verify that the data block is complete on the server side.

Each Datanode in the system stores data blocks. The data node performs integrity check on all stored data blocks at regular intervals, including the following steps: First, find a database and obtain data block information stored on all nodes in the database;

Then find a data block according to the obtained position information, and calculating a hash value of the data block at this time, if the data block file is unsuccessful to open, returning corresponding error information, and performing a repair operation; Finally, open the data block and get the hash value, compared with the original hash value stored in the database. If the values are consistent, the data block is intact and waits for the next check. Otherwise, the data block is damaged and needs to be repaired (Fig. 2).

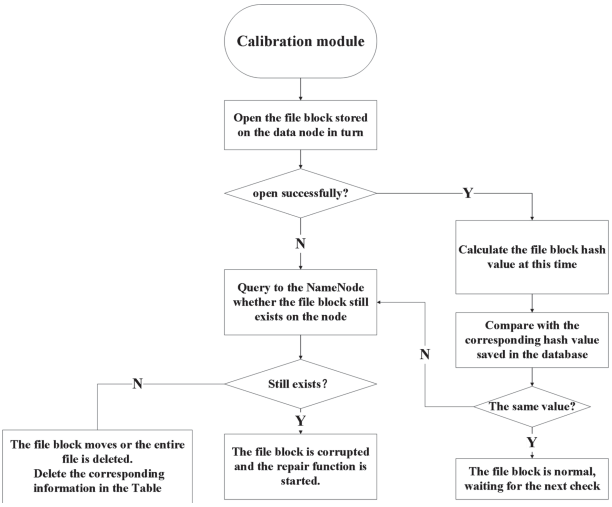

Fig. 2. Flow chart of cyclic check function

When the load of HDFS is unbalanced, the storage distribution of data on each node will be adjusted automatically. Therefore, before performing repair operations, the system needs to query whether the damaged data block still exists on this node. According to the current data block distribution information, the database is adjusted accordingly (Fig. 3).

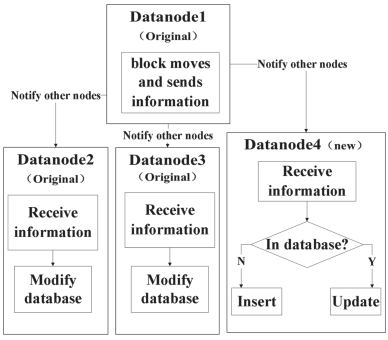

Fig. 3. Interaction graph between nodes

3.3 Data Block Repair

When the recalculated hash value is inconsistent with the corresponding hash value stored in the database, or when the hash value is calculated and the file block is no longer found, the check module may find that the file block is damaged, and sent the abnormal file block information to the repair module on the data node. Then the repair module repairs the damaged data block. The data file block repair function is implemented by the following steps: If the abnormal file block still exists on the data node, delete the damaged file block; The repair module sends the abnormal file block information to the query module on the NameNode, and queries the data node information of the backup file block storing the file block; Communicate with the datanode storing the backup file block, download the complete file block on the other node to the data node, and complete the file block repair.

Since each data node may have a file block corruption, the backup file block on the storage backup data node may also be incomplete, so further processing is required. The entire repair process is shown in Fig. 4.

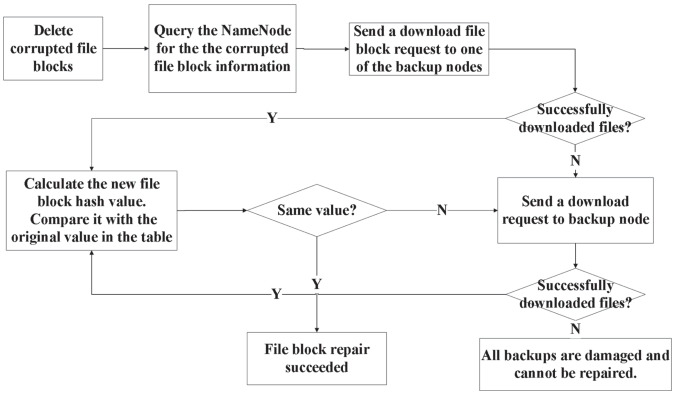

Fig. 4. Flow chart of cyclic check function

4 System Performance and Evaluation

The data integrity check and repair method designed in this paper has certain advantages in terms of security, scalability and flexibility compared with hdfs.

From the security perspective, the MD5 algorithm (Message-Digest Algorithm) is used to generate checksums for data blocks. The MD5 algorithm is a cryptographic algorithm. The data node can only generate a hash value by reading the file block in one direction, but cannot generate a corresponding file block according to the hash value. It is an irreversible algorithm, which can effectively prevent illegal users from tampering with the file block. In addition, in the process of data exchange between modules, the data is also encrypted, which also ensures the security of the system.

From the flexibility perspective, in the process of checking the data blocks, the datanode can adjust the check interval as needed. For files with high importance, users can narrow the check period of the check interval adjustment file. For general documents, the check interval can be appropriately increased to reduce the operating pressure of the system and improve the flexibility of verification.

From the aspect of scalability, the method proposed in this paper is relatively independent of the original HDFS system, and can be implemented without modifying the source code, and can independently deal with file changes such as file deletion and file block movement.

5 Conclusion

As time goes on, people will more and more realize the importance of data security in distributed storage. How to efficiently and securely store more and more big data is an urgent problem. This paper designs a data integrity verification and repair scheme for HDFS storage platform, which can guarantee the data integrity stored in the big data storage platform and support the dynamic operation of data. In the next step, we will study the security verification and repair scheme of the entire system, enabling the data holder to complete the data block verification on the client side. On the issue of distributed data query, more efficient data query algorithms and platforms need to be developed on the premise of ensuring data security. At the same time, in terms of data security strategies, adaptive and unsupervised anomaly detection algorithms can be developed to more effectively detect data anomalies.

Acknowledgment. This research was funded by the National Key Research and Development Program of China under Grant 2016QY06X1203 and the National Natural Science Foundation of China under Grant 61701019.

References

1. Wang, Y., et al.: Data integrity checking with reliable data transfer for secure cloud storage. Int. J. Web Grid Serv. **14**(1), 106–121 (2018)
2. Pardeshi, P.M., Tidke, B.: Improvement of data integrity and data dynamics for data storage security in cloud computing. In: Mandal, J.K., Satapathy, S.C., Sanyal, M.K., Sarkar, P.P., Mukhopadhyay, A. (eds.) Information Systems Design and Intelligent Applications. AISC, vol. 339, pp. 279–289. Springer, New Delhi (2015). https://doi.org/10.1007/978-81-322-2250-7_27
3. Liu, C., Yang, C., Zhang, X., et al.: External integrity verification for outsourced big data in cloud and IoT: a big picture. Future Gen. Comput. Syst. **49**, 58–67 (2015)
4. Maneas, S., Schroeder, B.: The evolution of the hadoop distributed file system. In: 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA). IEEE Computer Society (2018)
5. Dinsmore, T.W.: The Hadoop Ecosystem. Disruptive Analytics. Apress (2016)
6. Muthuram, R., Kousalya, G.: A survey on integrity verification and data auditing schemes for data verification in remote cloud servers. Electron. Gov. Int. J. **13**(1), 408–418 (2017)