*Applied Mathematics, Modeling and Computer Simulation*
*C. Chen (Ed.)*

*doi:10.3233/ATDE220098*

# Overview of Storage Architecture and Strategy of HDFS

Liwei TIAN[a,1], Yu SUN[b] and Lei YANG[a]
[a] *Guangdong University of science and technology, China*
[b] *Guangdong Innovative Technical College, China*

**Abstract.** At present, in the actual big data HDFS environment, because a large amount of data is stored locally, it is necessary to optimize the data security and transmission rate. At present, there are two main problems in HDFS. The background cluster is mainly divided into management process and business process. Therefore, in the actual use process, because the management process only has the main and standby dual machine backup, all the All the services of HDFS need to query the metadata through the namenode of the management end of HDFS, which leads to the emergence of hot spots, blocking the transmission link and increasing the load pressure of the equipment. In addition, the design concept of HDFS is that the underlying hardware is not reliable, which leads to the definition of three copy mechanism for data storage. The underlying device has already sacrificed part of the space (between 2 / 3-1 / 2) through raid data protection. When the data is protected through the triple copy mechanism, it will lead to too many data storage samples, which is not conducive to the actual storage space Use.This paper introduces the architecture and storage strategy of big data HDFS, and puts forward several solutions and ideas for the existing problems.

**Keywords.** HDFS, Big Data, Storage Strategy.

## 1. Introduction

HDFS (Hadoop Distribution File System) runs on general hardware (so-called universal hardware)Hardware means that the software has no requirements for the configuration and equipment of the underlying hardware platform, so it can be freely built and compatible[1-2]. For HDFS file system, including the overall Hadoop components, this can achieve perfect expansibility. Since the device itself has no requirements for hardware, then we can accumulate the hardware according to the infinite In this way, we can expand the performance until the requirements of big data processing system are met. This can reduce the cost in actual operation and provide better fault tolerance. If we have requirements for the model or performance of the equipment, we will inevitably use the same equipment for operation during the construction. In this way, if the equipment of a certain manufacturer has any Bug, then there will be the same problem on a batch of devices, resulting in overall performance degradation or system security crisis) distributed file system, which has high fault tolerance, high throughput and supports large file storage.HDFS mainly supports the stream data of large files, but it is weak for discrete small files, especially for delay

[1] Corresponding Author, Liwei TIAN, Guangdong University of science and technology, Dong guan, 523000, China;E-mail: 656453927@qq.com.

sensitive applications. Because HDFS needs to support high throughput, it is bound to sacrifice latency.

Since the file system of HDFS needs to load metadata in memory for maintenance, we also call this maintenance process namenode. The system needs to maintain about 150 for each data file Byte metadata, so storing small files and large files consumes the same metadata space. In this way, in terms of support, too many small files will affect the storage capacity of the final data. For the same metadata space, the larger the unit data we can store, the stronger the support of the large data file system will be HDFS in the same number of files, the cost of storing large files and small files is the same, so it is more reasonable to store large files. As the file system mainly used by big data, HDFS mainly provides file reading operation, so there is only one write process in the whole distributed process, and all other processes are read processes, and the write process is at the end of all processes. According to the processing characteristics of big data operating system, in order to protect data consistency and read-write performance, the designer proposed worm model as the overall system design goal of HDFS, worm write once read many, worm In the beginning, it was used in the storage system to protect the key data, such as the judgment files of the government and the court. These files can be read. However, due to the need to protect them from being tampered with, worm is needed to ensure that a file is written into the file system, and within the change period, We can also perform the rewriting operation. After entering the protection period, we can only read and cannot write. Here, any write operation cannot be performed. When the file size is 0 bytes, that is, the file is empty, then the file has an opportunity to append during the protection period. This is the characteristic of worm in storage. In HDFS, because our design goal is not to prevent file tampering, but to ensure efficient reading, we do not design worm very strictly. The file that we write is no longer allowed to be modified, but we can add and write infinitely at the end of the file[3].

## 2. HDFS Structure

HDFS is divided into three components: namenode,datanode,client[4].

(1) Namenode: used to store generated metadata and run an instance. The process is called into memory by HDFS. As the maintenance process of metadata, namenode carries the maintenance process of metadata in memory in order to improve the overall reading efficiency. However, the data in memory is easy to lose, so the metadata is usually maintained in datanode. After the system starts, the server will pull up the HDFS process, and then load the namenode into the memory, and then namenode will load the metadata image file into its own memory.

(2) Datanode: used to store the actual data. Each datanode will report the data block information maintained by itself to the namenode and run multiple instances. The default minimum storage space of HDFS is block, and the default size of each block is 128MB. Apart from maintaining data, datanode also needs to reserve some space for storing the metadata image file fsimage. If namenode and datanode are deployed together, the fsimage is on the datanode, which is equivalent to the storage medium of the server. If namenode and datanode are deployed separately, fsimage is stored on the server where namenode is deployed.

(3) Client: supports business access to HDFS, and obtains data from namenode and datanode, and returns it to users. Multiple services and instances run together. The client mentioned here does not refer to the actual user application, but the process of HDFS itself. Through this process, we can access HDFS, which is equivalent to that HDFS is a room. The client provides us with the access door. The interfaces provided by the client mainly include JDBC and ODBC interfaces.

In the active standby mode, the primary node provides services, and the standby node merges metadata and acts as the hot standby of the primary node. In order to protect the reliability of namenode and maintain the continuous operation of metadata and business, two processes are designed for protection. One process is used to provide normal business, and the other process is used as a standby process. However, the standby process is not cold standby, but is in hot standby state. Once the process fails, the standby process will be protected It can receive the message immediately and then switch the state. Since the standby process stores the fsimage metadata image file in the process of metadata persistence, the switching delay is very small. When switching, two file operations are involved, one is editlog and the other is fsimage. Editlog records the user's modification of metadata, and fsimage is the image of metadata.

The problem is that when HDFS is reading and writing massive data, if it is reading data, then relatively speaking, it can achieve efficient reading through distributed reading. However, before reading data, users need to contact namenode to read metadata through the distributed file system process. If you want to write, you need to contact the namenode to create metadata through the distributed file system, and then modify the metadata after the data is written. Therefore, the read will contact the namenode once, and the write operation will contact the namenode twice. However, the business model of China Merchants Bank is mainly for credit card related modules, so the user data is often written to small files frequently. At this time, as the external read-write agent process of HDFS, the client will frequently access the namenode process, but the namenode process itself has only one process, so it is easy to have business bottlenecks in high concurrency scenarios[5-6].

In addition, there is a switch function between namenodes. The main standby switch itself is protected by contacting the external zookeeper process through the zkfc process. The principle is as follows:

ZKFC (Zookeeper Failure Controller): used to control the active and standby state of namenode in case of failure. Fail over. The function of this process is to ensure that when the primary namenode fails, the service can be switched to the standby namenode for operation, so as to ensure the continuity of the business. Therefore, zkfc needs to detect the status of the primary and standby namenode in time, and report the heartbeat information to zookeeper in time. Therefore, zkfc process and namenode are There are as many processes and need to be deployed with namenode. The two main tasks of zkfc process are to obtain the heartbeat reported by namenode and perform the two operations of fail over. First of all, the zkfc process does not belong to zookeeper, but belongs to HDFS process. Zkfc process and namenode process are strongly coupled, so the two processes need to be deployed together. When namenode reports heartbeat to zookeeper, it is sent through zkfc. Therefore, we can understand that zkfc is a channel or a sending interface process. When zookeeper does not receive the heartbeat, it will issue the corresponding failure operation to zkfc. At this time, zkfc is responsible for controlling the standby namenode to take over the business.

JN(Journal node): used to share editlog files generated by namenode. The editlog file is the log file of HDFS operation. This information is not written to the metadata image file of fsimage. Therefore, we need to persist it to ensure that the overall metadata image can be loaded normally when the HDFS process is restarted.

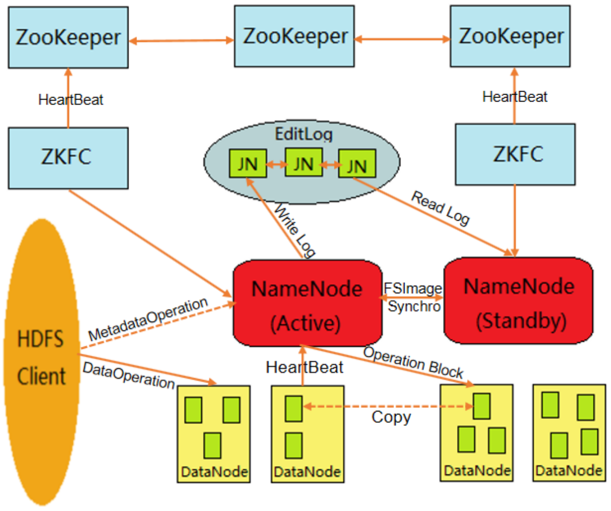

**Figure 1.** HDFS Structure

When the namenode executes the request of creating or moving a file submitted by HDFS, it will first record the editlog, and then update the file system image (in memory, since the operation of metadata image file recorded in editlog is in memory, the primary namenode will be in case of failure In this case, we need to recover the metadata according to the operations recorded in the editlog, and then we need to use the jn process to synchronize the editlog). The file system image in memory is used for namenode to provide services to the client, and editlog is used to provide services to the client It is used for the failure action in case of failure. Each operation recorded in the editlog is also called a transaction, and the transaction ID is used as the number. Therefore, the jn process has two functions: one is for metadata persistence; the other is to synchronize editlog when the primary namenode fails.

As shown in the figure 1, if a network attack occurs in the process, the attacker first cuts off the data synchronization process between the primary namenode and the standby namenode, and then the link between the primary namenode and the zkfc process. In this scenario, the zookeeper process will assume that the primary namenode has failed and replace the standby namenode with the active state, but it can be used as the client of HDFS It is detected that the primary namenode is still alive, so it will work normally. At this time, the primary namenode will be declared dead by zookeeper, but it is still working. The standby namenode will be set to the primary active state and work at the same time, which will lead to the inconsistency of data between the two. The metadata in the underlying data storage will be stored on two devices respectively to realize the data instability and metadata rewriting(It can be compared to the scenario that the database does not comply with the acid attribute, resulting in a read uncommitted scenario). Therefore, how to protect the security of data transmission between processes at this time, and how to solve the problem of low efficiency when single namenode works also needs to be solved.

## 3. HDFS Storage Strategy

HDFS namenode automatically selects datanode to save a copy of the data. In the actual business, there are the following scenarios[7-10]:

There are different storage devices on datanode, so it is necessary to select a suitable storage device to store data hierarchically. The importance of data in different directories of datanode is different. Data needs to be saved by selecting an appropriate datanode node according to the directory label. The data node cluster uses heterogeneous servers, and the key data needs to be saved in node groups with high reliability.

Data storage strategy can be widely used in all kinds of business, just like hierarchical storage used in storage We can also provide a high-level storage strategy for data assurance of different businesses. For example, as a video website, if a new TV series is put on the shelves, the access traffic will be large. At this time, the data can be put on the memory virtual hard disk or SSD After the end of the TV series, the number of visits will gradually decrease. At this time, the number of visits will gradually decrease. We can put the data in the SAS hard disk. As time goes on, when the access volume is very small, we can put the data on the SATA hard disk or archive it. The HDFS strategy is similar to the above storage strategy, but HDFS storage data involves few migration according to the access volume. It is mainly related to the data storage operation at the beginning. For example, the data of key business can be placed in the medium with fast access and high reliability, and normal business can provide a normal protection strategy.

### *3.1. Label Storage*

Users configure HDFS data block storage policy through data characteristics, that is to set a label expression for an HDFS directory, and each datanode can correspond to one or more tags; when the label based data block storage policy selects a datanode node for storing files in the specified directory, the datanode to be stored is selected according to the label expression of the file Node range, and then within the scope of this datanode node, the next specified data block storage policy is followed for storage.

The detailed explanation here can be understood as follows: we label the metadata, which is controlled by the namenode when allocating the write space. According to the data writing operation, we can find that no matter what the write operation is, the first step must be to access the metadata before writing. Then, we will print the metadata The label means that when the metadata allocates the write space, the write location of the data has been controlled. In fact, it is equivalent to that we have made the corresponding storage strategy for the data before the data is written, so as to ensure that the data is written to the corresponding media,as shown in figure 2.

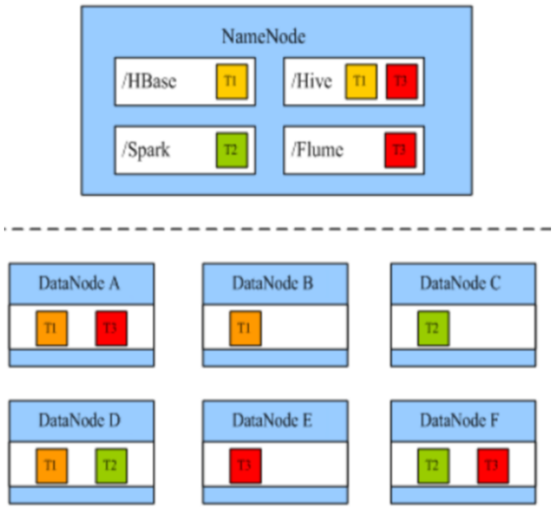

**Figure 2.** Schematic diagram of label storage

## *3.2. Node Group Storage*

Configure datanode to use node group storage: key data is saved in nodes with high reliability according to the actual business needs. At this time, datanode forms a heterogeneous cluster. By modifying the storage policy of datanode, the system can force the data to be saved in the specified node group.

The biggest differences between node group storage and label storage are as follows:

(1) Node group storage is performed by datanode, and label storage is done by namenode.

(2) The role of node group storage is replica data. The source of control is the first data. The object of tag storage is to write the first data, and the source of control is the directory label in metadata.

(3) Node group storage ensures the reliability of data, and label storage not only ensures the reliability of data, but also security and availability. Therefore, the control range of label storage is larger than that of node group storage.

Usage constraints: before using constraints, we need to configure constraints. We need to specify mandatory rack groups for data copies.

The first copy is selected from the forced rack group (rack group 2), and if there are no available nodes in the forced rack group, the write fails. Therefore, the first copy is compulsorily protected, and the successful writing must be guaranteed.

The second copy will be taken from a random node in the local client machine or rack group (when the client machine rack groupNot mandatory rack group).

The forced rack group only stores one copy of data. Therefore, when a node needs to create a replica, we first need to write the data to the forced rack group. When the data is written to the second replica, we need to check whether the rack group to which the node belongs is a mandatory rack group. If so, according to the mandatory policy of forcing the rack group to store only one copy, we will Data can no longer be written to this rack group. At this time, you need to select a random rack group to write. If the rack group to which the node belongs is not a mandatory rack group, then write normally.

The first data force rack group,second data - local device or local rack group,a third copy will be selected from the other rack groups. Each copy should be stored in a different rack group.If the number of copies required is greater than the number of available rack groups, the extra copies are stored in a random rack group,as shown in figure 3.

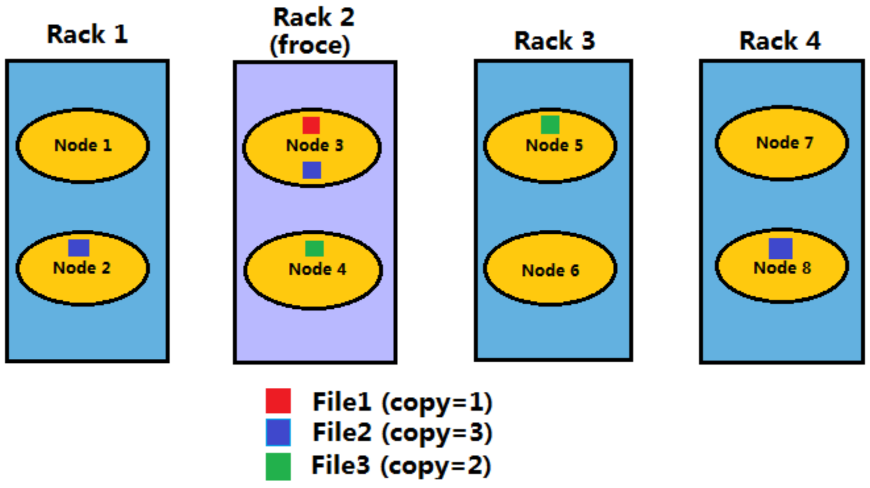

**Figure 3.** Storage diagram of node group

## *3.3. Hierarchical Storage*

Hierarchical storage is mainly for the purpose of providing high performance and high availability as mentioned above, and the specific significance will not be repeated. Configure datanode to use hierarchical storage: HDFS's heterogeneous hierarchical storage framework provides ram_ There are four types of storage devices: disk (memory virtual hard disk), disk (mechanical hard disk), archive (high density and low-cost storage medium), SSD (solid state disk).

Through the reasonable combination of the four storage types, the storage strategy suitable for different scenarios can be formed,as shown in table 1.

**Table 1.** Storage strategies for different scenarios

| Strategy ID | Name | Block placement location | Alternative Storage Strategy | Alternative Storage Strategies for replicas |
|---|---|---|---|---|
| 15 | LAZY_PERSIST | RAM_DISK:1, DISK:n-1 | DISK | DISK |
| 12 | All_SSD | SSD:n | DISK | DISK |
| 10 | ONE_SSD | SSD:1, DISK:n-1 | SSD, DISK | SSD, DISK |
| 7 | HOT（default） | DISK:n | <none> | ARCHIVE |
| 5 | WARM | DISK:1, ARCHIVE:n-1 | ARCHIVE, DISK | ARCHIVE, DISK |
| 2 | COLD | ARCHIVE:n | <none> | <none> |

For example, policy 10 has a policy ID of 10 and a name of one_ The placement mechanism of SSD and block is: the first data is stored in SSD, and the second and subsequent data are stored in hard disk If there is a problem with the media specified in the placement policy and it can't be written normally, then we need to use an alternative storage strategy to store it. According to the example, if there is a problem with the main policy, the SSD is preferred for the first copy. If there is no SSD, it will be stored on disk In the mechanical hard disk, the alternative storage strategy of the replica specifies the storage location of the replica data, and the meaning is consistent with the alternative storage strategy. To sum up, so block The placement strategy specifies the storage strategy of the data under normal conditions. If there is a problem, the alternate storage strategy specifies the storage strategy of the first data, and the alternate storage strategy of the replica specifies the storage strategy of the replica. If none occurs between the alternative storage strategy and the backup storage strategy, the storage strategy of the replica is specified If the main policy cannot be written, a write failure will be returned directly.

## 4. Solution Ideas

By introducing namenode cluster, the problem of inefficient access is first solved. Because namenode must ensure that the data must be consistent. As the customer business of China Merchants Bank is mainly based on frequent business data writing, which is in conflict with the idea of HDFS itself (the default design scenario of HDFS is to read massive large files and write a small amount). Therefore, in the design of HDFS, first of all, every device process in the namenode cluster should have the write function, but turning on too many write functions will lead to data Inconsistency occurs, so it is necessary to ensure that each device has the ability to write data through multi namenode blockchain technology. In addition, blockchain technology can also ensure the security of data writing and process.

We need to solve the above two problems separately. In this process, the first thing to solve is the read-write bottleneck problem of namenode. However, due to the conflict between the HDFS framework design and the business model of China Merchants Bank, we need to improve the HDFS framework to achieve high efficiency in small file high concurrency scenarios Read and write. The method that can be adopted here can add a new idea framework, which is to design a multi form access distributed framework. Through DHT related technology, each client can have the ability to store and read and write data. In addition, a new central site is set up, but the site is only responsible for the maintenance and authentication of each member, and does not do the business of relevant data.

DHT (distributed hash table) is similar to tracker's network which returns seed information according to seed signature. The full name of DHT is distributed hash table, which is a distributed storage method. In the case of no server, each client is responsible for a small range of routing, and responsible for storing a small part of data, so as to achieve the addressing and storage of the entire DHT network. The new version of BitComet allows you to connect to both the DHT network and the tracker. That is to say, it can be downloaded without connecting to the tracker server at all, because it can find other users who download the same file in the DHT network.

The first solution to the security problem of the process is to ensure the security of the process. That is, network messages will not be hijacked and tampered by hackers.

In terms of network, the number of server nodes of China Merchants Bank has exceeded 300, so the internal connection is made by using three-layer routing network. At this time, the first thing to ensure is the security of data transmission. Due to the use of encryption algorithm and other related measures in the internal cluster, it is easy to increase the pressure of nodes, which is not conducive to efficient forwarding. Therefore, it is suggested that the research on credit and message encryption between intranet server networks should not be conducted. The efficiency of message transmission should be improved as much as possible, and the security guarantee of accessing the intranet from the external network should be turned to. In addition to the conventional encryption of data transmission, it can be improved as much as possible Enhance the security of user and authority authentication. For example, a more efficient and reasonable encryption algorithm is used to implement[11-12].

## 5. Summary

This paper analyzes the architecture of HDFS, introduces the storage strategy of HDFS, and finds three problems.The namenode of HDFS has problems in actual security and utilization due to the influence of the active and standby processes.The switch between the main and standby processes of namenode and the data transmission between zookeeper are not protected by security, so they are vulnerable to attack.There are too many copies of datanode data storage, which is not conducive to the increase of storage utilization.Proposed solutions: through the introduction of namenode cluster to solve the problem of inefficient access; through DHT related technology, each client can have the ability to store and read and write data, and set up a new central site, but the site is only responsible for the maintenance and authentication of each member, and does not do related data business; more efficient and reasonable encryption is adopted The algorithm ensures security.

## Acknowledgement

The paper is funded by Dongguan social science and technology development (general) project (No.2020507154645) and Special project for key fields of colleges and Universities in Guangdong Province (new generation information technology): Research on cloud Storage mechanism based on blockchain technology; Cultivation Plan for Major Scientific Research Achievements of Universities: Research on the Blockchain Fragmentation Mechanism Based on Merkel Tree (GKY-2019CQYJ-13).

## References

[1] Attebury G, Baranovski A and Bloom K, “Hadoop distributed file system for the Grid”, Nuclear Science Symposium Conference Record,IEEE, 2009.

[2] Song A, Tu J and Zhao J, “Memory-Based Data Storing Technologies on Hadoop Distribution File System”in 2015 Third International Conference on Advanced Cloud and Big Data (CBD), IEEE, 2015.

[3] Fahmy M M, Elghandour I and Nagi M, “CoS-HDFS: Co-Locating Geo-Distributed Spatial Data in Hadoop Distributed File System”, in IEEE/ACM International Conference on Big Data Computing Applications & Technologies, ACM, 2017.

[4] Farag A, “Towards a scalable HDFS architecture” in International Conference on Collaboration Technologies & Systems, IEEE, 2013.
[5] Luo Y, Luo S and Guan J, “A RAMCloud Storage System based on HDFS: Architecture, implementation and evaluation”, Journal of Systems & Software, vol.86, no.3, p.744-750,2013.
[6] Li Z and Shen H, “Designing a Hybrid Scale-Up/Out Hadoop Architecture Based on Performance Measurements for High Application Performance”in International Conference on Parallel Processing, IEEE Computer Society, p.21-30, 2015.
[7] Song B Y, Wang J L and Wang Y, “Optimized storage strategy research of HDFS based on vandermonde code”, Chinese Journal of Computers, 2015.
[8] Zhao W T, Ding Y and Zhang X H, “Available Storage Space Sensitive Replica Placement Strategy of HDFS”, Applied Mechanics & Materials, p.644-650.
[9] Cao and Hui, “Research on the Optimization of the Storage Strategy of HDFS ”, Advanced Materials Research, p.989-994, 2014.
[10] Yuxin Guan, Ma Z and Li L, “HDFS Optimization Strategy Based On Hierarchical Storage of Hot and Cold Data”, Procedia CIRP, 2019.
[11] Shen Q, Yang Y and Wu Z, “SAPSC: Security Architecture of Private Storage Cloud Based on HDFS”, in 2012 26th International Conference on Advanced Information Networking and Applications Workshops. IEEE, 2012.
[12] Sui L, Jian H and Wen-Chao J, “A Secure Cloud Storage Model Based on HDFS”, Journal of Guangdong University of Technology, 2014.