Research on coal mine safety monitoring data storage based on HBase

Lizheng Jiang, Yanbin Li
School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China
liyanbin@eplian.com, li-yanbin@ncepu.edu.cn

Abstract. In order to solve the problem of low storage efficiency and slow query speed when massive coal mine safety monitoring data is stored in relational database, a method of storage and retrieval based on HBase was designed. By combining the hash value of the combination of coal mine code, substation code and transducer code as well as the timestamp after inversion, the hot spot and data skew caused by the unreasonable design of rowkey in HBase were avoided. In order to improve the efficiency of data storage, the swing door trending (SDT) algorithm was used to compress the monitoring data, and in order to improve the efficiency of data retrieval, a secondary index based on ElasticSearch was constructed. The experimental results on the real coal mine data show that the efficiency of data storage and retrieval has been significantly improved.

1. Introduction

With the rapid development of the Internet of Things technology and the country's great emphasis on production safety, coal mines are gradually realizing safety monitoring automation. Due to the limitation of storage technology, the security monitoring system rarely stores historical data, only has the function of real-time monitoring, and the value of the data has not been fully utilized. Through the mining of massive safety monitoring historical data, coal mine risks and hidden dangers can be analyzed and predicted, and potential safety problems can be discovered, which is of great significance for improving coal mine production safety. The storage of massive data is a prerequisite for data mining, and it is also a major problem currently facing[1].

Coal mine monitoring data is a typical time series data, and the large amount of data is its important feature. Traditional relational databases have performance bottlenecks when storing this type of data due to poor scalability, and data query has a high latency, which cannot satisfy real-time data storage and query[2]. As a highly reliable and high-performance distributed storage system, HBase provides a possibility for the storage of massive amounts of data. It has been applied in big data scenarios such as smart medical care[3]. According to the time series characteristics of coal mine monitoring data and the storage principle of HBase, rowkey and data compression were designed in this paper, and a secondary index based on ElasticSearch was designed, which can effectively improve the storage and retrieval efficiency of coal mine monitoring data.
2. Data Storage Mode Design

2.1. Table Structure Design
In order to ensure the uniformity of data formats and facilitate reporting to regulatory agencies, the data of coal mine safety monitoring systems generally meets relevant standards. The safety monitoring data used in this paper conforms to local standards in Shanxi Province. The data are stored in files, including transducer files and monitoring data files. Therefore, two tables are designed for storage, one is the transducer information table `transducer`, and the other is the monitoring data table `data`.

Because the transducer information data is relatively small, the row key of `transducer` is designed as a combination of coal mine code, substation code, and transducer code, which is convenient for quickly searching transducer information through the transducer code. `Transducer` contains a column family `info`, and the columns in `info` contain other information about the transducer.

The amount of monitoring data is very large, so `data` needs to be designed reasonably to avoid data tilt and hot issues. Rowkey design of `data` will be introduced in detail in section 2.2. `Data` contains a column family `i`, and the columns in `i` contain transducer status code `s` and transducer monitoring value `v`.

2.2. Table Structure Design
Rowkey design of `data` is very important. Rowkey information includes mine code, substation code, transducer code, monitoring time, etc. Rowkey should be as short as possible to reduce storage space and transmission time during query, and the characteristics of the operating system should be used to improve query performance. So the length of rowkey designed in this paper is 16 bytes. The first 6 bytes of rowkey use the high byte 6 bits of the MD5 hash value after the coal mine code, substation code, and transducer code are combined. On the one hand, this allows different transducers to correspond to different row key prefixes, avoiding hot issues caused by data being written to the same region; On the other hand, this can ensure that when obtaining the value of a certain transducer, a fixed row key is constructed for fast retrieval, and a full table scan is avoided to reduce retrieval efficiency. The last 10 bytes of rowkey are obtained by reversing the timestamp, that is, the timestamp of the monitoring time is subtracted from the largest 10 digits. Because HBase stores in ascending lexicographical order of rowkey characters, if you use the date directly, the most recent data will always be at the end. Generally, the frequency of querying recent data is much higher than the frequency of querying data for a long time in the past. This can bring a better query experience.

3. Data Compression Design

3.1. Swing Door Trending Algorithm
The swing door trending (SDT) algorithm is a lossy compression algorithm. It uses line segment fitting to compress data. The essence is to replace the middle data point with a connection between two points to reduce data storage[4]. As time series data has the characteristics of large data volume and high data proximity, using the revolving door compression algorithm to compress time series data can obtain a higher compression ratio.
The principle of SDT is to connect the most recently saved data point with the current data point to determine whether the resulting compression deviation zone can cover all points between the two, if not, save the data point just before the current time, Otherwise it will not be saved[5]. As shown in Fig.1, the "gate" formed by the current data point and the recently saved data point cannot contain all the data points in between. Therefore, the previous data point of the current data point is saved, and the remaining data points are not saved, which reduces stored data.

\[ \Delta E \]

Let \( \Delta E \) be the compression accuracy parameter of SDT, the compression process is: \( t_0 \) is the last stored point, and the upper and lower points with a distance of \( \Delta E \) to \( t_0 \) are used as fulcrums to establish two virtual doors. The door is closed when there is only one data point. As data points increase, the door will rotate to open, and once opened, it will not close; as long as the sum of the interior angles of the two doors is less than 180 degrees, the rotation operation can continue. When the sum of the internal angles of the two doors is greater than or equal to 180 degrees, the operation is stopped. The previous data point at the current moment is stored, and a new segment of compression is started from this point. As shown in Fig.2, after compression, the data points from \( t_0 \) to \( t_4 \) are replaced by straight lines from \( t_0 \) to \( t_4 \) in compression section 1; the data points from \( t_4 \) to \( t_7 \) are replaced from straight lines from \( t_4 \) to \( t_7 \) in compression section 2.

3.2. Monitoring Data Compression

Through the analysis of the principle of SDT, it is known that for compression, \( \Delta E \) must be selected first, and a suitable parameter \( \Delta E \) can ensure a better compression effect. Since various types of transducers generally have different ranges, and the monitoring data distribution of each transducer is also quite different. In order to get a better compression effect, the normal data of each transducer
according to the actual data was tested in this paper, and a more appropriate $\Delta E$ was selected, so a column was added to transducer to store $\Delta E$.

Status of transducer is divided into many types, including normal, alarm, power failure, over-range, adjustment, failure, disconnection, etc. Therefore, the monitoring data of abnormal status is not compressed but directly stored in the database to ensure the accuracy of abnormal data.

4. Secondary Index Design

Compared with traditional relational databases, HBase has a series of advantages, but it also has disadvantages. For example, HBase only supports fast query through rowkey and does not support secondary index. Therefore, if it is other complex query conditions, it is necessary to construct a filter for full table scan query, which greatly reduces the efficiency of data query. In order to improve the efficiency of data query, many secondary index schemes have been proposed, such as the hindex scheme based on HBase’s own Coprocessor, the SIHBase scheme based on Solr, etc[6]. A secondary index scheme based on ElasticSearch was used in this paper.

The design of the secondary index in this paper is to store the fields to be queried in ElasticSearch together with the rowkey. In actual use, fuzzy queries are generally performed on transducer status and transducer name, so transducer status and transducer name are stored in ElasticSearch together with rowkey and indexed. When searching by transducer code, directly construct the rowkey to query through HBase; if query by other information such as transducer name or status, first use ElasticSearch to query the corresponding rowkey, and then query the corresponding data from HBase through rowkey.

5. Experiment and Result Analysis

5.1. Experimental Environment

A Hadoop cluster with 3 nodes was built: 1 NameNode and 2 DataNode. HBase was configured with 1 Master and 2 RegionServer. The ElasticSearch cluster also used a 3-node configuration. The cluster was built on a server with 16GB of memory and 1TB of storage space through virtual machines, and each node was allocated 4GB of memory and 100GB of storage space. The data used in this experiment is part of the safety monitoring data of a coal mine in Shanxi Province in 2019.

5.2. Monitoring Data Compression

In the experiment, 70 pieces of monitoring data of a methane transducer were used for compression experiments. The upper limit of the alarm of the transducer was 0.56%. $\Delta E$ was selected as 0.04. The original data distribution is shown in Fig.3, and the compressed data distribution is shown in Fig.4. Through the analysis, it can be seen that the compression ratio is about 4:1, and the change trend of the data has been more realistically reflected, and the abnormal data has not been compressed, which ensures the accuracy of the abnormal data.

Fig.3 Transducer raw data
5.3. Monitoring Data Storage
First, the data writing performance test was performed, and a total of 10 batch data insertion experiments were carried out, with 50,000 data in each batch. Fig. 5 shows the time it takes to write data, with an average of 24,000 data per second, which can satisfy real-time data storage. And it can be seen from the figure that data writing speed is relatively stable, the fluctuation range is not large, and the performance will not decrease with the increase of written data.

Then, the test of data uncompressed storage and compressed storage and the test of data distribution were carried out. Fig.6 shows the data storage capacity of the two DataNode in HDFS system. It can be seen that the amount of data stored by the two data nodes is basically the same regardless of whether it is compressed or not. The reasonable rowkey design avoids hot spots and data skew problems. It can also be seen from the figure that the compressed data occupies significantly less space than the pre-compressed data, and the compression ratio is about 5:1.
5.4. Monitoring Data Query

The data query is to query the eligible rowkey in ElasticSearch according to various conditions and then query the corresponding data through HBase. A total of 10 tests were performed. Fig.7 shows the total time spent querying data, where time 1 is the time it takes to query the rowkey from ElasticSearch based on the query conditions, and time 2 is the time it takes to query data from HBase based on the rowkey. It can be seen that the total retrieval time is maintained at the order of milliseconds, which can meet the needs of fast data query under massive data.

6. Conclusion

This paper deeply studied the storage of coal mine safety monitoring data in HBase, designed a rowkey structure based on the combination of hash value and inverted timestamp, compressed the data using SDT, and designed a secondary index based on ElasticSearch. Through the analysis and verification of experiments on real coal mine safety monitoring data, the experimental results show that the design has a high efficiency for the storage and query of massive coal mine safety monitoring data.

Acknowledgments

This work was supported by the Institute of Network and Information Systems of North China Electric Power University.
References

[1] Zhang Zhongwen, Zhu Chenwu, et al. Networking big data storage technology for coal mine safety monitoring[J]. Coal Mine Safety, 2016, 47(07): 95-98 (in Chinese).

[2] Oussous A, Benjelloun F Z, et al. Big data technologies: A survey[J]. Journal of King Saud University Computer and Information Sciences, 2018, 30(4): 431-448.

[3] Zhang L, Li Q, et al. A Distributed Storage Model for Healthcare Big Data Designed on HBase[C]// 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2018.

[4] Wang Yangyang. Application of data compression based on the swing door trending algorithm in coal mine safety monitoring system[J]. Coal Mine Safety, 2020, 51(01): 115-117 (in Chinese).

[5] Liu Jinsong. Research and application of intelligent monitoring technology for CNC machine tools[D]. University of Chinese Academy of Sciences (Shenyang Institute of Computing Technology, Chinese Academy of Sciences), 2017 (in Chinese).

[6] Wang Wenxian, Chen Xingshu, et al. A Solr-based HBase secondary indexing scheme for massive data[J]. Information Network Security, 2017(08): 39-44 (in Chinese).