Hindawi
Scientific Programming
Volume 2022, Article ID 1371778, 10 pages
https://doi.org/10.1155/2022/1371778

Research Article

Unbalanced Big Data-Compatible Cloud Storage Method Based on Redundancy Elimination Technology

Tingting Yu

College of Internet of Things Technology, Wuxi Institute of Technology, Wuxi, Jiangsu 214121, China

Correspondence should be addressed to Tingting Yu; yutt@wxit.edu.cn

Received 18 November 2021; Revised 9 December 2021; Accepted 23 December 2021; Published 6 January 2022

Academic Editor: Baiyuan Ding

In order to meet the requirements of users in terms of speed, capacity, storage efficiency, and security, with the goal of improving data redundancy and reducing data storage space, an unbalanced big data compatible cloud storage method based on redundancy elimination technology is proposed. A new big data acquisition platform is designed based on Hadoop and NoSQL technologies. Through this platform, efficient unbalanced data acquisition is realized. The collected data are classified and processed by classifier. The classified unbalanced big data are compressed by Huffman algorithm, and the data security is improved by data encryption. Based on the data processing results, the big data redundancy processing is carried out by using the data deduplication algorithm. The cloud platform is designed to store redundant data in the cloud. The results show that the method in this paper has high data deduplication rate and data deduplication speed rate and low data storage space and effectively reduces the burden of data storage.

1. Introduction

In the big data environment, data security and privacy protection are facing great impact and challenges. In recent years, with the explosive growth of digital information, data occupies more and more storage space [1]. It is found that the redundancy in the big data saved by the application system is as high as 60%, and the redundancy increases with the passage of time. The traditional data storage technology reduces the redundancy of data through coding mapping according to the internal relationship of data, so as to increase the data density and finally reduce the space occupied by data [2, 3]. Moreover, the traditional data storage technology can only eliminate the redundant data inside the file, but it cannot do anything about the data redundancy between different files. It can be seen that the traditional data storage technology and management method have been difficult to meet the requirements of big data in terms of speed, capacity, storage efficiency, and security. Therefore, it is necessary to study an effective data storage method [4, 5].

Reference [6] presents a redundant optical fiber data storage optimization method based on traditional genetic algorithms and data compression algorithms. Combined with the Doppler transformation, the optimal basis function is globally optimized, and the redundant properties of the optical fiber data are analyzed and filtered. Furthermore, the fiber redundant data is initially compressed using the traditional genetic algorithm. The load reduction processing of optical fiber data storage based on K-L characteristics completes the optimization of optical fiber redundant data compression and realizes the optimal storage of optical fiber redundant data. The method can effectively compress the fiber redundant data and handle it more efficiently. However, the method only compresses redundant data and does not really delete redundant data, so redundant data still occupies a large storage space. Reference [7] proposes a design and optimization method of spatial vector data storage model based on HBase. Based on the analysis of relational spatial database storage model, the conversion rules from relational database storage mode to HBase storage mode are applied to the field of spatial vector data management, and a conversion method from spatial vector data relational storage mode to HBase storage mode is proposed, and a space vector data HBase storage model is designed. The model is optimized and improved by using the characteristics of HBase, such as entity nesting,

antinormalization, and no pattern. Using this method for data storage without auxiliary indexing, the data query is highly efficient. But the method only changes the storage method, and deduplication is poor. Reference [8] proposed a digital library data storage method based on compressed sensing, introduced the theoretical basis and mathematical model of compressed sensing, and tried to apply compressed sensing technology to the digital resource management of the library for the first time. This method applies the compression sensing method of the orthogonal matching tracking algorithm to the acquisition of scanning text resources and image electronic resources, with relatively high compression. However, the method does not completely delete the deduplication data, and the duplicate data occupies a large storage space.

Although the above methods improve the efficiency of data storage to a certain extent, there are some problems such as poor redundancy elimination effect and high data storage space. Aiming at the above problems, this paper explores a nonequilibrium big data compatibility cloud storage method based on redundancy elimination technology. Taking unbalanced big data as the main research object, it is hoped that the analysis of cloud storage technology will lay the foundation for further promotion of computer cloud computing data storage technology in the later period. The main research contents and innovations of the thesis include the following:

(1) Based on Hadoop and NoSQL, a data collection platform was designed, and multiple concurrent data collection function modules were opened on multiple machines at the same time, which improved the data collection efficiency of the entire platform

(2) Use the Huffman algorithm to compress the data, which can significantly reduce the storage space of the data and improve the data query speed in the storage mode

(3) Redundancy elimination algorithm in nonequilibrium big data encryption technology is for big data elimination, detecting duplicate data objects in the data flow according to redundancy, transmitting and storing only unique copies of data objects, and replacing other duplicates with the unique data object copy, so as to eliminate the same files or data blocks in the big data set, effectively reduce the storage space of big data, and reduce the amount of data transmitted by the network

2. Unbalanced Big Data Compatibility Cloud Storage Method

Unbalanced data usually means that the number of negative samples in the two types of problems is much larger than the number of positive samples. In reality, examples include credit card transaction fraud identification, telecom equipment failure prediction, enterprise bankruptcy prediction, and radar image monitoring of offshore oil pollution. However, most traditional classification methods are based on the assumption of data balanced distribution in data storage design, When these methods are applied to unbalanced data, they will lead to poor data storage performance. Therefore, unbalanced data storage has become a research hotspot in the field of data processing.

2.1. Design of Unbalanced Big Data Collection Platform. Big data, as the name implies, has a very high demand for the efficiency of data collection, storage, and retrieval. Normally, the data collection efficiency can reach MB/s or GB/s, and the data storage can reach the order of TB or even PB. Because of consistency constraints, traditional relational databases have been difficult to meet such high-intensity requirements [9, 10]. In response to this, a large number of nonrelational databases have emerged. Their common feature is that they perform read and write operations based on Key-Value, but they lack support for complex operations such as multicolumn retrieval and multitable joint statistical analysis. On the other hand, due to the limitation of its caching mechanism, it does not support rapid collection and retrieval of large amounts of data, resulting in low overall efficiency. In order to solve these problems, a new type of big data collection platform is designed based on Hadoop and NoSQL technologies, through which efficient unbalanced data collection is realized, and the data foundation is provided for the subsequent storage and processing of data [11, 12]. Figure 1 is a schematic diagram of the overall architecture of the unbalanced big data collection platform.

According to Figure 1, the overall architecture of unbalanced big data acquisition platform can be started from three levels: physical layer, logical processing layer, and network layer. The construction of functional modules should be considered from the aspects of unbalanced big data acquisition, audit, management, sharing, and security control. Among them, unbalanced big data acquisition module is the most front-end, and its task is to actively collect external information of the platform. There are two collection methods, one is automatic collection, and the other is manual collection. Both methods incorporate the unbalanced big data resources obtained on LAN, intranet, and Internet into the database of the collection platform [13, 14].

The big data collection cluster is equivalent to the internal entrance of the whole system. Taking the process as the execution unit, multiple concurrent data collection function modules are opened on multiple machines at the same time to improve the data collection efficiency of the whole platform; the concurrent data collection function module uses the 5CSEMA5F31C6FPGA core board of Altera cycloneV series as the processor and uses FPGA parallel collection technology to collect several bits of data code collected by the automatic data collection module and the manual data module simultaneously through multiple machine parallel channels. A character was divided into 8 bits to transmit an 8 bit signal once in parallel. Parallel acquisition module mainly includes FPGA acquisition module, digital demodulation module, and high-speed data transmission module. The FPGA terminal will determine whether the data can be collected in parallel at high speed,

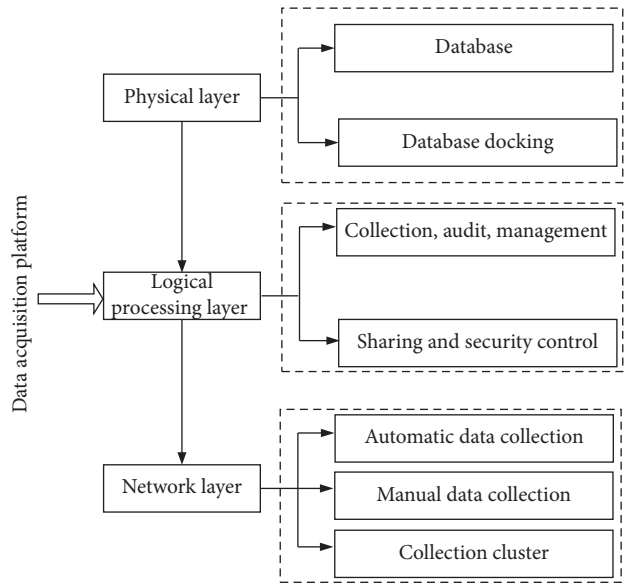

Figure 1: Overall architecture of unbalanced big data collection platform.

FIFO with clock frequency of 25M, after reading 4 bit check bit data, and Hamming coding, and the remaining data packets are collected in parallel, so as to ensure that the collected bit bit rate is 25 Mbps; digital demodulation module codes data according to Hamming code principle divides data by frequency and completes parallel collection and demodulation of multimachine data; high-speed data transmission module is mainly in P2P, based on backplane bus and coaxial transmission line, and vector signal analyzer sends IQ data to FPGA in parallel, trimmed by FPGA, and sends to database in MDA; the parallel acquisition process reduces signal attenuation, improves anti-interference capability, and realizes high-speed parallel data acquisition. After the completion of the data collection, the data analysis stage gradually receives the big data obtained in order, in this step, to complete the unbalanced data classification, operate according to the type, and finally complete the data storage.

2.2. Unbalanced Big Data Classification. Based on the data collected by the unbalanced big data collection platform, in order to further improve data storage efficiency and storage effectiveness, this article uses a classifier to classify the collected data, so as to improve the pertinence of data storage in the subsequent process.

The integration of the classifier in this paper is mainly divided into three steps: the first step is to use the MapReduce framework to process the initial unbalanced data set, thereby obtaining k balanced training subset; the second step is to use these k trains of the training subset to train the classifiers to obtain N classifiers; the third step is to use the majority voting strategy to integrate these N classifiers to complete the classification of unknown sample data [15–17]. The specific algorithm is described as follows:

(1) Enter the initial unbalanced data set D.

(2) Construct m balanced data sets $D = \{d_1, d_2, \ldots, d_m\}$ according to the unbalanced rate in data set D.

(3) Apply the ELM algorithm to train the training set $D = \{d_1, d_2, \ldots, d_m\}$ to obtain n subclassifiers $S = \{s_1, s_2, \ldots, s_n\}$.

(4) When classifying unknown sample data, each subclassifier obtains a classification result and votes on different subclassifiers. The final classification result of the sample is the one with the highest vote.

The overall algorithm process shown in Figures 2 and 3 is a schematic diagram of partial data classification results.

As can be seen from Figure 2, in the unbalanced big data classification process, in the construction of the initial unbalanced data set, each training set contains some examples in the negative class and all the positive examples, ensuring that all available data information in the training set will not be wasted. Classification results according to the data obtained by Figure 3 in the m balanced data set are trained, inspired by the principle of the MapReduce framework, the training of the m data set is distributed to each node, and the ELM algorithm is applied for parallel calculation, which greatly improves the classification efficiency [18]. Extreme Learning Machine (ELM) is an algorithm for single-implied layer feed-forward neural networks. The biggest feature is that both input weights and bias of implicit nodes are randomly generated within a given range, proven to be learning efficient and generalization. The main purpose of training is to solve the weights of the output layer. ELM has the advantages of high learning efficiency and strong generalization ability and is widely used in problems like classification, regression, clustering, and feature learning. For single hidden layer neural networks, ELM can randomly initialize the input weights and bias and get the corresponding output weights. Since only the required solution output weights, the ELM is essentially a linear parameter mode, and its learning process is easy to converge at global minima. For a given group N group training data, the SLFN containing L implied layers and M output layers using ELM learns in the following steps:

(1) Randomly assign the input weight vector and imply layer nodes of the ELM to complete the initialization

(2) Calculate the implied layer output parallel data matrix H

(3) Classify the matrix and calculate the output weight matrix in parallel

That is, the parallel calculation of the ELM neural network is completed.

2.3. Unbalanced Big Data Compression. Although the optimization of unbalanced big data can improve the efficiency of data processing to a certain extent, due to the huge capacity of big data, only classification cannot meet the needs of data processing. Therefore, Huffman algorithm is used to compress the classified unbalanced big data.

Huffman algorithm is a data compression method. When using the Huffman algorithm for data compression,

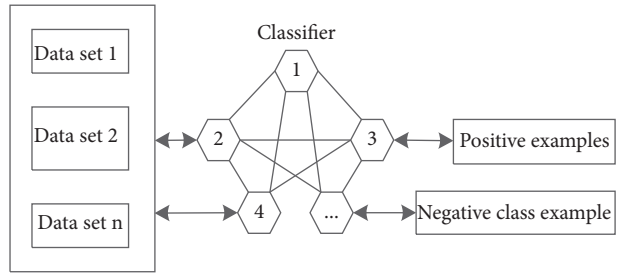

Figure 2: Unbalanced big data classification process.

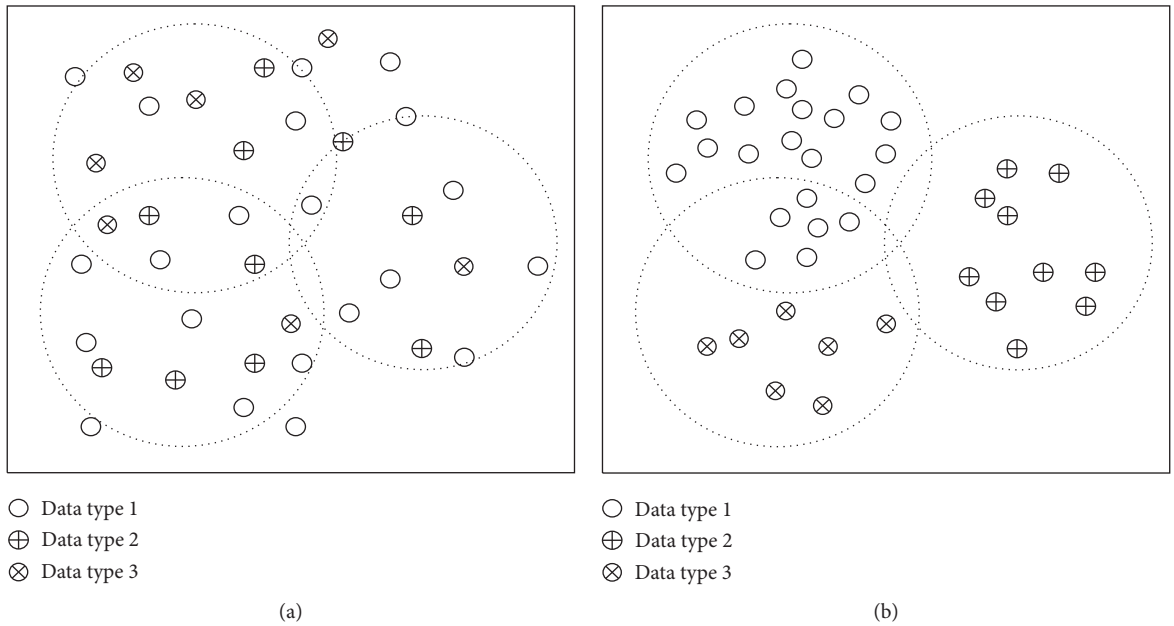

Figure 3: Schematic diagram of data classification results. (a) Before classification. (b) After classification.

the average code length of the data will not change, so this advantage is the coding efficiency uniqueness, which can significantly reduce the data storage space and improve the speed of the data query in the storage mode. In addition, the Huffman algorithm constructs the codeword with the shortest average length of different characters on the basis of the character appearance probability, which is more accurate. After the above classification processing, the data is transformed by wavelet decomposition method, and then Huffman coding. In the process of data compression using the combination of these two methods, the scale of wavelet decomposition should be smaller to reduce the amount of calculation of wavelet transform; binary coding of the transformed data can further improve the compression ratio [19, 20].

Huffman coding algorithm adopts optimized static coding technology, and the binary tree generated by the algorithm has the minimum weighted sum. The algorithm first arranges all the data in descending probability order, establishes a list, and then constructs a tree from bottom to top. Figure 4 is the flowchart of Huffman algorithm data compression.

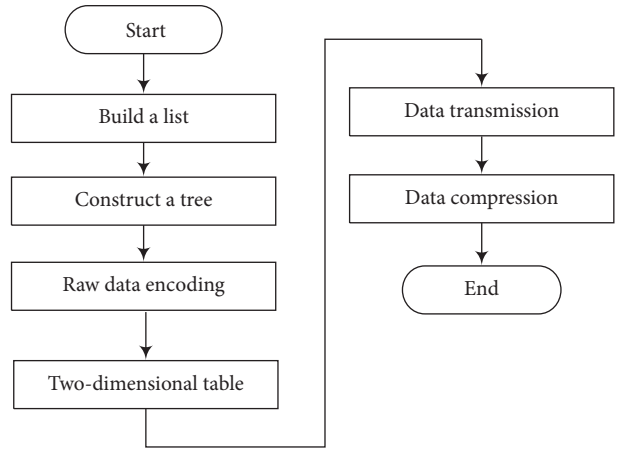

Figure 4: Flowchart of unbalanced big data compression.

According to the unbalanced big data compression process as shown in Figure 4, Huffman algorithm of each leaf is placed in a tree, and then the tree determines the coding of the original data. All the obtained codes form a two-

dimensional table. The size of the table is very small relative to the original data. The two-dimensional table and the encoded data are stored together or sent to the remote end through the communication network. The decoder does not need to traverse the tree but decompress it by looking up the table [21, 22].

2.4. Unbalanced Big Data Encryption. As users pay more and more attention to personal privacy, only realizing data processing can no longer meet the actual needs of users. Therefore, in order to meet the needs of user data security, it is necessary to encrypt the compressed data [23, 24]. The general chaotic encryption algorithm implements data encryption by superimposing a deterministic chaotic sequence on the original text. The decryption party uses the chaotic sequence reproduction method to achieve decryption. This approach is actually a process of superimposing white noise on the baseband signal. After smoothing and denoising, methods such as least squares processing and Kalman filtering can get the approximate original text, which reduces the confidentiality [25, 26]. At the same time, the chaotic sequence reappears in synchronization with the original text, which increases instability factors. The encryption algorithm proposed in this paper successfully solves these problems. The algorithm transforms the original data into a chaotic random sequence in the interval (0, 1) through a series of nonlinear transformations, which means that the baseband signal becomes white noise, so that the smoothing algorithm is useless. In addition, due to the good independence of the chaotic sequence itself, it increases the difficulty of being illegally deciphered [27, 28].

Suppose that the basic function of logistic mapping is

$$q(x) = w_t\delta_t + w_c\delta_c(D_h - S_h). \tag{1}$$

Among them, w_t and w_c represent the odd/even correlation characteristics of logistic map chaotic spread spectrum sequence, respectively; δ_t and δ_c respectively represent the influence of the finite length effect of the sequence on the odd/even correlation characteristics; D_h and S_h represent the balance rate of unbalanced data set and the balance rate of balanced data set, respectively.

The inverse mapping of formula (1) is

$$h = \frac{1}{2} \pm \sqrt{1 - \frac{q}{x}}. \tag{2}$$

When $0 < x \leq 1$, the Logistic map shows the desired chaotic characteristics, and this interval is the chaotic zone. Because the logistic mapping is simple in form and easy to analyze, the encryption algorithm described in this article uses it as the basic mapping. The bifurcation parameter c and the iteration parameter v of the mapping will be used as the encryption algorithm key, and their expressions are

$$\begin{aligned} c &= \sum_{i}^{N} (y_i - \hat{y}_i)^2 \\ v &= \sum_{i=1}^{N} t_i p_i \times T. \end{aligned} \tag{3}$$

Among them, y_i and $\hat{y}_i$ respectively represent the initial value and critical value of the bifurcation parameter; t_i represents the periodic response; p_i represents the chaotic response; T represents the parameter interval corresponding to the chaotic response.

After unbalanced big data is encrypted into ciphertext, it must be restored to the original text through the corresponding decryption algorithm. But logistic inverse mapping is a one-to-many mapping. To restore it, the one-to-one mapping problem of the mapping must be solved. This paper adopts the method of fixed word length discretization and adding disturbance term, which can successfully solve this problem. Finally, the corresponding encryption vector and decryption vector are generated from the key, and the original text encryption and ciphertext decryption can be performed.

2.5. Big Data Elimination Algorithm. Data redundancy technology refers to a large amount of data with the same content in the process of storing data, and the process of deleting redundant files and data blocks through repeated data detection, so that only unique data is stored in the system [29]. Compared with the traditional data compression technology, the redundancy elimination algorithm in the nonequilibrium big data encryption technology can not only eliminate the data redundancy in the files, but also eliminate the data redundancy between the files in the datasets, effectively reduce the storage space of big data, and provide new ideas for the storage and processing of big data [30].

2.5.1. The Basic Principle of the Deduplication Algorithm. Data deduplication technology detects duplicate data objects in the data stream based on the redundancy of the data itself, only transmits and stores the only data object copy, and uses a pointer to the unique data object copy to replace other duplicate copies. Data deduplication technology is committed to saving storage space and network bandwidth resources. In the process of unbalanced big data storage, the introduction of data deduplication technology can optimize the storage space of unbalanced big data and eliminate the same files or data blocks in the big data set, so as to reduce the workload of encryption processing [31, 32]. On the other hand, data deduplication effectively compresses the data, reduces the amount of data transmitted by the network, and reduces the bandwidth consumption. In order to improve the deduplication rate and deduplication efficiency of massive data deduplication system, this paper carries out big data deduplication.

Generally, the ratio of the number of bytes before deduplication B_{in} to the number of bytes after processing B_{out} is used to measure the data elimination ratio (DER) G_u, as shown in formula (5):

$$G_u = \frac{B_{\text{in}}}{B_{\text{out}}}. \tag{4}$$

G_u is usually determined by two factors: the type of partitioning strategy used and the average data block size. Although the data reduction rate shown in formula (5) takes into account the repeated data between data blocks after blocking and the data compression within a single data block, the metadata overhead is not considered. However, the metadata cost in the deduplication system cannot be ignored. Therefore, a correction formula for data reduction rate is proposed:

$$G_u' = \frac{B_{\text{in}}\left(U_i \times r\right)^2 Y}{B_{\text{out}}\left(U_j \times r\right)^2 Y'}. \tag{5}$$

Among them, U_i and U_j respectively represent the same data and similar data; Y and Y' respectively represent the space utilization rate before deletion and the space utilization rate after deletion; r represents the overhead size of metadata, and the calculation method is as follows:

$$r = \frac{\mu_k\left[G_u\left(x_k\right) - G_u'\right]}{\mu_k^{\text{avg}}}. \tag{6}$$

Among them, μ_k represents the size of the metadata; μ_k^{avg} represents the average value of the size of the metadata.

According to formula (6), there are two types of duplicate data: the same data and similar data. For these two types of data, duplicate data deletion technology is used to detect them, respectively. The specific processing methods are as follows:

(1) Same data detection technology: it divides the data set according to certain rules and replaces the same part of the data set with a pointer. According to different granularity, the same data detection technology is divided into complete file detection technology and data block detection technology. According to different blocking methods, data block detection technology is divided into fixed length block detection technology, content-based variable length block detection technology, and sliding window block detection technology [33].

(2) Similar data detection technology: according to the inherent similarity characteristics of data, the data set is detected. Delta coding technology is used to compress similar data to reduce data storage space and bandwidth occupied during transmission.

2.5.2. Implementation of Unbalanced Big Data Compatibility Cloud Storage. Section 2.5.1 discusses data deduplication technology, based on which a cloud platform is built to realize unbalanced big data compatibility cloud storage. The cloud platform has the ability of data reception, collection, and storage. It mainly obtains unbalanced big data resources through the Internet. The platform uniformly manages all kinds of unbalanced big data resources, establishes meta database and data catalog, and provides data browsing, query, download, and other services based on large screen for internal users and provides data sharing services for users based on the data distribution service submodule [34, 35]. Figure 5 shows the overall architecture of the cloud platform.

According to the overall architecture of the cloud platform in Figure 5, the software design of each platform is elaborated:

(1) Platform software support: cloud platform software support is the operating environment of various application software, and its components include operating systems, software operating platforms, application middleware, virtualized storage systems, and security systems. Among them, the data dynamic allocation strategy of the RSDO model supports the operation of the platform, which is mainly responsible for allocating unbalanced big data to different storage spaces according to the current storage situation of the data center to achieve the purpose of data balanced storage. The cloud scheduling service monitors the data of the entire cloud platform at the software support layer and provides task scheduling services for the storage of data in the cloud platform.

(2) Data distribution service submodule: the data distribution service submodule is the main portal for users to access the data storage center, and it is also the main platform for the center to provide external cloud services such as data sharing and application promotion. Users use the data resource cataloging service provided by the platform to search, query the metadata information of various resources, and understand the basic form of the data concerned. In addition, the system forms a data storage set through the classification and integration of data attribute characteristics, which is convenient for users to locate and understand the data of interest more quickly [36].

(3) Compatibility: the design of the unbalanced big data storage platform needs to organize the underlying database information in order to improve the compatibility of the data in the platform and avoid problems such as information corruption during the data storage process. Users can access the database uniformly through the data management portal and use the big data access unified interface to extract and store the required data. Finally, the data can be cached at the data relationship mapping layer and stored permanently in the bottom layer of the database. In addition, data storage platforms usually use nonrelational data characteristics to classify cached data to achieve data expansion and conversion, so as to optimize the data communication mode of the client. In the case of a relatively large

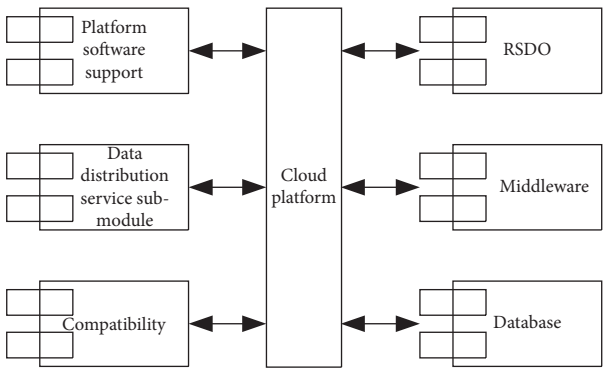

Figure 5: Schematic diagram of the overall architecture of the cloud platform.

amount of data, the cluster method needs to be used in the database cache system to optimize and store temporary data, so as to achieve the design goal of converting compatible large amounts of data.

Based on the above-built cloud platform, perform unbalanced big data-compatible cloud storage, and the specific steps are as follows:

(1) Assuming that the product of the CPU processing frequency, number of processors, and number of CPU cores of the a-th server in the cloud platform are represented by F_{all}, the expression is

$$F_{\text{all}} = F^{a}_{\text{CPU}} \cdot E_n \cdot L_{\text{CPU}}. \tag{7}$$

Among them, F^{a}_{CPU} represents the CPU processing frequency of the a-th server; E_n represents the number of processors; L_{CPU} represents the number of CPU cores.

(2) Assuming that the memory capacity of the cloud platform is represented by C_α, the disk capacity is represented by C_β, the disk read and write rate is represented by V_ε, and the network bandwidth throughput is represented by ∂_ω; then, the ratio between each parameter and the maximum value of the corresponding parameter of all servers is

$$\begin{cases} F^{a}_{\text{CPUC}} = \dfrac{C_\alpha}{C_{\alpha\max}}, \\ E_n = \dfrac{C_\beta}{C_{\beta\max}}, \\ L_{\text{CPU}} = \dfrac{V_\varepsilon}{V_{\varepsilon\max}}, \\ F_{\text{all}} = \dfrac{\partial_\omega}{\partial_{\omega\max}}. \end{cases} \tag{8}$$

The above ratio can reflect the performance of a certain parameter of the a-th server in the cluster server. The following uses the weighted average method to integrate these data together to calculate the overall performance value ℓ_a of the a-th server:

$$\ell_a = a_1 \times F^{a}_{\text{CPU}} + a_2 \times E_n + a_3 \times L_{\text{CPU}} + a_4 \times F_{\text{all}}. \tag{9}$$

From this, the weight value of the a-th server relative to the entire server cluster can be obtained:

$$\theta = \sum_{a \in A}^{n} \ell_a \times O_a. \tag{10}$$

According to the real-time performance of the cluster server statistics, define CPU_a as the actual CPU utilization of the a-th server, RAM_a represents the actual utilization of the memory, Disk_a represents the actual utilization of the disk capacity, Rate_a represents the average disk read and write rate, and Throughput_a represents the actual throughput of the network bandwidth, and it can calculate the actual load weight threshold of the server in the cloud platform:

$$\theta' = \frac{\theta}{\sum_{a \in A}^{n} \ell'_a} \times O'_a. \tag{11}$$

This article collects, classifies, compresses, and encrypts unbalanced big data and builds a cloud platform for data objects. In addition, the technology of eliminating redundancy is applied to unbalanced big data storage, so that running a cloud platform for data storage is self-contained. Features such as adaptability, fast response, and high efficiency were studied.

3. Simulation Experiment

In order to verify the effectiveness of the unbalanced big data compatibility cloud storage method based on redundancy elimination technology proposed in this paper, simulation experiments are carried out. In the experiment, reference [6] method and reference [7] method are used as comparison methods. The data redundancy elimination effect and data storage space are taken as the experimental indexes. The data redundancy elimination effect specifically refers to the duplicate data deletion rate and deletion rate.

3.1. Experimental Data Set. In this paper, six data sets are selected in the public machine learning database UCI, and one of the data sets in each data set is regarded as a positive class, and the rest are classified as negative classes, thus forming an unbalanced data set with different degrees of imbalance. In order to ensure the accuracy of the experimental results, all data sets are preprocessed with 0 mean and standard deviation 1. The total number of data is 74589, and the total size is 1130070.49 KB. The average size of the data block in the experiment is 8 KB, and the sliding window size is It is 48 Byte. Simulate the MapReduce computing environment in a stand-alone environment to verify the effectiveness of the method in this paper.

The specific experimental data set description is shown in Table 1.

Table 1: Experimental data set.

Data set	Number of samples/pieces	Category ratio (%) (positive/negative)
Letter	10237	5.24
Image	9892	10.29
Iris	11369	20.31
Class	10543	4.17
Wine	8446	9.30
Pima	24102	12.09

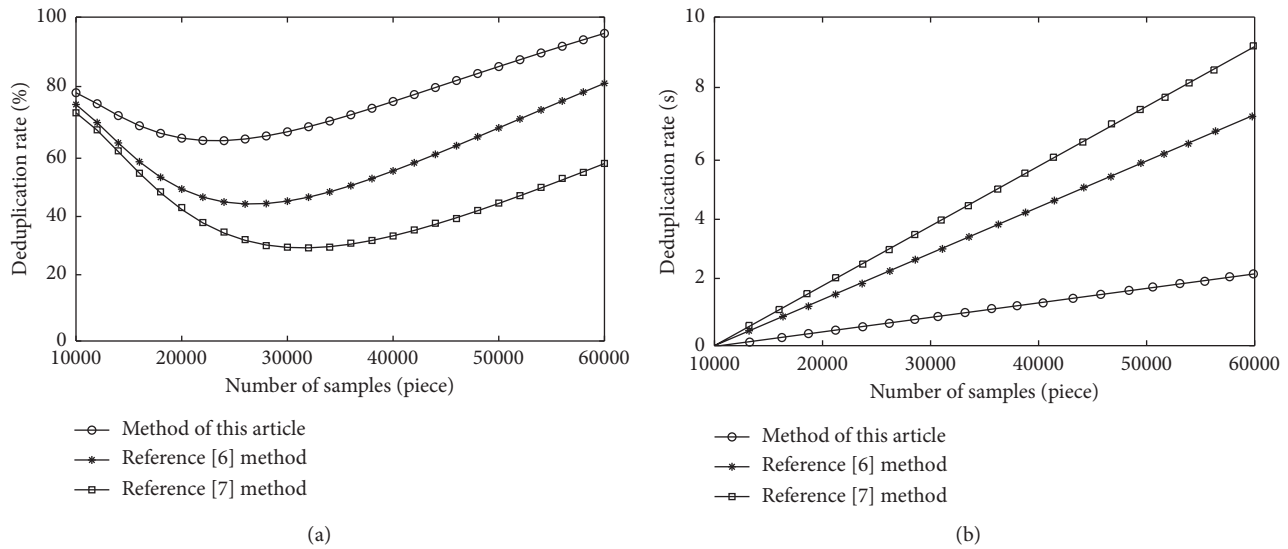

Figure 6: Comparison of data redundancy elimination effect. (a) Deduplication rate. (b) Deduplication speed rate.

3.2. Analysis of Experimental Results

3.2.1. Data Elimination Effect. The performance of this method, reference [6] method, and reference [7] method in data deduplication rate and deduplication speed rate is compared.

The performance test methods of the three methods are as follows: given a certain amount of data, stored in different directories and files, read these data, then store them in different methods, count the space occupied after storage, and calculate the deduplication rate. At the same time, count the total time from the beginning of reading data to the completion of the last storage, and calculate the deduplication speed rate. The data source is from Table 1, and the results are shown in Figure 6.

It can be seen from Figure 6(a) that, after using the method in this paper to eliminate redundancy, the deduplication rate is higher than that in reference [6] method and reference [7] method, and the highest value of the deduplication rate of the method in this paper is 96%. It can be seen from Figure 6(b) that the deduplication time is much lower than reference [6] method and reference [7] method, and the highest value of the duplicate data deletion time of the method in this paper is only 2.2s. The comparison results show that the method in this paper can not only ensure a high deduplication rate of duplicate data, but also improve the deduplication speed rate of duplicate data.

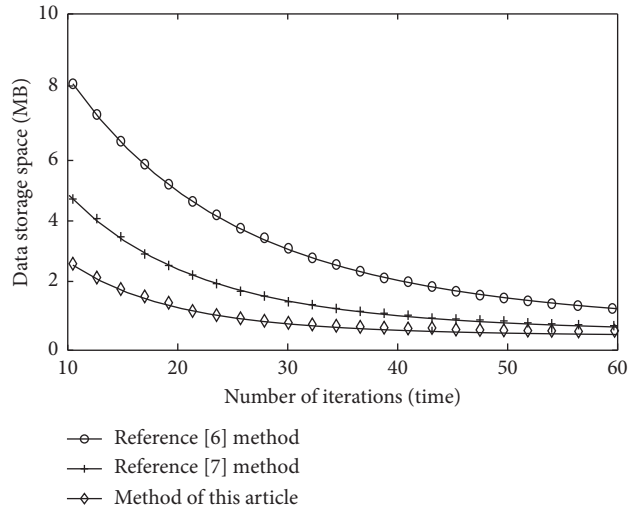

Figure 7: Data storage space.

3.2.2. Data Storage Space. In order to further verify the effectiveness of the method in this paper, the three methods are compared with the data storage space as an experimental indicator, and the results are shown in Figure 7.

It can be seen from the curve trend in Figure 7 that when processing the same amount of unbalanced big data, the data

storage space occupied by the method in this paper is significantly lower than that of reference [6] method and reference [7] method; although with iteration with the increase in the number of times, the gap between different methods is gradually narrowing, but the method in this paper can still maintain its advantages, indicating that it has a significant advantage in unbalanced big data storage space.

4. Conclusion

In order to improve the elimination capacity of the redundant data and reduce the storage space of the data occupied, a nonequilibrium big data-compatible cloud storage method based on the redundant elimination technology is proposed. Big data acquisition platform was designed based on Hadoop and NoSQL technologies, and data was collected in parallel using FPGA parallel acquisition technology. Based on this platform, the collected data is processed using the parallel utilization of the classifier classification. The Huffman algorithm is used to compress the redundancy elimination algorithm in the nonequilibrium large data encryption technique. Design a cloud platform to store big data. Experimental results show that the proposed method works better than traditional data storage methods, has high deduplication rate, and solves the problem of large data storage.

Data Availability

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding this work.

References

[1] A. Singh, S. Garg, K. Kaur, S. Batra, N. Kumar, and K. K. R. Choo, "Fuzzy-folded bloom filter-as-a-service for big data storage in the cloud," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2338–2348, 2019.

[2] D. Gupta and R. Rani, "A study of big data evolution and research challenges," *Journal of Information Science*, vol. 45, no. 3, pp. 322–340, 2019.

[3] H. Chergui and C. Verikoukis, "Big data for 5G intelligent network slicing management," *IEEE Network*, vol. 34, no. 4, pp. 56–61, 2020.

[4] K. Kaur, S. Garg, G. Kaddoum, E. Bou-Harb, and K. K. R. Choo, "A big data-enabled consolidated framework for energy efficient software defined data centers in IoT setups," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2687–2697, 2020.

[5] S. Kalaivani, C. Tharini, K. Saranya, and K. Priyanka, "Design and implementation of hybrid compression algorithm for personal health care big data applications," *Wireless Personal Communications*, vol. 113, no. 1, pp. 599–615, 2020.

[6] Z. P. Huang, L. Wang, S. X. Zhang, T. Z. Yu, and Q. R. Zhang, "Redundant optical fiber data storage optimization based on traditional genetic algorithm and data compression algorithm," *Laser Journal*, vol. 40, no. 3, pp. 135–139, 2019.

[7] P. Xie, C. C. Yang, S. Xiong, L. S. He, and X. D. Zhou, "Design and optimization of spatial vector data storage model based on HBase," *Acta Geodaetica et Cartographica Sinica*, vol. 49, no. 10, pp. 1365–1373, 2020.

[8] T. Chen, D. S. Wang, Z. J. Wang, and W. Liu, "An efficient data storage method in digital library based on compressed sensing," *Library Journal*, vol. 38, no. 09, pp. 4–11, 2019.

[9] D. de Santana Nunes, J. L. V. de Brito, and G. N. Doz, "A low-cost data acquisition system for dynamic structural identification," *IEEE Instrumentation and Measurement Magazine*, vol. 22, no. 5, pp. 64–72, 2019.

[10] F. Santoni, A. De Angelis, A. Moschitta, and P. Carbone, "A multi-node magnetic positioning system with a distributed data acquisition architecture," *Sensors*, vol. 20, no. 21, p. 6210, 2020.

[11] A. Elzanaty, A. Giorgetti, and M. Chiani, "Limits on sparse data acquisition: RIC analysis of finite Gaussian matrices," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1578–1588, 2019.

[12] E. Grimberg, A. Botzer, and O. Musicant, "Smartphones vs. in-vehicle data acquisition systems as tools for naturalistic driving studies: a comparative review," *Safety Science*, vol. 131, no. 6, Article ID 104917, 2020.

[13] A. Mehto, S. Tapaswi, and K. K. Pattanaik, "Virtual grid-based rendezvous point and sojourn location selection for energy and delay efficient data acquisition in wireless sensor networks with mobile sink," *Wireless Networks*, vol. 26, no. 5, pp. 3763–3779, 2020.

[14] M. Rapin, F. Braun, A. Adler et al., "Wearable sensors for frequency-multiplexed EIT and multilead ECG data acquisition," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 3, pp. 810–820, 2019.

[15] M. Irfan, Z. Jiangbin, M. Iqbal, Z. Masood, M. H. Arif, and S. R. u. Hassan, "Brain inspired lifelong learning model based on neural based learning classifier system for underwater data classification," *Expert Systems with Applications*, vol. 186, no. 1, Article ID 115798, 2021.

[16] D. Griffiths and J. Boehm, "A review on deep learning techniques for 3D sensed data classification," *Remote Sensing*, vol. 11, no. 12, p. 1499, 2019.

[17] N. Gomathi and N. P. Karlekar, "Ontology and hybrid optimization based SVNN for privacy preserved medical data classification in cloud," *The International Journal on Artificial Intelligence Tools*, vol. 28, no. 3, Article ID 1950009, 2019.

[18] F. Bensaid and A. M. Alimi, "Online feature selection system for big data classification based on multi-objective automated negotiation," *Pattern Recognition*, vol. 110, no. 1, Article ID 107629, 2020.

[19] R. Corda and C. Perra, "Hologram domain data compression: performance of standard codecs and image quality assessment at different distances and perspectives," *IEEE Transactions on Broadcasting*, vol. 66, no. 2, pp. 292–309, 2020.

[20] J. Uthayakumar, T. Vengattaraman, and P. Dhavachelvan, "A new lossless neighborhood indexing sequence (NIS) algorithm for data compression in wireless sensor networks," *Ad Hoc Networks*, vol. 83, no. 2, pp. 149–157, 2019.

[21] M. Tegmark and T. Wu, "Pareto-optimal data compression for binary classification tasks," *Entropy*, vol. 22, no. 1, p. 7, 2019.

[22] D. Greenfield, V. Wittorff, and M. Hultner, "The importance of data compression in the field of genomics," *IEEE Pulse*, vol. 10, no. 2, pp. 20–23, 2019.

[23] R. Geetha, T. Padmavathy, T. Thilagam, and A. Lallithasree, "Tamilian cryptography: an efficient hybrid symmetric key

encryption algorithm," *Wireless Personal Communications*, vol. 112, no. 1, pp. 21–36, 2020.
[24] M. S. Khoirom, D. S. Laiphrakpam, and T. Tuithung, "Audio encryption using ameliorated ElGamal public key encryption over finite field," *Wireless Personal Communications*, vol. 117, no. 2, pp. 809–823, 2021.
[25] X. Wang, N. Guan, and J. Yang, "Image encryption algorithm with random scrambling based on one-dimensional logistic self-embedding chaotic map," *Chaos, Solitons & Fractals*, vol. 150, no. 3, Article ID 111117, 2021.
[26] J. Mou, F. Yang, R. Chu, and Y. Cao, "Image compression and encryption algorithm based on hyper-chaotic map," *Mobile Networks and Applications*, vol. 5, no. 3, pp. 1–13, 2019.
[27] I. Yasser, M. A. Mohamed, A. S. Samra, and F. Khalifa, "A chaotic-based encryption/decryption framework for secure multimedia communications," *Entropy*, vol. 22, no. 11, p. 1253, 2020.
[28] X. Wang, S. Lin, and Y. Li, "A chaotic image encryption scheme based on cat map and MMT permutation," *Modern Physics Letters B*, vol. 33, no. 27, Article ID 1950326, 2019.
[29] S. Kumar and V. K. Chaurasiya, "A strategy for elimination of data redundancy in Internet of things (IoT) based wireless sensor network (WSN)," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1650–1657, 2019.
[30] Z. A. Pan, J. F. Wang, and M. F. Wang, "Research on cloud storage method of multi process spatial data based on SDN technology," *Computer Simulation*, vol. 38, no. 5, pp. 375–379, 2021.
[31] X. Xu, L. Liu, X. Zhang, W. Guan, and R. Hu, "Rethinking data collection for person re-identification: active redundancy reduction," *Pattern Recognition*, vol. 113, no. 4, Article ID 107827, 2021.
[32] S. Chandak, K. Tatwawadi, I. Ochoa, M. Hernaez, and T. Weissman, "SPRING: a next-generation compressor for FASTQ data," *Bioinformatics*, vol. 35, no. 15, pp. 2674–2676, 2019.
[33] M. Besseghier, A. B. Djebbar, A. Zouggaret, and I. Dayoub, "Joint channel estimation and data detection for OFDM based cooperative system," *Telecommunication Systems*, vol. 73, no. 4, pp. 545–556, 2020.
[34] Z. Qian, X. Wang, X. Liu, X. Xie, and T. Song, "An approach to dynamically assigning cloud resource considering user demand and benefit of cloud platform," *Computing*, vol. 102, no. 8, pp. 1817–1842, 2020.
[35] J. Yun, K.-W. Park, D. Koo, and Y. Shin, "Lightweight and seamless memory randomization for mission-critical services in a cloud platform," *Energies*, vol. 13, no. 6, p. 1332, 2020.
[36] L. S. Subhash and R. Udayakumar, "Sunflower whale optimization algorithm for resource allocation strategy in cloud computing platform," *Wireless Personal Communications*, vol. 116, no. 4, pp. 3061–3080, 2021.