Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 7849726, 11 pages
https://doi.org/10.1155/2022/7849726

Research Article

Load Balancing Cloud Storage Data Distribution Strategy of Internet of Things Terminal Nodes considering Access Cost

Jiansheng Wu,[1] **Weimin Xu**,[1] **and Jiarong Xia**[2]

[1]*China Tobacco Zhejiang Industrial Co., Ltd.,, Ningbo, Zhejiang 315504, China*
[2]*School of Mathmatics, Hangzhou Normal University, Hangzhou 311121, China*

Correspondence should be addressed to Jiarong Xia; jrxia@hznu.edu.cn

Received 29 November 2021; Revised 28 December 2021; Accepted 3 January 2022; Published 24 January 2022

Academic Editor: Akshi Kumar

With the rapid development of Internet of Things (IoT) technology, IoT terminal nodes are facing many challenges in data storage, distribution, and data management. In particular, in the IoT terminal nodes considering access cost, the corresponding data distribution and storage are professional, complex, and miscellaneous. Based on the abovementioned current situation, this article innovatively proposes a complex sensor data placement algorithm based on the cloud storage distribution of IoT terminal nodes. Under this algorithm, the accurate division of IoT data I/O methods is realized through reasonable configuration. Through the adaptive sensing algorithm, while fully considering the access cost of the algorithm, the performance of the IoT data storage system is further optimized. In the corresponding terminal node load balancing problem, this article innovatively proposes the terminal node data sorting and distribution algorithm through the node data. The sorting and distribution algorithm realizes the precise segmentation of the IoT data to be processed, thereby realizing the improvement of data reading and processing speed. Based on the proposed algorithm, this article designs a load balancing cloud storage data distribution optimization system of IoT terminal nodes considering access cost and carries out experimental verification in a real environment. The experimental results show that the data pattern division accuracy corresponding to the proposed distribution strategy is improved to 97.13% and the corresponding data access efficiency is improved to 98.3%, compared with the traditional distribution strategy. Therefore, the data distribution strategy proposed in this article has obvious performance advantages and further promotion value.

1. Introduction

The main architecture of the Internet of Things (IoT) includes the perception layer, network layer, management layer and application layer. The conventional sensing layer is mainly used to collect relevant physical information, such as temperature, humidity, air composition, and optical signals. The continuous collection and storage of data by the relevant sensor networks of the IoT cause problems of data distribution and data management [1–3]. The relevant data of the IoT are mainly classified into structured data and unstructured data. The corresponding structured data mainly depend on open-source databases and other products, while the storage of the corresponding unstructured data depends on the database software developed by the corresponding unstructured data model. At the data storage level of the IoT, cloud storage technology, as the development direction of data storage, mainly processes and analyzes a large number of different types of data generated by the IoT through cluster application, network technology, and distributed storage technology and provides external data access and storage [4, 5]. The conventional cloud storage architecture includes the application interface layer, infrastructure layer, and storage layer. The corresponding application interface layer mainly includes network access, user authentication, permission management, public API interface, application software interface, and network service interface. Moreover, the corresponding infrastructure layer includes the cluster system, distributed file system, grid system, content distribution and data deduplication, data compression, and data encryption processing. The corresponding storage layer mainly includes storage virtualization, centralized storage management, status monitoring, maintenance, and upgrading [6]. Data placement and node load balancing are important technologies of distributed storage in cloud storage technology. Load balancing mainly realizes the dynamic expansion of storage resources to further improve the utilization of resources, minimize the data response time, and further reduce the fault recovery and data switching time to ensure the quality of user data access in the IoT system [7, 8]. At the level of corresponding data placement and distribution, the flat distribution of the data system is of great significance. The directory organization corresponding to the traditional data distribution is unreasonable, and the node data distribution is too scattered or too centralized, which will further affect the response-ability of the data system, further reduce the system performance, and increase the data maintenance cost [9, 10]. Compared with other storage models, cloud storage has the following advantages: (1) easy to expand: the storage space is expanded in time according to the number of users and space of the server, which will also not affect the use of the front-end users; (2) reliable and safe: data synchronization effectively avoids media storage of data. Cloud storage can fully realize load balancing, which can realize the flow distribution control service of distributing access traffic to multiple back-end cloud servers according to the forwarding strategy. Moreover, the external service capabilities of the application system are extended through traffic distribution, and the availability of the application system is improved by eliminating single points of failure.

The traditional cloud storage data distribution strategy of load balancing of IoT terminal nodes has some disadvantages in data placement and load balancing technology. At the data placement level, the disadvantages are as follows: differences in data storage performance and availability storage cost between different data storage service providers and differences in different data reading and writing modes in the IoT sensor network. Therefore, the traditional data placement algorithm only meets a single optimization goal and cannot meet the application storage requirements under the huge amount of data in IoT. Based on this, constructing an efficient data placement algorithm to optimize the performance of different data access is of great significance to improve the performance of cloud storage data systems for load balancing of IoT terminals [11, 12]. For the problem of load balancing, after the corresponding IoT data are initialized, as the storage system continues to serve various big data-related applications on the upper layer, the data layout corresponding to each cloud storage system will be unevenly distributed, which will lead to a serious decline in the performance of data access [13–15]. After the storage system has been running for a period of time, it will become normal that the corresponding data load is seriously unbalanced. Once the corresponding load of a data storage service provider is very high, the system may be blocked and stagnant. When the data are stagnant, the corresponding outside world must send out sudden data access and the whole system will have a bottleneck [16, 17]. Based on this, we aim to realize the load balancing of IoT terminal nodes to make the system achieve load balancing and realize the application optimization access function.

Based on the disadvantages of the corresponding data distribution and storage of IoT terminals considering the access cost, this article will propose a complex perceptual data placement algorithm based on the cloud storage distribution of IoT terminal nodes and realize the I/O mode of accurately dividing IoT data by reasonably configuring hybrid perceptual algorithm and adaptive perceptual algorithm to optimize further the performance of the IoT data storage system considering the access cost. Aiming at the problem of terminal node load balancing, this article innovatively proposes the terminal node data sorting and distribution algorithm, which realizes the accurate segmentation of the IoT data to be processed through the node data sorting and distribution algorithm to further accelerate the data reading and processing speed. At the same time, a periodic load balancing algorithm is added to the terminal node data sorting and distribution algorithm so as to further solve the problem of unbalanced node data layout. Based on the proposed algorithm, this article designs a load balancing cloud storage data distribution optimization system of IoT terminal nodes considering access cost and carries out experimental verification in a real environment. The experimental results show that the data pattern division accuracy corresponding to the distribution strategy proposed in this article is improved to 97.13% and the corresponding data access efficiency is improved to 98.3% compared with the traditional distribution strategy. The data distribution strategy proposed in this article has obvious performance advantages and further promotion value.

The main structure of this article is as follows: the second section will mainly analyze the current research status of the load balancing cloud storage data distribution system of IoT terminal nodes considering access cost. The third section will focus on the analysis of complex perceptual data placement algorithm and terminal node data sorting and distribution algorithm and design the cloud storage data distribution optimization system for load balancing of IoT terminal nodes considering access cost. The fourth section will verify the system and analyze its data. Finally, the article is summarized.

2. Related Research: Research Status of Load Balancing Cloud Storage Data Distribution System of Internet of Things Terminal Nodes considering Access Cost

A large number of researchers and scientific research institutions have studied and analyzed the data placement and load balancing problems faced by the load of terminal nodes of the IoT. At the research level of data placement and distribution, relevant cloud storage manufacturers in the United States put forward simple storage services, and their corresponding storage mainly involves three indicators: basic storage unit, unique identifier, and storage object container. When there are a few corresponding IoT data, the corresponding distribution strategy is relatively simple; therefore, not too much data will be stored in the corresponding storage object container. However, once there is too much or blocked data, the performance of the cloud storage system will decline, which is not conducive to the maintenance and dynamic development of the system [18, 19]. European cloud storage service providers propose a static directory layering strategy, which mainly adopts a two-tier static object distribution model, alleviating the problem of too centralized data distribution on nodes to a certain extent, but this model still has certain limitations. The even distribution of the corresponding data depends on the corresponding IoT data level [20]. At the level of corresponding load balancing algorithms, relevant Chinese researchers have proposed a scheme to realize load balancing distribution in a distributed environment. However, this scheme does not consider the corresponding data copies, which may lead to multiple data copies appearing on the same data node after the relevant migration. At the same time, this load balancing strategy does not explicitly specify how much data can be migrated during the migration process. If the corresponding migration amount is too large, the load node will exceed the threshold [21]. Relevant European researchers have proposed a load balancing algorithm based on private cloud storage, which mainly realizes the balanced storage of data by calculating the weight corresponding to the storage node. However, the algorithm does not fully consider the heterogeneity of the node itself in the operation process. It treats all nodes equally, and it does not fully consider the heavy data at this time; if the data is too heavy, data migration will lead to the risk of increasing system bandwidth [2, 22]. Based on the analysis of the above research status, the current traditional load data placement algorithm and load balancing algorithm of IoT terminal nodes have more or fewer disadvantages, which leads to the accuracy of data mode division of cloud storage system and then leads to a serious decline in the corresponding data access efficiency.

3. Research on Load Balancing Cloud Storage Data Distribution Strategy of Internet of Things Terminal Nodes considering Access Cost

This section mainly analyzes and studies the corresponding key algorithms and system design of the IoT terminal node load balancing cloud storage data distribution system considering access cost. The corresponding system principle block diagram is shown in Figure 1. As shown in Figure 1, the key algorithms proposed in this article mainly have two points: the complex aware data placement algorithm and node load balancing strategy for processing data placement. At the corresponding hardware system design level, it also includes the distribution of IoT terminal nodes, IoT network element data management layer, IoT file service layer, and IoT data node control layer. There is no data management layer in the corresponding system, including IoT user and equipment management, IoT user data node allocation, IoT monitoring data management equipment, and IoT migration data management. The corresponding IoT file service layer includes the upload and download of IoT data files, the upload and download of quantity network audio and video files, the deletion of IoT files and other related modules. The whole system includes monitoring system components.

3.1. Complex Perceptual Data Placement Algorithm.

The complex perceptual data placement algorithm proposed in this article mainly solves the problem of data placement distribution in the IoT. The complex perceptual data placement algorithm is mainly divided into two processing levels. Firstly, the I/O mode of the terminal data nodes of the IoT is identified and classified based on the decision tree algorithm. Then, the placement strategy is formulated based on the processed IoT data. The principle block diagram of the corresponding complex perceptual data placement algorithm is shown in Figure 2.

The IoT data I/O mode discrimination and classification algorithm based on decision tree mainly discriminates the data I/O type based on the six key factors of IoT data. The corresponding six factors are as follows: IoT data size, data type, IoT data life cycle, IoT data creation time, the last creation time, and the last modification time of IoT data [23–25]. Based on the above six factors, the I/O mode of IoT data is determined. The modes determined in this section are as follows: read-only mode, write-only mode, read more and write less mode, read less and write more mode. The IoT data I/O mode discrimination and classification algorithm based on decision tree mainly follows the following operation steps:

Step 1: calculate the gain value of IoT terminal node data based on six factors: IoT data size, data type, IoT data life cycle, IoT data creation time, IoT data last creation time, and last modification time. The corresponding calculation formulas are shown in formulas (1), (2), and (3), the corresponding IoT data size is V, the data type is L, the life cycle of IoT data is S, the creation time of IoT data is C, the last creation time of IoT data is L, and the last modification time is D.

$$\begin{aligned} G(\text{data}, D) = {} & Q(\text{data}_1 + \text{data}_2 + \cdots + \text{data}_i) \\ & - Q(\text{data}_1|D_1) + \cdots Q(\text{data}_i|D_i), \end{aligned} \tag{1}$$

$$\begin{aligned} Q(D) = {} & -[\eta_1(a_1)\log(\eta_1(a_1))) + \cdots \\ & + \eta_i(a_i)\log(\eta_i(a_i))), \end{aligned} \tag{2}$$

$$\begin{aligned} Q(\text{data}|D) = {} & \eta(D_1) * Q(\text{data}_1|D_1) + \cdots \\ & + \eta(D_i) * Q(\text{data}_i|D_i). \end{aligned} \tag{3}$$

Step 2: divide the I/O mode of the IoT data according to the gain value calculated in Step 1, and repeat the operations of Step 1 and Step 2 for the IoT data until all the IoT data can be defined as unique data mode.

Step 3: generate a data I/O mode classification model of IoT terminal nodes based on a decision tree to determine the complete data I/O type.

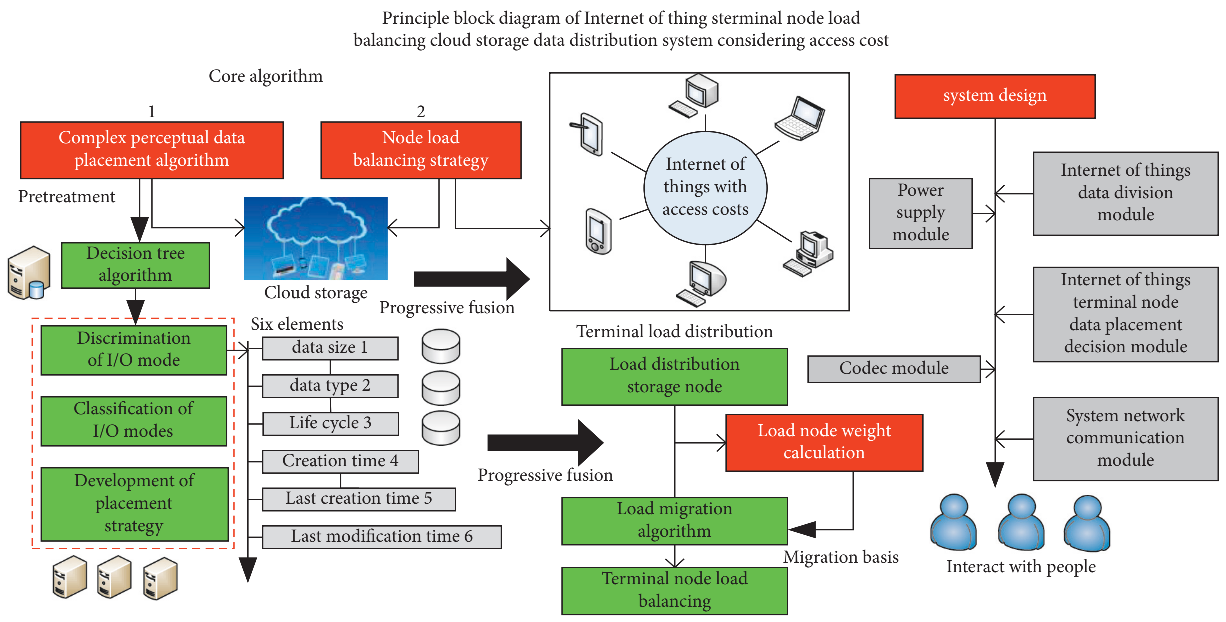

Principle block diagram of Internet of thing sterminal node load balancing cloud storage data distribution system considering access cost
Core algorithm
1
Complex perceptual data placement algorithm
2
Node load balancing strategy
Pretreatment
Decision tree algorithm
Cloud storage
Progressive fusion
Internet of things with access costs
Terminal load distribution
system design
Power supply module
Internet of things data division module
Internet of things terminal node data placement decision module
Codec module
System network communication module
Discrimination of I/O mode
Classification of I/O modes
Development of placement strategy
Six elements
data size 1
data type 2
Life cycle 3
Creation time 4
Last creation time 5
Last modification time 6
Progressive fusion
Load distribution storage node
Load node weight calculation
Load migration algorithm
Migration basis
Terminal node load balancing
Interact with people

Figure 1: Principle block diagram of Internet of Things terminal node load balancing cloud storage data distribution system considering access cost.

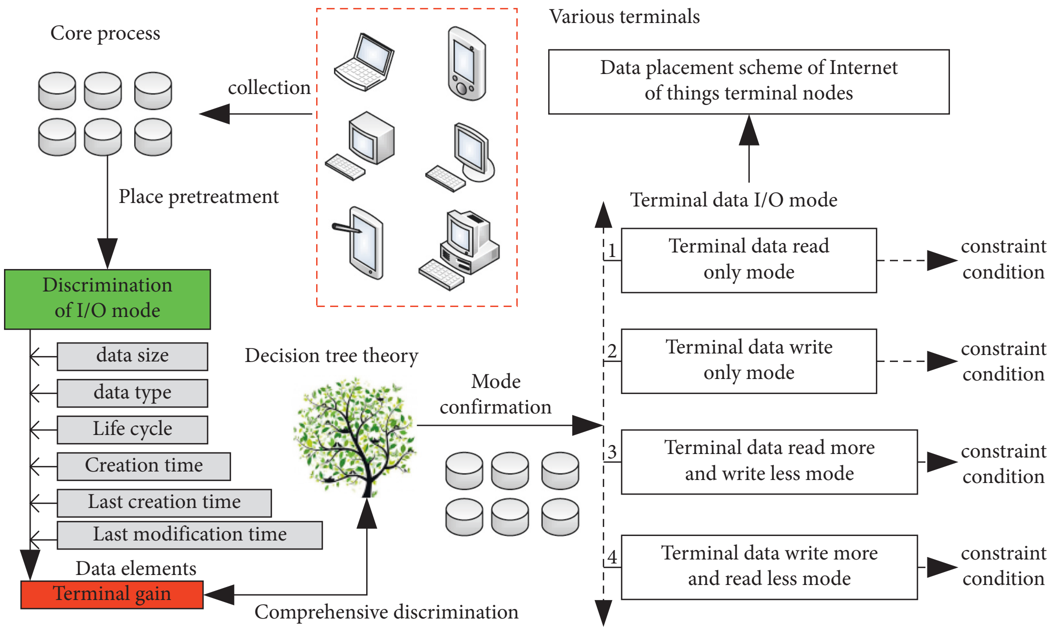


Figure 2: Principle block diagram of complex perceptual data placement algorithm.

Step 4: continuously collect the data to be placed, collect its corresponding six eigenvalues, call the corresponding decision tree IoT terminal node data I/O mode classification model, and constantly put new prediction results into the model so as to improve the accuracy of the model until all IoT terminal node data can be finalized.

After being processed by the I/O mode discrimination and classification algorithm of IoT data in the decision tree, it enters the link of the adaptive data placement algorithm. This algorithm is mainly based on the I/O mode type of IoT data to further determine its corresponding placement strategy and placement scheme. Based on this assumption, the read more write less mode is mode 1, the read less write multimode is mode 2, the read-only mode is mode 3, and the write-only mode is mode 4. Take the read more write less mode and read less write more mode as case modes, and formulate their corresponding placement strategies.

The placement strategy objective function corresponding to the read more write less mode is shown in formula (4). This mode has large downlink traffic and is sensitive to the read delay of the corresponding IoT data. The corresponding data placement constraints in this mode are shown in formulas (5)–(7) and (8). In the corresponding formula, A represents the cost proportion threshold and C represents the downstream data cost.

$$S = \min\left[\frac{(L * G_1(\text{mode1}))}{(A * d * w)}\right], \tag{4}$$

$$L * G_1(\text{mode1}) = \max\frac{\left(C_1^4 * \lg_1 + \cdots + C_i^4 * \lg_i\right)}{m_i, \quad 1 < i < n}, \tag{5}$$

$$A_1 = \left(\prod i \in N_i(1 - a_1) * \prod i \in N_i(a_1) + \cdots + \prod i \in N_i(1 - a_i) * \prod i \in N_i(a_i)\right), \tag{6}$$

$$A_1^s + A_1^g + A_1^T < a * A_1, \tag{7}$$

$$A_1^T = \left(a_1^4 * P_{\eta_1}^T * G_{d_1}\right) + \cdots + \left(a_1^4 * P_{\eta_i}^T * G_{d_i}\right). \tag{8}$$

The placement strategy objective function corresponding to the read less write multimode is shown in formula (9). This mode is mainly relatively sensitive to write delay. The corresponding data placement constraints in this mode are shown in formulas (10)–(12) and (13).

$$S = \min\left[\frac{(L * G_2(\text{mode2}))}{(A * d * w)}\right], \tag{9}$$

$$L * G_2(\text{mode2}) = \max\frac{\left(C_1^2 * \lg_1 + \cdots + C_i^3 * \lg_i\right)}{m_i, \quad 1 < i < n}, \tag{10}$$

$$A_2 = \left(\prod i \in N_i(1 - a_1) * \prod i \in N_i(a_1) + \cdots + \prod i \in N_i(1 - a_i) * \prod i \in N_i(a_i)\right), \tag{11}$$

$$A_2^s + A_2^g + A_2^T < a * A_2, \tag{12}$$

$$A_2^T = \left(a_1^2 * P_{\eta_1}^T * G_{d_1}\right) + \cdots + \left(a_1^2 * P_{\eta_i}^T * G_{di}\right). \tag{13}$$

Through the confirmation of the above data placement scheme, the data sorting and distribution of IoT terminal nodes can be further confirmed so as to further optimize the data placement of the system.

In summary, based on the above algorithm, the heterogeneity of IoT terminal node data can be fully considered so as to realize the rationality of IOT terminal node data placement considering access cost and then prepare for load balancing.

3.2. Terminal Node Load Balancing Strategy. The terminal node load balancing strategy proposed in this section mainly includes load distribution, storage nodes, and load migration. The principle block diagram of the corresponding load balancing strategy is shown in Figure 3. From Figure 3, we can see the load distribution, storage nodes, and load migration of the entire wireless sensor network.

In data node allocation, the weight of terminal node data needs to be calculated first. The corresponding calculation formula is shown in formula (14). The corresponding S represents the data state of the node, the corresponding a represents the heterogeneous weight of the data node, W_s represents the storage space used by the corresponding node data, and W_n represents the number of IoT nodes currently allocated. When the corresponding data weight is larger, the comprehensive available resources of the corresponding data node are smaller.

$$W = \frac{(S * W_s)}{(W_T * A) * W_n}. \tag{14}$$

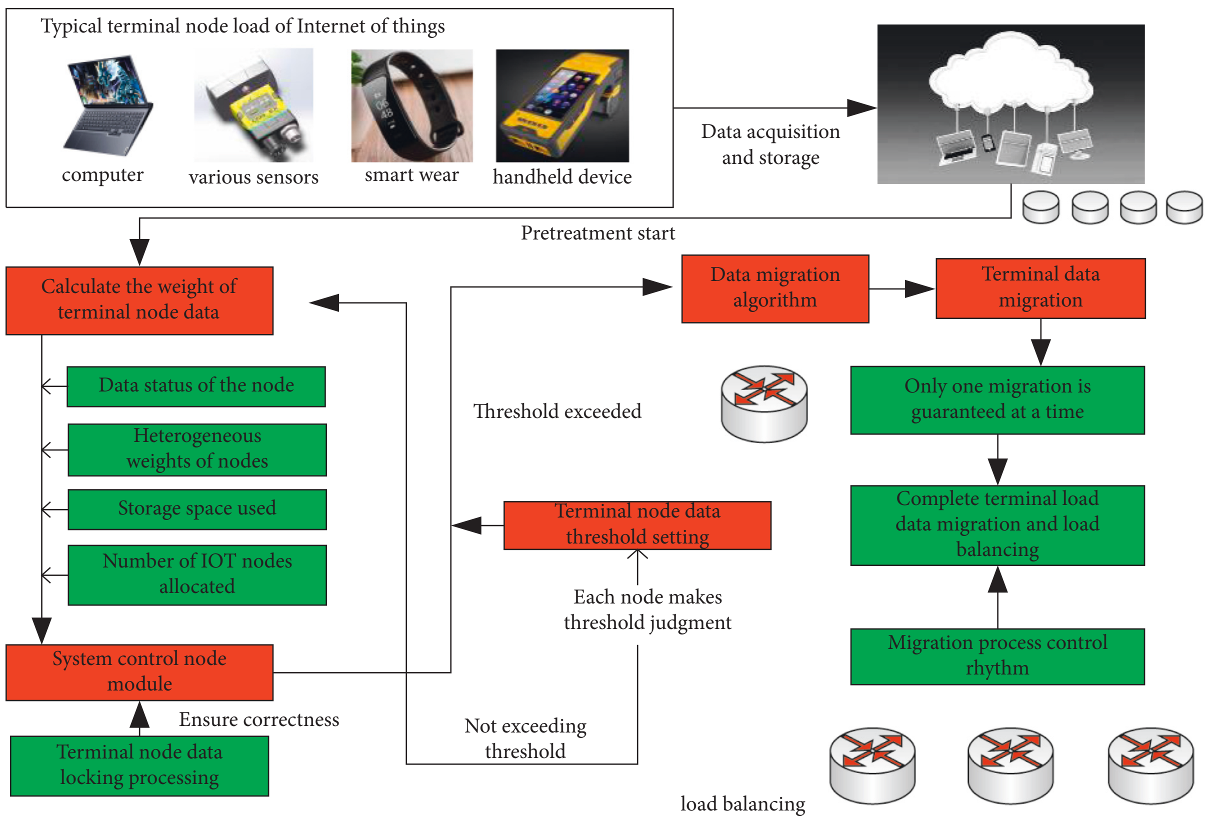

Typical terminal node load of Internet of things
computer
various sensors
smart wear
handheld device
Data acquisition and storage
Pretreatment start
Calculate the weight of terminal node data
Data status of the node
Heterogeneous weights of nodes
Storage space used
Number of IOT nodes allocated
System control node module
Ensure correctness
Terminal node data locking processing
Data migration algorithm
Terminal data migration
Threshold exceeded
Only one migration is guaranteed at a time
Complete terminal load data migration and load balancing
Migration process control rhythm
Terminal node data threshold setting
Each node makes threshold judgment
Not exceeding threshold
load balancing

FIGURE 3: Principle block diagram of terminal node load balancing algorithm.

The corresponding data node allocation process is shown in Figure 4. It can be clearly seen from Figure 4 that each storage node corresponding to the wireless sensor network needs to continuously calculate the heterogeneous weight of its corresponding data node and feed the data back to the corresponding control node of the wireless sensor network in real-time. At the same time, the data also cover key information such as the storage space used by data nodes and the number of IoT nodes allocated. In order to ensure the correct rate of the algorithm of the entire system, the system is implemented with a locking scheme when multiple IoT data nodes are concurrent.

After completing the data node allocation, carry out the data migration operation. First, set the data load threshold as *Wy*, judge whether the load of IoT terminal nodes is too heavy based on this threshold, and carry out relevant migration actions. When the load weight of the corresponding cloud storage system is far less than the load threshold and the weight of the corresponding single load node is greater than the load weight of the whole system, notify the system to trigger the corresponding data migration algorithm for data migration. When the corresponding system load weight is much greater than the load threshold, notify the system to add the corresponding data storage node and notify the system that the load is too heavy at this time. In the actual data migration process, in order to ensure the correctness of data migration, the system needs to ensure that only one data migration operation is allowed at the same time, and the system needs to control the migration rhythm.

3.3. Design of Cloud Storage Data Distribution System for Load Balancing of Internet of Things Terminal Nodes Considering Access Cost. The design framework of the cloud storage data distribution system for load balancing of IoT terminal nodes considering access cost is shown in Figure 5. It can be seen from the figure that the corresponding hardware system mainly includes IoT data division module, IoT terminal node data placement decision module, system network communication module, and IoT terminal node data encoding and decoding module.

At the design level of the corresponding IoT terminal node data division module, it mainly realizes the I/O mode division of the placed data. The module includes the data I/O model determination strategy, and the corresponding implementation method realizes the data type division model based on the decision tree through scikit-learn. The corresponding data placement decision module of the IoT terminal node is mainly a module to formulate the corresponding data placement strategy, which includes the data placement strategy algorithm. The corresponding network communication module is mainly to realize the interaction between multicloud storage, provide the function of cloud storage to obtain data and services from the upper server, provide the function of independent search and modification and other related operations of the IoT terminal, and provide the function of obtaining and monitoring cloud storage operators. The IoT terminal node data encoding and decoding module mainly helps restore relevant data. This module encodes the IoT

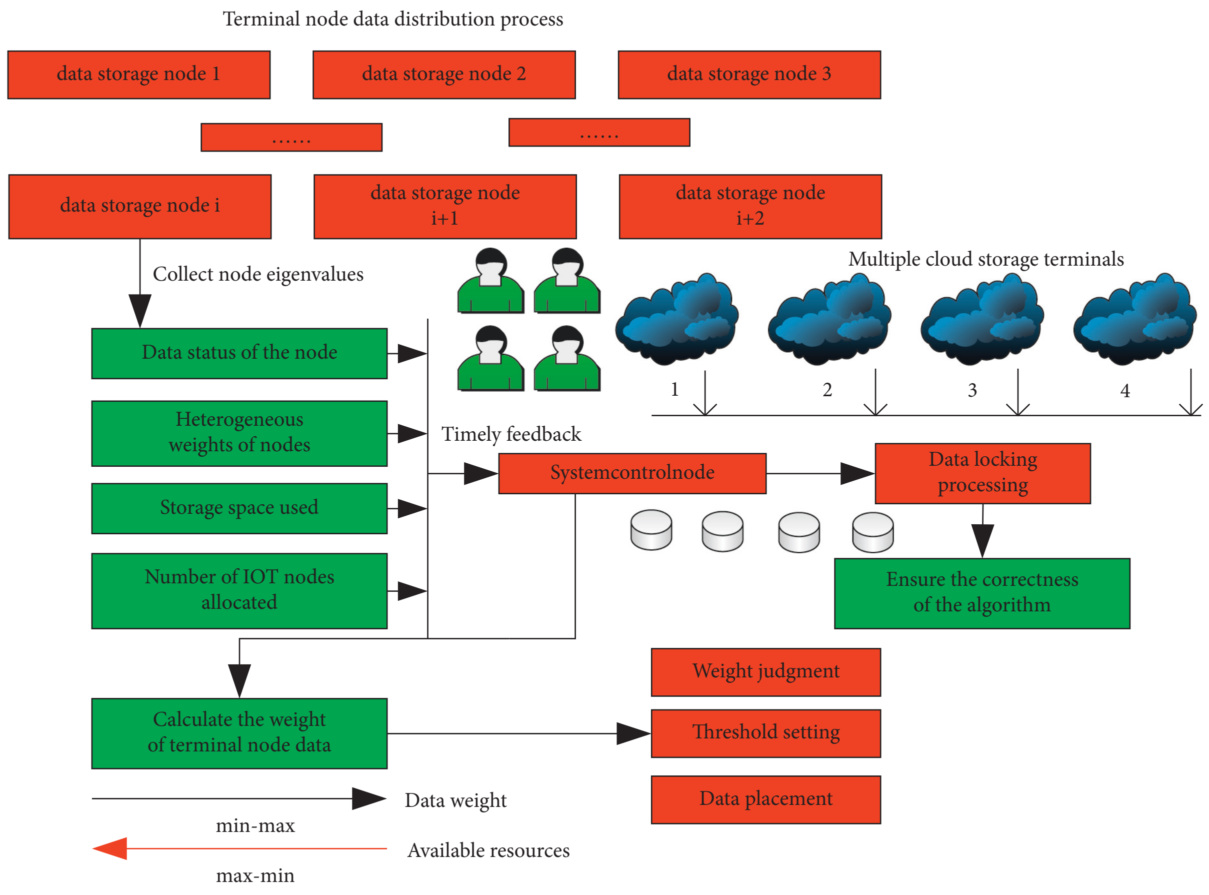


FIGURE 4: Principle block diagram of terminal node data distribution process.

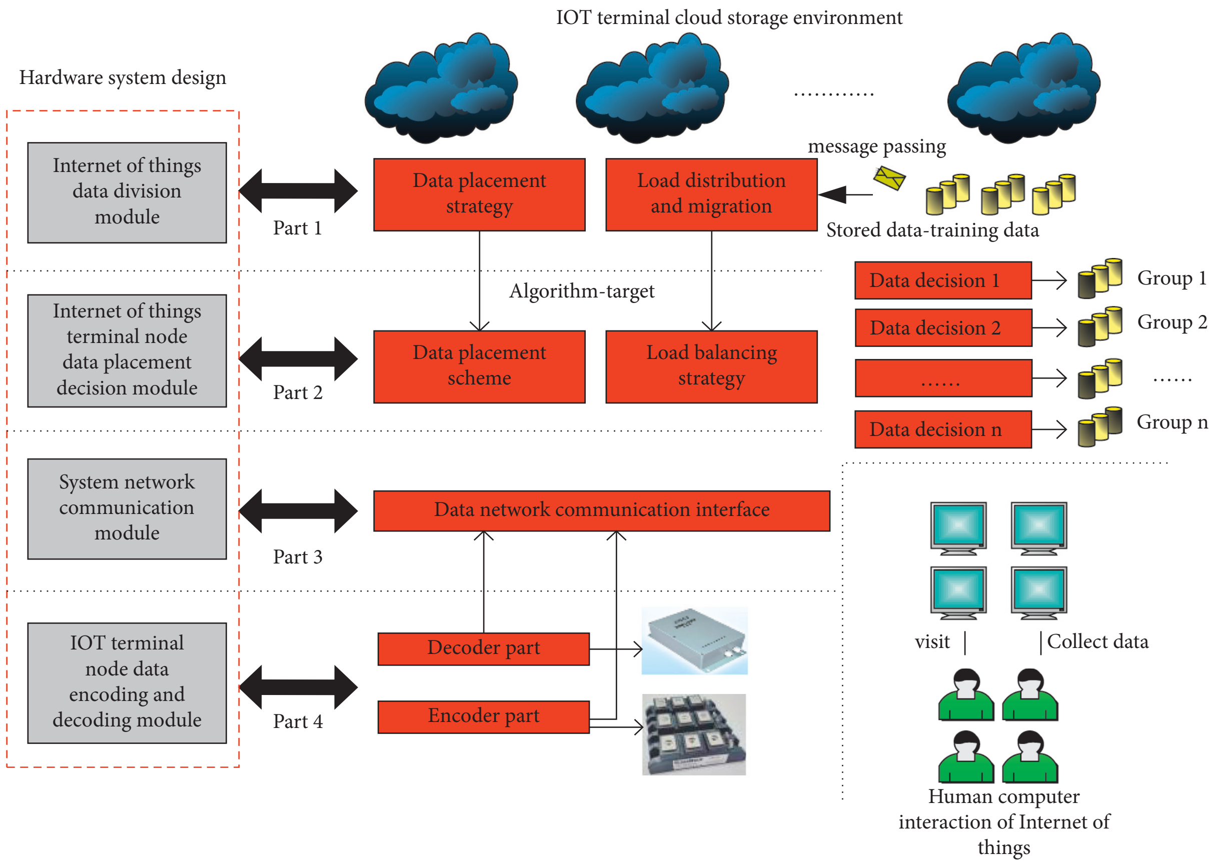

IOT terminal cloud storage environment
Hardware system design
....
message passing
Internet of things data division module
Part 1
Data placement strategy
Load distribution and migration
Stored data-training data
Algorithm-target
Data decision 1
Group 1
Data decision 2
Group 2
......
......
Data decision n
Group n
Internet of things terminal node data placement decision module
Part 2
Data placement scheme
Load balancing strategy
System network communication module
Part 3
Data network communication interface
IOT terminal node data encoding and decoding module
Part 4
Decoder part
Encoder part
visit
Collect data
Human computer interaction of Internet of things

FIGURE 5: Design framework of cloud storage data distribution system for load balancing of Internet of Things terminal nodes considering access cost.

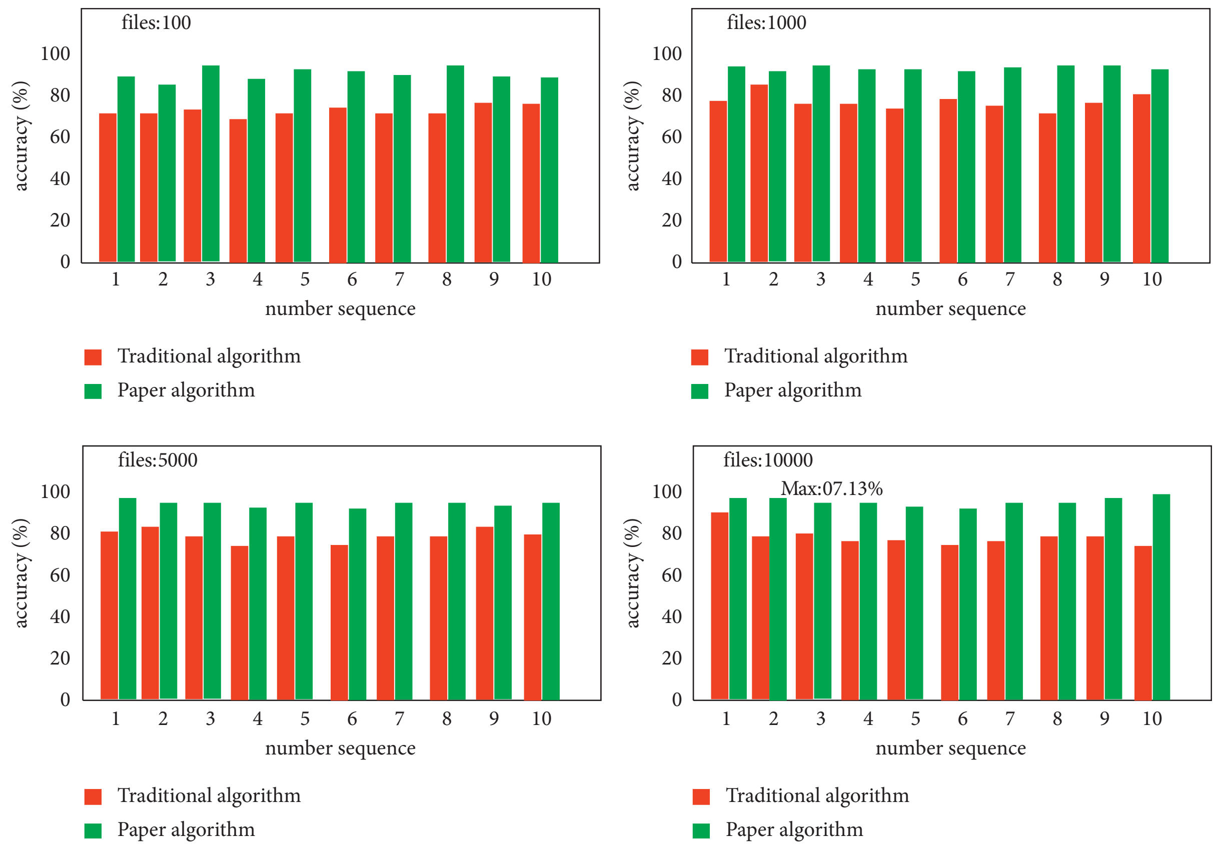


FIGURE 6: Column chart of data pattern division accuracy experiment.

terminal data through the encode() function and decodes the relevant IoT terminal data through the decode() function.

4. Experimental Verification and Data Analysis

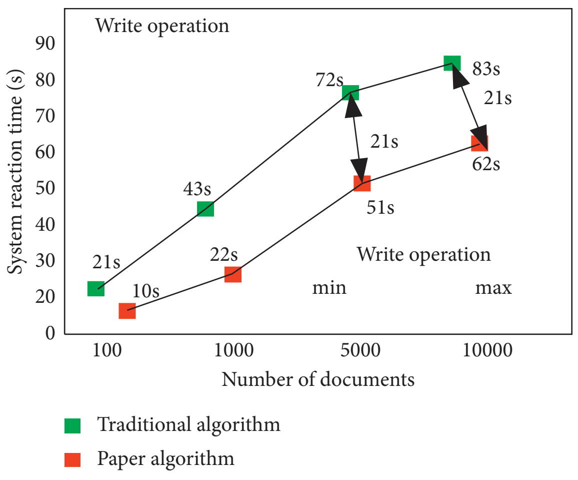
In order to further verify the advantages of the actual system and the corresponding data distribution algorithm, this article is verified by the performance comparison experiment with the traditional distribution algorithm.

In the data mode division accuracy experiment, this article selects 100, 1000, 5000, and 10000 IoT terminal node files for comparative experiments. The corresponding experimental data are shown in Figure 6. It can be seen from the figure that the data distribution strategy proposed in this article is improving with the increasing number of files. When the number of files reaches 10000, the corresponding data partition accuracy is improved to 97.13%, which has obvious advantages over the traditional data distribution strategy.

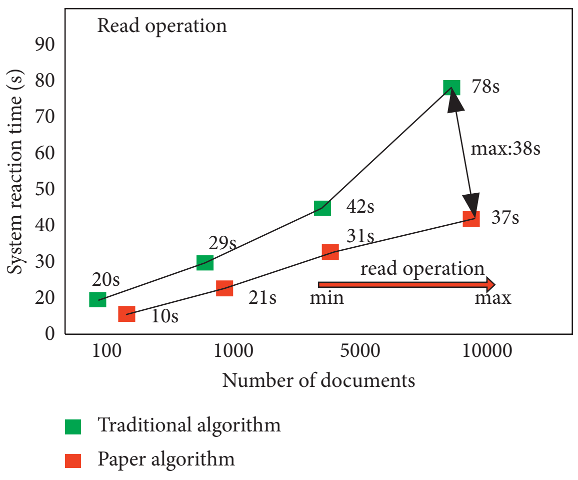
Verify the response speed advantages of the distribution algorithm at the read, write, store, and delete levels when the number of files corresponds to 100, 1000, 5000, and 10000. The corresponding experimental results are shown in Figures 7(a), 7(b), and 7(c). As can be seen from the figure, at the corresponding write operation level, the write speed of the algorithm proposed in this article is about 15% higher than that of the traditional algorithm. At the corresponding read operation level, the write speed of the proposed algorithm is about 20% higher than that of the traditional algorithm. At the corresponding storage operation level, the writing speed of the proposed algorithm is about 17% higher than that of the traditional algorithm. At the corresponding level of delete storage operation, the writing speed of the proposed algorithm is about 17% higher than that of the traditional algorithm. Based on the above data, it can be seen that the performance of the algorithm proposed in this article has been comprehensively improved compared with the traditional algorithm at the basic operational level.

In order to verify the advantages of this algorithm in the level of load balancing, the comparative verification is carried out when the number of files of IoT terminal nodes is 100, 1000, 5000, and 10000, and the corresponding load number of each storage node in the system is observed in the experiment. Under different file numbers, the corresponding load distribution is shown in Figure 8. It can be seen from the figure that the load distribution of the distribution algorithm proposed in this article is relatively uniform in each memory. At the same time, with the increase of the number of files, the number of files between the corresponding memories fluctuates little, while the corresponding load is seriously unbalanced under the corresponding traditional distribution strategy.

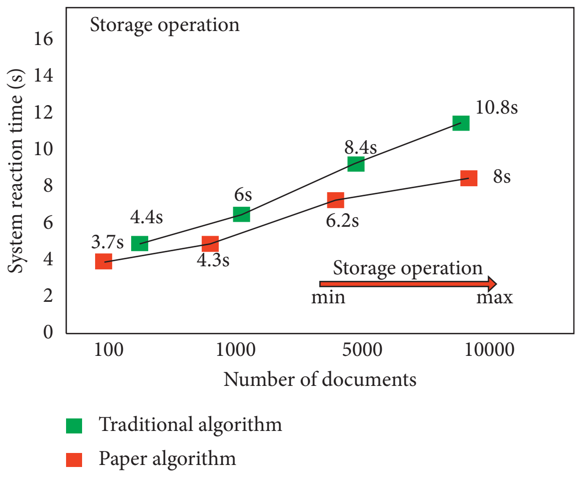
In order to verify the superiority of the algorithm proposed in this article in system data access efficiency, a comparative experiment is carried out based on the traditional distribution strategy. The experimental results are shown in Figure 9. It can be seen from the figure that the



(a)



(b)



(c)

Figure 7: (a) Write operation level performance comparison column chart. (b) Read operation level performance comparison column chart. (c) Storage operation level performance comparison column chart.

distribution strategy proposed in this article has the highest system data access efficiency of 98.3%, which is nearly 20% higher than the traditional distribution algorithm. Therefore, this algorithm has obvious advantages over traditional algorithms.

The above analysis and experimental results further show that the algorithm proposed in this article has obvious advantages over the traditional algorithm in the cloud storage data system of load balancing of IoT terminal nodes considering access cost, and it can realize the optimization of data placement and load balancing. This greatly improves the accuracy of the system's data mode division of IoT terminal nodes and further improves the data access efficiency of the system.

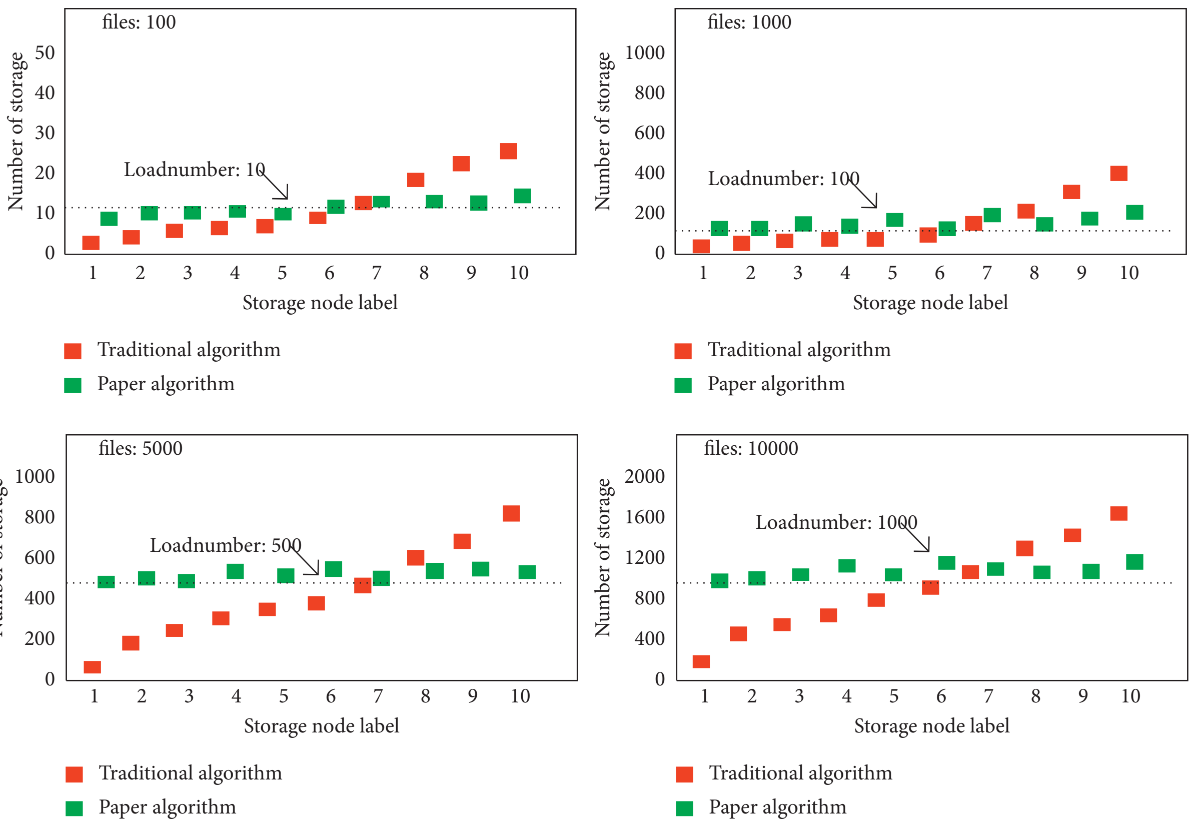


Figure 8: Broken line diagram of load distribution of each storage node under different numbers of files.

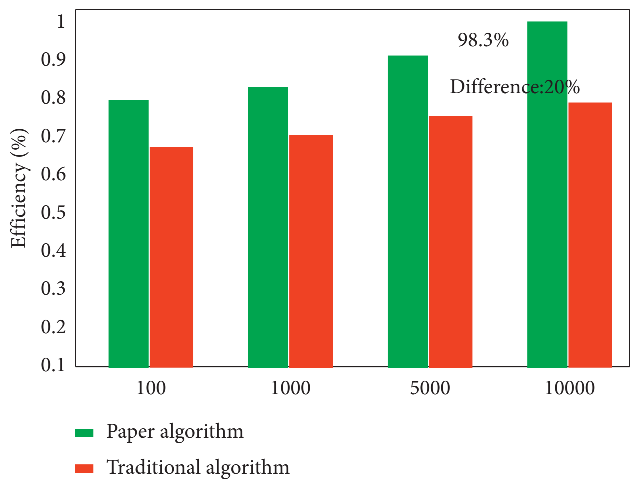


Figure 9: Comparison of system data access efficiency under different numbers of files.

5. Conclusion

This article mainly analyzes and studies the relevant research algorithms of cloud storage data distribution system with load balancing of IoT terminal nodes considering access cost and analyzes and compares their advantages and disadvantages. Based on the existing problems, this article proposes a complex sensing data placement algorithm based on the cloud storage distribution of IoT terminal nodes and realizes the accurate division of I/O mode of IoT data by reasonably configuring hybrid sensing algorithm and adaptive sensing algorithm so as to further optimize the performance of IoT data storage system considering access cost. Aiming at the problem of terminal node load balancing, this article innovatively proposes the terminal node data sorting and distribution algorithm. Through the node data sorting and distribution algorithm, the accurate segmentation of the IoT data to be processed is realized so as to further accelerate the data reading and processing speed. At the same time, a periodic load balancing algorithm is added to the terminal node data sorting and distribution algorithm. Thus, the problem of unbalanced node data layout is further solved. Based on the proposed algorithm, this article designs a load balancing cloud storage data distribution optimization system of IoT terminal nodes considering access cost and carries out experimental verification in a real environment. The experimental results show that the data pattern division accuracy corresponding to the distribution strategy proposed in this article is improved to 97.13% and the corresponding data access efficiency is improved to 98.3%, compared with the traditional distribution strategy. The data distribution strategy proposed in this article has obvious performance advantages and further promotion value. In the follow-up research, the article will focus on analyzing the energy consumption of wireless sensor networks. At the same time, it will further optimize the load balancing situation in the load balancing strategy. Moreover, it will further discuss the load balancing strategy problem of the traditional cloud storage network considering the cost.

Data Availability

The data used to support the study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Quezada-Naquid, R. Marcelín-Jiménez, and J. L. Gonzalez-Compeán, "RS-Pooling: an adaptive data distribution strategy for fault-tolerant and large-scale storage systems," *The Journal of Supercomputing*, vol. 72, no. 2, pp. 1–21, 2015.

[2] M. Li, J. Zhang, J. Wan et al., "Distributed machine learning load balancing strategy in cloud computing services," *Wireless Networks*, vol. 26, no. 8, pp. 5517–5533, 2020.

[3] J. Huang, W. Liu, and Y. Su, "Load balancing strategy and its lookup-table enhancement in deterministic space delay/disruption tolerant networks," *Advances in Space Research*, vol. 61, no. 3, pp. 811–822, 2017.

[4] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2374–2385, 2016.

[5] J. Kumar and A. Singh, "Secure and energy aware load balancing framework for cloud datacenter networks," *Electronics Letters*, vol. 55, no. 9, pp. 11–23, 2019.

[6] A. Ghasemi and A. T. Haghighat, "A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning," *Computing*, vol. 102, no. 9, pp. 2049–2072, 2016.

[7] A. Cucchetti, C. Ricci, G. Ercolani et al., "Efficacy and cost-effectiveness of immediate surgery versus a wait-and-see strategy for sporadic nonfunctioning T1 pancreatic endocrine neoplasms," *Neuroendocrinology*, vol. 101, no. 1, pp. 25–34, 2015.

[8] C. T. Yang, W. C. Shih, and C. L. Huang, "On construction of a distributed data storage system in cloud," *Computing*, vol. 98, no. 1, pp. 93–118, 2016.

[9] J. P. B. Mapetu, L. Kong, and Z. Chen, "A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing," *The Journal of Supercomputing*, vol. 77, no. 6, pp. 5840–5881, 2021.

[10] S. S. Sefati, M. Mousavinasab, and R. Z. Farkhady, "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation," *The Journal of Supercomputing*, vol. 11, no. 3, pp. 1–25, 2021.

[11] J. Prassanna and N. Venkataraman, "Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud," *Wireless Networks*, vol. 34, no. 3, pp. 2313–2322, 2019.

[12] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "MuR-DPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2015.

[13] J. K. Liu, K. Liang, W. Susilo, J. Liu, and Y. Xiang, "Two-factor data security protection mechanism for cloud storage system," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1992–2004, 2016.

[14] Z. Yin, X. Li, Z. Yao et al., "Divergences in reproductive strategy explain the distribution ranges of Vallisneria species in China," *Aquatic Botany*, vol. 132, no. 3, pp. 41–48, 2016.

[15] A. Hussain, M. Aleem, M. A. Iqbal, and M. A. Islam, "SLA-RALBA: cost-efficient and resource-aware load balancing algorithm for cloud computing," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 6777–6803, 2019.

[16] S. Vazquez, P. Acuna, R. P. Aguilera, J. Pou, J. I. Leon, and L. G. Franquelo, "DC-link voltage-balancing strategy based on optimal switching sequence model predictive control for single-phase H-npc converters," *IEEE Transactions on Industrial Electronics*, vol. 4, no. 19, pp. 11–19, 2019.

[17] A. Tchernykh, M. Babenko, N. Chervyakov et al., "AC-RRNS: anti-collusion secured data sharing scheme for cloud storage," *International Journal of Approximate Reasoning*, vol. 102, no. 11, pp. 60–73, 2018.

[18] Y. Li and Y. Wang, "Synergistic strategy for the geographical traceability of wild Boletus tomentipes by means of data fusion analysis," *Microchemical Journal*, vol. 140, no. 34, pp. 38–46, 2018.

[19] B. Hwa, Z. Yuan, and G. Kce, "Functional broadcast encryption with applications to data sharing for cloud storage," *Information Sciences*, vol. 502, no. 78, pp. 109–124, 2019.

[20] A. A. Periola, O. A. Osanaiye, and A. T. Olusesi, "Future cloud: spherical processors for realizing low-cost upgrade in underwater data centers," *The Journal of Supercomputing*, vol. 32, no. 2, pp. 1–27, 2021.

[21] K. Karmakar, R. K. Das, and S. Khatua, "An ACO-based multi-objective optimization for cooperating VM placement in cloud data center," *The Journal of Supercomputing*, vol. 91, no. 9, pp. 1–29, 2021.

[22] Z. Zhou, P. Q. Tran, and K. Kieft, "Genome diversification in globally distributed novel marine Proteobacteria is linked to environmental adaptation," *The ISME Journal*, vol. 141, no. 8, pp. 1–18, 2020.

[23] J. Wu, L. Shen, and L. Liu, "LSH-based distributed similarity indexing with load balancing in high-dimensional space," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 636–665, 2020.

[24] R. Regaieg, M. Koubàa, and Z. Ales, "Multi-objective optimization for VM placement in homogeneous and heterogeneous cloud service provider data centers," *Computing*, vol. 11, no. 3, pp. 1–25, 2021.

[25] M. Riahi and S. Krichen, "A multi-objective decision support framework for virtual machine placement in cloud data centers: a real case study," *The Journal of Supercomputing*, vol. 74, no. 7, pp. 2984–3015, 2018.