Research Article

Intelligent Data Processing and Optimization of University Logistics Combined with Block Chain Storage Algorithm

Li Li

Wenzhou University of Technology, Wenzhou 325035, China

Correspondence should be addressed to Li Li; 00011083@wzut.edu.cn

Received 27 May 2022; Accepted 28 June 2022; Published 18 July 2022

Academic Editor: Qiangyi Li

Copyright © 2022 Li Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The traditional Hadoop-based logistic data processing platform of universities cannot fully collect and control remote data in the process of data processing and has defects such as poor efficiency, high error, and difficult query. In existing designs, devices often need to store complete block data. When retrieving or verifying specific data on the chain, a large number of blocks need to be traversed to find the corresponding data, which reduces the response speed on the user side. In addition, the traditional consensus algorithm is not suitable for resource-constrained terminal devices. Therefore, this paper proposes an intelligent data processing and optimization method of university logistics combined with block chain storage algorithm. For unstructured data on the cloud storage network, the storage information can be obtained through domain analysis, and the storage frequency can be dynamically adjusted based on the estimated data flapping. Moreover, a data storage audit scheme is designed on the cloud storage network to improve data storage efficiency. In order to ensure the integrity verification efficiency of unstructured big data cloud storage, a multilayer block chain network model suitable for university logistics data is designed by introducing block chain technology. The throughput and average occupancy under the same conditions in different environments are evaluated and analyzed, and it is verified that the proposed method is not affected by the initial priority distribution and has good stability and optimization performance. The results show that the proposed method can significantly improve the cloud storage efficiency of unstructured data, can effectively respond to a large number of requests to access data, and has good big data processing capability.

1. Introduction

The era of “Internet plus” has changed the way of social life and the way of life and learning of teachers and students in universities, becoming an opportunity for innovation and reform of logistics management services in universities [1]. “Internet plus” technology can realize the sharing of data and information, and handle various matters on the Internet platform [2]. The improvement of social life quality and the transformation of lifestyle will inevitably have an impact on the way of life of universities. In the current situation of increasingly pursuing the quality of life, backward traditional logistic management services of universities cannot meet the needs of teachers and students, so it is imperative for universities to seek innovative reform of logistic management services [3]. If you want to be a beneficiary of technological change, you need to actively explore the combination of technology and reality, open mind, and dare to practice. At present, with the popularization of intelligent terminals, the establishment of big data platforms, and the development of computer technology, the full coverage of 4G signals will help universities to realize the innovation of logistics management services [4].

With the full coverage of 4G signals and the coming of 5G signal era, the convenience of the wireless network era is changing people’s daily life and also providing a foundation for information exchange and sharing [5]. Most of the management work of universities has been established on the basis of the Internet. The establishment of large database will lay a foundation for overall management and resource integration, help to improve the quality of service and management efficiency, and indirectly improve the efficiency of students’ learning and teachers’ teaching [6].
As there are more and more data in the logistic intelligent data processing platform of universities, in the case of massive data, the current logistic intelligent data processing platform of universities cannot guarantee security, stability, and efficiency [7]. The rich application scenarios of block chain are suitable for the application between different block platforms of logistics management in universities. It can effectively solve the security and integrity of the information of the logistic intelligent data processing platform in universities and realize the trust and consistency of assistance between multiple subjects.

Block chain is a chained data structure consisting of distributed data blocks with valid transactions and a decentralized shared ledger that is cryptographically guaranteed to be tamper proof and unfalsifiable [8]. The essence of block chain is an Internet of value. As a new, low-level science and technology, block chain has the characteristics of safety and openness, decentralization, stability, and so on.

Through network connection, the block chain system adopts consensus-based norms and protocols, and the users implement the point-to-point distributed consensus mechanism to collect and verify data information through their own ports [9]. Data entry through each point records the trace of data entry. These input data are added to the block chain through verification to protect transaction data, address, identity, and other types of sensitive information to ensure the security, stability, permanence, immutability, and traceability of the data. At the same time, all parties using the data can query and obtain the block chain data they need through the open port to realize the transparency and openness of the data.

Immutability is one of many features of block chain, which means that data in a historical block are difficult to change once it has been identified. Immutability ensures the integrity and reliability of the block chain’s historical data, but it also prevents the block chain from revising questionable historical data. Because most of the current mainstream block chains are decentralized ledgers with increasing amounts of data, this has a negative impact on system resources and availability. Therefore, an editable improvement of on-chain data is needed to fit the existing block chain architecture.

In traditional networks, data are mostly transmitted and stored in structured forms. With the development of cloud computing networks, a large number of unstructured data emerge [10]. Unstructured means that there is no fixed format and no convention rules. This kind of data is not easy to process uniformly and is not conducive to understanding. According to IDC’s statistical report, the annual growth rate of such irregular data exceeds 60% and will reach 40 ZB in 2020 [11]. Currently, Google processes more than 20 PB of web pages per day, while Facebook stores more than 15 TB of data per day [12]. The daily unstructured data generated by some laboratories and educational institutions are also in the TB or PB level, which is mainly composed of articles, web pages, audio, and video. Relying on cloud computing, unstructured data are developing toward mass, and its storage burden and storage cost are also rising sharply. At the same time, the persistence and operation of unstructured big data also have strict reliability and delay requirements. In the study of optimal storage, redundancy persistence mechanism is adopted to ensure reliability. In terms of improving timeliness, it is very difficult to expand the existing storage mechanism directly to unstructured because it adopts unified structure processing, so it is urgent to optimize the unstructured storage mechanism and storage method [13]. Hadoop cluster parallel storage is the mainstream persistence framework at present, but data cleaning and storage scheduling strategies need to be deployed on this basis. Some scholars [14] designed storage schemes related to node types and adopted differential storage for data according to node types and corresponding popularity. This storage mechanism helps reduce the transmission distance and access delay of cloud storage data, but does not optimize the processing of unstructured data. Some scholars [15] proposed UDSS storage and integrated nonrelational database with other components through the expanded query mechanism of MongoDB or NoSQL database. In addition, unstructured data are vulnerable to damage in the cloud storage process due to irregularity. Therefore, data audit is usually performed. At present, three-party audit is mostly used, which has some problems such as single point failure and time cost increase.

Considering that block chain network has good decentralization and expansion ability, it can replace three-party audit and improve interaction speed. In view of this, this paper introduces the multilayer block chain network model and proposes a method of intelligent data processing and optimization of university logistics combined with block chain storage algorithm. Data integrity is verified in the block chain network to further improve the cloud storage performance of unstructured big data. Experiments show that this model can effectively reduce the amount of block chain model data that edge nodes need to store, thus accelerating the speed of data retrieval.

The main contributions of this paper are as follows:

1. The cloud storage network uses F2 domain to obtain storage information, judge the data status according to the domain head, and update storage policies in real time.
2. Introduce storage audit policies to the cloud storage network to determine storage audit requirements based on data popularity and damage, and audit the storage duration and data packet quantity of stored data to optimize storage efficiency.
3. Designed the multilayer block chain network structure and realized the efficient verification of data integrity based on Merkle tree and Hash.

This paper consists of four main parts: the first part is the introduction, the second part is methodology, the third part is result analysis and discussion, and the fourth part is conclusion.

2. Methodology

2.1. Safe Storage and Fast Retrieval Based on Block Chain

Based on the multilayer block chain network model, a variety of services related to data storage and data retrieval are
provided on each layer for various nodes in the network to call. The basic service is the safe storage and retrieval of data. In order to achieve these two functions, smart contracts in the block chain are needed to accomplish them. At the same time, it also introduces the checking mechanism of edge nodes to the terminal side block chain and the detailed calculation formula of the adaptive workload proof algorithm.

Data storage smart contracts provide secure data storage services for terminals at the terminal layer. Data storage smart contract is deployed on the terminal side block chain. After the terminal device registers on the edge-node set in the current local network, the smart contract can be invoked to complete the secure storage of data. The original data collected by the terminal device and the corresponding index data information are stored on the data server and the terminal side block chain, respectively. In the terminal node side, the original data can be processed in advance, data segmentation, and other operations to achieve special applications. The index data information stored in the terminal side block chain contains the storage address of the original data for retrieving the original data. In addition, the index information also contains the time stamp, terminal node identification code, and other types of information.

The execution of a data storage smart contract is divided into three phases. The first stage is the preparation of the data index information. After the sensor on the terminal node collects data, the Message Digest Algorithm 5 (MD5) is used to calculate the corresponding MD5 value. The terminal node packages the MD5 value and other necessary information (hash value of original data, storage address, etc.) to generate the data index information RawJson. RawJson contains the MD5 value of original data, version number of original data, storage address of multiple original data storage sources, and storage timestamp information. RawJson is abstract information serialized and stored in JSON format. The storage addresses of multiple original data stores indicate that multiple copies of original data can be generated and stored on multiple trusted storage nodes to prevent original data loss due to a node failure. The end node then signs RawJson, packages it into a transaction, and invokes the data storage smart contract on the end-side block chain as an input parameter.

The second stage is the execution stage of data storage smart contract. In this phase, the data storage smart contract checks the entered transaction content to confirm that the signature node of the transaction has been registered with the edge node in the current local network. The smart contract then returns a signed message to the end node containing the transaction address to execute the contract. The current contract execution process enters a wait state, waiting for a wake-up message from the data server.

The third stage is the storage of raw data. The data collected by the end node are stored in the data server. The data server is the data storage source selected by the end node itself, ensuring that the private data itself are only owned by the end node and the data server trusted by the end node. After the data storage of the terminal node is successful, the data server will send a storage certificate containing its signature to the smart contract executed in the second stage to wake up the smart contract. The data storage smart contract continues to verify that the storage certificate currently received matches the storage source address in the previously received transaction. If yes, the original data are successfully stored. The smart contract notifies the end node to perform an adaptive proof-of-work algorithm, reaching consensus in the local network and publishing the transaction to the end-side block chain.

In IOTA’s tangle network, every node is a transaction. For every transaction generated, another transaction will be added to the chain. Because there is no block generation time limit in traditional block chain applications, IOTA’s data throughput can be theoretically unlimited. However, in the early stage of the network, when there are few participants, the whole network is not robust enough and the security is poor. Therefore, in the current implementation of IOTA, relying on a centralized coordinator to ensure the security of the system can effectively prevent double flower attack and parasitic chain attack. The coordinator posts a milestone transaction every two minutes, validating all current pending transactions. Due to the topology of the DAG, each milestone transaction directly or indirectly validates all transactions in the current network. Thus, only transactions confirmed by milestone transactions are considered 100% confidence transactions, and the consensus of the entire network is based on the coordinator.

Edge nodes adopt a similar check strategy, which issues a “check transaction” at an interval of time T to coordinate edge nodes in the local network. The edge node checks the transactions published by all edge nodes in time T and calculates the contribution value of all edge nodes. The contribution value is used as an indicator to evaluate the behavior of the terminal nodes. Then, the contribution value is published in the terminal side block chain in the form of “checking transactions” for the terminal nodes to query when mining or checking transactions. The details of how transactions are checked by edge nodes are shown below.

The terminal node is the producer of the transaction data, which can calculate the index information of the original data and upload the original data locally. Then, the end node packages the stored credentials and index information into a transaction, runs the PoW algorithm, and publishes the transaction to the end-side block chain. In fact, the whole block chain on the terminal side only stores the index of data storage, and the storage relationship between data is relatively independent without causal relationship. So, there is no need for strict input/output script checks on every transaction like Bitcoin and Ethereum. Different from block chain financial applications, the system for the traditional double flower attack, parasitic chain attack, and other attacks does not have special consideration. The malicious behaviors of terminal nodes that edge nodes mainly consider are lazy behaviors of terminal nodes and the behaviors of releasing a large number of meaningless transactions of terminal nodes. The latter will increase the load on the network and eventually cause the system to crash. Edge nodes detect these malicious behaviors, calculate the contribution value of nodes according to the
contribution calculation formula, and publish it to the terminal side block chain in the form of “check transaction.” When the terminal node executes the PoW algorithm, it needs to query its contribution value from “check transaction” to obtain the difficulty value of the PoW algorithm. Here, the PoW algorithm whose difficulty value dynamically changes according to the contribution value of nodes is called adaptive PoW algorithm.

After the edge node finishes checking the transactions in the terminal side block chain for a certain period of time, the published “check transactions” will link all the “pending transactions” in the local network. All subsequent transactions published in the local network must link to this “check transaction.” A “check transaction” represents a bundle of transactions that occurs periodically in the end-side block chain and can reduce the structural complexity in the end-side block chain.

In addition to releasing “check transaction” on the terminal side block chain at an interval of time T, the edge node will also extract the transaction information from the transaction traversed during this check and package it into a complete block and publish it on the edge-side block chain. Edge-side block chain is a storage structure composed of multiple single chains. Each chain represents a summary of the terminal-side block chain in a local network, and each block on the chain represents a snapshot of the data of transactions published in that local network during a certain time period. Data sharing and data retrieval across different local networks are realized by edge-side block chain. The structure of edge-side block chain is introduced below.

Edge-side block chain organizes the transaction set of terminal side block chain into a Merkle tree. The leaf node of Merkle tree contains the data label information of the corresponding transaction and the hash pointer pointing to the corresponding transaction in the terminal side block chain. Hash pointers are meaningful only in the corresponding local network. Merkle trees divide transactions into pairs from left to right, and each group yields a hash value that serves as the parent of the group, layer by layer, and finally get a strong hash-associated binary tree. The block header holds the root hash of the Merkle tree. Due to the irreversibility of the hashing algorithm, all nodes in Merkle tree are guaranteed to be immutable. The model structure of Merkle tree is shown in Figure 1.

In the edge computing scenario, data are regional and time-sensitive. Therefore, edge computing network is divided into multiple LNS according to the communication topology, which ensures that most data interaction requests occur in the same LNS, and there is less data interaction between different LNS. Because of the isolation between LNS, it is not necessary for a single edge node to record the information of all LANS in the edge computing network, it only needs to save the information of LANS directly connected to it. We refer to the difference between the block content stored by the full node and the light node in Bitcoin. In the edge block chain of the multichain structure, the single chain is divided into “full chain” and “light chain” according to the integrity difference of the block on the single chain. Blocks on the “full chain” contain the block header and block body structure, while blocks on the “light chain” contain only the block header structure. The local network on which an edge node resides and the local network directly connected to the edge node are stored in the edge-side block chain locally stored by the edge node in the form of “full chain.” The single chain corresponding to other LANS is stored as a “light chain.” The storage structure of different levels not only reduces the data content that needs to be stored by edge nodes but also ensures that the content of block can be verified between different chains to prevent the block chain data from being tampered with.

Data retrieval in multilayer block chain network model can be divided into fuzzy query and precise query. Fuzzy query is not clear in advance to find the specific transaction entity, but through the label information contained in the block to determine whether the data are needed. If the data are needed, a data request is made. Precise search means that the node knows the hash value of the target transaction in advance and finds the specific transaction entity through the hash value. Fuzzy query and precise query are divided into cross-LAN data retrieval and intra-LAN data retrieval. Among them, if it is the data retrieval within the local network in the fuzzy query, the edge node first traverses the corresponding “whole chain” of the local network. Then, starting with the block, we go forward through the Merkle tree child nodes in the block body one by one to retrieve the data we are interested in. After retrieving the data, we directly follow the hash pointer to find the corresponding transaction.

If the data are retrieved across the local network in a fuzzy query, the situation is slightly more complicated. We assume that edge-node NR needs to retrieve a piece of data of interest on other edge-node NA that is not directly connected. At this point, the chain of edge-node NA in the edge block chain locally stored by edge-node NR is a “light chain”; that is, the block on the chain only contains the block header information. In the first step, the edge-node NR still needs to be positioned according to a rough time range from back to front to the last block near the edge of the time range. After the block scope is determined, the edge-node NR still needs to update the local chain data, that is, to synchronize the complete information of the block within this time range to the “full-chain” data holder of the local network. During this synchronization, the edge-node NR matches the root hashes contained in the local bulk with those of the block on the “full-chain” data holder. Then, starting with the block, you traverse until you find the data you want and then make a request to the edge node, the data owner of the chain, to share the data.

If the edge node knows the hash value of the transaction, the edge node will first submit the query request to the edge node in the local network at the exchange. Edge nodes determine the last checkpoint transaction created in the terminal-side block chain based on the storage time range and then start from that checkpoint transaction and traverse backward until the corresponding transaction is found. At this point, the requested edge node needs to find Merkle path in the block corresponding to the transaction on the edge-side block chain. The Merkle path is shown in Figure 2.
This Merkle path is used by the requested edge node to prove to the requesting edge node that the data content of transaction 4 is not fabricated, but is real. When the requesting edge node receives such a Merkle path, it performs a hash check to see whether the root hash stored in the block header of the corresponding block in the local “light chain” is the same as the root hash of the Merkle path. If no error is found, no data are tampered with in the Merkle path. The Merkle path and corresponding data obtained in this query are considered to exist and not tampered with.

Through this mechanism, edge nodes only need to save the block header chain of other edge nodes to verify the correctness of data. At the same time, the edge node can effectively reduce the traversal search time after sorting the transactions in the local DAG network (i.e., terminal-side block chain) into a single-chain block chain in chronological order.

In traditional block chain applications, there are usually two roles: light node and full node. The full node stores all block chain data, while the light node only stores block chain header information. Different from traditional block chain applications, each edge node in this system is both a full node and a light node. It can also provide data service query to other edge nodes as a whole node. Like the light node, it only needs to save the block chain header information of other edge nodes, so it can query and verify the transaction data of other edge nodes through the fast data verification mechanism. Assuming that the total amount of data in a block is about 1M and the size of the block header data is fixed at 80 bytes, the storage space saved by each edge node is about 13,000 times that saved by the traditional block chain structure.

Figure 1: Model structure of Merkle tree.

Figure 2: Merkel path.

2.2. Big Data Cloud Storage Design. According to the distribution and preference differences in the cloud environment, the data scheduling equation is described as follows:

\[ w_{xy} = m_1 g + m_2 h + m_3 c + m_4 d. \]  \hspace{1cm} (1)

Here, \( m_1 \sim m_4 \) are the weighting coefficients, and their sum is 1. Item \( g \) is the cost of data packet sampling time in cloud storage, item \( h \) is the degree to which data packets meet the requirements, item \( c \) is storage quality, and item \( d \) is storage consumption. We assume that the total amount of unstructured data faced by the cloud environment is \( t \), and the corresponding persistence set is \( (1, 2, \ldots, t + g) \), \( g > 0 \). This results in arbitrary stored data, as shown below:

\[ j_x = \lambda x [i_1, i_2, \ldots, i_t], \]  \hspace{1cm} (2)

where \( \lambda x \) is a row vector in the field \( F_2 = \{0, 1\} \). \( I_x \) is data packet. To ensure the uniqueness of data packets, the
elements of vector $\lambda_x$ should conform to the following distribution rules.

$$U_x(\lambda_{xy} = s) = \begin{cases} \frac{\mu_{tt}}{t}, & s = 1, \\ 1 - \frac{\mu_{tt}}{t}, & s = 0, \end{cases} \quad (3)$$

where $s$ is the distance between nodes in the cloud storage network and $\mu_{tt}/t$ is the probability of node selection. From this $(t + g)$ number of data can be obtained the matrix $[\lambda_1, \lambda_2, \ldots, \lambda_{(t+g)}]^N$, thus, the final storage packet matrix is determined as follows:

$$[j_1, j_2, \vdots, j_{(t+g)}] = [\lambda_1, \lambda_2, \ldots, \lambda_{(t+g)}]^N, \quad (4)$$

Among them, $[\lambda_1, \lambda_2, \ldots, \lambda_{(t+g)}]^N$ is a matrix of order $(t + g) \times t$, and all its elements satisfy $U_x = (\lambda_{xy} = s)$ rule, the corresponding non-full-rank probability relationship in $F_2 = [0, 1]$ field is obtained as follows:

$$u_{\text{failure}} \leq t \sum_{q=1}^{t} \binom{t}{q} q^{(t+g)}, \quad (5)$$

where $q$ is the total number of nodes. When the selection probability of storage nodes is $0.5$, the non-full-rank probability relationship is reduced to $u_{\text{failure}} \leq (0.5)^q$. Considering that the loss in the stored procedure is affected by the number of nodes, its relationship can be expressed in the exponential form as follows:

$$\sum_{m=0,1,\ldots,q} \binom{q}{m} 2^{(q-1)} = 2^{(q-1)} \quad (6)$$

In the storage operation, this paper uses the $F_2 = [0, 1]$ domain to obtain the information required for storage. Scheduling control determines the data status based on the domain to obtain the information required for storage.

$$D(w, z) = \frac{|d_{(w+1)} - \hat{d}_{(w+1)}|}{d_{w,z}} \quad (8)$$

When $D(w, z)$ increases, that is, when the oscillation of unstructured big data in the cloud environment is active, the storage frequency should be accelerated. When $D(w, z)$ decreases, that is, the oscillation activity of unstructured big data in the cloud environment decreases, the storage frequency should be reduced.

To improve the efficiency of unstructured big data cloud storage, an audit scheme is introduced to optimize the storage. Firstly, the audit requirement analysis is carried out for cloud storage data packets. Whether the stored data packets are audited and to what extent the audit is necessary depend on the attributes $d_q, b_q$ of the data packets. $d_q$ represents the probability of data corruption, and $b_q$ represents the heat of data. The relationship between a single attribute and audit requirements is extended to the data packet level, and the storage audit requirements of cloud storage data packets are as follows:

$$V_q = \mu_1 \frac{V_{\text{max}}}{d_{\text{max}}} \times d_q + \mu_2 \frac{V_{\text{max}}}{b_{\text{max}}} \times b_q \quad (9)$$

$V_{\text{max}}$ is the largest audit requirement. $\mu_1$ and $\mu_2$ are the weighting coefficients of $d_q$ and $b_q$, respectively. Then, the data storage time is audited. Time audit is used to determine the time consumption of data storage. After data storage is complete, we minimize the storage time of nodes. We suppose the packet storage set is $\{D_1, D_2, \ldots, D_t\}$, the number of storage times of any packet $D_s$ is $C_s$, so the storage audit time can be obtained as follows:

$$N_{\text{audit}} = \sum_{x=1}^{t} \sum_{y=1}^{t} n_{x,y}, \quad (10)$$

where $n_{x,y}$ represents the $y$-th storage audit time. The number of cloud storage data packets is audited. The stored packets are arranged in positive order $\{N_1, N_2, \ldots, N_p\}$. The average time required for packet storage in the $x$-th cycle is $\tau_x$, so the number of stored packets can be audited as follows:

$$C_{\text{audit}} = \sum_{x=1}^{p} \left( N_x - N_{(x-1)} \right) / \tau_x \quad (11)$$

For periodic $N_x$, if the value is too small, the audit result will increase sharply. If the value is too large, the audit set for storing data packets is too large. During the audit process, the matching indexes are combined to ensure that packets are scattered in each cycle as much as possible. Here, the match indicator is defined as follows:

$$W_{\text{audit}} = \frac{(S_u - S_{\text{min}})}{S_{\text{min}}} \quad (12)$$

where $S_u$ indicates audit intensity, $S_{\text{min}}$ indicates the $S_u$ lower limit to meet data audit requirements, and $W_{\text{audit}}$ indicators are used to measure deviations in audit processing. When $W_{\text{audit}} > 0$, the audit intensity exceeds the required range. When $W_{\text{audit}} < 0$, it indicates that the intensity is within the required range. When $W_{\text{audit}} = 0$, it indicates that the intensity is exactly what is required. $S_u$ indicators reflect the audit effect of stored data, such as data identification capability. We suppose the data audit effect is expressed as key-
value pair form \( N_q, K_q \) about the audit cycle, where \( K_q \)
represents the data identification rate within the \( N_q \) cycle.
The corresponding \( S_u \) indicator is as follows:
\[
S_u = \eta_1 \times \frac{S_{\max}}{N_{\max}} + \eta_2 \times \frac{S_{\max}}{K_{\max}} \times K_q.
\]
(13)

\( S_{\max} \) indicates the maximum audit intensity, \( N_{\max} \) and
\( K_{\max} \) are the maximum values of \( N_q \) and \( K_q \), respectively. \( \eta_1 \)
and \( \eta_2 \) are the weighting coefficients of \( N_q \) and \( K_q \),
respectively.

Cloud computing uses arrays and Merkle trees to store unstructured big data. An array of unstructured data is mapped one-to-one to a Merkle tree, with an ID and a Merkle root address. That way, in the case of an increase in data, the array is converted to the corresponding Merkle tree operation. Merkle tree consists of inner and Leaf nodes. Leaf adopts a data structure in the form of \((O, Hash, D)\), and its Hash selects homomorphism Hash. \( O \) represents the encrypted data ID and array ID, and \( D \) represents the encrypted data of \( D \). Inner Hash selects MD5. \( O \) represents the maximum O value in leaf. We assume that any unstructured data \( D(\text{ID}, \text{NAME}, d_1, d_2, d_3) \), in the block chain, are distinguished by ID and NAME, and all nodes will realize encryption processing except for ID and NAME through shared keys. When data are updated, only the encrypted data need to be modified. Therefore, the data update process is described as follows:

\[
PD = (\tilde{d}_1 + \Delta d_1, \tilde{d}_2 + \Delta d_2, \tilde{d}_3 + \Delta d_3).
\]
(14)

According to data update calculation, this process conforms to homomorphic Hash. In block chain design, the first component has the parent Hash, the Merkle root Hash, the timestamp, and the block Hash. The parent Hash is used to construct a chain structure that points to the previous block Hash. The Hash of the block is calculated by the other three components using MD5. The Hash will be the parent Hash of the next block. Block body components include Hash Table, Merkle tree, and authentication path. The Hash Table is used to hold requests for block chain network validation, from which Merkle trees are created. In a Merkle tree, the array of Leaves is the authentication path. In genesis block of block chain network, both parent Hash and Merkle are null, and genesis block Hash is computed only by timestamp. Figure 3 depicts the block chain model for this article.

Storage data integrity audit relies on the storage efficiency reduction caused by three parties, and data integrity verification is realized based on block chain network. In the verification stage, the authorization center of the block chain network first obtains the key value according to the calculation, and then, the node encrypts and hazes the unstructured data to obtain \( L_z(\tilde{D}) \) and transmits it. After the cloud computing network receives the data, it is audited and stored, and meanwhile, the Merkle tree leaf nodes of the encrypted data \( \tilde{D} \) of the block chain network are obtained. When the block chain network detects an update to the data, it modifies the leaf node corresponding to Merkle. The nodes in charge of verification in the network will randomly select the data ID, and the nodes related to the data ID in the cloud storage network and block chain network will perform verification. The cloud storage network searches for the data label and sends back the response to the block chain network node, along with Merkle root Hash. The block chain node searches the corresponding label \( L_z(\tilde{D}) \) and compares the Merkle root Hash. When the Hash value is confirmed to be consistent, the verification is successful and the data integrity verification is completed.

3. Result Analysis and Discussion

3.1. Experimental Environment. The experimental platform adopts CloudSim open source cloud computing simulation platform and MATLAB data processing tool. The specific experimental environment is shown in Table 1. According to the modeling requirements of the problem studied in this paper, the task model and virtual machine model are adjusted on the platform, and the DatacenterBroker class and Cloudlet class are rewritten to implement the optimized algorithm in this paper.

Table 1 shows configuration table of experimental environment.

In the experiment, the optimized algorithm in this paper and five other algorithms in the literature [16–20] were used to conduct task set scheduling on the cloud platform to compare the scheduling performance of the six algorithms. Specific experimental parameters are shown in Table 2, in which the algorithm in the literature [16] adopts parameters in the original text. In the experiment, the time of six algorithms to perform different tasks in the case of fixed cloud platform resources was compared.

3.2. Experimental Parameters. To effectively evaluate the optimization effect of the proposed method, two experimental environments were set up. Based on the same conditions in different environments, the cloud storage throughput and average occupancy were taken as evaluation indexes for stability optimization, and the effect evaluation simulation was carried out before and after optimization. The details of the simulated environment are shown in Table 3.

3.3. Throughput Evaluation and Analysis. Through screening and processing 10 groups of experimental data, the mean value of the same request arrival rate is obtained. Figure 4 shows the effect of the proposed method before and after optimization based on throughput under different experimental environments.

As can be seen from Figure 5, when the initial priority of users of unstructured big data is low, the difference before and after optimization is not obvious. However, as the total distance difference between the home node and all nodes in the system is minimized, the cloud storage throughput performance optimized by the method in this paper is still slightly improved.

It can be seen from the above curves that there is no significant difference before and after optimization before
throughput reaches the peak. However, when the request arrival rate increased to about 2,400, the throughput values before and after optimization began to differ. As the calculation forms of energy consumption under different activity states are constructed, the throughput value after optimization is always in a high position. When the request arrival rate is about 5,600, the throughput value decreases before optimization. After optimization starts to decline when the request arrival rate is around 6,300, and the decline is relatively gentle.
3.4. Evaluation and Analysis of Average Occupancy Rate.
In three different preset environments under the same conditions, the average occupancy energy before and after optimization was simulated (see Figures 6–7). As can be seen from the curve trend in Figure 6, the difference in average occupancy rate before and after optimization in the case of low initial priority is small and there is no significant difference. As can be seen from Figure 7, request arrival rate 2,900 is the inflection point of the average occupancy index, and both trends show a linear growth trend. As the request arrival rate continues to increase, the average occupancy rate before and after optimization gradually opens up a gap. The latter shows significant advantages because it improves the fault tolerance performance of cloud storage of big data.

4. Conclusion
With the rapid development of science and technology, the infrastructure and functions of universities have been
constantly improved. The quality of life and study of teaching staff and students are gradually improved, which increases the workload and difficulty of logistic work in universities. The traditional logistics management in universities mostly adopts the Hadoop-based logistics data processing platform. Through a Hadoop-based software framework, students' accommodation, vehicle use, catering management, campus facilities, and equipment use are recorded to realize the management of logistics in universities. However, in the process of data processing, there are poor efficiency, high error, difficult query, and other defects, and cannot realize the standardization and scientific logistics management of colleges and universities. Therefore, this paper proposes a storage algorithm based on block chain technology for intelligent data processing and optimization of university logistics. In the cloud storage network, storage audit policies are introduced to determine storage audit requirements based on data popularity and damage, and audit the storage duration and data packet quantity to optimize storage efficiency. While optimizing the cloud storage method of unstructured big data, a multilayer block chain network model suitable for logistic data of universities is introduced, in which efficient data integrity verification based on Merkle tree and Hash is realized. Experiments prove that the proposed method has good universality for unstructured data with different formats and significantly improves the cloud storage efficiency of unstructured data. It can effectively respond to a large number of requests to access data and has good big data processing capability. In the future, it is planned to improve the service performance of the network system as the follow-up research direction.

**Data Availability**

The labeled data set used to support the findings of this study is available from the corresponding author upon request.

**Conflicts of Interest**

The author declares that there are no conflicts of interest.

**Acknowledgments**

This work was supported by the Wenzhou University of Technology.

**References**


