electronics

MDPI

Article

Cloud Storage Strategy of Blockchain Based on Genetic Prediction Dynamic Files

Jiali Tang [1], Chenrong Huang [2,*], Huangxiaolie Liu [3] and Najla Al-Nabhan [4]

1 College of Computer Engineering, Jiangsu University of Technology, Changzhou 213001, Jiangsu, China; tangjl@jsut.edu.cn
2 School of Computer Engineering, Nanjing Institute of Technology, Nanjing 211167, Jiangsu, China
3 Nanjing Vivo Software Technology Co., Ltd., Nanjing 211106, Jiangsu, China; char900814@hotmail.com
4 Department of Computer Science, King Saud University, Riyadh 11451, Saudi Arabia; nalnabhan@ksu.edu.sa
* Correspondence: huangcr@njit.edu.cn

Received: 10 February 2020; Accepted: 26 February 2020; Published: 28 February 2020



Abstract: With the rapid expansion of data volume, traditional data storage methods have been unable to meet the practical application requirements of blockchain cloud storage. Aiming for the cloud storage problem of blockchain, a new storage access method for predicting dynamic file load is proposed. By predicting the load status of cloud storage files in advance, the load of each blockchain data node at the next moment is first estimated. A hierarchical genetic algorithm is used to construct the connection weights between the hidden layer and the output layer, which makes the data network converge faster and more accurate, thereby effectively predicting the node load. In addition, based on the file allocation, an evaluation analysis model is constructed to obtain the time response capability of each file during the allocation process. The node's periodic load prediction value is used to calculate the corresponding weight of the node and it is continuously updated, retaining the advantages of the static weighted polling algorithm. Combined with the genetic algorithm to help predict the file assignment access strategy of the later load of each node, it can meet the system requirements under complex load conditions and provide a reasonable and effective cloud storage method. The experimental evaluation of the proposed new strategy and new algorithm verifies that the new storage method has a faster response time, more balanced load, and greatly reduced energy consumption.

Keywords: blockchain; dynamic file; load forecasting; time response

1. Introduction

Blockchain is regarded as a public ledger, in which all committed transactions are stored in a chain of blocks. This chain continuously grows when new blocks are appended to it. The blockchain technology has such key characteristics as decentralization, persistency, anonymity, and auditability. Blockchain can work in a decentralized environment, which is enabled by integrating several core technologies such as cryptographic hash, digital signature (based on asymmetric cryptography), and a distributed consensus mechanism. With blockchain technology, a transaction can take place in a decentralized fashion [1–6]. During the storage phase of blockchain, the problem to be solved is how to use redundant configuration, distribution, and cloud computing technologies to classify the blockchain data according to certain rules, reducing the storage capacity through filtering and de-duplication, and adding metadata that is easy to retrieve later to achieve low-cost, low-energy, and high-reliability goals. Cloud storage is developed on the basis of cloud computing technology. It regards data storage and management as a core task. Cloud storage can use related software to build on the basis of cluster applications, grid technology, and distributed file systems. Various types of storage devices in the blockchain network are combined to provide users with related data storage and business access

functions. The explosive growth of data volume has led enterprises and individuals to impose strict requirements on the persistence, balance, and availability of data. How to store these huge data quickly, efficiently, and reasonably becomes a blockchain calculation and the top priority of cloud storage technology. Therefore, in order to improve the access speed and accuracy of blockchain storage, while reducing traffic and data transmission, and reducing energy consumption, scholars in related fields have researched and experimented, and have proposed corresponding cloud storage methods and strategies [7,8].

Reference [9] proposed a grid-based blockchain storage method. After transferring computing tasks to the corresponding data storage nodes or adjacent nodes, this reduces the pressure on the network and provides new analysis for data processing in various fields. However, there is a problem of large comparison and analysis errors. Reference [10] proposed a method for summing up regular sites according to the weight of each sub data site. Although this method improves the processing efficiency of blockchain data, it has the defect of low robustness. Reference [11] proposed a blockchain parallel storage access strategy based on NAND FLASH. Through the different types of data, feature information is extracted and classified into files to realize parallel cloud storage access. This storage strategy is simple and feasible to implement, but the system is prone to hot spots and thus increases system energy consumption. Reference [12] introduced a key technology of an Orthrus cloud storage system. The system uses a multi-server form and implements load balancing among servers based on a variant genetic algorithm. However, the system is apt to fall into a local optimum and is unable to achieve optimal distribution of the system load due to the lack of a mutation process. Reference [13] used a rough set of serial data storage access methods to access the distribution list of data classes. This method effectively improves the storage access accuracy of the blockchain data, but the calculation is too complicated, which makes the system response time defective. Reference [14] proposed a parallel column storage access strategy. This study integrates the connection heuristic algorithm and the sub-slice aggregation theory, which effectively solves the fast storage and access of blockchain data under the cloud system, but the reliability is not ideal. Reference [15] introduced a Kuhn cloud storage system, which divided related files into blocks, built metadata, and stored the metadata in files. New file objects are stored in the file directory structure. The storage performance of metadata needs to introduce directories, which restrict the development and use of applications. References [16,17] proposed a distributed storage system of the blockchain network, which tried to use redundant data to repair the failure of the data stored in the node. Reference [18] discussed a type of blockchain developed for file storage. Data are stored in the blockchain forever and can be retrieved using a transaction hash as an identifier.

Aimed at various problems existing in current cloud storage strategies, a new blockchain cloud storage strategy for predicting dynamic files is proposed. This method performs model analysis on file assignment access, and provides a suitable load node for the storage layout of dynamic files by estimating the blockchain data node load model [19–21]. Combined with the fastest storage response strategy, the relationship between the system working environment and the node load is effectively realized to allow blockchain data to access to storage quickly and reliably under the cloud platform. Through simulation analysis, the proposed new cloud storage method is validated, which can adapt to various load modes of the blockchain storage system and can store storage access to the file system. Compared with traditional cloud storage strategy, it has a more prominent performance advantage.

The remaining of this paper is structured as follows. Section 2 introduces the cloud storage model of blockchain data, and the load forecast of blockchain storage technology is analyzed in Section 3. The genetic prediction algorithm is depicted to guarantee the reliability of system data in Section 4. Experimental comparison and comparative analysis are presented in Section 5, and finally, concluding remarks and future work are presented in Section 6.

2. Cloud Storage Model and File Assignment Access Analysis

Cloud storage is based on the distributed network technology. It is a data storage method for blockchain with high-efficiency and low-cost. Users do not need to consider the complicated underlying technical details such as storage capacity, storage devices, storage locations, and data characteristics. They can get near-infinite storage space and enterprise-class service quality from their service provider. As shown in Figure 1, the cloud data center is the core module of cloud storage services. It uses distributed technology and parallel databases for various types of data storage. Based on the cloud service level agreement, it provides professional resource services to cloud users through interfaces. The cloud client is the medium through which the user interacts with the system. The user customizes the service through the browser and operates the service within the authority. The function of the cloud service interface is to manage users' authorization, authentication, login, etc., and to manage available computing resources and services. It accepts user requests, forward corresponding programs according to user requests, and dynamically allocates, schedules, and recycles resources.

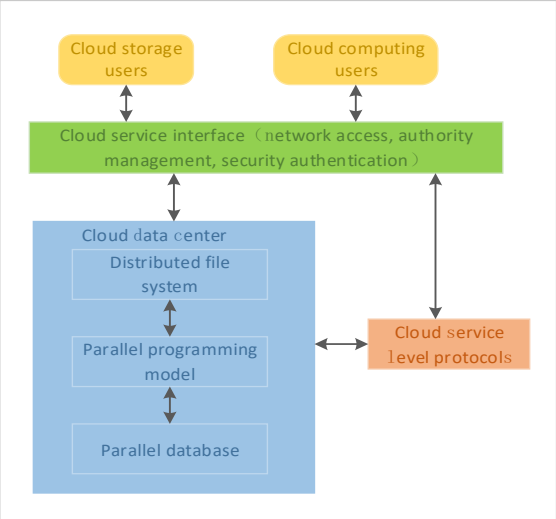


Figure 1. Cloud storage model of blockchain data.

Under the cloud platform, the data are distributed on a large number of nodes in the form of files. The method and advantages of the file layout will seriously affect the storage access performance of blockchain on the cloud platform system [22,23]. To analyze the dynamic file assignment problem of blockchain data, a mathematical model is used to describe the system model by abstracting files and nodes into two independent collections, so that the file processing under the cloud platform is transformed into the above two collection mapping problems, thereby analyzing and solving the response speed of blockchain storage access and the load balancing problem of the cloud storage system.

The load balancing algorithm can reasonably distribute requested tasks among multiple server nodes, balancing the load of each node basically. Therefore, the algorithm quality directly affects the overall system performance. Load balancing strategies can generally be used in two stages. One is to call a balancing algorithm when a request task arrives. It assigns the task to the appropriate node according to the algorithm; the other is that a node is overloaded in the working state and the algorithm is started. The tasks on the heavily loaded nodes are transferred to the lightly loaded nodes for processing.

Under the blockchain cloud storage platform, the data nodes can be expressed as:

$$D = \bigcup_{j=1,2,\cdots,m} \{D_j\} \tag{1}$$

where D_j represents the node j, and m represents the number of nodes.

The file can be expressed as:

$$F = \bigcup_{i=1,2,\cdots,n} \{f_i\}. \tag{2}$$

In the formula, f_i is referred to as a file, and n is the number of files.

According to the file set and the data node set, the mapping relationship between the two is established, which can be expressed as:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{3}$$

where $\forall i = 1, 2, \cdots, n$, $\forall j = 1, 2, \cdots, m$, and a_{ji} is the map of file f_i to node D_j in matrix A. When f_i is stored in D_j, $a_{ji} = 1$; when f_i is not stored in D_j, $a_{ji} = 0$.

We normalize the column vectors of matrix A, so

$$A_{ij} = \left(\frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \right). \tag{4}$$

We expand A_{ij} by rows and get

$$W^* = \left(\sum_{j=1}^{n} \frac{a_{1j}}{\sum_{i=1}^{n} a_{ij}}, \sum_{j=1}^{n} \frac{a_{2j}}{\sum_{i=1}^{n} a_{ij}}, \ldots, \sum_{j=1}^{n} \frac{a_{nj}}{\sum_{i=1}^{n} a_{ij}} \right)^T, \tag{5}$$

normalize W^* and get

$$W = (w_1, w_2, \ldots, w_n)^T, \tag{6}$$

and then get the maximum eigenvalue of A

$$\lambda = \frac{1}{n} \sum_{i=1}^{n} \frac{(AW)_i}{w_i}. \tag{7}$$

In addition, because f_i is only stored in D_j, so

$$\sum_{j=1}^{m} a_{ji} = 1. \tag{8}$$

A single file can also be defined as: $f_i = \langle s_i, \lambda_i, es_{ij}, h_i \rangle$, where s_i indicates the size of f_i, λ_i indicates access speed, es_{ij} indicates the expected time, and h_i indicates hot files.

For any data node, it can also be defined as: $N_j = \langle c_j, tr_j, l_j \rangle$, where c_j, tr_j, and l_j represent the capacity, read rate, and load of D_j, respectively.

Suppose the system request set is:

$$R = \bigcup_{k=1,2,\cdots,|R|} \{r_k\} \tag{9}$$

where r_k is *request k*, and $|R|$ is the requested quantity. At this point, the response time of the cloud storage can be obtained:

$$\theta = \sum_{k=1}^{|R|} rt_k / |R|. \tag{10}$$

rt_k in the formula is the response time of r_k.
After calculation, the node load is expressed as:

$$\rho = \frac{1}{m} \cdot \sum_{j=1}^{m} l_j. \tag{11}$$

3. Load Forecast Analysis

The cloud platform manages and schedules storage, computing, and network resources through the network to implement configuration optimization and resource integration, enabling each user to acquire and use computing resources on demand, minimizing application costs, and maximizing benefits. The realization of this process requires the cloud platform to allocate resources reasonably, and to reduce the waste of resources, but also to meet the resource requirements of the application. This requires us to make relevant predictions on the cloud platform resource load requirements, understand the future trend of resource requirements of application services, prepare for the increase of resource allocation for the arrival of high-load application services, and prevent violations of service level agreement (SLA) of applications.

The load blockchain data under the cloud platform are typical time series data. Before performing short-term load forecast, we normalize all the data first, fit the load data, and then convert quantitative data such as CPU, memory, hard disk, and network to the qualitative concept made up of multiple clouds.

As shown in Figure 2, short-term forecasting of the cloud platform resource load is an indispensable component of its resource optimization configuration process. It can obtain the resource load information of the blockchain application service in the future for a period of time through the relevant prediction model, and provide relevant decision support for the reasonable and systematic configuration of the cloud computing resource. The accuracy of the calculated value of the load prediction is the key to determine whether the cloud platform system can allocate storage resources reasonably and effectively.

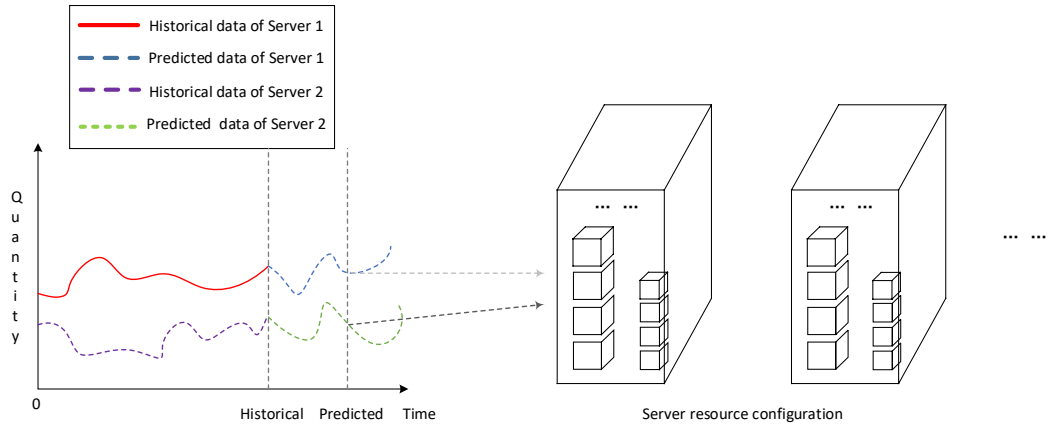


Figure 2. Cloud computing resource load short-term forecasting and resource optimization configuration.

Define *LB* as the load balancing parameter, which is used to evaluate the load balancing status of the cloud platform. Let the number of computing nodes in the cloud computing cluster be m,

the number of parameters be n, and the projection center of gravity of the computing nodes in the n-dimensional parameter space be $G(X_1, X_2, X_3, \ldots, X_n)$. The average value of the distance projected from each node to the center of gravity is:

$$A_G = \frac{1}{m}\sum_{i=1}^{m}\sqrt{\sum_{j=1}^{n}\left(x_{ij} - X_j\right)^2}. \tag{12}$$

Obviously, when the load in the cloud computing cluster is half empty and the nodes are full, the load balance of the cluster is the worst. The average value of the distance between the projection points of each node and the center of gravity in the calculation, and normalized, is defined as the node projection space load balance degree:

$$LB_{ij} = \frac{\sum_{i=1}^{m}\sqrt{\sum_{j=1}^{n}\left(x_{ij} - X_j\right)^2}}{m \times \frac{\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}}{2}}. \tag{13}$$

That is the system load balance degree when task i is assigned to node j.

The load balancing degree is an important indicator for measuring the load balancing status of cloud computing clusters, and quantitatively gives the load balancing degree of the cluster. In an ideal case, all computing blockchain nodes at a time have the same load, and their projection points are aggregated into one point. At this time, the load balancing degree of the cluster is 0, the cloud computing cluster is in an ideal load balancing state, and the maximum value of LB_{ij} is 1. Therefore, $LB_{ij} \in (0, 1)$, and the smaller the value, the better the load balance of the current cloud computing cluster.

Blockchain storage access under the cloud platform can be done with the load table representing each node, according to $AR(n)$ Model, which is calculated as:

$$l_t = \psi_1 l_{t-1} + \psi_2 l_{t-2} + \cdots + \psi_n l_{t-n} + a_t, a_t \sim NID(0, \sigma_a^2). \tag{14}$$

So, it can be deduced:

$$a_t = l_t - \psi_1 l_{t-1} - \psi_2 l_{t-2} - \cdots - \psi_n l_{t-n} \tag{15}$$

$$\sigma_a^2 = \frac{1}{N-n}\sum_{t=n+1}^{N}(l_t - \psi_i l_{t-i})^2. \tag{16}$$

$AR(n)$ model is expressed as the predictive model of $\psi_1, \psi_2, \cdots, \psi_n$. Combine the least squares method to convert Equation (14) into $Y = \Psi X + a$, in which $Y = \begin{pmatrix} l_{n+1} & l_{n+2} & \cdots & l_n \end{pmatrix}^T$, $\Psi = \begin{pmatrix} \psi_1 & \psi_2 & \cdots & \psi_n \end{pmatrix}^T$, $a = \begin{pmatrix} a_{n+1} & a_{n+2} & \cdots & a_N \end{pmatrix}^T$, and $X = \begin{pmatrix} l_n & l_{n-1} & \cdots & l_1 \\ l_{n+1} & l_{n+2} & \cdots & l_2 \\ \vdots & \vdots & \ddots & \vdots \\ l_{N-1} & l_{N-2} & \cdots & l_{N-n} \end{pmatrix}$.

At this point, the following is calculated:

$$\Psi = (X^T * X)^{-1} * X^T * Y. \tag{17}$$

To achieve a reasonable predictive load situation, a suitable function evaluation $AR(n)$ System model is required. The following functions are adopted:

$$FPE(n) = \frac{N+n}{N-n}\sigma_a^2. \tag{18}$$

When $FPE(n)$ in the case takes the minimum value, it corresponds to the best case for the system $AR(n)$ model.

4. Genetic Prediction File Storage Access Algorithm

In order to make the cloud storage response time as fast as possible, at the same time, each data node in the blockchain system is in a balanced load state as much as possible, and a dynamic file storage access strategy combined with the predicted load state is proposed. In blockchain cloud storage, files are transmitted and stored in batches. Therefore, the new algorithm proposed here distributes files to the corresponding nodes in batch format. When the new file is transmitted, first, the load status and change of each node are queried through the node load table, and then the node load model obtained by the previous analysis is used to estimate the load change of the subsequent cloud storage node. Finally, a genetic algorithm is used to assign the transmitted file to the predicted σ nodes, the occurrence of unbalanced nodes is prevented, the system response delay caused by node overload is reduced, and the reliability of the system data is also guaranteed.

Genetic algorithms have good parallel search and global optimization capabilities, and are also commonly used to train neural networks to better approximate the global minimum [24–26]. In each generation of population evolution, the individuals with the best fitness values are retained and copied directly to the next generation population. Elite retention strategy is an important guarantee for the convergence of genetic algorithms. It makes the best individuals not get destroyed by various genetic operators, and they can remain in the population until they are replaced by better individuals.

First, if any combination of storage characteristics is generated, the combination of file storage characteristics is directly encoded to form a chromosome individual. This process is repeated until the number of individuals is satisfied, thereby forming a population; then, the intra-class distance between classes is used to calculate the average intra-class distance between the storage feature combinations represented by each chromosome. The average inter-class distance is used as the aggregation degree determination index. The dominant individual is selected from the population by a certain selection algorithm and three different operations, such as replication, hybridization, and mutation operator, are performed with a certain probability to generate a new generation of population individuals. Repeat the above process until there are individuals in the group that meet the given degree of aggregation, or exit the loop after completing the specified number of iterations.

According to the genetic algorithm, the correspondence between the definition file and the node represents the unit individual; for the individual A, only when the following conditions (1) and (2) are true is A used to represent the desired individual.

(1) All files correspond to unique node.

$$\forall i, \sum_{j=1}^{m} a_{ji} = 1. \tag{19}$$

(2) The amount of files assigned to any node should be less than its total load.

$$\forall j, \sum_{i=1}^{n} a_{ji} * s_i < c_j. \tag{20}$$

p_t represents the existing population. Calculate the probability P_i that each individual is selected to be inherited for the next generation population.

$$P_i = \frac{F_i}{\sum_{i=1}^{n} F_i}, i = 1, 2, \ldots, n. \tag{21}$$

Generate a random number α between $[0, 1]$, and if $\sum_{j=1}^{i-1} P_j \leq \alpha \leq \sum_{j=1}^{i} P_j$, then select individual i to enter the offspring population.

Using the roulette algorithm [27–29] in existing populations p_t, extract individuals p_1 with p_2 as single point intersections, and then use the obtained new individual variation to obtain a new generation of samples, and select individuals from the current population according to individual fitness to form a new generation of population; use the simple elite strategy to preserve the optimal individual. In the process of evolution, the simple elite strategy always preserves the most adaptive individuals in the population, avoiding them being destroyed in various genetic operations such as mutation, inverse string, and string insertion. If the individual's fitness value is found in the current population, which is greater than the individual that is currently the best, it will be saved to replace the current one; finally, use the expectation function to calculate the good individuals that meet the system's rapid response,

Factors such as population size N, crossover probability P_c, and mutation probability P_m are considered in the design algorithm. The basic process is as follows:

(1) Use a binary string to encode the search solution space and randomly generate an initial population of N individuals.

(2) Calculate the fitness function value F_i of each individual in the population.

(3) Determine whether the fitness function value F_i meets the algorithm termination conditions. If the conditions are met, exit the algorithm, and if otherwise, continue.

(4) Use roulette algorithm to copy individuals and generate new populations according to the random number α.

(5) According to the certain cross probability P_c, cross two randomly selected individuals i_1 and i_2, and use a single-point crossover operator to generate two new individuals.

(6) According to the certain mutation probability P_m, mutate a randomly selected individual i and use a basic mutation operator to generate a new individual.

(7) Go to step 2.

The improved roulette strategy not only ensures that good individuals can enter the next-generation population with a higher probability, but also gives individuals with low fitness a certain opportunity to choose, without losing individuals, and it ensures the integrity and diversity of the population.

5. Experimental Simulation Proof

5.1. Simulation Environments

To validate the proposed final performance of blockchain cloud storage strategy for genetic prediction dynamic files, a cloud storage verification system [30,31] is built based on Linux. In the experiments, a Linux server with 16GB memory and Intel Core-i7 CPU is utilized to implement the simulation in the blockchain environment. The software environment uses CloudSim [32] to evaluate the improved algorithm through simulation experiments. We rewrite the *bindCloudletToVm* method in the *DatacenterBroker* class, and use the *Ant* tool to add the improved algorithm to the task scheduling unit of the platform for simulation experiments.

CloudSim is a cross-platform open source software that provides cloud computing data center virtualization technology, and also provides a series of interfaces for virtualized cloud modeling and simulation functions. The simulation experiment calls the simulation layer module in CloudSim through the *UserCode* layer. This layer mainly provides support for the simulation of cloud computing data center environments. The content involves dedicated management interfaces such as virtual machines, memory, storage, and bandwidth in cloud computing data centers. As shown in Figure 3, the *Simulation* layer can formulate and perform virtual machine deployment plans, perform host-to-virtual machine mapping, and dynamically monitor the system.

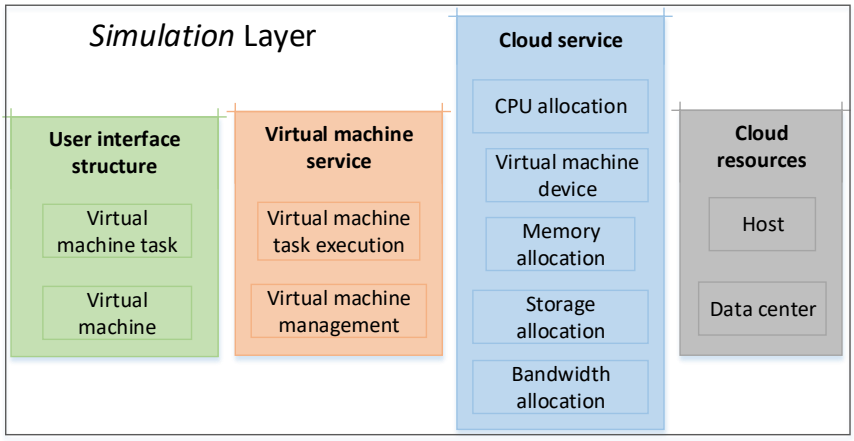


Figure 3. Structure of the *Simulation* layer in CloudSim.

We create 60 virtual machines as data nodes on the Linux server for scheduling 60–120 restore tasks. Virtual machine to host mapping and distribution is implemented by CloudSim's own *Time-Shared* algorithm. Priority weighting factor ω is set to 0.5. Experiments are performed to compare the execution time, packet loss rate, and load balance of the improved and traditional algorithms.

5.2. Parameter Configurations of Genetic Algorithm

After the load prediction analysis, the genetic algorithm is adopted to verify the reasonable mapping relationship between files and nodes. We use the improved algorithm proposed in Section 4 to establish an appropriate fitness function based on the elements of node memory, link bandwidth, and transmission path to obtain the optimal solution. In addition, we adopt a penalty function method to protect the survival of infeasible solutions, which can prevent the algorithm from prematurely falling into the local optimal solution to achieve the goal of global convergence. The main process of applying genetic algorithm in cloud storage strategy is shown in Figure 4.

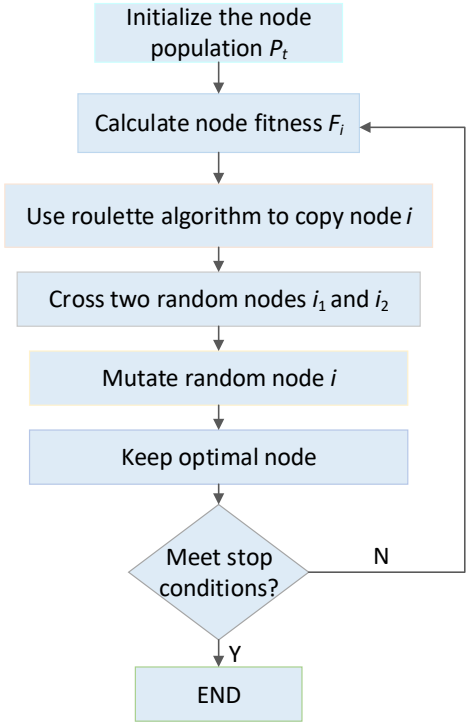


Figure 4. Process of applying the genetic algorithm in cloud storage strategy.

In the verification system, the user's attribute data space location does not change, and the amount of this data is relatively small. Assuming that the user's access rights change, the access control of the network data will also change. In this protection mechanism, the data in the cloud computing environment need to be segmented and divided into minimum attribute units. Then, we need to dynamically encrypt the data object of the smallest attribute unit to obtain the minimum attribute key. During the network operation, the secret key needs to be obtained according to the authorization, and the secret key is parsed according to the decryption method.

In order to verify the feasibility of the improved algorithm in blockchain cloud storage, all data nodes are assigned five mappings (*customer*, *date*, *supplier*, *part*, and *lineorder*) with a small amount of data. The nodes in this area include surviving, routing, and newly generated nodes. The corresponding genetic parameter settings are shown in Table 1.

Table 1. Parameter settings of the corresponding genetic algorithm.

Parameter	20GB	80GB	320GB	1TB
Initial population	200	1000	2000	5000
Max generations	1000	1000	2000	2000
Cross probability	0.5	0.5	0.6	0.6
Mutation probability	0.05	0.05	0.1	0.1
Iteration numbers	10^6	10^7	10^8	10^9

5.3. Simulation Results and Analysis

After the source data packets collected by the genetic algorithm convergence node are passed to some nodes in the cloud storage system, based on this part of the nodes, we arrange the historical blockchain data in the cloud storage data volume according to the time sequence into a time stamp sequence. Multi-level hierarchical sampling and storage of historical data are used to ensure the randomness of new samples stored in the blockchain by setting different sampling ratios. During the extraction experiment, the number of data nodes is increased from 15 to 60 (step size is 15), and five mappings and one restoration task were assigned to all data nodes. The details of the cloud storage system verification process are shown in Table 2.

Table 2. Experimental information.

Name	Data
customer	$40{,}000 \times SF$
date	$3556 \times SF$
supplier	$3000 \times SF$
part	$2000 \times [1 + b \times SF]$
lineorder	$7{,}000{,}000 \times SF$

b stands for the node's initial data value and SF stands for the incremental coefficient. During system verification, adjust SF value to get the data set of the desired size. The expected values here are 20 GB, 80 GB, 320 GB, and 1 TB (1024 GB).

Using the genetically predicted file storage access method, all nodes are effectively utilized, and compared with the traditional method in the same experimental situation, the experimental data of four set data sets are recorded, as shown in Figure 5. According to the data in the figure, it can be clearly seen that under the same amount of data, the execution time of the new storage access method is shorter, and the execution time is less affected by the amount of data, which proves that the new method is fast and efficient.

The fixed data volume is 80GB, and the data nodes are dynamically changed. The new method and the traditional method are respectively adopted, and the experimental data are obtained as shown

in Figure 6. When the number of nodes is the same, the execution time of the new method is faster, and in the case of a change in the number of nodes, there is still a significant speed advantage.

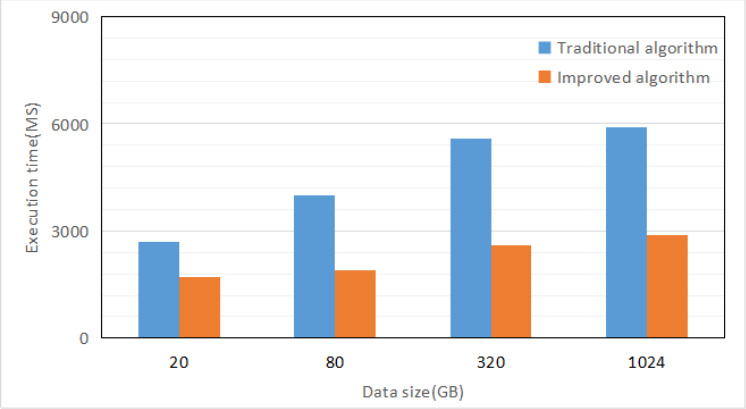


Figure 5. Execution time for different data volumes.

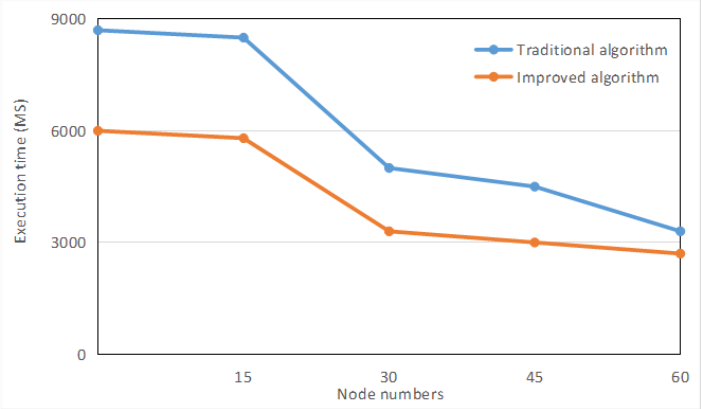


Figure 6. Execution curve for different node numbers.

Dynamically changing the amount of data, using new methods and traditional methods, the experimental data are shown in Figure 7. It can be seen that the method in this paper is based on the load balancing mechanism, which can prolong the network life cycle and reduce the network packet loss rate. The effect is better than the traditional method.

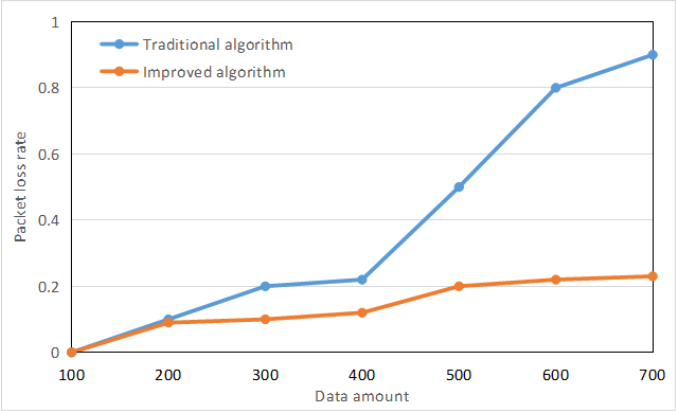


Figure 7. Execution curve for different data amount.

Table 3 shows the comparison of experimental data of a load balance degree. By comparing the experimental data of the two methods, it is proven that the blockchain cloud storage strategy of genetic forecasting dynamic file can reasonably distribute the node load and effectively improve the load balance, which is also beneficial to consumption control.

Table 3. Cloud storage system load balancing.

Number of Experiments (Times)	Traditional Algorithm Load Balance (%)	Algorithm Load Balance of Blockchain Cloud Storage Based on Genetic Prediction Dynamic File (%)
1	69	97
2	69	97
3	69	97
4	69	96
5	68	96
6	68	96
7	69	96
8	69	96
9	69	96

The information collection module collects the resource usage status information of the nodes in the cloud and submits this information to the message processing module. The message processing module performs quantitative processing on the collected information to obtain the information type that the inference engine can identify. Related rule information infers the load of the nodes in the system, and stores the obtained node load status information in the database. The coordination module selects the nodes that need to perform load balancing from the database, and executes the load balancing decision through the load balancing algorithm module. Then the number of migrated jobs is obtained through the migration strategy. The communication module specifically issues a load balancing instruction to each server.

6. Conclusions

Aiming at the status quo and existing problems of blockchain cloud storage, a blockchain cloud storage strategy for genetic prediction dynamic files is proposed. First, we establish a blockchain file assignment model, and analyze the node load situation and storage response time. Then, load prediction analysis is carried out. Finally, a genetic algorithm is adopted to assign the transmitted file to the predicted node, thereby realizing a reasonable mapping relationship between files and nodes. Through experimental comparison, the performance superiority of the blockchain cloud storage strategy for genetic forecasting dynamic file is verified. The system storage access speed is faster, the efficiency is higher, and it has good load balance. There are still some optimizations can be applied to minimize the cost of running the genetic algorithm in the blockchain data environment. We need to further optimize the algorithm process and design a faster population evolution method to achieve a more stable and efficient cloud storage strategy.

Author Contributions: Conceptualization, J.T.; methodology, H.L.; software, N.A.-N.; validation, C.H.; formal analysis, H.L.; investigation, N.A.-N.; resources, H.L.; data curation, H.L.; writing—original draft preparation, J.T.; writing—review and editing, C.H.; supervision, C.H.; funding acquisition, J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Opening Project of Jiangsu Key Laboratory of Advanced Numerical Control Technology (SYKJ201804), by the Project funded by Jiangsu Postdoctoral Science Foundation (2019K041), by Nanjing Subsidy Project of IUR Cooperation (201722043) and by Changzhou Sci&Tech Program (CE20195030).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
2. Dinh, T.T.A.; Liu, R.; Zhang, M.; Chen, G.; Ooi, B.C.; Wang, J. Untangling blockchain: A data processing view of blockchain systems. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1366–1385. [CrossRef]
3. Lin, I.C.; Liao, T.C. A Survey of Blockchain Security Issues and Challenges. *Int. J. Inf. Secur.* **2017**, *19*, 653–659.
4. Pan, Z.; Yang, C.N.; Sheng, V.S.; Xiong, N.; Meng, W. Machine learning for wireless multimedia data security. *Secur. Commun. Netw.* **2019**, *1*, 1–2. [CrossRef]
5. Tapscott, D.; Tapscott, A. How blockchain will change organizations. *MIT Sloan Manag. Rev.* **2017**, *58*, 10.
6. Bahga, A.; Madisetti, V.K. Blockchain platform for industrial internet of things. *J. Syst. Softw.* **2016**, *9*, 533–546. [CrossRef]
7. Cui, B.; Liu, Z.; Wang, L. Key-aggregate searchable encryption (kase) for group data sharing via cloud storage. *IEEE T. Comput.* **2016**, *65*, 2374–2385. [CrossRef]
8. Soulsby, D. Using cloud storage for nmr data distribution. *J. Chem. Educ.* **2012**, *89*, 1007–1011. [CrossRef]
9. Tian, Y.; Kaleemullah, M.M.; Rodhaan, M.A.; Song, B.; Al-Dhelaan, A.; Ma, T. A privacy preserving location service for cloud-of-things system. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *123*, 215–222. [CrossRef]
10. Song, B.; Hassan, M.M.; Alamri, A.; Alelaiwi, A.; Tian, Y.; Pathan, M.; Almogren, A. A two-stage approach for task and resource management in multimedia cloud environment. *Computing* **2016**, *98*, 119–145. [CrossRef]
11. Abbes, Y.; Najeh, S.; Besbes, H. Low-complexity detection and decoding scheme for LDPC-coded MLC NAND flash memory. *China Commun.* **2018**, *15*, 68–83. [CrossRef]
12. Wan, J.; Zhang, J.; Zhou, L.; Wang, Y.; Jiang, C.; Ren, Y.; Wang, J. ORTHRUS: A light weighted block-level cloud storage system. *Cluster Comput.* **2013**, *16*, 625–638. [CrossRef]
13. Song, J.; Tsang, E.C.; Chen, D. Minimal decision cost reduct in fuzzy decision-theoretic rough set model. *Knowl-Based Syst.* **2017**, *126*, 104–112. [CrossRef]
14. Kou, X.; Xu, G.; Yi, C. Belt-conveyor based efficient parallel storage system design and travel time model analysis. *Int. J. Prod. Res.* **2018**, *1*, 1–18. [CrossRef]
15. Zhang, X.Y.; Zhang, J.; Gong, Y.J.; Zhan, Z.H.; Chen, W.N.; Li, Y. Kuhn–Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks. *IEEE Trans. Evol. Comput.* **2015**, *20*, 695–710. [CrossRef]
16. Jiang, L.; Xie, S.; Maharjan, S.; Zhang, Y. Blockchain Empowered Wireless Power Transfer for Green and Secure Internet of Things. *IEEE Netw.* **2019**, *33*, 164–171. [CrossRef]
17. Dai, H.N.; Zheng, Z.; Zhang, Y. Blockchain for internet of things: A survey. *IEEE Internet Things J.* **2019**, *6*, 8076–8094. [CrossRef]
18. Gai, K.; Wu, Y.; Zhu, L.; Xu, L.; Zhang, Y. Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. *IEEE Internet Things J.* **2019**, *6*, 7992–8004. [CrossRef]
19. Rong, H.; Ma, T.; Cao, J.; Tian, Y.; Al-Dhelaan, A.; Al-Rodhaan, M. Deep rolling: A novel emotion prediction model for a multi-participant communication context. *Inform. Sci.* **2019**, *488*, 158–180. [CrossRef]
20. Ma, T.; Rong, H.; Hao, Y.S.; Cao, J.; Tian, Y.; Al-Rodhaan, M. A Novel Sentiment Polarity Detection Framework for Chinese. *IEEE Trans. Affect. Comput.* **2019**, *1*, 1. [CrossRef]
21. Al-Otaibi, B.; Al-Nabhan, N.; Tian, Y. Privacy-Preserving Vehicular Rogue Node Detection Scheme for Fog Computing. *Sensors* **2019**, *19*, 965. [CrossRef] [PubMed]
22. Prassanna, J.; Punitha, K.; Neelanarayanan, V. Towards an Analysis of Data Accountability and Auditing for Secure Cloud Data Storage. *Procedia Comput. Sci.* **2015**, *50*, 543–550. [CrossRef]
23. Küpçü, A. Official arbitration with secure cloud storage application. *Comput. J.* **2015**, *58*, 831–852. [CrossRef]
24. Yuan, X.; Elhoseny, M.; El-Minir, H.K.; Riad, A.M. A genetic algorithm-based, dynamic clustering method towards improved WSN longevity. *J. Netw. Comput. Appl.* **2017**, *25*, 21–46. [CrossRef]
25. Hiassat, A.; Diabat, A.; Rahwan, I. A genetic algorithm approach for location-inventory-routing problem with perishable products. *J. Manuf. Syst.* **2017**, *42*, 93–103. [CrossRef]
26. Kadri, R.L.; Boctor, F.F. An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *Eur. J. Oper. Res.* **2018**, *265*, 454–462. [CrossRef]
27. Cerf, R. The quasispecies regime for the simple genetic algorithm with roulette wheel selection. *Adv. Appl. Probab.* **2017**, *49*, 903–926. [CrossRef]

28. Qian, W.; Chai, J.; Xu, Z.; Zhang, Z. Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection. *Appl. Intell.* **2018**, *48*, 3612–3629. [CrossRef]
29. Cao, G.; Wang, X. Image encryption based on the combination of roulette wheel selection with linear congruence pixel transformation. *Multimed. Tools Appl.* **2019**, *78*, 10625–10647. [CrossRef]
30. Zhan, H.J.; Wei, Z. Cloud storage-based medical data integration technology. *Lect. Notes Elec. Eng.* **2014**, *279*, 1097–1102.
31. Barsoum, F.; Hasan, A.M. Verifying outsourced replicated data in cloud computing storage systems. *Int. J. Comput. Appl.* **2014**, *99*, 1–13.
32. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.* **2011**, *41*, 23–50. [CrossRef]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).