A Traceability System Based on the Quick Searching Blockchain

Youzhi Zhang, and Wanghu Chen

School of Computer Science and Technology, Northwest Normal University, Lanzhou 730070, China

Keywords: Kademlia, Blockchain tracing, Simplified Payment Verification

Abstract: With the improvement of the people's living quality. More and more people start to pay attention to the food tracing, but the ideal blockchain technology, as a solution to this problem, which have a low speed of searching and will occupy a large space of the storage, is unsuitable for the actual situation. Thus, in this paper, we mainly use and improve a method named Kademlia to make it to search the data rapidly. As for the data storage, we use the Simplified Payment Verification as the storage method. For this method, it is not necessary for most of the nodes to store a large amount of the data, nodes just need to verify that if the transaction data is stored in the block. In this way, we prevent the useless cost to the memory, and increase the efficiency of the searching at the same time.

1. Introduction

As an important aspect of food safety, food traceability has always been concerned, but now many traditional traceability systems are at risk of being tampered with and not trusted by the public. Blockchain is considered as a value exchange protocol. The system based on blockchain is faster, safer and cheaper than the traditional system in terms of value exchange [1]. Its transaction information is stored on multiple computers to ensure that under the premise of no consensus and common modification of multiple computers, the transaction cannot be tracked and modified [2], which can be well applied to this scenario. However, compared with the actual problems, blockchain also faces many problems in storage and query. Each node needs to store all the data, but for ordinary users, there is not so much storage space. For consumers, they just want to query the source of this batch of goods, not to store data and do a lot of calculations. And for node location, there is a lack of an effective and fast way, which needs to improve the storage and query methods.

For the storage problem, we improve the SPV algorithm. According to the heterogeneity of nodes, the nodes are divided into four categories: full node, light node, SPV node and query node. All nodes store all data, and form a consensus to confirm and store transactions, such as large farm contractors. While the light node only stores the data related to its own transaction, the full node stores the transaction data to itself and then to the light node, such as the smaller Contractor under the large contractor. SPV node uses simple storage, does not need to maintain complete blockchain information, only needs to use block header message [3]. Here, it synchronizes the block header of light node, only queries the transaction related to itself, sends the request to the light node, verifies the transaction and delivers goods, similar to planting farmers. The query node is directly for consumers, who scan the code to query the relevant transactions. The data and calculation required for the query are all carried out on the remote server, without excessive consumption.

For the query rate, we use the improved Kademlia algorithm. Based on the original algorithm, the count variable is added to each node stored in the routing table to indicate the number of times the node is searched. The nodes are arranged in order according to the count size, and the nodes with larger count are found first in each query. On the other hand, we store the transactions found in the K-V database according to the transaction batch number, and query them in the K-V database through the transaction batch number each time, avoiding the waste of computing power caused by multiple queries on the same transaction.

The following content of the article is arranged as follows: in the second part, we will describe some innovations that researchers have made in blockchain storage. In the third part, we will

  DOI: 10.25236/ictmic.2020.144

introduce the storage and query mechanism of the article in detail, and in the last part, we will make some summaries of the work of this article and make some prospects for the future work.

2. Related Work

Blockchain, as a new technology, has been attracting more and more attention. However, due to its large storage space and slow search speed in the later system, it has been unable to actually land. At present, many traceability systems based on mainstream frameworks such as Hyperledger Fabric [4], Ethereum [5], EOS, etc. still face the problems of storage and query rate. Storage is increasing and throughput is low, so many scientists put forward their own ideas based on their own needs.

In the paper of mobile charging system based on lightweight blockchain proposed by Nam ho Kim, Sun Moo Kang, etc. [6], SPV method is used as the solution of blockchain storage. Mobile chargers are put into use everywhere, and they are used as SPV nodes, while nodes in other regions and charging suppliers are used as full nodes. When users use SPV mobile charger, they will send the charging information to the full node or the parent node, and calculate the price according to the charging amount, which solves the storage problem well. Moreover, in view of the data redundancy of the parent node, the nodes in each region are simplified and only stored in the blockchain with the highest transaction related to themselves, which simplifies the storage and makes the system more efficient. However, there are some shortcomings in the system. The service providers in the system store all the data, and the storage pressure is great. With the increase of time, the storage cost will become higher and higher, and the system speed will become slower and slower. In addition, the number of users of the charging system is generally large, and the query method of data is also a challenge. It is necessary to find the data quickly to ensure the overall operation efficiency of the system.

In the paper based on search encryption cloud and blockchain proposed by Hu S, Cai C, Wang Q, etc. [7], the search method is implemented based on Ethereum. Methods Kademlia at the bottom of Ethereum was used for calculation, and Token was used for excitation to ensure the search speed and activity of the system. Through the blockchain system, the hacker's attack is well prevented, and the asymmetric secret algorithm is used to ensure the data security. However, for the actual traceability environment, it is impossible for every user to participate in the query of the blockchain system. Some edge SPV nodes or consumers do not need to consume computing power, so further improvement is needed in this environment.

To sum up, in this system, we use SPV method as storage, Kademlia algorithm as query, and divide the nodes into full node, light node and SPV node. The full node is used to store all the data, while the light node is only used to store part of the data, and the SPV node only needs to confirm some transactions before shipping. This method simplifies the storage, so that some nodes do not need to store all the data.

3. Detailed Explanation of Traceability Process

3.1 Storage Design

In the storage design, for the problem that each node of the blockchain stores a large amount of data, we add light nodes and query nodes based on Satoshi Nakamoto's SPV (simplified payment verification) idea. The nodes are divided into four categories: full node, light node, SPV node and query node. The functions of each node are as follows:

The full node stores all the data, which is equivalent to a large land contractor. The function of the full node is to reach a consensus on the transaction. The structure of the transaction data is shown in Table 1. The NodeID is a binary number, which records the serial number of nodes through which the transaction passes, and is generated in the order of joining the blockchain system. After the transaction is submitted, the full node stores the packed transaction block in its own node, and also stores the block in the light node of the corresponding NodeID of the transaction information, so as to facilitate the quick query of the light node later.

Light nodes store data related to themselves and only deal with the transactions they have processed. If the full node is compared to a large contractor, then the light node is a small Contractor under the full node, with small scale and weak storage capacity. This kind of node does not participate in consensus, only accepts the transaction data related to itself sent by the full node. When the transaction request from other nodes is received, the information will be sent to the full node. Each light node will have a certificate. After the full node confirms that the certificate is correct, a consensus will be reached. After the consensus is confirmed, the information will be returned to the light node. The data query is mainly the data query on the light node, so as to reduce the total amount of data search and optimize the efficiency.

Table 1 Transaction structure

Transaction attribute	Attribute specification
Tx_id	Transaction batch number
TimeStamp	Time to operate here
Place	Specific place to process the transaction
TotalNum	The total volume of these trading products
Operation	Operation of this batch of products, such as shipment, processing, hardcover, etc
NodeID	Node IDs that the transaction passes through

SPV node, also known as simple payment verification node, stores the block head of light node block, and only needs to confirm the transaction. A light node corresponds to multiple logically related SPV nodes and is responsible for transaction storage of these nodes. The structure of the block head is shown in Figure 1. It is mainly confirmed by Bloom filter and Merkle tree. After a transaction is confirmed reasonably, it can be processed or shipped, which is equivalent to the grower of the planting area. SPV starts to send the get header command to its own light node to get the block head of the light node and verify the transaction. If the transaction exists, it will operate according to the operation in the transaction. If it does not exist, the transaction will not continue. The verification process is as follows:

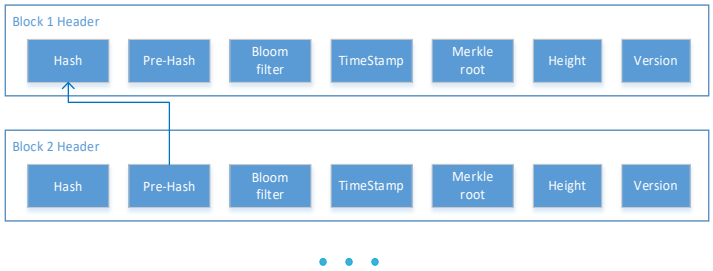

Figure 1 Block head structure

1) For a known transaction, the transaction information is hashed to get the transaction hash Tx_hash.

2) Tx_hash is calculated by Bloom filter to filter out the blocks in the light node that do not have this transaction.

3) For the remaining blocks after filtering, the node sends a request to the block to get the transaction hash value and the construction path needed by Tx_Hash to build Merkle_root.

4) Conduct a verification through the transaction hash and construction path obtained, if the result is equal to Merkle_root, the correctness of the transaction can be guaranteed.

In step 2, the fast location of the corresponding information block in the block is mainly realized by the Bloom filter in the block head of SPV node, as shown in Figure 2. Among them, Bloom filter is a spatial efficient probability data structure, which makes it establish a specific set of elements to support data query within the allowable range of error [8].

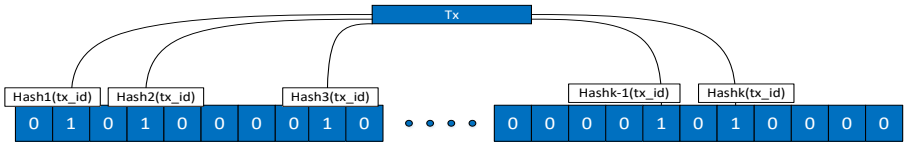

Figure 2 Structure diagram of Bloom filter

For each known transaction, k different hash functions are used to hash the transaction. Map the calculated values to the corresponding positions in the Bloom filter hash table, and set the values of these positions to 1. When searching again, we only need to hash again and compare the values at the corresponding positions in the hash table. If the values of K positions are all 1, we can judge that the transaction may be in this block, and realize the fast positioning of the corresponding block. The larger the K value is, the smaller the error is.

The query node is a node set up for consumers. The node is set up in the cloud. Consumers do not need to use their own computing power to query the corresponding nodes. They just need to scan the transaction batch number Tx_id and NodeID to the cloud server, and then query the target node through the routing table in the cloud node, and return the transaction information to consumers Now the traceability of products.

3.2 Query Method Design

The query between nodes mainly uses the improved Kademlia algorithm. Kademlia is based on a measure, which is defined by XOR or calculation between two nodes, and uses this distance to identify a node [9]. The distance is stored in a routing table. The table structure is shown in Figure 3. This table is called K-bucket. in K-bucket, all nodes are classified according to the logical distance. Bucket with number i represents all nodes with logical distance [2i,2i+1) from this node. The node distance is obtained by exclusive or calculation of ID between nodes.

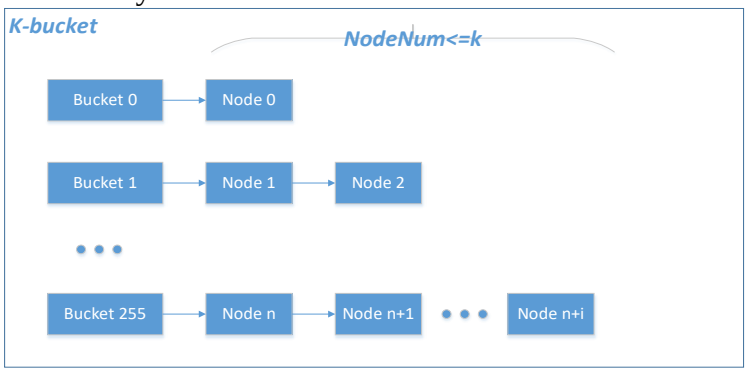

Figure 3 K-bucket structure

In this paper, we improve the Kademlia algorithm, design the data types of node nodes stored in each K-bucket, and add a count variable. The start of count is set to 0, and the Node is sorted according to count. If the Node is stored in K-bucket and the Node is the final node to be found, the count value will be increased by one, and then compared with the Node on the left. If it is larger than the count on the left, it will be traversed to the left. After finding the node greater than or equal to the count of the node, the node will be added to the right of the node. If the final Node found is not stored in the K-bucket, the node will be added to the K-bucket. After adding, the count value will be set to 1. For each query, select the first α nodes in the K-bucket to search, α is generally taken as the 3-most efficient. For 0110 nodes, the steps to find 1110 nodes are as follows:

1) XOR the two IDS, 0110 $\oplus$ 1110, and the result is 1000 (8) 10 as the distance between the two nodes.

2) According to the above description, 1110 nodes should be stored in the node No. 3, and access 0110 K-bucket No. 3 to obtain α nodes. If the number of nodes is less than α, then obtain from the nearby bucket.

3) If there is a target node in the acquired node, return the information of the target node in the

K-bucket, get the specific physical location of the node, obtain the relevant data and finish the search.

4) If the acquired node does not have a target node, XOR the target node with the acquired node, and then return to find α nodes from the k-bucket. For each returned node that does not exist in the K-bucket, initialize the count to 0 and add it to the bucket of the corresponding distance.

5) For each returned node, the original node will check whether these nodes are in its own K-bucket. If they do not exist, the K-bucket will be updated, and the count will be set as 1 and placed at the bottom of the bucket.

The search mode designed according to Kademlia algorithm is shown in Figure 4. For multiple queries of the same transaction, a K-V couchDB is set up in the query node to store the found data. The key value of each transaction process is stored in the database with Tx_id as the storage. The value contains all the transaction information of the same transaction batch number and is processed by time stamp Sorting: for each code scanning of consumers, first query the information in the database. If it does not exist, then send a message to other nodes to query the transaction, and then store the queried transaction data in couchDB according to the time stamp sequence.

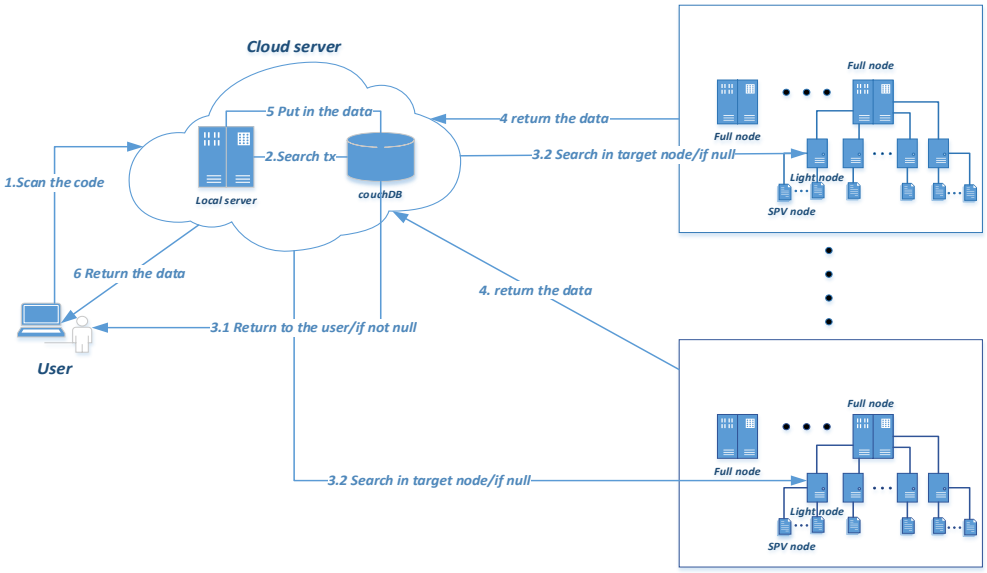

Figure 4 Traceability flow chart

The general process of traceability search is shown in Figure 4. The consumer submits the transaction batch number Tx_id and related node code NodeID to the cloud server by scanning the code.

In the cloud node, first query from couchDB according to Tx_id. If there is a relevant transaction, the relevant results will be returned to the consumer. If not, the transaction will be queried through Kademlia algorithm according to node number NodeID. After the relevant node is queried, the data will be searched in the block through Tx_id. After the transaction is queried, the data will be returned, and the queried data will be arranged according to the time stamp, stored in couchDB with Tx_id as the key, and then the relevant information will be returned to the consumer.

For the storage of transaction information, when the light node receives a transaction application, the full node selects the trusted node by consensus, submits the transaction to the chain, stores the transaction to itself and the corresponding light node, and then sends the message to the SPV node. The SPV node filters its block header through the bloom filter, and verifies the transaction content. After verifying that the transaction is reasonable, the goods can be delivered. If not, the goods will not be delivered.

For the generation of QR code, it is generated by the node that processes the transaction finally. Submit the transaction Tx_id and NodeID of all nodes through which the transaction passes to the full node. After the full node verifies the data, generate the QR code and bind it to the corresponding product. All NodeID that the transaction passes through is passed and accumulated

between nodes every time a transaction is made.

4. Future Work

This paper introduces a blockchain traceability system based on Kademlia query and SPV storage. On the basis of traditional blockchain traceability, the improved Kademlia query and SPV storage are used to solve the problems of query speed and data storage of blockchain itself, making it more suitable for the actual environment, and more efficient and rapid search.

However, there is no good solution for data redundancy. For many products that have been sold out, due to the characteristics that blockchain can't tamper with, we can't delete the corresponding data. With the accumulation of data, there will be a lot of useless waste for query and storage, and then we need to better find the data.

References

[1] Chen W, Liang X, Li J, et al. Blockchain Based Provenance Sharing of Scientific Workflows[C]//2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018: 3814-3820.

[2] Zheng Q, Li Y, Chen P, et al. An innovative IPFS-based storage model for blockchain[C]//2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI). IEEE, 2018: 704-708.

[3] Lin I C, Liao T C. A Survey of Blockchain Security Issues and Challenges[J]. IJ Network Security, 2017, 19(5): 653-659.

[4] Mohite A, Acharya A. Blockchain for government fund tracking using Hyperledger[C]//2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS). IEEE, 2018: 231-234.

[5] Paralkar K, Yadav S, Kumari S, et al. Photogroup: Decentralized Web Application Using Ethereum Blockchain[J]. Int. Res. J. Eng. Technol, 2018, 5: 489-492.

[6] Kim N H, Kang S M, Hong C S. Mobile charger billing system using lightweight Blockchain[C]//2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2017: 374-377.

[7] Hu S, Cai C, Wang Q, et al. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization[C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018: 792-800.

[8] Luo L, Guo D, Ma R T B, et al. Optimizing Bloom filter: Challenges, solutions, and comparisons[J]. IEEE Communications Surveys & Tutorials, 2018, 21(2): 1912-1949.

[9] Gattermayer J, Tvrdik P. Blockchain-based multi-level scoring system for P2P clusters[C]//2017 46th International Conference on Parallel Processing Workshops (ICPPW). IEEE, 2017: 301-308.