A High-Speed Data Retrieval Model on Blockchain

Jingang Yu[1]
[1]Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China

Yongkang Hou[1,2,*],
[1]Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
e-mail: flyingmonkey_hyk@163.com

Shu Li [3]
[3]Shenyang Ligong University, Shenyang 110158, China

Zhifeng Wen[1,2]
[1]Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China

2022 11th International Conference of Information and Communication Technology (ICTech) | 978-1-6654-9694-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICTECH55460.2022.00029

***Abstract*—Blockchain is also known as distributed ledger. All full nodes connected to the blockchain network participate in the maintenance of the data in the ledger. It is a technology in many fields such as computer science, cryptography, distributed storage, and finance. Industrial and academic research on blockchain technology has achieved great results, including research on blockchain networks, consensus mechanisms, and smart contracts. However, limited by the data storage mode and the characteristics of distributed storage at the bottom of the blockchain, there are still problems that need to be solved urgently, such as the single retrieval function and the low retrieval rate of the data retrieval on the blockchain. We focused on this problem, and based on the built-in index and external data warehouse method, we proposed a high-speed data retrieval model on blockchain. The model consists of three parts: a blockchain network with improved index storage, a data processing cluster, and application layer services. The new blockchain network improves the organization of transaction data in the traditional blockchain system, and designs a data structure suitable for high-speed retrieval to organize transaction data; the data processing cluster is responsible for ensuring data consistency and in accordance with high efficiency The synchronization strategy is to synchronize the data on the chain to the relational data warehouse under the chain; the application layer service encapsulates the rich query functions supported by the relational database, and finally provides services to the outside in the form of HTTP, RPC, etc. Experimental results show that the model can effectively expand the blockchain system in terms of query efficiency and query functions, improve the query rate of data on the blockchain, and meet people's needs for blockchain query functions.**

Keywords-blockchain; high-speed; retrieval; query; efficiency

I. INTRODUCTION

The development of blockchain technology can be traced back to the Bitcoin white paper "Bitcoin: A Peer-to-Peer Electronic Cash System"[1] published by Satoshi Nakamoto in 2008. Since then, the term "blockchain" has entered people's field of vision. In essence, the blockchain is a decentralized distributed ledger, and the ledger data is jointly maintained by all nodes connected to the blockchain network. Each node in the network is a transaction-driven state machine. The consistency of the nodes in the network is ensured by a consensus mechanism. After a user publishes a transaction in the network, it will be packaged into a block by the mining node, waiting for mining The node obtains the accounting right and publishes the block to the blockchain.

In today's Internet era, with the emergence of large amounts of data, the data query function and query rate on the blockchain have gradually exposed certain problems. When most blockchain projects are designed, they mainly focus on how to solve the problem of quickly reaching a distributed consensus in the process of data on the chain, while ignoring the problem of data retrieval on the chain caused by the emergence of a large amount of data.

At present, some practice has been carried out on the research of data query efficiency on the blockchain, Filecoin[2], storj[3] provide efficient solutions for distributed storage, and Polkadot[4] provides solutions for cross-chain communication. Literature [5] summarizes the current general solutions, and finally summarizes them into two. One solution is to optimize the data persistence mechanism at the bottom of the blockchain system, usually by improving the data organization form at the bottom of the blockchain or creating a new index layer on top of the original persistence mechanism [6]. Another solution is called an external data warehouse, which synchronizes blockchain data with limited query capabilities to a relational database with rich query capabilities. The main methods of external data warehouse can be summarized into two methods: synchronization and replication [7]. (1) Synchronization refers to the operation initiated by the data warehouse. Literature [8] proposed etherQL, which is an external query layer for the Ethereum [9] platform. The external relational database enriches the

978-1-6654-9694-0/22/$31.00 ©2022 IEEE
DOI 10.1109/ICTech55460.2022.00029

query function of the system. Literature [10] proposes a verifiable query framework vChain, which divides nodes into full nodes, light nodes and miner nodes, and uses cryptography to ensure the correctness and completeness of the query results received by light nodes. (2) Replication refers to the operation initiated by the blockchain network itself. The blockchain system is designed to take into account that the release of a new block will trigger the replication operation of the transaction data of the block. For example, chainSQL [11] uses the Ripple [12] network to write data in the form of playback. Into the database, FalconDB [13] uses a database server with an accessible verification interface on the client side, and stores digests on the blockchain for query/update authentication, so that a single user can participate in collaboration with high efficiency and low storage cost. Use blockchain technology to ensure the safety and trustworthiness of nodes.

Most of the blockchain systems, including the Bitcoin system and the Ethereum system, use Leveldb[14], which is a write-intensive database system specially designed by Google for write-intensive scenarios. We design the index based on the characteristics of the data on the chain, sort the leaf nodes of the Merkle Tree [15] index structure, and introduce Bloom filters [16] in the Merkle Tree to effectively avoid the drawbacks of Leveldb in terms of query efficiency. In response to the needs of enterprise-level data query, we designed a data synchronization cluster on the chain, through the synchronization server to synchronize the data on the chain to the local high-availability data warehouse according to the synchronization strategy, breaking the bottleneck of the data retrieval speed on the chain, and improving the data query Efficiency, enrich the data query function, and finally complete the design of the high-speed data retrieval model on the chain. Our contributions are as follows:

1) Improved the organization of transactions in the original blockchain network, introduced Merkle Tree with Bloom filter, pruned data retrieval operations on the chain, and greatly improved the efficiency of data retrieval.
2) Improve the structure of Merkle Tree, arrange the leaf nodes of Merkle Tree in order according to the hash value of the transaction, which can complement the false detection of the Bloom filter while adding the function of verifying the non-existence of a certain transaction information.
3) The off-chain query module is added. By synchronizing the data on the chain to the off-chain, functions such as range query and fuzzy query are added. Improve the data query rate, expand the data query query function, can effectively meet the needs of enterprise-level efficient query.
4) The data synchronization server monitors the changes of the blocks on the chain in real time, synchronizes the data on the chain to the relational database under the chain, and designs an efficient data synchronization strategy to solve the problems caused by the bifurcation and delay of the blockchain. Data inconsistency issues.

II. MODEL DESIGN

The overall architecture of the new high-speed data retrieval model on the chain proposed in this paper includes three parts: a sorted merkle tree-based blockchain network, data processing clusters, and application layer services. The overall architecture is shown in Figure 1. The blockchain network based on sorted merkle tree has made two improvements on the basis of merkle tree, (1) sorting the transaction information in the merkle tree. (2) Bloom filter is added to improve the efficiency of data query; the data processing cluster synchronizes the data on the chain to the relational data warehouse under the chain according to an efficient synchronization strategy, and solves the problem of data consistency, which is provided by the relational data warehouse. Data-rich query function; application layer service based on relational data warehouse provides data query function to users in the form of HTTP and RPC calls.

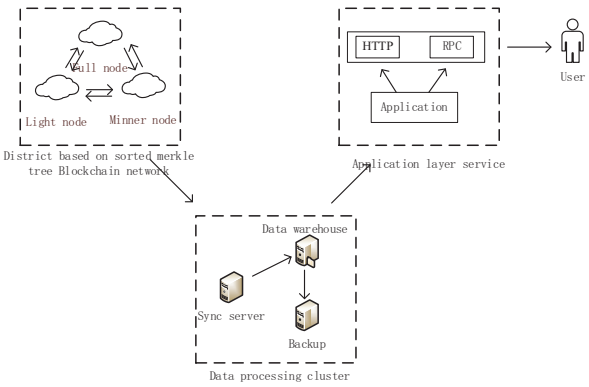

Figure 1. Overall system architecture

A. Improved Merkle Tree design

In traditional blockchain systems (such as the Bitcoin system), transaction data is usually organized in the form of Merkle Tree, and Merkle Tree is a general storage method for key-value data, with a single query function and a slow query rate. And other shortcomings. In addition, Merkle Tree only supports querying an existing transaction data, and when you want to quickly determine a non-existent transaction data, you need to traverse the entire blockchain to find that the transaction data does not exist, and the time complexity is high.

Aiming at the above shortcomings, this article improves Merkle Tree in two aspects. First, sort the Merkle Tree according to the hash value of the transaction data, so that you can quickly determine that a certain piece of data does not exist in the Merkle Tree. Second, on the basis of Merkle Tree, a Bloom filter is added to quickly locate existing transaction data, which greatly improves the efficiency of data query.

Bloom filter essentially consists of a fixed-size binary vector or bitmap and a series of mapping functions, and is mainly used to determine whether an element is in a set. When an element is added to the set, the variable is mapped

to K positions in the bitmap through K mapping functions, and they are set to 1. When querying an element, if one of these positions is not 1, the element being queried must not exist, and if these positions are all 1, it may exist. Note that the "may exist" here is because some hash functions may generate hash collisions, which leads to the same mapping position of two elements in the Bloom filter. Therefore, the Bloom filter has a property, it may be falsely detected, but it will never be missed.

Sorted Merkle Tree refers to sorting the hash values of the Merkle Tree leaf nodes to obtain an ordered Merkle Tree. When retrieving data, binary search and other methods can be used to reduce time complexity and improve data query efficiency. At the same time, it complements the false detection of the Bloom filter to avoid the occurrence of missed detection. The improved Merkle Tree structure is shown in Figure 2.

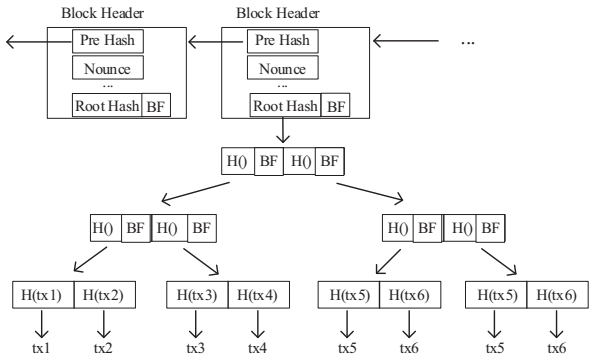

Figure 2. Improved merkle tree structure

The transaction data packaged in each block corresponds to tx1, tx2...txn of the leaf node, and these transaction information can be regarded as a set T(tx1, tx2,..., txn). The input of the query operation consists of two parts, (1) query the target block range (2) query transaction data. Sorted Merkle Tree based on Bloom filter searches for target transaction data according to the steps of Algorithm 1 to ensure the efficiency of data query on the chain. This process can also be understood as the process of providing Merkle Proof.

Algorithm 1 Sorted Merkle Tree search algorithm based on Bloom filter

Input: query target block range BSet, query transaction data target

Output: transaction data result

```
1.while IsEmpty(BSet)
2.   node = First(BSet).Root
3.   while node NotIn leafNode
4.     if BFHandler(node.BF,target)
5.        if BFHandler(node.Left.BF,target)
6.           node = node.Left
7.        end if
8.        if BFHandler(node.Right.BF,target)
9.           node = node.Right
10.       end if
11.    end if
12.  end while
13.  if node.Value == target && node IsIn leafNode
14.      Add(Result,node.hash.tx)
15.  end if
16.  RemoveFirst(BSet)
17.end while
18.Return Result
```

Analyze the process in algorithm 1, the first line traverses each block to be queried; the second line initializes the node node and takes the first element in the set BSet; lines 3-12 follow the Merkle filter results according to the Bloom filter Tree queries downwards until the leaf node; lines 13-15 add the query results to the Result set; line 16 deletes the first element in the block set BSet, and starts to traverse the next block. Finally, each element in the block set is traversed, and the result is collected in the Result set and returned.

B. Data processing cluster design

The data processing cluster includes three types of servers: data synchronization server, relational data warehouse and database backup. The main function of the data synchronization server is to synchronize the data on the chain to the off-chain high-availability relational data warehouse in accordance with the data synchronization strategy, and to solve the problem of data inconsistency caused by the blockchain bifurcation through the caching mechanism. The relational data warehouse is mainly used as the data source of application layer services, and it is the basis for providing users with rich query functions and high-speed retrieval efficiency. Database backup is the hot backup of relational data warehouse to prevent data loss.

The specific process is as follows: (1) The synchronization server cyclically monitors the state of the blockchain. (2) When a new block is published on the blockchain, the new block is loaded into the buffer queue. (3) Determine whether the queue length is greater than 6. If it is greater than 6, continue to the next step; otherwise continue to monitor the state of the blockchain. (4) Judge whether the team leader element is on the longest legal chain. If it is, the block data is cleaned, packaged, and persisted to the local data warehouse, and the head element is discarded; otherwise, the opponent element is directly discarded.

After the hot backup server detects changes in the local relational data warehouse data, it backs up the local relational data in real time. Finally, the data on the longest legal chain is persisted to the local data warehouse, and the local data warehouse can provide users with rich query functions. The specific process of the data synchronization strategy adopted by the synchronization server is shown in Figure 3.

If every time a block is added to the blockchain, the synchronization server synchronizes the block to the local data warehouse, which will cause certain problems. For example, the synchronization server has just synchronized a newly generated block to the local data warehouse. At this time, the blockchain has forked, and the newly synchronized block will become an obsolete block. Therefore, the fork of the blockchain is required. To process. This article introduces a caching mechanism. After the synchronization server

detects the new block, it will not immediately cache the new block to the data warehouse, but load it into a cache queue, and wait for the blockchain system to determine after six blocks. After becoming the longest legal chain, the block data on the longest legal chain is then cleaned, packaged and persisted to a local relational database.

C. Application design

Application layer service is a data query interface designed to meet the high-speed retrieval of enterprise-level application data. It is a data query interface designed for local high-availability relational databases and provides users with rich query functions in the form of HTTP and RPC.

The data processed by the application is mainly divided into three parts: block header, block body and transaction data. The data that users care about is mainly stored in the transaction data. The data in the block header and block body mainly provide Merkle Proof for the transaction data to ensure that the transaction data has not been tampered with.

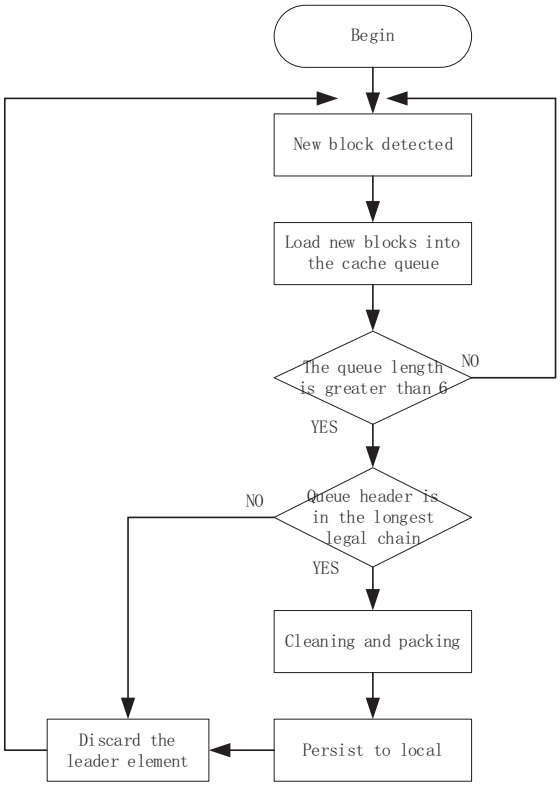

Figure 3. Data synchronization strategy process

III. Analysis and Experiment

A. Experimental design

Experimental environment: Blockchain 1 virtual machine 192.168.223.50, cluster 2 virtual machines 192.168.223.51 and 192.168.223.52, application layer service 1 virtual machine 192.168.223.53, a total of 4, all set static IP mode.

The experiment is mainly divided into two groups: (1) Comparison of Merkle Tree query rate. The traditional Merkle Tree is used to organize transaction data and the improved Merkle Tree is used to organize transaction data, and the query speed of transaction data is compared. (2) Off-chain data query verification, using the traditional blockchain system and the method proposed in this article to query different transaction quantities, and compare the results.

B. Merkle Tree Retrieval rate

This paper designs a set of experiments to compare the running time of Sorted Merkle Tree based on Bloom filter with the traditional BTC system. The result is shown in Figure 4. As the transaction data increases, it will show a linear growth trend (note that the abscissa transaction is exponentially processed, so it shows an exponential growth situation). The optimized Merkle Tree method we proposed showed a rapid growth at 1024 and 2048. It can be speculated that the main reason is the misdetection of the bloom filter.

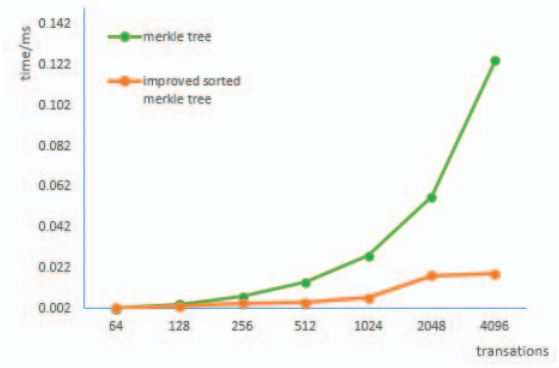

Figure 4. Merkle Tree Retrieval rate

C. Blockchain data throughput

This article first tests the throughput (qps) of the traditional BTC system to obtain block data and obtain TX data, and use this result as the baseline. Using the method of comparative experiment, we use our proposed method to obtain Block data and TX data of the same magnitude, and compare the result with the baseline. The result is shown in Figure 5. We are familiar with relational databases, which can usually reach a million-level throughput. The analysis system here is far from reaching the level of ordinary relational data warehouses. There are two main reasons: (1) The persistence method of the blockchain system itself is regional The bottleneck of data query efficiency on the blockchain. (2) The synchronization strategy implemented by the data synchronization server must ensure that the data on the blockchain system and the local relational data warehouse are in a consistent state, and solve the potential problems of the data (fork, delay, verification, etc).

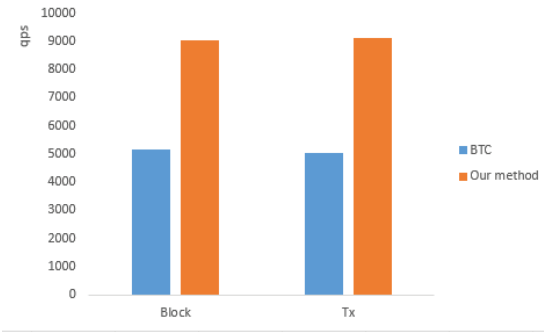

Figure 5. Query throughput

IV. Conclusion

With the massive increase of data on the blockchain, people's demand for high-speed retrieval of data on the chain is increasing. Limited to the key-value type of persistent databases of the blockchain, the traditional blockchain structure that organizes transaction data based on the Merkle Tree data structure can no longer meet the needs of high-speed queries. In this paper, by improving the Merkle Tree structure, on the one hand, the Bloom filter is introduced to prune the data query, and on the other hand, the Merkle Tree is sorted according to the hash value of the transaction data to reduce the time complexity of the query process and filter the Bloom Quickly locate the misdetection of the detector. In order to expand the blockchain query function, this paper introduces a data synchronization server, and designs an efficient synchronization strategy for data consistency issues. After cleaning and packaging the data on the chain, the data on the chain is synchronized to the local relational data warehouse. The application exposes the service to the user. The experimental results show that the high-speed data retrieval model on the chain proposed in this paper can effectively improve the data query rate on the blockchain in terms of throughput, batch retrieval of transaction data, and batch acquisition of block header information.

References

[1] NAKAMOTO S.Bitcoin.BitCoin:A peer-to-peer electronic cash system [EB/OL].https://bitcoin.org/bitcoin.pdf

[2] PROTOCOL LABS.Filecoin:A Cryptocurrency Operated File Storage Network[EB/OL].https://filecoin.io/filecoin.pdf

[3] SHAWN W.Storj A Peer-to-Peer Cloud Storage Network[EB/OL].https://storj.io/storj.pdf

[4] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. White Paper,2016

[5] Wang Qiange, He Pu, Nie Tiezheng, Shen Derong, & Yu Ge. (2018). Overview of data storage and query technology in blockchain systems.Computer Science, 45(012), 12-18.

[6] Zheng Haohan, Shen Derong, Nie Tiezheng, & Kou Yue. Query optimization of blockchain system for hybrid index. Computer Science, 47(10), 8

[7] Zhu Yanchao. Research on query processing for blockchain systems. 2020. East China Normal University, PhD dissertation.

[8] Yang, L., Kai, Z., Ying, Y., Qi, L., & Zhou, X. (2017). Etherql: a query layer for blockchain system. International Conference on Database Systems for Advanced Applications

[9] BUTERIN V.Ethereum:A next generation smart contract and decentralized application platform [EB/OL]. https://github.com/Ethereum/wiki/Whitecom/Ethereum/wiki/White-Paper

[10] Xu, C., Zhang, C., & Xu, J. (2018). Vchain: enabling verifiable boolean range queries over blockchain databases. [C]//Proceedings of ACM International Conference on Management of Data.2019:141-158

[11] Chainsql[EB/OL].http://www.chainsql.net/

[12] SCHWARTZ D, YOUNGS N, BRITTO A, et al. The ripple protocol consensus algorithm[R]. White Paper,2014.

[13] PENG Y, DU M, LI F, et al. Falcondb: Blockchain-based collaborative database[C]//Proceedings of the 2020 ACM International Conference on Management of Data.2020:637-652

[14] Leveldb[EB/OL].https://github.com/google/leveldb

[15] MERKLE R C. A certified digital signature[C] //Annual International Cryptology Conference. 1989: 218-238

[16] Xiao Mingzhong, & Dai Yafei. (2004). Overview of Bloom filter and its application. Computer Science, 31(004), 180-183.