An Efficient Data Query Method of Blockchain Based on Index

2021 7th International Conference on Computer and Communications (ICCC) | 978-1-6654-0950-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCC54389.2021.9674708

Mingmin Liu
Engineering Research Center of Information Network, Ministry of Education
School of Computer Science (National Pilot Software Engineering School)
Beijing University of Posts and Telecommunications
liumingmin@bupt.edu.cn

Hongman Wang*
Engineering Research Center of Information Network, Ministry of Education
School of Computer Science (National Pilot Software Engineering School)
Beijing University of Posts and Telecommunications
wanghm@bupt.edu.cn

Fangchun Yang
Engineering Research Center of Information Network, Ministry of Education
School of Computer Science (National Pilot Software Engineering School)
Beijing University of Posts and Telecommunications
fcyang@bupt.edu.cn

***Abstract*—In many blockchain applications, apart from storing the original data, it is also necessary store the data operation records in the blockchain for audit or security. Yet, the traditional blockchain, generally cannot meet the user's demand for data query. In order to solve this problem, we propose a novel block structure, the block body into an index layer and a data layer. On this basis, we further develop two in the index layer to aggregate the data and its operation records for efficient query. Specifically, we first build an index for the original data in blockchain by desigining the Abstract-Trie, which has efficient and stable query features. Secondly, we design another index called Operation-Record List to link the operation records to the corresponding original data, and store the first node of each list in the Abstract-Trie, so as to achieve the fast acquisition of the operation records of the original data. Simulation results based on Hyperledger Fabric verify the effectiveness of our optimized block structure in querying of the original data and its operation records in the blockchain.**

Keywords—Abstract-Trie, Blockchain, Operation-Record List, Query Efficiency

I. INTRODUCTION

Owing to the success of tranditional blockchain such as Bitcoin[1], Ethereum[2] and Hyperledger Fabric[3], blockchain technology has attracted much attention from both academia and industry. With the wide application of blockchain technology in distributed systems, how to efficiently obtain data in blockchain has become a new research hotspot. As shown in Fig.1, for the traditional blockchain, data is stored in each node of the network in the form of blocks. A block consists of a block header and a block body, where the block header mainly stores the basic information, such as the hash value of the previous block, and the block body stores the actual data. It is worth mentioning that the white paper of blockchain technology does not strictly define the organization form of the data, which suggests that we should traverse all the data in each block in order to acquire the required data.

In terms of data query of the traditional blockchain, the query operation can be carried out in two steps. Firstly, we need to traverse every block by previous hash pointer(preHash in Fig.1), then we can scan all the data(tx in Fig.1) in each block until find the required data. However, with the wide application of blockchain technology in data-intensive scenarios such as data sharing and data traceability, the above-mentioned inefficient data query method can no longer meet user's demand. In addition, in many blockchain applications, apart from storing the original data, application designers always store the data operation records in the blockchain for audit or security reasons. Against this backdrop, it is necessary to improve the query efficiency of the original data and its operation records in blockchain.

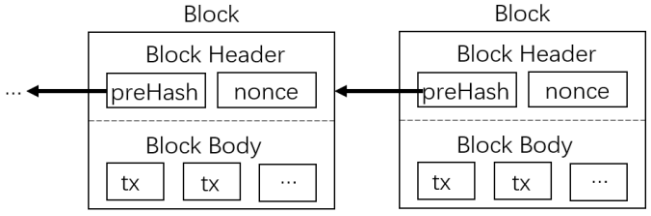

Fig. 1. Tranditional BlockChain Structure

Aiming at improving the inefficiency of data query in traditional blockchain, many methods and strategies have been proposed[4]-[12] for querying genetic data, government Information, and Electronic Certificate in blockchain[4]-[6], but these customized optimization schemes cannot be directly applied to other scenarios. In this regard, many scholars studied more general solutions to improve the efficiency of data query in blockchain. For example, many have tried to develop blockchain database solutions to support SQL-like queries[7]-[9]. Although these solutions can greatly improve the query efficiency, all of them assumed the existence of a trusted party which is contrary to the blockchain itself. In addition, Wood[2] and Xu Yuqin[10] improved the query efficiency by introducing Merkle Patricia Trie to build index for data in blockchain, which motivated for the design of the Abstract-Trie in this paper. However, these strategies adopted non-clustered index, data is separated from index storage, which will lead to data inconsistency to a certain extent. Also, they are not beneficial to batch query in blockchain, such as querying all historical operation records of an original data. In contrast to these works, Zeng Lei[11], Xu Cheng[12] and Zhang Ce[13] respectively develop different clustered index to improve the query efficiency in blockchain. Among which, the vchain proposed by Xu Cheng has the best performance and is

more universal. It develops ADS and introduces skip list to implement efficient query of any type of data in blockchain. However, this solution is still not perfectly applicable to data-intensive scenarios, because they still cannot support the efficient acquisition of historical operation records of the original data.

In order to futher improve the query efficiency of original data and its operation records in blockchain, we propose an novel block structure, and develop clustered indexes for the original data and its operation records respectively. Specifically, we first build an index for the original data in the blockchain by desigining the Abstract-Trie, which has efficient and stable query features. Secondly, we design another index called Operation-Record List to link the operation records for each original data, and store the first node of each list in the Abstract-Trie, so as to achieve the fast acquisition of the operation records of the original data. The results of emulational experiments show that compared with the traditional blockchain, the optimized block structure in this paper is more efficient to query the original data and its operation records.

II. METHOD

In this paper, an block structure is presented for the query optimization in blockchain. The block structure contains two novel indexes, namely, the Abstract-Trie and the Operation-Record List, which aims at improving the query efficiency of the original data and its operation records in blockchain. Incidentally, the indexes mentioned above are all stored in the block where the data is located, which can avoid the data insecurity caused by data inconsistency. A more detailed description of the proposed optimization scheme is as follows.

A. Optimization Method of the Original Data Query

In many scenarios, users need to check the validity of data frequently by judging whether its abstract exists is in the blockchain. In order to solve the problem that the traditional blockchain can't meet the query demands of the original data in many scenario, we develop an index for the original data by the Abstract-Trie. Specifically, we calculate hash values for the original data scattered in different blocks, and build a Abstract-Trie with these hash values as keys. The Abstract-Trie is a patricia trie, which guarantees of efficient and stable query as well as good local update. Because of these features, we can achieve efficient queries of the original data in blockchain at the expense of a little block storage space. In addition, the Abstract-Trie designed in this paper is a kind of clustered index, which is stored in the same block with the original data together. This design scheme ensures the consistency between the index and the original data, which ensures the data security in blockchain.

Figure 2 shows the structure of the block in blockchain after the introduction of the Abstract-Trie. Similar to the block stucture of traditional blockchain (as shown in Figure 1), the block is composed of block head and block body. The difference between them is that the block body is further divided into the index layer and the data layer, whereas the Abstract-Trie is stored in the index layer. The Abstact-Trie is a tree-like index structure, which organizes the original data abstracts scattered in different blocks in the form of trie, so as to implement the efficient acquisition of the original data in blockchain with the help of the fast and stable query features of trie. The Abstract-Trie contains three types of nodes, including index nodes (white nodes in Figure 2), data nodes (gray nodes in Figure 2) and external nodes (dotted nodes in Figure 2). The index node is a non-leaf node, which stores the intermediate state of the Abstract-Trie and links all the next nodes through the childs field. The data node is the node hit during query, which records the abstract of the original data through the abstact field. The external node is a special node, which plays a bridging role in the Abstract-Trie.

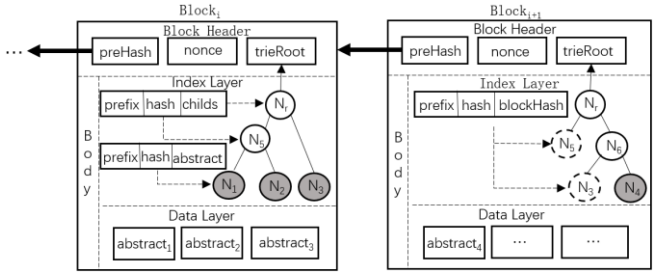

Fig. 2. Abstract-Trie Index Structure

This node can point to the index node or data node located in other block through the blockHash field. With this bridge approach, in each block, we can maintain only the portion of the index structure associated with the original data contained in the current block, rather than maintaining the full Abstract-Trie, which greatly reduces the memory overhead associated with building indexes. In addition, all the above nodes contain prefix and hash fields. The prefix field represents the common prefix from the root to the current node, which is used to assist the construction and query of the original data. The hash field represents the hash value of the current node and is used to verify the validity of the node.

Algorithm 1: Abstract-Trie Construction

```
Function: buildAbstractTrie(abstracts)
Input: list of data abstract in the new block
1. nodes = [];
2. for abstract ∈ abstracts
3.    node = new DataNode();
4.    node.prefix = hash(abstract);
5.    node.abstract = abstract;
6.    node.hash = hash(node);
7.    nodes.add(node);
8. end for
9. oldRoot = rootInTrie();
10. newRoot = new IndexNode();
11. newRoot.childs = transfer(oldRoot.childs);
12. for node ∈nodes
13.    insertIntoTrie(newRoot, node);
14. end for
15. abstractRoot = newRoot;
```

The detail about the construction and update of the Abstract-Trie in blockchain is described in Algorithm 1. Whenever a new block is generated, we will construct or

update the Abstract-Trie for the original data contained in the new block. The detail steps of the Abstract-Trie construction and update are as follows.

- We create new data nodes for all the original data in the new block, and assign values to the fields of these nodes.
- We obtain the root of the Abstract-Trie in the previous block by rootInTrie method, denoted as oldRoot, which will be null if it does not exist.
- We create a root for the new block, named newRoot, which is initialized as an external node pointing to oldRoot, so that we have maintained and tracked the existing index structure.
- We insert all the data nodes newly created in step 1 into the Abstract-Trie by insertIntoTrie method. It is worth noting that after all data nodes are inserted into the Abstract-Trie, the hash field values of all newly-created nodes need to be recalculated facilitate subsequent verification of the integrity and validity of these nodes.
- Finally, we calculate the hash value for the new newRoot created in step 3 and assign it to the abstractRoot field in the block header, so as to obtain the Abstract-Trie from the block header.

Algorithm 2: Query with Abstract-Trie

Function: abstractTrieQuery(hash)
Input: the hash value of the quering abstract
Output: query result R, verification object VO
1. trieRoot = rootInTrie ();
2. pNode = trieRoot;
3. **while** i ∈ hash **and** pNode != null
4. if pNode is an external node
5. pNode = getIndexNode(pNode.blockHash,
6. pNode.prefix);
7. add <$hash_{pNode}$, $hash_{childs}$> to VO;
8. index = hash.indexOf(i);
9. pNode = pNode.childs[index];
10. **If** pNode != null and pNode.prefix matchs hash
11. R = pNode;
RETURN <R, VO>

With the help of the Abstract-Trie, we can implement fast and efficient query efficiency of the original data in blockchain. The detailed steps of the query operation using the Abstract-Trie are described as follows.

- We obtain the root of the Abstract-Trie in the newest valid block by rootInTrie method. If it does not exist, there is no data in the blockchain currently.
- With the help of the Abstract-Trie, we can get the data node hit by the inputed hash value of the original data. Specifically, we use each character of the input hash value in turn to continuously obtain the next node until we find the data node. At the same time, We need to hash the current node and its children as part of the verify object(VO in Algorithm 2) for legitimacy proof.
- We can judge whether the query operation is successful through the prefix field of the above data node. If it is equal to the input hash value, the above data node(R in Algorithm 2)and the verify object(VO in Algorithm 2) will be returned; otherwise, it means that the required original data doesn't exist in the blockchain.
- Finally, the third query party can obtain the required original data by parsing the above data node(R in Algorithm 2), and verify the validity and integrity of the data with the help of VO.

B. Optimization Method of the Operation Records Query

In many blockchain applications, apart from storing the original data, it is also important to store the data operation records in the blockchain for audit or security. In order to improve the query efficiency of data operation records in the blockchain, we design the Operation-Record List to develop an index for the data operation records scattered in different blocks. Specifically, we first classify the data operation records in the same block according to the operated original data, and then link the same type of data operation records in different blocks by pointers. The Operation-Record List is a single linked list, which is simple and efficient, that is, it can link the same data operation record classes scattered in different blocks with a little pointer space overhead. In addition, by storing the head of the Operation-Record List in the Abstract-Trie, we can use the efficient query feature of the Abstract-Trie to quickly obtain the head of the required Operation-Record List, and then get the required records by traversing the Operation-Record List.

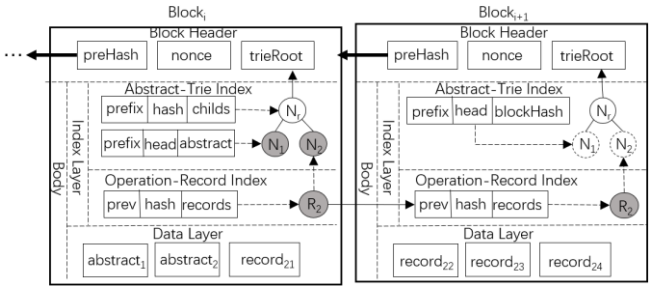

Fig. 3. Operation-Record Index Structure

Figure 3 shows the structure of the block in blockchain after the introduction of the Operation-Record List. Similar to the block structure shown in Figure 2, the block considered here consists of block header and block body, where the block body can be further divided into the index layer and the data layer. The difference between them is that the index layer of the new block contains the Abstract-Trie and the Operation-Record List together. The Operation-Record List is a cross-block list, which has two functions, classifying the data operation records in the same block according to the operated original data and linking the same type of data operation records in different blocks. To implement the above functions, the nodes in the Operation-Record List contain three fields: prev, records and hash. Among them, the prev field is used to link the same type of data operation records between different blocks, while the records field and the hash fild are used to record the same type of operation records in each block and verify the validity respectively. In addition, we add the head

field for the leaf nodes in the Abstract-Trie to record the head of each Opration-Record List, so as to implement the efficient acquisition of the Operation-Record List by utilizing the fast and stable query characteristics of theAbstract-Trie.

Algorithm 3: History-Record-List Construction

Function: buildRecordList(records)
Input:list of data operation records in the new block
1. recordGroup = partitionBySource(records);
2. **for** records ∈ recordGroup
3. dataNode = abstractTrieQuery(group.hash);
4. oldHead = dataNode.head;
5. head = new IndexNode();
6. head.prev = oldHead;
7. head.records = records;
8. head.hash = hash(head.prev | head.records);
9. updateAbstractTrie (head);
10. **end for**

The detail about the construction and update of the Operation-Record List in blockchain is described in Algorithm 3. Whenever a new block is generated, we will construct or update the Operation-Record List for the data operation records contained in the new block. The detail steps of the Operation-Record construction and update are as follows.

- We classify the data operation records in the new block according to the operated original data by partitionBySource method, so that the operation records of the same original data are classified into the same class.
- For each class of data operation record, we query the corresponding data node in the Abstract-Trie by abstractTrieQuery method and get the head of its Operation-Record List, named as oldHead, which will be null if it doesn't exist.
- For each class of data operation record, we create a new index node, let the records field store the corresponding operation records, and let the preNode filed point to the above oldHead in step 2.
- For each newly-created index node in step 3, we regard it as the newest head of each Operation-Record List and then assign it to the head field of the corresponding node in the Abstract-Trie, which will help us implement efficient acquisition of the Operation-Record List.

Algorithm 4: Query with History-Record-List

Function: recordListQuery(hash)
Input: the hash value of the relevant abstract
Output: query result R, verification object VO
1. < dataNode , VO1> = abstractTrieQuery(hash);
2. head = dataNode.head;
3. pNode = head;
4. **while** pNode != null
5. R.add(pNode.records);
6. VO2.add(<hash(pNode.prev), pNode.hash>);
7. pNode =getNode(pNode.prev, hash);
8. VO = <VO1, VO2>;
RETURN <R, VO>

With the help of the Abstract-Trie and the Operation-Record List, we can implement fast and efficient query efficiency of the operation records of the specific original data in blockchain. The detailed steps of the query operation are described as follows.

- We first receive the hash value of the original data associated with the operation records to be queried, and then obtain the correspond node and its verify Object(VO1 in Algorithm 4) from the Abstract-Trie by excuting algorithm 2.
- We extract the head of the corresponding Operation-Record List from the above node and record it as head.
- We traverse the whole Operation-Record List sequentially from the head through the prev pointer of each node, and the operation records of the traversed node are added into the result set R. Meanwhile, the hash values of each node and their previous node are added into a new verify object(VO2 in Algorithm 4) as the validity proof.
- Finally, we calculate the verify object of thie query operation by combining the above VO1 and VO2, and return it together with the query result set R.

III. EXPERIENCE

In order to verify the validity of the new block structure proposed in this paper, we have constructed a prototype experiment based on Intel Core I7 CPU and 8GB RAM. The blockchain network is built with the 2.1 version of Hyperledger Fabric, and it consists of two peer nodes and one sort node, each of which has 1GB RAM and 40G hard disk size. In addition, we implement the proposed block structure logically by designing smart contract, and evaluate the performance of the proposed block structure by experimental comparison with the traditional Hyperledger Fabric and vchain, which is proposed by Xu Cheng .

A. Dataset

The data set used in this paper comes from the papers of blockchain theme published in 2021 searched on Google Scholar, and the construction method of the data set is as follows. In order to ensure the stability of the operation records of each data, 1000 papers were extracted from the above-mentioned paper set, and the number of citations of each selected paper was set to 9.Then, we regard each citation of the paper as an operation, and construct a metadata for each paper or operation. It should be noted that the size of metadata will be 1KB by filling characters. Based on this, we obtained an experimental data set containing 1000 metadatas of the original paper and 9000 records of paper citation. Then, we divide the above 1000 metadatas of the original paper into 100 groups, and ensure that the data size of each group is 100KB by filling the citiation records. Finally, we build blocks for the above 100 groups and store them in the blockchain network which consists of two peer nodes and one sort node.

B. Evaluation Metrics

In order to evaluate the overhead and performance of the Abstrct-Trie and the Operation-Record List, we use three indexes to evaluate: block construction time, paper and its citiation metadata query time. Particularly, we can evaluate the cost of the new indexes of the proposed block by the first indicator, and evaluate the performance optimization of the original data query and operation records query by the other indicators.

C. Experimental Results and Analysis

According to the above three evaluation indexes, we have evaluated the performance of the optimized Fabric network through three groups of experiments, and compared with the traditional Fabric network and vchain[12]. Among them, the Fabric is a traditional blockchain system without external index, which has linear time query complexity, and the Vchain is a blockchain system optimized by introducing skipList, which has logarithmic time query complexity. As shown, compared with the traditional Fabric network, the optimized Fabric in this paper needs hundreds of milliseconds of block construction time to maintain the indexes between blocks, but at the same time, it can obtain several times of performance improvement when querying paper and its operation records in blockchain. In addition, compared with the vchain, the improved Fabric network also has certain advantages in the performance of the paper metadata query, and the performance of quering the paper operation records has been improved by nearly 2 times. The following is a detailed description of the experimental process and results.

1）Comparison of block construction time

In order to evaluate the performance cost of the Abstract-Trie and the Operation-Record List proposed in this paper, we conducted four experiments on the grouped data sets mentioned above. Specifically, we perform block building operations for each group of data in traditional Fabric, improved Fabric in this paper and vchain respectively, and record the average time spent building every 10 blocks. The comparison of experimental results is shown in Figure 4.

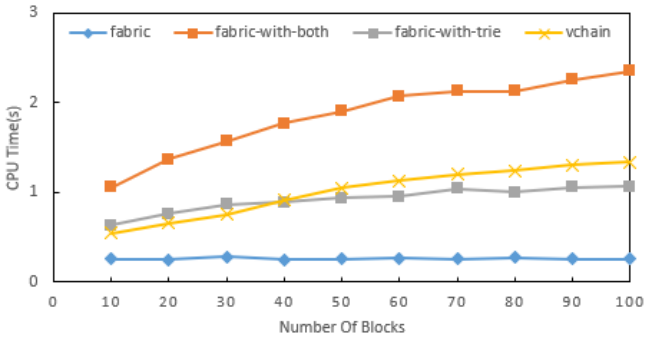

Fig. 4. Block Construction Performance

As shown in Figure 4, compared with the traditional Fabric, the block construction time has increased in the improved Fabric in this paper. Specifically, the introduction of the Abstract-Trie and the Operation-Record List increases the spent time of block building by about 400ms and 800ms on average, respectively. It should be noted that the time-consuming effect brought by the introduction of the Abstract-Trie will gradually stabilize with the increasement of the number of blocks, which is determined by the fixed number of paper metadatas contained in a single block. On the other hand, the time consumed of block building in the Fabric with the two indexes is about twice as long as than the Fabric with the Abstract-Trie, which is caused by the almost equal time cost on indexing the paper and its operation records metadata, and the number of paper metadatas and operation records classes contained in each block are almost equal in this experiment (the ratio of them is 10:9). Compared to the vchain, the improved Fabric in this paper takes hundreds of milliseconds longer to build blocks. In addition, in the experiment of this paper, we only consider the time-consuming of building blocks in a single node, but do not consider the process of spreading blocks in the blockchain among all nodes. When we consider the complete data uplink process, the time cost increasement of the improved Fabric in this paper will be less significant.

2）Comparison of the paper metadata query time

In order to evaluate the performance improvement of the paper metadata query in the Fabric optimized by the Abstract-Trie, we conducted three experiments on the grouped data sets mentioned above. Specifically, we execute the paper metadata query in the traditional Fabric, vchain and improved Fabric with the Abstract-Trie respectively. For each query request, it contains 50 random metadata hashes to be queried. Finally, the average query time of a single query request in different blocks is recorded in each experiment, and the comparison of experimental results is shown in Figure 5.

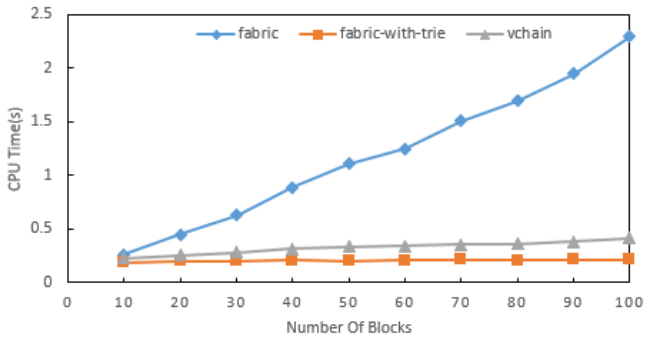

Fig. 5. Abstract Query Performance

As shown in Figure 4, compared with the traditional Fabric, after introducing the Abstract-Trie, the query efficiency of the paper metadata in the blockchain has been improved several times. Specifically, with the increment of the number of blocks in the blockchain network, the promotion effect brought by the Abstract-Trie is becoming more and more obvious, because the query method of the traditional Fabric determines that the time consumption for querying the paper metadata will increase linearly with the increment of the number of blocks, while the time cost of the improved Fabric with the Abstract-Trie is stable, which is determined by the depth of the trie. Compared with vchain, the Abstract-Trie proposed in this paper also has certain advantages for querying the paper metadata. Specifically, when the number of blocks is less than 100, compared with the vchain, the improved Fabric with the Abstract-Trie in this paper can save about 100ms of query time,

and the benefits will be gradually obvious with the increment of the number of blocks, because the time complexity of the vchain is logarithmic, while the Abstract-Trie is constant.

3) Comparison of the paper citation records query time

In order to evaluate the performance improvement of the specific paper citation records query in the improved Fabric in this paper, we conducted three experiments on the grouped data sets mentioned above. Specifically, we execute the citation metadatas query in the traditional Fabric, vchain and improved Fabric. Fo each query request, it contains 10 random hash values of the paper metadata whose citation records will be queried. Finally, the average query time of a single query request in different blocks is recorded in each experiment, and the comparison of experimental results is shown in Figure 6.

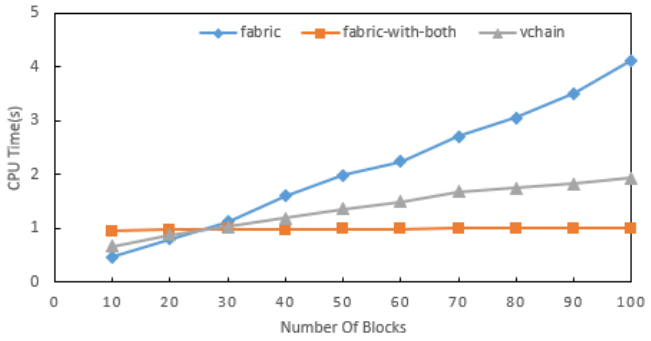

Fig. 6. Data Operation Record Query Performance

As shown in Figure 6, compared with the traditional Fabric, the query efficiency of historical citation records of the paper has been significantly improved. Specifically, the query performance of the improved Fabric is stable, which is due to the fact that the query procession of the papercitation records can be divided into two steps: obtaining the data node of the specific paper by traversing the Abstract-Trie and obtaining citation records by traversing the Operation-Record List whose head is extracted from the above data node, and the time loss of these two parts is almost fixed. When the number of blocks in the network is small, the performance of the traditional Fabric is slightly better than the improved Fabric, which is caused by the time-consuming of obtaining the index in the improved Fabric. However, with the increment of the number of blocks in the network, the performance of the improved Fabric is more and more superior to the traditional Fabric. In addition, compared with the vhain, the performance of the improved Fabric in querying the paper citation records has also been significantly improved, because the vhain can only determine the approximate range of the blocks where the result is located by skipList, while we have maintained the all blocks that contain the required citation records by the Opertion-Record List.

IV. Conclusion

In this paper, by developing two indexes in the block, we proposed an index-based efficient data query method for blockchain, which can maintain the association of the original data in every block at the expense of increasing a little time cost for building block. Numerical results show that, compared with the traditional Fabric network, the optimized Fabric in this paper needs to sacrifice hundreds of milliseconds for block construction to develop the indexes between different blocks, but at the same time, it can obtain several times of performance improvement when querying the paper metadata and its historical citation records on the chain. In addition, compared with the vchain, the improved Fabric network also has certain advantages in the performance of the paper metadata query, and the performance of quering the paper operation records has been improved by nearly 2 times.

References

[1] S. Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

[2] G. Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151, 1–32.

[3] Androulaki, Elli, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." Proceedings of the thirteenth EuroSys conference. 2018.

[4] Gürsoy, Gamze, Charlotte M. Brannon, and Mark Gerstein. "Using Ethereum blockchain to store and query pharmacogenomics data via smart contracts." BMC medical genomics 13 (2020): 1-11.

[5] Wang, Liang, Wenyuan Liu, and Xuewei Han. "Blockchain-based government information resource sharing." 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2017.

[6] Xu, Chenfu, et al. "Trusted and flexible electronic certificate catalog sharing system based on consortium blockchain." 2019 IEEE 5th International Conference on Computer and Communications (ICCC). IEEE, 2019.

[7] Peng, Zhe, et al. "VQL: Providing query efficiency and data authenticity in blockchain systems." 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW). IEEE, 2019.

[8] Li, Yang, et al. "EtherQL: a query layer for blockchain system." International Conference on Database Systems for Advanced Applications. Springer, Cham, 2017.

[9] Muzammal, Muhammad, Qiang Qu, and Bulat Nasrulin. "Renovating blockchain with distributed databases: An open source system." Future generation computer systems 90 (2019): 105-117.

[10] Xu, Yuqin, et al. "ECBC: A high performance educational certificate blockchain with efficient query." International Colloquium on Theoretical Aspects of Computing. Springer, Cham, 2017.

[11] Zeng, Lei, et al. "Transaction-based Static Indexing Method to Improve the Efficiency of Query on the Blockchain." 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). IEEE, 2021.

[12] Xu, Cheng, Ce Zhang, and Jianliang Xu. "vchain: Enabling verifiable boolean range queries over blockchain databases." Proceedings of the 2019 international conference on management of data. 2019.

[13] Zhang, Ce, et al. "Gem^ 2-tree: A gas-efficient structure for authenticated range queries in blockchain." 2019 IEEE 35th international conference on data engineering (ICDE). IEEE, 2019.