RESEARCH Open Access

A blockchain index structure based on subchain query



Xiaogang Xing¹, Yuling Chen^{1,4*} , Tao Li¹, Yang Xin^{2,3} and Hongwei Sun⁴

Abstract

Blockchain technology has the characteristics of decentralization and tamper resistance, which can store data safely and reduce the cost of trust effectively. However, the existing blockchain system has weak performance in data management, and only supports traversal queries with transaction hashes as keywords. The query method based on the account transaction trace chain (ATTC) improves the query efficiency of historical transactions of the account. However, the efficiency of querying accounts with longer transaction chains has not been effectively improved. Given the inefficiency and single method of the ATTC index in the query, we propose a subchain-based account transaction chain (SCATC) index structure. First, the account transaction chain is divided into subchains, and the last block of each subchain is connected by a hash pointer. The block-by-block query mode in ATTC is converted to the subchain-by-subchain query mode, which shortens the query path. Multiple transactions of the same account in the same block are merged and stored, which simplifies the construction cost of the index and saves storage resources. then, the construction algorithm and query algorithm is given for the SCATC index structure. Simulation analysis shows that the SCATC index structure significantly improves query efficiency.

Keywords: Blockchain, Query optimization, Hash pointer, Subchain

Introduction

With the advancement of computer science, the development of technologies such as big data [1], blockchain [2, 3], and the Internet of Things [4–6] has been promoted, and many convenient services [7] have also been brought to users. However, there are still many problems, such as user data privacy leakage [8–10], low algorithm efficiency [11], search efficiency [12], and other issues. Since traditional centralized institutions are not completely credible, users' data may be leaked. Blockchain with decentralized characteristics can store data safely and protect users' data privacy [13].

In 2008, Bitcoin was proposed by Satoshi Nakamoto in "Bitcoin: A Peer-to-Peer Electronic Cash System" [14], marking the emergence of blockchain technology. As the

Full list of author information is available at the end of the article

underlying technology of Bitcoin, blockchain has received extensive attention [15–18]. Blockchain is a distributed database technology that has the characteristics of decentralization [19–21], traceability, tamper-proof, collective maintenance, etc. [22]. The emergence of this technology solves a series of problems such as high cost, low efficiency, and low trust brought by centralized institutions [23]. However, the blockchain is a chain structure, which will cause the query efficiency to decrease as the number of blocks grows. Take the Bitcoin blockchain as an example. As of June 7, 2021, the block height has reached 674,000, which means that when querying historical data, hundreds of thousands of blocks may be traversed. Such a query method cannot meet the current query requirements.

Level-DB is the mainstream database in the blockchain system, which is based on the storage structure of the LSM tree. This leads to the lower reading performance of the blockchain [24]. Besides, Level-DB only supports simple Key-Value queries, not relational queries [25, 26].



^{*}Correspondence: ylchen3@gzu.edu.cn

¹ State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

⁴Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Weifang 262700, China

When querying transactions, users can only traverse in block order, which further reduces query efficiency [27]. The blockchain system only supports related queries with transaction hashes as keywords and does not query with account hashes as keywords. The query method is single. In response to this problem, some current solutions are to transfer the data on the chain to the off-chain storage [28, 29] to improve query efficiency, but the offchain storage violates the decentralized characteristics of the blockchain. Third-party databases are faced with trust issues, and they may also be attacked with a single point of failure, data loss, data tampering, and other issues. There are huge security holes in off-chain storage [25]. Therefore, under the premise of ensuring security, improving the retrieval efficiency on the chain is a current research hotspot.

You et al. [30].designed a hybrid index mechanism that supports blockchain transaction traceability based on the Ethereum state tree. In this mechanism, a hash pointer is embedded in the account transaction, which points to the block where the previous transaction. Through the pointer, the Account Transaction Trace Chain (ATTC) can be quickly traced. The query method based on ATTC improves the query efficiency of account transactions, but for some active accounts with longer transaction chain length, a longer chain still needs to be traversed. Besides, users do not always want to find all the historical transactions of an account, and it is still difficult to find target transactions in massive account data. In this regard, we improve the query scheme based on ATTC and propose a SCATC index structure, which solves the shortcomings of the ATTC index structure in the query effectively. The main contributions of this paper are as follows:

- 1. We divide the transaction chain into subchains and connect different subchains with hash pointers to shorten the query path when querying early historical transactions. This solution is not a query mode that uses space for time. While reducing the time complexity, the space complexity does not increase significantly.
- 2. We design a constructing algorithm and query algorithm for the SCATC index structure. The simulation results show that the SCATC-based query is more efficient when querying the early transactions of accounts.
- 3. Multiple transactions of an account in the same block are merged into one, and at most one index is built within each block for the same account. This reduces the cost of index construction and storage overhead.

The paper is organized as follows. "Related works" section of this article introduces the related work of blockchain in the data query; "Preliminaries" section introduces some preliminary knowledge of blockchain; "SCATC index structure" section elaborates on the construction method and query algorithm of SCATC index structure; "Experiment and analysis" section is efficiency

analysis and simulation experiment. The full text is summarized in "Conclusions" section.

Related works

In order to improve the efficiency of blockchain query, many researchers have made relevant studies. Xu et al. [31]. proposed an Educational Certificate Blockchain (ECBC) in response to the issue of education certificate management. ECBC has built a tree structure (MPT-Chain), which not merely supports effective query of transactions, but also supports historical transaction query of accounts. The index structure improves the efficiency of querying account transactions.

Morishima et al. [32]. propose to accelerate blockchain search through GPU using the higher computing power of GPU. Utilizing the feature that blockchain data does not need to be updated or deleted, an array-based Patricia tree structure is introduced, which is suitable for GPU processing. To study the identity verification and range query issues in the hybrid storage blockchain, Zhang et al. [33].used a unique gas cost model to design an authentication data structure GEM2-tree that can be effectively maintained by the blockchain. It not only saves gas consumption in smart contracts but also effectively supports identity verification queries. Aiming at the inefficient query of the Elastic-Chain [34] model on the blockchain, Jia et al. [35].propose an ElasticQM (elastic query model) query method based on the model. In the user layer, the model catches the user's first query result to improve the efficiency of the second query. In the data layer, the B-tree is combined with the Merkle tree to construct the blockchain data storage structure of the B-M tree. This storage structure improves the query efficiency of the internal data of the block. Jiao et al. [36]. propose a blockchain database system framework, which realizes the application of data management on the blockchain. Combining red-black trees with Merkle trees, they propose a tamper-resistance index based on hash pointers. Through the index can realize the fast positioning of the data in the block. Zheng et al. [37] divides the data attributes on the blockchain into discrete attributes and continuous attributes and proposed a MHerkle tree index structure for different attributes, which supports range query. H et al. [38].proposed the Ethereum data index structure EBTree based on the B+ tree index. The EBTree index supports real-time top-k query and range query. In addition, EBTree only stores the identifier of the blockchain data, which occupies a relatively small storage space and has better search and insertion performance. Ren et al. [39] introduce a DCOMB (Dual Combination Bloom filter) scheme, which converts the computing power used for Bitcoin mining into the computing power for data query. DCOMB has higher random read performance and lower error rate than COMB (Combination Bloom filter).

The encrypted signature tree data structure of the Merkel Block Space Index (BSI) [27] modifies the Merkle KD-tree to support fast Spatio-temporal query processing. In Ethereum, when a user initiates a transaction, the system checks the status of the account. Wan et al. [40]. built a Merkle Patricia tree account storage structure GMPT (Group Merkel Patricia Tree) to speed up the query of account status. However, GMPT does not support fast queries of historical transactions. For this, an index directory BKV (B-Key-Value) is constructed in combination with the B-tree index [41].

Preliminaries

Blockchain is a chain structure, as shown in Fig. 1. The internal structure of the block is divided into two parts: the block header and the block body. The block header records some information such as the timestamp, the hash value of the previous block, and the Merkle Root. A Merkle tree is recorded in the block body, the user's transaction is hashed to obtain the leaf node hash value. Combine the hash values of the two leaf nodes and perform a hash operation to obtain a new hash value, which is used as the hash value of the parent node. Through continuous iteration and hash operation, the hash value of Merkle Root can be finally obtained, which can be used to verify the transactions in the block.

Unlike traditionally linked lists, the pointers used in the blockchain are hash pointers, which store hash values instead of addresses in memory. The blocks are connected into a chain by hash pointers, and the pointers point from the new block to the old block in chronological order.

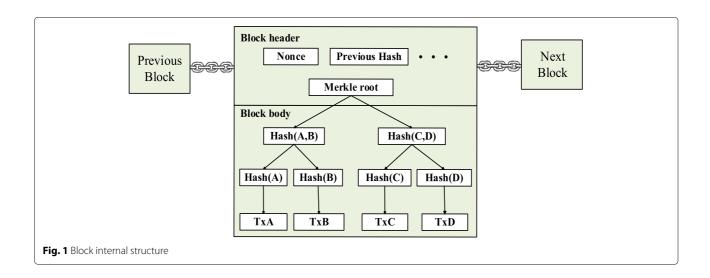
When querying transactions, users can only traverse from the new block to the old block through the hash pointer. The data in the block body is queried through the Merkle tree. First, check the Merkle Root, and then traverse the Merkle tree from top to bottom through the hash pointer in the Merkle Root. The hash pointer of the leaf node can locate the transaction storage location. If the target transaction is not found in the current block, the next block will be inquired until the target transaction is found. When querying early historical transactions, it is necessary to traverse a longer blockchain. If the transaction does not exist in the chain, the query will proceed to traverse the complete blockchain. This block-by-block traversal query method is extremely inefficient.

SCATC index structure

In this section, we have optimized the ATTC scheme and designed the SCATC index scheme on the basis of this scheme. The details of the SCATC index structure are introduced in detail. In addition, we also designed a construction algorithm and query algorithm for the SCATC index structure.

Index design

Given ATTC's shortcomings in retrieval, we improve it based on the index structure. In ATTC, the transactions of accounts in different blocks are connected by hash pointers. The hash pointers here are called FHP (First Hash Pointer). In the SCATC index structure, transaction chain is divided into subchains. Every k(k > 1) block is divided into a subchain, and each subchain has a subchain number. Each transaction of the account will identify the location of the transaction when it enters the chain. For example, $Account_{n,k}(Account$ is the account name, n and k are both positive integers)means that the account is in the kth block in the nth subchain of the transaction chain. Every time a user participates in a transaction of k blocks, another hash pointer is added to the account branch leaf node in the block $Account_{n,k}$ pointing to the block



 $Account_{n-1,k}$. The hash pointer connecting the blocks at the last block of the two subchains is SHP (Second Hash Pointer). The index structure of SCATC is shown in Fig. 2.

Figure 2 shows the chain structure of the blockchain. In the blockchain, each block connected by the FHP constitutes the transaction chain of the account, and each SHP will span a complete subchain. The FHP in SCATC is not embedded in the transaction but embedded in the leaf nodes of the Merkle tree. When querying early historical transaction, the system will directly filter the user's recent transaction data. For accounts with low activity, the latest transaction may exist in the earlier part of the chain. In the SCATC scheme, the state tree not only maintains the account balance status but also maintains the subchain number of the latest transaction. Through the status tree, users can quickly locate the block location where the latest transaction. The same account may generate multiple transactions in a short time, and transactions with higher transaction fees usually enter the chain first, so the same account may have multiple transactions in the same block. To simplify the construction of the index, we merge multiple transactions of the account in the same block for storage, and the account branch leaf nodes can directly access all the transaction of the target account in the block. The storage diagram is shown in Fig. 3.

Taking Account_A as an example, regardless of whether the transaction of Account_A is included in the latest block, the leaf node of the account branch of the latest block will maintain a hash pointer pointing to the latest transaction of the account in the block. While maintaining the global state, the state tree will also record the specific transaction records of each account whose state has changed in the block. All transaction records of the

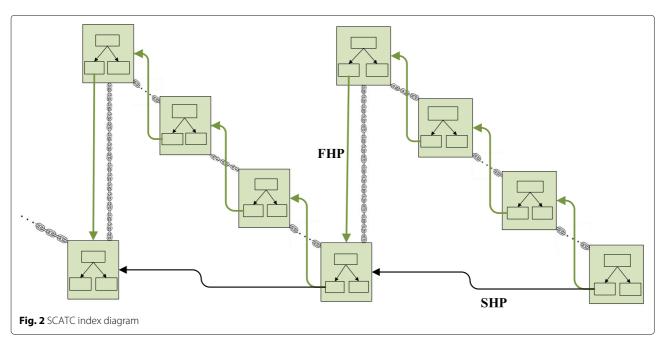
same account in the same block are combined and stored together, such as the transactions Tx_A1 and Tx_A2 of Account_A in block N, the transactions Tx_A3, Tx_A4 and Tx_A5 of block K. Specific account transactions can be accessed through the state tree, without the need to build a separate transaction tree, which reduces the cost of index construction.

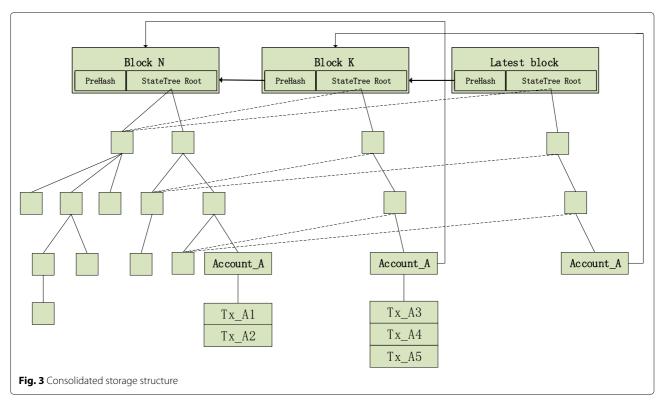
Algorithm design

We first designed the index construction algorithm, and then designed the query algorithm according to the SCATC index structure.

Index construction algorithm

The algorithm traverses all the accounts whose status has changed and judge whether the accounts are new users one by one. If it is a new user, assign the value of one to the subchain number of the transaction chain and the block number in the subchain. If not, judge whether the block number of the subchain of the previous block in the account transaction chain is less than k-1. If less than k-1, the subchain number of the new transaction is the same as the previous block, and the block number in the subchain is increased by one. If the block number in the subchain where the previous block is located is equal to k-1, the subchain number of the new transaction is the same as the previous block, and the block number in the subchain is assigned the value k. Then, add SHP to the account branch node corresponding to the new transaction, pointing to the kth block of the previous subchain. Through SHP, users can directly access the data of the previous subchain. If the block number of the previous block in the subchain is equal to k, the subchain number in





the new transaction will increase by 1, and the block number is assigned a value of 1. The block with block number 1 is the first block of the new subchain.

SCATC Index Construction Algorithm

```
input: The account transaction whose status has
      changed
output: SCATC index
1: for \ i \ in \ ACCOUNTS:
2:
       if i is new Account:
3:
            SubchainNum = 1
4:
            SubchainBlockNum = 1
5:
       if i is not new Account:
            if PreSubchainBlockNum < k - 1:
6:
7:
                 SubchainNum = PreSubchainNum
8:
                 SubchainBlockNum
                      = PreSubchainBlockNum + 1
9:
           if PreSubchainBlockNum == k - 1:
                 SubchainNum = PreSubchainNum
10:
                 SubchainBlockNum = k
11:
12:
                 SHP = PreHashvalue
13:
            if PreSubchainBlockNum == k:
14:
                 SubchainNum
                           = PreSubchainNum + 1
15:
                 SubchainBlockNum = 1
16 : end for
```

Query algorithm

When inquiring about historical transactions, users can directly access the kth block of the previous subchain from

the *k*th block of the latest subchain according to the SHP until the target subchain. Then traverse the blocks in the target subchain to obtain the transaction.

```
SCATC Index Query Algorithm
input : Target account subchain
output: Account transaction
1: TargetAccountData = []
2: p = LatestBlock.data
3: if p.SubchainBlockNum < k:
      for i in range(LatestBlock,
       LatestSubchainFirstBlock, -1):
5:
           q = i.data
6:
           for j in q:
7:
                ifj.Account == TargetAccount :
                     TargetAccountData.append(j)
9: for i in range(LatestSubchainKBlock,
   TargetSubchainKBlock, -k):
10:
        q = i.data
11:
       for j in q:
12:
           if j.Account == TargetAccount :
                TargetAccountData.append(j)
13:
14: for i in range(TargetSubchainKBlock,
   TargetSubchainFisrtBlock, -1):
        q = i.data
15:
16:
       for j in q:
17:
           if j.Account == TargetAccount :
                TargetAccountData.append(j)
19: return TargetAccountData
```

The algorithm first creates a list TargetAccountData to save the data of the target accounts that have been accessed. Lines 2-8 of the algorithm visit the latest block in the account transaction chain. If the sequence number of the block is less than *k*, traverse from the latest block to the first block in the subchain. Lines 9-13 of the algorithm, according to the hash pointer in the kth block, access the kth block of the previous subchain until the kth block of the target subchain. During this process, only one block is visited in each subchain. Lines 14-18 of the algorithm traverse all the blocks in the target subchain. Before the query reaches the target subchain, only one block is visited in all subchains except the latest subchain. The blockby-block traversal query method is transformed into a subchain-by-subchain query, which shortens the access path in the search process.

Experiment and analysis

Efficiency analysis

The length of the subchain affects the scope and efficiency of the query. Assuming that the transaction chain length of the current target account is s, the number of blocks in each subchain is $k(k > 1, k \in \mathbb{Z})$, and the number of subchains is $n(n > 1, n \in \mathbb{Z})$. When the transaction chain length s is determined, the number of subchains n and k are inversely proportional.

$$n = \frac{s}{k} \tag{1}$$

When k increases, the number of block accesses in the subchain will increase, and the query range will increase. The number n of subchains will continue to decrease as k increases, thereby reducing the frequency of SHP construction, because each subchain only constructs an SHP once for the transaction chain of the account. If k decreases, the length of the subchain becomes shorter, and the query range of a single subchain is reduced. If the user wants to increase the query range, the range from the initial subchain to the end subchain of the query needs to be given. In addition, the number of subchains will continue to increase with the k decrease, and the frequency of SHP construction will increase.

We define access to the block where the target transanction is located as valid queries, and queries other than valid queries as invalid queries. Invalid queries are represented by the symbol ψ . If ψ is larger, the query efficiency is lower, and it also means that more computing resources are wasted. In the SCATC-based query method, the number of blocks to be accessed by the initial subchain for querying is μ

$$\mu = \frac{s}{k} + k - 1 \tag{2}$$

The number of blocks of irrelevant subchains accessed is ψ_1 , then

$$\psi_1 = n - 1 \tag{3}$$

Because when the query proceeds to the target subchain, other subchains only access the last block, which reduces the number of irrelevant blocks that need to be visited when locating the target subchain. The transaction chain query method requires access to the complete transaction chain when querying the data of the initial subchain. The number of irrelevant blocks accessed is ψ_2

$$\psi_2 = n(k-1) \tag{4}$$

As s keeps increasing, n presents a monotonous increasing trend. Eqs. (3) and (4) can be regarded as a linear function. In Eq. (3), the coefficient of the independent variable n is 1, and in Eqs. (4), the independent variable coefficient is k-1(k>1). With the n continuous growth, $\psi_2 >> \psi_1$. Invalid queries based on the transaction chain have a faster growth rate, while invalid queries based on the SCATC query have a slower growth. The larger the n, the more obvious the advantages of the SCATC-based query.

Simulation experiment

The simulation environment is a host computer, where the CPU is Intel(R) Core(TM) i7-5500U, 12GB memory, and the 64-bit operating system Windows10 Professional Edition. The SCATC index structure is written and implemented in python language. The blockchain requires each full node to maintain a complete ledger, so the data retrieval of the simulation is performed locally.

The simulation compares the query efficiency of ATTC, MPT-Chain and SCATC query methods under different transaction chain lengths. Set the subchain length k to 10. The length of the transaction chain is set to 1000-6000 blocks, and the corresponding number of subchains is 100-600. The simulation experiments are divided into six groups according to different transaction chain lengths, and each group of simulations is repeated eight times. To better highlight the effect of simulation comparison, each query is tested with the initial Subchain. Three query methods start from the latest block to the earliest block, so the query time in the method in SCATC includes the time to locate the subchain. The simulation experimental data obtained are shown in Tables 1, 2 and 3.

The average value of simulation experimental data of ATTC, MPT-Chain and SCATC is plotted as a line chart shown in Fig. 4. As the length of the transaction chain continues to grow, the query time based on the ATTC and MPT-Chain query method is constantly increasing. However, the query method based on SCATC has not changed significantly in query efficiency.

Table 1 ATTC

Number of blocks	Qu	ery	time	e/m	s	AVG	Mean deviation			
1000	21	25	13	13	19	20	12	23	18.3	33.4
2000	32	42	30	35	61	56	48	43	43.4	69.8
3000	52	55	47	42	56	39	50	49	48.8	36.4
4000	46	54	73	88	56	69	75	61	65.3	88.0
5000	85	99	88	65	73	79	92	87	83.5	67.0
6000	74	109	85	77	83	95	88	101	89.0	76.0

The subchain length k is 10, and AVG is the average query time

For active users in the blockchain system, the length of the transaction chain has increased at a faster rate. From a theoretical analysis, no matter which of the above query methods, as the transaction chain grows, the length of the transaction chain that needs to be traversed will be longer, and the query efficiency will show a downward trend. However, after the SCATC index structure divides the transaction chain into subchains, it greatly reduces the number of block visits. The limited length of the transaction chain cannot cause a significant change in SCATC's query efficiency.

Take K to 50 and 100 to conduct a simulation experiment again, and compare the query efficiency of the three query methods when k takes different values. The simulation results obtained are shown in Figs. 5 and 6.

Compared with Figs. 4, 5 and 6 show no significant change in query efficiency. The query method based on ATTC and MPT-Chain has no obvious change in query efficiency. The main reason is that no matter how the value of k changes, the method needs to traverse a complete transaction chain. The SCATC-based query method has no obvious change in query efficiency. The reason is that after the transaction chain is divided into subchains, the number of blocks that need to be accessed is significantly reduced. The length of the transaction chain that needs to be accessed is not long enough to cause a significant drop in query efficiency.

Table 2 MPT-Chain

Number of blocks	Que	Mean deviation								
1000	15	21	18	12	16	19	22	18	17.6	19.8
2000	38	34	31	39	29	45	41	39	37.0	34.0
3000	53	58	45	56	49	44	39	51	49.4	41.0
4000	63	56	62	53	51	64	66	65	60.0	40.0
5000	78	73	85	81	76	75	77	75	77.5	23.0
6000	82	89	87	87	92	83	93	86	87.4	23.8

The subchain length k is 10, and AVG is the average query time

Table 3 SCATC

Number of blocks	Qu	ery	time	e/m	s	AVG	Mean deviation			
1000	15	12	16	15	15	17	13	13	14.5	11.0
2000	17	14	15	14	12	15	12	16	14.4	11.0
3000	16	18	14	13	15	17	16	13	15.3	12.0
4000	15	13	15	16	14	13	15	16	14.6	7.8
5000	17	17	17	13	16	16	19	18	16.6	9.8
6000	10	12	16	12	18	21	17	19	17.4	11.8

The subchain length k is 10, and AVG is the average query time

Time complexity

For an algorithm, its efficiency is related to the language it implements and the hardware configuration of the computer. Putting aside these factors related to software and hardware, it can be considered that the efficiency of the algorithm is only related to the scale of the problem.

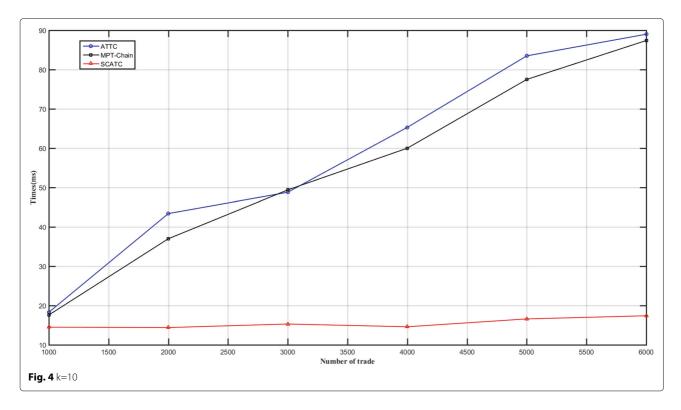
In the traditional traversal query method, it is assumed that the block height is h_1 , and the number of nodes of the Merkle tree in the block is p_1 . As the transactions of users in the system continue to increase, h_1 will continue to increase, while p_1 will be relatively unchanged, so h_1 is the scale of the problem. Since there may be multiple transactions in the same block in the same account, the traversal guery method needs to traverse a complete tree. In addition, the system does not know whether the next block also contains the target account transaction, so the system will continue to traverse the next block until the entire blockchain. So in the process of traversal query, the number of query operations that the system will execute is $\lambda(h_1) = p_1 \times h_1$. Since the block size usually varies little, the number of nodes p_1 can be regarded as a constant. Then the time complexity of the algorithm can be expressed as

$$T(h_1) = O(h) (5)$$

In ATTC, assuming that the block height of ATTC is h_2 , as the transactions of the account continue to increase, the length of the transaction chain continues to grow, so h_2 is the scale of the problem. The Merkle Patricia tree in the block is generated by the account ID, so the length of the query path is fixed. Suppose the number of fields in the transaction is g, and the number of query operations that the system will execute is also $\lambda(h_2) = p_2 \times h_2 \times g$. The number of fields in each transaction fluctuates slightly, so g can be regarded as a constant. Then the time complexity of the algorithm can be expressed as

$$T(h_2) = O(h) \tag{6}$$

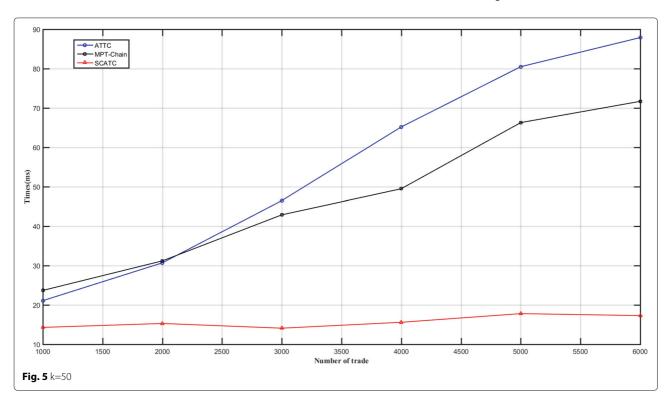
In MPT-Chain, suppose the height of the MPT-Chain block is h_3 , and the length of the query path p_3 in the Merkle Patricia tree remains unchanged. Compared with

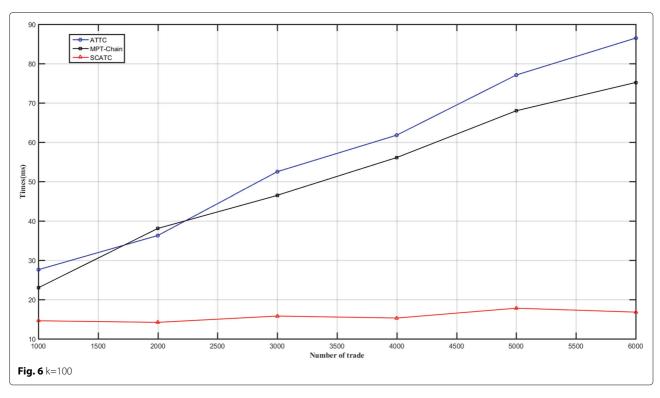


the ATTC scheme, MPT-Chain pointers are not embedded in specific transactions, so there is no need to access specific transactions. The number of query operations performed by the system is $\lambda(h_3)=p_3\times h_3$, and the time complexity of the algorithm can be expressed as

$$T(h_3) = O(h) \tag{7}$$

In SCATC, the index is also constructed based on the Merkle Patricia tree. Assuming that the height of the transaction chain of the target transaction to be checked





is h_4 , the length of the query path within the block is p_4 , and the length of the subchain is k. Then the number of query operations that the system will perform is $\lambda(h_4) = \frac{1}{k} \times h_4 \times p_4$. Then the time complexity of the algorithm can be expressed as

$$T(h_4) = O(h) \tag{8}$$

From the above analysis, it can be seen that the time complexity of any query method is linear order O(h), and it cannot reach the ideal constant order O(1). But in the case of linear order time complexity, the most important factor affecting query efficiency is the block height h. In the above-mentioned several query methods, there is $\frac{p_4}{k} < p_3 < p_2 g < p_1$, so $\lambda(h_4) < \lambda(h_3) < \lambda(h_2) < \lambda(h_1)$. Therefore, in the above scheme, SCATC needs to perform the least number of query operations during the query process, and the query efficiency is higher.

Conclusions

We improve the query efficiency of the ATTC index structure and proposes a SCATC index structure that supports querying account subchain data. We divide the transaction chain into subchains, add hash pointers to the account branch nodes of the block at the last block of each subchain, and each subchain is connected by hash pointers. Through this pointer, the query mode of traversing the transaction chain is converted to the subchain query mode, which effectively reduces the access to irrelevant block data and reduces the computational overhead. All transactions of the same account in the same block

are merged and stored together, which simplifies the construction cost of the index and reduces the storage overhead. Besides, we also design a query algorithm for the SCATC index. Simulation experiments and analysis show that the index structure based on SCATC can improve the query efficiency of account transactions effectively. However, the improvement in query efficiency of this solution is only for accounts with a longer account transaction chain, and there is no significant improvement for accounts with a shorter account transaction chain. At the same time, this solution is only for retrieval optimization in the plaintext state, and the data privacy of blockchain users cannot be guaranteed. Our next step will be dedicated to the optimization of ciphertext data retrieval in the blockchain.

Acknowledgements

We sincerely thank the School of Computer Science and Technology of Guizhou University and the State Key Laboratory of Public Big Data for the learning environment. At the same time, we thank Yufeng Li, Chaoyue Tan and Juan Ma for their help in this paper.

Authors' contributions

XX put forward the main ideas and drafted the manuscript. YC and TL conducted a feasibility analysis. XX and TL design and implement the index structure. YX and YC guided the research and participated in the discussion of the manuscript. HS made suggestions for the article. All authors read and approve the final manuscript.

Funding

This study is supported by Foundation of National Natural Science Foundation of China (Grant Number: 61962009); Major Scientific and Technological Special Project of Guizhou Province(20183001); Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2018BDKFJJ005); Talent project of Guizhou

Big Data Academy Guizhou Provincial Key Laboratory of Public Big Data.([2018]01).

Availability of data and materials

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

¹ State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China. ² School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China. ³ National Engineering Laboratory for Disaster Backup Recovery, Beijing 100876, China. ⁴ Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Weifang 262700, China.

Received: 17 June 2021 Accepted: 13 September 2021 Published online: 02 October 2021

References

- Mahmud MS, Huang JZ, Salloum S, Emara TZ, Sadatdiynov K (2020) A survey of data partitioning and sampling methods to support big data analysis. Big Data Min Analytics 3(2):85–101
- Xu X, Chen Y, Zhang X, Liu Q, Liu X, Qi L (2019) A blockchain-based computation offloading method for edge computing in 5G networks. Softw: Pract Exp 51:2015–2032. https://doi.org/10.1002/spe.2749
- Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W (2019) Become: blockchain-enabled computation offloading for iot in mobile edge computing. IEEE Trans Ind Inform 16(6):4187–4195
- Azrour M, Mabrouki J, Guezzaz A, Farhaoui Y (2021) New enhanced authentication protocol for internet of things. Big Data Min Analytics 4(1):1–9
- Wang Y, Yang G, Li T, Li F, Tian Y, Yu X (2020) Belief and fairness: A secure two-party protocol toward the view of entropy for iot devices. J Netw Comput Appl 161:102641
- Li F, Wang D, Wang* Y, Yu X, Wu N, Yu J, Zhou H (2020) Wireless communications and mobile computing blockchain-based trust management in distributed internet of things. Wirel Commun Mob Comput 2020. https://doi.org/10.1155/2020/8864533
- Jin Y, Guo W, Zhang Y (2019) A time-aware dynamic service quality prediction approach for services. Tsinghua Sci Technol 25(2):227–238
- Khazbak Y, Fan J, Zhu S, Cao G (2020) Preserving personalized location privacy in ride-hailing service. Tsinghua Sci Technol 25(6):743–757
- Chen Y, Sun J, Yang Y, Li T, Niu X, Zhou H PSSPR: A Source Location Privacy Protection Scheme Based on Sector Phantom Routing in WSNs. Int J Intell Syst. https://doi.org/10.1002/int.22666
- Qi L, Hu C, Zhang X, Khosravi MR, Sharma S, Pang S, Wang T (2020) Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. IEEE Trans Ind Inform 17(6):4159–4167. https://doi.org/10.1109/TII.2020.3012157
- Chen N, Wang Z, He R, Jiang J, Cheng F, Han C (2021) Efficient scheduling mapping algorithm for row parallel coarse-grained reconfigurable architecture. Tsinghua Sci Technol 26(5):724–735
- Wang Q, Liu X, Liu W, Liu A-A, Liu W, Mei T (2020) Metasearch: incremental product search via deep meta-learning. IEEE Trans Image Process 29:7549–7564
- 13. Xu X, Liu Q, Zhang X, Zhang J, Qi L, Dou W (2019) A blockchain-powered crowdsourcing method with privacy preservation in mobile environment. IEEE Trans Comput Soc Syst 6(6):1407–1419
- Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system[J]. Decentralized Bus Rev:21260
- 15. Li T, Wang Z, Yang G, Cui Y, Chen Y, Yu X (2021) Semi-selfish mining based on hidden Markov decision process. Int J Intell Syst 36:3596–3612

- Li T, Wang Z, Chen Y, Li C, Jia Y, Yang Y (2021) Is semi-selfish mining available without being detected? Int J Intell Syst:1-22. https://doi.org/10. 1002/int.22656
- Li T, Chen Y, Wang Y, Wang Y, Zhao M, Zhu H, Tian Y, Yu X, Yang Y (2020) Rational protocols and attacks in blockchain system. Secur Commun Netw 2020. https://doi.org/10.1155/2020/8839047
- 18. Wang Y, Yang G, Bracciali A, Leung HF, Yu X (2020) Incentive compatible and anti-compounding of wealth in proof-of-stake. Inf Sci 530:85–94
- Wang Y, Wang Y, Zhaojie W, Yang G, Yu X (2020) Research cooperations of blockchain: Toward the view of complexity network. J Ambient Intell Humanized Comput:1–14. https://doi.org/10.1007/s12652-020-02596-6
- Li P, Li K, Wang Y, Zheng Y, Wang D, Yang G, Yu X (2020) A systematic mapping study for blockchain based on complex network. Concurr Comput: Pract Exp:5712. https://doi.org/10.1002/cpe.5712
- Xu X, Chen Y, Yuan Y, Huang T, Zhang X, Qi L (2020) Blockchain-based cloudlet management for multimedia workflow in mobile cloud computing. Multimed Tools Appl 79(15):9819–9844
- 22. He P, Yu G, Zhang Y, Bao Y (2016) Survey on blockchain technology and its application prospects. Comput Sci 44(4):1–7
- 23. Yuan y., Wang F (2016) Blockchain technology development status and prospects. Acta Autom Sin 42(4):481–494
- Wang H, Dai B, Li c., Shaohua Z (2019) A query optimization model for blockchain applications. Comput Eng Appl 55(22):34–39
- 25. Wang Q, He P, Nie T, Derong S, Yu G (2018) Overview of blockchain system data storage and query technology. Comput Sci 45(12):12–18
- Yu B, Li X, Zhao H (2019) A structured data management method based on blockchain storage expansion. J Beijing Inst Technol 39(11):1160–1166
- Qu Q, Nurgaliev I, Muzammal M, Jensen CS, Fan J (2019) On spatio-temporal blockchain query processing. Futur Gener Comput Syst 98:208–218
- 28. Li Y, Zheng K, Yan Y, Liu Q, Zhou X (2017) Etherql: a query layer for blockchain system. In: International Conference on Database Systems for Advanced Applications. Springer, Suzhou. pp 556–567
- Yang X, Wang M, Xu D, Luo N, Sun C (2019) Data storage and query method of agricultural products traceability information based on blockchain. Trans Chin Soc Agric Eng 35(22):323–330
- You Y, Kong L, Xiao Z, Zheng Y, Li Q (2019) Hybrid indexing scheme supporting blockchain transaction tracing. Comput Intergrated Manuf Syst 25(04):978–984
- 31. Xu Y, Zhao S, Kong L, Zheng Y, Zhang S, Li Q (2017) Ecbc: A high performance educational certificate blockchain with efficient query. In: International Colloquium on Theoretical Aspects of Computing. Springer, Hanoi. pp 288–304
- Morishima S, Matsutani H (2018) Accelerating blockchain search of full nodes using gpus. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). IEEE, Cambridge. pp 244–248
- Zhang C, Xu C, Xu J, Tang Y, Choi B (2019) Gem[^] 2-tree: A gas-efficient structure for authenticated range queries in blockchain. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, Macao. pp 842–853
- 34. Jia D, Xin J, Wang Z, Guo W, Wang G (2018) Elasticchain: support very large blockchain by reducing data redundancy. In: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data. Springer, Macau. pp 440–454
- Jia D, Xin J, Wang Z, Guo W, Wang G (2019) Elasticqm: A query model for storage capacity scalable blockchain system. J Softw 30(09):2655–2670. http://www.jos.org.cn/1000-9825/5774.htm
- 36. Jiao T, Shen D, Tiezheng N (2019) Blockchain database: a queryable and tamper-proof database. J Softw 9:2671–2685
- Zheng H, Shen D, Tiezheng N (2020) Queriability optimization of blockchain system for hybrid indexing. Comput Sci 47(10):301–308
- XiaoJu H, XueQing G, ZhiGang H, LiMei Z, Kun G (2020) Ebtree: A b-plus tree based index for ethereum blockchain data. In: Proceedings of the 2020 Asia Service Sciences and Software Engineering Conference. pp 83–90
- Ren Y, Zhu F, Sharma PK, Wang T, Wang J, Alfarraj O, Tolba A (2020) Data query mechanism based on hash computing power of blockchain in internet of things. Sensors 20(1):207
- 40. Wan L (2020) A query optimization method of blockchain electronic transaction based on group account. In: International Conference on Big

- Data Analytics for Cyber-Physical-Systems. Springer, Shanghai. pp 1358–1364
- Wan L (2020) An optimization method for blockchain electronic transaction queries based on indexing technology. In: International Conference on Big Data Analytics for Cyber-Physical-Systems. Springer, Shanghai. pp 1273–1281

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ▶ springeropen.com