A SCATC-Based Blockchain Index Structure

Xiaogang Xing[1], Yuling Chen[1,4](✉), Tao Li[1], Yang Xin[2,3], and Hongwei Sun[4]

[1] State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
Ylchen3@gzu.edu.cn
[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

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; then, the 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

1 Introduction

In 2008, Bitcoin was proposed by Satoshi Nakamoto in "Bitcoin: A Peer-to-Peer Electronic Cash System" [1], marking the emergence of blockchain technology. Blockchain is a distributed database technology that has the characteristics of decentralization, traceability, tamper-proof, collective maintenance, etc. [2]. The emergence of this technology solves a series of problems such as high cost, low efficiency, and low trust brought by centralized institutions [3]. 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 [4]. Besides, Level-DB only supports simple Key-Value queries, not relational queries [5]. When querying transactions, users can

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 133–141, 2021.
https://doi.org/10.1007/978-3-030-89814-4_10

only traverse in block order, which further reduces query efficiency [6]. 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.

To quickly query account historical transactions, an index structure that supports querying account transaction chains was proposed in the Education Certificate Blockchain (ECBC) [7]. This index structure has the characteristics of low latency and high throughput. You et al. [8] 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 subchain-based account transaction chain (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 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.

The paper is organized as follows. Section 2 of this article introduces the related work of blockchain in the data query. Section 3 introduces the index structure based on SCATC and the query algorithm given in detail. Section 4 is efficiency analysis and simulation experiment. The full text is summarized in Sect. 5.

2 Related Works

To improve the efficiency of blockchain in data retrieval, Morishima et al. [9] 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. [10] 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 ElasticChain [11] model on the blockchain, Jia et al. [12] 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. [13] 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. [14] divide 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. Ren et al. [15] 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) [6] 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. [16] 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 [17].

3 SCATC Index Structure

3.1 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 the first hash pointer (FHP).

In the SCATC index structure, account transaction chain is divided into subchains. Every $k(k > 1)$ block is divided into a subchain, and each subchain has a subchain number. The index structure of SCATC is shown in Fig. 1.

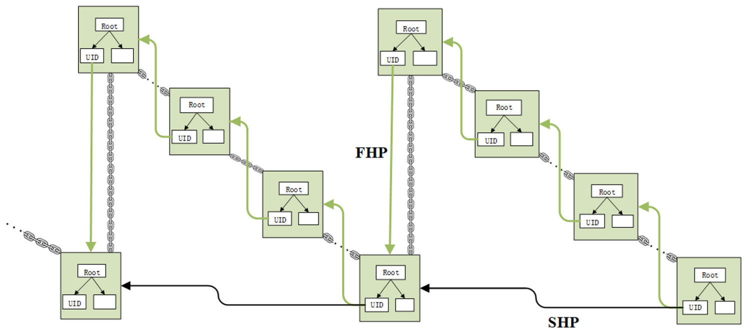

Fig. 1. SCATC index structure

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, $n \geq 1$ and $k \geq 1$, n and k are both positive integers) means that the account is in the k th block in the nth subchain of the transaction chain. In ATTC, 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 second hash pointer (SHP).

Within each block body, each leaf node of the Merkle tree represents an account, and both FHP and SHP are stored in the pointer variable defined in the leaf node. For new blocks on the chain, the system will detect each account whose status has changed, and update the transaction chain and subchain for each account. The specific steps for dividing subchains are as follows:

Step 1: Firstly, determine whether the account is a new user one by one. If so, set the transaction subchain number of the account transaction chain and the block serial number in the subchain to 1. If not, go to *step 2*.

Step 2: Determine whether the block sequence number of the sub-chain where the previous block in the account transaction chain is located is less than $k - 1$. If so, the subchain number of the new transaction is the same as the previous block, and the sequence number of the block in the sub-chain is increased by 1.

Step 3: Determine whether the block sequence number in the subchain where the previous block in the transaction chain is equal to $k - 1$, if so, the subchain number of the new transaction is the same as the previous block, and the block sequence number in the subchain is k. At the same time, the secondary hash pointer SHP is added to the account branch node of the new block to point to the k th block of the previous subchain.

Step 4: Determine whether the block sequence number in the subchain where the previous block is located is equal to k. If so, the subchain number in the new transaction is increased by 1, and the block sequence number is 1. This block is the initial block of the new subchain in the transaction chain.

3.2 Algorithm Design

When inquiring about historical transactions, users can directly access the k th 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. Before the query reaches the target subchain, only one block is visited in all subchains except the latest subchain. The block-by-block traversal query method is transformed into a subchain-by-subchain query, which shortens the access path in the search process. The FHP in SCATC is not embedded in the transaction but embedded in the leaf nodes of the Merkle tree. When querying early historical transactions, the system will directly filter the user's recent transaction data.

To achieve rapid retrieval of data, we design the query algorithm as shown below for the SCATC index.

Algorithm 1 SCATC query algorithm

```
Input：Target account subchain
Output：Account transaction
1：TargetAccount_data=[]
2：p = LatestBlock.data
3：if p.Subchain_BlockNum<k:
4：   for i in range(LatestBlock, LatestSubchain_FirstBlock,-1):
5：      q= i.data
6：      for j in q:
7：         if j.Account==Target_Account:
8：            TargetAccount_data.append(j)
9：for i in range(LatestSubchain_kBlock,TargetSubchain_ kBlock,-k):
10：   q= i.data
11：   for j in q:
12：      if j.Account==Target_Account:
13：         TargetAccount_data.append(j)
14：for i in range(TargetSubchain_kBlock, TargetSubchain_FisrtBlock,-1):
15：   q= i.data
16：   for j in q:
17：      if j.Account==Target_Account:
18：         TargetAccount_data.append(j)
19：return TargetAccount_data
```

The algorithm first creates a list *TargetAccount_data* to save the data of the target accounts that have been accessed. Lines 2–8 of the algorithm visit the latest block in the 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 k th block, access the k th block of the previous subchain until the k th 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.

4 Experiment and Analysis

4.1 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, and the number of blocks in each subchain is $k(k > 1)$. 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 with the increase of k, 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 ATTC-based query method requires access to the complete transaction chain; the number of irrelevant blocks accessed is $t_{1.}$

$$t_1 = n(k-1) \tag{2}$$

In the SCATC-based query method, the number of blocks to be accessed in the initial query subchain is t_2.

$$t_2 = \frac{s}{k} + k - 1 \tag{3}$$

The number of blocks in the irrelevant subchain accessed is t_3.

$$t_3 = n - 1 \tag{4}$$

With the increasing number of users' transactions, n tends to increase monotonically. Equations (2) and (4) can be regarded as a linear function of t to n. In Eq. (2), the coefficient of the independent variable n is $k(k > 1)$, and Eq. (4) where the coefficient of the independent variable is 1. With the growth of n, the number of irrelevant blocks that need to be accessed increases rapidly based on the ATTC query method. The SCATC-based query method has a slower growth rate, and the larger the n, the more obvious the advantage of the SCATC-based query method.

4.2 Simulation Experiment

The simulation environment is a host computer, where the CPU is Intel(R) Core(TM) i7-5500U, 12 GB 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 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. Both query methods start from the latest block forward, so the query time in SCATC includes the time to locate the subchain. The simulation experimental data obtained are shown in Tables 1 and 2.

Table 1. ATTC

Number of blocks	Query time/Ms								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

Table 2. SCATC

Number of blocks	Query time/Ms								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 average value of each subchain of simulation experimental data of ATTC and SCATC is plotted as a line chart shown in Fig. 2. As the length of the transaction chain continues to grow, the query time based on the ATTC query method is constantly increasing. However, the query method based on SCATC has not changed significantly in query efficiency as the length of the transaction chain continues to increase.

For active users in the blockchain system, the length of the transaction chain has increased at a faster rate. From a theoretical analysis, whether it is based on ATTC or SCATC 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 visits to irrelevant blocks. The limited length of the transaction chain cannot cause a significant change in SCATC's query efficiency.

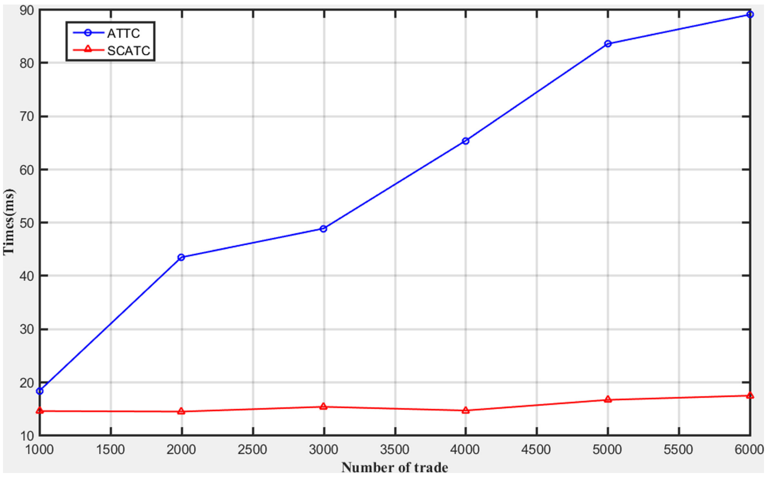

Fig. 2. Comparison of query efficiency

5 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. 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.

Acknowledgement. This work is financially supported by the National Natural Science Foundation of China under Grant No. 61962009, Major Scientific and Technological Special Project of Guizhou Province under Grant No. 20183001, Foundation of Guizhou Provincial Key Laboratory of Public Big Data No.2018BDKFJJ005, in part by Talent project of Guizhou Big Data Academy. Guizhou Provincial Key Laboratory of Public Big Data.([2018]01) .

References

1. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System[EB\OL] (2008). https://bitcoin.org/bitcoin.pdf.
2. He, P., Yu, G., Zhang, Y.F., Bao, Y.B.: Survey on blockchain technology and its application prospect. Comput. Sci. **44**(4), 1–7 (2017). (in Chinese with English abstract)
3. Yuan, Y., Wang, F.Y.: Blockchain: The state of the art and future trends. Acta Automatica Sinica 42(4), 481−494 (2016) (in Chinese with English abstract).[doi:https://doi.org/10.16383/j.aas.2016.c160158]

4. Wang, H.J., Dai, B.R., Li, C., Zhang, S.H.: Query optimization model for blockchain applications. Comput. Eng. Appl. **55**(22), 34-39+171 (2019)
5. Wang, Q.G., He, P., Nie, T.Z., Shen, D.R., Yu, G.: Overview of data storage and query technology in blockchain systems. Comput. Sci. **45**(12), 12–18 (2018)
6. Qu, Q., Nurgaliev, I., Muzammal, M., et al.: On spatio-temporal blockchain query processing. Futur. Gener. Comput. Syst. **98**, 208–218 (2019)
7. Yuqin, X., Zhao, S., Kong, L., Zheng, Y., Zhang, S., Li, Q.: ECBC: a high performance educational certificate blockchain with efficient query. In: Van Hung, D., Kapur, D. (eds.) Theoretical Aspects of Computing – ICTAC 2017, pp. 288–304. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-67729-3_17
8. You, Y., Kong, L.J., Xiao, Z.S., Zheng, Y.Q., Li, Q.Z.: Hybrid indexing scheme supporting blockchain transaction tracing. Comput. Intergrat. Manufact. Syst. **25**(04), 978–984 (2019)
9. Morishima, S., Matsutani, H.: Accelerating blockchain search of full nodes using GPUs. pp. 244–248 (2018)
10. Zhang, C., Xu, C., Xu, J., et al.: GEMˆ2-Tree: a gas-efficient structure for authenticated range queries in blockchain. In: 35th IEEE International Conference on Data Engineering (ICDE '19). IEEE (2019)
11. Jia, D., Xin, J., Wang, Z., Guo, W., Wang, G.: ElasticChain: support very large blockchain by reducing data redundancy. In: Cai, Yi., Ishikawa, Y., Jianliang, Xu. (eds.) Web and Big Data: Second International Joint Conference, APWeb-WAIM 2018, Macau, China, July 23-25, 2018, Proceedings, Part II, pp. 440–454. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96893-3_33
12. Jia, D.Y., Xin, J.C., Wang, Z.Q., Guo, W., Wang, G.R.: Efficient query model for scalable storage capacity blockchain system. J. Software **30**(09), 2655–2670 (2019)
13. Jiao, T., et al.: Blockchain database: a database that can be queried and tamper-proof. J. Software **30**(09), 2671–2685 (2019)
14. Zheng, H.H., Shen, D.R., Nie, T.Z., Kou, Y.: Query optimization of blockchain system for hybrid indexing. Comput. Sci. **47**(10), 301–308 (2020)
15. Ren, Y., et al.: Data query mechanism based on hash computing power of blockchain in Internet of Things. Sensors **20**(1), 207 (2019)
16. Wan, L.: A query optimization method of blockchain electronic transaction based on group account. In: Atiquzzaman, M., Yen, N., Zheng, Xu. (eds.) Big Data Analytics for Cyber-Physical System in Smart City: BDCPS 2020, 28-29 December 2020, Shanghai, China, pp. 1358–1364. Springer Singapore, Singapore (2021). https://doi.org/10.1007/978-981-33-4572-0_196
17. Wan, L.: An optimization method for blockchain electronic transaction queries based on indexing technology. In: Atiquzzaman, M., Yen, N., Zheng, Xu. (eds.) Big Data Analytics for Cyber-Physical System in Smart City: BDCPS 2020, 28-29 December 2020, Shanghai, China, pp. 1273–1281. Springer Singapore, Singapore (2021). https://doi.org/10.1007/978-981-33-4572-0_183