EtherH: A Hybrid Index to Support Blockchain Data Query

Pengting Du
Department of Computer Science and Technology, Ocean University of China
Qingdao, Shandong, China
dupengting@stu.ouc.edu.cn

Yingjian Liu*
Department of Computer Science and Technology, Ocean University of China
Qingdao, Shandong, China
liuyj@ouc.edu.cn

Yue Li
Department of Computer Science and Technology, Ocean University of China
Qingdao, Shandong, China
yueli@stu.ouc.edu.cn

Haoyu Yin
Department of Computer Science and Technology, Ocean University of China
Qingdao, Shandong, China
yhy@stu.ouc.edu.cn

Limin Zhang
Department of Computer Science and Technology, Ocean University of China
Qingdao, Shandong, China
zhanglimin1324@stu.ouc.edu.cn

ABSTRACT

As the underlying technology of cryptocurrency, blockchain is characterized by decentralization and immutability. It is considered as a new generation of disruptive technology. However, the underlying storage of blockchain only provides limited support for data query. This limits the utility of blockchain technology. To improve the efficiency of access to blockchain data, we design EtherH based on Ethereum, the most representative open blockchain system. EtherH is a hybrid index that can provide Single-V query and Range query on Ethereum blockchain. To make the index build more specific, EtherH supports elastic data building. A portion of the blockchain data is selected for index building. We connect the client to the Rinkeby network to obtain block data and conduct extensive experiments. Experimental results prove that EtherH performs well in both data insertion and data query.

CCS CONCEPTS

• **Information systems** → **Data management systems**; **Information storage systems**; *Information retrieval query processing*; Information systems applications.

KEYWORDS

Blockchain, Ethereum, Index, Data Query

ACM Reference Format:
Pengting Du, Yingjian Liu, Yue Li, Haoyu Yin, and Limin Zhang. 2021. EtherH: A Hybrid Index to Support Blockchain Data Query. In *ACM Turing Celebration Conference – China (ACM TURC '21), July 30-August 1, 2021, Hefei, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3472634.3472653

*Yingjian Liu is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ACM TURC '21, July 30-August 1, 2021, Hefei, China
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8567-1/21/07…$15.00
https://doi.org/10.1145/3472634.3472653

1 INTRODUCTION

As one of the underlying technologies of Bitcoin, blockchain technology is increasingly attracting attention. Blockchain is a new application of computer technology such as distributed data storage, point-to-point transmission, encryption algorithm, etc. It has the characteristics of decentralization, immutability and collective maintenance. Blockchain has huge application prospects in data-intensive fields such as medical management, finance and supply chain management. However, its limited data access limits its application.

As an open source public blockchain platform that introduces smart contracts, Ethereum is the most widely used blockchain application development platform at present. Ethereum is a peer-to-peer network, and each node needs to store data in the Ethereum, which is stored in LevelDB. It only supports insertion and query based on key values, and does not support complex queries.

In recent years, the research on blockchain data query has attracted more and more attention from scholars. EtherQL [1] was proposed to block data to an off-chain database for query. F. A. Pratama and K. Mutijarsa [2] enhanced and expanded query function defined in EtherQL. The method mentioned in VQL [3] was similar to [1], the difference is that [3] provides a verification service, which adds intermediate hashes to the database. That makes the query verifiable. M. Bartoletti, A. Bracciali et al. [4] and H. Kalodner, S. Goldfeder et al. [5] proposed to synchronize the blockchain data to obtain statistical information such as transaction graphs and transaction fees. These methods require more additional space consumption. S. Hu and C. Cai et al. [6] proposed a decentralized privacy-protection search solution to achieve trusted query, while S. Helmer et al. [7] used smart contracts to integrate part of the database functions into the blockchain to achieve diversified queries. However, restricted by smart contracts, their query capabilities are limited. vChain [8] was proposed a verifiable query framework, which adopts an accumulator-based authentication data structure (ADS) to focus on reducing the storage and calculation costs. GEM^2-Tree [9] was further proposed a new gas-efficient ADS, which supports authentication range query. A SGX-based blockchain range query verification scheme was proposed by Q. Shao et al. [10], the light client does not need to receive or validate any VO. These methods do not focus on improving query performance. Ebtree [11] was proposed to introduce the B+ tree index into Ethereum to support complex queries. But information that users can get from the returned results is limited. And B+ tree has extra pointers to handle.

In this paper, we propose a solution to the query problem of blockchain data. We use Ethereum as an experimental platform and propose a hybrid index – EtherH based on a B-Tree and a Skip-list. Our main contributions are as follows: (1) Construct EtherH and organize Ethereum blockchain data; (2) Based on EtherH, Single-V query and Range query are proposed to support the query of Ethereum blockchain data; (3) The effectiveness of the solution is proved by experiments.

2 PRELIMINARIES

In this section, we briefly introduce the data structure of blockchain and the architecture of Ethereum, a common blockchain platform.

2.1 Blockchain Structure

A blockchain consists of blocks linked by encrypted hash pointers. A complete block is divided into block header and block body. The block header contains more information. The contents of blocks are slightly different in diverse blockchain system. The data structure and relationship of blocks in Ethereum can be demonstrated in Figure 1. The block header contains the hash value to the previous block and uncle block (*ParentHash, UncleHash*), the address of the miner who excavates the block (*Coinbase*), the root of account state trie (*StateRoot*), receipts trie (*ReceiptsRoot*), transaction trie (*TransRoot*), the allowed Gas consumption (*GasLimit*) and Gas used by all transactions in the block (*GasUsed*), the height of block (*Number*), the time of block being mined (*Timestamp*), *Difficulty* in mining, *Nonce* used for validation. The block body consists of *TransList* and *UncleHeaderList*. Ethereum uses three modified MHT (Merkle Partricia Tree) to maintain the current block transactions, receipts generated after transactions, and the status of accounts in the system, namely transaction Tree, receipt Tree, and status Tree. The transaction tree is shown in Figure 1, and the other two trees have the same structure.

2.2 Ethereum Architecture

Through its proprietary cryptocurrency Ether, Ethereum provides a decentralized virtual machines (EVM) to handle peer-to-peer contracts. Ethereum is made up of three parts: the service layer, the core layer, and the application layer. The service layer includes *P2P* network, cryptography algorithm (*CryPto*), sharding (*Shard*) and *LevelDB*, etc. The core layer includes blockchain (*Blockchain*), consensus algorithm (*Consen*) and *EVM*, etc. The application layer includes smart contract (*Smart Contract*) and decentralized application (*DApp*), API interface (*API Interface*), etc. LevelDB is a Key-Value data storage system based on Log-Structured-Merge-Tree (LSM-Tree), which has high write performance. Ethereum has designed some interfaces to provide queries service. For example: *getBlock()* can query specific block information, *getUncle()* can query specific uncle block information, etc. By using these interfaces, users can also have some understanding of the Ethereum blockchain. Readers who want to learn more about other parts of Ethereum refer to [12].

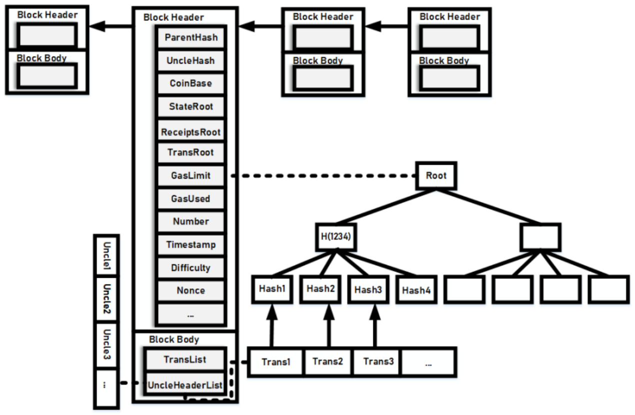

Figure 1: The data structure and relationships of blocks.

3 ETHERH DESCRIPTION

In the Section 3, we explain the existence of transaction data, receipt data, account data, etc in the Ethereum blockchain. Ethereum provides some APIs to meet the query requirements for blockchain. Although these APIs can provide some query functionality, the large amount of data in Ethereum blocks makes it inefficient and does not support complex queries. Based on this, we propose EtherH (Figure 2), which extracts block data from Ethereum and constructs indexes. EtherH consists of B-tree and Skip-list, which supports Single-V query, Range query.

3.1 EtherH Structure

As a multi-path self-balancing search tree, B-tree has better performance than binary search tree, balanced Binary (AVL) tree and other tree structures. Structurally, the B-tree is similar to the AVL tree, but the B-tree allows more children per node. Meanwhile, the B-tree also reduces the number of nodes and maximizes the effect of each disk I/O. Compared with B+ tree, all nodes in B-tree have Data domain, the query does not have to run to the leaf node and then return. That means the single query is faster. There is no extra leaf node pointer overhead. We consider using Skip-list to sort and merge block data into the B-tree index in batches. Skip-list in the original ordered linked list to increase the multi-level index. It has search performance comparable to AVL tree, red-black tree and so on. We serialize the index and store it on the local device. In this way, EtherH can help users achieve the purpose of monitoring and analyzing overall Ethereum blockchain.

In order to make EtherH more personalized, we propose the construction of elastic data attributes. That is, select part of the blockchain data to organize and insert these into the index for users to query. For example, we could optionally extract transaction hash, Gas consumption, and other data about transactions from the block. Then, we insert them into the index. There can be many combinations.

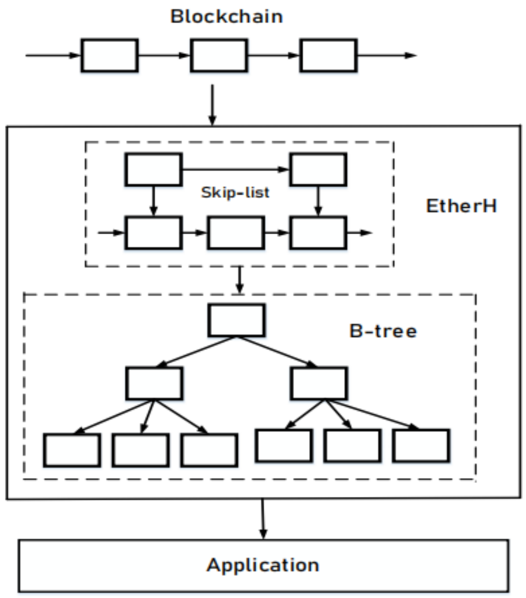

Figure 2: EtherH structure.

3.2 Insertion Blockchain Data

Considering that the dataset used to build EtherH may be different, multiple EtherHs are allowed. Users can adjust them according to their own needs.

After getting the data from the block, we need to insert the data into EtherH (Algorithm 1).

For the B-tree (set to order M):

(1) Before inserting a piece of block data, we need to load the root node of the B-tree index. If it does not exist, it means that the tree structure does not exist. Create a leaf node, then insert the data into it. This insertion ends. If not, go (2);
(2) Find the leaf node and insert the key and data according to the value of the key to be inserted. If the key values are equal during the insertion process, the two data will be merged. It also provides a preliminary organization of the data when building the index;
(3) After insertion, determine whether the current node needs splitting (whether the number of keys is greater than M-1). If it is not necessary, then stop. If it is necessary, splitting with the key in the middle of the current node as the center, insert the key in the middle into the parent node. The key points to the two parts of the original node after splitting respectively. Then, point the node to the parent node. Repeat the process until no splitting is needed;
(4) After the adjustment, the root of B-Tree is returned.

For the Skip-list, the insertion process is simple. And the data is inserted into the B-tree index at regular intervals, without further detail.

3.3 Query Blockchain Data

EtherH supports Single-V query, Range query for Ethereum blockchain.

Algorithm 1 Insertion Data (B-tree)

```
Input: The dataset from Blocks
Output: The root of B-Tree, the number of keys in B-Tree
1: Load root of B-Tree or create a new leaf node
2: Initialize $n = 0$
3: while $n < num(dataset)$ do
4:     Find the insertion position by searching within and between nodes
5:     Insertion the key and data
6:     if the number of keys in the node $> M - 1$ then
7:         split the node and point the node of parent node
8:     end if
9:     $n++$
10: end while
11: return the root and the number of keys
```

3.3.1 Single-V query. For the B-tree, search is divided into finding node and finding key in node. After the root node of the B-tree index is loaded, half-interval search is carried out inside the root node to compare the key to be searched with the key in the node. If equal, then record the data associated with this key; If not, compare other values within the node. After the node is compared, load the next node as needed. Repeat this process until finding a record equal to the key that is looking for. If not found until the leaf node, record the null value. When the search is complete, record the results and return them.

3.3.2 Range query. We define Range query as the generalized Single-V query. For the Range query, it is regarded as the Single-V query in the closed interval. All data corresponding to the key in the closed interval in the index is returned. As mentioned earlier, the result set returned is part of the overall blockchain data. Users can get the information they want from the returned data.

4 IMPLEMENTATION AND EVALUATION

To test the usability and performance of our method, we implement EtherH in Ethereum client – Geth v1.9. We build up the experiment platform on a computer with i5-6500U CPU and 16GB memory to synchronize block information over the Rinkeby network. We choose to use the Gas Used of transaction as the key, Number, Value and the identifier of the transaction in block as the data. These constitute the data attributes of the dataset. To better evaluate our proposal, we test EtherH in storage consumption, insertion performance and query performance with a starting point of 500,000 blocks and an end point of 4,000,000 blocks. Experiment adds 500,000 block each time. Due to the different of network speed or machine running state, the results of a single experiment is different. In the case of different number of blocks, we carry out multiple experiments to obtain the average value.

4.1 Storage Usage

We record the amount of data and storage usage when the number of blocks varied. From Figure 3, we can see that the changes of data volume and storage consumption are basically the same. That indicates that our scheme can adapt to the index construction requirements of different data volumes. From the storage usage,

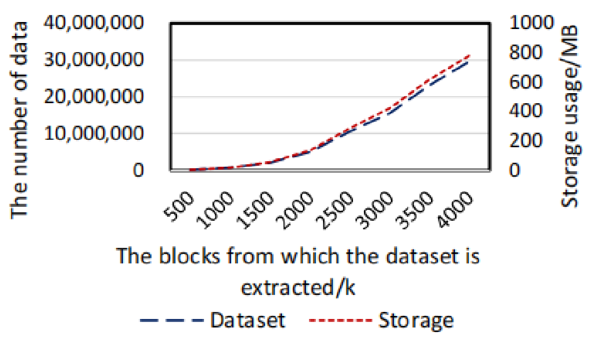

Figure 3: Storage space and amount of data.

Table 1: The average time to insert data for different number of blocks.

The number of blocks/k	The time to insert the dataset/ms	The time to insert a piece of data/ms
500	82.32	0.47984
1,000	160.95	0.23871
1,500	251.12	0.11896
2,000	337.41	0.06710
2,500	450.14	0.04256
3,000	620.99	0.03976
3,500	911.32	0.03929
4,000	1041.52	0.03511

although the space occupied more and more, but the overall consumption is small. For example, when the number of blocks reaches 4,000,000, the index size is 700 MBs. The block space used at this time is up to tens of GBs, while the storage space used by the index is only a few tenths of the whole blocks used. This is acceptable to the users.

4.2 Insertion Performance

To evaluate the insertion performance of our scheme, we compare the time required to insert data into the B-tree index with different numbers of blocks. Although the overall insertion time is getting longer and longer, we can find from Table 1 that the average insertion time of each data is shortened and tends to be stable. That indicates that our scheme is suitable for processing operations with large data volumes.

4.3 Query Performance

To better evaluate our solution, we conduct several tests on Single-V query and Range query, and record their respective response times. As we can see from Figure 4, query time increases as the block data increases. This is because when the amount of data increases, the data in the index and the number of nodes that the index needs to traverse will increase. So the time will naturally increase accordingly. On the whole, the query time is still at a relatively low level and is acceptable.

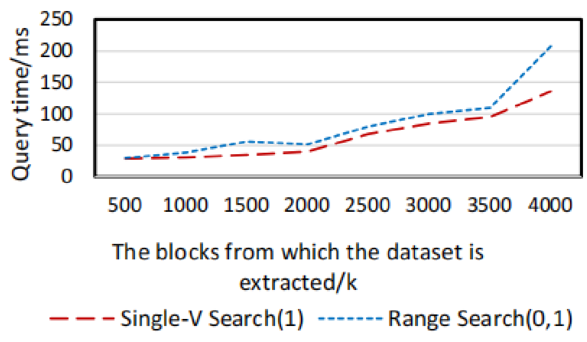

Figure 4: Time used for data query.

5 CONCLUSION

In this paper, we propose EtherH, a hybrid index that can query the Ethereum blockchain. EtherH is divided into a B-tree and a Skip-list. To make the index more targeted, we propose to construct the index with elastic data attributes. EtherH provides users with Single-V query and Range query, and inserts data in a relatively short time. We implement EtherH on Geth v1.9 and connect to the Rinkeby network to synchronize blocks. We conduct experiments to demonstrate the effectiveness of EtherH. In the future, we will further refine the details of expand its capabilities to achieve better storage, query performance and make it more universal.

ACKNOWLEDGMENTS

This work was supported partially by the National Natural Science Foundation of China under Grant No. 61572448, No. 61673357, and by the Key R&D Program of Shandong Province, China under Grant No. 2018GSF120015.

REFERENCES

[1] Yang Li, Kai Zheng, Ying Yan, Qi Liu, and Xiaofang Zhou. 2017. EtherQL: A Query Layer for Blockchain System. In Database Systems for Advanced Ap-plications, Selçuk Candan, Lei Chen, Torben Bach Pedersen, Lijun Chang and Wen Hua (eds.). Springer International Publishing, Cham, 556–567. DOI:https://doi.org/10.1007/978-3-319-55699-4_34
[2] Fariz Azmi Pratama and Kusprasapta Mutijarsa. 2018. Query Support for Data Processing and Analysis on Ethereum Blockchain. In 2018 International Symposium on Electronics and Smart Devices (ISESD), IEEE, Bandung, 1–5. DOI:https://doi.org/10.1109/ISESD.2018.8605476
[3] Zhe Peng, Haotian Wu, Bin Xiao, and Songtao Guo. 2019. VQL: Providing Query Efficiency and Data Authenticity in Blockchain Systems. In 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), IEEE, Macao, Macao, 1–6. DOI:https://doi.org/10.1109/ICDEW.2019.00-44
[4] Massimo Bartoletti, Andrea Bracciali, Stefano Lande, and Livio Pompianu. 2017. A general framework for blockchain analytics. arXiv:1707.01021 [cs] (November 2017). Retrieved May 8, 2021 from http://arxiv.org/abs/1707.01021
[5] Harry Kalodner, Steven Goldfeder, Alishah Chator, Malte Möser, and Arvind Narayanan. 2017. BlockSci: Design and applications of a blockchain analysis platform. arXiv:1709.02489 [cs] (September 2017). Retrieved May 8, 2021 from http://arxiv.org/abs/1709.02489
[6] Shengshan Hu, Chengjun Cai, Qian Wang, Cong Wang, Xiangyang Luo, and Kui Ren. 2018. Searching an Encrypted Cloud Meets Blockchain: A Decentralized, Reliable and Fair Realization. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, IEEE, Honolulu, HI, 792–800. DOI:https://doi.org/10.1109/INFOCOM.2018.8485890
[7] Sven Helmer, Matteo Roggia, Nabil El Ioini, and Claus Pahl. 2018. EthernityDB – Integrating Database Functionality into a Blockchain. In New Trends in Databases and Information Systems, András Benczúr, Bernhard Thalheim, Tomáš Horváth, Silvia Chiusano, Tania Cerquitelli, Csaba Sidló and Peter Z. Revesz (eds.). Springer

International Publishing, Cham, 37–44. DOI:https://doi.org/10.1007/978-3-030-00063-9_5

[8] Cheng Xu, Ce Zhang, and Jianliang Xu. 2019. vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases. In Proceedings of the 2019 International Conference on Management of Data, ACM, Amsterdam Netherlands, 141–158. DOI:https://doi.org/10.1145/3299869.3300083

[9] Ce Zhang, Cheng Xu, Jianliang Xu, Yuzhe Tang, and Byron Choi. 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, Macao, 842–853. DOI:https://doi.org/10.1109/ICDE.2019.00080

[10] Qifeng Shao, Shuaifeng Pang, Zhao Zhang, and Cheqing Jing. 2020. Authenticated Range Query Using SGX for Blockchain Light Clients. In Database Systems for Advanced Applications, Yunmook Nah, Bin Cui, Sang-Won Lee, Jeffrey Xu Yu, Yang-Sae Moon and Steven Euijong Whang (eds.). Springer International Publishing, Cham, 306–321. DOI:https://doi.org/10.1007/978-3-030-59419-0_19

[11] Huang XiaoJu, Gong XueQing, Huang ZhiGang, Zhao LiMei, and Gao Kun. 2020. EBTree: A B-plus Tree Based Index for Ethereum Blockchain Data. In Proceedings of the 2020 Asia Service Sciences and Software Engineering Conference, ACM, Nagoya Japan, 83–90. DOI:https://doi.org/10.1145/3399871.3399892

[12] Dr Gavin Wood. ETHEREUM: A SECURE DECENTRALISED G-ENERALISED TRANSACTION LEDGER. 39.