GBRQ: geographic location blockchain structure design supporting regional information query in Internet of Vehicles

Chang Zhou, Huimei Lu, Yong Xiang, and Jingbang Wu

Abstract—With the rapid development of the Internet of Vehicles, the data authenticity is also widely concerned. Blockchain has been introduced into the Internet of Vehicles by many works because of its decentralized, non-tamperable and traceable characteristics. However, the existing blockchain lacks contact with the real world, its application support for the Internet of vehicles is limited. In this paper, we propose a geographic location blockchain structure that supports regional geographic information queries. The Geohash encoded geographic information at the bottom of the blockchain to increase the authenticity of the blockchain data, which is used to re-establish a Region State Trie index structure with the geographic region as the key for the data on-chain, and has the ability of prefix query and branch region query, which improves the performance of the existing blockchain (Ethereum) to query the data on-chain in the region according to the geographic location. To further improve the query efficiency of the region state, a branch query method based on geographic region is proposed. Theoretical analysis and experimental research verify the effectiveness and practicability of the proposed technology.

Index Terms—Internet of Vehicles, blockchain, geographic location, Geohash, Region State Trie.

1 Introduction

With the rapid development of the Internet of Vehicles(IoV), people can obtain services such as traffic condition and safety warnings easily and quickly, to enjoy a safer and more comfortable driving environment[1]. The vehicles in the IoV are equipped with an On Board Unit(OBU). Through the OBU, the vehicle can obtain information such as GPS and driving status of vehicle. Malicious vehicles can seriously affect the effectiveness of IoV information by sending fake locations, etc[2], [3]. The major concern of the IoV are the credibility of the message sender and message reliability. Moreover, the huge data storage and access requirements make traditional centralized data management face great challenges[4]. Therefore, we need a storage access architecture that can ensure the authenticity and integrity of messages, the credibility of message sources and distributed data storage in the IoV.

As a distributed ledger technology, blockchain links the blocks storing transactions by hash. It has the characteristics of decentralization, data integrity and traceability, has been introduced into IoV by many works[5], [6], [7], [8], [9]. Ethereum[10] is another widely used blockchain platform after Bitcoin[11]. Many applications are built on Ethereum[12], [13]. The existing blockchain represented by Ethereum can identify the participating accounts, time, and content in each transaction, but it is not well related to the physical world, and the pure virtuality will reduce the cost of forgery transactions and the inability to track illegal users[14]. Therefore, we need to increase the connection between the blockchain and the physical world to improve its authenticity.

The IoV has a large number of location-based applications[1], [2], [3], [8], and its data information is usually bound to geographic location. Vehicles are only interested in information within a specific geographic region, so we can improve the authenticity of their data by adding geographic location information to the blockchain. Meanwhile, the existing blockchain has low support degree of geographic location-based transactions, which can not well meet the query requirements of geographic location-based large-scale transaction data on IoV. So that, blockchain needs to enhance geographic information support and rapid query ability of regional information to meet the needs of IoV.

The existing blockchain research on geographic location information mainly uses smart contracts to realize geographic location applications, such as authors in Ref. [8] realize vehicle location sharing through smart contract, so as to improve the accuracy of cooperative positioning; Chukwuocha et al.[15] use the distance between the vehicle node and the event location to measure the message credibility, and divide multiple overlapping regions according to the road network to limit the credibility of cross-regional messages; Kudva et al.[9] read the geo-

- *C. Zhou and H. Lu are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: changzhou@bit.edu.cn*
- *Y. Xiang is with Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.*
- *J. Wu is with School of Computer and Information Engineering, Beijing Technology and Business University, Beijing 102488, China.*

Manuscript received April 19, 2022; revised August 26, 2022.

graphic location and driving distance in the blockchain transaction simulated by the smart contract, and propose the proof of driving consensus mechanism to improve the consensus efficiency. The above research work shows that geographic location has great value in the application of IoV in the blockchain. However, these studies store geographic location data in the user data layer, and cannot use the blockchain's own security mechanism to ensure the security of location data, and there is no guarantee for the authenticity of the blockchain transaction data. In terms of the encoding method of geographic location, the latitude and longitude are used as the encoding method of geographic location of existing applications[8], [9], [15], and using one-dimensional characters to represent latitude and longitude coordinates can reduce the amount of data transmission and improve query efficiency[16]. Geohash[17] is a Base32-encoded quadtree index method that divides a block in layer M into n blocks in layer $M+1$. Due to its simplicity of update, Geohash is widely used in one-dimensional indexing to process spatial data[16], [18], [19]. Our previous study[16] also show that the vector map data structure encoded by Geohash has good performance in terms of data storage and data compression. Therefore, we choose Geohash as the encoding method of the geographic location in the geographic location blockchain.

The predecessors have carried out a lot of research works to improve the efficiency of blockchain data query[12], [20], [21], [22]. In general, there are three ways: One is off-chain query. The blockchain is only used to maintain data security, and an independent data query method is built off the blockchain. Such as IBM[20] establishes 4 different databases for ledger index, status data, historical data and block index to realize separating query; Oracle[22] uses CouchDB, which supports rich SQL queries, as the storage database to create indexes for both keys and values to improve query efficiency; the data of the blockchain is stored into the database of SAP[23] in the form of a virtual table to realize the off-chain query of the blockchain data. This method is mainly used for database suppliers. Although the query efficiency is improved, the data is transferred outside the blockchain, and its security is dependent on trusted entities very much. The second is application query, which realizes different query functions by building different applications on the blockchain. For example, FlureeDB[24] is a graphic database with blockchain core and ACID standard, which establish a data snapshot related to the block or time point for each query to improve query efficiency; Swarm[25] is a blockchain system built to handle high-level files or layered collections of various types of metadata, using the form of hierarchical set to improve the query efficiency; BigchainDB[21] is to add blockchain features on the basis of big data distributed database. Nodes establish different indexes and query APIs according to query requirements to realize different application queries and improve query efficiency. The third is contract query, which uses smart contract of blockchain to optimize the query function. vChain[12] uses smart contract to simulate the underlying storage structure of blockchain, and then establishes inter-block index and intra-block index of simulated blocks. Blocks and indexes are reconstructed in the smart contract to realize storage and query on-chain, but compared with that directly stored in the blockchain, the data amount reconstructed in the blockchain is much larger, and the internal query method of the blockchain is not improved. Compared with vChain, G Gürsoy et al. [13] do not reconstruct the block structure, but only establish an index structure for the stored content in the smart contract. The fourth is sharding query, which divides the large-scale blockchain into multiple independent parts by sharding, so as to improve the query efficiency by reducing the amount of data accessed. chainSQL[26] divides the blockchain into a fixed number of shards, and improves data query efficiency by dispersing transactions into different shards; Monoxide[27] divides the user address space into multiple zones, and each zone establishes an independent blockchain, thereby reducing the amount of data in each zone. This method improves the query efficiency within the shard by reducing the amount of data in the shard, but its regional division does not consider the actual geographic location characteristics, and it is not suitable for regional query of geographic location-related information.

In view of the above problems, we propose a blockchain structure named GBRQ based on a geographic location structure to support regional geographic information query, which integrates the hierarchical structure of the geographic location with the underlying data structure of the blockchain, which realize the linkage between the blockchain and the physical world, ensuring the security of location data and the efficiency of transaction query. We design a geographic Region State Trie based on the Merkle-Patricia Trie(MPT)[10] structure encoded by Geohash[17], increase the query ability of regional geographic information, and propose a branch region information query method. To summarize, our main contributions made in this paper are as follows:

1) We add Geohash encoded geographic location attribute at the bottom of the blockchain to ensure data security, increase the linkage between the blockchain and the real physical world and the authenticity of blockchain.
2) We use the hierarchical relationship encoded by Geohash to organize the hierarchical structure of geographic region state, realize the Geohash Merkle-Patricia Region State Trie (Regional State Trie), and improve the MPT structure to facilitate the data query of blockchain according to geographic region. We establish a branch query method based on branch geographic region to improve the efficiency of branch region data query.

The rest of the paper is organized as follows. Section 2 presents an overview of the geographic location blockchain structure, which are then described the schematic design in detail in Section 3. Section 4 intro-

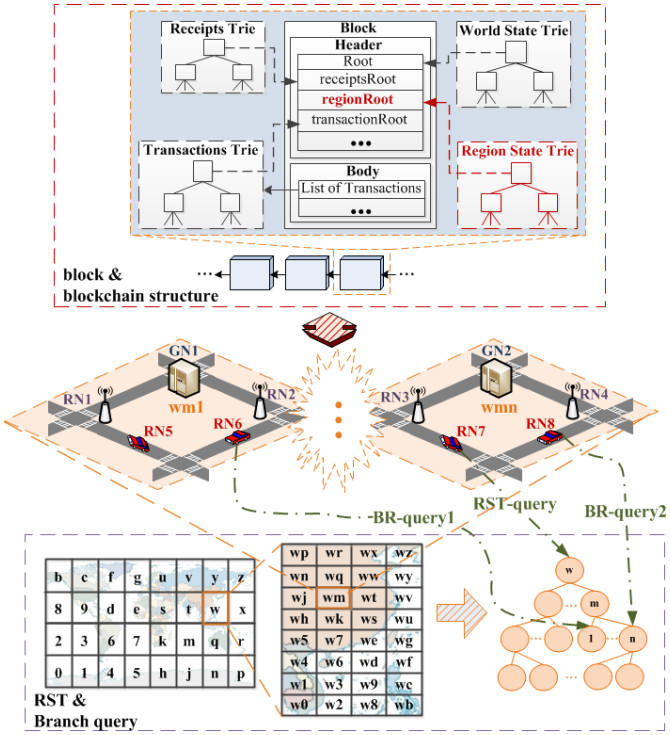


Fig. 1. Structure Model of GBRQ

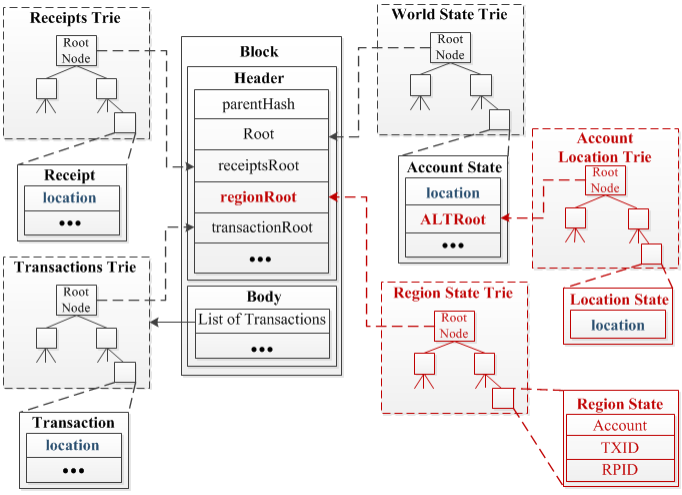


Fig. 2. Block Structure with Geographic Characteristics

duces the theoretical analysis. Section 5 shows the experiment and evaluation. Finally, we conclude our paper in Section 6.

2 Structure Overview

The structural diagram of GBRQ is shown in Figure 1, which consists of two parts: (1) On-chain data structure improvement (as shown in the block and blockchain structure part of Figure 1); and (2) query method(as shown in the Region State Trie and branch query of Figure 1).

On-chain Data Structure Improvement. We improved the data structure based on Ethereum to meet the geographic information query of IoV. Ethereum[10] adopts the account based data model, in addition to the parent block hash used to realize the chain structure and the transaction trie root hash of all transactions in a block, compared with Bitcoin, Ethereum uses Merkle Patricia Tree (MPT) to build its tree structure. The block also contains the root hash of the receipt trie for the transaction execution log and the root hash of the state trie for the account state data. As shown in the block and blockchain structure in Figure 1.

We implement the requirement of adding geographic location at the bottom of the blockchain by adding Geohash-encoded geographic location attributes to accounts, transactions, and receipts in the blockchain data structure; we add the root hash of the index structure of the geographic location attribute (Account Location Trie and Region State Trie) to the block to maintain the consistency and traceability of geographic location information, which is convenient for geographic information query and data credible verification.

1) Geographic location attributes are shown in blue $location$ in Figure2. The nodes in IoV are associated with the blockchain accounts. We add the geographic location attributes to the account to meet the mobility of vehicle nodes; vehicle locations are recorded in chronological order to facilitate historical location queries. The geographic location added in the transaction to achieve the purpose of adding geographic location attributes to the blockchain, which is also used to verify and update the account location of the transaction sender. The geographic location needs to be kept in the receipt as part of the transaction state information.
2) The root hash of the index structure of the geographic location attribute. Like the account balance, the Account Location Trie reflects the change of account state. Therefore, its root hash does not need to be written into the block header, but to the account state. As shown in the $ALTRoot$ in $Header$ of Figure 2. The Region State Trie records the global changes within the geographic region, which has the same status as the global changes of account state. Its root hash needs to be written into the block header. As shown in Figure 2, $regionRoot$, in $Header$.

Query Methods. In order to facilitate the query of geographic location attributes, we need to build an index structure for them and design branch query method according to the query requirements in different regions.

If there is no index structure for the account location in the blockchain, vehicle nodes in IoV need to go through two cyclic processes of traversing the block and traversing the transactions in the block when performing location verification or historical track query. Therefore, an Account Location Trie(ALT) should be established for each account. The index structure of geographic information

is required in the IoV[1]. Due to the increase in the amount of transactions within the region and the changes in account locations, the regional geographic information is the global state data. We need to establish a Region State Trie according to the user's query requirements for different levels of geographic region information. Nodes in different regions pay more attention to the state information of their own regions. If nodes in different branch regions all save the complete Region State Trie, each region information query needs to be carried out from the complete region. Therefore, we designed a branch region information query method to improve the efficiency of branch region information query.

1) The index structure of the geographic location attribute. Account Location Trie(ALT): An Account Location Trie of MPT structure with a time as key and a geographic location as $value$ is established for each account to realize the function of supporting geographic region information query within the blockchain. See details in Section 3.1. Region State Trie(RST): An improved MPT structure of region state with Geohash encoding features and hierarchical relationship of geographic regions, which is represented by the amount of transactions within the specified geographic region. We use the Geohash encoding of the geographic location as key, and the Region Element List in the current region (including account list(Account), transaction list(TXID) and receipt list(RPID)) as $value$ in this structure. See Section 3.2 for details.
2) The branch query method we designed requires that the branch index structure of the Region State Trie corresponding to the geographic region be saved according to the geographic location of the node, so as to narrow the query scope and improve the query efficiency. See Section 3.2.4 for details.

3 Design of GBRQ

The main challenge of GBRQ design is how to design the Account Location Trie and the Region State Trie to achieve the geographic location index, and the realization of the branch query method to improve the efficiency of branch region state query.

3.1 Account Location Trie

The vehicle nodes in the IoV have mobility, and the location changes are represented by the location attribute of the corresponding account in the blockchain. Like the balance of the account, all account locations are also written into the independent state database of each account as state information. In the IoV, no matter the historical track query or the specified geographical region traffic condition query, you need to query the historical location of the node. Therefore, an Account Location Trie(ALT for short) with MPT function is designed to implement the current account location data query.

1) Structure of Account Location Trie

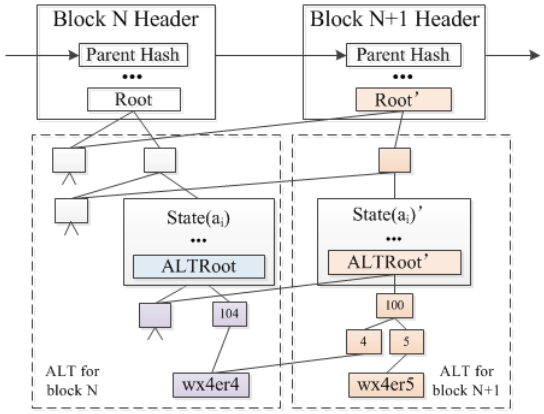


Fig. 3. Schematic Diagram of ALT Storage Changing

As the core data structure of Ethereum[10], MPT is a tree structure of single value identification key/value pairs formed by the combination of Merkle Tree and Patricia Tree. The advantage of the MPT structure is that the query efficiency brought by the compressed prefix and the fast location of differences brought by hash merging. Therefore, we use the MPT structure to build the Account Location Trie to achieve fast query of geographic information.

The Account Location Trie takes the record time of the account geographic location as the index, the account geographic location of the transaction sender verified by the miner node is stored in the leaf node; then calculate the hash value of leaf node, combine the hash values of two adjacent leaves into a string and store the hash value of the string in the parent node; the adjacent parent node repeats the above hash calculation process until the root hash value is obtained. At this time, the change of the entire Account Location Trie can be identified by the root node.

As shown in Figure 3, the storage of Account location Trie is divided into on-chain storage and index structure storage. The on-chain storage part is the hash value $Root$ of the root node of the state trie in the block header. The dashed box is the index structure, and the node of blockchain will maintain the latest index structure of Account Location Trie locally. The change of the account location will cause the change of the hash value of the root node of the Account location Trie in the account state $ALTRoot$, which will lead to the change of the hash value $Root$ of the root node of the state trie in the block header.

2) Account location query

The purpose of establishing an Account Location Trie is to improve the query efficiency of the current account location. The account location query is devided into latest location query and historical location query. Since the latest location of the

account is an attribute of the account, the latest location of the account can be obtained by reading this attribute, as shown in the Formula1, where $Location_{a_i}$ represents the location attribute value of the account a_i. The historical location needs to be queried based on the specific time. First, querying the latest account state, then reading the root node of the Account location Trie from the account state, and then querying the Account location Trie with the specified time as the index $alt_{a_i}(t)$, finally returning the index result, as shown in the Formula 2.

$$latest\_Location_{a_i} = Location_{a_i} \tag{1}$$

$$history\_Location_{a_i}(t) = alt_{a_i}(t) \tag{2}$$

3.2 Region State Trie

The purpose of the Account location Trie is to implement a location query related to the specified account. If you query the region state (which is detailed in Section 4.2 of Account location Trie), there are the following deficiencies: (1) First, you need to know all account IDs in advance and traverse the transaction; (2) Due to the original MPT structure, it can not achieve effective aggregation and fast query based on two dimensions of account and geographic region. We design a structure of improved MPT structure with Geohash encoding as the index – Geohash Merkle-Patricia Region State Trie, or Region State Trie (RST) for short. It records the region state information within a certain geographic region, and optimizes the MPT indexing method, and makes it efficiently support the prefix query, the latest regional information query and the branch query at the bottom of the blockchain. At the same time, we reduce the impact of external operations on the security of the blockchain, provide an interface for external access to facilitate application calls.

3.2.1 Structural Design

Although MPT structure can shorten the path by merging common prefixes, and quickly locate the difference locations of the two MPT structures. However only one value can be stored in the specified location. A bucket in MBT (Merkle Bucket Tree) is used to store the hash value of multiple key-values, which is the core index structure of Hyperledger[20] state data, but the hash value of the key-value cannot express the meaning of the key. We combine the merging common prefixes of MPT with the bucket of MBT to construct an index structure suitable for geographic location encoding and geographic region information query.

Figure 4 shows a schematic diagram of the Region State Trie structure. We followed the node classification method and hash method of MPT in the Region State Trie, modified the way of MPT hexadecimal encoding branches, uniformly adopted Geohash[17] as the encoding method of geographic region, branch node of Region State Trie and the geographic location of blockchain; we added

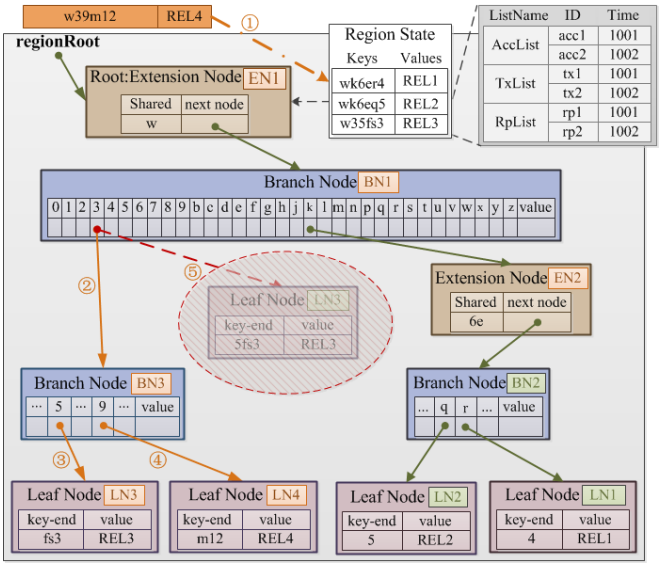


Fig. 4. Schematic Diagram of the Region State Trie Structure

the Region Elements List (REL for short) created by using the bucket of MBT to store multiple region state information at the same leaf node; the Region Elements Lists are arranged in chronological order, which is convenient to query the latest information; we designed a prefix query method to query the information of multiple leaf regions in the Region State Trie; we designed the branch query method to improve query efficiency of the region state in the specified branch region.

We use the Geohash encoding consistent with the encoding method of transaction location as the key of the Region State Trie, and the Region elements list of the state information in the region is $value$. There are three types of nodes in Region State Trie, namely:

Extension Node: Record the common prefix (shared) and followed branch node index (next node) of all unwritten nodes when traversing from the root node to this node in the key. As part of the key, the common prefix also uses Geohash encoding. As shown in the $Root$ node of Figure 4, since the three keys of $Region\ State$ have the same prefix w, the root node is an Extension Node. The purpose of constructing Extension Node is to expand the branches of the tree structure, so each Extension Node will follow a Branch Node.

Branch Node: Record the branch path traversed by different keys to this node. According to the Geohash encoding rules, the first 32 items of this node record the branch, and the 33rd item records the key end flag. As shown in the first branch node in Figure 4, since the second byte of the three keys of $Region\ State$ has two different encoding values of k and 3, it needs to be constructed into two branches of k and 3 in $BN1$. Since all the three keys have encoding values which are not traversed, there is no key end flag here. It is worth noting that if there is no combined common prefix before the branch junction, there is no extended node before the Branch Node, as shown in the Branch Node in Figure 4,

the previous node is the Branch Node $BN1$.

Leaf Node: Record the key encoding value and the corresponding Region Elements List that does not have a branch and has not been traversed. As shown in the leaf node $LN1$ of figure 4, because only the last encoding value 4 of the key of $REL1$ is left when it is traversed to $LN1$, the encoding value 4 is stored in the key-end of $LN1$, and the Region Elements List $REL1$ is stored in value. A Region State Trie with only one item is a leaf node.

Since the 14-byte Geohash encoding can meet the centimeter-level accuracy requirements for geographic locations at all latitudes of the earth[16], the key in Region State Trie is encoded with a 14-byte Geohash. The byte number of encoding keys in the Region State Trie is less than that of Ethereum and the depth of the tree is reduced. There is no need to transform and compress during storage, which improves the storage access efficiency. We adopt the Recursive Length Prefix (RLP for short)[10] storage method in Region State Trie to facilitate the storage requirements of different data amount.

3.2.2 Construction and Update

Region Elements List. The region state information is stored in the Region State Trie in the form of Region Elements List. The Region Elements List includes account list, transaction list and receipt list. The region element contains two attributes: ID and time. ID refers to the account ID, transaction ID and receipt ID in the account list, transaction list and receipt list respectively. The Region Elements List needs to include all region elements in the geographic region, and the Region Elements List of any two regions has no duplicate data. only the latest transaction time of the account is retained in the same account list to indicate the activity of the account.

Construction of Region State Trie. The construction of Region State Trie includes top-down storage path planning and bottom-up node value hash merging.

1) Storage path planning. The storage path planning refers to the construction process of all branch nodes starting to the final leaf node from the root node of the Region State Trie in accordance with the geographic location encoding region element list. The initial state of the Region State Trie is empty, the first data is written to the root node; when there is the same geographic location encoding as the region element list to be planned, the existing region element list should be updated; otherwise, the geographic region encoding where the current blockchain is located is used as the common prefix to plan the storage path.
   Take the Figure 4 as an example to describe the storage path planning process. There are three sets of region state information in region state list with key-values of $wk6er4 - REL1$, $wk6eq5 - REL2$ and $w35fs3 - REL3$ which are shown in Figure 4. The stored procedure is as follows: (1) the first region element list $REL1$ is stored in the empty Region State Trie, and a leaf node with key $wk6er4$ and value $REL1$ is constructed; (2) a leaf node already exists when the second region element list $REL2$is stored, the common prefix $wk6e$ of the geographic location encoding of $REL1$ and $REL2$ need to be found, and a new extension node with this prefix as the key is established, then an expansion node $BN2$ including two branches of r and q is established according to the subsequent encoding values of $REL1$ and $REL2$, and finally the leaf nodes of $LN1$ and $LN2$ are established with key-ends are 4 and 5 and values are $REL1$ and $REL2$; (3) the process of the third region element list $REL3$ is stored into the Region State Trie is similar to that of $REL2$. The storage path planning process of the Region State Trie is shown in Algorithm 1.

Algorithm 1 Storage path planning algorithm of Region State Trie

Require: R, w

function STORAGE_PATH_PLANNING(R,w) //w represents the Geohash region encoding that R belongs to
  cur_ Node is the node with the longest prefix of w
  sub_region = w removing the prefix of cur_node and w
  if $length\ of\ sub\_region > 0$ **then**
    LeafNode(sub_region[1:])=R
    BranchNode[sub_region[0]]=
      LeafNode(sub_region[1:])
    if $cur\_node.type \neq BranchNode$ **then**
      LeafNode(cur_node.key[1:])=cur_node
      BranchNode(cur_node.key[0])=
        LeafNode(cur_node.key[1:])
    end if
  else
    if $cur\_node.type == BranchNode$ **then**
      BranchNode.value=R
    else
      R is stored into REL
    end if
  end if
end function

2) The node hash value merging process of the Region State Trie is consistent with that of the Account Location Trie. As shown in Figure 4, the hash value of $BN2$ is the merged hash value of leaf node value hash of $LN1$ and $LN2$.

Region State Trie Update. The update process includes three parts: storage path update, update of region element list and update of the node hash value. When the storage path of the Region State Trie generated by the key to be updated does not overlap with the existing path in the tree, a new branch of the Region State Trie needs to be established, which is the storage path update. When the storage path of the Region State Trie generated by the key

to be updated is the same as the existing path in the tree, the new region element list needs to be updated to the original region element list, which is the region element list update. The above two update processes need to be accompanied by the update process of node hash value.

The key of the item to be updated in the Figure 4 is $w39m12$, and the $value$ is $REL4$ (step ①). When querying $BranchNode\,[3]$, the record is not empty, you need to compare the key-end of the leaf node. Since the key encoding of $REL4$ is traversed to $9m12$, the key of the original leaf node $LN3$ is $5fs3$, and the first encoding is different, it will enter the storage path update process, that is, a branch node is established (step ②), and then leaf nodes $LN3$ and $LN4$ which have keys of $fs3$ and $m12$ are created respectively (steps ③ and ④). Then, we enter the node value hash update process, that is, the hash values of $LN(3), LN(4)$ are written into the 5 and 9 branches of $BN\,[3]$ respectively; finally, the original $LN(5fs3)$ is deleted (step ⑤). One round of Region State Trie updating is over.

3.2.3 Prefix Query

The Region State Trie supports MPT query and prefix query. MPT query requires the same number of encoding bytes of the query area and the number of encoding bytes of the key of the construction tree, and the query result is empty or the value of the leaf node. It is impossible to query the regional information with the number of encoding bytes different from the key. To query different levels of regional information according to the prefix of the geographic location encoding, we designed a prefix query method. Prefix query requires the Geohash encoding of the region to be queried as the prefix of the key of Region State Trie, and the query result is empty or the combined value of all leaf node values of the branch where the prefix is located, as shown in the pink part in Figure 5. Prefix query includes Region State Trie query and query state cache.

1) Region State Trie query. A prefix matching method is designed to implement the prefix region query of the Region State Trie. The implementation process is shown in the Formula 3, where the $input$ represents the encoding of the region to be queried, the cur_key indicates the encoding of the current key to be queried, and the lck represents the encoding length of the cur_key. $parent\_node$ represents the parent node of the branch to be compared, the $parent\_key$ indicates the encoding of the key of the $parent\_node$, and the lpk represents the encoding length of the $parent\_key$. When comparing with the root node of Region State Trie for the first time, $cur\_key = input$; when $parent\_key$ is the prefix of cur_key, cur_key is the remainder without prefix; when the leaf node is not reached after traversing $input$, take the key $next\_key$ of the child node of $parent\_node$ as cur_key, and traverse the child nodes one by one until all the leaf nodes of $parent\_node$ are traversed.

$$cur\_key = \begin{cases} input, & parent\_node = \emptyset \\ cur\_key[lpk..lck], & cur\_key[0..lpk] = parent\_key \\ next\_key, & other \end{cases} \tag{3}$$

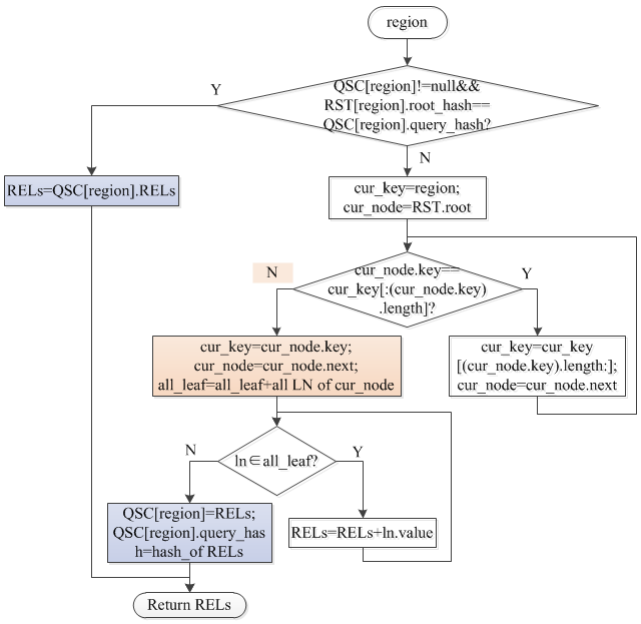


Fig. 5. the Flow Chart for Prefix Query of Region State Trie

2) Query state cache. Obtaining the regional geographic information of the IoV ultimately requires querying the region element list. When the region element list to be queried has not changed, caching the query state can improve the query efficiency of the region element list. We designed a query state cache (QSC) method to temporarily save the query results of the region element list. As shown in the blue part in Figure 5, $region$ is used as the key of region state cache to record the region encoding of the query request; $RELs$ stores the region element lists corresponding to the $region$; $query\_hash$ is the hash value of the query state cache. When the region information of $region$ is queried, to check whether $QSC[region]$ is empty firstly. If not empty, the hash value of the query state cache is compared with the region state tree node; if $RST[region].hash = QSC[region].query\_hash$, the query status cache result is returned directly.

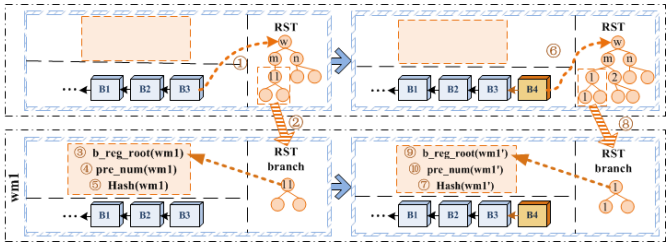


Fig. 6. Schematic Diagram of Branch Query Method

3.2.4 Branch Query

Compared with geographic information outside the region, vehicles need geographic information of their own regions provided by blockchain nodes to meet the requirements of IoV services. We propose the branch query method, the blockchain node caches the region state data of their own geographic region so that to improve the efficiency of branch query.

1) Branch region cache. The contents of the branch region cache are the root node, the Region State Trie branch of the geographic region where the node is located, and the branch node prefix. In this paper, we define the branch node prefix as the encoding length of all parent node keys of the Region State Trie branch which is cached by the node in the global Region State Trie.
Figure 6 is a schematic diagram of the establishment and update of the branch query method. In the figure, the node whose geographic region encoding $regionID$ is empty adopts the global Region State Trie query method, and the node whose $regionID$ is $wm1$ adopts the branch query method. In the figure, a blue dashed box is the storage state of the blockchain node at some point, including two parts: the orange box represents the state cache of the branch region of the current node, and the lower part represents the chain link relationship of the blockchain. The process of establishing branch region cache for node $wm1$ is as follows: ①read the root node of the global Region State Trie RST from the latest block $B3$, ② query the branch of $wm1$ in the global Region State Trie, and ③ obtain the root node $b\_reg\_root(wm1)$ with key 11, ④ get the prefix $pre\_num(wm1)$ of the branch region node $b\_reg\_root(wm1)$, ⑤ calculate the hash value $Hash(wm1)$ of the root node of the branch of Region State Trie.
After the node using the branch query method exits, its branch region cache is invalid, and the cache needs to be re-established when starting again. The algorithm for finding the root node of the branch region state and the prefix of the branch node is shown in the Algorithm 2.
2) Branch region update. When the region state branch corresponding to $regionID$ of the node using the branch query method changes, the local branch region state cache needs to be updated. As the structure of the Region State Trie will be changed, the hash value and prefix of the branch corresponding to $regionID$ will be changed, too. Therefore, both the hash value of the branch and the prefix of branch node need to be updated. After the node $wm1$ completes the synchronization process of the block $B4$, a new Region State Trie is generated (step ⑥), and then the root node hash of the branch of Region State Trie $Hash(wm1')$ needs to be calculated (step ⑦). $Hash(wm1') \neq Hash(wm1)$ indicates that the branch has changed, re-search the branch of Region State Trie of the node (step ⑧) and update the branch region state cache to $b\_reg\_root(wm1')$ and $pre\_num(wm1')$.
3) Branch query. When the actual query region is prefixed by the branch region where the node is located, there is a cache of the Region State Trie branch where the region to be queried is located locally, and the internal branch state query of the region where the node is located is realized by means of the cache. As shown in the Figure 6, the node of $wm1$ queries the region state of the range of $wm1$or $wm11$, which is the branch query. In the branch query, the calculation process of the current branch node prefix cur_pre_num, the current query region encoding cur_query_reg and the root node of current branch Region State Trie $cur\_b\_reg\_root$ is the query situation of the branch region of Formula 4-Formula 6.

$$cur\_pre\_num = pre\_num(regionID) \quad (4)$$

$$cur\_query\_reg = cur\_query\_reg[cur\_pre\_num ..length(cur\_query\_reg)] \quad (5)$$

$$cur\_b\_reg\_root = b\_reg\_root(regionID) \quad (6)$$

When the node does not set $regionID$, or when the actual query region is the upper region of the node's region or other regions that are not prefixed by the node's current branch region, because there is no corresponding region state cache, it needs to be queried in the global Region State Trie

Algorithm 2 Branch Region State Cache

Input:RST.root,regionID
Output:b_reg_root,pre_num

```
v_reg = regionID
pre_num=0
b_reg_root = RST.root
while v_Reg is not the prefix of b_reg_root.key do
    pre_num = pre_num + len(b_reg_root.key)
    v_reg = v_reg remove the prefix of b_reg_root.key
    b_reg_root = b_reg_root.next
end while
return b_reg_root,pre_num
```

of the corresponding region state. As shown in Figure 6, when the node of $regionID = \emptyset$ queries the region state or the node of $regionID = wm1$ queries the region state in the range of wm or $wm2$, which is the Region State Trie query.

4 Theoretical Analysis

In this section, we perform a theoretical analysis of the security and query time complexity of the geographic location blockchain structure.

4.1 Security Analysis

Storage security.

1) The blockchain adopts asymmetric encryption algorithm, which can resist many traditional security attacks, and its distributed storage structure ensures storage security[8], [15], [9], [5], [6], [7].
2) Locations are written into blocks through transactions, which can achieve consistency and tamper-resistance according to the distributed consensus mechanism and synchronization mechanism.
3) The geographic location attribute is added at the bottom of the blockchain. Although there is a risk of exposing the location, the authenticity of the transaction is increased by adding the location attribute to the transaction, which provides security guarantee on the other hand.

Query security.

1) Privacy. The original blockchain does not provide location information and there is no risk of exposing the location. We add location attributes inside the geographic location blockchain. Although the location proves the authenticity of the transaction, if the location access is unlimited, there is a risk of privacy leakage. Therefore, according to the usage scenarios of location information, we divide location access rights into internal index usage and whether it is one of the two parties to the transaction. During the internal indexing of the blockchain, the access to location information is unlimited. In other cases, only the both parties can access the specific location information, so as to achieve the purpose of restricting the use of location information.
2) The input is non-repeatable. Since only one transaction and one input of location can occur in the same account at one time, and the Account Location Trie takes time as the key, and the key is not repeated. That is, there will be no case that one time corresponds to two locations in the Account Location Trie.
3) Query reliability. When querying the state of the same region, any two nodes return the same query results. Suppose that there are two nodes that query the same region state information with different results. Since the honest nodes in the whole network account for the majority, the root node hash of the Region State Trie of all nodes is the same, so there will be no different branches. Contrary to the assumption, which also proves the query reliability.

It can be seen through the above security analysis that our solution has no impact on the security of the blockchain, and the reliability of the query can also be guaranteed. Although there is a risk of exposing account privacy, it can be improved by adding simple query restrictions.

4.2 Time Complexity Analysis of Querying

We analyze the query time complexity of querying region state information in the no-location index method, the Account Location Trie method, the global Region State Trie method, and the branch query method in this section.

1) No-location index method.
Existing blockchains such as Ethereum have no indexing method for geographic location, which is called no-location index. There are two ways to add location information to Ethereum. One is to use smart contracts to store locations, but they cannot be associated with ordinary transactions. If smart contracts are used to achieve storage and access data on-chain, the amount of data storage and storage cost will be increased. Another is to store location information in the transaction in the form of extra data. The location data needs to be stored in RLP encoding format and then written into ordinary transaction. The query process mainly includes a cyclic search process of block query and transaction query. When obtaining the transaction location, RLP decoding process needs to be completed, and then compared with the range to be queried. We select the latter one for comparative analysis.
Assuming the number of blocks is m, the transaction amount is n, the average transaction amount packaged in each block is $\frac{n}{m}$, then the query time complexity is $O(m \times \frac{n}{m}) = O(n)$.
2) Account Location Trie method. The steps to query the region state with the help of the Account Location Trie are as follows: First, get all account IDs. To get the state of the entire region from the perspective of the Account Location Trie (mainly refers to the number of transactions), we first need to count all historical account information in the region. Second, get all transactions of the account. We need to get the start and end time of the query. The start time is when the account sends the first transaction, and the end time is when the account sends the latest transaction. Due to the randomness of the transaction, we can determine it from the block generation time. We choose block 0, i.e. the timestamp of the genesis block as the start time, because genesis block does not contain

any transactions. We choose the timestamp of the latest block in the current blockchain as the end time to achieve full coverage in time. Third, transaction location filtering. In the process of step 2, it is also necessary to filter transactions according to the query region. According to Geohash encoding rules, it is only necessary to check whether the query region is the prefix of the transaction location.

Assuming the number of blocks is m, the transaction amount is n, the number of accounts is a, then the average transaction amount of each account is $\frac{n}{a}$. Account Location Trie is a Merkle tree, which has the query time complexity of $O(\log N)$, the N is the number of leaf nodes[28]. The Account Location Trie takes time as the key, and each account has an independent Account Location Trie. Therefore, the number of leaf nodes is the transaction quantity of the account, so the query time complexity of each account is $\log \frac{n}{a}$, and the overall query time complexity is $O(a \log \frac{n}{a}) \approx O(\log n)$, then the query time complexity of the Account Location Trie method is less than that of the no-location index method.

3) Global Region State Trie method.
The Region State Trie is also a Merkle tree, the Merkle tree has the query time complexity of $O(\log N)$, the N is the number of leaf nodes. Since the Region State Trie is stored in bucket structure at the leaf node, we suppose the number of transactions stored in the region element list at the leaf node in the Region State Trie is s, then the number of leaf nodes in the Region State Trie is n/s, and the query time complexity of the global Region State Trie is $O(log(n/s))$, the query time complexity of the global Region State Trie method is less than that of the Account Location Trie method.
4) Branch query method.
Suppose the number of leaf nodes of the Region State Trie cached in the branch region query method to l, the query time complexity of branch query method is $O(log(l))$. Because $l < n/s$, the query time complexity of the branch query method is less than the global Region State Trie method.

The theoretical analysis result of query time complexity is that in the case of region state query with the same transaction content and the same amount of data, the no-location index method $>$ Account Location Trie method $>$ global Region State Trie method $>$ branch query method.

5 Experiments and Evaluation

Based on the open source project Ethereum, we have implemented a prototype framework of geographic location blockchain structure supporting region state querying in IoV. In this section, we first introduce the experimental environment and experimental data used in this work. Then, we evaluate the performance of the geographic location blockchain structure with the region state query function.

5.1 Experimental Setup

1) Experimental environment. Our prototype framework is implemented on the basis of Ethereum version 1.9.12 (March 16, 2020) which is written in Go. The physical environment of the experiment is a desktop computer with 4-core Intel Core i5-4690k 3.5GHz processor and 7.7GB memory, running on Ubuntu 14.04.
2) Data generation. The transaction node randomly generates the geographic location data represented by 14-byte Geohash encoding in the specified geographic region and writes it into the transaction. Each transaction simulates the movement of the vehicle. The no-location index blockchain stores location data into the blockchain in the form of additional data of ordinary transaction, and the geographic location blockchain structure stores it in the form of location attributes of the transaction. Each transaction node sends a transaction to the blockchain every 200ms.
3) Data acquisition. The query node completes the query about transaction quantity of the region every 10s according to the fixed index method, which queries 200 times continuously.
4) Comparison protocol. As in the previous section, our comparison protocol includes four parts: original no index method, Account Location Trie index method, global Region State Trie index method and branch region query method.
Original no index method(ORG): Contents of part 1 in Section 4.2. In the original Ethereum, the location data of the transaction is stored in the form of additional information. Since there is no index for additional information in the blockchain, the query needs to be completed with blocks and intra-block transactions. This method with no dedicated location query is called the original no index method.
Account Location Trie index method(ALT): Contents of part 2 in Section 4.2. In the geographic location blockchain, the Account Location Trie is used to query the region state from the perspective of the account. This method is called the Account Location Trie index method.
Global Region State Trie index method(RST): Contents of Section 3.2.1-3.2.3. In the geographic location blockchain structure with Region State Trie, the nodes use a global query method to complete the region state query work, which is called the global Region State Trie index method.
Branch region query method(BR): Contents of Section 3.2.4. In the geographic location blockchain structure with branch query method, the nodes

use the branch query method to complete the query work for the cache of the current branch region state.

5) Experimental scenario. Four transaction nodes are in different regions (each node's location movement range is four 4-byte Geohash encoding regions, over an region of about $580km^2 * 4$, and the sending location is any continuous location within the moving range), sending one transaction every 200ms. One miner node is responsible for packaging transactions. One query node is responsible for the query work. In the geographic location blockchain structure, one of three query methods, such as the account location tree index, the global region state tree index or the branch region query method, in each query cycle is adopted by the query node independently.

5.2 Performance Analysis

In this section, we evaluate the performance of the geographic location blockchain structure prototype supporting region state query from different aspects. First, we measure the two performance indicators of build time and data amount of the no-location index blockchain and the geographic location blockchain; then we perform a quantitative analysis of the query time of the original no index method(ORG), Account Location Trie index method(ALT), global Region State Trie index method(global) and branch region query method(BR) when querying different geographic regions under different transaction amount; finally, we further analyze the impact of node region range and query range on branch region query method.

1) Build time. We take the successful packaging time of the same number and the same content of transactions as the build time, then we make statistics on the build time of five cases, such as 8000, 16000, 24000, 32000, 40000 transactions separately. Figure 7 shows the comparison of build time. The build time and time growth trend of the geographic location blockchain structure and the no-location index blockchain are basically the same, which both increase with the increase of the transaction amount on the whole. The build time of the geographic location blockchain structure increases about 0.31% on average compared with the no-location index blockchain, indicating that although the index structure is added to the geographic location blockchain structure, the build time does not increase significantly.
2) Data amount. The experimental settings of data amount statistics of is same with building time statistics. Since the data content stored by different blockchain nodes is different, we make statistics from the data amount on-chain and the local data amount respectively. Data amount on-chain: Only the data amount of the block itself is counted. Local data amount: Count the complete data amount saved locally, including state data. The comparison results of data amount are shown in Figure 8.

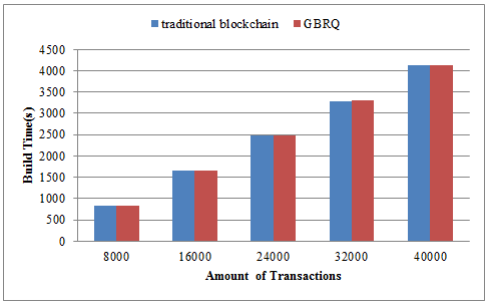


Fig. 7. Build Time

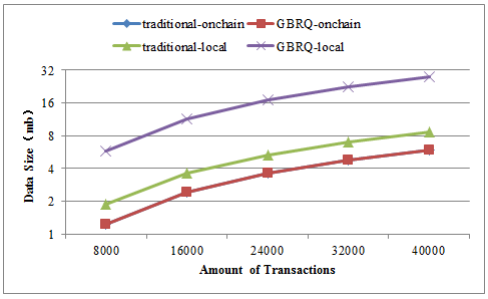


Fig. 8. Comparison of Data Amount

In terms of the data amount on-chain, as shown in Figure 8, the amount of on-chain data increases as the number of transactions increases in both cases. Compared with the no-location index blockchain, the on-chain data amount of the geographic location blockchain structure is slightly increased, with an average increase of about 0.27%. Because the geographic location blockchain structure only adds the root nodes of the Account Location Trie and the Region State Trie in the block header, and the data contained in the transactions included in the block body is consistent. It shows that the geographic location blockchain structure has no significant impact on the data on-chain.

From the perspective of local data amount, roughly speaking, the data amount of the two methods increases with the increase of the number of transactions; the data amount of geographic location blockchain structure has increased significantly, which is about 3.2 times more than that of the no-location index blockchain on average, indicating that the amount of database data of the geographic location blockchain structure is significantly higher than that of the no-location index blockchain. The reasons are as follows: 1) The storage of the account location trie built for each account will continue to expand as the number of account transactions increases; 2) The storage of the Region State Trie will continue to

expand according to the increase of the number of transactions in the region. The statistical results of the database data amount show that the index structure of the geographic location blockchain structure trades off space for time.

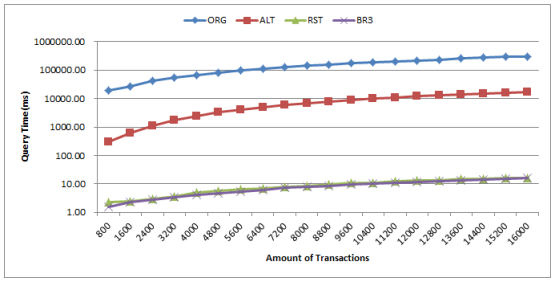
3) Query time. Query time refers to the time spent by different index methods to query the amount of transactions within the same geographic region when the amount of transactions on-chain is the same. During the query time statistics, we count the cumulative average value according to the fixed interval amount of transactions and count the query time of the transaction quantity in the geographic region with the query range of 3-6 byte Geohash encoding respectively. The statistical results are shown in Figure 9. The branch region selected by the region node of the branch query index in Figure 9 is the same as the Geohash range selected by the query region. Therefore, in the four query result diagrams in Figure 9(a)-(d), the query range of nodes corresponding to the branch region index is also different.

In general, when the query range is a 3-byte Geohash range, the amount of transactions contained in the region grows fast due to the large region. Therefore, the query time of the four query methods increases greatly with the increase of the amount of transactions in this range. When the query range is a 6-byte Geohash range, the query time increases, but at a small rate, because the region is small and the amount of transactions contained within the region grows slowly.

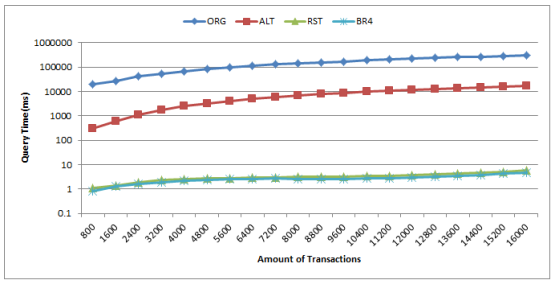
From the perspective of Account Location Trie index and original no index, when the query region is in the range of 3-6-byte Geohash respectively, the average query time of original no index is 19s-306s, while the average query time of Account Location Trie index is 0.3s-17s. The average query time of Account Location Trie index is 24 times higher than that of original no index, which shows that Account Location Trie index has greatly improved the query efficiency compared with original no index of querying region state.

From the perspective of global Region State Trie index and Account Location Trie index, when the query region is in the range of 3-6-byte Geohash respectively, the query time of global Region State Trie index is 1.4ms-9.2ms on average, which is at least 99.8% less than that of Account Location Trie index, indicating that the global Region State Trie index of Region State Trie has better region query efficiency.

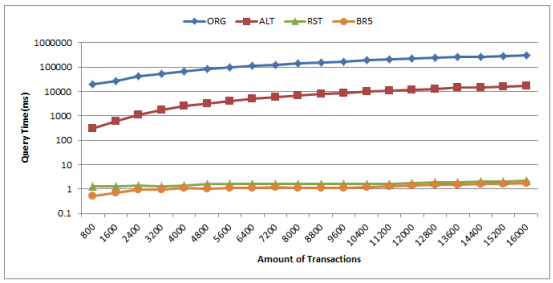
From the perspective of branch region index and global Region State Trie index, when the query region is in the range of 3-6-byte Geohash respectively and the query range of region index is also in the corresponding range of 3-6-byte Geohash, the average query time is 1ms-8.4ms, which is about 21.7 % less than that of global Region State Trie index, showing that the branch region index efficiency of Region State Trie is also improved. These experimental results have validated the time complexity analysis in theorem.



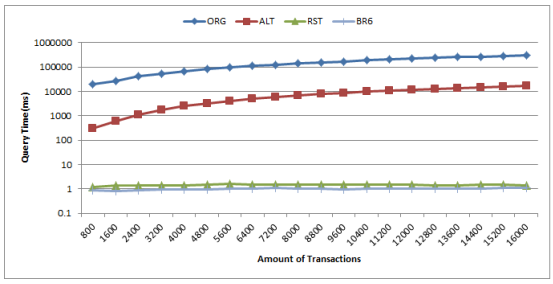
(a) The query range is geographic region of the 3-byte Geohash encoding



(b) The query range is geographic region of the 4-byte Geohash encoding



(c) The query range is geographic region of the 5-byte Geohash encoding



(d) The query range is geographic region of the 6-byte Geohash encoding

Fig. 9. Comparison of Query Time of the amount of transactions in different geographic regions

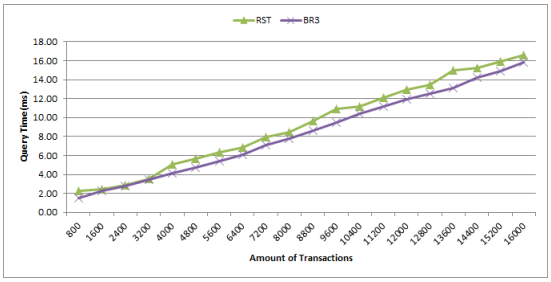
4) Analysis of branch region query method.

We further analyze the branch query method. We make statistics on the region query time of the

branch region index method under the geographic region of different region nodes in the same query range. The statistical results are shown in Figure 10. The $BR3 - BR6$ in the Figure 10 indicate that in different regions represented by Geohash encoding of different lengths, the region node completes the region state information query process.

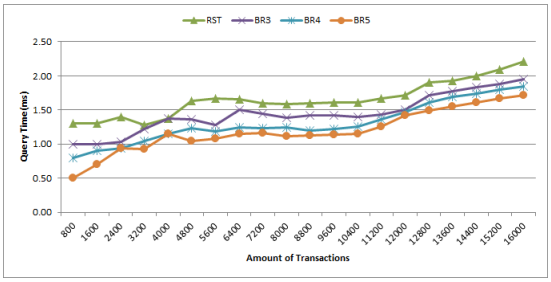
When the amount of transactions is small (about 1200), the region query time of global Region State Trie index is the same as that of branch region index when the range is 3-byte Geohash encoding; with the increase of the amount of transactions, the query advantages of the branch region query method are gradually improved. It can also be seen from the results that the query time is greatly affected by the amount of transactions within the query range. When the amount of transactions within the query range changes little, the query time is almost the same. For example, when the amount of transactions in Figure 10(b) is 5600-10400, the query time of $BR4$ is basically the same; in Figure 10(d), because the small query range, the amount of transactions in the region is small and changes little, so the query time does not change much.

When the region of node is consistent with the query range, the query time of branch region query method is reduced by about 21.7% compared with that of the global Region State Trie index on average; when the query range is in the 4-6-byte Geohash encoding range and the range of region index is 3 byte, the query time of branch region query method is about 7.6% less than that of the global Region State Trie index on average; when the query region is in the range of 5-6-byte Geohash encoding and the range of branch region query method is 4 byte, the query time of region query method is about 16.2% less than that of the global Region State Trie index on average; when the query region is in the 6-byte Geohash encoding range and the range of branch region query method is in the 5-byte, the query time of region query method is reduced by about 21% on average compared with that of the global Region State Trie index. It shows that when the range of region query method is consistent, the branch region query method has higher query efficiency than global Region State Trie index; the query range is about the same as the region range, and the branch region query method efficiency is higher. This plays a guiding role in region state query and the region selection of the region node.

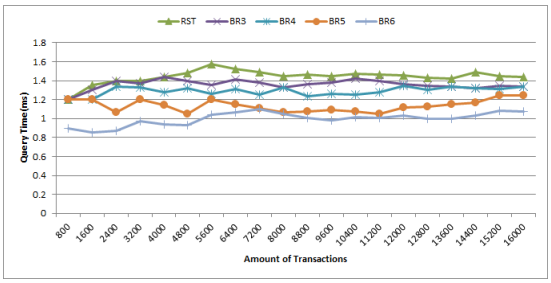


(a) The query range is geographic region of the 3-byte Geohash encoding

(b) The query range is geographic region of the 4-byte Geohash encoding



(c) The query range is geographic region of the 5-byte Geohash encoding



(d) The query range is geographic region of the 6-byte Geohash encoding

Fig. 10. Comparison Chart of Query Time of the Amount of Transactions in Specified Geographic Region of Branch Region Query Method

6 Conclusion

In this paper, our GBRQ structure increases the linkage between the blockchain and the real world by adding geographic location information at the bottom of the blockchain. In order to quickly obtain region state information in IoV, we study the Account Location Trie query method with account as the main body, and global Region State Trie query method with the geographic region as the main body; at the same time, we propose a branch query method to improve the efficiency of region query. We complete the changes to the underlying blockchain structure

on the basis of the open-source Ethereum, instead of using logical structures such as establishing smart contracts on the upper layer, which is more convenient for secondary development and utilization.

The experimental results show that in terms of construction time and the amount of data on-chain, the geographic location blockchain has an average growth of 0.31% and 0.27% compared with the no-location index blockchain, which are basically the same; in terms of the local data amount, due to the increase of the process of constructing the index structure of the geographic location, the data amount of the geographic location blockchain is 3.2 times of that of the no-location index blockchain on average, indicating that the geographic location blockchain uses storage space in exchange for query efficiency. From the perspective of query time, the query efficiency of the Account Location Trie index of the geographic location blockchain is 24 times higher than that of the original no index query; compared with the Account Location Trie index, the query time of the global Region State Trie index is reduced by 99.8%; compared with the global Region State Trie index, the query time of the branch region index is reduced by 21.7%, indicating that the geographic location blockchain has obvious advantages in region state query. In addition, we have further concluded that the more consistent the geographic location range of the node is with the branch query range, the better the query effect is. The effectiveness of our proposed structure and technology is proved by theoretical analysis and experimental results, including security and query time complexity.

However, our research can be further improved. At present, the query of the region state only involves a full-time query, and no further research has been done on the query by period; it does not involve issues such as physical storage and dynamic structure adjustment of blockchain and partitioned consensus. These are the research directions for the subsequent realization of physical multi-chains suitable for IoV.

REFERENCES

[1] *G. Manogaran, V. Saravanan, and C. H. Hsu, "Information-centric content management framework for software defined internet of vehicles towards application specific services,"* IEEE Transactions on Intelligent Transportation Systems, *vol. PP, no. 99, pp. 1–9, 2021.*

[2] *C. Lin, D. He, X. Huang, N. Kumar, and K. Choo, "Bcppa: A blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks,"* IEEE Transactions on Intelligent Transportation Systems, *vol. PP, no. 99, pp. 1–13, 2020.*

[3] *S. R. Maskey, S. Badsha, S. Sengupta, and I. Khalil, "Alicia: Applied intelligence in blockchain based vanet: Accident validation as a case study,"* Information Processing & Management, *vol. 58, no. 3, p. 102508, 2021.*

[4] *T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: Distributed network architecture and performance analysis,"* IEEE Internet of Things Journal, *vol. 6, pp. 4640–4649, 2019.*

[5] *R. Shrestha and S. Y. Nam, "Regional blockchain for vehicular networks to prevent 51% attacks,"* IEEE Access, *vol. 7, pp. 95033–95045, 2019.*

[6] *Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks,"* IEEE Internet of Things Journal, *vol. 6, no. 2, pp. 1495–1505, 2019.*

[7] *J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory,"* IEEE Transactions on Vehicular Technology, *vol. 68, no. 3, pp. 2906–2920, 2019.*

[8] *C. Li, Y. Fu, F. R. Yu, T. H. Luan, and Y. Zhang, "Vehicle position correction: A vehicular blockchain networks-based gps error sharing framework,"* IEEE Transactions on Intelligent Transportation Systems, *vol. 22, no. 2, pp. 898–912, 2021.*

[9] *S. Kudva, S. Badsha, S. Sengupta, I. Khalil, and A. Zomaya, "Towards secure and practical consensus for blockchain based vanet,"* Information Sciences, *vol. 545, pp. 170–187, 2021.*

[10] *G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014.*

[11] *S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.*

[12] *H. Wang, C. Xu, C. Zhang, and J. Xu, "vChain: A blockchain system ensuring query integrity," in* Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, *(Portland, OR, USA), pp. 2693–2696, June 2020.*

[13] *G. Gürsoy, C. M. Brannon, and M. Gerstein, "Using ethereum blockchain to store and query pharmacogenomics data via smart contracts,"* BMC Medical Genomics, *vol. 13, no. 1, 2020.*

[14] *F. A. Qi, A. Dh, B. Sz, C. Mkk, and D. Nk, "A survey on privacy protection in blockchain system,"* Journal of Network and Computer Applications, *vol. 126, pp. 45–58, 2019.*

[15] *C. Chukwuocha, P. Thulasiraman, and R. K. Thulasiram, "Trust and scalable blockchain-based message exchanging scheme on vanet,"* Peer-to-Peer Networking and Applications, *no. 14, p. 3092–3109, 2021.*

[16] *C. Zhou, H. Lu, Y. Xiang, J. Wu, and F. Wang, "Geohashtile: Vector geographic data display method based on geohash,"* International Journal of Geo-Information, *vol. 9, no. 7, p. 418, 2020.*

[17] *"Geohash." Website.*

[18] *J. Zhang, C. Yang, Q. Yang, Y. Lin, and Y. Zhang, "Hgeohashbase: an optimized storage model of spatial objects for location-based services,"* Frontiers of Computer Science in China, *pp. 1–11, 2018.*

[19] *K. Huang, G. Li, and J. Wang, "Rapid retrieval strategy for massive remote sensing metadata based on geohash coding,"* Remote Sensing Letters, *vol. 10, no. 11, pp. 1070–1078, 2018.*

[20] *I. Blockchain, "Ibm blockchain solutions," 2021.*

[21] *B. GmbH, "Bigchaindb 2.0: the blockchain database," 2018.*

[22] *Oracle, "Oracle blockchain platform," 2021.*

[23] *A. Schuster, "Sap hana blockchain – a technical introduction," 2018.*

[24] *Fluree, "Blockchain & fluree's ledger," 2021.*

[25] *S. Team, "Swarm writerpaper," 2021.*

[26] *PeerSafe, "chainsql blockchain write paper v3.0(in chinese)," 2020.*

[27] *J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in* NSDI, *2019.*

[28] *M. Szydlo, "Merkle tree traversal in log space and time," in* Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings, *2004.*