electronics

Article

CBCS: A Scalable Consortium Blockchain Architecture Based on World State Collaborative Storage

Jiashun Zhou [1,2], Na Wang [1,2,*], Aodi Liu [1,2], Wenjuan Wang [1,2] and Xuehui Du [1,2]

[1] School of Cryptographic Engineering, Information Engineering University, Zhengzhou 450001, China
[2] He'nan Province Key Laboratory of Information Security, Zhengzhou 450001, China
[*] Correspondence: twftina_w@126.com

Abstract: In the big data environment, data are characterized by a large volume, various types, and rapid changes. The consortium blockchain applied in this environment faces the problem of excessive storage of the ledger, and the ledger handling different types of business needs to be isolated to ensure the ledger's security. To this end, this paper proposes a scalable consortium blockchain architecture based on world state collaborative storage (CBCS). First, a business world state database update method is designed based on sparse Merkle multiproofs, where the collaborative storage of world state is realized under the premise of mutual isolation of the ledger between business domains. Then, a world state consistency verification method based on the rank B+ tree is designed to verify the consistency of the business world state in business domains by the checking sidechain, and a main-side chain cross-anchoring structure is designed to realize secure anchoring of the mainchain and the checking sidechain. Meanwhile, a blockchain transaction trusted tracing method based on two-level certification is developed to enable business nodes to obtain complete blockchain transactions. Finally, the feasibility and efficiency of the proposed mechanism to solve the storage scalability problem in the consortium blockchain are verified through experiments.

Keywords: consortium blockchain; storage scalability; world state; rank B+ tree; checking sidechain

Citation: Zhou, J.; Wang, N.; Liu, A.; Wang, W.; Du, X. CBCS: A Scalable Consortium Blockchain Architecture Based on World State Collaborative Storage. *Electronics* **2023**, *12*, 735. https://doi.org/10.3390/electronics12030735

Academic Editor: Juan M. Corchado

Received: 19 December 2022
Revised: 23 January 2023
Accepted: 31 January 2023
Published: 1 February 2023

1. Introduction

The blockchain [1] ensures high redundancy of ledger data through distributed storage of globally consistent blockchain ledger data by multiple nodes, thus avoiding single-point failures caused by corruption of the ledger data stored by an individual node. Though the redundant storage of the ledger improves the security of the blockchain system, it increases the pressure on the ledger storage of the blockchain nodes. As of October 2022, the ledger size of Bitcoin [2] is about 405 GB and is growing at a rate of about 50 GB per year. The large size and rapid growth of the blockchain ledger bring about serious blockchain storage scalability problems [3], which limit the application of blockchain technology [4].

Consortium blockchain [5] is a permissioned blockchain, which provides strict access control to the ledger data and has a faster throughput than the public blockchain [6], enabling fast consensus and up-chaining of blockchain transactions. Currently, consortium blockchain technology is widely used in practical scenarios, such as data circulation and sharing [7], data right confirmation [8], and data traceability [9]. However, due to the massiveness, heterogeneity, and frequent up-chaining demand of various business data in the actual business scenarios, consortium blockchain also faces serious storage scalability problems.

The current research on the scalability of blockchain storage mainly focuses on the processing of blocks in the chained ledger but ignores the critical role of the state database. The chained ledger ensures the security of a block by connecting it with its front and back blocks. Since the chained ledger is stored in the form of files and direct access to the blockchain transactions in it requires searching the block files one by one, which is inefficient. The world state database contains critical data extracted from the chained ledger and stores the latest values of the world state in the form of key-value pairs. Owing to its characteristics of high value, a small storage space occupation, high query efficiency, and fine-grained slicing, the world state database occupies an important position in the consortium blockchain system. In practice, we can retrieve the required ledger data by searching the world state database, thus satisfying most of the application requirements in the consortium blockchain system. Chen et al. [10] proposed a high-performance consortium blockchain storage architecture for a big data environment. In this architecture, the consortium blockchain ledger is divided into continuous data and state data, an index-based method and a multi-level cache method are designed to process continuous data and state data, respectively. However, this scheme focuses on improving the read–write performance of the ledger but does not address the scalability issue of the consortium blockchain ledger storage.

Meanwhile, in existing application scenarios, a consortium blockchain system often has the need to process blockchain transactions of multiple business scenarios simultaneously, in which the ledger data need to be isolated to enhance ledger security. Hyperledger fabric [11], an open-source platform for consortium blockchain, uses channel technology to realize the processing in different scenarios. In each business scenario, its ledger is processed and recorded through a separate channel. However, the organizations joining multiple channels need to store the ledger of multiple channels in the blockchain nodes, which increases the pressure on the blockchain nodes to store the ledger.

This paper focuses on the world state data in the consortium blockchain ledger. According to different business types, the complete world state database is divided into different business world state databases. Then, the consortium blockchain nodes of different business departments process different business world state to realize collaborative storage of the world state database. In this way, the storage space of the consortium blockchain node can be efficiently saved, and the ledger data in different business scenarios can be isolated safely. Collaborative storage is a common approach for solving the storage scalability problems of blockchain [12]. However, the existing studies mostly use random or sequential order of transactions to divide the chained ledger into different pieces and store them in different blockchain nodes, which increase the communication and time costs of acquiring the local unstored ledger data. The world state data are in the form of a single key-value pair, which has a smaller granularity than the chained ledger. By dividing the world state database according to different business scenarios, the consortium blockchain node can store the world state data most frequently used in its business, thus effectively improving the efficiency of querying ledger data.

In addition, this paper fully considers the security of the world state data. The world state in the consortium blockchain can improve the accessing speed of the ledger data and serve as a basis for determining the validity of the read–write sets of blockchain transactions. Moreover, it can update and read the world state through write operation and read operation, respectively, but there are big differences in the execution process of the two kinds of operations. When a blockchain transaction is initiated to perform a write operation to the ledger, an endorsement policy can be set up to stipulate that the blockchain transaction can only write new data to the blockchain ledger after it has been verified by enough organizations. However, when a blockchain transaction is initiated to read the world state, the endorsement of multiple organizations is not required. The blockchain node obtains the corresponding state data through the locally stored world state database and returns it to the applicant. After completing the write operation, the world state database has a high independence degree, which is less secure than the chained ledger stored through the chained structure, especially for those scenarios where data management is not strict or other security issues exist. If the data stored in the world state database are corrupted or maliciously modified, the operation of reading the world state has the risk of returning non-trusted data [13]. In our study, the consistency of the state data is periodically verified to ensure the security of the business world state databases stored by the consortium blockchain nodes.

To sum up, this paper comprehensively considers the application scenarios of consortium blockchain and proposes a scalable consortium blockchain architecture based on world state collaborative storage (CBCS) to address the excessive pressure of ledger storage and the isolation of ledger data in different business scenarios. Meanwhile, the security of ledger data storage is fully considered. The contributions of this paper are as follows.

(1) A business world state database update method is designed based on sparse Merkle multiproofs, where collaborative storage of world state is realized under the premise of mutual isolation of ledger between business domains.
(2) A world state consistency verification method based on the rank B+ tree is designed to verify the consistency of the business world state in business domain by the checking sidechain, and a main-side chain cross-anchoring structure is designed to realize secure anchoring of the mainchain and the checking sidechain.
(3) Considering the demand that business nodes may need complete blockchain transactions, the main-side chain cross-anchoring structure is used to design the blockchain transaction trusted tracing method based on two-level certification to support business nodes' access to blockchain transactions.
(4) The feasibility and efficiency of the proposed architecture to address the storage scalability issue of consortium blockchain is verified through experiments.

2. Related Work

At present, many scholars have researched the blockchain storage scalability problem from different perspectives for different application scenarios, and the methods can be classified into the off-chain storage-based method and the on-chain storage-based method.

The off-chain storage-based method transfers the storage pressure of the ledger to off-chain storage resources, such as cloud storage [14], distributed storage system IPFS (InterPlanetary File System) [15], etc. On-chain nodes store only the metadata of the ledger (including off-chain storage addresses, hash values of transaction data, etc.) to reduce storage pressure. Zheng et al. [16] proposed to store blockchain transactions generated by miner nodes in the IPFS network, and pack the storage addresses of transactions returned by the IPFS network instead of the original blockchain transactions into blocks for storage, using the feature that the storage address occupies less storage space to save the on-chain storage space. He et al. [17] provided the Chameleon architecture. In this architecture, the complete ledger data are stored on the cloud storage system, and the blockchain nodes only store the ledger of the most recent period. Xu et al. [18] proposed SlimChain, where the off-chain nodes store some or all ledger data according to their storage capacity and verify the validity of transactions. The on-chain nodes only store part of the block header data such as the transaction hash and the previous block hash, and based on this, the validity of the ledger data stored by the off-chain nodes can be verified. The above methods of transferring the ledger to off-chain storage space can dynamically adjust the demand for off-chain storage space with the change in ledger volume. However, these methods reduce the decentralization of the blockchain system, and it is necessary to ensure the off-chain ledger storage security to prevent the ledger from being destroyed.

In the on-chain storage-based method, all the ledger data are stored in the blockchain nodes. The on-chain storage-based method can be further divided into the light node-based method and the collaborative storage-based method.

The light node-based method is widely used in cryptocurrency wallets [19] or Internet of Things (IoT) [20] scenarios to solve the problem that the weak storage capacity of terminal nodes is not suitable for storing larger-scale blockchain ledger. For blockchain systems using the unspent transaction outputs (UTXO) model [21], Palai et al. [22] proposed the block summarization method, which merges the inputs and outputs of transactions in adjacent blocks to generate the summarized blocks. Then, these blocks are stored on resource-constrained light nodes, enabling the light nodes to verify blockchain transactions independently. However, this method can only be applied to blockchain systems using the UTXO model, and it is not suitable for blockchain systems that deploy smart contracts. In blockchain-based IoT applications, IoT nodes often have limited storage resources. To address this issue, Kim et al. [23] proposed the storage compression consensus (SCC) algorithm, which improves the practical Byzantine fault tolerance (PBFT) consensus algorithm [24] to increase the block compression process. In this algorithm, the Merkle tree is constructed by calculating the hash of the existing blocks, and the root node of the Merkle tree is recorded in the compressed blocks. After the end of the consensus cycle, the IoT nodes with limited storage resources store the compressed blocks and the latest blocks, and delete the blocks stored in the previous cycle. In this method, the IoT nodes need to rely on the full node to extract the blockchain transactions in the compressed blocks, which reduces the decentralization of the blockchain system and increases time consumption in requesting blockchain transactions. Liu et al. [25] designed an irrelevant block filtering method to filter the blocks that cannot be invoked by subsequent blockchain transactions according to the actual application scenario of industrial IoT. The dynamic storage nodes only store the useful blocks that can be invoked subsequently. However, this method is closely related to the actual application scenario and is not suitable for the scenario where all existing blocks are likely to be invoked by subsequent transactions.

The ledger data can be processed by encoding or slicing to realize collaborative storage of ledger data. Qi et al. [26] combined erasure coding with the Byzantine fault tolerance (BFT) consensus protocol to propose the BFT-store scheme, which divides and encodes the original block into multiple smaller block slices, and the block slices are stored separately on blockchain nodes to reduce the redundancy of blocks in the system to $O(1)$. Guo et al. [27] proposed to encode blockchain transactions using residue number system (RNS) [28], where blockchain nodes can dynamically adjust the compression ratio according to their storage capacity to accommodate the growth of the ledger data. In the methods of using encoding technology to segment ledger data, the node needs to send a request of obtaining slices to other nodes and recode them to obtain the required ledger data. The process of slicing, merging, and recoding will result in more time consumption. Chen et al. [29] proposed to recode the fields such as the version and hash of blockchain transactions in blocks to reduce the storage space. However, the ledger data storage space that can be saved by this method is limited. Xu et al. [30] proposed to divide the blockchain nodes into different consensus units (CU). The nodes in the CU store some blocks in the chained ledger according to the frequency of the blocks being used, and the block headers of all blocks. More frequently used blocks have more copies in the CU, and at least one complete copy of the blockchain ledger is stored in a CU. However, the method needs to record and count the frequency of each block being used, thus increasing the computation and storage overhead. In addition, as new blocks are generated, the number of copies of blocks in the CU gradually decreases, and the time consumption in the CU for block retrieval gradually increases. Chen et al. [31] divided blocks into hot blocks and cold blocks according to the frequency of blocks being used, and constructed a cooperative storage network to store hot blocks based on the Chord protocol [32] and stored cold blocks through the IPFS network. Xie et al. [33] adopted the Kademlia protocol [34] to assign a unique node ID to each blockchain node and divided the nodes into different clusters, each of which stores a complete copy of the ledger. Meanwhile, they proposed a dynamic reorganization mechanism to dynamically adjust the cluster size according to the storage capacity of the nodes in the cluster.

3. Preliminaries

3.1. Consortium Blockchain

Consortium blockchain is a kind of blockchain technology specially developed for enterprises or other groups to realize trusted distributed data processing. Unlike Bitcoin, which is oriented to the field of cryptocurrency, there is no need for miners to mine in the consortium blockchain, and no cryptocurrency is generated. The consortium blockchain achieves consensus through consensus algorithms such as PBFT, which run by voting, consume less energy and computing resources, and can reduce the delay of consensus.

Consortium blockchain consists of different organizations. Each organization is a group of different types of nodes. The consortium blockchain carries out strict authentication and permission management on nodes, and the ledger data can only be accessed by nodes in the organization, thus isolating ledger data from external nodes. The submitting nodes in the consortium blockchain initiate blockchain transactions through chaincodes, and the blockchain transactions are formed into blocks through the ordering service after endorsement. These blocks are linked back and forth to form a chained ledger. Meanwhile, the world state is generated and stored as a key-value pair (*key*, *value*). The *key* identifies the world state, and the *value* records the version *rev* of the world state and other data related to specific business scenario. The chaincode of the consortium blockchain contains several smart contracts [35], which enable the creation, query, and update of the world state. When the world state is updated, the *key* is kept unchanged, and the version *rev* and other data related to the specific business scenario are updated.

The ledger stored by the consortium blockchain nodes includes a history database as well as an index database in addition to the chained ledger and the world state database. The history database stores the information of blockchain transactions that update the world state data database. The index database holds the locations of the blocks in the file where the chained ledger is stored.

3.2. B+ Tree

The B+ tree [36] is a multiplexed balanced lookup tree for indexing and retrieving data in file systems or databases. As shown in Figure 1, the leaf nodes in a B+ tree store multiple data and sort the data in increasing order of keywords. The incremental ordering approach ensures the determinism of data structure, and facilitates the construction of an entirely consistent data structure in distributed data systems. The non-leaf nodes store the ranges of data in the leaf nodes and the pointers to the leaf nodes. The range of data stored in non-leaf nodes can be used to quickly locate the data stored in the leaf nodes, enabling efficient data retrieval.

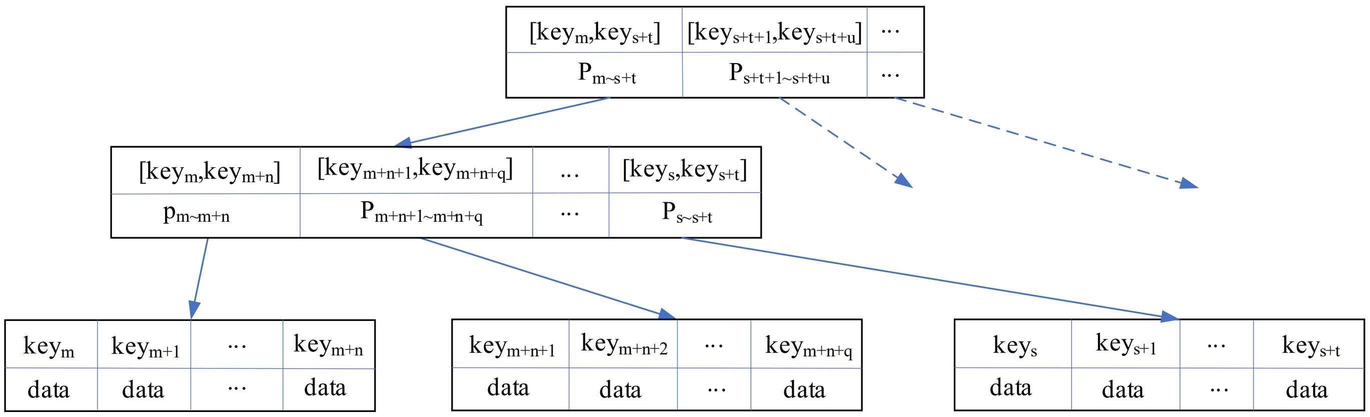

Figure 1. B+ tree structure.

4. Execution Framework

As shown in Figure 2, this paper designs world state collaborative storage method to alleviate the ledger storage pressure of business nodes. The master nodes in the system perform periodic consensus on the consistency information of each business world state database and the mainchain cycle interval summary, and the information is stored by the checking sidechain. The business node obtains the consistency information in the sidechain block and compares it with its consistency information to ensure the consistency of the business world state database stored by it. To obtain complete blockchain transactions, business nodes can send transaction acquisition request to the master node and verify the authenticity of transactions by using the mainchain cycle interval summary recorded in the block headers of the sidechain block.

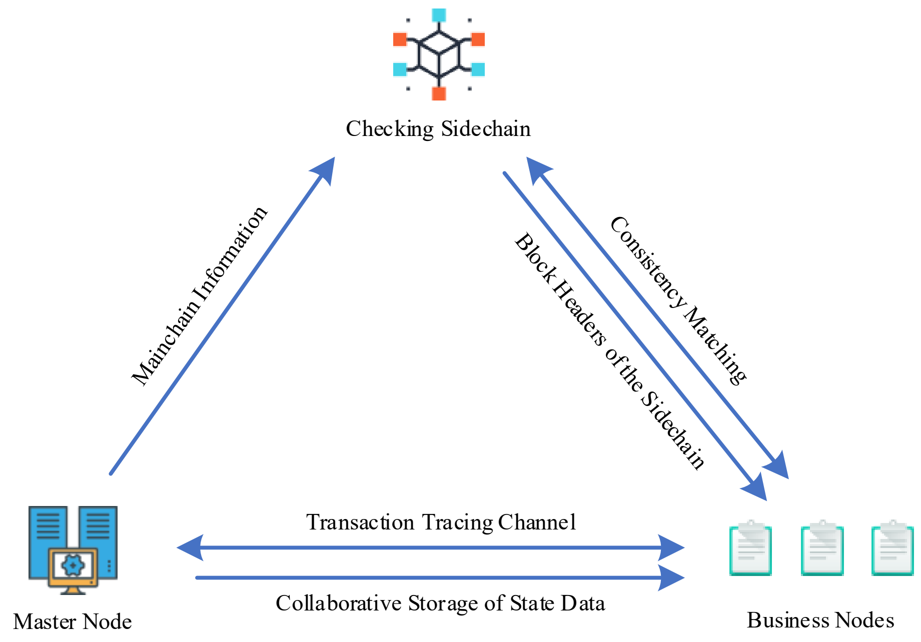
Checking Sidechain
Mainchain Information
Consistency Matching
Block Headers of the Sidechain
Transaction Tracing Channel
Collaborative Storage of State Data
Master Node
Business Nodes

Figure 2. System overview.

4.1. World State Collaborative Storage Method

4.1.1. Collaborative Storage Architecture

This study divides the participants in the consortium blockchain organization into an organization manager and business departments. Taking the data sharing scenario as an example, the shared data can be divided into different business types such as video data, text data, audio data, and image data. The consortium blockchain organization can set up its business departments for each data type. The organization manager constructs the master node of the organization, and the business departments make their respective business nodes. Similarly, the study can also be applied to other fields such as medical treatment and banking. This can solve the problem of excessive storage pressure of ledger data and meet the need of ledger data isolation between different business departments.

In this case, the master node and business nodes use different ledger storage strategies. To guarantee that the ledger is fully available in the organization, the master node stores all ledger of consortium blockchain, including the chained ledger, the world state database, the history database, the index database and the checking sidechain proposed in our scheme. However, the master node does not increase the management privileges to the business nodes, and the business nodes still operate in a decentralized manner. A master node is set up in each organization, and all master nodes form the master domain. The business nodes in the same organization are responsible for different businesses, and the business nodes responsible for the same business in different organizations form a business domain. The business nodes in the same business domain handle the same business based on the same business chaincode and maintain a consistent business world state database. Meanwhile, the business node stores the business world state database in the business domain to which it belongs, and the block headers of the checking sidechain. The business world state database is a class of data in the complete world state database for handling the same business, and the business world state database in a business domain is generated by the same chaincode.

The world state collaborative storage architecture is shown in Figure 3. There are three business domains {*Busi_dom_a*, *Busi_dom_b*, *Busi_dom_c*}, with three business nodes in each business domain. These three business nodes belong to organizations {*Org1*, *Org2*, *Org3*}, respectively. Specifically, the three business nodes in *Busi_dom_a* process the business *busi_a* based on business chaincode *busi_code_a*, and store the business world state database *busi_base_a*. The complete world state data in the system is stored by collaborative storage of business world state databases in different business domains, which can effectively reduce the pressure on the ledger storage of business nodes and help the consortium blockchain system to process more businesses. The business nodes can only access the business world state database in their respective business domains, thus realizing controlled access of business nodes to consortium blockchain ledger data.

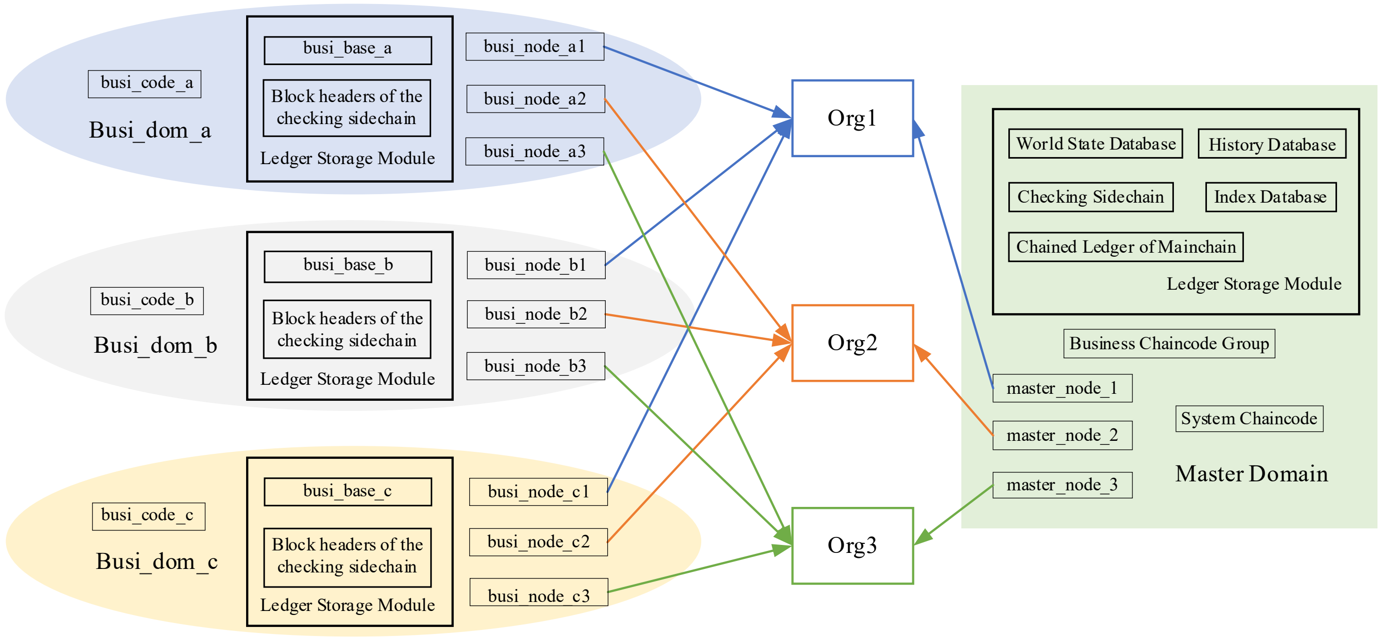

Figure 3. The collaborative storage architecture.

4.1.2. Business World State Database Update Method Based on Sparse Merkle Multiproofs

The master node updates the stored world state database based on the blockchain transactions in the complete block. The complete block contains a block header and a block body. The block header of the complete block stores the block body hash *block_i_body_hash*, the block header hash *block_(i−1)_header_hash* of the previous block, and the block number *blockNum*. The block body of the complete block stores the blockchain transactions in order of achieving consensus. In this study, the Merkle tree is built for the blockchain transactions in the complete block, and the hash recorded by the root node of the Merkle tree is used as *block_i_body_hash*.

The complete block contains blockchain transactions of different business domains. To prevent business nodes from accessing transactions of other business domains, business nodes cannot directly access the complete blocks to update the business world state databases. In this regard, this paper proposes the concept of the business block. The business node receives only business blocks related to its business and uses the transactions to update its stored business world state database based on the authenticity verification of blockchain transactions in the business blocks.

The ordering service [37] is a technique to ensure that a consistent order of blockchain transactions is obtained in the consortium blockchain system. It sorts the blockchain transactions of multiple concurrent points and obtains the globally recognized transaction serial number in the block. Meanwhile, it groups blockchain transactions according to the parameter in them, and distributes the grouped transactions to specific consortium blockchain nodes.

In our scheme, businesses in the same business domain are handled with the same chaincode, and the name of the chaincode used to generate the blockchain transaction is recorded in the blockchain transaction. The process of business block generation in our scheme is presented in Figure 4. As shown in this figure, the ordering service filters the blockchain transactions in the complete block to obtain the business block of each business domain based on the business chaincodes. The block body of the business block contains the blockchain transactions generated using the same business chaincode, as well as the serial number of each blockchain transaction in the complete block.

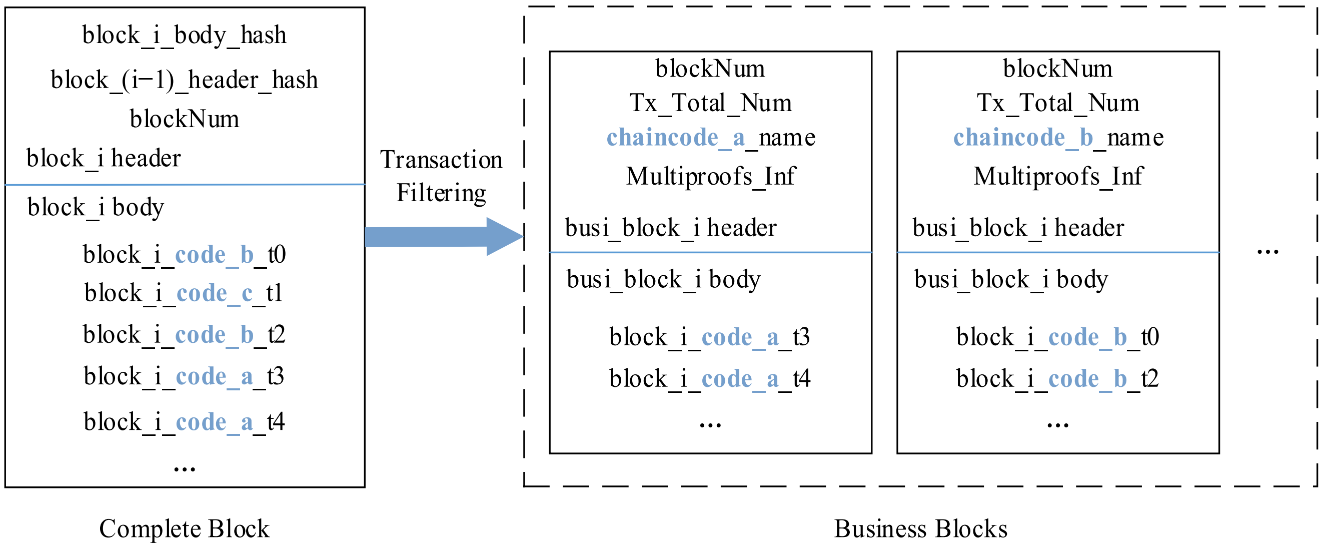

Figure 4. Business block generation.

Since the transaction data in the business block is a part of the transaction data in the complete block, cannot directly use the Merkle root hash in the block header of the complete block to verify the authenticity of the transaction data in the business block. Merkle proof [38] is required to verify transactions one by one, which will lead to large communication and time costs. To this end, this paper uses sparse Merkle multiproofs [39] to merge the Merkle proof paths, thus simultaneously verifying the authenticity of all transactions in the business block. In this study, the block header of a business block contains the block number *blockNum*, the total number *Tx_Total_Num* of the blockchain transactions in the corresponding complete block, the business chaincode name *chaincode_name*, and the multiproofs auxiliary information *Multiproofs_Inf* that helps to prove the existence of blockchain transactions in the business block. The block number of a business block is the same as that of the corresponding complete block.

- Multiproofs Auxiliary Information Generation;

In our scheme, Merkle tree nodes are recorded as key-value pairs {*merkle_key*, *merkle_value*}. The key *merkle_key* is graded and labeled as N_j^i, where the superscript i indicates the layer where the node is located in the Merkle tree; i = 0 indicates the node is obtained by calculating the blockchain transaction hash; $i > 0$ indicates that the node is obtained by merging its two child nodes, $i = Root$ indicates that the node is the Merkle tree root node; a larger value of i indicates that the node is closer to the Merkle tree root node; the subscript j indicates the node's serial number in layer i. The subscript of each Merkle tree node increases from left to right. The value *merkle_value* indicates the hash recorded at this Merkle tree node.

The non-leaf nodes of the Merkle tree have both left and right child nodes. If two Merkle proof paths pass through a Merkle tree node from the left and right sub-paths of the node, respectively, then the Merkle tree node can be obtained by calculating its left and right child nodes. This node is directly obtainable and does not need to be calculated using auxiliary information.

Figure 5 shows the schematic diagram of generating multiproofs auxiliary information for a business block. In this figure, *t0*~*t7* denote the transactions in the complete block, where {*t0*,*t2*,*t3*,*t6*} are the transactions in a business block. It needs to verify that {*t0*,*t2*,*t3*,*t6*} is presented in the complete block. The Merkle proof path for the blockchain transaction *t2* is $\{N_2^0, N_1^1, N_0^2, N_0^{Root}\}$, and the multiproofs auxiliary information $\{N_3^0, N_0^1, N_1^2\}$ is required. When performing Merkle proof on blockchain transaction *t3*, the Merkle proof path is $\{N_3^0, N_1^1, N_0^2, N_0^{Root}\}$, and the multiproofs auxiliary information $\{N_2^0, N_0^1, N_1^2\}$ is required. The two proof paths pass through the Merkle tree node N_1^1 from its left and right sub-paths, respectively, N_1^1 can be obtained using the existing nodes N_2^0 and N_3^0 in the two proof paths. Then the multiproofs auxiliary information to prove the existence of blockchain transactions *t2* and *t3* is $\{N_0^1, N_1^2\}$. Similarly, the multiproofs auxiliary information to prove the existence of blockchain transactions {*t0*,*t2*,*t3*,*t6*} is $\{N_1^0, N_7^0, N_2^1\}$.

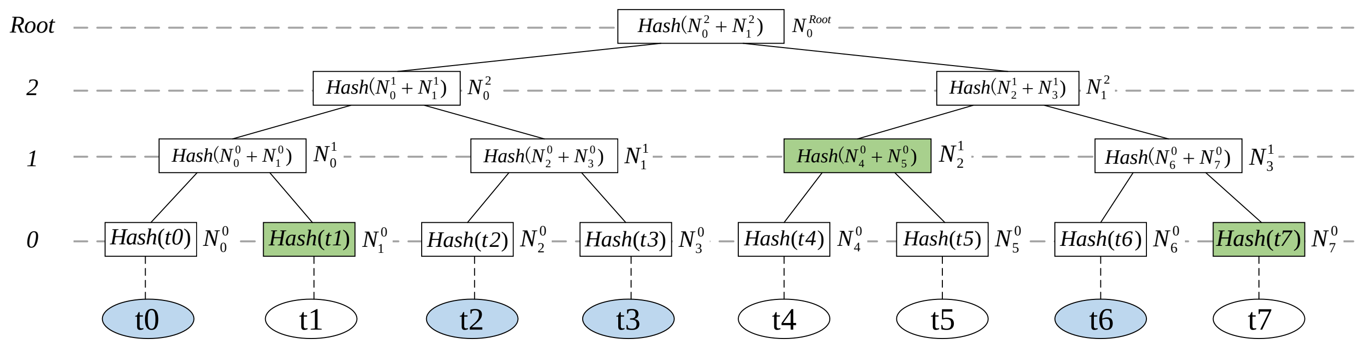

Figure 5. The schematic diagram of sparse Merkle multiproofs for business block.

- Block Broadcasting;

The master node of each organization is connected with the ordering service to obtain the complete block generated by the ordering service, and broadcasts the block header of the complete block and the amount list $Amount_{list}$ for the business nodes contained in each organization. $Amount_{list}$ records the number of blockchain transactions in each business block. Each business domain elects business docking node to connect with the ordering service, obtain the business blocks of their respective business domains, and broadcast them in their respective business domains.

- Business World State Database Update.

After the business node obtains the business block, the block header of the complete block, and $Amount_{list}$, the authenticity of the blockchain transactions in the business block need to be verified. First, it is verified whether the block number recorded in the business block header matches the block number recorded in the complete block header to ensure that the business block matches the complete block. Then, it is verified whether the number of entries recorded in $Amount_{list}$ is consistent with the actual number of blockchain transactions contained in the business block. Finally, a layer-by-layer verification approach is designed, and the multiproofs auxiliary information is used to prove the existence of the blockchain transactions obtained from the business block in the complete block.

The hash of the blockchain transactions in the business block is calculated as *merkle_value* of the node at the bottom of the Merkle tree, and *merkle_key* corresponding to each *merkle_value* is obtained, where the superscript *i* is 0, the subscript *j* is the serial number of each blockchain transaction in the complete block. The obtained node is recorded at *Merkle_level0*, i.e., the 0th level of the Merkle tree.

When the subscript *j* of *merkle_key* is even, it indicates that the node is connected to the upper-level node through the left sub-path, and when the subscript is odd, it indicates the node is connected to the upper-level node through the right sub-path. The nodes in *Merkle_level0* are analyzed one by one in the increasing order of the subscript of *merkle_key*. When the subscript *j* of *merkle_key* is even, it is checked whether there is a node with subscript *j* + 1 in *Merkle_level0*, and if not, the node is obtained from the multiproofs auxiliary information. If the subscript *j* of *merkle_key* is odd, the node with subscript $j-1$ is obtained directly in the multiproofs auxiliary information. Then, the two obtained left and right child nodes are used to calculate the node at *Merkle_level1*. The obtained node has a superscript of 1 and a subscript of 1/2 of the subscript of the left child node. Subsequently, the nodes in *Merkle_level1* are processed in the same way as *Merkle_level0* to obtain the Merkle tree nodes in the higher level *Merkle_level2*, and until the root node of the Merkle tree is obtained. At this time, the hash recorded in this root node is compared with the block body hash recorded in the block header of the complete block. If consistent, it is proved that the blockchain transactions in the business block all originate from the corresponding complete block. The business node verifies the read–write sets of the blockchain transactions in the business blocks, and the write sets are used to update the business world state database after the validation is passed. After the business world state database is successfully updated, the business node discards business block, the block header of the complete block, and $Amount_{list}$.

The blockchain transactions to be proved in Figure 5 are {*t0*,*t2*,*t3*,*t6*}, then *Merkle_level0* = $\left\{N_0^0, N_2^0, N_3^0, N_6^0\right\}$, and the multiproofs auxiliary information recorded in the block header of the business block is $\left\{N_1^0, N_7^0, N_2^1\right\}$. The nodes in *Merkle_level0* are analyzed in an increasing order of the subscript of *merkle_key*. N_0^0 with an even subscript 0 is first looked up in *Merkle_level0* to check if there is a node with an odd subscript 1, and if it is found that the node does not exist in *Merkle_level0*, then N_1^0 is obtained in the multiproofs auxiliary information. The upper-level node N_0^1 is calculated by using N_0^0 and N_1^0, and the node N_0^1 is recorded with *Merkle_level1* = $\left\{N_0^1\right\}$. Afterward, the node N_2^0 in *Merkle_level0* is analyzed, and the subscript of N_2^0 is even 2. Then, it is first checked whether there is a node in *Merkle_level0* whose subscript of *merkle_key* is odd 3. From *Merkle_level0*, the node N_3^0 can be found, and N_2^0 and N_3^0 are used to calculate the upper-level node N_1^1, and the node N_1^1 is recorded in *Merkle_level1*, *Merkle_level1* = $\left\{N_0^1, N_1^1\right\}$. Subsequently, the node N_6^0 is analyzed, and the node N_3^1 is recorded in *Merkle_level1*, *Merkle_level1* = $\left\{N_0^1, N_1^1, N_3^1\right\}$. Next, *Merkle_level2* is obtained by analyzing *Merkle_level1* in the same way. This process is continued until the root node of the Merkle tree is obtained. Based on this, the authenticity of the business block can be verified, and the business world state database is updated.

4.2. World State Consistency Verification Method Based on Rank B+ Tree

To reduce the ledger storage pressure of the business nodes, the business nodes in our scheme do not store a chained ledger. As a result, the business nodes cannot check the consistency of the business world state database by reconstructing based on the chained ledger. To address this issue, this paper proposes a world state consistency verification method based on the rank B+ tree. Firstly, the B+ tree is improved to construct the rank B+ tree. Then, the rank B+ tree is employed to obtain the consistency information of the business world state database, and this information is used to initiate checking transactions. Based on this, the information between the mainchain and the checking sidechain is matched through the main-side chain cross-anchoring structure. Meanwhile, the business nodes verify the consistency of their stored business world state database by obtaining the consistency information in the checking block.

4.2.1. Rank B+ Tree

The B+ tree incrementally sorts the keywords of the data to ensure the determinism of the data structure. However, when the data are corrupted, the B+ tree cannot be used to obtain a basis for determining whether the data structure is consistent, and the specific location of the data corruption cannot be discovered. For this, this paper improves the B+ tree by proposing the rank B+ tree, as shown in Figure 6. The rank layer is constructed based on the original B+ tree structure to group the data managed by the B+ tree, and the hash of the sorted data group is calculated as the consistency information of the group. Then, the consistency information of all the data managed by the rank B+ tree can be obtained through the consistency information layer. There are four layers in the rank B+

tree, and from top to bottom, they are the consistency information layer, the rank layer, the index layer, and the data layer.

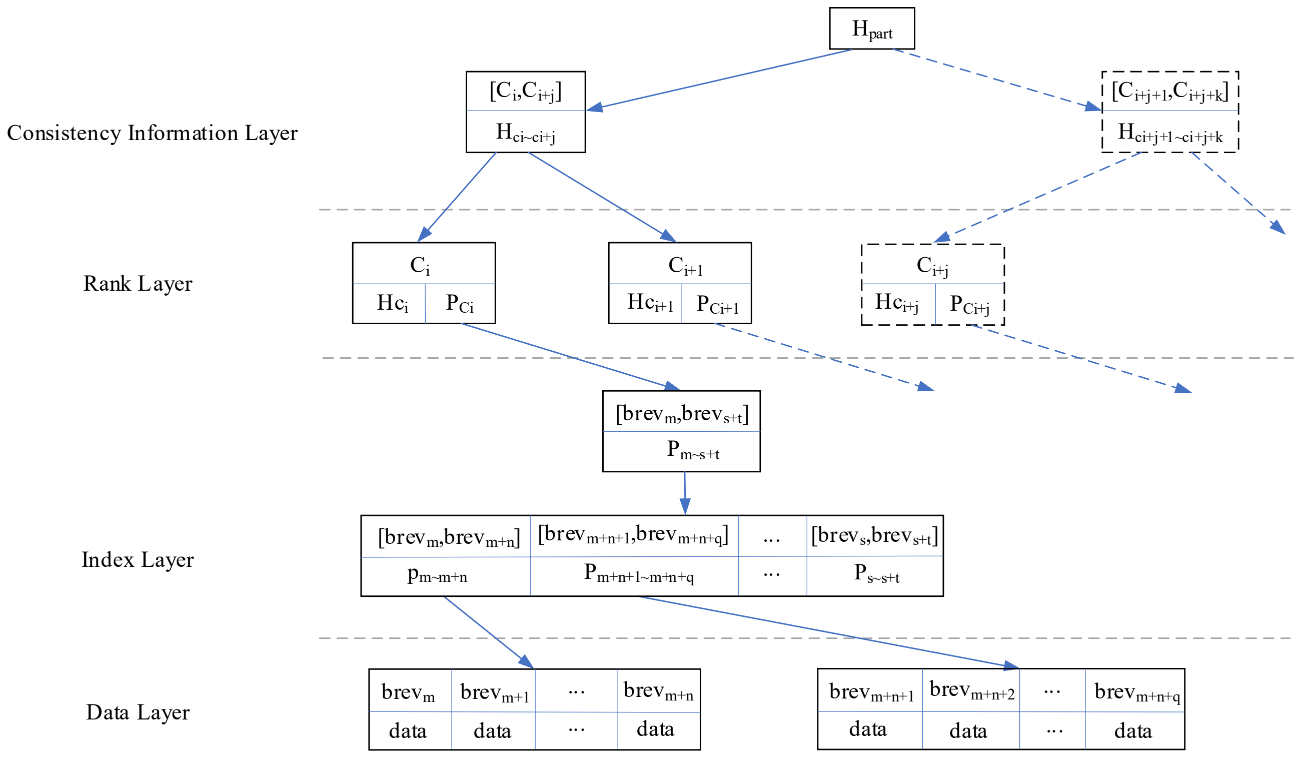

Figure 6. The schematic diagram of the rank B+ tree.

The data layer stores the data nodes that are represented in the form of key-value pairs, denoted as $\{brev_m, data\}$, where $brev_m$ and *data* denote the key and the content of the data node, respectively. Each data node corresponds to a rank identifier C_i, and is classified according to C_i. Different ranks of data nodes are managed by different rank nodes. Each rank node indexes the data nodes of the rank through a multinomial tree, where the interval $[brev_m, brev_{m+n}]$ of $brev_m$ and the pointer $P_{m\sim m+n}$ to the location of the data node interval are stored. The data nodes in the interval are sorted in increasing order of $brev_m$, the data node with the smallest $brev_m$ at the leftmost end and the data node with the largest $brev_m$ at the rightmost end. The adjacent intervals are also stored in increasing order of $brev_m$, the $brev_m$ of data nodes in the left interval are smaller than those in the right interval. The hash H_{ci} of the sorted data nodes managed by each rank node is calculated as the data consistency information of that rank, i.e., H_{ci} = hash($sortedD_{ci}$), where $sortedD_{ci}$ represents the sorted data in the data layer managed by the rank node. The rank node stores its rank identifier C_i, the data consistency information H_{ci}, and a pointer to the root node of a multinomial tree in the index layer.

The incremental sorting method is also used in the consistency information layer to sort the rank node identifiers C_i, and the consistency information layer is calculated by merging the hashes of the rank nodes through a multinomial tree, i.e., $H_{ci\sim ci+j} = \text{hash}(H_{ci}||H_{ci+1}||H_{ci+2}||\ldots||H_{ci+j})$. The consistency information nodes are identified by $[C_i,C_{i+j}]$, which indicates the range of the rank nodes identified by this consistency information node. The consistency information nodes are similarly merged by the hash merge method, and finally, the root node H_{part} of this consistency information layer is obtained.

4.2.2. Main-Side Chain Cross-Anchoring Structure

To avoid affecting the generation of blockchain transactions in the mainchain, the consistency of business world state data cannot be verified by initiating blockchain transactions in the mainchain. To address this issue, this paper sets the checking sidechain and then initiates checking transactions in the checking sidechain to achieve consensus on the consistency information of the business world state database. Meanwhile, the main-side chain cross-anchoring structure is proposed, which uses a two-way recording of the mainchain and the checking sidechain to achieve information matching and improve the security of the checking sidechain.

The main-side chain cross-anchoring structure contains a mainchain and a checking sidechain. Specifically, the mainchain is the original chain in the consortium blockchain system. The checking sidechain uses the genesis block of the mainchain as the starting block. The generation of checking blocks is determined by the block height of the mainchain. Let the checking height threshold be *c*. When *c* blocks are generated in the mainchain, a consistency checking transaction is initiated to perform consensus on the consistency of the business world state database in each business domain. The checking block is generated through the ordering service.

The schematic of the main-side chain cross-anchoring structure is shown in Figure 7, where there are two mainchain cycles [*block_1*,*block_c*] and [*block_c* + *1*,*block_2c*]. When the block height of the mainchain reaches *c*, a checking transaction is initiated, and the first checking block *cblock_1* is generated; when the block height of the mainchain reaches 2*c*, the checking sidechain generates the second checking block *cblock_2*. The block in the mainchain with a block height of *nc* is called the mainchain cycle point block, the range between adjacent mainchain cycle point blocks is called the mainchain cycle interval, and the multiplicity *n* is called the checkpoint number *CheckNum*. In Figure 7, *cycle_1* and *cycle_2* are the mainchain cycle intervals corresponding to checking blocks *cblock_1* and *cblock_2*, respectively.

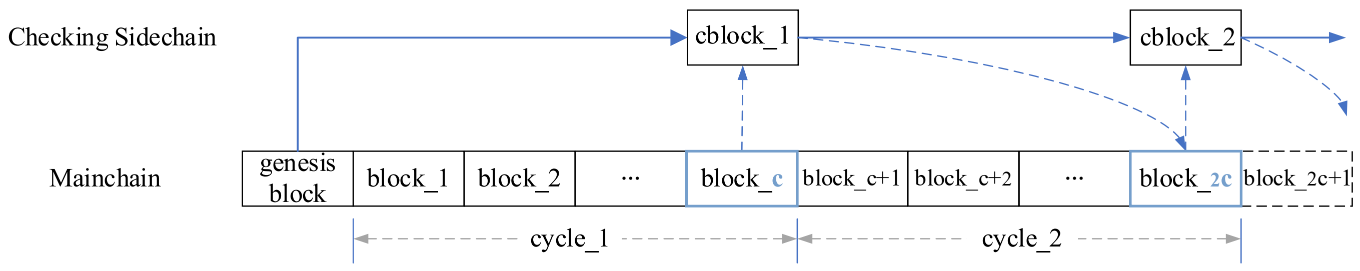

Figure 7. The main-side chain cross-anchoring structure.

The structure of the checking block is illustrated in Figure 8. The block header of the checking block records the block header hash *cblock_(i−1)_header_hash* of the previous checking block, the block body hash *cblock_i_body_hash* of the checking block, the checkpoint number *CheckNum*, as well as the block header hash *block_cycle_i_header_hash* of the corresponding mainchain cycle point block *block_cycle_i*. Meanwhile, the Merkle tree is constructed for the block headers of the mainchain blocks in the mainchain cycle interval, and the root hash of the Merkle tree is obtained as the mainchain cycle interval summary *cblock_i_sum* of *cycle_i*, and it is stored in the block header of the checking block *cblock_i*. Due to the small number of blocks generated in the checking sidechain, the latest checking block does not have enough subsequent blocks to connect, which reduces the security of the checking sidechain. In this regard, this paper uses the block header of the mainchain cycle point block *block_cycle_i* to record the block header hash of the previous checking block *cblock_(i−1)*, using the mainchain to enhance sidechain security.

The block body of the checking block contains the checking transactions, which perform consensus on the consistency of the business world state database in each business domain, respectively. The business nodes obtain the consistency information recorded in the checking transaction corresponding to their business, and use the information to perform consistency checking on their stored business world state databases. After the verification process is completed, the block body of the checking block is discarded, and the block header of the checking block is stored.

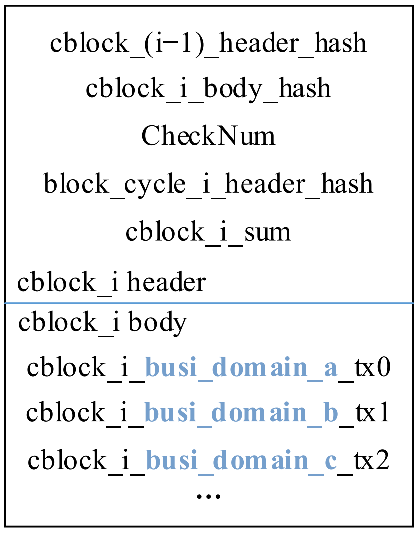
cblock_(i−1)_header_hash
cblock_i_body_hash
CheckNum
block_cycle_i_header_hash
cblock_i_sum
cblock_i header
cblock_i body
cblock_i_busi_domain_a_tx0
cblock_i_busi_domain_b_tx1
cblock_i_busi_domain_c_tx2
…

Figure 8. The structure of checking block.

4.2.3. Checking Block Generation and Consistency Matching

In our scheme, the version *rev* of the world state is recorded in a composite form, i.e., *rev* = *CheckNum_BlockNum_TxNum_VerNum*, by using underscores to connect the subparagraphs. Let the blockchain transaction that generates the current version of the world state be *TX*. *CheckNum* is the checkpoint number corresponding to the mainchain cycle interval in which *TX* is located, *BlockNum* is the block number of the complete block in which *TX* is located, *TxNum* is the serial number of *TX* in the complete block, and *VerNum* is the version number of the world state. *VerNum* records the times the world state has been updated. When the world state is created, *VerNum* equals to 1; each time the world state is updated, *VerNum* is incremented by 1, and the old version of the world state is deleted when the world state obtains a new version.

This paper uses *CheckNum* in the composite version *rev* as the identifier C_i of the rank node. The world state updated in the same mainchain cycle interval corresponds to the same *CheckNum* and the same rank node. The other parameters in *rev* are used as the key *brev* of the data node, i.e., *brev* = *BlockNum_TxNum_VerNum*. Each rank node calculates the consistency information H_{ci} for the world state data it manages, and then obtains the root node H_{part} of the consistency information layer. In this paper, H_{part} is taken as the consistency information of the business world state database.

In our scheme, the checking block contains checking transactions that are initiated by the master nodes in the master domain. When the block height of the mainchain reaches *nc*, the master node constructs a rank B+ tree for the locally stored business world state database of each business domain and obtains the consistency information, respectively. Then, by using the business domain name, the checkpoint number and the consistency information constitute a checking transaction for each business domain.

Meanwhile, to prevent over-reliance on a single master node, the checking transaction initiating master node is rotated by the master nodes in the system. The master nodes that do not act as the initiating node verifies and endorses the checking transactions to ensure the reliability of the checking transactions. In addition, the warning time *t1* and the replacement time *t2* are set, where *t2* > *t1*. If the current master node does not initiate all checking transactions within *t1* after current mainchain cycle point block *block_cycle_i* is generated, a warning is issued to this master node. If this initiation process is still not completed within *t2*, a certain penalty is imposed on this master node, and the next checking transaction initiating master node initiates the remaining checking transactions.

After the checking transactions pass endorsement, the checking block is generated and broadcast to all ledger-keeping nodes in the consortium blockchain system through the ordering service.

As shown in Figure 9, the business node obtains the checking block, extracts the consistency information of the business world state database from the checking block, and uses its stored business world state to construct a rank B+ tree and obtains the root node of the consistency information layer. Then, the obtained root node information is compared with the consistency information obtained from the checking transaction. If the information is consistent, it proves that the business world state stored by this business node is consistent with the business world state stored by the master domain; otherwise, it indicates that there is inconsistent world state in the business world state database stored by the business node.

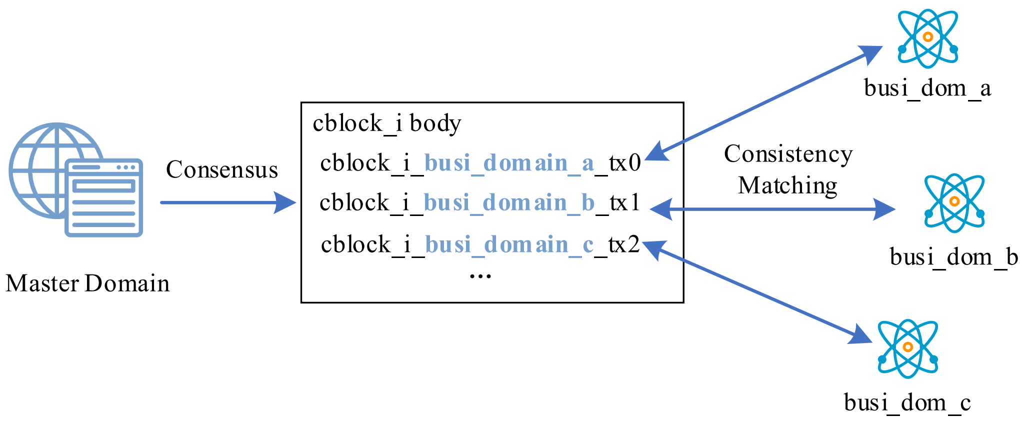

Figure 9. Consistency matching.

When the business node finds inconsistent data in the business world state database stored by itself, it requests the consistency information layer data and the rank layer data in the rank B+ tree from other business nodes in the business domain, and compares them with that obtained from its business world state database to determine the inconsistent rank nodes. Then, the business node requests the data of the data layer managed by the inconsistent rank nodes from other nodes of this business domain, replaces the inconsistent data of the data layer, recalculates the consistency information of the business world state database, and compares it with that recorded in the corresponding checking transaction. If it is consistent, the repair of the business world state database is completed.

4.3. Blockchain Transaction Trusted Tracing Method Based on Two-Level Certification

In our scheme, business nodes only store the business world state related to their business and the block headers of the checking blocks, without storing the chained ledger containing blockchain transactions. However, information such as transaction signatures and transaction endorsements are recorded in blockchain transactions, and business nodes need to obtain the blockchain transactions when they need to re-audit this information. In this regard, the blockchain transaction trusted tracing method based on two-level certification is designed, which enables the business node to initiate a request to the master node of the organization to obtain the transactions and verify the authenticity of the obtained transactions.

4.3.1. Block-Level Proof and Interval-Level Proof

The master node in the master domain stores history database, which record the update history of each world state. Generate a blockchain transaction traceability request by using the *key* of the world state and the name of the business domain where the business node is located. The generated request is signed by the private key of the business node, and sent to the master node of the organization where the business node is located.

After the master node receives the request, it first verifies the signature of the request to ensure its authenticity. After the verification is passed, the history database corresponding to the requested business world state database is retrieved by using the *key* of the world state to obtain the update history of the world state, i.e., $His\_Inf = \{his\_inf_1, his\_inf_2, \ldots, his\_inf_n\}$, where *his_inf* contains the identification *tx_ID* of the transaction obtained by the retrieval, the corresponding checkpoint number *tx_CheckNum*, the block number *tx_BlockNum* that the transaction is in, and the serial number *tx_TxNum* of the transaction in the block. If this update history is empty, it indicates that there is no world state with *key* in the business world state database and the application fails.

If the update history of *key* exists, the location information recorded in the update history is exploited to retrieve the chained ledger to obtain the update transactions $His\_Tx = \{his\_tx_1, his\_tx_2, \ldots, his\_tx_n\}$. At this time, the master node needs to generate the block-level proof *block_prove* and the interval-level proof *interval_prove*, which prove the existence of the transaction in the mainchain block and the mainchain cycle interval, respectively.

- Generation of *block_prove*;

The transaction fetched from the chained ledger is denoted as his_tx_i, where $1 \leq i \leq$ n. For each transaction, *block_prove* needs to be generated to prove that the transaction exists in the respective mainchain block. Let the block number of the mainchain block where the blockchain transaction his_tx_i is located be tx_Block_i. A Merkle tree is constructed by using all blockchain transactions in tx_Block_i, and then the Merkle proof is generated by using some Merkle tree nodes and the hash of his_tx_i. This Merkle proof and the block header of tx_Block_i constitute the block-level proof $block\_prove_i$ for his_tx_i.

- Generation of *block_prove*.

Instead of storing the block headers of the mainchain blocks, the business node in our scheme stores the block headers of the checking blocks, thus taking up less storage space. After the master node proves the existence of the blockchain transaction his_tx_i in the mainchain block tx_Block_i, it needs to prove the existence of tx_Block_i in the mainchain cycle interval. To this end, the master node constructs a Merkle tree by using the block headers of the mainchain blocks in the mainchain cycle interval where tx_Block_i is located. Then, it generates a Merkle proof by using some of the Merkle tree nodes and the block header hash of tx_Block_i, and it constructs $interval\_prove_i$ by using this Merkle proof and the corresponding checkpoint number of the mainchain cycle interval.

The master node packages the blockchain transactions and the transaction proof corresponding to each transaction, signs the packaged data with its private key, and sends the packaged data and the signature information to the business node that initiates the request.

4.3.2. Blockchain Transactions Verification

After receiving the packaged data and the signature information from the master node, the business node first verifies the signature information to ensure that the data has a trustworthy source. It is checked whether the account of blockchain transactions is the same as that of items identified by the version number *VerNum* of the world state. If it is, the authenticity of the blockchain transactions will be verified one by one. First, the checkpoint number in $interval\_prove_i$ is used to obtain the block header of the checking block corresponding to it. The mainchain cycle interval summary *cblock_i_sum* in the block header of the checking block and the Merkle proof in $interval\_prove_i$ are exploited to verify that the block header of tx_Block_i is presented in the mainchain cycle interval. Afterward, the existence of the blockchain transaction his_tx_i in the mainchain block tx_Block_i is verified by using the Merkle proof in $block\_prove_i$. If the verification passes, it is indicated that the blockchain transactions is authentic in the mainchain, and the business node can use the obtained data to re-audit the blockchain transaction.

5. Security Analysis

5.1. Business World State Database Security Isolation and Synchronization

This study generates business blocks through the ordering service. The ordering service is a decentralized consensus module in the consortium blockchain, which ensures the reliability of operation by executing the system chaincodes. The ordering service connects with business docking nodes in each business domain and distributes the business blocks. The distribution of business blocks does not rely on central agency, which ensures that the business block distribution process is highly decentralized.

The business nodes in the same business domain receive $Amount_{list}$ and the block header of the complete block from the organizations they belong to. The business nodes do not need to maintain complete trust in the business docking node in this business domain. They can verify whether the account of the blockchain transactions in the business block is consistent with the $Amount_{list}$ to prevent blockchain transactions in the business block from being maliciously deleted; meanwhile, they use the received multiproofs auxiliary information and the blockchain transactions in the business block to calculate the root node of the sparse Merkle multiproofs. Then, they compare the hash recorded in this root node with the block body hash recorded in the complete block header to ensure the authenticity of the blockchain transactions in the business block. When a business node in the business domain finds that the business block received from the business docking node does not correctly match, it can agree with other business nodes in the business domain on the reason for the unmatch and then determines whether the behavior of the business docking node is trustworthy. If the business docking node is found to have untrustworthy behaviors, it is replaced and penalized.

The business nodes can exploit the blockchain transactions in the verified business blocks to update their respective business world state databases, keeping the ledger synchronized with the mainchain. According to the business world state database, the business node can verify the legitimacy of the read–write sets in the blockchain transactions, and only the blockchain transactions containing the legal read–write sets can update the world state data. In this process, the business node does not need to access the ledger in other business domains, thus realizing security isolation of the ledger and ensuring controlled access to the ledger while ensuring a high decentralization degree.

5.2. Business World State Database Consistency Security

In the consortium blockchain system, the world state database has high independence and no similar chained structure. In this way, the security of the world state database is ensured, and data corruption due to accidental factors or attacker attacks is more difficult to be detected compared with the chained ledger.

To address this issue, this paper constructs a rank B+ tree for the business world state database. The data layer of the rank B+ tree sorts the world state data in a fixed order by the composite key *BlockNum_TxNum_VerNum*. Then, by calculating the hashes of the sorted data of data layer, the consistent ledger in the same business domain can obtain the same rank layer and consistency information layer, thus obtaining the same consistency information H_{part}. The corrupted business world state database stores the ledger that is inconsistent with the normal nodes, which causes the data layer of the constructed rank B+ tree to be also inconsistent. In this case, the mainchain cycle interval to which the corrupted data belongs can be found by comparing the rank layer and the consistency information layer, enabling accurate location of data corruption and reducing the cost of restoring the business world state database. Meanwhile, the consistency verification transactions need to be audited and endorsed by multiple organizations, thus ensuring the credibility of the consistency information.

5.3. Main-Side Chain Cross-Anchoring Structure Safety

In the chained structure of blockchain, the block extracts the block header hash of the previous block and stores it in its block header. To make tampering with a block not be detected, the hash recorded in the block header of all subsequent blocks of that block needs to be modified. The chained structure of the blockchain increases the cost of tampering with the block data, but the newer the block, the fewer the subsequent blocks it is connected to, and the lower the cost of being tampered with.

In our scheme, each checking block corresponds to a mainchain cycle interval, and the checking block has the characteristics of fewer generated blocks and longer generation interval. By storing the block headers of the checking blocks, the business node can verify the authenticity of the blockchain transactions and occupy less space. However, the long interval between the blocks generated results in weaker security of checking blocks compared to mainchain blocks. This paper improves the safety of the checking sidechain by the main-side chain cross-anchoring structure. It records the block header hash of the checking block in the mainchain cycle point block. After the mainchain cycle point block is generated, the mainchain blocks are generated quickly and the sidechain information is stored inside the mainchain. In this case, an attacker who wants to tamper with the checking blocks needs to modify many mainchain blocks together, thus achieving enhanced sidechain security.

6. Experimental Evaluations

The proposed CBCS is implemented and tested by using Go language (version 1.16.4) on the open-source consortium blockchain platform Hyperledger fabric (version 2.2). The hardware and software environment of the experiment platform is as follows: the operating system is Ubuntu 16.04, the CPU is Intel Xeon (Skylake) Platinum 8163 @ 2.5 GHz, and the memory size is 256 GB.

By setting up three organizations {*org1*,*org2*,*org3*} using Docker containers [40], each organization contains five ledger-keeping nodes {*master_peer0_orgi*, *part_peer1_orgi*, *part_peer2_orgi*, *part_peer3_orgi*, *part_ peer4_orgi*}, $(1 \leq i \leq 3)$, including one master node *master_peer0_orgi* and four business nodes *part_peerj_orgi* $(0 \leq j \leq 4)$. Meanwhile, four business domains are set up, and each of the four business nodes in each organization belongs to one business domain. In addition, the number of blockchain transactions in a mainchain block is set to *Tx_Total_Num* = 128, and the checking height threshold is set to $c = 1024$. Then, the number of blockchain transactions in a mainchain cycle interval is *cycle_txcount* = *Tx_Total_Num***c* = 131,072. Assuming that each blockchain transaction has the same probability of belonging to each business domain, the average number of new transactions in a single business domain in a mainchain cycle interval is *cycle_part_txcount* = *cycle_txcount*/4 = 32,768. This paper uses the sample network provided by Hyperledger fabric to construct a system with three organizations, each containing five ledger-keeping nodes, as the comparison system. The ledger-keeping nodes in the comparison system are represented as fabric nodes. The ledger stored by the fabric nodes includes the chained ledger, the world state database, the history database, and the index database.

6.1. Ledger Data Volume Test

The same blockchain transactions are generated in the CBCS system and the comparison system, respectively. The ledger data volume of the business node in the CBCS system is compared with that of the fabric node in the comparison system.

Firstly, the change in the ledger volume is tested when the initiated transactions are all world state creation transactions. At the end of each mainchain cycle, the average ledger volume of the business nodes in the CBCS system and the average ledger volume of the fabric nodes in the comparison system are tested, respectively. Figure 10 shows the variation of the ledger volume during 20 mainchain cycles. When the mainchain cycle number is 2, the amount of generated world state data is 2*cycle_txcount = 262,144, and the average amount of world state data stored by a single business node is 262,144/4 = 65,536. It can be seen that the average ledger volume of the fabric node in the comparison system is 1129 Mb, while that of the business node in the CBCS system is 30 Mb. When the mainchain cycle number is 20, the amount of generated world state data is 20*cycle_txcount = 2,621,440, and the average amount of world state data stored in a single business node is 2,621,440/4 = 655,360. The ledger volume of the fabric node in the comparison system is 11,254 Mb, while that of the business node in the CBCS system is 292 Mb.

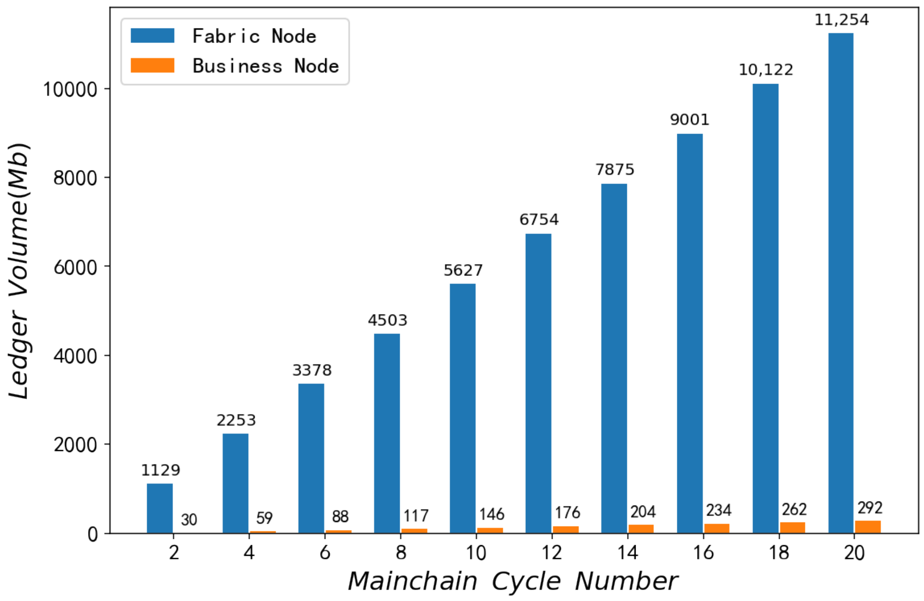

Figure 10. Comparison of the ledger data volume (single world state creation transactions).

To realize a more accurate comparison, the ratio of the ledger volume of the business node in the CBCS system to the ledger volume of the fabric node in the comparison system is calculated to obtain the compression ratio, *com_ratio* = *volume_business/volume_fabric*. A smaller compression ratio indicates that the business node in the CBCS system occupies less space for ledger storage compared to the fabric node in the comparison system.

The compression ratio over 20 mainchain cycles is calculated and shown in Figure 11. When all blockchain transactions are world state creation transactions, the average ledger volume of the business node in our scheme is 2.6053% of that of the fabric node in the comparison system, indicating that our scheme can efficiently reduce the ledger storage space of business nodes.

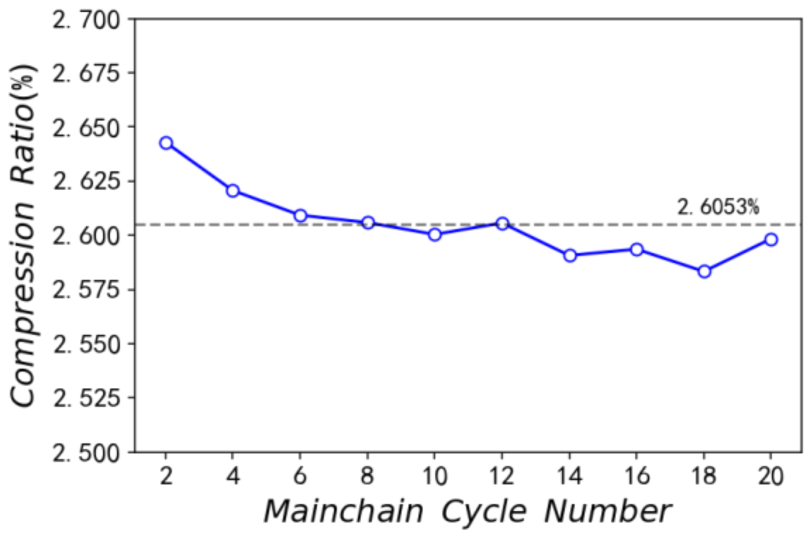

Figure 11. The compression ratio (single world state creation transactions).

6.2. Transaction Type Impact Test

In the test of Section 6.1, the initiated blockchain transactions are all world state creation transactions. However, in practical applications, blockchain transactions that update world state are used more frequently. Different types of blockchain transactions have different impacts on the blockchain's ledger. To investigate the changes in the ledger volume when the consortium blockchain system involves different proportions of world state creation and update transactions, the world state update transactions are initiated based on the experiment in Section 6.1. The ledger volume of the business nodes in the CBCS system is tested and compared with that of the fabric nodes in the comparison system, and the result is shown in Figure 12. The blockchain transactions in the 21st mainchain cycle to the 60th mainchain cycle are all world state update transactions, and no new world state is created.

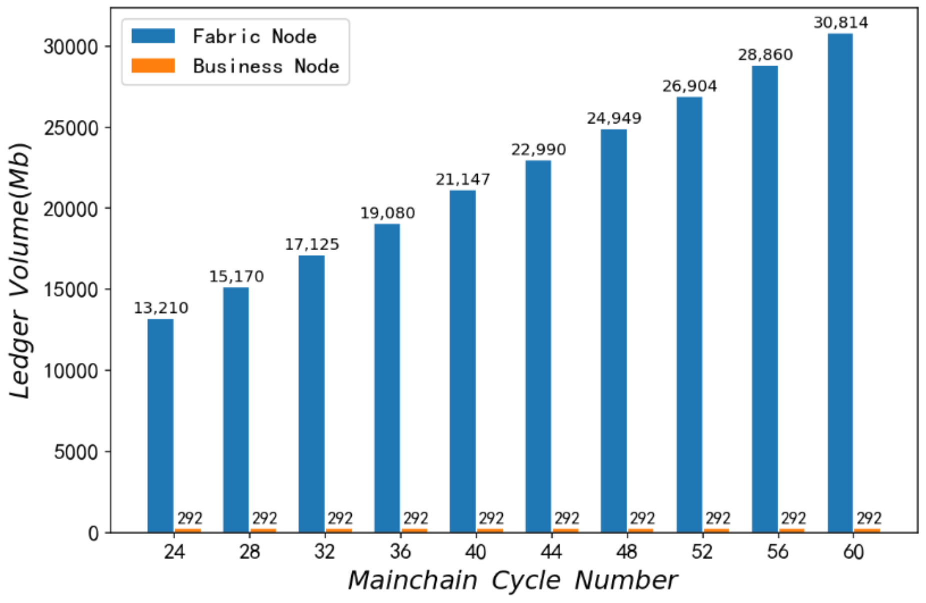

Figure 12. Comparison of the ledger data volume (world state creation and update transactions).

It can be seen from Figure 12 that as the number of world state update transactions increases, the ledger volume of the fabric node in the comparison system still grows at a faster rate, while the ledger volume of the business node in the CBCS system remains almost unchanged. The difference is that the world state creation transaction adds records to the chained ledger and the world state database at the same time; the world state update transaction increases the amount of data in the chained ledger without generating new world state data, so it has less impact on the world state database.

Then, the compression ratio in the 21st mainchain cycle to the 60th mainchain cycle is calculated, and the result is presented in Figure 13. From the figure, it can be seen that as the proportion of world state update transactions gradually increases, the compression ratio of the ledger of the business node and the fabric node in our scheme gradually decreases. When the mainchain cycle number is 40, the ratio of world state creation transactions to world state update transactions is 1:1. In this case, the ledger volume of the business node is 1.38% that of the fabric node in the comparison system; when the mainchain cycle number is 60, the ratio of world state creation transactions to world state update transactions is 1:2. In this case, the ledger volume of the business node is 0.95% that of the fabric node in the comparison system. These results indicate the higher the ratio of world state update transactions, the better the effect of saving business node ledger storage space in our scheme.

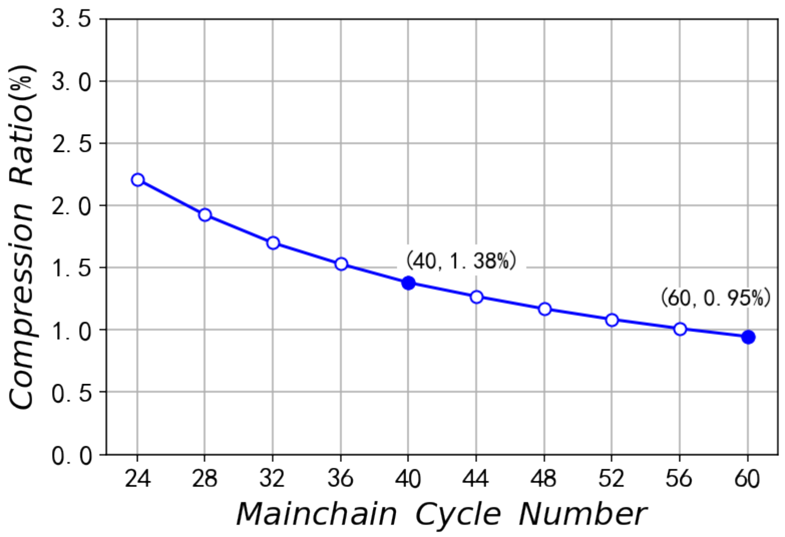

Figure 13. The compression ratio (world state creation and update transactions).

Table 1 shows the comparison between our scheme and the existing schemes. It can be found that our scheme obtains a better compression rate than existing schemes, which can save the storage space of the blockchain nodes better. The work [18] and our scheme achieve a similar compression rate, but the blockchain nodes in the former only store data such as blockchain transaction hash and the previous block hash. When the blockchain nodes need ledger data, it needs to initiate a request to obtain ledger data from the off-chain storage nodes that store the complete blockchain ledger data, thus incurring more time and communication consumption. In the literature, [30] uses the same way of collaborative storage as our scheme. However, our scheme extracts the ledger data at the granularity of world state, which can save the storage space of business nodes more fully compared with [30] which divides the ledger data at the granularity of block. Meanwhile, the business nodes in our scheme can efficiently obtain the required ledger data by retrieving the locally stored business world state database.

Table 1. Comparison of CBCS and the related schemes.

Scheme	Main Technology	Storage Location of Ledger Data	Query Method	Compression Rate
Liu et al. [25]	filter transactions	on-chain	local retrieval	56.65%
Guo et al. [27]	encode blocks	on-chain	encode fragments	20%
Zheng et al. [16]	storage in IPFS	off-chain + on-chain	ask from off-chain	8.17%
Xu et al. [30]	collaborative storage	on-chain	local retrieval or ask from others	5%
Xu et al. [18]	storage in off-chain nodes	off-chain + on-chain	ask from off-chain	3~1%
Ours	collaborative storage	on-chain	local retrieval	0.95%

6.3. Sidechain Data Impact Test

In our scheme, business nodes store the business world state database and the block headers of the checking blocks. The setting of the checking sidechain increases the ledger storage consumption of the business nodes. This paper investigates the proportion of the sidechain block headers in the business nodes' ledger volume at the end of different mainchain cycles, and the result is shown in Figure 14. After the 20th mainchain cycle, the blockchain transactions initiated are all world state update transactions, and the data volume of the business world state database stops increasing. Meanwhile, the checking blocks continue to be generated. This makes the proportion of the block headers of the checking sidechain in the ledger data volume of the business node gradually increase. The proportion reaches the highest of 0.00632% after the end of the 60th mainchain cycle. At this time, the block headers of the checking sidechain still account for a small percentage of the overall ledger volume of the business node.

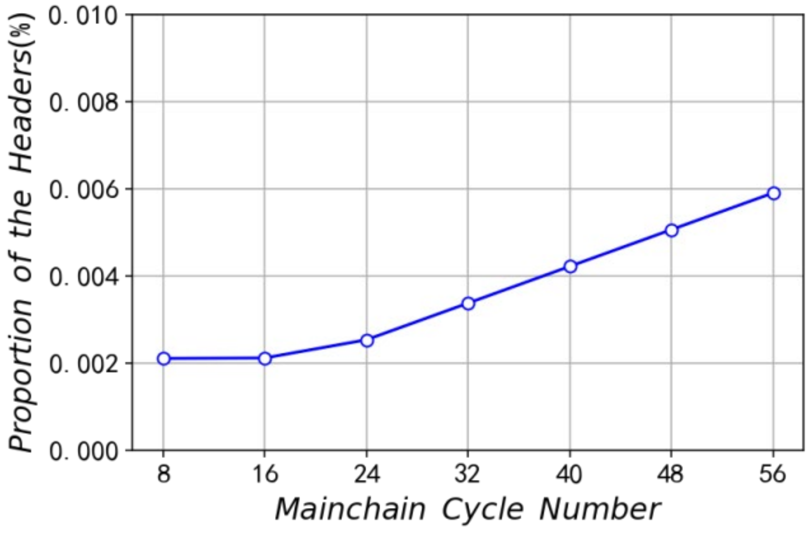

Figure 14. The proportion of sidechain data in the ledger volume of the business node.

6.4. Efficiency Test of Retrieving State Data

To verify the efficiency of the business node in obtaining the state data, the business node in the CBCS system retrieves the stored business world state database, records the retrieval time and compares it with the time spent by the fabric node in the comparison system to obtain Figure 15. It can be seen that the state data retrieval time of fabric node is slightly higher. When the amount of retrieved data is consistent, calculate the difference between the retrieval time of the business node and the fabric node, and their difference does not exceed 0.1 s. This result indicates that the business node can efficiently obtain the required ledger data through its own stored business world state database.

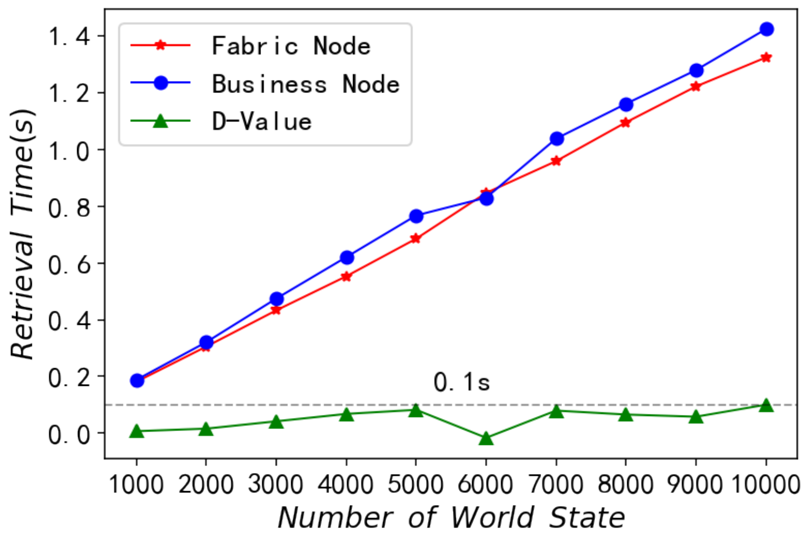

Figure 15. Retrieve state data test.

7. Conclusions

This paper proposes a scalable consortium blockchain architecture based on world state collaborative storage. In this architecture, the business nodes collaboratively store the world state database to effectively reduce the pressure of ledger storage and isolate the ledger data between different business domains. The consistency of the business world state stored by the business nodes can be verified by initiating consistency verification transactions through the checking sidechain. The proposed scheme can effectively save the ledger storage space of the consortium blockchain nodes while ensuring a high decentralization degree and retrieval efficiency of the ledger data. Meanwhile, the structure design of the scheme has a high similarity with the internal organizational structure of the social organization or group, which is convenient for the actual deployment in the social organization or group. It is important to note that the checking height threshold c in our scheme determines the time interval between two consistency checkings, which affects both the timeliness of discovering world state data corruption and the computational overhead of consistency checkings. It is necessary to set the checking height threshold c according to different needs in different application scenarios. In the future, we will apply our designed scheme to projects related to specific scenarios such as big data sharing, and conduct a more detailed evaluation of the practical application effect of the scheme.

Author Contributions: Conceptualization, J.Z., N.W., A.L., W.W. and X.D.; methodology, J.Z., N.W., A.L., W.W. and X.D.; validation, J.Z. and N.W.; formal analysis, J.Z.; investigation, J.Z., N.W., A.L., W.W. and X.D.; writing—original draft preparation, J.Z., N.W. and A.L.; writing—review and editing, J.Z., N.W., A.L., W.W. and X.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (No. 2018YFB0803603), the Key Research and Development and Promotion Program of Henan Province (No. 222102210069), and the National Natural Science Foundation of China (No. 62102449, No. 61802436).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berdik, D.; Otoum, S.; Schmidt, N.; Porter, D.; Jararweh, Y. A survey on blockchain for information systems management and security. *Inf. Process. Manag.* **2021**, *58*, 102397. [CrossRef]
2. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 16 September 2022).
3. Sun, Z.; Zhang, X.; Xiang, F.; Chen, L. Survey of storage scalability on blockchain. *J. Softw.* **2021**, *32*, 1–20.
4. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
5. Dib, O.; Brousmiche, K.L.; Durand, A.; Thea, E.; Hamida, E.B. Consortium blockchains: Overview, applications and challenges. *Int. J. Adv. Telecommun.* **2018**, *11*, 51–64.
6. Gorenflo, C.; Lee, S.; Golab, L.; Keshav, S. FastFabric: Scaling hyperledger fabric to 20,000 transactions per second. *Int. J. Netw. Manag.* **2020**, *30*, e2099. [CrossRef]
7. Zhang, A.; Lin, X. Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. *J. Med. Syst.* **2018**, *42*, 1–18. [CrossRef] [PubMed]
8. Gao, Z.; Cao, L.; Du, X. Data right confirmation mechanism based on blockchain and locality sensitive hashing. In Proceedings of the 2020 3rd International Conference on Hot Informationcentric Networking (HotICN), Hefei, China, 12–14 December 2020.
9. Nyaletey, E.; Parizi, R.M.; Zhang, Q.; Choo, K.K.R. BlockIPFS-blockchain-enabled interplanetary file system for forensic and trusted data traceability. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019.
10. Chen, X.; Zhang, K.; Liang, X.; Qiu, W.; Zhang, Z.; Tu, D. HyperBSA: A high-performance consortium blockchain storage architecture for massive data. *IEEE Access.* **2020**, *8*, 178402–178413. [CrossRef]
11. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Yellick, J. Hyperledger fabric: A distributed op-erating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018.
12. Yu, G.; Wang, X.; Yu, K.; Ni, W.; Zhang, J.A.; Liu, R.P. Survey: Sharding in blockchains. *IEEE Access* **2020**, *8*, 14155–14181. [CrossRef]
13. Gao, Z.; Zhang, D.; Zhang, J. A security problem caused by the state database in Hyperledger Fabric. In Proceedings of the International Conference on Frontiers in Cyber Security, Tianjin, China, 15–17 November 2020; pp. 361–372.
14. Sharma, P.; Jindal, R.; Borah, M.D. Blockchain technology for cloud storage: A systematic literature review. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–32. [CrossRef]

15. Benet, J. Ipfs-Content Addressed, Versioned, p2p File System. Available online: https://arxiv.org/abs/1407.3561 (accessed on 16 September 2022).
16. Zheng, Q.; Li, Y.; Chen, P.; Dong, X. An innovative IPFS-based storage model for blockchain. In Proceedings of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Santiago, Chile, 3–6 December 2018; pp. 704–708.
17. He, G.; Su, W.; Gao, S. Chameleon: A scalable and adaptive permissioned blockchain architecture. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 15–17 August 2018; pp. 87–93.
18. Xu, C.; Zhang, C.; Xu, J.; Pei, J. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing. *Proc. VLDB Endow.* **2021**, *14*, 2314–2326. [CrossRef]
19. Suratkar, S.; Shirole, M.; Bhirud, S. Cryptocurrency wallet: A review. In Proceedings of the 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India, 28–29 September 2020.
20. Nauman, A.; Qadri, Y.A.; Amjad, M.; Zikria, Y.B.; Afzal, M.K.; Kim, S.W. Multimedia Internet of Things: A comprehensive survey. *IEEE Access* **2020**, *8*, 8202–8250. [CrossRef]
21. Chakravarty, M.M.; Chapman, J.; MacKenzie, K.; Melkonian, O.; Peyton Jones, M.; Wadler, P. The extended UTXO model. In Proceedings of the International Conference on Financial Cryptography and Data Security, Kota Kinabalu, Malaysia, 14 February 2020; pp. 525–539.
22. Palai, A.; Vora, M.; Shah, A. Empowering light nodes in blockchains with block summarization. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018.
23. Kim, T.; Noh, J.; Cho, S. SCC: Storage compression consensus for blockchain in lightweight IoT network. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 11–13 January 2019.
24. De Angelis, S.; Aniello, L.; Baldoni, R.; Lombardi, F.; Margheri, A.; Sassone, V. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. In Proceedings of the Italian Conference on Cyber Security, Milan, Italy, 6 February 2018.
25. Liu, Y.; Wang, K.; Lin, Y.; Xu, W. LightChain: A lightweight blockchain system for industrial internet of things. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3571–3581. [CrossRef]
26. Qi, X.; Zhang, Z.; Jin, C.; Zhou, A. BFT-Store: Storage partition for permissioned blockchain via erasure coding. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1926–1929.
27. Guo, Z.; Gao, Z.; Liu, Q.; Chakraborty, C.; Hua, Q.; Yu, K.; Wan, S. RNS-based adaptive compression scheme for the block data in the blockchain for IIoT. *IEEE Trans. Ind. Inform.* **2022**, *18*, 9239–9249. [CrossRef]
28. Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Math. Comput. Simul.* **2020**, *177*, 232–243. [CrossRef]
29. Chen, X.; Lin, X.; Yu, N. Compression of bitcoin blockchain. *Chin. J. Netw. Inf. Secur.* **2021**, *7*, 76–83.
30. Xu, Z.; Han, S.; Chen, L. CUB, a consensus unit-based storage scheme for blockchain system. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 16–19 April 2018; pp. 173–184.
31. Chen, L.; Zhang, X.; Sun, Z. Scalable Blockchain Storage Model Based on DHT and IPFS. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 2286–2304.
32. Morris, R.; Kaashoek, M.F.; Karger, D.; Balakrishnan, H.; Stoica, I.; Liben-Nowell, D.; Dabek, F. Chord: A scalable peer-to-peer look-up protocol for internet applications. *IEEE/ACM Trans. Netw.* **2003**, *11*, 17–32.
33. Xie, J.; Li, Z.; Jin, J.; Zhang, B.; Hua, Y. Research on Blockchain Storage Extension Based on DHT. In Proceedings of the 2021 4th International Con-ference on Big Data Technologies, Zibo, China, 24–26 September 2021; pp. 79–85.
34. Marandi, A.; Sehat, H.; Lucani, D.E.; Mousavifar, S.; Jacobsen, R.H. Network Coding-based Data Storage and Retrieval for Kademlia. In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021.
35. Tolmach, P.; Li, Y.; Lin, S.W.; Liu, Y.; Li, Z. A survey of smart contract formal specification and verification. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
36. Zhou, W.; Lu, J.; Luan, Z.; Wang, S.; Xue, G.; Yao, S. SNB-index: A SkipNet and B+ tree based auxiliary Cloud index. *Clust. Comput.* **2014**, *17*, 453–462. [CrossRef]
37. Sousa, J.; Bessani, A.; Vukolic, M. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In Proceedings of the 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN), Luxembourg, 25–28 June 2018; pp. 51–58.
38. Mizrahi, A.; Koren, N.; Rottenstreich, O. Optimizing Merkle proof size for blockchain transactions. In Proceedings of the 2021 International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 5–9 January 2021; pp. 299–307.
39. Ramabaja, L.; Avdullahu, A. Compact merkle multiproofs. *arXiv* **2020**, arXiv:2002.07648.
40. Potdar, A.M.; Narayan, D.G.; Kengond, S.; Mulla, M.M. Performance evaluation of docker container and virtual machine. *Procedia Comput. Sci.* **2020**, *171*, 1419–1428. [CrossRef]