Scalable Blockchain Storage Model Based on DHT and IPFS

Lu Chen[1,2,3] , Xin Zhang[1,2,3], Zhixin Sun[1,2,3*]
[1] Engineering Research Center of Post Big Data Technology and Application of Jiangsu Province, Nanjing University of Posts and Telecommunications, Nanjing 210003 China
[2] Research and Development Center of Post Industry Technology of the State Posts Bureau (Internet of Things Technology), Nanjing University of Posts and Telecommunications, Nanjing 210003 China
[3] Engineering Research Center of Broadband Wireless Communication Technology of the Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003 China
[e-mail:2018070261@njupt.edu.cn, sunzx@njupt.edu.cn]
*Corresponding author: Zhixin Sun

Received October 22, 2021; revised March 2, 2022; revised April 27, 2022; accepted June 12, 2022; published July 31, 2022

Abstract

Blockchain is a distributed ledger that combines technologies such as cryptography, consensus mechanism, peer-to-peer transmission, and time stamping. The rapid development of blockchain has attracted attention from all walks of life, but storage scalability issues have hindered the application of blockchain. In this paper, a scalable blockchain storage model based on Distributed Hash Table（DHT) and the InterPlanetary File System (IPFS) was proposed. This paper introduces the current research status of the scalable blockchain storage model, as well as the basic principles of DHT and the InterPlanetary File System. The model construction and workflow are explained in detail. At the same time, the DHT network construction mechanism, block heat identification mechanism, new node initialization mechanism, and block data read and write mechanism in the model are described in detail. Experimental results show that this model can reduce the storage burden of nodes, and at the same time, the blockchain network can accommodate more local blocks under the same block height.

Keywords: blockchain, DHT, IPFS, storage optimization, storage scalability

This work is supported by the National Nature Science Foundation of China (No. 61972208, No.61672299), Postgraduate Research &Practice Innovation Program of Jiangsu Province(SJKY19_0770). The authors thank the sponsors for their support and the reviewers for helpful comments.

http://doi.org/10.3837/tiis.2022.07.009 **ISSN : 1976-7277**

1. Introduction

Satoshi Nakamoto first proposed the concept of Bitcoin in 2008 in "Bitcoin: A Peer-to-Peer Electronic Cash System" [1], and blockchain is the core technology of digital cryptocurrencies such as Bitcoin. After 2015, the rapid development of the second-generation blockchain Ethereum [2] has made blockchain technology gradually understood and explored by more people. Blockchain is a decentralized infrastructure and distributed computing paradigm. It uses cryptography technology to ensure that data cannot be tampered with and cannot be forged, uses distributed consensus mechanisms to generate and update data, and uses automated script code to program and manipulate data [3]. These properties of blockchain make it applicable to many non-cryptocurrency fields. However, the storage scalability problem limits the widespread application of blockchain. Therefore, this paper constructs a Distributed Hash Table (DHT) and InterPlanetary File System(IPFS)-based blockchain storage model (DIBS). Our model adopts an improved collaborative storage scheme and block replacement algorithm, which solves the storage scalability problem that restricts the implementation of current blockchain applications.

According to the different data models in the block, blockchain can be divided into transaction type blockchain and account type blockchain [4], among which transaction type blockchain can be divided into narrow transaction type and generalized transaction type. Represented by Bitcoin, the blocks in the narrow transactional blockchain store transaction records, which are composed of transaction input, transaction output, and other information. And the traceability of transactions is achieved by linking the input and output of different transactions. In a generalized transactional blockchain, any serializable data such as authentication information, storage information, authentication information can be regarded as a transaction. In the account blockchain led by Ethereum, the block needs to record the current balance and status of each account.

With the development of blockchain technology, the amount of data on the blockchain has grown rapidly. As of the end of January 2021, the size of the Bitcoin ledger using the transactional data model has reached 317GB. This makes nodes that join the Bitcoin network have to synchronize such a huge blockchain ledger first, which affects the willingness of nodes to join the Bitcoin network. According to the growing trend of the ledger in the past ten years, it is expected that it will reach 761GB in 2025 and an astonishing 1637GB in 2030. Moreover, the current tens of millions of Bitcoin nodes use a total of PB of space to maintain only about 300G of the original ledger, and the storage efficiency is very low. On the other hand, Ethereum, the representative of the account-based blockchain, also faces the same problem. According to data from Etherscan [5], at the end of January 2021, the size of the OpenEthereum client has also reached 375GB. The continuous expansion of the blockchain ledger makes any transaction or account blockchain that uses a highly redundant storage mechanism (each node stores complete data) will face storage scalability problems.

At present, scholars have conducted relevant research on the scalability of blockchain storage. The Zheng team [6] proposed a blockchain storage model based on IPFS, which relieves the store pressure of the narrow transactional blockchain by transferring verified transactions from the local transaction pool to the IPFS system. Another way to reduce the storage pressure of a narrow transactional blockchain is the light node mode [7]. Unlike full nodes that need to store a complete ledger, independently verify transactions, and update the blockchain in real-time, light nodes only need just download all the block headers. Although the light node model that does not store the complete ledger reduces the storage burden, it still needs to rely on the full nodes in the network to be able to verify transactions. Literature [8]

designed an improved light node that does not need to rely on full nodes. After joining the network, it only need to download the latest six blocks, the Unspent Transaction Outputs (UTxO) shard hash list, and the block hash list to verify the transaction. Literature [9] uses the characteristic that transactions are composed of inputs, outputs, and amounts to merge and summarize transactions in a certain range of blocks, so it reduces the number of transactions and blocks, and ultimately achieves the purpose of reducing the storage space occupied by the ledger. In order to cope with the problem of tight node storage resources caused by the need to store all smart contract codes in Ethereum, the solution in [10] transfers the contract code to the off-chain IPFS system when the smart contract is created, avoiding smart contract codes with low usage rates that take up a lot of storage space. Literature [4] proposes an improved account-based blockchain storage model, which saves all account state data and shards the block data to reduce the consumption of storage space. The above studies have optimized the storage models of narrow transactional blockchains and account-based blockchains, but the blockchain application scenarios of these two data models are more limited to the field of digital encryption currency.

The generalized transactional blockchain has a wider range of application scenarios, such as finance [11-13], Internet of Things [14-15], government affairs [16-17], copyright protection [18-19], information sharing [20-21] and traceability [22-23] and other fields. These application scenarios above pay more attention to storage scalability requirements than digital encryption currency scenarios. Therefore, the storage scalability of the generalized transactional blockchain, which is mainly used in the field of non-digital encrypted currencies, directly affects whether the blockchain can be widely used.

This paper will aim at the scalability storage requirements of the generalized transactional blockchain, based on the distributed hash table technology and the interplanetary file system, to build a DHT and IPFS-based blockchain storage model (DIBS), which solves the storage scalability problem that restricts the implementation of current blockchain applications. The details of our proposal are as follows：

(1) Propose a scalable storage model applied to generalized transactional blockchains. Through the off-chain IPFS system, part of the storage tasks of generalized transactional blockchain nodes is shared, reducing the pressure on the local storage of blockchain nodes, and avoiding the continuous linear growth of the local storage load of blockchain nodes.

(2) Propose an improved collaborative storage scheme.

Construct a DHT network based on the Chord protocol according to the node's hardware capabilities and storage willingness. Among them, the blockchain node and neighboring nodes cooperate to jointly maintain a complete blockchain ledger. With the same storage consumption, the network can accommodate a larger amount of local block data.

(3) Propose a block replacement algorithm.

In the process of querying block data, nodes frequently request the off-chain storage system, which will cause the inefficiency of the query. The algorithm distinguishes the block data based on the frequency with which the block data is queried, realizes the efficient replacement of local hot block data and off-chain cold block data, and improves the efficiency of block data query.

2. Related Work

All paragraphs except the first one must be indented. All references are numbered in the order they appear in the text. Reference numbers in the text are red [1].

The current methods to improve the scalability of blockchain storage include off-chain storage

and on-chain storage [7]. Off-chain storage [24] means that the blockchain node transfers part of the data in the block body to the off-chain storage system, and the block body only stores "pointers" to these data. The on-chain storage [7] method does not need to rely on an additional off-chain storage system, nor does it require each node to store a complete blockchain ledger. Nodes only need to store the corresponding part of the ledger according to the pre-arranged agreement. Common on-chain storage methods include collaborative storage mode and light node mode. In collaborative storage, the status of blockchain nodes is equal, and the nodes cooperate to achieve the same functions as "full nodes". The light node mode needs to rely on full nodes to restore the complete blockchain ledger. This feature also makes the light node model mainly used in the narrow transactional blockchain, such as Simplified Payment Verification（SPV) light nodes. For generalized transactional blockchains with broader application scenarios, off-chain storage and collaborative storage are the two main methods to improve storage scalability.

At present, many studies use off-chain storage to improve the storage scalability of the blockchain. Literature [25] optimizes the traditional transaction storage mode, no longer requires nodes to store a complete blockchain ledger. It saves the original block data to a storage system based on the Kademlia protocol [26]. The system is maintained by nodes unrelated to the blockchain network to relieve the storage pressure of blockchain nodes. Literature [27] proposed a large-scale (Internet of Things) IoT data storage and protection scheme, using edge computing [28] and an off-chain DHT system to make up for the lack of computing and storage capacity of IoT devices. Literature [29] builds a modular alliance architecture based on blockchain and IPFS, which reduces the degree of data centralization of IoT and also solves the problem that traditional blockchain cannot store massive amounts of data. Literature [30] uses the IPFS system to store metadata and large data in the service market, and the hash value of these data is stored in the Ethereum block. In the social media application built by the Xu team [31] based on Ethereum and IPFS, Ethereum is responsible for storing user data, and IPFS is responsible for storing large file data. Chameleon [32] believes that in some scenarios, the correlation between data in different time dimensions is extremely low, so only the data in the most recent period can be stored in the blockchain, and the rest of the data can be stored on the cloud. The chain stores the hash value of the latest block on the cloud to maintain data consistency.

Through the off-chain DHT system, IPFS, or cloud storage, the pressure generated by storing the relevant data in the original block is transferred to the off-chain storage system, which can reduce the storage consumption of blockchain nodes. But at the same time, when nodes in the blockchain or applications built by the blockchain query these data, they need to frequently access the off-chain storage system, which is less efficient and not suitable for the query scene in generalized transactional blockchain applications.

In addition to off-chain storage, collaborative storage is also a hot topic in academic research. Literature [33] proposed a blockchain sharding storage model based on an improved Shamir threshold secret sharing. It first constructs a $k-1$ degree polynomial through the improved Shamir threshold mechanism and constructs n data blocks through the polynomial. Then it distributes the $n-1$ data blocks that are not stored by itself to other $n-1$ nodes in the network. Literature [34] first divides the original block into k sub-blocks and uses network coding technology to generate n sub-coded blocks. Then it divides the blocks into $2k$ sub-blocks and repeats the above process until the nodes in the blockchain network all hold the coded sub-blocks corresponding to the current block. The node only needs to request a certain number of remaining coded sub-blocks from other nodes in the network, and the original block can be reconstructed through the network coding mechanism. The Kaneko team [35] built a

blockchain storage load balancing mechanism through the Kademlia cluster, which performs an exclusive OR (XOR) operation between the hash value of the new block and the node ID. The cluster to which the node closest to the hash value belongs is responsible for storing the block. Literature [36] uses a low-density check code algorithm to enable blockchain nodes to detect malicious nodes and reduce the storage cost consumed by the blockchain system. The Gadiraju team[37] determines the block sequence length L according to the number of nodes (n_s) in the shard and the number of nodes stored (α). And it then uses the characteristics of the safe regeneration code to increase the speed of new node initialization and reduce the storage cost to $n_s\alpha/L$. The above research uses a collaborative storage method based on encoding, clustering, or threshold secret sharing, and only a part of the original burden needs to be stored. When these nodes that adopt a cooperative storage mechanism need a complete block, the block can be reconstructed through the agreed rules. But these methods are all $1/k$-like methods, that is, the storage cost of the node is reduced to $1/k$ before optimization. With the continuous generation of new blocks, the optimized node storage overhead will continue to grow, and the future will also face storage bottlenecks. Therefore, we will propose a scalable storage model for generalized transactional blockchains, which reduces the storage consumption of nodes while avoiding the continuous linear growth of node storage burden.

3. Preliminaries

3.1 DHT

DHT [38] stores resources on different nodes through a predetermined agreement, and uses unique keys to identify specific resources. In the DHT network, each resource has a unique identifier, so the storage location of each resource is fixed. DHT does not rely on a centralized server. All nodes maintain routing information in the network to undertake the task of addressing and storing resources. DHT performs hash mapping on resource keywords and obtains a string as the unique identification K of the resource. Similarly, hash mapping is performed on the IP or other keywords of the node in the DHT network, and a string is obtained as the unique identification ID of the node. Finally, according to the rules, the identification K of the resource is mapped to the identification ID of the node, and the resource identification K corresponds to a certain number of nodes, that is, these corresponding nodes will assume the task of storing the resource. Common DHT protocols include Chord protocol [39], Pastry protocol [40], Content-Addressable Network (CAN) protocol [41].

3.2 Chord protocol

Chord, one of the mainstream DHT protocols, was proposed by MIT in 2001. It appears to solve the basic problem encountered in P2P applications: how to find the nodes in P2P networks that hold specific data. Chord defines how to query where key values are stored, how to add and remove nodes, and how to recover from node failures.

Chord is a structured, fully distributed topology. It ensures consistent hashing by mapping nodes and keys into the same space, and to ensure that hashing is non-repetitive, Chord chooses SHA-1 as the hash function. Sha-1 produces a 2^160 space, each of which is a large 16-byte integer. The integers are joined end to end to form a loop called a Chord loop. Integers are arranged clockwise by size on the Chord ring. Node (machine IP address and Port) and Key (resource identification) are hashed onto the Chord ring, thus assuming the state of the entire P2P network as a virtual ring.

3.3 IPFS

IPFS [42] is a distributed hypermedia distribution protocol based on a point-to-point model. It uses the advantages of DHT, BitTorrent [43], and Git [44], and uses Merkle Directed Acyclic Graph (MerkleDAG) as the underlying data structure to provide a permanent decentralized content-addressed file storage method. After the client uploads the file to the IPFS node, it will receive a hash string that is the unique identifier of the file returned by the IPFS node. The client can obtain the file from any node of IPFS through the identifier. The IPFS node that receives the file acquisition request finds the node location where the file is stored through the underlying DHT service. Then it retrieves the file and verifies the data and returns it to the requesting client.

4. Scalable Blockchain Storage Model Based on DHT and IPFS

This section introduces our proposed scalable blockchain storage model (DIBS model) based on DHT and IPFS. It includes model construction, DHT network construction mechanism, scalable block heat identification mechanism, new node initialization mechanism, and block data reading and writing mechanism.

As shown in **Fig. 1**, the DIBS model we constructed is specifically composed of storage resource-constrained nodes, full nodes, and an off-chain IPFS system:

Storage resource-constrained nodes: storage-resource-constrained nodes refer to blockchain nodes that objectively have insufficient storage capacity or subjectively low willingness to store.

Full node: Full node refers to a blockchain node that is not troubled by storage resources. It has enough storage space and can undertake all responsibilities.

Off-chain IPFS system: The off-chain IPFS system mainly stores cold block data dynamically replaced by resource-constrained nodes through a block replacement algorithm.

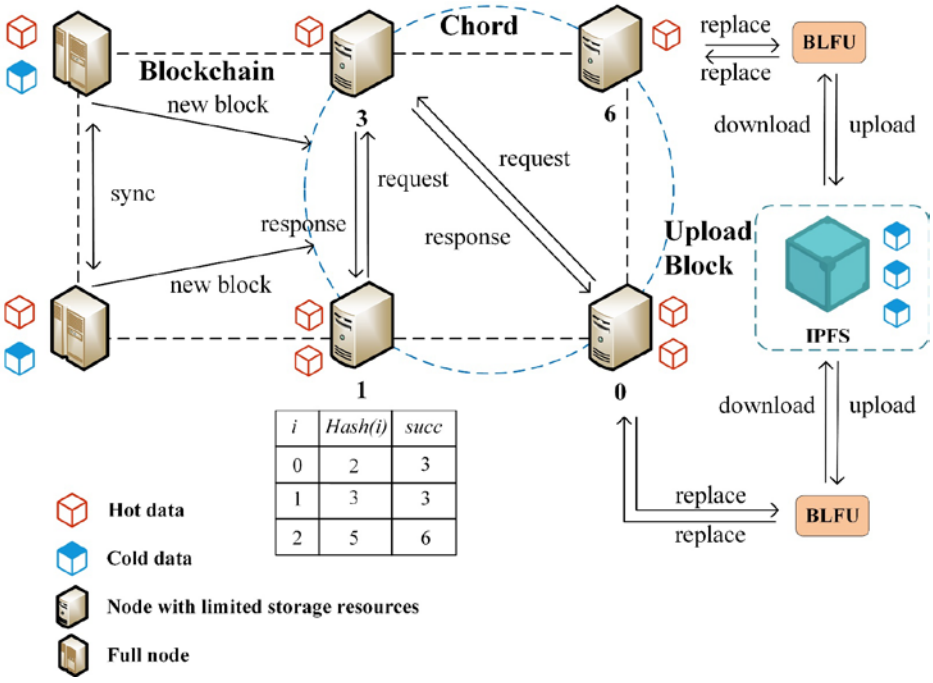

Fig. 1. DIBS model

In the DIBS model, nodes with limited storage resources first form the Chord ring through the DHT network construction mechanism and then store the corresponding block data according to the scalable block heat identification mechanism, which can cooperate with other nodes in the Chord ring. This model can realize the function of a full node without the need to store a complete blockchain ledger. For newly joined nodes with limited storage resources, they only need to use the new node initialization mechanism to join the Chord ring that is already in operation. The nodes in the Chord ring can synchronize with the full node to ensure the normal operation of the blockchain application. When a read and write request is received, the full node can respond to the request by operating the local block; Resource-constrained nodes complete read operations in local storage, in-ring nodes, or IPFS systems through the block read mechanism, and complete the storage tasks of the new node through the block write mechanism.

Sections 3.1 to 3.4 will sequentially introduce the DHT network construction mechanism, scalable block heat identification mechanism, new node initialization mechanism, and block data read and write mechanism of the DIBS model.

4.1 DHT Network Construction Mechanism

For the storage resource-constrained nodes that need to be optimized in the blockchain network, the DHT network can be established to cooperate to achieve the purpose of reducing the pressure on each storage-resource-constrained node. The DIBS model will use the Chord protocol to build a DHT network. In the Chord protocol, each node uses a hash algorithm to determine the location of the node in the Chord ring network based on its IP address and port number. Each node randomly determines its position in the Chord ring through a hash algorithm, so the distance between adjacent nodes is not related to their hardware limitations or subjective wishes.

As shown in **Fig. 2**, there are three nodes 0, 1, and 2 in the Chord ring network, and B_i represents block i. The storage pressure of node 1 with weaker storage capacity is much greater than that of node 0 with stronger storage capacity. Therefore, only determining the location of the node through the hash algorithm will cause nodes with relatively high hardware levels to be responsible for only a small part of the storage tasks, while nodes with weak computing power and small storage space are responsible for storage tasks over a large distance.

Therefore, this mechanism introduces virtual nodes to solve the problem of mismatch between node performance and storage pressure. Let Sm_i be the maximum storage capacity of the node. Com_i represents the computing power of the node, Sw_i represents the storage size that the node is willing to bear, and W_i represents the network bandwidth of the node. Then, the node performance is shown in formula (1):

$$P_i = k_1 Sm_i + k_2 Com_i + k_3 Sw_i + k_4 W_i \tag{1}$$

Among them, k_i represents the corresponding weight coefficient. Suppose the number of virtual nodes is N_v, then the number of virtual nodes allocated to each node is

$$num_i = N_v \times \frac{P_i}{\sum_{i=0}^{n} P_i} \tag{2}$$

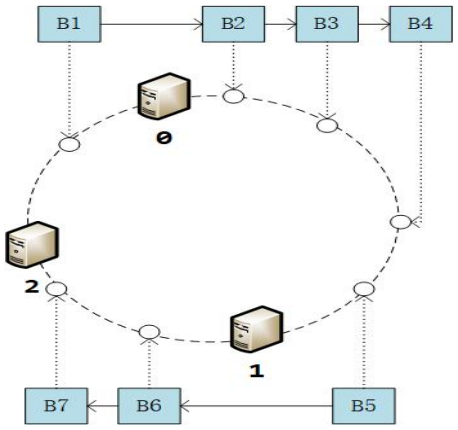

Fig. 2. The relation of block and node

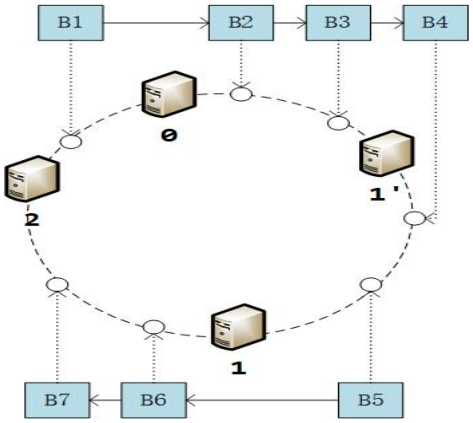

Fig. 3. Joining of new nodes

The type of a new node joining the blockchain network is determined by its storage capacity and storage willingness. If it has enough storage space and is willing to take full responsibility, it will be a full node; Otherwise, the role of the newly added node is that of a node with limited storage resources. When a node with limited storage resources joins, in the traditional way of determining the location by hashing the IP and port number, the successor nodes of the node need to recalculate each hash. Then assign the corresponding part belonging to the new node to the newly added node, and update the routing table. When a node exits, the node's successor node will assume the storage task originally belonging to the exit node. As shown in **Fig. 3**, after the introduction of virtual nodes, the newly added nodes with limited storage resources need to first calculate their performance, and then each node recalculates the proportion of its corresponding virtual node.

According to the above calculation method, the system can calculate the performance of all nodes that choose to join the Chord ring, and then determine the relationship between each physical blockchain node and the virtual node according to the preset number of virtual nodes. As shown in line 3 of **Fig. 4**, the block body $data_b$ is queried according to the block number $blockNo$. In line 4, when a new block is generated, each node calculates the corresponding hash value according to the block body, and the hash value is used as the location $location_b$. In line 5, judge whether $location_b$ is within the scope of its responsibility. If the block address matches the position of the Chord ring where the block is located, the block is stored locally, and at the same time, a notification R and the new block are sent to the succeeding node.

After receiving the notification R, the successor node stores the block locally, and then sends the block and notification $(R-1)$ to the successor node. Then it repeats this step until it is zero, to realize the R redundancy of the block.

Algorithm 1 Algorithm for Block Data Location Determination

Input: $blockNo$, $node$

Output: $result$

```
1: function ShouldStorage(blockNo, node)
2:     init result = false
3:     data_b = getBlockDataByNo(blockNo)
4:     location_b = hash(data_b)
5:     if location_pre < location_b ≤ location_node then
6:         result = true
7:     end if
8:     return result
9: end function
```

Fig. 4. Block storage location determination based on Chord protocol

4.2 Scalable Block Heat Identification Mechanism

Nodes newly added to this model can obtain hot block data from other nodes in the DHT or the IPFS system through the initialization mechanism. Among them, hot blocks represent the most frequently accessed blocks. However, unlike the narrow transactional block chain which is mainly used in the field of digital encryption currency, the generalized transactional blockchain can be applied to many fields, and there are bound to be differences in the block heat identification mechanism in different scenarios.

Therefore, as shown in **Fig. 5** and **Fig. 6**, this model needs a scalable blockchain block heat identification mechanism to meet the needs of different scenarios. DIBS's scalable block heat identification mechanism has a built-in default block heat identification mechanism, and also has the ability to expand, for the customized implementation of the blockchain in specific scenarios, and uses the factory model to obtain the actual block heat identification mechanism.

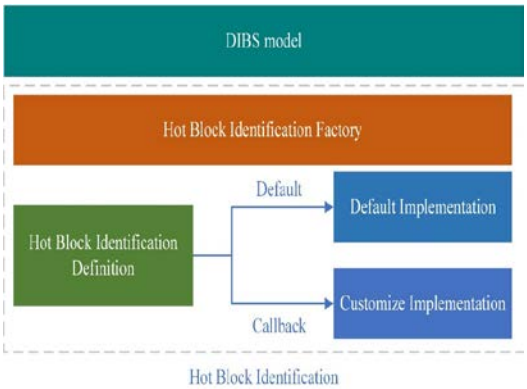

Fig. 5. Scalable heat identification architecture

Algorithm 2 Algorithm for Hot Block Identification

Input: $blockNo$, $gmtCreate$, $Count_{total}$, Ext

Output: $isHot$

```
1: function IsHotBlock(blockNo, F_hot-f, F_hot-e, Ext)
2:     init isHot = false
3:     init impl = ImplementFactory.GetDefault()
4:     if HasCustomizedImplementation() then
5:         impl = ImplementFactory.GetCustomized()
6:     end if
7:     isHot = Execute(impl, blockNo, gmtCreate, Count_total, Ext)
8:     return isHot
9: end function
```

Fig. 6. Block heat identification

Let the block be B, the block number is $blockNo$, the block generation time is $gmtCreate$, the number of blocks in the current blockchain is $Count_{tota}$, and the extension information is Ext.

Then, the block heat $Temp(B)$ can be expressed as：

$$Temp(B) = TempFunction(blockNo(B), gmtCreate(B), Count_{total}, Ext) \quad (3)$$

The default implementation of the heat identification mechanism adopts the identification mechanism applicable to the most classic blockchain system (Bitcoin). Literature [45] analyzed the transaction situation of Bitcoin within a week, and analyzed the distribution of the number of visits to the blockchain block. The most frequently visited block interval is near the latest block and close to the genesis block. Before each node joins the Chord network, the thresholds T_{hot-f} and T_{hot-e} need to be set, which respectively represent the hot block data threshold close to the genesis block and the hot block data threshold close to the latest block.

$$F_{hot-f} = Count_{total} \times T_{hot-f} \quad (4)$$

$$F_{hot-e} = Count_{total} \times T_{hot-e} \quad (5)$$

Among them, F_{hot-f} represents the right boundary of the hot block interval close to the genesis block, and F_{hot-e} is the length of the interval close to the latest block. Therefore, the hot blocks include blocks with block numbers 0 to F_{hot-f} blocks and the latest consecutive F_{hot-e} blocks. Then, the calculation method of block heat $Temp(B)$ is as follows:

$$Temp(B) = \begin{cases} hot & Count_{total} - F_{hot-e} \le blockNo(B) \le Count_{total} \\ cold & other \\ hot & 0 \le blockNo(B) \le F_{hot-f} \end{cases} \quad (6)$$

Fig. 6 shows the block heat identification mechanism algorithm. In line 3, the default block heat identification mechanism is obtained through the mechanism factory $ImplementFactory$ (Formula (6) is its specific implementation). Then use the function $HasCustomizedImplementation()$ in line 4 to determine whether there is a custom extension. If it exists, get the custom implementation in $ImplementFactory$ (formula (3)), as shown in line 5. Finally, in line 7, the $EXECUTE$ function is executed to determine whether the block is a hot block.

4.3 New Node Initialization Mechanism

After a new node joins the model, it needs to perform initialization operations to synchronize historical blocks. The new node synchronizes blocks from DHT or IPFS in this model according to the blocks ' heat attribute. **Fig. 7** is a sequence diagram of the initialization mechanism of the new node.

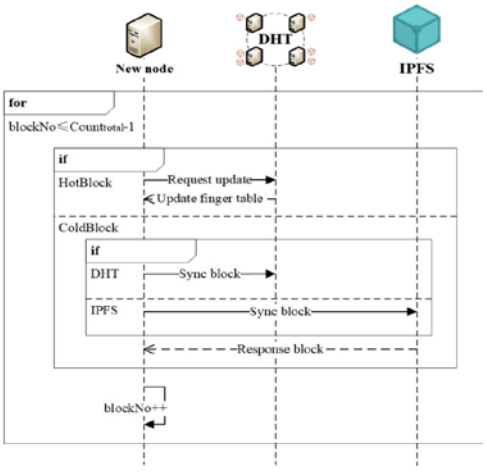

Fig. 7. The initialization of new node

Algorithm 3 Algorithm for New Node Synchronization

Input: $node$, $S_{expected}$

Output: $sync$

1: **function** INIT($node$, $S_{expected}$)
2: init $sync = true$
3: set T_{hot-f}, T_{hot-e}
4: **if** $(T_{hot-f} + T_{hot-e}) \times Count_{total} \geq S_{expected}/avg(B)$ **then**
5: **return** $false$
6: **end if**
7: **for** $blockNo = 0 \rightarrow Count_{total}$ **do**
8: **if** ISHOTBLOCK($blockNo$, $T_{hot-f} \times Count_{total}$, $T_{hot-e} \times Count_{total}$) **then**
9: **if** $shouldStorage(blockNo)$ **then**
10: **if** $isNodeStoraged(neighbors, blockNo)$ **then**
11: $sync$ &= $syncFromDHT(blockNo)$
12: **else**
13: $sync$ &= $syncFromIPFS(blockNo)$
14: **end if**
15: **else**
16: $sync$ &= $updateFingerTable(blockNo)$
17: **end if**
18: **end if**
19: **end for**
20: **return** $sync$
21: **end function**

Fig. 8. The algorithm for the new node initialization

The new node initialization algorithm is shown in **Fig. 8**. When a new node joins this model, it is first necessary to set the size $S_{expected}$ that the node is willing to participate in the local storage of the blockchain and initialize the synchronization state to $true$. Then, set the threshold pre-interval T_{hot-f} and post-interval T_{hot-e} respectively. If $(T_{hot-f} + T_{hot-e}) \times Count_{total} \geq S_{expected}/Avg(B)$, no synchronization is required, as shown on lines 4 and 5. Next, traverse the block number from 0 to $Count_{total}$. As shown in line 8 (see Algorithm 2 for the specific content of $IsHotBlock$), if the current block B is identified as a hot block and happens to be the successor node where the hash value of the block is located, then the block data is requested from the successor node of the node. If the successor node happens to store the block locally, the block data is directly returned to the new node, and update the local finger table ($updateFingerTable()$, in line 16 of Algorithm 3); Otherwise, it finds the block in the Chord ring through the local finger table and returns the block data to the new node ($syncFromDHT()$, in line 11 of Algorithm 3); If all nodes do not store the block locally, then they synchronize from IPFS($syncFromIPFS()$, in line 13 of Algorithm 3). If the current block is identified as a cold block, the new node only needs to request the index of the block in the IPFS system from other nodes in the network.

4.4 Block Data Read and Write Mechanism

In the DIBS model, other nodes in the Chord ring are used to relieve the storage pressure of the node's hot block, and the IPFS system is used to relieve the storage pressure of the node's cold block. Therefore, when there is a request to read a block, if the block is not locally, the request needs to be forwarded to other nodes in the Chord ring through a routing strategy, or the IPFS system is requested to access the cold block. The specific block reading mechanism sequence diagram is shown in **Fig. 9**. When there is a request to write a block, if the local storage is full, a block is replaced by a replacement strategy for the new block to write. The specific block writing mechanism sequence diagram is shown in **Fig. 12**.

(1) Block read mechanism

The algorithm for block reading is shown in **Fig. 10**. When a node receives a read request for a specific block (no indicates the block number), as in line 2 of algorithm 4, the node first adds one to the number of accesses of the block in the local finger table, and then the node finds whether the block exists in the local storage (such as LevelDB). If it exists, execute $readFromLocalStorage()$ directly, and return the block data through local storage, such as line 4; If it does not exist, it will judge whether the requested block is hot block data or cold block data according to the block heat identification mechanism described in the previous section. If it is cold block data (line 7), read the block data file from IPFS; Otherwise, execute $findTargetNodeInDHT()$ (in line 9). It requests the data file from the node with the largest identifier and less than the target block identifier in the local finger table and then repeats this step until the target node is found in the DHT. In line 10 of the algorithm 4, if the target node stores the block, it will obtain the data file through the IPFS system. If it is hot block data, then execute $readFromDHT()$, and it will be directly returned to the requesting node.

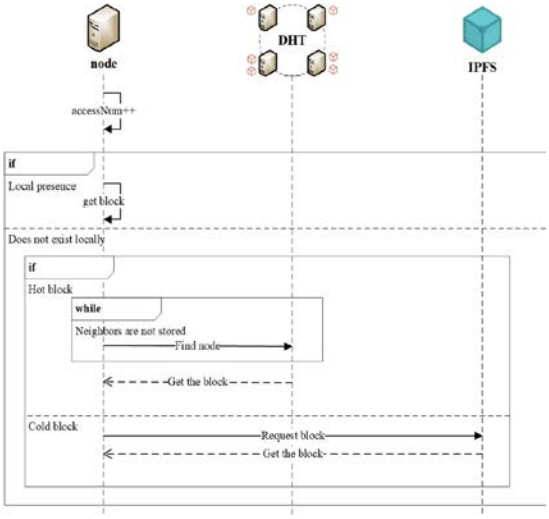

Fig. 9. The process of block reading

Algorithm 4 Algorithm for Block Reading

Input: $node$, no

Output: $data_b$

```
1:  function ReadBlock(node, no)
2:      fingerTable[no] += 1
3:      if hasBlockInLocalStorage(no) then
4:          data_b = readFromLocalStorage(no)
5:      else
6:          if isColdBlock(no) then
7:              data_b = readFromIPFS(no)
8:          else
9:              target = findTargetNodeInDHT(no)
10:             if isNodeStoraged(target, no) then
11:                 data_b = readFromDHT(no, target)
12:             else
13:                 data_b = readFromIPFS(no)
14:             end if
15:         end if
16:     end if
17:     return data_b
18: end function
```

Fig. 10. The algorithm for block reading

(2) Block writing mechanism

In the Bitcoin blockchain network, every time a node executes the consensus algorithm, the miner will generate a new block and broadcast it to all nodes in the entire network. After the node is successfully verified, it will be attached to the end of the local blockchain, and the block height will be increased by one. The above process is the full-node high-redundancy storage strategy adopted by the Bitcoin blockchain, and each node needs to store new blocks locally. When a node in our scheme joins the blockchain network, it will pre-set the size $S_{expected}$ that the node is willing to participate in the local storage of the blockchain. Obviously, as time passes, new blocks are constantly added to the end of the blockchain, and the storage size required by the node will gradually exceed the originally set $S_{expected}$. Therefore, a mechanism needs to be set up to ensure that each node stores the most important hot block data in the limited and precious local storage space. Our solution uses the Block Least Frequently Used (BLFU) algorithm to dynamically replace the blocks in the node's local storage to save space for new block storage. The block replacement algorithm is implemented as shown in **Fig. 11**. First, set the block number to be replaced as $replaceIndex = -1$ in line 3. Then, in line 4 of the algorithm, it traverses $localHotBlockList$ to find the number of the least used hot block (i.e. lines 5 to 7 of the algorithm) as $replaceIndex$.

Algorithm 5 Algorithm for BLFU

Input: $fingerTable$

Output: $replaceIndex$

```
1:  function GetReplaceBlockNoByBLFU(fingerTable)
2:      init minUseNum = INT.MAX
3:      init replaceIndex = -1
4:      for data_b ∈ localHotBlockList do
5:          if fingerTable[blockNo(data_b)] < minUseNum then
6:              minUseNum = fingerTable[blockNo(data_b)]
7:              index = blockNo(data_b)
8:          end if
9:      end for
10:     return replaceIndex
11: end function
```

Fig. 11. The algorithm for block replacement

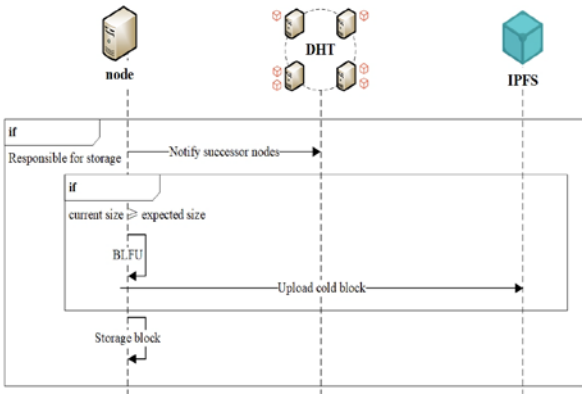

Fig. 12. The process of block writing

The block writing algorithm is shown in Fig.13. In this scheme, each node verifies the new block. If the verification is successful, the node needs to make a storage judgment on the new block. The storage method based on the Chord protocol does not require every node to store the same data content. The node first calculates the hash value of the new block according to the Chord protocol and determines whether the block belongs to the position between the predecessor node and itself (line 3 of the algorithm). If it is, then the node will be responsible for the local storage task of this block. According to the preset redundancy number R, the node executes $notifyStroageToNext(R-1)$ (in line 4). The node needs to send a storage notification to the successor node, and the successor node performs the same operation until the R redundancy is completed. If the local storage size is still larger than the expected value $S_{expected}$ at this time, execute the $GetReplaceBlockNoByBLFU$ (in line 6), and delete the block locally. Finally, store the new block locally as shown in line 9.

Our solution uses a method of locally storing hot block data. Obviously, the newly generated block must be hot block data. Therefore, after the node determines that it needs to store a new block, it will determine whether the current local storage size has exceeded the threshold set in the initialization phase. If it is exceeded, the hot block replacement algorithm is executed to replace the hot block that has been most rarely accessed recently. Then upload it to the IPFS system, and store the new block locally.

Algorithm 6 Algorithm for Block Writing

Input: $data_b$, R

```
1:  function WriteBlock(node, no)
2:      no = blockNo(data_b)
3:      if ShouldStorage(no, node) then
4:          notifyStroageToNext(R − 1)
5:          if getCurrentStorageSize(node) ≥ S_expected then
6:              replaceBlockNo = GetReplaceBlockNoByBLFU(fingerTable)
7:              deleteFromLocal(replaceBlockNo)
8:          end if
9:          storageInLocal(node, no)
10:     end if
11: end function
```

Fig. 13. The algorithm for block writing

5. Experiment and Analysis

The experimental environment of this paper is on a server with Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz CPU and 32G memory. With the help of Hyperledger Fabric 1.0, a scalable blockchain storage model based on DHT and IPFS is realized.

Our solution sets up 5 blockchain nodes (node-1~5) with different performances to simulate and test the DIBS model. According to formula 1 and formula 2, the number of virtual nodes corresponding to nodes in the DHT network is affected by four factors （Sm_i， Com_i， Sw_i， W_i）. This paper mainly studies the storage optimization problem of blockchain. Therefore, this paper considers Sm_i and Sw_i as the main influencing factors, and Com_i and W_i as secondary factors. If the coefficients of the secondary factors are set relatively large, the experimental results will not reflect the real effect. Therefore, the parameter selection in this

paper sets the coefficients of the main factors to be relatively large. Set the weight coefficient k_1 of the performance of the blockchain node to 0.4, k_2 to 0.1, k_3 to 0.4, and k_4 to 0.1. Set 16 virtual nodes, and select the main frequency data to represent the computing power of the node. The parameter settings of blockchain nodes are shown in **Table 1**.

Table 1. The settings of blockchain nodes

Blockchain node name	Number of corresponding virtual nodes	Maximum storage capacity	Calculate ability	Affordable storage size	Network bandwidth
node-1	3	800M	2.50GHz	750M	10Mbps
node-2	2	600M	2.50GHz	500M	5Mbps
node-3	4	1000M	2.50GHz	1000M	10Mbps
node-4	3	750M	2.50GHz	750M	5Mbps
node-5	4	1050M	2.50GHz	1000M	10Mbps

As shown in **Fig. 14**, when the redundancy number of the blockchain DHT network is set to $R = 2$, that is, in addition to the node to which the block originally belongs, the subsequent (R-1)=1 nodes are all redundantly stored. When the number of blocks is 500, the current average storage size $avg(\mathrm{bc-node})$ of the bc-node using the literature [45] model is 546.5M, while the node storage consumption of the DIBS model using our scheme is significantly smaller. When the number of blocks is small, our scheme is the same as the scheme in literature [45], and the storage consumption of nodes using the DIBS storage model increases linearly. When the number of blocks reaches a certain amount (greater than 500), local storage in our scheme is no longer increased. This is because the Chord protocol is adopted in our scheme to disperse storage pressure, and the BLFU block replacement algorithm is used to save space for local storage. In this way, when a node needs to store new blocks, it checks whether the current local storage capacity exceeds the threshold set during initialization. If the number of hot blocks is exceeded, BLFU is used to replace the hot blocks that are least frequently accessed recently, and the new blocks are stored locally so that the local storage does not increase linearly.

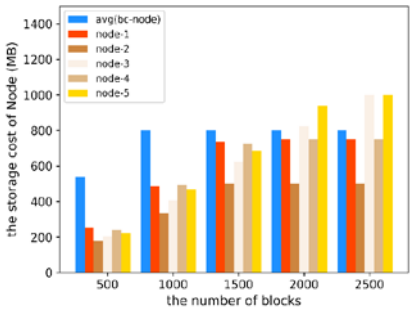

Fig. 14. The node storage when $R = 2$

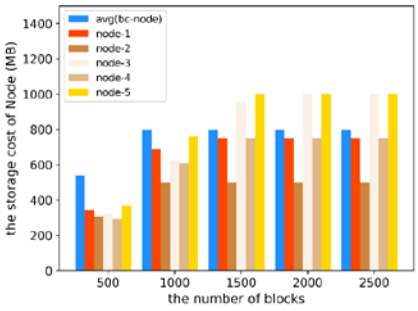

Fig. 15. The node storage when $R = 3$

When the redundancy number R=3 is set, the node storage consumption using the DIBS model and the model in literature [45] is shown in **Fig. 15**. Compared with the literature [45] and the full-node storage model, our scheme occupies a smaller node storage space in the case of the same number of blocks. When the local storage reaches the preset upper limit, the BLFU block replacement algorithm will be triggered, and the cold block data will gradually be transferred to the IPFS system. Therefore, local storage will no longer continue to grow linearly, but will gradually stabilize. The relationship between the number of blocks and the storage capacity of nodes is shown in **Fig. 16**.

According to the results, when the number of blocks reaches a certain amount (greater than 1000), the local storage capacity stops growing and tends to be stable. Therefore, compared with literature [45], when the number of blocks is large, our DIBS model saves the local storage space more and ensures that each node stores the hot block data that should be stored most in the limited and precious local storage space, thus improving the query efficiency.

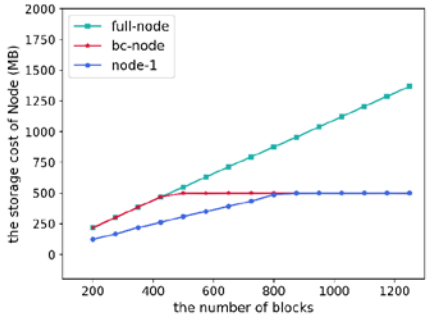

Fig. 16. The comparison of node storage capacity

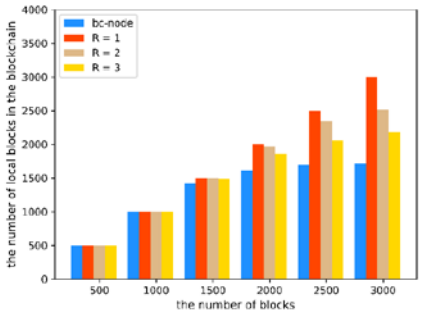

Fig. 17. The number of locally stored blocks at the same block height

As shown in **Fig. 17**, under the same block height, the number of blocks stored locally in the blockchain network node of this scheme (when R is set to 1, 2, and 3 respectively) is greater than or equal to the number of blocks stored by the bc-node [45]. This result is due to the decentralized storage of DHT in our solution, rather than the full node high redundancy storage strategy adopted by Bitcoin. In this scheme, after determining the need to store new blocks, the node will judge whether the current local storage size has exceeded the threshold. If so, the BLFU algorithm will be used to replace the cold block data to provide space for storing hot blocks. Therefore, for the same total storage capacity, our scheme can store more hot block data locally.

Table 2. The comparison of schemes

Storage model	Main technique	Block atomicity	Auto-adjustability	Storage consumption	Block query time complexity
literature [33]	Shamir threshold	N	-	S_{btc}/k	$O(n)$
literature [34]	Network coding	N	N	S_{btc}/k	$O(n)$
literature [36]	Low-Density Check Code	N	N	S_{btc}/k	$O(n)$
literature [37]	Safe regeneration code	N	N	S_{btc}/k	$O(n)$
literature [45]	IPFS	Y	N	$\min\{S_{btc}, S_{expected}\}$	$O(1)$
DIBS Model	Chord+IPFS	Y	Y	$\min\{S_{btc}R/n, S_{expected}\}$	$O(\log n)$

Table 2 summarizes 6 models to solve the scalability problem of blockchain storage. Compared with many models that adopt threshold secret sharing and encoding methods (the storage consumption is S_{btc}/k, where S_{btc} is the consumption when using the full-node storage model, and k is the threshold in the secret sharing mechanism or the number of data blocks in the encoding mechanism), the DIBS storage model does not split block data and does not destroy the atomicity of blocks in the blockchain [7]. From the perspective of block query, for the full-node high-redundancy storage model, all nodes are stored in the local key-value database, and the query efficiency is $O(1)$. For the storage model that cannot guarantee the

atomicity of the block, it is necessary to request n nodes (the block is divided into n parts and distributed to n nodes), so the query time complexity is $O(n)$. In our solution, the node is not only a node of the blockchain but also a node of the Chord network, so the query time complexity is only $O(\log n)$ instead of $O(n)$, and the query efficiency is higher.

6. Conclusion

Based on the analysis of current collaborative storage methods for generalized transactional blockchains, this paper proposes a scalable blockchain storage model (the DIBS model). This model builds a DHT network based on the Chord protocol, according to factors such as node hardware capabilities and storage willingness. So that blockchain nodes can jointly maintain a complete blockchain ledger and reduce the burden of local storage on nodes. At the same time, according to how frequently the block data is queried, the heat of the block data is identified. It is found from the results that when the number of blocks reaches a certain amount, the BLFU replacement algorithm adopted in our model will be triggered to effectively replace the cold block data into the IPFS system under the blockchain, making the storage cost tend to be stable and avoiding the continuous linear growth of the local storage cost of nodes. Because the proposed DIBS model adopts DHT for decentralized storage rather than the high redundancy storage mode of the full node, the storage pressure of resource-constrained nodes is significantly reduced. Therefore, with the same storage consumption, the blockchain network in our solution can store more hot block data, which improves the efficiency of block query. This paper discusses the problem of optimizing the storage cost of the blockchain from the perspective of the network layer and the data layer. In the future, we plan to study the blockchain consensus mechanism that can reduce storage space. It enables the nodes in the blockchain to efficiently and collaboratively store block data in the process of reaching a consensus with each other, so as to achieve the purpose of alleviating the storage pressure of blockchain nodes. At the same time, from a security point of view, we are going to study security mechanisms based on attribute encryption, homomorphic encryption and other technologies. The purpose is to further improve the security of off-chain storage and on-chain storage, and to ensure the data security when applying blockchain storage in various fields.

References

[1] S. Nakamoto, “Bitcoin: a peer-to-peer eletronic cash system,” 2009. [Online]. Available: http://bitcoin.org/bitcoin.pdf.

[2] G. Wood, “Ethereum: a secure decentralised generalised transaction ledger,” 2015. [Online]. Available: http://gavwood.com/Paper.pdf.

[3] Y. Yuan and F.Y. Wang, “Blockchain: the state of the art and future trends,” *Acta Automatica Sinica*, vol. 42, no. 4, pp. 481-494, Apr. 2016. Article(CrossRef Link)

[4] X.H. Zhang, B.N. Niu and T. Gong, “Account-based blockchain scalable storage model,” *Journal of Beijing University of Aeronautics and Astronautic*, vol. 48, pp. 708-715, 2022. Article(CrossRef Link)

[5] G.A. Oliva, A.E. Hassan and Z.M.J. Jiang, “An exploratory study of smart contracts in the Ethereum blockchain platform,” *Empirical Software Engineering*, vol.25, no.2, pp.1864–1904, May,2020. Article(CrossRef Link)

[6] Q. Zheng, Y. Li, P. Chen and X.H. Dong, “An innovative IPFS-based storage model for blockchain,” in *Proc. of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, IEEE, pp. 704-708, Dec. 2018. Article(CrossRef Link)

[7] Z.X. Sun, X. Zhang and L. Chen, "Survey of storage scalability on blockchain," *Journal of Software*, vol. 32, no.01, pp.1-20, Jan. 2021. Article(CrossRef Link)
[8] D. Frey, M.X. Makkes, P.L. Roman, F. Taïani and S. Voulgaris, "Bringing secure bitcoin transactions to your smartphone," in *Proc. of the 15th International Workshop on Adaptive and Reflective Middleware*, pp. 1-6, Dec. 2016.Article(CrossRef Link)
[9] A. Palai, M. Vora and A. Shah, "Empowering light nodes in blockchains with block summarization," in *Proc. of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, pp. 1-5, Feb. 2018. Article(CrossRef Link)
[10] R. Norvill, B. BF. Pontiveros, R. State and A. Cullen, "IPFS for reduction of chain size in Ethereum," in *Proc. of 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, pp.1121-1128, Aug. 2018. Article(CrossRef Link)
[11] T. Huang, "Application and challenge of blockchain in supply chain finance," in *Proc. of International conference on Big Data Analytics for Cyber-Physical-Systems*, Springer, Singapore, pp.1372-1377, Sep. 2020. Article(CrossRef Link)
[12] M. Kowalski, Z. W. Y. Lee and T. K. H. Chan, "Blockchain technology and trust relationships in trade finance," *Technological Forecasting and Social Change*, vol.166, p.120641, May. 2021. Article(CrossRef Link)
[13] L. Zhang, Y. Xie, Y. Zheng, W. Xue and X.R. Zheng, "The challenges and countermeasures of blockchain in finance and economics," *Systems Research and Behavioral Science*, vol. 37, no. 4, pp. 691-698, Jun. 2020. Article(CrossRef Link)
[14] G. Wang, Z. Shi, M. Nixon and S. Han, "ChainSplitter: Towards Blockchain-Based Industrial IoT Architecture for Supporting Hierarchical Storage," in *Proc. of 2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 166-175, Jul. 2019. Article(CrossRef Link)
[15] A. Ekramifard, H. Amintoosi and A.H. Seno, "A systematic literature review on blockchain-based solutions for IoT security," in *Proc. of the 7th International Conference on Contemporary Issues in Data Science*, Springer, Cham, pp. 311-321, Jan. 2019. Article(CrossRef Link)
[16] A. Ponza, S. Scannapieco, A. Simone and C. Tomazzoli, "Envisioning the digital transformation of financial documents: a blockchain-based bill of exchange," in *Proc. of International Congress on Blockchain and Applications*, Springer, Cham, pp. 81-90, Jul. 2020. Article(CrossRef Link)
[17] C. Wei, Q. Wang and C. Liu, "Research on construction of a cloud platform for tourism information intelligent service based on blockchain technology," *Wireless Communications and Mobile Computing*, no. 2, pp.1-9, Oct. 2020. Article(CrossRef Link)
[18] Z. Xu, L. Wei, J. Wu and C.N. Long, "A blockchain-based digital copyright protection system with security and efficiency," in *Proc. of China Blockchain Conference*, Springer, Singapore, pp. 34-49, Jan. 2021. Article(CrossRef Link)
[19] C.Q. Zhao, M.Z. Liu, Y.H. Yang, F.X. Zhao and S.J. Chen, "Toward a blockchain based image network copyright transaction protection approach," in *Proc. of International Conference on Security with Intelligent Computing and Big-data Services*, Springer, Cham, pp.17-28, Jan. 2020. Article(CrossRef Link)
[20] S.M. Pournaghi, M. Bayat and Y. Farjami, "MedSBA: a novel and secure scheme to share medical data based on blockchain technology and attribute-based encryption," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 4613-4641, Nov. 2020. Article(CrossRef Link)
[21] S.K. Dwivedi, R. Amin, S. Vollala and R. Chaudhry, "Blockchain-based secured event-information sharing protocol in internet of vehicles for smart cities," *Computers & Electrical Engineering*, vol.86, p.106719, Jun. 2020. Article(CrossRef Link)
[22] T.K. Agrawal, V. Kumar, R. Pal, L.C. Wang and Y. Chen, "Blockchain-based framework for supply chain traceability: a case example of textile and clothing industry," *Computers & industrial engineering*, vol.154, p.107130, Jan. 2021. Article(CrossRef Link)
[23] S. Xu, X. Zhao and Z. Liu, "The impact of blockchain technology on the cost of food traceability supply chain," in *Proc. of IOP Conference Series: Earth and Environmental Science*. IOP Publishing, vol.615, p.012003, Dec. 2020. Article(CrossRef Link)

[24] J. Poon and T. Dryja, "The bitcoin lightning network: scalable off-chain instant payments," 2016. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf
[25] G. Zyskind, O. Nathan, "Decentralizing privacy: using blockchain to protect personal data," in *Proc. of 2015 IEEE Security and Privacy Workshops*, pp.180-184, May. 2015. Article(CrossRef Link)
[26] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. of International Workshop on Peer-to-Peer Systems*, Berlin, Heidelberg, pp.53-65, Oct. 2002. Article(CrossRef Link)
[27] R. Li, T. Song, B. Mei, H. Li, X. Z. Cheng and L. M. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, vol. 12, no.5, pp. 762-771, Sept.-Oct. 2019. Article(CrossRef Link)
[28] Y. Mansouri and M. A. Babar, "A review of edge computing: features and resource virtualization," *Journal of Parallel and Distributed Computing*, vol.150, pp.155-183, Apr. 2021. Article(CrossRef Link)
[29] M. S. Ali, K. Dolui and F. Antonelli, "IoT data privacy via blockchains and IPFS," in *Proc.of the seventh international conference on the internet of things*, pp.1-7, Oct. 2017. Article(CrossRef Link)
[30] M. Klems, J. Eberhardt, S. Tai, S. Härtlein, S. Buchholz and A. Buchholz, "Trustless intermediation in blockchain-based decentralized service marketplaces," in *Proc. of International Conference on Service-Oriented Computing*, Cham, pp.731-739, Oct. 2017. Article(CrossRef Link)
[31] Q. Xu, Z. Song, R. S. M. Goh and Y. J. Li, "Building an ethereum and IPFS-based decentralized social network system," in *Proc.of 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 1-6, Dec. 2018. Article(CrossRef Link)
[32] G. He, W. Su and S. Gao, "Chameleon: a scalable and adaptive permissioned blockchain architecture," in *Proc. of 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pp. 87-93, Aug. 2018. Article(CrossRef Link)
[33] G. C. Zhang and R. J. Wang, "Blockchain shard storage model based on threshold secret sharing," *Journal of Computer Applications*, vol. 39, no.9, pp. 2617-2622, Sep. 2019. Article(CrossRef Link)
[34] M. Dai, S. Zhang, H. Wang and S. Jin, "A low storage room requirement framework for distributed ledger in blockchain," *IEEE Access*, vol.6, pp. 22970-22975, Mar. 2018. Article(CrossRef Link)
[35] Y. Kaneko and T. Asaka, "DHT clustering for load balancing considering blockchain data size," in *Proc. of 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, pp. 71-74, Nov. 2018. Article(CrossRef Link)
[36] H. H. Wu, A. Ashikhmin, X. D. Wang, C. Li, S. C. Yang and L. Zhang, "Distributed error correction coding scheme for low storage blockchain systems," *IEEE Internet of Things Journal*, vol.7, no.8, pp. 7054-7071, Mar. 2020. Article(CrossRef Link)
[37] D. S. Gadiraju, V. Lalitha and V. Aggarwal, "Secure regenerating codes for reducing storage and bootstrap costs in sharded blockchains," in *Proc. of 2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 229-232, Nov. 2020. Article(CrossRef Link)
[38] Y. Doi, S. Wakayama and S. Ozaki, "A design for distributed backup and migration of distributed hash tables," in *Proc. of 2008 International Symposium on Applications and the Internet*, pp. 213-216, Jul. 2008. Article(CrossRef Link)
[39] I. Stoica, R. Morris, D. Liben-nowell, M. F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on networking*, vol.11, no.1, pp.17-32, Feb. 2003. Article(CrossRef Link)
[40] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, Berlin, Heidelberg, pp. 329-350, Nov. 2001. Article(CrossRef Link)
[41] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable content-addressable network," in *Proc. of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 161-172, Oct. 2001. Article(CrossRef Link)

[42] Y. Chen, H. Li, K. Li and J. Zhang, "An improved P2P file system scheme based on IPFS and Blockchain," in *Proc. of 2017 IEEE International Conference on Big Data (Big Data)*, pp. 2652-2657, Dec. 2017. Article(CrossRef Link)
[43] W. Mazurczyk and P. Kopiczko, "Understanding BitTorrent through real measurements," *China Communications*, vol. 10, no. 11, pp.107-118, Nov. 2013. Article(CrossRef Link)
[44] D. Spinellis, "Git," *IEEE software*, vol. 29, no.3, pp.100-101, Jun. 2012. Article(CrossRef Link)
[45] I. T. Chou, H. H. Su, Y. L. Hsueh and C. W. Hsueh, "BC-Store: a scalable design for blockchain storage," in *Proc. of the 2020 2nd International Electronics Communication Conference*, pp. 33-38, Jul. 2020. Article(CrossRef Link)

Lu Chen received the B.Eng. degree in network engineering from Nanjing University of Posts and Telecommunications, Nanjing, China. She is currently pursuing the M.D.-Ph.D. degree in information network from the Nanjing University of Posts and Telecommunications. Her primary research interests are in cloud storage, network security, and blockchain technology.

Xin Zhang received his B.Eng. degree from Nanjing University of Posts and Telecommunications, Nanjing, China. He is a master student of Nanjing University of Posts and Telecommunications. His main research area is blockchain storage technology and application.

Zhixin Sun is the dean of School of Modern Posts, Nanjing University of Posts and Telecommunications. He received his PHD degree in Nanjing University of Aeronautics and Astronautics, China in 1998 and worked as a post doctor in Seoul National University, South Korea between 2001 and 2002. He has published more than 90 literatures on journals worldwide. His research area includes information security, computer networks, computer science, etc.