A Collaborative Ledger Storing Model for Lightweight Blockchains based on Chord Ring

ZiXiang Nie (niezixiang@bupt.edu.cn)
Beijing University of Posts and Telecommunications
Jin Li
Beijing University of Posts and Telecommunications
FengHui Duan
Beijing University of Posts and Telecommunications
Yueming Lu
Beijing University of Posts and Telecommunications

Research Article

Keywords: Lightweight Blockchains, Ledger Structure, Asynchronous Consensus, Distributed Hash Table

Posted Date: August 17th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3254799/v1

Additional Declarations: No competing interests reported.

A Collaborative Ledger Storing Model for Lightweight Blockchains based on Chord Ring

ZiXiang Nie[1*], Jin Li[1†], FengHui Duan[1†], Yueming Lu[1†]

[1*]Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing University of Posts and Telecommunications, Xitucheng, Haidian, 100876, Beijing, China.

*Corresponding author(s). E-mail(s): niezixiang@bupt.edu.cn;
Contributing authors: li_jin@bupt.edu.cn; duanfh@bupt.edu.cn;
ymlu@bupt.edu.cn;
†These authors contributed equally to this work.

Abstract

A Blockchain is one distributed ledger system, and keeps the high redundancy of ledger copies to make the assurance of network security. However, the continuously duplicated full copies also impose a tremendous amount of demand on some nodes for data storage. The development of blockchain technologies in the IoT(Internet of Things) application scenario is hampered by the restricted storage capacities of terminal devices used in the IoT edge computing scenario, which makes it difficult to load the full copy with infinite growth. Our paper suggests a collaborative ledger storing model based on *Chord Ring* to address the issues with lightweight blockchains in data storage. On-chain blocks are split by *Chord Ring* structure and stored in various node clusters in a decentralized manner, and off-chain blocks at various levels are provided with PoW(Proof of Work) consensus asynchronously and archived centrally to the cloud storage on a regular basis. The theoretical and experimental analysis indicates that this model can reduce the data storage redundancy of blockchains while ensuring the high availability of data and the high decentralization of the network.

Keywords: Lightweight Blockchains, Ledger Structure, Asynchronous Consensus, Distributed Hash Table

1 Introduction

A blockchain is a distributed ledger technology with consensus mechanism, cryptography, peer-to-peer network, smart contracts, etc., maintains a complete and consistent copy of data on each node of the system, in contrast to conventional distributed ledger technology which splits data for storage. This serves to prevent the entire ledger from being destroyed on the network, which means not possible to destroy blockchain ledger as long as sufficient nodes contribute to the upkeep of the system. The more copies of the whole ledger there are, the more redundant data there is, and the more secure, stable, and tamper-resistant the blockchain becomes. Since the length of blockchain ledgers are constantly increasing, new nodes require hundreds of gigabytes of storage space in order to even participate in the maintenance of the blockchain. This strategy, however, ignores the massive amount of duplicate data storage when network nodes expand quickly.

Data volume of both Bitcoin[1] and Ethereum[2] surpassed 500GB by 2023 and keep expanding quickly, meaning any additional node that wishes to join the two blockchains would face a significant storage overhead as a result of this volume of data. The idea of "edge computing" has been put forth with the miniaturization of hardware and containerization of software development environments. In the age of the IoT, end devices are no longer just sensors that can gather and upload information and clients that request services from the central server, but also self-organizing nodes with computational power that are capable of carrying out some computing activities, such as data processing, caching, device management, privacy protection, etc. The concept of decentralization, in which end devices become data consumers as well as data providers, coincides with the idea of blockchains and has garnered a lot of interest from both academic and industry. The current blockchain and IoT integration can be separated into functional and application-oriented categories, with the functional category concentrating mostly on Data Management[3], System Security[4], Identity Verification[5], Privacy Protection[6], Access Control[7], etc. and the application-oriented mainly focuses on Internet of Vehicles[8], Industry 4.0[9], Smart City[10] and Supply Chain Management[11].

A Blockchain have a triadic paradox: it can only achieve two of decentralization, security, and scalability. The IoT blockchain senario prioritizes scalability, transaction performance, and throughput compare to the financial blockchain senario. However, the unlimited growth of ledger data and highly redundant storage strategy will put huge storage pressure on edge computing devices, and the storage scalability of blockchains is a major issue that limits application implementations. The blockchain lightweight ledger structure which can reduce the data storage pressure and improve the scalability have been studied. The feasible research ideas are broadly divided into two types: one is to realize the lightweight on-chain, such as reducing the storage pressure of blockchain nodes with ledger slicing[12], light nodes[13], ledger coding[14], etc., and the other is to transfer part of block data to off-chain external storage, such as combining IPFS(InterPlanetary File System)[15], DHT(Distributed Hash Table)[16], and other cloud storage technology to store most blockchain data. However, both methods have downsides and this study presents a collaborative ledger storing model

with both on-chain way and off-chain way, the innovative work in our paper are mainly included in the following three aspects:

1.We propose an on-chain distributed storing model and a route addressing mechanism for on-chain blocks based on *Chord Ring* structure for lightweight blockchains. Each node in the blockchain network only keeps a portion of the blocks and has flexible control over the parameters that determine how redundant the data is, ensuring the network's decentralization and reduced node data storage stress. Furthermore, we design a multi-level node relationship. A node that has just joined the network can only save ledger data and generate transactions, and according to its resource abundance in the IoT, it may be responsible for managing a node cluster or the whole blockchain network in the subsequent time, increasing the scalability of the blockchain.

2.We propose an off-chain blockchain ledger periodic uploading cloud strategy based on asynchronous consensus to divide blocks into blocks of three different security levels: minute blocks, hour blocks, and day blocks. Asynchronous consensus represents that the generation of blocks with different security levels is asynchronous, which could reduce the transaction throughput pressure caused by node size growth. In addition, the combination of asynchronous consensus and regular cloud upload strategy ensures that blockchain data is highly available and high TPS(Transaction Per Second) without sacrificing security and tamper resistance.

3.We combine the on-chain and the off-chain blockchain ledger data lightweight methods and simulating a IoT edge computing scenario, and design the relevant experiments. We make statistics on the distribution uniformity of nodes and blocks. The experimental results show that nodes and blocks can be randomly and uniformly distributed in the network topology according to the algorithm. Further more, the experiments on the nodes of the blockchain in reducing data storage pressure are also counted to prove the feasibility of this design.

2 Related Work and Background Knowledge

We divide this section into two parts. The first part introduces how to realize the lightweight of blockchain ledgers in the previous work. Referring to the previous work, we choose the DHT architecture as the basis of the blockchain network topology in this article, so in the second part, we will introduce the background of DHT.

2.1 Lightweight and scalability of blockchain ledger architecture

In recent years, research has focused on enhancing the scalability of blockchain, studying the lightweight of its ledger, and lowering the redundancy of data storage across the network. The primary categories from an on-chain perspective are ledger sharding, light nodes, ledger splitting, ledger coding. From the off-chain perspective, it mainly relies on external storage to realize blockchain data transfer.

Ledger sharding is one of the popular methods for scaling a blockchain that divides the original blockchain network into numerous shards, with each shard's transactions being independently handled by a cluster of nodes using a parallel verification strategy. For example, Elastico[12] interposes blockchain processes with the idea of "slice

epochs", where each round of epochs ends with a random variable calculated for the next round of random slicing. However, secondary slicing stops almost any node in the entire blockchain network, significantly reducing blockchain performance in terms of time and overhead. SSChain[17] adopts a market incentive policy where nodes can freely migrate between different shards, thereby dynamically adjusting the degree of computing power balance in each shard. The problem with lightweight solutions based on ledger sharding is that most of the considerations are to improve the overall data load capacity of the system without reducing the load on individual nodes. Regular restructuring of shards will bring corresponding data migration loads, and the interaction between shards will bring additional system complexity and security issues.

The term "light node" refers to a node that only keeps the block header and can send transactions but cannot independently verify them, and it originates from the SPV(Simplified Payment Verification) protocol[1], a financial-oriented design concept. When a light node needs to confirm a transaction, it must request data from a full node (i.e., a node that houses the entire blockchain ledger and data with full functionality) and use hash functions like Merkle Root to check the consistency of the data. Frey D[13] suggests a more intricate consensus model with the goal of lessening the reliance of light nodes on full nodes, and divides the full nodes into five categories of roles that carry out the mining, storing, and verifying tasks. In addition to full nodes, light nodes can also ask DHT nodes for data. For light nodes with limited resources, the protocol does lessen the degree of dependence on the full node, but it only shifts the dependency to the DHT node, which is actually still a design with a significant external dependency. Kim[18] adopts a more compromise block compression method, which uses a selective compression scheme to quickly validate data and prevent compressed block stacking by updating partial checkpoints. Although lightweight solutions based on lightweight nodes can reduce the storage load of a single node, they completely store data in the entire node, resulting in low data independent verification ability, low security, low throughput, and high degree of centralization.

By moving data from the chain to external cloud storage, it is also possible to increase the scalability of the blockchain ledger storage while simultaneously easing the load on blockchains and achieving the lightweight nature of block ledgers. It is common practice to use the IPFS system, for instance, Zheng[19] stores validated transactions in IPFS and uses its ability to distinguish between transactions that have been verified by the presence or absence of the file's index hash in the block as evidence. Mani[20] designs a Hyperldger combined with IPFS transaction verification model. In this model, blockchain miners will first check whether there is a hash of the transaction locally when verifying the transaction. If it does not exist, they will download the transaction data uploaded from other nodes in IPFS to quickly filter verified transactions and improve efficiency. Sarathchandra[21] constructs a social media application that combines Ethereum and IPFS implementation, transferring large file data that is difficult to load on the blockchain to IPFS for storage. Only user metadata information is stored on the chain, reducing the storage load on Ethereum when creating smart contracts. Ali[22] establishes a three-layer architecture consisting of blockchain, DHT, and IPFS. Blockchain is responsible for permission management and access

control, DHT stores data reference information, and IPFS stores data entities, reducing the storage load of blockchain through a three-layer architecture. Hassanzadeh[23] proposes a separated storage blockchain ledger that stores data entities in DHT and saves data references in the blockchain.

In conclusion, the research on blockchain ledger storage lightweight can be separated as on-chain and off-chain. Ledger sharding's communication loss and resharding problem, light nodes' external dependency issue, and ledger coding's computational burden are some of the drawbacks of on-chain approaches. The off-chain external cloud storage based solution will also result in additional security concerns due to the level of trust with external cloud service providers. Another problem is that the users will no longer be able to view data directly from the chain; instead, they must go through a cloud download-verification procedure, which increases the system delay.

2.2 Distributed Hash Table and Route Addressing

Structured P2P(Peer-to-Peer) models are suggested as a solution to the issues of blindness, inefficiency, low accuracy, and high information redundancy in traditional unstructured resource discovery. The design concept is to use the compatibility hash algorithm to map the index of resources to a structured overlay network, and then to design routing tables and indexing algorithms between nodes to achieve accurate and quick location of resources. The unidirectivity, anti-collision, and load balancing of the hash algorithm, which ensures the uniqueness and balance criteria of the structured P2P model for resource address distribution, are the foundation for the implementation of DHT technology. Pastry[24], Tapestry[25], and Chord[26] are typical designs of DHT-based P2P models, with Chord being the most popular due to its design's simplicity, accuracy, and aesthetic appeal.

Chord assigns an m-bit identifier to each node and resource according to the SHA-1 algorithm, called NodeID and ObjectID respectively. Each node and resource is arranged clockwise according to its ID on a circular ring with address space 2^m . This ring is called *Chord Ring*, which is an overlay network built on top of the IP network. Each node maintains its own routing table (called finger table) containing the information of m nodes that are close to its neighbors. Usually, when a node receives a resource index, it does not find the corresponding resource node directly from its own routing table, but keeps narrowing down the storage range of the target resource with a dichotomy like jumping search, and finally finds it by a *successor node*.

Because the concepts of DHT using hash value to locate data resources and blockchains using hash value to determine the uniqueness of blocks are similar, there is a chance to merge the two. The concept for this paper is inspired by the topology of *Chord Ring*: blocks are distributedly stored in the corresponding nodes on the *Chord Ring* according to their own hash values, and the IP address of a node in the blockchain network is used to generate its location identification on the *Chord Ring*. Our paper also suggests a two-layer cluster structure based on the *Chord Ring* to address the issues of communication overhead and service delay brought on by nodes frequently joining or leaving the network. The specific network topology design and associated algorithms will be covered in more detail in the following chapters.

3 The Collaborative Storing Model Combining On-Chain and Off-Chain

In this section, we take into account the two aforementioned research concepts and propose an collaborative storing model combining the on-chain method and off-chain method for lightweight blockchains. We also provide a flow chart that includes the process from the establishment to the normal operation of the model.

On the chain, this model uses the *Chord Ring* to divide nodes into clusters. Nodes within a same cluster no longer store full ledger data, but jointly maintain a portion of the ledger data, which all the portions then combine to form the entire blockchain data. In addition, the roles of the blockchain nodes are divided into four types including emphleader nodes, *vice-leader nodes*, *Successor nodes*, and *follower nodes*, which helps improve the efficiency of this model. Comparing to the light node model, this design retains the ability of nodes to independently verify data, providing higher security, and has the advantage of decreasing communication loss between nodes due to sharding and relieving the full node of this responsibility that the ledger sharding paradigm unpossessed.

Off the chain, an asynchronous consensus based upload cloud storage strategy is employed. The asynchronous consensus represents that blocks are hierarchical with different security levels, and different generation intervals are given according to different levels. The higher the level, the longer the generation interval, and the more stable and secure the block. High level blocks include low-level blocks, and the highest level blocks are regularly uploaded to external cloud storage for persistence. Most blockchain nodes will erase the low-level blocks after a predetermined amount of time (usually seven days) and begin storing the freshly created blocks. Since data often accessed by users is always freshly generated, this design of regularly clearing data enables blockchain transactions to be maintained on the edge device for a week, ensuring high accessibility of blockchain data and lessen the storage demand brought on by the exponential expansion of blockchain data on edge devices. Furthermore, keeping data for a week also help blockchain nodes to monitor the behavior of external cloud storage, meaning that blockchain nodes can immediately stop uploading blockchain data to cloud storage once it is found to be evil, which ensures the data sovereignty of edge computing nodes within a certain period.

3.1 On-chain distribution of Blocks based on *Chord Ring*

3.1.1 Node network topology design

In this subsection, we propose a multi-layer architecture based on the *Chord Ring* design to organize the chain's nodes rather than a flat hierarchy. According to the hierarchy from top to bottom, the roles of nodes are specified as *leader nodes*, *vice-leader nodes*, *Successor nodes*, and *follower nodes*. The blockchain network's topological architecture is depicted in Fig. 1 along with the transformational relationship between nodes, which includes the following concepts:

- *Chord Ring*: The key idea of topology design in this study. When nodes join the network, they are all assigned to a cluster of nodes on the *Chord Ring*. The management,

interaction and work between nodes are based on this *Chord Ring* architecture. All nodes work together to enable the efficient running of the blockchain system. Compared with the traditional architecture[**?**], our paper improves its hierarchy design and routing addressing.

- *Chord cluster*: A collection of nodes on the *Chord Ring*, each *chord cluster* corresponds to a number, which is the result of modulo a fixed number after hashing the IP information of the node (Fig. 1 shows the result of modulo 8, so there are a total of 8 clusters on the whole network, with serial numbers from 0 to 7 arranged clockwise on the *Chord Ring*). The blockchain ledger is stored in part by each *chord cluster*, and the data between clusters is related in a non-intersecting way. The essential component of the *chord cluster* is changing the data storage from maintaining one whole ledger per node to storing only a portion of the ledger per node, hence minimizing the data redundancy rate over the entire blockchain network.
- *Leader node* : A *leader node* node is responsible for collecting the transaction information sent by all the nodes in the network and packing transactions into blocks containing PoW(Proof-of-Work) and has the most authority among the network nodes. A *leader node* node is distinguished by its protracted online status, robust computational power, and large storage capacity. Each node can only serve as the *leader node* for one week continuously before stepping down and being replaced by the *vice-leader nodes* in a specific order. This is known as the "term" concept for the *leader node*. The transformation of a *leader node* into a *follower node* following resignation is seen in Fig. 1.
- *Vice-leader node* : A *vice-leader node* is the successor of a *leader node*, elected from *follower nodes* or succeeded by a *successor node*(each *vice-leader node* has a *successor node* corresponding to it). *A vice-leader node* is responsible for receiving blocks from a *leader node* and performing validation work, then broadcasting blocks to the *follower nodes* it manages. It also in charge of obtaining the verification signature data of a few blocks from *follower nodes* and sending it to a *leader node*. *Vice-leader nodes* have no concept of term and will keep working until they transfer to a *leader node* or drop out. Fig. 1 depicts the process by which a *vice-leader node* replaces a *leader node* as the network's new *leader node*. Theoretically, each *vice-leader node* will act as a *leader node* once after $n-1$ terms if the fixed number is assumed to be n.
- *Successor node* : A *successor node* is the successor of a *vice-leader node*, and is mainly responsible for listening to the mutual heartbeat with the *vice-leader node*. A *successor node* will automatically become the new *vice-leader node* of the *chord cluster* and begin a election process for a new *successor node* if the status of a *vice-leader node* changes which includes taking over as the *leader node*, leaving the network, or quiting. Fig. 1 depicts how the *successor node* replaces the previous *vice-leader node* in the *chord cluster* 5 and takes over as the cluster's new *vice-leader node*.
- *Follower node* : *Follower nodes* are the lowest level nodes in blockchain network, which are only in charge of producing transaction data and storing blockchain ledger at regular intervals. Their storage capacity, computational power, and online time are very random and heterogeneous. When a *follower node* joins the blockchain

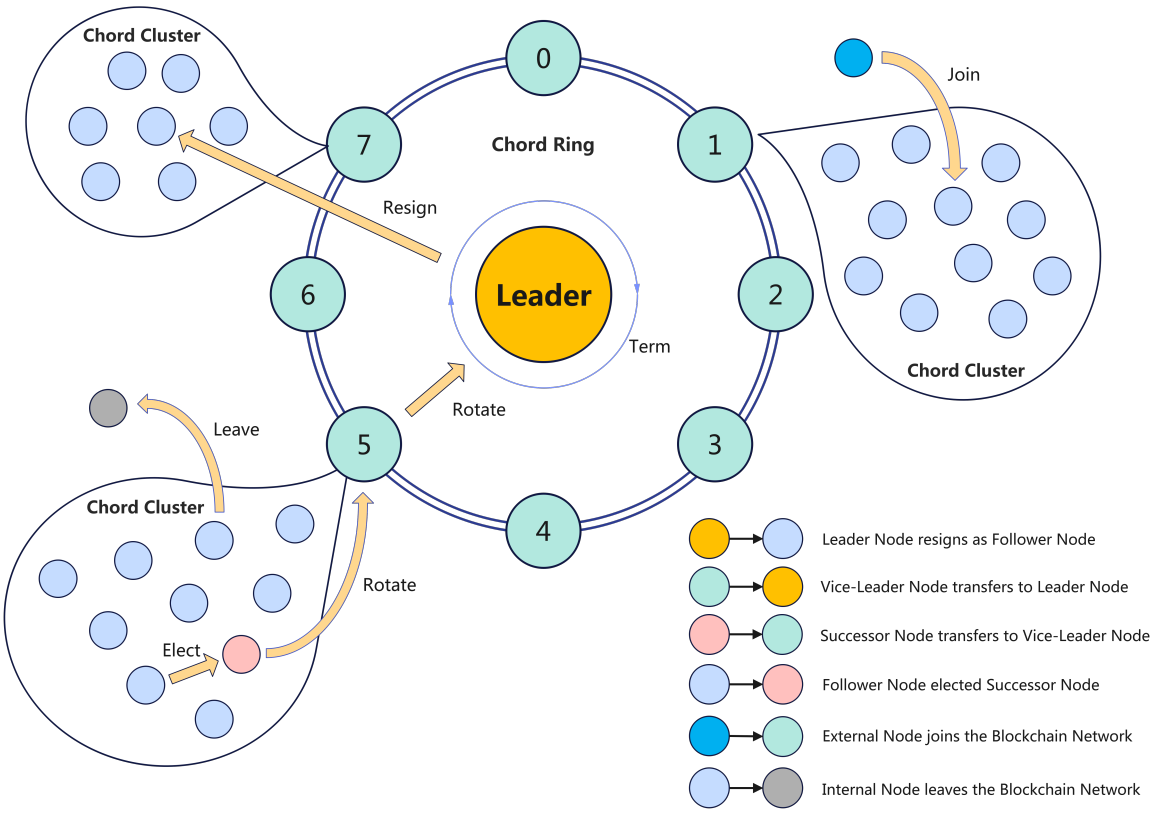

Figure 1 Network topology design based on *Chord Ring*

network, it is given a *chord cluster* to belong to, and the nodes in a same cluster will consistently store block data. Since *follower nodes* store the blocks' underlying transaction data, a *leader node* must use *follower nodes* to verify any special blocks it generates (hour blocks or day blocks). As a consequence, a *leader node*'s ability to tamper with the transactions in the resulting "enhanced block" is reduced, and the network's decentralization and data security are improved. In a *chord cluster* election, some of the more inventive *follower nodes* may be elected as a *successor node* and eventually have the chance to advance to *leader node* across the network. The specific election method can be introduced in our previous work[27].

The network topology proposed in this subsection classifies nodes with different levels responsible for different work. First, a *leader node* is responsible for collecting transactions, generating blocks, and broadcasting blocks. Secondly, *vice-leader nodes* are responsible for managing clusters, verifying and forwarding blocks, and acting as the backup of the leader node. Thirdly, *successor nodes* are responsible for acting as the backup of the vice header node. Finally, *follower nodes* are responsible for generating

transactions, storing blocks, and supervising the leader node. To sum up, this design can achieve the scalability of the blockchain, and the low-level nodes as the backup of high-level nodes can ensure that the blockchain network will not stop when some node status changes. Algorithm 1 represents the building process of the whole blockchain network based on *Chord Ring*, starting from the building of clusters, then *follower nodes* are randomly allocated to clusters, and then *vice-leader nodes*, *successor nodes*, and a *leader node* are generated in turn.

Algorithm 1 Blockchain Network Establishment.

$ChordclusterGroup[\ ] \leftarrow$ **CreateChordRing**$(clusterNum)$
for $i = 0; i < clusterNum; i++\{$
  if Hash$(Node.IP)Mod(clusterNum) == i\{$
  $Node$ **Join** $ChordclusterGroup[i]$
  break$\}$
$\}else$ **continue**
for $j = 0; j < clusterNum : j++\{$
  $ChordclusterGroup[j].ViceLeader \leftarrow$ **ElectViceLeader**()
  $ChordclusterGroup[j].Successor \leftarrow$ **ElectSuccessor**()
$\}$
$Leader \leftarrow ViceLeader$
NormalWork$(7\ Days)$

3.1.2 Finger Tables and block distributed storage

In this subsection, we introduce the differences between the routing tables maintained by four levels of nodes, and explain how a block is allocated to a specific *chord cluster*. The role of a node may change, and when this happens, the role of other nodes in the network will also change, and the routing information maintained by the nodes will be modified. This process can be described by algorithm 2.

Fig. 2 depicts the Finger Tables that various kinds of network nodes maintain, along with the way that each block is stored on the *Chord Ring*. Each *vice-leader node* in the network has an IP address listed in the *leader node*'s Finger Table. That becauses the *leader node* will send the block to the associated *vice-leader node* according the result of moduloing a fixed number (8 in the illustration figure) by a block's hash. For example, if Block $N-2$ has a $BlockHashmod8 = 5$, the block will be sent by the *leader node* to the *vice-leader node* in *chord cluster* 5. The Fig. 2 shows that from block $N-2$ to block $N+2$, they are assigned to different *chord clusters*, forming a complete blockchain together.

A *vice-leader node* has a Finger Table in which it logs the IP addresses of the current *leader node* in the network, the vice-leaders of other *chord clusters*, the *successor node*, and a few *follower nodes* in the *chord cluster* it manages. The block's verification information, gathered from the *chord cluster* will be sent to the *leader nodes*. A *vice-leader node* can forward blocks that do not belong to its own cluster to other

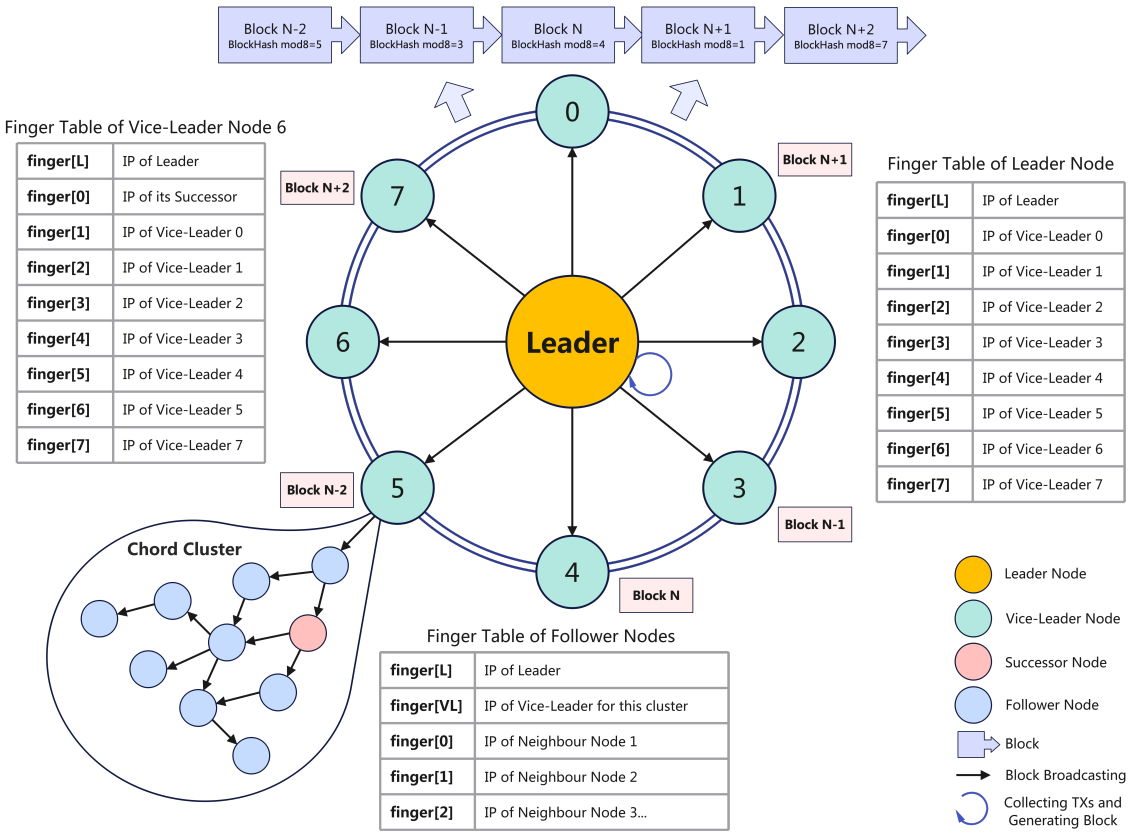

Figure 2 Finger tables of different kinds of nodes

vice-leader nodes when it receive blocks storing the IP information of other *chord clusters.*The IP address of a associated *vice-leader node* as well as the IP addresses of a few *follower nodes* in the same cluster are both recorded in a *successor node*'s Finger Table. A *successor node* will listen to a *vice-leader node* to see if it is online all the time and take over the role of the vice-leader when the vice-leader drops or becomes a new *leader node*.

A *follower node*'s Finger Table stores IP addresses of the *leader node*, the vice-leader information that supervises it, and a few nearby *follower nodes* that belong to the same cluster. Each *chord cluster*'s *follower nodes* submit transaction information to the *leader node*, and these information serves as the foundation for constructing a block. At the same time, a block from a *vice-leader node* is propagated in the cluster in the form of flooding until all nodes have stored this block.

Algorithm 2 Node State and Role changes.

```
1: BlockchainNormalOperate()
2: ifTrue NodeTypeChange(){
3:   if NodeType == Leader{
4:     Leader = NextViceLeader
5:     NewViceLeader = ElectViceLeader()
6:     ChangeFingerTable()
7:   }else if NodeType == ViceLeader{
8:     NewViceLeader = Successor
9:     NewSuccessor = ElectSuccessor()
10:    ChangeFingerTable()
11:  }else if NodeType == Successor{
12:    NewSuccessor = ElectSuccessor()
13:    ChangeFingerTable()
14:  }else if NodeType == Follower{
15:  ChangeFingerTable()
16: }
```

By introducing the routing tables maintained by nodes at all levels, the interaction logic between nodes can be represented. It can be seen that the communication between the *vice-leader node* and all other level nodes is bidirectional, but the communication between the *leader node* and *follower nodes* is unidirectional. Besides, this subsection also shows how different blocks maintained by each cluster form a complete blockchain.

3.2 Off-chain Storage of Blockchain based on Asynchronous Consensus

3.2.1 Hierarchical block architecture

In this subsection, we propose a hierarchical block architecture containing three types of blocks, and describe the relationship between the three types of blocks, how nodes at different levels handle different types of blocks, and why this design is adopted.

The hierarchical block architecture in this work includes three different sorts of blocks: minute blocks, hour blocks, and day blocks. Blocks are named by the time interval they are generated, i.e., minute blocks are generated every minute, hour blocks are generated every hour, and day blocks are generated every day. Each hour block is a collection of all minute blocks packaged in the previous hour, and each day block is a collection of all hour blocks packaged in the previous day.

All three types of blocks are produced by the *leader node*, as seen in Fig. 3. When a minute block is formed, the *leader node* broadcasts it to the relevant *chord cluster* for 7 days of permanent storage. A minute block contains all the transactions submitted by *follower nodes* to the *leader node* within one minute. The robustness of the blockchain network data is improved by this design, which makes sure that block data is not solely held in a small number of "central nodes" in the network, and the key benefit of keeping minute blocks on the *follower nodes* is that when a *leader node* creates an hour or day block, *follower nodes* must verify it, which prevents a malevolent *leader*

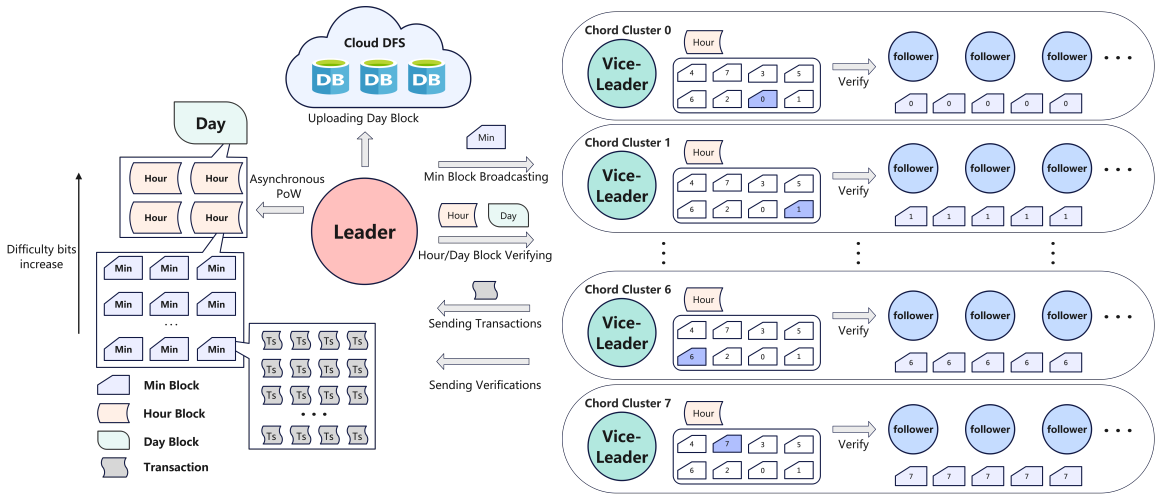

Figure 3 Hierarchical block architecture and asynchronous consensus process

node from tampering with the blockchain. The algorithm 3 shows the pseudo code how a *leader node* generates blocks of three types at different intervals.

Algorithm 3 Blocks Generated by Leader Node.

```
1:
2:  while{
3:    TxGroup[ ] ← Collect(Transactions)
4:    MinBlock ← GenerateMinBlock(TxGroup[ ], ...)
5:    TimeWait(1 Minute)
6:  }
7:  while {
8:    PreHourBlock ← GenerateHourBlock(MinBlockGroup[ ], ...)
9:    ifTrue VerifyBlock(PreHourBlock){
10:   HourBlock[ ] append PreHourBlock}
11:   else RegenerateHourBlock()
12:   TimeWait(1 Hour)
13: }
14: while {
15:   PreDayBlock ← GenerateDayBlock(HourBlockGroup[ ], ...)
16:   ifTrue VerifyBlock(PreDayBlock){
17:   DayBlock[ ] append PreDayBlock}
18:   else RegenerateDayBlock()
19:   TimeWait(1 Day)
20: }
```

Fig. 3 also shows that the *leader node* will transfer the day blocks or hour blocks to the *vice-leader node* of each *chord cluster* for verification. When a *vice-leader node*

receives the block, it will first confirm its legality by checking information like whether the block's hash is accurate and carries the *leader node*'s signature information. Upon successful completion of the verification, a *vice-leader node* then splits the block, locates the minute blocks associated with the *chord cluster* it maintains, and broadcasts these minute blocks to *follower nodes* for validation. Theoretically, a *follower node* stores the identical block information in the day blocks or hour blocks, and would compares the received blocks with its own stored blocks. When the verification is passed, *follower nodes* attache signatures to the verification information and send it to the *vice-leader node* in the same cluster. When the *leader node* receives the verification result, it confirms that the day/hour block can be stored persistently. Algorithm 4 describe the procedures of how hour blocks are verified and stored, and the procedures of day blocks are close to hour blocks.

Every seven days, the *leader node* will upload a collection of day blocks, and *follower nodes* will empty all of the minute blocks they stored. This is an off-chain transfer of data storage from the on-chain. It should be noted that each *leader node* synchronizes the block header data stored by the previous *leader node* with the block body data in the distributed storage off the chain when it assumes control. Since users are considered to be the most willing to access recently created data, the goal of this approach is to ensure that no forking or transaction rollback takes place prior to the data being persisted on the cloud storage by storing a batch of recently generated data across numerous nodes to improve data availability.

Algorithm 4 Verification of Hour Blocks.

```
1:
2: Viceleader.BlockVerify(Block){
3:   if BlockType == hourBlock{
4:     ifTrue CheckSignature(Block){
5:       blocks[ ] = Viceleader.SplitHourBlock(Block)
6:       for block in blocks[ ]{
7:         if block.clusterNum == Viceleader.clusterNum{
8:         minBlocks[ ] = append block
9:         }
10:        ifPass ←Viceleader.SendAndCollect(minBlocks[ ])
11:        ifTrue ifPass{
12:          Viceleader.BlockStorage()
13:          }else Discard()
14:      }
15:    }else Discard()
16: }
```

In the general blockchain architecture, there is only one block type and the blocks are generated serially, which limits the throughput of the system. By adopting a multi-level block architecture and setting the generation interval of low-level blocks as fixed, throughput of blockchain can be increased. At the same time, high-level blocks include low-level blocks, and long-term PoW is used to ensure subsequent security. This is the asynchronous consensus, which will be described in detail in the next subsection.

3.2.2 Asynchronous PoW Consensus

This part introduces the concept of PoW(Proof-of-Work) and the defects of traditional PoW, and explains what asynchronous consensus is and the necessity of adopting it. PoW is a technique that is frequently employed in blockchain systems to offer reliability. After determining a blockchain system's total network arithmetic strength, the fundamental notion is to select an appropriate hash bit difficulty using the one-way nature and collision resistance of hash functions, which usually means finding a sufficiently small value (in the binary representation, the first x bits representing the result of the calculation need to be 0). An attacker aiming for the blockchain network would theoretically have to pay the same amount of computational resources in order to tamper with a block because of the nature of the hash function (making it possible to try to locate the given computation result can only through traversal). More crucially, while an attacker is working to alter a specific block, new valid blocks are continuously created. As a result, if an attacker wishes to alter a entire blockchain, its arithmetic resources must vastly outnumber those of the entire blockchain network in use. However, PoW also has clear disadvantages: carrying out numerous hash calculations can dramatically decrease a blockchain network's throughput. For example, Bitcoin's block-generating time is in the range of ten minutes, and need to wait for an average of six blocks before the transaction on the first block is actually confirmed, which is not at all comparable to a centralized system with millisecond-level response.

In order to increase the accessibility of the block data and further boost the security and tamper-proofness of a blockchain without affecting its throughput. With the intention of giving multiple security levels to hierarchical blocks, we suggest an asynchronous PoW algorithm in conjunction with a hierarchical block structure. The security level is the first x bits of the hash value calculated by a block is zero, the larger the x, the more difficult the calculation, and the higher the security level. In the three-layer architecture of minute blocks/hour blocks/day blocks, minute blocks have the lowest security level and the day blocks have the highest security level. Minute blocks are generated at minute intervals, and the transactions in them are accessible to users for fast access. Hour blocks are equivalent to a reinforced authentication of the security and consistency of the transactions in minute blocks, while day blocks are equivalent to a reinforced authentication of the security of hour blocks. The PoW of the three are asynchronous and can be processed by the *leader node* at the same time. A day block with the highest security level is generated every day to ensure that the attacker needs to pay the corresponding cost for tampering with a blockchain afterwards. Users can take advantage of the high throughput of minute blocks for fast access to block data, while the security of blocks relies on hour blocks and day blocks that are continuously enhanced over time.

3.3 Flowchart

This subsection provides a flow chart from the beginning to the normal operation of the storing model(including the role replacement of different nodes) for better understanding. As Fig .4 shows, the overall process can be divided into three parts: network establishment, normal operation and node status change.

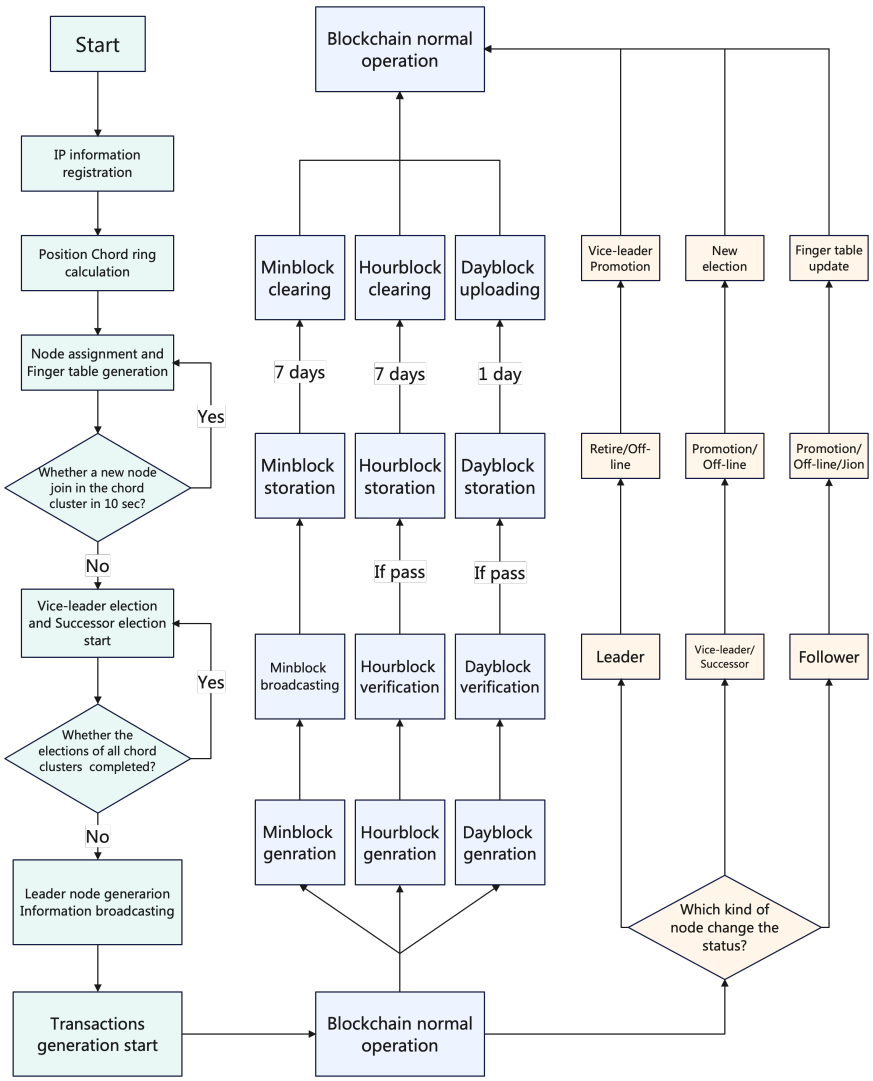

Figure 4 Flowchart of lightweight blockchain.

In the network establishment part, first of all, each node will register its own IP information when joining the network and calculate the position on *Chord Ring* and join into the *chord cluster* it belongs to. Then, each node will generate a finger table to record IP information of the *chord cluster*, and wait for new nodes to join in. When the number of nodes in a cluster stops growing within ten seconds, the election in a cluster will begin, which includes the *vice-leader node* election and the *successor node* election. After elections of all clusters are over, the *vice-leader node* in the first place on the *Chord Ring* in clockwise order transmits to the *leader node* of the *Chord Ring* network. Once the *leader node* appears, it will broadcast to the whole network, start to

collect transactions, and generate blocks. Here, the network establishment part finish and the normal operation of blockchain part start.

The normal operation part mainly includes the generation, verification and storage procedures of different levels of blocks. Firstly, the *leader node* collects all the transactions with in one minute and packs them into a minute block. Then the hash of the minute block will be calculated by modulus to locate its position on *Chord Ring*. After that, the generated minute block will be send to a *chord cluster* depends on the number of modulus calculation. The *vice-leader node* of corresponding *chord cluster* will receive it first and then verify the Block. If the verification pass, this minute block will be broadcasted to the whole *chord cluster*, and all *follower nodes* in the cluster will receive the minute block and store it. Sencondly, every hour an hour block will be generated by the *leader node* which contains all minute blocks generated in the last hour. The PoW of hour blocks is higher than minute blocks and the *leader node* send hour blocks to all *vice-leader nodes*. As mentioned above, the minute blocks contained in an hour block are separated and stored by all clusters. Therefore, once a *vice-leader node* of a *chord cluster* receives an hour block, it unpacks the block and filter out the minute blocks belong to its own *chord cluster*. Then the minute blocks filtered will be send to all the *follower nodes* in the cluster to verify. After *follower nodes* receive the minute blocks, then they compare the blocks to the minute blocks in their local storage. If the blocks are same, the verification passes and the *follower nodes* send their significations to the *vice-leader node*. Once a *vice-leader node* receives the significations more than the two thirds of *follower nodes*, it confirms the hour block is legal and correct. Then the *vice-leader node* send its signature to the *leader node* to confirm the hour block. The verification process of day block is similar to that of hour block, but the difference is that every seven days, only day blocks will be uploaded to the cloud, and other type of blocks will be cleared by *follower nodes*.

The node status change part including role changes and status changes of various nodes. Role changes refer to the conversion of nodes at different levels according to the inherent design during the normal operation of the model. For example, from a *follower node* to a *successor node*, from a *successor node* to a *vice-leader node*, and from a *vice-leader node* to a *leader node*. The retirement of a *leader node* is also a kind of role change, which happens every seven days. When the original *leader node* retires, it will become a common *follower node* and continue to participate in the election of a *successor node* in the subsequent work. On the other hand, status change refers to some abnormal situations, such as a sudden disconnection or an exit of a node. When this happens, a replacement node will start to work and the related IP information will also be updated synchronously. As mentioned above, a *successor node* is the replacement node of a *vice-leader node*, while a *vice-leader node* is the replacement node of a *leader node*.

4 Theoretical Analysis

In this section, we analyze the performance of the storing model designed based on this paper in reducing the network storage load, including the storage load of four types of nodes under blockchain normal operation. Use BS for block size, BC for block count,

and CN for cluster number. Let N denote the number of all nodes, DS represent the data size.

In a generic blockchain architecture, each node stores a complete copy of the blockchain ledger, so the data size of the entire network can be expressed as (1):

$$DS_{GenericChain} = BS \times BC \times N \tag{1}$$

The *leader node* stores three types of blocks, namely minute blocks, hour blocks and day blocks. Since high-level blocks are generated by packing blocks of lower levels with different PoW, the overall size of these three types of blocks is the same, which means that the *leader node* stores data equivalent to three times the size of a generic blockchain node. And in the design of our paper, *vice-leader nodes* store minute blocks and hour blocks, but due to the existence of clusters, the size of data stored by the nodes within each cluster is divided by the number of clusters. Then, both *successor nodes* and *follower nodes* store only minute blocks, so these two types of nodes have the smallest data size. The data size of the *leader node*, a *vice-leader node*, a *successor node* and a *follower node* are expressed as (2), (3), (4) respectively.

$$DS_{Leader} = BS \times BC \times 3 \tag{2}$$

$$DS_{Vice-leader} = \frac{BS \times BC \times 2}{CN} \tag{3}$$

$$DS_{Successor} = DS_{Follower} = \frac{BS \times BC}{CN} \tag{4}$$

By adding up the data sizes of all types of nodes, we can get the data size of our storing model for a whole blockchain in (5). Then, we use $BDSR$(Blockchain Data Size Ratio) in (6) to denote the degree of reduction of blockchain storage load by the model in our paper. The calculation results show that when N is much larger than the CN, the $BDSR$ is a number approximately equal to $1/CN$, but greater than $1/CN$. This result demonstrates that by adjusting the parameter CN, the amount of data redundancy across the network can be significantly reduced.

$$\begin{aligned} DS_{OurModel} = DS_{Leader} + CN \times (DS_{Vice-leader} + \\ DS_{Successor}) + (N - 1 - 2CN) \times DS_{Follower} \end{aligned} \tag{5}$$

$$BDSR = \frac{DS_{OurModel}}{DS_{GenericChain}} = \frac{N + 4CN - 1}{N \times CN} \approx \frac{1}{CN} \tag{6}$$

5 Experiment and analysis

In this section, we establish two blockchain systems to compare the data size, one is based on our storing model, another is based on a normal blockchain architechture. We let the two blockchain systems run for one day normally, and calculate the total amount of data generated by both from the backend. According to our design, our

model would generate 1440 minute blocks, 24 hour blocks and 1 day block, sense the normal blockchain architechture does not have multi layer blocks, let the normal blockchain generate 1440 blocks. To prove the scalability of our model, we set the number of nodes that make up the blockchain network to 100, 200, 300, 400 and 500. At the same time, according to the design in the previous chapters, we have established different numbers of clusters in the network, namely 4, 5, 6, 7 and 8. The software enviroment is based on open source such as TDengine, Docker, Go, Jmeter and etc., the hardware environment of this experiment is shown in the table 1.

Table 1 Experiment Set Up of Hardware and Software

Experiment set up	Description
CPU	Phytium,FT-2000+/64
RAM	128GB
HDD	4TB
System	KylinOS

Then, we counted three main indicators, which are respectively used to reflect the average degree of assigning nodes to each cluster, the average degree of assigning blocks to each cluster, and the reduction of the overall data storage capacity of the blockchain in our paper design. All indicators are the average values calculated from running the program ten times under the same conditions. The first indicator is defined as "coefficient of variation of nodes(CVN)" in (7), calculated from the number of nodes x in each cluster. The second indicator is defined as "coefficient of variation of blocks(CVB)" in (8), calculated from the number of blocks β in each cluster and number of clusters m($C\bar{V}B$ is the average of CVB). The third indicator is $BDSR$ mentioned in the previous section.

$$CVN = \frac{\sqrt{\frac{\sum(x_i-\bar{x})^2}{n-1}}}{\bar{x}} \tag{7}$$

$$C\bar{V}B = \frac{\sum^{m} \frac{\sqrt{\frac{\sum(\beta_j-\bar{\beta})^2}{n-1}}}{\bar{\beta}}}{m} \tag{8}$$

Fig. 5 shows that when the number of clusters is the same, as the number of nodes increases, the nodes will be more evenly distributed to the network, and the number of nodes owned by each cluster will be closer. Under the condition of the same number of network nodes, the more the number of clusters, the more uneven the distribution of nodes will be, but when the number of nodes is large, the impact of the number of clusters will be accordingly reduced. Generally speaking, both the number of clusters and the number of nodes will affect the uniformity of node dispersion, the number of clusters is inversely proportional to CVN, and the number of nodes is directly proportional to CVN. CVN is generally within a small range, that is, less than 0.4, which means that all nodes can be evenly distributed to each cluster according to the design.

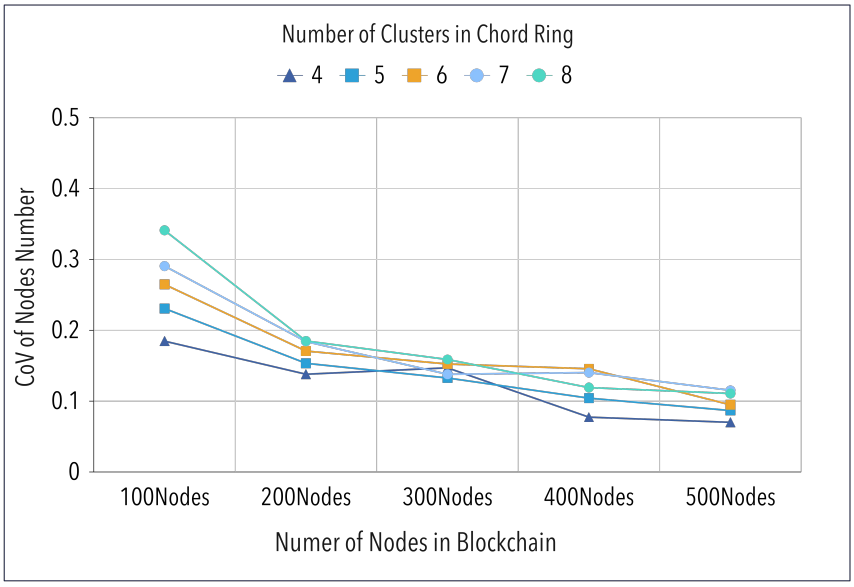

Figure 5 CVN statistics from 100 nodes to 500 nodes and from 4 clusters to 8 clusters

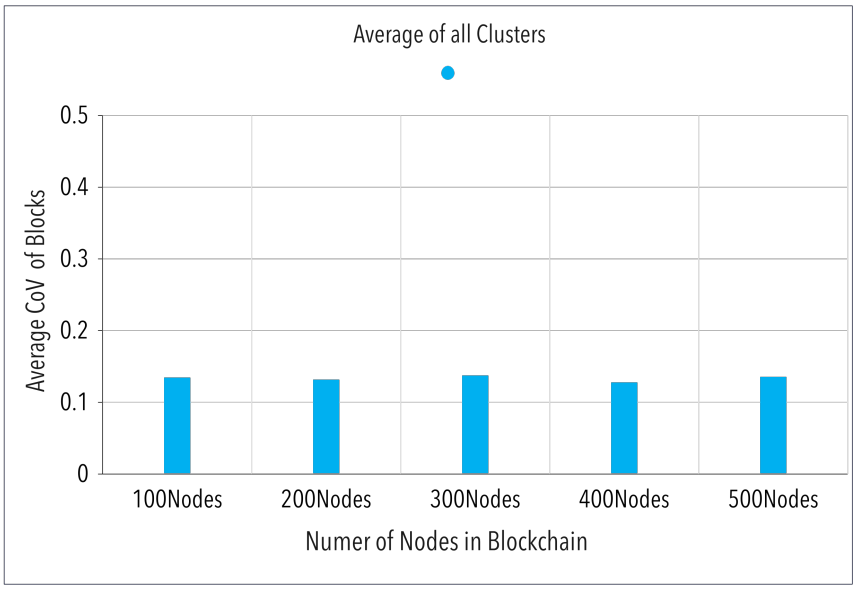

Figure 6 $C\bar{V}B$ statistics from 100 nodes to 500 nodes(average of all clusters)

Fig. 6 shows the uniform distribution of blocks under different numbers of nodes. Since the number of blocks is constant, here we no longer compare the differences between each cluster, but average them. It can be seen that when increasing from 100

nodes to 500 nodes, there is almost no change in $C\bar{V}B$(around 0.15), which means that after a day of normal operation of the blockchain designed based on our model, blocks will be evenly distributed to each cluster. The measurement results of $C\bar{V}B$ show that the storage load of the blockchain is uniform, and there will be no obvious difference in storage pressure for a specific cluster.

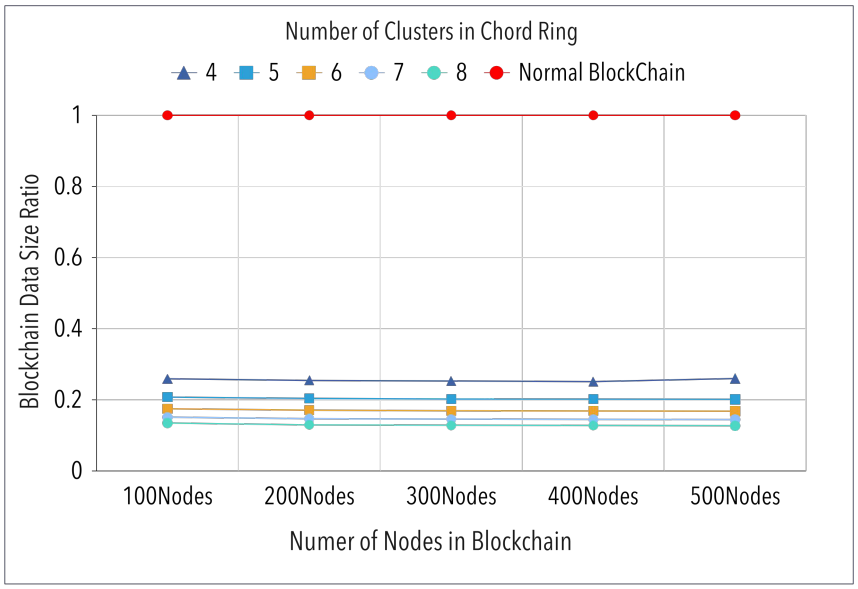

Figure 7 *BDSR* statistics from 100 nodes to 500 nodes and from 4 clusters to 8 clusters

Fig. 7 compares the total amount of data storage on all nodes under two blockchain systems. It can be seen that when the total amount of data under the normal chain architecture is 1, the *BDSR* on the blockchain nodes based on the our model is always less than 1, and will further decrease as the number of clusters increases. As the number of nodes increases, the total amount of data hardly changes, which means that the only factor affecting the total amount of data is the number of clusters. This is in line with the design of our paper, which is to split the data into disjoint sets and assign them to different clusters. The more clusters there are, the more sets are split, and the smaller the amount of data stored on each node is, which will reduce the total data volume of the whole network.

6 Conclusion

This paper addresses the problem of excessive storage load of existing blockchains, and describes the existing solutions of on-chain ledger and off-chain ledger lightweighting methods and their shortcomings. For the resource-constrained end devices in the IoT edge computing scenario, a collaborative ledger storing model is proposed, whicn adopts the *Chord Ring* architechture to cluster the nodes and split the ledger on-chain,

and adopts a block data storage strategy of periodically clearing and uploading to the cloud, combining a hierarchical block structure off-chain. The experiment proves that the method can significantly reduce the storage load of blockchains, and this design of distributed storage of blocks can also ensure the trustworthiness and availability of blockchain data, which helps to realize the application of lightweight blockchain in the IoT scenario.

Declarations

- **Funding** This research is founded by National Key Research and Development Program of China(2022YFB3104900).
- **Conflict of interest** No conflicts of interest are disclosed by the authors.
- **Ethics approval** The research is compatible with ethical standards.
- **Availability of data and materials** The study's data is available publicly and can be easily accessed from the internet.
- **Authors' contributions** Nie did the whole research and wrote the manuscript under the supervision of Lu, the major supervisor, and Li and Duan, the co-supervisors. All authors read and approved the final manuscript.

References

[1] Nakamoto, S., Bitcoin, A.: A peer-to-peer electronic cash system. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf **4**(2), 15 (2008)

[2] Wood, G., *et al.*: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)

[3] Yaqoob, I., Salah, K., Jayaraman, R., Al-Hammadi, Y.: Blockchain for healthcare data management: opportunities, challenges, and future recommendations. Neural Computing and Applications, 1–16 (2021)

[4] Zhu, L., Liang, H., Wang, H., Ning, B., Tang, T.: Joint security and train control design in blockchain-empowered cbtc system. IEEE Internet of Things Journal **9**(11), 8119–8129 (2021)

[5] Chen, R., Shu, F., Huang, S., Huang, L., Liu, H., Liu, J., Lei, K.: Bidm: A blockchain-enabled cross-domain identity management system. Journal of Communications and Information Networks **6**(1), 44–58 (2021)

[6] Liang, W., Yang, Y., Yang, C., Hu, Y., Xie, S., Li, K.-C., Cao, J.: Pdpchain: A consortium blockchain-based privacy protection scheme for personal data. IEEE Transactions on Reliability (2022)

[7] Han, D., Zhu, Y., Li, D., Liang, W., Souri, A., Li, K.-C.: A blockchain-based auditable access control system for private data in service-centric iot environments. IEEE Transactions on Industrial Informatics **18**(5), 3530–3540 (2021)

[8] Gupta, M., Patel, R.B., Jain, S., Garg, H., Sharma, B.: Lightweight branched blockchain security framework for internet of vehicles. Transactions on Emerging Telecommunications Technologies, 4520 (2022)

[9] Javaid, M., Haleem, A., Singh, R.P., Khan, S., Suman, R.: Blockchain technology applications for industry 4.0: A literature-based review. Blockchain: Research and Applications **2**(4), 100027 (2021)

[10] Ullah, F., Al-Turjman, F.: A conceptual framework for blockchain smart contract adoption to manage real estate deals in smart cities. Neural Computing and Applications **35**(7), 5033–5054 (2023)

[11] Wang, Y., Chen, C.H., Zghari-Sales, A.: Designing a blockchain enabled supply chain. International Journal of Production Research **59**(5), 1450–1475 (2021)

[12] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30 (2016)

[13] Frey, D., Makkes, M.X., Roman, P.-L., Taïani, F., Voulgaris, S.: Bringing secure bitcoin transactions to your smartphone. In: Proceedings of the 15th International Workshop on Adaptive and Reflective Middleware, pp. 1–6 (2016)

[14] Jing, N., Liu, Q., Sugumaran, V.: A blockchain-based code copyright management system. Information Processing & Management **58**(3), 102518 (2021)

[15] Kumar, S., Bharti, A.K., Amin, R.: Decentralized secure storage of medical records using blockchain and ipfs: A comparative analysis with future directions. Security and Privacy **4**(5), 162 (2021)

[16] Yu, B., Li, X., Zhao, H.: Virtual block group: A scalable blockchain model with partial node storage and distributed hash table. The Computer Journal **63**(10), 1524–1536 (2020)

[17] Wang, X., Wang, W., Zeng, Y., Yang, T., Zheng, C.: A state sharding model on the blockchain. Cluster Computing **25**(3), 1969–1979 (2022)

[18] Kim, T., Lee, S., Kwon, Y., Noh, J., Kim, S., Cho, S.: Selcom: Selective compression scheme for lightweight nodes in blockchain system. IEEE Access **8**, 225613–225626 (2020)

[19] Zheng, Q., Li, Y., Chen, P., Dong, X.: An innovative ipfs-based storage model for blockchain. In: 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 704–708 (2018). IEEE

[20] Mani, V., Manickam, P., Alotaibi, Y., Alghamdi, S., Khalaf, O.I.: Hyperledger healthchain: patient-centric ipfs-based storage of health records. Electronics

10(23), 3003 (2021)

[21] Sarathchandra, T., Jayawikrama, D.: A decentralized social network architecture. In: 2021 International Research Conference on Smart Computing and Systems Engineering (SCSE), vol. 4, pp. 251–257 (2021). IEEE

[22] Ali, S., Wang, G., White, B., Cottrell, R.L.: A blockchain-based decentralized data storage and access framework for pinger. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 1303–1308 (2018). IEEE

[23] Hassanzadeh-Nazarabadi, Y., Küpçü, A., Özkasap, Ö.: Lightchain: Scalable dht-based blockchain. IEEE Transactions on Parallel and Distributed Systems **32**(10), 2582–2593 (2021)

[24] Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12–16, 2001 Proceedings 2, pp. 329–350 (2001). Springer

[25] Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on selected areas in communications **22**(1), 41–53 (2004)

[26] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM computer communication review **31**(4), 149–160 (2001)

[27] Nie, Z., Zhang, M., Lu, Y.: Hpoc: A lightweight blockchain consensus design for the iot. Applied Sciences **12**(24), 12866 (2022)