applied sciences

MDPI

Article

Ring-Overlap: A Storage Scaling Mechanism for Hyperledger Fabric

Wenxuan Liu, Donghong Zhang, Chunxiao Mu, Xiangfu Zhao and Jindong Zhao *

School of Computer and Control Engineering, Yantai University, Yantai 264005, China
* Correspondence: zhjdong@ytu.edu.cn

Abstract: Currently, blockchain is facing a serious storage explosion problem. While most storage scaling schemes are focused on permissionless blockchain, we propose the ring-overlap mechanism for consortium blockchain and use it to scale Hyperledger Fabric. In our scheme, all accounting nodes are divided into clusters, and each cluster contains several nodes; then, a portion of the entire block data is stored in a cluster. Block data is stored overlappingly on some cluster nodes, and each block is guaranteed to have some copies in a cluster. Theoretical analysis and simulation show that the storage occupied by nodes is significantly reduced in blockchain applications with frequent transactions, and the mechanism can still guarantee data integrity in the case of partial node failures in a single cluster. Furthermore, for transaction-frequent applications, storage space consumption can be significantly reduced without increasing excessive query time overhead.

Keywords: blockchain; storage limitations; overlap; shard; hyperledger fabric

Citation: Liu, W.; Zhang, D.; Mu, C.; Zhao, X.; Zhao, J. Ring-Overlap: A Storage Scaling Mechanism for Hyperledger Fabric. *Appl. Sci.* **2022**, *12*, 9568. https://doi.org/10.3390/app12199568

Academic Editors: Nadejda Komendantova and Hossein Hassani

Received: 6 August 2022
Accepted: 20 September 2022
Published: 23 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

Blockchain technology arose from Satoshi Nakamoto's paper "Bitcoin: A Peer-to-Peer Electronic Cash System" [1], which describes blockchain as a decentralized distributed data ledger. It allows all nodes in the network to jointly own, manage, and supervise data through cryptographic methods, and the system operates without the control of any single node, thus providing characteristics such as unforgeability, untamperability, and traceability. Although blockchain technology improves data security and reliability, its current storage scalability is inadequate. The blockchain is shared by all nodes in the system, all nodes in the blockchain are equal, and each accounting node holds all transactions and blocks of the network [2]. As the network continues to expand and the data stored in the blockchain rises dramatically, the blockchain will face the problem of the storage limitation of single nodes. Therefore, the study of blockchain storage scalability is imminent, and a reasonable and effective storage strategy is essential.

In recent years, sharding has emerged as an effective candidate technique for addressing the blockchain scalability problem [3]. It divides the network state into different sub-committees, each of which stores a portion of the data on each node, and the members of sub-committees can only process the set of transactions assigned to them [4,5]. In other words, transactions are assigned to different shards so that different transactions can be processed in parallel between the shards [6].

A blockchain platform can be classified into two types: public blockchain and consortium blockchain. Public blockchains are represented by Bitcoin and Ethereum, while the most widely used consortium blockchain is Hyperledger Fabric. Hyperledger Fabric differs from blockchains such as Ethereum or Bitcoin not only in terms of type, but also in terms of internal mechanisms [7]. Furthermore, it does not require expensive mining computation to submit transactions, which helps to build a scalable blockchain with shorter latency.

Most blockchain storage scaling schemes are proposed for public blockchain, but public blockchain scaling suffers from the problem of not being able to support large-scale

networks with high TPS, low latency, and security [8]. To address these problems, we propose ring-overlap, a storage scaling mechanism for consortium blockchain.

The major contributions of the paper are the following:

- The mechanism is to realize the storage scaling of the blockchain without increasing too much extra communication overhead;
- It tolerates the failure of some nodes without causing the loss of ledger data, which ensures the reliability of the system;
- It is compatible with the write and query mechanisms of Hyperledger Fabric, and only needs to add tiny extra maintenance mechanism.

Finally, theoretical analysis and experiments of the proposed mechanism are given. The results show that a single node's storage space tends to s/mn, which greatly improves blockchain scalability. In addition, the security analysis of this mechanism demonstrates that if the number of failed nodes within a single cluster is less than $s - 1$ (s is the number of copies a single block within a single cluster), the mechanism can still guarantee the blockchain's data integrity.

We note that a shorter conference version of this paper appeared in the literature [9]. Our original conference paper was only an initial idea of the ring-overlap mechanism and did not include experiments on system reliability or storage cost. This manuscript includes this section as well as additional analysis.

The rest of this paper is organized as follows: the background and related works are reviewed in Section 2. Sections 3 and 4 introduce the problem description and scheme design. Then the analysis and simulation experiment are given in Sections 5 and 6. Finally, the paper is summarized and future work is outlined.

2. Background and Related Work

Sharding technology is regarded as the most promising solution for breaking the performance and capacity bottleneck of blockchain [10], and more and more scholars have started to invest in the research of blockchain sharding schemes. There are many sharding-based protocols, such as Bitcoin-NG [11], Elastico [12], OmniLedger [13], RapidChain [14], Monoxide [15], and others [16–18]. This section provides an overview of the representative shard-based blockchain protocols and related work.

2.1. Shard-Based Blockchain Protocols

2.1.1. Bitcoin-NG

To improve the throughput of blockchain networks, Eyal et al. proposed the Bitcoin-NG protocol. Although Bitcoin-NG significantly improves the throughput of Bitcoin, frequent generation of transaction information can also cause network congestion. In addition, it is also vulnerable to the same attacks as Bitcoin, such as the 51% attack [19,20]. As a result, when the number of miners rises, Bitcoin-NG will not be able to scale the network.

2.1.2. Elastico

Elastico was proposed by Luu et al. The key idea is to divide the nodes in the network into random committees to process different shards. While Elastico does improve the throughput and latency of Bitcoin, it has some limitations. For example, Elastico only divides the nodes, not the blockchain. This results in all nodes needing to store the full blockchain. Therefore, when a block is generated, it needs to be sent to all nodes in the network. This leads to a high level of complexity in communication.

2.1.3. OmniLedger

OmniLedger is a sharding-based blockchain design proposed by Kokoris-Kogias et al. at SP2018. It is composed of an identity chain and multiple sub-chains, uses RandHound and VRF (verifiable random function) protocols [21] to randomly assign verifiers to different shards, and it has the same consensus within the shards as Elastico [22]. Compared with

the previous two protocols, the OmniLedger protocol makes a better trade-off between system security, scalability, and decentralization.

2.1.4. RapidChain

The RapidChain protocol was presented by Zamani et al. at the 2018 SIGSAC conference, and it accomplishes a further extension of blockchain performance by improving the state sharding technique. It is a fully sharded blockchain protocol that provides complete sharding of computational, storage, and communication overhead for handling transactions. Additionally, it divides nodes into smaller groups of nodes, called committees. Each committee maintains a storage shard and handles a separated set of blocks, and it can scale the throughput of the system proportionally to the number of committees. Unlike the previous protocols, the RapidChain protocol is the first blockchain protocol that is fully sharded.

2.2. Related Work

Moreover, many scholars have also studied the problem of sharding. SSMAB, a scalable storage model based on account-based blockchain, was proposed by Zhang et al. [23]. The model stores state data completely redundantly to ensure transaction verification, while block data is stored in shard storage to reduce redundancy. This model conserves storage space while ensuring data security and availability. Meepo, a systematic study on sharded consortium blockchain, was proposed by Zheng et al. [24]. Meepo improves cross-shard efficiency by utilizing cross-epoch and cross-call. Kim et al. [25] proposed a selective compression scheme that uses checkpoint chains to prevent compression results from accumulating. A large number of blocks can be verified by using a small number of updated checkpoints, allowing blockchain nodes to reduce the blockchain ledger's storage capacity. Liu et al. [26] proposed a sharding mechanism based on overlap: During the mapping process, OverlapShard alters the one-to-one model of nodes and shards and randomly maps nodes to multiple actual shards at the same time. Each node stores only the transaction history information (ledger subset) for the shard to which it belongs. OverlapShard not only reduces performance losses due to cross-shard transactions, but also improves system performance. Li et al. [27] proposed the ICIStrategy, a multi-node collaborative storage strategy based on intra-cluster integrity. It alleviates the storage pressure by reducing the amount of data that each participating node needs to store, and reduces the communication overhead by the nodes in the cluster cooperating to store and check blocks. Jia et al. [28] proposed a blockchain storage capacity scalable model. The main idea is to store the blocks in the blockchain in a certain proportion in the nodes, and a strategy for blockchain data copy distribution is proposed. Experiments show that the model reduces the storage space of massive nodes while maintaining certain stability and security, effectively increasing the blockchain's storage scalability. Table 1 compares the storage scalability schemes proposed in the preceding literature.

Compared with the schemes in Table 1, the model proposed in the paper has the following main advantages:

1. Cannot increase too much extra communication overhead;
2. Tolerates the failure of some nodes without causing the loss of ledger data;
3. Only needs to add tiny extra maintenance mechanism.

However, the model introduces some extra network communication overhead for transaction queries. If specific transactions are not queried frequently, these added communications overheads will not affect the system too much in the current good network infrastructure.

This paper, inspired by the literature [23–34], proposes a storage scaling mechanism for consortium blockchain, and uses it to scale Hyperledger Fabric.

Table 1. Blockchain storage scalable schemes comparison.

Storage Scalable Schemes	Proposer	Main Content
SSMAB	Zhang et al. [23]	Store stated data in a fully redundant manner to ensure transaction verification; store block data in a sharded manner to reduce redundancy; and implement an economic incentive mechanism to reduce storage consumption while ensuring data availability.
Meepo	Zheng et al. [24]	Cross-nesting and cross-call are used to improve the efficiency across shards; a partial cross-call merging strategy is used to handle multi-state dependencies in contract calls and achieve flexibility across shards; a backup algorithm called shadow shard-based recovery is also used to improve the robustness of shards.
SELCOM	Kim et al. [25]	To avoid the accumulation of compression results, a selective compression scheme based on checkpoint chains is proposed. An update process is also proposed to prevent the accumulation of checkpoints by merging them.
OverlapShard	Liu et al. [26]	By mapping each node to multiple actual shards, the adverse effects of cross-shard transactions are mitigated. To handle cross-shard transactions, virtual slices made up of overlapping nodes can be used.
ICIStrategy	Li et al. [27]	Divide all participants into clusters. Each cluster needs to store all of the network's data, and the nodes in the cluster are not required to maintain data integrity. Storage pressure is alleviated by reducing the amount of data that each participant needs to store, and communication overhead is reduced through collaborative storage and block verification by cluster nodes.
ElasticChain	Jia et al. [28]	Which fragments a blockchain replica and stores the fragments in a part of nodes. Validation nodes are added to perform real-time testing of nodes storing data based on data retrievability proof methods, record updated storage node stability values, and then select high-stability nodes to store newly generated data copies.

3. Problem Description

At the moment, blockchain platforms are classified as either public blockchains (Permissionless Blockchain) and consortium blockchains (Permissioned Blockchain). Bitcoin and Ethereum are examples of public blockchains, and the most widely used consortium blockchain is Hyperledger Fabric. The majority of the work mentioned above is an exten-

sion of public blockchain platforms and involves node consensus after sharding. Therefore, most of the work requires major changes to the platform to achieve realization, and there are many difficulties in practice. In contrast, the Hyperledger Fabric is deployed in a planned consortium structure and uses order sorting to complete consensus, which does not require accounting nodes to participate in a consensus process. Moreover, node joining is planned and managed, so network topology changes are not as frequent as in the public blockchain. As a result, this paper proposed a ring-formed overlapping shard storage strategy for a consortium blockchain such as Hyperledger Fabric in order to reduce storage pressure on accounting nodes and improve blockchain scalability.

3.1. Double-Shard Model

In a consortium chain, the number of nodes is assumed to be planned when the network is established, and the number and roles are relatively fixed. Consider a blockchain network with $m \times n$ nodes. First, by dividing the clusters, the peers who are responsible for maintaining the ledger and state in the network are divided into m clusters, each cluster consisting of n nodes. The data of the whole network is divided into non-overlapping m parts, and each cluster stores $1/m$ data. Then, in the clusters, the data is divided into n non-overlapping parts, each node stores s copies ($1 < s < n$), and each peer has overlapping data with some other nodes to avoid the data loss problem caused by a single node failure. On a certain node in a cluster, the tail of the data it stores is overlapped with the head of the data restored on the next node. Then the storage structure formation of the data in a cluster is a ring. Therefore, we call the mechanism "Ring-Overlap".

3.2. Ring-Overlap Storage Model

The data stored in Hyperledger Fabric is composed of four parts: world state, block index, ledger data, and historical state (optional). The two most important components of the ledger, which together from the blockchain, are the world state and ledger data. The specific architecture is shown in Figure 1.

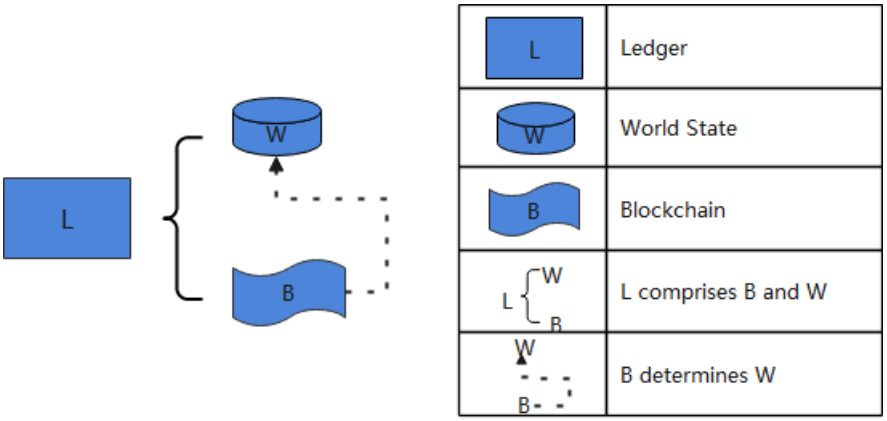

Figure 1. Fabric data storage structure.

The world state represents the current value of all ledger states and is usually implemented using a K-V database. The current value of a ledger state can be accessed directly through the world state, which is represented as a key-value pair. The ledger data is composed of blocks that are connected together by pointers. Each block contains a set of world state transaction logs with a Merkle Tree structure that cannot be tampered with, which records all changes that contribute to the current world state.

In the Hyperledger Fabric, the peers on the same channel all store the same blockchain data. According to the properties of ledger data, blockchain data is the primary cause of data inflation, and the world state database is unsuitable for sharding. Therefore, the model in this paper takes blocks as the basic unit of shard data.

4. Scheme Design

Since in the traditional blockchain storage mechanism, each peer node needs to store all the data in the network, which causes great storage pressure on the nodes, this paper proposes a double-shard storage scaling mechanism for consortium blockchain. The mechanism divides the data for storage using clustering, and each node within the cluster stores part of the data according to overlapping rules to solve the storage pressure problem by reducing the amount of data that must be stored by a single node. This mechanism is aimed at the Hyperledger Fabric because the number of nodes in the consortium chain is relatively fixed.

4.1. Cluster Division

The proposal is to use static clustering to divide the peer nodes into m clusters. Each cluster consists of n nodes, and each node keeps the cluster number and node numbers. If the number of nodes cannot be divided by n, make the last cluster have more nodes than n but less than $2n$. In this way, the block data of the whole network is divided into non-overlapping m parts, and each cluster stores $1/m$ of the data.

After generating a new block, it is broadcast to the network. Unlike the original mechanism of the Hyperledger Fabric, not all peer nodes save the block. The nodes in each cluster determine whether to save the data according to the block number, and only those whose cluster number matches the block number save the data. Furthermore, the nodes in the cluster then determine which nodes save the data according to the matching rule between node number and block number and store block data in overlapping storage mode.

Taking $m = 4$, $n = 3$ as an example, the whole network divides 12 peer nodes into four clusters. Each cluster contains three nodes, and each cluster stores 1/4 of the data. The system structure of the ring-overlap is illustrated in Figure 2.

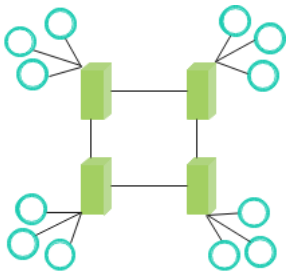

Figure 2. System structure of ring-overlap.

4.2. Intra-Cluster Data Storage

In [27], the authors divide all nodes in the network into multiple clusters by the dynamic clustering algorithm. Each cluster needs to store all data, but nodes within the cluster are not required to maintain data integrity. The new proposed mechanism in the paper differs from [27] in that each cluster only stores $1/m$ of the block data. If the nodes in a single cluster also follow the random storage method, the integrity of all the data in the network cannot be guaranteed if some nodes in the cluster are corrupted. Therefore, based on the division of clusters, we proposed overlapping storage in a single cluster, which can still ensure data integrity even if some nodes in a single cluster fail.

Block data is divided and stored in the cluster based on parameters such as the number of nodes in the cluster and the total data volume of the block. The nodes in the cluster store the block data in an overlapping manner, and the storage capacity available in a single node must be greater than the size of a single block, and the total storage capacity of all the nodes in the cluster must be greater than the total data volume of the blockchain. Still, with $m = 4$ and $n = 3$ as an example, it is divided into 4 clusters by static clustering, with three nodes in each cluster, and each cluster stores 1/4 of the block data, naming the block data as A. Using overlapping shard storage, A is divided into three parts, and two parts

should be stored on each node. The data storage structure of a single cluster is illustrated in Figure 3.

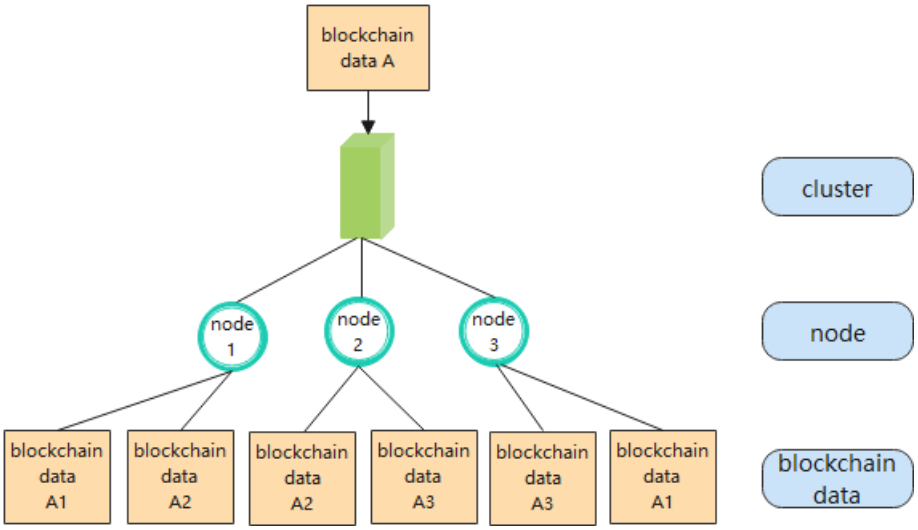

Figure 3. Single cluster data storage structure.

4.3. Storage and Query Mechanism

4.3.1. Storage Process

In our proposal, all nodes in the network are divided into m clusters by static clustering, and the number of nodes in each cluster is n. Each node keeps its own cluster ID i, node ID in the cluster j, the total number of nodes in the cluster n and the number of shardings to be kept by each node s, where $i = 1, \ldots, m$, $j = 1, \ldots, n$, $s > 1$ and $s < n$.

Since block data are not static but grow continuously, in practice, it is not possible to divide the dynamic block data into a prescribed number of n parts in advance. In this scheme, when a new block is generated, it is broadcast to the whole network. The peer nodes in the network receive it and determine whether to save it based on the block number b, the number i of their own cluster, and their own node number j. The specific rules are as follows.

Define the operation $\oslash$, for any integers x and non-zero integer y, with:
Divide y into x, if the remainder is not 0, then $x \oslash y$ equals the remainder.
Otherwise: $x \oslash y = y$.
Define the operation $\div$, for any integers x and non-zero integer y, with:
Divide y into x, $x \div y$ equal the answer regardless of the remainder.
The block number b starts from 1 and increases in order, $b = 1, 2, 3, \ldots \ldots$.

If $b \oslash m = i$, the nodes in cluster i need to save the block data, then each node of cluster i calculates $l = [(b - i) \div n] + 1$. The nodes whose ID are $l \oslash n$, $(l + 1) \oslash n$, $\ldots$, $(l + s - 1) \oslash n$ will save a received block respectively.

If the number of nodes is not divisible by n, the number of nodes in the last cluster is greater than n and less than $2n$, and the data storage rules are the same as described above.

In a cluster with n nodes, such a data storage model makes each block stored in only s nodes, and the data stored in each node are theoretically s/n of all blocks in the cluster when b/m is an integer multiple of n. The effect is the same as when the overall data are divided into n parts and each node stores s parts.

Taking $m = 4$, $n = 3$ and $s = 2$ as an example:

1. The first generated block ID is 1, we can obtain $1 \oslash 4 = 1$ according to the storage cluster location formula to determine that the block should be stored in cluster 1, then we get $l = [(1 - 1) \div 4] + 1 = 1$ according to the storage node location formula in a single cluster, then $l \oslash 3 = 1$ and $(l + 1) \oslash 3 = 2$ to determine that the block should be stored in nodes 1 and 2.

2. The next generated block IDs are 2, 3, 4, then according to the storage cluster location formula yields 2⊘4= 2, 3⊘4= 3, 4⊘4= 4, it can be determined that the block should be stored in clusters 2, 3, 4, node storage process is the same as step (1).
3. When the generated block ID is 5, so we get 5⊘4 = 1, and it is determined that the block should be stored in cluster 1, with the result that $l = [(5 - 1)4] + 1 = 2$, then l⊘3= 2, $(l + 1)$⊘3= 3, it is determined that the block should be stored in nodes 2 and 3.

Similarly, when the generated block ID is 9, it is determined that the block should be stored in cluster 1. Then, according to the formula for the location of the storage nodes in a single cluster, $l = [(9 - 1)4] + 1 = 3$, then l⊘3 = 3 and $(l + 1)$⊘3 = 1, it is determined that the block should be stored in nodes 3 and 1.

After the network generates nine blocks, the data storage structure of cluster 1 is shown in Figure 4.

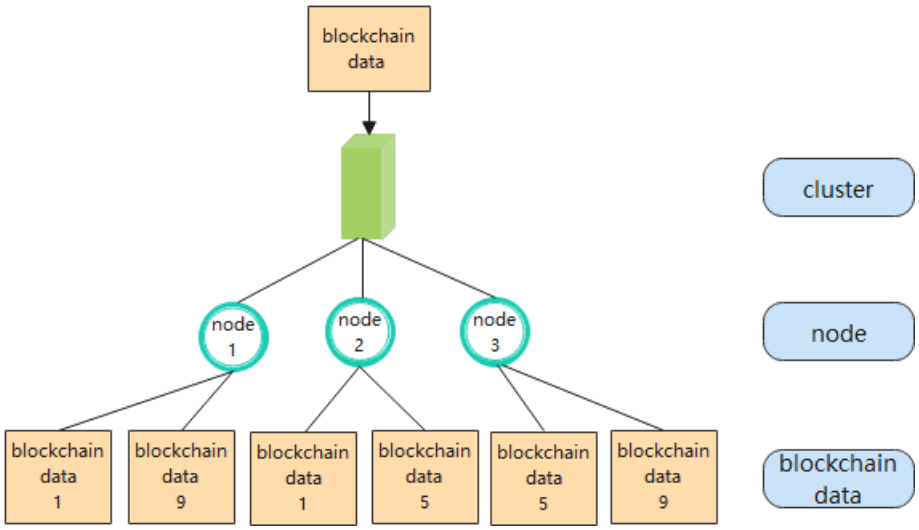

Figure 4. Data storage structure of cluster 1.

4.3.2. Query Process

There are two kinds of queries in Hyperledger Fabric. One is querying the world state. For example, querying the current assets of a user can be done directly by querying the K-V database. Since each peer in our proposal still keeps the complete world state, the process of such a query is exactly the same as that of the Hyperledger Fabric. Another type of query is to look up transaction information, such as a user's asset transfer history. This information is stored in blocks. According to the new storage design scheme, the blocks containing this transaction exist only on a few nodes, making the query process different from the original query mechanism of the Hyperledger Fabric.

The Hyperledger Fabric peer nodes form a P2P network. A gossip module is responsible for connecting the order service to the peer nodes to disseminate messages in the network. The Gossip protocol was proposed in the paper "Epidemic Algorithms for Replicated Database Maintenance" by Alan Demers et al. It is a decentralized distributed protocol in which data spreads through nodes one by one like a virus. Because of the exponential propagation, the overall propagation speed is very fast. However, instead of simply broadcasting, the gossip protocol sends messages to the dominant peer node, which in turn sends messages to some (randomly selected) predetermined number of other peer nodes. The peer nodes that receive the messages then disseminate the message in the same way until all peers receive them. In the Hyperledger Fabric, any peer node on a channel keeps the same complete blockchain, so the first node to receive the query can return the query result. However, in the new proposed mechanism, the blocks on the first peer node that receives the message may not contain the transaction data. As a result, the

data cannot be returned and the message must be forwarded until a peer node that contains the transaction receives the query message, which may cause additional network delays.

5. Scheme Analysis

5.1. Storage Efficiency Analysis

In Hyperledger Fabric, the data stored is mainly the world state and the blockchain. The scaling mechanism proposed in the paper is that each node within a cluster only keeps a portion of the blocks belonging to this cluster but still needs to keep the entire world state database. In blockchain applications, the world state is similar to a traditional database where the current state information of specific entities is stored, so the size of the world state is usually predictable and grows slowly and linearly.

Conversely, the main content saved by blockchain is the transaction information of entities in the world state, similar to database transaction log, and the storage cost grows rapidly. In different types of blockchain applications, the proportion of world state and blocks in the overall storage space will be different. In ring-overlap, only the block data is saved in pieces, and this proportion of the transactions to world state will affect the storage overhead. Let $|B|$ denote the size of the blockchain, $|S|$ was the world state data size, m be the number of clusters, n denote the number of nodes in a single cluster, and s be the number of copies of a block kept in a cluster. For the sake of discussion, only the case where $|B|/m$ is divisible by n is considered, and in the case where $|B|/m$ is much larger than n, the computational difference can be ignored.

In the new scheme, the number of blocks to be stored in each cluster is $(|B|)/m$, and the total amount of data that should be stored by each node is $s/(m \times n) \times |B| + |S|$. Thus, the storage proportion of each node to the case without sharding is:

$$R_v = \frac{\frac{s}{m \times n} \times |B| + |S|}{|B| + |S|} \tag{1}$$

Equation (1) shows that when the block data capacity is much larger than the world state data capacity, R_v tends to be s/mn. If $s = n$, it means that each node stores all the data of this cluster and the storage overhead is maximum; if $s = 1$, it means that the data of the whole network are equally distributed and stored on each node and the storage overhead is minimum.

5.2. Security Analysis

5.2.1. System Reliability

According to the above analysis, it is clear that the smaller the number s is in the cluster, the less storage space is occupied by a single node. However, the smaller the s, the lower the reliability of the system, and the failure of a small number of nodes can result in partial data loss in the blockchain, preventing the ledger from being recovered. Therefore, the size of s is related to the reliability requirements of the system. A larger value of s is required if the system allows more nodes to fail.

In the new proposed scheme, the number of nodes in a single cluster is n, and $1/m$ of the total block data is to be stored. Each block in the cluster is stored repeatedly on s nodes, which means the overall data are the cluster is divided into n parts, and each node stores s part. To simplify the description, only the results after the first round of storage is discussed, and the first n blocked IDs stored in a cluster are normalized to $1, 2 \ldots, n-1, n$. According to the storage rules of the scheme, these n data can be sequentially arranged as a ring R, $1 \rightarrow 2$, $2 \rightarrow 3$, $\ldots$, $n-1 \rightarrow n$, $n \rightarrow 1$. Each node in the cluster stores s consecutive blocks with different starting block ID on R.

The shard IDs of each node in a cluster are different combinations of s consecutive IDs selected from 1 to n. Any two combinations have at least one element different and at most $s-1$ elements different, and each combination is only $n-s$ elements away from completing the entire set of numbers.

In a cluster, the reliability of the system can be understood as the number of nodes allowed to fail without causing data loss; in other words, the data stored in r nodes, which are chosen from n nodes arbitrarily, could cover all blocks. We want to obtain the minimum value of r taken for a particular n.

Worst case: among the selected r nodes, any node α is selected, there exists another node β, and they exist $s-1$ identical elements. In this case, $n-(s-1)$ nodes are needed to obtain all the block data. Therefore, if n nodes, each node stores s shard according to the rules of this scheme, at most $s-1$ nodes can be allowed to fail.

It is known that the system reliability is related to s when n is a fixed value. When $s = 1$, each block exists in only one node, and the failure of any node will lead to an unrecoverable blockchain; when $s = n$, the storage in the cluster degenerates to the Hyperledger Fabric native way, and the full block data can be recovered even if only one node survives.

Since the underlying design used in this scheme is still the traditional P2P design model, its system reliability can be expressed by Equation (2).

$$q = \sum_{k=1}^{s-1} C_n^k \lambda^{n-k} (1-\lambda)^k \tag{2}$$

In Equation (2), q denotes the system reliability, λ denotes the reliability of a single node, s denotes the number of single block storage copies within a single cluster, n denotes the number of nodes within a single cluster, and k denotes the number of failed nodes. When λ is determined, the system reliability q corresponding to different values of s can be found by Equation (2).

5.2.2. Gossiping Guarantees

The Gossip protocol allows for arbitrary additions and subtractions of nodes in the network, and the state of newly added nodes will eventually match that of other nodes. Because of its natural fault-tolerance properties for distributed systems, downtime and restarts of any node in the network have no effect on the propagation of Gossip messages. Thus, the gossip protocol can guarantee that the message gossiped by any honest node will eventually reach all other honest nodes.

5.2.3. Epoch Security

The reliability analysis in Section 5.2.1 shows that the mechanism can still guarantee block data integrity when the number of node failures in a single cluster does not exceed $s-1$. If the reliability of a single node is labeled as λ, when λ is the same, the larger the value of s, the higher the reliability of a single cluster; when s is the same, the failure rate of a single cluster decreases as λ increases. The system reliability q corresponding to different values of s can be calculated by Equation (2).

6. Simulation Experiment

6.1. Experimental Setup

A series of experiments were carried out in order to evaluate the performance of the scheme proposed in the paper. The experimental machine configuration was Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz dual CPU and 256G RAM with CentOS 8 64-bit operating system. The experiments were implemented in the Python language, using python-igraph to create the Watts–Strogatz small-world network, with multiple experiments using different network sizes.

6.2. Experimental Design

The experiments are conducted to simulate the storage efficiency and communication efficiency of this scheme. During the experiments, a small-world network with 100 node sizes ($dim = 5$, $size = 3$, $nei = 1$, $p = 0.1$) was created and the network was divided into four clusters, each of which is set to $s = \lceil 2n/3 \rceil = 17$. That is, s is rounded up by two-thirds of the number of nodes. The structure of the small-world network is shown in Figure 5.

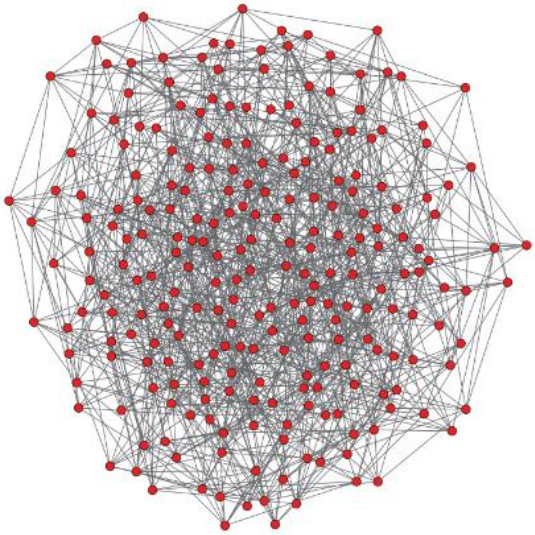

Figure 5. The structure of the generated small-world network.

6.2.1. Experiment 1: Storage Consumed on a Single Node

In Hyperledger Fabric, the ledger consists of two parts: the world state and the blockchain. The world state is the database that stores a set of current values, and the blockchain is the transaction log (transactions) that records all changes resulting in the current world state. If an asset (a record in the world state) only has one corresponding transaction, the storage space consumed by a node is unchanged when compared to Hyperledger Fabric; if an asset has multiple transaction logs, the storage space consumed by a single node will change in the new proposal when compared to Hyperledger Fabric. When each asset corresponds to more and more transaction logs, more and more blocks are generated, but the size of world state increases slowly, which means the proposed mechanism is more dominant, and it will consume less storage space on a single node compared to the traditional storage model on Hyperledger Fabric.

In the network used for the experiments, set the data size S_w for one record of the world state, the data size S_t for one transaction and the block size S_b, ignoring block header. When the number of transactions achieved the block size threshold, a new block is generated and sent to the network, and the peer nodes in the network save the block according to the rules of proposed scheme. Set two variables N_w and N_t, where N_w denotes the number of records in the world state and N_t is the number of transactions. Increasing the value of N_w and N_t sequentially and continuously, until N_t is an integer multiple of S_b, called a round, and records the value of Equation (1).

6.2.2. Experiment 2: Query Efficiency

Because each node in the network keeps the world state, queries to the world state are the same as for Hyperledger Fabric. However, since each node only keeps a portion of the blocks, when a query operation is made for each specific transaction, if the first peer node receiving a query field does not contain the corresponding transaction, the query request needs to be delivered through the gossip protocol until the node containing the transaction receives the request.

In the experiment, using the results of the previous experiments, a node was randomly selected to query a particular transaction. If the node contains the transaction, the network communication overhead of this query is recorded as 1; otherwise, up to $n/3$ neighbors are randomly selected as the next targets to deliver the query request. If the query result is obtained in one hop, the network communication overhead is added by 1. Follow this procedure to execute 1000 queries and record the statistical analysis of the network communication overhead required to obtain the result for each query.

6.2.3. Experiment 3: Storage Cost

With the blockchain network's continuous expansion, the original storage mechanism will cause data explosion in the near future, resulting in a massive waste of resources. Sharding can not only alleviate the storage issue for a single node, but also reduce overall resource consumption from a macro perspective, which is consistent with the low-carbon and energy-saving social strategy.

We use four created small-world networks of different sizes to store 5 MB and 10 MB of data using the Hyperledger Fabric blockchain system, the scalable storage model proposed by Jia et al., and the ring-overlap storage mechanism, respectively. In the experiments, the minimum number of copies of each shard in the scalable storage model is two, and the failure rate of a single cluster node in the ring-overlap storage mechanism does not exceed 33.3%.

6.3. Experimental Result

6.3.1. Experiment 1

During the experiment, S_b is set to 512k according to the default value of Fabric. Each block contains 10 transactions, S_t is 51.2k. The experimental results are shown in Figure 6.

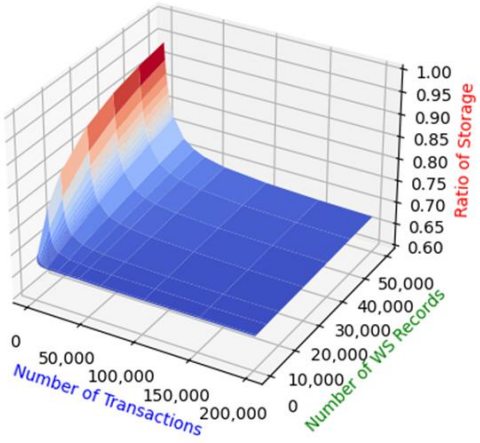

Figure 6. The result of experiment 1.

In Figure 6, the x-axis represents N_w, the y-axis represents N_t, and the z-axis represents the value of the node storage rate R_v. It can be seen that R_v approaches s/mn, when N_t is much larger than N_w in clusters of various sizes.

6.3.2. Experiment 2

The experimental results are shown in Figure 7 and the statistic result is shown in Table 2. In Figure 7, the x-axis is the number of experiments, and the y-axis is the network communication overhead. The graph shows that most of the queries access network overheads greater than 1, and the average value is about 3.4. From Table 2, we can know that most of the query can obtain a result in 4 hops, though the longest route contains 6 hops.

Table 2. The statistic result of experiment 2.

Hops	Frequency
1	150
2	52
3	260
4	346
5	173
6	19

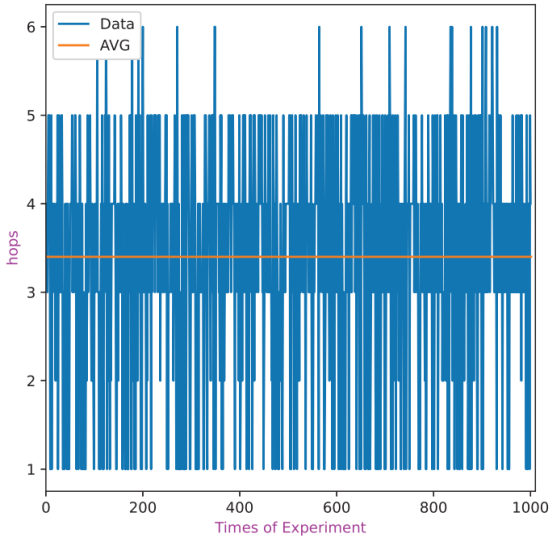

Figure 7. The statistic result of query efficiency.

6.3.3. Experiment 3

The experimental results are shown in Figure 8, where the x-axis is the number of nodes and the y-axis is the total amount of storage. The graph shows that when the number of nodes increases, the storage space occupied by the ring-overlap mechanism is significantly reduced compared to the Fabric blockchain system and the scalable storage model.

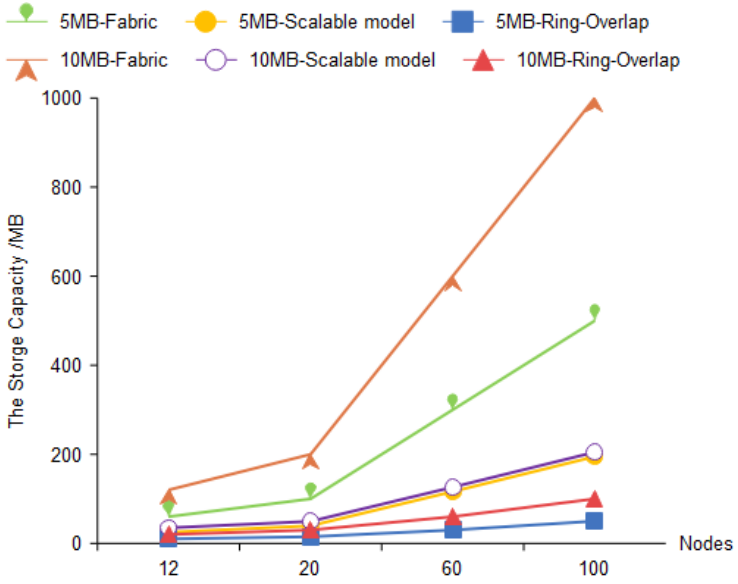

Figure 8. Storage space occupied by Ring-Overlap with Hyperledger Fabric and scalable model.

Despite the additional network overhead introduced by this solution, in a real Hyperledger Fabric application, unless the network state is extremely poor and network communication takes up more time in the whole query transaction process, the increased network communication overhead is well worth it relative to the storage space savings.

6.4. Analysis of Experimental Results

From Equation (1), it is known that the smaller the s, the lower the storage rate; however, the number of tolerable failure nodes becomes smaller, and the system reliability decreases. This requires a careful choice of s versus n when designing the scheme to compromise between the storage rate and system reliability.

Take a network of 100 nodes with 25 nodes per cluster as an example. Let the number of transactions be large enough to calculate the storage rate in terms of s/mn. Table 3 lists the storage ratios and the number of tolerant failure nodes of the system corresponding to different s values. For clarity purposes, the n is omitted in calculating the result, which means that we only focus on the storage ratio on a single node.

Table 3. Storage ratios and the number of tolerant failure nodes.

Overlapping Block Number s	Storage Rate	Number of Failure Tolerant Nodes
2	8%	1
3	12%	2
4	16%	3
5	20%	4
6	24%	5
7	28%	6
8	32%	7
9	36%	8
10	40%	9
11	44%	10
12	48%	11
13	52%	12
14	56%	13
15	60%	14
16	64%	15
17	68%	16
18	72%	17
19	76%	18
20	80%	19
21	84%	20
22	88%	21
23	92%	22
24	96%	23

As can be seen from Table 3, in a cluster of 25 nodes, if $s = 2$, a block is repeatedly kept in 2 copies within a cluster, only 1 node failure in the system can be tolerated; when $s = 3$, 2 nodes can be tolerated to fail at the same time, and such a result is already able to provide high system reliability.

Assuming that the reliability λ of nodes in the blockchain network is 0.9, 0.95, and 0.99, the probability of node failure is 0.1, 0.05, and 0.01. In a network of 100 nodes divided into four clusters, when s takes different values, the reliability q of the system can be calculated according to Equation (2), and the system reliability is shown in Figure 9.

By analyzing the experimental results in Figure 9, the following can be concluded. When λ is the same, the larger the value of s, the higher the reliability of the system. When $\lambda = 0.95$, the number of single block storage copies in a single cluster needs to be greater than eight for the system to reach 99.9% reliability. When $\lambda = 0.99$, the number of single block storage copies in a single cluster only needs to be greater than four for the system reliability to reach 99.9%. This indicates that the higher the node reliability, the more applicable the scheme is. Combining Table 3 and Figure 9, we know that when $\lambda = 0.95$, the storage rate only needs to be 32% for the system reliability to reach 99.9%. Almost 2/3 of the storage space is saved compared with the original Hyperledger Fabric storage. This experimental result demonstrates that the scheme is capable of effectively scaling Hyperledger Fabric storage.

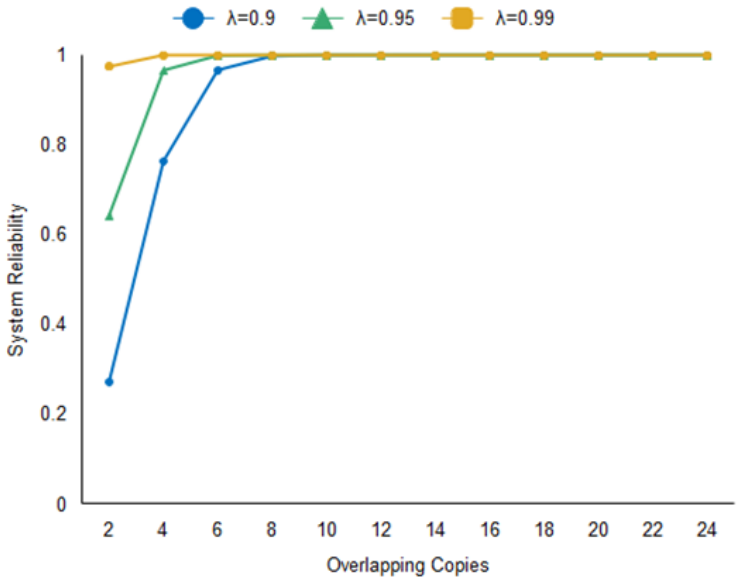

Figure 9. System reliability.

7. Conclusions

In blockchain networks, all nodes need to synchronize all block data to participate in the blockchain, which causes the exhaustion of storage resources. Therefore, this paper proposes a ring-overlap storage scaling mechanism for consortium blockchain, which employs double-sharding to reduce blockchain node storage capacity. By reasonably selecting the number of copies of a block to be repeatedly saved, the storage limitation problem of the blockchain can be effectively solved under the premise of ensuring system reliability.

Of course, the design of this solution introduces some additional network communications overhead for transaction queries. However, if specific transactions are not frequently queried, these added communication overheads will not affect the system too much in the current good network infrastructure.

For the Hyperledger Fabric consortium chain design, the scheme can be used as a reference for blockchain systems that do not need a cross-verify mechanism. One drawback of our scheme is that it is inapplicable to blockchains that use PoW consensus. This is because such blockchain would involve cross-block consensus. We will continue to improve this scheme in the future so that it can be applied to other blockchain platforms.

Author Contributions: Conceptualization, W.L. and J.Z.; investigation and methodology, W.L. and J.Z.; writing of the original draft, W.L.; writing of the review and editing, W.L. and J.Z.; validation, C.M.; software, X.Z.; data curation, W.L. and D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by the National Natural Science Foundation of China (No.61972360) and Shandong Provincial Natural Science Foundation (No.ZR2020MF148).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 22 March 2021).
2. Decker, C.; Seidel, J.; Wattenhofer, R. Bitcoin meets strong consistency. In Proceedings of the 17th International Conference on Distributed Computing and Networking (ICDCN '16), Singapore, 4–7 January 2016; pp. 1–10.
3. Gilad, Y.; Hemo, R.; Micali, S.; Vlachos, G.; Zeldovich, N. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17), Shanghai, China, 28 October 2017; pp. 51–68.
4. Yu, G.; Wang, X.; Yu, K.; Ni, W.; Zhang, J.A.; Liu, R.P. Survey: Sharding in Blockchains. *IEEE Access* **2020**, *8*, 14155–14181. [CrossRef]
5. Zhang, P.; Zhou, M.; Zhen, J.; Zhang, J. Enhancing Scalability of Trusted Blockchains through Optimal Sharding. In Proceedings of the 2021 IEEE International Conference on Smart Data Services (SMDS), Chicago, IL, USA, 5–10 September 2021; pp. 226–233.
6. Wang, G. RepShard: Reputation-based Sharding Scheme Achieves Linearly Scaling Efficiency and Security Simultaneously. In Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, 11 December 2020; pp. 237–246.
7. Underwood, S. Blockchain beyond Bitcoin. *Commun. ACM* **2016**, *59*, 15–17. [CrossRef]
8. Li, C.L.; Zhang, J.; Yang, X.M. Scalable blockchain storage mechanism based on two-layer structure and improved distributed consensus. *Supercomput* **2022**, *78*, 4850–4881. [CrossRef]
9. Liu, W.X.; Zhang, D.H.; Zhao, J.D. Ring-Overlap: A Storage Scaling Mechanism for Consortium Blockchain. In Proceedings of the 2022 International Conference on Service Science (ICSS), Zhuhai, China, 13–15 May 2022; pp. 33–40. [CrossRef]
10. Min, X.P.; Li, Q.Z.; Kong, L.J.; Zhang, S.D.; Zheng, Y.Q.; Xiao, Z. Permissioned Blockchain Dynamic Consensus Mechanism Based Multi-centers. *Chin. J. Comput.* **2018**, *41*, 1005–1020.
11. Eyal, I.; Gencer, A.E.; Sirer, E.G.; Renesse, R.V. Bitcoin-NG: A Scalable Blockchain Protocol. In Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI' 16), Santa Clara, CA, USA, 16–18 March 2016; pp. 45–59.
12. Luu, L.; Narayanan, V.; Zheng, C.D.; 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 (CCS '16), Vienna, Austria, 24–28 October 2016; pp. 17–30. [CrossRef]
13. Kokoris-Kogias, E.; Jovanovic, P.; Gasser, L.; Gailly, N.; Syta, E.; Ford, B. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 583–598.
14. Zamani, M.; Movahedi, M.; Raykova, M. RapidChain: Scaling Blockchain via Full Sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), Toronto, ON, Canada, 15–19 October 2018; pp. 931–948. [CrossRef]
15. Wang, J.P.; Wang, H. Monoxide: Scale out blockchain with asynchronous consensus zones. In Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation (NSDI'19), Boston, MA, USA, 26–28 February 2019; pp. 95–112.
16. Danezis, G.; Meiklejohn, S. Centrally Banked Cryptocurrencies. In Proceedings of the 23rd Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA, USA, 21–24 February 2016; pp. 1–14.
17. Gencer, A.E.; van Renesse, R.; Sirer, E.G. Short paper: Service-oriented sharding for blockchains. In Proceedings of the International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017; pp. 393–401. [CrossRef]
18. Nguyen, L.N.; Nguyen, T.D.T.; Dinh, T.N.; Thai, M.T. OptChain: Optimal Transactions Placement for Scalable Blockchain Sharding. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 525–535. [CrossRef]
19. Niu, J.Y.; Wang, Z.Y.; Gai, F.Y.; Feng, C. Incentive Analysis of Bitcoin-NG, Revisited. *ACM SIGMETRICS Perform. Eval. Rev.* **2020**, *48*, 59–60. [CrossRef]
20. Mao, Z.L.; Liu, Y.N.; Sun, H.P.; Chen, Z. Research on Blockchain Performance Scalability and Security. *Netinfo Secur.* **2020**, *20*, 56–64.
21. Micali, S.; Rabin, M.; Vadhan, S. Verifiable Random Functions. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), New York, NY, USA, 17–19 October 1999; pp. 120–130. [CrossRef]
22. Wang, H.; Wang, L.C.; Bai, X.; Liu, Q.H.; Shen, X.Y. Research on Key Technology of Blockchain Privacy Protection and Scalability. *J. Xidian Univ.* **2020**, *47*, 28–39.
23. Zhang, X.H.; Niu, B.N.; Gong, T. Account-based Blockchain Scalable Storage Model. *J. Beijing Univ. Aeronaut. Astronaut.* **2022**, *48*, 708–715.
24. Zheng, P.; Xu, Q.; Zheng, Z.; Zhou, Z.; Yan, Y.; Zhang, H. Meepo: Sharded Consortium Blockchain. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 1847–1852. [CrossRef]
25. Kim, T.; Lee, S.; Kwon, Y.; Noh, J.; Kim, S.; Cho, S. SELCOM: Selective Compression Scheme for Lightweight Nodes in Blockchain System. *IEEE Access* **2020**, *8*, 225613–225626. [CrossRef]
26. Liu, Y.; Sun, H.P.; Song, X.; Chen, Z. OverlapShard: Overlap-based Sharding Mechanism. In Proceedings of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 05–08 September 2021; pp. 1–7.

27. Li, M.; Qin, Y.; Liu, B.; Chu, X. A Multi-node Collaborative Storage Strategy via Clustering in Blockchain Network. In Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 1275–1280.
28. Jia, D.Y.; Xin, J.C.; Wang, Z.Q.; Guo, W.; Wang, G.R. Storage Capacity Scalable Model for Blockchain. *J. Front. Comput. Sci. Technol.* **2018**, *12*, 525–535.
29. EI Azzaoui, A.; Choi, M.Y.; Lee, C.H.; Park, J.H. Scalable Lightweight Blockchain-Based Authentication Mechanism for Secure VoIP Communication. *Hum.-Cent. Comput. Inf. Sci.* **2022**, *12*, 8. [CrossRef]
30. Fan, X.; Niu, B.N.; Liu, Z.L. Scalable blockchain storage systems: Research progress and models. *Computing* **2022**, *104*, 1497–1524. [CrossRef]
31. Ren, L.; Ward, P.A.S. Understanding the Transaction Placement Problem in Blockchain Sharding Protocols. In Proceedings of the 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 27–30 October 2021; pp. 0695–0701.
32. Fajri, A.I.; Mahananto, F. Hybrid lightning protocol: An approach for blockchain scalability issue. *Procedia Comput. Sci.* **2022**, *197*, 437–444. [CrossRef]
33. Yang, C.L.; Li, X.X.; Li, J.J.; Qian, H.F. Linear Scalability from Sharding and PoS. In Proceedings of the Algorithms and Architectures for Parallel Processing, New York, NY, USA, 2–4 October 2020; pp. 548–562.
34. Li, M.Y.; Qin, Y.; Liu, B.; Chu, X.W. Enhancing the efficiency and scalability of blockchain through probabilistic verification and clustering. *Inf. Processing Manag.* **2021**, *58*, 102650. [CrossRef]