Computers, Materials & Continua
DOI: 10.32604/cmc.2022.028305
Article

CLEC: Combination Locality Based Erasure Code for Permissioned Blockchain Storage

Jiabin Wu[1,3], Boai Yang[2], Yang Liu[1], Fang Liu[3,*], Nong Xiao[1] and Shuo Li[4]

[1]National University of Defense Technology, Changsha, 410073, China
[2]Hunan University, Changsha, 410082, China
[3]Sun Yat-sen University, Guangzhou, 510275, China
[4]The University of Edinburgh, Edinburgh, EH8 9BT, The United Kingdom
*Corresponding Author: Fang Liu. Email: fangl@hnu.edu.cn
Received: 07 February 2022; Accepted: 31 May 2022

Abstract: Building a new decentralized domain name system based on blockchain technology is helping to solve problems, such as load imbalance and over-dependence on the trust of the central node. However, in the existing blockchain storage system, the storage overhead is very high due to its full-replication data storage mechanism. The total storage consumption for each block is up to O(n) with n nodes. Erasure code applied to blockchains can significantly reduce the storage overhead, but also greatly lower the read performance. In this study, we propose a novel coding scheme for blockchain storage, Combination Locality based Erasure Code for Permissioned blockchain storage (CLEC). CLEC uses erasure code, parity locality, and topology locality in blockchain storage, greatly reducing reading latency and repair time. In CLEC, the storage consumption per block can be reduced to O(1), and the repair penalty can also be lowered to O(1). Experiments in an open-source permissioned blockchain Tendermint show that CLEC has a maximum repair speed of 6 times and a read speed of nearly 1.7 times with storage overhead of only 1.17 times compared to the current work, a great improvement in reading performance and repair performance with slightly increased storage overhead via implementation.

Keywords: CLEC; blockchain; erasure code; delay circle; read performance; throughput; repair performance

1 Introduction

Domain name system is an important infrastructure of the Internet. Its main function is to establish the mapping between domain names and IP addresses so that users can access other applications on the Internet through domain names. According to the 47th Statistical Report on The Development of Internet in China released by China Internet Network Information Center (CNNIC) in February 2021, the number of Internet users in China had reached 989 million, and the scale of Internet applications is still growing steadily, with significant growth in users of live streaming, short videos, and online shopping. Internet applications have been integrated into our modern life, providing convenience to the public while putting greater pressure on basic Internet applications.

The Domain Name System (DNS) system provides a domain name resolution service for Internet queries from all over the world. China's top-level domain name ".cn" is requested more than 1 billion times every day. However, the DNS protocol was not secure enough at the beginning of design, resulting in frequent malicious attacks on the DNS system. Attacks such as Distributed denial of service attack (DDos), domain name hijacking, and domain name poisoning have brought huge losses to all parties [1,2]. The vulnerability of the DNS system makes it very important to design a new domain name system.

Blockchain, as an append-only ledger maintained by all nodes, can work in an untrusted environment. To prevent blocks from being changed by malicious nodes, blocks in a blockchain are joined by cryptographic hashes in the block header. In general, traditional blockchains require a complete copy of the blockchain data on each node, as well as consensus protocols, such as the Pow protocol used in public blockchains, and the Practical Byzantine Fault Tolerance (PBFT) protocol most commonly used in permissioned blockchains [3] which ensures data consistency across the network. However, for each block, the storage overhead is O(n), where n is the number of nodes in the network. For such a full-replication blockchain, each node needs to spend a large amount of storage overhead to preserve the entire chain. For example, the size of the Bitcoin blockchain is 360 GB at present and is steadily increasing by 0.2 GB per day. Such a large storage overhead is undoubtedly a huge threshold for others to join the blockchain. Ultimately, only enterprises or individuals with enough storage resources can have the qualification to join the network, which virtually forms a monopoly situation of giants and destroys the original intention of decentralization [4–6].

1.1 Motivation

The BFT-Store scheme [7] proposed by the Qi team provides a method to combine blockchain with erasure code technology. It performs erasure coding on the complete blockchain data and stores erasure coding chunks on different nodes, and this greatly reduces the storage overhead. However, to read a block, the data needs to be fetched from the node where the block is stored, rather than locally, as is the case with traditional blockchain storage. Therefore, when a node fails (e.g., when it refuses to respond to a read request or gives an incorrect data chunk), a recovery request needs to be sent to all nodes. When receiving the recovery request, each node will send the locally stored chunk to the requesting node. Only after the requesting node collects the correct $n - 2f$ chunks and carries out the recovery process, can the data chunk be recovered. In PBFT protocol, the probability of chunk failure is 1/3 at most. Each failure requires $n - 2f$ chunks to be transmitted in the network, i.e., $f + i(1 \leq i \leq 3)$ chunks. This will undoubtedly result in significant network overhead. However, in Zilliqa [8] protocol and Ripple protocol, the probability of chunk failure is about 1/4 and 1/5 respectively, and the repair penalty is $2f + i(1 \leq i \leq 4)$ and $3f + i(1 \leq i \leq 5)$ chunks respectively. Undoubtedly, in the latter two cases, it is very important to reduce the repair penalty of node failure. In BFT-Store, by setting multiple copies, the probability of chunk lost is reduced, the throughput rate of the system is improved, and the read delay is reduced. But it also multiplies the storage overhead. Therefore, it is reasonable to make some improvements in the coding scheme.

The Reed-Solomon code (RS code) is used in the BFT-Store, which is the most common coding scheme. Reference [9] describes a wide strip coding scheme that calculates how many chunks should be placed in each rack and generates a local parity chunk in each group to balance storage overhead and repair performance. The method proposed in this article is the trade-off of Parity Locality and

Topology Locality. It reduces storage overhead and ensures a small repair penalty when a node fails. The latency of communication within the same rack is low, so chunks in the same rack are combined and then transmitted. But the context of the article is set in a non-Byzantine environment which is not suitable for blockchain storage. Credibility and reliability are highly similar for nodes within the same rack, i.e., when a node fails (attacked or powered down), there is a high probability that other nodes in the same rack will fail. However, in the blockchain environment, the reliability and trustworthiness of each node are independent of each other, and there should be no strong correlation between each other. Because chunks are assembled inside a rack and then transferred out, the correctness of the chunks generated during this process cannot be promised. Because in the consensus process of blockchain, only the hash value of each chunk is retained. The combined chunk is a completely new chunk whose information is not recorded by the blockchain.

1.2 Contribution

The main contributions of this work are described as follows:

(1) We propose Combination Locality Based Erasure Code for Permissioned Blockchain Storage (CLEC). CLEC improves the coding scheme based on blockchain storage. In this scheme, each node in the blockchain belongs to a certain delay circle. You can select each *r* delay circle as a group and generate a local parity in each group. When chunk repair is needed, the chunks within the same group are combined and then transmitted in the form of a combined chunk to the node with the missing chunk. This greatly reduces the transmission cost across the delay circle.

(2) We do not store the combined chunks in each delay circle, but we can judge the correctness of each combined block in the recovery process. Every time to repair, the node that loses the block (the target node) requests all the nodes in the group to repair the block. In each delay circle of the group, a lead node is selected, and the lead node collects all chunks in the delay circle to synthesize a new combined chunk and transmit it to the target node. However, a problem is introduced, that the reliability of the combined chunk that each lead node transmits to the target node is not guaranteed, and the newly generated combined chunk is not stored on any node beforehand, i.e., there is no way to check whether the combined chunk is correct. Therefore, we in the process of encoding, also generate and record the hash value of the combined chunk at the same delay circle, but do not keep the combined chunk. When the target node receives the combined chunk from other delay circles, can determine whether it is correct.

The structure of the rest of paper is composed of the following four sections. Section 2 is about related works. Section 3 presents our proposed CLEC system, including description and some explanations of the system, delay circle division, node architecture, coding scheme, an important algorithm of recover to read, and the improved read performance and repair performance analysis for CLEC. In Section 4, we present the evaluation of our system. The last section concludes this paper.

2 Related Work

There are 13 DNS root servers in the world, including 10 in the United States, 2 in Europe, and 1 in Japan, which shows the highly centralized characteristics of root servers and may lead to a security crisis. Namecoin is designed to replace root servers with a blockchain for mapping domain names to DNS records. On basis of that, blockstack [10] was proposed by M. Ali, a new blockchain-based naming and storage system that powers a production public-key infrastructure system for 55,000 users.

Blockchains that was first proposed by Satoshi Nakamoto, the inventor of Bitcoin, provide an append-only ledger maintained by all nodes. Blockchains have the advantages of decentralization, independence, security, openness and anonymity. However, it also occupies too many computing and storage resources [11].

To solve the problem of the high storage overhead of blockchain, many solutions have been proposed. For example, Zyskind [12] and other scholars use the method of off-chain storage to separate data from data abstract. Only the abstract generated by SHA-256 is retained on the blockchain, and the corresponding data of this abstract is stored in the distributed object storage based on a distributed hash table. Zheng [13] and other scholars proposed an IPFS-based blockchain data storage model to solve the storage bottleneck of blockchain technology. In this model, miners store transaction data in the IPFS network and package the returned transaction IPFS hash into blocks, which significantly reduces the amount of data stored in the blockchain. Both methods transfer the storage pressure shared by all nodes in the on-chain storage to the off-chain storage. Although the storage pressure of each node is reduced and the entry threshold of nodes is lowered, the security of data is also reduced at the same time. That is, when the data stored under the chain is tampered with, the stored data can only be found to be wrong through the blockchain, and the correct data cannot be recovered through the blockchain.

In addition to off-chain storage, it is more common to use on-chain storage. To reduce the storage overhead of on-chain storage, a team proposed a collaborative storage method based on grouping [14], that is, the whole network is divided into multiple groups, and each group stores part of the data. In the same group, each node also keeps a full copy of some data chunks. In addition, erasure code is also a novel idea for on-chain data chunks. Perard D and other scholars proposed that a block could be encoded into chunks and distributed to each node for storage. However, such a method destroys the integrity of the block. Every time to read a block, users cannot read it directly but need to collect enough chunks that the block generates from the network and combine or decode for reading, resulting in poor reading performance [15]. Xiaodong Qi's team has also proposed a solution combining erasure code with blockchain [7]. Different from the Perard D team, the Qi team did not encode a block into individual chunks, but unified coding after collecting certain blocks. Compared with the method of block coding, the advantage of this method is to ensure the integrity of the block. When you want to read a block, you do not need to collect all data chunks from the network, but only need to find the node where the corresponding data chunk is stored and to read. In addition, to improve reading performance, the Qi team used multi-copy and caching technologies and considered the recoding problem when a node joins or exits.

To improve repair performance, many studies exploit either parity locality or topology locality to improve the performance of erasure coding. In terms of parity locality, locally repairable codes [16,17] reduce the repair bandwidth and I/O costs by associating local parity chunks with different groups of fewer than k data chunks. Product codes [18–20] associate local parities with both horizontal and vertical groups of data chunks for high fault tolerance. Several studies exploit hierarchical parity locality to associate local parity chunks with different levels of groups of data chunks to handle multiple failures [21,22]. In terms of topology locality, existing studies exploit rack-level locality to reduce cross-rack data transfers in repair or update operations. Some studies propose repair-optimal erasure code constructions [23,24] that minimize the cross-rack repair bandwidth.

3 System Design

3.1 System Architecture

As shown in Fig. 1, the whole system is divided into several delay circles and each node belongs to a certain delay circle according to some factors such as geographical distance or communication latency. For each node, the specific system architecture is shown in Fig. 2. Compared with traditional blockchain, each node has an erasure coding layer. These are explained below.

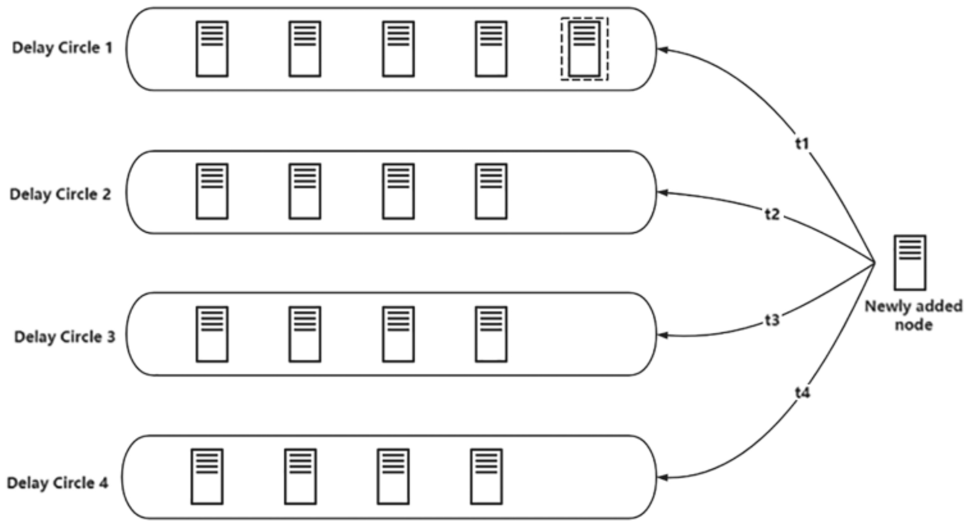


Figure 1: Delay circles division of nodes

3.2 Delay Circle Division

Algorithm 3.1 shows the process when a new node is added to CLEC. We will choose the delay circle for it. The node first needs to send a message to each node in the network, then calculate the latency of the message to each node, and find the node N with the lowest communication latency. If the latency is smaller than $t_{threshold}$, the newly added node N_{new} will be into the delay circle where N resides; otherwise, a new delay circle will be created and this node is added to it.

Algorithm 3.1 delay circle division

```
Input: $N_{new}$
Output: Delay Circle DC
1    $t_{ping} = N_{new}$ ping a big packet to every node;
2    get min($t_{ping}$) and $N_{\mathrm{minPing}}$ //$N_{\mathrm{minPing}}$ is the node with the smallest ping time
3    if min($t_{ping}$) $< t_{threshold}$:
4        DC = which DelayCircle Node_minPing is in.
5    else:
6        DC = new DelayCircle()
7    return DC
```

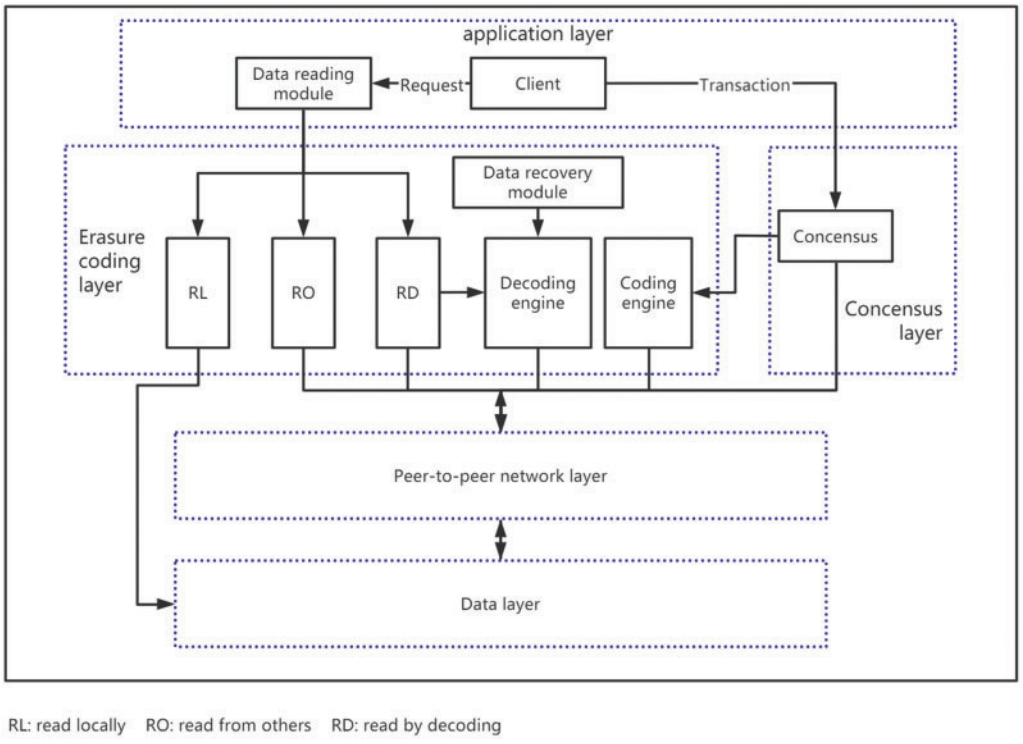


Figure 2: Single-node architecture

3.3 Node Architecture

As shown in Fig. 2, a node architecture is mainly composed of the application layer, erasure layer, data layer, etc. The block confirmed by the consensus algorithm will be broadcast to the network, and part of the data will be retained by erasure coding.

A node caches the blocks validated by the consensus algorithm locally until the number of blocks reaches the encoding threshold, and then encodes the blocks and chooses to retain some of the data. After encoding, each node only preserves a certain chunk. Once a node receives the access request, it will carry out three kinds of data access through the data reading module, namely local reading, remote reading, and decoding reading. If the accessed block is stored locally, the corresponding data is returned directly. If the block accessed is not local, the remote node will be accessed to obtain the corresponding block. However, if the block is not local and the remote target node does not respond, the node performs data recovery through the decoding engine and calculates the corresponding block. The data recovery module can recover blocks in the event of nodes failure. Through the decoding engine, the block can be recovered. There are 2 modes of recovery, i.e., global repair and local repair. When there is only one node failure in a group, local repair using local parity starts, otherwise global repair collects enough global parities and data chunks to recover the block.

3.4 Coding Scheme

The important symbols used in this part are shown in Tab. 1. CLEC does not encode a block into various chunks separately, because this will cause that every time a block is read, it needs to collect all data chunks from the network and splice them into a complete block, which will lead to a significant decline in reading performance. After $n - 2f - lp + 1$ blocks are collected, they are encoded into N chunks, including $n - 2f - lp + 1$ data chunks, $2f - 1$ global parities, and lp local parities. There are several delay circles in a group, each chunk is placed in a node in a group, and a local parity is arranged for each group. When a node fails, users only need to fetch chunks from nodes in the group to recover the target block.

Table 1: The important symbols used in this part

Symbol	Description	Symbol	Description
α	Fault factor	lp	Number of local parities
g	Group, containing several delay circles	dc	Number of delay circles
gp	Number of global parities	n	Number of nodes
γ	Maximum storage redundancy	ddc	Number of delay cycles containing data blocks

The coding process consists of three steps: 1. k blocks are collected; 2. encode them into k data chunks, $2f - 1$ global parities, and generate a local parity only for each group containing data chunks. Each node holds the corresponding chunk and its hash values. This rule is valid for all nodes, and each node knows what chunks are in its delay circle and its group. In addition, each node will xor each chunk in the delay circle containing data chunks, calculate a combined chunk but only store its hash value. 3. each node retains the corresponding data chunk or parity chunk based on the placement algorithm and the metadata of all chunks, including the combined block of each delay circle, but does not store the combined chunk described in the previous step. Computing combined chunks and saving their metadata in preparation for recovery reads is described in more detail in 3.3. Besides, according to 3.1, when a node cannot obtain the target block locally or from the node preserving this block, the recovery (repair) process is started.

Without considering Byzantine attack nodes and with the introduction of local parities, the system can resist $gp + 1$ nodes failure [17]. So, we can get:

$$n = k + lp + f - 1 \tag{1}$$

$$gp = f - 1 \tag{2}$$

$$\gamma \geq \frac{n}{k} \tag{3}$$

According to Eqs. (1) and (3), we can get:

$$lp \leq n(1 - \frac{1}{\gamma}) - f + 1 \tag{4}$$

The number of nodes in each delay circle may be different, and the grouping can only be based on the delay circle. We divide nodes into lp groups, and only groups containing data chunks need a local parity. To balance the number of delay circles contained in each group, the first *ddc modlp* groups contain $\left\lceil \frac{ddc}{lp} \right\rceil$ delay circles, and the last $ddc - (ddc\ modlp)$ groups contain $\left\lfloor \frac{ddc}{lp} \right\rfloor$ delay circles.

Reference [7] proves that: to resist f malicious nodes, if RS coding is adopted, the coding scheme RS(n − 2f, 2f) should be used. In the blockchain scenario, the proportion of data chunks to nodes must be greater than $\frac{1}{\alpha}$, so $k = (\alpha - 2)f + i(1 \le i \le f)$. In the blockchain environment, due to the existence of malicious nodes, the above various adjustments need to be made [25,26]:

$$n = \alpha f + i + lp - 1(1 \le i \le f) \tag{5}$$

$$gp = 2f - 1 \tag{6}$$

$$f \le \frac{n - lp}{\alpha} \tag{7}$$

$$lp \le n(1 - \frac{1}{\gamma}) - 2f + 1 \tag{8}$$

From Eq. (5), we can get $f = \frac{n - lp - i + 1}{\alpha}$, and thus we can get Eq. (7) due to $1 \le i \le f$. Take the maximum of f, namely, $f = \left\lfloor \frac{n - lp}{\alpha} \right\rfloor$. The more lp, the faster the recovery of the single chunk failure process and the lower the bandwidth of the repair across the delay circle needs, so $lp = \left\lfloor n(1 - \frac{1}{\gamma}) - 2\left\lfloor \frac{n-lp}{\alpha} \right\rfloor + 1 \right\rfloor$. The number of global parities is 2f − 1, and the number of data chunks is $n - lp - 2f + 1$. Take $n = 20$, $\gamma = 1.7$, $dc = 7$, $ddc = 5$ for example, set $\alpha = 5$, i.e., in Ripple consensus, we can get $f = 3$, $lp = 3$, $gp = 5$, $k = 12$. And the delay circles containing data chunks are divided into three groups. The first 2 groups contain two delay circles and the third group contains just one delay circle, as shown in Fig. 3.

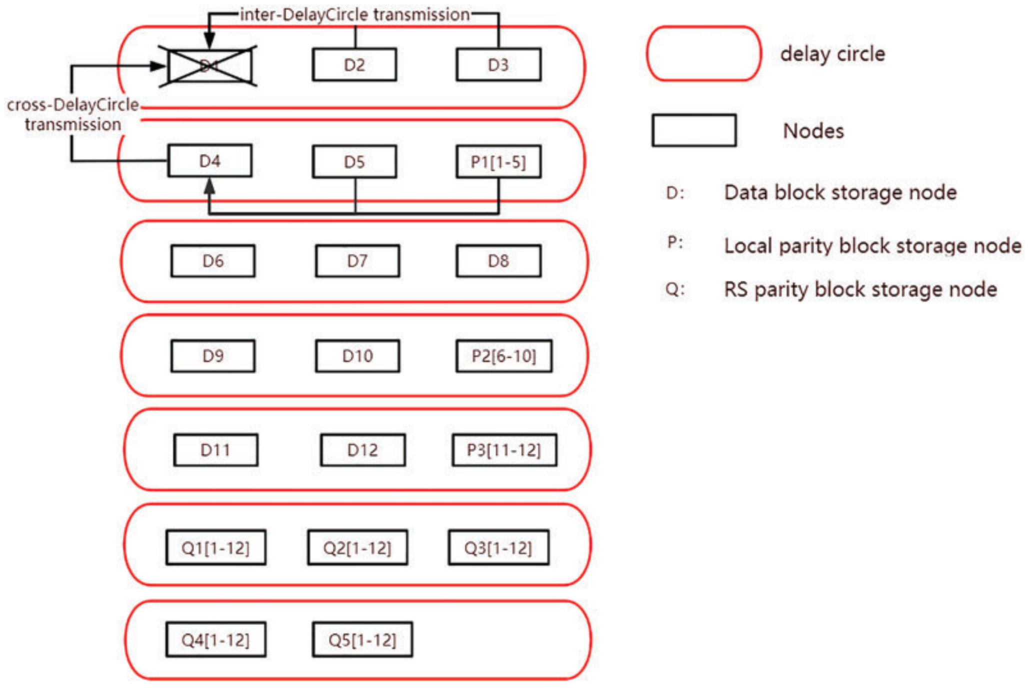

inter-DelayCircle transmission
cross-DelayCircle transmission
D1
D2
D3
D4
D5
P1[1-5]
D6
D7
D8
D9
D10
P2[6-10]
D11
D12
P3[11-12]
Q1[1-12]
Q2[1-12]
Q3[1-12]
Q4[1-12]
Q5[1-12]
delay circle
Nodes
D: Data block storage node
P: Local parity block storage node
Q: RS parity block storage node

Figure 3: Example for CLEC

3.5 Recover to Read

When a client initiates a read request, it first checks whether the target block exists on the local machine. If not, it searches for the location of the target block based on the local metadata information and obtains the data chunk from the target node. When a node fails, the client starts the recovery reading process. This section focuses on the recovery reading.

Algorithm 3.2 Recover to read

```
Input: $N_c$ initiate a recovery read request <recover, name of target block>
Output: target block $B_t$
1    Get the target group where the target node $N_t$ in
2    Combine every chunk in each delay circle using local parity
3    Transmit combined chunk to $N_c$
4    if receiving enough right combined chunks:
5        Recover to get target block
6    else:
7        Send request <recover, name of target block, global>
8        While receiving n − 2f correct chunks:
9            Recover to get target block
10   Return target block
```

As shown in algorithm 3.2, N_c is the node that initiates the read request. When N_c cannot be obtained directly from the local node or other nodes, a recovery read request <recover, name of target block> will be sent to all nodes in the group of the target node N_t. The 1–3 lines refer to through the metadata N_c stored locally, we can know which node N_t with the target chunk B_t is and its group, and each chunk in the group can be combined in each delay circle. As local parity is used in the group, only xor operation is required for the combination of each chunk. Lines 4–5 refer to that each delay circle in the group sends a combined chunk to the target node, and the node starts the recovery process after receiving enough correct combined chunks. There is a problem here, that is, the combined chunk of each delay circle is not stored on the node, so it is hard to judge whether it is correct or not. To solve this problem, in the process of coding, each node will calculate the combined chunk of every delay circle and its hash value, and store the hash value, but do not store the combined chunk. Therefore, in the process of recovery, when the combined chunk is received from other groups, its correctness can be verified by the hash of the previously stored combined chunk.

3.6 Improved Read Performance and Repair Performance

Compared with BFT-Store, this scheme greatly reduces the repair cost. In BFT-Store, n−2f blocks need to be taken from the network during node repair, and the maximum probability of node failure reaches $\frac{1}{\alpha}$. In PBFT, Zilliqa, and Ripple protocols [27–31], the values are $\frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ respectively. This is undoubtedly a great burden on the network overhead of the system. However, in CLEC, the repair cost of node failure is equal to the transmission time of each block within each delay circle plus the time of sending the combined chunk to the requesting node. However, the transmission time within the delay circle is far less than the transmission time across the delay circle, so the former can be almost ignored. Therefore, in CLEC, the network cost of chunks repair is approximately the number of delay circles within a group, which is much lower than that of the BFT-Store scheme.

For example, in Ripple, when the total number of nodes is 20, the BFT-Store scheme is RS(14,6). Therefore, when a chunk is missing, 14 chunks need to be transmitted from the network to recover the block, consuming 14 chunks of network bandwidth, and having to fetch chunks from different nodes leads to high latency. In the case shown in Fig. 3, only one combined chunk is needed to be transmitted across the delay circle, plus the transmission chunk within the delay circle, the network bandwidth consumed is 5 chunks. In addition, the transmission speed within the delay circle is fast, and the transmission speed across the delay circle is low, which will greatly reduce the transmission time. Therefore, the read performance and repair performance can be improved.

4 Evaluation

We implemented BFT-Store with CLEC on Tendermint, an open-source blockchain system. At the network layer, each node maintains a TCP connection with its peers, so that all nodes can communicate with each other via P2P protocol. To conduct a comprehensive evaluation, we also simulated nodes with Byzantine faults for the test, which may keep silent with sending no message or send forged messages.

All experiments are conducted on the Ali Cloud platform, and the specific parameters are shown in Tab. 2. Twenty machines are used in the experiment, where each node is equipped with 1 CPU core with 2.5 GHz, 1 GB RAM, and 40 GB disk space. The network bandwidth is 200 Mbps. All machines run on centos7. The machines are distributed in 7 regions which are 7 delay circles. Tab. 3 shows the time required for transferring 4 MB blocks between and within delay circles. The content of the diagonal is the delay required for communication within the delay circle.

Table 2: The parameters in experiments parameter

Parameter	Value	Parameter	Value
#of machines	20	#of nodes n	20
faulty factor r	3∼5	Space per node	1 GB
Block size	4 MB	request distribution	Uniformed Distribution
#of delay circle	7		

Table 3: Latency between delay circles

Time/s	Qingdao	Beijing	Wulanchabu	Shanghai	Chengdu	Heyuan	Guangzhou
Qingdao	0.0379	0.1628	0.2201	0.2188	0.5680	0.4826	0.4922
Beijing		0.0506	0.1525	0.3302	0.6796	0.4984	0.4744
Wulanchabu			0.074	0.3730	0.6821	0.5681	0.5203
Shanghai				0.0544	0.4083	0.3154	0.3008
Chengdu					0.0440	0.4196	0.4012
Heyuan						0.0545	0.1094
Guangzhou							0.0258

4.1 Storage Consumption

We first test the storage overhead of the nodes. The number of nodes varies from 15 to 20. We assume that each block of the blockchain is 4 MB in size, that is, each chunk is 4 MB in size. As can be seen from Fig. 4, under different α, that is, under different consensus protocols, the storage overhead of CLEC and BFT-Store is almost the same. The storage redundancy of BFT-Store is lower than that of CLEC. Take $= 5$, $n = 20$ for example, in BFT-Store, the average storage overhead per block is 6 MB and storage redundancy is 1.5, while in CLEC, the average storage overhead per block is 7 MB and storage redundancy is 1.75. The storage redundancy of CLEC is about 1.17 times that of BFT-Store.

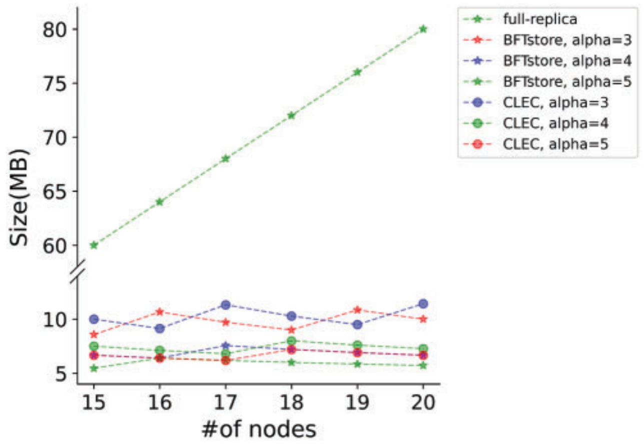
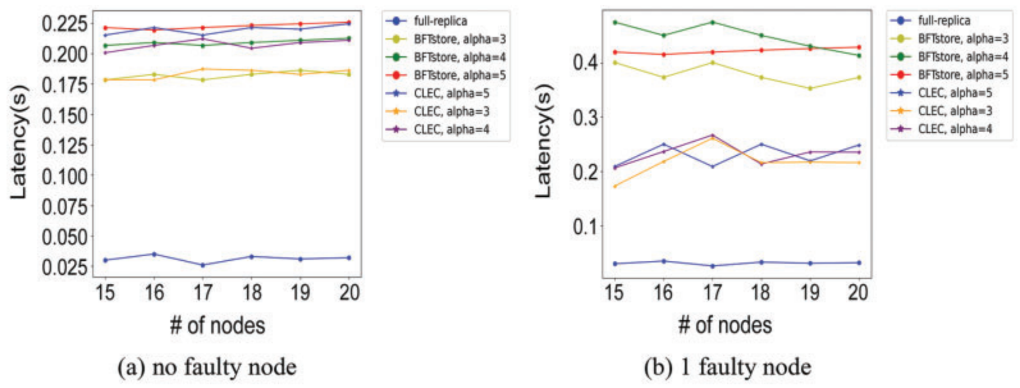


Figure 4: Storage overhead per block

4.2 Read Performance

Because the throughput is not significant in our experiment, we just study the latency of different storage schemes [7]. The latency of various storage schemes against the number of nodes from 15 to 20 is as shown in Fig. 5. The Uniform distribution is applied for requested blocks in Figs. 5a–5c. Figs. 5a–5c show the read latency under different faulty nodes respectively. They show that no matter how many faulty nodes exist, the full-replica blockchain has the lowest read latency, only if the number of faulty nodes is no more than f. Because users in the full-replica blockchain store the whole chain locally and can read the block locally, they can read any block immediately without a network. But it requires n times the storage overhead of CLEC, which is unacceptable in some cases. Fig. 5a reports the read latency of CLEC and BFT-store when there is no faulty node. From it, we know the read latency of these two schemes are almost the same, but they are both higher than that of full-replica, because each node only reserve one block in a coding epoch, and if the user needs to acquire other blocks, it ought to ask the other node for this certain block. Figs. 5b and 5c show that when there exist faulty nodes, the read latency of BFT-Store is about 1.7 times that of CLEC, because when reading the block stored in a faulty node, decoding to recover this block is needed, and the time cost of this process in CLEC is much lower than that of BFT-Store. Besides, with faulty nodes increasing, the probability of accessing a block on the faulty node increases, and thus the read latency of BFT-Store and CLEC both increases. In addition, when there are more than 2 faulty nodes, if more than 1 faulty node in a group, we can only collect k chunks as BFT-Store does to recover this chunk instead of using the combined chunk. Thus, with faulty nodes increasing, the gap of reading latency between CLEC and BFT-Store is getting seemly smaller.



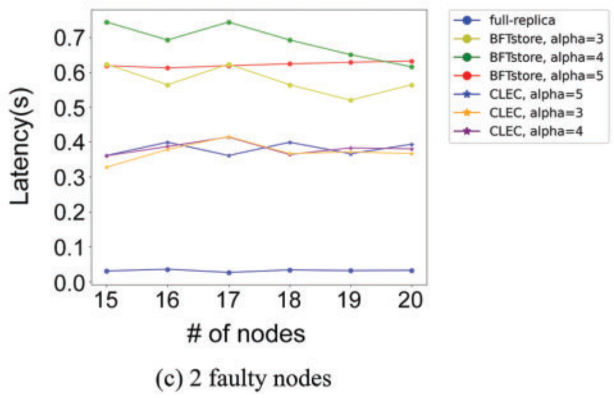


Figure 5: Latency against the number of nodes

4.3 Repair Performance

Taking 20 nodes, $\alpha = 5$ as an example, the maximum number of tolerable faulty nodes is 3. It is assumed that there are two local parities in CLEC, and all nodes are distributed in seven delay circles. Test the repair speed of different faulty nodes. As shown in Fig. 6, the average repair time of BFT-Store slightly increases with the increasing number of faulty nodes. However, the repair time of CLEC increases more sharply with the increase of the number of faulty nodes, because if nodes in the same group fail at the same time, global parities are needed to repair the chunk, which will prolong the repair time. The repair time of BFT-Store is about 4–6 times that of CLEC when a node fails, indicating that CLEC can significantly reduce the repair time after block failure.

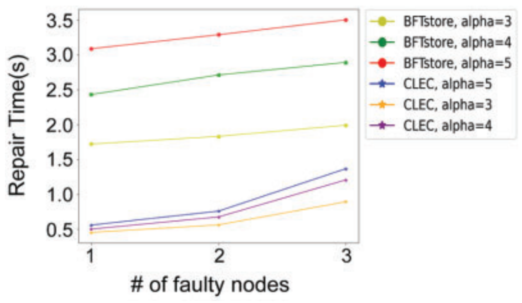


Figure 6: Repair performance

5 Conclusion and Future Work

Blockchain in domain naming resolution system will introduce a lot of storage overhead, erasure coding technology can effectively reduce the storage overhead, but also lower its read performance. CLEC proposed in this paper effectively reduces the repair penalty and thus improves the read efficiency. Our contributions include: (1) exploit Locally Repairable Codes in blockchain to improve its read performance and repair performance with slightly higher storage overhead. (2) considering the heterogeneity of node networks, we introduce a new notion of *delay circle*. The concept of time delay circle is combined with coding further improves the read performance. For the future work, we will focus on the change of coding scheme when nodes add or exit.

Funding Statement: This work is supported by The National Key Research and Development Program of China (2019YFB1804502).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] W. Wang, N. Hu, B. Liu, X. Liu and S. Li, "Research review of DNS security protection technology," *Journal of Software*, vol. 31, no. 7, pp. 2205–2220, 2020.
[2] M. Hu, B. Li and B. Yan, "Overview of domain name system security research," *Journal on Communications*, vol. 28, no. 9, pp. 91–103, 2007.
[3] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. of the Third Symp. on Operating Systems Design and Implementation (OSDI '99)*, Berkeley, CA, USA, pp. 173–186, 1999.
[4] T. Zheng, Y. Luo, T. Zhou and Z. Cai, "Towards differential access control and privacy-preserving for secure media data sharing in the cloud," *Computers & Security*, vol. 113, no. 1, pp. 1–15, 2022.
[5] T. Yang, C. Wang, T. Zhou, Z. Cai, K. Wu *et al.,* "Leveraging active decremental TTL measuring for flexible and efficient NAT identification," *Computers, Materials & Continua*, vol. 70, no. 3, pp. 5179–5198, 2022.
[6] T. Yang, B. Hou, Z. Cai, K. Wu, T. Zhou *et al.,* "6Graph: A graph-theoretic approach to address pattern mining for Internet-wide IPv6 scanning," *Computer Networks*, vol. 203, pp. 1–12, 2021.
[7] X. Qi, Z. Zhang, C. Jin and A. Zhou, "A reliable storage partition for permissioned blockchain," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 1, pp. 14–27, 2021.
[8] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert *et al.,* "A secure sharding protocol for open blockchains," in *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security, CCS'2016*, New York City, NY, USA, pp. 17–30, 2016.
[9] Y. Hu, L. Cheng, Q. Yao, P. P. C. Lee, W. Wang *et al.,* "Exploiting combined locality for wide-stripe erasure coding in distributed storage," in *Proc. of the 19th USENIX Conf. on File and Storage Technologies, FAST 2021*, pp. 232–248, 2021.
[10] M. Ali, J. Nelson, R. Shea and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. of the 2016 USENIX Conf. on Usenix Annual Technical Conf., FAST 2016*, Denver, CO, USA, pp. 181–194, 2016.
[11] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll *et al.,* "Research perspectives and challenges for bitcoin and cryptocurrencies," in *2015 IEEE Symp. on Security and Privacy, SP 2015*, San Jose, CA, USA, pp. 104–121, 2015.
[12] G. Zyskind, O. Nathan and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, San Jose, CA, USA, pp. 180–184, 2015.
[13] Q. Zheng, Y. Li, P. Chen and X. Dong, "An innovative IPFS-based storage model for blockchain," in *2018 IEEE/WIC/ACM Int. Conf. on Web Intelligence, WI 2018*, Santiago, Chile, pp. 704–708, 2018.

[14] H. Dang, T. Dinh, D. Loghin, E. Chang, Q. Lin *et al.,* "Towards scaling blockchain systems via sharding," in *Proc. of the 2019 Int. Conf. on Management of Data (SIGMOD '19)*, New York, NY, USA, Association for Computing Machinery, pp. 123–140, 2019.

[15] D. Perard, J. Lacan, Y. Bachy and J. Detchart, "Erasure code-based low storage blockchain node," in *2018 IEEE Int. Conf. on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Halifax, NS, Canada, pp. 1622–1627, 2018.

[16] P. Gopalan, C. Huang, H. Simitci and S. Yekhanin, "On the locality of codeword symbols," *IEEE Transactions on Information Theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[17] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder *et al.,* "Erasure coding in windows azure storage," in *Proc. of the 2012 USENIX Conf. on Annual Technical Conf. (USENIX ATC'12)*, Boston, MA, USA, pp. 1–12, 2012.

[18] K. S. Esmaili, L. Pamies-Juarez and A. Datta, "CORE: Cross-object redundancy for efficient data repair in storage systems," in *2013 IEEE Int. Conf. on Big Data*, Santa Clara, CA, USA, pp. 246–254, 2013.

[19] J. Hafner, "HoVer erasure codes for disk arrays," in *Int. Conf. on Dependable Systems and Networks (DSN'06)*, Philadelphia, PA, USA, pp. 217–226, 2006.

[20] M. Li, J. Shu and W. Zheng, "GRID codes: Strip-based erasure codes with high fault tolerance for storage systems," *ACM Transaction on Storage*, vol. 4, no. 15, pp. 1–22, 2009.

[21] J. Li and B. Li, "Beehive: Erasure codes for fixing multiple failures in distributed storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1257–1270, 2017.

[22] B. Sasidharan, G. K. Agarwal and P. V. Kumar, "Codes with hierarchical locality," in *2015 IEEE Int. Symp. on Information Theory (ISIT)*, Hong Kong, China, pp. 1257–1261, 2015.

[23] H. Hou, P. P. C. Lee, K. Shum and Y. Hu, "Rack-aware regenerating codes for data centers," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 4730–4745, 2019.

[24] Y. Hu, X. Li, M. Zhang, P. P. C. Lee, X. Zhang *et al.,* "Optimal repair layering for erasure-coded data centers: From theory to practice," *ACM Transactions on Storage*, vol. 13, no. 33, pp. 1–24, 2017.

[25] A. Indumathi and G. Sumathi, "Construction of key-dependent S-box for secure cloud storage," *Intelligent Automation & Soft Computing*, vol. 32, no. 3, pp. 1509–1524, 2022.

[26] N. Ragavan and C. Y. Rubavathi, "A novel big data storage reduction model for drill down search," *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 373–387, 2022.

[27] P. Wang and W. Susilo, "Data security storage model of the internet of things based on blockchain," *Computer Systems Science and Engineering*, vol. 36, no. 1, pp. 213–224, 2021.

[28] X. R. Zhang, W. F. Zhang, W. Sun, X. M. Sun and S. K. Jha, "A robust 3-D medical watermarking based on wavelet transform for data protection," *Computer Systems Science & Engineering*, vol. 41, no. 3, pp. 1043–1056, 2022.

[29] X. R. Zhang, X. Sun, X. M. Sun, W. Sun and S. K. Jha, "Robust reversible audio watermarking scheme for telemedicine and privacy protection," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3035–3050, 2022.

[30] W. Sun, G. Z. Dai, X. R. Zhang, X. Z. He and X. Chen, "TBE-Net: A three-branch embedding network with part-aware ability and feature complementary learning for vehicle re-identification," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2021, https://doi.org/10.1109/TITS.2021.3130403.

[31] W. Sun, L. Dai, X. R. Zhang, P. S. Chang and X. Z. He, "RSOD: Real-time small object detection algorithm in UAV-based traffic monitoring," *Applied Intelligence*, pp. 1–16, 2021, https://doi.org/10.1007/s10489-021-02893-3.