Erasure code-based low storage blockchain node

Hamidreza Khoshakhlagh

Supervisor: Vitaly Skachek

March 12, 2019

Abstract

Blockchain technology, first conceptualized by Satoshi Nakamato is an open distributed ledger that records transactions between different parties in a verifiable manner. As the popularity and public awareness of this technology and cryptocurrencies is growing fast, one of the main issues in this technology has been the possibility of a risk that this technology could not be able to keep up with growing demand. This issue which is called *Scalability* has been the focus of many research works in the last years.

In this report, we mainly focus on the storage scalability problem. We first try to survey existing solutions for this problem and then describe an approach, proposed by [PLBD18] in 2018, which uses erasure codes to improve the tradeoff between availability and storage.

1 Introduction

Blockchain technology may be considered as one of the most important innovations in financial services in the twenty-first century. This technology is significant for various reasons. By combining a distributed database and a decentralized ledger, this technology can completely remove the need for verification by a central authority (e.g., trusted third party).

An inherent property of decentralized ledgers is that each node has to store the entire data. Obviously, this would cause the following issue in popular blockchains, where they can have hundreds of GB storage: low-capacity nodes do not participate to the global storage effort and this makes the blockchain more centralized by only few nodes, so in contrary to the core idea behind the concept of a blockchain. Another issue similar to this is that despite the fact that the number of transactions is increasing exponentially as more people use the blockchain network, but the size of each block is not growing with the number of users. In a general view, this problem is called the scalability problem and is likely to be happened, not only in the case of storage, but also in the case of CPU and network load. For example in the case of CPU and process complexity,

each transaction has to be processed by every node which can consequently lead to a restriction on the throughput and latency of the blockchain.
Several solutions were proposed to overcome this problem. For example, considering process issue, a mechanism is proposed by Ethereum team which increase the blockchain throughput. Considering the storage scalability problem which is the main focus of this report, there are three types of solutions. While in a general case, each node stores the entire blockchain, these solutions try to decrease the storage space complexity but still guaranteeing the availability and reliability of the blockchain at the same time.
The structure of this report is as follows: We first start with some explanations about blockchain and storage scalability problem in this technology. Next, we explain two basic solutions for this problem. Then we bring the description of recent erasure code-based solution in more details. Finally, we will present a conclusion.

2 Blockchain and Storage Scalability problem (SSP)

The blockchain is essentially a chain of blocks. A block contains the data of all transactions within a specific period of time, and a reference to the previous block as well. Cryptographic hashing algorithms are used to guarantee that all blocks are not tampered with, and thus the blockchain keeps itself secure. To do this, the underlying hash function in the blockchain should be a secure and standard hash algorithm which has two important properties:

1 it should be collision resistant;

2 finding a preimage should be difficult (i.e., given x, it should be hard to find a different value x' such that $hash(x) = hash(x')$).

One important property in the blockchain technology is that it is not provided from a single server, but is run on a widespread network of computers as a distributed ledger. All participants in the network hold all data in the blockchain, and all work together to expand it. That's where the issue comes up: low-resource devices can not participate to the global storage effort and this makes the blockchain more centralized, which is in contrary with the core idea behind this technology. This problem which is called storage scalability problem, has been one of the main challenges in blockchain technology in recent years. Despite many solutions were proposed to solve the problem, none of them solves the problem fully.

3 Basic Solutions for SSP

Before describing two basic solutions for SSP, we recall storage policy in traditional blockchains, where as depicted in Fig. 1, each node stores the entire blockchain.

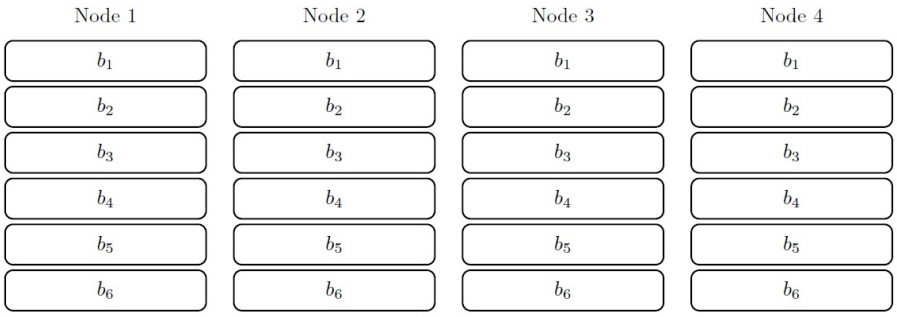

Figure 1: Traditional blockchain [PLBD18]

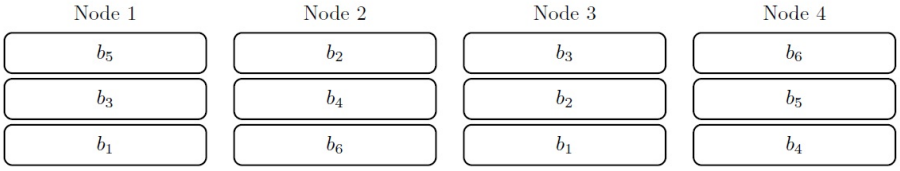

Figure 2: Distributed replication blockchain [PLBD18]

- **Replicated systems.** The first and most trivial solution for storage scalability problem is to reduce the required storing capacity of every node. This means that every block will be sustained by several nodes, but not by all. A toy example is presented in Fig. 2. By this approach, if one of the nodes storing a block becomes unavailable at a moment, the block will be still available in some other nodes. Despite a drastic improvement in space complexity, there is a serious weakness within this solution: block availability is reduced and thereby it is very tempting for malicious users to target those blocks of the blockchain that are poorly replicated.

- **Light nodes.** The second solution is offered by light clients. The idea behind defining light clients is to allow nodes with limited resources, in particular storage limitation, to participate to the blockchain and perform verification of transactions, while just storing block headers. The main disadvantage of this solution is that these nodes never verify the entire blockchain and rely their trust on other non-light nodes (i.e., full nodes) which is certainly in contrary with the core idea behind the blockchain technology, where the the main assumption is that no user in the network can be trusted.

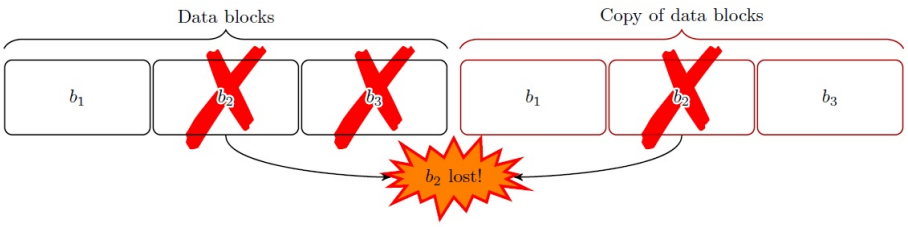

Figure 3: Traditional system [PLBD18]

4 Erasure code-based low storage node

To describe the third solution, we first consider a replicated system as depicted in Fig. 3. There are 3 data blocks b_1, b_2 and b_3 and they are simply replicated once for redundancy. If one block and it's copy are deleted at a moment (Here b_2), this means that the corresponding block can no longer be recovered by other nodes.
Here we give some explanations about the idea behind using erasure codes to improve the storage complexity: the fact that different blocks could be possibly unavailable at some moments means equivalently that the storage could be assumed to be over an erasure channel. In particular, from a coding-theory point of view, when there is a scenario where part of information over a communication channel can be deleted or unavailable, we model this scenario as an erasure channel. Therefore, it is reasonable to apply existing solutions for this problem from a coding theory approach to our storage scalability problem.
In [PLBD18], the authors propose that the redundancy blocks in the replicated system are replaced by a linear combination of data blocks. This solution, as represented in Fig. 4 would be more efficient and can provide better block availability at the same time. In this example, there are three available blocks, where the last two blocks are two linear equations correspond to the three original blocks b_1, b_2 and b_3. Now, as b_1 is known and we have only two unknown variables b_2 and b_3, so it is possible to recover these blocks by the two linear equations. The only needed computations are to invert the corresponding coefficient matrix and then multiply this matrix by the output vector as depicted in Fig. 4. We explain this solution and the description of encoding and decoding algorithms with more details in the following subsections. Before starting the encoding procedure, let us first define some notations:

- $N^{(i)}$: The ith node in the blockchain network. Note that i can be considered as a unique identifier for characterizing each node.
- $B^{(j)}$: The jth block in the blockchain. The first block is denoted by $B^{(0)}$.
- s_B is the maximum size of a block in the blockchain.
- k is an integer which corresponds to the number of fragments of a block.

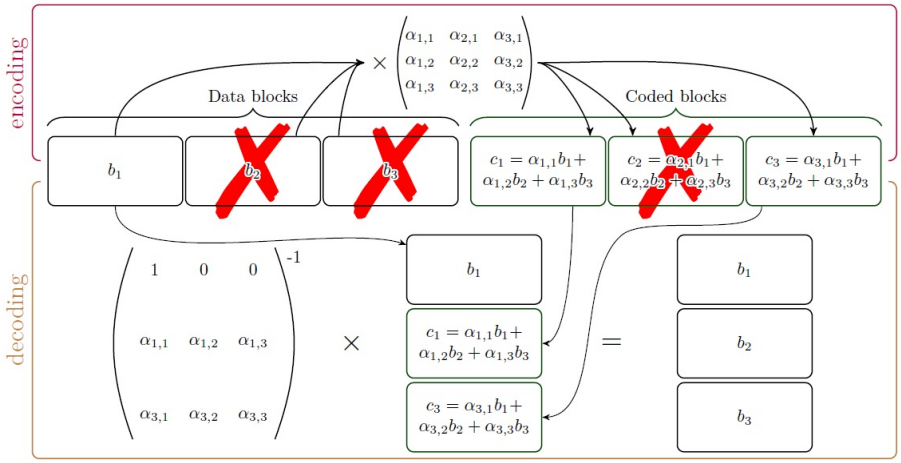

Figure 4: Coded storage system [PLBD18]

- r is an integer which corresponds to the number of coded fragments stored by a node.

4.1 Encoding algorithm

Assume that an arbitrary node (say $N^{(i)}$) wants to encode a block $B^{(j)}$. As depicted in Fig. 5, it first splits the block into k fragments, represented by $F_0^{(j)}, ..., F_{k-1}^{(j)}$. Next, it generates $k \cdot r$ entries of the coefficient matrix, denoted by $\{\alpha_0^{(i,j)}, ..., \alpha_{(k.r-1)}^{(i,j)}\}$, by initializing a pseudo-random number generator with a seed defined as the concatenation of binary expressions of indices i and j. Finally, the coded fragments are built by multiplying the coefficient matrix to the block fragments. i.e., the node performs the following linear computations for $0 \leq u \leq r-1$:

$$\zeta_u^{(i,j)} = \alpha_{k.u}^{(i,j)} \cdot F_0^{(j)} + ... + \alpha_{k.(u+1)-1}^{(i,j)} \cdot F_{k-1}^{(j)}$$

After doing this computations, the node removes the block $B^{(j)}$ and replaces it with the coded fragments $\zeta_u^{(i,j)}$ for $0 \leq u \leq r-1$.

4.2 Decoding algorithm

When a node $N^{(i)}$ wants to recover a block $B^{(j)}$, it firsts download coded fragments $\zeta_0^{(i_0,j)}, ..., \zeta_{r-1}^{(i_{r-1},j)}$ from different nodes $N^{(i_0)}, ..., N^{(i_{r-1})}$. It is important to note that the node requests can be from different nodes or only from a small set of nodes. In particular, there is no restriction on node selections as by each request, the node will retrieve a linear equation corresponding to the desired block fragments. Next, due to the fact that this node knows the indices

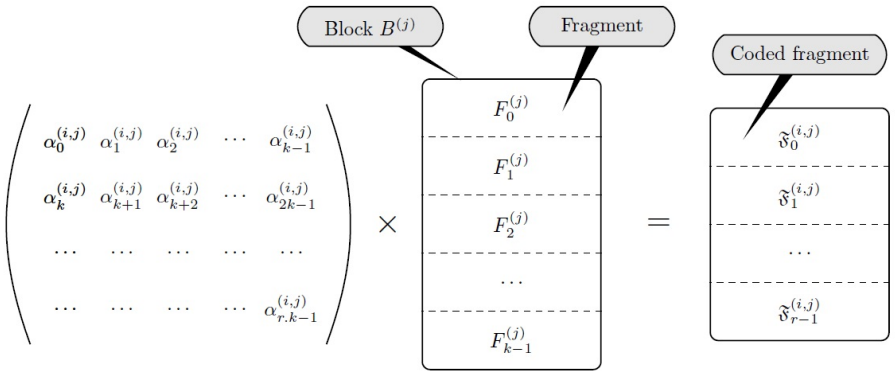

Figure 5: Block encoding procedure [PLBD18]

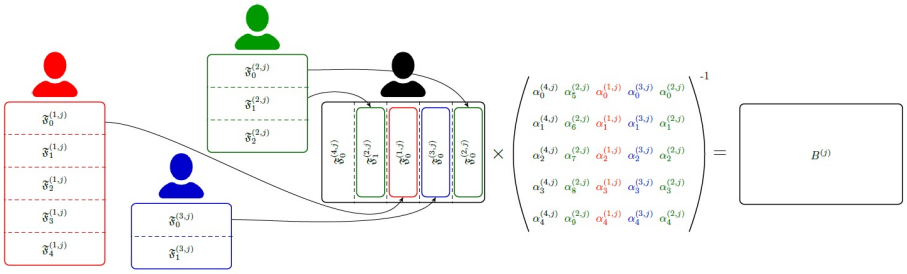

Figure 6: Block recovery procedure [PLBD18]

$i_0, ..., i_{r-1}$ and j, so it is able to initialize the pseudo random number generator by the concatenation of binary expression of these indices, and then computes the entries of the coefficient matrix. Now, node $N^{(i)}$ knows r linear equations that correspond to r coded fragments. Thus after downloading a sufficient number of these fragments, it will be able to recover the k fragments and thus the entire value of block $B^{(j)}$ by performing linear algebra computations. A toy example of decoding procedure is illustrated in Fig. 6.

4.3 Choosing parameters

To find a good tradeoff between storage and time complexity, one needs to determine the parameters k and r. Depending on the blockchain and type of the node, different parameters could offer the best tradeoff and this fact makes it impossible to determine a generic value for k and r. But based on some observations and system constraints, it is possible to analyze these parameters. In what follows, we first analyze the size of the finite field, where the entries of the coefficient matrix are chosen from. Next, we summarize the complexity results with respect to encoding and decoding complexity from [PLBD18]. Some

important observations from [PLBD18] on choosing k and r parameters are summarized at the end of the discussion.

a. **Size of the finite field.** The importance of the size of finite field is as follows: first, it affects the error probability in recovering a block from the downloaded coded fragments, which is minimized by choosing a larger finite field. Also, there is a direct relation between the size of the finite field and the encoding and decoding complexity. i.e., by choosing a larger field, we have an encoding and decoding algorithm with a higher computation complexity.

   The authors in [PLBD18] propose to use the finite field $\mathbb{F}_{256}$. To choose this field, the authors use a lemma in [SB10] which shows that if the entries of the coefficient matrix in the underlying linear equations are selected from the field $\mathbb{F}_{2^m}$, then the probability of retrieving a block from $k+s$ downloaded fragments can be approximated by $1-2^{-m\cdot(s+1)}$.

b. **Processing Complexity.** To find the complexity of encoding, we observe that it consists in one matrix multiplication with $r \times s_B$ operations. This means that the encoding complexity is independent of parameter k, only depending on r. Considering the decoding, the complexity consists in one matrix inversion and one matrix-vector multiplication. Since the inversion of coefficient matrix has complexity $O(k^3)$, the total decoding complexity would be $O(k^3) + k \times s_B$. Fig. 7 represents the speed of encoding and decoding for 1MB blocks and for different values of k and r. For this computations, an erasure code implementation on a core i7-6700 GHz CPU is used. Based on this graph, it can be concluded that the process cost is negligible, as the coding speed is always larger that 15 Gbps for $r \leq 20$.

c. **Choosing k and r.** In the case of k, it can be shown that by choosing larger values of k, the decoding procedure becomes more efficient. Moreover, the compression factor which is defined as $c = k/r$ will also increase, which is desirable from the efficiency point of view. Therefore, it can be concluded that k should be chosen as large as possible, yet not exceeding $2^m/2 = 128$.

   The choice of r on the other hand is up to the end user and depends on the type of the user. Based on [PLBD18], choosing a large value of r will improve the block recovery, but reduce the network load. Moreover, increasing r will increase the storage effort of a node and thereby improve the overall availability of the blockchain. It can be concluded that the parameter r must be chosen according to the properties of the node.

5 Conclusion

In this report, we first present different existing solutions for Storage Scalability Problem (SSP), which was introduced in Section 1. Next, we explain

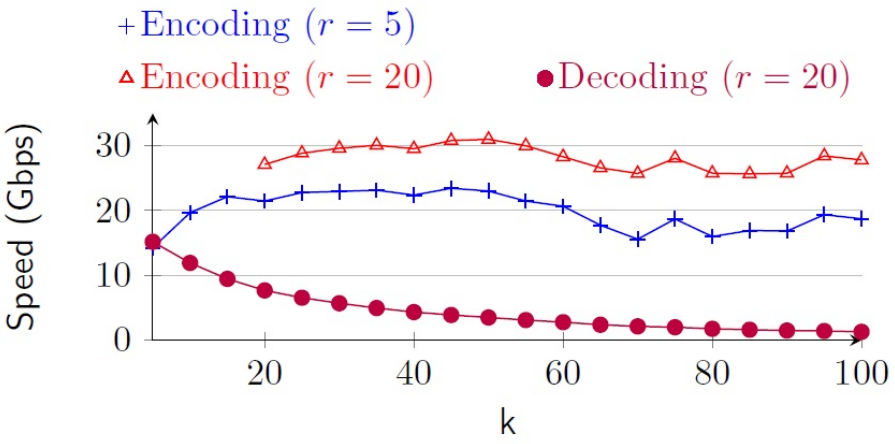

Figure 7: Encoding and Decoding speed [PLBD18]

the solution given by [PLBD18]. The idea behind this solution is to introduce new nodes called light storage and then letting these nodes to store only some *coded fragments* of each block. When a light storage node wants to retrieve a block from these coded fragments, it first downloads a sufficient number of coded fragments corresponding to the desired block and then recover the block by solving a system of linear equations. In this case, even though blocks are no longer stored in their original form, they are still recoverable by downloading sufficient related information and performing some algebraic computations. The authors in [PLBD18] show that this solution improves the overall availability of the blockchain, without affecting the data integrity.

References

[PLBD18] Doriane Perard, Jérôme Lacan, Yann Bachy, and Jonathan Detchart. Erasure code-based low storage blockchain node. *CoRR*, abs/1805.00860, 2018.

[SB10] Chris Studholme and Ian F. Blake. Random matrices and codes for the erasure channel. *Algorithmica*, 56(4):605–620, 2010.