Abstract

Data availability problem is one key challenge of future blockchains with large-scale applications. In this paper, we propose a dynamic data sharding to address the data availability problem, especially its sub-problems such as proof of publication, proof of storage, and proof of retrievability. We apply the dynamic data sharding to a decentralized KV store on top of an EVM-compatible blockchain, and our early analysis suggests that the cost of storing large values can be reduced to ≈ 1/100x compared to the fully-replicated EVM-native KV store via SSTORE opcode while ensuring tens or hundreds of replicas of the values in the network.

Introduction

One of the key challenges of future blockchains is the data availability for a large amount of data that far exceeds the capacity of a single node. The data availability problem can be generally divided into the following sub-problems:

- Proof of Publication

, which ensures that the data show up on the network initially and the nodes can choose to process or discard the data;

- Proof of Storage

, which ensures that the data is stored somewhere in the network and prevents losing the data;

- Proof of Retrievability

, which ensures that anyone is able to retrieve the data in the presence of some malicious nodes that withhold the data.

In this paper, we propose a dynamic data sharding to achieve data availability with applications for a large decentralized key-value (KV) store on top of an EVM-compatible blockchain. The KV store maintains two data structures

- Metadata of the KV store

, which is maintained in the KV contract and is fully replicated to all the nodes in the network;

- Values of the KV store

, which are partitioned into multiple fixed-size shards. Each node may serve zero or multiple shards and claim the rewards of each shard via proof of replication. When the number of KV entries increases, the shards will be dynamically created to serve new values of KV entries.

The key to achieving data availability is to build an on-chain oracle of the estimate of the replication factors of each shard and then to reward those nodes that prove the replications of the shard over time. When a node is launched, the node operator is free to choose the shards (or no shard) to serve, most likely depending on the shard that will offer the best-expected reward, i.e., lower replication factors based on the oracle. The reward of each shard is distributed to the nodes that can prove the replication of the shard via proof of random access (PoRA).

The PoRA is a mining process that heavily relies on read IOs that are performed over the shard data. Similar to the proof of work mining, the KV store contract maintains a dynamic difficulty parameter for each shard and adjusts the parameter after accepting the submission result of the PoRA (a.k.a., a mini-block). Therefore, we could estimate the replication factor of the shard by calculating the hash rate, i.e., read IO rate, vs the read IO rate of most mining-economical storage devices (e.g., 1TB NVME SSD).

When some nodes of the shard leave the network, the hash rate of the shard will decrease, and therefore new nodes are incentivized to join the shard in order to receive better rewards. This dynamical procedure guarantees the replication factor over time and thus preventing loss of the data, i.e., achieving proof of storage.

Further, we could adjust the token reward (paid from users) to guarantee some levels of the replication factor. Given the assumption that most of the nodes that run the shard replicas are honest (in fact, 1-of-N is sufficient), then we could address the data withholding attack from malicious nodes and thus achieve proof of retrievability.

Our early economic analysis shows that with the dynamic data sharding, the cost of storing KVs with large values can be decreased to \approx

1/100x of the current fully-replicated storage model in EVM (via SSTORE opcode) while ensuring tens or hundreds of replicas based on the IO performance of the targeted storage device.

The rest of the paper are organized as follows. Section 2 describes the semantics of the proposed KV store. Section 3.1 explains the data structures of the KV store. Section 4 illustrates how to achieve data availability. Section 5 discusses some limitations and optimizations of the KV store. Section 6 lists the attack vectors, and Section 7 concludes the paper.

(See the attached for the complete paper)

dynamic_data_sharding v0.1.4.pdf (257.5 KB)

Acknowledgment

Thank Micah for providing early feedback on the paper.