Canyon: Permanent Storage Layer for Limitless Scalability

Liu-Cheng Xu

Canyon Labs

Abstract

Canyon is a permanent storage network built on Substrate, which records the hashes of files on-chain and stores the files off-chain. By blending PoS and a probabilistic proof-of-storage scheme inspired by Arweave, Canyon greatly reduces the barriers to entry for miners, who are incentivized to store as much data as possible for the rewards.

Contents

1 Introduction **3**

1.1 Motivation . . . 3

2 Background **3**

2.1 Filecoin . . . 3

2.2 Crust . . . 3

2.3 Arweave . . . 3

3 Design Principles **4**

3.1 Security and Privacy . . . 4

4 System Design **4**

4.1 Consensus . . . 4

4.1.1 Proof of Access . . . 4

4.1.2 Proof of Stake . . . 5

4.2 Economy Model . . . 6

4.2.1 Bandwidth . . . 6

4.2.2 Transaction Fee . . . 6

4.2.3 Data Oblivion . . . 6

4.2.4 Payments . . . 6

5 Conclusion **6**

A Appendix **7**

A.1 Data Structure 7
A.1.1 poa 7

1 Introduction

1.1 Motivation

It can't be denied that the storage infrastrusture will be an essential part of Web 3.0 in the future.

2 Background

2.1 Filecoin

Filecoin[1] proposes a sophisticated cryptographic solution based on zero-knowledge proofs (ZKPs) to prevent the common attacks on the decentralized storage verification, which uses pure mathematical methods and is able to achieve high-security guarantees. The whole mining process consumes too much computing power, rendering the actual storage cost prohibitively expensive. Furthermore, the high hardware requirements largely raise the threshold for small miners, preventing those with pure common commodity hardware from joining the network, the storage distribution today becomes centralized.

Due to the lack of authentic storage needs and system design of Filecoin, the miners themselves are economically impelled to store tons of garbage data in the network, increasing the storage power and maximizing their mining profit. Despite the Filecoin team has proposed the off-chain governance approach like Filecoin Plus[1], it does not mitigate this problem substantially, leaving the promotion of the useful storage still a huge unresolved challenge in the Filecoin network.

2.2 Crust

To solve the problem of decentralized storage verification, Crust[2] adopts a hardware-based solution Trusted Environment Execution (TEE) to make sure the miners store a specific number of data copies as promised. Each of the storage nodes is required to enroll on Crust chain through TEE before it's allowed to deal with the storage orders from client. The TEE module of the nodes will periodically check and report whether the files are properly stored in the local storage space in a trusted way.

There are three major TEE providers with different implementations: Software Guard Extensions (SGX) on the Intel platform, Secure Encrypted Virtualization (SEV) on the AMD platform, and TrustZone on the ARM platform. The SGX of Intel is the most widely used TEE platform. Although Crust has a relatively low hardware demand for mining compared with Filecoin, the miners are heavily dependent on a fairly narrow range of hardware that supports TEE, thus being greatly affected by the hardware manufacturers.

2.3 Arweave

Unlike the ephemeral storage services such as Filecoin, Crust, Arweave[3] serves as a permanent storage layer using a probabilistic and incentive-driven approach to maximize the number of redundant copies of any individual piece of data in the network, which fills in a crucial aspect of decentralized storage need in Web 3.0. The natural feature of perpetual storage is perfectly suitable for the NFTs, and is the cornerstone of new storage-based computation paradigm like everFinance [2].

Without the periodical audit of the data replicas, Arweave is much effortless than the other solutions in terms of the constraint and cost of storage consensus. Nevertheless, the deficiency

[1] `https://docs.filecoin.io/store/filecoin-plus`

[2] `https://medium.com/everfinance/a-storage-based-computation-paradigm-enabled-by-arweave-de799ae8c424`

of this scheme is the potential risk of data centralization that ultimately there possibly will be a single storage provider serves the whole data. The Arweave team has introduced and deplopyed an improved version of consensus Succinct Proofs of Random Access, attempting to disincentivize miners from retrieving data on demand from the network.

3 Design Principles

3.1 Security and Privacy

4 System Design

4.1 Consensus

Canyon network is profoundly inspired by Arweave, especially the storage consensus, aiming to be a permanant decentralized storage network using the Substrate framework. The key contribution of Canyon is that it adapts PoS into the storage consensus of Arweave, whereas Arweave uses PoW, making Canyon a more scalable and environment-friendly network.

In order to mine a block, a legitimate POA, which proves the block author has the access to the data of a random historical block, is required to be included in the block header. According to the computed result of POA, we can estimate the proportion of data stored locally by a node in the network-wide data. Based on this, we link the PoS rewards to the estimated storage capacity of a node. The more data a node stores, the more rewards it can receive. Besides, as the number of blocks mined by the node continues to grow, the estimation of the node's storage capacity will be getting closer to the actual value.

4.1.1 Proof of Access

In Arweave, the probability that a node can mine a block is a function of the node's hashing power relative to the average hashing power of all of the nodes in the network that also possess the recall block.

$$P(\text{win}) = P(\text{has recall block}) * P(\text{finds hash first}) \tag{1}$$

In Canyon,

$$P(\text{win}) = P(\text{has recall block}) * P(\text{claims slot}) \tag{2}$$

The steps for generating the storage proof are as follows:

1. Input the value of `parent_hash` and `weave_size` respectively. And depth is initialized to 1.
2. Compute `recall_byte`, i.e. a random byte in the network-wide data history ranging from 0 to `weave_size`, excluding `weave_size`.
3. Find out the `recall_tx` the contains the `recall_byte` computed from the previous step.

Algorithm 1: Generation of POA

Input :
The random seed S;
The weave size W;

Output:
The proof of accessing the recall block POA;

Initialize the number of repeats x with 1;
repeat
  Draw a random byte B with $\text{MULTIHASH}(S, x) \mod W$;
  Find the TX in which the random byte B is included;
  $x \leftarrow x + 1$;
until *The data of TX is available*;
$POA \leftarrow \text{CONSTRUCTPOA}(TX)$;
return POA;

4. If the miner already has stored the data of `recall_tx` locally, extract the original data of `recall_tx` and split the data into a list of chunks, then construct a storage proof `poa { tx_path, chunk_root, chunk_path, chunk }` (See A.1.1 poa Data Structure).
5. If the miner does not have the data of `recall_tx` locally, increase the value of `depth` by 1 and repeat the step 2 to 4.
6. Return **poa**.

As you can see, if a node stores 100% of data, the value of `depth` always remains 1. The node only needs to go through the above steps once every time when it attempts to generate a `poa` proof. If the node stores 50% of data, the value of `depth` is expected to approach 2 as it produces enough blocks. If the node stores 10% of data, the value of `depth` approaches 10.

If a node produces a total of N blocks, and the poa.depth of each block is $depth_{i \in 1,2,\ldots,N}$, the average `depth` value of the N blocks $\hat{x}$ is:

$$\hat{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

With time, the node produces enough blocks (i.e. $N \to \infty$), we can calculate the proportion of data stored locally by the node relative to the network-wide data more precisely with the value of $\hat{x}_{N \to \infty}$. The formula is as follows:

$$R = \frac{1}{\hat{x}_{N \to \infty}}$$

With the knowledge of the storage capacity of nodes R, we can incentivize the miners to store more users' data by allocating the rewards accordingly.

4.1.2 Proof of Stake

4.1.2.1 Validator Election

The validators on Canyon network have two duties: author blocks as a virtual miner and provide unstoppable storage services. The security of a PoS system is rooted in the number of tokens locked in the staking pool.

WIP

4.1.2.2 Staking Rewards

The reward for a miner producing a block is composed of three parts:

$$R_{total} = R_{inflation} + R_{fees} + R_{endowment}$$

4.2 Economy Model

4.2.1 Bandwidth

For any storage platform to be valuable, it must be careful not to lose the data it was given, even in the presence of a variety of possible failures within the system. What's more, a storage platform is of no use unless it also functions as a retrieval platform. Only when the data can be retrieved without any pains whenever the client wants to can we hope a fully functional storage service.

4.2.2 Transaction Fee

The fee of data storage transaction is composed of the perpetual storage cost and one-shot bandwidth cost. The perpetual storage cost is basically modeled after Arweave, on the observed pattern that the cost of commercially available storage media has been decreasing at a significant rate over the last decades and this trend is foreseed to last for hundreds of years.

4.2.3 Data Oblivion

Specifically, the perpetual storage needs can be classified into two kinds: immortal and indefinite storage. The immortal storage means the file that is doomed to be preserved from the creation, such as the metadata of NFTs. Indefinite storage is such a kind of data that has an uncertainty of ending time, but can be deleted once it's used or the client simply does not wish the contract to be continued for any reason.

For these use cases, Canyon allows the origin uploader opts to terminate the contract and partially refund the charged perpetual storage fee. The files that have been refunded will be disqualified to be selected for the future storage consensus procedure, thus the miners that keep storing them will receive no rewards. For the sake of scarce storage resources, the miners are expected to purge these data from local storage, consequently, sooner or later, the data will be eliminated from the network.

4.2.4 Payments

An inherent advantage of Canyon is the easier interoperability with the Polkadot[4] ecosystem by using Substrate[5], which is the same framework Polkadot is built on. Apart from the native token of Canyon CYN, Canyon will support the users pay the storage fee by introducing the stable coins like USDT from Polkadot network, reducing the impact of the churn of token price on providing a stable storage service for the custom.

5 Conclusion

Canyon network is designed to be a permanent decentralized storage network, putting the emphasis on both light-weight storage consensus and highly-avaliable data retrieval. Being a permanent storage layer, Canyon is better suited for the storage-based computation paradigm.

References

[1] Protocol Labs, Filecoin: A Decentralized Storage Network, `https://filecoin.io/filecoin.pdf`.

[2] `https://crust.network/`

[3] Sam Williams, Viktor Diordiiev, Lev Berman, India Raybould, Ivan Uemlianin. Arweave: A Protocol for Economically Sustainable Information Permanence.

[4] Gavid wood. Polkadot: Vision for A Heterogeneous Multi-chain Framework. `https://polkadot.network/PolkaDotPaper.pdf`

[5] Substrate: The platform for blockchain innovators. `https://github.com/paritytech/substrate`

A Appendix

A.1 Data Structure

A.1.1 poa

Each storage proof is composed of the following fields:

- *tx_path*: Merkle path of recall transaction to the transaction root, which is included in the block header.
- *chunk_root*: Merkle root of transaction data chunks.
- *chunk_path*: Merkle path of *chunk* to *chunk_root*.
- *chunk*: Raw bytes of recall chunk, at most 256 KiB.

There are two merkle proofs generated per poa: *tx_path* and *chunk_path*. With the origin *chunk*, anyone can verify that the block author did have access to the data of recall transaction without downloading the entire data of that block. To be specific, everyone can verify the storage proof by merely downloading the block header, for the full content of poa is included in the header as a digest item.

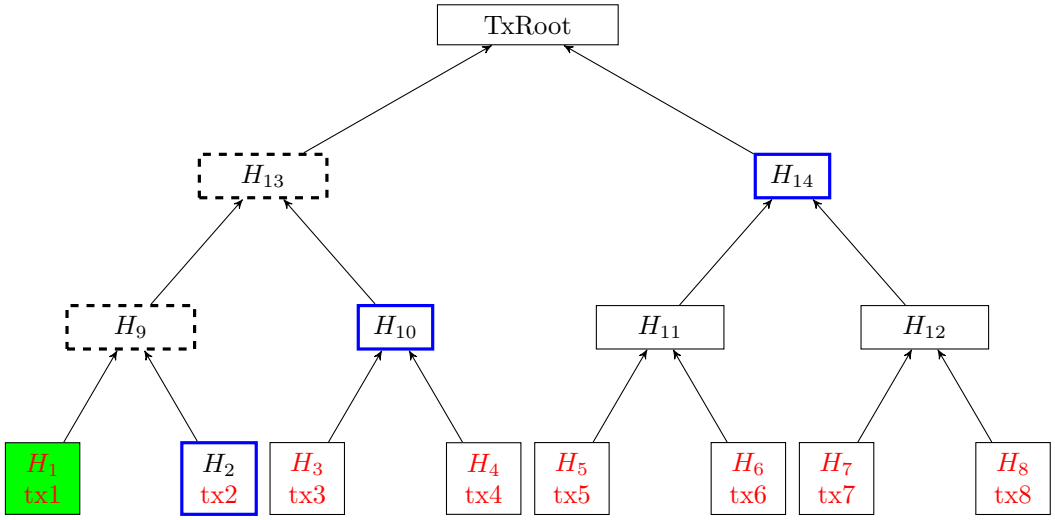

Figure 1: Tx Path

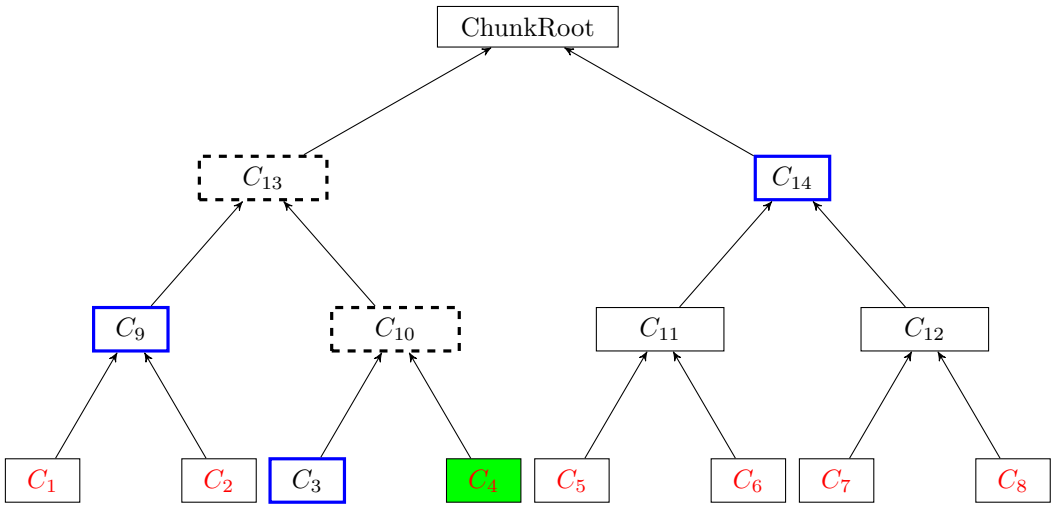

Figure 2: Chunk Path