Canyon: A Permanent Storage Layer for Web3.0

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 actual files off-chain. By blending PoS and a probabilistic proof-of-storage scheme inspired by Arweave, Canyon is able to pledge pretty high data durability in theory and greatly reduces the barriers to entry for storage miners who are incentivized to store as much data as possible for winning more rewards, realizing the permanent storage in a trustless way with a low cost, rendering Canyon Network a highly sustainable decentralized storage platform.

Contents

1 Introduction **3**

1.1 Motivation 3

2 Background **3**

2.1 Filecoin 3

2.2 Crust 4

2.3 Arweave 4

3 System Design **4**

3.1 Consensus 4

3.1.1 Proof of Access 5

3.1.2 Proof of Stake 6

3.2 Economy Model 8

3.2.1 Bandwidth 8

3.2.2 Transaction Fee 9

3.2.3 Data Oblivion 9

3.2.4 Payments 9

4 Conclusion **9**

A Appendix **10**

A.1 SPoRA 10

A.2 Data Structure 11

A.2.1 poa 11

1 Introduction

1.1 Motivation

Bitcoin[6] brought us a global-scale consensus mechanism and a fully decentralized electronic cash system. From there, Ethereum[9] proposed the goal of a world computer that will never stop by introducing a Turing-Complete smart contract layer. Thus, the vision of Web3.0 has once again germinated in people's minds.

Web1.0, also called the static web, was the first and most reliable internet in the 1990s despite only offering access to limited information with little user interactions. Web2.0, or the social web, was largely driven by the innovations in mobile, social, and cloud, rendering the internet a lot more interactive.

Web3.0 has been described as early as 2006 by Professor Tim Berners-Lee, the inventor of the World Wide Web, when he proposed the concept of the Semantic Web. Since then, Web3.0 has gradually emerged as a movement away from the centralization of services as well as the monopoly of big data by tech giants. Later, in 2014, Dr. Gavin Wood, co-founder of Ethereum and creator of Polkadot, described his views on four components of Web3.0[7] in his blog: static content publication, dynamic messages, trustless transactions and an integrated user-interface.

The future is yet to come. We believe the vision of Web3.0 is to make the Internet more decentralized, verifiable and trustless. There is no doubt that the infrastructure, providing a secure, highly available, low-cost, and easy-to-use decentralized data access service, will be an essential part of Web3.0 applications.

2 Background

2.1 Filecoin

Filecoin[1] proposes a sophisticated cryptographic solution based on zero-knowledge proofs (ZKPs) to prevent the common attacks on 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.

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

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 the 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 demand 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 mandatary periodical audit of the data replicas, Arweave is much effortless than the other solutions in terms of the mining equipment as well as the final total cost of realizing the data storage. Nevertheless, there are some deficiencies about this super lightweight storage scheme: firstly there is no data durability guarantee due to PoA itself is purely based on the probability without any bound and so PoW is, which means the data loss does have a chance to happen even Arweave paper claims that the PoA algorithm incentives miners to store 'rare' blocks more than it incentivizes them to store well-replicated blocks. Furthermore, there is a potential risk of data centralization that ultimately there possibly will be a single storage provider that serves the whole data, which has been already observed on the Arweave network. The Arweave team had introduced and deployed an improved version of consensus Succinct Proofs of Random Access(SPoRA), attempting to discourage miners from retrieving data on demand from the network. SPoRA can mitigate the pool issue in some extent but is unable to threoretically fix it completely.

3 System Design

3.1 Consensus

Canyon network is profoundly inspired by Arweave, especially the storage consensus, aiming to be a permanent 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

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

PoW, making Canyon a more scalable and environment-friendly network. Additionally, high data durability, comparable to the one provided by traditional centralized cloud storge providers, can be achieved due to the validator set is known for a PoS system and the minimal storage ratio can be applied for each validator. This will be expanded in 3.1.2.2.

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.

3.1.1 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, a PoS-based system, the node is allowed to produce a block only when it succeeds in claiming the slot and owns the data of recall block:

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

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 $\textsc{MultiHash}(S, x) \mod W$;
    Find the TX in which the random byte B is included;
    $x \leftarrow x + 1$;
until *The data of TX is available*;
$POA \leftarrow \textsc{ConstructPOA}(TX)$;
return POA;

The steps for generating the storage proof are as follows:

1. Input the value of `parent_hash` (S) and `weave_size` (W) respectively. And the repeats x, hereinafter called `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.
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 { depth, tx_path, chunk_root, chunk_path, chunk }` (See A.2.1 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 $x_{i \in 1,2,\ldots,N}$, 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.

3.1.2 Proof of Stake

3.1.2.1 Validator Election

The validators on Canyon network have two responsibilities: author blocks as a virtual miner and provide unstoppable storage services. Given the security of a PoS system is rooted in the number of tokens locked in the staking pool, both the validators and stakers should be obviously rewarded for the contribution to the security of consensus. Additionally, the value of Canyon network in essence is the high availability in the perpetual storage service served by the whole validator set. The validators who win more stakes and provide superior storage services with larger storage capacity will gain a great advantage in the staking election.

There is a minimal storage ratio constraint for each validator. Once the calculated storage ratio on chain of a validator is lower than its claimed minimum value, the reward for the particular validator will be deducted and even be slashed if it's extremely lower for not contributing enough storage resources as promised. The storage ratio of a validator will be converted into a virtual stake and considered in the validator election process. The minimum storage ratio will be adjusted according to the number of total validators and the size of entire weave. At the early stage of network, the weave is so small that all the validators can afford the storage cost for storing the entire weave locally, hence the minimum storage ratio will be pretty high. With the weave becoming larger and reaching the EiB level, the minimum storage ratio of each validator will be decreased to less than 1% in the case of Filecoin data distribution.

3.1.2.2 Data Durability

From the view of the traditional cloud storage providers, the data durability means the ability to keep the stored data consistent, intact without the influence of bit rot, drive failures, or any form of corruption, expressed as an annual percentage in nines, as in two nines before the decimal point and as many nines as warranted after the decimal point. For example, eleven nines of durability is expressed as 99.999999999%. What this means is that the storage vendor is promising that your data will remain intact while it is under their care without losing any more than 0.000000001 percent of your data in a year (in the case of eleven nines annual durability)[8].

Of the major vendors, Azure claims 12 nines and even 16 nines durability for some services, while Amazon S3, Google Cloud Platform, and others claim 11 nines or 99.999999999% annual durability.

As a permanent storage network, the minimal storage ratio of each validator inherently guarantees high data durability. Assuming the minimal storage ratio for each validator is x, the number of total validators in the network is y, the probability that some file has been uploaded but somehow actually not stored by any validator due to various reasons, in another word, the file is, unfortunately, missing now, is $P_{missing}$:

$$P_{missing} = (1 - x)^y$$

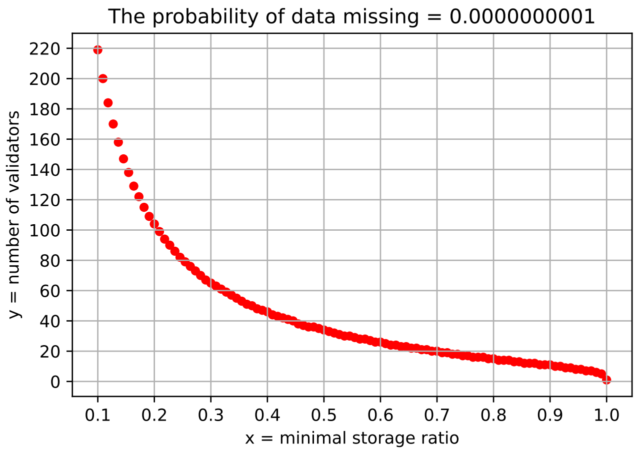

Figure 1: 12 nines of data durability($P_{missing} = 0.0000000001$)

3.1.2.3 Data Redundancy

Redundancy means the act of duplicating the data. Although the high data durability ensures the data loss happens in a pretty low chance, there is no guarantee about the number of data copies that actually exist in the network, for PoA is an incentive-driven approach and does not specify an exact number of redundant copies that should be provided. To put it simply, the system can only promise there is at least one copy for a given piece of data in the network and nothing else. The data redundancy is not a trivial issue, Canyon does not resolve it using a hard way but using a market-leading means with a principle that clients should pay for reading data. A client has to pay some fee for reading data due to the bandwidth cost, hence the validators are motivated to store the data that are read frequently because they can earn some money by providing the read service except the basic storage service. In another word, the data with a frequent read need will obviously have more copies, the data that is rarely read will have fewer copies, but no data will be lost even it is never read by anyone after stored in the network.

3.1.2.4 Staking Rewards

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

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

3.2 Economy Model

3.2.1 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 is able to be retrieved without any pains whenever the client wants to can we hope a fully functional storage service.

3.2.2 Transaction Fee

The fee of data storage transaction in Canyon 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 foreseen to last for hundreds of years.

3.2.3 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 to opt 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.

3.2.4 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 using stable coins like USDT which are bridged from Polkadot network, reducing the impact of the volatility of token price on providing a stable storage service for the customer.

4 Conclusion

Canyon network is designed to be a permanent decentralized storage network, putting the emphasis on both lightweight storage consensus and highly available data retrieval. Being a permanent storage layer, Canyon is better suited for the storage-based computation paradigm. Canyon guarantees high data durability by enforcing an adaptive minimal storage ratio constraint for each validator based on the total number of validators. The solution for the data redundancy is a retrieval-driven approach relying on the retrieval market itself.

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`

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://www.bitcoin.org/bitcoin.pdf`, 2008.

[7] Gavin Wood. "ÐApps: What Web 3.0 Looks Like". `https://gavwood.com/dappsweb3.html`, 2014.

[8] `https://www.backblaze.com/blog/cloud-storage-durability-vs-availability/`

[9] Vitalik Buterin. A next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2014.

A Appendix

A.1 SPoRA

Succinct Proofs of Random Access(SPoRA) is a new type of consensus deployed by Arweave with an intention of resolving the observed storage and computation pool issue with the PoA consensus [3].

SPoRA requires the miners to compute a slow RandomX(S, x) instead of MultiHash(S, x) used in 1 Algo 1, reducing the number of trials allowed for finding the recall block in the context of the average block time of a PoW chain. In another word, an Arweave miner won't gain much more mining advantages with more computing powers. Canyon might not adopt this change as it's a PoS system.

In addition, the `mod` operation for locating the random recall byte in 1 Algo 1 is replaced by searching in a limited search space. The entire weave is divided into N equal parts, the random recall byte locating will be constrained to be performed in only one part of them W_{search}.

[3] `arweave.medium.com`

Algorithm 2: Generation of SPoRA

Input :
The count of search subspaces N;
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
    $H0 \leftarrow \textsc{Random}\text{X}(S, x)$;
    $W_{search} \leftarrow H0 \mod N$;
    Draw a random byte B from the limited search subspace W_{search};
    Find the TX in which the random byte B is included;
    $x \leftarrow x + 1$;
until *The data of TX is available*;
$POA \leftarrow \textsc{Construct}\text{POA}(TX)$;
return POA;

A.2 Data Structure

A.2.1 poa

Each storage proof is composed of the following fields:

- *depth*: the number of total attempts to construct a valid poa for producing block.
- *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.

Technically, the way to prove the miner did store the data of recall block, is that it can be verifiably proved that the miner holds one piece of data belonging to that block, which requires two Merkle proofs to be generated: *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. Furthermore, 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.

Assuming the recall byte is located in the chunk C_4 which is part of the data of $tx1$.

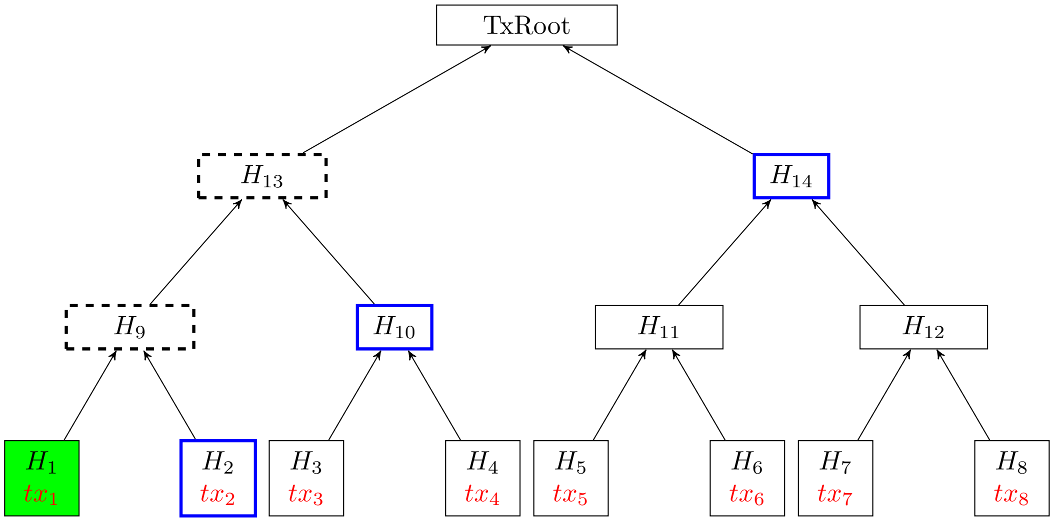

Figure 2: Tx Path

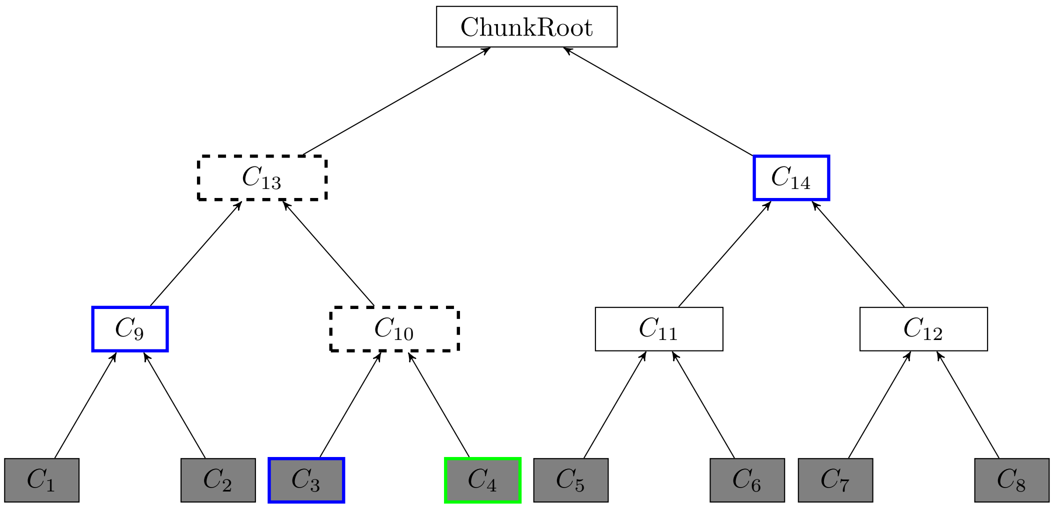

Figure 3: Chunk Path: the transaction data of tx_1 is segmented into 256-kilobyte chunks (gray).