A Pool allocation algorithm

Algorithm: MainChainWeightAllocation

Data: $BC_{(i,j)}$ am ordered subset of the blockchain
Result: $blockProfits$ with the weighted profit per block

begin
  $sum \leftarrow 0$
  $w \leftarrow [\emptyset$ for k in $range(j-i+1)]$ # Python-like creation of an array of $\emptyset$s
  for k *in* $0 : j-i$*:* **do**
    wins=getWinningSolves($BC_(i,j)$ $[k]$) # get $Solve_{CT}$'s that won their competition
    for *win in wins:* **do**
      $sum \leftarrow sum + getPF(win)$ # Half the promised fee of the competition of win
      $w[k] \leftarrow$ append($w[k]$,$getWinnerID(win)$)
    end
  end
  $W_t = getTotalCardinalWeight(w)$
  $blockProfits \leftarrow [0$ for k in $range(j-i+1)]$
  for k *in* $0 : j-i$*:* **do**
    $blockProfits[k] = \frac{|w[k]|*sum}{W_t}$
  end
  Return blockProfits
end

B Hash puzzles

Definition B.1 A hash function H maps input strings to output strings and have the following five properties [1]:

1. *Any input, fixed output*: H takes an input of any size and outputs a string of a fixed size n. That is, $H : \{0,1\}^* \rightarrow \{0,1\}^n$.

2. *Collision resistance*: it is computationally infeasible to find two inputs x and y such that $x \neq y$ and $H(x) = H(y)$.

3. *Hiding*: It is infeasible to learn what x is from knowing $H(r||x)$ where $||$ denotes concatenation and r is a randomly selected value from a distribution F whose minimal entropy is high.

4. *Puzzle friendliness*: there is no algorithm with an average case complexity better than $o(2^n)$ to find x given $H(k||x)$, where k is a randomly selected value from a distribution F whose minimal entropy is high.

5. Cheap validation: for any input x of size m, $H(x)$ is computable in $O(m)$.

Hash puzzles are only good for showing that the miner performed many computations and cannot be used for practical computational tasks. This major drawback can prevent using PoW in very large systems as the energy consumption can be too large.

C Extant storage blockchains

C.1 Filecoin

Filecoin [2] manages two markets. First, the *Storage Market* allows clients to pay the storage miners for storing data. This market audits *in-chain*: the management process is done using transactions that are recorded in the blockchain. Second, the *Retrieval Market* allows clients to retrieve data by paying data retrieval miners for delivering data. This market audits off-chain: there is a private process taken between the client and the provider and the internal steps are not published in the blockchain. In both markets, clients submit *bid* orders to the market to request a service. In response, the miners submit ask orders to offer a service, when both parties agree on a price, they jointly create a deal order. In both cases the output deal is signed digitally and is published as a transaction in the blockchain. The off-chain component of data retrieval is that the client-miner negotiation is done privately by them. A filecoin block is divided into three parts:

I. Orderbook: a data structure that holds in-chain bid, ask, and deal transactions of the Storage Market.

II. Transactions: records that hold payment transactions and client transaction. Filecoins transactions act as collateral - clients deposit the funds specified in the order, guaranteeing commitment and availability of funds during settlement.

III. Allocation table: holds pointers to the stored data. The miners storage is partitioned into sectors, where each sector contains data pieces assigned to the miner. The Network keeps track of each sector assignments to miners via this allocation table. When a storage miner sector is filled, the sector is sealed. Sealing is a slow, sequential operation that transforms the data in a sector into a replica, a unique physical copy of the data that is associated with the public key of the storage miner.

Trustless storage systems have to prove that it stores the data. This is achieved in Filecoin using a method called Proof-of-Storage. The miner publishes a proof on chain using a challenge-response protocol every t time. Filecoin introduced new methods that implement this protocol such that a miner can prove the data exists in storage. Their challenge-response mechanism is called PoRep (proof of replica) and PoSt (proof of spacetime) and it is different from other blockchains like Storj. PoSt schemes allow a user to check if a storage provider is storing the outsourced data for a certain time range (and not only at the time of the challenge).

In order to retrieve the data efficiently, the storage system cannot wait for auditing each operation in-chain. Therefor filecoin uses an incremental micropayment scheme. In this scheme, two party setup a multi signature *channel*, funds are placed into this channel. This channel is represented as an entry on the public ledger. In order to spend funds from the channel, both parties must agree on the new balance. The current balance is stored as the most recent transaction signed by both parties. Bitcoin lightning network is an example for a system that uses such schemes [3].

Data encryption: The encryption is made only by the client. There is no additional mechanism.

Blockchain maintenance: Filecoin is based on a Proof-of-Stake consensus protocol where the stakes are based on the storage assigned to each miner. This is an elegant PoS solution with is similar in spirit to our vstake mechanism. However, Filecoin does not address the N@S problem, nor is it designed to handle computational work.

Flow in the Storage Market:

- To store the data, clients first publish a bid transaction on the blockchain. Miners can then see it and in response publish their ask transaction. If the negotiation is successful, the client and the miner both sign a deal and publish it on the blockchain.
- A pointer to the data is written in the allocation table.

- The miner is paid for storing the data over time. Every t time it publishes a PoSt and gets paid a fraction per proof.

Flow in the Retrieval Market:

- Retrieving the data is done off-chain through a negotiation between the client and the miners holding the data (exchange ask, bid, deal p2p without auditing it on-chain). When the deal is done and signed by both parties, the client receives the data.
- The payment is done using a micropayment mechanism, which is similar to a lightning network, and a payout signature is published when retrieval is done (for payment extraction). Thus, the public ledger contains information that the deal was completed.

C.2 Storj

Storj [4] is an application running on the Ethereum blockchain. Storj uses an ERC-20 token called STORJ for making payments. Storj uses a trusted server called bridge. The bridge is used for delegating trust to dedicated server that manages data ownership and payments.

The Storj market is based on a contract negotiation between the storage provider and the client. If they both agree on a price then data can be stored. The client splits the file into shards (encrypted parts of the file). The negotiation is mediated by the bridge. The bridge can sniff the network and find storage providers. Note that Storj heavily relies on bridges throughout the pipeline and it is therefore not fully decentralized.

After the negotiation, the client sends the shards to the miners. Payment is done by micropayment mechanism (like in lightning networks). The bridge verifies that the miner holds the shard periodically by Proof-of-Replication mechanism (challenge-response) and pays the storage provider on every proof.

Retrieving the data is performed by negotiation of the bridge and the miners holding the data. Payment is also carried by a micropayment mechanism.

Data encryption: Encryption is made by the user. The bridge also takes a park in maintaining the privacy of the data by dividing the file into shards. Only the client and the bridge know where these data are. As a result, gathering all the data for going back to the information in the input files cannot be done by a malicious adversary.

D Our adapted storage protocol

Clearly, one can use any storage to keep task data (e.g., centralized cloud services such as Amazon or Google Cloud). However, the entire protocol would become centralized and tightly dependent on an external provider. Another issue is inefficiency in that: (1) data will be copied twice for validations, and (2) it will not exploit the fact that miners that hold the data can easily validate solutions and there is practically no added heavy cost to the hardware they need to keep. The dTMNs (task-specific masternode networks) take upon themselves the maintenance of the task data and the validation of the solutions once the competition phase is over. Our default suggestion is to take Filecoin as base code, and make the storage ´smarter´ by supporting validations.

Storage constraints and collaterals

In this section we adopt standard storage-based notation that uses time span instead of counting blocks to allow more granularity. Below we define parameters and their constraints.

- Let T_x be the block number at time T auditing the transaction x.
- Let $timeout_x$ be the number of blocks for timeout transaction x.

- *bid* request becomes invalid after $timeout_{publish}$ blocks starting from $T_{publish}$
- A miner that publishes an ask request in T_{ask} must lock the needed resources for $timeout_{publish} + T_{bid} - T_{ask}$
- The dTMN of CT promises to store the dataset for at least $timeout_{freeze}$ starting from T_{stored}
- After T_{solved} storage miners must keep the data for a $timeout_{retrive}$

dTMN problem gathering

The client starts by submitting a bid request (in-chain) in a transaction called $Publish_{CT}$. Then, storage miners submit ask requests. Only after at least r_s miners publish an ask request, the client and the miners sign joint deal transaction, which we previously denoted as $Stored_{CT}$. Using a multi-signature process, this group becomes the dTMN that provides services. All subsequent transactions, including those providing PoSt, are made as a group via one multi-signature transaction.

In order to minimize the number of in-chain transactions, storage services are audited using micro-payment method. Micro-payment channel is used by storage miners and computational miners, as well as with the client on solution retrieving. Proofs of micropayments (e.g., claim micropayment transaction) are done once by the dTMN after the computational competition process is completed. The same holds also for PoSt proofs. Having said that, a single node of the dTMN may also submit a transaction when the dTMN parties are not honest. dTMN may become smaller due to parties failure, therefore the network keeps following the dTMN size. If the dTMN gets too small (e.g., less than $\frac{r_s}{2} + 1$) then the network considers this dTMN as failed and the computation process is stopped.

The dTMN, as an efficient storage network unit will send and receive data in chunks. In order to keep high-availability, each node in the dTMN holds the whole copy of the data. This is achieved by chunk exchange. Chunk exchange is highly beneficial here because it is much faster than sending the whole dataset to each miner.

References

[1] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies Introduction to the book*. 2016.

[2] ProtocolLabs, *Filecoin: A Decentralized Storage Network*. PhD thesis, 2017.

[3] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," *Technical Report (draft)*, p. 59, 2016.

[4] S. Wilkinson, T. Boshevski, J. Brandoff, J. Prestwich, G. Hall, P. Gerbes, P. Hutchins, and C. Pollard, *Storj: A Peer-to-Peer Cloud Storage Network*. PhD thesis, 2016.