Written in collaboration with Siva Dirisala, CTO,0Chain.net

A brief intro - At 0Chain we wanted our dStorage solution to be as secure and reliable as possible but still offer a cheaper, faster, more secure and more available storage than AWS or DropBox to make it easy for the cloud to move to the edge.

Advanced storage solutions initially centered around dedicated hardware and pre-allocated capacity for a given workload. That changed with the advent of hyper-converged infrastructure (HCI) that separated compute from storage and allowed each to scale independently. Storage solutions were also stratified into primary and secondary storage; primary storage is accessed by the realtime workloads such as databases and email servers, and secondary storage is used for replication, backup, and recovery.

Many players have emerged to offer dedicated primary storage and dedicated secondary storage solutions. In addition, the arrival and growth of the cloud storage in the last decade made these traditional storage providers to rethink from dedicated appliance mindset to offering cloud native solutions with a pure software play.

In order to avoid a vendor lock-in, most solutions provide multi-cloud and hybrid cloud integration strategies. The adoption of blockchain technology as a mainstream solution has made some players to consider making storage itself a decentralized commodity that can be offered at a much cheaper price without sacrificing the reliability of the big players. Some of the already existing initiatives dating back to a couple of years are Sia, IPFS, and FileCoin.

First, let’s discuss some inherent challenges with storing data in a decentralized environment.

1. How to ensure that the data is not lost?

2. How to make sure that even if one or some of the decentralized parties are down, the overall service is not effected for every single stored file or object?

3. How to make certain that the decentralized parties are really storing the data for which they are getting rewarded?

Below, we will explore each of these challenges and the approach 0Chain dStorage has taken.

Data Availability

Traditional storages offer two types of solutions. One is replication where each data is stored multiple times and in multiple locations so when one disk or location has a fateful incident, the data is recoverable from the other.

The other option is to store the data in a special format by running it through what is known as erasure coding. Erasure coding can be best understood as a set of equations required to solve a set of variables. That is, if there are 4 variables, one only needs 4 equations to solve them. Anything more either produces inconsistent result or is redundant. In case of redundant equations, any 4 will give the same result for the 4 variables. Hence, by converting the data into a format that provides the inherent redundancy, it is possible to recover if some parts are lost. An example encoding is 10-16 erasure coding where 10 original data blocks go through expansion into 16 blocks. This gives a safety where even with the loss of any 6 of the 16 blocks, the original 10 blocks can be recovered.

0Chain dStorage uses the erasure coding technology to ensure data is available.

Data Performance & Reliability

While decentralized systems bring their own complexity, they do well in the inherent reliability they provide. That is, when one or some of the decentralized parties are not active, the rest can move forward. So, after going through the erasure coding and expanding the original data, each expanded piece is stored with a different service provider (we call them blobbers). This results in a few advantages.

1. Even with 6 blobbers down, the original data can still be restored from the remaining 10. As these are independent blobbers, their failure (hardware, network, natural disaster) probability is mutually exclusive, resulting in a very high reliability.

2. Data can be uploaded/downloaded in parallel to all the 16 blobbers. This results in a tremendous network time savings. For example, a 10 MB file expands into 16 MB due to erasure coding but each blobber is uploaded only 1 MB. So, in effect, the upload time will be as if the client is uploading just 1 MB file. Of course, the assumption here is that the client has enough network bandwidth to upload 16 MB simultaneously to 16 different service providers. But the point is, the upload time is now dependent on the bandwidth of the client rather than the bandwidth of a single server on a traditional storage platform.

3. We are all used to sub-second search results on Google. One way they do it is by sending your search terms to multiple servers but consolidate and rank only the ones that arrived within a short amount of time. This helps eliminate the long-tail effect and provide an improved quality of service. This type of sophisticated algorithms are possible only with distributed systems with ample redundancy. The point here is that, when you download your file from 0chain dStorage, the client tries to download it from all the 16 blobbers but it only needs to wait for the first 10 to reconstruct the data. This results in a fast and smooth download experience. It would be interesting to note that the 10 blobbers might be dynamically changing as the data is being streamed. This eliminates jitters and makes the streaming

experience much smoother.

Always-on Data Protection

Data protection is an issue for a traditional storage providers as well, but in a different context than Byzantine conditions, and it is more due to bit rotting and issues related to hardware. For this, the traditional storage systems have a means of self-validating and recovering.

In a decentralized system, there is no way to enforce the operational practices and quality of each service provider just by policy. This assurance needs to be part of the protocol. Just like in Proof Of Stake based consensus algorithms that offer rewards as well as slashing of stake, the service providers can be rewarded and punished for passing or failing a challenge that ensures that the provider is doing what is required. 0Chain uses smart contract driven challenges for the blobbers. The challenge itself is completely random (but deterministic for a given block to ensure consensus). The blobber and the specific file and a fragment within the file is picked for a challenge in a random manner. Such randomized systems have statistical outcomes that can be tuned to achieve a desired level of quality of service.

Ensuring high quality requires doing frequent challenges. Hence the challenge protocol should be as light as possible in terms of bandwidth requirements. At the same time, it is important to be able to validate as much as possible. Say a file is stored in 64KB blocks and the file size is 10MB. Challenging the entire file content stored at a blobber (1MB after erasure coding) is a waste of lot of network bandwidth. Challenging any of the random blocks reduces it to just 64KB overhead. However, this has a potential for a blobber to download the content, reconstruct it and then serve the 64KB he is supposed to have stored. Note that 0chain storage offers both private and public content. The private content can only be read by the owner or any user authorized by the owner. Hence, a blobber will not be able to download the private content to pass the challenge. The following attack scenario and the defense against is valid only for public content although the challenge behaves exactly the same in both scenarios.

Outsourcing Attack

Our protocol avoids outsourcing attack by ensuring that the content provided for verification is 64KB but the content required to create this verified content is the full file fragment. This is done as follows. The file fragment of 1MB stored with the blobber is divided into 16 64KB blocks. Each of these 64KB blocks are further are divided into 64 byte chunks (just for discussion here, but the final chunk size will be fine tuned or dynamic). There are 1024 such chunks in each 64KB block that can be addressed using an index of 1 to 1024. Now, imagine that the data at each of these indexes across the blocks is treated as a continuous message and hashed. Then these 1024 hashes serve as the leaf hashes of the Merkle Tree. The root of this Merkle tree is used to roll up the file hashes further up to directory/volume level. The Merkle proof provides the path from the leaf to the file root and from the file root to the volume level. With this model, in order to pass the challenge for a file for a given index (between 1 and 1024) a dishonest blobber first needs to download all the content and do the chaining to construct the leaf hash. This discourages them to avoid storing the content and engage in an outsourcing attack.

Validators

In addition to blobbers, the dStorage also relies on a special role called Validators. Their role is to validate and give verification signatures that the blobbers can collect and with enough signatures, they establish their validity to the blockchain. The fact that there are multiple validators is beneficial for the following reasons

1. Scales with storage growth and number of blobbers on the network
2. Individual validators cannot get the blobbers punished
3. Increases the reliability of the network in case of validator downtime

Parallel Async blockchain transactions

All reads and writes have to get registered on the blockchain for the following reasons

1. the payments to the blobbers are happening on the blockchain via the storage smart contract (hence completely decentralized)
2. there is an audit trail of every read and write to the storage. This audit trail is useful in creating not only security driven applications but also monetization applications that needs an indisputable proof of what actions have been performed by users for which the content owners need to be paid.

Because of this requirement to record every read and write on the blockchain, it might indicate a bottleneck compared to a traditional storage model. 0Chain protocol is designed specifically to reduce this bottleneck. This is done by making the blockchain transactions asynchronous to the actual read/write operations. This is done by making use of read and write markers that can be cryptographically verifiable. The users present these markers to the service providers and get started right away and the service providers redeem these markers for rewards offline. Since the blockchain itself is trusted, and the markers are valid, the service providers are guaranteed to be paid.

I would be interested to know what dApps on Ethereum are currently using and any feedback to our solution would be appreciated. Thanks in advance.