Atomchain Storage: A distributed-decentralized storage (ddStorage)

Author: Okpara Okechukwu D.
Email: okpara.net@gmail.com
Website: www.okpara.net
Date: 01-05-2022

Abstract:
We propose a method of distributed decentralization of files, data structures, and persistent algorithms for atomchain technology, which aims to eliminate the inefficiencies of a traditional decentralized storage (dStorage).

Keywords: atom network, atomchain, dStorage, ddStorage

1. Introduction

Decentralized storage systems consist of a peer-to-peer network of user-operators (or nodes) that hold a portion of the overall data, creating a resilient file storage sharing system. All data of atomchain is stored mainly in active nodes. These nodes are connected through the P2P protocol.

When a user initiates a transaction, the transaction is broadcasted through the P2P protocol, and the atom creator (or miner) node will verify it, package it, and then broadcast it to the atom network.

If atomchain were to grow steadily, expanding to large amounts of data it will not be feasible for all atomic (reactor) nodes to continue to run. And the cost of deploying such huge data in the future to Atomchain's Atom Network would be prohibitively expensive due to computation energy fees.

2. Distributed-Decentralized Storage (or ddStorage)

This involves the combined effort of applications on atom networks running atomchain protocols. Atom network is both decentralized (there is no central node or server that controls the operations of the network) and distributed (multiple nodes work together, for instance by sending messages).

Atomchain’s ddStorage service is like a decentralized program running on a distributed platform using trie data structures to manage data [1]. We shall delve into ddStorage mechanisms by considering four aspects of a decentralized storage (dStorage):

- Persistence mechanism / incentive structure
- Data retention enforcement
- Decentrality
- Consensus

3. Persistence Mechanism / Incentive Structure

Persistence mechanism is needed for a piece of data to persist permanently. Here, the whole chain needs to be accounted for when running an atomic (reactor) node. A new piece of data is bonded

into the last atom (atomsphere), and it continues to grow – requiring every node to replicate all the encapsulated or embedded data. This persistence Mechanism has an incentive structure, in which a cryptocurrency payment is made to the atomsphere creator. The atom creator (or miner) node is paid to add the data on the atomchain. Moreover, quark objects (using omega atom wallet addresses) can be used to store the hash of the location of a piece of data. To keep the data persisted, these quark objects or codes must be continually funded.

4. Data Retention Enforcement

To ensure data retention, Atomchain is expected to use some type of cryptographic challenge that is issued to the nodes to make sure they still have the data. Here Atomchain issues a challenge to the active nodes to see if they have the data at both the most recent atom and a random atom in the past. If a node is found wanting or deficient, it is penalized.

5. Decentrality

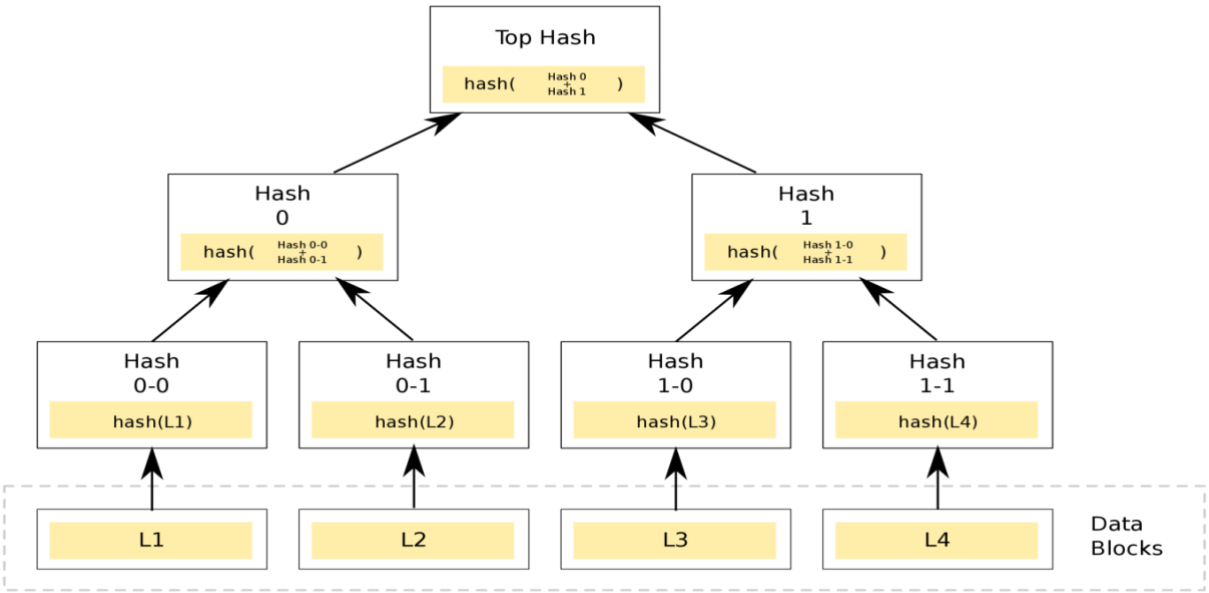
Atomchain decentralization may be measured by an application that can account for the entire atom network. In particular, a decentralized application that can check a given cluster of atom fields' analytics and metrics.

6. Consensus

Atomchain main consensus mechanism is based on atomic bonding.

7. Transaction Storage

According to Wikipedia [2], a Merkle tree is typically a binary tree typical binary tree.



Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains. Therefore, using a Merkle tree for our transactions will allow atomchain to be as compact as possible in terms of disk space, as well as allowing for secure verification of transactions.

From the Atomchain Network Whitepaper [3], atomic nodes will contain the whole transaction history while orphaned atomic nodes will contain the Merkle root hash.

8. Conclusion

In the future, atomchain ddStorage may also support compatible content distribution and file referencing dStorage platforms for storing data; as well as multi-chain ddStorage compatibility.

9. Definitions

- A **binary tree** is a tree whose elements have at most 2 children. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

- A **Merkle Tree** is a tree in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes. For example, in the picture hash 0 is the result of hashing the concatenation of hash 0–0 and hash 0–1. That is, hash 0 = hash(hash(0–0) + hash(0–1)) where + denotes concatenation.

- A **tree** is a collection of nodes, where each node is a data structure consisting of a value, together with a list of references to nodes.

10. References

[1] Okpara O. D. 2022. The Atomchain Wallet Whitepaper, https://okpara.net/AtomWallet.pdf

[2] Retrieved on 01-05-2022 from https://en.wikipedia.org/wiki/Merkle_tree

[3] Okpara O. D. 2022. The Atomchain Network Whitepaper, https://okpara.net/AtomNetwork.pdf