Decentralized storage solutions on the Blockchain

Dimple Madhwani
D17B 38

Introduction to Data Storage

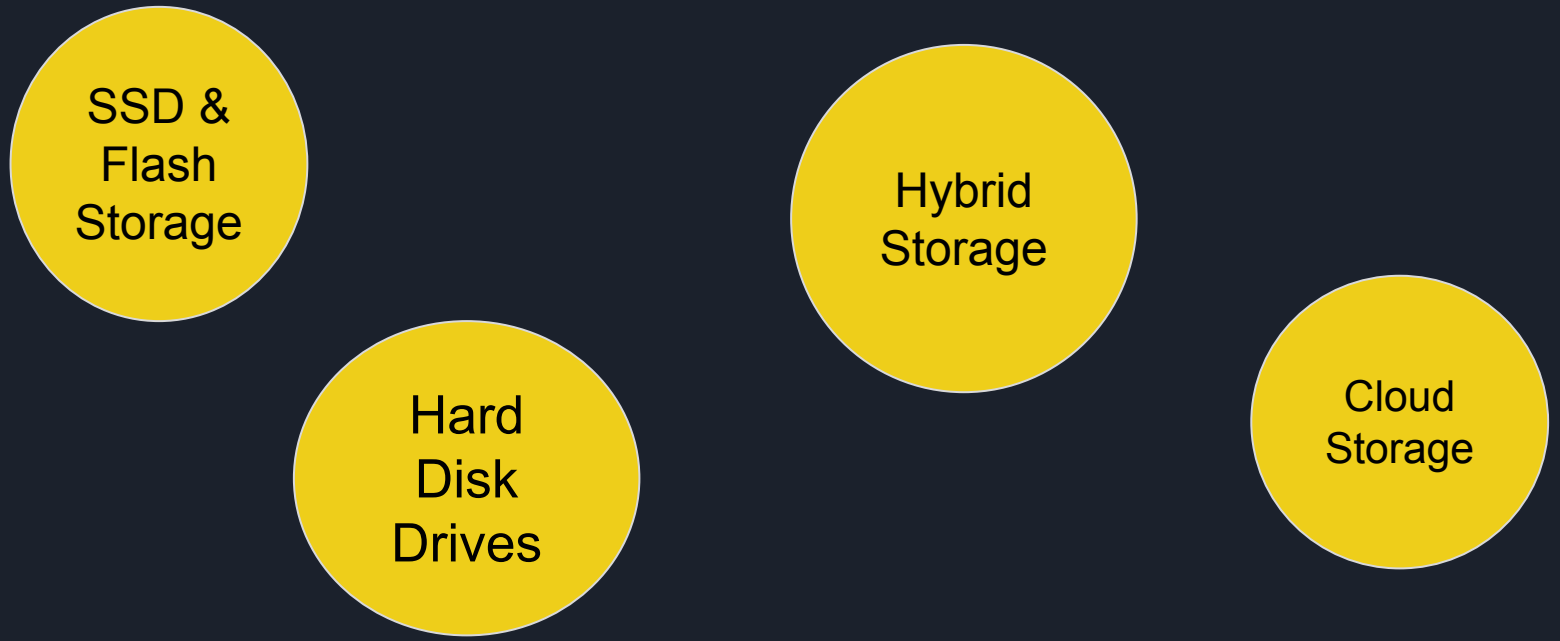
Data Storage Server System:

A storage server is used for storing a large amount of data files

It is used to transfer large files over a network.

When organization expands, the need for data storage increases.

Different Types of Data Storage Devices



Centralized Storage System

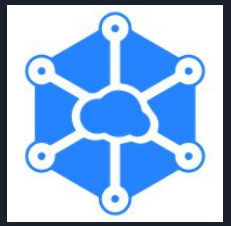
When data is stored at a single location and shared among users with the help of servers. It is also known as networked storage.

Example:

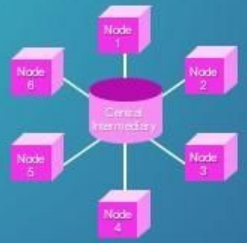
Decentralized Storage System

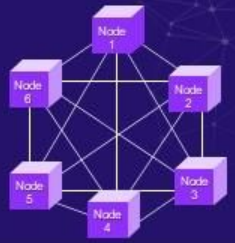
Decentralized Cloud storage is a system where data is stored on multiple servers. It is a peer-to-peer decentralized cloud storage solution.

Example:



Centralized vs Decentralized Data Storage System





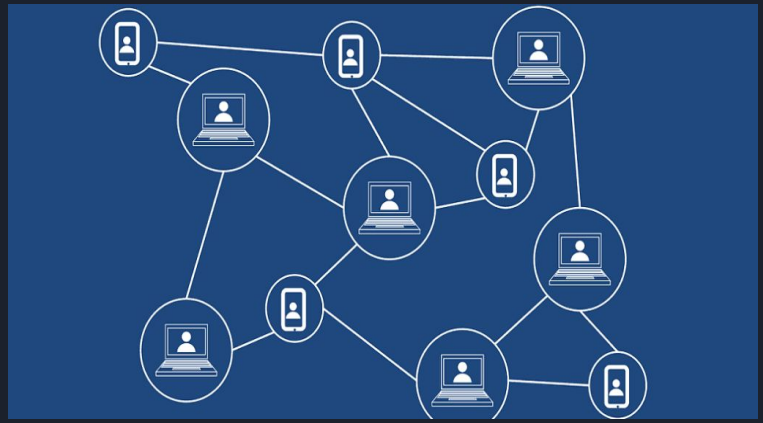
Centralized	Comparison Characteristics	Decentralized
Data must flow through a central point (server)	Data Flow	Data flows through points without any single specific point
Data communication flow is vertical	Data Communication Flow	Data flow is always open and free
The responsibility lies with the central point, which can be one server	Decision Making	Difference in bullet indent; please check
Mass adoption	Adoption	In the early phases of adoption
Systematic reservation of authority in a network	Authority	Systematic distribution of authority among access points

Blockchain based Decentralized Data Storage

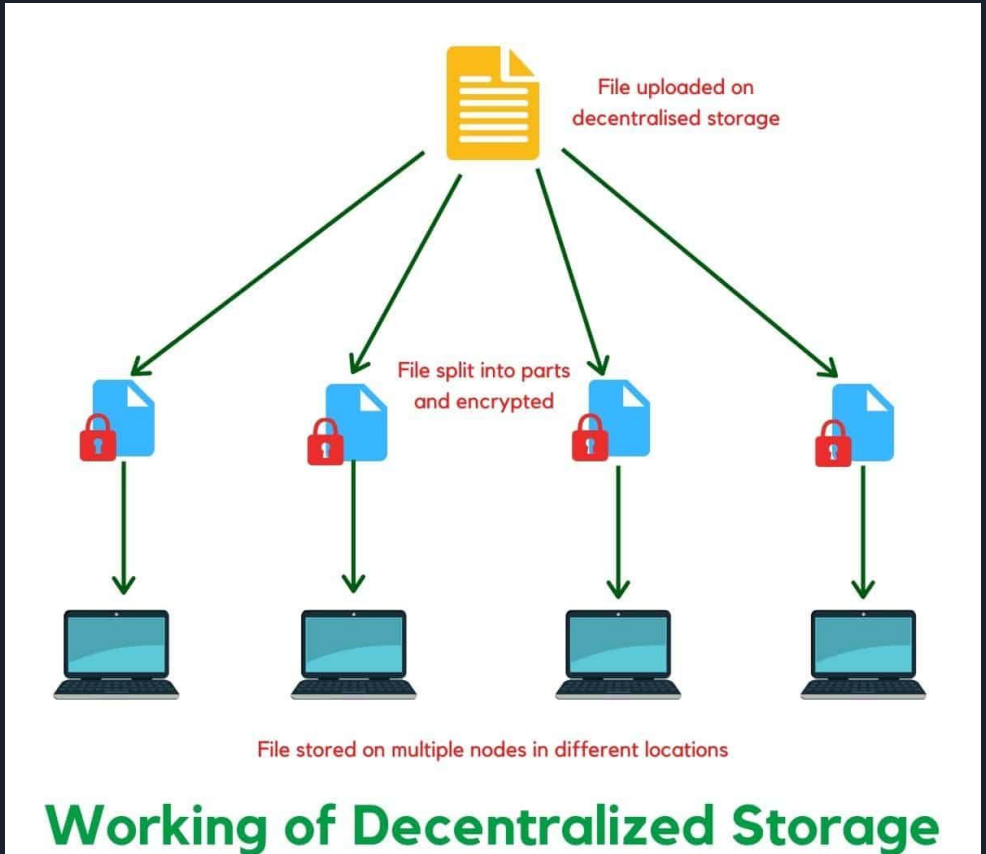
The blockchain based decentralized cloud storage system divides users files into small chunks of a data in the form of blocks.

Each block is then encrypted with a unique hash or with public-private keys.

Some blockchain-based storage systems use tokens or cryptocurrencies to incentivize node operators and users to contribute storage space and bandwidth to the network.



Blockchain based Decentralized Data Storage



Advantages of decentralized blockchain storage

Low Costs -> decentralized storage can be more cost-effective than traditional centralized storage, especially for large-scale or long-term data storage.

Data Security -> Decentralized storage relies on encryption and cryptographic techniques, making it highly secure

Scalability -> Decentralized storage systems can potentially scale more easily than centralized ones.

Conclusion

In conclusion the content elaborated about the importance of the decentralized storage networks that are intrinsically based on blockchain technologies. case. Due to the decentralized and peer-to peer nature, blockchains have the potential to make a significant impact on business across many industries.

References

A Comprehensive Survey on Blockchain-Based Decentralized Storage Networks by MUHAMMAD IRFAN KHALID1 , IBTISAM EHSAN 2 , AYMAN KHALLEL AL-ANI 3 , JAWAID IQBAL 4 , SADDAM HUSSAIN 5,6, SYED SAJID ULLAH7,8, AND NAYAB

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10026822

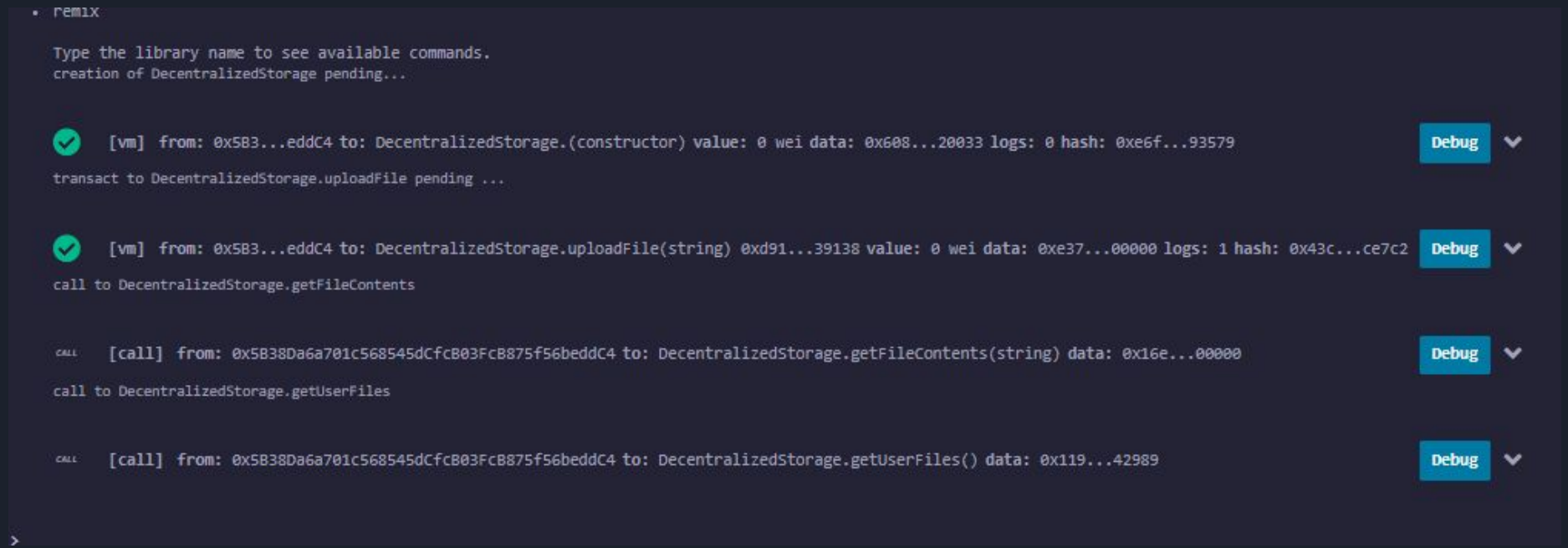
Smart Contract

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract DecentralizedStorage {

    // Mapping to store file hashes by user

    mapping(address => string[]) private userFiles;

    event FileUploaded(address indexed user, string fileHash);

    modifier fileExists(address user, string memory fileHash) {

        bool exists = false;

        for (uint i = 0; i < userFiles[user].length; i++) {

  if (keccak256(bytes(userFiles[user][i])) ==
keccak256(bytes(fileHash))) {
                exists = true;
                break;
            }
        }
        require(exists, "File does not exist.");
        _;
    }

    // Store a file hash associated with the sender's address
    function uploadFile(string memory fileHash) public {
        address sender = msg.sender;
        userFiles[sender].push(fileHash);
        emit FileUploaded(sender, fileHash);
    }
```

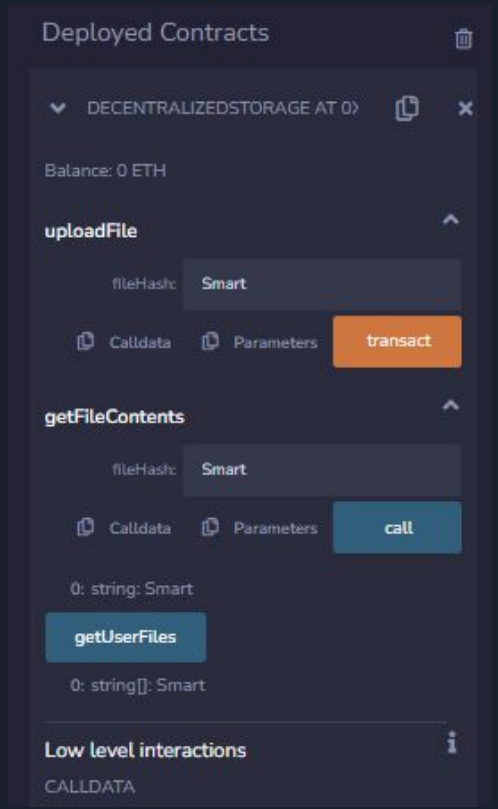
Smart Contract

```
  // Store a file hash associated with the sender's address

    function uploadFile(string memory fileHash) public {

        address sender = msg.sender;

        userFiles[sender].push(fileHash);

        emit FileUploaded(sender, fileHash);  }

    function getUserFiles() public view returns (string[] memory) {

        address sender = msg.sender;

        return userFiles[sender];

    }

    function getFileContents(string memory fileHash) public view fileExists(msg.sender, fileHash) returns (string
memory) {

        return fileHash;

    }

}
```

Smart Contract



Smart Contract



Thankyou