CSE223B: Blockchain File Storage Service

Jiachen Niu
UC San Diego
j4niu@ucsd.edu

Aaron Lee
UC San Diego
acl049@ucsd.edu

Abstract

Blockchain has been widely regarded as the future medium of financial transactions. It's decentralized, trust-based, and purely virtual. As blockchain is being adapted to new scenarios, new demands also require adaptations of the original design of blockchain. Higher-throughput and faster block synchronization and communication are desirable properties of a blockchain system. However, the blockchain trilemma: decentralization, scalability, and security, forces developers to make a tradeoff that prevents them from achieving all three aspects.

In this project, we built a file storage system on top of Solana (Yakovenko, 2018), a commercial blockchain that utilizes modern consensus algorithms like Proof-of-Stake and ordering algorithms like Proof-of-History to give high throughput and consistency guarantees. Our source code can be found at https://github.com/lee-aaron/CSE223B.

1 Background

Solana is a relatively new blockchain released in March 2020. It touts being able to support thousands of transactions per second for a relatively low cost per transaction. We chose this blockchain to benchmark its performance given what it claims to be able to handle. We were interested in seeing if we could provide a low latency, high performance file storage service on top of a blockchain. We also wanted to examine how scalable this service is so we plan to test it on multi node clusters.

Solana utilizes a Proof-of-History algorithm that does not require nodes to synchronize with the entire network before generating the next block. We would like to examine Solana to determine their performance in a file storage setting. At a high level, we plan to use metrics such as transaction throughput, ping time, network bandwidth, and CPU utilization to illustrate the different scenario performances. We'd also like to develop a client / application that can interact with the blockchains to help measure performance for the scenarios mentioned above. After obtaining metrics, we would like to optimize the application by either implementing it with HTTP3 or finding ways to reduce compute and latency times.

Solana also uses a Proof-of-History algorithm to validate transactions. Instead of trusting the timestamp on a transaction, this algorithm uses a high frequency Verifiable Delay Function to cryptographically verify the passage of time between two events. Thus, it does not really require consensus from a large amount of nodes.

2 Consensus Algorithms

Different from the traditional blockchain design (Nakamoto, 2009) which uses Proof-of-Work as the consensus mechanism, modern blockchains like Ethereum and Solana adopts a more eco-friendly and less computation-intensive approach called Proof-of-Stake.

In essence, consensus mechanisms make sure every participating nodes connected to the block chain network agrees on the next block to add to the chain. There are two possible approaches - either every nodes on the chain negotiates with each other to reach an agreement, or the system makes sure there's only one proposer for any single block. Taking the former approach would inevitably leads us to implement a Paxos-like consensus algorithm. Besides tricky to implement, such an algorithm doesn't scale well with thousands of nodes living on-chain (300,000 for Ethereum (Maeng et al., 2020)) and new nodes constantly joining and leaving. Therefore, both proof-of-work and proof-of-stake take the second approach.

2.1 Proof of Work

Proof-of-work (PoW) uses cryptography traits to ensure blocks are proposed at roughly a fix interval between each other (around 10 minutes (Nakamoto, 2009)). To propose a block, a node needs to solve a hard computation task. Examples of such tasks may be to compute a magic number called "nounce", which when included in the block, can make the hash of that block to have a fixed number of leading zeros. The number of leading zeros required can be adjusted to make the problem harder or easier based on the number of miners to ensure a roughly constant flow of newly mined blocks. By spacing out blocks being mined, the same block is unlikely to have 2 competing proposers at any moment.

Despite simple and elegant, running PoW on a blockchain has an unwanted side-effect of wasting too much computation power. When the number of miners increases, the computation task becomes harder, taking longer to solve and wasting more computation power collectively when a task is eventually solved. Those wasted computation power translates to wasted money, over-powerful hardware and generated heat. In addition, the 10 minutes interval for generating blocks is too slow for high-throughput applications.

2.2 Proof of Stake

Proof-of-stake (PoS) takes a less computation-intensive approach than PoW. It eliminates the need for hard computation tasks. In PoS, blocks are proposed by only validated nodes in the system called "validators". The system chooses only one validator in the validators pool for generating the next block and only accepts the block proposed by that validator. Since validators are delegated to propose a block instead of competing with others to propose a block as in PoW, this process of generating the next block is also called "validating" a block.

The process of becoming a validator is discussed in the next section (3.2), but a validator can be regarded as a system-trusted node to generate the next block. Since every block is validated by exactly one validator, there's no contest between validators. Validators can be idle while they are not assigned blocks, thus saving their computation power.

3 Security

A key selling point of blockchain is that it's decentralized, without single point of control, and robust against various attacks by malicious users. Those guarantees are both preserved and reflected in PoW and PoS.

There are mainly two kinds of attacks on a blockchain - adding malicious blocks and modifying existing blocks with malicious data. The design of blockchain already protect itself from the second type of attack. Since blocks are chained and their hashes are propagated down the chain, any modifications up the chain would be easily detected by later blocks. Therefore, both PoW and PoS need to protect against the second type of attack.

3.1 Proof of Work

The robustness of PoW against inserting malicious data rely on computationally expensive tasks. Nodes compete with each other to propose a block. Similarly, malicious nodes must compete with benign nodes to propose a block with malicious data. As long as no single entity can control over 50% of the total blockchain's computation power, the system is robust against malicious blocks being added into the blockchain. Because with more than half of the total computation power, the group of benign nodes will more likely to beat the group of malicious nodes in solving tasks and proposing blocks.

3.2 Proof of Stake

PoS takes a different approach by relying only on trusted validator nodes to generate blocks. A validator is recognised by the system as trustworthy by "staking" some of its blockchain currency in the system. Staked currency is withheld by the system while the node is actively validating new blocks.

A validator has a chance of validating the next block proportional to the amount itself staked divided by the total staked amount in the system. This constraint shows that the system innately regards validators with more assets staked as more trustworthy. However, since this selection process is random, it doesn't eliminate the possibility of a malicious validator introducing malicious data when it eventually is selected to validate a block.

To solve this problem, the system installs a punishment scheme for misbehaving nodes. Once a malicious block is detected with other checking measures by the system (discussed in 4.2), the system would incur a fine on the validator that proposed the block. When the staked money is reduced below the amount a validator can validate, that validator is kicked out. In Solana, the staked

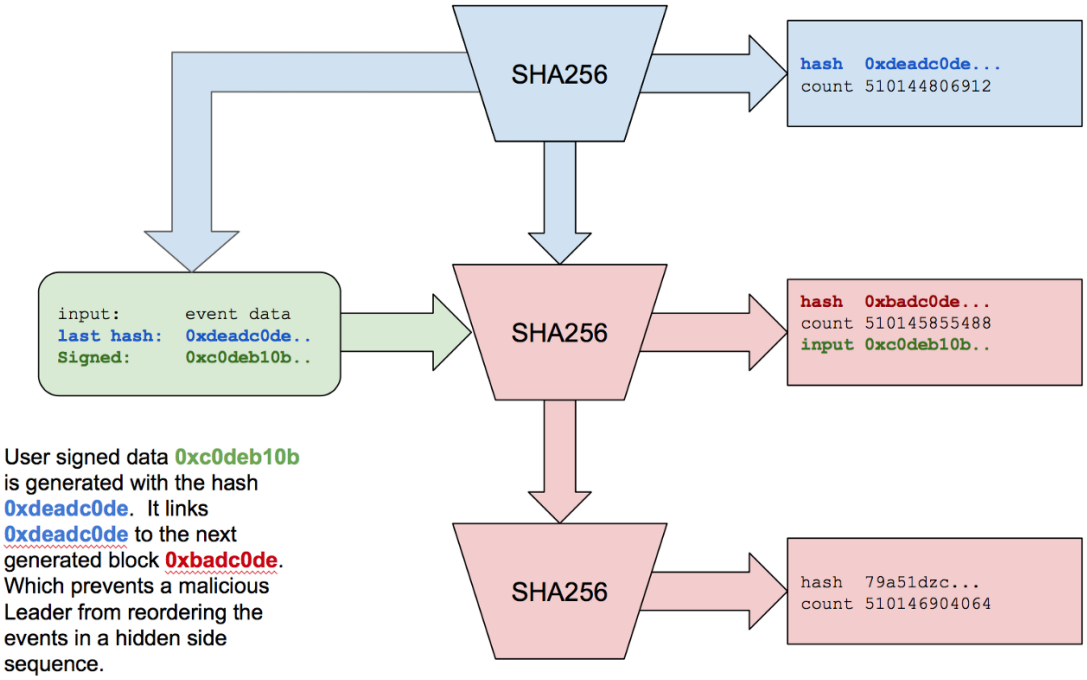

Figure 1: Proof of History Event Sequence Diagram

currency represents real money. Therefore, would-be-malicious users are strongly motivated to play by the rule as the reward and punishment of misbehaving aren't even.

Like PoW, as long as no one single entity controls over half of the total staked amount in the system, the blockchain is robust against inserting malicious blocks. However, this guarantee is less impossible to break compared to that of PoW since one can obtain an disproportional amount of wealth but not time spent on solving PoW tasks.

4 Proof of History

In both PoW and PoS, the "winning" node determines which transactions to include and the order of them in the next block on the chain. This is sufficient for serializing events and preventing double-spending as only the first of two duplicate transactions is deemed valid.

However, Solana takes one step further and uses Proof-of-History (PoH) (Yakovenko, 2018) to ensure orderings of transactions as they are issued by a client. The process is illustrated in Figure 1.

Without diving deep into the concept, PoH employs the same cryptographic function, as in any blockchain, to compute hashes for consecutive events and include their hashes in their next event blocks. The timestamp of each event is also incorporated in the computation of each hash. This timestamp records and ensures the order of events. Correct event ordering is easily confirmed by re-computing the hashes following the event sequence. Any reordering of the events will result in hashes mismatch.

Because of this guarantee on event ordering, Solana transactions are strongly consistent **if** they are issued or signed by the same user. This is a nice property to have for our application since our storage service don't have to worry about file operations, issued in transactions to the blockchain, being reordered as long as only one user issues the transaction at a time.

5 Application

Our application is a file storage service built on top of Solana blockchain. We use the Anchor framework to develop a smart contract with operations in the same style as that of a regular file system. As of now, our interface only supports creating and modifying files.

We build the smart contract backend using Rust and a client interface using TypeScript. The blockchain system is deployed on a local network connected to a cluster with 1-2 virtual nodes run as processes on a local machine.

5.1 Data Storage

Considering the structure of a blockchain and the immutable nature of previous blocks, we intuitively decided to mimic a log-structure file system in designing ours - data are stored as data blocks and inodes are used to reference and organize data blocks.

Because Borsh, the serialization library used by the Anchor framework, limits the amount of data to be transferred, each transaction issued to the blockchain can only have a fixed size of around 1 kilobyte. Therefore, we divide files into around 900 bytes chunks to be stored/read to the blockchain. Every block of file data in the blockchain has a unique hash which Solana calls a `PublicKey`. Note this hash is different from the actual hash used to maintain blockchain's integrity. It's an application-level hash Solana uses to identify blocks on chain.

We use `PublicKey` to reference data blocks, and group `PublicKeys` referencing blocks in the same file into inodes. Inodes are also stored on the blockchain and are subject to the same space limitation. One inode block can store up to 30 direct data block `PublicKeys` and 1 inode block `PublicKey` referencing the next inode block. This chaining of inode blocks is simple to implement and space-efficient since client only needs one `PublicKey` pointing to the head of the inode block to reference an arbitrarily large file stored in the chain.

5.2 Smart Contract

We currently have two sets of smart contract operations on the two different types of blocks on-chain. For both data and inode blocks, we have implement the following operations:

- `create(userPubKey)`
- `get(blockPubKey): data`
- `set(blockPubKey, data)`

The application currently assumes only the creator of a block can modify but anyone can read that block. Therefore, the creator must sign the `create` and `set` transactions.

5.3 Client Application

We implement a more user-friendly client interface which provides the following operations to interact with the blockchain:

- `upload(userPubKey, filePath): inodePubKey`
- `download(inodePubKey, filePath)`

To upload a file, its content is cut and wrapped into fixed sized blocks and stored on-chain with inode blocks. Because Solana uses application-level `PublicKeys` to reference blocks and those hashes can be obtained before data is stored on-chain, we can make the uploading step concurrent and reduce response time. For a 1 megabytes file, this optimization reduces the uploading time from 16 minutes to 1 minute.

To download a file, the opposite operations are performed. Direct and chained inode blocks are read and data blocks referenced are read to a file. We could make this step concurrent by dispatching each thread to grab all data referenced by one inode block in the inode chain. However, we didn't make this optimization since reading a block is very fast compared to block writes given that no signing and security checks are performed. The client can be easily modified to concurrent reads if necessary.

5.4 Testing Framework

To test the performance of our system, we wrote a simple client that uploads a file to the blockchain and then downloads that same file. The content of the downloaded file is compared against the original file for data integrity checks. The runtime of downloading and uploading operations are recorded from the client side.

6 Experiment Results

Our system is deploy both in an WSL2 system on Windows and AWS Ubuntu 20.04 machine in the cloud. Our tests are exclusively performed locally to the machine that the system is deployed on. We believe the results obtained from our benchmark tests establish the absolute optimal running time of our system.

We measured the performance of our system and identify where optimizations or bottlenecks are. We also tracked the TPS and other metrics via a Grafana dashboard monitoring our cluster. We tested our service on a single node and two node cluster. We used various sized files and measured times to upload and download those files. Furthermore, we examined the TPS and Confirmation Time metrics from the Grafana dashboard at the end of running all our tests.

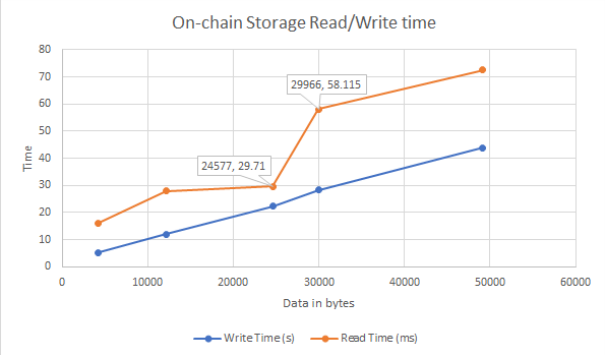

Figure 2: Performance of Upload and Download

File Size (KB)	Upload (s)	Download (ms)
10	0.954	11.592
20	1.171	17.675
30	1.565	27.489
40	2.029	31.47
50	2.163	34.236
100	4.379	68.164
150	6.369	95.255
200	8.338	119.586
1000	62.04	839.93
10000	631.75	6733.00

Table 1: One Node Cluster with Different File Size

6.1 Upload and Download Performance

We did some initial correctness test with arbitrary text file we found on the web. The result is shown in Figure 2. The upload time grows proportionally with file size due to increasing numbers of data blocks being transferred. The download time also grows with file size but the trendline has a steep slope when the file size grows to 30KB. This is because when a file grows to be larger than 30KB, it needs an additional inode block since there's only 30 direct block reference slots. This increase in runtime is spent on reading that additional inode block for files over 30KB and less than 60KB.

We can see from Figure 3 the overall dashboard of how the network handled transactions for just one node. The network was able to handle a max of 135 transactions per second and have average confirmation times for transactions to be less than 1 second.

We also analyzed the upload and download times for various sized files. In Table 1, we can see that the performance sort of inversely scales with the file size. It's intuitive that reading a file from the blockchain should be much faster than uploading because the validators need to verify the transactions in order to append the transactions in the block.

Validators	Upload Time (s)	Download Time (ms)
1	27.880	579.2838
2	34.784	726.2854

Table 2: Two Node Cluster: File Size 349 kilo bytes

Validators	Upload Time (s)	Download Time (ms)
1	0.878	11.727
2	1.039	27.0658

Table 3: Two Node Cluster: File Size 1 byte

6.2 Scalability

Then we tested how adding new validators can impact system performance. We deployed a 2-validator cluster and run the same test with both a 300 kilo bytes and a 1 byte test file. From Table 2, we can see the average upload time for 1 node cluster is 7 seconds faster than a 2 node cluster. This is expected as data uploaded to one node needs to be propagated to other nodes to be persisted.

We also attempted to measure the transaction overhead in clusters with different number of validators. From Table 3 we observe that the transaction overhead increases by 0.2s with a 2-validators cluster compared to the baseline transaction overhead of 0.8s in a 1-node cluster. If we add a 0.2s time increase to all transactions in the previous test, the upload time for a 300KB file in 2-validator cluster would be around 70s. This shows our concurrent upload code reduces the upload speed at least by half.

7 Discussion

7.1 Advantages of Blockchain

The major advantage of a file storage system built on top of a block chain is that trust is guaranteed. There is no central authority so users' files cannot be modified or lost as if stored in a centralized server.

In addition, downloading files from the block chain is safer than other traditional means since the integrity of the blockchain translates directly to the integrity of the file. A user reading files from a trustworthy node connected to the blockchain doesn't need additional steps to verify the authenticity of the file.

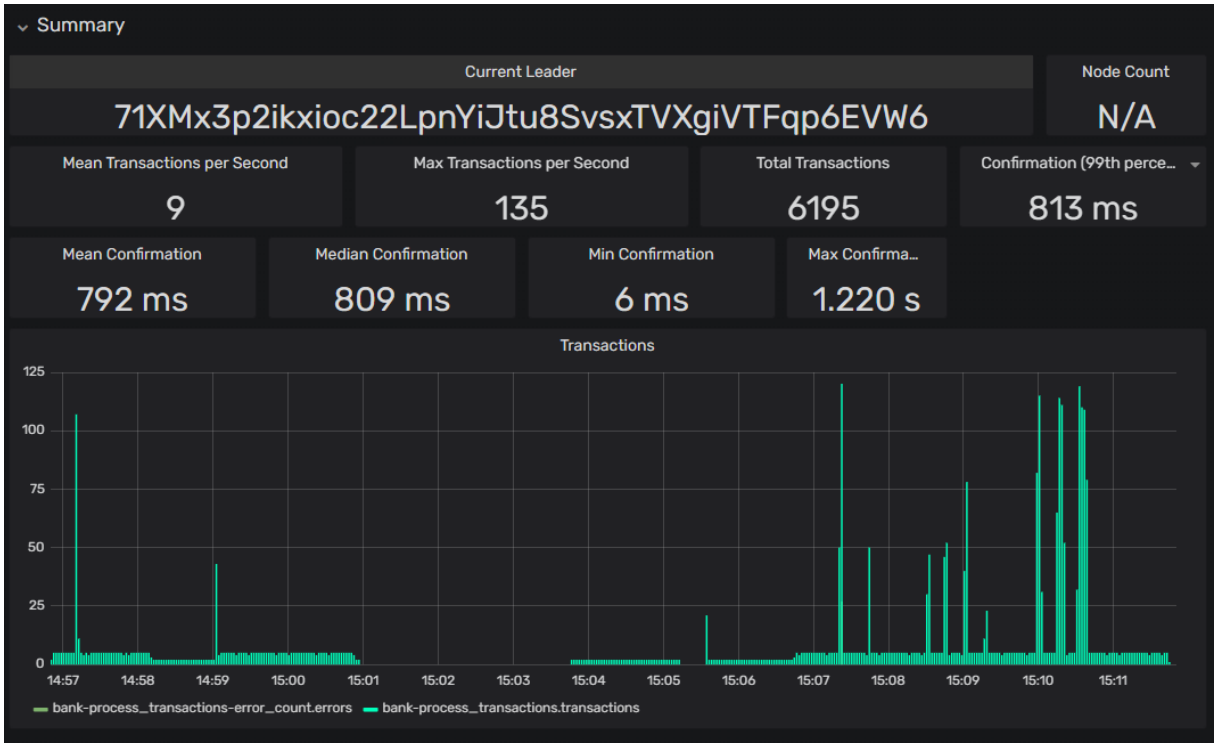

Figure 3: Solana Dashboard Metrics

7.2 Disadvantages of Blockchain

Latency is perhaps one of the largest tradeoffs made by using a blockchain based system. In our experiment, uploading and downloading a file doesn't scale very well for large files but small files ($<$ 1MB) will take less than a second to download.

Furthermore, such a system cannot afford to store very large files when there are a lot of users. Compared to traditional file storage services like Google Drive, our system has too much runtime and memory overhead to store data. Ensuring each transaction is valid for every 1KB of data transfer is too much a price to pay. In addition, files stored on-chain are propagated to every node in the system. This default replicating protocol is not efficient when it comes to large file storage.

In addition, using this blockchain model means there is an associated transaction fee to incentivize validator nodes to verify our transactions. However, this means that using our service to handle many files would probably be very costly to upload and download often.

8 Future Works

8.1 Blockchain Design

We believe there are several optimizations that could be made in our smart contract but our performance so far seems to be limited by the Solana blockchain design. The design of the system generally has one leader validator followed by many others in the same cluster. The leader needs to broadcast the same ordering to all nodes in its cluster to maintain a total ordering but it hurts performance. We haven't measured what the impact of changing this leader follower system to another will be.

Another blockchain, Avalanche, uses a modified BFT algorithm which is leaderless (Rocket et al., 2019). Their system can achieve high TPS and is quite scalable as well. It will be interesting to see if there is a blockchain that can utilize the speed of Proof-of-History and Avalanche's modified BFT algorithm.

In addition, Solana has imposed some constraints on the amount of data being transferred. If we can test another blockchain system which doesn't have this constraints, we may be able to make our transactions more efficient by grouping small transactions to reduce transaction overhead.

8.2 Robustness

Solana has had blockchain outages in the past. These outages are part of the growing pains of the blockchain but for particularly a distributed file storage service, it may be better to build out one's

own blockchain or P2P service. However, for a P2P service, there isn't really a monetary incentive to run one's own node.

8.3 Smart Contract Design

Another optimization may be to rewrite our smart contract design. Currently, the issue is that we have to wait for the blockchain to validate the transactions so we know the entire file has been uploaded. If it's possible to design our smart contract in such a way that it does not need to entirely depend on encoding data in the blockchain but rather perhaps sharded and stored locally on many nodes, then it is possible this could scale out more efficiently and effectively. Another idea is to send batch transactions to reduce cost if possible. This way there can be less reliance or cost needed to upload / download files.

We also designed this smart contract just as a proof of concept to test performance. This could be more securely designed by figuring out a way to hash the data that is stored in the data blocks on the blockchain or just hashed and stored on a node's local client.

9 Conclusion

A blockchain based file system may be useful in the sense that you don't need to trust a central authority to not leak or overwrite your files. However, in other cases, this system does not make so much sense because of the loss of performance when you scale. We think a different design of our smart contract would prove to be more scalable.

Furthermore, Solana is touted to reach tens of thousands of transactions per second. While we didn't stress test this due to our limited hardware, we noticed that in the Solana whitepaper, the transactions have to go through a leader validator. As the network grows, we believe this will be a pain point for the system. Solana's Proof-of-History is very fast but its perhaps by design that the reliance of this leader validator may be a bottleneck.

References

Soo Hoon Maeng, Meryam Essaid, and Hong Taek Ju. 2020. Analysis of ethereum network properties and behavior of influential nodes. In *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 203–207.

Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system.

Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. 2019. Scalable and probabilistic leaderless BFT consensus through metastability. *CoRR*, abs/1906.08936.

Anatoly Yakovenko. 2018. Solana: A new architecture for a high performance blockchain v0.8.13.