Objective

: design a database which is horizontally scalable and decentralized

Properties

:

- Horizontally scalable

. The database scales by adding more nodes, which incurs linear cost.

- Decentralized

. The database architecture is trust-minimized. The node operators cannot “mint money”.

Design.

Let’s build a traditional distributed database. The database system is composed of a master node and a cluster of tablet nodes (a simplification of BigTable’s architecture). The database is sharded, whereby each tablet node is responsible for a slice of the database. To add more storage capacity, you add more tablet nodes.

Users send INSERT

and SELECT

tranasctions to the master, who routes them to the tablet nodes.

This isn’t decentralized, because the computation is not verified. If this database tracked money, a tablet node could easily print money by modifying their balance. We need to verify the computations.

The problem with scaling blockchains is that verifying transactions is O(n)

cost (where n

is computational steps, ie. gas

). It’s prohibitive. We can’t expect the master node to run every transaction themselves to verify.

However, what if we used a ZK-VM? The cost of proving a ZK-VM’s execution is polylogarithmic, roughly O(log^2 n)

. This scales much better. Cairo is a production-ready ZK-VM based on ZK-STARKS, and there areother designs based on EVM too.

Imagine that the master node and tablet nodes both ran their database (sqlite.exe) atop a Cairo VM. Then we could verify the computation was done.

New design - the master node distributes work via messages to the tablet nodes. The tablet node does the work, and generates a proof, which it sends back to the master node.

1. What about state growth?

Well, Cairo program can employ nondeterministic programming. We don’t have to prove the entire database table shard in the computation, we only have to prove we’re modifying it according to some rules! Simply put - imagine the database shard was merklized, and running an INSERT

tx is proving the assertion that the row was inserted into the merkle tree correctly. Using a Sparse merkle tree, we can construct an ordered map of (index→item), which is efficient to prove adjacent leaves of.

1. What if a tablet node is byzantine and decides to “rollback” the transaction

? How do we ensure they only ever advance?

Simple, we make every node a Cairo blockchain. Instead of messages, nodes communicate via transactions. Each transcation increments the node’s clock T+1, and naturally, references the previous block so that they form an immutable chain. The master node keeps track of every tablet node’s latest clock in their state too, which binds everything together.

Now we have a system where the tablet nodes are verifiably doing their job, the database state is sharded in a cryptographically authentifiable way, and the master node only incurs O(log^2 t)

cost to verify these things are happening.

The last and easy part is now decentralizing the master node. The master node is mostly a coordinator - it distributes work to

different tablet nodes and verifies they did their work. We can put this program on Cairo’s CPU. This is recursive proofing

in an async context - while the master node is generating its own proof of its computation, it is awaiting the tablet node’s proof.