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

When a blockchain has a fast finality, like 0chain, the state is rapidly changing. So it becomes critical to sync fast and be operational even with a partial state to contribute to the overall consensus.

The state management is an integral part of the fast finality claim together with the execution time of a smart contract. The production of a block involves executing smart contract logic associated with the transactions in the block that result in mutating the underlying DLT state that is securely verifiable.

So, when comparing blockchains and their finality claims, it's important to keep this full picture in mind, and not in isolation of being able to do a fast Byzantine consensus.

Sync is expensive today

In most blockchains using MPT, when a new node comes online, for it to participate in the blockchain process of generating or validating a block, it first needs to sync up the entire MPT. The MPT can be a very large data structure depending on the number of keys present running into several millions. If the blockchain is popular, such as Ethereum, it can easily go into billions over time. This makes it very time consuming to sync the entire tree.

One of the key observations of transactions on a blockchains is, like many other things, the 80-20 rule. That is, 80% of the blockchain is used by 20% of users and smart contracts (there is a good chance that these numbers are even more skewed). Hence, it doesn't make sense for someone to wait loading the entire state represented by MPT synching up data related to a long and growing tail of users and smart contracts that hardly transact and in the process impact the quality of service for the rest of the users. Ideally, if it's possible to make progress with a partial state that helps the nodes to start earning rewards as soon as possible.

But is it possible to support this given that the MPT updates are expected to be securely verifiable with each block produced on the blockchain? The answer, not surprisingly, turns out to be yes. The mathematical properties of the MPT provides a solution with proof of correctness.

0Chain's Fast Sync MPT (FS-MPT)

Let's first define some terminology. An application using MPT has an API to store a value with a given key, represented as (k,v) and storing several such values can be represented as (K,V) (the upper case just defines a set while the lower case represents a single element). In order to provide security proof, MPT has to store some intermediate nodes in addition to V and hence collectively represented as (N). Further, the application keys (K) end up representing the paths in the MPT to access the values, while hashes of the nodes, H, are used as keys to the Nodes. Hence, MPT is a mapping (K,V) -> (H,N). The (H,N) pairs are persisted into a key, value store (such as rocksdb used by 0Chain). Note that the pair (H,N) is self validating because H = hash(N).

Say at a given blockchain round, x, the MPT is represented as (Hx, Nx). Subsequently the MPT got modified as the blockchain progressed. For a given block, let's say the state got updated (inserted/deleted) to a small set of keys and values, (Kb, Vb). The new MPT after applying the changes (Kb, Vb), contains a subset of (Hx ,Nx) intact and a new set of hashes and nodes (dHb, dNb) together representing the new state, (Hxb, Nxb). The key observation here is, in order to compute and verify the new state, any partial state (Hxp, Nxp) which is a subset of (Hx, Nx) is sufficient so long as it is possible to compute (dHb, dNb) using this partial state. When the new state is so computed from the partial state, the root hash will be in agreement with that computed with the entire state. This is very important and coupled with the commutative property, it is possible to keep operating on the partial states while simultaneously synching the missing state, (Hm, Nm), block by block or ad-hoc and applying it (as in just storing the new hash, node pairs) to the existing state. The order in which these delta changes are applied doesn't matter.

0Chain blockchain uses the above concept to support the ability to make progress with the blockchain (to generate a block, validate a block, respond to client queries for balances, or other state related queries). While it is good to be able to make progress, eventually the entire state needs to be synched which can happen in parallel. The state sync itself happens via two separate partial sync strategies mentioned below.

1. Block level Partial State Sync

2. That is, applying transactions of a block to an initial state as mentioned above results in (dHb, dNb) delta state changes. This delta change includes the root key and node and since the root of the state MPT is part of the block hash, it is possible to verify that a given delta state matches with a given block. As a result, when a node is making progress with the blockchain and temporarily unable to validate the state of a block, it can request for the block specific partial state, validate that his partial state is correct and start using it. This helps with any temporary downtime such as a short network outage and allows updating the partial state to full state without having to replay the transactions in all the intermediate blocks to compute the state. This makes the temporary state sync process very fast.

3. Missing State Sync

4. As mentioned, MPT is a versioned data structure by default and hence it accumulates a lot of (H, N) tuples. These need to be pruned to contain the size of the state db. 0Chain has a state pruning logic that is similar to a garbage collection of Mark and Sweep logic. During this process, it is possible to identify the set of missing state (Hm, Nm).

When missing state is detected during the “marking” process, it will sync that state (randomly asking peers for Nm related to a set of hashes, Hm) and abandon the “sweeping”. This can continue several times as discovering some missing nodes can lead to discovering additional missing nodes that are children of the original missing nodes. Eventually when the entire state is synced, the older state that is no longer required is pruned. Note that the “marking” process traverses the tree while the “sweeping” process just traverses the underlying database. As a result the “marking” process can be expensive and 0chain has a configuration to specify the frequency at which this process runs. However, when the state is missing, this configuration is ignored to ensure the state is synched as fast as possible without waiting for the next pruning cycle. It is likely that the “marking” process can be further optimized by storing the nodes of each level in a separate database or column family to reduce random I/O, something that can be explored in the future.

MPT is used for each smart contract to maintain its own state and the root hashes of each of these are stored as values in the global state that maintains state of the client wallets and smart contracts. This is similar to how Ethereum manages it’s state. Each MPT can be independently synched and it will be possible to prioritize synching some smart contract states over the other based on popularity for example.

The FS-MPT approach of state pruning and synching provides 0chain with a robust state management. The ability to sync up partial state changes block by block provides the benefit of synching state faster for frequent users and smart contracts (and hence get ready to support them with their subsequent transactions), while eventually syncing the entire state by discovering the missing state in the background. This improves the stability and quality of service of the blockchain.

@vbuterin would Ethereum 2.0 be able to use this construct to enable miners do a fast sync?