To make sure every node in the network agrees on the same result of transaction execution, a lot of blockchains such as Ethereum, Cosmos Hub, include the hash of the state trie, where the state is represented as a key-value map. After a transaction is executed and parts of the key-value pairs in the state are updated, the state trie is also updated, and the hash is re-calculated and included in the block header. As long as all nodes agree on the same hash, we ensure that all the nodes also agree on exactly the same state.

However, reading and writing the state trie can be expensive - a read will traverse the internal nodes of the path to the key and a write will update several internal nodes of the path to the key-value pair. Consider a trie with depth 5, a trie read may perform 5 underlying DB reads, and a trie write may perform 5 underlying DB writes besides 5 DB reads. When the number of entries in the state trie is large, the executing of a transaction can be slow, and thus system throughput will be lowered.

The proposed fast tx execution without state trie uses a traditional key-value DB to represent the state. To make sure every node agrees on the same tx result, instead of including the hash of the state in each block, we include the hash of the updates of the state, i.e., a list of DB write and delete operations, after performing all the transactions in the block. As a result, a read from the state always takes one DB read, and a write to the state always takes one DB write - no matter how many entries are in the state. In addition, as long as we ensure the hash of the updates (deltas) are the same for every node, we could ensure every node will have the same state of the network.

However, using the hash of updates instead of the hash of the state creates several questions:

- If the chain is re-organized, how to recover the state of a previous block? A way is to undo the transactions like Bitcoin does, or we could resort to the underlying DB snapshot feature, which is very cheap for most DBs such as leveldb.
- How to quickly synchronize a node? A quick sync will only download the block headers and then the state of a close-to-latest block. It could use the state trie hash to verify the correctness of the state of the block downloaded from another untrusted client. Without state trie hash, a solution may require every node to periodically create a snapshot block (maybe every 2 weeks or 80000 blocks) - a normal block containing the hash of the state of the previous snapshot block (likely using a trie). This means that a client could use 2 weeks to re-calculate the hash of the state trie instead of every block. As a result, a quick sync can be done by obtaining the hash of the latest snapshotted state, downloading the state, and verifying it. After the quick sync is done, the node could download the remaining blocks and replay them to obtain the latest state.
- How to lightly check if a key-value pair is in the latest state? Given the hash of a state trie, a user could quickly check the existence of a key-value pair (e.g., such as the balance of an account) by querying the cryptographic proof of the inclusion of the key-value pair (the paths to the key-value pair in the state trie). Without state trie in each block, we could check by using the most recent snapshot and its hash, which can be out-dated. To obtain the latest result, there are a couple of ways. One way is that a user may need to replay the remaining blocks, or a mini trie (likely a sparse tree mostly in memory) that represents the latest updated key-value pairs since the latest snapshot is maintained and the hash is included in the header.