My team has been working on research to prove or disprove an approach to solving the “state availability” problem outlined here: https://notes.ethereum.org/5VVFi_8FRo2hSnTiLFB_6g

Overview of the approach

A rough sketch of the direction we are looking to take:

- The network is a DHT (likely built on discv5)
- Account and Contract storage data is stored in their individual trie nodes.
- Nodes in the network are aware of the header chain.
- New trie data from each block is pushed into the network in the form of proofs.

We refer to this approach as the GetNodeData

approach because it is similar to how existing fast

sync works when retrieving state.

Trie Node vs Leaf+Proof storage

We have chosen to focus on storging data by individual trie node because it is simple.

The alternate approach is to store only the leaf values with accompanying proofs. This approach incurs complexity since the proofs need to be continually updated. Updating the proofs can be done locally at the cost of EVM computation and distribution of full block witnesses. EVM computation is expensive and full block witnesses are large.

By storing individual trie nodes, the only work that network nodes need to do is store the data, and verify merkle proofs for new data as it arrives.

Findings Thus Far

Expected Latency

Based on experience with the DiscV5 DHT, we expect to see network lookup times to be around 100ms.

Trie Nodes per Transaction

Nick Gheorghita has been researching the number of trie nodes touched by common transaction types. Preliminary results from a small sample size:

- Simple value transfer: ~30 trie nodes
- ERC20 transfer/approve: ~50 trie nodes

At 100ms latency, that suggests an upper bound of 3 seconds and 5 seconds respectively to do eth_estimateGas/eth_call

. These numbers can be reduced with basic optimizations like concurrently looking up sender and recipient of the transaction.

A more in depth experiment is underway to perform this measurement on a large block of mainnet transactions.

Garbage Collection & Cold State

Brian Cloutier has done some investigative work on cold state access patterns.

We define cold state here in the Glossary

Brian can clarify details, but the high level findings are that most blocks touch state that has not been touched for 1 million blocks.

This has interplay with garbage collection.

- If

the network has sufficient capacity to store the full archive state then we do not need garbage collection.

- if

the network has insufficient

capacity to store the full archive state then the network must

implement some mechanism to prevent cold state from being lost from the network.

Open Research Questions

Deduplication and Garbage Collection

Two contracts with identical storage tries will have identical trie nodes.

Similarly, two accounts with identical balance, nonce, code, and state will have identical leaf nodes storing their account data. If nodes are stored using their node hash as the key, then reference counting is required to be able to implement garbage collection, otherwise you cannot know that a node that was removed from one location in the trie is not being used in another trie.

One way to address this is to key nodes by both their location in the trie and their node hash. This would allow deletion of nodes using exclusion proofs while incuring the extra cost of redundant storage for duplicate data.

The open question is how much this increases the storage requirements.

Archive vs Garbage Collection

We need to figure out one of the ideas for garbage collection or

validate that the network can be an archive node.

Options for addressing garbage collection

1. Remove the de-duplication and use (trie_path, node_hash)

as the key for looking up data.

1. Monitor the network and actively re-add cold state
2. Figure out if there is a subset of garbage collection that can occur on just the intermediate trie nodes in the account trie.
3. Ensure the network has the capacity to operate as an archive node.

Data Ingress

We need to push newly create trie data into the network. Nodes in the network can be expected to have an up-to-date picture of the header chain, allowing them to anchor proofs against recent state roots.

Open questions are:

1. How big are full block proofs of only the new trie data.
2. How big are individual proofs for each individual node in the block proof.