During the 1.x workshop in Paris last weekend, a couple ideas have been discussed for the transition from hexary tries to binary tries. It has been agreed that each proposal should be published on ethresearch for comparison and further discussion.

TL;DR

A hexary to binary conversion method in which new values are stored directly in a binary tree sitting “on top” of the hexary, while the “historical” hexary tree is converted in the background. When the process is finished, both layers are merged.

Snapshotter

In the present proposal, I am relying on an up-and-coming feature of geth

, the snapshotter. I won’t go too much in the details of the snapshotter here, just the general principle:

- it uses series of diff layers to store recent modification,
- an on-disk, storage layer contains a more ancient, stable version of the tree and is used as the base of the diff

This method promises a lot of improvements for speed and sync, including the method proposed here to switch between hexary and binary trees.

The binary tree overlay

The base idea consists in laying a binary tree “over” the hexary tree. The hexary tree is then declared read-only:

- all new values are therefore inserted into the binary tree,
- when reading a value, both trees are being searched in parallel:
- If a value is found in the binary tree, it is returned,
- If a value is found in the hexary tree and not in the binary tree, it is returned,
- If a value is found in the binary tree, it is returned,
- If a value is found in the hexary tree and not in the binary tree, it is returned,
- a background process converts the base layer from hexary to binary
- when the base layer conversion has been achieved, the miner signals that they are ready for the switch by publishing the new root in the block header. When enough (N

) miners have reported the same conversion, that trie is considered the new base layer and the top layer can then be merged.

Transition process

In the subsequent paragraph, root^{2}

represents the root of a binary tree, and root^{16}

that of a hexary tree.

[

statemachine

385×636 19.9 KB

](https://ethresear.ch/uploads/default/original/2X/a/a6752fc77965717e886135acc0c8d6288a1247e1.png)

The transition goes through 4 phases:

1. The block format is unchanged until block number n_{activation}
2. At block number n_{activation}

, blocks contain two state roots: the root root_{base}^{16}

of the hexary base layer and the root root_{overlay}^{2}

of the overlay binary tree. Participants have N

blocks to convert the base layer from hexary to binary in a background process. During those N

blocks, the value of root_{base}

remains the value of the hexary state root at block number n_{activation}-1

(that is, root_{base}^{16}

), and any other value will be refused by honest nodes.

1. At block number n_{activation}+N

, the value of root_{base}

becomes that of the translated base tree to binary, i.e. root_{base}^{2}

. This block format is used until N

consecutive blocks have provided the same value for root_{base}^{2}

. A background process starts to insert elements represented by root_{overlay}^{2}

into the tree with root root_{base}^{2}

. This is done in the background and the block should keep reporting an identical value for root_{base}

. At this stage, a block can only be considered valid if they provide either the hexary value or the binary value previously seen. Any other value has to cause a fork at block number n_{activation}+N

.

1. Once N

consecutive blocks have occured, the two layers are merged and only root_{base}^{2}

is present in subsequent blocks. From this point on, the consensus rules are the same as during stage 1, except that the state tree is now binary.

Required changes

- state roots

: during the transition period, blocks should have two state roots: that of the hexary tree (which should not change over time) and that of the binary tree. When the conversion of the base layer has happened, the hexary root is replaced with the result of the binary conversion. After N

blocks have had the same root, the hexary root is deprecated in favor of the binary root. After another N

blocks, the two binary layers are merged and the top state root is discarded.

- consensus

: both state roots have to be considered by the consensus algorithm

I believe that this can be done in one hard fork, and some people have expressed reservations about this. I'm looking forward to discussing this point.

Key metrics

Here are key metrics that a prototype should be able to measure in order to demonstrate the feasibility of this concept

- Memory usage: it depends on the conversion process
- Disk space: it is estimated to require around 30 extra GB during the transition
- CPU usage: this model proposes a conversion that is done by miners and full node to the benefit of the entire ecosystem, and is very costly to these actors. Computation costs could be shared with the community by a gas price increase during the process.

Open questions

- Should we delete values present in both trees?

This could reset the calculation of the hash, and therefore create an attack vector that would forever stall the conversion. It would also simplify the final hash calculation.

- Symmetrically, should reads returning values from the hexary tree that are not present in the binary tree be inserted there?

This adds extra calculations which should cause a gas price increase. This would make the final merge easier, though.

- How long will this realistically last?

Current estimates range from between two weeks to a month, testing will answer this question more precisely.

- Who is going to pay for it?

Presumably the miners, if an increase in gas price can offset the extra costs. This has the advantage that, for a couple weeks, the price of gas will be more expensive and therefore the amount of transactions lower, which will in return help reduce the increase in required resources.

Thanks

This has benefited from the valuable feeback from Sina Mahmoodi and Tomasz Stanczak.