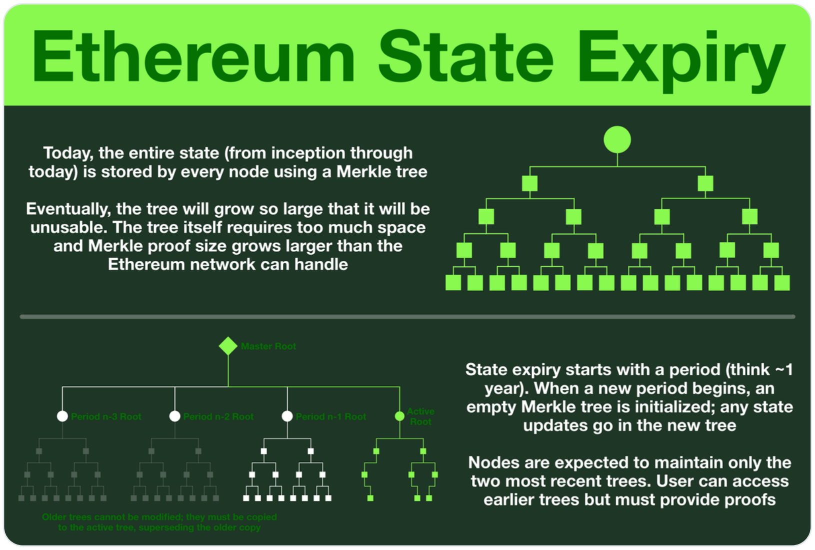
Haym @SalomonCrypto

Nov 3 • 21 tweets • SalomonCrypto/status/1588302097413332992

(1/20) @ethereum Roadmap: State Expiry

As the World Computer attracts more use, the amount of resources needed to keep itself running increases. Today, that burden will increase to infinity, eventually crushing the network.

A roadmap to a sustainable Ethereum.



Ethereum State Expiry

Today, the entire state (from inception through today) is stored by every node using a Merkle tree

Eventually, the tree will grow so large that it will be unusable. The tree itself requires too much space and Merkle proof size grows larger than the Ethereum network can handle

State expiry starts with a period (think ~1 year). When a new period begins, an empty Merkle tree is initialized; any state updates go in the new tree

Nodes are expected to maintain only the two most recent trees. User can access earlier trees but must provide proofs

Older trees cannot be modified; they must be copied to the active tree, superseding the older copy

(2/20) @ethereum is the World Computer, a single, globally shared computing platform that exists in the space between a network of 1,000s of computers (nodes).

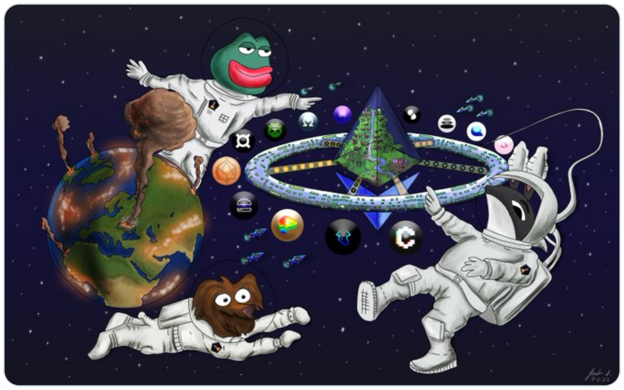
The nodes provide the hardware, the EVM provides the virtual computer and the blockchain records Ethereum's history.

Haym
@SalomonCrypto · **Follow**

(1/21) @ethereum: The Big Picture

From 1492 to 2022, the context, technology and vision of the World Computer. The complete, top-to-bottom case for $ETH.

An (unprecedented) mega-thread.



3:00 PM · Sep 3, 2022

Read the full conversation on Twitter

960 Reply Copy link

Read 45 replies

(3/20) The EVM sits at the center of @ethereum, providing a decentralized computing platform to the world.

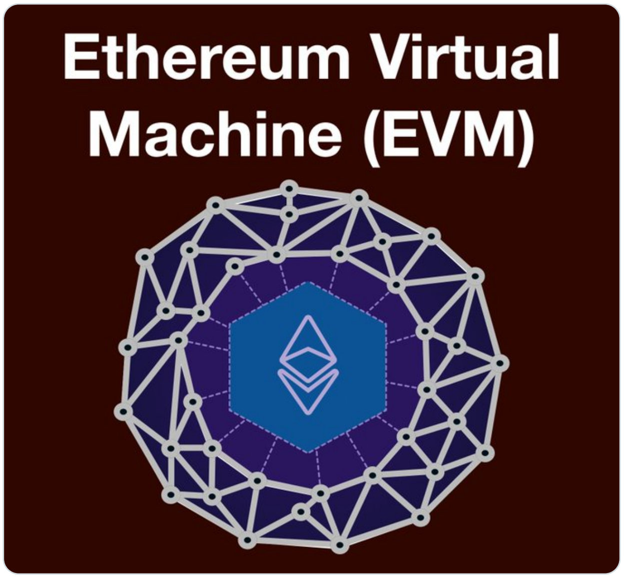
Everything is designed to construct this virtual machine and expose it to the world, so that anyone who is interested can permissionlessly interact with it.

Haym
@SalomonCrypto · **Follow**

(1/23) @ethereum Virtual Machine (EVM)

Ethereum is the World Computer, the future's internet-native global settlement layer. The EVM is the core of Ethereum; it provides the world in which settlement and decentralized computation happens.

Read on to learn about core $ETH tech!



4:33 AM · Sep 27, 2022

Read the full conversation on Twitter

378 Reply Copy link

Read 22 replies

(4/20) The internal state of the EVM is stored in a data structure known as a Merkle tree. Merkle trees are very powerful, but they are not perfect.

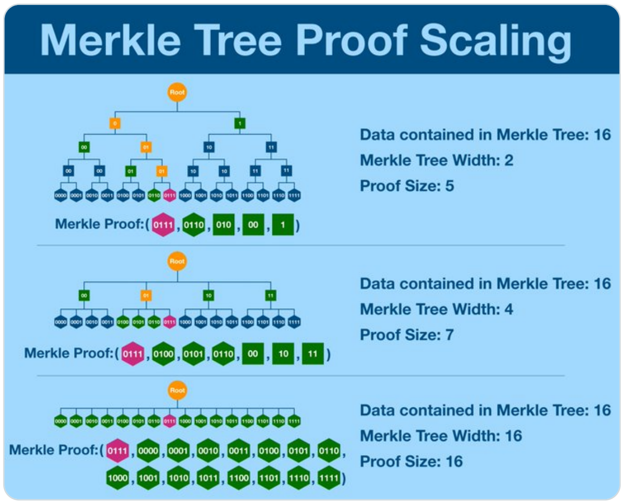
Specifically, they don't scale efficiently enough for the World Computer.

Haym
@SalomonCrypto · **Follow**

(1/18) The Problem with Merkle Trees

At the heart of @Bitcoin, @ethereum and many blockchain computers is the Merkle Tree. While this data structure has served us well, it is not perfect... and if you look ahead, you can see impending problems.

Let's talk Merkle proof scaling.


Merkle Tree Proof Scaling
Root
Merkle Proof:(0111 , 0110 , 010 , 00 , 1)
Data contained in Merkle Tree: 16
Merkle Tree Width: 2
Proof Size: 5
Merkle Proof:(0111 , 0100 , 0101 , 0110 , 00 , 10 , 11)
Data contained in Merkle Tree: 16
Merkle Tree Width: 4
Proof Size: 7
Merkle Proof:(0111 , 0000 , 0001 , 0010 , 0011 , 0100 , 0101 , 0110 , 1000 , 1001 , 1010 , 1011 , 1100 , 1101 , 1110 , 1111)
Data contained in Merkle Tree: 16
Merkle Tree Width: 16
Proof Size: 16

4:14 PM · Oct 29, 2022

Read the full conversation on Twitter

906 Reply Copy link

Read 53 replies

(5/20) Today, the World Computer stores the entire state of the EVM; every account, every entry, all the way back to genesis. For now, that amount of data is acceptable, but without intervention the size of the state will grow to infinity.

We'll kiss decentralization goodbye.

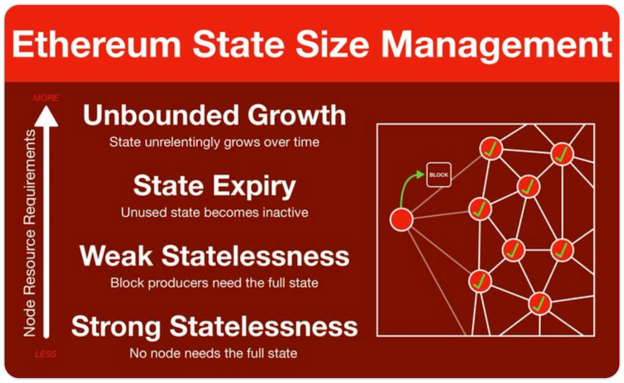
(6/20) Fortunately, the gigabrains developing @ethereum have been thinking ahead for a while now. And so, we have a framework out of which to build potential solutions.

In particular, we are interested in the concept of state expiry.

Haym
@SalomonCrypto · **Follow**

(1/22) As the World Computer gets more use, the size of the EVM's state grows. Unceasing, unrelenting, unending growth. Eventually, the individual computers that make up the network wont be able to keep up.

@VitalikButerin's theory of @ethereum State Size Management.


Ethereum State Size Management
MORE
Node Resource Requirements
LESS
Unbounded Growth
State unrelentingly grows over time
State Expiry
Unused state becomes inactive
Weak Statelessness
Block producers need the full state
Strong Statelessness
No node needs the full state
BLOCK

1:42 AM · Nov 3, 2022

Read the full conversation on Twitter

227 Reply Copy link

Read 8 replies

(7/20) With state expiry, parts of the state default to inactive and must be explicitly renewed.

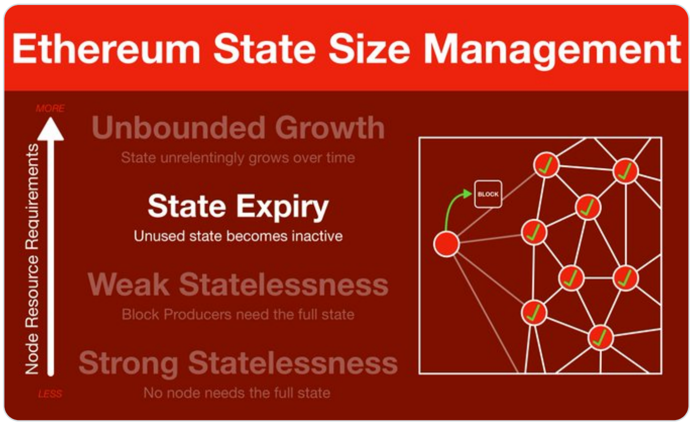
While there are many ideas around renewal, we are particularly interested in schemes that refresh via "touching;" simply accessing the state is enough to defer expiry.

Haym
@SalomonCrypto · **Follow**

Replying to @SalomonCrypto

(10/22) Instead of allowing the state of the EVM to grow into infinity, we could look at schemes that move inactive parts of the state into retirement.

State Expiry schemes do not look to delete data out of the EVM state; any inactive nodes can be cryptographically resurrected.


Ethereum State Size Management
MORE
Node Resource Requirements
LESS
Unbounded Growth
State unrelentingly grows over time
State Expiry
Unused state becomes inactive
Weak Statelessness
Block Producers need the full state
Strong Statelessness
No node needs the full state
BLOCK

1:42 AM · Nov 3, 2022

10 Reply Copy link

Read 1 reply

(8/20) State expiry keeps the size of the EVM's state at a manageable level; nodes will be allowed to offload stale data in order to make room for new objects.

Over time, the state size should remain (somewhat) stable, configurable by the scheme we decide to implement.

(9/20) In any (good) state expiry scheme also has a method to revive expired/inactive parts of the EVM state.

Think of it this way: each node holds the last X days of state, if you want to access further back, you must provide proof it existed in the state >X days ago.

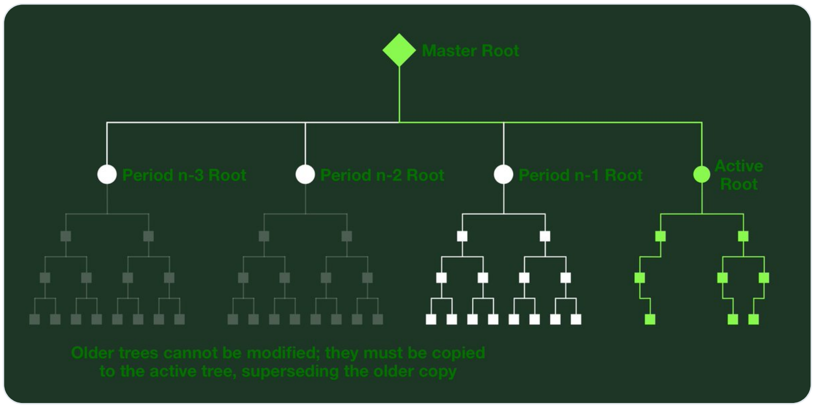
(10/20) Here's the state expiry scheme we are interested in. @VitalikButerin's proposal (below) has all the details.

(you'll notice talk about 32-byte addresses; don't worry we'll get there)

Resurrection-conflict-minimized state bounding, take 2
See also v1: Alternative bounded-state-friendly address scheme This document describes a scheme for how accounts and storage slots can be stored in a way that allows them to be pruned over time (see…
https://ethresear.ch/t/resurrection-conflict-minimized-state-bounding-take-2/8739

(11/20) Below is a graphical representation of the proposal. Each period (~1 year) @ethereum will archive the current state tree and will initialize a new state tree.

Archived trees cannot be modified; any changes to objects require first copying them over to the new state tree.


Master Root
Period n-3 Root
Period n-2 Root
Period n-1 Root
Active Root
Older trees cannot be modified; they must be copied to the active tree, superseding the older copy

(12/20) Each node is required to hold a certain number of archived trees (proposal: 1 archive tree plus active tree). When a period ends, a node is free to release older trees.

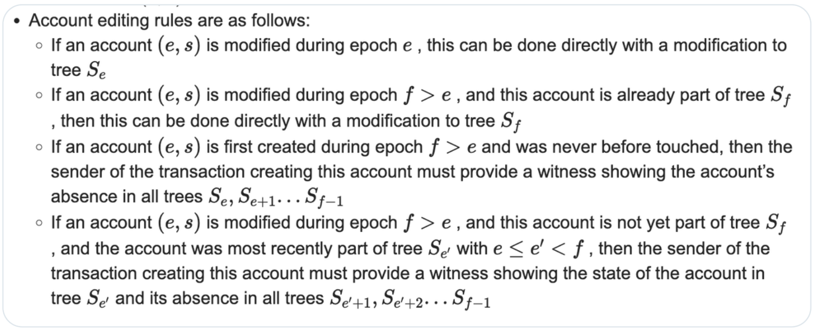
Thus, the EVM state size remains manageable; at most it will be two full trees covering 1 period each.

(13/20) While nodes (can) release the older archive trees, they keep the Merkle root in perpetuity (this is a single string, does not introduce scaling issues).

Reviving a piece of the inactive state is as simple as providing the Merkle proof for that old state.

(14/20) Ok in reality it's not quite so simple, but it's still straightforward.

The section below basically says "if the state is active, modify it. If it's inactive, you have to prove that it was part of the state, is now inactive, and was not accessed between now and then."



- Account editing rules are as follows:
  - If an account (e, s) is modified during epoch e, this can be done directly with a modification to tree S_e
  - If an account (e, s) is modified during epoch $f > e$, and this account is already part of tree S_f, then this can be done directly with a modification to tree S_f
  - If an account (e, s) is first created during epoch $f > e$ and was never before touched, then the sender of the transaction creating this account must provide a witness showing the account's absence in all trees $S_e, S_{e+1} \dots S_{f-1}$
  - If an account (e, s) is modified during epoch $f > e$, and this account is not yet part of tree S_f, and the account was most recently part of tree $S_{e'}$ with $e \leq e' < f$, then the sender of the transaction creating this account must provide a witness showing the state of the account in tree $S_{e'}$ and its absence in all trees $S_{e'+1}, S_{e'+2} \dots S_{f-1}$

(15/20) Interestingly, this state expiry scheme reframes our conception of statelessness.

At the network level, this isn't stateless as nodes still need to hold the most recent state. But for objects older than "most recent," we can access the state solely via Merkle proofs.

(16/20) Let's say we follow the proposal and allow nodes to release archive trees that are older than 1 period.

Well, some node operators might choose to store a few extra, just to optimize proof generation. And we could create state-archive-nodes which hold every tree.

(17/20) We can also go in the opposite direction. Again, assume the proposal's rule of 1 archive node.

Theres no reason you MUST access the state through the tree; you could verify using only Merkle proofs. Then the question becomes, why hold any of the trees at all?

(18/20) What's interesting about state expiry schemes is that they offer a way for different nodes to access different levels of statelessness based on their specific needs.

Gas costs (and the like) will be optimized for 1 archive tree, but the choice is the node operator's.

(19/20) State expiry is a very exciting and important area of @ethereum research. Bottom line, we are going to need to implement some sort of scheme.

But thing's are quite so easy. Picking a scheme is one thing, implementing it is a whole other animal.

(20/20) We are talking about changing the data structures that hold the state of the EVM, the molten core of @ethereum. It cannot be overstated how delicate and high stakes a task this will be.

Fortunately, we have some experience with these kinds of things.

Haym
@SalomonCrypto · Follow

Welcome to Proof of Stake

```
:42:53.002 INFO Ready for the merge             current_difficulty: 58750003716598352816469, terminal_total_
6:42:59.749 INFO New block received             root: 0x810a00400a80cdffc11ffdcf17ac404ac4dba215b95221955a9d
6:42:59.954 INFO

e: beacon
6:42:59.954 INFO Proof of Stake Activated       slot: 4700013, service: beacon
6:42:59.954 INFO                                Terminal POW Block Hash: 0x55b11b918355b1ef9c5db810302ebad0b
6:42:59.957 INFO                                Merge Transition Block Root: 0x810a00400a80cdffc11ffdcf17ac4
:42:59.957 INFO                                 Merge Transition Execution Hash: 0x56a9bb0302da44b8c0b3df54
```

7:03 AM · Sep 15, 2022

80 Reply Copy link

Read 1 reply

@ethereum More of a long-form reader? Try this out:

Haym
@SalomonCrypto

State Expiry

State Expiry | Haym

(1/20) @ethereum Roadmap: State Expiry As the World Computer attracts more use, the amount of resources needed to keep itself running increases. Today, that burden will increase to infinity, eventua…

https://typefully.com/SalomonCrypto/ynuVQEb

Like what you read? Help me spread the word by retweeting the thread (linked below).

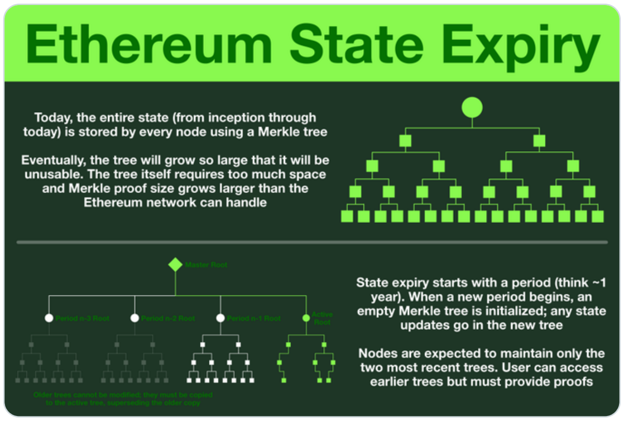
Follow me for more explainers and as much alpha as I can possibly serve.

Haym
@SalomonCrypto · Follow

(1/20) @ethereum Roadmap: State Expiry

As the World Computer attracts more use, the amount of resources needed to keep itself running increases. Today, that burden will increase to infinity, eventually crushing the network.

A roadmap to a sustainable Ethereum.



Ethereum State Expiry

Today, the entire state (from inception through today) is stored by every node using a Merkle tree

Eventually, the tree will grow so large that it will be unusable. The tree itself requires too much space and Merkle proof size grows larger than the Ethereum network can handle

State expiry starts with a period (think ~1 year). When a new period begins, an empty Merkle tree is initialized; any state updates go in the new tree

Nodes are expected to maintain only the two most recent trees. User can access earlier trees but must provide proofs

Master Root
Period n-3 Root
Period n-2 Root
Period n-1 Root
Active Root
Older trees cannot be modified; they must be copied to the active tree, superseding the older copy

10:48 PM · Nov 3, 2022

Read the full conversation on Twitter

4 Reply Copy link

Read 2 replies

• • •