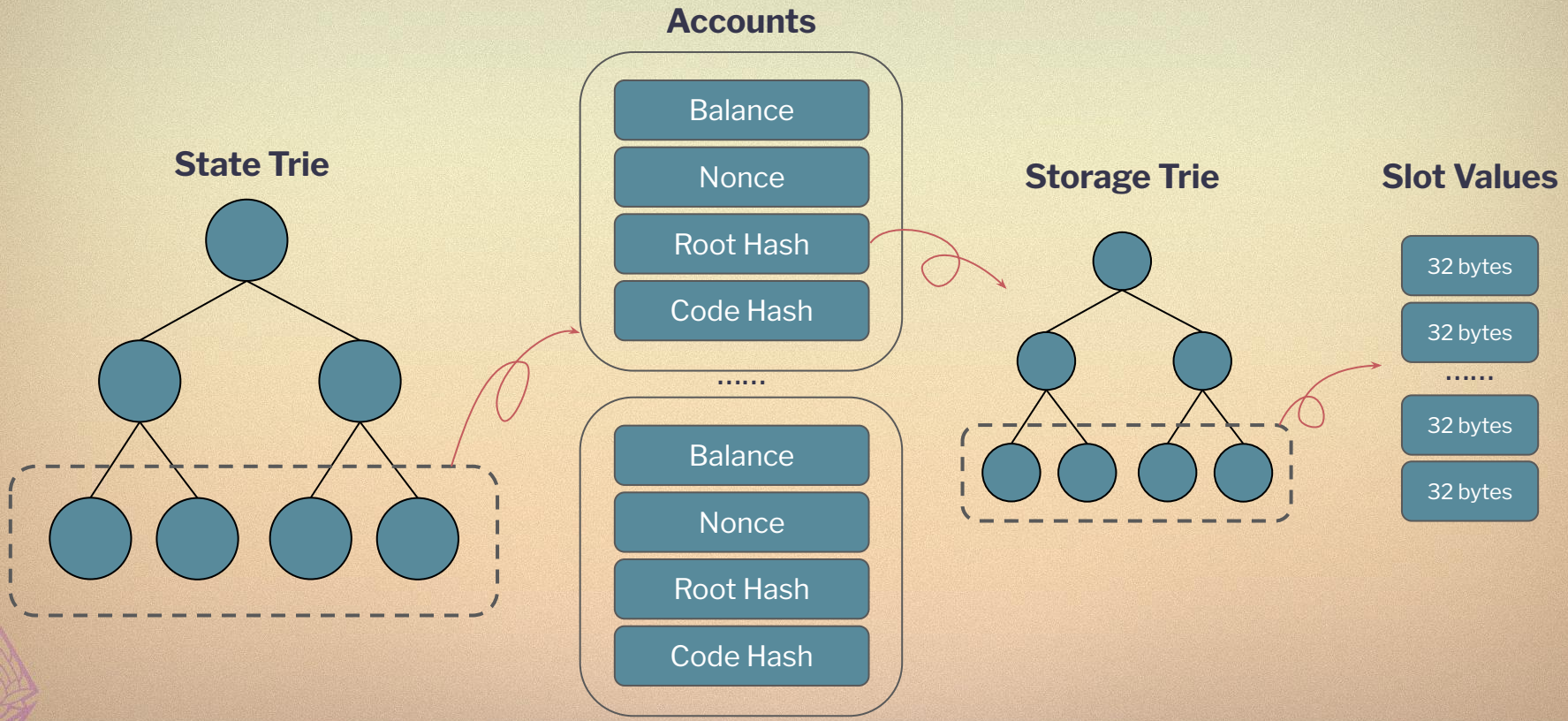
Everything you need to know about state expiry

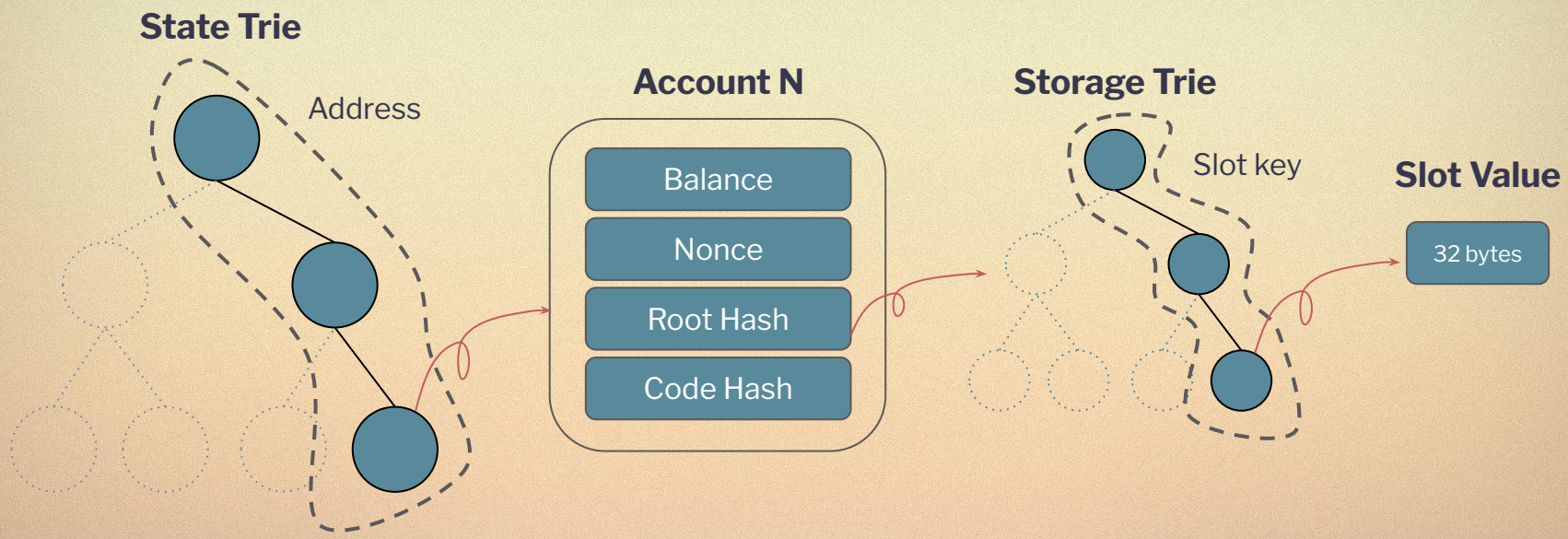
Han
Engineer @ Nethermind

STATE

State


Accounts
State Trie
Balance
Nonce
Root Hash
Code Hash
.....
Balance
Nonce
Root Hash
Code Hash
Storage Trie
Slot Values
32 bytes
32 bytes
.....
32 bytes
32 bytes

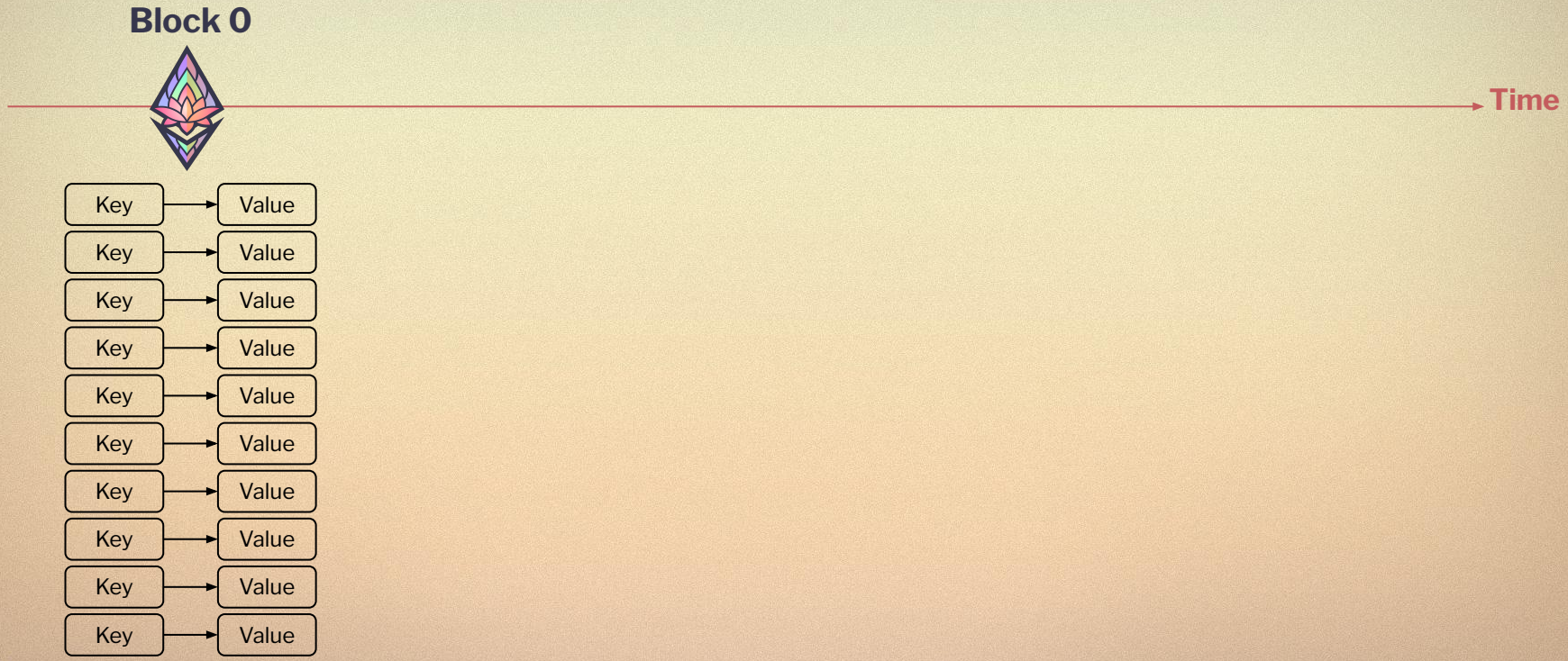
State


State Trie
Address
Account N
Balance
Nonce
Root Hash
Code Hash
Storage Trie
Slot key
Slot Value
32 bytes

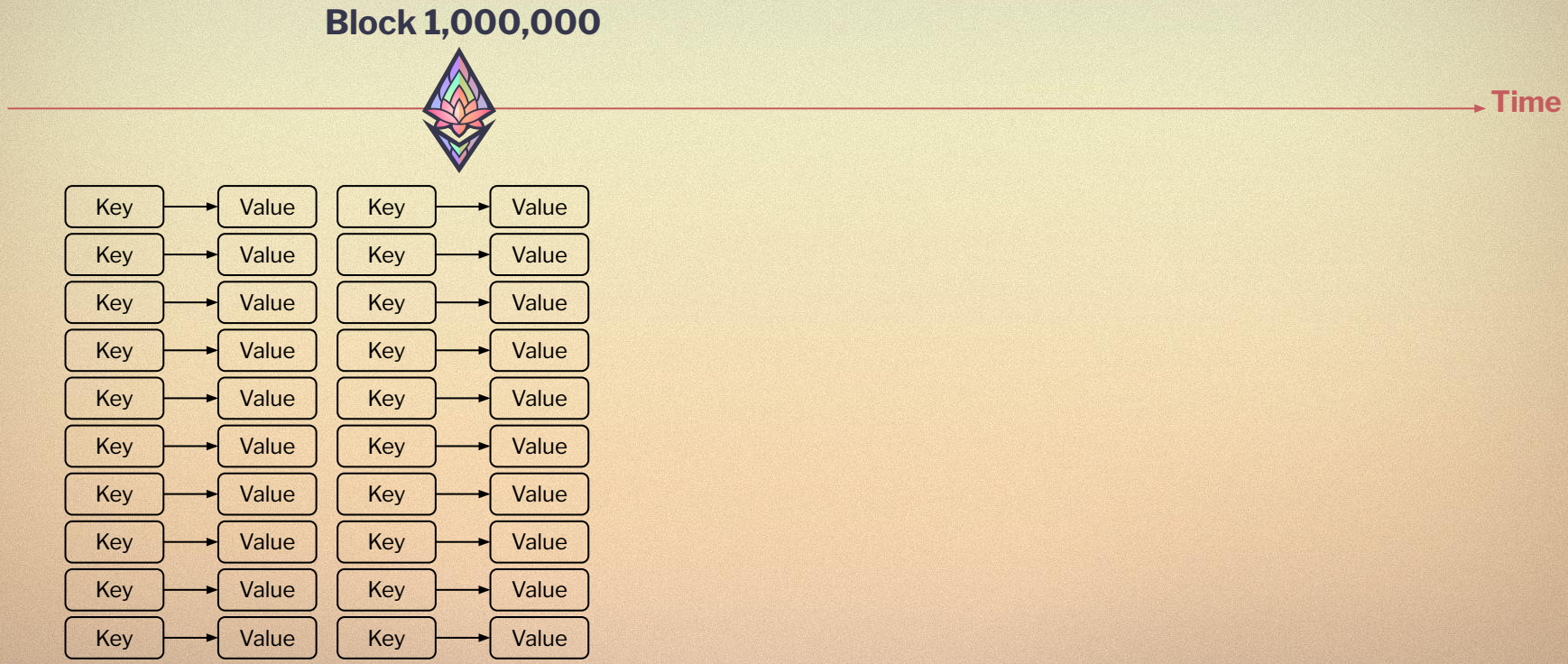
State

Key-value pair is the **basic unit** to measure state on Ethereum.

State


Block 0
Time
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value

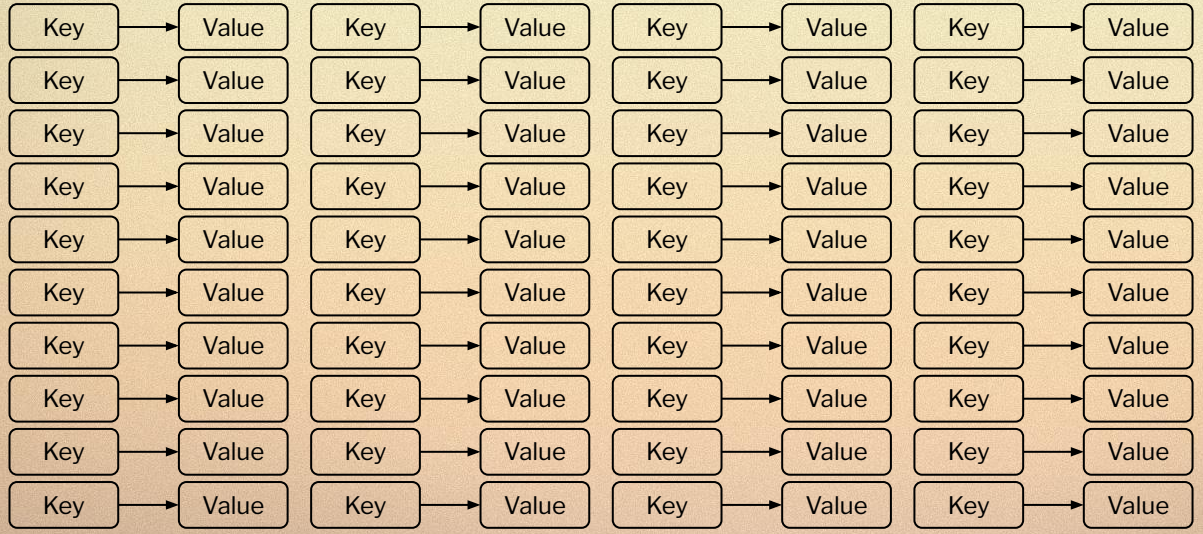
State



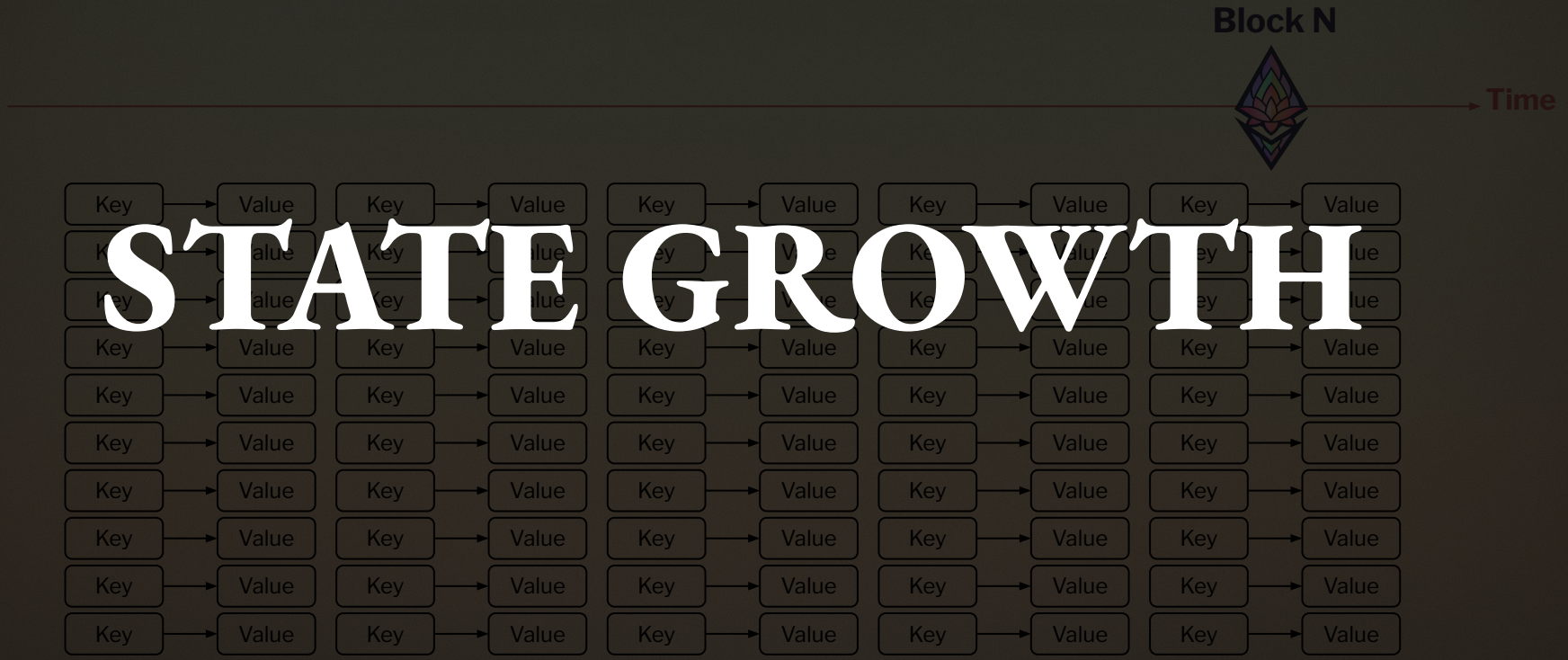
State

Block 10,000,000

Time



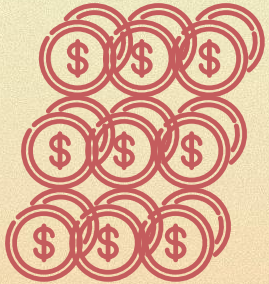
State



State Growth

State Growth

State grows



Running a node costs more

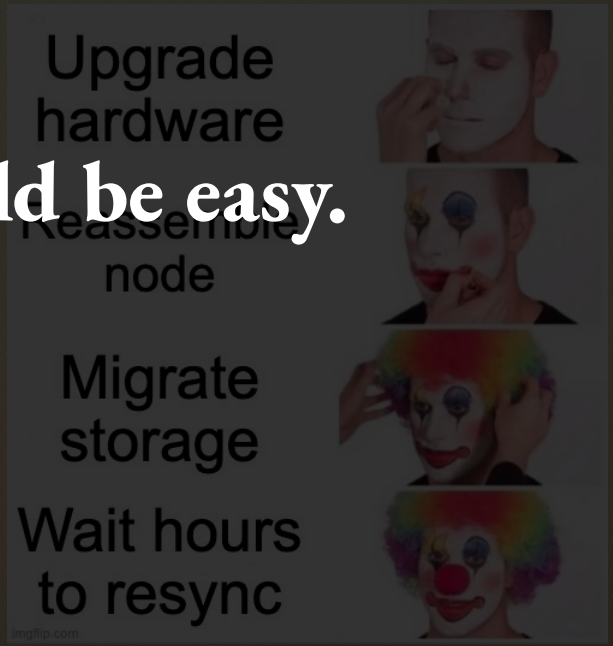
Less people run nodes

More centralized

State Growth

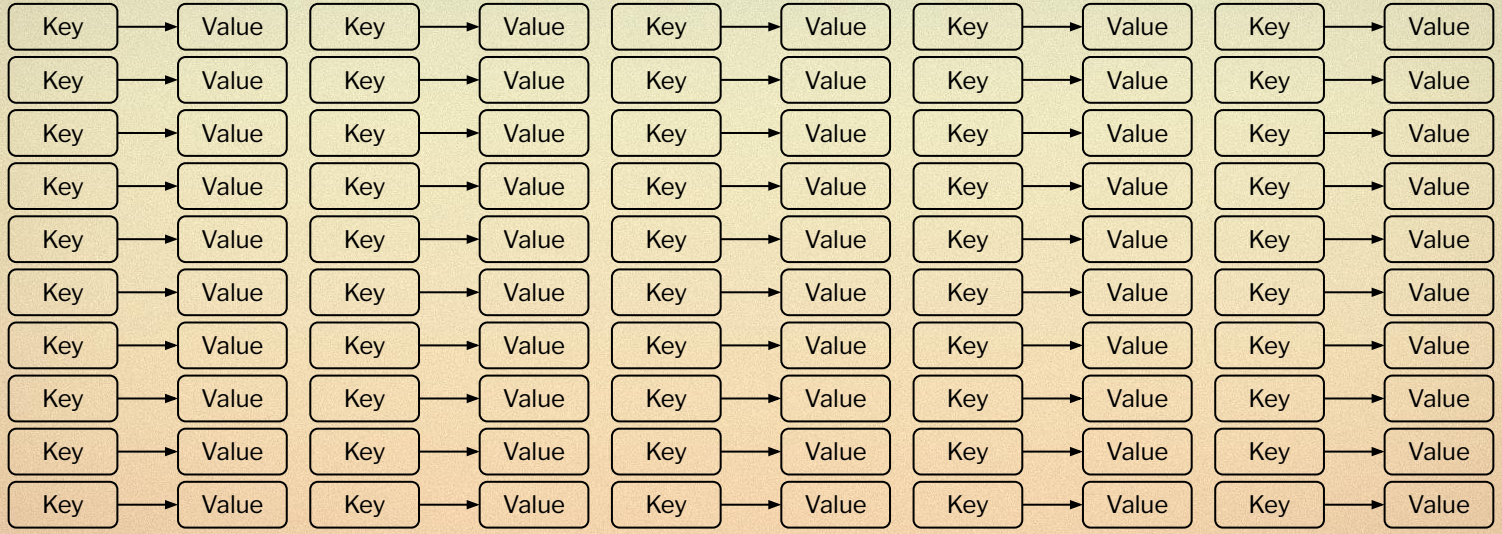
Expectation

Reality



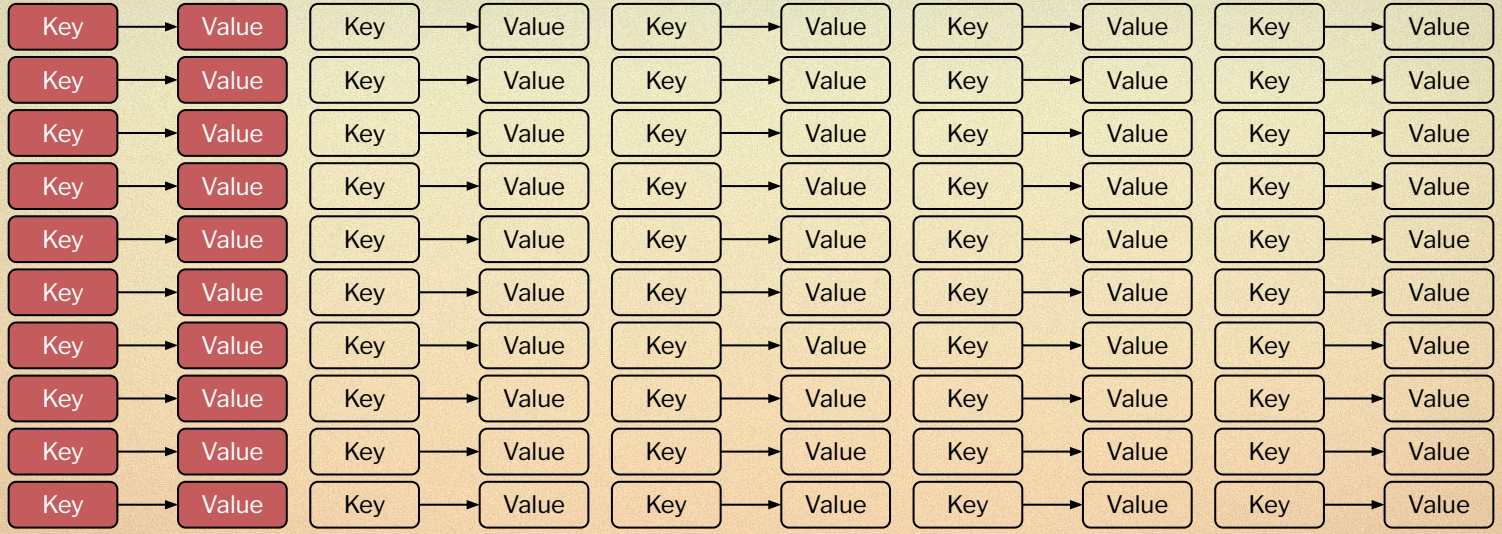
Running nodes should be easy.

State



How many key-value pairs were accessed in the past 1 year?

State



Only about **20%**.

**Analysis was done up to Block #14907811*

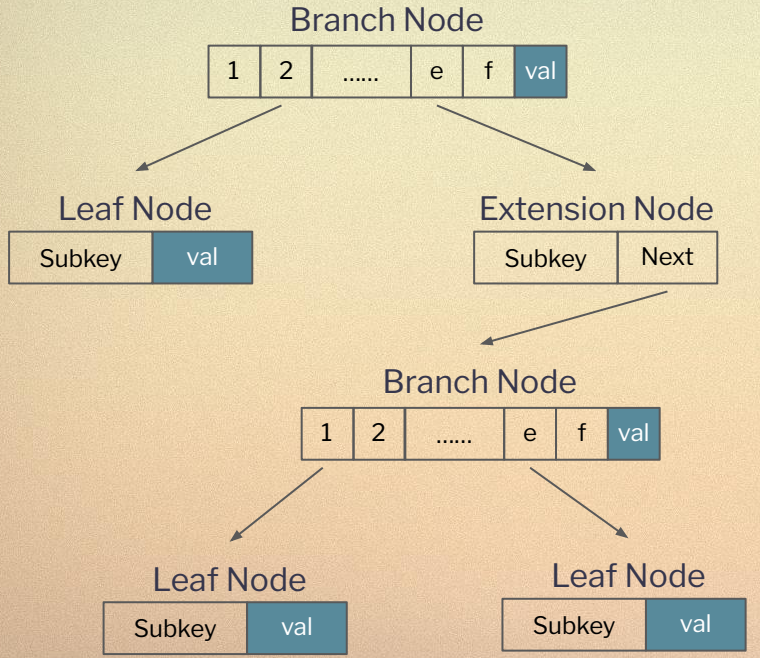
STATE EXPIRY

How do we know if a state is expired?

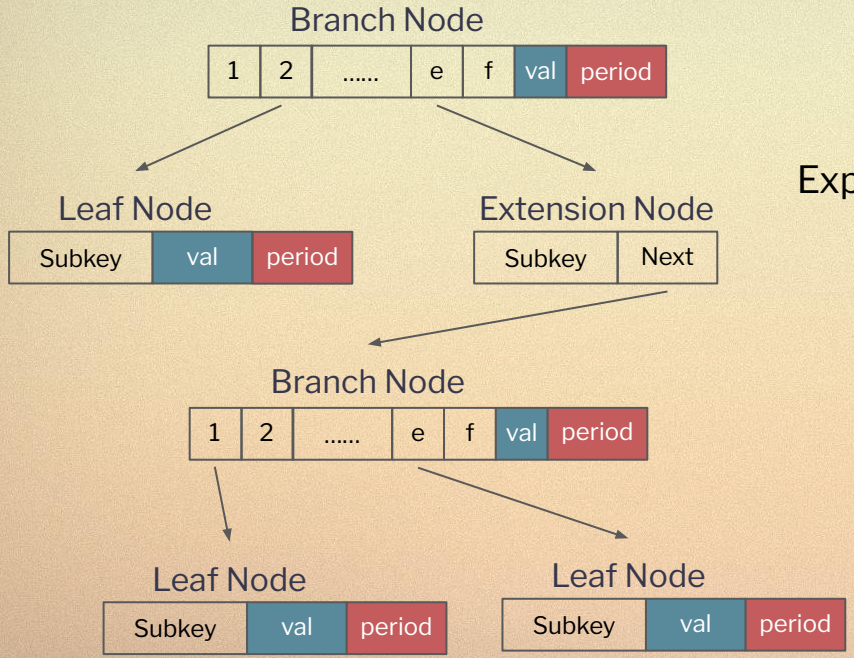
Expire now, revive later.

How to revive expired state?

State Expiry on MPT



State Expiry on MPT

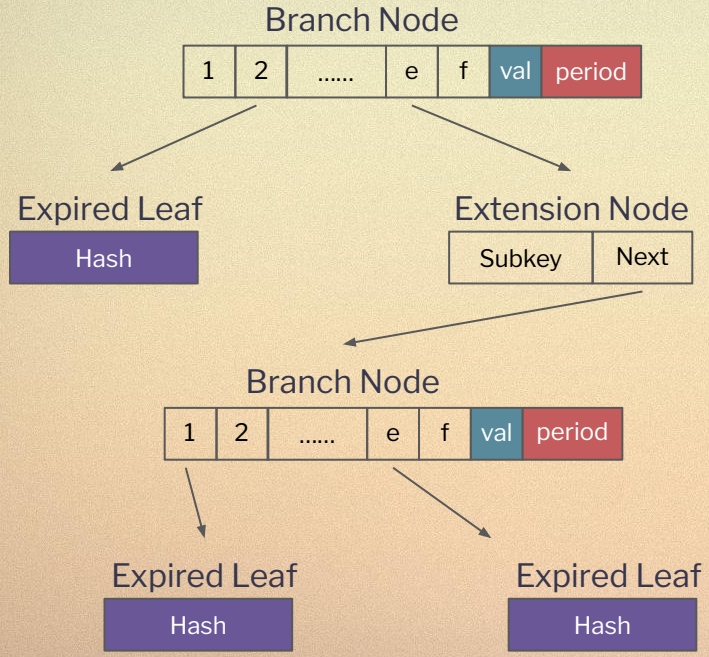


State Expiry

1 period = 15778800s (6 months)

Expires when value was last accessed 1 year ago

State Expiry on MPT



State Expiry

1 period = 15778800s (6 months)

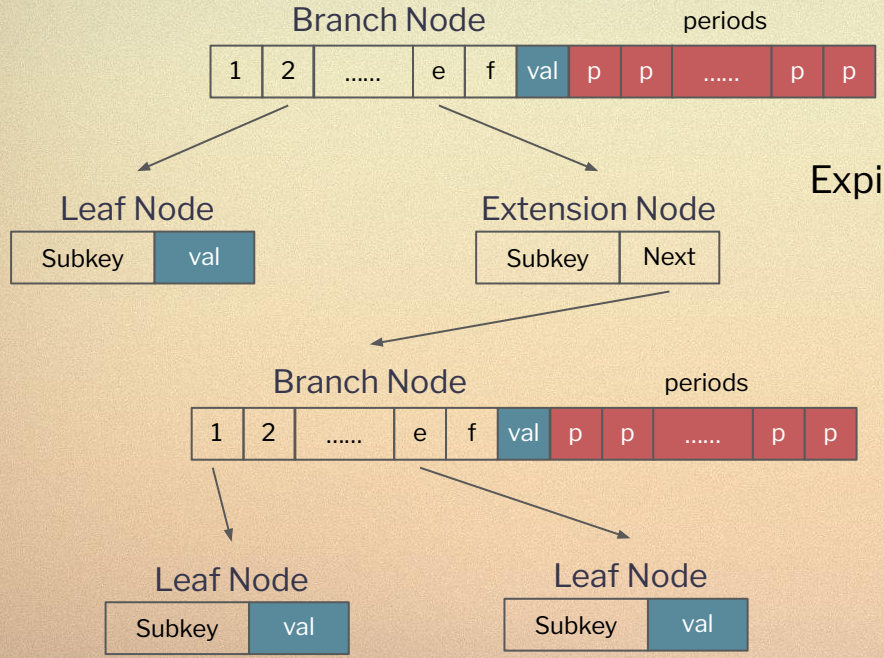
Expires when value was last accessed 1 year ago

State Revive

Provide **key** + **value** + **period**

We only save few bytes per leaf...

State Expiry on MPT

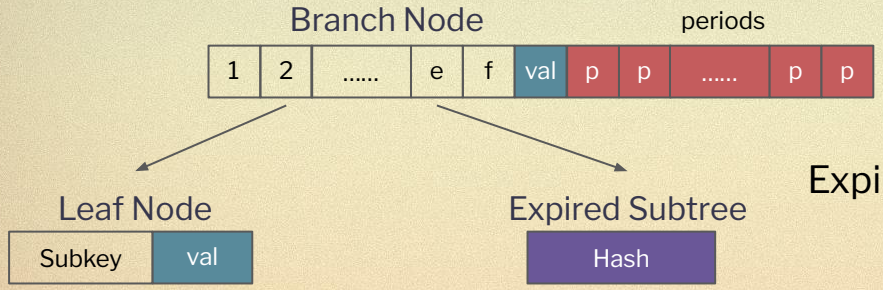


State Expiry

1 period = 15778800s (6 months)

Expires when child was last accessed 1 year ago

State Expiry on MPT



State Expiry ✓

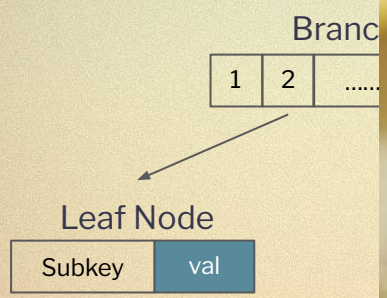
1 period = 15778800s (6 months)

Expires when child was last accessed 1 year ago

State Revive ✓

Provide **path** + **Merkle proof** of expired subtree

State Expiry



State Expiry ✓

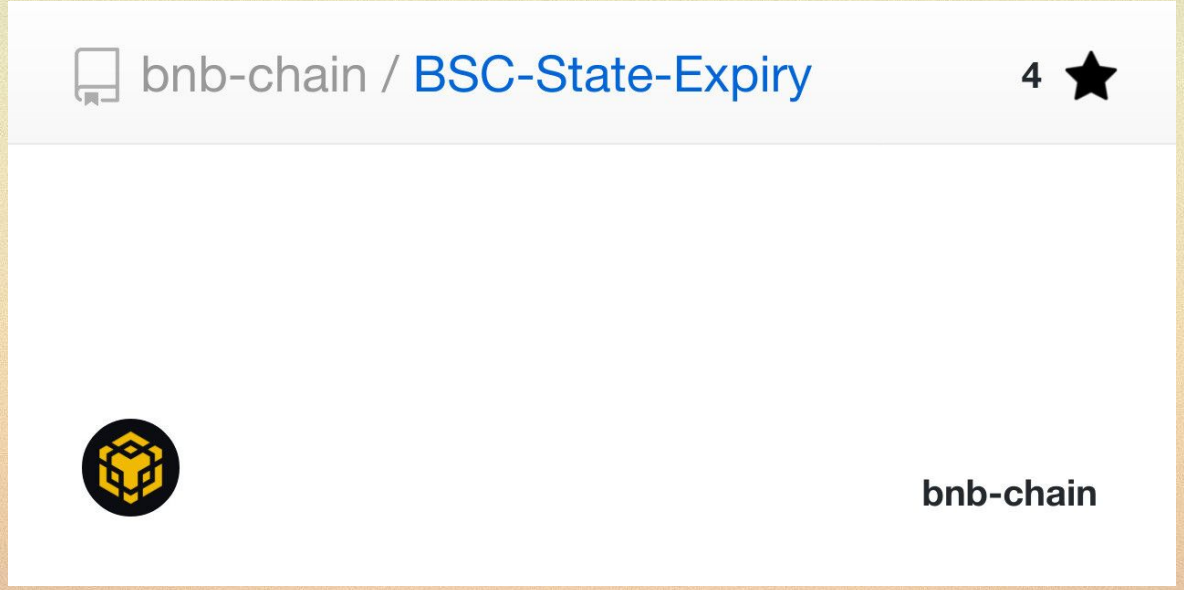
300 blocks (6 months)

st accessed 1 year ago

State Revive ✓

oof of expired subtree

State Expiry on MPT



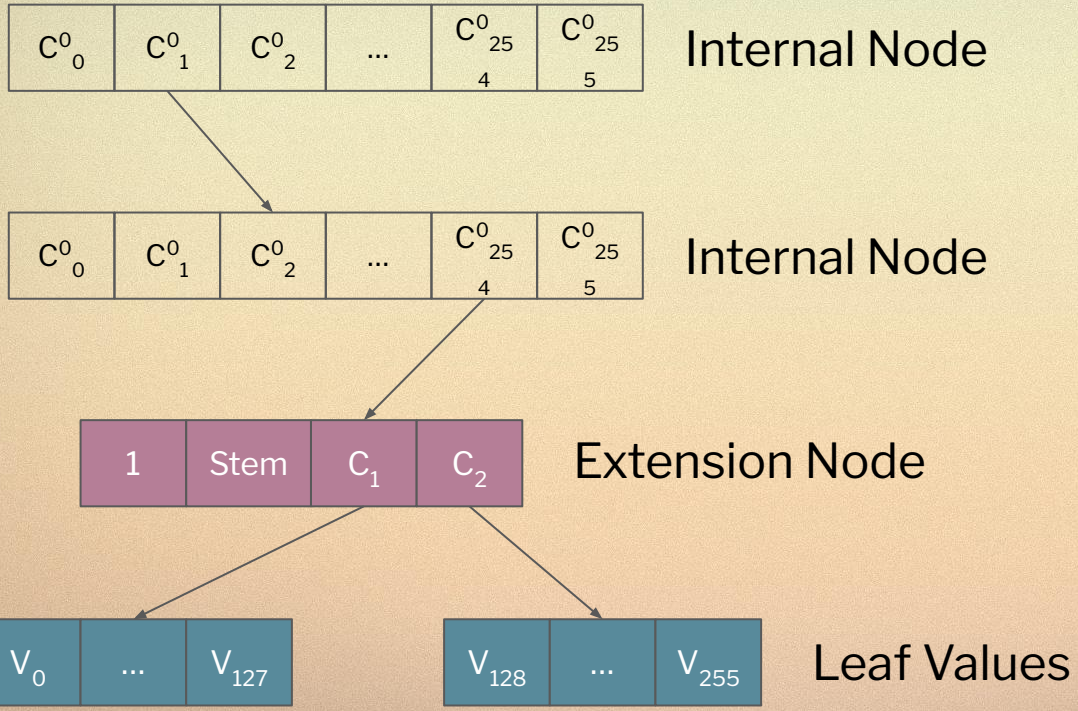
github.com/bnb-chain/BSC-State-Expiry

There’s a big issue.

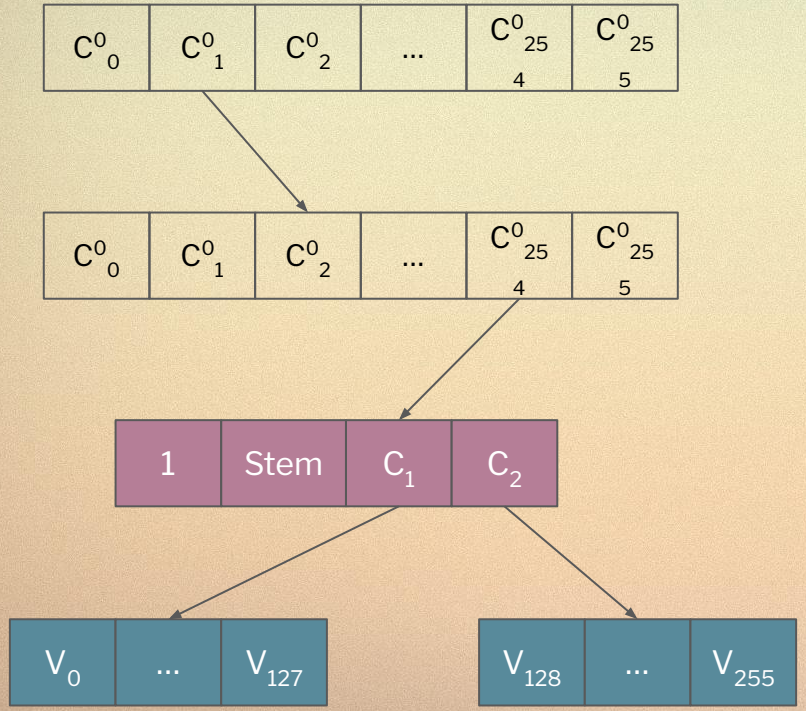
Proof size.

Solution: Verkle Tree

Verkle Tree


C^0_0
C^0_1
C^0_2
...
C^0_{254}
C^0_{255}
Internal Node
C^0_0
C^0_1
C^0_2
...
C^0_{254}
C^0_{255}
Internal Node
1
Stem
C_1
C_2
Extension Node
V_0
...
V_{127}
V_{128}
...
V_{255}
Leaf Values

Verkle Tree



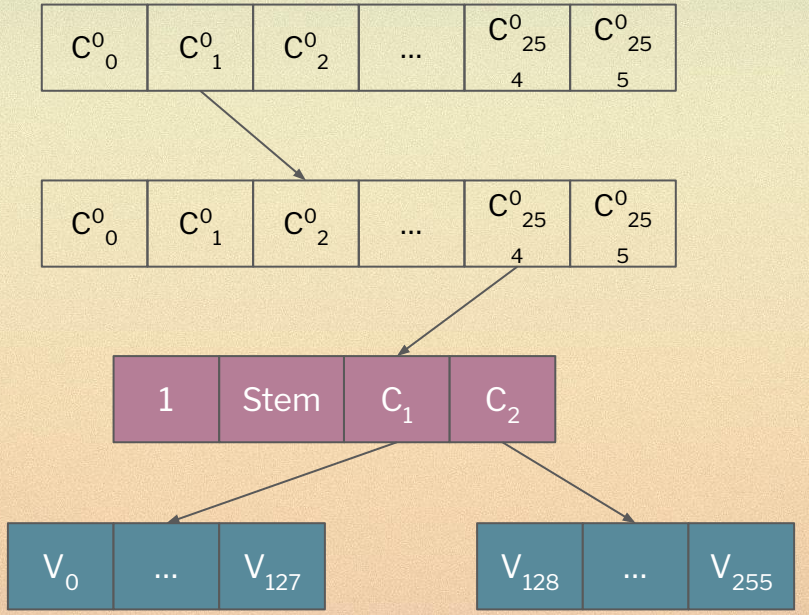
Why Verkle?

- Small proof sizes
- Allows for stateless clients
- Reduces hardware requirements
- Improves node syncing experience
- ZK-friendly

Major Change

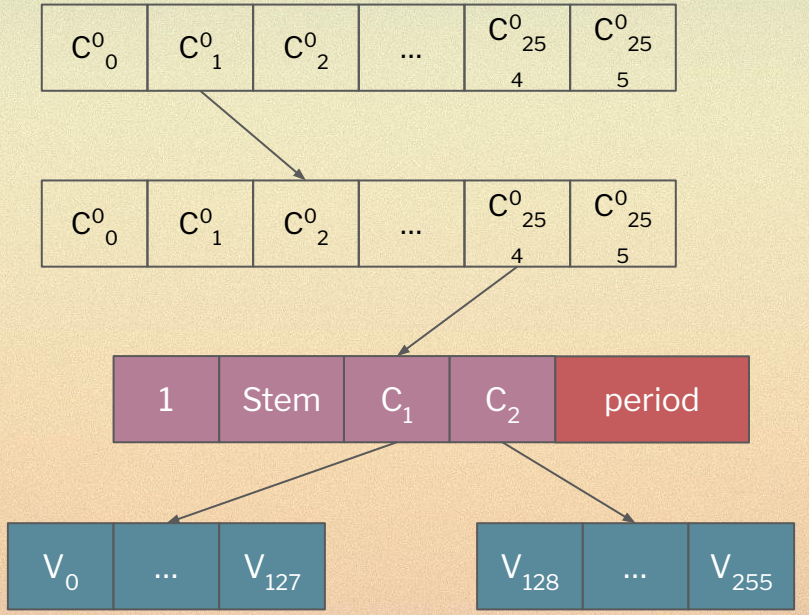
- Single tree approach instead of double layer

State Expiry on Verkle Tree



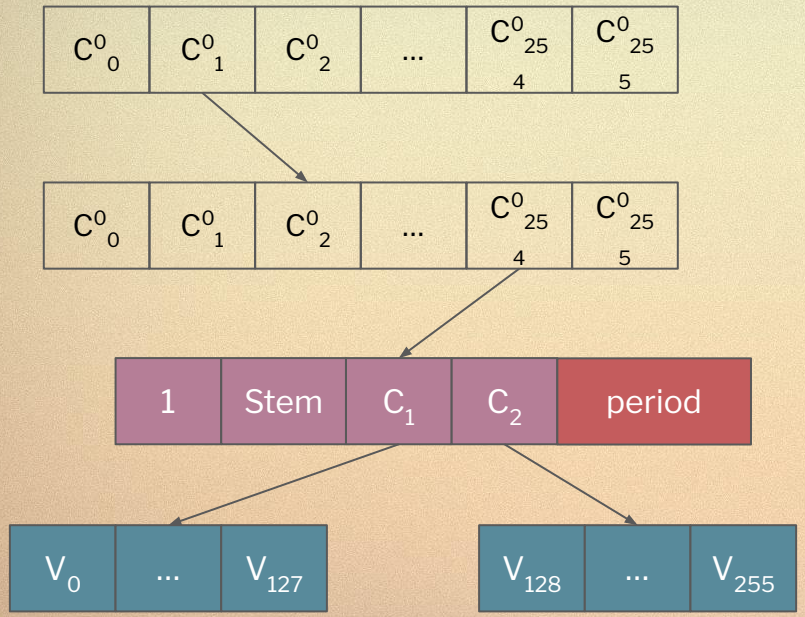
In Verkle Tree, leaf values are **grouped together** under an extension node.

State Expiry on Verkle Tree



We do the same thing by adding **period** to each extension node.

State Expiry on Verkle Tree

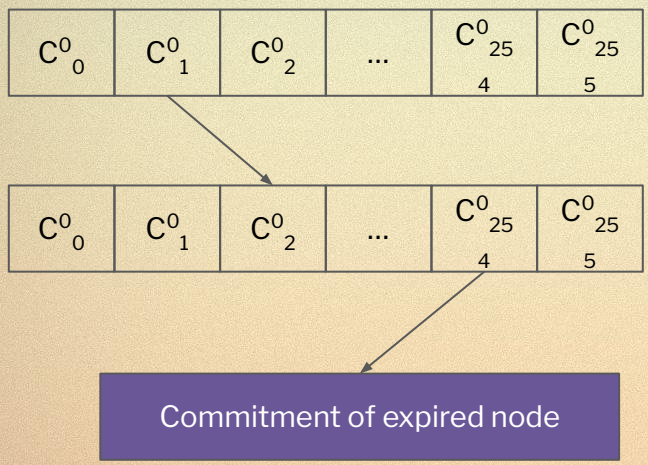


State Expiry

1 period = 15778800 blocks (6 months)

Expires when node was last accessed 1 year ago

State Expiry on Verkle Tree



State Expiry ✓

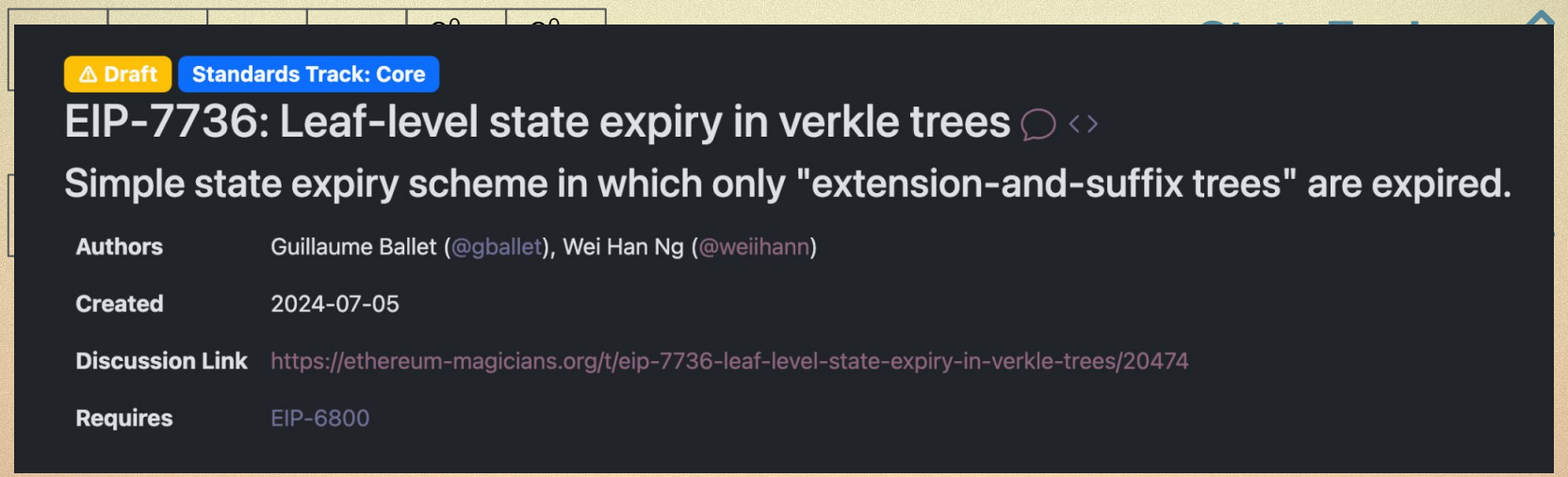
1 period = 15778800 blocks (6 months)

Expires when node was last accessed 1 year ago

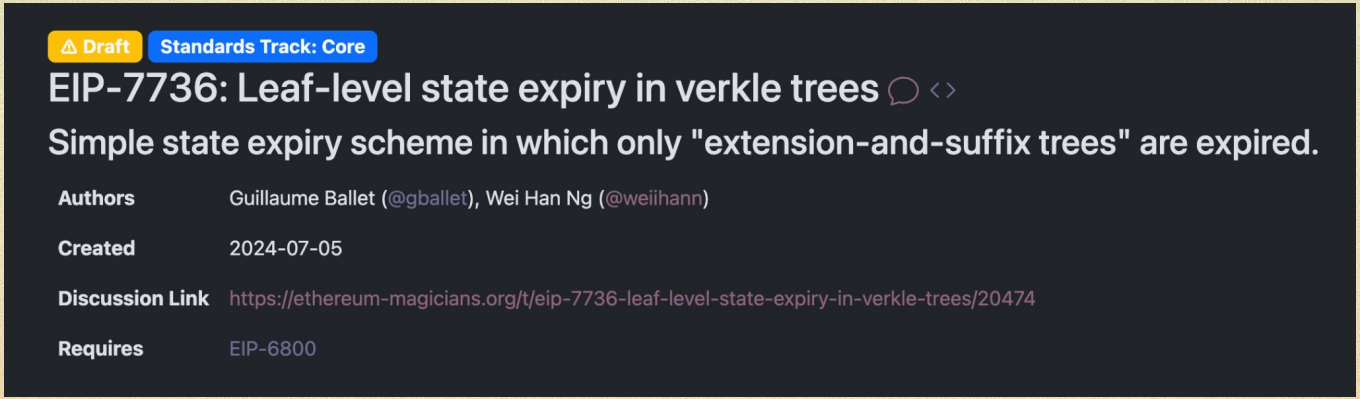
State Revive ✓

Provide **path to the extension node** + **period** + **values**

State Expiry on Verkle Tree



State Expiry on Verkle Tree



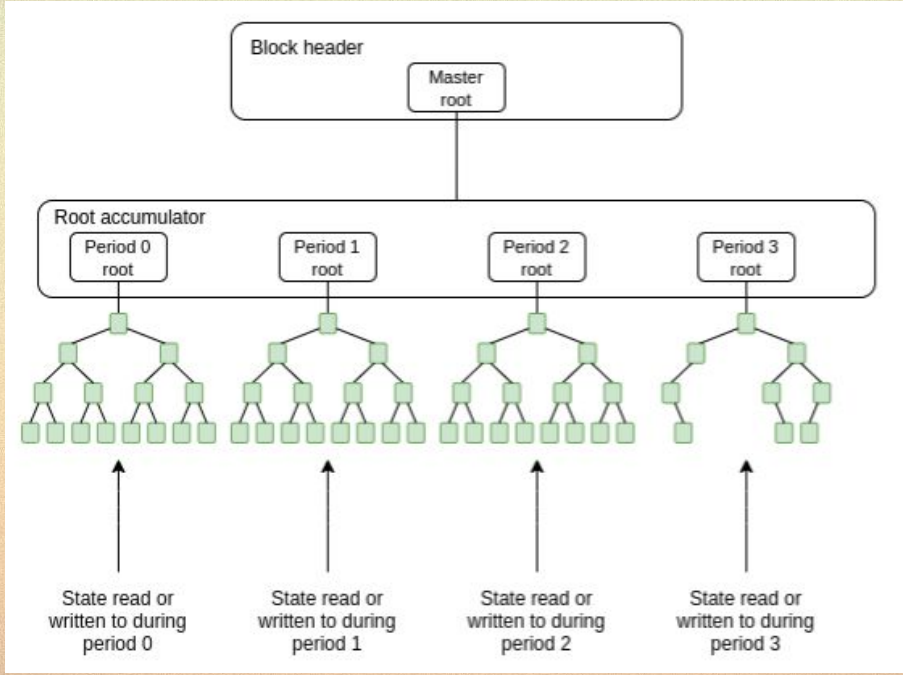
Pros

- Simple to implement
- Clear gas costs
- Small revive proofs
- Backward compatible with Verkle

Cons

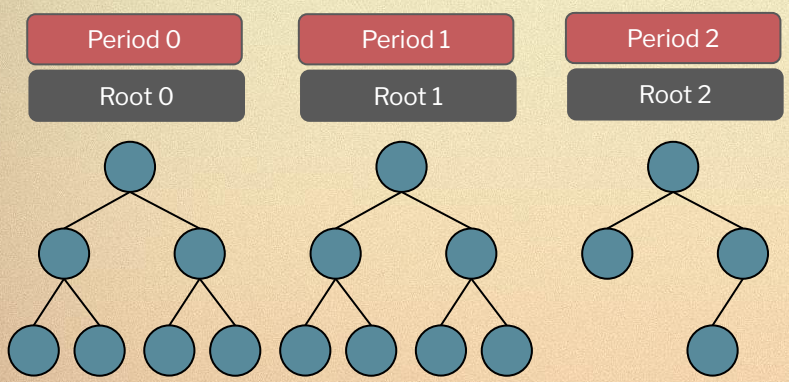
- Only expires state partially

Full State Expiry on Verkle Tree



Vitalik's multi period tree approach, published in 2021.

Full State Expiry on Verkle Tree



State Expiry

A new state tree is created every 1 period.

State older than 2 periods are expired.

State Revive

Requires the last known state tree of the account.

Proof of all subsequent periods up until the current period - 1.

Super complex proposal

Requires address space extension

Data duplication

Larger proof size

FUTURE

State Expiry

We get:

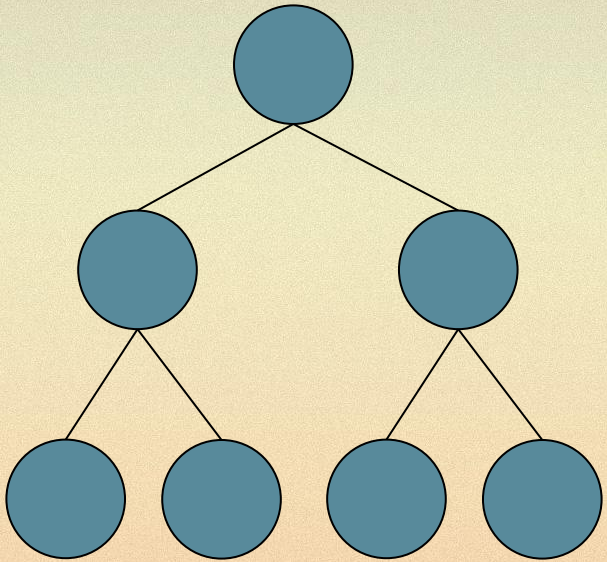
- Slower state growth
- Reduce hardware requirements
- More efficient execution client

In exchange for:

- Worse user experience
- Additional state revive cost

That's why we need:

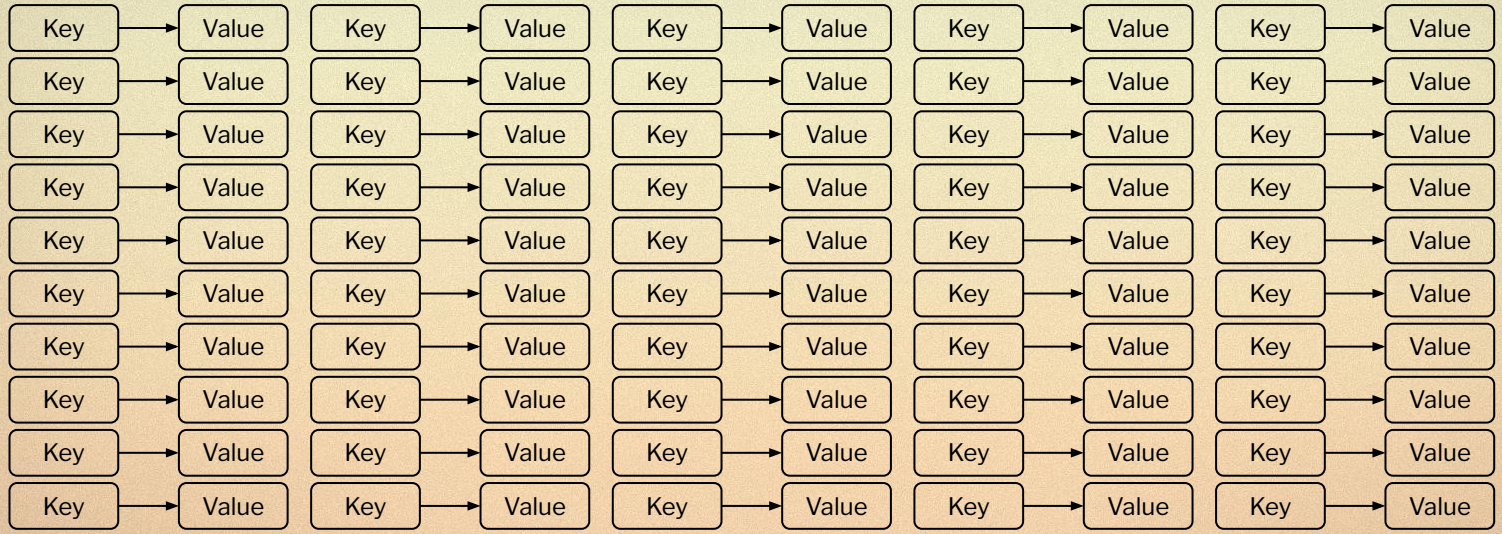
- Decentralized proving service (Portal Network)



We might go for **binary tree** instead.

Block 21,178,332

Time



Ethereum's state just grew by **1.9MB**.

Let's build an Ethereum that scales in time.

Thank you!

Han
Engineer @ Nethermind
@ngweihan_eth