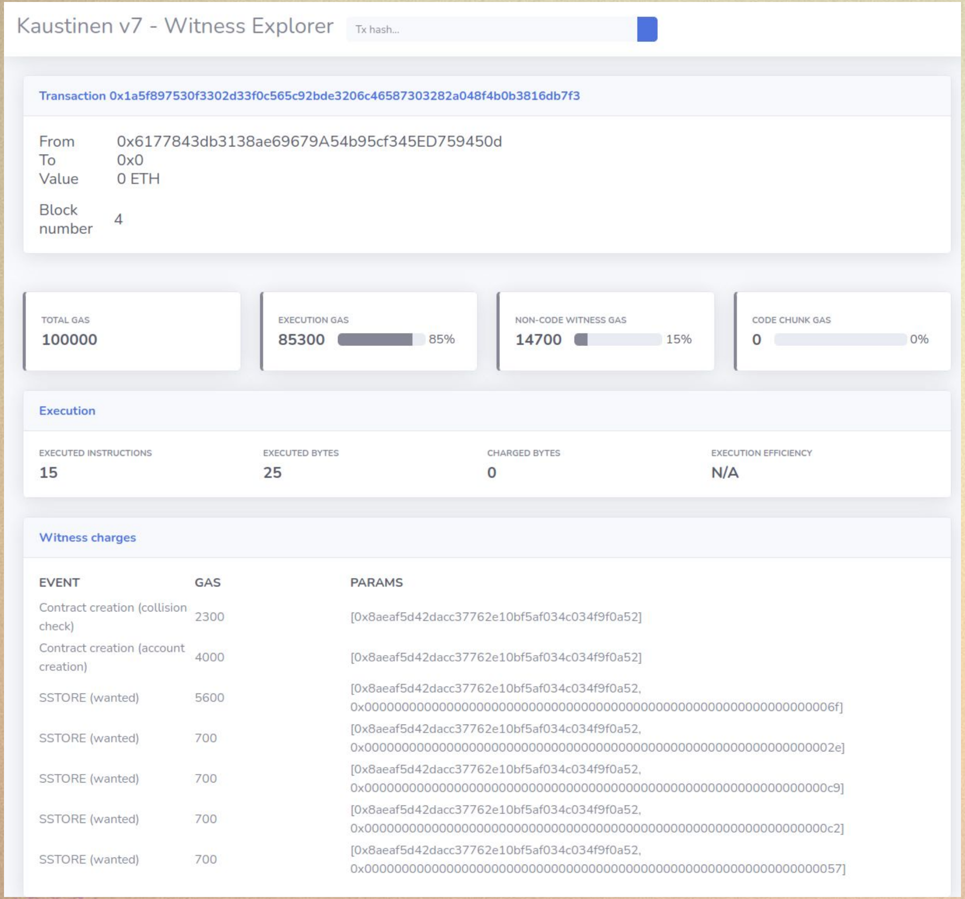
The verkle advantage

Guillaume Ballet

November 13, 2024

Section 1

Readiness


Kaustinen v7 - Witness Explorer

Transaction 0x1a5f897530f3302d33f0c565c92bde3206c46587303282a048f4b0b3816db7f3

From	0x6177843db3138ae69679A54b95cf345ED759450d
To	0x0
Value	0 ETH
Block number	4

TOTAL GAS	EXECUTION GAS	NON-CODE WITNESS GAS	CODE CHUNK GAS
100000	85300 85%	14700 15%	0 0%

Execution

EXECUTED INSTRUCTIONS	EXECUTED BYTES	CHARGED BYTES	EXECUTION EFFICIENCY
15	25	0	N/A

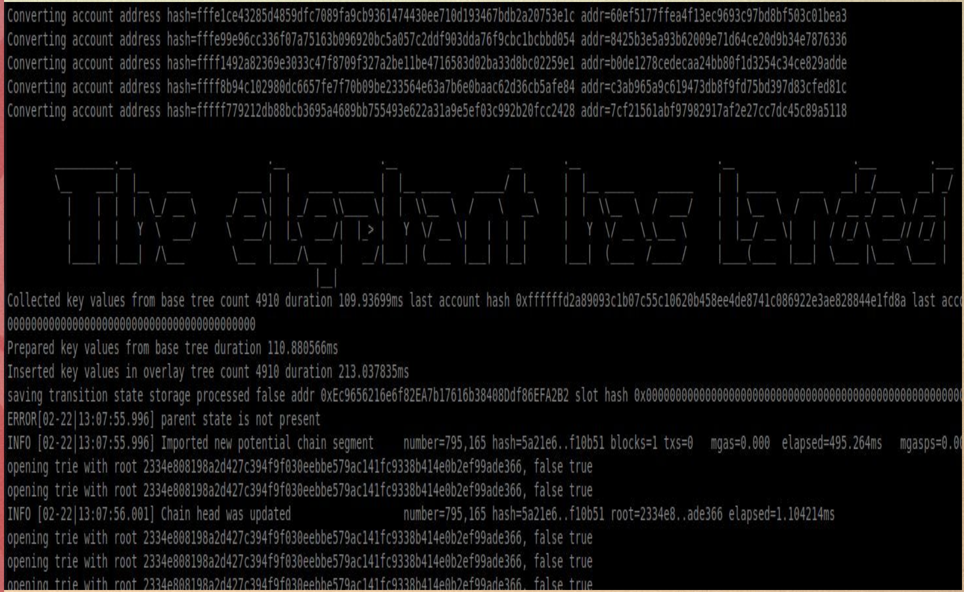
Witness charges

EVENT	GAS	PARAMS
Contract creation (collision check)	2300	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52]
Contract creation (account creation)	4000	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52]
SSTORE (wanted)	5600	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52, 0x000000000000000000000000000000000000000000000000000000000000006f]
SSTORE (wanted)	700	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52, 0x000000000000000000000000000000000000000000000000000000000000002e]
SSTORE (wanted)	700	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52, 0x00000000000000000000000000000000000000000000000000000000000000c9]
SSTORE (wanted)	700	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52, 0x00000000000000000000000000000000000000000000000000000000000000c2]
SSTORE (wanted)	700	[0x8aeaf5d42dacc37762e10bf5af034c034f9f0a52, 0x0000000000000000000000000000000000000000000000000000000000000057]

Devnet 7 launched !

- Latest eip-4762 spec
  - except FILL_COST
- More stress
- Using execution spec tests
- Eip 7709 spec
- https://verkle-gen-devnet-7.ethpandaops.io

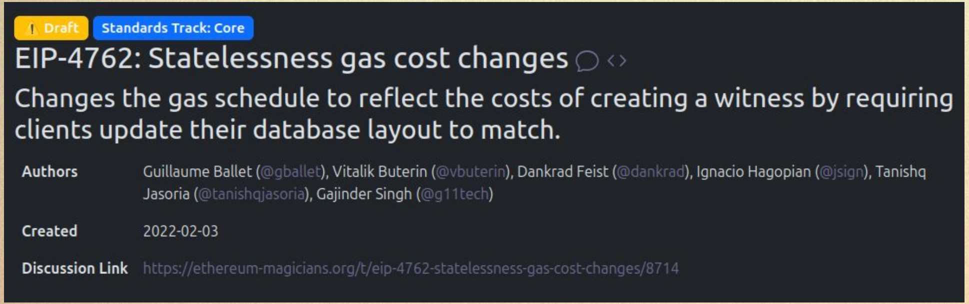
Holesky shadowfork



```
Converting account address hash=fffe1ce43285d4859dfc7089fa9cb9361474430ee710d193467bdb2a20753e1c addr=60ef5177ffea4f13ec9693c97bd8bf503c01bea3
Converting account address hash=fffe99e96cc336f07a75163b096920bc5a057c2ddf903dda76f9cbc1bcbbd054 addr=8425b3e5a93b62009e71d64ce20d9b34e7876336
Converting account address hash=ffff1492a82369e3033c47f8709f327a2be11be4716583d02ba33d8bc02259e1 addr=b0de1278cedecaa24bb80f1d3254c34ce829adde
Converting account address hash=ffff8b94c102980dc6657fe7f70b09be233564e63a7b6e0baac62d36cb5afe84 addr=c3ab965a9c619473db8f9fd75bd397d83cfed81c
Converting account address hash=fffff779212db88bcb3695a4689bb755493e622a31a9e5ef03c992b20fcc2428 addr=7cf21561abf97982917af2e27cc7dc45c89a5118


    The elephant has landed

Collected key values from base tree count 4910 duration 109.93699ms last account hash 0xffffffd2a89093c1b07c55c10620b458ee4de8741c086922e3ae828844e1fd8a last acc
0000000000000000000000000000000000000000
Prepared key values from base tree duration 110.880566ms
Inserted key values in overlay tree count 4910 duration 213.037835ms
saving transition state storage processed false addr 0xEc9656216e6f82EA7b17616b38408Ddf86EFA2B2 slot hash 0x0000000000000000000000000000000000000000000000000000
ERROR[02-22|13:07:55.996] parent state is not present
INFO [02-22|13:07:55.996] Imported new potential chain segment     number=795,165 hash=5a21e6..f10b51 blocks=1 txs=0   mgas=0.000  elapsed=495.264ms   mgasps=0.0
opening trie with root 2334e808198a2d427c394f9f030eebbe579ac141fc9338b414e0b2ef99ade366, false true
opening trie with root 2334e808198a2d427c394f9f030eebbe579ac141fc9338b414e0b2ef99ade366, false true
INFO [02-22|13:07:56.001] Chain head was updated                   number=795,165 hash=5a21e6..f10b51 root=2334e8..ade366 elapsed=1.104214ms
opening trie with root 2334e808198a2d427c394f9f030eebbe579ac141fc9338b414e0b2ef99ade366, false true
opening trie with root 2334e808198a2d427c394f9f030eebbe579ac141fc9338b414e0b2ef99ade366, false true
opening trie with root 2334e808198a2d427c394f9f030eebbe579ac141fc9338b414e0b2ef99ade366, false true
```


Draft Standards Track: Core

EIP-4762: Statelessness gas cost changes

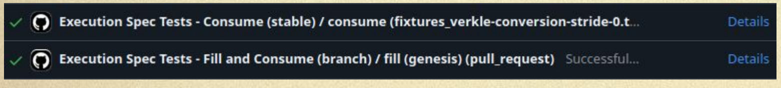
Changes the gas schedule to reflect the costs of creating a witness by requiring clients update their database layout to match.

Authors	Guillaume Ballet (@gballet), Vitalik Buterin (@vbuterin), Dankrad Feist (@dankrad), Ignacio Hagopian (@jsign), Tanishq Jasoria (@tanishqjasoria), Gajinder Singh (@g11tech)
Created	2022-02-03
Discussion Link	https://ethereum-magicians.org/t/eip-4762-statelessness-gas-cost-changes/8714

Gas cost spec

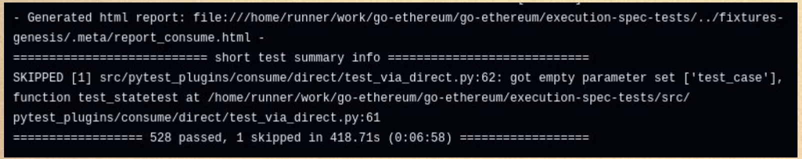
- Simplifications
- Clarified a lot of corner cases
- Prepares for the upcoming forks (eip-7702, EOF, ...)

Execution spec tests



✓ Execution Spec Tests - Consume (stable) / consume (fixtures_verkle-conversion-stride-0.t... Details

✓ Execution Spec Tests - Fill and Consume (branch) / fill (genesis) (pull_request) Successful... Details

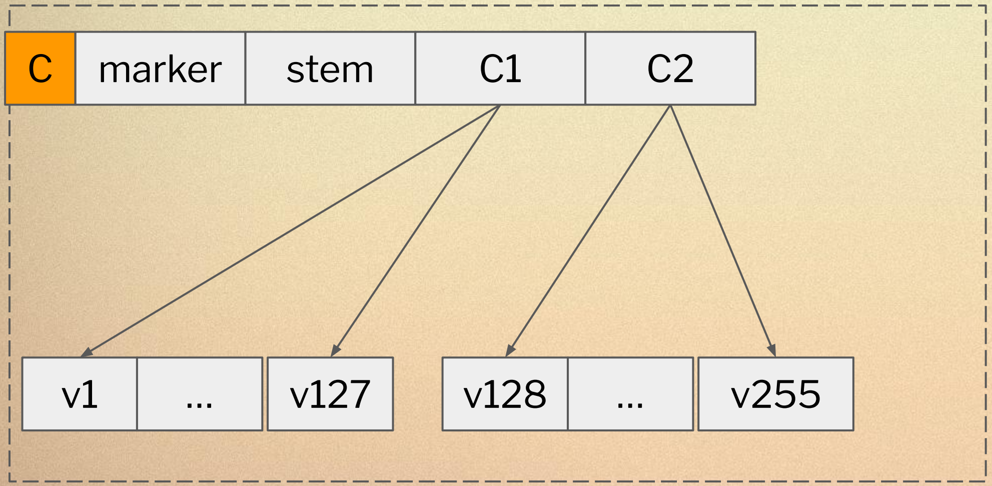


```
- Generated html report: file:///home/runner/work/go-ethereum/go-ethereum/execution-spec-tests/../fixtures-
genesis/.meta/report_consume.html -
=========================== short test summary info ============================
SKIPPED [1] src/pytest_plugins/consume/direct/test_via_direct.py:62: got empty parameter set ['test_case'],
function test_statetest at /home/runner/work/go-ethereum/go-ethereum/execution-spec-tests/src/
pytest_plugins/consume/direct/test_via_direct.py:61
================== 528 passed, 1 skipped in 418.71s (0:06:58) ==================
```

- Created > 200 execution spec tests
- Found countless consensus bugs before the testnet launch
- Runs in (verkle) geth CI on every PR
- Better developer experience for execution clients that want to check their implementation

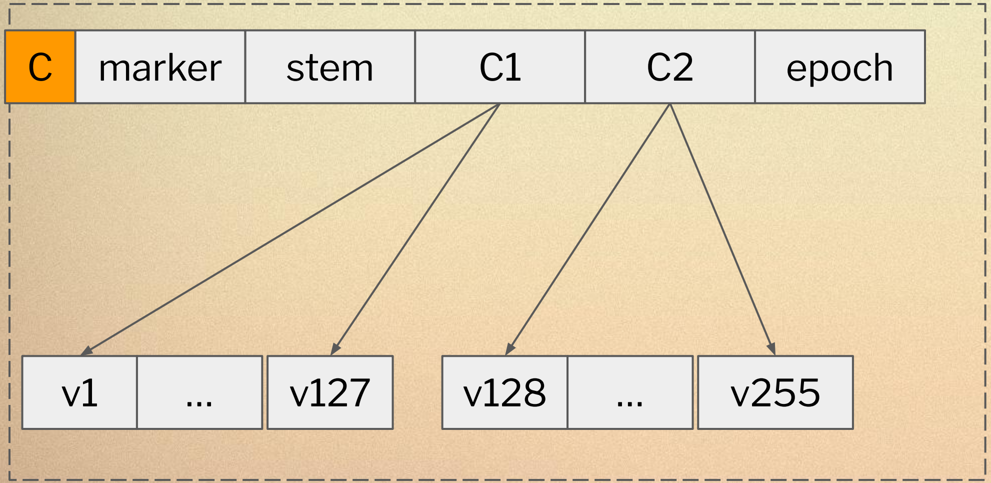
Verkle sync

- Adaptation of snap sync, using verkle proofs & witnesses
- Demonstrated on the verkle testnet
- Big shoutout to Tanishq from Nethermind for building it
- First stateless feature that wasn't first implemented in geth



State expiry

- EPF project by Han
- Add an epoch to the leaf node
  - Polynomial -> backwards compatible!
- Only delete the leaves
  - Worth it, because the verkle tree is very small, and grows slowly
- Expire and resurrect 256 leaves at once
- Single tree, simpler design
  - E.g. no “epoch” in address

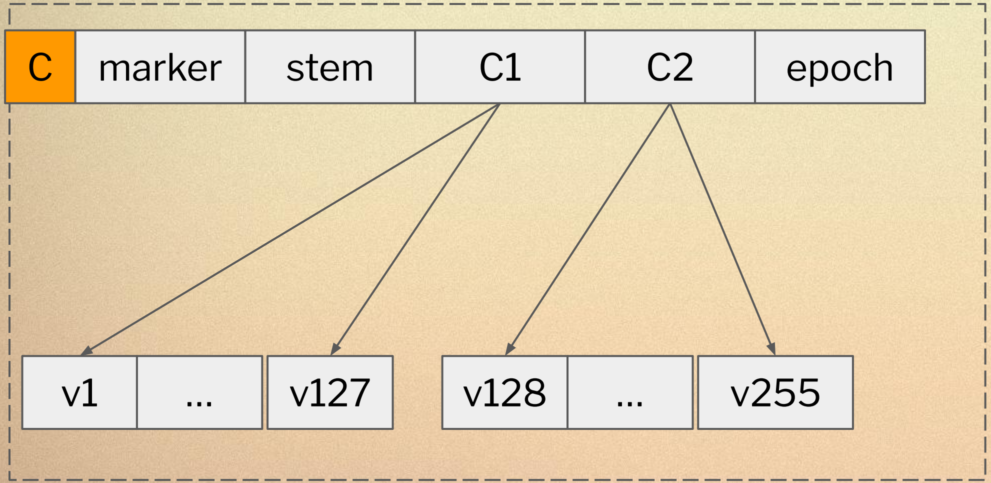


State expiry

- EPF project by Han
- Add an epoch to the leaf node
  - Polynomial -> backwards compatible!
- Only delete the leaves
  - Worth it, because the verkle tree is very small, and grows slowly
- Expire and resurrect 256 leaves at once
- Single tree, simpler design
  - E.g. no “epoch” in address

State expiry

- EPF project by Han
- Add an epoch to the leaf node
    - Polynomial -> backwards compatible!
- Only delete the leaves
    - Worth it, because the verkle tree is very small, and grows slowly
- Expire and resurrect 256 leaves at once
- Single tree, simpler design
    - E.g. no “epoch” in address



State expiry

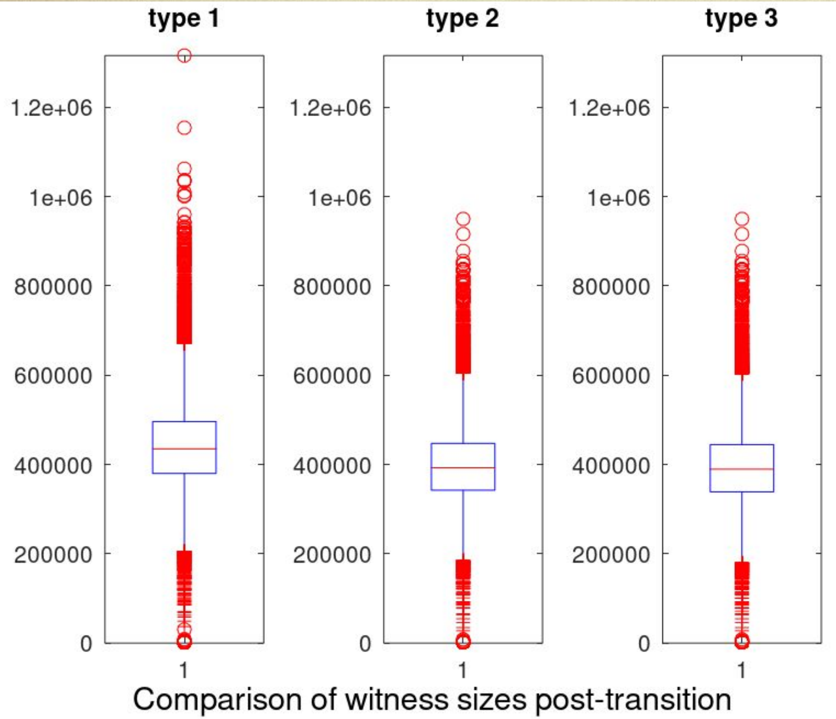
- EPF project by Han
- Add an epoch to the leaf node
  - Polynomial -> backwards compatible!
- Only delete the leaves
  - Worth it, because the verkle tree is very small, and grows slowly
- Expire and resurrect 256 leaves at once
- Single tree, simpler design
  - E.g. no “epoch” in address

After 4 years of development, verkle is ready for its prime

- Major questions answered
- Implemented to some degree in most EL clients
- Integration work still left to do
    - 2 EL clients missing
    - Preimage distribution
    - RPCs
    - Some client optimizations are still on the roadmap

Section 2

Verkle pros & cons



Comparison of witness sizes post-transition

Verkle proofs + witnesses are small

<400kb average at current gas limit

- Witness size grows linearly with gas limit
- Worst observed case is < 1MB
- Good for bandwidth
- Good for decentralization
- Fast too: proofs can be built within a slot on an average-powered machine

Verkle proofs + witnesses are small

25% smaller than the MPT

- Less nodes
- Homomorphism => less storage
- Size of commitments is inflated because compressing them takes a lot of time
- Optimizing that would further save 10s of GB
- Disk size requirements grow smaller than the MPT
- Improved DB access patterns

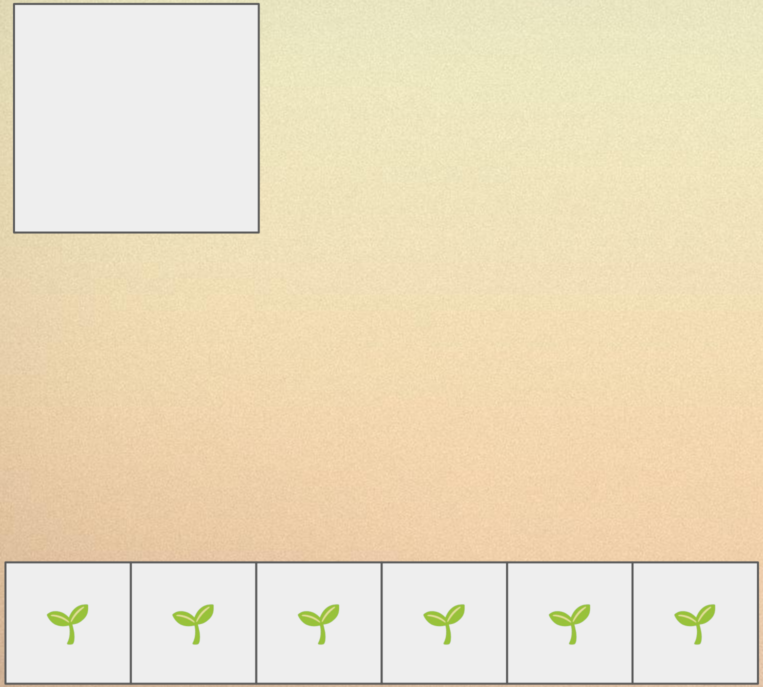
Verkle *trees* are small

Improved sync UX

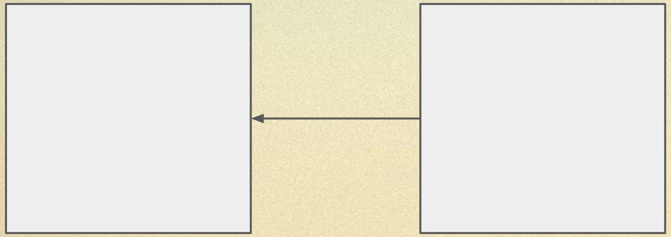
- Immediately join the network
- Healing is also immediate
  - Necessary data is bound to block
  - No endless back-and-forth over the network

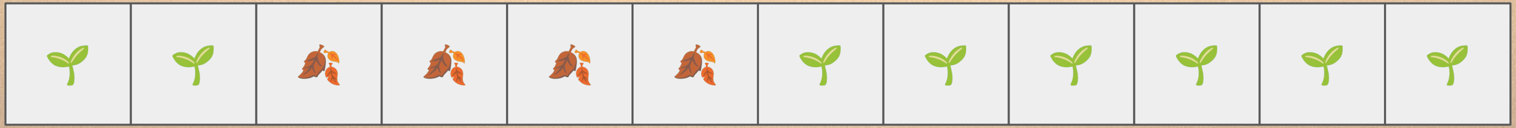
Verkle makes sync easier

Sync: Start downloading state

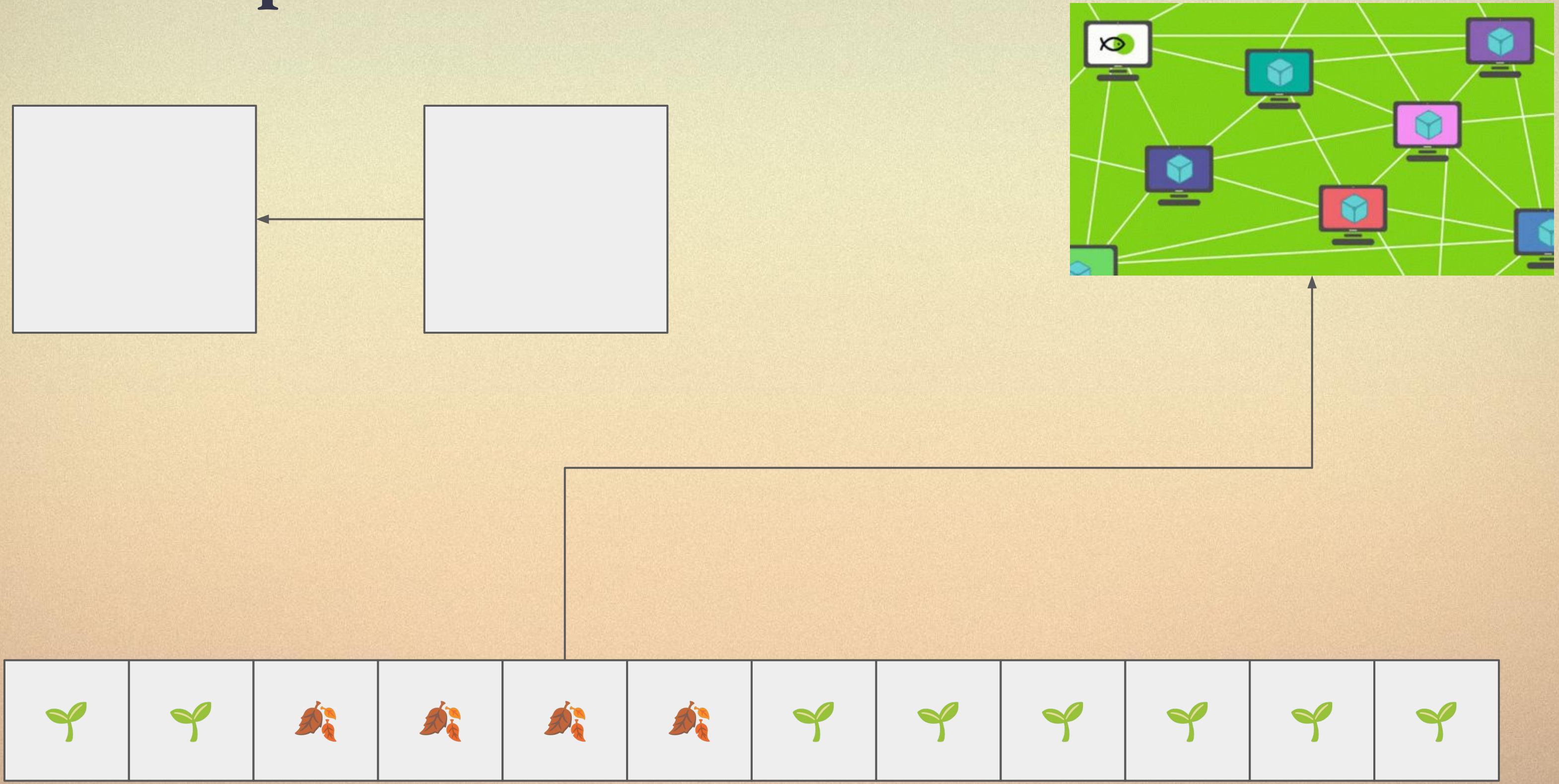


Sync: Complete state download

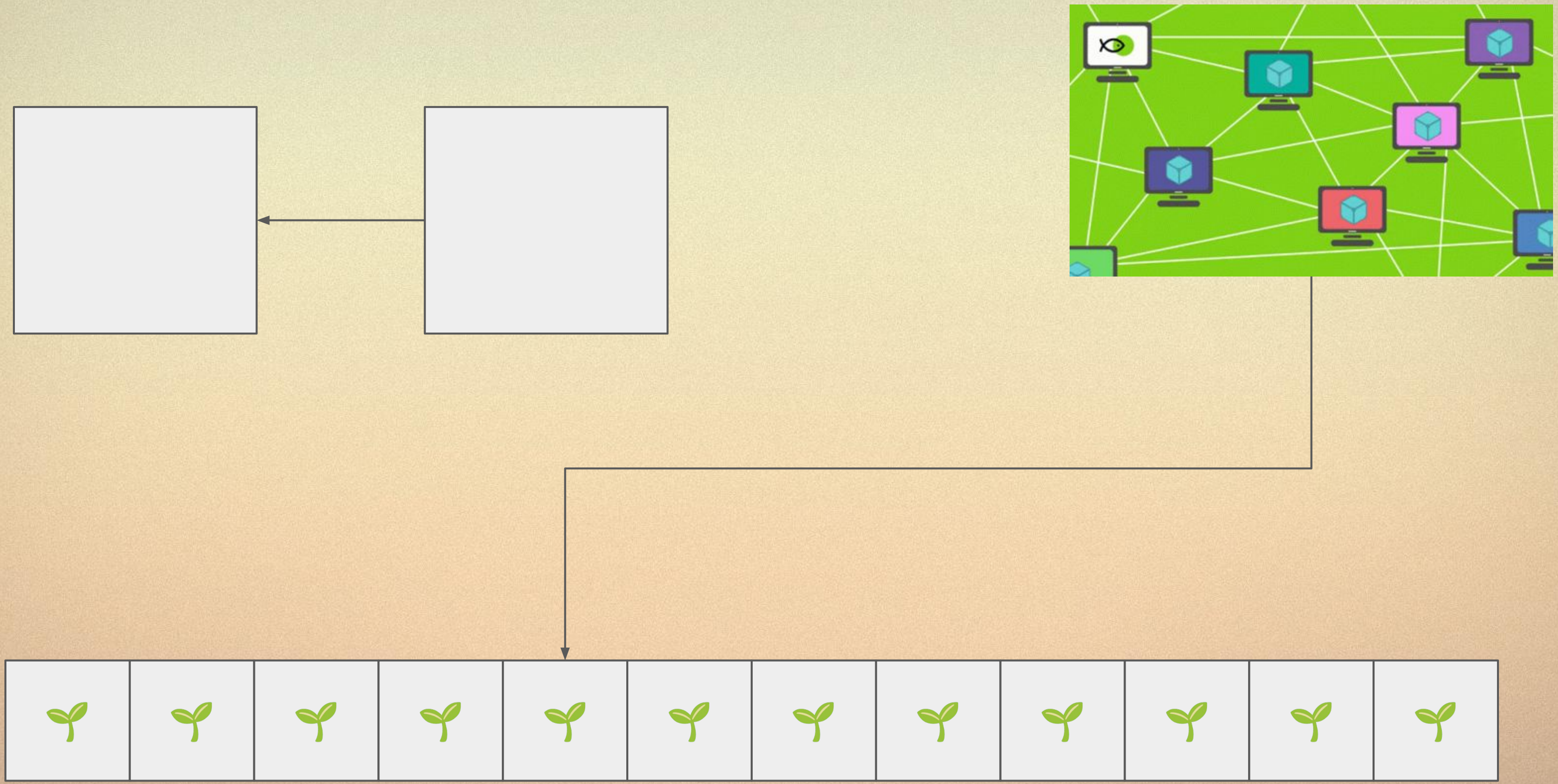




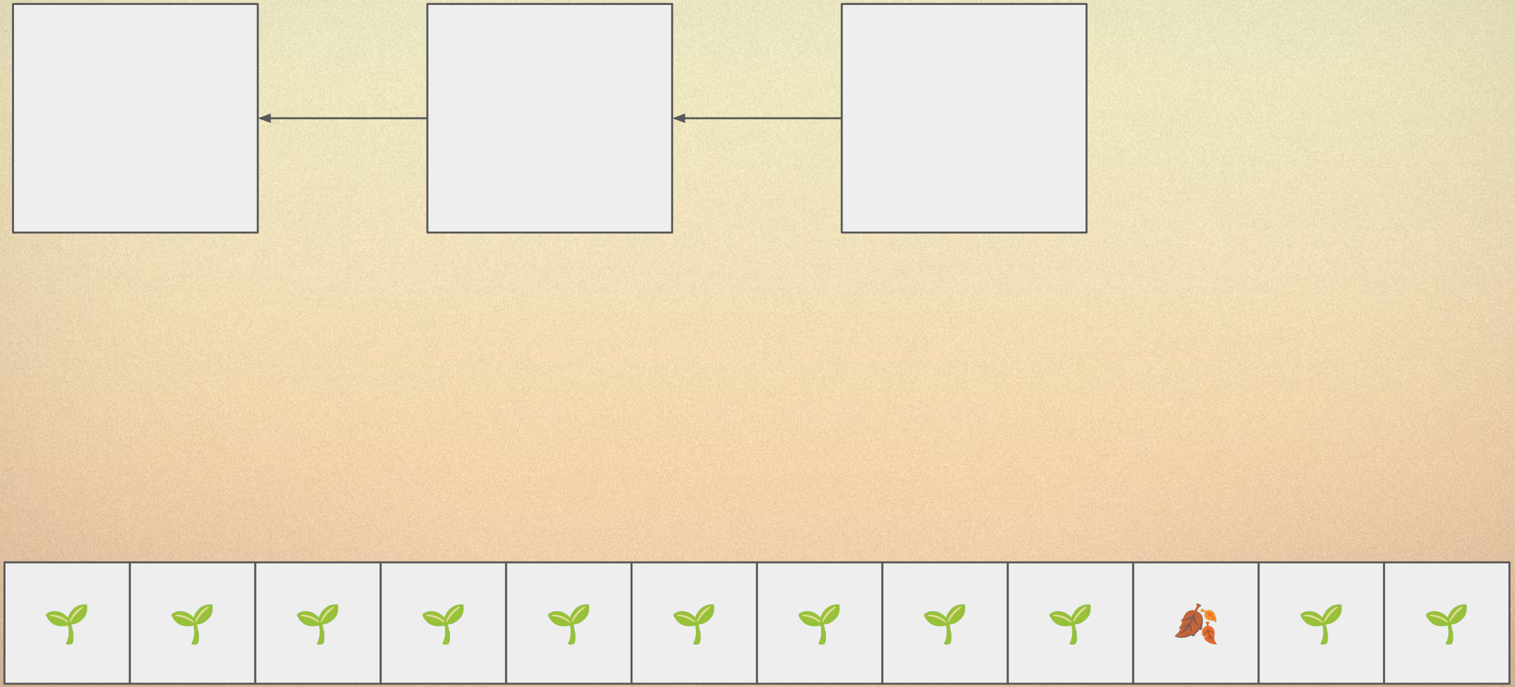
Sync: Complete state download



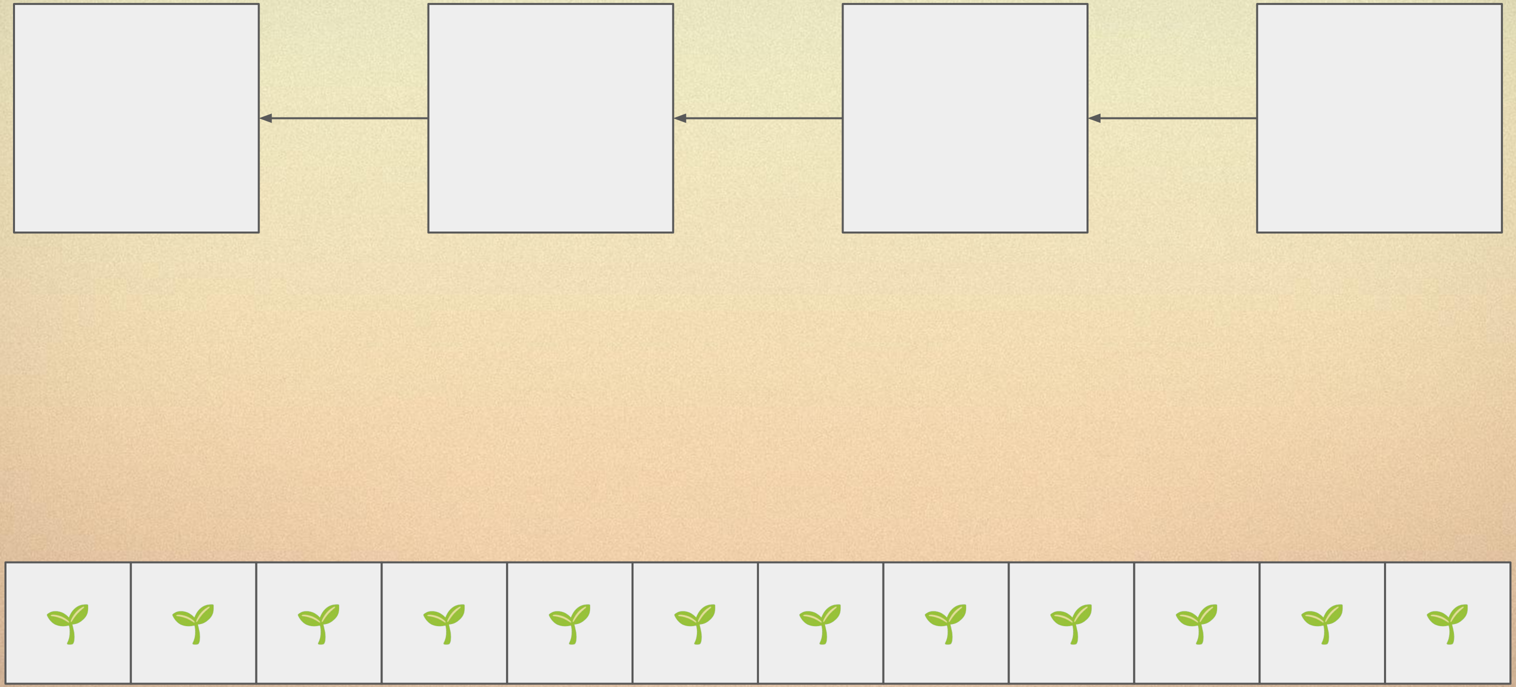
Sync: Heal Phase



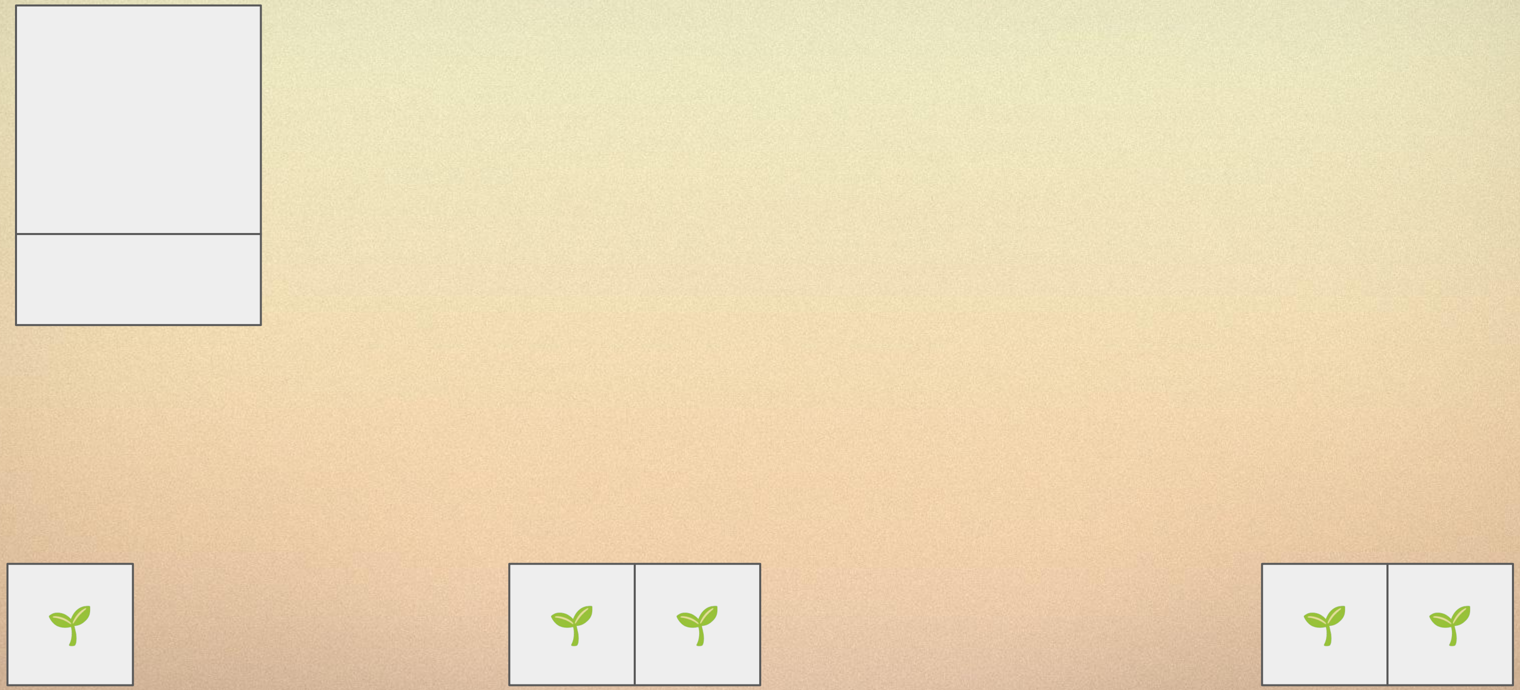
Sync: Heal Phase



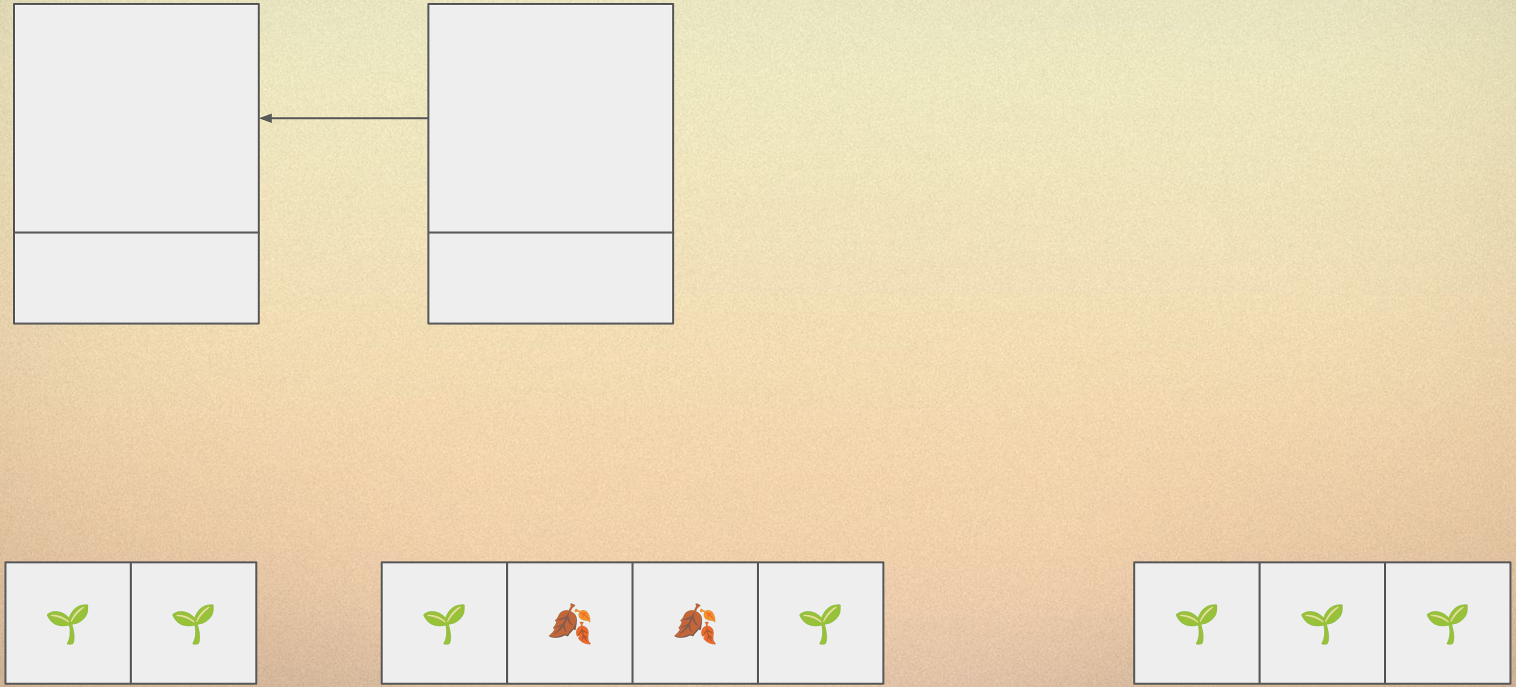
Sync Completion



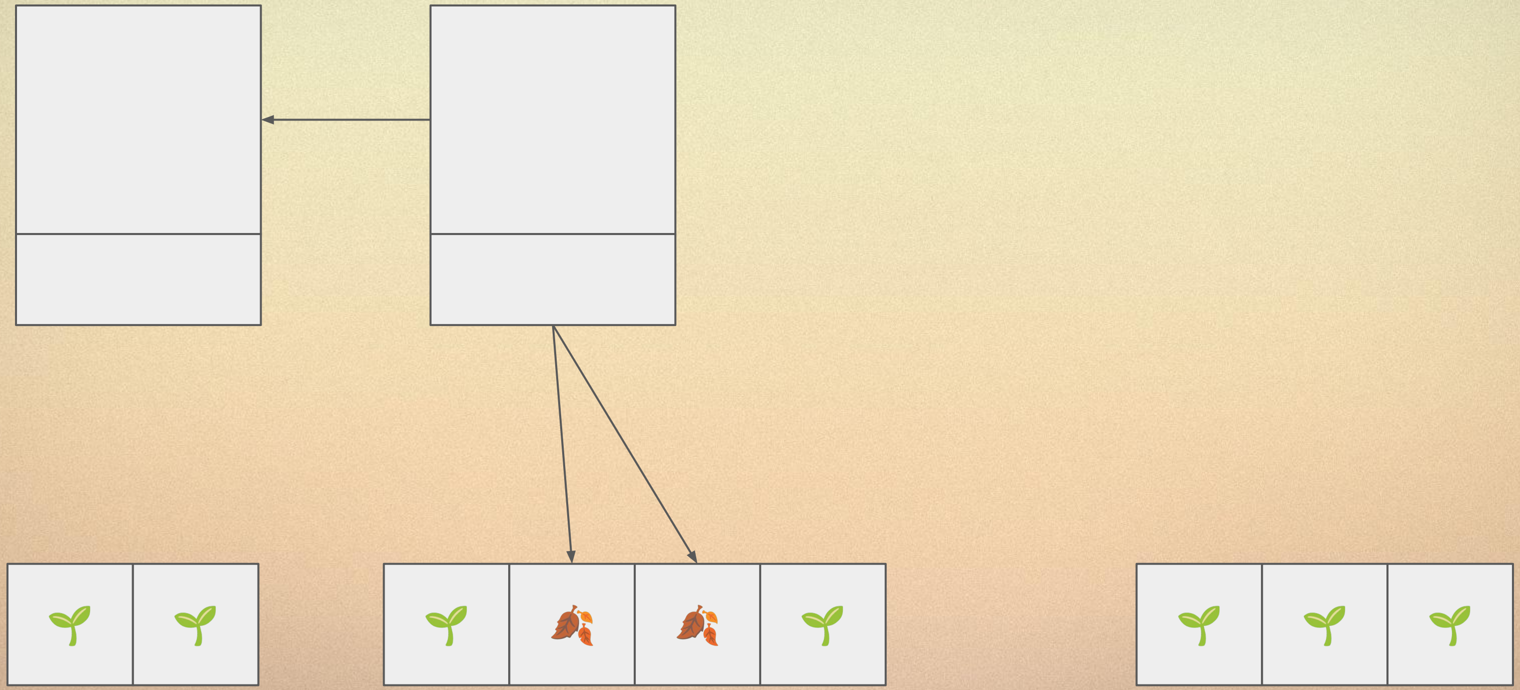
Verkle Sync: “Instant heal”



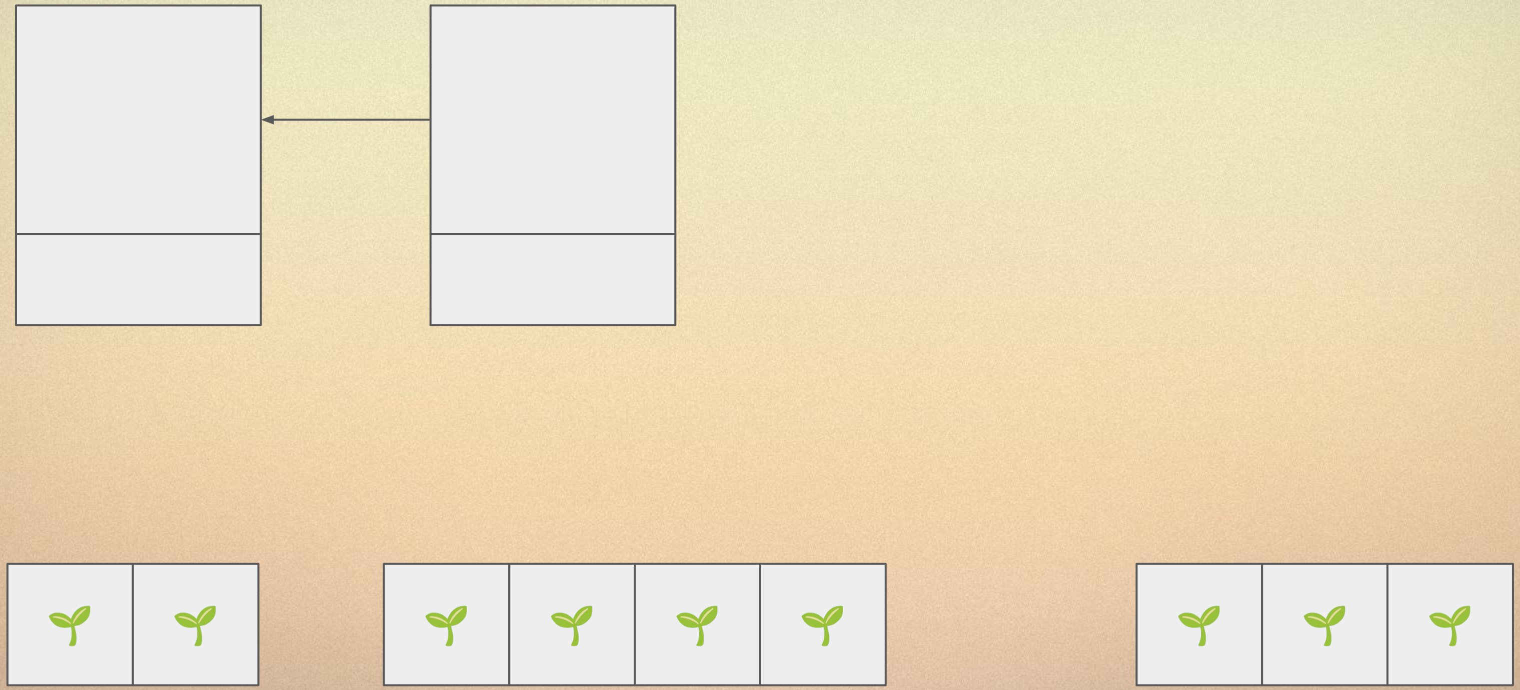
Verkle Sync: “Instant heal”



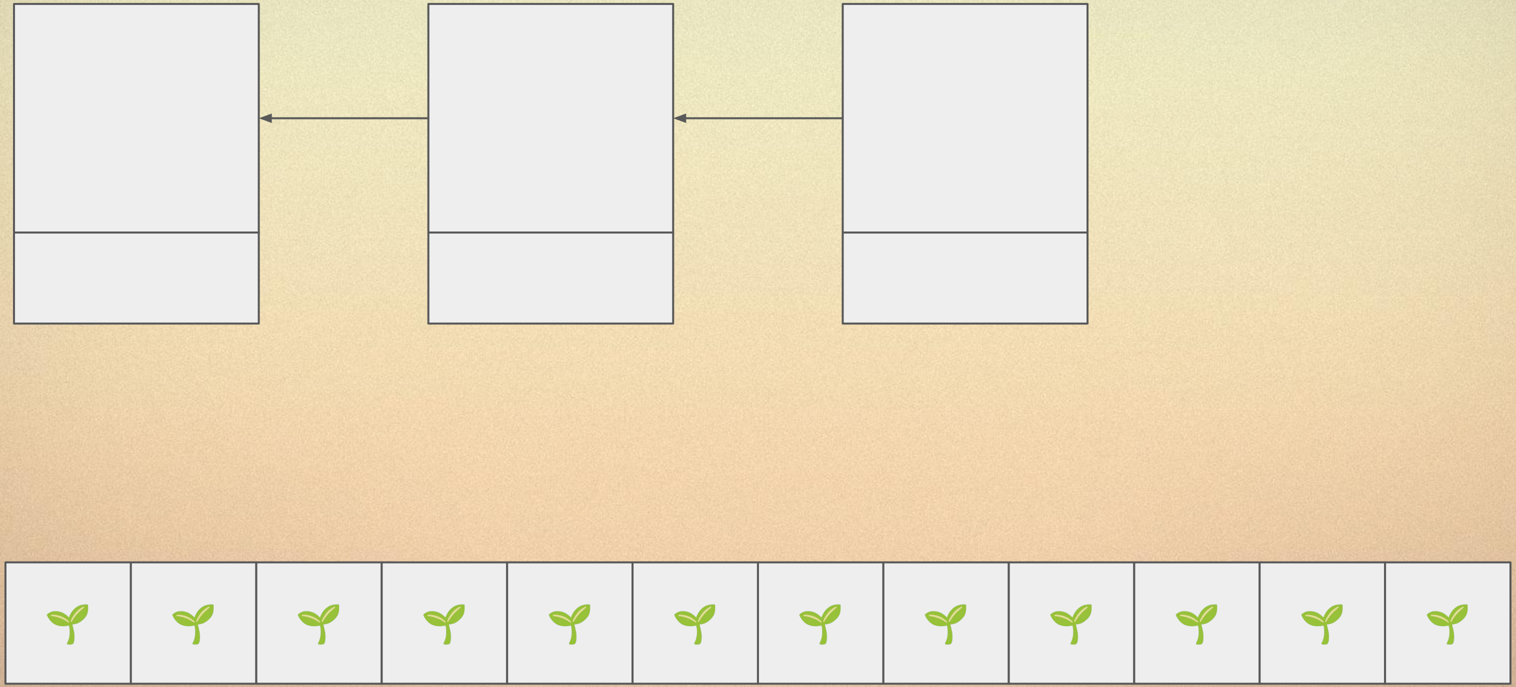
Verkle Sync: “Instant heal”



Verkle Sync: “Instant heal”



Verkle Sync: “Instant heal”



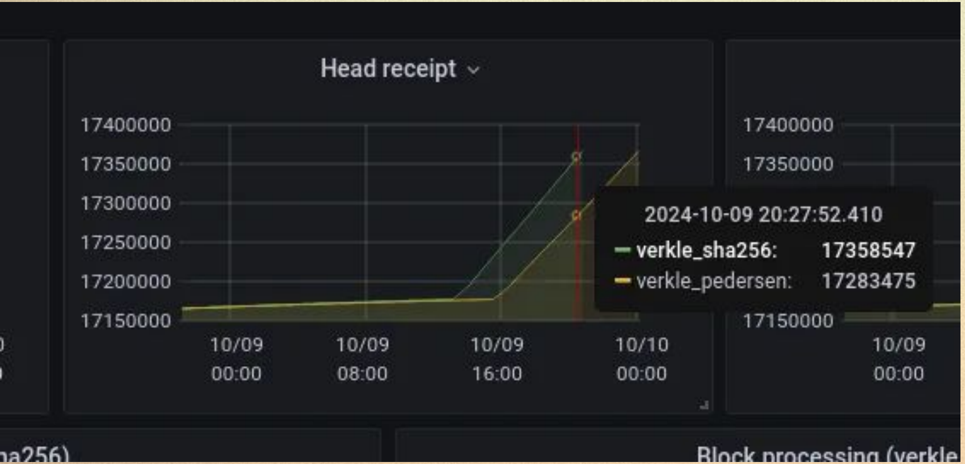
Issues

- No quantum resistance
  - ECDLP is broken by quantum computers
  - Pedersen hashes are no longer secure
    - Keys can be crafted to deepen the tree
    - Fake proofs of existence can be made
- Less zk-friendly than binary trees with Poseidon
  - Still very zk-friendly
  - Zk-friendliness comes at the cost of an expensive Pedersen hash function

Potential fixes:

- Classic hash for tree keys
  - No more unbalancing
  - Negative impact on zk-friendliness
- Crafting a fake proof is still possible, creating an account of thin air…
  - …but only stateless nodes would be fooled
  - Worst case scenario: forced fallback to statefulness
- Quantum-resistant, zk-friendly crypto in verkle (e.g. lattices)
  - Needs more research

Issues : potential mitigations



Potential fixes:

- Classic hash for tree keys
  - No more unbalancing
  - Negative impact on zk-friendliness
- Crafting a fake proof is still possible, creating an account of thin air…
  - …but only stateless nodes would be fooled
  - Worst case scenario: forced fallback to statefulness
- Quantum-resistant, zk-friendly crypto in verkle (e.g. lattices)
  - Needs more research

Section 3

Binary trees

The good

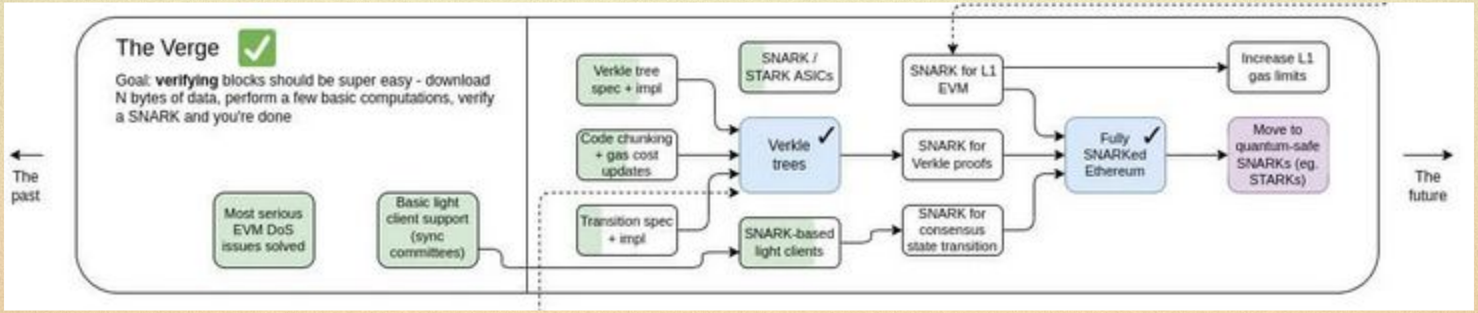
- Quantum secure
- Fast hash function
- Plays nice with small field STARKs (circle, binius, ...)

The bad

- Classic proof size
- Verkle still does better if you use zk proofs
  - 500kb + public input size vs. 400kb
  - Double the size if accesses are considered public
- Tree is significantly larger
  - Some tricks might be used to smooth that out, but haven’t been tested yet
- Doesn’t solve the “sync race” issue
  - Unless witness contains state diffs

Section 4

Should we skip to the endgame?

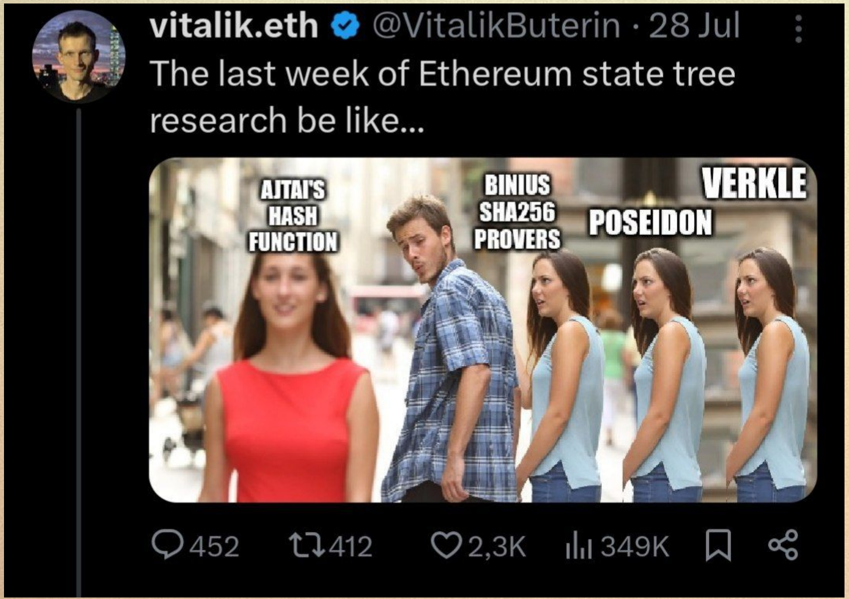

The Verge ✅
The past
Most serious EVM DoS issues solved
Basic light client support (sync committees)
Verkle tree spec + impl
Code chunking + gas cost updates
Transition spec + impl
SNARK / STARK ASICs
Verkle trees ✓
SNARK-based light clients
SNARK for L1 EVM
SNARK for Verkle proofs
SNARK for consensus state transition
Fully SNARKed Ethereum ✓
Increase L1 gas limits
Move to quantum-safe SNARKs (eg. STARKs)
The future

Goal: **verifying** blocks should be super easy - download N bytes of data, perform a few basic computations, verify a SNARK and you're done

Assumptions

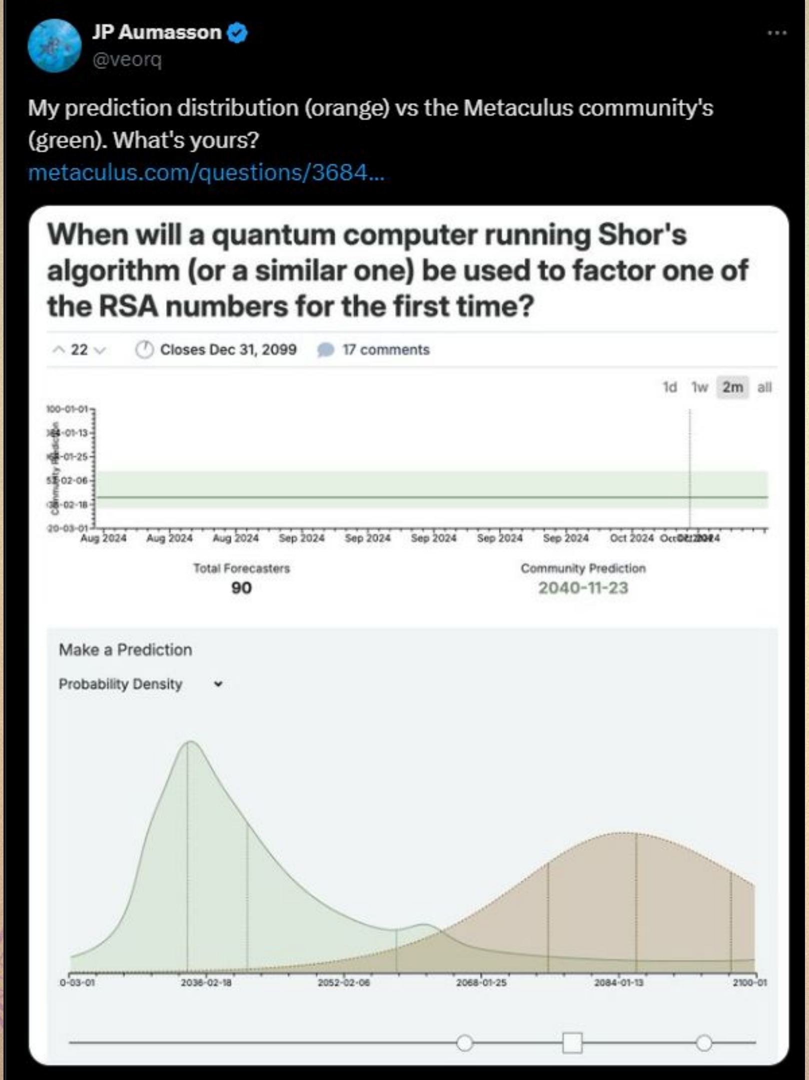
- The endgame will remain the same
- No other tree update will ever be necessary after binary trees
    - No further, better crypto primitives
    - Sync is never going to be a problem
        - Some form of snap-sync is possible
- No quantum-resistant, zk friendly verkle tree design is found
    - Lattice-based commitments
    - Anything polynomial based would work
- Poseidon is secure
    - Otherwise, zk performance is still far off
- Solo proof-building is possible, in an acceptable time and on accessible hardware
- Fullnodes remain easy to maintain
    - Ideally, become easier to maintain
- No new, yet unknown zk framework is found
    - Binius, circle STARKs, GKR... it keeps coming, some might work better with verkle
- Quantum computing risks realize
    - Exactly the way we expect them to realize

Better ideas keep coming


vitalik.eth @VitalikButerin · 28 Jul

The last week of Ethereum state tree research be like...

- Poseidon is secure
  - Otherwise, zk performance is still far off
- Solo proof-building is possible, in an acceptable time and on accessible hardware
- Fullnodes remain easy to maintain
  - Ideally, become easier to maintain
- No new, yet unknown zk framework is found
  - Binius, circle STARKs, GKR... it keeps coming, some might work better with verkle
- Quantum computing risks realize
  - Exactly the way we expect them to realize



JP Aumasson
@veorq

My prediction distribution (orange) vs the Metaculus community's (green). What's yours?
metaculus.com/questions/3684...

When will a quantum computer running Shor's algorithm (or a similar one) be used to factor one of the RSA numbers for the first time?

22 · Closes Dec 31, 2099 · 17 comments

Total Forecasters 90

Community Prediction 2040-11-23

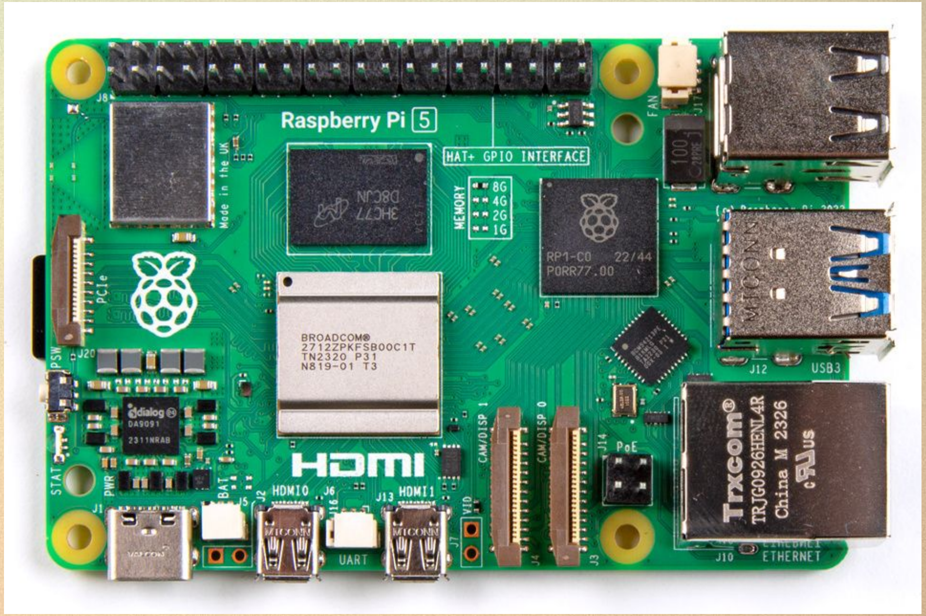
Make a Prediction

Probability Density

- Poseidon is secure
  - Otherwise, zk performance is still far off
- Solo proof-building is possible, in an acceptable time and on accessible hardware
- Fullnodes remain easy to maintain
  - Ideally, become easier to maintain
- No new, yet unknown zk framework is found
  - Binius, circle STARKs, GKR… it keeps coming, some might work better with verkle
- Quantum computing risks realize
  - Exactly the way we expect them to realize

Open questions

- Structure of the binary tree
- Structure of a zk witness
    - Can it be reused for the sync?
- Hash function
    - Is Poseidon secure?
- ZK proving systems
    - Binius? Circle starks? GKR?
- As a community, do we want to keep solo staking a viable option?
    - What are the “minimum hardware requirements” of Ethereum?
    - What are the “minimum bandwidth requirements” of Ethereum?



Uncertainty = delays

STATELESS NOW
STATELESS SOME DAY
imgflip.com

Done is better than perfect

What can be reused?

EIP-6800	🧩	Tree structure
EIP-4762	✅	Stateless gas costs
EIP-7748	✅	State conversion
EIP-7709	✅	Historical root contract
EIP-7736	🧩	State expiry

Thank you!

Guillaume Ballet

https://verkle.info

https://kaustinen-testnet.ethpandaops.io

@gballet