stateless block verification

Ethereum Sharding Research

Jeongho Jeon <maczniak@gmail.com>

July 9, 2018

Whitepaper Foundation, Nonce
(for internal discussion purposes only)

Whitepaper
Foundation

Outline

Stateless Client Proposal

How to Implement It

Let’s Extend It Further

Stateless Client Proposal

Whitepaper
Foundation

Stateless Client

Vitalik proposed the stateless client idea in Oct 2017.[1] It had been discussed until Feb 2018.

Stateless clients store only headers like Bitcoin SPV, but should verify transactions by its own. For stateless clients, the transaction senders (or miners) bundle a witness with a transaction. The witness contains state Merkle trees of affected accounts.

[1] https://ethresear.ch/t/the-stateless-client-concept/172

Advantages

- Full nodes in general no longer need to store any states.
- Fast sync can be done within a few seconds.
- Discussions about rent and storage cost disappear.
- Disk I/O that was the primary target of DoS attacks decreases.
- It supersedes the by-account parallelization (EIP-648, that includes affected accounts in a transaction).
- State-storing clients can prefetch account storage data from a disk in parallel.
- Fast reshuffling nodes across shards helps better security.

One of Ethereum's key advantages is the platform's ease of use, and the fact that users do not have to care about details like storing private state.

Who stores state?

- Nodes drop state randomly after 3 months. Data availability decreases over time.
- Nodes make contacts that pay to nodes that keep its state for a period of time, in the form of a state channel.
- DApps make web browsers store partial states in localstorage, and make it accessible by web3 API.

How to Implement It

Witness Optimization[2]

multi-tries divides a trie into 2^n tries by n-bit prefix.

partial statelessness stores up to n level.

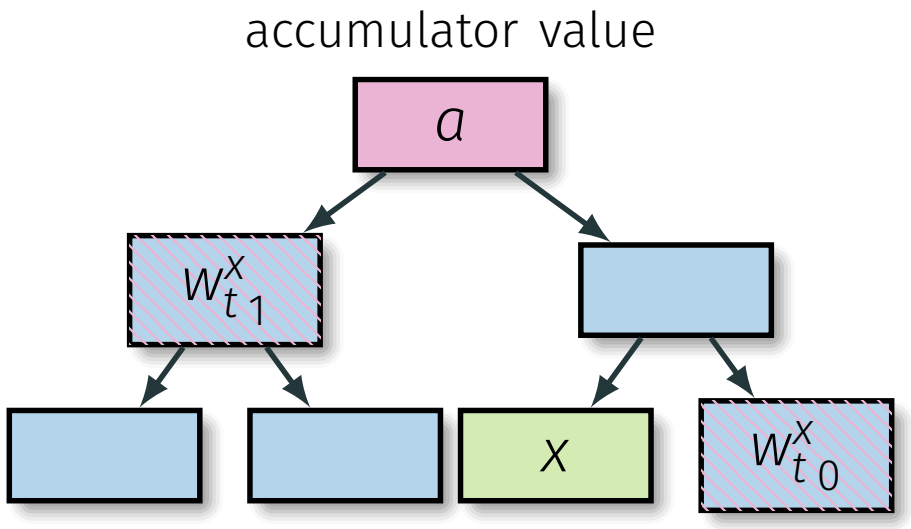
[2] https://ethresear.ch/t/multi-tries-vs-partial-statelessness/391

Accumulator

Accumulator is the function that aggregates recursive data structure (such as lists and trees) into a single value. It is also called as fold and reduce.

Cryptographic accumulator is the one-way membership function that could tell whether the given item is in the set without revealing set members. For example, Merkle tree is a cryptographic accumulator.

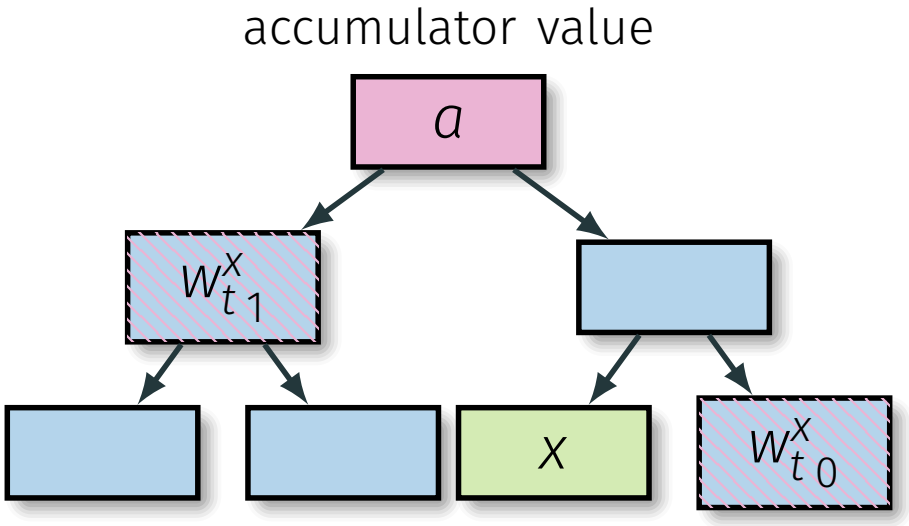
Merkle tree as an Accumulator



Merkle proof $\{(w^x_{t\,0}, right), (w^x_{t\,1}, left)\}$

$$a = h(w^x_{t\,1} || h(x || w^x_{t\,0}))$$

Merkle tree as an Accumulator



Each time you put a new item, accumulator value (Merkle root, a) and witness (Merkle proof, w^x_t) should be updated.

Accumulator Types

dynamic accumulator can remove members.

universal accumulator can tell whether the item is *not* in the set, too.

Let’s Reduce the State Size!

Ethereum uses Patricia Merkle trie accumulator now, and disregards non-state information (such as receipts) because it is unreachable from transactions (smart contracts).

But witness can be prepared by off-chain computation. Non-state information (history) can be useful in future.

If there are efficient accumulators for append-only data (history), we could make a contract use history than state, and adapt dual accumulator VM. History-driven Application could use very small state (actually one hash due to TrueBit or zero knowledge) or state only for temporary purposes.

Low update frequency accumulator[5]

It is called by many names: asynchronous accumulator[3], Merkle mountain range (MMR)[4], delayed (U)TXO commitment and so on.

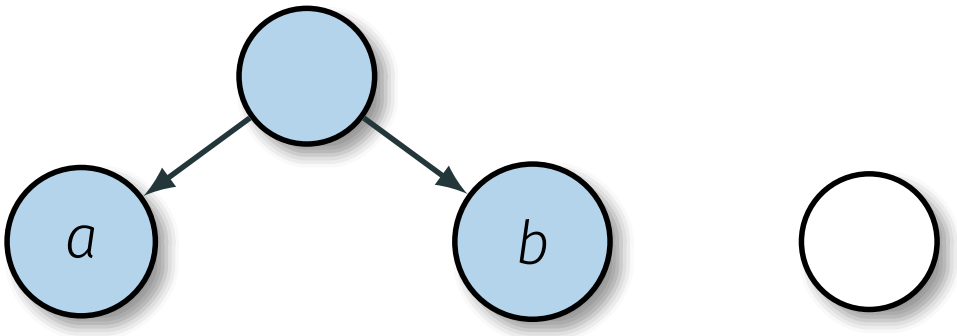
Even if accumulator value and witness are not synchronous, i.e., witness is older than accumulator value or accumulator value is older than witness, it can verify a member. Then updates can be delayed. It makes $log(n)$ times updates, and take up $log(n)$ times space (i.e., moutain summits).

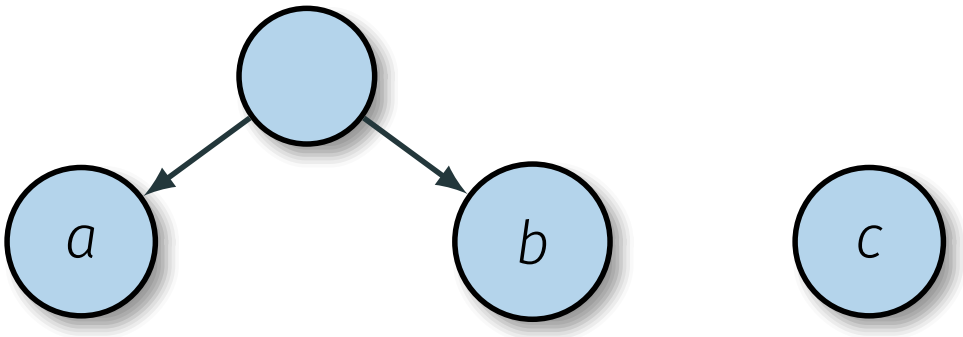
---

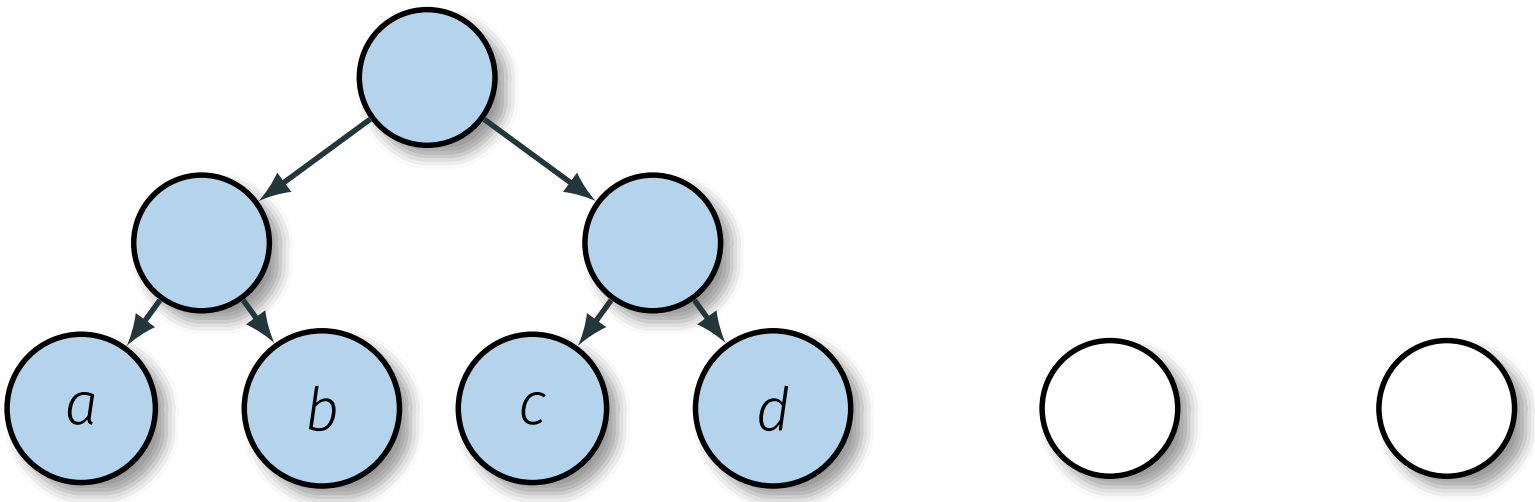
[3] https://eprint.iacr.org/2015/718.pdf

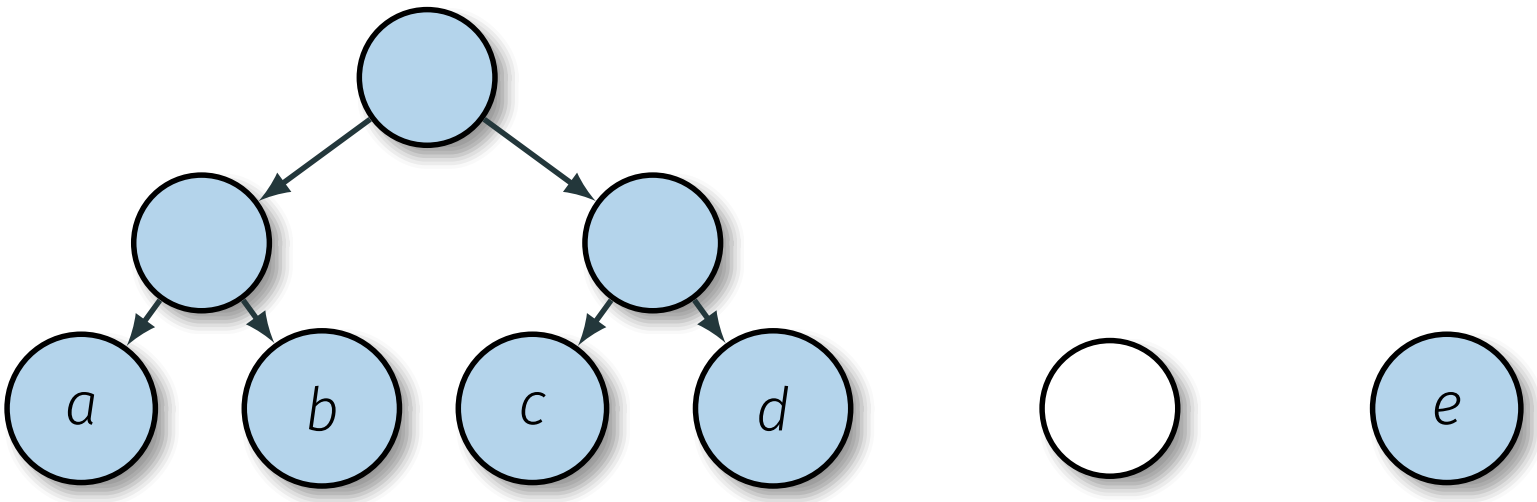
[4] https://petertodd.org/2016/delayed-txo-commitments

[5] https://ethresear.ch/t/history-state-and-asynchronous-accumulators-in-the-stateless-model/287


a
b


a
b
c


a
b
c
d


a
b
c
d
e

Double-Batched Merkle Log Accumulator

By stacking two MMRs, double-batched merkle log accumulator seems to be best candidate.[6]

...(diagram here)...

[6] https://ethresear.ch/t/double-batched-merkle-log-accumulator/571

if you then use a model where users delay sending receipts as long as possible and only use receipts to create new receipts then you've basically reinvented the UTXO model.

Let’s Extend It Further

Dynamic Accumulator for State

We can make a dynamic accumulator by using an append-open accumulator.[7] We use `[add, o]` and `[del, o]` records instead of the sole items. It looks like an event sourcing database. Conflict proofs may be prevented by economic incentives.

[7] https://ethresear.ch/t/a-cryptoeconomic-accumulator-for-state-minimised-contracts/385

State Minimised Execution[9]

We can replace stateful shards with log shards[8]. On-chain validators will be replaced with off-chain executors such as TrueBit. Stateful executos make virtual transactions without witness possible.

[8] https://ethresear.ch/t/log-shards-and-emv-abstraction/747
[9] https://ethresear.ch/t/state-minimised-executions/748