State Execution Separation

2018. 7. 26

Geore Han
<kr8534@gmail.com>

Outline

- Delayed State Execution
- “Cryptoeconomic light client” Designs
- Revisit Cross-shard Communication
- (DoS Attack on Delayed State Execution)

What is Delayed State Execution

Current Ethereum blockchain, consensus on transaction ordering and state calculation are tightly coupled together

AS-IS; A block contains

- A set of transactions
- A post-state root

So that represents a claim about both what the current TX history and the state after executing this history

TO-BE;

- Consensus process happens only on transaction ordering
- then a *separate* process exists to incentivize calculating state roots

all times aware of the state roots at all block heights that it has already processed

What is Delayed State Execution

Current Ethereum blockchain, consensus on transaction ordering and state calculation are tightly coupled together

AS-IS; A block contains

- A set of transactions
- A p

So tha

what the current TX history and the state after executing this history

TO-BE;

- Consensus process happens only on transaction ordering

to

ots

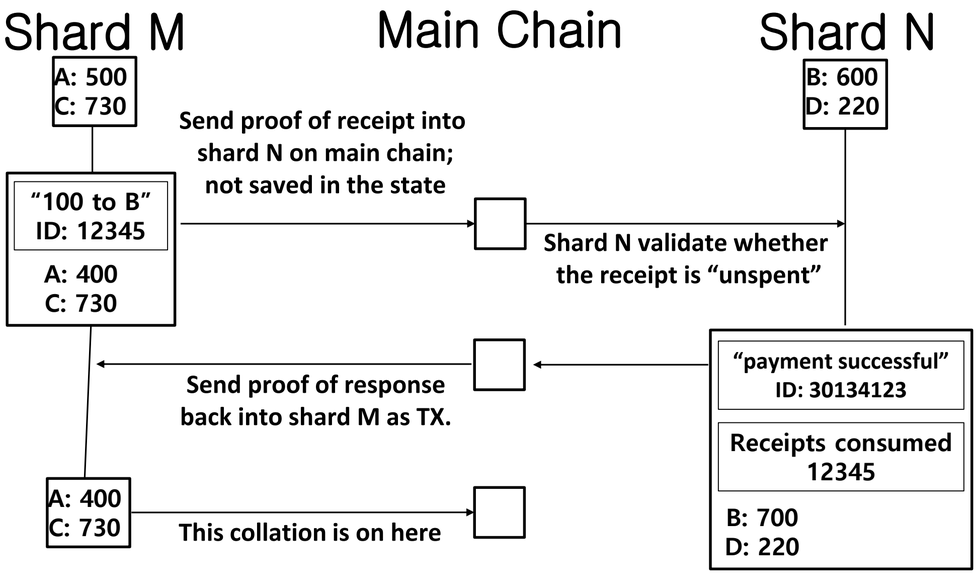
all times aware of the state roots at all block heights that it has already processed

State execution **is not** **the consensus game**

Some Old Research

- An executor submitting a claim (H, X, Y) would be incentivized as follows:
  - Reward of R if the block hash at height H (which represents the entire history up until that point) is X and the state root is Y
  - Large penalty of -D if the block hash at height H is X but the state root is NOT Y
  - No reward and no penalty if the block hash at height H is not X.

Back to the Cross-shard Communication



1. Operation on shard M creates a receipt on shard M
2. The receipt on shard M gets confirmed
3. An operation on shard N incorporates a proof that the receipt on shard M was confirmed, and perform execution based on this

- What if shard M has a large reorg?
  ➔ if N doesn't do reorg, consistency fail
  ➔ if N has a reorg, this could be a DoS attack point
  waiting shard M's finalization is non-sense!

- **Delayed stated execution!**
  ➔ Instead of reorg any tx on shard N after M's reorg, let the executors recalculate the state roots of shard M

Back to the Cross-shard Communication

- If shard A does a reorg ➔ don’t reorg any TXs on shard B, but rather let executors to recalc state roots
- it is possible to calculate ahead of time that some operation on shard B is not part of the dependency cone of something in shard A simply by looking at the access lists of transactions
- so users would have private knowledge that their operation on shard B is safe and sound without waiting for confirmation from the global state root

Back to the Cross-shard Communication

- If shard A does a reorg ➔ don't reorg any TXs on shard B, but rather let executors to recalc state roots

- it is possible to calculate ahead of time that some operation on shard B is not part of the dependency cone of something in shard A simply by loc

How *do* clients figure out what the state is?

- so shard B is safe and sound without waiting for confirmation from the global state root

A Simple “Cryptoeconomic light client” Design

- Allow anyone with ETH in any shard to deposit their ETH (with a 4 month lockup period), and at certain points (eg. once every Casper epoch) give depositors the ability to make claims about the state at some given height. These claims can be published into the blockchain
- Claim would be of the form [height, shard, state_root, signature]
- Given some reward proportional to the deposit (eg. corresponding to an interest rate of 5%), and a false claim means the claimer is penalized.

**“if too few nodes are super-full nodes,
it may not matter what happens to the deposits in the real state, because no one will know what the real state is”**

–Vitalik Buterin

A Simple “Cryptoeconomic light client” Design v2

- Still, even in the presence of a 90% attack on any single shard a client can still figure out the correct state
- Switch from claims being **[height, shard, state_root, signature]** to **[height, collation_hash, shard, state_root, signature]**
  - A claim that specifies the wrong collation_hash for a given height does not receive any reward or penalty
  - only a claim with the *correct hash* but the *wrong state root* is penalized

A Simple “Cryptoeconomic light client” Design v2

- It’s the client’s responsibility to determine what the correct chain is
- It means that state executors do not have to assume “reorg risk”; because their claims are now conditional on a particular history, and state execution is a deterministic process, they can only lose money if they deliberately provide the wrong state root
- it becomes safe to have very harsh penalties for claimers giving bad answers. Specifically, we can make the penalty be the entire deposit

A Simple “Cryptoeconomic light client” Design v3

- Switch claims to **[height, collation_hash, shard, prev_state_root, post_state_root, signature]**

- Imagine claims **as a graph**; state: vertex, collation_hash: edge, total deposit size of claims: weight ➔ from the genesis, follow a GHOST-like protocol

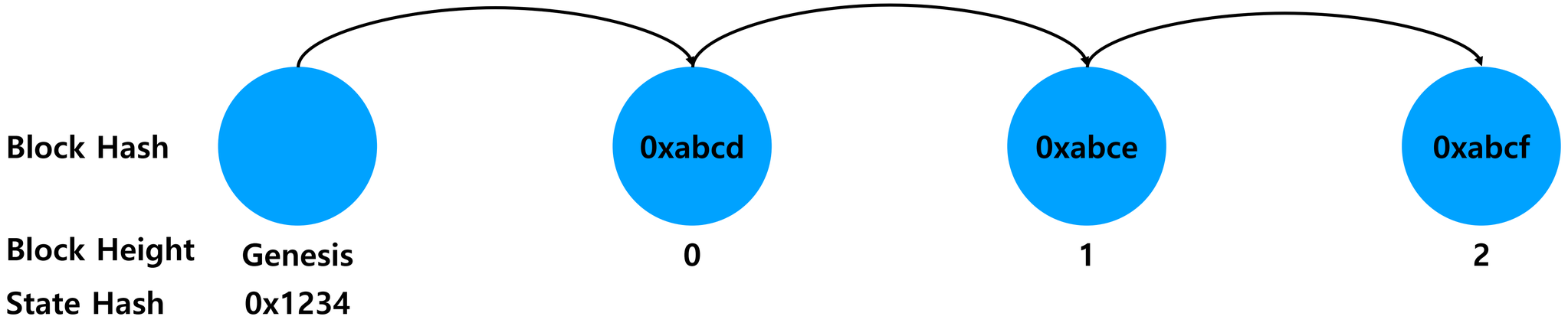
A Chain of Reasoning

- I know the pre-state root at height 0 is 0x1234 because it's the genesis

Claim A of 520000 ETH
Pre-state root: 0x1234
Block Hash of 0: 0xabcd
➔ State root at 0 is 0x5678

Claim B of 437000 ETH
Pre-state root: 0x5678
Block Hash of 1: 0xabce
➔ State root at 1 is 0x9012

Claim C of 619000 ETH
Pre-state root: 0x9012
Block Hash of 2: 0xabcf
➔ State root at 2 is 0x3456



- I know that either (i) the post-state root at height 2 is 0x3456 or (ii) at least 437000 ETH will be LOST

Revisit the Cross-shard Communication!

Cross-shard Communication

- Add *prior_meta_state_root* on previous claims where as meta_state_root means that the root hash of a Merkle tree of all the state roots (For make claims be conditional on the prior state of ***every*** shard)

  ➔ but may cause inefficiency

  ➔ Randomly sample so that a specific 1% of validators are allowed to make a claim during each block Have clients execute the collation themselves if > 10% of the validators that vote disagree with the majority

  ➔ Have individual claims be over a few consecutive epochs for allowing signatures to be reused

“I actually am fine with claims having to pay for gas.”

–Vitalik Buterin

“Execute-Order-Validate approach chosen by the Hyperledger Fabric platform (Linux Foundation).”

– kladkogex

References

- https://ethresear.ch/t/delayed-state-execution-finality-and-cross-chain-operations/987
- https://ethresear.ch/t/delayed-state-execution-in-practice/1041
- https://ethresear.ch/t/state-execution-scalability-and-cost-under-dos-attacks/1048