Ethereum sharding FAQs notes (Draft)

Source doc

阅读策略 BFS instead of DFS

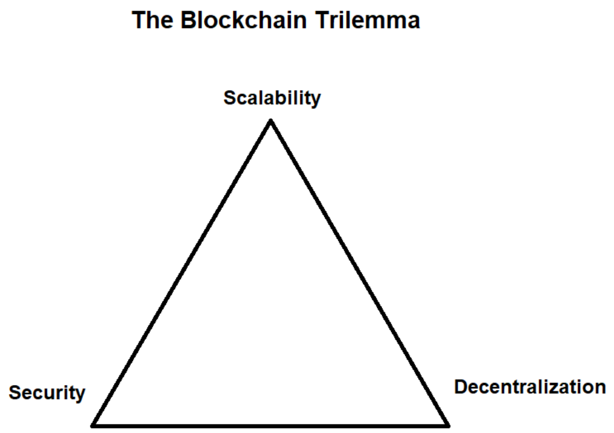
- Current tps: Bitcoin: 3~7; Etherum: 7~15

trivial but flawed solutions:

- N new blockchain & N altcoins; => N-factor decrease in security.
- bigger block; => only supercomputers can support => centralizaiton risk
- merge mining; ~ bigger block => more load to miner & less miner involved => centralization

scalability trilemma

- **Decentralization** (defined as the system being able to run in a scenario where each participant only has access to O(c) resources)
- **Scalability** (defined as being able to process O(n) > O(c) transactions)
- **Security** (defined as being secure against attackers with up to O(n) resources)



Modeling the scaling target

- **Metcalfe's law** gives O(n^2).
- But for large group: O(n*log(n)) for market cap & Txs given n users.

Other options than sharding

- In simple PoW blockchains, there are high centralization risks and the safety of consensus is weakened if capacity is increased to the point where more than about **5% of nodes' CPU time** is spent verifying blocks
- bitcoin-NG
  - compatible with other sharding solutions
- Channel-based strategies
  - lightning network, Raiden
  - scale Tx by constant factor; cannot scale storage;
  - DoS
  - off-chain scaling via channels can be used with on-chain scaling via sharding
- advanced cryptography (E.g. ZK-SNARK) => less initial synchronization
- Plasma
  - conducting off-chain transactions while relying on the underlying Ethereum blockchain to ground its security.
  - constant factor improvement
  - lock account attack
- State channel
  - tradeoff between versatility and speed of finality
- Proof of stake with Casper (used in layer 1) would also improve scaling—it is more decentralizable

Basic sharding solution

Basic design

- Each **shard** consists of proposers and collators; Each shard has a blockchain as collation.
- **Notaries** (in shard) check shards with periodic shuffling. Notaries votes on availability of data in collation, if w/o EVM. They execute, validate data, and vote, if with EVM.
- **Committee** to check the votes from notaries and generate new block for main chain with collation headers.
- A "main chain" processed by everyone still exists, but this main chain's role is limited to storing collation headers for all shards.
- Node type:
  - Super-full node: with data of all shards
  - Top-level node: store mian chain + light client for all shards
  - Single-shard node: top-level node + all blocks for some shards
  - Light node: only store block head of main chain

Potential flaws

- **Single-shard takeover attacks**: 51% attach in a shard; typically prevented by random sampling
- **State transition execution**: with random sampling, how to get authoritative up-to-date state info?
- **Fraud detection**: how to detect fraud, especially for light nodes?
- **Data availability problem** what if data is missing from one collation?
- **Cross-shard communication**: including cross-shard dependencies.
- **Superquadratic sharding**: what if top-level chain is still too big? shards of shards?

> CAP theorem does not make anything fundamentally harder.
>
> CAP: for a distributed data store system, at most two of three of following features can be achieved simultaneously.
>
> - consistency
> - availability
> - partition tolerance
>
> "in the cases that a network partition takes place, you have to choose either consistency or availability, you cannot have both".

Attacks and solutions

- Economical model for attack cost [here]
- **Single-shard takeover attacks**: random sampling
  - Notries are taken from the sample for that shard.
  - Reshuffling can be semi-frequently (e.g every 12 hours) or maximally (every block)
  - Explicit: select committees to vote / Implicitly: validate the longest chain
  - Random sampling with few nodes for verification is **as secure as** every node to do validation. (e.g size of sample: 150)
  - **Conclusion**: Because of the imperfections of sampling, the security threshold does decrease from 50% to ~30-40%, but this is still a surprisingly low loss of security for what may be a 100-1000x gain in scalability with no loss of decentralization.

Implementation Tricks

Random Sampling

- Using **PoS is easy** to do sampling.
  - Run the random function based on the stake.
  - Either run in-protocol algorithm to choose 150 validators for each shard
  - or each validator get random number to decide with shard.

- Using **PoW** is more **difficult**
  - Malignant node can keep running random function until it is assigned to a shard, which is desired to it.
  - "Stateful" PoW (using random number from last round to assign shard in this round) ==> **verifiable randomness**
- Is random sampling **random enough**?
  - For shard size **N = 150**, 1/3 of toal nodes are bad => 0.000183% 51% attack occurs. (if fully random & based on [binomial distribution](#))
  - Small shard size (< 100) is risky.
  - It cannot survive **bribing attacker** or **coordinated majority model**

Random sampling reshuffling frequency

- Too infrequent, adaptive adversaries can attack
- Too frequent, huge overhead for downloading whole states in new shard.
  - Control state size by deleting old accounts and/or limiting new account
- Downloading the whole **Ethereum state snapshot** take 2~8 hours ==> reshuffling every few days.
- Client or interactive verification protocol to execute state ==> less overhead. (Validation is still done by nodes)

User-side state

???

Split data and execution

???

SNARK(zero knowledge proof)

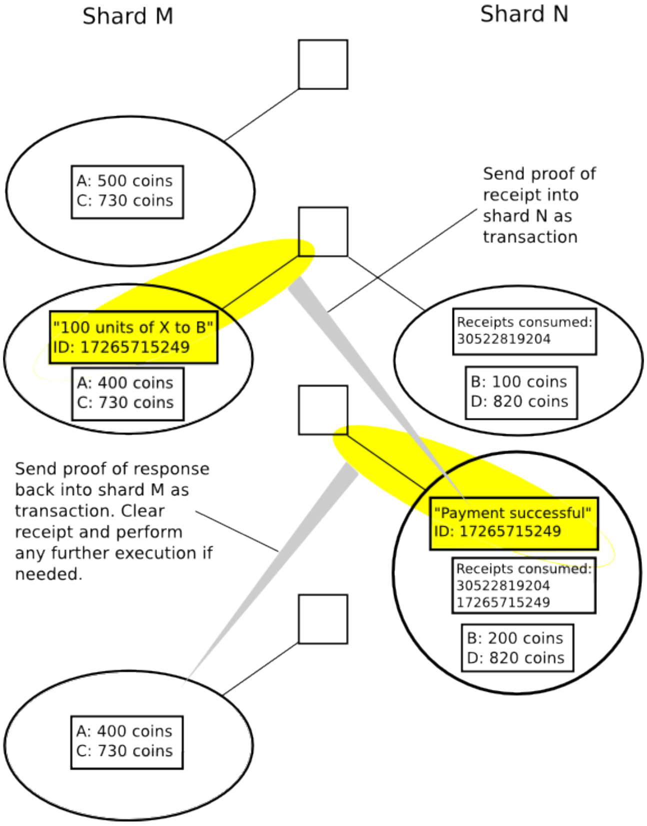
- Can be used in second level shard chain.

Cross Shard

Simpliest scenario

- Applications individually run on each shard, and there are few cross-shard communications;
- Use **receipts**
  - Shard M, where Tx is sent from, create receipt (not save on chain by verifible through Merkle

proof).
    - Wait for finalization.
    - Send Tx to shard N, where Tx is sent to. Create receipt and wait for finalizaiton.
    - Send response with receipt back to shard M.



    - Shard layers > 2 is much more complicated.

General scenario

- **cross-shard atomic synchronous transactions** with asynchronous messages only.
- E.g. train-and-hotel problem
- Lock shard A and shard B. Revert if Tx failed in one shard.
- Resolving deadlock
    - time-out; Slow!
    - [wound-wait mechanism](): Tx1 requests a data locked by Tx2. If timestamp(Tx1) < timestamp(Tx2) Tx1 forces Tx2 to revert; If timestampe(Tx1) > timestamp(Tx2) Tx1 wait.
- Question: how to ensure the cross-shard message from one shard is included in another shard within some fixed period of time ???

[Cross-shard contract yanking]()

- Yank one contract from one shard to another, so that cross-shard becomes intra-shard.

Attacks & solutions

- bribing attacker & 51% attack
    - interactive verification protocols
    - [Truebit]
- [Data availability problem]
    - The block creator broadcasts a block with insufficient date for validation. Data not available cannot directly prove the block creator is malignant, because of possible bad network connection. The block might pass validations.
- The attack by sending a lot cross-sharding Txs with high fee is unavoidable. "Transparent sharding" (automaticlly re-shard nodes based on usage) might help. (The design of transparent sharding is not mentioned in the source file)
- Attacker sends a cross-shard call from every shard into shard X at the same time?
    - Request to pre-purchase "congealed shard B gas" with penalties.
    - **Congealed gas**: fast demurrage rate: once ordered, it loses 1/k of its remaining potency every block.

Synchronous cross-shard messages

- A transaction may specify a set of shards that it can operate in.
- In order for the transaction to be effective, it must be included at **the same block height** in all of these shards ???
- Transactions within a block must be put in order of their hash (this ensures a canonical order of execution).
- Be careful to low-cost DoS attack

Semi-asynchronous messages

- The state keeps track of all operations that have been recently made and graph of dependencies.
- It can propagate the revert message twice as fast as other kind of messages ???
- The overhead has not been studied
- this is one of the more promising research directions for advanced sharding.

Guaranteed cross-shard call

- Cross-shard calls is not guaranteed.
    - tragedy-of-the-commons effects
    - volatile gas fee
    - possible timeout, no gas fee earned.

Heterogeneous sharding

- Not been essentially studied.
- Not in sharing road map.
- The main issue at hand is whether this interferes with random sampling from the notary registry and consequently allows coordinated attacks on shards.
- MicroChain of MOAC ???

Concepts mentioned && Questions

- **Verifiable Random Function(VRF)**
  - E.g. RANDAO; BLS aggregate signature; beacon chain for randomness
- Casper Proof of Stake ???
- Vlad Zamfir's presentation which talks about merge blocks (with introduction of Casper)
- The **deterministic threshold signature approach** works better in consistency-favoring contexts and other approaches work better in availability-favoring contexts ???
- User-side state ???
- Split data and execution ???
- Cross-shard contract yanking ???
- interactive verification protocols ???
- Truebit ???
- Data availability problem ???
- Synchronous cross-shard messages && semi-asynchronous messages && guaranteed cross-shard calls ???
- Congealed gas ???
- Guaranteed scheduling ???
- Over O(c^2)?