TLDR

: This post describes a possible “end game” for sharding. Assuming the existence of data availability proofs we can design a non-trivial sharding scheme that is “fork-free”, i.e. where every collation header added to the MVC has exactly one child. Fork-free sharding enjoys fantastic properties by construction:

1. Optimal head fetching
2. Optimal asynchronous (or synchronous) cross-shard communication
3. Security and finality elevated to that of the main shard

Background on consensus

As I see it there are three fundamental local impediments to global consensus. Each impediment manifests itself at a physics level and at a networking level:

1. Ordering

: What comes first?

In physics, time is relative and there is no absolute clock. In networking, routing speeds are unpredictable because topologies are inhomogeneous and transports are unreliable.

1. Availability

: What can be seen?

In physics, the speed of light is finite and we have black holes. In networking, communication has latency and partitions happen.

1. Validity

: What is real?

In physics, we have Heisenberg’s uncertainty and other quantum effects. In networking, software and hardware stacks are inhomogeneous and buggy.

In distributed consensus the traditional approach to overcoming local impediments to consensus is fork-choice rules. In a quantum-like fashion, we accept simultaneous candidate realities to co-exist as parallel forks, and global consensus emerges/collapses from fork-choice rules and local “observation”. Unfortunately following fork-choice rules comes at a cost:

1. Evaluation cost

: Evaluating forks for availability may require downloading lots of data, and evaluating forks for validity may require executing many transactions. With fork-choice rules, evaluation costs are technically unbounded in paralellism, and this is a vector for DoS attacks (at least until we have quantum computers and quantum networking). Evaluation costs in a fork-free setting are bounded in the classical computing setting.

1. Forking risk

: There is opportunity cost following one fork versus another, i.e. a risk in choosing the wrong fork. For example, PoW mining on a wrong fork is wasting electricity. With a single chain you are always on the right chain, so opportunity costs as a miner/validator/executor goes away.

1. Impedance to security, speed and finality

: Having the possibility to follow the wrong fork dilutes security of consensus, wastes proposals that don’t advance state, and/or impedes finality.

In the current sharding scheme we have to pay the price of fork-choice rules twice

: once for the main shard, and once within the VMC. (When we separate ordering from execution, we may actually have three

nested levels of fork-choice rules. One providing ordering to the VMC, another providing availability, and yet another providing validity.)

Below we give a sharding scheme where ordering, availability and validity is guaranteed by construction by the VMC from the collation headers, i.e. there is no fork-choice rule within the VMC.

Construction

The VMC already provides ordering. That is, given two collation headers added to the VMC, we know which was added first. For validity, we know of several ways to have valid-by-construction collations:

1. Put a SNARK/STARK in the collation header which proves that the collation root corresponds to a valid state transition, and have the VMC's addHeader

method check the proof.

1. Use log shards where any (hashable) blob of data can be parsed as a bounded list of logs (and/or transactions where garbage/invalid transactions are no-ops) and push state execution to a second-layer protocol, or to the application layer.

For data availability—and this is today's key missing ingredient—we assume the existence of succinct and efficiently computable/verifiable proofs which the VMC checks.

Discussion

With the VMC gating collation headers to guarantee ordering, availability and validity of collation bodies, the logic for head fetching collapses to just "get the last collation header from the VMC". Cross-shard communication is significantly easier because there is no re-org risk, as least not from the point of view of the VMC. The main thing that needs to be provided is replay protection in case the main shard reorgs, and this is provided by construction with stateless client witnesses.

The security of collation bodies elevates to the security of collation headers in the VMC. In particular, we have tight coupling, i.e. breaking a single shard means breaking the VMC, which means breaking the main shard. The time-to-finality of child shards also matches that of the main shard. So in a setup where the main shard has full Casper PoS with finality in minutes, child shards also enjoy finality in minutes.

As a side note for choosing the log shard approach vs the SNARK/STARK approach for validity, notice that the SNARK/STARK approaches is much stricter by disallowing "shard forks" below the VMC level. The state execution rules are frozen in the VMC until it is updated/replaced/forked, which requires intervention in the main shard. When execution is decoupled from availability (e.g. the log shard approach) forking a shard's execution rules can be done independently at the shard level.