TLDR

: We propose a sharding strategy where collation headers are kept offchain, i.e. off the main chain. Potential benefits include:

1. No censorship of collation headers by main chain miners or validators
2. No gas fees for inclusion of collation headers, avoiding stalled shards if main chain gas prices skyrocket
3. Faster periods (e.g. 5-second periods) compared to the currently proposed 75-second periods
4. Lower variance periods compared to the main chain's high-variance block times
5. No collation header processing bottleneck, allowing for more shards
6. Shard upgradability without main chain forking
7. Cheaper protocol-layer signature abstraction
8. Faster finality (~7.5 seconds on average) compared to ~2 weeks pre-Casper and ~20 minutes post-Casper
9. Competing sharding implementations running off the main chain

For simplicity of exposition we use Dfinity chains as described in their consensus whitepaper, although other constructions may be more suited. Special thanks to @karl for collaborating on this.

Construction

The SMC in the main chain allows for notaries to post fixed-size ETH deposits. We assume 2/3 of the notaries are honest, and that the main chain reaches finality in MAIN_CHAIN_FINALITY := 40320

block times (~2 weeks).

We now instantiate a Dfinity chain called the "beacon chain" that sits below the main chain. The beacon chain provides a high-grade random beacon with a low-variance period length of 5 seconds. The beacon chain processes all transactions necessary for the random beacon, in particular the BLS Distributed Key Generation (DKG) transactions. The beacon chain processes no user transactions.

The beacon chain and the main chain are synchronised via beacon checkpoints posted on the main chain. A beacon checkpoint is a beacon output that occurs every ~10 minutes, i.e. for which the period number is a multiple of BEACON_CHECKPOINT_GRANULARITY := 120

. Not all checkpoints have to be included in the main chain. We want chronological ordering of the beacon checkpoints which can be enforced by the SMC or via client-side filtering. Notice the beacon output chain is forkfree so beacon checkpoints don't roll back.

Synchronisation allows for unambiguous two-way light-clienting:

- Main chain light-client for the beacon chain

: At period p

the beacon chain finds the most recent beacon checkpoint in the main chain with period at most p - MAIN_CHAIN_FINALITY * 3

and uses the post-state root of the block containing that checkpoint for light-clienting. In particular, the beacon chain uses the set of deposits as specified by the main chain light-client, which is final and well-defined at every period of the beacon chain.

- Beacon chain (and shards) light-client for the main chain

: At block b

the main chain finds the second most recent beacon checkpoint included in the main chain. This checkpoint is at least 10 minutes old which is enough for finality of the beacon chain (and the shards). Indeed, assuming the worst case where an attacker has 1/3 of the deposits and he always successfully causes the committees to sign off on equivocated proposals at every height then the attacker can cause competing forks to last 10 minutes with probability 1/3^{120} < 10^{-57}

.

We now instantiate the shards below the beacon chain. Those are Dfinity chains which use the random beacon from the beacon chain. In other words, a committee of size 423 (honest majority with high probability) is randomly sampled for every collation at every period and for every shard, along with ranked proposers. The collations are notarised for availability via

quorums of size 212.

Discussion

The beacon chain acts as a “synchronised sidechain” to the main chain. Separation of concerns (deposits management vs beacon management) reduces load on the main chain and allows for upgradability of the sharding infrastructure (beacon chain and shards) without forking the main chain. More importantly, the beacon chain and the shards operate “unchained” (pun unintended

) from the limitations of the main chain, especially with regards to censorship, gas price, period length and period variance. This is somewhat reminiscent of plasma chains, but without the exit mechanism.

Interestingly in this construction the shards have better finality than the main chain despite shard security deriving from that of the main chain. One reason is that the shards only care about deposit snapshots from the main chain’s finalised past. This is somewhat reminiscent of state channels being able to provide instant game theoretic finality via deposits.

An important consideration is the bottlenecks of the sharding phase 1 draft spec likely to be retired, of which there are three:

1. Collation header processing

: Processing of collation headers is now distributed across shards instead of being centralised in a single chain. This bottleneck goes away here, allowing for 1000 shards instead of 100.

1. Light-client access

: Every shard has (delayed) light-client access to every other shard. With 200 bytes headers, 1000 shards and 5-second periods the bandwidth overhead is 200*1000/5 = 40kB/s. The double-batched Merkle accumulator imposes a storage overhead of ~1MB for light-client access across all shards. This bottleneck is not too constraining.

1. Notary bandwidth

: Assuming 10,000 notaries, 1000 shards, 50kB collation bodies, and 5-second periods we get a notary bandwidth of 423kB/s. This is the only significant bottleneck for going above 1000 shards, and it is potentially addressable with erasure codes.

As a final remark, notice the main chain can provide basic infrastructure for different sharding implementations running in parallel. This basic infrastructure could be deposits, a RNG scheme, a commitment scheme, a challenge-response game, checkpoints, etc. This makes the main chain a uniquely well suited platform for the bootstrapping, experimentation and competition of sharding designs.