What is Sharding? Guide to this Ethereum Scaling Concept Explained

As the scaling debate in cryptocurrencies continues, some potential solutions have actually been in development for quite some time now. Specifically, in the case of Ethereum, where a large focus is placed on decentralization and security at the expense of scalability, the application of sharding in conjunction with implementing Proof of Stake consensus is seen as the much needed mechanism through which the network can scale to practical levels for applications while still retaining its decentralization and security.

Sharding is a complex topic, especially when applied to a decentralized, peer to peer network such as Ethereum where the global state of the network constantly is updated. So what exactly is sharding and how can it help blockchain networks to scale?

Sharding and Distributed Computing Background

Sharding is actually much older than blockchain technology and has been implemented in a variety of systems from business database optimizations to Google’s global Spanner database. Essentially, sharding is a particular method for horizontally partitioning data within a database. More generally, the database is broken into little pieces called “shards”, that when aggregated together form the original database.

In distributed blockchain networks, the network consists of a series of nodes connected in a peer to peer format, with no central authority. As is the case with current blockchain systems, each node stores all states of the network and processes all of the transactions. While this provides the high level security through decentralization, especially in Proof of Work systems such as Bitcoin and Ethereum, it leads to legitimate scaling problems.

Using Ethereum as an example, a full node in the Ethereum network stores the entire state of the blockchain, including account balances, storage, and contract code. Unfortunately, as the network increases in size at an exponential pace, the consensus only increases linearly. This limitation is due to the communication needed between the nodes needed to reach consensus.

Nodes in the network do not have special privileges and every node in the network stores and processes every transaction. As a result, in a network the size of Ethereum’s, issues such as high gas costs and longer transaction confirmation times become noticeable problems when the network is strained. The network is only as fast as the individual nodes rather than the sum of its parts.

Sharding helps to alleviate these issues by providing an interesting, yet complex solution. The concept involves grouping subsets of nodes into shards which in turn process transactions specific to that shard. It allows the system to process many transactions in parallel, thus significantly increasing throughput.

A simpler way to put it would to be imagining the division of the United States into states. While each state (a shard in this case) is part of the larger United States (Ethereum network), they have their own specific rules, boundaries, and subsets of populations. However, they do share a universal language and culture as part of their larger network that makes up the country.

Or even better, in Vitalik Buterin’s own words:

“Imagine that Ethereum has been split into thousands of islands. Each island can do its own thing. Each of the islands has its own unique features and everyone belonging on that island i.e., the accounts, can interact with each other AND they can freely indulge in all its features. If they want to contact other islands, they will have to use some sort of protocol.”

As you can see, the concept of fragmenting the network into more efficient pieces allows the network to function as the sum of its parts, rather than being limited by the speed of each individual node.

How Does Sharding Work in Blockchains?

We will continue to use Ethereum as an example in this as it is the most well-known and arduous sharding attempts in the blockchain arena, as the Ethereum developers are implementing what is known as “state sharding”.

The current state of the Ethereum blockchain is known as the “global state” and is what everyone can see when they look at the blockchain at a specific instance. The tricky part in implementing sharding in Ethereum is that by sharding the nodes into smaller subsets, these subsets need to be able to process specific sets of transactions while simultaneously updating the state of the network, all while ensuring it is valid.

Sharding in Ethereum is supposed to be implemented in a two phase rollout, more than likely after Proof of Stake is implemented in the network. Phase one will be the data layer consisting of the consensus of what data is in the shards. Phase two is the state layer. All of this is very fluid, so a general breakdown of how it may work is below.

Ethereum breaks down the network into specific shards. Each shard is assigned a specific group of transactions that is determined by grouping specific accounts (including smart contracts) into a shard. Each transaction group has a header and a body that consist of the following.

Header

The shard ID of the transaction group

Assignment of validators through random sampling (verify the transactions in the shard)

State Root (state of the merkle root of the shard before and after transactions added)

Body

All of the transactions that belong to the transaction group that are part of the specific shard.

Transactions are specific to each shard and occur between accounts native to that shard. When transactions are verified, the state of the network changes and account balances, storage, etc are updated. In order for the transaction group to verify as valid, the pre-state root of the transaction group must match the shard root in the global state. If they match, the transaction group is validated and the global state is updated through the particular shard ID state root.

Instead of only containing a state root, each block of the Ethereum blockchain now contains both a state root and the transaction group root. The transaction group root is the merkle root of all of the transaction groups from the specific shards for that block of transactions. Basically, there is a merkle root of all of the different shards that contain the updated and verified transaction groups. This root is stored in the blockchain along with the updated state root.

The employment of merkle tree concepts in this structure is vital to ensuring validity of the blockchain. Understanding how a merkle tree and specifically a merkle root work, can help you to grasp these concepts much more easily.

Consensus within a shard is reached through a Proof of Stake consensus of randomly selected nodes that are applied to a shard for specific consensus round. This not only provides finality to the consensus, which is necessary within the shards, but also provides a particular defense to an attack that a Proof of Work blockchain would be susceptible to in this instance.

The hash power required to overrun a specific shard in a PoW sharded network is drastically reduced and the ability for a malicious actor to take over a shard through computational power is feasible. Through this, the bad actor could attack other shards through the communication protocol which is one of the more complicated and important features of sharding architecture. Random sampling selection of the validators within a shard manages to stifle this type of attack since a bad actor will not know which shard they are being placed in before they are actually placed in it. Further, random sampling will be used to select the validators that are actually validating from that random validating set.

The communication protocol is vital to the sharding architecture functioning correctly in the system. You can think of the communication protocol as the universal language that is consistent among the states as part of the larger United States. However, designing this protocol is highly challenging and needs to be performed so that it is only used when necessary. It becomes necessary when a specific node requires information that is not stored within its own shard and needs to find the shard with the requisite information. This communication is known as cross-shard communication.

The cross-shard communication is achieved through applying the concept of transaction receipts. The receipt for a transaction is stored in a merkle root that can be easily verified but that is not part of the state root. The shard receiving a transaction from another shard checks the merkle root to ensure that the receipt has not been spent. Essentially, the receipts are stored in a shared memory that can be verified by other shards, but not altered. Therefore, through a distributed storage of receipts, shards are able to communicate with each other.

Sharding Moving Forward

Sharding in Ethereum is expected to be implemented after the Casper PoS upgrade. Recently, there have been some developments regarding Ethereum 2.0 which involve implementing both Casper and sharding. Casper seems to be slated for 2019 while sharding will follow in 2020 or 2021.

Sharding has also been implemented in a few other platforms, most notably Zilliqa. However, Zilliqa does not implement state sharding at this time and instead focuses on providing a high throughput blockchain by utilizing transaction and computational sharding.

Conclusion

Sharding serves to offer some promising solutions to the elephant in the room of blockchain platforms right now, scalability. While Bitcoin’s lightning network is in the testing phase and has been showing some very promising progress so far, Ethereum’s solution brings with it some unique challenges as it is pegged as a world computer that is Turing complete.

Sharding will directly work only at the protocol level, so to the end user or dapp developer it may not be necessarily relevant to learn about. Regardless, Ethereum’s attempt at state sharding for a vast, decentralized network is an impressive endeavor and will be an enormous feat of accomplishment if successfully implemented.