SCDO Blockchain Sharding Introduction

Why do we need sharding?

As we all know, in 2018, Ethereum's CryptoKitties game become very popular among a large number of users, but a lot of transactions created in a short period also caused a large area of congestion on the Ethereum network, making many transactions have to enter the waiting queue. Some transactions even needed to wait for 20-30 hours. This incident had blocked the entire Ethereum network for several days and let us know that in order to support various applications, the blockchain network must be able to and need to scale up.

Regarding Bitcoin, a new block is generated every ten minutes and there are no more than seven transactions that can be processed per second. In fact, it takes about 6 block length for one block to be confirmed. Therefore, the actual transaction delay is around 1 hour.

Compared with Visa and MasterCard, it is obvious that the transaction processing speed of Bitcoin and Ethereum is far from enough for daily usage. Not to mention that there is an increasing demand of various blockchain applications, the requirements of speed and throughput of one blockchain increase rapidly.

But in reality, these major public blockchains are facing the embarrassing situation of either unable to reach an expansion consensus or unable to upgrade the existing network to scale up.

A simple and straightforward way to solve the blockchain throughput and speed may be vertical expansion: increasing the block size, or in the case of Ethereum, increasing the upper limit of the use of GAS. Although these are the correct methods in theory, there are some other issues brought up as well. First, increasing the GAS limit inevitably increases the uncle block rate, that is, adding more temporary blocks to the blockchain. At present, the uncle block rate of Ethereum has already reached about 30%. Constantly increasing the GAS limit may cause higher uncle block rate and further damage to the security of the Ethereum network, because users may affect each other due to their transactions that will be rejected later; Second, since data needs to be propagated and verified in the P2P network, increasing the block size will increase the pressure of message communication and data synchronization per unit time. Moreover, the more the block size increases, the more transactions can be packed in one block. The superpower nodes in the network may have more advantages than nodes on transaction packing, which will damage the decentralization of the blockchain and weaken its security. So, using the method of increasing block size to increase the throughput is a double-edged sword. Its effect may be not good as expected;

So how can we achieve the expansion of the blockchain without increasing the pressure on the network and weakening the network security?

Let us take a look at the basic components which may affect blockchain performance: network communication, message encryption and decryption, consensus mechanism, transaction verification mechanism and so on. In terms of consensus mechanism, its goal is to allow the nodes in the network to agree on a block or a transaction with the same protocol, but reaching a consensus in a highly decentralized system itself is a time-consuming task. If we consider there may be some malicious nodes, this will further increase the processing complexity and time.

Therefore, some consortium chains have introduced some weakened consensus algorithms from the perspective of performance. For example, the consensus algorithm used by Fabric is “endorsement”, but it can only be used to handle recovery faults such as machine down, but cannot handle malicious behavior of nodes. This method has certain feasibility in the consortium chain, because the consortium chain has some entry requirements and all members are assumed to be accountable. However, for the open-source public chain platforms, it brings great challenges to data security. Another approach is to use a “voting” consensus mechanism to limit the consensus participation to a small number of elected nodes. This for sure will increase the speed and efficiency of consensus reaching, but it also impairs the decentralization and cannot prevent malicious nodes from collaborating with fraud.

Another possible solution is to improve scalability through off-chain methods (also called layer 2 expansion), which mainly include: isolation verification, lightning network, RSK side chain, layering. These methods mainly assign part of the tasks to off-chain or other chains for processing, or divide the entire transactions into several parts and process them separately, and only interact with the main chain when they have to. Although these methods can increase the capacity of the network, they are not strict on-chain expansion since all the processes are completed off-chain.

As more and more complaints about the slow network of Ethereum from the developer community, Vitalik Buterin once stated in a tweet that the most important principle of Ethereum's sharding should be "to be as close as possible to the same properties as a single blockchain". However, Ethereum's sharding is not conducted on the main chain, but transferred to the second layer of the network. And its consensus algorithm has to use the "voting" mechanism and the sharding requires super nodes to handle the sharding logics, which greatly reduces the decentralization of the network. This may not be called as “consistent sharding”. Moreover, in order to make Ethereum looks like before, the sharding proposal and process is separately handled in another protocol layer and the sharding logics will not be exposed to developers. Such as, how to create shards, how to balance the states between shards or how to prevent the shards from being too small, etc. All of these will be done behind the scenes, this is not very friendly to open source projects.

In summary, none of the methods mentioned above are consistent expansion. They expand network either in the second layer network, off-chain, or sacrificing the security and decentralization of the main chain.

So how to consistently expand the network without damaging its security has become a challenge of the blockchain community. Good news is SCDO have successfully developed a secure and fast on-chain sharding protocol. Each shard has consistency, namely they are all identical. Even more the network can be expanded by simply increasing the configured number of shards. So far, with 4 shards, the transition speed can be easily as high as 1000 per second, which can be increased if needed.

Before introducing the SCDO sharding technology, let’s first talk about the sharding technology and its possible modes.

Principle of Sharding: divide and conquer

First, what is sharding technology?

Blockchain sharding technology is a kind of expansion technology similar to the traditional concept of database sharding which horizontally divides the database into multiple shards and places these shards on different servers. For the public blockchain, the transactions will be divided into different networks, and different nodes will process them parallelly. Therefore, each node only needs to process a small part of the transactions, and a large amount of verification work can be completed in parallel with other nodes in the network. So, splitting one network into partitioned networks will allow more transactions to be processed and verified at the same time. This is also called horizontal expansion.

You can imagine this: the existing blockchain is like a busy highway, and there is only one toll gate on this highway. This structure can easily lead to traffic jams because all cars have to wait in only one line to pass through the toll gate. Blockchain sharding is like adding several identical toll gates on this highway. This will greatly increase the speed of cars passing through the toll gates since several cars now can pass the toll gates simultaneously.

In brief, the blockchain sharding will have several benefits. First, the speed of transaction processing in the network can easily reach thousands per second or more, which will change people’s perceptions of the low efficiency of cryptocurrency as a payment method; Second, with increasing throughput, more applications can be deployed on the blockchain; Third, sharding technology can increase the number of blocks generated per unit time, so the reward of miners per unit time is multiplied, making mining more profitable and attracting more nodes to join to the network. Fourth, sharding technology can help reduce transaction costs, since the times of verifying a single transaction now is reduced. The combination of low transaction fees with fast transaction speed will make it more and more attractive to users to participate.

Although sharding technology can bring so many benefits to public blockchains, how can we implement it?

Sharding strategy

One blockchain has three core components: P2P network, transaction, state, based on which the sharding technology can be implemented: P2P network sharding, transaction sharding and state sharding.

Through network and transaction sharding, the nodes in the network are divided into different shards, and each shard network can form an independent processing subgroup and reach the consensus among its subgroup parallelly. But the communication between two shards is usually limited since the nodes in one shard do not have the data for the other shard.

On the other hand, for today’s mainstream public blockchains, in order to verify each transaction, full nodes have to store all transactions, smart contracts, and accounts states. This costs them more with more storage space required.

In order to solve this problem, a feasible method called state sharding has been proposed. The key to this technology is to separate the entire storage and allow each node to be only responsible for hosting data of its own shard, rather than storing the entire network state data.

Complexity of Sharding

Although these different forms of sharding may be very intuitive, the implementation of them may be not as simple as they seem to be. There are a lot of complexities and potential challenges for these methods, some of which are easy to overcome, while others are not. Generally speaking, P2P network sharding and transaction sharding are easier to implement, while state sharding is much more complicated. Below, for different sharding mechanisms, we will discuss some of the challenges they face and their feasibility.

P2P Network Sharding

The first and most important question of sharding is how to create shards. Developers need to develop a mechanism or protocol to determine which shard for one node should be assigned in a safe manner, to avoid attacks by those who control some specific shard.

The best way to solve this (at least in most cases) is to take advantage of a verifiable randomness based on which the protocol can randomly select nodes in the network to form shards. It means nodes themselves cannot predict which shard they will join until they are selected. So, this can prevent malicious nodes from over occupying one specific shard.

But how do we establish verifiable randomness? The easiest way is to use hash information stored in the block header itself, since we cannot predict the hash value but on the other hand we can verify it anytime, such as the **Merkle Tree Root Hash** of the **transactions**.

However, shard-shard interaction is still not solved.

Transaction Sharding

Let us first consider the transaction sharding (without smart contracts) in a system similar to Bitcoin, the state of the system is defined by UTXOs. We assume that the network is already composed of shards. One user sends out a transaction and each transaction has two inputs and one output. So, how can the transaction be assigned to a shard?

The most intuitive method is to determine the shard based on the sum of the last few bits of the transaction hash value. For example, if the sum of the last few bits of the hash value is 0, then the transaction will be allocated to the first shard, and if the value is 1, the shard is 2 and so on. This allows us to verify transactions in a single shard. However, if the user is malicious, he may create another transaction with two identical inputs but different outputs--yes, it is a double-spending transaction. The second transaction will have a different hash value, so these two transactions may be allocated to two different shards. Then, each shard will verify the received transaction separately without knowing the other double-spending transaction in the other shard.

In order to prevent the double-spending problem, the shards have to communicate with each other during the verification process. In fact, since double-spending transactions may appear in any shard, transaction receivers in a particular shard have to communicate with other shards. In fact, the cost needed for mutual communication may completely compensate for the benefits of sharding since this kind of mutual communication network among nodes require a fully connected graph.

However, if we have an account-based system (without smart contracts), the problem will become simpler. Each transaction will have a sender's address, and then the system can allocate it to a shard based on the sender's address. This ensures that two double-spending transactions will be verified in the same shard, so the system can easily detect double-spending transactions without cross-shard communication. But the handling of the recipient transaction is still a challenge.

State Sharding

State sharding may be the most challenging but the best technical solution for sharding so far.

Let us still use an account-based system (without considering smart contracts). In a state-sharded blockchain, a specific shard will only keep a part of all states.

Assume we have two shards and there are two user accounts (called as Selina and Cindy respectively) in each shard. Each shard has their own user's balances.

Suppose Selina want to pay Cindy some cryptocurrency and she created a transaction. This transaction will be processed and verified by the first shard. Once the transaction passes verification, balances of Selina's and Cindy's accounts will be updated and shared with the other shard. If there are a lot of cross-shard transactions between these two accounts, it may require frequent cross-shard communication and state exchanges. So, ensuring that cross-shard communication does not exceed the performance gain of state sharding is a subject that requires comprehensive research.

One possible way to reduce cross-shard communication cost is to restrict users from making cross-shard transactions. In our example above, it means not allow Selina to directly trade with Cindy. If Selina had to trade with Cindy, she would have to have an account in the other shard. Although this does eliminate cross-sharding communication, it does not really fulfill the original goal of sharding.

The second challenge of state sharding is the availability of data. We can consider a scenario where, for some reason, some specific shards are attacked and cause them offline. Since one shard does not replicate the entire state of the system, the network can no longer verify transactions that rely on offline shards. Therefore, the blockchain is basically unusable under such circumstances. The solution to this problem is to maintain data archives or perform node backups, which can help the system to repair faults and restore unavailable data. However, in this way, the node will have to store the entire state of the system, which increases the pressure of data propagation on the network.

Other important points that we need to consider in all sharding mechanisms (not specific to state sharding) are: 1) to ensure that sharding is resilient against attacks and failures; 2) the network must accept new nodes and allocate them in a random manner. In other words, the network may need to redistribute nodes within a period of time in order to get balanced.

However, in the case of state sharding, reallocating nodes is very tricky. Since each shard only keeps a part of all states, reallocating the network means not only the nodes of shards need to be reallocated, but the corresponding state data needs to be updated. This naturally brings additional data synchronization and network communication. And before the synchronization is completed, there may be problems that cause the entire system to fail. In order to prevent system interruption, we must gradually adjust the network to ensure that each shard still has enough data before all nodes are reallocated.

Similarly, once a new node joins a shard, the system must ensure that the node has enough time to synchronize with the shard states; otherwise, the node will completely reject every transaction.

In summary, blockchain sharding is an exciting technology. It gives us hope that it can solve the expansion problem without affecting decentralization and transparency. However, there is no doubt that sharding technology, especially state sharding, is very challenging at the design and implementation levels.

5. SCDO Sharding

SCDO's sharding technology combines the characteristics of P2P network sharding, transaction sharding, and state sharding: The SCDO network adopts the account balance model, and uses its innovative proof-of-work consensus algorithm (ZPoW). Sharding is implemented on-chain and each shard is completely replicated and absolutely equivalent to each other. Each shard does not need to store the complete information of other shards, and the shard communicates with each other at the minimum frequency to ensure the verification of cross-shard transactions. Moreover, the SCDO Sharding protocol ensures that the SCDO network can add more shards at a minimum cost when needed in the future to achieve higher expansion.

Sharding in the SCDO

The basic idea of sharding in SCDO mainnet is to partition the nodes and network into several shards. A transaction between two shards is called a cross-shard transaction. Because a cross-shard transaction involves accounts of two different shards, this transaction should be recorded in both shards. The transaction verification on the receiver’s side is a big challenge because validators in the receiver’s shard do not have the sender’s information at hand. They have to send requests to a remote peer to retrieve the desired information and carefully examine the responses. In SCDO mainnet, we use light chains as archives of the other shards, with respect to the shard of the local node. The validity of the light chains is examined based on the POW solutions, which means it is impossible to forge a light chain without computation. If the responses from remote peers are consistent with the light chains, we have a high level of confidence that the cross-shard transaction has been confirmed on the sender's side. But the maintenance of the light chains should not occupy too much resource such as CPU, storage and bandwidth. To this end, only the block headers are stored in the light chains. We will introduce more details in the next section.

At the setup stage, each node chooses a shard to join permanently. Nodes in the same shard share the full information including state, transaction history, etc., while nodes in different shard only share lightweight information such as block headers.

Typically, a node maintains a full chain associated with its shard and several light chains associated with other shards. See Figure 1 for an illustration. The full chain stores all the information of this shard while the light chains only store the block headers of other shards. While synchronizing a full chain, the local node downloads full blocks and verifies every transaction in the blocks. While synchronizing a light chain, only the block headers are downloaded and examined. The light chains serve as references for the full chain to verify cross-shard transactions.

Transactions within a shard are relatively straightforward as all the information needed is accessible in a full chain. The logic of cross-shard transactions is more complicated because each SCDO node only uses light chains to keep track of other shards.

Generally, the process of a cross-shard transaction can be divided into two phases. At Phase 1, the transaction is packed in a block by a miner in the sender's shard and the fund is deducted from the sender's account balance. After a certain amount of blocks are mined thereafter, we can say that the block with the cross-shard transaction is confirmed. Then miners create a debt and put it in the outgoing debt pool. The outgoing debt pool is a pool for the management of outgoing debts. Every once in a while, the outgoing debt pool checks the debts and sends them out to the nodes of other shards. Outgoing debt pool will remove the debts once it finds them get confirmed in the target shards.

At Phase 2, the debt reaches the receiver's shard and gets into an incoming debt pool. The miners in the receiver's shard send a request to the nodes of the sender's shard for the verification of the debt. If the nodes of the sender's shard confirm that the cross-shard transaction does exist, they send back the corresponding block header hash. Miners in the receiver's shard then search for the returned block header hash in their local light chain. If the search is successful, then miners confirm that the debt is valid and move it into a packing queue. Finally, miners pack the debts in the block from the queue and add the fund to the receiver's account balance.

After a certain confirmation period, the block validator checks in the receiver's shard whether the debt amount has been deducted from the sender's balance. If the debt is forged by a malicious user, the block verification program can easily find the debt before entering the debt pool.

Since cross-shard transactions are recorded on the sender and receiver's shards, there are two possible inconsistencies. The sender’s balance decreases, but the receiver’s balance remains the same, or the sender’s balance remains the same, but the receiver’s balance increases.

The first situation will not happen in the SCDO mainnet. As long as the transaction is recorded on the sender, when certain conditions are met, the nodes in the sender's shard will issue the corresponding debt again. Therefore, the receiver's shards always have the opportunity to record transactions. The second situation may hinder the expansion of the entire SCDO main network. But for this to happen, the attacker needs to create a malicious long fork chain in the sender's shards and try to replace the normal chain. This kind of attack is very difficult, because SCDO's ZPoW algorithm combines the distribution of probability, thereby making the distribution of block nodes more decentralized, and there is an additional mechanism in the SCDO main network to check the consistency of cross-shard transactions.

At present, SCDO uses 4 slices, and SCDO can reach a peak of about 1000 TPS, which can already meet the needs of many DApps. In addition, shards can be added to the SCDO mainnet in the future. In the future, SCDO will combine its sharding and sub-chains to achieve a more scalable system.

More Words

Blockchain sharding technology seems to be very promising, but is there no problem at all? Not really! Previously, attacking a network requires 51% computing power. Once the sharding is implemented, the entire network computing power is divided, so the malicious nodes can more easily attach some specific shard with less computing power, which is commonly **k**nown as **1% computing power attack.**

How can we solve this problem? In addition to increasing computing power, it is even more urgent to ensure the decentralization of each shard network. SCDO's high decentralized proof-of-work consensus algorithm is particularly important. At the same time, SCDO is actively researching differentiated **difficulty setting protocols** to increase the difficulty of consecutive block generation.