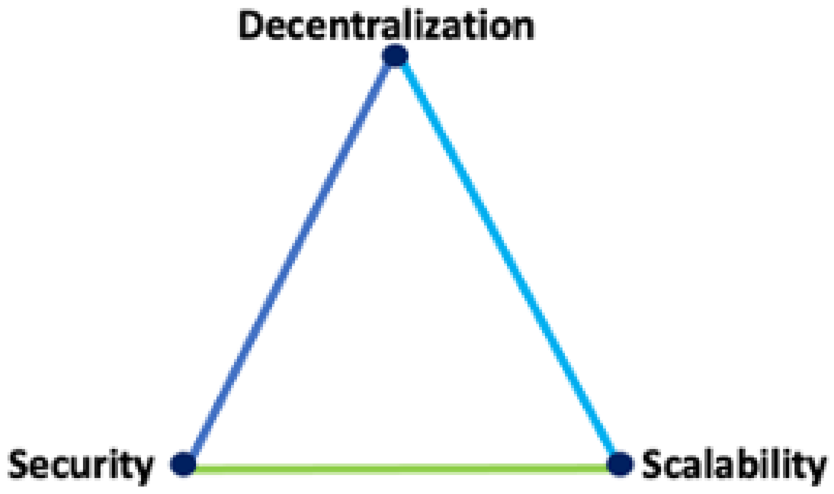
Introduction

Blockchain networks pose an issue that requires a choice between two of three essential components of a blockchain system; decentralization, security or scalability. This posed issue is known as the scalability trilemma. Sharding is a proposed architecture that aims to solve this issue. In this paper we will explore the scalability trilemma as well as how sharding architecture can be used as a solution to it.

What is the scalability trilemma?

The scalability trilemma occurs in blockchain networks due to the difficulty of compromising on one of three main characteristics; decentralization, security or scalability. It's argued that it's impossible to build a blockchain network with all three of these components -- making it especially difficult to *scale* blockchains.



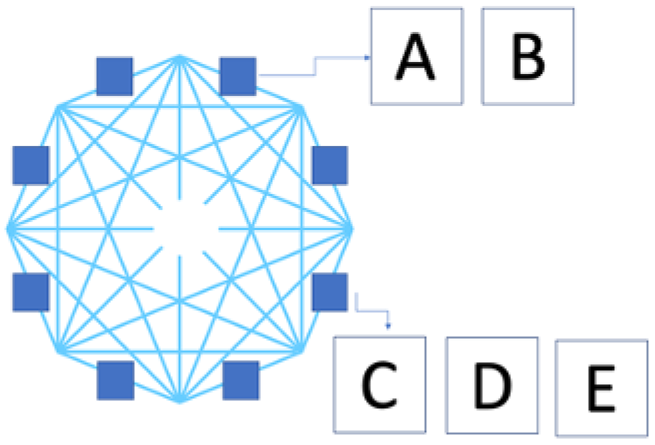
Decentralization is one of the blockchain's greatest strengths and can't be omitted from a blockchain's infrastructure. However, compromising on security will leave the network faulty and prone to risk and attack. While, compromising on scalability will mean network capabilities are limited with very little means of being able to grow beyond a few simple services. The scalability trilemma exists because the nature of decentralized and secure networks induces latency in transaction throughput, making the performance of blockchains subpar. When comparing the Bitcoin blockchain's transactions per minute to Visa's, we find Bitcoin's at 3 transactions/minute, while Visa is at 65,000/second. Of course, Bitcoin's technology, in terms of security, is far superior to Visa's. However, without the means to scale, the Bitcoin blockchain and other blockchains will have a hard time competing with existing technologies.

Overcoming the verification of using *every* node with sharding

Blockchain networks have decentralized nodes that verify *every* transaction that is generated, in a linear manner. *Every* node in the network *needs* to verify *every* transaction. Without getting every node in the network to verify every transaction, there is a risk of compromising security.

In order to maintain the decentralization of nodes and the security of the blockchain, sharding is introduced.

Sharding is a concept that's been used in computing well before having been adopted in blockchain networks. The idea is to split databases, in the traditional sense, or blockchain nodes into segments or committees and split the workload amongst each committee. This allows for more efficient data processing.



Sharding aims to divide workload and information across the nodes in the network, so that each node is no longer responsible for verifying *every* transaction. Instead, each node is only responsible for the information in its own shard. Each shard contains independent states or sets of information. Including the different kinds of cryptocurrencies available in the shard as well as independent smart contracts and conditions of the smart contracts.

Each shard also has its own set of validators who work to verify the transactions taking place in said shard, running consensus independently of the other shards in the network.

This gives rise to a parallel verification system where nodes in each shard can work simultaneously on verifying various information, therefore creating a more efficient process. Moreso, each shard can still share the information it stores and verifies with other shards in the network. With the use of Merkle Trees, information on the network is accessible and visible to all nodes on the mainchain, without the need for *all* nodes to verify *all* transactions.

There are, however, challenges and limitations to sharding a blockchain.

When a blockchain is sharded, each shard is seen as its own standalone blockchain. Therefore, **communication** between each shard becomes difficult. By implementing various communication mechanisms, the issue can be solved. However, it’s a time consuming problem in and of itself to create these mechanisms.

Many security challenges are also posed when sharding a blockchain. These security challenges include:

Difficulty in uniformly splitting all nodes: developers will need to build committees that generate a high probability of using the consensus from the honest majority.

Preventing bad actors from gaining advantage: shards must tolerate some inconsistencies, but need to find a way to prevent fraudulent validators from gaining advantage over the full operation.

Establish identity management systems: the ability to identify bad actors from creating multiple malicious virtual nodes that could overtake the majority.

Lack of transactional history: shards do not contain a transaction's full history and this compromises the security of the network. Without access to the transaction's full history, the network becomes more susceptible to hacking and immutability.

Lack of standardization: blockchain projects use different consensus protocols which use various sharding methods – creating a lack of standardization across networks.

The above are some of the challenges for developers to consider when building a sharded blockchain. In the following paragraphs, solutions to some of the above challenges are discussed, such as dynamically reshuffling members of a shard or committee.

Achieving consensus in a sharded environment

Since shards operate in silos, achieving consensus on the state of the entire system requires cross-shard transactions to communicate with one another. With cross-shard transactions, there are multiple inputs and outputs in different shards, and all cross-shard transactions should be executed by different shards.

To process transactions, the coordinator node sends transactions to all the shards. Applying the Atomic Commit Protocol requires all participants to agree on one output for the transaction, either commit or abort. Since all cross-shard transactions should be executed by different shards, the Atomic Commit Protocol creates a consistent method for inter-shard consensus. Inter-shard consensus ensures access to the state of different shards atomically. So, when one shard commits, all shards commit. Conversely, if one shard aborts, all shards abort.

By publishing Merkle proofs onto the mainchain, each shard's history and state information is accessible to all on the mainchain. This allows for each shard to process the Merkle proof and either commit or abort a transaction given the information from the mainchain.

Consensus protocols work together, in various ways, in order to achieve consensus in shard architectures. Proof-of-X (Proof-of-Work, Proof-of-Stake, etc.) and Byzantine Fault Tolerance protocols each play a role in creating consensus across shards.

Proof-of-X protocols are responsible for committee formation and the establishment of committee members. Byzantine Fault Tolerance protocols are responsible for intra-committee consensus.

Dynamically changing committee members maintains a high level of security and ensures consensus from the honest majority. This can be done by frequently reshuffling validators and nodes between each shard. As well as reshuffling the data and information stored in each shard between other shards.

Decentralized applications use various methodologies to maintain interoperability and composability.

Interoperability aims to provide the exchange of information from one network to another. Allowing two completely separate blockchains, with various shards and dApps to share information from one blockchain and synthetically recreate it to exist in another blockchain.

There are four main technicalities that are used to achieve interoperability across blockchains.

- **Federated side chains** relay proof of ownership of assets and information and through this process, bridge assets and information.

· **Off-chain atomic swaps** trade information between Peer 2 Peer networks off-chain (layer 2) and employ hash locking to lock balances on-chain.

· **On-chain atomic swaps** exchange assets and information across multiple blockchains and employ hash locking to lock balances on-chain.

· **Intermediary networks** reach consensus on events that have happened on connected blockchains.

Communication bridges are built for incompatible blockchains to connect to one central blockchain that allows for cross-sharing of information. For example, The Cosmos network does this by using a Central Hub to connect various Zones (different blockchains) to it. This allows for incompatible blockchains, using different protocols and languages to easily communicate with one another over the Cosmos Hub.

Composability aims to allow the change of smart contract conditions within any environment. The concept combines different software stacks in different ways to allow for a function on shard A to take place on shard B.

The functionality of dApps across multiple blockchains is made possible with yanking. To yank a smart contract is to move the smart contract from shard A to the target shard, issue a receipt that contains the state of the smart contract and finally process the receipt in the target shard. When yanking a smart contract, the developer has the freedom to change the conditions of the contract as well.

With yanking, a smart contract on a shard that is used for booking tickets for movies can be moved to another shard across blockchains that is used for booking a doctor’s appointment. In this instance, the function of shard A is moved and has been amended to carry out another function on shard B.

There are numerous blockchain projects that have used sharding to scale the functionality of distributed ledger technology.

Casper and the beacon chain:

ETH 2.0 will use Casper protocol, a new proof of stake protocol, to shard nodes into subsets which will validate transactions and then share them with the main beacon chain. Casper will group transactions into "transaction packages" and will use a double verification process. Validators will be assigned to each shard and will vote on the validity of each transaction package. Once the validity of the transaction package is confirmed, a separate committee is formed for double verification. If the transaction package is validated once again, the transaction package will be published on the beacon mainchain. Validators are then released into a larger pool and the beacon chain will randomly select new validators to join the shards for future transaction verifications.

OmniLedger:

Omniledger uses the Byzantine shard Atomic Commit Protocol for its sharding practices. It has a three-step initialize, lock and unlock protocol. Transactions are atomically processed across shards and shards either commit or abort the transactions. Omniledger uses a lock/unlock process while tasking the client to drive the unlock process. The client is permitted to allow a validator to take over if the process is stalled. The client then gossips the transactions to all input shards, each input shard locks the corresponding input and issues a proof-of-acceptance if the transaction is valid. Afterwards, the client collects all responses from the input shards and issues an "unlock to commit" to the output shards.

Conclusion and parting thoughts

Sharding is a clear solution for solving the latency issue on blockchain networks, although most sharded architectures pose issues in and of themselves. It's promising to see the developments being made in projects such as Cosmos, Polkdadot, Honeybadger and many others. Developers have found creative ways to append the rigid structures of consensus algorithms and open up communication across chains. As the underlying technology of interoperable systems grows, the way forward will be paved and an opportunity for more creativity will be available. Many of the systems and services we rely on today will be re-shaped entirely by these advancements, albeit on the backend more so than consumer facing.