Overview of a Solana Cluster

A Solana cluster is a set of validators working together to serve client transactions and maintain the integrity of the ledger. Many clusters may coexist. When two clusters share a common genesis block, they attempt to converge. Otherwise, they simply ignore the existence of the other. Transactions sent to the wrong one are quietly rejected. In this section, we'll discuss how a cluster is created, how nodes join the cluster, how they share the ledger, how they ensure the ledger is replicated, and how they cope with buggy and malicious nodes.

Creating a Cluster

Before starting any validators, one first needs to create agenesis config . The config references two public keys, amint and abootstrap validator . The validator holding the bootstrap validator's private key is responsible for appending the first entries to the ledger. It initializes its internal state with the mint's account. That account will hold the number of native tokens defined by the genesis config. The second validator then contacts the bootstrap validator to register as avalidator . Additional validators then register with any registered member of the cluster.

A validator receives all entries from the leader and submits votes confirming those entries are valid. After voting, the validator is expected to store those entries. Once the validator observes a sufficient number of copies exist, it deletes its copy.

Joining a Cluster

Validators enter the cluster via registration messages sent to itscontrol plane . The control plane is implemented using agossip protocol, meaning that a node may register with any existing node, and expect its registration to propagate to all nodes in the cluster. The time it takes for all nodes to synchronize is proportional to the square of the number of nodes participating in the cluster. Algorithmically, that's considered very slow, but in exchange for that time, a node is assured that it eventually has all the same information as every other node, and that information cannot be censored by any one node.

Sending Transactions to a Cluster

Clients send transactions to any validator's Transaction Processing Unit(TPU)port. If the node is in the validator role, it forwards the transaction to the designated leader. If in the leader role, the node bundles incoming transactions, timestamps them creating anentry , and pushes them onto the cluster'sdata plane . Once on the data plane, the transactions are validated by validator nodes, effectively appending them to the ledger.

Confirming Transactions

A Solana cluster is capable of subsecondconfirmation for thousands of nodes with plans to scale up to hundreds of thousands of nodes. Confirmation times are expected to increase only with the logarithm of the number of validators, where the logarithm's base is very high. If the base is one thousand, for example, it means that for the first thousand nodes, confirmation will be the duration of three network hops plus the time it takes the slowest validator of a supermajority to vote. For the next million nodes, confirmation increases by only one network hop.

Solana defines confirmation as the duration of time from when the leader timestamps a new entry to the moment when it recognizes a supermajority of ledger votes.

Scalable confirmation can be achieved using the following combination of techniques:

1. Timestamp transactions with a VDF sample and sign the timestamp.
2. Split the transactions into batches, send each to separate nodes and have each node share its batch with its peers.
3. Repeat the previous step recursively until all nodes have all batches.

Solana rotates leaders at fixed intervals, calledslots . Each leader may only produce entries during its allotted slot. The leader therefore timestamps transactions so that validators may lookup the public key of the designated leader. The leader then signs the timestamp so that a validator may verify the signature, proving the signer is owner of the designated leader's public key.

Next, transactions are broken into batches so that a node can send transactions to multiple parties without making multiple copies. If, for example, the leader needed to send 60 transactions to 6 nodes, it would break that collection of 60 into batches of 10 transactions and send one to each node. This allows the leader to put 60 transactions on the wire, not 60 transactions for each node. Each node then shares its batch with its peers. Once the node has collected all 6 batches, it reconstructs the original set of 60 transactions.

A batch of transactions can only be split so many times before it is so small that header information becomes the primary consumer of network bandwidth. At the time of this writing (December, 2021), the approach is scaling well up to about 1,250 validators. To scale up to hundreds of thousands of validators, each node can apply the same technique as the leader node

to another set of nodes of equal size. We call the techniqueTurbine Block Propagation .
Previous Solana Validator Architecture Next Available Solana Clusters