Gossiper

Let's explore how Cassandra uses gossip protocol to keep track of the state of the system.

We'll cover the following

How does Cassandra use gossip protocol?#

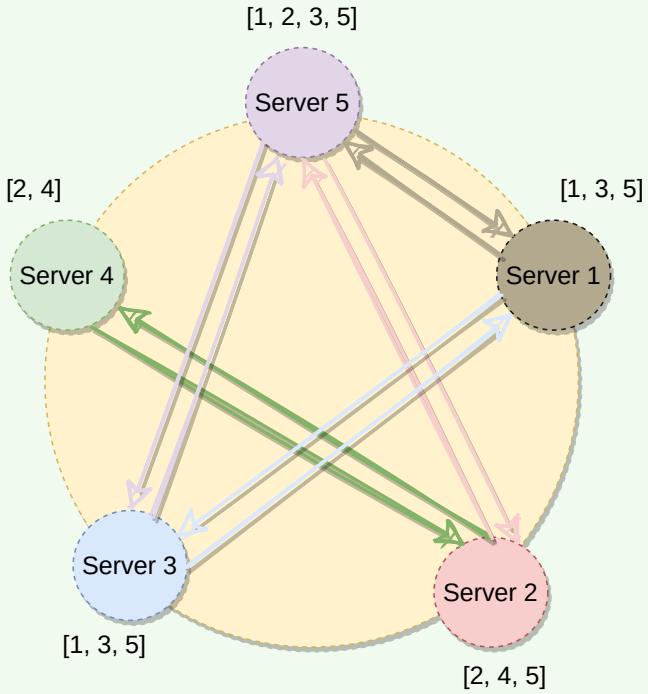
Cassandra uses **gossip protocol** that allows each node to keep track of state information about the other nodes in the cluster. Nodes share state information with each other to stay in sync. Gossip protocol is a peer-to-peer communication mechanism in which nodes periodically exchange state information about themselves and other nodes they know about. Each node initiates a gossip round every second to exchange state information about themselves (and other nodes) with one to three other random nodes. This way, all nodes quickly learn about all other nodes in a cluster.

Each gossip message has a version associated with it, so that during a gossip exchange, older information is overwritten with the most current state for a particular node.

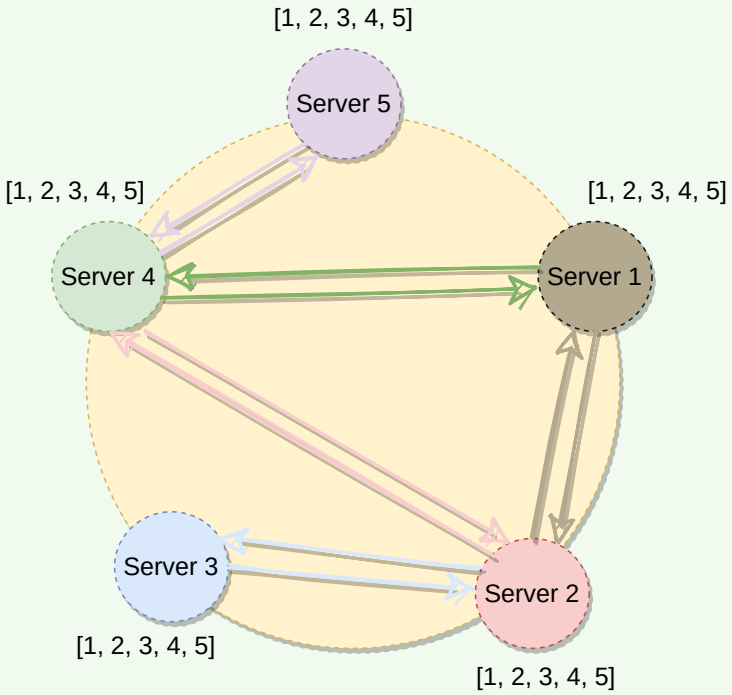
Generation number: In Cassandra, each node stores a generation number which is incremented every time a node restarts. This generation number is included in each gossip message exchanged between nodes and is used to distinguish the current state of a node from its state before a restart. The generation number remains the same while the node is alive and is incremented each time the node restarts. The node receiving the gossip message can compare the generation number it knows and the gossip message's generation number. If the generation number in the gossip message is higher, it knows that the node was restarted.

Seed nodes: To prevent problems in gossip communications, Cassandra designates a list of nodes as the seeds in a cluster. This is critical for a node starting up for the first time. By default, a node remembers other nodes it has gossiped with between subsequent restarts. The seed node designation has no purpose other than bootstrapping the gossip process for new nodes joining the cluster. Thus, seed nodes are not a single point of failure, nor do they have any other special purpose in cluster operations other than the bootstrapping of nodes.

Every second each server exchanges information with one randomly selected server



Every second each server exchanges information about all the servers it knows about



Gossip protocol

Node failure detection#

Accurately detecting failures is a hard problem to solve as we cannot say with 100% surety that if a system is genuinely down or is just very slow in responding due to heavy load, network congestion, etc. Mechanisms like Heartbeating outputs a boolean value telling us if the system is alive or not; there is no middle ground. Heartbeating uses a fixed timeout, and if there is no heartbeat from a server, the system, after the timeout, assumes that the server has crashed. Here the value of the timeout is critical. If we keep the timeout short, the system will be able to detect failures quickly but with many false positives due to slow machines or faulty networks. On the other hand, if we keep the timeout long, the false positives will be reduced, but the system will not perform efficiently for being slow in detecting failures.

Cassandra uses an adaptive failure detection mechanism as described by **Phi Accrual Failure Detector**. This algorithm uses historical heartbeat information to make the threshold adaptive. A generic Accrual Failure Detector, instead of telling that the server is alive or not, outputs the suspicion level about a server; a higher suspicion level means there are higher chances that the server is down. Using Phi Accrual Failure Detector, if a node does not respond, its suspicion level is increased and could be declared dead later. As a node's suspicion level increases, the system can gradually decide to stop sending new requests to it. Phi Accrual Failure Detector makes a distributed system efficient as it takes into account fluctuations in the network environment and other intermittent server issues before declaring a system completely dead.

Now that we have discussed Cassandra's major components, let's see how Cassandra performs its read and write operations.

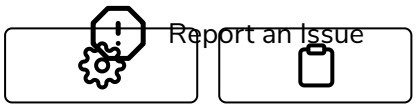
← Back

Next →

Cassandra Consistency Levels

Anatomy of Cassandra's Write Operat...

Mark as Completed


Report an Issue