Cassandra Architecture

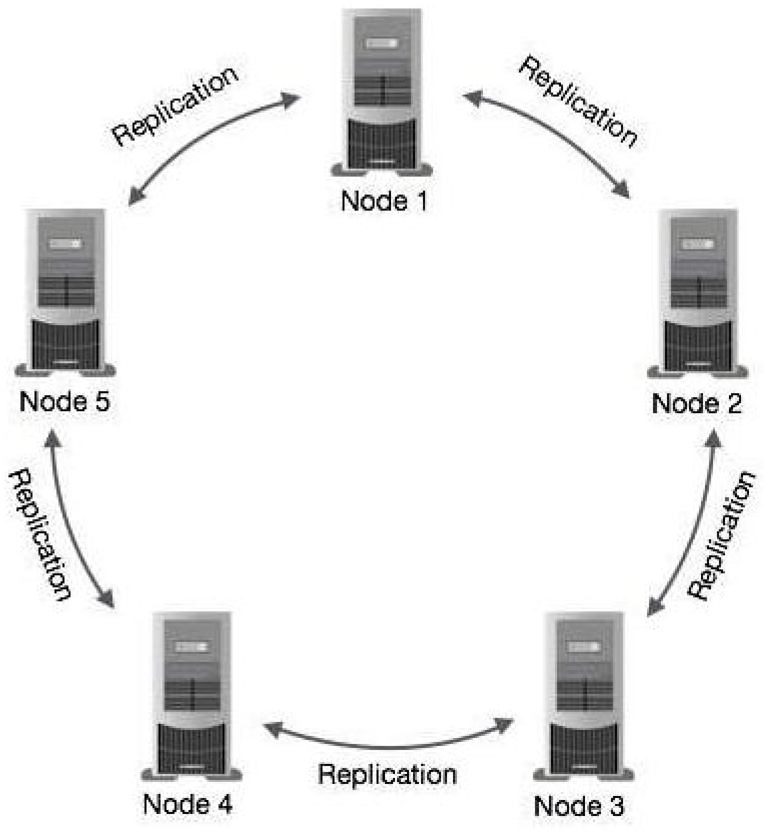
Cassandra was designed to handle big data workloads across multiple nodes without a single point of failure. It has a peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster.

- In Cassandra, each node is independent and at the same time interconnected to other nodes. All the nodes in a cluster play the same role.
- Every node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- In the case of failure of one node, Read/Write requests can be served from other nodes in the network.

Data Replication in Cassandra

In Cassandra, nodes in a cluster act as replicas for a given piece of data. If some of the nodes are responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a read repair in the background to update the stale values.

See the following image to understand the schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure.



Components of Cassandra

The main components of Cassandra are:

- **Node:** A Cassandra node is a place where data is stored.
- **Data center:** Data center is a collection of related nodes.
- **Cluster:** A cluster is a component which contains one or more data centers.
- **Commit log:** In Cassandra, the commit log is a crash-recovery mechanism. Every write operation is written to the commit log.
- **Mem-table:** A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.
- **SSTable:** It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.
- **Bloom filter:** These are nothing but quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.

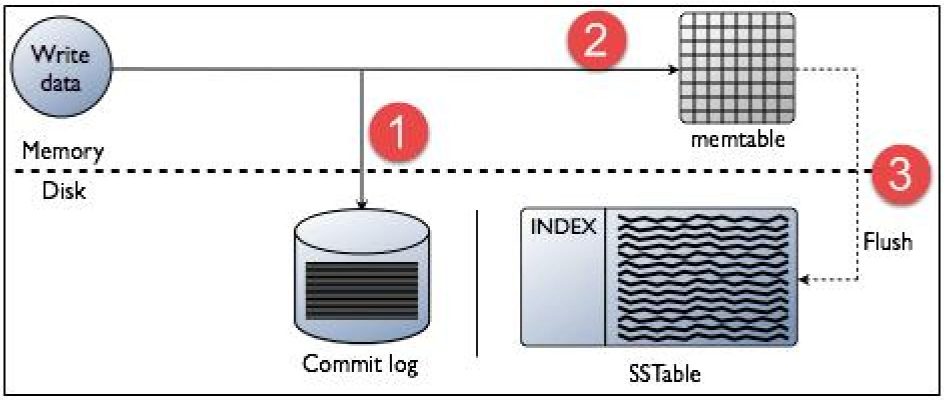
Cassandra Query Language

Cassandra Query Language (CQL) is used to access Cassandra through its nodes. CQL treats the database (Keyspace) as a container of tables. Programmers use cqlsh: a prompt to work with CQL or separate application language drivers.

The client can approach any of the nodes for their read-write operations. That node (coordinator) plays a proxy between the client and the nodes holding the data.

Write Operations

Every write activity of nodes is captured by the commit logs written in the nodes. Later the data will be captured and stored in the mem-table. Whenever the mem-table is full, data will be written into the SStable data file. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, discarding unnecessary data.



Read Operations

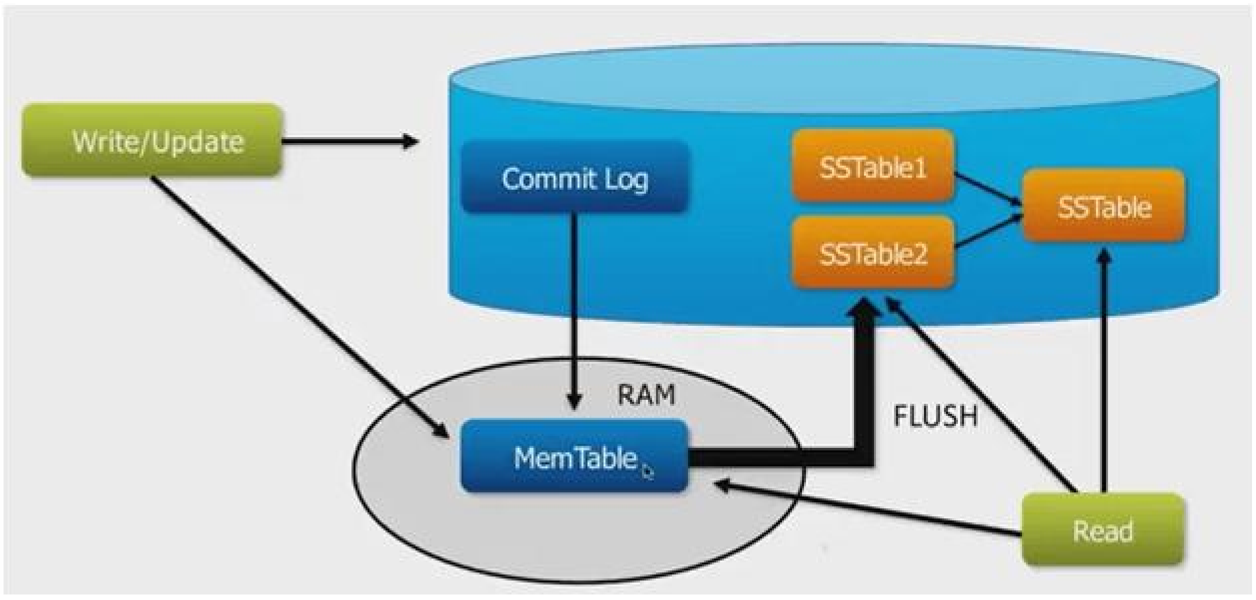
In Read operations, Cassandra gets values from the mem-table and checks the bloom filter to find the appropriate SSTable which contains the required data.

There are three types of read request that is sent to replicas by coordinators.

- Direct request
- Digest request
- Read repair request

The coordinator sends direct request to one of the replicas. After that, the coordinator sends the digest request to the number of replicas specified by the consistency level and checks if the returned data is an updated data.

After that, the coordinator sends digest request to all the remaining replicas. If any node gives out of date value, a background read repair request will update that data. This process is called read repair mechanism.


Write/Update
Commit Log
SSTable1
SSTable2
SSTable
RAM
MemTable
FLUSH
Read