

Cassandra is NoSQL database which is designed for high speed read and write , online transactional data. The specialty of Cassandra lies in the fact, that it works without a single point of failure.  
Cassandra uses gossip protocol, to keep the updated status of surrounding nodes in the cluster. In case one node goes down, another node takes its responsibility, till the time failed node is not up.

# Cassandra: Daughter of DynamoDB and BigTable

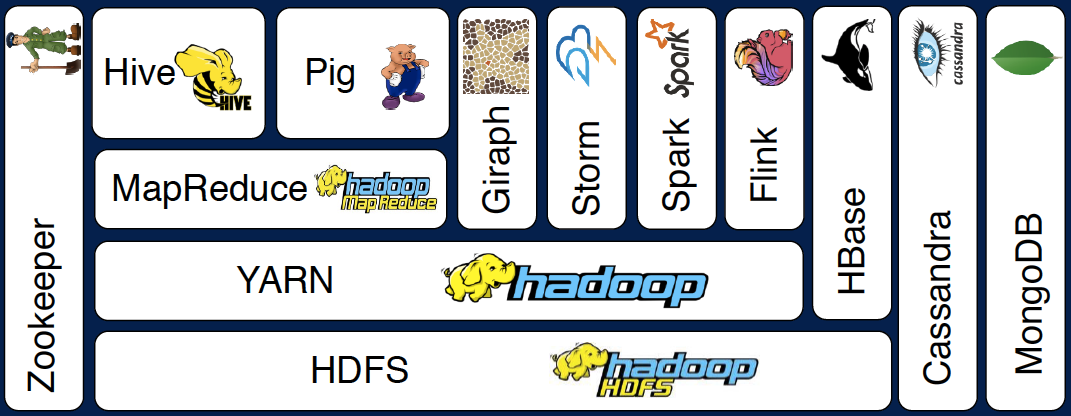
Google’s  BigTable (cf)

Amazon’s highly available DynamoDB

Data Modeling

Unlike relational models where the schema and tables are designed to represent objects and their relationships, data in NoSQL should be modeled around the specific questions that the end-user will ask. This often means using multiple tables with the same data, but the redundancy of this denormalized approach is often worth the better performance on massive volumes of data. While Cassandra doesn’t support ad-hoc data manipulation with functions like JOIN, SUM, and MAX, it can handle a higher throughput of data if you design a data model specifically for the desired queries.

Hadoop Eco System Architecture



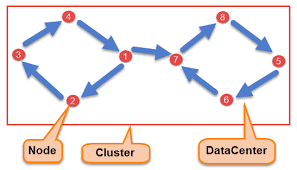
CAP

Cassandra is considered available and partition-tolerant (AP). Since there is no ‘master’ node to coordinate reads and writes, data is always available on one of the nodes, but it’s possible it could be out of date for about a second before the other nodes gossip about new updates. However, one of the main differences from other NoSQL databases is Cassandra’s ability to tune the level of consistency for each query.

**HOW DOES A DISTRIBUTED FILE SYSTEM WORK?**

A normal file system stores files on a single hard disk or partition. This limits the size of the file we can store to the size of the harddisk.  
A distributed file system bypasses this problem by storing data on multiple disks either on the same system or over a network. The design and implementation of a distributed file system is more complex than a conventional file system due to the fact that the users and storage devices are physically dispersed.

Cassandra Architecture



* **Node**

Node is the place where data is stored. It is the basic component of Cassandra.

* **Data Center**

A collection of nodes are called data center. Many nodes are categorized as a data center.

* **Cluster**

The cluster is the collection of many data centers.

* **Commit Log**

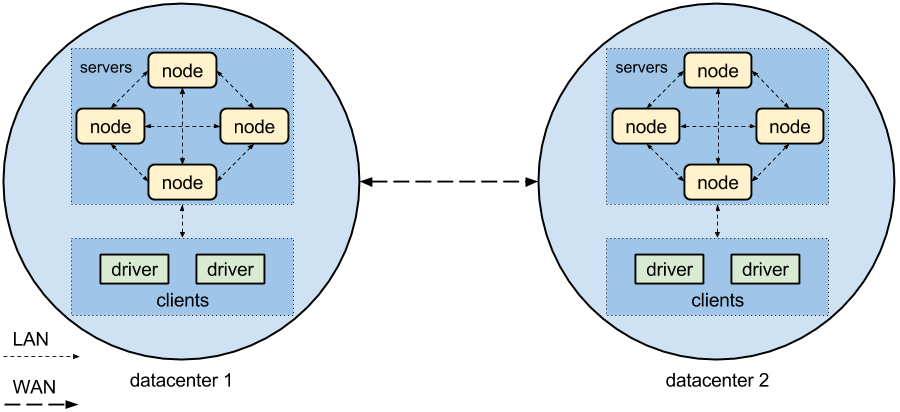
Every write operation is written to Commit Log. Commit log is used for crash recovery.

* **Mem-table**

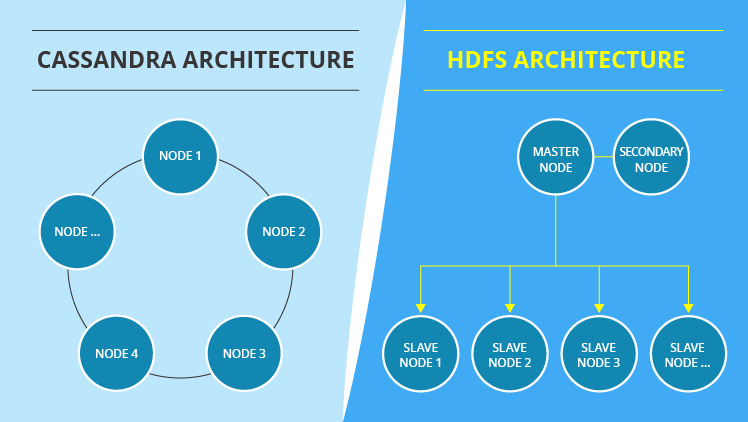
After data written in Commit log, data is written in Mem-table. Data is written in Mem-table temporarily.

**SSTable(**Sorted Strings Table)

When Mem-table reaches a certain threshold, data is flushed to an SSTable disk file.



Since Cassandra uses a decentralized approach, it works well for use cases that require multiple datacenters in different locations



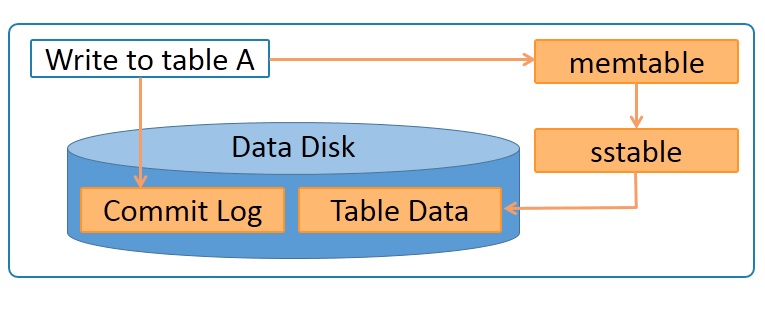
Cassandra Write Process

The Cassandra write process ensures fast writes.

Steps in the Cassandra write process are:

1. Data is written to a commitlog on disk.
2. The data is sent to a responsible node based on the hash value.
3. Nodes write data to an in-memory table called memtable.
4. From the memtable, data is written to an sstable in memory. Sstable stands for Sorted String table. This has a consolidated data of all the updates to the table.
5. From the sstable, data is updated to the actual table.
6. If the responsible node is down, data will be written to another node identified as tempnode. The tempnode will hold the data temporarily till the responsible node comes alive.

The diagram below depicts the write process when data is written to table A.



Data is written to a commitlog on disk for persistence. It is also written to an in-memory memtable. Memtable data is written to sstable which is used to update the actual table.

## Cassandra Read Process

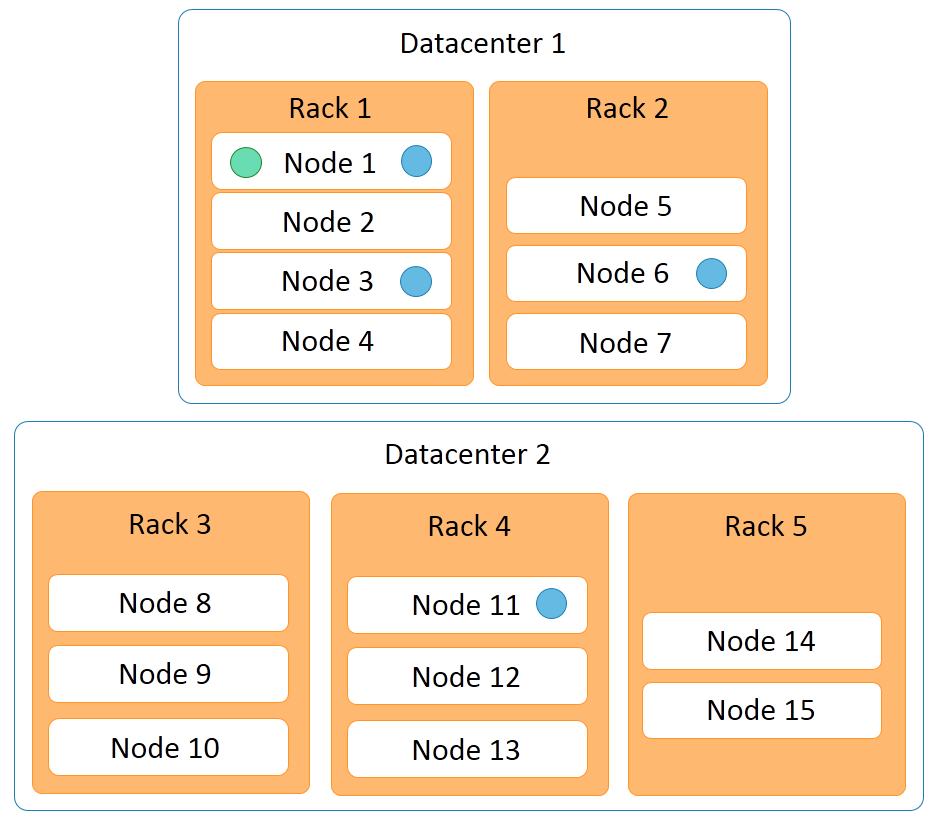
The Cassandra read process ensures fast reads. Read happens across all nodes in parallel. If a node is down, data is read from the replica of the data. Priority for the replica is assigned on the basis of distance.

Features of the Cassandra read process are:

* Data on the same node is given first preference and is considered data local.
* Data on the same rack is given second preference and is considered rack local.
* Data on the same data center is given third preference and is considered data center local.
* Data in a different data center is given the least preference.

Data in the memtable and sstable is checked first so that the data can be retrieved faster if it is already in memory.

The diagram below represents a Cassandra cluster.



It has two data centers:

* data center 1
* data center 2

Data center 1 has two racks, while data center 2 has three racks. Fifteen nodes are distributed across this cluster with nodes 1 to 4 on rack 1, nodes 5 to 7 on rack 2, and so on.

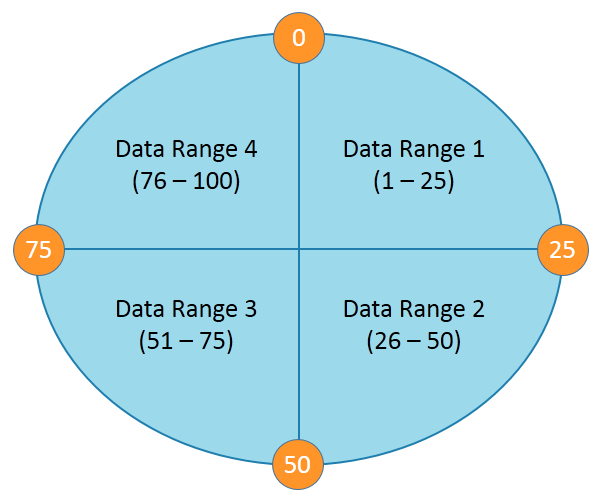
Let us discuss the example of Cassandra read process in the next section.

## Data Partitions

Cassandra performs transparent distribution of data by horizontally partitioning the data in the following manner:

* A hash value is calculated based on the primary key of the data.
* The hash value of the key is mapped to a node in the cluster
* The first copy of the data is stored on that node.
* The distribution is transparent as you can both calculate the hash value and determine where a particular row will be stored.

The following diagram depicts a four node cluster with token values of 0, 25, 50 and 75.



For a given key, a hash value is generated in the range of 1 to 100.

Keys with hash values in the range 1 to 25 are stored on the first node, 26 to 50 are stored on the second node, 51 to 75 are stored on the third node, and 76 to 100 are stored on the fourth node.

Please note that actual tokens and hash values in Cassandra are 127-bit positive integers.

Let us discuss replication in Cassandra in the next section.

## Replication in Cassandra

Replication refers to the number of replicas that are maintained for each row. Replication provides redundancy of data for fault tolerance. A replication factor of 3 means that 3 copies of data are maintained in the system.

In this case, even if 2 machines are down, you can access your data from the third copy. The default replication factor is 1. A replication factor of 1 means that a single copy of the data is maintained, so if the node that has the data fails, you will lose the data.

Cassandra allows replication based on nodes, racks, and data centers, Replication across data centers guarantees data availability even when a data center is down.

Replications

 If his data needed to be quickly accessible to several locations, Cassandra would easily scale to multiple datacenters with flexible configurations. For example, a piece of data can be stored in

1 replicas on the America

1 replicas on the Asia and 1 in Europe

|  |
| --- |
| a piece of data can be stored in two replicas on the east coast and three replicas on the west coast using:  cqlsh:rams> CREATE KEYSPACE messages WITH REPLICATION={'class': 'NetworkTopologyStrategy', 'us-west': 3, 'us-east': 2}; |

Using a SQL-like language known as the Cassandra Query Language (CQL), we were able to learn and test queries immediately using the [CQL shell](http://docs.datastax.com/en/cql/3.1/cql/cql_using/about_cql_c.html).

Cassandra’s native support of time to live (TTL) that allows columns of data to expire automatically.