1. Nosql Databases

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. NoSQL is an approach to [database](http://searchsqlserver.techtarget.com/definition/database) design that can accomodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only [SQL](http://searchsqlserver.techtarget.com/definition/SQL)," is an alternative to traditional relational databases in which data is placed in tables and data [schema](http://searchsqlserver.techtarget.com/definition/schema) is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

1. Types of Nosql Databases

There are 4 basic types of NoSQL databases:

a. Key-Value Store – It has a Big Hash Table of keys & values {Example- Riak, Amazon S3

(Dynamo)}

b. Document-based Store- It stores documents made up of tagged elements. {Example- CouchDB}

c. Column-based Store- Each storage block contains data from only one column, {Example- HBase, Cassandra}

d. Graph-based-A network database that uses edges and nodes to represent and store data. {Example- Neo4J}

1. CAP Theorem

In theoretical computer science, the **CAP theorem**, also named Brewer's **theorem** after computer scientist Eric Brewer, states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees: Consistency. Availability. Partition tolerance. For example, if consistency and availability (CA) are important, you can’t partition the data. If consistency is not that important, then you can partition the data for high availability (AP). If you have to partition the data and consistency is important (CP), then availability suffers.  In the past decade, a vast range of new systems have emerged, as well as much debate on the relative merits of consistency and availability.The new breed of NoSQL solutions play around with these concepts so you can choose your tolerance for your desired properties.

1. HBase Architecture

**HBase Data Model**

HBase data model stores semi-structured data having different data types, varying column size and field size. The layout of HBase data model eases data partitioning and distribution across the cluster. HBase data model consists of several logical components- row key, column family, table name, timestamp, etc. Row Key is used to uniquely identify the rows in HBase tables. Column families in HBase are static whereas the columns, by themselves, are dynamic.

* **HBase Tables –** Logical collection of rows stored in individual partitions known as Regions.
* **HBase Row –** Instance of data in a table.
* **RowKey -**Every entry in an HBase table is identified and indexed by a RowKey.
* **Columns -** For every RowKey an unlimited number of attributes can be stored.
* **Column Family –** Data in rows is grouped together as column families and all columns are stored together in a low level storage file known as HFile.

**HBase Use Cases- When to use HBase**

HBase is an ideal platform with ACID compliance properties making it a perfect choice for high-scale, real-time applications. It does not require a fixed schema, so developers have the provision to add new data as and when required without having to conform to a predefined model.

It provides users with database like access to Hadoop-scale storage, so developers can perform read or write on subset of data efficiently, without having to scan through the complete dataset. HBase is the best choice as a NoSQL database, when your application already has a hadoop cluster running with huge amount of data. HBase helps perform fast read/writes.

HBase provides low-latency random reads and writes on top of HDFS. In HBase, tables are dynamically distributed by the system whenever they become too large to handle (Auto Sharding). The simplest and foundational unit of horizontal scalability in HBase is a Region. A continuous, sorted set of rows that are stored together is referred to as a region (subset of table data).  HBase architecture has a single HBase master node (HMaster) and several slaves i.e. region servers. Each region server (slave) serves a set of regions, and a region can be served only by a single region server. Whenever a client sends a write request, HMaster receives the request and forwards it to the corresponding region server.

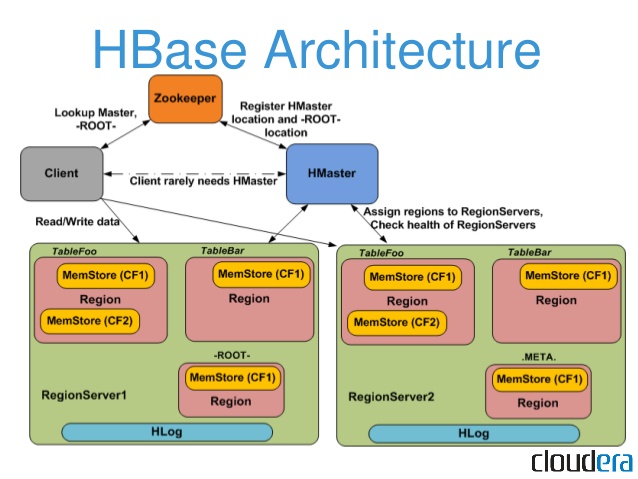


Image Credit : Cloudera

HBase can be run in a multiple master setup, wherein there is only single active master at a time. HBase tables are partitioned into multiple regions with every region storing multiple table’s rows.

### ****Components of Apache HBase Architecture****

HBase architecture has 3 important components- HMaster, Region Server and ZooKeeper.

#### ****HMaster****

HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster for load balancing. Responsibilities of HMaster –

* Manages and Monitors the Hadoop Cluster
* Performs Administration (Interface for creating, updating and deleting tables.)
* Controlling the failover
* DDL operations are handled by the HMaster
* Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.

#### ****Region Server****

These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the hadoop cluster. Region Server runs on HDFS DataNode and consists of the following components –

* Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted.
* MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore.
* Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage.
* HFile is the actual storage file that stores the rows as sorted key values on a disk.

#### ****Zookeeper****

HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes.

ZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts ZooKeeper to get the details of region servers. Various services that Zookeeper provides include –

* Establishing client communication with region servers.
* Tracking server failure and network partitions.
* Maintain Configuration Information
* Provides ephemeral nodes, which represent different region servers.

1. HBase vs RDBMS

Difference between **RDBMS** and **HBase**. **Hadoop** and **RDBMS** are varying concepts of processing, retrieving and storing the data or information. While **Hadoop** is an open-source Apache project, **RDBMS** stands for **Relational** Database Management System. ... SQL is utilized to retrieve needed data which is stored in such tables

