NoSQL Databases

The term NoSQL was coined by Carlo Strozzi in the year 1998. He used this term to name his Open Source, Light Weight, DataBase which did not have an SQL interface.

NoSQL is a non-relational (generic data model) database management systems, designed for distributed data stores with very large scale of data storing needs. This type of data storing may not require fixed schema (non-relational), avoid join operations(de-normalized) and typically scale horizontally.

Types of Nosql Databases

There are various NoSQL Databases. Each one uses a different method to store data. Some might use column store, some document, some graph, etc., Each database has its own unique characteristics.

## Document Oriented Databases

It’s a collection of documents that stores data. A document is a key value collection where the key allows access to its value. Documents are stored into collections in order to group different kinds of data and can contain many different key-value pairs, or key-array pairs, or even nested documents. E.g. MongoDB

## Key-value stores

Key-value stores are most basic types of NoSQL databases designed to handle huge amounts of data.In the key-value storage, database stores data as hash table where each key is unique and the value can be string, JSON, BLOB (Binary Large OBjec) etc. For example a key-value pair might consist of a key like "Name" that is associated with a value like "Robin". Key-Value stores follow the 'Availability' and 'Partition' aspects of CAP theorem and can be used as collections, dictionaries, associative arrays etc. e.g. Redis, Dynamo, Riak. Etc

## Column-oriented databases

Column-oriented databases primarily work on columns and values of a single column are stored contiguously.A ll data within each column datafile have the same type which makes it ideal for compression. Column stores can improve the performance of queries as it can access specific column data. e.g. HBase, BigTable, Cassandra, SimpleDB etc.

## Graph Databases

A graph database stores data in a graph structure which consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices. Each node represents an entity (such as a student or business) and each edge represents a connection or relationship between two nodes. Every node and edge are defined by a unique identifier. E.g. Neo4j

CAP Theorem

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.

* **Consistency** - A guarantee that every node in a distributed cluster returns the same, most recent, successful write. Consistency refers to every client having the same view of the data. There are various types of consistency models. Consistency in CAP (used to prove the theorem) refers to linearizability or sequential consistency, a very strong form of consistency.
* **Availability** - Every non-failing node returns a response for all read and write requests in a reasonable amount of time. The key word here is every. To be available, every node on (either side of a network partition) must be able to respond in a reasonable amount of time.
* **Partition Tolerant** - The system continues to function and upholds its consistency guarantees in spite of network partitions. Network partitions are a fact of life. Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.

The CAP theorem categorizes systems into three categories:

* CP (Consistent and Partition Tolerant) - At first glance, the CP category is confusing, i.e., a system that is consistent and partition tolerant but never available. CP is referring to a category of systems where availability is sacrificed only in the case of a network partition.
* CA (Consistent and Available) - CA systems are consistent and available systems in the absence of any network partition. Often a single node DB servers are categorized as CA system. Single node DB servers do not need to deal with partition tolerance and are thus considered CA systems. The only hole in this theory is that single node DB systems are not a network of shared data systems and thus do not fall under the preview of CAP.
* AP (Available and Partition Tolerant) - These are systems that are available and partition tolerant but cannot guarantee consistency.



HBase Architecture

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.

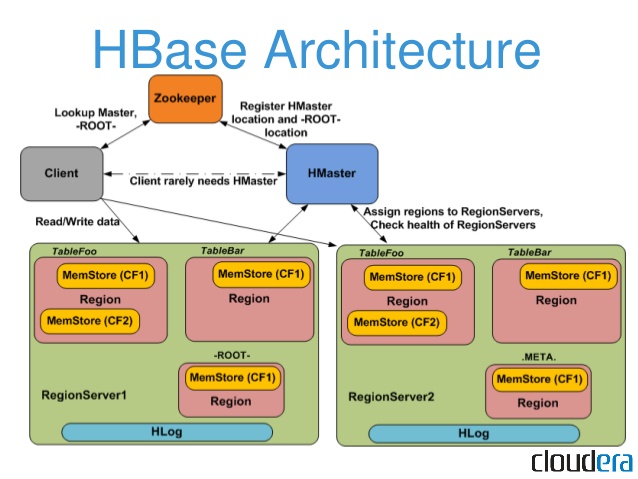


Image Credit : Cloudera

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

HBase architecture has 3 important components

•Region servers serve data for reads and writes.

•HBaseMaster process handles the Region assignment, DDL (create, delete tables) operations

•Zookeeper maintains a live cluster state.

# Regions

HBaseTables are divided horizontally by row key range into “Regions.”

•A region contains all rows in the table between the region’s start key and end key.

•Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve data for reads and writes.

•A region server can serve about 1,000 regions.

# HBase Master

Region assignment, DDL (create, delete tables) operations are handled by the HBaseMaster. A master is responsible for:

•Coordinating the region servers

•Assigning regions on startup

•Re-assigning regions for recovery or load balancing

•Monitoring all RegionServerinstances in the cluster (listens for notifications from zookeeper)

**Admin functions**

•Interface for creating, deleting, updating tables

# ZooKeeper

•HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster.

•Zookeeper maintains which servers are alive and available, and provides server failure notification.

•Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines for consensus.

HBase vs RDBMS

|  |  |
| --- | --- |
| **RDBMS** | **HBASE** |
| RDBMS is row-oriented databases | HBaseis a distributed, column-oriented data storage system |
| RDBMS tables have fixed-schema | Hbasetables do not have fixed-schema |
| RDBMS tables guarantee ACID properties | Hbasetables guarantee consistency and partition tolerance |
| RDBMS uses SQL (Structured query Langauge) to query the data | Hbaseuses Java client API and Jruby |