**Session 9 Assignment 3**

1. **NoSQL Databases**

* NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. It is which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

examples: mongoDB (document based), cassandra (columnar), neo4j (graph base) etc.

1. **Types of NoSQL Databases**

* There are 4 types of NoSQL databases below are the same,
  + Key-Value databases: It stores are the simplest NoSQL data stores to use from an API perspective. The client can either get the value for the key, put a value for a key, or delete a key from the data store. The value is a blob that the data store just stores, without caring or knowing what's inside; it's the responsibility of the application to understand what was stored. Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled. Example- Riak, Amazon S3 (Dynamo).
  + Document databases: Documents are the main concept in document databases. The database stores and retrieves documents, which can be XML, JSON, BSON, and so on. These documents are self-describing, hierarchical tree data structures which can consist of maps, collections, and scalar values. The documents stored are similar to each other but do not have to be exactly the same. Document databases store documents in the value part of the key-value store; think about document databases as key-value stores where the value is examinable. Example- CouchDB.
  + Column family stores: Column families are groups of related data that is often accessed together. For a Customer, we would often access their Profile information at the same time, but not their Orders. Each column family can be compared to a container of rows in an RDBMS table where the key identifies the row and the row consists of multiple columns. When a column consists of a map of columns, then we have a super column. A super column consists of a name and a value which is a map of columns. Think of a super column as a container of columns. Example- HBase, Cassandra
  + Graph Databases: Graph databases allow you to store entities and relationships between these entities. Entities are also known as nodes, which have properties. Think of a node as an instance of an object in the application. Relations are known as edges that can have properties. Edges have directional significance; nodes are organized by relationships which allow you to find interesting patterns between the nodes. The organization of the graph lets the data to be stored once and then interpreted in different ways based on relationships. Example- Neo4J

1. **CAP Theorem**

* C: Consistency: Is the state of the databases should be consistent compared to an operation after its completion.

A: Availability: Is the system is always available.

P: Partition Tolerance: Database is having tolerance towards the partitioning the nodes in the database.

CAP theorem: says that in a distributed environment it is not possible for a database to have all the features as C, A, P.

NoSQL databases lies in AP (availability, partition tolerance) & CP (availability, partition tolerance) domains.

NoSQL databases are generally schema less or having dynamic schema.

AP: DynamoDB, Cassandra, CouchDB etc.

CP: mongoDB, Redis, memcache, HBase etc.

1. **HBase Architecture**

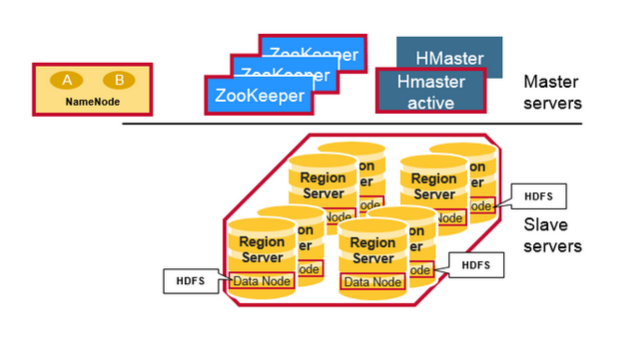
* HBase is composed of three types of servers in a master slave type of architecture. Region servers serves data for reads and writes. When accessing data, clients communicate with HBase Region Servers directly. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process.

Zookeeper, which is part of HDFS, maintains a live cluster state.

The DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the Region Servers. HBase data is local when it is written, but when a region is moved, it is not local until compaction.

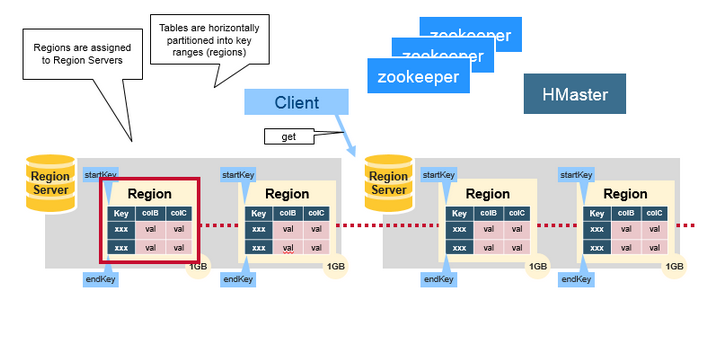
The NameNode maintains metadata information for all the physical data blocks that comprise the files.

Below is the HBase Architecture -



**Regions**

HBase Tables 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 HMaster**

Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.

A master is responsible for:

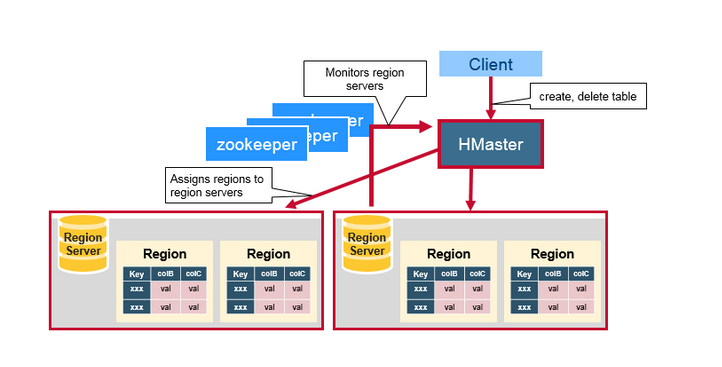
* + Coordinating the region servers

- Assigning regions on start-up, re-assigning regions for recovery or load balancing

- Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)

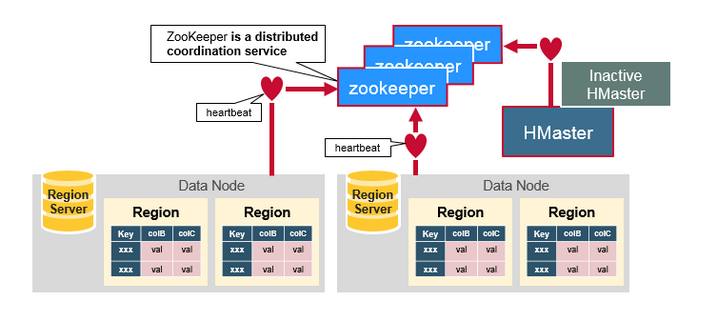
* + Admin functions

- Interface for creating, deleting, updating tables



**ZooKeeper: The Coordinator**

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.



1. **HBase vs RDBMS**

|  |  |  |
| --- | --- | --- |
| **Feature** | **NoSQL** | **RDBMS** |
| **Data Volume** | Handles Huge Data Volumes | Handles Limited Data Volumes |
| **Data Validity** | Highly Guaranteed | Less Guaranteed |
| **Scalability** | Horizontally | Horizontally & Vertically |
| **Query Language** | No declarative query language | Structured Query Language (SQL) |
| **Schema** | No predefined schema or less rigid schemas | Predefined Schema (Data Definition Language & Data Manipulation Language) |
| **Data Type** | Supports unstructured and unpredictable data | Supports relational data and its relationships are stored in separate tables |
| **ACID/BASE** | Based on BASE principle (Basically, Available, Soft State, Eventually Consistent) | Based on ACID principle (Atomicity, Consistency, Isolation and Durability) |
| **Transaction Management** | Weaker transactional guarantee | Strong transactional guarantees |
| **Data Storage Technique** | Schema-free collections are utilized to store different types and document structures, such as {“color”, “blue”} and {“price”, “23.5”} can be stored within a single collection. | No collections are used for data storage; instead use DML for it. |