**Explain in brief**

● **Differences between HBASE and HDFS.**

**HDFS**

HDFS is a distributed file system and has the following properties:

1. It is optimized for streaming access of large files. You would typically store files that are in the 100s of MB upwards on HDFS and access them through MapReduce to process them in batch mode.   
2. HDFS files are write once files. You can append to files in some of the recent versions but that is not a feature that is very commonly used. Consider HDFS files as write-once and read-many files. There is no concept of random writes.  
3. HDFS doesn't do random reads very well.

**HBase**

HBase on the other hand is a database that stores its data in a distributed file system. The file system of choice typically is HDFS owing to the tight integration between HBase and HDFS. Having said that, it doesn't mean that HBase can't work on any other file system. It's just not proven in production and at scale to work with anything except HDFS.

HBase has the following properties:

1. Low latency access to small amounts of data from within a large data set. You can access single rows quickly from a billion row table.  
2. Flexible data model to work with and data is indexed by the row key.  
3. Fast scans across tables.  
4. Scale in terms of writes as well as total volume of data.

**● List and explain the main components of HBASE.**

HBase is composed of three types of servers in a master slave type of architecture.

• **Region servers** serve data for reads and writes.

• **HBase Master** process handles the Region assignment, DDL (create, delete tables) operations

• **Zookeeper** maintains a live cluster state.

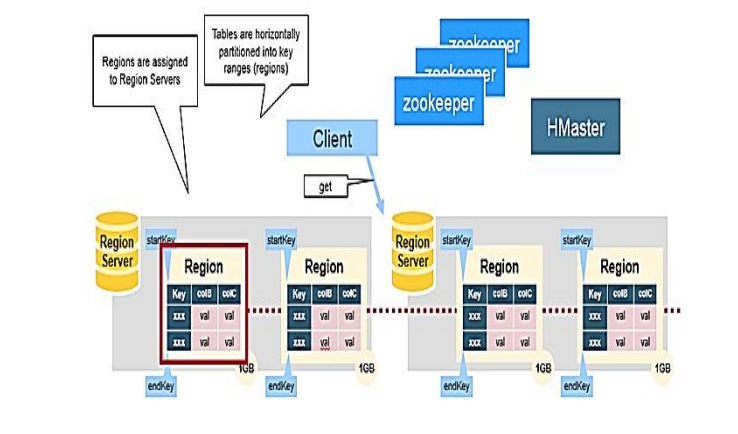
**Regions**

• 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

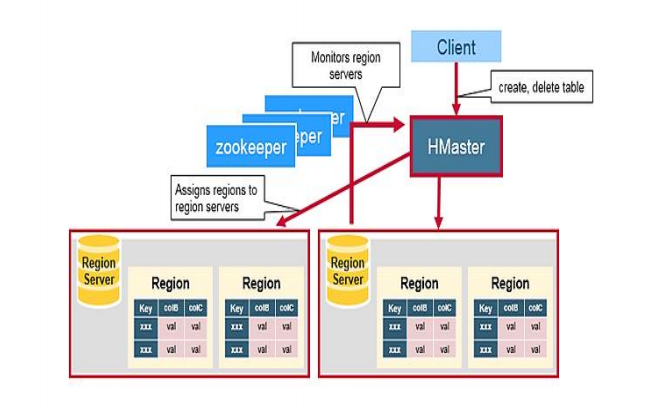
• Assigning regions on startup

• Re-assigning regions for recovery or load balancing

• Monitoring all Region Server instances in the cluster (listens for notifications from zookeeper)

Admin functions

• Interface for creating, deleting, and updating tables

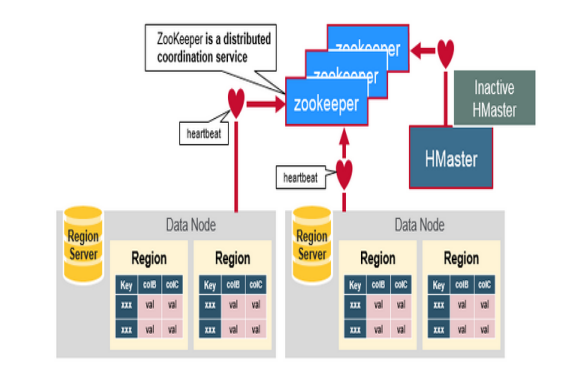


**Zookeeper: The Coordinator**

• 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.



● **Does Hbase support sql?**

There are various technologies being developed in Hbase that support sql. HBase provides random, read-write access to Big Data stored in very large tables as a distributed columnar store. HBase thus gained immediately popularity as a Big Data technology that unlike Hadoop which was primarily used in the backend data warehousing infrastructure could be deployed to service online transactions as well. Just as Hive brought SQL to Hadoop, there are many alternate projects providing SQL for HBase which is the subject of this blog article.

**Phoenix**

[Phoenix](https://github.com/forcedotcom/phoenix) is a technology developed by Salesforce.com to put a SQL skin over HBase.

## Impala

Cloudera has been promoting Impala heavily since its announcement in Strata last year. Impala is a SQL engine that can run on HDFS or HBase or both.

## Drill

Apache Drill (the only in this series which is an Apache project) was inspired by Google Dremel. It is the most interesting in terms of its emphasis on interactive analysis of large scale datasets. It is similar to Impala in many ways but is community driven.