**SESSION 17: HBASE INTRODUCTION**

**ASSIGNMENT 1**

1. **Give a brief difference between HBASE and HDFS.**

HBase is a non-relational database that can run on top of Hadoop and provides you random data access/querying capabilities. HDFS, by itself has no support for reads/writes at random location.

Another primary difference would be the way data is stored in the two. HBase stores data as key/value pairs as in a column database while data in HDFS is stored as flat files.

To put it simply, HBase is an extension for the Hadoop environment that allows you to quickly read/write data.

1. **List the main components of HBase.**

HBase architecture has 3 major components:

* 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 server data for reads and writes.

* 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.

* HMaster

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

A master is responsible for:

* Coordinating the region servers
  + Assigning regions on startup, 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

1. **Does HBase support SQL?**

Unlike relational database systems, HBase doesn’t support a structured query language like SQL. Instead, HBase is a column oriented NoSQL database built on top of Hadoop.

1. **When should we use HBase? List some of the scenarios.**

HBase is an ideal platform with ACID compliance properties making it a perfect choice for high-scale, real-time applications. It is suitable for use cases where you need real time and random read/write access to huge volumes of data (Big Data). As a NoSQL DB, HBase offers a lot of good functionalities, but it is still not a “Fit for All” solution. Following are some key scenarios to be considered:

* Volume of data is the most common point to be considered. You must have a large amount of data to be processed in a distributed environment. Otherwise for a small amount of data, it will be stored and processed in a single node, keeping other nodes idle which will be a misuse of technology.
* HBase is not suitable for transactional applications, large volume MR jobs, relational analytics, etc. It is preferred when you have a variable schema with slightly different rows. It is also suitable when you are going for a key dependent access to your stored data.
* HBase runs on top of HDFS. And HDFS works efficiently with a large number of nodes. So if you have good hardware support, then HBase can be a good selection.
* The application shouldn’t have any requirement for RDBMS features like transaction, triggers, complex query, joins, etc. If the application can be built without these features, then HBase is a good fit.
* If you need a random and real time access to your data, then HBase is a suitable candidate. It is also a perfect fit for storing large tables with multi structured data. It gives flashback support to queries, which makes it more suitable for fetching data in a particular instance of time.

Apart from the above points, HBase is also suitable when you need fault tolerant, fast and usable data management in a non-relational environment.

1. **What are the different modes in which HBase can be run?**

Hbase has two run modes, as below:

* Standalone HBase – This is the default mode. In this mode, HBase does not use HDFS but the local filesystem instead and it runs all HBase daemons and local zookeeper all up in the same JVM.
* Distributed – Distributed mode can be subdivided into pseudo-distributed and fully-distributed modes.

A pseudo-distributed mode is simply a distributed mode in which all daemons run on a single node, whereas in fully-distributed mode the daemons are spread across all nodes in the cluster.

1. **Why is zookeeper needed in HBase?**

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.

Zookeeper maintains which servers are alive and available, and provides failure notifications. It uses consensus to guarantee common shared state.

1. **HBase is a schema less database. What does it mean?**

Being a schema less database, means that it doesn’t have the concept of fixed columns schema; it only defines column families. Within a column family, any record can have any number of columns which in turn can hold any values irrespective of the data type.

1. **What is the minimum number of column family every HBase table should have?**

ONE.

1. **What is the benefit of using connection pool in HBase?**

Creating connections to a server component from an application is a heavy weight operation and it is much pronounced when connecting to a database server. That being the reason database connection pooling is used to reuse connection objects and HBase is no exception.

In HBase, data from meta table that stores details about region servers gets cached at the individual connection level that makes HBase connections much heavier. So if there are region movements for balancing or if a region server fails, the meta data needs to be refreshed for each connection object which is a performance overhead. For these reasons, applications need to try to reuse connection objects created.