**Assignment-12.2**

**1.Difference between Hbase and HDFS?**

* **HDFS**

It is mainly used for accessing large files. You can store files which are having 100s of MB on hdfs and access them through MapReduce.

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.

HDFS doesn't do random reads very well.

* **HBase**

It is a database that stores it's data in a distributed filesystem. The filesystem 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 filesystem. It's just not proven in production and at scale to work with anything except HDFS.

HBase provides you with the following:

* 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.
* Flexible data model to work with and data is indexed by the row key.
* Fast scans across tables.
* Scale in terms of writes as well as total volume of data.

**2. List and explain 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.

**3. When should we use HBASE, list some of the scenarios for the same.**

* Data volume: The volume of data is the most common point to be considered. You should have peta bytes 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. So, it will be a misuse of technology framework.
* Application Types: HBase is not suitable for transactional applications, large volume MapReduce 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.
* Hardware environment: HBase runs on top of HDFS. And HDFS works efficiently with a large number of nodes (minimum 5). So, if you have good hardware support, then HBase can be a good selection.
* No requirement of relational features: Your application should not have any requirement for RDBMS features like transaction, triggers, complex query, complex joins etc. If you can build your application without these features, then go for HBase.
* Quick access to data: 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.

4**. What are the different modes in which Hbase can be run?**

* Standalone
* Distributed

5. **Why is zookeeper needed in Hbase?**

* ZooKeeper is a high-performance coordination service for distributed applications.
* It exposes common services like naming, configuration management, synchronization, and group services, in a simple interface so you don't have to write them from scratch.
* It can be used off-the-shelf to implement consensus, group management, leader election, and presence protocols. And you can build on it for your own, specific needs.

6. **Hbase is a schema less database, what does it mean?**

Actually HBase called schemaless Data store. But to store values in particular column, we should need to specify the table and then column family then column name then the {Actual value}.

**7. What is the minimum number of column family every Hbase table should have?**

HBase currently does not do well with anything above two or three column families so keep the number of column families in your schema low.

8. **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 that can serve data for specific key ranges 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 need to be refreshed for each connection object which is a performance overhead.
* For these reasons, applications need to try to reuse connection objects created.

**9. What is the difference between memstore and hfile in HBase?**

* **Memstore**:
* The MemStore is a write buffer where HBase accumulates data in memory before a permanent write.
* Its contents are flushed to disk to form an HFile when the MemStore fills up.
* It doesn't write to an existing HFile but instead forms a new file on every flush.
* The HFile is the underlying storage format for HBase.
* HFiles belong to a column family(one MemStore per column family). A column family can have multiple HFiles, but the reverse isn't true.
* size of the MemStore is defined in hbase-site.xml called hbase.hregion.memstore.flush.size.
* **Hfile:**
* Row key is primary identifier.
* HFiles store the rows as sorted KeyValues on disk.
* HFile is Unit of Storage used by HBase
* HFile is data file HBase which is stored on HDFS

**10.Describe compactions in HBase.**

* HBase is a distributed data store based upon a log-structured merge tree, so optimal read performance would come from having only one file per store (Column Family).
* However, that ideal isn’t possible during periods of heavy incoming writes.
* Instead, HBase will try to combine HFiles to reduce the maximum number of disk seeks needed for a read. This process is called compaction.

**11.List and explain the logical entities in HBase.**

* An HBase table is made of column families which are the logical and physical grouping of columns.
* The columns in one family are stored separately from the columns in another family.
* The column family and column qualifier names are repeated for each row.
* Therefore ,keep the names as short as possible to reduce the amount of data the HBase stores and reads.

12.**What will happen if we do not create a row key while inserting the data?**

* Using INSERT IGNORE effectively causes MySQL to ignore execution errors while attempting to perform INSERT statements.
* This means that an INSERT IGNORE statement which contains a duplicate value in a UNIQUE index or PRIMARY KEY field does not produce an error, but will instead simply ignore that particular INSERT command entirely.
* The obvious purpose is to execute a large number of INSERT statements for a combination of data that is both already existing in the database as well as new data coming into the system.

13.**How can filters be applied in HBase and what are the benefits?**

* When reading data from HBase using Get or Scan operations, you can use custom filters to return a subset of results to the client.
* While this does not reduce server-side IO, it does reduce network bandwidth and reduces the amount of data the client needs to process.
* Filters are generally used using the Java API, but can be used from HBase Shell for testing and debugging purposes.

1. **What are the data model operations in hBase?**

* The four primary data model operations are Get, Put, Scan, and Delete
* Get returns attributes for a specified row.
* Put either adds new rows to a table (if the key is new) or can update existing rows (if the key already exists).
* Scan allow iteration over multiple rows for specified attributes.
* Delete removes a row from a table.

15.**How MapReduce can be used with HBase?**

* HBase supports scaling far beyond traditional RDBMS capabilities, it supports automatic sharding and massive parallel processing capabilities via Mapreduce.
* HBase can be used as data source as well as data sink for Mapreduce jobs.

1. **What is regionserver?**

* RegionServers are the software processes (often called daemons) you activate to store and retrieve data in HBase (Hadoop Database).
* In production environments, each RegionServer is deployed on its own dedicated compute node.