**SESSION 17: HBASE INTRODUCTION**

**ASSIGNMENT 2**

1. **What is the difference between memstore and HFile in HBase?**

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.

In contrast, 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 is not true.

1. **Describe compactions in HBase.**

Compaction is the process by which HBase cleans up after itself. When the storage area of HBase is almost filled with data, it starts creating compressed files which occupy less memory. There are two types of compactions:

* Minor Compaction
  + Bigger HFiles are created by combining smaller HFiles.
  + HBase keeps the deleted file with them.
  + Increases space in memory, useful to store more data.
  + Merge sorting is used in process.
* Major Compaction
  + Data present per column family in one region is accumulated in 1 Hfile.
  + During the process, all deleted files or expired cells are deleted permanently.
  + Increased read performance of newly created Hfile.
  + Accepts lots of I/O.
  + Possibilities of traffic congestion.
  + This is also known as Write Amplification Process.
  + This process must be scheduled at a minimum bandwidth of network I/O.

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

The data model in HBase is made of different logical components such as:

* **Tables** – The HBase tables are more like logical collection of rows stored in separate partitions called Regions. Every region is then served by exactly one Region Server.
* **Rows** – A row is one instance of data in a table and is identified by a row key. Row keys are unique in a table and are always treated as a byte array.
* **Column Families** – Data in a row are grouped together as Column Families. Each column family has one or more columns and these columns in a family are stored together in a low level storage file knows as an Hfile. Column Families form the basic unit of physical storage to which certain HBase features like compaction are applied. Hence it’s important that proper care be taken when designing column families in a table.
* **Columns** – A column family is made of one or more columns. A column is identified by a column qualifier that consists of the column family name concatenated with the column using a colon (ColumnFamilyName:ColumnName). There can be multiple columns within a column family and rows within a table can have varied number of columns.
* **Cell** – A cell stores data and is essentially a unique combination of row key, column family and the column (column qualifier). The data stored in a cell is called it value and the data type is always treated as byte array.
* **Version** – The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data in a column family is configurable and this value by default is 3.

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

HBase will throw a “wrong number of arguments” error, and the put command will fail.

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

Filters in HBase shell and filter language was introduced in Apache HBase 0.92. It permits you to perform server-side filtering when accessing HBase over Thrift or within the HBase shell. When reading information from HBase using get or scan operators, you’ll be able to use custom filters to return a set of results to the client. This, however, doesn’t reduce server-side IO, it will only cut back network information measure and reduces the amount on information the client has to process. Filters are typically implemented using the Java API, however, are often used from HBase shell for testing and debugging purposes.

Syntax:

scan ‘<table name>’,{FILTER=>”<filter name>”}

For example:

hbase(main):007:0> scan ‘bulktable’,{FILTER=>”FirstKeyOnlyFilter()”}

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

The four primary data model operations are:

* Get – Get returns attributes for a specified row.
* Put – Put either adds new rows to a table (if the key is new) or can update existing rows (if the key already exists).
* Scan – Scan allow iteration over multiple rows for specified attributes.
* Delete – Delete removes a row from a table. Hbase doesn’t modify data in place, and so deletes are handled by creating new markers called tombstones. These tombstones, along with the dead values, are cleaned up on major compactions.

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

* Hbase provides a TableInputFormat, to which you provided a table scan, that splits the rows resulting from the table scan into the regions in which those rows reside.
* The map process is passed an ImmutableBytesWritable that contains the row key for a row and a Result that contains the columns for that row.
* The map process outputs its key/value pair based on its business logic.
* The reduce process builds its results but emits the row key as an ImmutableBytesWritable and a Put command to store the results back to Hbase.
* Finally, the results are stored in Hbase by the Hbase MapReduce infrastructure. (You don’t need to execute the Put commands.)

Note that you can MapReduce applications using Hbase as a data source (the source of the data you’re analyzing), a sink (the destination to where your output will be written), or both.

1. **What is Region Server?**

In Hbase, the slaves are called Region Servers. Each region server is responsible to serve a set of regions, and one region (range of rows) can be served only by one region server.

The Hbase architecture has two main services: HMaster that is responsible to coordinate the cluster and execute administrative operations, and the HRegionServer responsible for handling a subset of the table’s data.