**SESSION 14: HIVE INTRODUCTION**

**ASSIGNMENT 1**

1. **When Hive is best suited and when is it not?**

Hive is a query engine that allows you to impose a logical relational schema on top of a wide variety of physical storage mechanisms and file formats, both internal and external to the Hadoop cluster, and to run SQL like queries against those schemas as MapReduce jobs.   It also gives you the ability to manage exactly how a logical table is stored and structured as physical files on HDFS. It is designed for large analytical queries and batch transformations; it has limited write capabilities.

Hive is not suitable for OLTP (OnLine Transaction Processing) transactions. It only supports OLAP (OnLine Analytical Processing) transactions. For OLTP transactions, one needs to switch to NoSQL databases. HQL queries have higher latency, due to their conversion into MapReduce jobs.

1. **When should one use Hive over MapReduce?**

Big Data enterprises require fast analysis of data collected over a period of time. Hive is an excellent tool for analytical querying of historical data. It is to be noted that the data needs to be well organized, which would allow Hive to fully unleash its processing and analytical prowess. Querying real time data with Hive might not be the best idea, as it would be a time consuming job, defeating the original goal of fast processing.

1. **What is Hive metastore?**

The Hive metastore is a service that stores the metadata for Hive tables and partitions in a relational database. It provides clients (including Hive) access to this information using the metastore service API.

1. **How can Hive improve performance with orc file format tables?**

ORC file format stores collections of rows in one file and within the collection, the row data is stored in a columnar format. This allows parallel processing of row collections across a cluster. Each file with the columnar layout is optimized for compression and skipping of data/columns to reduce read and decompression load.

ORC goes beyond RCFile and uses specific encoders for different column data types to improve compression further, e.g. variable length compression on integers. ORC introduces a lightweight indexing that enables skipping of complete blocks of rows that do not match a query. It comes with basic statistics — min, max, sum, and count — on columns. Lastly, a larger block size of 256 MB by default optimizes for large sequential reads on HDFS for more throughput and fewer files to reduce load on the namenode.

1. **What is Thrift server and client, JDBC and ODBC driver importance in hive?**

Apache Thrift is a software framework for scalable cross-language services development, which combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Perl, C#, JavaScript, Node.js and other languages. HiveServer is a service that allows a remote client to submit requests to Hive, using a variety of programming languages, and retrieve results. It is built on Apache Thrift, therefore it is sometimes called as the Thrift server.

In the context of Hive, Java language can be used to access Hive server. The Thrift interface acts as a bridge, allowing other languages to access Hive, using a Thrift server that interacts with the Java client.

A JDBC driver is a software component enabling a Java application to interact with Hive, whereas ODBC is an open interface which can be used by any application to communicate with Hive.

1. **What is the importance of partitions in Hive?**

Hive organizes tables into partitions. Partitioning is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partitions, it is easy to query a portion of the data. It distributes the execution load horizontally.

As the data is stored as slices/parts, query response time is faster to process a small part of the data instead of looking for a search in the entire data set. For example, in a large user table where the table is partitioned by country, selecting users of country ‘IN’ will just scan one directory ‘country=IN’ instead of all the directories.

1. **What is the use of Bucketing in Hive?**

At times, even after partitioning on a particular field or fields, the partitioned file size doesn’t match with the actual expectation and remains huge and we want to manage the partitioned results into different parts. To overcome this problem of partitioning, Hive provides the concept of Bucketing, which allows users to divide table data sets into more manageable parts. When we write data in bucketed table in hive, it places the data in distinct buckets as files. Hive uses some hashing algorithm to generate a number in range of 1 to N buckets and based on the result of hashing, data is placed in a particular buckets as a file.

Bucketed tables offer more efficient sampling than non-bucketed tables. As the data files are equal sized parts, map sized joins are performed faster on bucketed tables. Bucketing concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. Similar to partitioning, bucketing provides faster query responses.

1. **What is the difference between static partitioning and dynamic partitioning in hive?**

The major difference between static and dynamic partitions is that in case of a static partition, the name of the partition is hardcoded in the insert statement, whereas with a  
dynamic partition, Hive will automatically determine the partition based on the value  
of the partition field.

Static partitioning needs to be applied when we know data (which is to be inserted) belongs to which partition. In static partitioning, every partitioning needs to be backed with individual hive statement which is not feasible for large number of partitions as it will require writing a lot of hive statements.

In that scenario, dynamic partitioning is suggested as we can create as many number of partitions with single hive statement.