**ACD\_BDD\_Session\_14\_Assignment\_1\_Main**

**Problem Statement**

**Give brief answers to the below questions:**

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

Hive is best suited for Data Warehousing Applications where data is stored, mined and reporting is done based on processing. As most Data Warehousing applications are base in relational database models, so Hive bridges the gap between these applications and Hadoop.

Even though Hive gives SQL dialects it does not give SQL like latency as it ultimately runs MapReduce programs underneath. As we know, MapReduce framework is built for batch processing jobs, it has high latency; even the fastest Hive query would take several minutes to get executed on relatively smaller set of data in few megabytes.

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

We should use Hive over MapReduce when we need easy and effective data aggregation, ad-hoc querying and analysis of huge volume of data.

**3. What is Hive metastore?**

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

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

-> The Optimized Row Columnar (ORC) file format provides a highly efficient way to store Hive data. It was designed to overcome limitations of the other Hive file formats. Using ORC files improves performance when Hive is reading, writing, and processing data.

Compared with RCFile format, for example, ORC file format has many advantages such as:

* a single file as the output of each task, which reduces the NameNode's load
* Hive type support including datetime, decimal, and the complex types (struct, list, map, and union)
* light-weight indexes stored within the file
* skip row groups that don't pass predicate filtering
* seek to a given row
* block-mode compression based on data type
* run-length encoding for integer columns
* dictionary encoding for string columns
* concurrent reads of the same file using separate RecordReaders
* ability to split files without scanning for markers
* bound the amount of memory needed for reading or writing
* metadata stored using Protocol Buffers, which allows addition and removal of fields

**5. What is thrift server and client, jdbc and odbc driver importance in hive?**

-> Thrift allows clients using languages including Java, C++, Ruby and many others, to programmatically access Hive remotely.

**6. What is the importance of partition in hive?**

-> When any query is fired, Hive reads the entire dataset. If the dataset is large, then it takes a long time and it become a bottleneck for MapReduce jobs. To overcome this issue, Hive allows a special option called Partitioning. It is recommended to partition the Hive table while analyzing large datsets. Based on the values of particular columns segregate the input record into different files. We can partition multiple columns also. Instead of analyzing vast amount of data, we can partition and analyze target data to get desired output results. It is the best approach to improve the query performance on large tables.

**7. What is the use of bucketing in hive?**

To overcome the problem of over partitioning, Hive provides bucketing concept, another technique for decomposing table datasets into more manageable parts.

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

In static partitioning we need to specify the partition column value in all the LOAD statements.

Suppose we are having partition on column country for table t1 (userid, name, occupation, country), so each time we need to

Provide profession value.

hive> LOAD DATA INPATH '/hdfs path of the file' INTO TABLE t1 PARTITION(country="Trump")

hive> LOAD DATA INPATH '/hdfs path of the file' INTO TABLE t1 PARTITION(country="Hillary")

Dynamic partition allows us not to specify partition column value each time. The approach we follow is as below:

1. Create a non-partitioned table x and insert data into it.

2. Now create a table y partitioned on intended column (say candidate).

3. Load data in y from x as below:

hive> INSERT INTO TABLE y PARTITION(country) SELECT \* from x;

4. Make sure that partitioned column is always the last one in non partitioned table (as we are having country column in y)