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

Ans: Use of HiveQL saves the user from writing complex MapReduce functions. Hive allows programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express the logic in HiveQL.

The limitation of Hive is It provides no support for updation/ deletion of data. Data needs to be loaded from a file using LOAD command.

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

Ans :It take very less time to write Hive Query compared to Map Reduce code.

For example, the word count problem which takes around 50 lines of code can be written in 5 lines in Hive. So, you save time.

It was developed so that people who have SQL knowledge can write the MR Job.

It supports many SQL Syntax which means that it is possible to integrate Hive with existing BI tools.

So, business Analyst, or non-java guys can also work on the large data set. Similarly, the code which were earlier used in RDBMS can be used in Hive (with changes). With MR, it is a separate exercise in itself.

It is very easy to write query involving joins (if there are few joins) in Hive.

Comparing it with MR code, you have to do caching of data and do several operations to reach to the same point.

It has very low maintenance and is very simple to learn & use (low learning curve).

**3. What is Hive metastore?**

Ans : 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?**

Ans : 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?**

|  |  |
| --- | --- |
| The Hive server is exposed as a Thrift service, so it’s possible to interact with it using any programming language that supports Thrift. There are third party projects providing clients for Python and Ruby. | |
| Using jdbc driver a Java application will connect to a Hive server running in a separate process at the given host and port. | | |
| An ODBC driver allows applications that support the ODBC protocol (such as business intelligence software) to connect to Hive. |

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

Ans : With its proven ability to speed performance, partitioning is a must-have feature in the tool set of any Big Data query engine. Hive is no exception; it has had partition support since its early versions. Although this blog will touch on static partitioning, it will primarily focus on when and how to best employ dynamic partitioning—a method we believe is often underutilized as an effective means for partitioning data and improving performance.

Dynamic and Static Partitioning

Hive supports two partitioning models: static and dynamic. When either is used, queries are run against only a portion of the data, providing significant performance gains. But what type of partitioning should you use, and when?

Static Partitioning—Used when the values for partition columns are known well in advance of loading the data into a Hive table

Dynamic Partitioning—Used when the values for partition columns are known only during loading of the data into a Hive table

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

Ans : 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

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

Ans : in static partitioning we need to specify the partition column value in each and every LOAD statement.

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

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

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

dynamic partition allow us not to specify partition column value each time. the approach we follows is as below:

create a non-partitioned table t2 and insert data into it.

now create a table t1 partitioned on intended column(say country).

load data in t1 from t2 as below:

hive> INSERT INTO TABLE t2 PARTITION(country) SELECT \* from T1;

make sure that partitioned column is always the last one in non partitioned table(as we are having country column in t2)