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.Hive is best suited for Data Warehousing applications where data is structured, static and formatted. Hive is not OLTP (Online Transaction Processing) tool for sure but its closer to OLAP (Online Analytical Processing) but again it conflicts with word Online in it due to high latency.Hive does not provide row wise update and insert which is a biggest disadvantage of using it.

**Hive Is**

Hive is a data warehousing tool based on Hadoop. As we know Hadoop provides massive scale out on distributed infrastructure with high degree of fault tolerance for data storage and processing. Hadoop uses Map Reduce algorithm to process huge amount of data with minimal cost as it does not require high end machines to process such amount of data. Hive processor converts most of its queries into a Map Reduce job which runs on Hadoop cluster. Hive is designed for easy and effective data aggregation, ad-hoc querying and analysis of huge volumes of data.

**Hive Is Not**

Even though Hive gives SQL dialect it does not give SQL like latency as it ultimately runs Map Reduce programs underneath. As we all know, Map Reduce 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. We cannot simply compare the performance of traditional SQL systems like Oracle, MySQL or SQL Server as these systems are meant to do something and Hive is meant to do else. Hive aims to provide acceptable (but not optimal) latency for interactive querying over small data sets for sample queries

Like we said earlier, hive is not an OLTP (Online transaction Processing) application and not meant to be connected with systems which needs interactive processing. It is meant to be used to process batch jobs on huge data which is immutable. A good example of such kind of data would be Web logs, Application Logs, call data records (CDR) etc.

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

MapReduce is a powerful programming model for parallelism based on rigid procedural structure. Hadoop MapReduce allows programmers to filter and aggregate data from HDFS to gain meaningful insights from big data. The Map and Reduce algorithmic functions can also be implemented using C,  [Python](https://www.dezyre.com/data-science-in-python/36)and Java. The only drawback to use the coding approach of [Hadoop MapReduce](https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83) is that Hadoop developers need to write several lines of basic java code requiring extra effort and time for code review and QA.

1. What is Hive metastore?

Hive data (not metadata) is spread across Hadoop HDFS DataNode servers. Typically, each block of data is stored on 3 different DataNodes. The NameNode keeps track of which DataNodes have which blocks of actual data.

For a Hive production environment, the metastore service should run in an isolated JVM. Hive processes can communicate with the metastore service using Thrift. The Hive metastore data is persisted in an ACID database such as Oracle DB or MySQL. You can use SQL to find out what is in the Hive metastore:

Here are the tables in the Hive metastore:

select table\_name from user\_tables;

DBS

DATABASE\_PARAMS

SEQUENCE\_TABLE

SERDES

TBLS

SDS

CDS

BUCKETING\_COLS

TABLE\_PARAMS

PARTITION\_KEYS

SORT\_COLS

SD\_PARAMS

COLUMNS\_V2

SERDE\_PARAMS

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

The ORC File (Optimized Row Columnar) format provides a more efficient way to store relational data than the RC File, reducing the data storage format by up to 75% of the original. The ORC file format performs better than other Hive files formats when Hive is reading, writing, and processing data. Specifically compared to the RC File, ORC takes less time to access data and takes less space to store data. However, the ORC file increases CPU overhead by increasing the time it takes to decompress the relational data. Also, the ORC File format comes with the Hive 0.11 version and cannot be used with previous versions.

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

**Thrift**

The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages.

**JDBC**

HiveServer2 has a JDBC driver. It supports both embedded and remote access to HiveServer2. Remote HiveServer2 mode is recommended for production use, as it is more secure and doesn't require direct HDFS/metastore access to be granted for users.

**ODBC**

The Hive ODBC Driver is a software library that implements the Open Database Connectivity (ODBC) API standard for the Hive database management system, enabling ODBC compliant applications to interact seamlessly (ideally) with Hive through a standard interface. This driver will NOT be built as a part of the typical Hive build process and will need to be compiled and built separately according to the instructions below.

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

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

Tables or partitions are sub-divided into buckets, to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

For example, a table named Tab1 contains employee data such as id, name, dept, and yoj (i.e., year of joining). Suppose you need to retrieve the details of all employees who joined in 2012. A query searches the whole table for the required information. However, if you partition the employee data with the year and store it in a separate file, it reduces the query processing time. The following example shows how to partition a file and its data.

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

Uses of bucketing in Hive.

* Bucketed tables offer efficient sampling than by non-bucketed tables. With sampling we can try out queries on a fraction of data for testing and debugging purpose when the original datasets are very huge.
* As the data files are equal sized parts map side join will faster on bucketed tables. In map side join mapper processing a bucket of the left side table knows that the matching rows in the right table will be in its corresponding bucket so it only retrieves that bucket.
* Similar to partitioning, bucketed tables provide faster query responses than non-bucketed tables.
* It also provides the flexibility to keep the records in each bucket to be sorted by one or more columns.

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

partitioning in Hive is very useful to purne data during query to reduce query times.

Partitions are created when data is inserted into table. Depending on how you load data you would need partitions. Usually when loading files (big files) into Hive tables static partitions are preferred. That saves your time in loading data compared to dynamic partition. You "statically" add a partition in table and move the file into the partition of the table. Since the files are big they are usually generated in HDFS. You can get the partition column value form the filename, day of date etc without reading the whole big file.

Incase of dynamic partition whole big file i.e. every row of the data is read and data is partitioned through a MR job into the destination tables depending on certain field in file. So usually dynamic partition are useful when you are doing sort of a ETL flow in your data pipeline. e.g. you load a huge file through a move command into a Table X. then you run a inert query into a Table Y and partition data based on field in table X say day , country. You may want to further run a ETL step to partition the data in country partition in Table Y into a Table Z where data is partitioned based on cities for a particular country only. etc.

Thus depending on your end table or requirements for data and in what form data is produced at source you may choose static or dynamic partition.