Objective

- ✓ What is Hive?
- ✓ Hive v/s RDBMS
- ✓ Architecture
- ✓ Components
- ✓ Metastore
- ✓ Data Types
- ✓ HiveQL
- ✓ Tables
- ✓ Partition
- ✓ Bucket
- ✓ Sorting
- ✓ Joins
- ✓ User Defined Function
- ✓ Limitation of Hive

What is Hive?

- ✓ Data warehousing package built on top of Hadoop.
- ✓ Hive works with Hadoop to allow you to query and manage large-scale data using a familiar SQL-like interface.
- ✓ For managing and querying structured data.
- ✓ Allows you to define a structure for your unstructured Bigdata.
- ✓ Simplifies analysis and queries with an SQL-like scripting called **HiveQL**.
- ✓ Hive is most suited for data warehouse applications, where relatively static data is analysed, fast response times are not required and when the data is not changing rapidly.
- ✓ Hive is best suited for applications, where a large data set is maintained and mined for insights, reports, data analysis etc.
- ✓ Hive runs as a client application that processes HiveQL queries, convert them into MapReduce jobs and submits these to a Hadoop cluster.
- ✓ Developed by Facebook and contributed to community.

Hive is Not….

…A relational database. Hive uses a database to store metadata, but the data that Hive processes is stored in HDFS.

…. Designed for on-line transaction processing. Latency for Hive queries is generally high (even for small jobs).

…. Suited for real-time queries and row-level updates. It is best suited for batch jobs over large set of data.

Advantages….

- ✓ Targeted towards users comfortable with SQL.
- ✓ Abstracts complexity of Hadoop.
- ✓ No need to learn Java or Hadoop API.

Hive v/s RDBMS

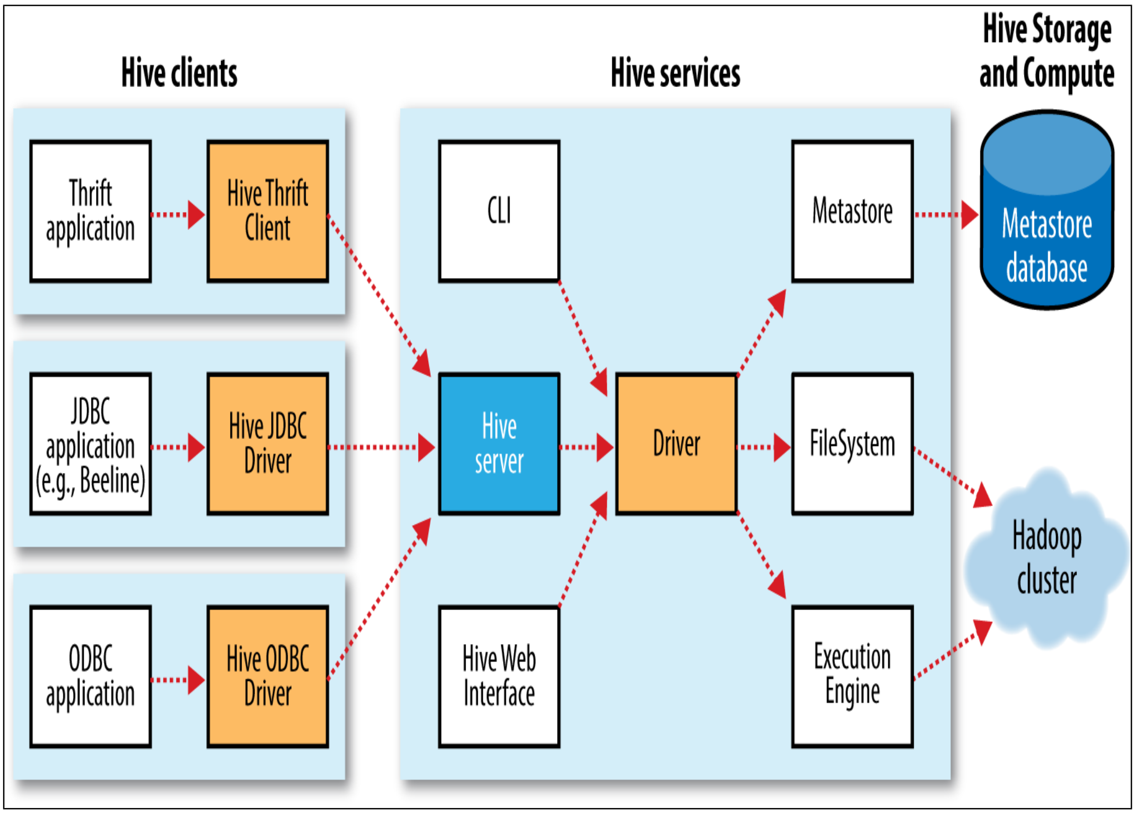
- ✓ **Schema on write:** In RDBMS, table's schema is enforced at data load time. This design is called schema on write.

- ✓ **Schema on read**: Hive does not verifies the data when it is loaded, but rather when a query is issued. This is called **Schema on read**.

- ✓ If data schema mismatch occurs at query time, hive show NULL value.

- ✓ Schema on read is useful for fast data loading. It is just a file copy or move.

Hive Architecture



Hive Components

1. **Metastore:** Central repository of Hive metadata.

2. **Execution Engine:**

✓ MapReduce execution engine is default engine for Hive to execute jobs.
✓ It contains compiler and optimizer.

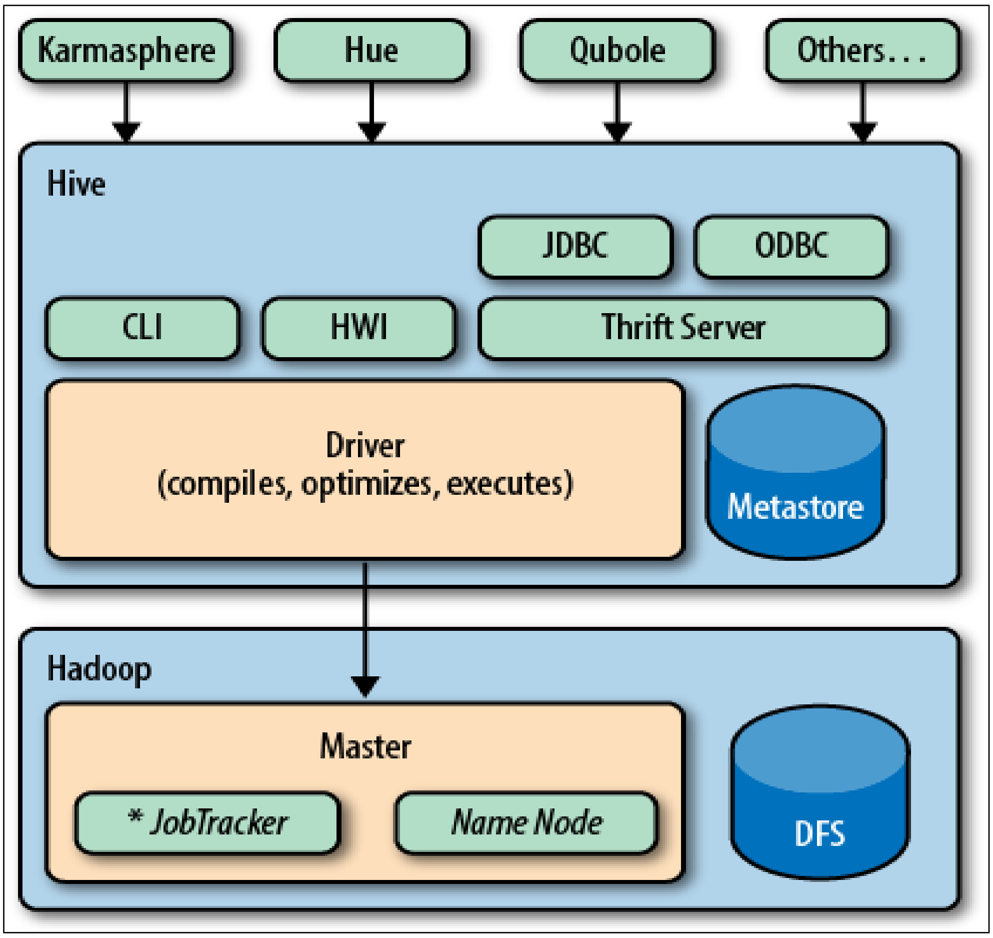
3. **Hive Services:** Used to connect with Hive

i. Hive CLI (Old): The command-line interface to Hive (the shell). This is the default service.

ii. Beeline CLI (New): It's a JDBC client that is based on the SQLLine CLI.

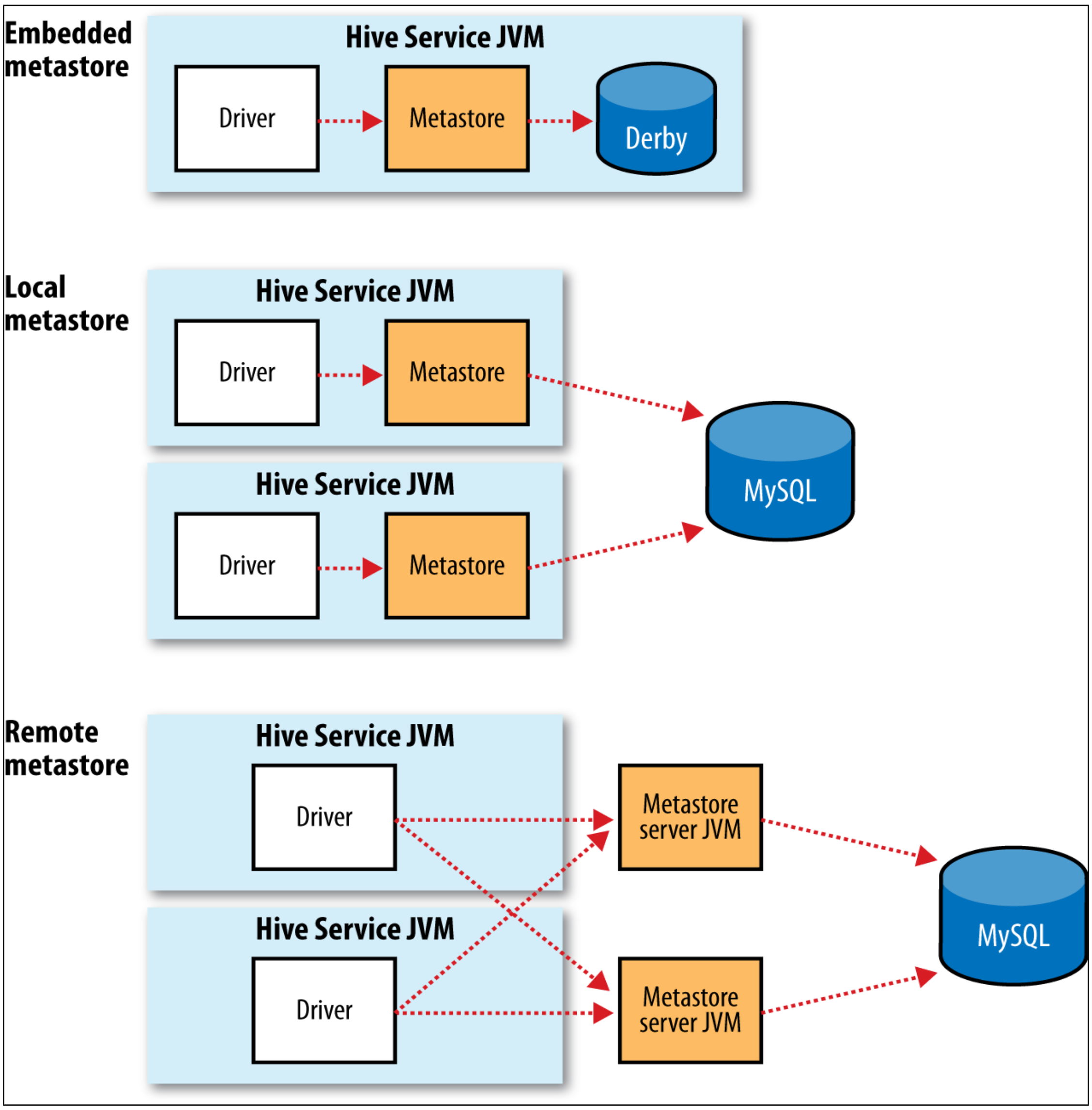
iii. hiveserver2: Runs Hive as a server exposing a **Thrift service**, enabling access to clients programs written in different languages.

iv. HWI: The Hive Web Interface. A simple web interface that can be used as an alternative to the CLI without having to install any client software.



Hive Metastore

- ✓ The Metastore is the central repository of Hive metadata.
- ✓ The Metastore is a separate relational database (usually a MySQL instance) where Hive persists table schemas and other system metadata.
- ✓ The Metastore is divided into two pieces: a service and the backing store for the data.
- ✓ By default, the Metastore service runs in the same JVM as the Hive service and contains an embedded Derby database instance in local disk. This is called the **embedded** Metastore configuration.
- ✓ In embedded Metastore, we can have only one Hive session open at a time that access the same Metastore. It means only single user can access that Metastore at a time.
- ✓ The solution to supporting multiple sessions (and therefore multiple users) is to use a standalone database. It is called **local** Metastore configuration.



Hive Data Types

Category	Type	Data Type	Value/Length
Primitive	Boolean	BOOLEAN	TRUE/FALSE
	Integers	TINYINT	1 Byte
		SMALLINT	2 Byte
		INT	4 Bytes
		BIGINT	8 Bytes
	Floating Point	FLOAT	Single Precision
		DOUBLE	Double Precision
		DECIMAL	Arbitrary-precision signed decimal number
	String	STRING	Unbounded variable-length character string
		VARCHAR	Variable-length character string
		CHAR	Fixed-length character string
	Date	DATE	Date
		TIMESTAMP	Timestamp with nanosecond precision
Complex	Array	ARRAY	An ordered collection of fields.
	Map	MAP	An unordered collection of key-value pairs
	Structure	STRUCT	A collection of named fields
	Union	UNION	A value that may be one of a number of defined data types.

Hive Table

✓ A Hive table consists of:

- Data: typically a file or a group of files in HDFS
- Schema: in the form of metadata stored in a Relational database

1. Internal / Managed Table

- Create the table in Hive warehouse directory.
- Default warehouse directory is **/user/hive/warehouse**
- Metadata and Data is managed by Hive.
- Hive delete the table and metadata.
- Data reside in HDFS and metadata store in Metastore of Hive.
- When data is not using by multiple users or not updated by multiple resources, we use internal tables.
- If we are processing our data with Hive only then Internal table should be used.

2. External Table

- Create the table in another HDFS location and not in warehouse directory.
- Data not managed by Hive.
- Hive does not delete the table (or HDFS files) even when the tables are dropped. It leaves the table untouched and only metadata about the tables are deleted.
- Data reside in HDFS and metadata store in Metastore of Hive.
- When data size is large or data is uses/updated by multiple resources, we use External tables.
- If we want to use Hive and some other tools to processing our data then external table should create.
- If we wish to associate multiple schemas with the same dataset we should use external table.

HiveQL

- ✓ It provides the basic SQL-like operations.
- ✓ Filtering the data based on condition in where clause.
- ✓ We can store the result of a query in Hadoop dfs directory or into another table.
- ✓ We can manage tables and partitions. (create, drop, alter)
- ✓ Supports multi table inserts.
- ✓ Converts SQL queries into MapReduce jobs.
- ✓ Also supports plugging custom MapReduce scripts into queries.
- ✓ Hive query language supports for familiar SQL operations including joins, subqueries, order by, sort by and so on.

Partitions

- ✓ Partition means dividing a table into a coarse grained parts based on the value of a partition column such as a date.
- ✓ This make Hive faster to do queries on slice of data.
- ✓ A table may be partitioned in multiple dimensions
- ✓ Partitions are defined at table creation time using the PARTITIONED BY clause which takes a list of column definitions.
- ✓ Column used for partition is called Partitioned Columns.
- ✓ When we load the data into a partitioned table, the partition values are specified explicitly.

Example:

```
CREATE TABLE logs (ts BIGINT, line STRING)
PARTITIONED BY (dt STRING, country STRING);


LOAD DATA LOCAL INPATH 'input/hive/partitions/file1'
INTO TABLE logs
PARTITION (dt='2001-01-01', country='GB');
```

Buckets

- ✓ Buckets give extra structure to the data that may be used for efficient queries.
- ✓ To populate the bucketed table, we need to set the **hive.enforce.bucketing** property to true
- ✓ Physically, each bucket is just a file in the table (or partition) directory.
- ✓ A job will produce as many buckets (output files) as reduce tasks.
- ✓ Buckets are created on the basis of hash function of columns of the column.
- ✓ **Bucket = [hash of the column] % number of buckets.**
- ✓ We use the **CLUSTERED BY** clause to specify the columns to bucket on and the number of buckets.

Example:

```
CREATE TABLE bucketed_users (id INT, name STRING)
CLUSTERED BY (id) INTO 4 BUCKETS;
```

Sorting

- ✓ ORDER BY: Parallel total sort of the input.
- ✓ SORT BY: Produces a sorted file per reducer.
- ✓ DISTRIBUTE BY: Control which reducer a particular row will go.
- ✓ CLUSTER BY : If the columns for SORT BY and DISTRIBUTE BY are the same, then we can use cluster by.

Joins

- ✓ A single join is implemented as a single MapReduce job.
- ✓ Multiple joins can be performed in less than one MapReduce job per join if the same column is used in the join condition.
- ✓ Hive try to minimize the number of MapReduce jobs to perform the joins.
- ✓ You can see how many MapReduce jobs Hive will use for any particular query by prefixing it with the **EXPLAIN** keyword

Types of Joins

1. Inner Join: Default join
2. Outer Join: Outer joins allow you to find non matches in the tables being joined.

- ✓ Left outer join
- ✓ Right outer join
- ✓ Full outer join

3. Map Join: If one table is small enough to fit in memory, Hive can load it into memory to perform the join in each of the mappers.

User Defined Function

- ✓ When desired output can’t available using built in function, we have to write our own function to perform task. It is called User defined function.
- ✓ UDFs have to be written in Java.

Types of UDF

1. **UDF :**

- ✓ Regular User defined function.
- ✓ A UDF operates on a single row and produces a single row as its output.
- ✓ Most functions, such as mathematical functions and string functions, are of this type.

2. **UDAF:**

- ✓ User-defined aggregate functions
- ✓ A UDAF works on multiple input rows and creates a single output row.
- ✓ Aggregate functions include such functions as COUNT and MAX.

3. **UDTF:**

- ✓ User-defined table-generating functions.
- ✓ A UDTF operates on a single row and produces multiple rows, a table as output.

Limitation of Hive

- ✓ No record level insert, update or delete.
- ✓ Not design for online transaction processing.
- ✓ Hive queries have higher latency, due to the start-up overhead for MapReduce jobs.
- ✓ Hive does not provide transactions.
- ✓ Hive doesn’t provide crucial features required for OLTP, Online Transaction Processing.
- ✓ It’s closer to being an OLAP tool, Online Analytic Processing.