Hive

Data Warehouse on Hadoop

Agenda

- ✓ Introduction to Hive
- ✓ Hive Architecture and Hive Shell
- ✓ Configuring hive
- ✓ Hive services and execution flow
- ✓ Schema on read vs Schema on write
- ✓ Serde
- ✓ Data Partitioning
- ✓ Hive Application

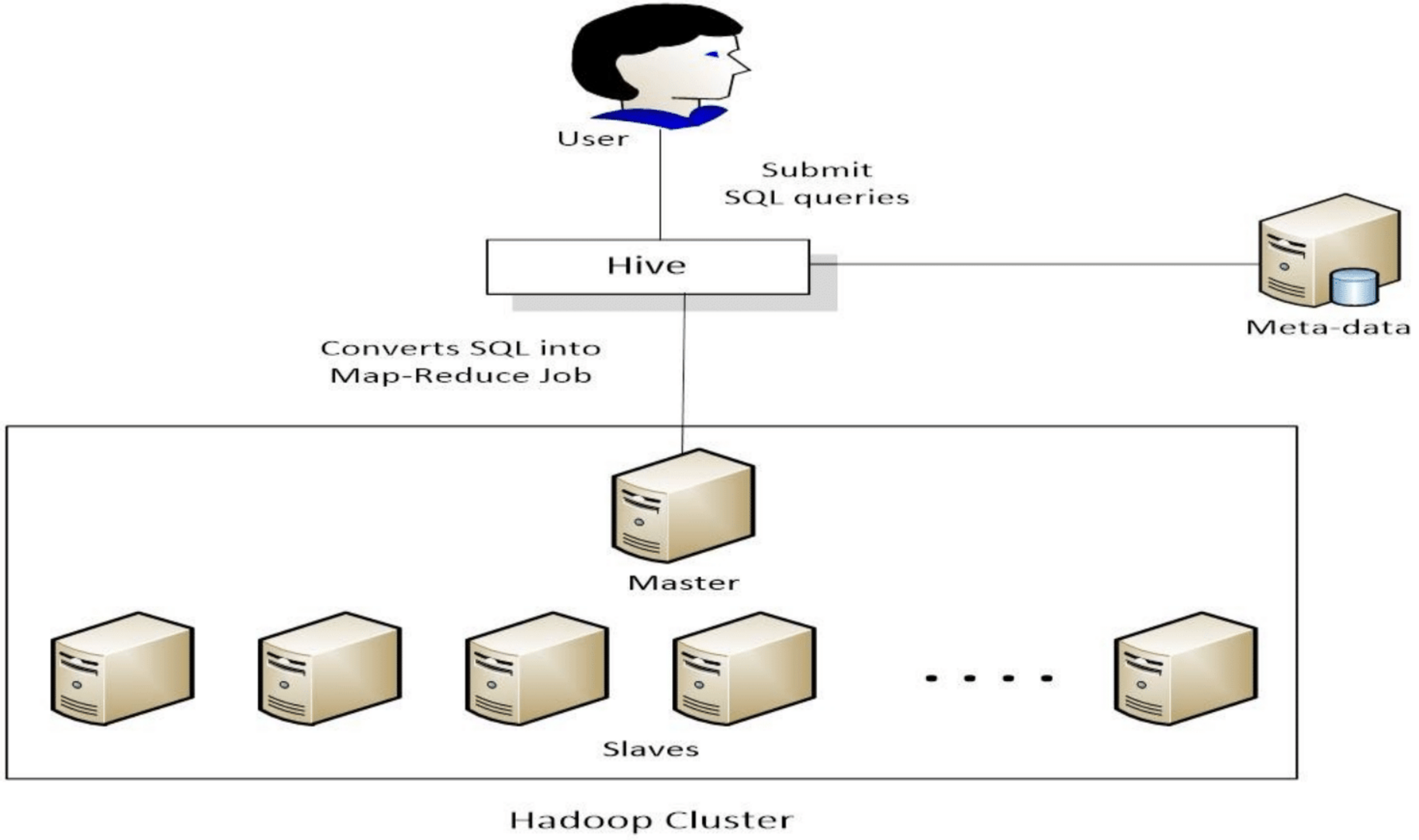
Motivation

- Analysis of Data made by both engineering and non-engineering people.
- The data is growing and growing fast.
- Current solution are not capable, not scalable, Expensive and Proprietary.
- Hadoop supports data-intensive distributed applications. However...
  - Map-reduce hard to program (users know SQL/bash/python)
  - No schema

Introduction to Hive

- A data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.
  - Structure
  - Access to different storage
  - Query execution via Map Reduce
- Key Building Principles:
  - SQL is a familiar language
  - Extensibility – Types, Functions, Formats, Scripts
  - Performance
- Hive is designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data

Hive Architecture



HiveQL

- HiveQL is the Hive query language. Like all SQL dialects in widespread use, it doesn't fully conform to any particular revision of the ANSI SQL standard.
- No support for row level inserts, updates, and deletes.
- Hive doesn't support transactions.
- Adds extensions to provide better performance in the context of Hadoop

Examples – DDL & DML Operations

- **CREATE TABLE** sample (foo INT, bar STRING) **PARTITIONED BY** (ds STRING);
- **SHOW TABLES** '.*s';
- **DESCRIBE** sample;
- **ALTER TABLE** sample **ADD COLUMNS** (new_col INT);
- **DROP TABLE** sample;
- **LOAD DATA LOCAL INPATH** './sample.txt' **OVERWRITE INTO TABLE** sample **PARTITION** (ds='2012-02-24');
- **LOAD DATA INPATH** '/user/hdadmin/sample.txt' **OVERWRITE INTO TABLE** sample **PARTITION** (ds='2012-02-24');

Built-in Functions

- Mathematical: round, floor, ceil, rand, exp...
- Collection: size, map_keys, map_values, array_contains.
- Type Conversion: cast.
- Date: from_unixtime, to_date, year, datediff...
- Conditional: if, case, coalesce.
- String: length, reverse, upper, trim...

The Hive Shell

- The shell is the primary way that we interact with the Hive, By issuing commands in the HiveQL.
- Hive QL (HQL), a dialect of the SQL. It is heavily influenced by the MySql.
- You can also run the Hive shell in the non-interactive mode. the -f option runs the command in the specified file

  For ex hive -f script.q

Working with Hive

- Starting the Hive Shell:
  - ✓ Start a terminal and run

```
$ cd hive-VERSION/
$ bin/hive
```

- Should see a prompt like:

```
hive>
```

- Creating tables

```
hive> SHOW TABLES;
hive> CREATE TABLE shakespeare (freq INT, word STRING)
        ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
        STORED AS TEXTFILE;
hive> DESCRIBE shakespeare;
```

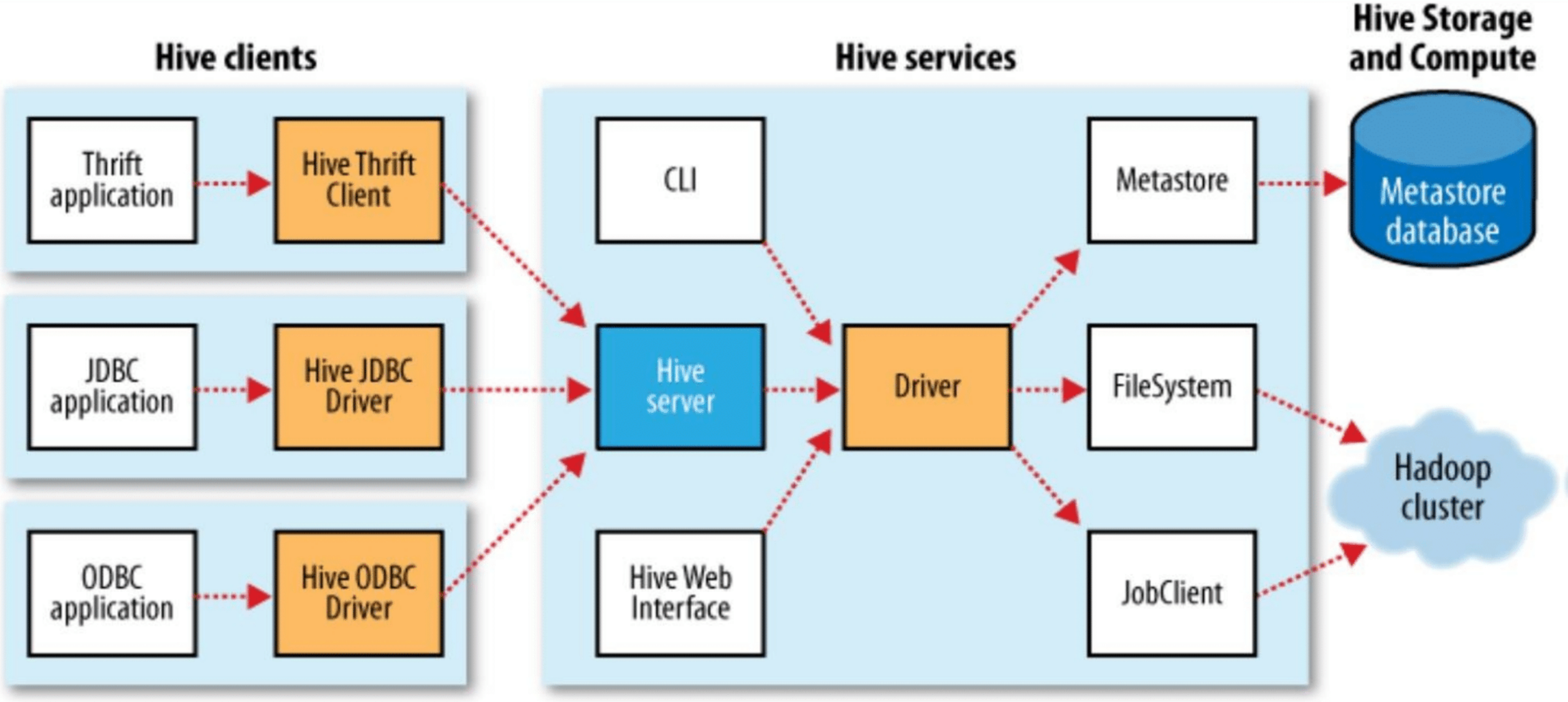
Hive Services

- The Hive shell is only one of several services that you can run using the hive command .
- You can specify the service to run using the -- service option.
  - Command Line Interface
  - jar
  - metastore

Hive client

- If you run Hive as a server (hive --service hiveserver), then there are a number of different mechanisms for connecting to it from applications
- Thrift Client
- JDBC Driver
- ODBC Driver

Hive Execution Flow



The Metastore

- Metastore is the central repository of Hive metadata.
- Metastore is divided into two pieces: a service and the backing store for the data.
- Metastore service runs in the same JVM as the Hive service and contains an embedded Derby database instance backed by the local disk. This is called the **embedded** metastore configuration

Schema on Read Versus Schema on Write

- In a traditional database, a table's schema is enforced at data load time. If the data being loaded doesn't conform to the schema, then it is rejected. This design is sometimes called ***schema on write***, since the data is checked against the schema when it is written into the database.
- Hive, on the other hand, doesn't verify the data when it is loaded, but rather when a query is issued. This is called ***schema on read***.
- Schema on read makes for a very fast initial load, since the data does not have to be read, parsed, and serialized to disk in the database's internal format.

Serde

- The SerDe interface allows you to instruct Hive as to how a record should be processed.
- SerDe is a combination of a **Serializer** and a **Deserializer** (hence, Ser-De).
- The Deserializer interface takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate.
- The Serializer will take a Java object that Hive has been working with, and turn it into something that Hive can write to HDFS or another supported system.
- Deserializers are used at query time to execute SELECT statements, and Serializers are used when writing data, such as through an INSERT-SELECT statement

Pros

- Hive is a great supplement of Hadoop to bridge the gap between low-level interface requirements required by Hadoop and industry-standard SQL which is more commonplace.
- Support of ODBC/JDBC which enables the connectivity with many commercial Business Intelligence and/or ETL tools.
- Having Intelligence Optimizer (naïve rule-based) which optimizes logical plans by rewriting them into more efficient plans.
- Support of Table-level Partitioning to speed up the query times.
- A great design decision by using traditional RDBMS to keep Metadata information (Metastore) which is more optimal and proven for random access.

Cons

- HiveSQL is not 100% ANSI-Compliant SQL.
- No support for UPDATE & DELETE
- Rule-based Optimizer doesn’t take into account available resources in generating logical and physical plans.
- No Access Control Language supported

Data Partitioning

Hive Application

- Log Processing