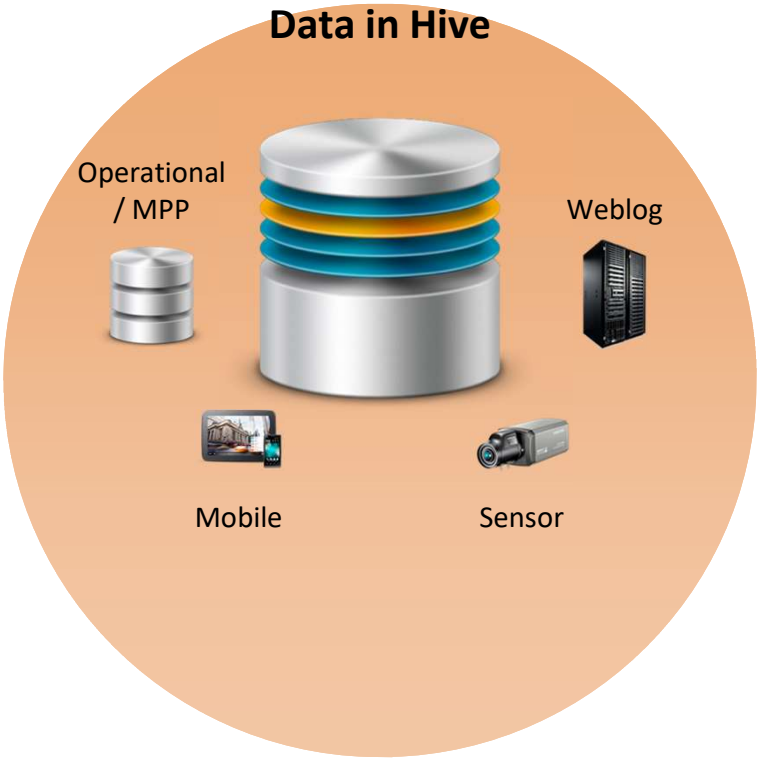
Hive Programming

Topics Covered

- About Hive
- Comparing Hive to SQL
- Hive Architecture
- Submitting Hive Queries
- Defining Tables
- Loading Data into Hive
- Performing Queries
- *Lab: Understanding Hive Tables*
- Hive Partitions, Buckets, and Skewed
- *Demo: Understanding Partitions and Skew*
- Sorting Data
- *Lab: Analyzing Big Data with Hive*
- Hive Join Strategies
- *Demo: Computing ngrams*
- *Lab: Joining Datasets in Hive*
- *Lab: Computing ngrams of Emails in Avro Format*

About Hive

Store and Query all Data in Hive



Use Existing SQL Tools and Existing SQL Processes

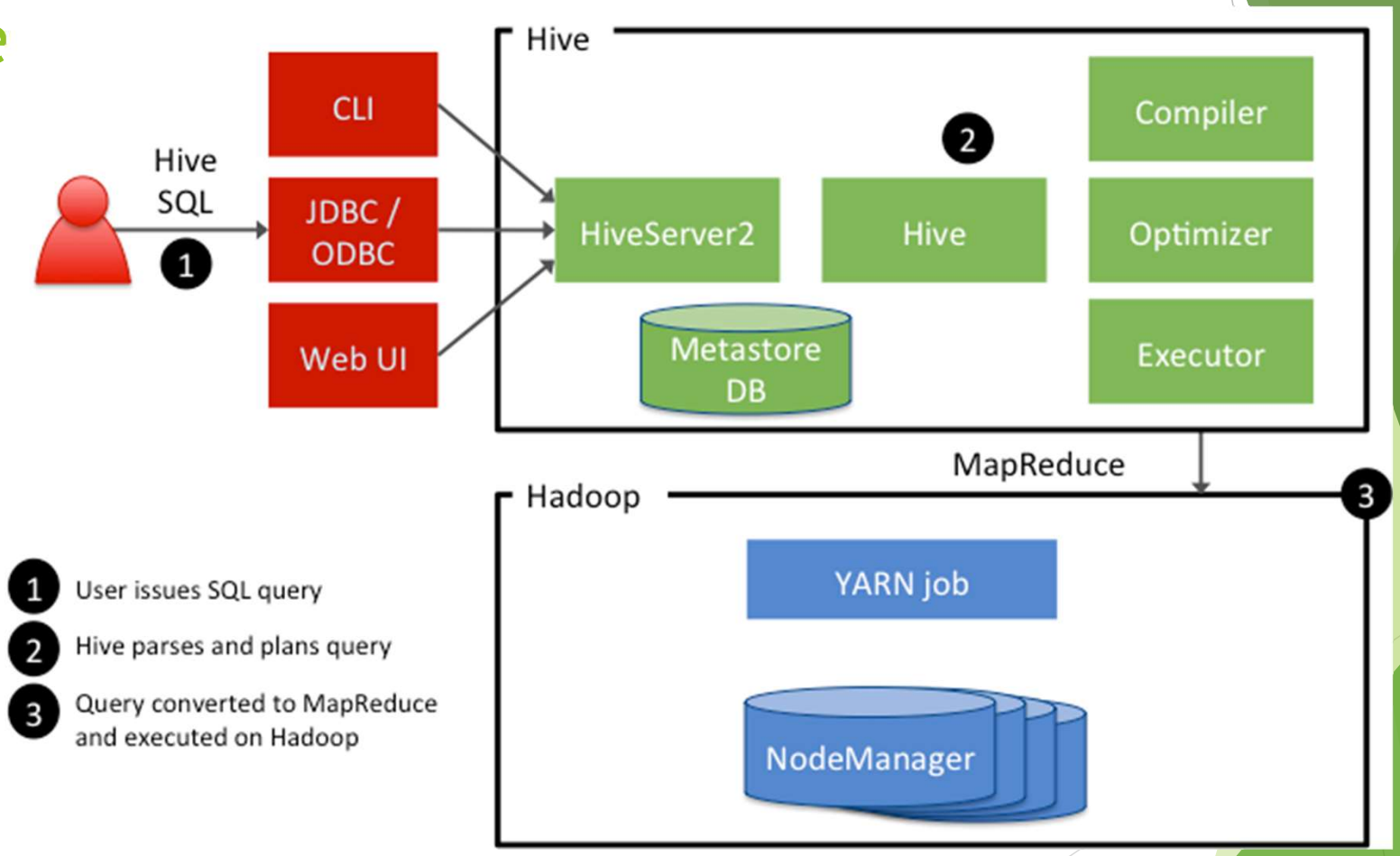
About Hive – cont.

- It is a data warehouse system for Hadoop
- It maintains metadata information about your big data stored on HDFS
- It treats your big data as tables
- It performs SQL-like operations on the data using a scripting language called **HiveQL**

Hive's Alignment with SQL

SQL Datatypes	SQL Semantics
INT	SELECT, LOAD, INSERT from query
TINYINT/SMALLINT/BIGINT	Expressions in WHERE and HAVING
BOOLEAN	GROUP BY, ORDER BY, SORT BY
FLOAT	CLUSTER BY, DISTRIBUTE BY
DOUBLE	Sub-queries in FROM clause
STRING	GROUP BY, ORDER BY
BINARY	ROLLUP and CUBE
TIMESTAMP	UNION
ARRAY, MAP, STRUCT, UNION	LEFT, RIGHT and FULL INNER/OUTER JOIN
DECIMAL	CROSS JOIN, LEFT SEMI JOIN
CHAR	Windowing functions (OVER, RANK, etc.)
VARCHAR	Sub-queries for IN/NOT IN, HAVING
DATE	EXISTS / NOT EXISTS

Hive


Hive
CLI
Hive
SQL
JDBC /
ODBC
Web UI
1
HiveServer2
2
Hive
Compiler
Optimizer
Executor
Metastore
DB
MapReduce
Hadoop
3
YARN job
NodeManager
1 User issues SQL query
2 Hive parses and plans query
3 Query converted to MapReduce
and executed on Hadoop

Submitting Hive Queries

- Hive CLI
  - Traditional Hive “thick” client
  - ```
    $ hive
    hive>
    ```
- Beeline
  - A new command-line client that connects to a HiveServer2 instance
  - ```
    $ beeline –u url –n username –p password
    beeline>
    ```

Defining a Hive-Managed Table

```
CREATE TABLE customer (
          customerID INT,
          firstName STRING,
          lastName STRING,
          birthday TIMESTAMP
  ) ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ',';
```

Defining an External Table

```
CREATE EXTERNAL TABLE salaries (
   gender string,
   age int,
   salary double,
   zip int
 ) ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ',';
```

Defining a Table LOCATION

```
CREATE EXTERNAL TABLE SALARIES (
   gender string,
   age int,
   salary double,
   zip int
 ) ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ','
   LOCATION '/user/train/salaries/';
```

Loading Data into Hive

```
LOAD DATA LOCAL INPATH ’/tmp/customers.csv'
OVERWRITE INTO TABLE customers;
```

```
LOAD DATA INPATH '/user/train/customers.csv'
OVERWRITE INTO TABLE customers;
```

```
INSERT INTO TABLE birthdays
        SELECT firstName, lastName, birthday
        FROM customers
        WHERE birthday IS NOT NULL;
```

Performing Queries

```
SELECT * FROM customers;
```

```
FROM customers
        SELECT firstName, lastName, address, zip
        WHERE orderID > 0
        ORDER BY zip;
```

```
SELECT customers.*, orders.*
        FROM customers
        JOIN orders ON
        (customers.customerID = orders.customerID);
```

Lab: Understanding Hive Tables

Hive Partitions

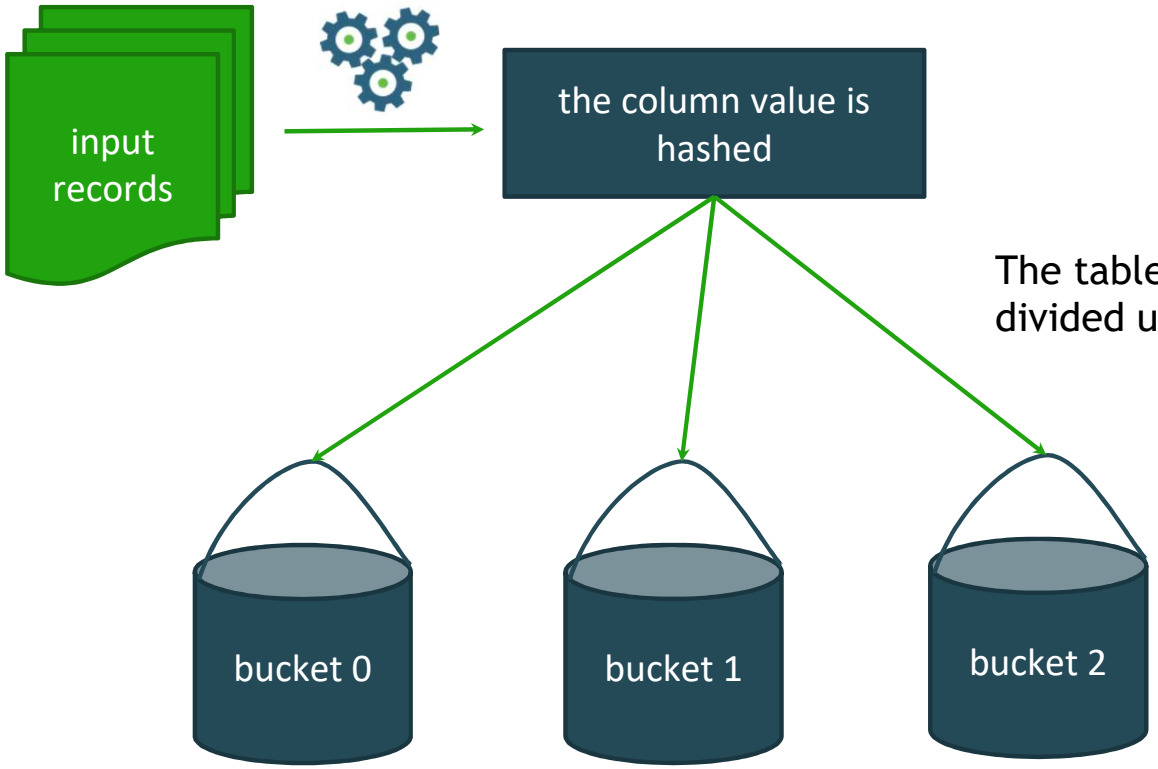
- Use the **partitioned by** clause to define a partition when creating a table:

```
create table employees (id int, name string, salary double)
partitioned by (dept string);
```

- Subfolders are created based on the partition values:

```
/apps/hive/warehouse/employees
    /dept=hr/
    /dept=support/
    /dept=engineering/
    /dept=training/
```

Hive Buckets



The table's data is
divided up into buckets

Skewed Tables

```
CREATE TABLE Customers (
    id int,
    username string,
    zip int
)
SKEWED BY (zip) ON (57701, 57702)
STORED as DIRECTORIES;
```

Demo: Understanding Partitions and Skew

Sorting Data

Hive has two sorting clauses:

- **order by**: a complete ordering of the data
- **sort by**: data output is sorted per reducer

Using Distribute By

```
insert overwrite table mytable
    select gender,age,salary
    from salaries
    distribute by age;
```

```
insert overwrite table mytable
      select gender,age,salary
    from salaries
    distribute by age
    sort by age;
```

Storing Results to a File

```
INSERT OVERWRITE DIRECTORY
    '/user/train/ca_or_sd/'
    from names
        select name, state
        where state = 'CA'
        or state = 'SD';
```

```
INSERT OVERWRITE LOCAL DIRECTORY
    '/tmp/myresults/'
    SELECT * FROM bucketnames
    ORDER BY age;
```

Specifying MapReduce Properties

```
SET mapreduce.job.reduces = 12
```

```
hive -f myscript.hive
  -hiveconf mapreduce.job.reduces=12
```

```
SELECT * FROM names
        WHERE age = ${age}
hive -f myscript.hive -hivevar age=33
```

Lab: Analyzing Big Data with Hive