Welcome to

Big Data & Hadoop

Session

Session 6 - HIVE Basics

+91-9538998962

sandeep@knowbigdata.com

WELCOME - KNOWBIGDATA

- Interact - Ask Questions
- Lifetime access of content
- Class Recording
- Cluster Access
- 24x7 support

- Real Life Project
- Quizzes & Certification Test
- 10 x (3hr class)
- Socio-Pro Visibility
- Mock Interviews

ABOUT ME

2014	**KnowBigData**	Founded
2014 2012	**Amazon**	Built High Throughput Systems for Amazon.com site using in-house NoSql.
2012	**InMobi**	Built Recommender after churning 200 TB
2011	**tBits Global**	Founded tBits Global Built an enterprise grade Document Management System
2006 2002	**D.E.Shaw**	Built the big data systems before the term was coined
2002	**IIT Roorkee**	Finished B.Tech somehow.

COURSE CONTENT

I	Understanding BigData, Hadoop Architecture
II	Environment Overview, MapReduce Basics
III	Adv MapReduce & Testing
IV	Pig & Pig Latin
➡ V	Analytics using Hive
VI	NoSQL, HBASE
VII	Oozie, Mahout,
VIII	Zookeeper, Apache Storm
IX	Apache Flume, Apache Spark
X	YARN, Big Data Sets & Project Assignment

TODAY'S CLASS

- Introduction
- Limitations
- Modules Chart
- Word Count Example
- Getting Started
- Hive Prompt
- Data Types
- Databases & Tables Basics
- Managed, External, Paritioned Tables
- Alter Table
- Load
- Simple Select Statements
- Aggregations
- Quick Demo

INTRODUCTION

Hadoop Is Great but How To Move Existing

- Data Relational & Sql Infrastructure?
- Larges base of SQL Users, DB Designers and Admins?

MapReduce

- Difficult even for developers
- Lot of Repetitive logic (such as join)

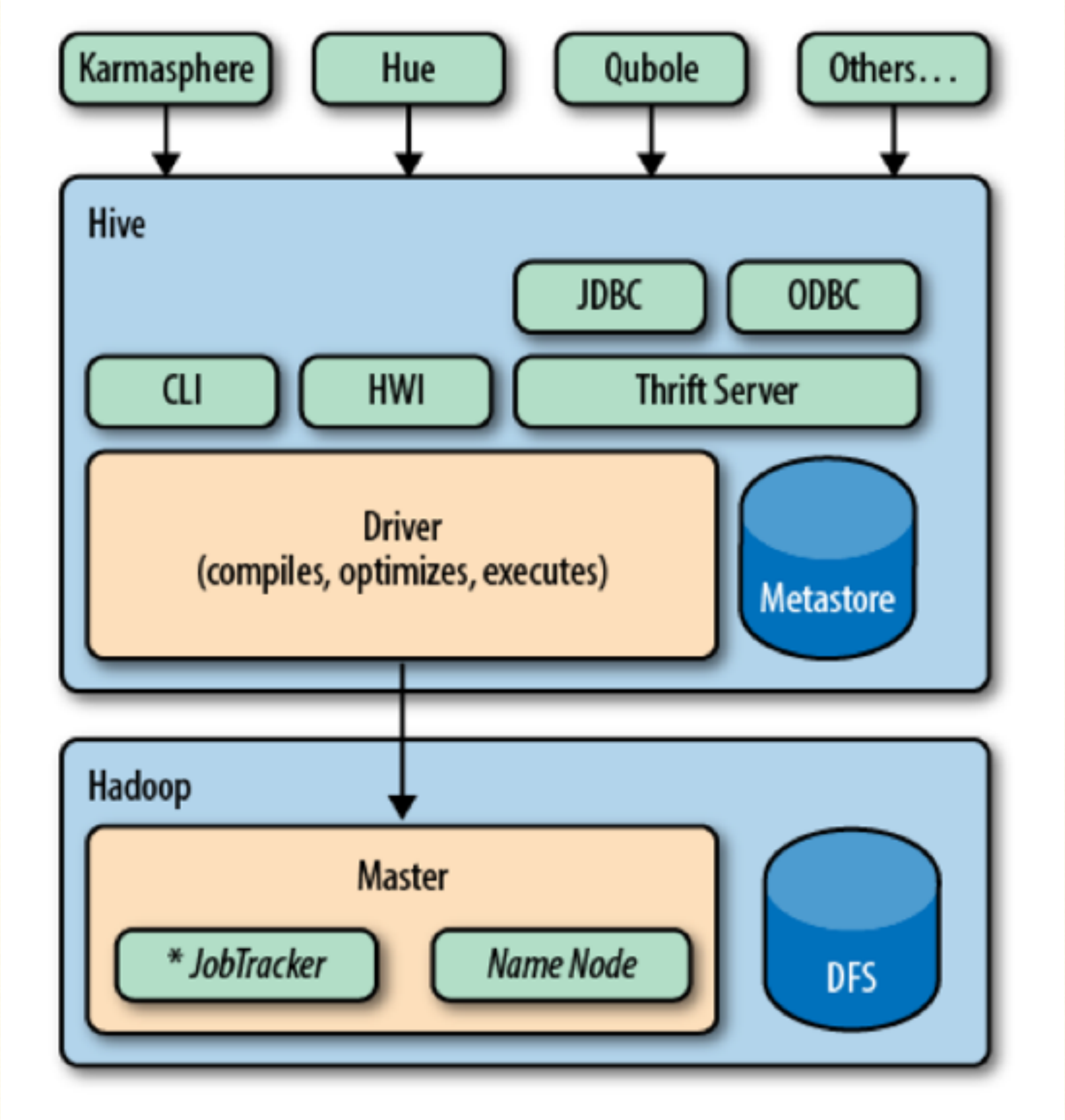
INTRODUCTION - VALUE PROP.

- Most suited for data warehouses
- Relatively Static Data is analysed
- Fast Response Not Required
- Makes it easier port existing projects
- Provides SQL Dialect called HIVEQL
- Maintains metastore(mysql) for the metadata
- Allows querying HDFS Files and HBASE

INTRODUCTION - LIMITATIONS

- Does not provide record-level
  - Update, insert or delete.
- But Can generate new tables
- Not suitable for OLTP
  - Queries have higher latency
  - Hadoop being batch-oriented,
  - Start-up overhead for MapReduce jobs
- Best when large Dataset is maintained and mined

MODULES


Karmasphere
Hue
Qubole
Others…
Hive
JDBC
ODBC
CLI
HWI
Thrift Server
Driver
(compiles, optimizes, executes)
Metastore
Hadoop
Master
* JobTracker
Name Node
DFS

THE WORD COUNT EXAMPLE

```
— Copy File
hadoop fs -cp 'sgiri/wordcount/input/big.txt' ‘sgiri/wordcount/input/big-copy.txt'
```

```
— Count Words Using the Copied File
LOAD DATA INPATH 'sgiri/wordcount/input/big-copy.txt' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
  SELECT word, count(1) AS count FROM
    (SELECT explode(split(line, '\s')) AS word FROM docs) w
  GROUP BY word ORDER BY word;
select * from word_counts;
```

GETTING STARTED…

```
ssh student@hadoop1.knowbigdata.com
```

or

```
ssh student@hadoop2.knowbigdata.com
```

```
hive
```

or

```
hive -e "SELECT * FROM mytable LIMIT 3”;
```

or

```
hive -f /path/to/file/withqueries.hql
```

1. CREATE TABLE x (a INT);
2. SELECT * FROM x;
3. DROP TABLE x;

ABOUT THE PROMPT - **HIVE>**

- Can run hadoop dfs commands
  - dfs -ls / ;
- Load a hive script
  - source /path/to/file/withqueries.hql;
- Comments start with --
- Set env variables using set
  - YEAR=2012
  - hive -e "SELECT * FROM x WHERE year = ${env:YEAR}”;
- Execute the shell commands with !
  - ! ls -la

DATA TYPES

Numeric Types

- TINYINT (1-byte)
- SMALLINT (2-byte)
- INT (4-byte signed)
- BIGINT (8-byte signed)
- FLOAT (4-byte single precision floating)
- DOUBLE (8-byte double precision floating)
- DECIMAL

Misc Types

- BOOLEAN
- BINARY (0.8.0)

String Types

- STRING
- VARCHAR (> 0.12.0)
- CHAR (> 0.13.0)

Date/Time Types

- TIMESTAMP (> 0.8.0)
- DATE (> 0.12.0)

DATA TYPES

Complex Types

```
  arrays: ARRAY<data_type>
  maps: MAP<primitive_type, data_type>
  structs: STRUCT<col_name : data_type [COMMENT col_comment], ...>
  union: UNIONTYPE<data_type, data_type, ...> (> Hive 0.7.0)
```

```
CREATE TABLE employees (
name STRING,
salary FLOAT,
subordinates ARRAY<STRING>,
deductions MAP<STRING, FLOAT>,
address STRUCT<street:STRING,
city:STRING,
state:STRING, zip:INT>
auth UNION<fbid:STRING, gid:STRING, email:STRING>
)
```

DATABASES

Essentially catalog or namespace of tables

```
hive> SHOW DATABASES;
  default
  financials
hive> CREATE DATABASE human_resources;
hive> SHOW DATABASES;
  default
  financials
  human_resources
hive>DESCRIBE DATABASE financials;
hive>USE financials;
(set hive.cli.print.current.db=true;)
hive>DROP DATABASE financials;
hive> DROP DATABASE IF EXISTS financials;
```

TABLES

Similar to SQL Tables

```
SHOW tables;
    Shows you the list of tables
select * from my_table
    runs select on my_table
describe my_table
describe extended my_table
describe formatted my_table
```

TABLES

Two Kinds of Tables

Managed Tables
aka Internal
lifecycle managed by Hive
data is stored in the file you added
Example: The previous all tables

External table
The lifecycle being managed by someone else
Hive does not assumes that it owns the data
dropping the table
  does not delete the data
  metdadata will be deleted

HIVE METASTORE

- Stores the metadata of tables into a db
- Meta data includes
  - What all dbs are there?
  - Which tables are there
  - table definitions: name of table, columns, partitions etc.

MANAGED TABLES

```
CREATE TABLE nysedaily (
exchange1 STRING,
symbol STRING,
ymd STRING,
price_open FLOAT,
price_high FLOAT,
price_low FLOAT,
price_close FLOAT,
volume INT,
price_adj_close FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LOCATION '/data/NYSE_daily';
```

LOADING DATA

From Local Directory (copies)

```
Load data local inpath '/home/student/sgiri/hive/NYSE_daily'
overwrite into table nysedaily;
```

From HDFS (moves)

```
hadoop fs -copyFromLocal /home/student/sgiri/hive/NYSE_daily sgiri/
#check if the file is in hdfs
hadoop fs -ls sgiri/NY*
Load data inpath 'sgiri/NYSE_daily' overwrite into table nysedaily;
#Check the file should not be in hdfs
hadoop fs -ls sgiri/NY*
```

Loads the data from file separated by ctrl-a
Will create the directory for partition, if any
Then copy the data
If target table is not partitioned you omit partition
Data is Stored on HDFS (/data/nysedaily in e.g.)

QUERY

```
select * from nysedaily where symbol = ‘CMC'
```

WHERE IS DATA LOCATED

Either in /apps/hive/warehouse
or
The location you mentioned in table definition.
In our case, it should be in ‘/data/NYSE_daily'
Check: hadoop fs -ls /data/NYSE_daily;

SAVING DATA

In Local File System

```
insert overwrite local directory  '/tmp/onlycmc'
select * from nysedaily where symbol = 'CMC'
```

IN HDFS

```
insert overwrite directory  '/tmp/onlycmc'
select * from nysedaily where symbol = 'CMC'
```

PARTITIONS

- To avoid the full table scan. Instead Partial table scan.
- The data is stored in different files based on various combinations of values of columns.
- You have to define the partitions using "partition by" in "create table"
- Partition can happen on multiple columns
- You can also add partition later.

PARTITIONS

Horizontal Partitioning using HDFS

```
$ cat sgiri/hive/TheEmp
sandeep,10,btech
sravani,9,ma
```

```
CREATE TABLE employees(Name String, Sal int, degree string)
PARTITIONED BY(Qualification STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

```
describe formatted employees;
hadoop fs -ls /apps/hive/warehouse/employees/
load data local inpath 'TheEmp' into table Employees partition (Qualification=1);
load data local inpath 'TheEmp1' into table Employees partition (Qualification=2);
SHOW PARTITIONS employees;
```

```
hadoop fs -ls /apps/hive/warehouse/employees/
  drwxr-xr-x   - student hdfs          0 2014-12-06 09:30 employee/qualification=1
  drwxr-xr-x   - student hdfs          0 2014-12-06 09:30 employee/qualification=2
```

TABLES - DDL - ALTER

Renaming a Table

```
ALTER TABLE log_messages RENAME TO logmsgs;
```

Changing Columns

```
ALTER TABLE log_messages
CHANGE COLUMN hms hours_minutes_seconds INT
COMMENT 'The hours, min’ AFTER severity;
```

Adding Columns

```
ALTER TABLE log_messages ADD COLUMNS (
    app_name STRING COMMENT 'App name',
    session_id LONG COMMENT 'The current id’
);
```

Drop a partition:

```
ALTER TABLE log_messages
    DROP IF EXISTS PARTITION(year = 2011, month = 12, day = 2);
```

BUCKETING

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
      page_url STRING, referrer_url STRING,
      ip STRING COMMENT 'IP Address of the User')
 COMMENT 'This is the page view table'
 PARTITIONED BY(dt STRING, country STRING)
 CLUSTERED BY(userid) SORTED BY(viewTime) INTO 32 BUCKETS
 ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\001'
   COLLECTION ITEMS TERMINATED BY '\002'
   MAP KEYS TERMINATED BY '\003'
 STORED AS SEQUENCEFILE;
```

TABLES - EXTERNAL

Reading a CSV

```
CREATE EXTERNAL TABLE IF NOT EXISTS mystocks (
  exchange1 STRING,
  symbol STRING,
  ymd STRING,
  price_open FLOAT,
  price_high FLOAT,
  price_low FLOAT,
  price_close FLOAT,
  volume INT,
  price_adj_close FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LOCATION ‘/data/NYSE_daily';

select * from mystocks;
```

S3 BASED EXTERNAL TABLE

```
create external table miniwikistats
(projcode string, pagename string, pageviews int, bytes int)
partitioned by(dt string)
row format delimited fields terminated by ' '
lines terminated by 'n'
location 's3n://paid-qubole/default-datasets/miniwikistats/';
```

TABLES - EXTERNAL PARTITIONED

```
CREATE EXTERNAL TABLE IF NOT EXISTS log_messages (
 hms INT,severity STRING,server STRING,
 process_id INT,
 message STRING
)
PARTITIONED BY (year INT, month INT, day INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

```
ALTER TABLE log_messages ADD PARTITION(year = 2012, month = 1, day = 2)
LOCATION "hdfs://hadoop1.knowbigdata.com/data/log_messages/2012/01/02";
```

```
ALTER TABLE log_messages PARTITION(year = 2011, month = 12, day = 2)
SET LOCATION 's3n://ourbucket/logs/2011/01/02';
```

```
describe   formatted log_messages partition (year = 2012, month = 1, day = 2);
```

NOTES

- Each table has got a location
- By default that table is a directory under the location /apps/hive/warehouse
- You can override that location by mentioning 'location' in create table clause.
- Load data copies the data if it is local
- Load moves the data if it is on hdfs for both external and managed table.
- Dropping managed table deletes the data the 'location'
- Dropping external table does not delete the data the 'location'
- The meta data is stored on the relational database

SELECT STATEMENTS

```
hive> SELECT name, salary FROM employees;
hive> SELECT e.name, e.salary FROM employees e;
```

Compution on Columns

```
    SELECT
        upper(name), salary,
        deductions["Federal Taxes"],
    round(salary * (1 - deductions["Federal Taxes”]))
    FROM
        employees;
```

AGGREGATIONS

Standard Aggregation functions

```
SELECT count(*), avg(salary) FROM employees;
```

To improve performance

```
SET hive.map.aggr=true;
--To do "top-level" aggregation in the map phase
```

VIEWS

```
CREATE VIEW tweets_clean AS
  SELECT
    id,  ts,  text,  m.country
   FROM tweets_simple t
  LEFT OUTER JOIN time_zone_map m
  ON t.time_zone = m.time_zone;
```

```
Select id, country from tweets_clean
```

CREATING CUSTOM FUNCTIONS

1. Create Custom Class

```
package com.example.hive.udf;

import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public final class Lower extends UDF {
  public Text evaluate(final Text s) {
    if (s == null) { return null; }
    return new Text(s.toString().toLowerCase());
  }
}
```

CREATING CUSTOM FUNCTIONS

2. Create JAR

3.Add jar your.jar

4. Define You function

create temporary function my_lower as 'com.example.hive.udf.Lower';

5. Call your function

select my_lower(title), sum(freq) from titles group by my_lower(title);

6. See a more detailed example

LOADING JSON DATA

- Download JSON-SERDE BINARIES
- add jar json-serde-1.1.6-SNAPSHOT-jar-with-dependencies.jar
- Create Table

```
CREATE EXTERNAL TABLE tweets_raw (
....
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION '/user/student/sgiri/senti/upload/data/tweets_raw'
;
```

SORTING & DISTRIBUTING

ORDER BY x

- Guarantees global ordering
- Does this by pushing all data through just one reducer.
- This is basically unacceptable for large datasets.
- You end up one sorted file as output.

SORTING & DISTRIBUTING

SORT BY x

- Orders data at each of N reducers
- Each reducer can receive overlapping ranges of data.
- You end up with N or more sorted files with overlapping ranges.

SORTING & DISTRIBUTING

DISTRIBUTE BY x

- Ensures each of N reducers gets non-overlapping ranges of x,
- But doesn't sort the output of each reducer.
- You end up with N or unsorted files with non-overlapping ranges.

SORTING & DISTRIBUTING

CLUSTER BY x

- Ensures each of N reducers gets non-overlapping ranges
- Then sorts by those ranges at the reducers.
- This gives you global ordering
- Is the same as (DISTRIBUTE BY x and SORT BY x).
- You end up with N or more sorted files with non-overlapping ranges.
- CLUSTER BY is basically the more scalable version of ORDER BY.

QUICK DEMO

1. Create a Table:

```
CREATE TABLE u_data_8nov (
  userid INT,
  movieid INT,
  rating INT,
  unixtime STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;
```

2.Download Data

```
hive>!wget http://files.grouplens.org/datasets/movielens/ml-100k.zip
```

3. Unzip

```
hive>! unzip ml-100k.zip
hive>! find .
```

4.LOAD DATA LOCAL INPATH 'ml-100k/u.data' overwrite into table u_data;

5. hive> select * from u_data limit 5;

QUICK DEMO

1. For each movie how many users voted it
2. For each movie what is the average ratings

Big Data & Hadoop

Thank you.

+91-9538998962

sandeep@knowbigdata.com