CSCE 5300 Introduction to Big Data and Data Science

Lesson 3- Hive

Professor: Dr. Zeenat Tariq

Agenda

- Motivation
- Overview
- Architecture
- Application
- Cons and Pros
- Data Model / Metadata
- Related Work

Data Analysts with Hadoop

Hadoop MR

- MR is very low level and requires customers to write custom programs
- HIVE supports queries expressed in SQL-like language called HiveQL which are compiled into MR jobs that are executed on Hadoop
- Hive also allows MR scripts
- It also includes MetaStore that contains schemas and statistics that are useful for data explorations, query optimization and query compilation

Motivation

- Limitation of MR
  - Not Reusable
  - Error prone
- For complex jobs:
    - Multiple stage of Map/Reduce functions
    - Just like ask dev to write specify physical execution plan in the database

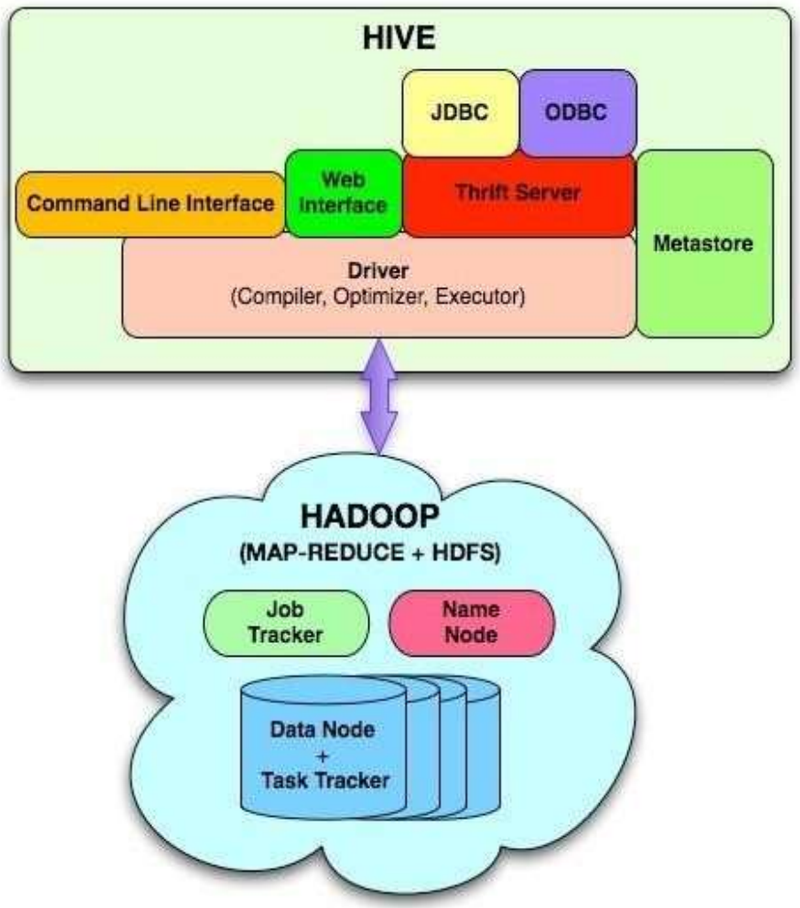
Motivation

- Yahoo worked on Pig to facilitate application deployment on Hadoop
- Their need mainly was focused on unstructured data
- Simultaneously Facebook started working on deploying warehouse solutions on Hadoop that resulted in Hive
- The size of data being collected and analyzed in industry for business intelligence (BI) is growing rapidly making traditional warehousing solution prohibitively expensive

What is HIVE ?

- A data warehousing system to store structured data on Hadoop file system
- Provide an easy query these data by execution Hadoop MapReduce plans
- Make the unstructured data looks like tables regardless how it really lay out
- SQL based query can be directly against these tables
- Generate specify execution plan for this query

Hive architecture



Metastore: stores system catalog

Driver: manages life cycle of HiveQL query as it moves thru' HIVE; also manages session handle and session statistics

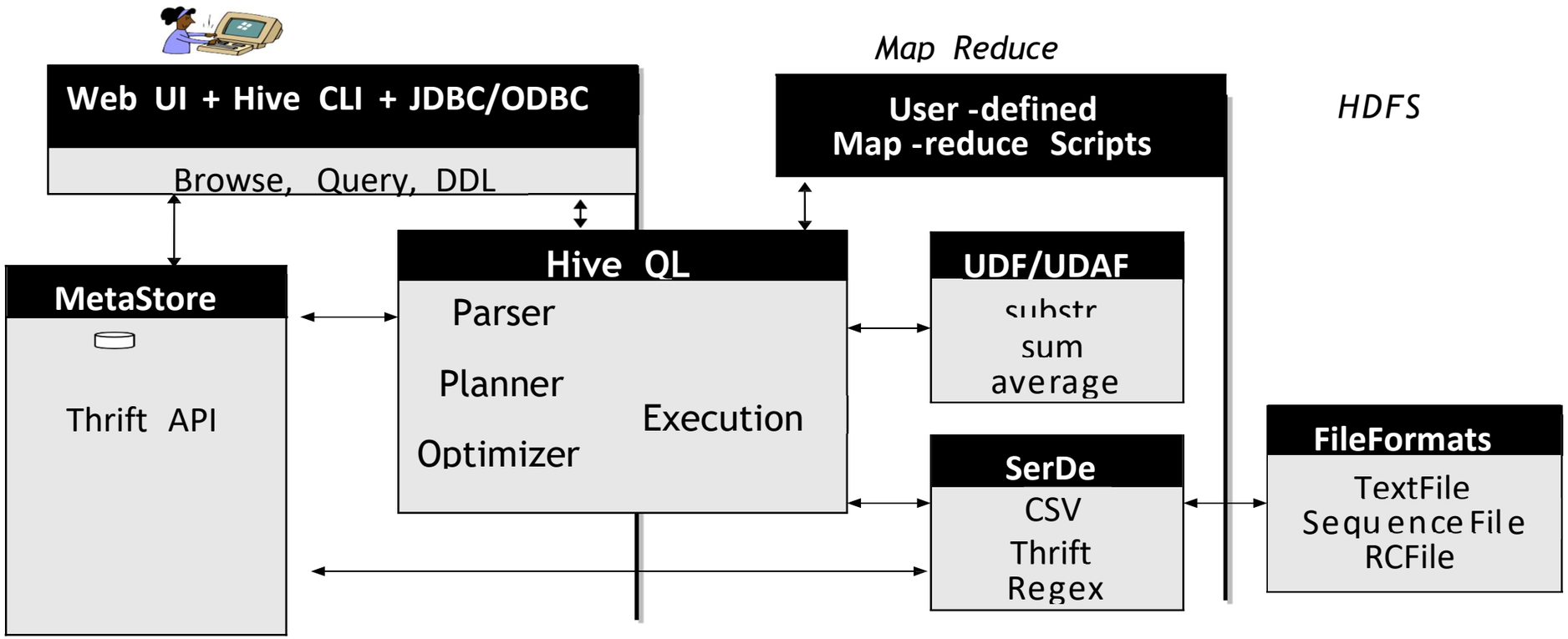
Query compiler: Compiles HiveQL into a directed acyclic graph of map/reduce tasks

Execution engines: The component executes the tasks in proper dependency order; interacts with Hadoop

HiveServer: provides Thrift interface and JDBC/ODBC for integrating other applications.

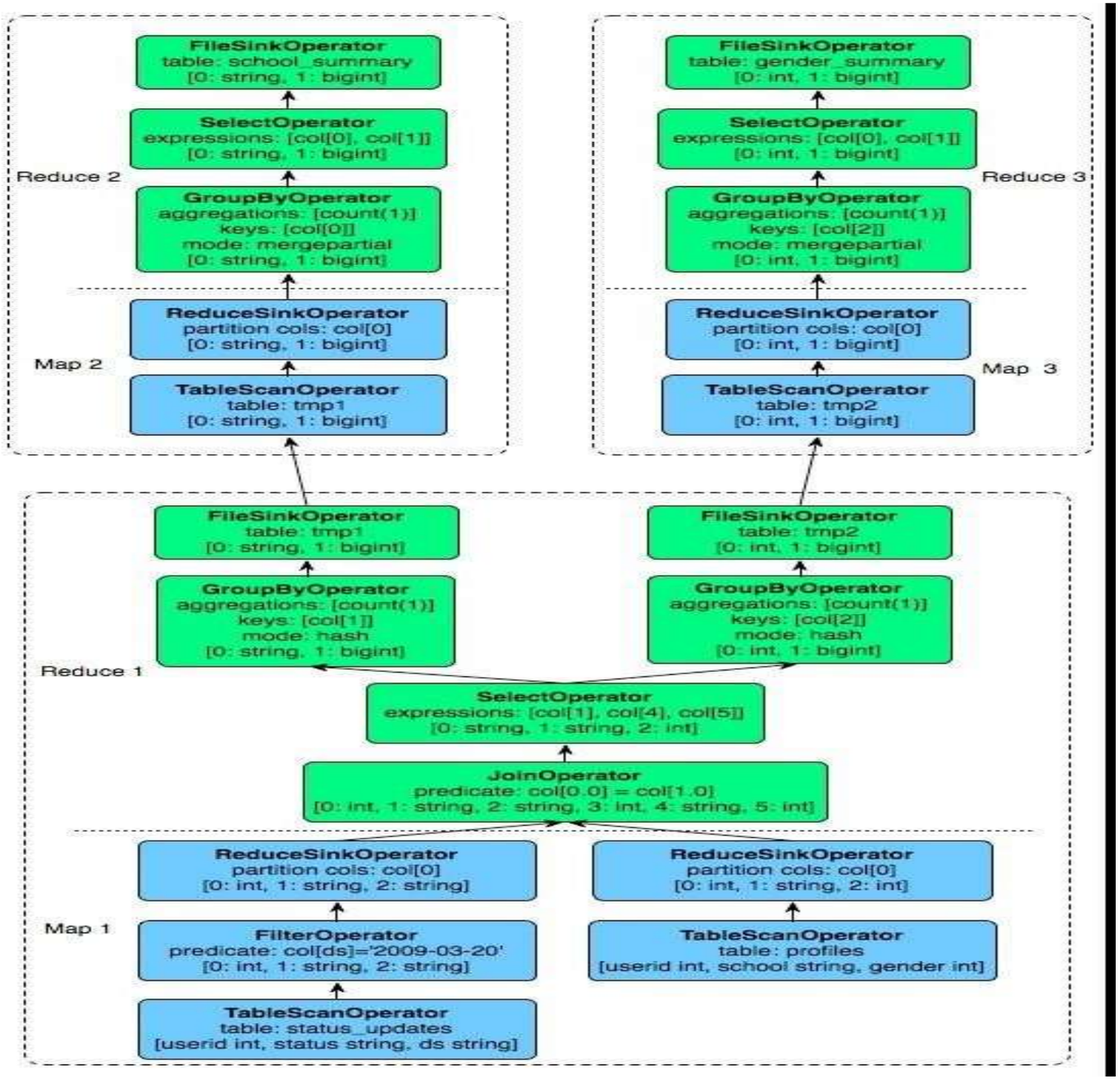
Client components: CLI, web interface, jdbc/odbc interface Extensibility interfaces include SerDe, User Defined Functions and User Defined Aggregate Function.

Architecture(Detailed)



http://www.slideshare.net/cloudera/hw09-hadoop-development-at-facebook-hive-and-hdfs

Sample Query Plan


FileSinkOperator
table: school_summary
[0: string, 1: bigint]
SelectOperator
expressions: [col[0], col[1]]
[0: string, 1: bigint]
Reduce 2
GroupByOperator
aggregations: [count(1)]
keys: [col[0]]
mode: mergepartial
[0: string, 1: bigint]
ReduceSinkOperator
partition cols: col[0]
[0: string, 1: bigint]
Map 2
TableScanOperator
table: tmp1
[0: string, 1: bigint]
FileSinkOperator
table: gender_summary
[0: int, 1: bigint]
SelectOperator
expressions: [col[0], col[1]]
[0: int, 1: bigint]
Reduce 3
GroupByOperator
aggregations: [count(1)]
keys: [col[2]]
mode: mergepartial
[0: int, 1: bigint]
ReduceSinkOperator
partition cols: col[0]
[0: int, 1: bigint]
Map 3
TableScanOperator
table: tmp2
[0: int, 1: bigint]
FileSinkOperator
table: tmp1
[0: string, 1: bigint]
GroupByOperator
aggregations: [count(1)]
keys: [col[1]]
mode: hash
[0: string, 1: bigint]
FileSinkOperator
table: tmp2
[0: int, 1: bigint]
GroupByOperator
aggregations: [count(1)]
keys: [col[2]]
mode: hash
[0: int, 1: bigint]
Reduce 1
SelectOperator
expressions: [col[1], col[4], col[5]]
[0: string, 1: string, 2: int]
JoinOperator
predicate: col[0.0] = col[1.0]
[0: int, 1: string, 2: string, 3: int, 4: string, 5: int]
ReduceSinkOperator
partition cols: col[0]
[0: int, 1: string, 2: string]
ReduceSinkOperator
partition cols: col[0]
[0: int, 1: string, 2: int]
Map 1
FilterOperator
predicate: col[ds]='2009-03-20'
[0: int, 1: string, 2: string]
TableScanOperator
table: profiles
[userid int, school string, gender int]
TableScanOperator
table: status_updates
[userid int, status string, ds string]

Application

- Log processing
  - Daily Report
  - User Activity Measurement
- Data/Text mining
  - Machine learning (Training Data)
- Business intelligence
  - Advertising Delivery
  - Spam Detection

Pros

- A easy way to process large scale data
- Support SQL-based queries
- Provide more user defined interfaces to extend
- Programmability
- Efficient execution plans for performance
- Interoperability with other database tools

Cons

- No easy way to append data
- Files in HDFS are immutable
- Future work
    - Views / Variables
    - More operator
        - In/Exists semantic

Hive Data Model

Data in Hive organized into :

- Tables
- Partitions
- Buckets

Hive Data Model Contd.

- Tables
  - Analogous to relational tables
  - Each table has a corresponding directory in HDFS
  - Data serialized and stored as files within that directory
  - Hive has default serialization built in which supports compression and lazy deserialization
  - Users can specify custom serialization –deserialization schemes (**SerDe's**)

Hive Data Model Contd.

- Partitions

  - Each table can be broken into partitions

  -Partitions determine distribution of data within subdirectories

- Example

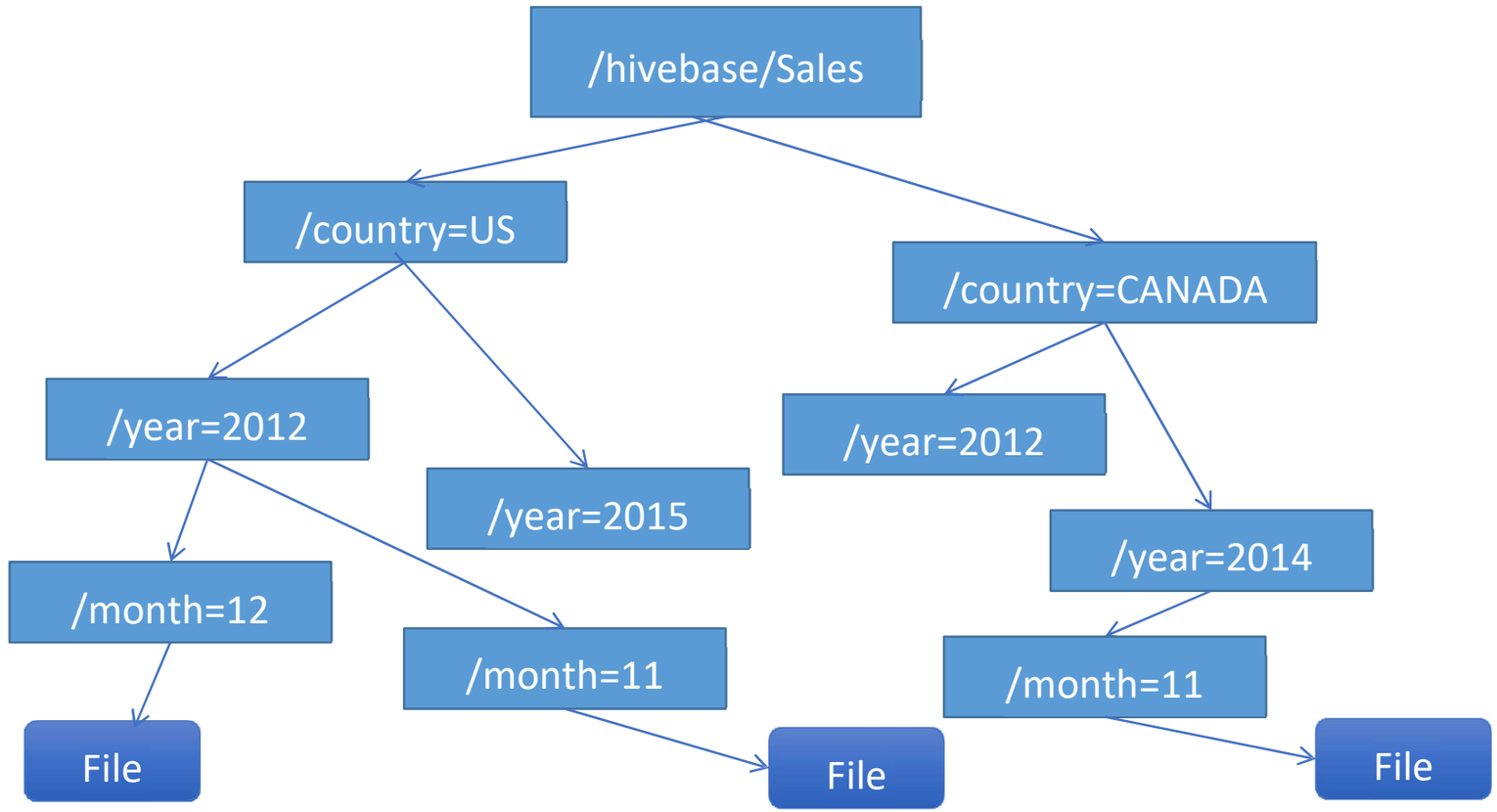
  CREATE_TABLE Sales (sale_id INT, amount FLOAT)

  PARTITIONED BY (country STRING, year INT, month INT)

  So each partition will be split out into different folders like

  Sales/country=US/year=2012/month=12

Hierarchy of Hive Partitions



Hive Data Model Contd.

- Buckets
  - Data in each partition divided into buckets
  - Based on a hash function of the column
  - **H(column) mod NumBuckets = bucket number**
  - Each bucket is stored as a file in partition directory

HiveQL

DDL :

- CREATE DATABASE
- CREATE TABLE
- ALTER TABLE
- SHOW TABLE
- DESCRIBE

DML:

- LOAD TABLE
- INSERT

QUERY:

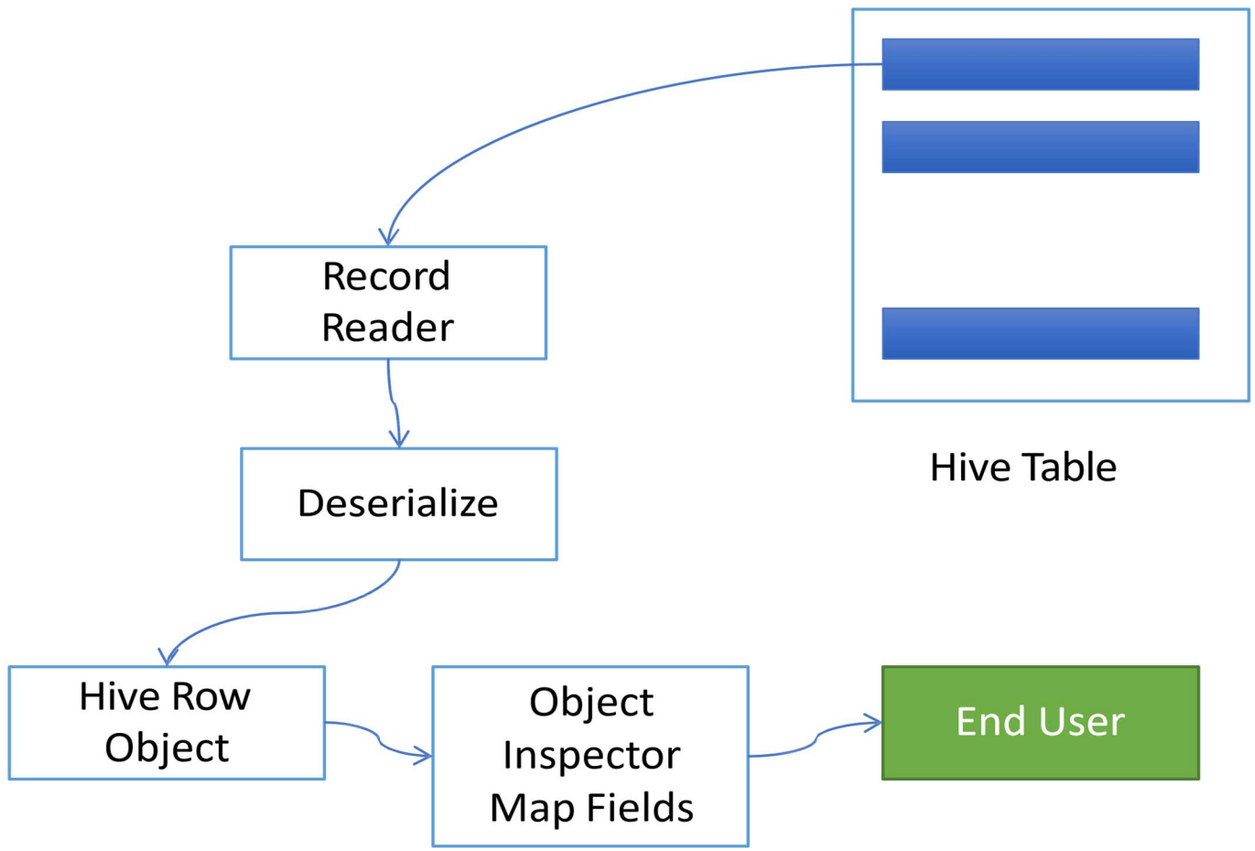
- SELECT
- GROUP BY JOIN
- MULTI TABLE INSERT

Hive SerDe

- SELECT Query

➢Hive built in Serde:
Avro, ORC, Regex etc

➢ Can use Custom SerDe's (e.g. for unstructured data like audio/video data, semistructured XML data)



Data model.. cont

- It supports primitive types: integers, floats, doubles, and strings
- Hive also supports complex types:
  - arrays: map<key-type, value-type>
  - Lists: list<element type>
  - Structs: struct<file name: file type…>
- SerDe: serialize and deserializedAPI is used to move data in and out of tables

Query Language (HiveQL)

- Subset of SQL
- Meta-data queries
- Limited equality and join predicates
- No inserts on existing tables (to preserve worm property)
- Can overwrite an entire table

Data Storage

- Tables are logical data units; table metadata associates the data in the table to hdfs directories.
- Hdfs namespace: tables (hdfs directory), partition (hdfs subdirectory), buckets (subdirectories within partition)
- /user/hive/warehouse/test_table is a hdfs directory

Hive Usage @ Facebook

- Statistics per day:
  - 4 TB of compressed new data added per day
  - 135TB of compressed data scanned per day
  - 7500+ Hive jobs on per day
- Hive simplifies Hadoop:
  - ~200 people/month run jobs on Hadoop/Hive
  - Analysts (non-engineers) use Hadoop through Hive
  - 95% of jobs are Hive Jobs

http://www.slideshare.net/cloudera/hw09-hadoop-development-at-facebook-hive-and-hdfs

Hive v/s Pig

Similarities:

- Both High level Languages which work on top of map reduce framework
- Can coexist since both use the under lying HDFS and map reduce

Differences:

- **Language**
  - Pig is a procedural ; (A = load ‘mydata’; dump A)
  - Hive is Declarative (select * from A)

- **Work Type**
  - Pig more suited for adhoc analysis (on demand analysis of click stream search logs)
  - Hive a reporting tool (e.g. weekly BI reporting).

Hive v/s Pig

Differences:

- **Users**
  - Pig – Researchers, Programmers (build complex data pipelines, machine learning)
  - Hive – Business Analysts
- **Integration**
  - Pig - Doesn’t have a thrift server(i.e no/limited cross language support)
    - Hive - Thrift server
- **User’s need**
  - Pig – Better dev environments, debuggers expected
  - Hive - Better integration with technologies expected(e.g JDBC, ODBC)

Practise

https://www.ukdataservice.ac.uk/media/604456/hiveworkshoppracti cal.pdf

In Class Exercise

Part1

Hive command use case: petrol

ColumnNO.	Name	Example	DataType
Column1:	District.ID	I4N 1M1	varchar
Column2: ,	Distributor name	shell	varchar
Column3:	Buy rate (million)	$957.70	varchar
Column4:	Sell rate(million)	$5779.92	varchar

Column5:	volumeIN(millioncu bic litter)	933	int
Column6:	volumeOUT(million cubic litter)	843,	int
Column7:	Year	1624	int

Hive Commands Use Case - Petrol

Creation of Table in Hive and Loading of data

```
hive> create table petrol (distributer_id STRING,distributer_name STRING,amt_IN STRING,amy_OUT STRING,vol_IN INT,vol
_OUT INT,year INT) row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.096 seconds
hive> load data local inpath '/home/acadgild/Downloads/petrol.txt' into table petrol;
Loading data to table default.petrol
Table default.petrol stats: [numFiles=1, totalSize=19081]
OK
Time taken: 0.345 seconds
hive>
```

Create table petrol

(distributer_idSTRING,distributer_nameSTRING,amt_INSTRING,amy_OUTSTRING,vol_

IN INT,vol_OUT INT,year INT) row format delimited fields terminated by ‘,’ stored as

textfile;

load data local inpath ‘/home/acadgild/Downloads/petrol.txt’ into table petrol;

Hive Commands Use Case - Petrol

1) In real life what is the total amount of petrol in volume sold by every distributor?
SELECT distributer_name,SUM(vol_OUT) FROM petrol GROUP BY distributer_name;

```
hive> SELECT distributer_name,SUM(vol_OUT) FROM petrol GROUP BY distributer_name;
Query ID = acadgild_20161202170606_1b5ecb9f-6533-4450-bf8b-661696b52b3f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1480676763240_0001, Tracking URL = http://localhost:8088/proxy/application_1480676763240_0001/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1480676763240_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-12-02 17:06:38,626 Stage-1 map = 0%,  reduce = 0%
2016-12-02 17:07:18,093 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.55 sec
2016-12-02 17:07:54,312 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 3.27 sec
2016-12-02 17:08:01,530 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.7 sec
MapReduce Total cumulative CPU time: 5 seconds 700 msec
Ended Job = job_1480676763240_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.7 sec   HDFS Read: 19294 HDFS Write: 56 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 700 msec
OK
Bharat  83662
hindustan       71767
reliance        76558
shell   69266
Time taken: 125.357 seconds, Fetched: 4 row(s)
hive>
```

Hive Commands Use Case - Petrol

2)Which are the top 10 distributors ID's for selling petrol and also display the amount of petrol sold in volume by them individually?

SELECT distributer_id,vol_OUT FROM petrol order by vol_OUT desc limit 10;

```
hive> SELECT distributer_id,vol_OUT FROM petrol order by vol_OUT desc limit 10;
Query ID = acadgild_20161202172020_230cd2e8-52c6-41cb-bbbf-3614a5d84800
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1480676763240_0002, Tracking URL = http://localhost:8088/proxy/application_1480676763240_0002/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1480676763240_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-12-02 17:21:00,832 Stage-1 map = 0%,  reduce = 0%
2016-12-02 17:21:19,226 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2016-12-02 17:21:42,624 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.91 sec
MapReduce Total cumulative CPU time: 3 seconds 910 msec
Ended Job = job_1480676763240_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.91 sec   HDFS Read: 19294 HDFS Write: 120 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 910 msec
OK
S8W 0P4 899
T1A 9W4 899
V8U 2T6 898
O8A 6Z5 897
O9P 9S3 897
F6W 6H3 896
E6O 9P1 895
N5Q 8E5 895
M6S 1P4 895
J4M 4G3 895
Time taken: 63.329 seconds, Fetched: 10 row(s)
hive>
```

Hive Commands Use Case - Petrol

3)Find real life 10 distributor name who sold petrol in the least amount

SELECT distributer_id,vol_OUT FROM petrol order by vol_OUT limit 10;

```
hive> SELECT distributer_id,vol_OUT FROM petrol order by vol_OUT limit 10;
Query ID = acadgild_20161202172828_86b85e7d-f0d0-4b89-beec-23adca69eb8b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1480676763240_0003, Tracking URL = http://localhost:8088/proxy/application_1480676763240_0003/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1480676763240_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-12-02 17:28:48,236 Stage-1 map = 0%,  reduce = 0%
2016-12-02 17:28:57,910 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.47 sec
2016-12-02 17:29:22,617 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.28 sec
MapReduce Total cumulative CPU time: 3 seconds 280 msec
Ended Job = job_1480676763240_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.28 sec   HDFS Read: 19294 HDFS Write: 120 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 280 msec
OK
F4D 6K2 602
H7M 4M4 603
G9F 6U7 607
R3W 2E3 608
H4P 6A9 610
O5D 2R6 610
W0M 8R7 612
V0Z 0F6 612
O0D 0L1 612
L9H 1K6 613
Time taken: 45.751 seconds, Fetched: 10 row(s)
hive>
```

Hive Commands Use Case - Petrol

4)Try One yourself

The constraint to this query is the difference between volumeIN and volumeOuT is illegal in real life if greater than 500. As we see all distributors are receiving patrols on every next cycle.

List all distributors who have this difference, along with the year and the difference which they have in that year. Hint: (vol_IN-vol_OUT)>500

In Class Exercise

Part2

Hive Commands Use Case - OLYMPICS

Hive Command Use case- Olympics

ColumnNO.	Name	Example	DataType
Column1:	AthleteNam e	Michael Phelps	STRING
Column2: ,	Age	23	INT
Column3:	Country	United States	STRING
Column4:	Year	2008	INT
Column5:	Closing Date	8/24/2008	STRING
Column6:	Sport	Swimming	STRING
Column7:	Gold Medals	8	INT
Column8:	Silver Medals	0	INT
Column9:	Bronze Medals	0	INT
Column10:	Total Medals	8	INT

Hive Commands Use Case - OLYMPICS

Creation of Table in Hive and Loading of data

create table olympic (athelete STRING,age INT,country STRING,year STRING,closing STRING,sport STRING,gold INT,silver INT,bronze INT,total INT) row format delimited fields terminated by '\t' stored as textfile;

```
hive> create table olympic(athelete STRING,age INT,country STRING,year STRING,closing STRING,sport STRING,gold INT,silver INT,bronze INT,total INT) row form
at delimited fields terminated by '\t' stored as textfile;
OK
Time taken: 2.762 seconds
hive> load data local inpath '/home/acadgild/Downloads/olympic_data.csv' into table olympic;
Loading data to table default.olympic
Table default.olympic stats: [numFiles=1, totalSize=518669]
OK
Time taken: 3.173 seconds
hive>
```

load data local inpath '/home/acadgild/Downloads/olympic_data.csv' into table olympic;

Hive Commands Use Case - OLYMPICS

1) Find total number of medals won by each country in athletics.
   - we use SUM function to define total medal and also, we need only Athletics so use where condition.
   - we need separate country and their relevant total number of medal then we group by country.

   SELECT Country,SUM(Total) FROM olympic WHERE Sport = "Athletics" GROUP BY Country;

2) Find the total number of medals each country won display the name along with total medals.

   select country,SUM(total) from olympic GROUP BY country;

3) Find Total number of medals won by each country in swimming.

   Select country,sum(total) from Olympic where sport='swimming' group by country;

Try it yourself.

4) Which country got medals for Shooting, year wise classification?

• In Class Exercise

Part 3

Hive Commands Use Case - MovieLens

Create Hive Tables and Perform Queries for Use Case based on Movielens dataset which has 3 datasets as movies, users and ratings. Perform following tasks:

- Create a table for 3 for movies, user, Rating.
- Now join the two tables. (Movies and rating)
- Find which day of the week most of ratings are posted.

• In Class Exercise

Part 4

Task

Perform the following tasks:

Using the same data set as Part 3. Perform the following tasks:

- Create exploded view of movie id and genre.
- Find for each user, the rank of genre. Ideally you would like to compute weighted.
- Create a temporary view for user and his total ratings by genre.
- Find top 3 genres for each user and create a temporary table for that.

Reference

- A.Thusoo et al. Hive: a warehousing solution over a mapreduce framework. Proceedings of VLDB09', 2009.
  https://mapr.com/products/apache-hive/
  https://cwiki.apache.org/confluence/display/Hive#HomeApacheHive
- Hadoop 2009:
  http://www.slideshare.net/cloudera/hw09-hadoop-development-at-facebook-hiveand-hdfs
- Facebook Data Team:
  http://www.slideshare.net/zshao/hive-data-warehousing-analytics-on-hadoop-presentation
- Hive Examples:

https://umkc.box.com/s/1dcugk08caqzitgqvrthiqe5n6sgznd5