Hive - A Warehousing Solution Over a Map-Reduce Framework
Agenda

• Why Hive??
• What is Hive?
• Hive Data Model
• Hive Architecture
• HiveQL
• Hive SerDe’s
• Pros and Cons
• Hive v/s Pig
• Graphs
Data Analysts with Hadoop
Challenges that Data Analysts faced

- **Data Explosion**
  - TBs of data generated everyday

Solution – HDFS to store data and Hadoop Map-Reduce framework to parallelize processing of Data

What is the catch?

- Hadoop Map Reduce is Java intensive
- Thinking in Map Reduce paradigm can get tricky
... Enter Hive!
Hive Key Principles

- Defines SQL-Like query language called QL
- Allows programmers to plug-in custom mappers and reducers
- Data Warehouse Infrastructure
- It provides tools to enable easy data ETL
HiveQL to MapReduce

Hive Framework

SELECT COUNT(1) FROM Sales;

rowcount, N

Sales: Hive table

MR JOB Instance
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)
• 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

```
/hivebase/Sales
  /country=US
    /year=2012
      /month=12
      File
    /year=2015
    /year=2012
      /month=11
      File
  /country=CANADA
    /year=2012
      /month=11
      File
    /year=2014
      /month=11
      File
```
Hive Data Model Contd.

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

**External Interfaces** - CLI, WebUI, JDBC, ODBC programming interfaces

**Thrift Server** – Cross Language service framework.

**Metastore** - Meta data about the Hive tables, partitions

**Driver** - Brain of Hive! Compiler, Optimizer and Execution engine
Hive Thrift Server

- Framework for cross language services
- Server written in Java
- Support for clients written in different languages
  - JDBC(java), ODBC(c++), php, perl, python scripts
Metastore

- System catalog which contains metadata about the Hive tables
- Stored in RDBMS/local fs. HDFS too slow (not optimized for random access)
- Objects of Metastore
  - Database - Namespace of tables
  - Table - list of columns, types, owner, storage, SerDes
  - Partition – Partition specific column, Serdes and storage
Hive Driver

- **Driver** - Maintains the lifecycle of HiveQL statement
- **Query Compiler** – Compiles HiveQL in a DAG of map reduce tasks
- **Executor** - Executes the tasks plan generated by the compiler in proper dependency order. Interacts with the underlying Hadoop instance
Compiler

- Converts the HiveQL into a plan for execution
- Plans can
  - Metadata operations for DDL statements e.g. CREATE
  - HDFS operations e.g. LOAD
- Semantic Analyzer – checks schema information, type checking, implicit type conversion, column verification
- Optimizer – Finding the best logical plan e.g. Combines multiple joins in a way to reduce the number of map reduce jobs, Prune columns early to minimize data transfer
- Physical plan generator – creates the DAG of map-reduce jobs
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)
Good Things

• Boon for Data Analysts
• Easy Learning curve
• Completely transparent to underlying Map-Reduce
• Partitions(speed!)
• Flexibility to load data from localFS/HDFS into Hive Tables
Cons and Possible Improvements

- Extending the SQL queries support (Updates, Deletes)
- Parallelize firing independent jobs from the work DAG
- Table Statistics in Metastore
- Explore methods for multi query optimization
- Perform N-way generic joins in a single map reduce job
- Better debug support in shell
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)
Head-to-Head
(the bee, the pig, the elephant)

Hive, PIG and Hadoop benchmark

- Grep select: Hadoop 136.1, Pig 125.4, Hive 247.8
- Rankings select: Hadoop 26.1, Pig 31, Hive 38.4
- Uservisits aggregation: Hadoop 533.8, Pig 768.8, Hive 855.4
- Rankings uservisits join: Hadoop 470, Pig 471.3, Hive 763.9

Version: Hadoop – 0.18x, Pig:786346, Hive:786346
REFERENCES

- https://hive.apache.org/
- https://cwiki.apache.org/confluence/display/Hive/Presentations
- Hortonworks tutorials (youtube)