Outline

• Motivation
• Overview
• Data Model
• Working with Hive
• Wrap up & Conclusions
Background

- Started at Facebook
- Data was collected by nightly cron jobs into Oracle DB
- “ETL” via hand-coded python
- Grew from 10s of GBs (2006) to 1 TB/day new data (2007), now 10x that.
Hadoop as Enterprise Data Warehouse

• Scribe and MySQL data loaded into Hadoop HDFS
• Hadoop MapReduce jobs to process data
• Missing components:
  – Command-line interface for “end users”
  – Ad-hoc query support
    • … without writing full MapReduce jobs
  – Schema information
Hive Applications

- Log processing
- Text mining
- Document indexing
- Customer-facing business intelligence (e.g., Google Analytics)
- Predictive modeling, hypothesis testing
Hive Components

- Shell: allows interactive queries like MySQL shell connected to database
  - Also supports web and JDBC clients
- Driver: session handles, fetch, execute
- Compiler: parse, plan, optimize
- Execution engine: DAG of stages (M/R, HDFS, or metadata)
- Metastore: schema, location in HDFS, SerDe
Data Model

• Tables
  – Typed columns (int, float, string, date, boolean)
  – Also, list: map (for JSON-like data)

• Partitions
  – e.g., to range-partition tables by date

• Buckets
  – Hash partitions within ranges (useful for sampling, join optimization)
Metastore

• Database: namespace containing a set of tables
• Holds table definitions (column types, physical layout)
• Partition data
• Uses JPOX ORM for implementation; can be stored in Derby, MySQL, many other relational databases
Physical Layout

- **Warehouse directory in HDFS**
  - e.g., `/home/hive/warehouse`
- **Tables stored in subdirectories of warehouse**
  - Partitions, buckets form subdirectories of tables
- **Actual data stored in flat files**
  - Control char-delimited text, or SequenceFiles
  - With custom SerDe, can use arbitrary format
Starting the Hive shell

• Start a terminal and run
  $ cd /usr/share/cloudera/hive/
  $ 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;
Generating Data

• Let’s get (word, frequency) data from the Shakespeare data set:

```bash
$ hadoop jar \\
$HADOOP_HOME/hadoop-*-examples.jar \\
grep input shakespeare_freq \w+
```
Loading data

- Remove the MapReduce job logs:

  $ hadoop fs -rmlr shakespeare_freq/_logs

- Load dataset into Hive:

  hive> LOAD DATA INPATH "shakespeare_freq"
      INTO TABLE shakespeare;
Selecting data

hive> SELECT * FROM shakespeare LIMIT 10;

hive> SELECT * FROM shakespeare
    WHERE freq > 100 SORT BY freq ASC
    LIMIT 10;
Most common frequency

```
hive> SELECT freq, COUNT(1) AS f2
       FROM shakespeare GROUP BY freq
       SORT BY f2 DESC LIMIT 10;
```

```
hive> EXPLAIN SELECT freq, COUNT(1) AS f2
       FROM shakespeare GROUP BY freq
       SORT BY f2 DESC LIMIT 10;
```
Joining tables

• A powerful feature of Hive is the ability to create queries that join tables together

• We have (freq, word) data for Shakespeare

• Can also calculate it for KJV

• Let’s see what words show up a lot in both
Create the dataset:

```
$ tar zxf ~/bible.tar.gz -C ~
$ hadoop fs -put ~/bible bible
$ hadoop jar \n    $HADOOP_HOME/hadoop-*-examples.jar \n    grep bible bible_freq '\w+'
```
Create the new table

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

hive> SHOW TABLES;

hive> DESCRIBE kjv;
```
Import data to Hive

$ hadoop fs -rmr bible_freq/_logs

hive> LOAD DATA INPATH "bible_freq"
   INTO TABLE kjv;
Create an intermediate table

```
hive> CREATE TABLE merged
    (word STRING, shake_f INT, kjv_f INT);
```
Running the join

```
hive> INSERT OVERWRITE TABLE merged  
      SELECT s.word, s.freq, k.freq FROM  
     shakespeare s JOIN kjv k ON  
       (s.word = k.word)  
    WHERE s.freq >= 1 AND k.freq >= 1;

hive> SELECT * FROM merged LIMIT 20;
```
Most common intersections

• What words appeared most frequently in both corpuses?

```
hive> SELECT word, shake_f, kjv_f,  
    (shake_f + kjv_f) AS ss  
FROM merged SORT BY ss DESC  
LIMIT 20;
```
Some more advanced features…

- "TRANSFORM:" Can use MapReduce in SQL statements
- Custom SerDe: Can use arbitrary file formats
- Metastore check tool
- Structured query log
Project Status

- Open source, Apache 2.0 license
- Official subproject of Apache Hadoop
- 4 committers (all from Facebook)
- First release candidate coming soon
Related work

- Parallel databases: Gamma, Bubba, Volcano
- Google: Sawzall
- Yahoo: Pig
- IBM Research: JAQL
- Microsoft: DryadLINQ, SCOPE
- Greenplum: YAML MapReduce
- Aster Data: In-database MapReduce
- Business.com: CloudBase
Conclusions

• Supports rapid iteration of ad-hoc queries
• Can perform complex joins with minimal code
• Scales to handle much more data than many similar systems