Apache Hive

CMSC 491
Hadoop-Based Distributed Computing
Spring 2016
Adam Shook

What Is Hive?

- Developed by Facebook and a top-level Apache project
- A data warehousing infrastructure based on Hadoop
- Immediately makes data on a cluster available to non-Java programmers via SQL like queries
- Built on HiveQL (HQL), a SQL-like query language
- Interprets HiveQL and generates MapReduce jobs that run on the cluster
- Enables easy data summarization, ad-hoc reporting and querying, and analysis of large volumes of data

What Hive Is Not

- Hive, like Hadoop, is designed for batch processing of large datasets
- Not an OLTP or real-time system
- Latency and throughput are both high compared to a traditional RDBMS
  - Even when dealing with relatively small data (<100 MB)

Data Hierarchy

- Hive is organised hierarchically into:
  - Databases: namespaces that separate tables and other objects
  - Tables: homogeneous units of data with the same schema
    - Analogous to tables in an RDBMS
  - Partitions: determine how the data is stored
    - Allow efficient access to subsets of the data
  - Buckets/clusters
    - For subsampling within a partition
    - Join optimization

HiveQL

- HiveQL / HQL provides the basic SQL-like operations:
  - Select columns using SELECT
  - Filter rows using WHERE
  - JOIN between tables
  - Evaluate aggregates using GROUP BY
  - Store query results into another table
  - Download results to a local directory (i.e., export from HDFS)
  - Manage tables and queries with CREATE, DROP, and ALTER

Primitive Data Types

Type	Comments
TINYINT, SMALLINT, INT, BIGINT	1, 2, 4 and 8-byte integers
BOOLEAN	TRUE/FALSE
FLOAT, DOUBLE	Single and double precision real numbers
STRING	Character string
TIMESTAMP	Unix-epoch offset *or* datetime string
DECIMAL	Arbitrary-precision decimal
BINARY	Opaque; ignore these bytes

Complex Data Types

Type	Comments
STRUCT	A collection of elements If S is of type STRUCT {a INT, b INT}: S.a returns element a
MAP	Key-value tuple If M is a map from 'group' to GID: M['group'] returns value of GID
ARRAY	Indexed list If A is an array of elements ['a','b','c']: A[0] returns 'a'

HiveQL Limitations

- HQL only supports equi-joins, outer joins, left semi-joins
- Because it is only a shell for mapreduce, complex queries can be hard to optimise
- Missing large parts of full SQL specification:
  - HAVING clause in SELECT
  - Correlated sub-queries
  - Sub-queries outside FROM clauses
  - Updatable or materialized views
  - Stored procedures

Hive Metastore

- Stores Hive metadata
- Default metastore database uses Apache Derby
- Various configurations:
  - Embedded (in-process metastore, in-process database)
    - Mainly for unit tests
  - Local (in-process metastore, out-of-process database)
    - Each Hive client connects to the metastore directly
  - Remote (out-of-process metastore, out-of-process database)
    - Each Hive client connects to a metastore server, which connects to the metadata database itself

Hive Warehouse

- Hive tables are stored in the Hive "warehouse"
  - Default HDFS location: /user/hive/warehouse
- Tables are stored as sub-directories in the warehouse directory
- Partitions are subdirectories of tables
- External tables are supported in Hive
- The actual data is stored in flat files

Hive Schemas

- Hive is schema-on-read
  - Schema is only enforced when the data is read (at query time)
  - Allows greater flexibility: same data can be read using multiple schemas
- Contrast with an RDBMS, which is schema-on-write
  - Schema is enforced when the data is loaded
  - Speeds up queries at the expense of load times

Create Table Syntax

```
CREATE TABLE table_name
    (col1 data_type,
     col2 data_type,
     col3 data_type,
     col4 datatype )
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ','
   STORED AS format_type;
```

Simple Table

```
CREATE TABLE page_view
    (viewTime INT,
     userid BIGINT,
     page_url STRING,
     referrer_url STRING,
     ip STRING COMMENT 'IP Address of the User' )
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\t'
   STORED AS TEXTFILE;
```

More Complex Table

```
CREATE TABLE employees  (
    (name STRING,
    salary FLOAT,
    subordinates ARRAY<STRING>,
    deductions MAP<STRING, FLOAT>,
    address STRUCT<street:STRING,
                   city:STRING,
                   state:STRING,
                   zip:INT>)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t'
  STORED AS TEXTFILE;
```

External Table

```
CREATE EXTERNAL TABLE page_view_stg
    (viewTime INT,
     userid BIGINT,
     page_url STRING,
     referrer_url STRING,
     ip STRING COMMENT 'IP Address of the User')
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\t'
   STORED AS TEXTFILE
   LOCATION '/user/staging/page_view';
```

More About Tables

- CREATE TABLE
  - LOAD: file moved into Hive's data warehouse directory
  - DROP: both metadata and data deleted
- CREATE EXTERNAL TABLE
  - LOAD: no files moved
  - DROP: only metadata deleted
  - Use this when sharing with other Hadoop applications, or when you want to use multiple schemas on the same data

Partitioning

- Can make some queries faster
- Divide data based on partition column
- Use PARTITION BY clause when creating table
- Use PARTITION clause when loading data
- SHOW PARTITIONS will show a table's partitions

Bucketing

- Can speed up queries that involve sampling the data
  - Sampling works without bucketing, but Hive has to scan the entire dataset
- Use CLUSTERED BY when creating table
  - For sorted buckets, add SORTED BY
- To query a sample of your data, use TABLESAMPLE

Browsing Tables And Partitions

Command	Comments
`SHOW TABLES;`	Show all the tables in the database
`SHOW TABLES 'page.*';`	Show tables matching the specification (uses regex syntax)
`SHOW PARTITIONS page_view;`	Show the partitions of the page_view table
`DESCRIBE page_view;`	List columns of the table
`DESCRIBE EXTENDED page_view;`	More information on columns (useful only for debugging)
`DESCRIBE page_view PARTITION (ds='2008-10-31');`	List information about a partition

Loading Data

- Use LOAD DATA to load data from a file or directory
  - Will read from HDFS unless LOCAL keyword is specified
  - Will append data unless OVERWRITE specified
  - PARTITION required if destination table is partitioned

```
LOAD DATA LOCAL INPATH '/tmp/pv_2008-06-8_us.txt'
  OVERWRITE INTO TABLE page_view
  PARTITION (date='2008-06-08', country='US')
```

Inserting Data

- Use INSERT to load data from a Hive query
  - Will append data unless OVERWRITE specified
  - PARTITION required if destination table is partitioned

```
FROM page_view_stg pvs
  INSERT OVERWRITE TABLE page_view
  PARTITION (dt='2008-06-08', country='US')
  SELECT pvs.viewTime, pvs.userid,
         pvs.page_url, pvs.referrer_url
  WHERE pvs.country = 'US';
```

Inserting Data

- Normally only one partition can be inserted into with a single INSERT
- A multi-insert lets you insert into multiple partitions

```
FROM page_view_stg pvs
INSERT OVERWRITE TABLE page_view
PARTITION  ( dt='2008-06-08', country='US‘ )
SELECT pvs.viewTime, pvs.userid, pvs.page_url, pvs.referrer_url WHERE pvs.country = 'US'
INSERT OVERWRITE TABLE page_view
PARTITION ( dt='2008-06-08', country='CA' )
SELECT pvs.viewTime, pvs.userid, pvs.page_url, pvs.referrer_url WHERE pvs.country = 'CA'
INSERT OVERWRITE TABLE page_view
PARTITION ( dt='2008-06-08', country='UK' )
SELECT pvs.viewTime, pvs.userid, pvs.page_url, pvs.referrer_url WHERE pvs.country = 'UK';
```

Inserting Data During Table Creation

- Use AS SELECT in the CREATE TABLE statement to populate a table as it is created

```
CREATE TABLE page_view AS
  SELECT pvs.viewTime, pvs.userid, pvs.page_url,
         pvs.referrer_url
  FROM page_view_stg pvs
  WHERE pvs.country = 'US';
```

Loading And Inserting Data: Summary

Use this	For this purpose
`LOAD`	Load data from a file or directory
`INSERT`	Load data from a query • One partition at a time • Use multiple INSERTs to insert into multiple partitions in the one query
`CREATE TABLE AS (CTAS)`	Insert data while creating a table
Add/modify external file	Load new data into external table

Sample Select Clauses

- Select from a single table

```
SELECT *
   FROM sales
   WHERE amount > 10 AND
            region = "US";
```

- Select from a partitioned table

```
SELECT page_views.*
 FROM page_views
 WHERE page_views.date >= '2008-03-01' AND
      page_views.date <= '2008-03-31'
```

Relational Operators

- ALL and DISTINCT
  - Specify whether duplicate rows should be returned
  - ALL is the default (all matching rows are returned)
  - DISTINCT removes duplicate rows from the result set
- WHERE
  - Filters by expression
  - Does not support IN, EXISTS or sub-queries in the WHERE clause
- LIMIT
  - Indicates the number of rows to be returned

Relational Operators

- GROUP BY
  - Group data by column values
  - Select statement can only include columns included in the GROUP BY clause
- ORDER BY / SORT BY
  - ORDER BY performs total ordering
    - Slow, poor performance
  - SORT BY performs partial ordering
    - Sorts output from each reducer

Advanced Hive Operations

- JOIN
  - If only one column in each table is used in the join, then only one MapReduce job will run
    - This results in 1 MapReduce job:

      ```
      SELECT * FROM a JOIN b ON a.key = b.key JOIN c ON b.key = c.key
      ```

    - This results in 2 MapReduce jobs:

      ```
      SELECT * FROM a JOIN b ON a.key = b.key JOIN c ON b.key2 = c.key
      ```

  - If multiple tables are joined, put the biggest table last and the reducer will stream the last table, buffer the others
  - Use left semi-joins to take the place of IN/EXISTS

    ```
    SELECT a.key, a.val FROM a LEFT SEMI JOIN b on a.key = b.key;
    ```

Advanced Hive Operations

- JOIN
  - Do not specify join conditions in the WHERE clause
    - Hive does not know how to optimise such queries
    - Will compute a full Cartesian product before filtering it
- Join Example

```
SELECT
  a.ymd, a.price_close, b.price_close
FROM stocks a
JOIN stocks b ON a.ymd = b.ymd
WHERE a.symbol = 'AAPL' AND
      b.symbol = 'IBM' AND
      a.ymd > '2010-01-01';
```

Hive Stinger

- MPP-style execution of Hive queries
- Available since Hive 0.13
- No MapReduce
- We will talk about this more when we get to SQL on Hadoop

References

- http://hive.apache.org