HIVE

Data Warehousing with Hadoop

Related Work

* Google's Sawzall
* Yahoo's Pig
* IBM Research's JAQL
* Microsoft's SCOPE
* Greenplum's MapReduce
* Aster Data's In-Database MapReduce

Running Hive

* <install Hadoop>
* wget http://mirror.facebook.com/facebook/hive/hadoop-0.17/dist.tar.gz
* tar zxvf dist.tar.gz
* cd dist
* export HADOOP=<path to bin/hadoop from your Hadoop distribution>
* bin/hive
* hive>

Hive Config

* **conf/hive-default.xml:**
  * **hadoop.bin.path:** Points to bin/hadoop in your Hadoop installation.
  * **hadoop.config.dir:** Points to conf/ in your Hadoop installation.
  * **hive.exec.scratchdir:** HDFS directory where execution information is written.
  * **hive.metastore.warehouse.dir:** The directory inside of HDFS "managed" by Hive.
  * The rest are MetaStore-related, and that's in flux, so we'll worry about it later.
* **conf/hive-log4j.properties**
  * Puts data in /tmp/{username}/hive.log by default
* **conf/jpox.properties**

Populating Hive

* <set hive.metastore.warehouse.dir to “hively”>
* CREATE TABLE users (userid INT, name STRING, num_friends INT)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
* (in the DFS: /hively/users is created)
* LOAD DATA LOCAL INPATH 'users.csv' INTO TABLE users;
* (in the DFS: /hively/users/users.txt is created)
* Advanced: use a SerDe; use CLUSTERED BY (userid) INTO 512 BUCKETS
* Use distcp to get data from filer into HDFS, then use LOAD to put it into Hive

Create Table Syntax

CREATE [EXTERNAL] TABLE table_name (col_name data_type [col_comment], col_name data_type [col_comment], ...)
[PARTITIONED BY (col_name data_type [col_comment], col_name data_type [col_comment], ...)]
[CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name, ...)] INTO num_buckets BUCKETS]
[ROW FORMAT row_format]
[STORED AS file_format]
[LOCATION hdfs_path]

NB:
- PARTITION columns are virtual columns; they are not part of the data itself but are derived when loading the data set.
- CLUSTERED columns are part of the data set, and are used for hash partitioning, often inside of a partitioned column.
- ROW FORMAT can be used to specify delimited serialization formats or more complex formats via a SerDe.
- If your data already lives in HDFS and you'd like to query it in place, you can use EXTERNAL and LOCATION.

Load Table Syntax

LOAD DATA [LOCAL] INPATH '/path/to/file'
[OVERWRITE] INTO TABLE table_name
[PARTITION (partition_col = partition_col_value, partition_col = partiton_col_value, ...)]

NB:
- Data can be loaded from the local file system or HDFS.
- Data can overwrite be appended to existing data.

Running Custom MapReduce

* import sys
  vowels = ['a', 'e', 'i', 'o', 'u']
  for line in sys.stdin:
   line = line.strip()
   userid, name = line.split('\t')
   num_vowels = sum([name.count(vowel) for vowel in vowels])
   print '%s,%s,%s' % (userid, num_vowels, len(name))

* FROM users a
  INSERT OVERWRITE LOCAL DIRECTORY 'hiveout'
  SELECT
   TRANSFORM (a.userid, a.name)
   AS (userid, num_vowels, len_name)
   USING 'python /path/mapper.py';

MetaStore

* Uses Derby embedded database for persistence
* Next release will use MySQL
* Would like to have pluggable storage backend
* Interface: src/contrib/hive/metastore/if/hive_metastore.thrift

Query Processor

- Compiler
  - Parser
  - Type Checking
  - Semantic Analysis
  - Plan Generation
  - Task Generation
- Execution Engine
  - Plan
  - Operators
  - UDFs and UDAFs

Future Work

- Integrate with Hadoop trunk
- Multiple GROUP BYs in a single scan
- Registries for UDFs and UDAFs; cf. Piggy Bank
- JDBC driver
- SerDe refactoring
- ALTER TABLE enhancements: change column name, set SerDe properties
- Table statistics in MetaStore
- Integrate Pig with MetaStore
- Allow HBase tables to be managed by MetaStore and serve as execution targets
- Move insert clause to bottom, as in SCOPE
- Better debugging support in shell

FIN.