Big Data Paper Summary

Jing Tai Wu
Database Management
Professor Labouseur

HIVE: Main Idea

- The design goal of HIVE is to be able to process extremely large data efficiently.
- Hadoop, a map-reduce implementation, had already been used by big companies like Facebook to process large data.
- However, Hadoop is very low-leveled and simple. The end users (programmers) must write customized map-reduce programs to perform simple query tasks, and customized codes are generally not reusable.
- Also, Hadoop is not expressive compare to SQL Languages.

How it's implemented

- HiveQL is built on top of Hadoop.
- Unlike Hadoop which is unstructured, HiveQL uses tables, columns, and partitions like many other query languages.
- HiveQL includes a system catalog called Metastore, and supports primitive types such as Integers, Floating point Numbers and Strings as well as complex types like associative arrays and structs.
- The logical data units are table. Tables' information is stored in HDFS (Hadoop file system).

Continued

- There are three types of primary data units:
  - Tables: stored in a directory
  - Partitions: subdirectory of a table's directory
  - Buckets: a file within a directory that tells if a table is partitioned or not.
- There are a few important building blocks in Hive:
  - Metastore: system catalog and metadata about tables
  - Driver: Manages HiveQL statements
  - Query Compiler: converts queries into map/reduce tasks.
  - Execution Engine: executing converted tasks.
  - HiveServer: allows interaction with other applications
  - Extensibility: allows user defined aggregate functions

Analysis

- It is definitely more efficient to write queries in SQL-like languages than writing customized map/reduce code.
- Hive supports complex types and allows nested complex types. Personally I think nesting complex types is very confusing, but for people who know what they are doing, it could make the database more effective.
- I can treat HiveSQL like PostgreSQL except there is more unique features to it.
- All inserts in Hive overwrites existing table.

Continued

- Hive does not support INSERT INTO, UPDATE, and DELETE. This could be very dangerous because when we are designing databases, we want to make sure we follow three rules for normal forms (four if we include BCNF). These rules ultimately eliminates INSERT, DELETE, and UPDATE anomalies. I am not too sure if that limitation would cause potential problems in implementations.

Comparison

- Since Hive is built on top of Hadoop, one must examine the speed of various tasks between Hadoop and traditional SQL languages.
- Hadoop excels in data loading. Compared to Vertica and DBMS-X, the load times for Hadoop is significantly less than both database systems.
- Besides the data loading time, Hadoop does not perform very well in pattern-matching queries, aggregation tasks, as well as join tasks. This underlying performance issue may greatly impact the usage of Hive in large scale data.

Continued

- Although transactions were not really mentioned in Hive's paper, in Large-Scale data summary, they suggested that MR (Hadoop in this case) is able to recover from faults in the middle of query execution. This can significantly increase the reliability of Hive and be used as a great advantage when comparing to other languages.

Advantages and Disadv.

- Advantages:
  - Hive is faster at loading data.
  - MR (Hadoop) is able to recover from faults if transactions fail. Since Hive is built on top of Hadoop, this makes Hive more reliable.
  - More SQL like, supports relational data model, uses tables, columns.

- Disadvantages:
  - Slower in every other aspect (pattern-matching, aggregation, selection)
  - Does not support INSERT INTO, UPDATE and DELETE.
  - only equality is supported in a join predicate.