Hive - A Petabyte Scale Data Warehouse Using Hadoop

A Comparison of Approaches to Large-Scale Data Analysis

Michael Stonebraker on his 10-Year Most Influential Paper Award

25 October 2017

Thusoo, Ashish, et al. “Hive - A Petabyte Scale Data Warehouse Using Hadoop.”.

Andrew, Pavlo, et al. “A Comparison of Approaches to Large-Scale Data Analysis.”.

“Michael Stonebraker on his 10-Year Most Influential Paper Award.” 2005.

By: Ariele Silva

Main Idea

- Hadoop - a popular open-source map-reduce implementation to store and process extremely large data sets on commodity hardware.
- Map-reducing benefits
- Extensions of Hadoop
  - Hive - an open-source data warehousing solution built on top of Hadoop
- HiveQL
  - A subset of SQL

Implementation

- Hadoop executes map-reducing jobs and provides scalability using commodity hardware.
- Hadoop was used to replace Facebook’s previous infrastructure
- HiveQL enables users to plug in custom map-reduce scripts into queries
    - Allows users to extend the system with their own types and functions
- Hive buckets, a bucket that is stored in a file within the partitions or tables directory

Analysis of Idea & Implementation

- HiveQL helps reduce the number of scans done on input data
- Facebook being one of Hadoop's customers, the scalability portion is really beneficial considering Facebook's size and it's ever growing community
- The customization of map-reduce and the fact that the users have the ability to expand on the system through their own functions and types gives the user so much freedom to explore and "mess around" with Hadoop
- The similarities between HiveQL and SQL are beneficial since most users are already familiar with all of SQL's functions and features.

Main Ideas: Comparision Paper

- Comparing both paradigms
  - MapReduce
  - SQL DBMS
- Cluster computing - harnessing large numbers of (low-end) processors working in parallel to solve a computing problem.
  - MapReduce clusters
- Parallel databases

Implementation: Comparison Paper

- MapReduce clusters are being used to teach students about distributed programming
- Collections of partitions are used to store MapReduce’s data on each node of a cluser
- Split functions are the results of the Map function producing output records
- Parallel DBMS’s consist of 3 phases
  - Filter subquery
  - Join algorithms
    - Small DBMS → replicate the number of records on all nodes
    - Large DBMS → number of records are distributed across multiple nodes
  - “Roll up” computation

Analysis of Ideas & Implementation: Comparison Paper

- By implementing a large number of low-end servers, the processors will work in parallel to solve the computing problem, rather than using a smaller number of high-end servers
- Programmers don’t need to worry about the underlying storage details of indexing and join strategies, they just specify their goal in the programming language
- Writing in a declarative language like SQL over Java can be beneficial since programmers will have to be more specific with their coding
- Hadoop is written in Java
    - I prefer Java rather than C++ (Google’s MapReduce is written in C++)

Main Idea: Stonebraker Talk

- Making RDBMS the answer to any question
- RDBMS → one size fits all (ultimately never was going to work)
- Based on a sample of 3, one size did not fit all
  - One size fits none
- C-Store’s are faster
- OLTP (online transaction processing) engines have greater market share

Advantages & Disadvantages

Advantages

- SQL DBMS were the fastest compared to the others
- MapReduce programming model is simple
- Hadoop’s ability to allow users to add in their own types and functions
- Ability to increase scalability
- MapReduce scheduler automatically restarts the task on alternate nodes if it fails the first time

Disadvantages

- SQL DBMS took longer to tune and load the data.
- RDBMS are not one size fits all
  - One size fits none
- Traditional row stores are not good for markets like:
  - Streaming market
- Hadoop is limited by speed
  - Must parse the tables on the fly