Christina Ramos

December 8, 2014

Hive – A Petabyte Scale Data Warehouse Using Hadoop

Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden,

Michael Stonebraker

Main Idea of First Paper

Hadoop is awesome for Facebook because it can handle petabytes of data very quickly using a map-reduce function. However, end-users find Hadoop cumbersome and confusing to use, so Hive was created by Facebook to feel like SQL but plug into Hadoop. Hive uses tables with rows and columns so that querying feels familiar and easy.

How This is Implemented

Hadoop uses map-reduce functions, so writing programs for Hadoop is a lot harder than good old SQL. So, Hive uses a Query Compiler that compiles SQL into a directed acyclic graph of map-reduce functions that Hadoop can work with.

Hive also implements a Java SerDe that allows users to plug in legacy data or data created by another program without having to transform the data.

HiveQL is Hive's flavor of SQL. It's very similar, but lacks support for UPDATE, INSERT, and DELETE. It adds a bit of functionality in their steads, though. This includes a CREATE AS SELECT statement.

Hadoop also has its own file system, the HDF. Data is mapped to directories in the HDF, which allows Hadoop to find data faster by only searching relevant directories.

Analysis

I think this is pretty neat. I'm all for making something weird and ugly into a more accessible platform for the average programmer (or in this case, database developer). I personally enjoy SQL, so it's pretty cool to hear that a big company like Facebook decided to create something to make big data processing easier for the end user, even it was really for themselves originally.

I did a bit more searching and found that Hive does not support nearly as much functionality as SQL does, and I was especially surprised that it doesn't support INSERT, DELETE, or UPDATE. However, the paper assured me that a company like Facebook does not need such functionality, which I found strange. I found it amusing that by avoiding those statements, it was actually easier for Facebook because they didn't have to implement complicated locking systems to prevent finagling.

Comparison To Second Paper

The second paper explored the advantages and disadvantages of both DBMSs and MRs. The second paper used several benchmarks to measure the performance of Hadoop versus that of two DBMSs, Vertica and DBMS-X. In general, Vertica seems to come out on top. Hadoop excels in load times, but fails in query times. The gist of the second paper is that Hadoop is great for loading once and never re-accessing data, while Vertica is great for when you need to repeatedly access the same data.

This isn't entirely true for Hive. Hive has the loading speed and power of Hadoop, but with its HiveQL layer, somewhat better query times. It's still preferable to use Hive for a warehouse that doesn't need repeated access.

Advantages and Disadvantages

Advantages

Hive still runs on Hadoop, so it's great at loading data very quickly, as shown in the benchmark study done in the second paper.

Easy to program for because of the SQL layer.

Tolerates faults better because of Hadoop.

Disadvantages

Hive still runs on Hadoop, so it's not so great at doing fast queries and repeated access, as shown by the benchmark study.

Hive doesn't support indexing.

Hadoop fails miserably at every other benchmark test besides the first data loading test.