Building a High-Level Dataflow System on Top of Map-Reduce: The Pig Experience

Alan F. Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, Utkarsh Srivastava

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

•••

Michael Stonebraker on his 10-Year most Influential Paper Award at ICDE 2015

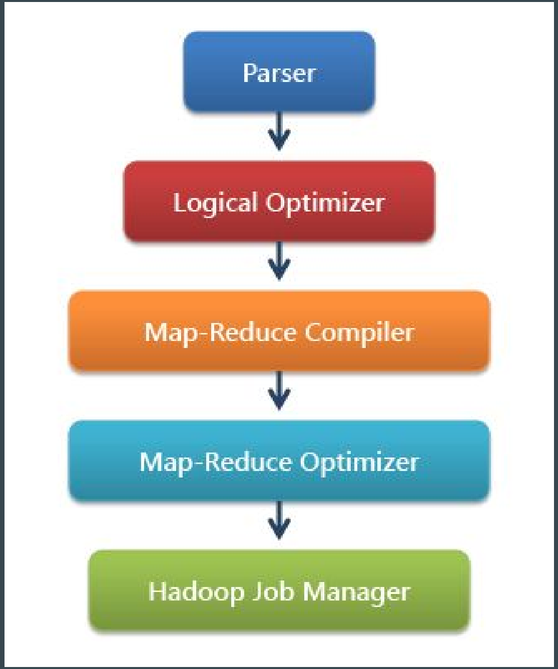
Mackenzie O'Brien 3/7/2017

Main Idea- Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience

- Potential with Map-Reduce and its scalability
- Map-Reduce’s simplicity lead to problems in practice
    - “Does not directly support complex N-step dataflows”
    - “Lacks explicit support for combined processing of multiple data sets” such as joins
    - “Frequently-needed data manipulation primitives like filtering, aggregation, and top-k thresholding must be coded by hand”
- Yahoo!’s answer to the problem was Pig, which is a combination of SQL and Map-Reduce which allows SQL-esque manipulation on top of Map-Reduce
- The programs allow encoded explicit dataflow graphs like Map-Reduce instead of SQL’s implicit dataflow structure

Implementation- Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience

Every Pig Program goes through a series of steps before being executed:



1. Parser
   a. Verifies the program's syntax is correct and all variables are defined
   b. Type checking and schema inference are also performed here
2. Logical Optimizer
   a. Logical optimizations such as projection pushdown are performed here
3. Map-Reduce Compiler
   a. Pig translates the logical plan to a physical plan and turns it into a Map-Reduce plan, assigning physical operators to Hadoop stages for optimization
4. Map-Reduce Optimizer
   a. Pig breaks down optimization of distributive and algebraic aggregation functions into 3 steps
      i. Initial- generate(sum,count) pairs MAP
      ii. Intermediate- combine n(sum,count) pairs COMBINE
      iii. Final- combine n (sum, count) pairs and take the quotient REDUCE
5. Hadoop Job Manager
   a. Generates a Java .jar file that contains the Map and Reduce classes as well as any user-defined functions that will be used

Analysis- Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience

- Pig aims to provide both the scalability of Map-Reduce and the high-level data manipulation of SQL
- Pig is implemented on top of Map-Reduce and breaks down the high-level functions into stages easily read by Hadoop

The idea behind Pig is great; combining Map-Reduce with SQL gives extremely high data manipulation. However, as the earliest data manipulation construct created for Map-Reduce environments, it is not better than later creations, but it did set a high benchmark for other constructs to measure up to. Even so, the best suited environment depends on what the task at hand is and each system has its benefits.

Main Idea - A Comparison of Approaches to Large-Scale Data Analysis

Hadoop and Map-Reduce are not perfect despite the growing enthusiasm for them

When tested against 2 separate parallel SQL DBMS (Vertica and DBMS-X), the DBMS's achieved faster results with less code

There is no current real-world example needing all 1,000 nodes that Map-Reduce is capable of computing (as of 2009)

While both achieve the same resultss, there are benefits to both

- The DMBS is more expensive, but faster
- Map-Reduce systems take longer but are cheaper up-front and easy to use with scalability

Implementation- A Comparison of Approaches to Large-Scale Data Analysis

3 systems were tested within a benchmark environment on different tasks

Systems:

- Hadoop - most popular open-source implementation of Map-Reduce
- Vertica - Parallel DBMS from a major relational database (stored in rows)
- DBMS-X - Parallel DBMS designed for large data warehouses (stored in columns)

Tasks:

- Grep Task - "Original MR Task"
    - Scan through 100 byte data set looking for a 3 character pattern that only appears 1 in 10,000 records from input stored in plain text files
- Analytical Tasks
    - Data loading, selections, aggregations, joins, UDF aggregations

Analysis- A Comparison of Approaches to Large-Scale Data Analysis

Both DBMS's outperformed the Map-Reduce system when tested with the benchmarks. The paper concludes that both systems have their uses and the user should determine which aspects of big data the value more; ease of use or minimizing data loss

The paper gave credit to Hadoop/Map-Reduce in terms that the system is written primarily in Java, an object oriented language that is more familiar to most than SQL

The user must determine whether ease of use and scalability outweigh the pros of DBMSs performance speed and low maintenance needs.

Overall, more testing should be done to stress both systems and compare the results

Comparison of the Ideas and Implementations of the 2 papers

The first paper discusses how to implement a high level dataflow system on top of Map-Reduce in order to have data manipulation similar to SQL, while still maintaining Map-Reduce's scalability and simplicity.

This was done by converting the "Pig Latin" to "Map" and "Reduce" through a series of phases. This increased the performance performance ratio to 1.5 when compared to raw Map-Reduce

The second paper compared Map-Reduce systems to DBMSs, and while the DBMS's outperformed Map-Reduce in the given tasks, it still credited Map-Reduce with being easy to use

Both papers agree that Map-Reduce is not perfect, and there are different versions better suited for some tasks, but that it does have its uses for working with big data. THe papers also agree that there is not a definitive "Winner" in terms of best system to work with

Main Idea - Stonebraker Talk

Michael Stonebraker insists that one size does not fit all, and instead suggests that one size fits none for DBMSs.

Row-based relational databases are being phased out for other options such as column-based and NoSQL

Stonebraker goes through different markets that use Database Management Systems and explains how the old model of relational databases no longer works or is the best option for ANY of the markets.

He believes that the market is not stagnant, but rather bursts with different “best fit” systems for each market

Advantages and Disadvantages of Main Paper vs. Comp Paper and Stonebraker Talk

Advantages:

- Vs. Comparison Paper
  - Pig tries to be “the best of both worlds” in terms of fast performance and scalability/ease of use
  - Modifies Hadoop with Pig to use SQL-like queries on top of Map-Reduce data to increase performance
- Vs. Stonebraker Talk
  - Moves away from traditional relational databases that are “dying”
  - Has potential to manage petabytes of data
  - Is a system that a database researcher would work on

Disadvantages:

- Vs. Comparison Paper
  - Argues that DBMSs are still currently better in performance than Map-Reduce systems
  - Map-Reduce is not as effective as DBMSs
- Vs. Stonebraker Talk
  - Not one of the “best fit” systems described in the talk

Overall, the comparison paper and talk agree that there is no best fit, as everything has a specific use and context where it should be used