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 Abadi, David DeWitt, Samuel Madden, and Michael Stonebraker

G Leaden
10/19/2016

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

Saw potential with *Map-Reduce* and its extreme simplicity and scalability.

In practice, the simplicity of *Map-Reduce* leads to problems:

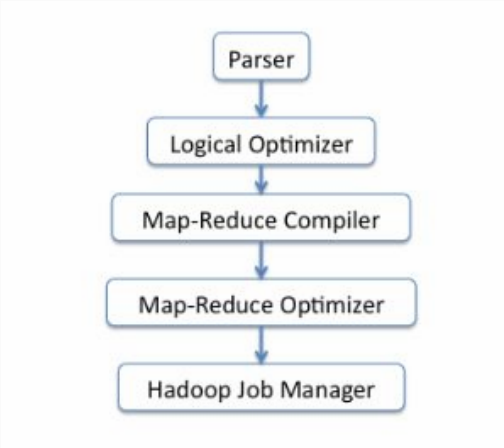
1. *"Map-Reduce* does not directly support complex N-Step dataflows"
2. *"Map-Reduce* lacks explicit support for combined processing of multiple data sets (e.g., joins and other data matching operations)"
3. "Frequently-needed data manipulation primitives like filtering, aggregation and top-*k* thresholding must be coded by hand"

Pig is Yahoo!'s answer to Map-Reduce's problems. *Pig* "aims at a sweet spot between SQL and Map-Reduce" which offers SQL-style manipulation on top of Map-Reduce, thus retaining the properties of Map-Reduce systems which make Map-Reduce appealing in the first place.

Pig programs, as with Map-Reduce, encode **explicit** dataflow graphs, as opposed to **implicit** dataflow utilized in SQL.

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

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



1. Parser
   a. Verifies the program is syntactically correct and all referenced variables are defined
   b. Other checks such as type checking are also performed
2. Logical Optimizer
   a. Logical optimizations such as projection pushdown are performed
3. Map-Reduce Compiler
   a. Pig translates the now optimized logical plan to a physical plan and translates that physical plan to a Map Reduce plan - assigning physical operators to Hadoop stages
4. Map-Reduce Optimizer
   a. There is room for more optimization
   b. Pig only does one: breaking down distributive and algebraic aggregation functions into 3 smaller 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

At the time of the paper (2009) Pig is used in roughly 60% of all ad-hoc and 40% of all production Hadoop jobs within Yahoo.

Outside of Yahoo there has been steady user growth due to it being easy to learn and its open source nature.

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

Pig aims to be a solution that lies in-between SQL and Map-Reduce providing both the scalability with Hadoop, and the high-level data manipulation constructs modeled in the spirit of SQL.

Pig is implemented on top of Map-Reduce and translates / breaks down its high level functions to Map, Combine, and Reduce stages to be easily read by Hadoop.

I think the idea of giving Map-Reduce an easy way to manipulate data on a high level like SQL is a fantastic idea that gives you the best of both worlds (see above). The next question is: “Is Pig better than other options that are currently available?” Pig was developed before or alongside DryadLINQ, Hive, Jaql, and Scope, all of which offer high level data manipulation constructs for Map-Reduce like environments.

Ultimately, Pig does what it aims to do with a surprising performance benchmark. Pig is **not**, however, the perfect solution. Each system listed above has its use within Map-Reduce environments and even outside of those systems, Map-Reduce may not be the system best suited for the task. It is all about context and what you need your system to do.

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

Hadoop / Map-Reduce is not perfect despite the hype (“considerable enthusiasm”)

Tested Hadoop vs two separate parallel SQL DBMSs (Vertica, DBMS-X)

In General the SQL DataBase Management Systems (DBMS) achieve in faster results with less code to come to the same result when compared to Hadoop

While Map-Reduce is capable of 1,000s of nodes, there is no clear real world example where that many nodes is necessary as of the paper’s publication date (2009)

There are best practices for both DBMSs and Map-Reduce systems. There is a way to achieve the same results whether or not the user uses a DBMS or a Map-Reduce system. Overall SQL DBMSs outperform Map-Reduce systems in almost every area. Map-Reduce systems are still appealing due to their low up-front cost, ease of use, and extreme scalability which may one day become necessary.

Map-Reduce is not the DBMS killer yet and should not be considered a “better” alternative, rather in today’s uses it is just that, an alternative.

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

Tested three systems within a benchmark environment using different tasks

- Tested Systems:
  - Hadoop
    - Most popular open-source implementation of the Map-Reduce framework.
  - DBMS-X
    - Parallel SQL DBMS from a major relational database (data stored in rows)
  - Vertica
    - Parallel DBMS designed for large data warehouses (data stored in columns)
- Tasks Used:
  - Grep Task “Original MR Task”
    - Data loading and task execution
  - Analytical Tasks:
    - Data loading
    - Selections
    - Aggregations
    - Joins
    - UDF aggregations

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

Both the parallel DBMSs heavily outperformed the Map-Reduce system with Hadoop in terms of the paper's benchmarks. The paper concludes claiming that each have their uses and it is up to the user to determine what aspects of big data they value more, performance vs ease of use and minimizing loss of data

Hadoop and in turn Map-Reduce systems were given some credit however when it comes to ease of use where it excels due to Map-Reduce programs being written primarily in Java, an object oriented language that is much more familiar than SQL to an "average programmer". Now whether or not an "average programmer" should be writing such a system that requires the scalability Map-Reduce offers, is something else entirely.

There is a fine line between sacrificing performance for ease of use, obviously not many programmers are going to use assembly just because it is faster than java, but there needs to be a happy medium where the sacrifice is not too great. Map-Reduce's pros (ease of use, scalability, lower up-front cost) do not outweigh the pros of DBMSs.

Map-Reduce's ease of use is only argued up to a point, where SQL again takes the lead due to the DBMS being much easier to maintain over time, with less changes throughout the code when the schemas are modified.

The scope of the benchmarks should also be questioned as to whether or not it is enough to fully stress both the DBMS and Map-Reduce system.

Comparison of the Ideas and Implementations of the two papers

The first paper discussed a way to build a high level dataflow system on top of a Map-Reduce system.

This was done in order to have SQL-like data manipulation with the scalability and ease of use given by Map-Reduce.

This was implemented by a series of phases and translations that converted the “Pig Latin” to “Map” and “Reduce” while also producing surprising benchmark results stating the performance ratio of a pig execution vs raw Map-Reduce times to be 1.5 (1 would result in parity).

The second paper compared Map-Reduce to the already widely used and battle tested DBMS.

The second paper stated that Map-Reduce was not the perfect solution and that DBMSs still outperformed Map-Reduce at almost every task given. The paper did, however, credit Map-Reduce to having a much easier up-front set-up as well as ease of use due to its use of java over SQL as its primary programming language.

I believe both papers agree on the fact that Map-Reduce is not perfect and the first paper is an attempt to improve or modify Map-Reduce to fit specific needs. Both papers also agree that Map-Reduce does have a place within big data but it is not clear as to whether or not there is a definitive best system to work with big data.

Main Idea - Stonebraker Talk

Michael Stonebraker is adamant that one size does NOT fit all and instead one size fits none in terms of DBMSs. Row based relational database management systems are being almost entirely phased out for other options such as column based, and NoSQL.

Stonebraker goes through many different markets that use database management systems and explains in each one how the old model of relational DBMSs does not work or is no longer the “best fit” for ANY of the markets.

Stonebraker states that contrary to the widely held belief in the 80s and 90s the market is not becoming stagnant but rather as of now exploding with different “best practice” or “best fit” systems / paradigms for each individual market.

It’s a great time to be a database researcher

Examples:

- Data Warehouse market - column stores (2 orders of magnitude faster)
- OLTP market - main memory deployments
- NoSQL market - key value / big table clones / JSON stores
- Streaming market - stream processing and OLTP engines combined

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

Advantages of Pig and creating a high level dataflow system on top of Map-Reduce:

- Vs. Comparison Paper
  - Pig entertains the idea of a “best of both worlds” scenario where one can achieve both faster performance than Map-Reduce and better scalability and ease of use
  - Modifies Hadoop with Pig to increase performance and use an SQL-like query system on Map-Reduce data
- Vs. Stonebraker Talk
  - Is moving away from the standard relational database model that Stonebraker claims is dying
  - Has the capability of managing petabytes of data, something that has yet to be realized to its full potential
  - Is a system that a database researcher would work on, and it is a great time to be a database researcher

Disadvantages:

- Vs. Comparison Paper
  - The comparison paper argues that DBMSs are still the best performance wise and users should not jump on the Hadoop hype train too soon
  - Uses Hadoop and Map-Reduce for data that the comparison paper claims can be put in a DBMS and be used more effectively
- Vs. Stonebraker Talk
  - Not explicitly one of the “best fit” systems described in the talk

The overall gist of both the comparison paper and the stonebraker talk is that there is no best fit and nothing should be claimed as such, everything has a specific use and in the context of its use is where one can really judge a big data management system and its effectiveness.