“MapReduce: Simplified Data Processing on Large Clusters”

(Jeffrey Dean and Sanjay Ghemawat)

“A Comparison of Approaches to Large-Scale Data Analyis”

(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

at ICDE 2015

Bridget Leahy

March 7, 2017

MapReduce Main Ideas:

- This paper discussed MapReduce, a system designed to processes and generate large data sets. The design was based on the need to perform simple computations while concealing the details of data distribution, fault-tolerance, parallelization, and load-balancing.
- MR takes a set of input key/value pairs and produces a set of output key/value pairs. These are both written by the user and done in two different functions: map and reduce.
- MR is useful for systems such as Google’s production web search service, sorting, data mining, and machine learning.
- MR relies on in-house cluster management system that is responsible for distributing and running user tasks on a large collection of shared machines.

MapReduce - Implementation

- Map takes an input pair and produces a set of intermediate key/value pairs. All intermediate values associated with the same key are passed to the reduce function, where the values are merged to form a smaller set. This provides a solution to handle lists of values to large to fit in memory.
- MR can be good form computing distributed greb, count of URL access frequency, Reverse Web-Link Graph, Term-Vector per Host, Inverted Index, and Distributed Sort.
- Different implementations of MR are better depending on the environment. What worked best at Google will not work best everywhere else.
- MR conserves network bandwidth, re-executes map tasks on failure, has an optional Combiner function to merge data before it is sent over the network, finds a solution for machines taking too long to complete a task, allows new input types based on implementation of a reader interface, along with other developments and features.
- MR was tested on a cluster of approximately 1800 machines to measure its performance.

Analysis of MapReduce and Implementation

- The success of MR is credited to allowing the possibility to write a simple program which runs efficiently on a thousand machines, speeding up development and prototyping cycle. Those who lack experience with distributed or parallel systems are able to easily utilize this.
- While there are shortcomings including crashing on records and debugging, MR allows simple data processing across a wide domain. The simplistic nature makes it attractive.
- User is writing the code for the Map and Reduce functions, which are easy to understand.

“A Comparison of Approaches…”

- This paper discussed and explored the differences between MapReduce and parallel database systems. While there are common elements, notable differences are considered throughout tests.
- Approaches to data analysis, as with anything, have both strengths and shortcomings - must choose a trade-off depending on individual circumstances.
- MapReduce provides a simple model for more sophisticated use. MR consists of two functions, Map and Reduce, written by users. MR reads records and outputs them in key-value pairs, stored on collection of partitions in distributed file system deployed on each node in a cluster. An MR scheduler decides how to allocate nodes and run both Map and Reduce functions,
- Parallel DBMS , supporting standard relational tables and SQL, have existed since the late 1980s. Most tables in DBMS are partitioned over the nodes in a cluster and the system translates SQL commands into a query plan.

Implementation of the Comparison

- The authors compared MR and parallel DBMS based on architectural elements consisting of schema support, indexing, programming model, data distribution, execution strategy, flexibility, and fault tolerance.
- They then conducted tasks to test performance benchmarks. They used *Hadoop,* an implementation of the MR framework, written entirely in Java. For parallel DBMS, they used both *DBMS-X* and *Vertica*.
- Each system was deployed on 100-node cluster and given Grep tasks, such as system installation and configuration, data loading, task execution, and analytical tasks such as selection, aggregation, join, and UDF aggregation.
- It was found that Hadoop outperformed the parallel DBMS for installation. However, Hadoop's shortcomings began there. The parallel database systems have performance advantages in most tasks.

My Analysis:

- Despite their own drawbacks, each system does its job to manage big data effectively in their own way.
- The choice for a data management system is dependent on what the specific case for data analysis may be. The Grep Task and others provide insight on the matter. The tests chosen were able to showcase the abilities of the systems to carry out equivalent functions.
- This type of comparison work is important. It sparks the opportunity for adjustments on both sides. The paper ends with the idea that the two systems can move towards each other. Extracting the best features from systems to integrate together may lead us into the future for big data.

MapReduce vs “A Comparison...”

Both papers are written with a common goal: dealing with big data efficiently and successfully. However, the intent and content for each paper was different.

- In the MapReduce paper, the authors are focused only on MapReduce and educating readers solely on that topic.
- They highlight what need brought on the development of MapReduce. They discuss what features MR can do and how it works.
- Discussed cluster configuration, the Map and Reduce functions, sample code, and performance tests which had been implemented.

- The comparison paper seeks to establish a comparison between MR and parallel DBMSs.
- Describe both in detail and then illustrate tests performed on both.
- This revealed strengths and shortcomings for both, allowing readers to see what situations would be best to use which system.
- Suggested the two classes of systems could move toward each other.

Stonebraker Talk Main Ideas

- Michael Stonebraker introduces his talk saying he was trying to find a “one size fits all”, universal relational database. However, he realized this would never work.
- He realized that one size actually fits none and that traditional row stores are now obsolete and good for nothing.
  - Data Warehouse market → vendors will have column stores soon, faster than row stores
  - OLTP market → moving toward main memory deployments, different techniques from traditional row stores and lightweight transactions
  - NoSQL market → no standards, none appear traditional
    - key -value stores, record stores, bigtable clones, JSON stores
  - Complex Analytics → Regression, SVD, data clustering, etc. all defined on arrays not on tables
    - Waiting to see how this will unfold
  - Streaming market → not based on traditional row stores, doing well
    - S-Store → add streaming to OLTP engine
  - Graph Analytics market → simulate in a column or array engine, traditional row stores aren’t even considered
- Stonebraker concludes that there is a huge diversity of engines in which traditional row stores have no place. He says it is the time for new ideas: NVRAM, big main memory, processor diversity, higher speed networks, LLVM, and vectorization.

MapReduce paper in the context of the comparison paper and the Stonebraker talk

ADVANTAGES:

- The MapReduce paper provides great detail into MR itself - its functionality, uses with large clusters.
- It was useful to learn such in depth detail about MR.
- Written by people who are passionate and knowledgeable about MR.

DISADVANTAGES:

- While the MapReduce paper did highlight some limitations, it did not fully speak to all that it lacks.
- The comparison paper discusses MR's shortcomings in much greater detail - highlighting where it did not match up to parallel DBMS.
- Stonebraker brought up many ideas and providing insight on a variety of topics, rather than learning about just one thing.