Big Data Paper Summary

By: Katelyn Boylan
CMPT 308: Database Systems
March 7, 2017

A Comparison of Approaches to Large-Scale Data Analysis by Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel Abadi, David DeWitt, Samuel Madden, & Michael Stonebraker. *OSDI.* Published on July 2, 2009.

MapReduce: Simplified Data Processing on Large Clusters by Jeffrey Dean & Sanjay Ghemawat. *Google, Inc.* Published in 2004.

Paper 1: MapReduce: Simplified Data Processing on Large Clusters

- MapReduce is a “programming model and an associated implementation for processing and generating large data sets.”
- MapReduce is easy to use and is easily expressible for a large variety of problems.
- The Map function processes a key/value pair that generates a set of intermediate key/value pairs.
- The Reduce function merges all intermediate values associated with the same implementation key.
- Distributed Grep, Count of URL Access Frequency, Reverse Web-Link Graph, Term-Vector per Host, Inverted Index, & Distributed Sort are all programs that can be easily expressed as MapReduce computations.
- Restricting the programming model makes it easy to parallelize and distribute computations and to make such computations fault-tolerant.
- Network bandwidth is a scarce resource and can be saved through the use of intermediate data.
- Redundant execution can be used to reduce the impact of slow machines as well as handle machine failures and data loss.

MapReduce Implemented

1. Splits Map function input files (M).
2. The Master copy of the program assigns M and R tasks.
3. Worker is assigned a map task reads contents of the corresponding input split.
4. The buffered pairs are written to local disk and are partitioned into R regions.
5. The reducer uses remote procedure calls to read the buffered data from the Map workers.
6. The reduce worker iterates over the intermediate data and passes the values to the user's Reduce function.
7. Master wakes up the program and MapReduce returns back to the user code.
8. The output of the MapReduce execution is available in the R output files.

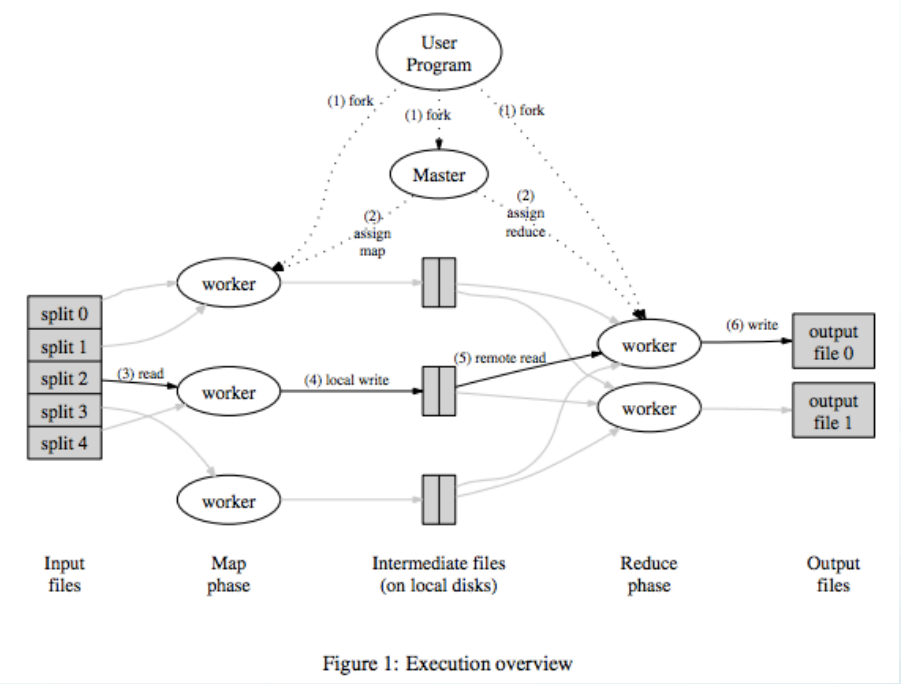

Figure 1: Execution overview

Analysis of MapReduce & its Implementation

- Hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters everyday.
- MapReduce hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library.
- In most cases, the Map function produces an input pair and produces a set of intermediate key/value pairs. The Reduce function accepts this key/value for that key and merges value together to a smaller set of values.
- Many different implementations of the MapReduce interface are possible and the right choice depends on the environment and the size of the project.
- The MapReduce library must tolerate machine failures gracefully given the size of this data. Failures can be from an idle state or local disk problems.
- The master stores the state of the map and reduce tasks. Because there is only one Master, failure is unlikely.
- Compared to other systems, MapReduce exploits a restricted programming model to parallelize the user program automatically and provide transparent fault-tolerance.

Paper 2: A Comparison of Approaches to Large-Scale Data Analysis

- MapReduce (MR) provides a simple model through which users can express relatively sophisticated distributed programs, leading to significant interest in the educational community.
- Parallel Database Systems have been commercially available for nearly two decades, and there are about a dozen in the marketplace.
- DBMS requires that data conform to a well-defined schema. This system is fast and requires less code but takes longer to tune and load the data.
- MR permits data to be in any arbitrary format and is suited for development environments with a small number of programmers and a limited application domain.
- Both systems achieve parallelism by dividing any data set to be utilized into partitions, which allocate to different nodes to facilitate parallel processing.
- The Benchmark in implementation is made up of a collection of tasks that has been run on an open source version of MapReduce and two parallel Database Management systems.
- Higher level interfaces are using the MR foundation and parallel databases are used by commercial and open-source systems.

Benchmarks Implemented

Each system must scan through a data set of 100-byte records looking for a three-character pattern

Hadoop (MR framework)

- Stores all input & output data in distributed file system (HDFS)
- Uses a "master" HDFS daemon to coordinate node activities
- Took more time before all nodes were running at full capacity
- Ease of use is shown in installation
- Performed to a significant disadvantage compared to parallel DBMSs

DBMS-X (Parallel SQL)

- Each table is hash partitioned across all nodes on the salient attribute for that table, and then sorted and indexed on different attributes
- Has the ability to compress tables using dictionary-based scheme
- Surprised system didn't detect administrative errors
- Much faster than MR system in benchmark analysis
- Much more challenging to install and configure properly

Vertica (Parallel DBMS)

- All data is stored as columns, rather than rows
- Vertica processes one query at a time
- Compresses data by default since its executor can operate directly on compressed tables
- Resource manager sets the amount of memory given to queries
- Much more challenging to install and configure properly

Analysis of Benchmark Implementation

- **Figure 1:** (535 MB/node) Compared to Hadoop and Vertica, DBMS-X takes a significantly longer amount of time to process data.
- **Figure 2:** (1TB/Cluster) DBMS-X does not decrease as less data is stored per node because it loads data on each node sequentially.
- **Figure 4:** (535MB/node Data set) Two parallel databases perform the same, more than a factor or two faster than MR system due to the small size of data.
- **Figure 5:** (1TB/Cluster Data set) As fixed cost increases, so does the amount of time it takes for each system to process data. Vertica outperforms DBMS systems with its aggressive use of data compression.
- **Figure 7:** (Aggregation Task Results (2.5 Mil Groups)) 2 DBMSs perform about the same for a large number of groups.
- **Figure 8:** (Aggregation Task Results (2,000 Groups)) It is more adventurous to use a column-store system when processing fewer groups.
- **Figure 9:** (Join Task Results) Performance of Hadoop is limited by the speed with which the large User Visits table can be read off disk and the DBMSs take advantage of the join key.
- **Figure 10:** (UDF Aggregation Task Results) Both DBMS-X & Hadoop have constant performance for this task but with more nodes, DBMS-X is slower with row-by-row interaction.

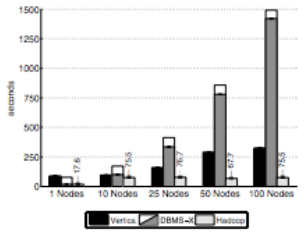

Figure 1: Load Times – Grep Task Data Set (535MB/node)

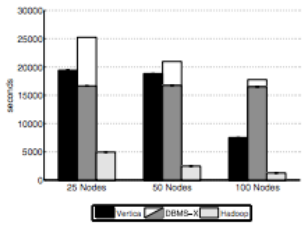

Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

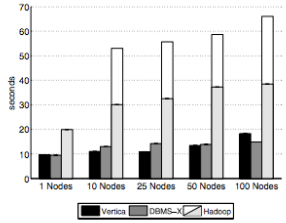

Figure 4: Grep Task Results – 535MB/node Data Set

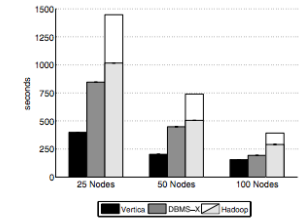

Figure 5: Grep Task Results – 1TB/cluster Data Set

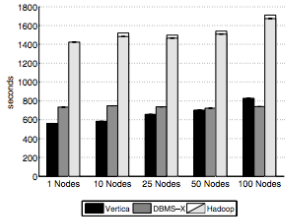

Figure 7: Aggregation Task Results (2.5 million Groups)

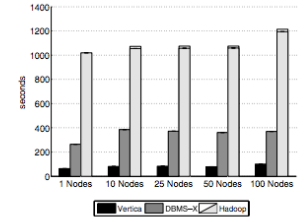

Figure 8: Aggregation Task Results (2,000 Groups)

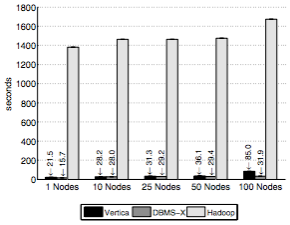

Figure 9: Join Task Results

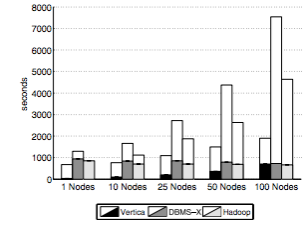

Figure 10: UDF Aggregation Task Results

Comparison Paper vs. MapReduce Paper

Comparison

- Both papers agree that MapReduce is easy to use and appropriate for large sets of data.
- Both papers use examples of how MapReduce is implemented and produce small values from large values through the Master function.

Contrast

- Comparison paper explains that parallel database are the superior system due to its speed and efficiency.
- Comparison paper says that MapReduce should be used with a small number of programmers but the large corporation, *Google Inc.* uses MR.

Conclusion

Comparison paper explains that the Google MR system is different and more complex than the Hadoop system they used, therefore, there is different outcomes for each.

Each system could be used for different reasons, depending on the environment of the big data and the preference of the team.

Stonebraker Talk

- Analysts thought that Relational Database Management Systems were the only answer from 1970-2000.
- In 2005, they realized that one size doesn't fit all and it is a great time to be a DBMS researcher.
- In 2015, one size fits none
  - Data Warehouse stores will be entirely column stores due to the speed
  - OLTP (transaction processing) different SQL interpretation
  - NoSQL Market has 100 or so vendors
  - Complex Analytics consists of Business Intelligence products and graphical front ends to SQL aggregates
  - Streaming Market is the Workflow of record processing with SQL like characteristics.
  - Graph Analytics: Simulation in a column store or an array engine. Examples are Facebook and measuring distance
- There is now a huge diversity within engines. Traditional row stores are good at none of the markets.
  - This creates a great opportunity for new ideas and implementations that legacy vendors will need to keep up with.

Conclusion - Pros & Cons

Comparison paper says that Parallel databases are superior to MR systems, however, Stonebraker explains that “one size fits none” is the future of database systems.

Stonebraker says that traditional row stores are good at none of the markets but Comparison paper says that it is efficient and faster than the new MapReduce system.

The various new database systems have the potential to create complex and powerful databases that will make businesses more efficient and increase the speed of DBMSs.

One disadvantage or advantage of new database systems is that database researches may be irrelevant and taken over by these new systems.

Stonebraker explains that legacy vendors need to keep up with the new wave in database systems in order to not be wiped out by new superior systems. This puts the old vendors at a disadvantage therefore, they must invest in R&D.

In the future, we can expect a massive change in Database systems with the MapReduce function and new relational databases, however, for now relational databases are still functional and effective for multiple purposes.