

# Big Data Paper Summary

Paper 1: “MapReduce: Simplified Data Processing on Large Clusters”

Citation: Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce." *Communications of the ACM Commun. OSDI* 51.1 (2004): 107.

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

Citation: Pavlo, Andrew, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. Dewitt, Samuel Madden, and Michael Stonebraker. "A Comparison of Approaches to Large-scale Data Analysis." *Proceedings of the 35th SIGMOD International Conference on Management of Data - SIGMOD '09* (2009):

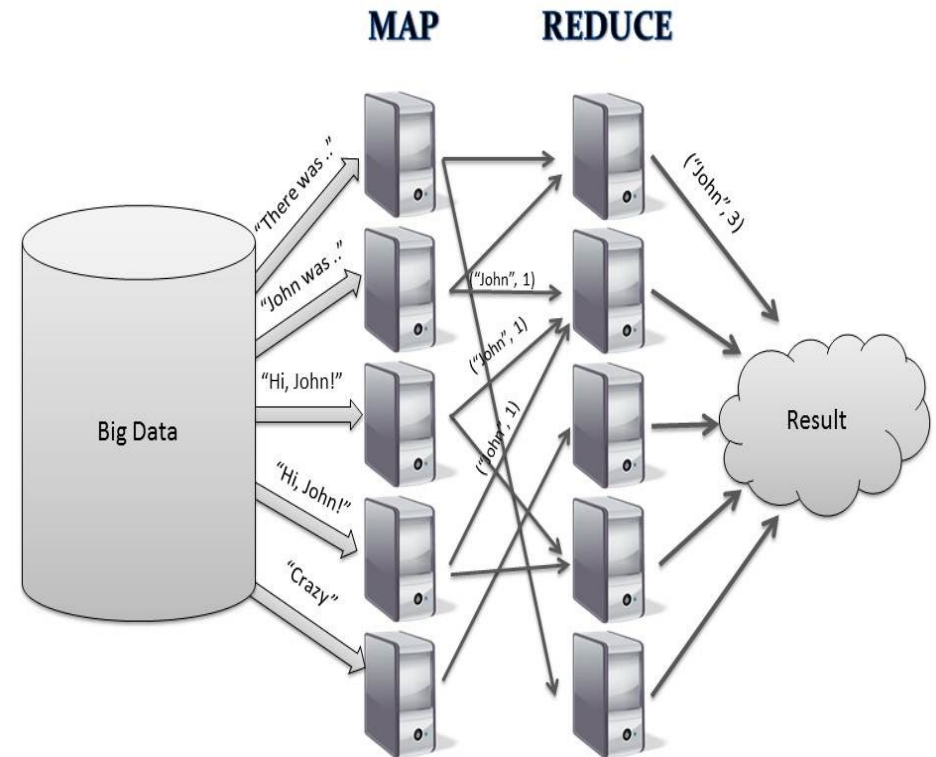
Video: Michael Stonebraker’s ICDE 2015 talk about his “10 Year Test of Time” Paper



By Edward Achziger 3/14/16

# Main Idea of MapReduce Paper

- MapReduce is “a programming model and an associated implementation for processing and generating large data sets”
- MR’s goal is to make processing large clusters of data on parallel and distributed systems simpler, more reliable in regards to system failures and data loss, and more efficient on network bandwidth use
- Examples of MR computations include: Counting # of Occurrences, Distributed Grep, Count of URL access frequency, Reverse Web-Link Graph, Term-Vector per Host, Inverted Index, Distributed Sort
- It was realized that most data computations involved “a map operation to each logical ‘record’ in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key, in order to combine the derived data appropriately”
- MapReduce programming model has been successfully used at Google for many different purposes



# How MapReduce is Implemented

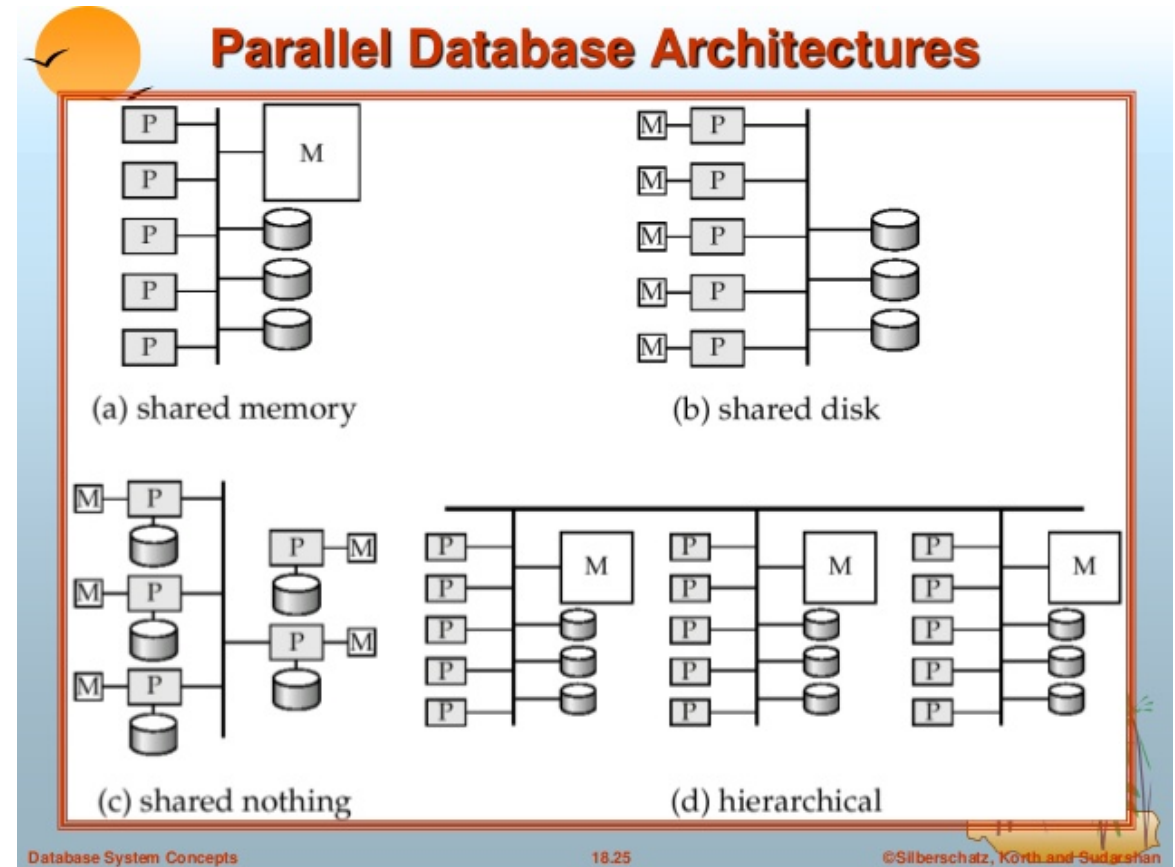
- MapReduce can be implemented many different ways depending on the computing environment of the parallel and distributed systems, whether it be shared-memory, shared disk, or shared none memory
- The most common implementation and the one used at Google is the shared nothing system, in which large clusters of commodity PC's all have they own disk and memory processor and are connected sharing data on the network bandwidth
- The simple thing with MR is the program handles this interaction between parallel systems on its own and all users have to do is define the map and reduce commands.
- In a simple overview of the execution, an input file is sent to the User program, that will then break that data into M splits, or clusters of data that will be copied multiple times across the many systems
- The master, then assigns work to the computers, either Map work (the Map function defined by user) or Reduce work (the reduce function made by the user)
- The Map work makes key/value pairs and goes through them making intermediate key/value pairs, which then sit on the memory until they are picked up by a Reduce worker
- The Reduce work picks up the intermediate key/value pair and performs its function, which is then grouped together with other reduce outputs on a final output file that is send back to the user once all tasks are complete
- The master keeps several data structures, stores many copies of key/value pairs along with the state of its completion, which helps lessen the percent times of failure and loss of data

# Analysis on the Idea and Implementation

- Personally, I like the idea of MapReduce because of its simplicity and lack of complicated data structure
- After reading the performance results I was a little taken back by the overall efficiency of computation and its speed when processing data
- I think that its overall user friendly experience is what is making it so popular now, as the field of data analysis is still in its growing stages but I would if it's a model that can with stand the test of time
- Definitely for sorting large amounts of unstructured data inputs, for example large lists or large frequencies, this program probably is the best to use right now
- A main problem with the program with potential arise from the thing that makes it great, because it is so simply and unstructured, it can be hard to pass information from a MR program to one user to the next, as everyone will use their own techniques and there will be no universality to it

# Main ideas of the Comparison Paper

- Compare and contrast the MapReduce parallel model to other types of parallel SQL database systems
- Parallel database systems have been around for 20 plus years and over a dozen are now offered on the market
- A main theme of discussion is that “MR model is well suited for development environments with a small number of programmers and a limited application domain, but its lack of constraints may not be appropriate for longer-term and larger-sized projects”
- On most tests involving speed and efficiency, the Vertica and DBMS-X models significantly out performed MapReduce



# How those Ideas are Implemented

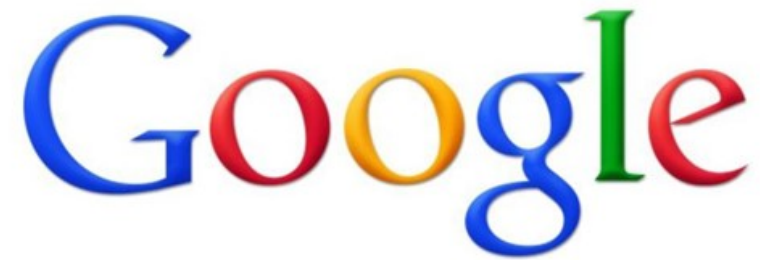
- The paper talks about 7 main points on how the implementation of the two systems are different
- The first was schema support, in which it described the problems MR has with data integrity as there is no defined schema for the data inputs and information added into the systems has no constraints
- The next was indexing, which talks about the use of views in the modern DBMSs and how difficult a task like that can be accomplished in MR
- The next was how data is called, whether it be presenting an algorithm which MR uses, or stating what you want, which the relational model supports
- The others go to concepts of data distribution, execution strategy, the flexibility among different users, and fault tolerance

# Analysis of the Comparison Paper

- I thought the paper made a lot of strong points why MR might be a problem for long term data analysis structure
- While everyone is worried about how easy we can analyze data and get it into the hands of the people making business or analytical decisions, it is important to remember why we have database models, which is to give data context which in return turns it into information
- Relating it to my education in finance, there is no such thing as “Too Big to Fail” and we must remember this when dealing with Big Data
- While I’m more personally interested in the MR program to analyze the data I would want to look at, like long lists of stock prices or sports statistics, I think the paper presents some strong points on why MR is not yet the end all be all

# Comparison of Paper 1 and Paper 2

- Both papers presented interesting points, with the MapReduce having the Google bias and the Comparison paper having a bias towards relational parallel models
- Like previously stated, I thought Paper 2 brought up some strong points to the things that we have learned in class, for example universal use among different users and data integrity

The Google logo, featuring the word "Google" in its characteristic multi-colored font (blue, red, yellow, blue, green, red).

vs.

**THE WORLD**



# Main Ideas of Stonebreaker Presentation

- Relational Databases are not a “one size fit all model” and perhaps now a “one size fits none”
- Slowly relational databases are not becoming good at anything due to the fact that traditional row stores have no place in the future and he provides 6 main reasons why they are being replaced (data warehousing, rise in NoSQL market, and lack of complex analytics skills just to name a few)
- Sees the future of databases being centered around the consumer need of Non volatile RAM, bigger main memory, higher processor diversity, higher speed networks, LLVM, and vectorization
- Innovators dilemma will be the big challenge facing the legacy vendors like Oracle, but they have slowly been moving away

# The Big Conclusion

- Main take away is that unlike Lord of the Rings, there is “not one ring (or database model) to rule them all” anymore
- MapReduce has some advantages but lacks areas in speed, database schema, and overall computer efficiency
- Parallel Database models may be faster and stay more true to the relational database rules but lack the simplicity and system failures can be just as costly
- Relational Database models are slowly becoming a thing of the past and has helped us learn a lot about how data should work together
- In conclusion, it will be exciting to track the progress of database science for years to come as new developments are rising frequently
- In my opinion, I believe programs like Hive and Pig, that work on top of the MapReduce face will be the next big phase to see if there can be away to add the structure we love about relational database to the simplicity of the MR