# Mapreduce: Simplified Data Processing on Large Clusters, A Comparison of Approaches to Large-Scale Data Analysis

JASON HIPOLITO

OCTOBER 30, 2017

# MAPREDUCE – MAIN IDEA

- A program that is run through hundreds/thousands of machines to reduce time spent on sorting through big data
- usually “MapReduce commutates processes many terabytes of data on thousands of machines”
- Most contributions are a simple and powerful interface that allows automatic parallelization and distribution of large-scale computations
- Large computations are split up and Combined with an interface that achieves high performance on large clusters of PCs
- Since the MapReduce is meant to use a lot of machines to process a lot of data, it must he able to deal with failures without breaking all other processes

# HOW THE IDEA IS IMPLEMENTED

- Different implementations of MapReduce are used depending on the size of the PC/network
- There are: small shared-memory machine, large NUMA multi-processor, and larger collection of networked machines
- This is how google has implemented Map Reduce
  - Machines must have dual-processor x86 processors running Linux, with 2-4 GB of memory
  - While using Commodity networking hardware, either 100mb/s or 1gb/s at the machine level
  - Clusters consist of hundreds to thousands of machines so its more prone to fail with some machines
  - They provide inexpensive IDE disks that go directly into the individual machines
  - Users then submit jobs based on a scheduling system ones the task the machine was assigned is done

# MY ANALYSIS OF THE IDEA/IMPLEMENTATION

- This is important because it reduces the amount of time it takes to preform these long processes
- that way the process can be finished in a reasonable amount of time
- More processes can be started because more are being completed
- Even though this can be very useful to preform large processes it can turn out bad if the failsafe is not implemented properly
- you don’t want a machine that failed its task to effect other machines that are doing other tasks

# COMPARISON PAPER MAIN IDEAS

- Compares the performance and development complexity of both MapReduce and the Database management system
- Map reduce is popular because it provides a simple model that users can create somewhat sophisticated distributed programs
- IBM and Google want to make a 1000 processor MapReduce cluster that will teach students distributed programming
- The MR and DBMS are different in a few ways
  - DBMS – requires all data to follow a well-defined schema, MR – works with any format
  - Indexing and comparison optimization
  - Program models
  - Way data is distributed
  - Query execution strategies

# COMPARISON PAPER IMPLEMENTATION

- MR model is better for development environments with a small number of programmers and a limited application domain
- DBMS requires data to fit into the relational paradigm of rows and columns
- MR model does not require that data files follows a schema defined by the relational data model, the programmer is free to structure the data however they want, even no structure if the feel like it
- SQL needs the programmer to specify the “shape” of the data in a data definition facility.
- The MR programmer must write a custom parser so the appropriate semantics for their input records can be derived

# MY OPINION ON THE COMPARISON

- I feel that the DBMS and the MR both have there own advantages and disadvantages and should be used in different situations
- As said before, for a smaller team the DBMS would be more beneficial to use
- Where if there was a big team the MR would be better because there is no set schema that needs to be followed

# COMPARISON OF THE IDEAS AND IMPLEMENTATIONS OF THE TWO PAPERS

- MapReduce
  - Goes into detail about how this model is implemented and the advantages of using it
  - Talks about how there are many ways to implement MapReduce
- Comparison Paper
  - Talks about both MapReduce and DBMS pointing out the advantages of both of them in certain situations
  - Talks about how both MR and DBMS are implemented and some tedious things that can be avoided by using one over the other

# STONEBRAKER'S MAIN IDEAS

- No DBMS will be "one size fits all"
- "One size fits none"
- SQL server was good for everything but now thinks they are good for nothing
- Column stores are faster then row stores when talking about market warehouses
- Complex analytics using regression, eigenvectors, SVD, data clustering and predictive models all use arrays instead of tables
  - Can simulate in SQL but it is a lot slower
  - Can CAST tables to arrays/implement an array DBMS, but these are inconvenient
- Streaming market is not based on traditional row stores, OLTP seem to have a greater market share
- Graph analytics simulate a column store (Facebook uses this), so do array engines

# ADVANTAGES/DISADVANTAGES OF MAPREDUCE COMPARISON & STONEBRAKER TALK

- Advantages
  - There are many option od database models and engines that can be used for all types of different tasks.
  - Both parallel database systems displayed a significant performance advantage over Hadoop MR
- Disadvantages
  - Can take time to adapt to these different methods used
  - The more processors used to manage data, more energy will be used as seen in the comparison with the 100 and 1000 node test
  - MapReduce model with multi-thousand node clusters is a brute force solution that is not very energy efficient