# MapReduce: Simplified Data Processing on Large Clusters

# A Comparison of Approaches to Large-Scale Data Analysis

# One Size Fits All – An Idea Whose Time Has Come and Gone (2005)

BY THOMAS FAMULARO

10/20/2016

# MAPREDUCE – MAIN IDEA

- MapReduce is a model for processing and generating large data sets
- It is written to run on a cluster of many machines
- The Map refers to an input pair which produces the intermediate key/value pairs and then those are Reduced to zero or one output value
- It is designed to process data that is way to big for one computer to process
- It does this by creating a master worker relation ships with other computers

# MAPREDUCE – IMPLEMENTATION

- Implementation starts with the user creating the input and reduce functions
- Then a master computer is chosen, and it divides up the work to give to the worker computers.
- The goes to the workers during the map phase where the data is processed
- The intermediate data is then moved on to the reduce phase where the data is combined back
- Finally the data is out put to zero or one files

# MAPREDUCE - ANALYSIS

- The idea is simple break up the work into a lot of small parts so that many different computers can work on it
- But implementation is not simple, there is a lot decisions that have to be made
- Map reduce created a solution that lets the user have a lot of control over the part that the user cares about and the automation of the parts they don't

# LARGE-SCALE DATA ANALYSIS – MAIN IDEAS

- Why is MapReduce such a radical idea why not just use parallel DBMS’s
- Parallel DBMS’s are much more efficient, 3.1 to 6.5 times more efficient on a scale of 100 nodes
- As you get to more and more nodes MapReduce becomes more efficient but not many pieces of data are that big yet.

# LARGE-SCALE DATA ANALYSIS – IMPLEMENTATION

- When using MapReduce and a different programmer has to know how the whole function is written to be able to use the data
- When using DBMS's the data is stored on a separate schema which is easier to access
- MapReduce proposes the idea that it has no rigid structure of data fields, programmers at some point do have to agree on a format for their data for it to be useful.
- In MapReduce it is agreed upon outside of the program, in DBMS's its is inforced inside of the program

# LARGE-SCALE DATA ANALYSIS - ANALYSIS

- The programming model of presenting an algorithm for data access is typical of a DBMS system and seems to be the preferred way to ask for information, and seems to be the efficient.
- Hadoop seems to be more efficient on the computers no matter how many nodes are in the system for loading data but not doing calculations on data in any way.

# MAPREDUCE VS. LARGE-SCALE DATA ANALYSIS

- MapReduce according to the first paper seemed like a great new way to process data that will let us use all of our equipment so much better
- But the second paper introduced DMBSs and how they have been around for ever and out preform MapReduce on any realistic level at the current time.
- When our data gets so big that we need 1000 computers in a cluster to be able to handle it then at that point MapReduce will have the advantage.

# STONEBRAKER TALK – MAIN IDEAS

- One data processing system is not going to meet all the needs of the world
- Making relational databases universal is not possible
- Column store is 2 orders of magnitude faster than row store
- The big data base companies are not going to adapt to the new system, but if anyone will it will be SQLServer

# STONEBRAKER - ADVANTAGES & DISADVANTAGES

- Data Warehouse Market – Colum stores are 2 magnitudes faster than column stores
- OLTP Market – All data in main memory lightweight SQL is necessary
- NoSQL Market – No standards what so ever but none look like traditional row stores
- Complex Analytics – Data scientist are going to replace business analysisists
- Streaming Market – adding streaming to a OLTP engine much easier than adding persistence to a streaming engine