MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

A Comparison of Approaches to Large-Scale Data Analysis

Pavlo, Paulson, Rasin, Abadi, DeWitt, Madden, Stonebraker

Jason Wong

May 9 2014

MapReduce: Overview

- A system for processing large data sets
- Designed and widely used at Google
- Runs on a large cluster of relatively weak CPUs
- Very efficient and fault-tolerant
- Hides complexity of distributed computing from programmers

MapReduce: Implementation

- Start with key-value pairs of data in a large file
- Write programs with two functions: map and reduce
  - map :: k -> v -> [(k', v')]
    - produces a list of key-value pair results, where keys are not unique
  - reduce :: k' -> [v'] -> [v']
    - merges values with the same keys
- Use a master machine to split the file up and distribute data evenly among workers that run map, then tell other workers to run reduce on the output

MapReduce: It's awesome!

- Map/Reduce format allows programmers that have not done any sort of parallel computing to easily utilize it
- Logic handling the distribution of data and fault recovery is separate from program logic
- Does not require hardware that is powerful or reliable, or a lot of network bandwidth

Comparison Paper

- Any parallel processing task can be written in either MapReduce or a set of database queries
- Filter records in a table, and send them to other tables. Those tables calculate aggregates on their data, then the aggregates are merged together
- Benchmarks were run comparing the performance of Hadoop, an open-source implementation of MapReduce, to two different parallel database management systems: DBMS-X and Vertica

MapReduce: Not so awesome?

Advantages

- More flexible/extensible
- Very fault-tolerant
- Easier to set up - less tuning required
- Easier to write programs initially

Disadvantages

- Parallel DBMS are faster by about **half an order of magnitude**
- Less storage and bandwidth required too
- Easier to maintain programs