Morgan Stippa

3/9/16

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

A Comparison of approaches to Large-Scale Data Analysis

Andrew Pavlo, Erik Paulson, Daniel Abadi, David DeWitt, Sameul Madden, Alexander Rasin, Michael Stonebraker

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

Michael Stonebraker
Ugur Cetintemel

MapReduce

- Programming model implemented to process and generate large data sets. Has two functions:
  - Map function takes a set of data and generates an intermediate set of data. Data is broken down into key/value pairs
  - Reduce function combines output of a map into a smaller set of rows using the same intermediate key
- Allows programs to be automatically parallelized and executed on a large cluster of machines
- Model hides complexities of parallelization, fault-tolerance, data distribution and load balancing in a library
- Functional model with map and reduce functions lets the user:
  - Parallelize large computations easily
  - Use re-execution as the primary mechanism for fault tolerance

Implementation

- Different implementations are possible – it depends on the environment
  - Large collection of networked machines
  - Small shared-memory machine
- Google implements in a computing environment of large clusters of commodity PCs connected by switched Ethernet
  - Cluster consists of hundreds or thousands of machines
  - Scheduling system manages jobs that users submit and a is mapped to set of available machines within a cluster by a scheduler
    - Jobs consist of a set of tasks
  - Input data is partitioned into a set of splits which allows Map invocations to be distributed across numerous machines
    - Different machines will process input machines in parallel
  - A partition function is used to distribute Reduce invocations by partitioning the intermediate key space into pieces

Analysis

- Model has been successful because it hides complexities of parallelization, fault-tolerance, locality optimization, and load balancing
  - Locality optimization saves network bandwidth because data gets read from local disks and a single copy of the intermediate data is written on a local disk
- Many problems involving data can be easily accomplished using MapReduce
  - Data Mining
  - Data Sorting
  - Machine Learning
- Can be implemented for different environments which makes it scalable
  - Can be efficiently used on thousands of machines which makes it suitable for many companies facing large computational problems
    - Problems with machines can be a common occurrence when thousands of machines are being used, but redundant execution is used to reduce problems of machine failures

Large-Scale Data Analysis

- Hadoop MapReduce and DBMSs get evaluated to see their performance abilities and development complexity
- Goal of the paper was to figure out where each system was different and what caused the differences in performance
- DBMSs took a lot longer than MapReduce to process data and prepare for execution of parallel
- DBMSs were consistently and significantly faster than MapReduce, ranging from a factor of 3.1 to 6.5
  - Also required less code to implement each task
- Growing popularity of MapReduce can be credited to its simple model that hides the complex distributed programs that users express

Implementation

- Authors ran 5 analytic tasks on a cluster of 100 nodes to measure system's performance
  - Data Loading: Hadoop outperforms the DBMSs because data is loaded into Hadoop using only a single replica per block
  - Selection Task: The DBMSs outperformed the Hadoop MR across all cluster scaling levels due to Hadoop's increased start up costs as more machines are added
  - Aggregation Task: The DBMSs outperform Hadoop – they executed the queries by having each node scan its local table, extract the sourceIP and adRevenue fields, and perform a local group by
  - Join Task: Hadoop is much slower in the task because it is limited by the speed at which the large table can be read of the disk and the DBMSs were able to take advantage of join key that partitioned two tables
  - UDF Aggregation Task: Systems perform about the same because each machine has the same data to process and the amount stays the same as more machines are added

Analysis

- Hadoop MapReduce did not perform as well as the DBMSs but its popularity comes from:
  - Less expensive upfront cost
  - Easier to set up and use compared to the DBMSs
  - Much better job of minimizing the amount of work that is lost when there is a hardware failure
- Problems for MapReduce to fix:
  - Lack of powerful analytic functions
  - Lack of schema
    - Compression is less valuabe
    - A reason why MapReduce get outperformed by DBMS
    - Schema contains important information which is critical in optimizing queries
  - Complex tasks are difficult to code in the MapReduce style
    - Higher level interfaces are fixing this
  - Model's brute force solution wastes too much energy

Comparison

- Both agree that the main reason MapReduce is popular is because its model is easier to understand and there is less of a learning curve
- Hadoop was much faster loading data then the DBMS systems which left Hadoop with a lot of time to execute queries before the other two systems were done loading the data. The authors of the paper did not mention this in the conclusion
- First paper did not address the issue about the amount of energy that gets wasted
  - DBMSs systems that provide the same response time as MapReduce model use far less machines
- MapReduce is new and has a lot of room to improve while DBMSs have been using the same framework for 20 years

Stonebraker

- In the 80s and 90s he believed that “one size fits all” in the field of DBMS
- That led him to polishing up the relational model – thinking it could be used for any case
- He went through most of the major markets to show that there’s a huge diversity of engines each oriented toward a specific application
  - So one size could possibly not fill all
- Believes that there is no longer going to be a market for row stores because evidence shows that column stores faster
- We’re starting to see the transition from old implementations of DBMS to new implementations
- Many new ideas are being implemented today

Final Comparison

- Advantages of MapReduce:
  - Simple model that hides the complexities of distributed programs
  - Much faster than DBMSs in loading data
  - Compared to DBMSs, less work will be lost if there is a hardware failure
  - Completely different paradigm from the RDBMS
- Disadvantages of MapReduce:
  - DBMSs systems outperform MapReduce in large-scale data analysis
  - Not suited for every task
  - Lack of Schema
  - MapReduce style systems can make complex tasks hard to code
  - SQL take much less code on tasks