Big Data Paper Summary: “Pregel: A System for Large Scale Graph Processing” & “A Comparison of Approaches to Large-Scale Data Analysis

Michael Read

October 20, 2016

Pregel Paper:

http://labouseur.com/courses/db/papers/malewicz.sigmod10.pegel.pdf

Comparison Paper:

http://labouseur.com/courses/db/papers/pavlo.sigmod09.comparison.pdf

Stonebreaker Talk:

http://kdb.snu.ac.kr/data/stonebraker_talk.mp4

Main idea of Pregel Paper

- Create a scalable, fault-tolerant platform flexible enough to use for arbitrary graph algorithms
- Have the graph database synchronous, by using supersteps, to make it easier to implement algorithms and make it better for distributed systems
- Making balancing the workloads on the machines easier, to help remove excessive latency between supersteps
- Contains a pure message passing model, meaning there is no way to read from a remote machine, or share memory, which is better for performance

Implementation of Pregel Paper idea

- Each vertex is uniquely identified by a string value, called a vertex identifier
- Each edge has a modifiable, user defined value
- After the initial inputs, initializing the graph and defining the algorithms, the computation continues in a sequence of supersteps until every vertex is inactive
- Vertices start active, and only become inactive when they send themselves a “vote to halt” message
- If an inactive vertex receives a message, it becomes an active vertex until it sends another “vote to halt” message
- Designed for Google cluster architecture, which consists of thousands of PCs in racks with high intra-rack bandwidth
- Persistent data is stored on a distributed storage system, Google File System, or in Bigtable, while temporary data is stored on local disk

Analysis of Pregel Paper idea and implementation

- Pregel system seems to be very good for relatively small graphs, or for larger graphs with simple algorithms
  - The more complex the algorithm, and the more vertices are added, the more computations will be required
- Would work very well with computer clusters
  - The more computations that can be performed per second, the faster each superstep can occur, meaning the faster the algorithms can be resolved
- Synchronicity makes the analysis of the state of the graph along each superstep very simple
  - Since there is no order to any operation or message which happens during a particular superstep, the state of the graph can be very easily defined based on the superstep

Main ideas of comparison paper

- Parallel processing is becoming much more dominant in the data processing field
  - Replacing small numbers of high-end processors with large numbers of cheaper, lower-end processors processing in parallel
- MapReduce and Parallel DBMSs both divide the input data into partitions to allow for faster processing by processing each partition in parallel
- MapReduce does not require the data to be structured in any particular way, while Parallel DBMSs need structured data
- Parallel DBMSs allow for simpler querying by using a high-level language, removing the need to think about how to join the data, and how to index the data

Implementation of comparison paper ideas

- MapReduce uses two different functions, Map and Reduce, to process the data
  - The map function takes the input data and does some preliminary processing, then outputting the data into multiple intermediate records as key/value pairs, separated by a “split” function based on the keys
  - The Reduce function takes the intermediate records, and processes them or combines them in some way, and then writes them to an output file, which makes up part of the final output
- Parallel DBMSs distribute the data for a table on multiple nodes based on some attribute, and then processes the query in parallel on each node

Analysis of comparison paper ideas and implementation

- Both MapReduce and Parallel DBMSs seem very powerful in their parallel processing abilities
- MapReduce seems better for smaller datasets, or datasets that are harder to put into a defined structure, whereas Parallel DBMSs seem to be the better choice for larger, more defined datasets
- Parallel DBMSs are much faster to create queries for, as queries can be written in high level languages like SQL
- MapReduce seems to be much better for running the same algorithms on multiple datasets, as once the algorithms are written, the turnaround on running them would be faster than on the Parallel DBMSs

Comparison of ideas and implementations between the two papers

- Pregel, as a graph database model, has more defined states between the input and output than either MapReduce or Parallel DBMSs, with the usage of supersteps to differentiate between different states as the graph evolves
- Parallel DBMSs are the simplest of the three to query, as being able to write in a high-level language makes generating queries extremely easy
- MapReduce seems to be the best way of taking a small dataset and getting an answer quickly, as little to no structure is required to use MapReduce

Main points of Stonebreaker Talk

- Legacy database systems such as Oracle and DB2, which use traditional row-store methods, are not optimal for many, if any, modern markets
- Column-stores work much more quickly than traditional row-stores
- Smaller databases for things like online transaction processing are much more efficient when stored entirely on main memory, for which legacy systems are much too heavyweight to implement
- NoSQL market is going strong with many options to chose from
- Complex Analytics market will end up using complex functions (i.e. regressions, eigenvectors, predictive models, etc.) to analyze data, which are all defined on arrays and not tables
- Graph Analytics can use simulations in column-stores and array engines, or utilize graph engines, not row-stores

Advantages/disadvantages of Pregel paper ideas

- As relational database models are falling out of use, and parallel processing is rising in popularity, the Pregel database model is becoming an increasingly more popular choice
- Pregel is good with distributed graphs, and has very defined states with the use of supersteps, making it easy to trace the algorithms as they progress
- Pregel has the downside of not being able to utilize shared memory or read from a remote machine, meaning there is no way to access data not stored on the machines running the graph database