Pregel: A system for Large-Scale Graph Processing by: Grzegorz Malewicz

A comparison of Approaches to Large-Scale Data analysis by: Andrew Pavlo

StoneBraker Talk by: Michael Stonebraker

Carlie Maxwell 3/14/16

Main Ideas of Pregel

- There was no computational model that was able to process extremely large graphs until the Pregel program was created, which was modeled after the Bulk Synchronous Parallel model
- The model is known as "vertex-centric," meaning that the graph is broken down into vertices, which perform supersteps, or iterations, allowing each vertex to perform a user specified function and then pass/receive messages along to other vertices using the compute method
  - The model has no remote reads or shared memory
  - The model will eventually stop when all vertices are inactive, which happens when there are no longer any messages to be sent
- Pregel is implemented in day to day use through the use of several workstations with one of the different algorithms developed by Pregel that could potentially be used, such as Shortest Paths
  - ShortestPaths is used to find the single shortest path between vertices
- After plotting the program onto graphs, the information doesn't accurately demonstrate the efficiency of the program, but instead shows that you don't need to code a lot in order to have a decent performance
- Overall, the program is highly efficient, flexible, and uses a "fault-tolerant implementation" in order to limit errors

Implementation

- There is a cluster of machines, in which one one of the copies of the user program is the master, where all messages are sent to
- The Master determines how many sets of vertices there are and gives at least one to each computer – each computer is responsible for executing the compute function on each vertice in the set → which processes messages sent to that vertex and then sends other messages out to other vertices
- The Master gives some user input to each worker (the input is a number of vertices and edges which are then marked as active)
- The Master tells all computers to do a superstep while executing compute in order to send updated messages to the master
- The workers will then save their parts and it will be compiled after
- Aggregators are used for global monitoring and data → The resulting values are made available to all of the vertices = sum, min, max
- In order to have no errors, failures are issued through ping messages
  - If you don't get a ping message, then your program will shut down and the master has to reassign their work load

My analysis of Pregel

- I believe the Pregel program is highly flexible and useful for the task at hand (processing large graphs such as social networks)
- Since Pregel uses messages and iterations, it doesn't use shared memory or remote reads which makes it more efficient
- Pregel is different than most programs because rather than using only user input and output, its able to modify the local state of the graph and constantly update it quickly

Main ideas of the Comparison Paper

- This paper consists of comparisons amongst the traditional DBMS vs. the new MapReduce paradigm for the analysis of large-scale data and how they ultimately match up against each other
- The comparison between how the systems should be used, which is later elaborated upon to find more complex uses. Initial uses consist of things such as for large scale projects (DBMS) vs small scale (MR) and what environment each system is appropriate for, such as development or corporate.
- The differences between the structure of the systems – MR is more flexible/less rigid structure, while DBMS has a single schema
- The comparison of different systems operating assumptions and the way in which we interpret them to make one experiment – important that the graphs showed that the DBMS systems, such as Vertica, were faster each time
- Lastly, the article describes the difference between how the design of the systems are crucial to the overall performance

Implementation of MR

- MR
  - Map
    - It reads records from an input file, sorts/filters them, and outputs intermediate records as new key values
    - There is a split function run on the records as they are being outputted, which assigns them a hash value; if you have the same hash value you will be grouped together
  - Reduce
    - It processes the values that were grouped together and assigned to that specific node in order to create a final output record
  - MR is used mainly for small scale projects and development environments
  - MR doesn't have a set/defined schema, causing vast flexibility
  - In order to prevent errors/failure, the MR user can restart the task on a separate node
  - Installation requires simply setting up directories, libraries, and configuration files

Implementation of DBMS

- DBMS
  - It runs on clusters of shared nodes
  - It uses parallel execution by having tables that are partitioned over the nodes in a cluster and the system can translate SQL into a usable query that can be executed among various nodes
  - It is similar to a relational database in the sense that it uses rows and columns; this creates a set structure
  - It uses hashes and b-tree indexes in order to search through the data more efficiently
  - It's essentially written in english so there is not a strong correlation between the language and the instructions
  - Large transactions are restarted in the case of failure rather than recovered on a separate node, making this worse than the MR for this aspect
  - They are always up and running, simply waiting for a query to execute on multiple threads, allowing the DBMS to multitask and continue on your other code, rather than waiting for it to constantly start up when there's a new query

Analysis of Comparison Paper

- In each section, the MR model seems to be viewed down upon in regards to the DBMS model.
- The one main thing that the MR model has going for it was it's extreme flexibility and easy implementation, but as we learned earlier, too much flexibility can be dangerous. Additionally, the DBMS has drastically improved over the years to be flexible enough to compete, while maintaining its rigid structure to give it an edge.
- Since the MR model has no structure, it ultimately creates more work for the programmer in the end, as they usually have to design their own support methods for things such as a custom parser and indexes
  - The programmers then have to decide on a single schema that they actually want to use, which creates conflict, which again has to be approved by another set of programmers to fit all constraints, which is a lot of overhead.
- SQL, which is used in DBMS, has been proven to be easier to write/modify/understand, making it overall more efficient for both the programmer and the user
- The biggest complaints of the DBMS system is that it was hard to initially configure and that is doesn't recover data in the middle of a query execution – I think in retrospect, the other features outweigh this issue, making the DBMS model ultimately superior from things I read outlined in this article

Comparison of the ideas/ implementations of 2 papers

- Both papers deal with how a system processes large amounts data
- In the first paper, it deals with how a model is able to break down large graphs such as social networks in order to efficiently process them
- The second paper deals with 2 systems that analyze large scale data, one being the traditional way called DBMS and the other being the newer version called MapReduce that ultimately fell flat in my opinion
- The first paper implements the idea of the pregel system by breaking down the large graph, ultimately assigning it to many workstations, into a series of vertices, which receive and send out messages individually, rather than as a whole
- The comparison paper implements the 2 systems differently. The MapReduce model is extremely flexible and has no schema, but sorts through records and groups them, which are then counted by the reduce function. The DBMS has a structured schema, that uses SQL, in a similar manor of a relational database where it consists of rows and columns. The DBMS translates sql into a query and executes it among various nodes

Main ideas of Video

- In 2005, they believed that making relational DBMS was the answer to any question by making them broad in scope, abstract data types, referential integrity, and triggers
  - Back then, DB2 and Oracle were considered to be not good for everything, whereas are now, they are good for nothing
  - In 2015, they realized that “one size fit nothing”
- He explained the differences between data warehouses, OLTP market, NoSQL market, complex analytics, streaming market, and graph analytics market and why they ultimately failed
  - In the end, they were seen as various engines that are geared towards certain applications, meaning that the traditional rows had no market share creating “one size fits none”
- Now, there needs to be new idea techniques offered such as NVRAM, big main memory and vectorization
- The DMBS research didn’t work in the 80’s and 90’s because they believed they simply had to redo the relational model, which wasn’t the case; now all that is left are the new ideas that can be implemented in the future

Advantages/disadvantages

- The comparison paper would see Pregel as a good system, as it sees the DBMS as the dominant model, which has a rigid structure
- Pregel, in my opinion, is extremely structured, as each vertex follows a complete series of instructions. It runs its own method, called compute, that allows it to be able to first receive and process messages and then send out messages to the next vertex until all nodes are eventually deactivated, meaning no more messages are being sent out
- I believe the video would see Pregel as a good system. When he discusses the major markets, he claims that the traditional row store would not make the cut, which Pregel is not. Pregel is innovative and new because it stores the data in a series of messages that move amongst the vertices.