Nicholas Carmello

Map reduce: Simplified Data processing on Large Clusters.

By : **jeffrey dean and sanjay Ghemawat**

Comparison of Approaches to Large-Scale Data Analysis

By **: Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker**

10/24/2020

Main idea

The main idea of the mapReduce paper is Mapreduce is a programming model to take input key and value pairs. Much like a dictionary. The Map Function computes a set of output keys and value pairs. This is the mapping part. The reduce part is they group all of the values that have the same key and combines the values together to form a smaller value. This is beneficial to people because it we can shrink the size down of large memory to small memory. Some mapReduce programs are Distributed Grep, Count of URL Access frequency, reverse Web-Link Graph, Term-Vector per host. There are many ways to speed up processes such as backup tasks and sorting. Some records could be bad records so there is a need to skip over it until the next task. MapReduce is used for data mining, web crawling, machine learning and more. It is the future of the internet for huge amounts of data where we need fast interpretation.

How these Ideas are implemented

There are many ways to implement map reduce and it all depends on the machines being used such as small memory machine or a large memory machine. Many factors come into play.

The main idea is to have many computers connected that process the data. The computers splits the program into many input files, distributes the files across the machines and assigns a master to one of the files that will overlook all of the other files. The master assigns the files either a map job or a reduce job.

The map workers reads the split file and produces a key and value pair which is passed to the map function.

The Master then finds where the the key and value pair are stored and passing them to the reduce function. These workers group all of the values together with the same key. Then it finally appends every key and value to the output file.

My analysis of the First paper

I thought learning about how mapreduce changed the google ad-hoc system is pretty incredible among the other things it does. It sped up the processes and made the code easier to understand. I've read about how the computers are chained together through a switch with low processors. I was wondering why this couldn't be down with higher processors. It's pretty intelligent that the master controls the whole system and operates how every task works. It can assign a job to workers and also reassign if one of those workers fail on a node. Overall the performance of mapreduce was pretty good. Most of the tasks finished before 15000 seconds. The 2 exceptions were top left and top middle tests.

I love the idea of when a task fails, the master can just assign a task to another worker. If it is a map worker that fails, the reduce workers are sent a message that the map worker failed. They then know the new map worker. I thought this idea was very smart because no data is being left behind but found a way to overcome this problem.

Main idea of comparison paper

The main idea for the comparison paper was to compare mapreduce to database in terms of speed and performance. In the tests they tested how fast the data loaded, how long it took for the data to be executed and how long the data took to give a result. Overall, they found out that the parallel databases are faster than the hadoop map reduce method. The paper said, “ DBMS-X was 3.2 times faster than MR and Vertica was 2.3 times faster than DBMS-X”. DBMS-X and vertica were the 2 databases that were tested against the hadoop mapreduce.

They believe that the hadoop mapreduce method would be faster if there were more nodes to be tested. The map reduce method is easier to set-up than the database because there is no set structure in the mapreduce whereas in a database there is tables of rows and columns. Data would have to be manually inserted into the databases. The vertica Database is only a database of columns and not of rows like DBMS-X. There are many technologies that make the Database systems faster such as b- tree indexing. Mapreduce does not have indexing already. Programmers would’ve had to manually make up their indexes and then distribute that structure to the other programmers to make the tests work properly.

How the ideas of the comparison paper are implemented

The ideas of comparison paper are implemented in multiple tasks. Each task would should how each of the systems would do under circumstances in terms of speed. The tasks are Grep Task, Join Task, aggregate task, and selection task. The number of nodes that were being tested on was 1 node, 10 nodes, 25 nodes, 50 nodes and 100 nodes.

Test one was the Grep task test. This would measure how long it would take for the data be loaded for the 3 systems. Hadoop mapreduce ended up outperforming vertica and DBMS-X.

Another task was the selection task which is to find the page Url in a table. MapReduce only need to use a map function for this test since the nodes were all unique. The databases outperform hadoop again because of the databases excellent indexing strategy. Hadoop has a slow startup time so it made the execution very slow.

My analysis of the ideas and implementation in the comparison paper

Overall, I believe that the mapreduce was slower because of the slow start up time. I believe if the nodes were above 1,000, the mapreduce would be significantly better in the results. The paper says that there is no need for 1,000 nodes because that is a enormous data scale but I believe one day that the data will be that big and mapreduce will be there to simplify the data. Databases have a huge advantage due to its incredible b-tree indexing strategy but I feel like hadoop weighs more in its favor due to its fault tolerance. Hadoop can just start up a task on a different node if one node fails. This could be completely helpful in the future with big data sets because databases would have to startup the whole query again. These seemed like fair tests. They are different in many ways so it's hard to really compare results.

Comparison of the ideas and implementations

The first paper and second paper were kind of similar in regards to mapreduce. Both of them talked about mapreduce heavily and gave some excellent points such as backing up tasks. A master will assign a task if a machine fails. They both performed tests on map reduce but the first paper used 1800 nodes while the second paper used 100 nodes at max. This is because 100 nodes is more than enough for everyday tasks for now. I believe the 1800 nodes is what accurate depicts mapReduce. MapReduce should be used for bigger tasks while database systems used for smaller tasks. The second paper was mainly comparing a database to mapReduce. The first and second paper are different by the tests they have done. The 2nd paper used selection task and aggregate task but the first paper only did grep and indexing. The 2nd paper almost compared mapreduce to databases. We found that overall databases were faster on smaller tasks.

The main idea of Stonebrakers talk

The main idea of Stonebreakers talk is that one size doesn't fit all. He says this is terms of relational databases. We are seeing a transition from relational databases to other databases such as streaming databases, c-store, and more. All of these new databases keep coming out with something new in them and its transforming the industry. Stonebraker realized this when he was working on a streaming dbms and C-store. A C-store is made up of column storing instead of row storing. Column stores are much faster than row stores. He goes on to explain some markets that are going to be useless soon such as data warehouses, OLTP, NOSQL market, and more. Complex analysis might transition to array stores. Streaming market is going to transition to s-store. Graph engines are used for places like facebook like finding mutual friends. There is going to be a huge diversity of engines and everything is moving away from the row stores. Exactly why there isn't a one size fits all.

Advantages and disadvantages of main paper

The advantages of the first paper is that mapreduce can be used for more data and speed up processes for bigger data. Another advantage is that relation databases are transitioning to different databases so we might see an influx in mapreduce if it were to be improved. Another advantage of mapreduce is that it isn't being compared to anything in the first paper. There is nothing to really compare how it does to other things. The 2nd paper gives a broader idea of the situation. Another advantage of the mapReduce paper is its fault tolerance. There is almost no fault tolerance.

Disadvantages for the main paper is that relational databases seem to be faster for now for smaller data.. Another disadvantage of the main paper is the paper showed how many seconds it would take if there was no backup tasks. This would be more than 1000 seconds. Another