The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlov, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker

One Size Fits All

Michael Stonebraker

CHRISTIAN ISOLDA

MARCH 7, 2017

Overview of The Google File System

- The Google File System is a scalable distributed file system for large distributed data intensive applications. It is an efficient system that includes fault tolerance, uses low price hardware and has the ability to be accessed by a large number of clients at once.
- Main needs of the system:
  - Component Failures: Be able to cope and deal with down hardware and constantly update the system to adapt the currently down hardware or hardware that doesn't come back up.
  - Handle large files: Be able to handle millions of files that are upwards of multiple GB's per file and be able to access them with ease for clients.
  - Use Inexpensive Hardware: Use inexpensive parts that fail but are budget worthy and utilize the failure detection system above to be able to use this kind of hardware for the system.

Implementing Google File System

- Google File System is divided into clusters that have different functions and context to allow for simplicity.
  - Master Servers, Chunk Servers and Clients are the main 3.
- **Master Server**: Having a single master makes the design and functionality simple. To make sure it does not bottleneck, the master never reads or writes, rather supplies the data to the client.
- **Chunk Server**: Store data in "chunks" or more specific 64MB files. It is standard to have 3 replicas of the data but can be altered in the process for more or less.
- **Client**: To keep flow and simplicity the client requests from the master for chunk location to either alter or read the requested chunk.

Analysis of Google File System

- GFS makes a very complicated and cast process into a simple and straight through process. I enjoy the simplicity and minimalistic style of the system.
  - Master: Having a master to handle the admin like operations rather than the client directly go in is an awesome idea and way to access and alter data.
  - Chunks: I enjoy the idea of storing data in chunks rather than full files or big files. Storing them at this size makes it easy and fast to access or change the data as well as store it. Additionally, it is more efficient to make copies of this data at this size rather than a bigger size.
  - Hardware: Being able to use hardware that isn't top notch or expensive makes implementation and usability achievable. It is also genius to couple the use of inexpensive hardware with a built in fault monitoring system to make sure the system stays up and running through faults.

Overview of A Comparison to Large-Scale Data Analysis

- Introduces the audience to the advantages and disadvantage of MapReduce and DBMS
- Discusses tested systems such as DMBS-X, Hadoop, and Vertica
- Talks about the advantages of being a programmer dealing with certain systems as well as usability and performance based on opinions of people that have used both first hand.
- Uses Hadoop as a main example of executing the MR framework and the ups and downs of this program.

Implementing the Comparison

- To compare the two systems benchmarks and tasks were used to measure the efficiency of completion and determining which is better in either scenario.
  - The tasks were repeated 3 times to get a good sample data for each task. To make sure the performance were not altered doing parallel tasks they first executed each task on a single node. Then executed the tasks on different cluster sizes to show the scaling as both the amount of data was processed and resources made available
- The main tasks that were measured was a selection task, aggregation task, join task, and a UDF aggregation task

Analysis of The Comparison

- After the completion of all the tasks the data and averages were explained in the paper:
  - Selection Task: Both DBMS systems outperformed Hadoop because of the startup time Hadoop has implemented.
  - Aggregation Task: Both DBMS systems outperformed Hadoop in this task.
  - Join Task: Going by time, DBMS beat Hadoop because of the limited speed that Hadoop has implemented which is 20GB/node.
  - UDF Aggregation Task: Hadoop beat DBMS in this task because of the row by row interaction between the UDF and input file stored out of the database.
- DBMS outperformed Hadoop in three out of the four tasks tested. It shows that DBMS even though it is older still outperforms a new breakthrough like Hadoop.

GFS vs. Large Scale Data Analysis

- The main comparison I had for the papers is just the structure and format in which they are written in. If you look at the overall look at both they look very similar with structure and format. On the other hand, the information in both of them are completely different. GFS focuses on one system and the functionality and how it works to a tee while the other focuses on the good and bad of multiple systems rather than just one.
- When it comes to becoming more knowledgeable, I would start with Large Scale Data Analysis paper being it has a lot of base knowledge on the topic. Once becoming acquainted with the topic and chosen GFS, I would read the GFS to learn more about the practicality of going into this field and how it actually works in a system highlighted such as GFS.

Overview of One Size Fits All

- Stonebraker was a believer that relational database is never going to be the "One Size Fits All" database.
- He speaks about how the future of data storage will be based on the idea on column stores replacing row stores.
- He also believes that the DBMS does not do anything or nearly as much as he wants or needs it do.
- He says the DBMS is "One Size Fits None" where there is no adaption or ability to change when dealing with different systems.

Comparison of GFS to LSDA and Stonebraker

- Advantages: GFS works insanely well with Google and its needs being Google designed it to their needs and wants it works in a simple and logical way so it is reliable and constantly accessible.
- Disadvantages: Comparing GFS to the other papers it has a main disadvantage which is that to be successful it has to be adaptable which GFS is not. GFS is tailored to what Google needs and how they function. Not every major company works how Google does. To be successful in this field you need to be adaptable and able to change at an instant.

- To wrap up, GFS is a very good baseline and structure for a system. Changing GFS would be hard because of the simplistic style it has but would be easy to add onto if need be. Being able to add onto is a start of being adaptable and can lead to a system that is widely used rather than just for Google.