The Google File System & A Comparison of Approaches to Large-Scale Data Analysis

By: Blaise Spinelli

The Google File System: *Main Ideas*

- The purpose of the Google File System article is to give the reader a extensive understanding of the file system's functions.
- The Google File System's interface extensions were designed to support allocated applications, aspects of the system's design, and reports of different measures which included 'micro-benchmarks' and real world practices of the file system were outlined.
- The Google File System was inspired by the demands of Google's data processing needs of their workload.
- The GFS is able to support large scale data while using clusters of inexpensive commodity hardware.

The Google File System: *Implementation*

- The GFS's files are hierarchically organized within the repositories and classified by path name.
- The GFS's architecture is applied through the use of cluster computing
- Through the utilization of commodity hardware, files must be stored in clusters which makes persistent monitoring, error detection, fault tolerance, and automatic recovery an essential part of the system's functions
- Data is stored in an unorthodox fashion. In the GFS the fault tolerance system allows for cheap and efficient storage methods.
- The GFS is necessary for the following reasons: processing of robust data files, durability, data redundancy, and error tolerance.

The Google File System: *Analysis of Implementation*

- The distinguishing factor of the GFS is that component failures are treated as the norm rather than a rare occurrence.
- Without the GFS's built in tolerance systems, Google wouldn't't be able to carry out with daily operations at such an efficient rate.
- The implementation of the GFS has allowed Google to allocate immense amounts of data clusters to the entire world.
- All of the GFS capabilities have been shaped by the fact that google is able to store multiple copies of the internet throughout the entirety of the system.
- The GFS is a crucially important tool which has enabled Google to innovate and attack different problems on the scale of the world wide web.

Comparison to Approaches: *Main Ideas*

- There are two central approaches in the analysis of large-scale data.
- The first: MapReduce or MR in short. A paradigm which some have called a dramatically new computing model. Popular for its simplicity and pays extra attention to the user end, quicker loading times and fault tolerant characteristics.
- The second: Parallel DBMS. Even though loading data may be proven to be more time consuming, the overall performance and results were shockingly more intriguing than those of the MR. Even though it's attributes are less omnipotent, the DBMS's supportive tools prove to be very valuable.
- Despite the multiple architectural differences between both paradigms, they are more than suitable for the processing of large data.
- Both approaches have their variations, and as time progresses we see more and more versions of these paradigms being built: Hadoop and Hive are two major examples of these variations.

Comparison of Approaches: *Implementation*

- When analyzing fault tolerance in both paradigms, it’s important to understand the distinctions of each, respectively.
- In terms of the parallel DBMS, files are saved across a network rather than in a local environment. For this reason, files saved become accessible for multiple users at a time.
- In the case of the MR, a node system is implemented to cope with fault tolerance. When failure occurs, the task is simply run on another node, rather than restarting the query.
- From a programming perspective, MapReduce offers more flexibility than the parallel DBMS. In the MR there is no outside support, so programmers are allowed to implement their own.
- Altogether, the MR seems to be focused around sole application situations, opposing the parallel DBMS, where we see it being manipulated across multiple application platforms.

Comparison of Approaches: *Analysis of Implementation*

- When comparing DBMS-x, MR, and Vertica, it’s apparent that the MR lacks significantly in speed, being nearly twice as slow as Vertica, and practically three times slower than the DBMS-x. Due to this liability, MR would not be my first choice if time management was the priority, and time is ALWAYS the priority…
- “Storage is cheap, time is expensive”-Alan
- While executing different tasks, MR may require more code than the DBMS and Vertica.
- MR uses Hadoop, which allows for quicker and easier use within the MR system. Essentially, Hadoop is a Java implemented program that processes data very effectively with little to no errors.

The Final Comparison

- After final analysis, I find the idea and implementations of the Google File System to be the most intriguing.
- The volume of data that the Google File System is able to retain and regurgitate is more than adequate, and seems to be working pretty well for Google.
- With the built in fault tolerance operations, the Google File System is virtually indestructible. So GFS, you receive Blaise Spinelli’s vote.
- Looking at the remaining data systems, both the MR and DBMS are definitely impressive to say the least.
- Out of the two, I would have to go with the MR simply because of it’s time management advantages.

Stonebraker: *Main Ideas*

- Initially, their intent was to have the RDBMS a "*one size that fits all"* design concept. After consulting with "The Team," (Zdonik, Cetintemel, Cherniack, Tathul, Madden) it became glaring obvious that this idea was impossible.
- As the video progresses, Stonebraker dwells into how many of the database markets he encountered were not very productive at all. This list included: Data Warehouse, noSQL, and OTLP.
- Stonebraker seems to be very intrigued by the concept of column stores. He makes the claim that they will most likely replace row storage systems and could be the solution to complex analytics.
- With the growing diversity of engines within major database systems, the traditional row format is becoming obsolete.

The Google File System & Stonebraker: *Advantages and Disadvantages*

- The most impressive attribute of the Google File System which makes it so advantageous is its built in fault tolerance system. It is also very attractive because of its ability to allocate data efficiently, process large scale data.
- Primarily, The Google File System seems especially beneficial as a database system because of it's large storage chunk (64 MB). It is important to note what exactly the significance behind this chunk size is.
  - ❖By being so large, this reduces direct interactions between the master and the client, reducing the size of metadata in conjunction.
- The collection methodologies of the Google File System pose very interesting advantages, as well as a fair share of disadvantages.
  - ❖One perk of the GFS is that it is very simplistic. Tasks are very simple to perform at a very efficient rate.
  - ❖However, with the GFS, since the storage capacity is compromised, it may be difficult to make data precise.