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

By: Daniel Njoku

GFS Summary

- The main idea of the Google File System article is to give the reader a deeper understanding of how Google file system operates.
  - This file system was initially built in order to meet the specific needs of their desired workload.
- GFS understands its purpose while performing the tasks it was assigned for.
- GFS supports large scale data processing while using commodity hardware.
- Other primary points within the paper discussed the importance of design overview, system interactions, and master operations.

GFS Implementation

- Why is Google File System necessary? GFS is needed for:
  - Robust data files, reliability, replication of data, and fault tolerance.
- The primary design of GFS would assume the position of:
  - Storing large amounts of data and monitoring components.
- The architecture of GFS is implemented through the use of cluster computing
- The idea is implemented through the means providing a fault tolerance system while storing files in cluster over commodity hardware.
  - This means that data is stored in way that is cheap but yet efficient
- Files are organized hierarchically within the directories and are classified by path name.
- Multi-way merge

My personal analysis on GFS implementation

Files are divided into fixed sized chunks equating to to 64MBs.

- GFS implementation has the capability of distributing multiple clusters across the worlds.
- The capability to analyze millions of queries a minute is a unique ability, however Google understood that in order to fulfill this niche they would need to develop their own autonomous file system.
- The following factors shaped the implementation of GFS
  - An individual query consumes billions of cpu cycles
  - A sole query is responsible for reading hundreds of megabytes of data
  - The fact that google stores multiple copies of the internet throughout its system
- In conclusion, the need for a largely distributed, fault tolerant file system is necessary in order to carry out the tasks performed by Google on a daily basis.

Comparison Paper Summary

- The objective of the comparison paper is to broaden the readers knowledge on how large scale data can be viewed through a comparative viewpoint.
- There are two primary approaches to analyzing large scale data.
    - MapReduce (MR) & Parallel DBMS
        - Both of these approaches are suitable for large scale data processing
- There are several architectural differences between these two approaches.
- MapReduce is known for its simplicity in regard to the end user, fast loading times, and its enticing fault tolerant attributes.
- Although it is not as omnipotent as MapReduce, the DBMSs are beneficial due to their supportive tools and hastily query times.
- Both MapReduce and DBMS have their pros and cons. Due to the beneficial aspects on both ends, several hybrids are becoming more popular as time goes on.
    - Hadoop, Hive, Pig, etc...

Comparative Implementation

- Scheme Support
  - MR is primarily used in single application situations, where as Parallel DBMS can be used on multiple application platforms making it more versatile than MR in some capacity.
- Program Modelling
  - MR seems to be a lot more flexible compared to parallel DBMS in regard to programming.
- Indexing
  - Due to the fact that there is no native support within MR, programmers can implement their own. Although this is beneficial on one end, it is difficult to share across multiple applications.
- Fault Tolerance
  - Rather than saving results locally, parallel DBMS save files across a network. This allows it to become accessible to multiple users.
  - The fault tolerance within MR seems to hold up better due to the fact that when a node fails, the same task is run on another node; rather than re-running an entire query.
- Transfers of Data

My Analysis of Comparative Implementation

- MR requires more code than DBMSs and Vertica in order to perform each task.
- MRs is fairly easy to use and is known to take less time to set up while using Hadoop
  - Hadoop is Java based program that is widely known for its ability to processes large amounts of data without any errors
- Compared to DBMSx, MR was nearly three times slower than DBMS-x and nearly twice as slow than Vertica.
  - Due to this discrepancy MR would not be a good choice when considering time management.

Comparison of the Two Papers

I find that all data systems covered within the both papers can be viewed as beneficial. I find GFS the most intriguing due the the volume of data that they are able to retain. Although radical, the GFS works for Google with minimal disadvantages when it comes to scalability. MRs are fairly intriguing; although they are difficult to share across different applications I completely understand why they are becoming more prevalent, leaving DBMS in the dust. With big data serving a more dominant role within business, in regard to data analysis, at times I wonder what the future of data storage has in store.

Stonebraker's Main Concepts

- The initial intent was to make it so that RDBMS the "one size that fits all" according to Stonebraker.
  - However, after Stonebraker & Setintemal collaborated on *One Size Fits All*, they realized that this desire was not feasible.
- In 2015, Stonebraker came to realization that in regard to RDBMS, one size fits none.
- Stonebraker continued to explain that a multitude of database markets are indeed not useful for much, such as:
  - Data warehouse, OLTP, noSQL, and streaming markets were not good for anything.
- Stonebraker continued to elaborate on how row storage systems will soon be replaced by column stores.
- The viewpoints of Stonebraker shows that he is jaded when it comes to complex analytics but this believes that column stores are the answer.
  - Stonebraker is open to other implementations, such as, arrays.
- Within the major markets there is a huge diversity of engines, and traditional row stores are beginning to become obsolete.
- The idea that one size fits is a fleeting concept. This leave plenty of room for future research and implementation.

Comparative Advantages and Disadvantages

- Google's file system has several advantages, as well as disadvantages. In regard to its design, its advantageous that that GFS allows such large chunk sized (64MB).
  - This reduces interactions between the client and master, as well as reduces the size of metadata and network overhead.
  - Due to this internal fragmentation occurs, however, this can be solved by using lazy space allocation as specified in the paper.
- GFS garbage collection methodologies are fairly unique while having its advantages as well as disadvantages. For example;
  - It's simplistic and reliable and tasks are performed when the master is free
  - On the other spectrum, it is hard to make data precise when storage capacities are at risk.
- It becomes obvious that GFS is the way to go, due to its
  - Fault tolerance, high aggregate throughput, and large scale data processing.