The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

A Comparison Approach to Large-Scale Data Analysis

Andrew Pavlo, Eric Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker

One Size Fits All – An Idea Whose Time Has Come and Gone

Micheal Stonebraker (2005)

Brian Henderson

October 20th, 2016

The Google File System: Main Idea

- A proprietary distributed file system designed to provide reliable, efficient, and flexible access to data intensive applications using large clusters of commodity hardware
- Designed to:
  - Meet the rapidly growing demands of Google's processing needs
  - Have a single master to simplify design
    - Enables the master to make sophistical chunk placement and replication decisions convenient using global knowledge.
    - Clients never need to read or write file data through the master, instead the client receives which chunk-servers it should contact from the master.
  - Use inexpensive commodity hardware
    - Inexpensive commodity hardware fails often, therefore constant self-monitoring, error detection and correction, fault tolerance, and automatic recovery are essential for the GFS to operate successfully.

The Google File System: Implementation

- GFS is organizes into clusters (networks of computers) where there are three distinct entities with unique operations:
  - ***Clients, Chunk Servers, Master Servers***
- **Clients**: Make's requests via the master to add or alter (mutations) data files on the GFS cluster. Master sends the client the requested chunk location.
- **Chunk Servers**: store the data in 64-mB chunks. Sends requested chunks directly to the client. Copies every chunk multiple times, known as *replicas,* and stores on different chunk servers.
  - Metadata stored: Checksums for each block of user data
- **Master Servers:** Keeps an operation log of all operations on the GFS cluster.
  - Metadata stored: File name, file ownership/permissions, mapping from files to chunks, each chunks current version, and replica location.

The Google File System: Analysis

- Sophisticated system that seems to be very efficient, reliable, and flexible.
- The Use of Inexpensive Hardware
  - Google capitalizes on the GFS's design and strength to compensate for the cheaper Linux based hardware. Although chance for failure is high, sophisticated approach that allows for self monitoring and repairing outweighs negatives.
- Single master
  - A single master creates a simpler, and cleaner design, without creating a data bottleneck which is essential when dealing with large data.
- 64 mB clusters
  - Having a flat sized clusters make design and implementation simpler, but optimization would ensure all data is being used and not wasted within each cluster.
- Replicas
  - Replica data allows for extensive backups for any faults. Replicas are used in duplicating of chunks and masters, in-case of failure to ensure smooth transition.

Comparison Approach to Large Scale Data: Main Idea

- Compares advantages to the MapReduce (MR) paradigm and Database Management Systems (DBMS).
- The Hadoop system, demonstrating the MapReduce framework, is efficient and quick in loading data initially, but query functionality is sub-par compared to DBMS.
- DBMS take longer to load data, but once fully implemented, the performance is significantly better.
- Data sharing is not convenient in MR as a programmer must decipher the original programmers code to determine how to process the input file, while in DBMS, the schema is separated from the application and stored in a set of system catalogs that can then be queried.

Comparison Approach to Large Scale Data: Implementation

- Using Benchmark Execution, the two database systems can be tested comparatively without having to coordinate parallel tasks. The tasks are executed three times, reporting the average, separately to ensure exclusive access to the cluster's resource, ensuring consistent fields.
- Measured the time for each system to load the data.
- In the Benchmark Execution, both DBMS-X and Vertica execute most of the tasks much faster than Hadoop at all scaling levels.
- The tasks performed were a selection, an aggregation, a join, and a UDF aggregation task.
  - **Selection Task***:* DBMS outperformed Hadoop by a significant amount due to Hadoops start up time.
  - **Aggregation Task**: Two DBMS outperformed Hadoop.
  - **Join Task**: DBMS outperformed Hadoop due to it being limited by the speed with which the table (20GB/node) can be read off the disk.
  - **UDF Aggregation Task:** Hadoop outperformed DBMS due to the the added overhead of row-by-row interaction between the UDF and the input file stored outside of the database

Comparison Approach to Large Scale Data: Analysis

- The database management system is, in my opinion, the more superior choice when dealing with large scale data.
- Although the fault tolerance for DBMS is low, DBMS outperformed the MR system 3 to 1 in the tests.
- Data sharing is very easy and convenient with DBMS, where in MR, data sharing can be very complex.
- Data in a MR framework can be easily corrupted with bad data. In DBMS, the integrity of the data is enforced with no additional work on the programmers behalf due to separating the constraints from the application. Overall, that makes the data in DBMS have a higher integrity which is very favorable when dealing with big data.

Paper Comparison

- The two papers compare different ways and technology to store big data.
- The Google File System paper really goes into depth of the how the GFS works and points out any flaws, but shows how the flaws are not an issue most of the time. Whereas the second paper really points out all the goods and bads of the other data storage systems.
- The Google File System is a good option when the purpose of the data is tailored towards the GFS's design, where the Database Management System seems to be a very good option for most data as it is more streamlined for user interaction, especially with data sharing, a very important consideration when dealing with large amounts of data.

One Size Fits All – An Idea Whose Time Has Come and Gone: Main Idea

- Stonebreaker believes that the relational database management system will never be the "One Size Fits All" and universal concept everyone wants it to be.
- He believes that DBMS has become obsolete and good at nothing, ("One Size Fits None!") shown in the following markets:
  - Data Warehouses: Column stores are faster
  - OLTP: Moving towards main memory, Row storage too heavy weight for the lightweight transactions
  - NoSQL: 100+ vendors
  - Complex Analytics: Data scientists will replace business analytics and Data Scientists don't want to run SQL analytics, they want to run regressions, etc…
  - Streaming: OLTP engines have a greater market share as they are much easier than adding persistence to a streaming engine.
  - Graph Analytics: Simulates column store and array engine.

Comparing *The Google File System* to *A Comparison Approach to Large-Scale Data Analysis* and One Size Fits All – *An Idea Whose Time Has Come and Gone*:
Advantages and Disadvantages

- Advantage: The GFS was designed surrounding Google's needs and purposes. GFS works very well for Google's needs and intentions.

- Disadvantage: GFS is very tailored towards Google and would not be as practical for all other large scale data systems. Stonebraker discus's the huge diversity of engines in the market. The different engines are targeted towards specific applications, which could be much more practical depending on the application.