The Google File System

by Ghemawat, Gobioff, and Leung

&

A Comparison of Approaches to Large-Scale Data Analysis

by Pavlo, Paulson, Rasin, Abadi, DeWitt, Madden, and Stonebreaker

Ryan Bertsche

May 9, 2014

The Main Idea of **The Google File System**

- Create a file system that can handle large amounts if distributed data
- Make this system efficient for data intensive applications
- Make this system failure tolerant
- Allow a large number of clients to access the data simultaneously
- Run on inexpensive hardware, that can easily and cheaply be replaced
- Support distributed applications
- Create a system that is fast and efficient
- Spread the data across multiple computers using clusters

Implementation of **The Google File System**

- One main master server holds meta-data about the mapping of files to chunks, access control to those chunks, and it runs system wide processes
- Data files are blocked into 64 MB files, and stored as a plain linux file
- Chunks are stored on *chunkservers*, which are servers comprising of low cost hardware
- Chunks have a program defined number of duplicates, usually stored on different chunkservers
- Mutations to files are all logged, and checksums are run on the data to make sure all the data is correct over all the copies of the chunks
- Outdated chunks are marked for deletion, and garbage collected in the background
- When chunkservers go offline, data is copied based on number of expected copies vs actuall copies
- Master server gives the adresses of the data requested, and leases that data for use, while the application gets direct acces to data

Analysis of **The Google File System**

- This file system allows for great flexibility in the type and structure of the data, which is its biggest advantage
- There is a great level of redundancy across many clusterservers that are cheap, so hardware failure is well accounted for, planning well ahead for inevitable hardware failures
- This system is implemented in such a way that is highly specific for the client needs. It expects to have clients request large amounts of data to be used for a long time, so its efficiencies are highly tuned for this purpose, making it less useful in general
- There needs to be many different clusterservers to be able to facilitate multiple users to ensure that there aren't bottlenecks of clients trying to simultaneously access data
- An abundance of clusterservers could then put strain on the singular master to perform maintenance, and direct multiple simultaneous file requests in an implementation dissimilar to Google's

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

- The comparison paper stated and proved that Google style map reduce file systems are inferior to parallel DBM systems in terms of times of queries
- These DBMS are readily available and more advanced at this time
- The structure of the DBM systems allow for data to be more accurately maintained, and queries on that data to be returned much faster
- The inherent structure and logic of Relational databases require more time to set up the data into schemas, as well as increased file load times compared to the GFS
- The increases structure limits some flexibility, but this can be worked around by newly developed DBMS tools
- Less nodes are needed to store massive amounts of data, decreasing space and power needs in their implementations

Advantages and Disadvantages of **The Google File System**

- The GFS offers more flexibility in how and what data is stored, not having to adhere to strict Relational Database rules, or take the time to establish schemas. Companies seem to have specific needs from their data that cannot not be easily implemented into a relational system
- While GFS is slower in the present, the idea is in its infancy, and lacks the fine tuning and optimization tools already adapted by traditional DBM systems. This means that given time to mature, Map- Reduce systems could evolve to have higher capacity and speed
- The background processes that ensure data duplication, availability and reliability are not standardized in Map-Reduce systems, unlike relational systems, meaning the rules have to be explicitly provided in GFS
- GFS will most likely have to take design principles of relational systems and apply them in order to achieve the lower query speeds of relational systems, or inversely, the map reduce concept could be applied to DBMS to create a super-hybrid of awesome, fast, reliable, efficient big-data storage