The Google File System, Big Data, & A Word from Michael Stonebreaker

By Charles Grippaldi

October 10th, 2017

Ghemawat, Sanjay, et al. "The Google File System." *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, Jan. 2003, p. 16., doi:10.1145/1165389.945450.a

Pavlo, Andrew, et al. "A Comparison of Approaches to Large-Scale Data Analysis."*Proceedings of the 35th SIGMOD International Conference on Management of Data - SIGMOD '09*, 2009, doi:10.1145/1559845.1559865.

Stonebraker, M., and U. Cetintemel. "'One Size Fits All': An Idea Whose Time Has Come and Gone." *21st International Conference on Data Engineering (ICDE'05)*, doi:10.1109/icde.2005.1.

The GFS: Main Idea

- Data-intensive applications use the GFS
- It runs on hardware that constantly self-monitors and is inexpensive (so it fails often)
- It's a storage platform for research & development as well as the generation and processing of data
- Observations of application workloads and technology environment were used to create it's design

The GFS: How It's Implemented

- It's cluster has single master and multiple chunk servers which are accessed by multiple clients
- The files get cut into block sizes of 64 MB (this is bigger than block sizes of average systems)
- The chunk replicas are stored on a chunkserver as plain Linux files (extended only as needed)
- The master stores metadata of 3 types:
  - File and chunk namespaces
  - Mapping from files to chunks
  - Locations of each chunk's replicas

The GFS: A Personal Analysis

- Highly productive
  - Utilizes diagnostic tools which identify issues right away
  - Chunks provide a great way to organize data
    - Easily cached
    - Reduces need to interact with the master file
    - Reduces network overhead
    - Reduces the size of metadata
- Components are inexpensive, which allows for failing to be a normal and expected occurrence that can be easily planned for
- Fast recovery
- Efficient replication
- Utilizes a relaxed consistency model which supports vastly distributed applications all the while staying simple and efficient to implement

Comparison Paper: Main Idea

- This paper compares and evaluates both the MapReduce paradigm and Parallel Database Management Systems (with regards to complexity and performance)
- As a whole, PDMS's are faster and more advanced. It supports standard relation tables and SQL. It takes a larger staff as well as a larger budget to maintain.
- MapReduce can load data quicker, leaves some coding up to the programmers, and is overall a simpler program.
- PDMS turns out to be better than MapReduce overall.

Comparison Paper: How It's Implemented

- The authors used DBMS-X, Hadoop, and Vertica to test differences in speed, performance, flexibility, and fault tolerance.
- MapReduce
  - Structures data in anyway, if at all.
  - Uses mainly object oriented programming.
  - No built-in indexes
  - Did better during the Grep Tasks testing
- PDMS
  - Use a relational model to structure data, and mainly use SQL
  - Current ones utilize hash or B-tree indexes. This improves access to data.
  - Found to be superior
- Both offer runtime support

Comparison Paper: A Personal Analysis

- PDMS
  - Reigns supreme in most scenarios and should be a go-to choice over MapReduce for larger companies.
  - Outperforms MR and provides something that MR doesn't: structure. Structure is very beneficial.
- MapReduce
  - A better pick for smaller businesses, leaving it open-ended for programmers to really customize the system
  - Uses more resources
- We still have much to learn for both systems.

Comparing These Ideas and Implementations

- MapReduce provides little to no structure, but GFS is a cluster archictecture including GFS masters, chunk servers, and clients.
- GFS stores data in chunks which are given a unique 64-bit identifier on a linux chunk based.
- PDMS implements by using a relational model, whereas MapReduce might use something or not anything at all.

Michael Stonebreaker: Main Idea

- Relational databases are supposed to be "one size fits all"
- The "innovators dilemma" states that vendors who sell old technology have difficulties when switching over to new technology
- Streaming applications had no place on the traditional row-store of RDBMS implementation
- In actuality, one size fits none.
  - Data warehouse market is going to column stores because they are much faster
  - OLTP market is moving toward lightweight transaction systems
  - There aren't even standards in the NoSQL market
  - Complex analytics is moving to business intelligence products
  - Streaming market has a larger market share
  - Graph analytics market simulates an array matrix or column store architecture

Advantages & Disadvantages of the GFS in the Context of the Comparison Paper and Stonebreaker Talk

Advantages

- Modular data from chunks on chunk servers
- Duplicate data helps prevent data corruption

Disadvantages

- Component failures
- Inexpensive components give way to abundant overhead

- GFS doesn't use a traditional DBMS
- GFS is a good example of Stonebreaker's noSQL market. Google created its own architecture and data model based off its current applications.

Thank you!