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

Graham Burek

March 16, 2016

References:

1. Pavlo, Andrew, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. Dewitt, Samuel Madden, and Michael Stonebraker. "A Comparison of Approaches to Large-scale Data Analysis." *Proceedings of the 35th SIGMOD International Conference on Management of Data - SIGMOD '09* (2009): 165-78. Web. 13 Mar. 2016.
2. Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google File System." *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles - SOSP '03* (2003): n. pag. Web. 13 Mar. 2016.

The Google File System

Main Ideas:

- A storage system is needed that can run data-intensive applications
- Data processing for these applications is done with distributed, inexpensive hardware
  - This hardware is highly prone to failure (software bugs, networking outages, etc.)
- Files to be processed are very large
  - Multiple GBs
- Files are mostly modified by appending new data onto an existing file
- Files are mostly read sequentially in large streams or randomly in small pieces

The Google File System

Implementation:

- The Google File System (GFS)!
- Architecture:
  - GFS clusters made of:
    - **Chunkservers** – Store chunks (fixed-size Linux files) containing the data
    - **Masters** – Have the metadata for the file system, periodically communicate with and command chunkservers
  - Chunks replicated on different machines, as well as master metadata
- Operation
  - Client wants to write data
    - Replicated chunks written to chunkservers in linear sequence
    - Writes may be split if too large to fit in chunk

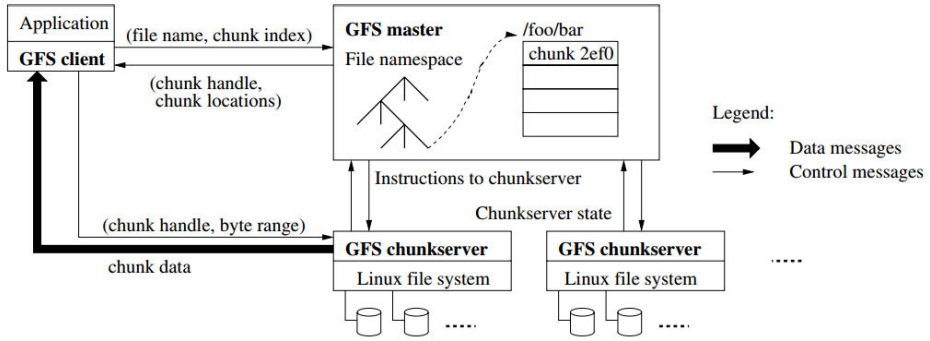

Figure 1: GFS Architecture

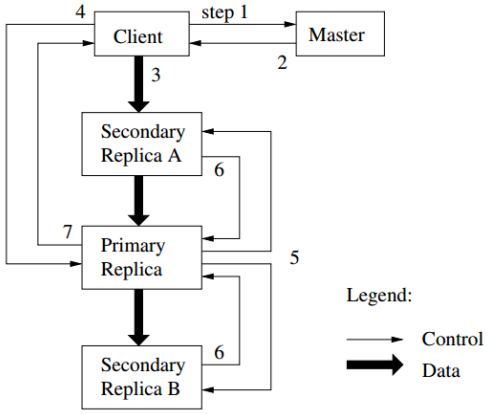

Figure 2: Write Control and Data Flow

The Google File System

Analysis:

- Well suited to a distributed system
  - Data is replicated on different chunkservers, as well as on different racks for backup
  - Good if hardware breaks down
- Not good solution for many small reads
  - Getting data from many different chunks could get expensive
- Good error recovery capabilities
  - Master server polls chunkserver for data
- Favors speed over space

A Comparison of Approaches to Large-Scale Data Analysis

Main Ideas:

- Aim is to compare new cluster computing strategies (GFS, MapReduce) with parallel DMBS
- Want to find the trade-offs between the two
- Focus of comparisons is on performance, as well as complexity of use and development using the systems

A Comparison of Approaches to Large-Scale Data Analysis

Implementation:

- Testing is done using **Hadoop**, **DBMS-X**, and **Vertica**
  - **Hadoop** – Open source MapReduce implementation
  - **DBMS-X** and **Vertica** – two parallel DMBS

Results:

- The parallel DMBS have much longer load times, but perform the tested tasks faster than Hadoop
  - Hadoop, however, is much easier to get up and running
- Hadoop is more easily extensible

A Comparison of Approaches to Large-Scale Data Analysis

Analysis:

- Hadoop
  - Although it executes slowly, is better than Parallel DMBS at error recovery during query execution
    - An important consideration for large amounts of data
  - Is programmed in an object-oriented style (arguably more familiar)
- Parallel DMBS
  - Are preferred for efficient processing
    - Are well optimized with indexes
    - Use much less power (require fewer nodes in cluster)

Google File System VS Parallel DMBS

- When is GFS's slower clustered computing strategy useful?
  - On unreliable, distributed hardware (better error recovery)
  - When large amounts of data need to be quickly queued for processing (but not necessarily quickly processed)
  - When storage is not a consideration (compression slows execution further)
- On the other hand, Parallel DMBS are favored for fast, efficient processing

Stonebreaker Talk

Main Ideas:

- The ‘one size fits all’ ideology for row-based, traditional DMBS is false
  - In fact, row-based DMBS are becoming obsolete!
- Column-oriented DMBS will take over (better overall performance)
- Alternate systems could be better than relational system for many use cases
  - Streaming
  - Complex analytics
  - Graph analytics

GFS and Other Markets

In what other markets would GFS be well-suited?

- GFS is better suited than traditional, row-based DMBS for specialized tasks (not constrained by SQL for more complicated data processing)
  - Therefore, GFS could be suitable for complex and graph analytics
- GFS is able to process huge data sets at once
  - Therefore, GFS may be suitable for the analysis of the Internet of Things
- GFS is designed for sequential access to large data sets
  - Therefore, the GFS (with enough space) would be well-suited for streaming