The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

Mike Mucci & 5/9/14

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo
Brown University
pavlo@cs.brown.edu

Erik Paulson
University of Wisconsin
epaulson@cs.wisc.edu

Alexander Rasin
Brown University
alexr@cs.brown.edu

Daniel J. Abadi
Yale University
dna@cs.yale.edu

David J. DeWitt
Microsoft Inc.
dewitt@microsoft.com

Samuel Madden
M.I.T. CSAIL
madden@csail.mit.edu

Michael Stonebraker
M.I.T. CSAIL
stonebraker@csail.mit.edu

The Google File System

- To have constant monitoring, error detection, fault tolerance, and automatic recovery; failure is a norm rather an exception.
- Standard I/O operations and block sizes are reconsidered due to the high frequency of big files
- Data appending is a performance focus based on simultaneous and multi-terminal edits
- Simple so that general and concurrent application access is smooth and unhindered

The GFS Implementation

- Single master with multiple Linux chunkservers that handle the major client/application communications (read/write/etc…).
- Operation log of critical changes is used to keep track of operations performed. Master and chunckservers are made to have a fast start-up,replicate its data for system reliability and for user reliability, respectively. The master checks for dead chunckservers with handshakes between all chunckservers.
- Mutations (writes/appends) are handled by the chunk. Mutated data is sent to all chunks, the primary chunk (chosen by the master) then calls to write the data to the primary. Once written, the primary sends write requests to the other chunks.
- Applications adapt the techniques used from other system design choices: Focusing on data appends, utilizing checkpoints, and using checksums to check data integrity.

Personal Analysis of GFS

- I find that viewing failure as a norm creates an extreme amount of overhead that would hinder other systems/companies. However due to the size of Google, they have the ability to afford that cost.
- The use of chunkservers is a good way modularize the data. Duplicating data on each chunk, while costly for some, is a good way to combat possible data corruption.
- The operation log acts like a normal database log with checkpoints roll forwards/backwards, which is a design that is and should be implemented regardless.
- Google had a set of goals for what their system should do, and they constructed a system that achieves each goal. Meaning this design is good for them but definitely not for all

GFS vs Large-Scale Data Analysis

- A MapReduce (MR) model is slower than a parallel model. Although not confirmed, GFS, being a MR instance, is expected to share this disadvantage
- MR model, with Hadoop as the example, is easy to set up
- MR model handles data loss due to hardware superiorly compared to parallel, however with potential performance losses.

GFS: Advantages & Disadvantages

- Engineered to handle big files with an explicit focus on data appending. Unlikely sharing the degrade in performance like GFS's MR relatives.
- Extensible, seen with the integration of GFS myriad of applications
- Since GFS, and its applications, are specifically tailored for its goals, it is likely to not suffer to some of the pitfalls of MR
- Fault tolerance overhead has the potential to hinder performance greatly.
- Performance on retrieval and insert are slower than parallel systems.
- Synchronization processing with control messages reduces performance


Thank you alot
for reading!