BIG DATA PAPER SUMMARY

THE GOOGLE FILE SYSTEM
A COMPARISON OF APPROACHES TO LARGE-SCALE DATA ANALYSIS
MICHAEL STONEBREAKER TALK

BY: MATTHEW LAKE
3/4/2017

BIBLIOGRAPHIC DATA:

HTTP://LABOUSEUR.COM/COURSES/DB/PAPERS/GHEMAWAT.SOSP03.GFS.PDF

HTTP://LABOUSEUR.COM/COURSES/DB/PAPERS/PAVLO.SIGMOD09.COMPARISON.PDF

HTTP://KDB.SNU.AC.KR/DATA/STONEBRAKER_TALK.MP4

MAIN IDEA OF THE GOOGLE FILE SYSTEM (GFS)

- Designed and implemented to meet the rapidly growing demands of Google’s data processing needs.
- Support Google’s workloads and technological environment, both current and anticipated.
- Constant monitoring, error detection, fault tolerance, and automatic recovery.
- Ability to efficiently handle extremely large data sets.
- Appending becomes the focus of performance optimization and atomicity guarantees.
- Co-designing the applications and the file system API benefits the overall system by increasing our flexibility.

HOW THE MAIN IDEA IS IMPLEMENTED

- The system is built from many inexpensive commodity components that often fail creating a constantly monitored environment within the system.
- A GFS cluster consists of a single master and multiple chunkservers and is accessed by multiple clients.
- GFS has a relaxed consistency model that supports Google’s highly distributed applications well but remains relatively simple and efficient to implement.
- GFS decouples the flow of data from the flow of control to use the network efficiently.
- GFS provides an atomic append operation called record append.
- Fast recovery and replication.

MY ANALYSIS OF THIS IDEA AND ITS IMPLEMENTATION

- The Google File System was designed to handle large data sets and to be run over commodity hardware.
- The GFS was created in order to operate in ways that a traditional file system could not.
- The design delivers high aggregate throughput to many concurrent readers and writers performing a variety of tasks.
- The system places a high priority on fault tolerance within its implementation.
- GFS successfully met their storage needs and is widely used within Google for research and development, as well as production.

MAIN IDEAS OF THE COMPARISON PAPER

- What is MapReduce?
- What is a Parallel DBMS?
- How do these two classes of systems make different choices in several key areas?
- Understand the approach of both MapReduce and Parallel DBMS in regards to performing large-scale data analysis.

HOW THOSE IDEAS ARE IMPLEMENTED

- Review of the two alternative classes of systems.
- The architectural trade-offs.
- Benchmark consisting of a variety of tasks from each class of system.
- Reasons for the differences in approaches.
- Suggestions on the best practice for any large-scale data analysis engine.

MY ANALYSIS OF THOSE IDEAS AND THEIR IMPLEMENTATIONS

- Both kinds of systems have a lot to offer, when deciding on one or the other you must take into consideration which kind will best utilize your needs.
- MapReduce borrows many key ideas from parallel database systems.
- MapReduce and Parallel DBMS are moving toward one another.
- Solutions for integrating SQL with MR offered by Greenplum and Asterdata is evidence.

COMPARISON OF THE TWO PAPERS

- Both papers involve a system used to work with large scales of data.
- Include observations that led to conclusions
- Discussion of data flow
- Use diagrams in order to illustrate specific data and information
- Discussion of fault tolerance

MAIN IDEAS OF THE STONEBREAKER TALK

- 1970-2000: attempt to make RDBMS “universal”
  2005: realized RDBMS had no chance of continuing
  2015: one size fits none
- All major vendors have (or soon will have) column stores
- Move to putting all data in main memory
- Data scientist soon to replace business analyst
- Traditional row stores are good at none of the markets discussed in the video
- We expect to see a lot of new implementations

ADVANTAGES AND DISADVANTAGES

Advantages:

- The GFS addresses fault tolerance by keeping the master state small and fully replicated on other machines. MR frameworks also have sophisticated failure model. Unlike parallel DBMS the GFS does not have large bodies of work that have to be restarted in an event of a failure.
- The GFS had a design focus similar to what Michael Stonebraker spoke about. That new implementations are upon us. GFS design was an effort to be current but also anticipate the future of the database field.

Disadvantages:

- Biggest problems with the GFS were disk and Linux related
- Mismatches would cause the drive and the kernel to disagree about the drive's state and lead to a corruption of data.