BIG DATA PAPER SUMMARY

1. *"10 Year Test of Time" paper award*. Perf. Michael Stonebraker. N.p., 2015. Web. 1 May 2017. <http://kdb.snu.ac.kr/data/stonebraker_talk.mp4>.
2. Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google File System." SOSP'03, 19 Oct. 2003. Web. 1 May 2017.
3. 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." *Www.science.smith.edu*. Sigmod 2009, June & july 2009. Web. 1 May 2017. <http://www.science.smith.edu/dftwiki/images/6/6a/ComparisonOfApproachesToLargeScaleDataAnalysis.pdf>.

Deborah Sylvester
Alan Labouseur

Database Systems
2 May 2017

THE GOOGLE FILE SYSTEM

A Scalable Distributed File System for data-intensive applications

- Provides fault tolerance.
- Uses low-cost hardware.
- Clustered Storage.
- Allows concurrent access by many clients.
- Conforms to the ACID model.

THE GOOGLE FILE SYSTEM

Uses low-cost hardware:

- Off the shelf hardware

Clustered Storage:

- Hundreds of storage systems
- Many machine racks
- Communication via network switches

Allows concurrent access by many clients:

- Record append – while allowing for atomicity
- Snapshot – creates copies at low cost

Provides fault tolerance:

- Many replicas of the data – Assumes fails will occur

Conforms to the ACID model:

- Operation Log and Checkpoints
  1. Replicated on many machines
  2. Can restart an activity if it fails
  3. Chunk Servers maintain checksums
- Master Data – stores metadata
  1. File and chunk namespaces
  2. Mapping from files to chunks
  3. Locations of chunk's replicas
- HeartBeat messages
  1. Detects orphaned chunks
  2. Leases and mutations tag along
- Locks for reads and writes

THE GOOGLE FILE SYSTEM

- Clear, focused paper about the design and how it's implemented.
- Good design based on current and anticipated workloads.
  - Highly fault tolerant – System was designed assuming faults will occur.
    - Use of low-cost machines.
    - Three replicas of chunks.
    - Master maintains metadata about the chunks.
    - Provides automatic recovery.
  - Design optimization assumes more record appends than writes.
  - Files are large compared to conventional measures.
- Different approach from traditional distributed file systems.
- HeatBeat messages with the master enables a simple solution to lost or broken chunks, ie..Garbage Collection.
- A disadvantage to this design is the amount of energy required to run all the systems.
- Paper achieved it's purpose explaining the design and real world use.

A COMPARISON OF APPROACHES TO LARGE-SCALE DATA ANALYSIS

Map Reduce (Hadoop) and two parallel SQL database management systems (DBMS-X & Vertica) were compared for the following:

- Data Format
- Indexing
- Programming Model
- Data Distribution
- Flexibility
- Query Execution
- Fault Tolerance
- Performance

A COMPARISON OF APPROACHES TO LARGE-SCALE DATA ANALYSIS

Data Format
- DBMS - rows and columns
- MR - no structure needed

Indexing
- DBMS - Hash or B-Tree
- MR - Programmer enforced

Programming Model
- DBMS - Relational
- MR - Algorithm Based

Data Distribution
- DBMS - Query Optimizer
- MR - Programmer

Flexibility
- Both are flexible

Query Execution
- DBMS - Data is "Pushed"
- MR - Data is "Pulled"

Fault Tolerance
- MR minimizes work loss better than DBMS's

Performance
- DBMS performed significantly better than MR overall
- MR uses much more energy

A COMPARISON OF APPROACHES TO LARGE-SCALE DATA ANALYSIS

- Extremely well thought out benchmark comparing two different systems.
  - Map Reduce versus parallel DBMS.
- Very detailed focused paper.
  - Clearly broken down in sections by benchmark test and systems.
- Map Reduce does much better with respect to fault tolerance than the DBMS. Overall, DBMS performed much better on the benchmark tests.
- Benchmark tasks tested included:
  - "Grep task"
  - Analytical Tasks such as: Selection, Aggregation, Join and UDF Aggregation
- An informative, understandable research paper.

COMPARISON OF BOTH PAPERS

- *The Google File System* paper focused extensively on it's design and implementation of a distributed file system.
- *A Comparison of Approaches to Large-Scale Data Analysis* paper provided benchmark testing of two designs, including one similar to the Google File System.
- Both papers had some form of benchmark testing data.
- Each paper included architectural elements of their design.
- The comparison paper generally describes how the different systems work, whereas *The Google File System* paper was very detailed.

STONEBRAKER TALK
"ONE SIZE FITS NONE"

Data Warehouses

- Column store – 2x faster than row stores

OLTP – Transaction Processing

- Lightweight transactions
- Main memory transactions in the future

No SQL

- Key-Value
- Record
- Big Table Clones
- JSON Stores
- No Standards

Complex Analytics

- Business Intelligence
- Predictive Models
- Data Clustering
- Column store or Arrays – Not tables

Streaming

- OLTP engines stream real time
- Processing engine

Graph Analytics

- Simulate Column Store
- Simulate Array
- Graph Engine

SUMMARY

- The Google File System is effective in achieving it's goal of many concurrent process's running while maintaining fault tolerance.
- Disadvantages in this system include:
  - Large amount of energy consumed by the many machines.
  - Overhead to recombine the output into a single file greatly affects performance as shown in the comparison paper.
  - Input files must be scanned in their entirety compared to DBMS files which uses sorting.
- Stonebraker's talk clearly emphasizes that "one size fits none"
  - Depending on the market, a different approach is needed. Ex. Graph Analytics would look to simulate column and array stores.
  - The Google File System design is specific to their needs and was designed around such.
- Stonebraker believes vertical stores or specific applications will be developed in the future and row stores would be good for none of the markets.