Big Data Paper

By Alexander Richin

Date: March 7, 2017

# Bigtable:

# A Distributed Storage System for Structured Data

Andrew Pavlov

Brown University

Pavlo@cs.brown.edu

&

# A Comparison of Approaches to Large-Scale Data Analysis

Fay Chang
Google, Inc.
Fay@google.com

&

# One Size Fits All - An Idea Whose Time Has Come and Gone (2005)

Michael Stonebraker
Brown University
Database Group

# Main Idea - Bigtable

- A compressed, high performance, & proprietary data storage system
- Built on:
  - Google File System
  - Chubby Lock Service
  - SSTable (log-structured storage like LevelDB)
  - Etc.
- May 6, 2015
  - Public version of **Bigtable** available as service

# Bigtable Implementation

- 3 Major Components -  Bigtable Implementation
  a. A library that is linked to every client
  b. One master server
  c. Many tablet servers
- Tablet Location
  a. Three-level hierarchy analogous to store info.
- Tablet Assignment
  a. Each tablet assigned to 1 tablet server at a time
- Tablet Serving
  a. Persistent state of tablet stored in GFS
- Compactions
  a. Shrinks memory usage of tablet server
  b. Reduces amount of data read from commit log if server dies

# Analysis of Bigtable

- Description - A distributed system for storing structured data @ Google
- What  users like:
  - Performance & high availability
  - Scale capacity of clusters via adding more machines as resource demands change
- Difficult for new users to get adjusted to
  - If you are used to Relational Databases
- However, Google has demonstrated Bigtable’s success
- New Features (coming soon!):
  - Support secondary indices & infrastructure for building cross-data-center replicas
    - Multiple master replicas
  - Serviced to product groups
    - No need for maintenance of clusters
- As service clusters scale:
  - Deal w/ more resource-sharing issues
    - API, Implementation

# Main Idea (MapReduce vs Parallel DBMS)

- “Cluster Computing” - Constructing a data structure via large # of low end processors rather than smaller sets of high-end servers.
- MapReduce:
  - Attractive, simple, expresses sophisticated distributive programs
  - Data can be in arbitrary format
- Parallel DBMS
  - Vertica, Oracle, Teradata
  - Robust high-computing platforms
  - Data conforms to well-defined schema
- Main Idea:
  - “The purpose… to consider these choices, and the trade-offs they entail”

# Implementation

- Architectural Elements
    - Schema Support
    - Parallel DBMS req. data  to fit in relational paradigm
        - Rows / Columns
    - MR does not
        - Most popular implementation of MR framework - Hadoop
  - Indexing
    - MR does not have built in indexing
    - Parallel DBMS - hash / B-tree
  - Data Distribution
    - Parallel - use data distribution & location to advantage
    - MR - must do manually
  - Node configuration
    - Hadoop, Vertica, DBMS-X
    - Implemented on 100 node cluster

# Analysis

- MapReduce
  - Very simple to use
  - Better for smaller projects
  - Most flexible
  - Better Fault Tolerance
  - Well suited for development env. with:
    - Small # of programmers
    - Limited application domain
  - Not appropriate for long-term / larger-sized projects
    - Lack of constraints
- Parallel DBMS
  - More complex
  - Harder to understand
  - Not as flexible as MR
  - Better Data Distribution

# Bigtable vs. MapReduce / Parallel DBMS

- Comparisons:
    - All three deal with having a structured data storage
    - Deal with large amounts of data
    - Each is good in its own way
        - Depends upon what you use it for
    - Compress files and store them in an efficient manner
    - Can perform similar tasks to each other
    - Each is specified for its own purpose and has it’s  trade offs
        - Some perform certain tasks more efficiently than others
        - One is not better than the other
    - Implementation is similar

# Michael Stonebraker’s Main Idea

- “Dead on our feet” (80’s & 90’s)
    - B/c we believed:
        - “One Size Fits All”
            - Engines are too diverse
            - We have moved on
                - “One size fits none”
- We live in very interesting times
    - Good for DBMS researchers
        - New ideas (NVRAM, LLVM, Xeon/Phi)
    - People should explore DBMS

# Advantages vs. Disadvantages

## Bigtable Advantages:

- Substantial flexibility
  - Designing own data module
- Removes:
  - Bottlenecks
  - Inefficiencies

## Large-Scale Data Analysis:

- Uses a significant amount of less energy
- Large user community
- Easy to set up

## Parallel DBMS

- Can handle massive amounts of data
- Not so flexible
  - Follows structure
- Difficult to understand
  - If you’re new

## MapReduce:

- Simplest to use
- Most flexible
- Very efficient
- Only good for smaller projects
  - Less programmers