A Comparison of Approaches to Large-Scale Data Analysis

(Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker)

Bigtable: A Distributed Storage System for Structured Data

(Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, RObert E. Gruber)

Reyhan Ayhan, 10/20/19

What is Bigtable?

- The paper discussing Bigtable and its relevance as a storage system for structured data generally overviews what Bigtable is, how it works, and what it serves to the world of storage
- Uses a simple data model that supports dynamic control over data layout and format
- Bigtable is built upon several other pieces of Google infrastructure
    - Uses distributed Google File System (GFS) to store log and data files
- Operates on a shared pool of machines, Bigtable processes often share the same machines with processes from other applications
- Bigtable is said to reliably scale to petabytes of data and thousands of machines while still being used for a variety of workloads
    - Throughput oriented batch-processing jobs to latency-sensitive serving of data to end users
- Bigtable is essentially a multidimensional sorted map that is indexed by a row key, column key, and a timestamp
- Relies heavily on a persistent distributed lock service known as *Chubby*
- Chubby consists of five active replicas, uses the Paxos algorithm to maintain replicas, provides a namespace that includes directories

How is this storage system implemented?

- *The implementation contains 3 major components:*
  - A library linked into every client
  - A single massive server
  - Several tablet servers: dynamically added/removed from a cluster
- *Master's responsibilities:* assigning tablets to specific tablet servers, recognizing the addition and expiration of tablet servers, balancing tablet-server load, and garbage collection of files, and changes in the schema
- *Tablet server manages a set of tablets:* handles read/write requests and splits overgrown tablets
- Clients communicate directly with tablet servers when reading and writing
- # of tables are stored within each cluster, each table consists a set of tables

Bigtable: Analysis

- The paper regarding Bigtable described the system to be organized and its implementation strategies resembling a database (made up of rows and columns)
- It was impressive to know that projects like Google Earth and Google Finances use Bigtable as a source of storage.
    - Demonstrates the reliability of both the platform the system was built off of and how far its come

```
// Open the table Table *T =
OpenOrDie("/bigtable/web/webtable");

// Write a new anchor and delete an old
anchor RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com"); Operation
op; Apply(&op, &r1);
```

- The commands and script style that go into writing to Bigtable seem very direct while maintaining details.
- There seem to be a variety of more or less complicated ways for users could manipulate the data

Comparison of SQL DBMS and MapReduce

- The purpose of the Large-Scale Data Analysis paper is to demonstrate and evaluate the differences between the MapReduce paradigm and the preexisting framework in parallel SQL database management systems (DBMS)
- The paper is an overview of how MR and DBMS achieve parallelism by dividing datasets to be utilized into partitions that are allocated to specific nodes
- **MapReduce (MR):** new, simple, written by a user to process key/value data pairs, input data sets exists as a single or several collections of partitions in the distributed file system
- **Parallel DBMS:** existed since the 1980s, supports standard relational tables, nodes receive necessary data that calculate the preliminary aggregate function

Implementation of SQL DBMS and MapReduce

SQL DBMS	MR
- Most tables are partitioned over nodes in a cluster - The system uses an optimizer that translates SQL commands into a query plan - A datatable's content could be replicated on all nodes if classified as small - A datatable's content could be distributed across multiple nodes if classified as large - Requires data to fit into the relational paradigm of rows and columns -Separates the schema from the application, stores it in a set of system catalogs that can be queried - Most functionality is built-in	- Written by a user to process key/value data pairs - Input data is stored in a collection of partitions - Made up of two phases: The Map function → reads, filters, transforms, partitions a "split" function The Reduce process → combines the records, writes records to an output file, forms part of the computations final output - No requirement to adhere to a scheme - All structure must be built into the Map & Reduce programs - No knowledge of programming rules = allows for bad data - MR model → works best with developing environments with small numbers of programmers and limited application domain - Performs on a more manual and customizable basis

Analysis of SQL DBMS and MapReduce

- The paper served to prove that although MapReduce is a new and exciting paradigm for large-scale data analysis, DBMSs can do the same
- The discussion of the architectural elements, schema support, indexing, programming models, data distribution, execution strategies, flexibility and fault tolerances were all in favor in DBMS in respect with what MR can offer and the purpose it can serve in the Big Data community
- Particularly in elaborating on the comparison of *flexibility*, the MR model is arguably more flexible due to its generality and user customized functions and procedures but the authors continued to side with DBMS because it still strives to improve the flexibility of DBS

Bigtable vs. SQL DBMS and MapReduce

- The ideas presented in both the Bigtable project and the Large-Scale Data analysis paper vary for several reasons
- The Bigtable project was surrounded around the idea of storage and maintaining a strong data storage system with a *simple data model* making it easy for a user can easily access their data
- Bigtable ISN'T a database but resembles one
- On the other hand, the paper comparing SQL DBMSs and MapReduce compared both paradigms using a variety of important factors (schemas, flexibility, etc.) in which both achieve parallelism but in different ways
    - SQL DBMS requires data to fit into a relational paradigm (rows and columns like Bigtable)
    - MR has no scheme, all structure is manually implemented into the functions

Stonebraker Talk

- “One size fits all” approach within relational databases was no longer relevant
  - This was based on a sample of 3 in Stonebraker’s paper in 2005
- RDBMS could be the universal/best solution to database model because it provided abstract data types and referential integrity
- “One Size Fits None” (2015):
  - Data Warehouse market: all major vendors currently or will soon have column stores, riw stores will disappear because column stores have been proven to be faster
  - OLTP market: transaction processing the databases aren’t all that big, involves lightweight transactions
  - NoSQL market: 100+ vendors with a potpourri of data models and architectures. Key-value stores, record stores, Big Table clones, JSON stores
- Business analytics use data warehouses: can simulate things such as standard SQL analytics
- Stream processing engines have some market shares: they have a workflow of processing with SQL-like characteristics
- Can add streaming to an OLTP engine much simpler than adding persistance to a streaming engine
- Graph analytics stimulate in column stores, array engines, graph engines

Advantages and Disadvantages - BigTable - SQL DBMS vs. MR - Stonebraker Talk

Advantages:

- Bigtable supports dynamic control over data form and layout similar to that of MapReduce → functions are customizable
- Bigtable is multidimensional sorted map but isn't a database → still utilizes the idea of rows and columns but removes SQL and queries → slowly moving in Stonebraker direction

Disadvantages:

- Share machines with other applications → not an entirely unique platform, works off of other ideas. Similar to DBMS (a traditional approach)
  - Unlike MR
  - Unlike RDBMS
- One size does not fit all but in the comparison paper, DBMS is said to excel in most areas when there needs to be a diversity of engines because of the great variety of markets (the data warehouse markets, OLTP market, NoSQL market)