# JESSICA RIEGER OCTOBER 16TH, 2016

## *Bigtable: A Distributed Storage System for Structured Data*

By: Fay Chang, Jeffery Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber

Google, Inc: OSDI ’06

## *A Comparison of Approaches to Large-Scale Data Analytics*

By: Andrew Pavlo, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Erik Paulson, Alexander Rasin, and Michael Stonebraker

SIGMOD ’09

## ***“10 Year Test of Time”***

Michael Stonebraker’s ICDE 2015 Talk

# *BIGTABLE: A DISTRIBUTED STORAGE SYSTEM FOR STRUCTURED DATA*

- Bigtable is a good way to manage large amounts of structured data through its distributed storage system.
- Bigtable is a model that can be represented as a tree structure that has 3 levels.  A root tablet, other metadata tablets, and user tablets.
- Clients have dynamic control over the data layout and format.  Clients can decide, using the simple model, about the locality of data, such as the schema of their data as well as whether or not their data is in memory or stored on the disk.
- Wants to serve throughput-oriented batch-processing jobs, latency sensitive jobs, and jobs that require huge amounts of data storage.
- Bigtable strives to achieve wide applicability, high-performance, scalability, and high availability.

# HOW BIGTABLE IMPLEMENTED

- Bigtable data storage can be represented by as a “sparse, distributed, persistent, multi-dimensional map.”
- Bigtable is a flexible, high-performance solution that is used by Google and other companies. Its beginning to be sold as a service so customers do not have to manage their own server clusters.
- Bigtable uses Google File System, MapReduce, Chubby, and Google SSTable to supplement its operations.
- Data is sorted by column name, then grouped into column families. Families can be compressed to reduce space usage and grouped into locality groups that are readily available to the user. Reduces number of necessary accesses to the disk.
- The data is spread across multiple tablet servers that are all connected to a master server and a library. Tablets can be added or deleted dynamically as needed.
- The structure of this model is a map that is indexed by row key, column key, and timestamp. Each row range for a table is called a tablet and is stored on its own server. By choosing the correct range, a client can keep relevant information easy to access.
- The user can use the API to delete tables or column values or manage the structure and access control rights.
- To improve its performance, uses 2 level compression, multiple types of caching, bloom filters(to as a client what data present before accessing disk), and a single log file of commits.

# MY OPINION

- Flawed by its dependency on Chubby's availability.
- Though the API may seem simple to the developers, it seems much more complex than the relational model that is so popular today.
- I question why are we going back to showing the user data storage when the whole point of the relational model was abstraction into a simple table.
- Does seem like a flexible system that trades time for space which is a plus. Has many ways to implement time efficiency.
- Has a recovery system using memtable to recover lost tablets.  This plus the storage of indexes in many places makes it reliable
- New system so faces many unexpected problems. Needs time to mature.
- Good because can be scaled as the company develops over time. (It grows with the enterprise that it serves.)

# A COMPARISON OF APPROACHES TO LARGE-SCALE DATA ANALYTICS

- Compares the performance of DBMS-X, Vertica, and MapReduce (Hadoop) systems.
- Concluded DBMSs were harder to install and configure initially however once up and running they were easier to manipulate.
- MapReduce program is inefficient due to lack of structure, slow start-up time, lack of database level rules making it hard to share among developpers, and its excessive use of CPU processing/energy.
- In terms of the user interface, Hadoop was easier to install and configure, but the DBMS were easier to interact with thanks to the several tools that go with them and the ability to query the system catalog.
- The DBMSs (DBMS-X and Vertica) outperformed Hadoop in almost every test due to their short/non-existent start-up time, the fact that they don’t need to combine results into one file, their use of compression effectively, and their execution strategies to reduce network traffic.
- DBMSs don’t deal with failures effectively, and the size of buffer pool and sort heaps are too small for current systems.

# COMPARISON PAPER'S TESTS AND RESULTS

- The implementation of the comparison of the three systems was executed through a series of trials.  For each system they performed the test three times and took the average performance time. As well, they observed how the performance was altered as the test grew in size (namely through the number of nodes involved and the amount of data.)
- By playing with the configuration of each system and adjusting the capabilities, they tried to make each system perform its best. Whether it be writing code for Hadoop or adjusting the distribution of the processing, they did everything they could to improve the results.
- The tests that they ran were a join task, aggregation task, selection task, and the UDF aggregation task. MapReduce was slow due to the amount of data it must transfer, the map and reduce functions it has to execute, and its limitation to the speed it can read off the disk.
- The DBMS outperformed the MapReduce system in all except the UDF aggregation task.

# COMPARISON OF THE TWO PAPERS

- In my opinion, I think that the Database management system is still the way to go for a database model. The reason for this is because it has many more years of foundation and though it presented its own problems, it out performed the Map Reduce model Significantly.
- Perhaps once the Map Reduce model gains tools and a simpler interaction system, then I will consider it more thoroughly as a model to use.
- We must address the lack of failure recovery in Database Management Systems.
- Overall, this experiment was useful to explore the faults in both systems and push both to make the necessary user interface and model changes.

# COMPARISON OF THE TWO PAPERS

- Both stress the importance of a system’s performance and the trade-offs between time and space.
- Emphasis is placed on the structure of the systems and the importance of this structure in the facility of data access.
- One paper supports the use of MapReduce, while the other points out many of its flaws. Both acknowledge that MapReduce is far from a perfect system.
- In each of the papers, the user interface and ease of use is addressed. In the first, the user interface seems more complex than the Database Management System interface that the second promotes.
- Both papers present their ideas on how to address large amounts of data. Each approaches big data from a clustered distributed server approach and use compression.
- The first paper uses many different programs to manage the data whereas the second promotes DBMS which is a single system.

# MAIN IDEAS OF STONEBRAKER TALK

- The Relational Database Management System Is the essentially useless. It is the "one size that fits none."
- RDBMS are too heavy-weight, too slow. SQL is too slow and incapable of doing complex analytics such as graph analytics.
- The popularity of the column store, the speed of the column stores, the NoSQL market (potpouri of different models very different from RDBMS), the battle for more complex data management and statistic programs combined are all forcing RDBMS into oblivion.
- Warehouse storage, Transaction processing, analytics, streaming, and graph analytics markets are all using column store and other technologies so row-store is becoming obsolete.
- Networking in DBMS are the bottleneck which conflicts with the needs of our world and its markets today. Other systems deal with networking much better so they are gaining popularity.

# ADVANTAGES AND DISADVANTAGES OF BIGTABLE

- Advantages: Bigtable is abandoning the traditional relational model.
- It implements a column-based storage and is abandoning much of the relational model's abstraction in favor of allowing the user to customize the locality and layout of their data.
- The model supports complex modifications by the user that are clearly being demanded by markets described in the talk.
- Disadvantages: The Bigtable system still has row storage to some degree so that may hinder its ability to keep up with the complex and fast systems being developed today.
- It is reliant on several other Google Products, thus if a different product is dominant in providing one of the service it needs, it may have trouble related to interoperability.