Big Data Paper Summary

By Simoni Handoo
03/05/2017

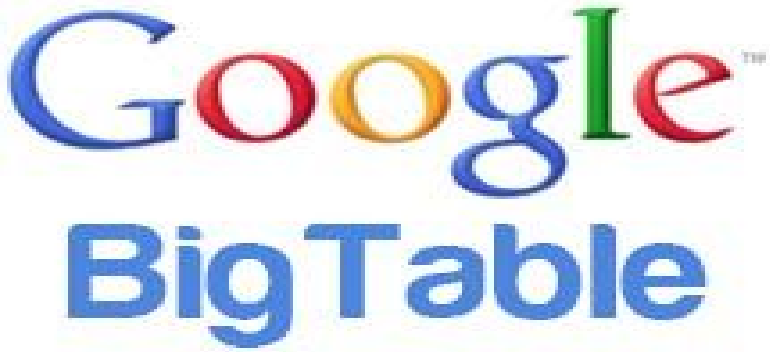
Paper Chosen: Bigtable
Comparison Paper: A Comparison of Approaches to Large-Scale Data Analysis

BIBLIOGRAPHY:

1. Petuchowski, Posted By Ethan. "With Pith." *Bigtable Paper Summary - With Pith*. N.p., n.d. Web. 12 Feb. 2017.
2. Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hseih, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. *Bigtable: A Distributed Storage System for Structured Data*. Publication. Google Inc, 1 June 2008. Web. 12 Feb. 2017. <http://labouseur.com/courses/db/papers/chang.osdi06.bigtable.pdf>.
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*. Tech. N.p.: n.p., n.d. Print.
4. *Michael Stonebraker on his 10-Year Most Influential Paper Award*. Michael Stonebraker, n.d. Web. 1 Mar. 2017. <http://kdb.snu.ac.kr/data/stonebraker_talk.mp4>.

THE MAIN IDEA OF BIGTABLE

- BigTable is a well compressed data storage system that was built by the Google File System. It is a database management system which allows you to define tables, write and update data, and run queries against the stored data.
- It is quite similar to a relational database in a way where it supports a simplified model of the RDBMS structure.
- Its development had began in 2004 and BigTable is now used by multiple Google applications, and example would be MapReduce.
- The main idea behind BigTable is the separation of storage and logic.
- BigTable is able to handle large structured data by building indexes on it.
- It has a simple design, therefore it can be used in several applications and it is quite easy to maintain.
- Since Google has terabytes/petabytes of data in its system, an innovation such as BigTable was needed since traditional centralized database systems are not known to scale well with such huge data set.



Implementation of Bigtable

The main idea behind implementation is separation of storage and logic. Storage is handled by GFS, which provides reliability, scalability and availability for free. This separation enables each tablet having only one server responsible for it. There are 3 major components to the model implementation:

1. ***Library that is linked into every client:*** The client library caches tablet locations. If the client does not know the location of a tablet, or if it discovers that cached location information is incorrect, then it recursively moves up the tablet location hierarchy.
2. ***One Master server:*** The master manages tablet assignment, server cluster membership changes (with help from Chubby), load-balancing of tablets among servers, garbage collection, and schema changes. Clients only need to communicate with the master to find the relevant tablet servers.
3. ***Many tablet servers:*** A tablet is a complete alphabetical interval of all the data in a table. The tablet server handles read and write requests to its tablets, and splits tablets when they get too big. Each METADATA row stores approx. 1KB of data in memory. There is a generous limit if 128 MB METADATA in tablets. BigTable uses Chubby to keep track of tablet servers.

My Analysis of the idea and it's implementation

(Row, column, Time) → String

The paper informs us How google uses 'chubby' and 'gfs' (google file system) centrally to shape services on top of it.

In a traditional manner, databases do a lot of analytical processing on the stored data, and plain data is presented through key value stores. The master and tablet server management is similar to the master and chunk server in the GFS system. just like in gfs, the client don't rely fully on the master and interact directly with the servers managed by the master.

I believe that the design of distinctly isolating the tablet server's tasks from the back-end storage is a smart idea. This would let the developers scale the system by adding more tablet servers as the data increases exponentially. The master can also be made to take into account the location of the tablet in gfs to assign tablet servers. The most intriguing aspect of bigtable's implementation is the way it is designed to be so simple. The bigtable's data model supports only a single "Big Table" that is organized in rows of attributes with a specific timestamp. With such a simplistic database model, all that is left to solve is efficiently storing and redeeming data on a distributed group of machines.

A comparison of approaches to large-scale data analysis- Main Idea

- This scholarly paper focuses on a comparison of approaches to Large-scale data analysis, featuring MapReduce and Parallel DBMS.
- MapReduce consists of only 2 functions 'Map()' and 'Reduce()' that are processed by a user to extract values.
- Parallel DBMS on the other hand stores data on multiple machines which is enabled by parallel executions.
- The paper tells us that parallel dbms provides much better performance, whereas mapreduce provides a much better ease/simplicity. It's loading time is also better.
- At the end, the paper presents thoughts about the test results. The main idea behind writing this paper is to provide
- database designers a clear and detailed comparison of MapReduce and parallel DMBS so that the database designers can think about the trade-offs objectively and figure out what would be the best way for them to construct their database.
- An example would be that the mapreduce programmers must write the program so that it can extract and convert the data and store it in memory for execution. This would require a lot of work from the programmer, which is something a database designer should know about.
- That is the main purpose of the paper- to inform the readers about the pros and cons of both mapreduce and parallel DBMS

A comparison of approaches to large-scale data analysis- Implementation

- Indexing: MapReduce frameworks don't provide in-built indexes. Since the mapreduce model is basic, the programmers themselves implement any indexes that they may need to speed up access to data inside the application.
- Programming model: in terms of programming in MapReduce, the programmer must write algorithms in a low-level language in order to execute record-level manipulation.
- Data distribution: First of all, the map function in MapReduce scans the documents and creates a histogram of frequently occurring words. The documents are then pass down to a Reduce function that groups files by their site of origin.

A comparison of approaches to large-scale data analysis- My Analysis of the main idea and its implementation

I believe that the paper established its analysis in a quite well-rounded manner and it used a simplistic way to present the flow of the two approaches to large scale data. I liked how the paper first discussed the similarities and differences of how mapreduce and parallel dbms handle large-scale data processing from a conceptual level.

The main highlight of this paper was how it ran a comparatively full-scope juxtaposition of parallel dbms and mapreduce implementation. The paper also provided the readers with a discussion of what a user should examine while deciding which system to use, as well as what could be done to enhance/boost each of the systems.

In my interpretation, i feel as if the paper was a little subjective. I believe the paper seemed to be a little biased to the parallel dbms's side. Most of the analysis done in the paper is dedicated to prove how parallel dbms can surpass MapReduce. This is what i think was one of the drawbacks of the paper, however it could have been simply because parallel dbms is merely a better approach to large-scale data analysis than mapreduce. We should have been given a little more information regarding the popularity of MapReduce and the fact that even though Parallel dbms is more efficient than MapReduce, MapReduce is more dominant and has gained more popularity over the years.

My analysis of the ideas and implementation

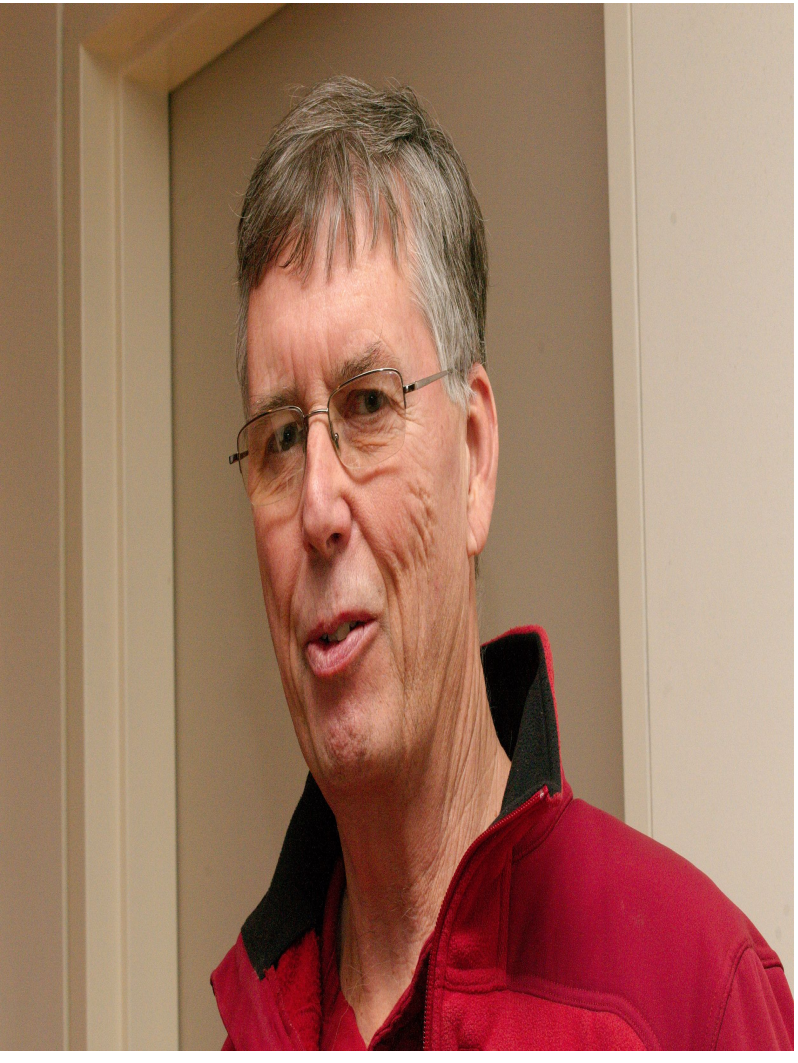
I believe that the paper established its analysis in a quite well-rounded manner and it used a simplistic way to understand the flow of the two approaches to large scale data. I liked how the paper first discussed the similarities and differences of how mapreduce and parallel dbms handle large-scale data processing from a conceptual level.

The paper provided the readers with a discussion of what a user should examine while deciding which system to use, as well as what could be done to enhance/boost each of the systems.

In my interpretation, I feel as if the paper was a little subjective. I believe the paper seemed to be a little biased to the parallel dbms's side. Most of the analysis done in the paper is dedicated to prove how parallel dbms can surpass MapReduce. This is what i think was one of the drawbacks of the paper, however it could have been simply because parallel dbms is merely a better approach to large-scale data analysis than mapreduce. We should have been given a little more information regarding the popularity of MapReduce and the fact that even though Parallel dbms is more efficient than MApReduce, MapReduce is more dominant and has gained more popularity over the years.

Comparison between the 2 papers

- Both papers discuss handling large sets of data
- Both papers give a clear main idea, implementation and a discussion/conclusion for each of the methods used for handling large sets of data
- The Papers inform us that BigTable can be used with MapReduce. **“**set of wrappers that **allow a Bigtable to be used both as an input source and as an output target for MapReduce jobs.”**
- The comparison paper talks about the differences and trade-offs between “Parallel DBMS” and “MapReduce”, and is a little biased towards MapReduce. BigTable Paper on the other hand focuses principally on the implementation and main idea of BigTable. There is no comparison.



Main Ideas of the Stonebraker talk

- From 1970-2000, RDBMS (SQL) was the answer to everything
- The idea was to make relational databases a model where “one size fits all”
- However, by 2005 the world realized that wouldn’t practically work. The case was actually that “One size fits none.”
- Basically talks about how SQL Server, DB2 etc. are bad systems
- Column stores > Row stores → much faster
- “No SQL market” has 100 or more vendors.
- In the future data scientists will run regressions, singular value decomposition, data clustering etc. All of these are defined on arrays, NOT tables. All of this can be simulated on SQL but it is extremely slow.
- Graph analytics not possible with Row stores (RDBMS)

Advantages and disadvantages of the "BigTable" paper

Advantages	Disadvantages
The simplicity of the design. It was made simple to reduce the problem into a manageable size	Long Queries required - time consuming
It is a vital tool for storing huge amounts of data	It is an exclusive system (only used for Google file systems)
It is easily scalable	Queries take some time to complete because data in BigTable is stored on many servers within multiple locations
It is a very adaptable system (flexible)	