Big Data Paper Summary

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)

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)

Michael Stonebraker on his 10-Year most Influential paper Award at ICDE 2015

Liliya Yerassilova 03/11/2016

Bigtable: A distributed Storage System for Structured Data

- Main Idea:
- Bigtable is a distributed storage system that was designed for storing petabytes of data across thousands of servers. Many projects at Google store data in Bigtable, such as Google Earth, finance.
- The paper discussed the main purpose of Bigtable, which is primarily to solve scale problem, in addition to make managing of storage easier.
- Bigtable resembles a database, but it is not really one. It is like a 3-dimentional map that includes not only rows and columns, but also timestamp, which is used to index multiple versions of the same data. Rows represent arbitrary strings, that are maintained in lexicographic order. Columns are grouped into column families to form a basic unit of access control.

Bigtable: A distributed Storage System for Structured Data

- How the idea is implemented:
- Bigtable is built on several other pieces of Google infrastructure. It uses Google File System to store persistent data (also called SSTable) and Chubby, which is a lock service.
- There are 3 major components of implementation: a library linked into every client, one master server, and many tablet servers, which can be dynamically added and removed.
- The master server assigns tablets to tablet servers to balance tablet-server load , collects garbage, and performs metadata operations.
- Tablets hold a range of rows, Chubby is used to keep track of all these tablet servers and master server makes sure each tablet has some load of work to do.

Bigtable: A distributed Storage System for Structured Data

- Analysis of the idea and its implementation:
- To improve Bigtable, there were a few refinements: locality groups, compression, bloom filters, etc. Locality is used to enable more efficient reads by grouping multiple column families into one locality group. Bloom filters allow us to ask whether an SSTable contains any data for a certain row or column.
- Bigtable can be used for many different purposes storing large amounts of data and being flexible to scale at the same time. The idea is great since it addresses the issues we face. However, the system is somewhat vulnerable to failures, rebalancing algorithm to different tablets is not perfect, compaction reduces recovery period for tablets. Even though, it is easy to scale, the ratio of tablets in use does not proportionately translate into efficiency, therefore, speed.

A Comparison of Approaches to Large-Scale Data Analysis

- The main ideas:
- To understand the differences between the MapReduce approach to performing large-scale data analysis and parallel database systems.
- What the trade-offs between the choices? DBMSs require a well-defined schema whereas MR permits data to be any format. MR is simpler and easy to install, whereas DBMSs support relational data model and have higher start-up costs. But this might not necessarily mean that MR is the choice to go with.
- Overall, the conclusion of the paper says that DBMSs outperform many data analyses.

A Comparison of Approaches to Large-Scale Data Analysis

- How the ideas are implemented:
- To understand which system is better, or in which case one should prioritize one over the other, performance, flexibility, fault tolerance should be considered. MR is considered more flexible than DBMS. MR also provides more sophisticated tolerance to failure. For execution strategy, MR is vulnerable because it has to separate between Map and Reduce functions that might not always work smoothly.
- To evaluate performance, Hadoop was taken as an example of MapReduce as its most popular open-source framework. DBMS are separated into DBMS-X and Vertica.
- DBMS-X outperforms Hadoop in most respects concerning performance: data loading, task execution, and analytical tasks, such as selection, aggregation, join, and UDF aggregation.

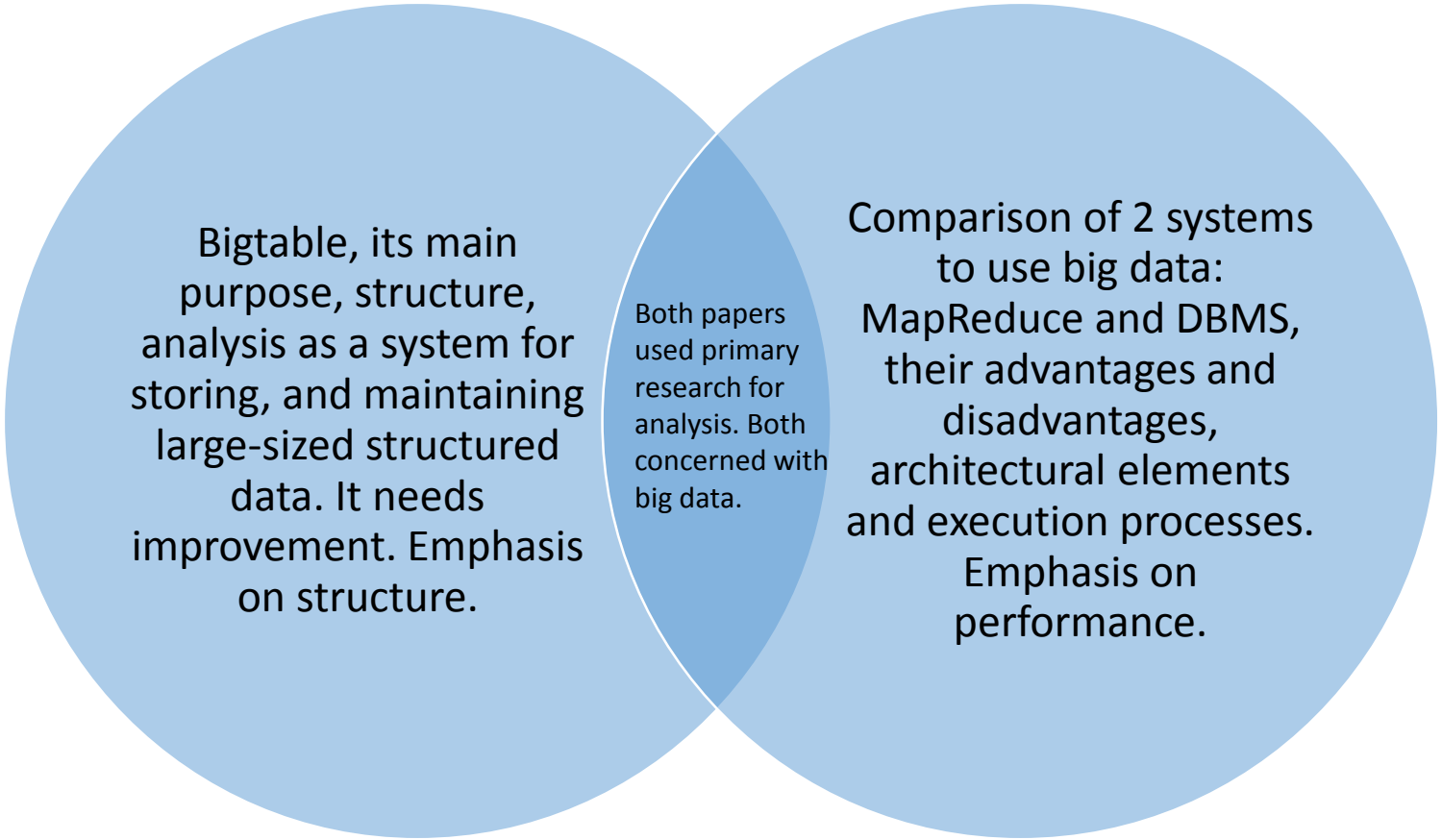
A Comparison of Approaches to Large-Scale Data Analysis

- Analysis of the ideas and their implementation:
- It is difficult to say which system is better, but the paper provides a deep analysis on different types of performances that each system can do and how well they execute. If sharing is needed, then DBMSs are better, because MR doesn't have built-in indexes.
- For example, a procedure to load data from the nodes' local files into each system's internal storage representation shows that Hadoop clearly outperforms both DBMS-X and Vertica, which is all explained in detail as to why. However, selecting task is done better by parallel DBMSs by a significant factor. For aggregation and join tasks, DBMSs outperform Hadoop as well. UDF Aggregation is performed better by Hadoop, despite DBMS's seemingly easy implementation.
- Overall, DBMS provides the same responses with far fewer processors, which uses far less energy, which might provide savings despite higher start-up costs.

Comparison of the ideas and implementation of the 2 papers

Bigtable: A distributed Storage System for Structured Data

A Comparison of Approaches to Large-Scale Data Analysis



The Main Ideas of the Stonebraker talk:

- Relational DBMS models were used for all sorts of projects: small, medium, and large, and for any purposes. It was “one size fits it all.”
- In his paper, they proved (using a sample of 3 only) that Relational models do not “fit it all”
- 10 years later, in 2015 he claims that “One size fits None,” DB2, Oracle, Sequel Server are good for nothing.
- Warehouse markets: have column store, they are 2 orders of magnitude faster than row stores.
- OLTP: row stores are heavy-weight transaction system, light-weight transactions are better
- NoSQL Market: about 100vendors; no standards
- Complex Analytics: business intelligence through warehouses, can do standard data analytics. More complex analytics will run regression analysis and other things that are not available in SQL, which you can simulate on SQL, it is very slow and difficult, though.
- It is going to be either column stores or array stores, but not the traditional rows stores (DBMSs)
- Streaming Market (from 2005): is not based on traditional row stores. Stream processing engines have some market share.
- Graph Analytics: by simulating in a column store or in an array engine. Not traditional Row stores
- Apparently, there is a huge diversity of engines, they have specific applications, but traditional row stores (traditional DBMSs) are not good in any of these applications.
- Processor diversity, memory will become much bigger, networks will become much faster, and in general many new implementations will evolve.

Main idea of Bigtable in the context of the comparison paper and the Stonebraker talk

Advantages:

- The Bigtable paper is very specific and in great detail explains the structure, execution process, and evaluation of performance for only Bigtable. It provides many examples and graphs that enable reader to understand and visualize some concepts.

Disadvantages:

- Bigtable explores only one way of dealing with large-sized data, whereas, the comparison paper compares two different models and discusses their advantages and disadvantages. The talk offers a completely different point of view, saying that DBMSs are not good for anything and that there are many other engines that should be implemented for specific applications. Bigtable offers only one point of view and does not inform readers about other possible choices of dealing with Big Data.