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 {fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com Google, Inc.

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel Abadi, David DeWitt, Samuel Madden, Michael Stonebraker

Tenkorang Darko

3/15/16

BigTable: Main Idea

- Invented by Google Inc. in 2006
- Bigtable is a distributed storage for managing structured data
  - Designed to scale to a very large size
- Does not support full relational data model (not an sql database)
  - provides simple data model that supports dynamic control over data layout and format
  - allows clients to reason about the locality properties of the data represented in the underlying storage
- Self Managing, Very Flexible, High-performance,wide applicability, scalability, high performance, and high availability
- Gives dynamic control over data layout and format
- Used for many Google products
  - Google Finance, Google Analytics, Personalized Search, Google Documents, Google Earth…
- Designed to scale petabytes of data and thousands of machines

BigTable: How It Is Implemented

- **Three Major Components:** (1) One master server, (2) Many tablet servers, (3) Client library
  - (1) - Responsibility: Balancing tablet server load; Assigning tablets to tablets servers; Detecting addition/expiration of tablet servers, Garbage collection of files
  - (2) - Responsibility: Manage set of tablets; Handle read/write requests to tablets; Split tablets that have grown too large
  - (3) - Responsibility: Does not rely on the master for a tablet location information; Communicates directly with tablet servers for reads/writes
- **Tablet Location - (5.1):** Three level hierarchy (1) Chubby File, (2) Root Tablet, (3) Metadata Tablets/Table
  - (1) - Is used as a name to access the Root Tablet. The Chubby contains the location of the root tablet.
  - (2) - The Root Tablet contains the location of all tablets in a Metadata table.
  - (3) - Contains the location of user tables/tablets with assigned rows and keys.
- **Tablet Assignment - (5.2):**
  - Master Server
    - Keeps track of live/unassigned tablets; Detects the status of each tablet server; Reassign tablet server when no longer being used; Tablets are assigned to 1 tablet server
- **Tablet Serving - (5.3):** Updates committed files to a commit log; Writes recently committed updates in the meltable; Older updates are stored in sequence of SSTable
- **Compactions - 5.4:** Used to control memtable size. Consist of Minor, Merging, and Major compaction.
  - Minor Compaction: Transforms the meltable to an SSTable and help reduce memory usage
  - Merging Compaction: Merges SSTables to exactly one SStable
  - Major Compaction: Ensures that an SSTable contains no deleted information or data.

BigTable: Analysis of Idea and Implementation

- Having three major components each with its own set of different functions is a great way to keep things organize and clean. If there is a problem anywhere in the system, it will be easier to pinpoint the component causing the issue.
- I love how Google took a NoSQL approach by making it different from a relational database model. Although it contains similarities of a relational model, BigTable offers many more options to data management and database design by giving flexibility to coders.
- I very much dislike the fact that Google has kept this software proprietary. BigTable is a very cool software that should be made public.
- The Timestamp feature added to the row and column is a great way to keep duplicated data unique.
- Although BigTable uses its own language called CQL, it will be cool to also have it support SQL.

Main Idea: Comparison Paper

Main Idea: The articles compares MapReduce to Parallel DBMS systems in relation to large datas.

- **Mapreduce**
  - **MapReduce tries to keep things simple by having only two main Functions: Map & Reduce. MapReduce is more efficient for processing large data sets and gives the programmer flexibility when writing codes. MapReduce also don't require changes or loading of Schemas and Indexes.**

- **Parallel DBMS**
  - **Parallel DBMS are more for performances, task execution, or loading. In addition to this, analysis tasks can also be run much faster than MapReduce. Parallel DBMS gives you the flexibility of joining multiple tables through joins and subqueries.**

Ideas Implemented

- **MapReduce**
  - Has two functions which is “Map and Reduce.” The Map job is to process input data and break it down into small chunks of data. The "Reduce" stage processes the small chunks of data that was created and produces a new set of output. Data is partitioned into groups to make it easily accessible and maintainable. Although Indexes are not supported, a programmer is free to create a custom Index if needed. Also, MapReduce allows for high scalability and is very easy for a programmer to adapt to.

- **Parallel DBMS**
  - Requires a Schema and uses SQL to write queries. The functions to a Parallel DBMS are already defined such as stored procedures, user-defined functions, and aggregates. In addition to this, a built in Index is already made to be accessible if needed.

Analysis of Ideas/Implementations

- It seems like MapReduce is the better program to use in my eyes simply because of the flexibility a programmer will have. In addition to this it is easier to set up and start using than Parallel DBMS.
- I like how Parallel DBMS has a set of built in functions ready to be used by programmers.
- Parallel DBMS is much more expensive to use and maintain.
- A Parallel DBMS is better suited for structured and multiple applications
- Parallel DBMS had a significant performance advantage over MapReduce when compared
- With MapReduce Data Loading is fast and no schema is required

Two Paper Comparison

- **A Comparison of Approaches to Large-Scale Data Analysis**
  - According to this article, Mainly focuses on different approaches to Large data by using either MapReduce or Parallel DBMS. MapReduce uses a distributed file system, Functions such as Map and Reduce, and a Scheduler to make tasks more efficient. Parallel DBMS uses the standard relational tables using SQL for writing queries. MapReduce should be used for single applications while Parallel DBMS should be used for multiple applications.

- **BigTable**
  - Invented by Google Inc takes the MapReduce approach by ensuring that large amounts of Data can be stored effectively without any problems. In addition to this, coding is made easier which in turns leads to much more productiveness when coding.

Stonebraker Talker

- In the 1980's and 90's, Relational database was made as a way to solve every problem by adapting the one size fits all mentality. Soon, they realized Relational database model is not good for everything. This mentality is now dead because it left them clueless and stuck.
- One size does not fit all and one size fits none.
- Row Stores and Column Stores are very different. Column stores are faster and more popular.
- There is a NoSQL market with all Column Stores. There is also a streaming market.
- There are a huge diversity of engines all oriented toward specific verticals or applications

Overall Advantages and Disadvantages

- BigTable is NoSQL, does not support full relational data model and is very flexible.
- BigTable is a part of the NoSQL market
- It is not column store database or row store database.
- Parallel DBMS have referential integrity rule, triggers, and Abstract Data types to give more flexibility and control.