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

Google, Inc.

A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abad,i David J. DeWitt, Samuel Madden, Michael Stonebraker

This work was supported in part by NSF Grant CluE - 0844013/0844480.

Mutiat Alagbala

3/15/16

Bigtable: An Overview

- Bigtable is not a relational data model.
  - Bigtable stores data at the intersection of a row, column and timestamp.
  - The stored data can be structured or unstructured.
- Operations within Bigtable (DBMS for Bigtable)
  - DBMSs for Bigtable is an API implemented through client applications
  - Client “Applications can write or delete values ... look up values from individual rows, or iterate over a subset of the data” (207).
- Building Blocks
  - Bigtable is one part of the overall system on which google is built.
  - Chubbies are how Bigtable minimizes data loss and maintain physical data independence.
  - Each Chubby has 5 replicas, one is an active session called the “master.”
  - When Sessions are expired, the chubby “loses any locks” which are files and directories that the chubbies store.

Bigtable: Implementation

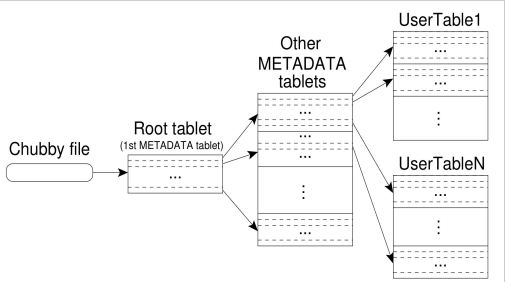

Figure 4: Tablet location hierarchy.

- “Three major components: a library, … a master server, and tablet servers” (208).
- Library of Locations
  - A Chubby has the location of the “*root tablet*” which has the location of all METADATA tablets which has the location of a set of all user tablets.
- Master Server
  - “Each tablet is assigned to [a] tablet server... The master [tracks]... the live tablet servers [their] current assignment of tablets [and] which tablets are unassigned” (209).
  - The master will kill “itself if its Chubby session expires” (209).
  - When starting up the master it: “Grabs a unique master lock”, “Finds all live servers”, Finds all tablets that are assigned to those servers and “Scans the METADATA table” for tablets that are not assigned.
- Tablet Servers
  - Write and read operations are checked by the server. Then data is changed or output respectively.
  - The three types of compaction, minor, major and merging, are used to minimize “memory usage” and irrelevant data, allow recovery of deleted data and merging data into one table, like joins in SQL.

Bigtable: Analyzed

- Bigtable’s implementation into various Google Services prove it is a fitting solution for Big Data in search engines specifically.
- The approach to the general problem, sifting through searchable data, is logical from its hierarchical structure and use of chubbies to the master operations.
- Google used or is using it so it is a practical implementation for like database problems
- One might even say it is correct, though I am not sure how Codd would justify it.

Comparison Paper: An Overview

Compare Database Management Systems (DBMSs) to MapReduce (MR) paradigm.

DBMS

- The schema is rigid and complex compared to MR schemas
- Important features are built-in for example preservation of referential integrity and compound keys are supported.
- Physical data independency
- SQL compatible
- ...and the list goes on

MR

- Schema is defined by the programmer or could be unstructured
- “programmer must explicitly write support for more complex data structures, such as compound keys” (167).
- Data must be reinterpreted by each programer to be useful information.
- “there is widespread sharing of MR code fragments to do common tasks, such as joining data sets” (167).
- ...and the list goes on

Overall MR is more simple than DBMS but operations that may seem trivial in DBMS like joins must be hard coded from scratch by each programer that accesses the data.

Comparison Paper: Implementation

To compare DBMSs and MRs the Authors used 3 different models (Vertica [DBMS], DBMS-X[DBMS], Hadoop [MR]) to test 5 different data manipulation and retrieval tasks(Data Loading, Analytics, Aggregation, Joins, UDF Aggregation). There were a few main differences between task executions on the DBMSs vs the MR

- In 3 out of 4 tasks the execution was carried out by a moderately simple SQL query (ies) on the DBMSs. On the other hand all tasks performed on the MR required more effort on the programmer's part. For example, JOIN is a function built into SQL while in order to do a JOIN in the MR program it took several phases of work to finally receive the sought after data returned.
- In every task one of the three models had a slowest load times.
    - DBMS-X had the slowest load times during execution of the Data Loading and UDF Aggregation Tasks.
    - Hadoop was the slowest to load during execution of the Grep's (Anylitics), Selection, Aggregation and Join Tasks.

Comparison Paper: Analyzed

- The point of the paper was to compare the effectiveness of MRs and DBMSs considering different types of tasks and amounts of data and to understand where one was better than the other.
- The paper confirmed that the relation data model is not the answer to all of a developer's data issues.
- In the context of big data MRs are more efficient when
  - the amount of data is larger
  - AND the task is simpler
- In the same context DBMSs are more efficient when
  - the amount of data is small
  - EVEN WHEN the task is complex.

Bigtable vs. Comparison Paper

- A semi-relational database that uses a structured data type rather than an atomic one to organize data for the purpose of ultimately searching through large amounts of said data.
- This is an idea that is or was used in multiple Google Services as a means of retrieving and manipulating data.
- This is an implementation that would not be well replaced by a relation database.

- A comparison of the MapReduce and Database Management Systems database structures and their efficiency based on load time and data amount.
- The structures explored in this paper were used to determine speed as per amount of data for each service.
- The implementations showed that relational databases are not always the best solution.

BOTH

- Both of these papers explore ideas that have been implemented.
- Both reference the Relational Data Model (RDM) that this class has focused on
- Both claim relational data model is not the only, nor best solution to every issue.
- Both address the possibility that Relational Database Models as we know them today will not be used in the context of Big Data or possibly at all.

Stonebraker Talk: An Overview

- Stonebraker's Paper from 2005 claimed that SQL would not be the "one size fits all" solution. Now in 2015 he believes that "one size fits none"
- There are many data markets and several solutions to the Big Data in all of them
- In coming years none of them will use row-stores like SQL thus "one size fits none"
- There are lots of new ideas and implementations coming to the forefront of database study
- Older vendors will be keeping their user interfaces and upgrading their engines to keep up with the coming changes.
- Stonebraker admits that in the 80s and 90s DB researchers were stuck on relational databases and that caused them to wrongly assume that all data could be best handled by SQL.
- From that misconception many new implementations and great opportunities in DBMS research are now being born

Wrap Up

- Bigtable has the great advantage of being a great DBMS for search engine data.
- It would not be a good implementation where a Full Relational Database Model might be expected (i.e. a small business’ records or a video game database)
- As Stonebraker said there is no one size fits all solution.
- Regardless of whether a solution calls for a MapReduce paradigm, a Bigtable or a Relational Database Model the point is every database implementation will be different and should be different especially in Big Data.