Bigtable: A Distributed Storage System for Structural Data

By: Kara Lusabia

Due Date: 05/09/14

Main Idea

- Has a structure of a multidimensional sorted map, uniquely identified by a row key, column key and timestamp, where the values are in the form of array of bytes.
- Data are clustered in groups for easier access.
- Can execute client-supplied scripts through the address space of its servers.
- Used for MapReduce, where it acts as an input and an output target.
- Operates by sharing machines with other applications.
- "Bigtable uses the Google File System (GFS) to store log and data files" (207).
- Is stored through the Google SSTable file format.
- Uses and is registered on Chubby.
- Applications include Google Earth, Personalized Search, Google Analytics and Google Finance.

Implementation

- Client library - library linked to every client.
- Master server - coordinates activities of tablets and tablet servers.
  - assign tablets to tablet servers.
  - create, delete, split, merge tablets.
- Tablet servers - handles the reads and writes, and splits tablets that are too large.
  - stops serving its tablets when network partition occurs
  - once dead, its server file should be deleted.
  - if tablet changed server, tablet server performs two minor compactions to reduce recovery time of logs.

Analysis of its implementation

- Since it is time-based, the column family can have records with multiple version, which is beneficial for a storage system that is continuously growing.
- Client friendly, because in Webtable, data are grouped by their domain, which results to easy access and efficiency.
- Mechanisms of Chubby provides reliability and maintainability.
  - Through Chubby, inactive server tablets are detected, how much traffic is present in the servers is known, can easily track down clusters and aware of which tablets are assigned to which server.
  - Tablet server is aware of when to release its lock to the master, so the master can reassign its tablets.
  - Tablet servers kill themselves if their lease is expired, but reduces the availability of servers.

Approches to analyzing large-scale data

- **MapReduce**
  - Better used for small applications, rather than large applications.
  - No constraints on the structure of the data, because the parser is customized. However, this can cause problems when sharing data. The programmer has to keep modifying the code and retest the programs to ensure the new schema is still compatible with the programs.
  - The programmer must implement his own index search when searching for data.
  - Data distribution has to be done manually and must go through map and reduce.
  - Excellent at handling node failures win the midst of its execution, because each split file is considered. Therefore, if a task failed, this single task can be restarted.
  - Hadoop did not implement tuning tools, so the code had to be changed often for it to function with the system.
  - Record-level and block-level compression slowed down execution due to frequent modifications in the code when inputting data.
- **Parallel DBMS**
  - Has an optimizer that translates queries and is divided amongst many nodes for execution.
  - Can easily do data sharing due to the implementation of schema and indexing.
    - Parsing occurs at load time
    - Has a schema, that is separated from the application, and store it in system catalogs. However, only one schema is allowed, and the data models must commit to that structure.
    - Hashing or B-tree indexes is already built in for faster access of data.
  - Has the ability to minimize data transfers over the network through the parallel query optimizer.
  - If a task failed, large chunks of work needs to be restarted, because intermediate results are not stored in the disk. Instead, large clusters of data are deployed.

Compare and contrast

- **Advantages**
  - Similar data are clustered, which results to an efficient way of reading data, easier access to data, because it does not require accessing the disk, and great compression ratios.
  - Presence of Bloom filters reduces disk access for read operations.
  - One way of indexing is through timestamp, so it is possible to have different versions of data.
  - For an increased read performance, two levels of caching is used for the tablet servers.
  - Has a great implementation of transferring a tablet into another server, in case the tablet fails.
  - Has a schema for easier and faster data access.
  - Has created solutions to the problems as they come along.
  - Only implements the features that are needed at that point.
  - Bigtable runs through Chubby for easier traffic detection, how big the clusters are and more.
- **Disadvantages**
  - Vulnerable to many types of failures.
  - Dealing with these failures involves calling different protocols and having too many can cause complications later.