Goel, Ajay: Nu Id (001897443)

INFO 7250
Engineering Big-Data Systems
Summer Full 2019
Assignment 1

Bigtable: A Distributed Storage System for Structured Data

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size petabytes of data across thousands of commodity servers.

- Bigtable has successfully provided a flexible, high performance solution for all of these Google products. Bigtable has achieved several goals: wide applicability, scalability, high performance and high availability.
- Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format.
- Data is indexed using row and column names that can be arbitrary strings.
- A Bigtable is a sparse, distributed, persistent multidimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.

Rows:

- Bigtable maintains data in lexicographic order by row key.
- The row range for a table is dynamically partitioned. Each row range is called a tablet, which is the unit of distribution and load balancing.
- In Web table, pages in the same domain are grouped together into contiguous rows by reversing the hostname components of the URLs.

Column families:

- It forms the basic unit of access control.
- All data stored in a column family is usually of the same type.
- The number of distinct column families in a table be small and that families rarely change during operation.
- Access control and both disk and memory accounting are performed at the column-family level.

Timestamps:

- Each cell in a Bigtable can contain multiple versions of the same data; these versions are indexed by timestamp. Bigtable timestamps are 64-bit integers. They represent real time in microseconds or be explicitly assigned by client applications.
- Applications that need to avoid collisions must generate unique timestamps themselves. Different versions of a cell are stored in decreasing timestamp order.
- The client can specify either that only the last n versions of a cell be kept or that only new-enough versions be kept.

API:

The Bigtable API provides functions for creating and deleting tables and column families. It also provides functions for changing cluster, table, and column family metadata, such as access control rights.
Client applications can write or delete values in Bigtable, look up values from individual rows, or iterate over a subset of the data in a table.

Bigtable supports several other features that allow the user to manipulate data in more complex ways:

- Bigtable supports single-row transactions, which can be used to perform atomic read modify write sequences on data stored under a single row key.
- Bigtable allows cells to be used as integer counters.
- Bigtable supports the execution of client-supplied scripts in the address spaces of the servers.

Bigtable can be used with MapReduce, a framework for running large-scale parallel computations developed at Google.

Building blocks:
Bigtable is built on several other pieces of Google infrastructure.
Bigtable depends on a cluster management system for scheduling jobs, managing resources on shared machines, dealing with machine failures, and monitoring machine status.

- An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings.

Implementation:

- The Bigtable implementation has three major components: a library that is linked into every client, one master server, and many tablet servers.
- Tablet servers can be dynamically added from a cluster to accommodate changes in workloads.
- The master is responsible for assigning tablets to tablet servers, detecting the addition and expiration of tablet servers, balancing tablet-server load, and garbage collection of files in GFS
- A Bigtable cluster stores a number of tables. Each table consists of a set of tablets, and each tablet contains all data associated with a row range.

Tablet Location:

Levels:

- The first level is a file stored in Chubby that contains the location of the root tablet. The root tablet contains the location of all tablets in a special METADATA table.
- The METADATA table stores the location of a tablet under a row key that is an encoding of the tablet's table identifier and its end row.

Tablet Assignment:

- Each tablet is assigned to one tablet server at a time.
- The master keeps track of the set of live tablet servers and the current assignment of tablets to tablet servers including which tablets are unassigned.
- When a tablet is unassigned, and a tablet server with sufficient room for the tablet is available, the master assigns the tablet by sending a tablet load request to the tablet server.

Compactions:
The size of the memtable increases. When the memtable size reaches a threshold, the memtable is frozen, a new memtable is created and the frozen memtable is converted to an SSTable and written to GFS.
This minor compaction process has two goals:

- It shrinks the memory usage of the tablet server and it reduces the amount of data that has to be read from the commit log during recovery if this server dies.
- Incoming read and write operations can continue while compactions occur. Every minor compaction creates a new SSTable.
- If this behavior continued unchecked, read operations might need to merge updates from an arbitrary number of SSTables.
- Instead, the number of such files by periodically executing a merging compaction in the background is bounded. A merging compaction reads the contents of a few SSTables and the memtable and writes out a new SSTable.
- The input SSTables and memtable can be discarded as soon as the compaction has finished.

Refinements:
The implementation described in the previous section required a number of refinements to achieve the high performance, availability, and reliability required by our users. This section describes portions of the implementation in more detail in order to highlight these refinements.

- Locality groups
- Compression
- Caching for real performance
- Bloom filters
- Commit log implementation

- Speeding up tablet recovery
- Exploiting immutability

Performance Evaluation:
A Bigtable cluster with N tablet servers was set up to measure the performance and scalability of Bigtable as N is varied.

- The tablet servers were configured to use 1 GB of memory and to write to a GFS cell consisting of 1786 machines with two 400 GB IDE hard drives each N client machines generated the Bigtable load used for these tests.
- Each machine had two dual-core Opteron 2 GHz chips enough physical memory to hold the working set of all running processes, and a single gigabit Ethernet link. The machines were arranged in a two-level tree-shaped switched network with approximately 100-200 Gbps of aggregate bandwidth available at the root.
- All of the machines were in the same hosting facility and therefore the round-trip time between any pair of machines was less than a millisecond.
- The tablet servers and master, test clients, and GFS servers all ran on the same set of machines.
- Every machine ran a GFS server. Some of the machines also ran either a tablet server, or a client process, or processes from other jobs that were using the pool at the same time as these experiments.
- Single tablet server performance
- Scaling:
    - Aggregate throughput increases dramatically by over a factor of a hundred, as we increase the number of tablet servers in the system from 1 to 500. For example, the performance of random reads from memory increases by almost a factor of 300 as the number of tablet server increases by a factor of 500.
    - This behavior occurs because the bottleneck on performance for this benchmark is the individual tablet server CPU.

Real Applications:

- **Google Analytics**
    - Google Analytics (analytics.google.com) is a service that helps webmasters analyze traffic patterns at their web sites.
    - It provides aggregate statistics, such as the number of unique visitors per day and the page views per URL per day, as well as site-tracking reports, such as the percentage of users that made a purchase given that they earlier viewed a specific page.

- **Google Earth**
    - Google operates a collection of services that provide users with access to high-resolution satellite imagery of the world's surface, both through the web-based Google Maps interface (maps.google.com) and through the Google Earth (earth.google.com) custom client software.
    - These products allow users to navigate across the world's surface: they can pan, view, and annotate satellite imagery at many different levels of resolution.
    - This system uses one table to preprocess data, and a different set of tables for serving client data.

- **Personalized Search**
    - Personalized Search (www.google.com/psearch) is an opt-in service that records user queries and clicks across a variety of Google properties such as web search, images, and news.
    - Users can browse their search histories to revisit their old queries and clicks, and they can ask for personalized search results based on their historical Google usage patterns.

Lessons:

- Large distributed systems are vulnerable to many types of failures, not just the standard network partitions and fail-stop failures assumed in many distributed protocols.
- It is important to delay adding new features until it is clear how the new features will be used.
    - Initially, it was planned to support general-purpose transactions in our API because we did not have an immediate use for them, however, we did not implement them. Now that many real applications running on Bigtable, it is examined by their actual needs, and have discovered that most applications require only single-row transactions.
    - Where people have requested distributed transactions, the most important use is for maintaining secondary.
- Indices and we plan to add a specialized mechanism to satisfy this need. The new mechanism will be less general than distributed transactions but will be more efficient and will also interact better with our scheme for optimistic cross-data-center replication.
- Importance of proper system-level monitoring.

Conclusion:

- The value of simple designs. Given both the size of our system (about 100,000 lines of non-test code), as well as the fact that code evolves over time in unexpected ways, It has been found that code and design clarity are of immense help in code maintenance and debugging.
- Given the unusual interface to Bigtable, an interesting question is how difficult it has been for our users to adapt to using it. New users are sometimes uncertain of how to best use the Bigtable interface, particularly if they are accustomed to using relational databases that support general-purpose transactions.
- Nevertheless, the fact that many Google products successfully use Bigtable demonstrates that our design works well in practice.