Bigtable

Sirui Tan st2957

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. Being the data storage solution to many Google projects, it has successfully reached flexibility and high-performance and gives clients dynamic control over data layout and format.

Bigtable is designed to reliably scale to petabytes of data and thousands of machines. It has achieved several goals: wide applicability, scalability, high performance, and high availability. It resembles a database, but provides a much simpler data model that supports dynamic control over data layout and format. Bigtable treats data as uninterpreted strings. Its schema parameters let clients dynamically control whether to serve data out of memory or from disk.

A Bigtable is a sparse, distributed, persistent multidimensional sorted map which is indexed by a row key, column key, and a timestamp. Row keys are arbitrary strings which anatomically read and write data. It also serves as the key for sorting table data in lexicographical order. Column keys are grouped into column families, which form the basic unit of access control. Timestamp is used for solving the possible conflict of multiple versions of the same data.

The Bigtable API provides functions for creating and deleting tables and column families, as well as functions for changing cluster, table and column family metadata. Using API, client applications of Bigtable can write and delete values in Bigtable and lookup values from individual rows, or iterate over a subset of the data in a table.

Bigtable relies on Google File System (GFS) to operate in a shared pool of machines that run a variety of other distributed applications. Its data are stored using the Google SSTable format. As for contingency, Bigtable applies Chubby, a highly-available and persistent distributed lock service. Chubby helps Bigtable to ensure whether there is at most one active master, store the bootstrap location, schema information, access control lists, and discover tablet servers and finalize tablet server deaths.

The Bigtable implementation has such a hierarchy of three major components: a library that is linked into every client, one master server, and many tablet servers. The master server does administrative jobs such as tablet assignment, and tablet server monitor. Tablet servers, on the other hand, handle read and write requests.

Such a distributed implementation of Bigtable requires a number of refinements to achieve high performance, availability and reliability. Such refinements include locality groups, compression, caching, bloom filters, commit-log implementation and so on.

Finally, the performance of Bigtable is illustrated using a series of experiments on a cluster of N tablet servers.