[Back to index]

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, and Robert E. Gruber
Google

One-line Summary

BigTable is a distributed storage system for managing structured data with a simple data model and providing flexible controls over data layout, locality, storage medium, and format, but has poor performance and consistency.

Overview/Main Points

- Data model
  - a sparse, distributed, persistent multi-dimensional sorted map
  - indexed by a row key, column key, and a timestamp (a 64-bit integer, “real time” in ms)
  - row
    - row key is 64KB in size, but typical 10-100 bytes
    - every r/w for a single row key is atomic
    - data sorted in lexicographically by row key
    - each tablet contains all data associated with a row range by dynamically partition
    - reversed URL: com.cnn.www
  - column
    - keys group as column families as the basic unit of access control and both disk and memory accounting, and they are always the same type and compress stored.
    - family:qualifier
  - Each tablet is represented by several SSTables
    1. key - value pairs
    2. key - offset index
- APIs
  - create
  - lookup
  - update
  - delete
  - scan
  -
- Arch
  - master
    - assign tablets to tablet servers
    - tablet server management: add or expire
    - load balance of tablet-server
    - garbage collection of files in GFS
    - handle schema changes including table and column family creation
  - tablet server
    - Only one tablet is assigned to one server

        - handle r/w to the tablets
        - split tablets if too large
    - Chubby server
        - a high-available and persistent distributed lock service based on Paxos.
        - 5 active replicas, one of which is master to serve requests if the majority live and can talk to each other.
        - namespace consists of directories and small files, which act as locks
        - Each r/w to files are atomic
        - Chubby client registers callbacks for notification of changes or session expirations.
        -
    - client
        - cache tablet locations
        -
    - GFS server for storing logs and data files in SSTable file format
        - SSTable: a persistent, ordered immutable key-value map
        - Each SSTable is represented by a set of blocks associated with indices (load into memory when opened for future lookup in a single disk seek)
    - cluster management system
        - job schedule
        - resource management on shared machiens
        - machine failure recovery, including master restart
        - machine status monitor
- Operations
    - Lookup
        - performs a three-level search starting from location of the root tablet, which contains the location of all tablets in a METADATA table, which in turn contains the location of user tablets
        - The METADATA table stores the location tablets whose row keys are an encoding of the tablet's table identifier and its end row.
        -
    - Read
        - merge data from in-memory Memtables and on-disk SSTables
        - Bloom filters for only necessary reads
    - Write
        - write both in-memory Memtables (in a sorted manner) and on-disk commit logs
    - Compaction in tablet servers
        - minor compaction: flush in-memory Memtables to on-disk SSTables (GFS)
        - major compaction: merge on-disk SSTables
- Recovery
    - Tablet server failure: the same to assign a tablet to a new tablet server
    - Master failure: restart by the cluster management system
        1. the master acquires a unique master lock in Chubby
        2. scan Chubby's servers directory for live servers
        3. talks to live servers about their assigned tablets
        4. scan the METADATA table for the set of tablets, and processes a tablet that has no associated server
- User's dynamic control
    - Data layout: segregate several column families to one SSTable
    - Locality: leverage row-key range in tablets
    - Storage medium: load all SSTables into memory
    -

Relevance

Flaws