Google Cloud Bigtable

By Group #04

- Akshay Dhabale (40163636)
- Kshitij Yerande (40194579)
- Mrinal Rai (40193024)
- Yogesh Yadav (40202192)

Google Bigtable - Motivation

- Increased storage requirements
- High scalability & applicability
- High performance and high availability
- Real-time, Interactive data serving or batch processing
- Ideal for use cases such as e-commerce, ad tech, fintech, digital media, and IoT

Google Bigtable - Topics

Architecture:

a. Google File System
b. Map Reduce
c. Lock Service (Chubby)

Data Model and API

d. Row and column families
e. Timestamp

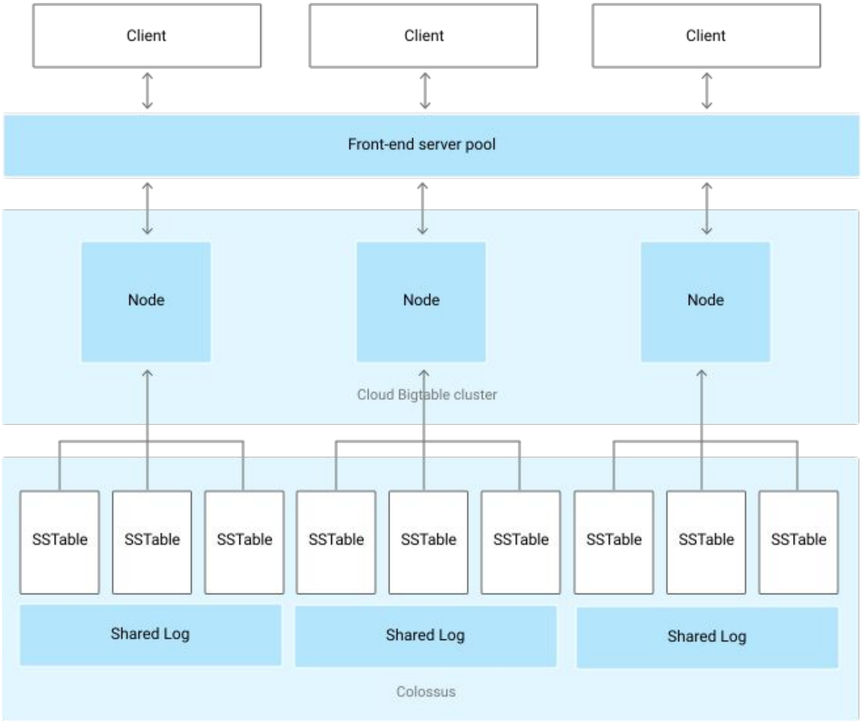
Implementation:

a. Tablet Location
b. Tablet Assignment
c. Tablet Serving

Performance Improvement refinements

a. Locality Groups
b. Caching
c. Bloom Filters

Bigtable Architecture



- ➢ All the client requests go through a frontend server
- ➢ Nodes are organized into a Bigtable cluster
- ➢ Cluster belongs to a Bigtable instance
- ➢ Each Node handle a subset of requests sent to cluster
- ➢ The Bigtable table is sharded into blocks of contiguous rows called tablets
- ➢ Tablets are stored on colossus GFS, in SSTable format
- ➢ Supports rebalancing of tablets and recovery from failure

Bigtable Architecture - Building Blocks

Google File System

- ➢ A distributed File storage containing thousands of commodity servers
- ➢ Files are subdivided into multiple chunks and are shared across multiple chunk servers

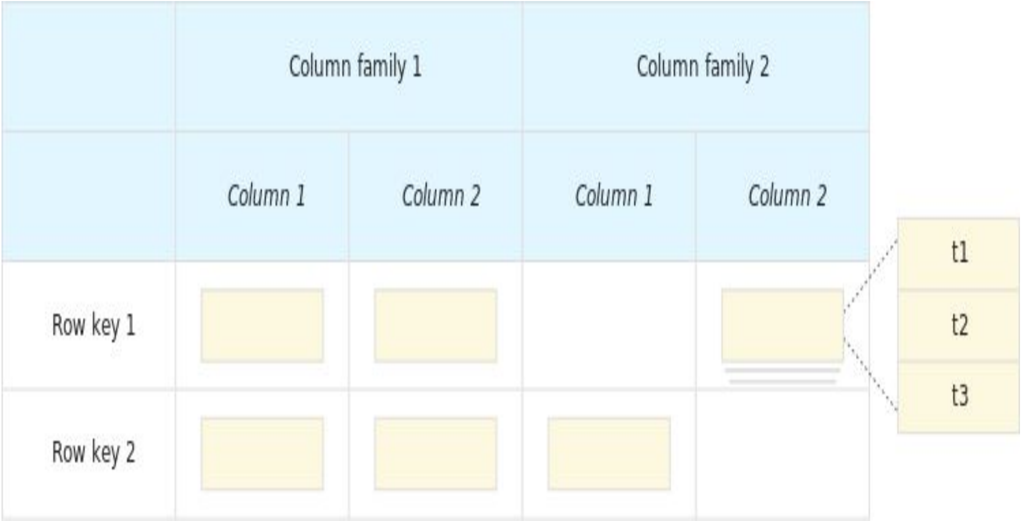
Map Reduce

- ➢ A programming model where the code is pushed onto multiple servers and those servers process or run that code
- ➢ Files are subdivided into multiple chunks and are shared across multiple chunk servers

Chubby

- ➢ A highly-available and persistent distributed lock service that is used by Bigtable
- ➢ It consists of five active replicas, among which one of the replicas is elected as the Master and listens to active requests

Data Model



	Column family 1		Column family 2	
	Column 1	Column 2	Column 1	Column 2
Row key 1	▨	▨		▨
Row key 2	▨	▨	▨	

Row Keys and Column Families

- ➢ Row keys are arbitrary string
- ➢ Row is the unit of transactional consistency
- ➢ Big Table maintains data in lexicographic order by row key.
- ➢ Columns in the Bigtable are grouped together called Column Families
- ➢ A column family must be created before loading data into any columns

Data Model

Timestamp

- ➢ Each cell in Bigtable contain multiple versions of data with the timestamp.
- ➢ Timestamps can be specified by the Bigtable or from the client’s application.
- ➢ Bigtable stores data in decreasing timestamp order keeping recent data first.

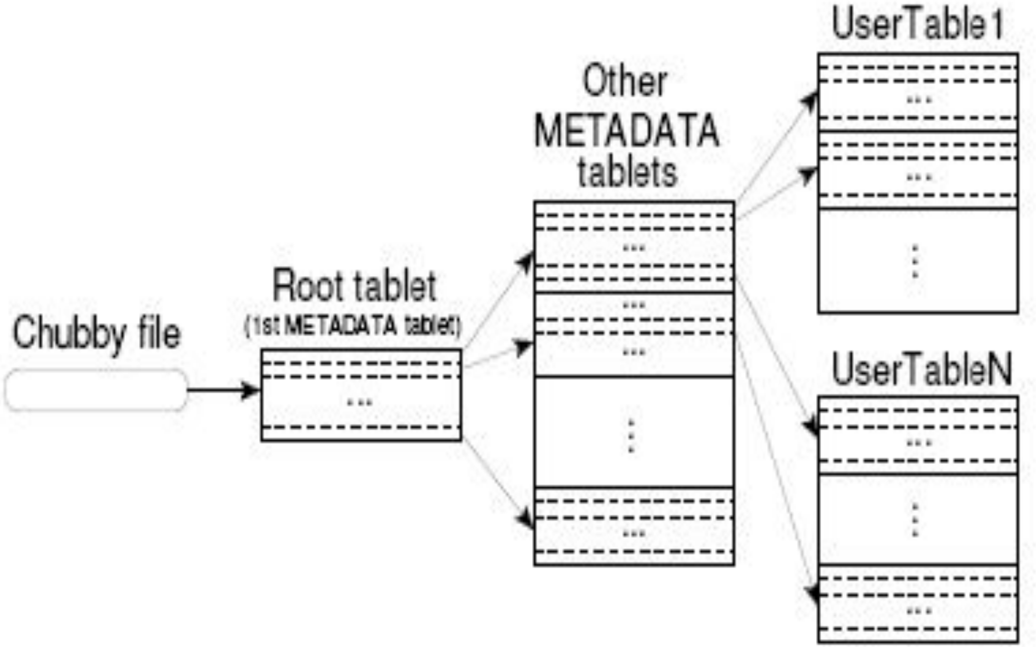
API

- ➢ The Bigtable APIs provide functions for creating, deleting tables and column families.
- ➢ Bigtable also support client supplied scripts which can executed in the server space.

Implementation

- **Tablet Location :**
  a. Uses chubby file to locate tablets in a hierarchical fashion from METADATA file.
- **Tablet Assignment**
  a. Master server keeps track of all live tablet servers.
  b. Uses chubby to gain lock on tablet servers.
  c. Performs complete scan of metadata in case of master server restart.
- **Tablet Serving**
  a. read/write operation are written to tablet log file.
  b. Operations performed on shared view(memtable + SSTable) stored on tablet in order of log file.

Implementation - Locating a Tablet



Data stored in key value pair:

- Key: tablet identifier + end row
- Value: location of tablet

Caching enabled at client side for faster tablet location.

- If no data found perform hierarchical search.

Implementation - Tablet Assignment

Master Server keeps track of all live tablet servers.

Bigtable uses Chubby:

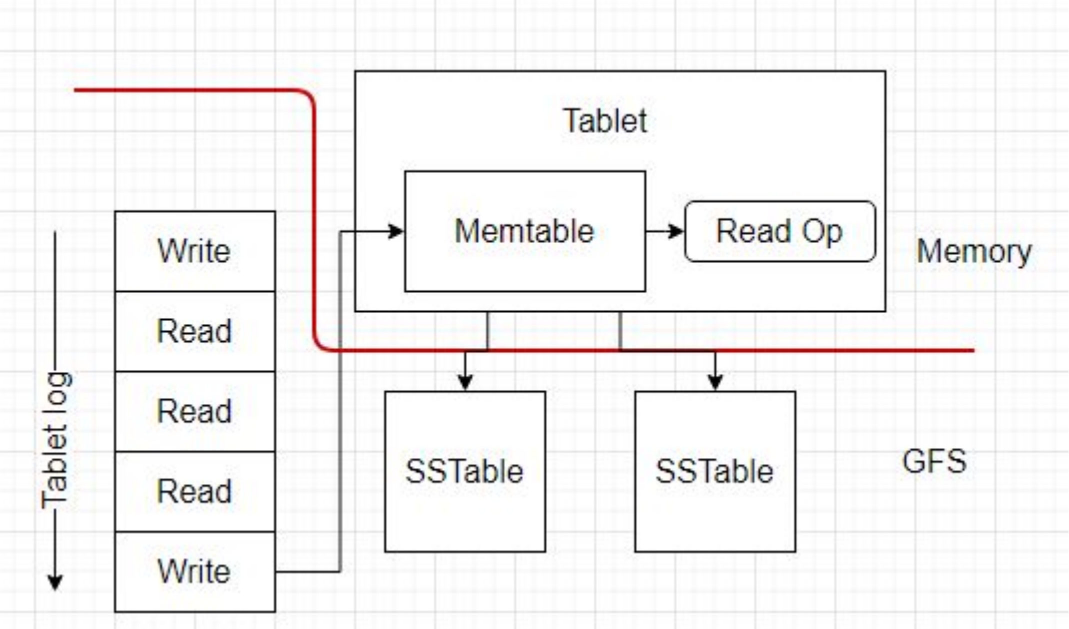
1. Tablet server acquires lock on file in Chubby directory.
2. Server Loses lock in case of network issues.
3. Server re attempts to acquire lock. Success if the file exists or else terminate itself.

Master Server restart:

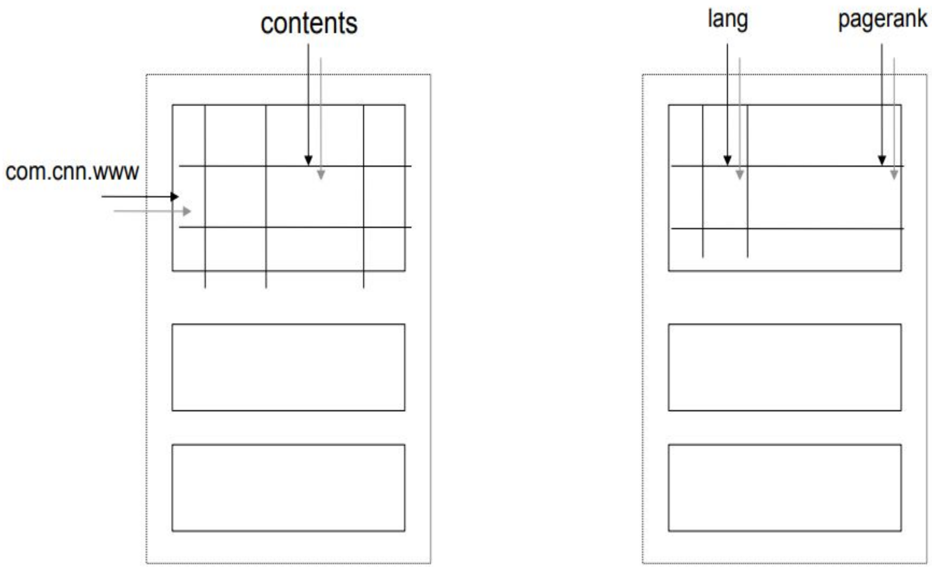
1. Grabs unique lock on Chubby Directory.
2. Scan directory to retrieve live servers.
3. Communicate to each live tablet server to get the tablets assigned to them..
4. scan the METADATA table to learn the set of tablets.
5. Perform assignment eligibility check of unassigned tablet.

Implementation - Tablet Serving

Read/Write operations are logged in commit log, then applied to an in-memory memtable



Refinements - Locality Groups



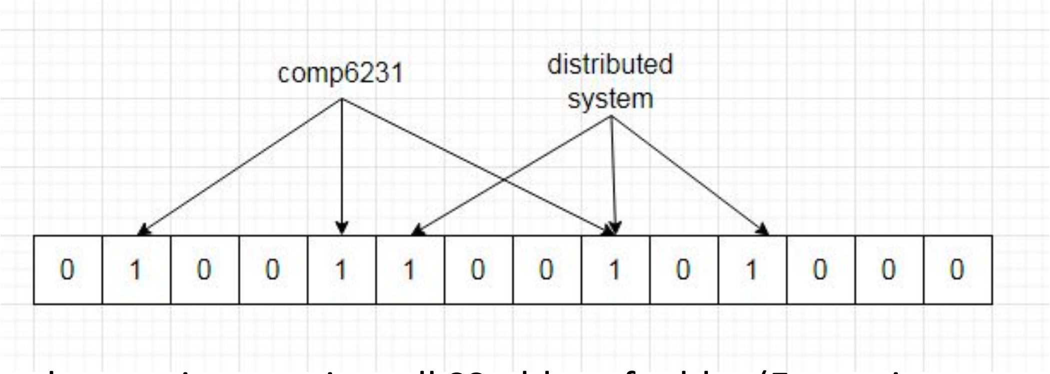
1. Group multiple column families together into a locality group which are accessed together.

2. Can keep locality group in memory to avoid frequent disk access.

3. Used for efficient read access. For e.g while scanning webpage metadata, content is not required.

Refinements - Bloom Filters

A Bloom filter is a **space-efficient probabilistic** data structure that is used to test whether an element is a member of a set

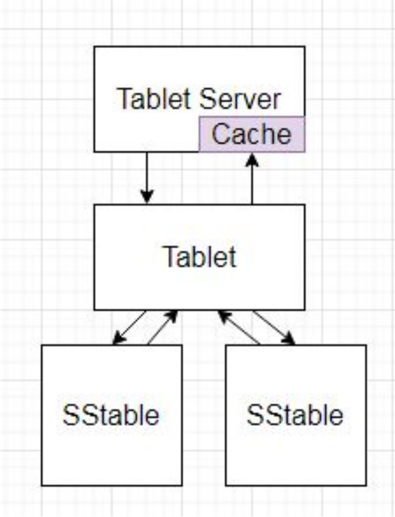


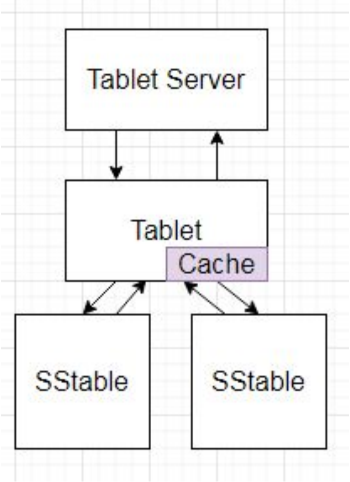
M-bit array: to store the hashes of data item.

K hash functions: to compute hashes of given data item

- Read operation requires all SStables of tablet (Expensive operation)
- Reduce disk access by bloom filter:
    a. Check if the data item is present in the SStable before performing read operation.

Refinements - Caching





Scan Cache:

- High level Cache
- Cache Key value Pair returned by SSTable

Block Cache:

- Low Level Cache
- caches SSTables blocks that were read from GFS.

BigTable : Pros and Cons

Pros

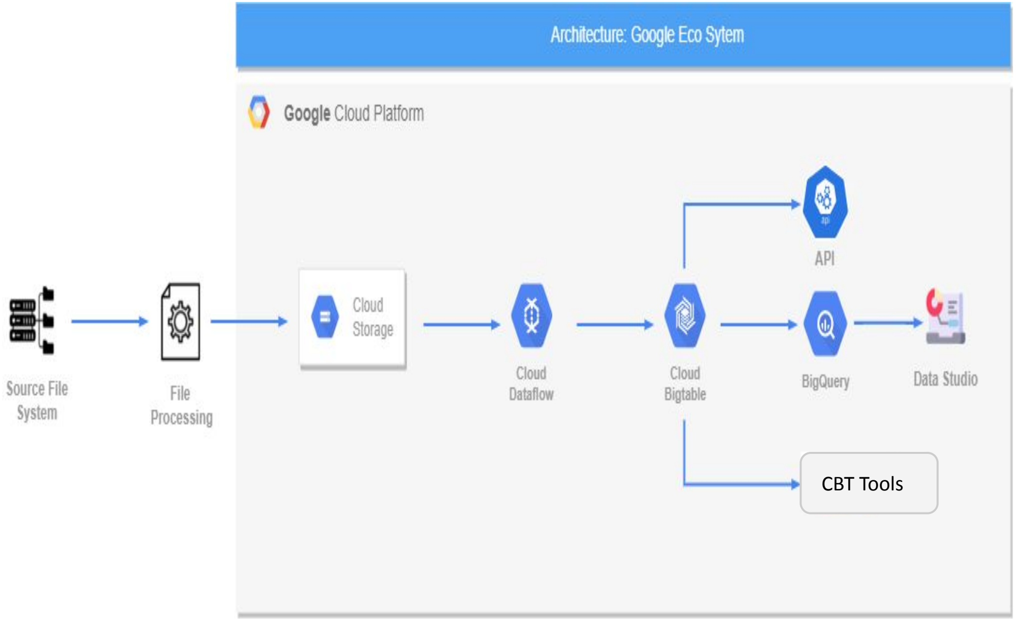
- High Performance and scalable
- Fully Managed and reliable database
- Data replication
- Backups
- Support for automatic failover

Cons

- It is not an open source database.
- Requires adaptation to the Bigtable approach for application writing.
- Demands manual query programming as Structure query language is not supported by Bigtable.
- No support for ACID transactions as used in RDBMS.

Demo

Data Flow Diagram



Distributed Feature Aspects

- Autoscaling : Cluster resizing without downtime
- High Availability
- Automatic Failover
- Read Write Throughput
- Query Statistics

Data Set : Amazon US Customer Reviews Dataset

- Size: 2.5GB
- Includes Customer rating and overall rating for each appliance under different product categories

References

- Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." *ACM Transactions on Computer Systems (TOCS)* 26.2 (2008): 1-26.

- Kalid, Sultana, et al. "Big-data NoSQL databases: A comparison and analysis of “Big-Table”,“DynamoDB”, and “Cassandra”." 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA). IEEE, 2017.

- Amazon US customer reviews Dataset - https://www.kaggle.com/cynthiarempel/amazon-us-customer-reviews-dataset

Any questions?

Thank you.