Bigtable

Yingchen Wang, Tiancheng Ge
Nov. 27, 2019
Introduction

Bigtable is a distributed storage system.
• Resembles a database.

Lots of applications build upon Bigtable.
• Web indexing
• Google Earth
• Gmail
• etc.
History

- Bigtable is build upon GFS, and
- Can be used with MapReduce
  - As input source/output target for MapReduce jobs.
Motivation

- **Scalability**
  - Thousands of servers.
  - Petabyte of data.
  - Millions of requests per second.

- **Commodity hardware**
  - Disks, *good at sequential reads/writes*.
  - System should be fault-tolerant to hardware failures.
Data Model

- Tables in Bigtable.
  - A *sparse*, distributed, multidimensional *sorted* map.

\[
\text{(row:} \text{string, column:} \text{string, time:} \text{int64}) \rightarrow \text{string}
\]

- Key of the map is three dimensional.
  - Row key
  - Column key
  - Timestamp
- Content of the map is an uninterpreted array of bytes.
Row

Table entries are ordered by row keys.
- Select row key to get good locality.
- Short row range means small number of machines.
- e.g. reversed URL as key
  
  images.google.com/... → com.google.images/...
  maps.google.com/... → com.google.maps/...

Row is the unit of transactions.
- Atomic transaction for a single row key.
- No single transaction across rows.
Column

Columns grouped into *column families*

- Data stored in a column family is usually of the same type
  - Compressed together.

- Small number of distinct column families.
  - In the hundreds at most.
  - Families rarely change during operation.
Column (continued)

Column families must be declared explicitly before insert data.
- Unlike rows.
- Part of the schema.

Column key is "family:qualifier".

Column family is basic unit of access control.
Data Model

(row:string, column:string, time:int64) → string
**Timestamps**

Store different versions of data in a cell.
- Webtable example: times when pages were crawled.

Entries stored in decreasing timestamp order.
- most recent version read first.

Lookup options:
- Return most recent K values.
- Return all values.

Garbage collection options:
- Keep last N versions.
- Keep last N days of data.
APIs

• Create delete tables and column families.
• Change cluster, table and column family meta-data.
• Single-row transactions.
  • Atomic read-modify-write on a single row.
• Does not support single transaction across row keys.
  • Does support batching writes across rows.

• Supports scan operation.
  • Sequential reads from a disk are fast.
Building Blocks

Three Google infrastructure supports.

- GFS to store log and data files.
  - Files are replicated with GFS.
- Google Sorted Strings Table (SSTable), file format.
- Distributed lock service: Chubby.
**SSTable (Sorted Strings Table)**

SSTable immutable-file format

- Keys and values are arbitrary byte strings.
- Lookup and iterate over key/values in a range.

SSTable is stored on disk.

- Can be completely mapped into memory.

SSTable contains blocks (64KB in size)

- Block index (at the end of the SSTable) to locate blocks.
- Lookup is one disk seek:
  - Perform a binary search in the in-memory index.
  - Reading the appropriate block from disk.
Chubby

Lock service for distributed system.

- Uses Paxos consists of five active replicas.
- Reads/writes to a file are atomic.

Master need to acquire a unique master lock.

- Ensures at most one active master at any time.

Bigtable is heavily relied on Chubby.

- Chubby unavailability -> Bigtable unavailable.
- Some data in Bigtable unavailable: 0.0047%.
Tablets

Smallest unit for distribution and load balancing.

Each table consists of a set of tablets.

- Each tablet contains all of the data associated with a row range.

Table automatically split into multiple tablets

- Each tablet about 100-200 MB in size.
Implementation

Chubby Service

Provide service to every one.

Master

Master manages tablets and tablet servers.

Client

Most of time...

Tablet Server

10~1000 Tablets

Tablet Server

10~1000 Tablets

Tablet Server

10~1000 Tablets
Implementation

• Master
  • Assigns tablets to tablet servers.
  • Detects addition & expiration of tablet servers.
  • Load Balancing.
  • Garbage collection of GFS files.
  • Schema changes (table & column family additions and deletions).

• Client
  • Communicates with tablet servers for data.
    • Most clients don't communicate with the master.
  • Cache tablet server location information.
Tablet Location

Three-level hierarchy like a B+ tree.

Chubby stores the location of the root tablet.

- Clients send request to Chubby, then cache the tablet location.
Tablet Location

Root is never split.

- Hierarchy is always 3 levels.
- Up to $2^{34}$ tablets addresses.
Let’s write

T: Tiancheng
Y: Yingchen

Memtable
How Bigtable works

Let’s write

User Manos

M: Manos
T: Tiancheng
Y: Yingchen

Sorted

Memtable
How Bigtable works

M: Manos
T: Tiancheng
Y: Yingchen

Write more!
Ummm.. It’s too much for me

Memtable

Sorted
How Bigtable works

User Manos

Time to transform

Sorted

M: Manos
T: Tiancheng
Y: Yingchen

Memtable
How Bigtable works

User Manos

Memtable

Memory

Well it may not be safe

Tablet log

GFS

A Tablet

M: Manos
T: Tiancheng

SSTable 0

D: Donald

SSTable 1
How Bigtable works

Well it may not be safe

User Manos

Memtable

Memory

M: Manos
T: Tiancheng

SSTable 0

D: Donald

SSTable 1

Tablet log

GFS

Write to log first!!!
How Bigtable works

Each METADATA tablet contains the location of a set of user tablets.
How Bigtable works

User Manos

Let's read

Memtable

Memory

Read from a merged view of SSTables and Memtable

SSTable 0

D: Dump
T: Tronald

M: Manos
T: Tiancheng

GFS

SSTable 1

D: Donald
T: Trump
Tablet Assignment

Tablet assigned to at most one tablet server

Chubby used to keep track of tablet servers
Tablet Assignment (tablet server)

When a tablet server starts, Chubby creates and acquires exclusive lock on file in specific Chubby directory

A tablet server stops serving its tablets if it loses its exclusive lock (i.e. network partition causes loss of chubby session)

A tablet server will try to reacquire an exclusive on its file as long as the file still exists. If no, then kill itself.
Tablet Assignment (master server)

Master keeps track of live tablet servers and assignment of tablets to tablet servers

Master executes the following steps at startup:

- Grabs a unique *master* lock in Chubby
- Scans the server directory in Chubby to find the live servers
- Communicates with each live tablet server to discover what tablets are already assigned to each server
- Scans the METADATA table to find unassigned tablets

Master periodically asks each tablet servers for the status of its lock
Compactions

Minor compaction
- Memtable increases in size with write operations
- The memtable frozen
- New memtable created, and the old one converted to an SSTable and written to GFS

Merging compaction
- Periodically executed in the background
- Merge a few SSTables and the memtable to one SSTable
Compactions

Major compaction - leaves only one SSTable
- A merging compaction which rewrites all SSTables into exactly one SSTable
- Leaves no deleted data
Refinements

Commit log

• If commit logs were separate files, lots of disk seeks for writes
• One commit log per tablet server
• When the tablet server crashes, the master server sorts the logs
## Refinements (commit log)

<table>
<thead>
<tr>
<th>Before sort</th>
<th>After sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tablet 2, 14:00:00</td>
<td>Tablet 1, 14:00:20</td>
</tr>
<tr>
<td>Tablet 3, 14:00:10</td>
<td>Tablet 1, 14:00:45</td>
</tr>
<tr>
<td>Tablet 1, 14:00:20</td>
<td>Tablet 2, 14:00:00</td>
</tr>
<tr>
<td>Tablet 4, 14:00:30</td>
<td>Tablet 3, 14:00:10</td>
</tr>
<tr>
<td>Tablet 3, 14:00:40</td>
<td>Tablet 3, 14:00:40</td>
</tr>
<tr>
<td>Tablet 1, 14:00:45</td>
<td>Tablet 4, 14:00:30</td>
</tr>
</tbody>
</table>
Refinements

Bloom filters

• A probabilistic data structure
• A bloom filter allows us to ask whether an SSTable might contain any data for a specified row/column pair
Evaluation

Number of 1000-byte values read/written per second.

The table shows the rate per tablet server; the graph shows the aggregate rate.

<table>
<thead>
<tr>
<th>Experiment</th>
<th># of Tablet Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>random reads</td>
<td>1212</td>
</tr>
<tr>
<td>random reads (mem)</td>
<td>10811</td>
</tr>
<tr>
<td>random writes</td>
<td>8850</td>
</tr>
<tr>
<td>sequential reads</td>
<td>4425</td>
</tr>
<tr>
<td>sequential writes</td>
<td>8547</td>
</tr>
<tr>
<td>scans</td>
<td>15385</td>
</tr>
</tbody>
</table>

Values read/written per second vs. Number of tablet servers
References

[1] https://www.slideshare.net/marloxo1/google-big-table-62397242

Q&A
Revisions

Compression

- Clients can control whether or not the SSTables for a locality group are compressed.
- Emphasized speed instead of space reduction.