BigTable
BigTable

In the early 2000s, Google had way more data than anybody else did.

Traditional databases couldn’t scale.

Want something better than a filesystem (GFS).

BigTable optimized for:

- Lots of data, large infrastructure
- Relatively simple queries

Relies on Chubby, GFS
Uninterpreted strings in rows and columns

\[(r : \text{string}) \rightarrow (c : \text{string}) \rightarrow (t : \text{int64}) \rightarrow \text{string}\]

Mostly schema-less; column “families” for access

Data sorted by row name

- lexicographically close names likely to be nearby

Each piece of data versioned via timestamps

- Either user- or server-generated

- Control garbage-collection
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
# Tablets

<table>
<thead>
<tr>
<th>a</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>data</td>
</tr>
<tr>
<td>c</td>
<td>data</td>
</tr>
<tr>
<td>d</td>
<td>data</td>
</tr>
</tbody>
</table>

Each table composed of one or more tablets
Starts at one, splits once it’s big enough
- Split at row boundaries

Tablets ~100MB-200MB
# Tablets

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td>data</td>
</tr>
<tr>
<td>b</td>
<td></td>
<td>data</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>data</td>
</tr>
<tr>
<td>d</td>
<td></td>
<td>data</td>
</tr>
<tr>
<td>e</td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>

Each table composed of one or more tablets

Starts at one, splits once it’s big enough

- Split at row boundaries

Tablets ~100MB-200MB
# Tablets

<table>
<thead>
<tr>
<th></th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

Each table composed of one or more tablets

Starts at one, splits once it’s big enough

- Split at row boundaries

Tablets ~100MB-200MB
Tablets

A tablet is indexed by its range of keys
- `<START>` - "c"
- "c" - `<END>`

Each tablet lives on at most one tablet server

Master coordinates assignments of tablets to servers
Tablets

- Tablet locations stored in METADATA table
- Root tablet stores locations of METADATA tablets
- Root tablet location stored in Chubby
Tablet serving

Tablet data persisted to GFS

- GFS writes replicated to 3 nodes
- One of these nodes should be the tablet server!

Three important data structures:

- memtable: in-memory map
- SSTable: immutable, on-disk map
- Commit log: operation log used for recovery
Tablet serving

Writes go to the commit log, then to the memtable
Reads see a merged view of memtable + SSTables
  - Data could be in memtable or on disk
Compaction and compression

Memtables spilled to disk once they grow too big
  - “minor compaction”: converted to SSTable
Periodically, all SSTables for a tablet compacted
  - “major compaction”: many SSTables -> one
Compression: each block of an SSTable compressed
  - Can get enormous ratios with text data
  - Locality helps—similar web pages in same block
Bloom filters

Goal: efficient test for set membership: member(key) -> true/false

false ==> definitely not in the set
true ==> probably is in the set

Generally supports adding elements but not removing them

Basic version: m bit positions, k hash functions

For insert: compute k bit locations, set to 1
For lookup: compute k locations, check for 1

BigTable: avoid reading SSTables for elements that are not present; saves many seeks
KV Store Design Options

LSM: Log structured merge trees (1996)

NoSQL Key-value Stores

- RocksDB
- Google BigTable
- mongoDB
- SQLite
- LinkedIn
- Amazon DynamoDB
- Cassandra
- Apache HBase

Applications:
- Machine learning
- Social media
- Smart homes
- Web browsers
- Smart phones
- Web-based apps
- Security
- Health devices
- Graphs
- Analytics
BigTable components

- Client
- Tablet Server
- Tablet Server
- Tablet Server
- Tablet Server
- Tablet Server

- Master

- Chubby
- GFS
Master

Tracks tablet servers (using Chubby)
Assigns tablets to servers
Handles tablet server failures
Master startup

- Acquire master lock in Chubby
- Find live tablet servers (each tablet server writes its identity to a directory in Chubby)
- Communicate with live servers to find out who has which tablet
- Scan METADATA tablets to find unassigned tablets
Master operation

Detect tablet server failures
- Assign tablets to other servers

Merge tablets (if they fall below a size threshold)

Handle split tablets
- Splits initiated by tablet servers
  - Master responsible for assigning new tablet

Clients never read from master
BigTable components

Client

Tablet Server  Tablet Server

Master

Tablet Server  Tablet Server

Chubby

GFS
BigTable components

Client

Where is the root tablet?

Chubby

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Master

Tablet Server

Tablet Server

Tablet Server

GFS
BigTable components

- Client
- Tablet server 2
- Master
- Tablet server
- Tablet server
- Tablet server
- Tablet server
- Chubby
- GFS
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
BigTable components

Client

Where is the METADATA tablet for table T row R?

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Client

GFS

Chubby
BigTable components

Client

Tablet server 1

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Master

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
BigTable components

Where is table $T$ row $R$?

Client

Master

Tablet Server Tablet Server Tablet Server Tablet Server Tablet Server Tablet Server

Chubby

GFS
BigTable components

Client

Tablet server 3

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS

Chubby
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS

Chubby
BigTable components

Client

Read table $T$ row $R$

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS

Chubby
BigTable components

Client

Row

Master

Tablet Server
Tablet Server
Tablet Server
Tablet Server
Tablet Server
Tablet Server

Chubby

GFS
Optimizations

Clients cache tablet locations

  Tablet servers only respond if Chubby session active, so this is safe

Locality groups

  Put column families that are infrequently accessed together in separate SSTables

Smart caching on tablet servers

Bloom filters on SSTables