BigTable

CS 452
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.
Chubby
Chubby

Distributed coordination service

Goal: allow client applications to synchronize and manage dynamic configuration state

Intuition: only some parts of an app need consensus!

- Lab 2: Highly available view service
- Master election in a distributed FS (e.g. GFS)
- Metadata for sharded services

Implementation: (Multi-)Paxos SMR
Why Chubby?

Many applications need coordination (locking, metadata, etc).

   Every sufficiently complicated distributed system contains an ad-hoc, informally-specified, bug-ridden, slow implementation of Paxos

Paxos is a known good solution

(Multi-)Paxos is hard to implement and use
How to do consensus as a service

Chubby provides:

- Small files
- Locking
- “Sequencers”

Filesystem-like API

- Open, Close, Poison
- GetContents, SetContents, Delete
- Acquire, TryAcquire, Release
- GetSequencer, SetSequencer, CheckSequencer
Back to BigTable
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

Tablet Server

Chubby

GFS
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS

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

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
  - Or, some columns in each
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
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS

Chubby
BigTable components

Client

Master

Chubby

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS
Master

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

Master

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

Chubby

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS
BigTable components

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
BigTable components

Where is the METADATA tablet for table $T$ row $R$?

Client

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

GFS
BigTable components

Client

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

GFS

Chubby
BigTable components

Client

Tablet server 3

Master

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
BigTable components

Client

Tablet Server

Tablet Server

Tablet Server

Tablet Server

Master

Tablet Server

Tablet Server

Tablet Server

Chubby

GFS
BigTable components

Client

Master

Read table $T$ row $R$

Tablet Server Tablet Server

Tablet Server Tablet Server

Tablet Server Tablet Server

Chubby

GFS
BigTable components

- Client
- Master
- Tablet Server
- GFS

Row
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