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, Robert E. Gruber

OSDI 2006

Presenter: Nghia Vo
Motivation

• Google scale:
  • Lots of requests
  • Needs suitable system to back services like Webpages, Emails, Maps...

• No commercial service big enough by then
• In-house means well suited for future scaling
• Built on top, relies on existing infrastructure (GFS, Chubby, etc...)
Model

- Multidimensional map
- (Row string, column string, timestamp integer) as key
  - Value is an arbitrary string
- Sparse row, column oriented
- Distributed and persistent
• **Rows (string):**
  - Ordered lexicographically
  - Row read/write is atomic
    - Exclusive lock
  - Range of rows is called tablet
• **Tablet:**
  - Dynamically partitioned
  - Unit for distribution
- Column (string)
- Family: qualifier naming
- Family is unit for access control.
  - Client might only access some families within a row.
- Small number of families and rarely change.
- Unbounded number of columns.
- Timestamp (int)
  - Store each version of the data
  - Can set to store last k-values
  - Can set to store values within the last few hours, etc...
Webtable:
- Row: reverse URL
  - Helps with locality
- Column: content, languages, anchor, etc...
- Keep contents for different time stamp
Service/APIs

• Metadata operation:
  • Create/delete table, define column families, change metadata like access control

• Read:
  • Scanner API provides read (batch mode)
  • Filtering:
    • Support regex filter for columns
    • Client-supplied script for more filtering option

• Write:
  • Set data in some cells on a row
  • Delete some cells or all on a row
- One master
  - Meta operation
  - Monitor tablet server
  - Lightly loaded
- Multiple tablet servers
  - Handle read/write request
  - Split overgrown tablets
- Linked-to-client library
  - Cache tablet location
- SSTable, Tablet and GFS:
  - How BigTable is formatted/stored
- Chubby and Tablet Server:
  - How to access BigTable
SSTable & Tablet

- SSTable:
  - Immutable map for key-value store
  - Provide key range lookup
- Tablet:
  - Consists of multiple SSTables
  - Contains range of rows
  - Tablets do not overlap
  - Unit of distribution:
    - Stored in GFS
    - Must be assigned to a tablet server before being served
Tablet directory

- 3-level B+ tree design
- Leaf level: actual data tablets
- 1st and 2nd level are METADATA tablets
- METADATA tablet stores:
  - Key range -> tablet location
  - Root tablet stored in Chubby
Client’s perspective

- Request for a key
- Ask for tablet location
  - root tablet (in Chubby)
  - METADATA tablet
  - UserTable
- Tablet location = tablet’s ID & its end row
- Caches tree for performance
- When cache stales, trace back and repeat.
Tablet assignment

- Each tablet is assigned to one tablet server
- Master server’s job:
  - Track tablet -> server
  - Remember unassigned tablets
  - Remember live servers
  - Assign whenever possible
Tablet operation

- Master handles:
  - Create tablets
  - Merge tablets
  - Delete tablets

- Tablet server handles:
  - Split tablet:
    - Write new tablet information to parent METADATA tablet
  - Send notification to master
Tablet server liveness

- Tablet server asks Chubby:
  - Gets exclusive lock on a uniquely named file
  - Holds lock as long as it serves tablets

- Tablet server is considered bad:
  - File isn’t locked on Chubby
  - Master can acquire the server’s lock but can’t pings tablet server

- Result:
  - File is deleted, server terminates.
Master server liveness

• When master starts:
  • Ask Chubby for a master lock
  • Scan Chubby server directory for live tablet server
  • Ask tablet servers for tablets being served
  • Scan METADATA tablet for existing tablets
  • Assign accordingly
• Tablet server needs:
  • GFS stores existing tablets
  • In-memory memtable
    • Construct from tablet log in GFS
    • Stores committed update from redo point
    • Due to immutability of SSTable
Read and Write

• Require authorization from Chubby
• Read:
  • Merge SSTable in GFS with memtable
• Write:
  • Write to tablet log then to memtable
  • What happens when memtable gets large?
Why compaction:
- SSTable is immutable
- Must store commit log for writing request
- Memtable size grows as well
Minor compaction

- When memtable gets large:
  - Converts to SSTable
  - Write to GFS
  - Allow incoming read/write
Merging compaction

- Periodically executed
- Read converted memtables and related SSTables
- Merges into one SSTable
- Safely delete old SSTables
• Merging compaction that:
  • Rewrites all SSTables into one within a tablet
  • Contains no deleted information or data

Major compaction
Major compaction

- Use case:
  - Might exist a sensitive data
  - Its delete command written to GFS
  - Major compaction gets rid of the data and free resource
Refinement

- Locality Group:
  - By client
  - Combine families that are frequently accessed together into a SSTable
- Compressing each LG’s SSTable separately:
  - Reduce size (10-1)
  - Only decompress one SSTable when access that LG
- Declare LG as “load-to-memory”:
  - Load SSTable lazily into tablet server memory
Refinement

• Caching:
  • By tablet server
  • Cache K/V pair
  • Cache SSTable blocks, fast access to neighbors

• Bloom Filter:
  • Client creates a hash for locality group
  • Check whether a K/V exists in a SSTable
  • Helpful to avoid disk read

• Immutability of SSTable can be exploited
  • Example:
    • When tablet splits, its children can use its SSTable
Refinement

• Recovery speed up:
  • Tablet can be assigned to a new server
  • Perform minor compactions on the old server before transferring

• Commit-log implementation:
  • Tablets on the same server share commit log.
  • Reduce number of files on GFS
  • Complicated recovery, since tablet can be assigned to different server
Performance

• Setup:
  • N = # of clients = # of tablet servers
  • N <= 500
  • 1786 machines:
    • Each runs a GFS server
    • Some runs a tablet server, client process, other job’s process
  • Choose # of row keys so that each test read/write ~ 1 GB to each tablet server
Throughput

- Number of 1000-byte value r/w per second
- Sequential
  - read/write key from range
- Random
  - Read/write key uniformly chosen
  - Read (mem)
    - locality group in-memory optimization
- Scan
  - Sequential read with RPC optimization

<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>
Aggregated throughput

- Increasing but not linearly!
  - CPU (bottleneck of tablet server)
  - Network
    - Read 64KB SSTable per 1KB value R/W
    - Only good for sequential R/W and random read (mem)
- Load balancing doesn’t work perfectly
  - Load is dynamic as benchmark goes
  - Limited due to tablet movement
Conclusion

• Scale well
• But credits due to refinement as well
• Exists HBase (open source) that model BigTable
• Thanks for Google’s infrastructure:
  • Easy for optimization on lower-level
  • Well suited for Google’s projects:
    • Maps
    • Personalized Search
    • Etc...
Q & A
Vs Relation DB (SQL)

- Hard to compare.
- From model perspective:
  - By default, index lexicographically on row key
    - Divided into ranges, similar to B+ tree
  - Tighter control on column family
  - Versioning-focused by timestamp key
  - RDBM is more structured, support queries, data aggregation
- From distributed system perspective:
  - Load balancing on tablet/tablet server (by GFS)
  - Tablet serving (by GFS and SSTable)
  - Concurrency control on tablet/tablet server (by Chubby)
Quiz

• Under which dimension (row/column/time) is BigTable lexicographically ordered?

• What is a tablet again?

• How exactly is tablet distributed and served to client!?!?
Quiz

• In few words:
  
  • What does GFS store?
  
  • What is the relationship between Chubby and tablet server?
  
  • What does the master server do?