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

Google, Inc.

Presented by:
Emily Sassano
Bigtable

• Super high scalability
  – Dynamic control over data layout and format
• High availability by High Replication Datastore (HRD)
  – Synchronous write on multiple data centers
• Supports strong consistency among multiple rows
Bigtable

• Scalable, distributed, highly-available and structured storage
  - Semi-structured data
  - Multi-level map
  - Self-managing
    • Servers can be added/removed dynamically
    • Servers adjust to load imbalance

• Consistency
  - Strong consistency for a single row
  - Eventually consistency for multi-row level

• Google
  - In production starting April 2005
  - Searching the web, YouTube, Google Earth, Google Finance, etc (over 60 Google products).
  - The largest is used on 3000TB of data
Bigtable

- Primary database technology used at Google
  - Reliability over thousands of nodes
- Non-relational
- Diverse requirements
- \((\text{row, column, timestamp})\rightarrow\text{string}\)
  - Built to meet the needs of many of the teams at Google
  - A focus was for search
  - Need huge read/write bandwidth
- Arbitrary keys
- Rows partitioned lexicographically into tables
- Lookup, scan, row-atomic write
A single table can be huge! Too large for most commercial databases
- Petabytes of data
- Over thousands of servers

Rows → In a very large table are given to different servers in chunks
- Rows that are close together (usually) related and will more likely end up on the same server
- Table is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes

(row:string, column:string, time:int64) → string
Bigtable Data Model

- Distributed multi-dimensional sparse map
- Key value data storage
- A row has a Key and Columns
- Sorted by Key
  - In Lexical order
  - Enables range query application
Webtable Example

- Many applications and users need the same data
- Store webpages and related information: billions of URL's with many versions/page
  - Row keys: URL's
  - Columns: various aspects of the webpages
  - Contents of webpages
    - Contents: under the timestamps when they were fetched
Other Applications

• Per-user data: Millions
  – User preference settings, recent queries and search results

• Geographic data: 100TB+ of image data
  – Physical entities, roads, satellite imagery, annotations,...
Tablets

- Large tables broken into tablets at row boundaries
  - Tablet holds contiguous range of rows
    - Client can often choose row keys to achieve locality
  - Aim for ~100MB to 200MB of data per tablet
- Serving machine responsible for ~100 tablets
  - Fast recovery when a machine fails:
    - 100 machines each pick up 1 tablet from failed machine
  - Fine-grained load balancing
    - Migrate tablets away from overloaded machine
    - Master makes load-balancing decisions
Automatic Scalability
Automated Sharding

- Table has too many write requests to a tablet
- Needs to multiply
- Now we have two sets of tablet servers
- Load is now distributed
- This happens automatically
- You don't have to think about if you need to increase the number of servers or tablets
- End up with two independent tablets (each roughly half)
- Can move the tablets to different machines
Automatic Scalability
Automated Sharding

Tablet splitting with randomly distributed keys

writes
Dog
Zebra
Gopher
Walrus

Tabby, our hot tablet server
The split happens

MUST... SPLIT...
I handle A-L

SHAZAM
Life is good
And I handle M-Z

Dog & Gopher
Zebra
Walrus
• Clients can control the locality of their data through careful choices in their schemas
Locality Groups

• Clients can group multiple column families together into a locality group (segregate columns from other columns)

• Useful for when you need to scan in columns that are rather small (language and rank of a webpage)

  – Your search is then proportional to only the data in these columns rather than all the columns
Locating Tablets

• Tablets move around from server to server, given a row how do the clients find the right machine?
  – Need to find the tablet whose row range covers the target row

• Store spacial tablet containing the tablet location info in Bigtable cell itself
Locating Tablets

- 3-level hierarchical lookup scheme

![Diagram of tablet location hierarchy]

Figure 4: Tablet location hierarchy.
• Amazon's SimpleDB
  - Similar to Google's Bigtable
  - Most cost effective than DynamoDB
  - Just a thought, might be something to look at if you are still having DB issues