Bigtable: A Distributed Storage System for Structured Data by Google

SUNNIE CHUNG
CIS 612
Google Bigtable

- A distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers.

- Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance.

- These applications place very different demands on Bigtable, both in terms of:
  - Data size (from URLs to web pages to satellite imagery) and
  - Latency requirements (from backend bulk processing to real-time data serving).
Google Bigtable

- Wide Applicability
- Scalability
- High Performance
- High Availability
- Bigtable is used by:
  - Google Analytics
  - Google Finance
  - Google Earth
  - Google Search / Personalized Search
  - Orkut etc.
Google Bigtable is

- a sparse,
- distributed,
- persistent,
- multi dimensional sorted map which is indexed by
  - a Row Key,
  - a Column Key and
  - a Timestamp,
- where each value in the map is an un-interpreted array of bytes.
Data Model

(row key : string, column : string, timestamp : int64) → string

Fig: A slice of an eg table (consider name Webtable) storing web pages
<table>
<thead>
<tr>
<th>Row Key</th>
<th>Time Stamp</th>
<th>ColumnFamily contents</th>
<th>ColumnFamily anchor</th>
<th>ColumnFamily people</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;com.cnn.www&quot;</td>
<td>19</td>
<td></td>
<td>anchor:cnnsi.com = &quot;CNN&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;com.cnn.www&quot;</td>
<td>18</td>
<td></td>
<td>anchor:my.look.ca = &quot;CNN.com&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;com.cnn.www&quot;</td>
<td>16</td>
<td>contents:html = &quot;&lt;html&gt;...&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;com.cnn.www&quot;</td>
<td>15</td>
<td>contents:html = &quot;&lt;html&gt;...&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;com.cnn.www&quot;</td>
<td>13</td>
<td>contents:html = &quot;&lt;html&gt;...&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;com.example.www&quot;</td>
<td>15</td>
<td>contents:html = &quot;&lt;html&gt;...&quot;</td>
<td></td>
<td>people:author = &quot;John Doe&quot;</td>
</tr>
</tbody>
</table>
Bigtable Rows and Column Families

Rows:
- Arbitrary strings; currently up to 64KB in size allowed; typical size for string data 10-100 bytes
- Every read or write is atomic
- Lexicographic ordering of keys
- Row range definition for a table is dynamic called a Tablet

Column Families:
- Column keys are grouped into sets called column families
- **Syntax:** `family:qualifier`
- **Eg:** “anchor:my.look.ca”: in the figure from previous slide
- `language:LanguageID`: This stores the language for webpage
Timestamps

**Timestamps:**

- Multiple versions of the data are allowed in a single cell; indexed by timestamps
- Timestamps are 64 bit integer values
- Client or Bigtable can generate timestamps for the data
- Stored in decreasing order keeping newest data on top of storage
- Bigtable column family can have their own garbage collection mechanism defined to help keep only recent ‘n’ no. of data values
Bigtable API

Bigtable API Provides functions for

• Creating and Deleting tables
• Creating and Deleting column families
• Changing cluster, table and column family metadata
• Access to data rows for write and delete operations
• Scan through data for particular column families, rows and data values with filters applied
• Batch and atomic writes
Bigtable Building Blocks

Google File System
Google SSTable
Chubby
Bigtable Building Blocks

Google File System:
To Store log and data files

Google SSTable:
Used internally to store data in Bigtable

Chubby:
Paxos based system for consensus in network of unreliable processors
Provides a namespace to store Directories and small files. Each Dir or file can be used as a lock and every access is atomic
Chubby Unavailable, Bigtable Unavailable
Implementation

Three Major components:

A Library that is linked to every client

One master server:
  Responsible for assigning tablets to Tablet Servers

Many Tablet servers:
  Dynamically added and removed from a cluster to accommodate changes in workloads

Recall: Tablet: Every table in Bigtable is dynamically partitioned on the basis of row keys / range. Each row range is called a tablet.
Implementation Cntd.

One master server:
- Responsible for assigning tablets to Tablet Servers
- Detecting the addition and expiration of Tablet Servers
- Balancing Tablet Server Load
- Garbage collection of files in GFS

Note: No client practically communicates with Master Server

Many Tablet servers:
- Dynamically added and removed from a cluster to accommodate changes in workloads
- Each Tablet Server manages a set of Tablet
- Handles Read and Write requests to the Tablets
- Split tablets when they have grown too large
Fig: Tablet Location Hierarchy
Tablet Assignment

**Master Server and Tablet Assignment:**
- Assign a tablet to only one Tablet Server at a given time
- Keep track of all the Tablet servers including tablets assignments and un-assignments
- Master is responsible for:
  - Detecting when a tablet server is no longer serving its tablets
  - Keep track of lock / Session with Chubby

**Steps Taken by Master at Startup:**
- Grab a unique Master lock in Chubby
- Scan the servers directory to find live servers
- Communicate with every live server to check assignments
- Scan the Metadata table to learn the set of Tablets
Tablet Serving

Write Operation on Tablet
When write operation arrives at Tablet:
  • Server checks for well formedness and proper authorization
  • Valid mutation is written to the commit log
  • Group commit is used to improve the throughput
  • After write is committed, contents are inserted to memtable

Read Operation on Tablet
  • Checked for Well formedness and proper authorization
  • Valid read operation is executed on a merged view of sequence of SSTables and memtable

When write operation arrives at Tablet:
  • Server checks for well formedness and proper authorization
  • Valid mutation is written to the commit log
  • Group commit is used to improve the throughput
  • After write is committed, contents are inserted to memtable
Tablet Compactions

**Minor Compactions:**
- Shrinks memory usage of Tablet server
- Reduces amount of data that has to be recreated from log in case of failure

**Merging Compactions:**
- Performed in background on SSTables created by minor compactions

**Major Compactions:**
- All SSTables are merged and rewritten to one SSTable.
- Bigtable performs this operation regularly.

Everything that happens with data here is **atomic!!!**
Refinements

- Locality Groups
- Compression
- Caching for Read performance
- Bloom Filters
- Commit Log Implementation
- Speeding up Tablet recovery
- Exploiting immutability
Summary

Bigtable is a distributed storage system for storing structured data at Google.

In operation since 2005, by August 2006 more than 60 projects are using Bigtable.

Effective performance, high availability and scalability are the key features for most of the clients.

Control over architecture allows Google to customize the product as needed.

Use by old and new clients demonstrates that Bigtable architecture works.
This Lecture Notes from the paper published by Google in OSDI 2006

http://grail.csuohio.edu/~sschung/cis612/googlebigtable-osdi06.pdf