

BigTable

CSE-291 (Cloud Computing)

Fall 2016

Data Model

- Sparse, distributed persistent, multi-dimensional sorted map
- Indexed by a row key, column key, and timestamp
- Values are uninterpreted arrays of bytes
- (row:string, column:string, time:int64) → string

Slice of an Example Table

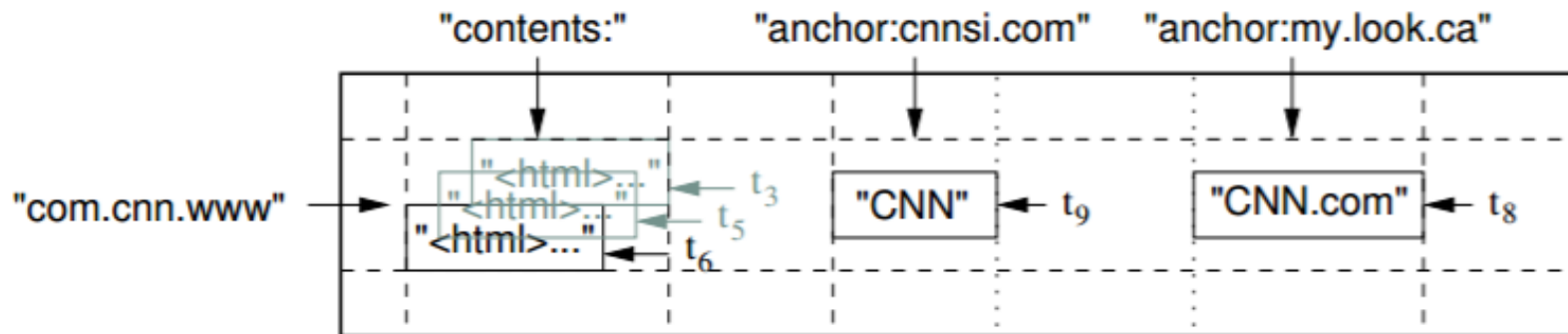


Figure 1: A slice of an example table that stores Web pages. The row name is a reversed URL. The **contents** column family contains the page contents, and the **anchor** column family contains the text of any anchors that reference the page. CNN's home page is referenced by both the Sports Illustrated and the MY-look home pages, so the row contains columns named **anchor:cnnsi.com** and **anchor:my.look.ca**. Each anchor cell has one version; the contents column has three versions, at timestamps t_3 , t_5 , and t_6 .

Rows

- Arbitrary strings
- Reads and writes of rows are atomic (even across columns)
 - Makes it easier to reason about results
- Data is lexicographically sorted by row
- Row range partitioned into *tablets*
 - Used for distribution and load balancing
 - Reads of short row ranges are efficient
 - If data has good locality w.r.t. tablets, efficiency boost

Column Families

- Not columns, groups of columns
- Unit of access control, disk and memory accounting
- Column families group column keys
- Data within column family is usually same type, compressed together
- Columns within column family rarely change
- Relatively few column families

Timestamps

- Cells are versioned
- Timestamps are times in microseconds
 - Or, alternately, user can assign, e.g. version number, etc
 - Need user-assigned unique timestamps, if want to avoid collisions
- Automated garbage collection
 - Most recent n
 - Within m amount of time

API

- Create and destroy tables and column families
- Change metadata, e.g. access control, etc
- Write or delete values
- Iterate across rows
- Iterate over subset of data
- Single row transactions (read-update-write)
- Cells to be used as counters
- Client-provided server-side scripts for transformation, filtering, summarization etc.

Building blocks

- Uses GFS for log and data files
- Shares servers with other applications

Building blocks: *SSTables*

- *SSTable* file format used to store tables
 - *Sorted String Table*
 - Key-value pairs, sorted by key
 - Blocks of data followed by binary-searchable index at end
 - Index in memory, one seek to data
 - Or, load all into memory
 - Lookup value by key, iterate over key/value pairs in range

Building Blocks: Chubby

- Distributed lock service
- 5 active replicas, one is master
 - Only master serves requests
 - Needs majority to work
 - Paxos based
- Namespace is directories and tiny files
 - Directories and files can be used as locks
- Locks are leased and callbacks can be requested
- Used to ensure one active master, location of tablet servers, schema information, access control lists

Master Server

- Master server
 - Assigns tablets to tablet servers
 - Detecting and managing changes to tablet servers
 - Load balancing across tablet servers

Tablet Server

- Tablet server
 - Can be dynamically added and removed
 - Manages read and writes
 - Split tablets that are too large

BigTable Cluster

- Cluster stores Tables
- Tables store Tablets
- Tablets store Row Range
- Initially, one Tablet per Table, then split with growth

Tablet Location

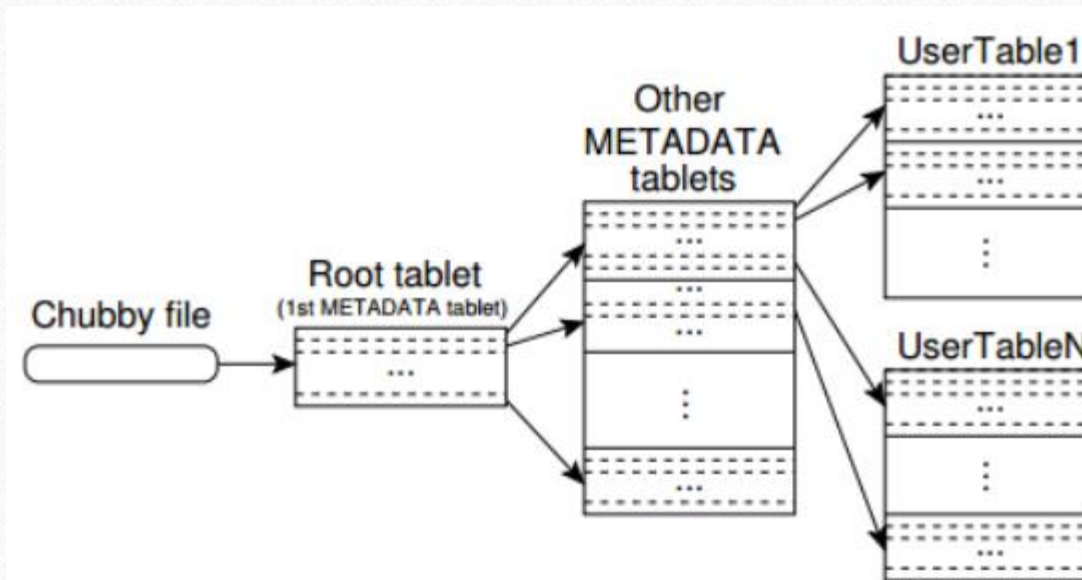


Figure 4: Tablet location hierarchy.

Tablet Assignment

- Each tablet lives on one server at a time
- Master keeps track of live tablet servers and assignments
- Chubby used to keep track of tablet servers
 - Master monitors Chubby directory
- Tablet servers can't serve if they lose exclusive lock on tablet
- Tablets reassigned when not reachable
 - Notify master if lost lock
 - Master heartbeats (asks for status of lock from tablet server)

Tablet Serving

- Persistent stores redo record stored in GFS
- Commit log
- Recent updates in in-memory *memtable*
- Older updates in *sstable*

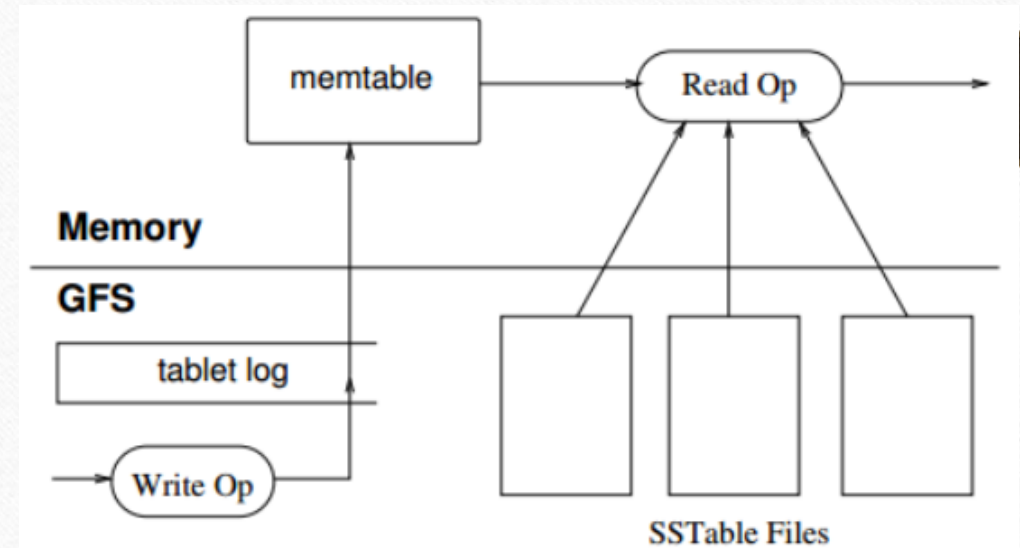


Figure 5: Tablet Representation

Locality Groups

- Grouping of multiple column families together
 - Separate SSTable for each locality group
- Makes it faster to access data that is accessed together

Compression

- Per block or across blocks
 - Per block enables small portions to be read without decompression of larger block
- Sometimes 2 pass schemes
- Want to emphasize speed over compression

Read Caching

- Scan cache
 - Key-value pairs from sstable
- Block cache
 - Sstable blocks read from GFS

Bloom Filters

- Reads need to read from all SStables that make up table
- Bloom filters reduce the number that are accessed by don't have matching row/column pair.
 - Ditto for non-existent pairs

Commit Log

- Logs are written concurrently
 - Different purposes
 - Different files would mean different locations on disk
- Use one per tablet server, not one per tablet
 - Reduces the number of files written, improves seek locality, reduces overhead, etc.
- Complicates recovery
 - Few log entries relate to any one tablet server
 - Parallel sort by key first, then entries for one server are together.

Tablet Migration

- Process
 - Compact
 - Freeze
 - Compact
 - Migrate
- Log is clean for move with only a short freeze time

Immutability

- Only memtable allows reads and writes
- Everything else is versioned
- Allows asynchronous deletes
- Mitigates need for locking.