

Distributed Systems

18. BigTable

Paul Krzyzanowski
pkx@cs.rutgers.edu

BigTable

- Highly available distributed storage for structured data
- Built with structured data in mind
 - URLs: content, metadata, links, anchors, page rank
 - User data: preferences, account info, recent queries
 - Geography: roads, satellite images, points of interest, annotations
- Large scale
 - Petabytes of data across thousands of servers
 - Billions of URLs with many versions per page
 - Hundreds of millions of users
 - Thousands of queries per second
 - 100TB+ satellite image data

Uses

- At Google, used for:
 - Google Analytics
 - Google Finance
 - Orkut
 - Personalized search
 - Writely
 - Google Earth & Google Maps
 - Dozens of others...

A big table

- BigTable is NOT a relational database
- BigTable appears as a large table
 - "A BigTable is a sparse, distributed, persistent multidimensional sorted map"

	"language:"	"contents:"		
com.aaa	EN	<DOCTYPE html PUBLIC...		
com.cnn.www	EN	<DOCTYPE HTML PUBLIC...		
com.cnn.www/TECH	EN	<DOCTYPE HTML>...		
com.weather	EN	<DOCTYPE HTML>...		

Webtable example

*Bigtable, OSDI 2006

Table Model

- (row, column, timestamp) → cell contents
 - Contents are arbitrary strings (arrays of bytes)

	"language:"	"contents:"		
com.aaa	EN			
com.cnn.www	EN	<div style="border: 1px solid black; padding: 2px;"> <DOCTYPE html... </div>	← t ₁	
com.cnn.www/TECH	EN	<div style="border: 1px solid black; padding: 2px;"> <DOCTYPE html... </div>	← t ₇	
com.weather	EN	<div style="border: 1px solid black; padding: 2px;"> <DOCTYPE html... </div>	← t ₁₅	

Webtable example

Tablets: Pieces of a Table

- Row operations are atomic
- Table partitioned dynamically by rows into tablets
 - Unit of distribution and load balancing
 - Nearby rows will usually be served by the same server
 - Accessing nearby rows requires communication with a small # of machines
 - Select row keys to ensure good locality
 - E.g., reverse domain names:
com.cnn.www instead of www.cnn.com

Table splitting

- A table starts as one tablet
- As it grows, it splits into multiple tablets
 - Approximate size: 100-200 MB per tablet by default

	"language:"	"contents:"		
com.aaa	EN	<DOCTYPE html PUBLIC...		
com.cnn.www	EN	<DOCTYPE HTML PUBLIC...		
com.cnn.www/TECH	EN	<DOCTYPE HTML>...		
com.weather	EN	<DOCTYPE HTML>...		

tablet

Splitting a tablet

	"language:"	"contents:"		
com.aaa	EN	<DOCTYPE html PUBLIC...		
com.cnn.www	EN	<DOCTYPE HTML PUBLIC...		
com.cnn.www/TECH	EN	<DOCTYPE HTML>...		

com.weather	EN	<DOCTYPE HTML>...		
com.wikipedia	EN	<DOCTYPE HTML>...		
com.zcorp	EN	<DOCTYPE HTML>...		
com.zoom	EN	<DOCTYPE HTML>...		

Columns and Column Families

- Column Family
 - Group of column keys
 - Column family is the basic unit of data access
 - Data in a column family is typically of the same type
 - Implementation compresses data in the same column family
- Operations
 - (1) Create column family
 - (2) Store data in any key within the family
- Column families will typically be small
 - ≤ hundreds of keys; a table may have an unlimited #
- Identified by
 - family:qualifier

Column Families: example

- Three column families
 - "language:" – language for the web page
 - "contents:" – contents of the web page
 - "anchor:" – contains text of anchors that reference this page.
 - www.cnn.com is referenced by Sports Illustrated (snisi.com) and MyLook (mlook.ca)
 - The value of ("com.cnn.www", "anchor:snisi.com") is "CNN", the reference text from snisi.com.

	"language:"	"contents:"	anchor:snisi.com	anchor:mlook.ca
com.aaa	EN	<DOCTYPE html PUBLIC...		
com.cnn.www	EN	<DOCTYPE HTML PUBLIC...	"CNN"	"CNN.com"
com.cnn.www/TECH	EN	<DOCTYPE HTML>...		
com.weather	EN	<DOCTYPE HTML>...		

sorted ↓

Timestamps

- Each column family may contain multiple versions
- Version indexed by a 64-bit timestamp
 - Real time or assigned by client
- Per-column-family settings for garbage collection
 - Keep only latest *n* versions
 - Or keep only versions written since time *t*
- Retrieve most recent version if no version specified
 - If specified, return version where timestamp ≤ requested time

API: Operations on BigTable

- Create/delete tables & column families
- Change cluster, table, and column family metadata (e.g., access control rights)
- Write or delete values
- Read values from specific rows
- Iterate over a subset of data in a table
 - All members of a column family
 - Multiple column families
 - E.g., regular expressions, such as anchor: *.cnn.com
 - Multiple timestamps
 - Multiple rows
- Atomic read-modify-write row operations
- Allow clients to execute scripts (written in Sawzall) for processing data on the servers

Implementation: Supporting Services

- GFS
 - For storing log and data files
- Cluster management system
 - For scheduling jobs, monitoring health, dealing with failures
- Google SStable
 - Internal file format
 - Provides a persistent, ordered, immutable map from keys to values
 - Memory or disk based

Implementation: Supporting Services

- Chubby
 - Highly-available & persistent distributed lock (lease) service
 - Five active replicas; one elected as master to serve requests
 - Majority must be running
 - Paxos used to keep replicas consistent
 - Namespace of files & directories. Each file or directory can be used as a lock
- Chubby is used to:
 - Ensure there is only one active master
 - Store bootstrap location of BigTable data
 - Discover tablet servers
 - Store BigTable schema information
 - Store access control lists

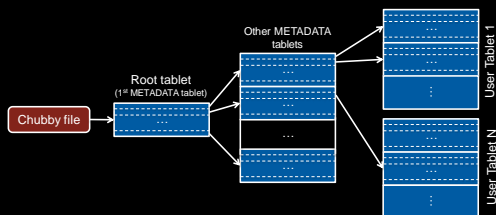
Implementation

1. Many tablet servers
 - Can be added or removed dynamically
 - Each manages a set of tablets (typically 10-1,000 tablets/server)
 - Handles read/write requests to tablets
 - Splits tablets when too large
2. One master server
 - Assigns tablets to tablet server
 - Balances tablet server load
 - Garbage collection of unneeded files in GFS
 - Schema changes (table & column family creation)
3. Client library

Client data does not move through the master
Clients communicate directly with tablet servers for reads/writes

Implementation

- Three-level hierarchy
 - Balanced structure similar to a B+ tree
 - Root tablet contains location of all tablets in a special METADATA table
 - METADATA table contains location of each tablet under a row key = f(tablet table ID, end row)



Implementation

- Tablet assigned to one tablet server at a time
- Chubby keeps track of tablet servers
 - When tablet server starts:
 - It creates & acquires an exclusive lock on a uniquely-named file in a Chubby *servers* directory
 - Master monitors this directory to discover tablet servers
- When master starts:
 1. Grabs a unique master lock in Chubby (prevent multiple masters)
 2. Scans the *servers* directory in Chubby to find live servers
 3. Communicate with each tablet to discover what tablets are assigned to each server
 4. Scan the METADATA table to learn the full set of tablets
 - Build a set of unassigned tablets – these are eligible for tablet assignment

BigTable Replication

- Each table can be configured for replication to multiple BigTable clusters in different data centers
- Eventual consistency model

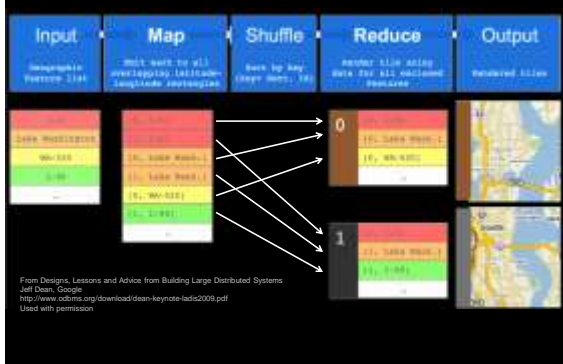
Sample applications

- Google Analytics
 - Raw Click Table (~200 TB)
 - Row for each end-user session
 - Row name: {website name and time of session}
 - Sessions that visit the same web site are sorted & contiguous
 - Summary Table (~20 TB)
 - Contains various summaries for each website
 - Generated from the Raw Click table via periodic MapReduce jobs

Sample applications

- Google Maps / Google Earth
 - Preprocessing
 - Table for raw imagery (~70 TB)
 - Each row corresponds to a single geographic segment
 - Rows are named to ensure that adjacent segments are near each other
 - Column family: keep track of sources of data per segment (this is a large # of columns – one for each raw data image – but sparse)
 - MapReduce to preprocess data
 - Serving
 - Table to index data stored in GFS
 - Small (~500 GB) but serves tens of thousands of queries with low latency

MapReduce for Rendering Map Tiles



The End