Distributed Storage

Wolf-Tilo Balke & Pierre Senellart
IFIS, Technische Universität Braunschweig
IC2, Telecom ParisTech

Google

Detour

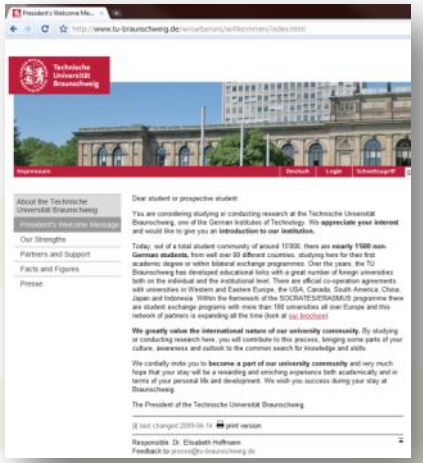
- Currently, Google offers a multitude of services
  - Google Search
  - Google Mail
  - Google Maps
  - Google Earth
  - Google Documents
  - Picasa Web Album
  - etc.

- Thus, Google hosts and actively uses several Petabytes of data!

Google Challenges

Detour

- Google needs to **store** and **access** lots of (semi-)structured data
  - **URLs** and their contents
    - Content, meta data, links, anchors, pageranks, etc.
  - **User data**
    - User preferences, query history, search results
  - **Geographic information**
    - Physical entities (shops, restaurants, etc), roads, annotations, POIs, satellite images, etc.



Bigtable

- **Bigtable**
  - F. Chang et al, **"*Bigtable: A Distributed Storage System for Structured Data*"**, ACM Transactions on Computer Systems (TOCS), Vol 26, Iss 2, June **2008**
  - Bigtable is a high-performance proprietary **database system** used by multiple Google services
    - e.g. used in Google Maps, Google Books, Google Earth, Gmail, Google Code, etc.
    - Uses an abstracted and very flexibly row and column storage model
    - Is based on versioning for updates

Bigtable Requirements

- Originally designed for storing Google's **Web index**
- Special requirements
  - Processes **continuously** and **asynchronously update** different pieces of data
    - i.e. continuous Web crawling
    - Store version, usually access just newest one
    - Multiple version can be used to examine change of data in time
  - Very **high read / write rates** necessary
    - Millions per seconds
  - Support efficient **scanning** of interesting data subsets

Bigtable Requirements

- Additional requirements as usual for web-scale applications
  - Fault tolerant, persistent
  - Use cheap hardware
  - Scale to huge sized infrastructures
    - Support **incremental scaling**
    - Thousands of servers
      - Terabytes of in-memory data
      - Petabytes of disk-based data
  - Self-managing
    - Servers auto-load balance
    - Servers can be dynamically added and removed

Bigtable Cells

- Each distributed Bigtable cluster is responsible for the data of one or multiple applications
  - Called a "cell"
    - Several hundred cells are deployed
    - Cell size range from 10-20 up to thousands machines
    - In 2006, the largest cell was 0.5 PB
      - Now it is probably much larger…

Bigtable Environment

- Bigtable heavily relies on additional systems and concepts
  - **Google File System (GFS)**
    - A distributed and fail-safe file system
    - Physically stores Bigtable data on disks
      - S. Ghemawat, H. Gobioff, S.T. Leung. **"The Google File System",** ACM Symp. Operating Systems Principles, Lake George, USA, 2003
  - **Google Chubby**
    - A distributed lock manager, also responsible for bootstrapping
      - M. Burrows. **"The Chubby Lock Service for Loosely-Coupled Distributed Systems",** Symp. Operating System Design and Implementation, Seattle, USA, 2006
  - **Google MapReduce**
    - Programming model for distributing computation jobs on parallel machines
      - J. Dean, S. Ghemawat. **"MapReduce: Simplified Data Processing on Large Clusters",** Symp. Operating System Design and Implementation, San Francisco, USA, 2004

Bigtable Implementation

- Bigtable is a "database" especially designed to run ontop of GFS
  - Bigtable data model also focuses on appends
    - Assumption: rows are frequently added, but rarely updated
    - Row "updates" will just result in new rows with a different timestamp
  - GFS takes care of replication and load-balancing issues
- To accommodate for Google's applications, Bigtable uses a very flexible data model

Bigtable: Data Model

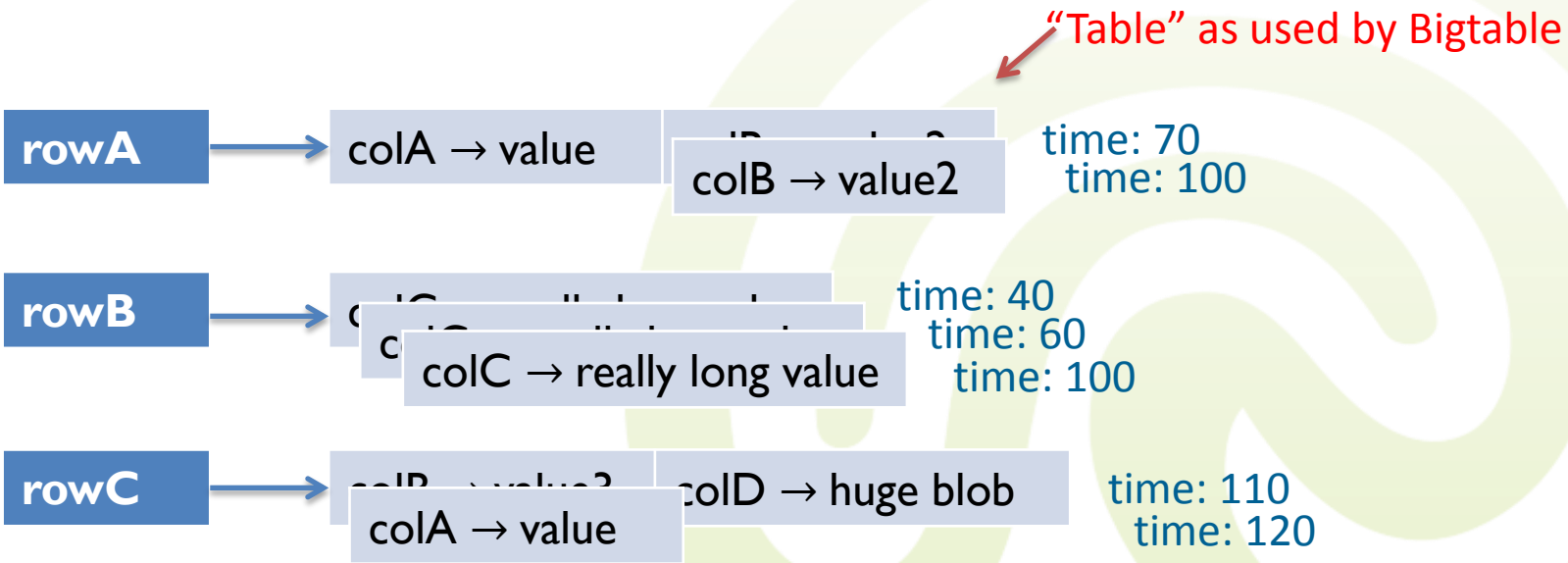
- Don't think of Bigtables as spreadsheet or traditional DB table
  - Unfitting name….
  - e.g. not each row has a fixed size of attributes
    - Not: Each column has a data type
    - Not: Missing values denoted as null

Table as NOT used by Bigtable

	colA	colB	colC	colD
rowA				NULL?
rowB	NULL?			
rowC			NULL?	
rowD				

Bigtable: Data Model

- Instead, Bigtable implements a **multi-dimensional sparse map**
  - Think of columns just as available tags
    - "Cells" are referenced by $(row\_name, col\_name, timestamp)$
  - Each row can use just some columns and story any value
    - Columns are just roughly typed, i.e. binary, string, numeric, …



Bigtable: Data Model

- **Rows**
  - Each row has a **unique name**
    - Name is just an arbitrary **string**
      - e.g. "www.ifis.cs.tu-bs.de"
  - Each access to a **row** is **atomic**
    - Load and store whole rows
  - Rows are **ordered lexicographically**
    - Idea: after partitioning the table, lexicographically similar rows are within the same or a nearby fragment
      - e.g. "www.ifis.cs.tu-bs.de" is close to "www.ifis.cs.tu-bs.de/staff"
  - **Rows will never be split** during partitioning

Bigtable: Data Model

- **Columns**
  - Each column has a two-level name structure
    - **Family** name and **qualifier** name
      - e.g. <family:qualifier>
  - All **column families** must be created explicitly as part of schema creation
    - Columns within a family have usually a similar type
    - Data of a row within a family are often stored and compressed together
  - **Individual columns** can be used by application freely and **flexibly**
    - Individual columns are not part of schema creation
      - Flexible data model
  - **Aims**
    - Have a few (max. 100 (!)) column families which rarely change
    - Let application create columns as needed

Bigtable: Data Model

- **Timestamps**
  - Of each cell, different **versions** are maintained with their respective timestamps
    - 64 Bit integers
  - **Updates** to a cell usually create a new version with the current system time as timestamp
    - But timestamp can also be set explicitly by application
  - During **column family** creation, **versioning options** are provided
    - Either "keep *n* copies" or "keep versions up to the age of *n* seconds"
  - Typical queries ask for timestamp ranges

Bigtable: Data Model

- The base unit of load balancing and partitioning are called **tablets**
  - i.e. **tables** are **split** in multiple **tablets**
  - Tablets hold a **contiguous** range of rows
    - Hopefully, row ordering will result in locality
  - Tablets are **disjoint**
    - No overlapping value ranges
  - **Tablets** are rather large (1 GB by default) and are later stored in **GFS**
    - i.e. tablets will usually have multiple GFS chunks
    - Tablets need to contain full rows
      - A single row should not exceed several hundred MB such that it will fit into a tablet…

Bigtable - API

- Bigtable provides only very simple **native API interfaces** to applications
  - e.g. in C++ or Python
  - No complex query language like SQL
  - API can
    - Create and delete tables and column families
    - Modify cluster, table, and column family metadata such as access control rights,
    - Write or delete directly addressed values in Bigtable
      - Supports just single row transactions (i.e. read-modify-write)
      - No multi-row transactions
    - Look up values from individual rows
    - Iterate over a subset of the data in a table,
      - Can be restricted to certain column families or timestamps
      - Relies on regular expressions on row and columns names

Bigtable - API

Summary

- **Recap**
  - Semi-flexible schemas are supported
  - A table consist of named **rows** and **columns**
    - All data cells are **versioned** with **timestamps**
    - Columns are grouped in **column families** which are defined in the schema
      - Families are usually stable during application life
    - Columns can be dynamically used and added by applications as they seem fit
    - As a result, table is **very sparse**
      - i.e. it resembles a **multi-dimensional map**

  - Tables are broken down into **tablets**
    - Tables hold a **continuous** and **ordered non-overlapping row name** range
    - **Horizontal fragmentation**

Bigtable

Detour

- Application 1: **Google Analytics**
  - Enables webmasters to analyze traffic pattern at their web sites.
  - Provides statistics such as:
    - Number of unique visitors per day and the page views per URL per day
    - Percentage of users that made a purchase given that they earlier viewed a specific page
  - How is it done?
    - A small JavaScript program that the webmaster embeds in their web pages
    - Every time the page is visited, the program is executed
    - Program records the following information about each request
      - User identifier
      - The page being fetched

Bigtable

Detour

- Application 2: **Google Earth & Maps**
  - Functionality: Storage and display of satellite imagery at different resolution levels
  - One Bigtable stores raw imagery (~ 70 TB):
    - Row name is a geographic segments
      - Names are chosen to ensure adjacent geographic segments are clustered together
    - Column family maintains sources of data for each segment.
  - There are different sets of tables for serving client data, e.g., index table

Bigtable

Detour

- Application 3: **Personalized Search**
  - Records user queries and clicks across Google properties
  - Users browse their search histories and request for personalized search results based on their historical usage patterns
  - One Bigtable
    - Row name is userid
    - A column family is reserved for each action type, e.g., web queries, clicks
    - User profiles are generated using MapReduce.
      - These profiles personalize live search results
    - Replicated geographically to reduce latency and increase availability

Bigtable

Summary

- Google Bigtable is a NoSQL database
  - **No complex query language supported**
    - Mainly based on scans and direct key accesses
  - **Single table data model**
    - **No joins**
    - **No foreign keys**
    - No integrity constraints
  - **Flexible schemas**
    - Column may be added dynamically
  - Usually, Bigtable is not a direct replacement for a distributed database

HBase

Detour

- **Hbase** is an open-source **clone** of Bigtable
  - http://hbase.apache.org/
  - Created originally at Powerset in 2007
- Hbase is a **Apache Hadoop** subproject
  - Hadoop is strongly supported by Microsoft and Yahoo
  - http://hadoop.apache.org/
  - Hadoop reimplements multiple Google-inspired infrastructure services
    - MapReduce ←Google Map And Reduce
    - Hbase ← Bigtable
    - HDFS ← GFS
    - ZooKeeper ← Chubby