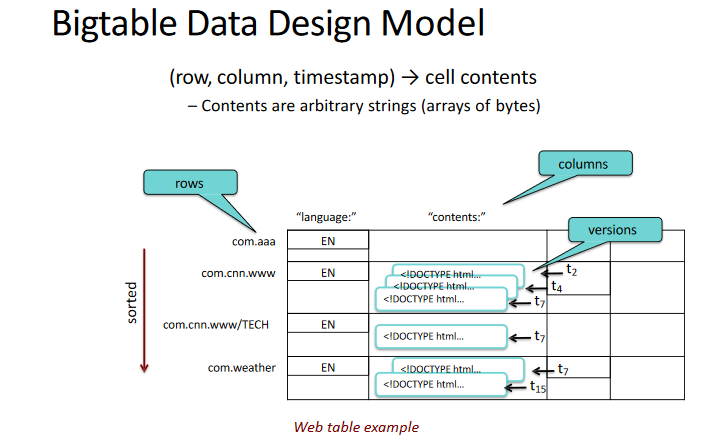
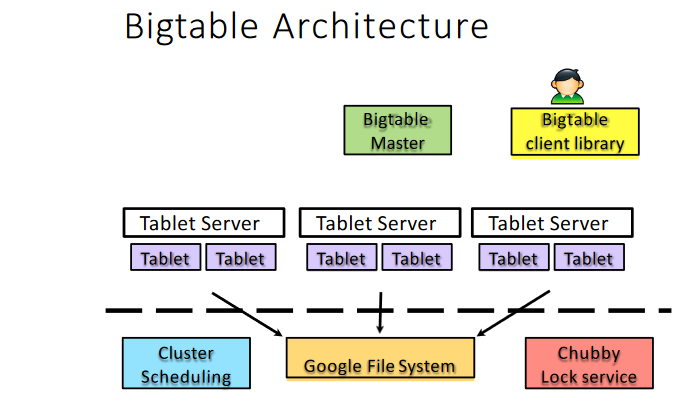
BigTable

* Lots of semi-structured data at Google
  + URLs: Content, crawl metadata, links, anchors
  + Per-user data: User pref settings, queries
  + Geographic locations: Physical entities and stellite image ….
  + Scale of Googles stuff is LAAAARGE
    - Billions if URLs, many versions… hundreds of millions of users, thousands of queries, 100TB+ satellite images
* All this data TOO LARGE for reg databases
* If it wasn’t, still cost V high
* Low-level storage optimization help performance a lot
* Atomic Transactions
  + Transaction- operation composed of sequence of discrete steps
  + All steps must be completed for transaction to be committed (results made permanent)
  + Transaction aborted if all steps not completed, and state of system revert to what it was before.
    - Rollback=reverting to prev state
* BigTable
  + Sparce, distributed, persistent multidimensional sorted map:
  + Fault Tolerant, persistent
  + Scalable (thousands of servers, terabytes of in-memory data, petabyte of disk-based data, millions of reads/writes per second, efficient scan)
  + Column and Column Families
    - Group of column keys, column family is basic unit of data access
    - Data in column family typically same type, implementation compresses data in same column family
  + Operations
    - Create column family -> admin task done when table created
    - Store data in any key within family -> can do anytime
  + Typically be small number of column families
    - Hundreds of column families
    - Table may have unlimited # of columns that are often sparsely populated

Tables and Tablets

* Row operations atomic
* Table partitioned dynamically by rows into tablets
* Tablet=range of contiguous rows
  + Unit of distr and load balancing
  + Nearby rows usually be served by same server
  + Accessing nearby rows require comm with small # machines
  + Need to select row keys to ensure good locality
* Table Splitting
  + Table start as one tablet
  + As grow, split into multiple tablets
* TimeStamps
  + Each column family may contain multiple versions
  + Each version indexed by 64 bit timestamp
  + Per-column-family settings for garbage collection
    - Keep only latest n versions or keep versions written since time t

IMPLEMENTATION

* Many tablet servers coordinate requests to tablets
  + Can add or remove dynamically
  + Each manages set of tablets (10-1000/server)
  + Handlres read/write requests to tablets
  + Splits tablets when too large
* 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)
* Client Library
  + Client data doesn’t move through master
  + Client communicate directly with tablet servers for read/write

HOW BIG DATA STORED ON DATABASES

* In big data have lots of dif data like url, per user data, geographic location NOT EZ to store in traditional db. This data changes quickly, and traditional db don’t handle semi structured data well
* GFS MAP REDUCE AND BIGTABLE are applications from CS Innovation.
* Big table is db version for big data, how can store.
* Atomic Transactions
  + Transaction need to be completed properly. If stuck somewhere in between, roll it back and restart. NO PAUSE AND RESTART. If something start need to finish.