Distributed Systems
[Fall 2012]

Lec 20: Bigtable (cont'ed)

Slide acks: Mohsen Taheriyan

(http://www-scf.usc.edu/~csci572/2011Spring/presentations/Taheriyan.pptx)

Chubby (Reminder)

- Lock service with a file system interface

- Intuitively, Chubby provides locks with possibility to store a bit of data in them, which can be read but not written unless you have a writer’s lock

- It also provides notifications for file updates and others

- Uses Paxos to provide reliability and availability

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

OSDI 2006

Slide acks to: Mohsen Taheriyan

(http://www-scf.usc.edu/~csci572/2011Spring/presentations/Taheriyan.pptx)

Bigtable Description Outline

- Motivation and goals (last time)
- Schemas, interfaces, and semantics (with code) (today)
- Architecture (today)
- Implementation details (today, or you’ll read on your own)

Bigtable Goals (Reminder)

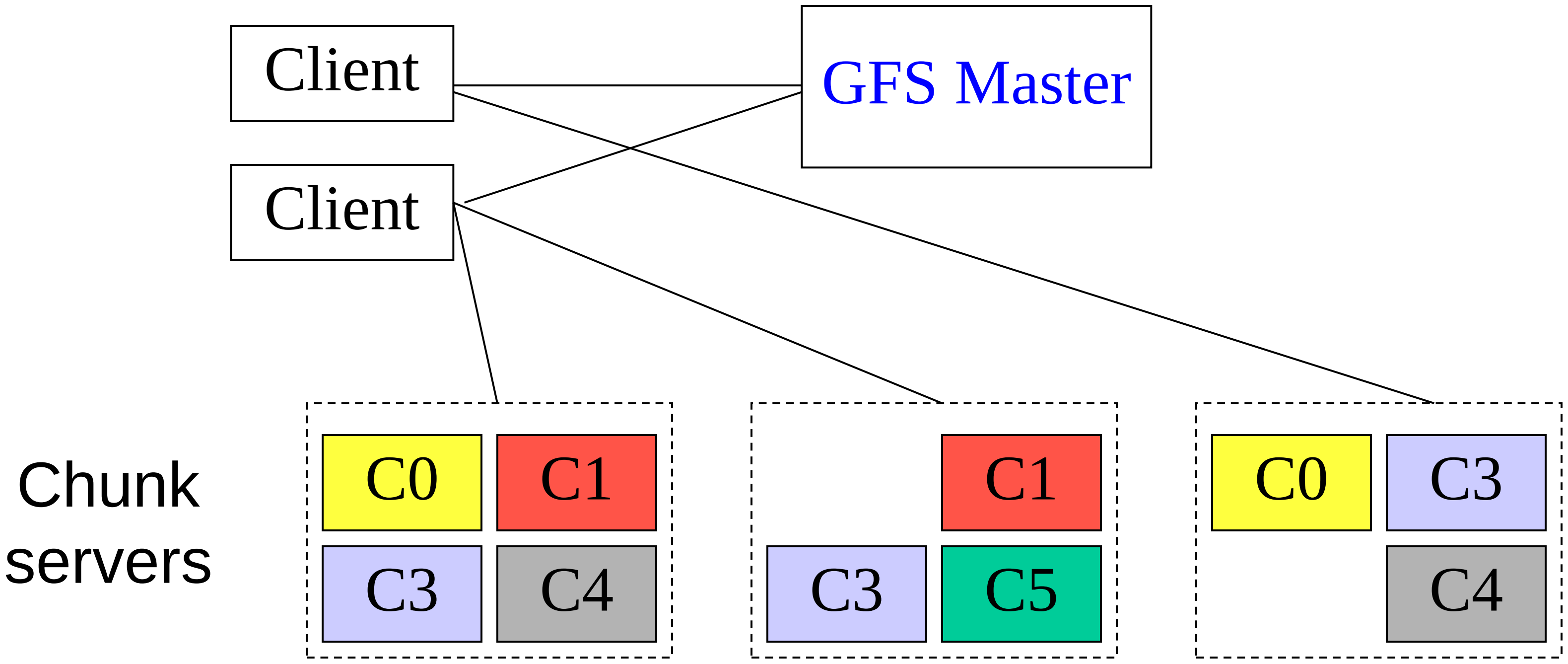
- A distributed storage system for (semi-)structured data
- Scalable
  - Thousands of servers
  - Terabytes of in-memory data
  - Petabyte of disk-based data
  - Millions of reads/writes per second, efficient scans
- Self-managing
  - Servers can be added/removed dynamically
  - Servers adjust to load imbalance
- Extremely popular at Google (as of 2008)
  - Web indexing, personalized search, Google Earth, Google Analytics, Google Finance, …

Background

- Building blocks
  - Google File System (GFS): Raw storage
  - Scheduler: Schedules jobs onto machines
  - Chubby: Lock service

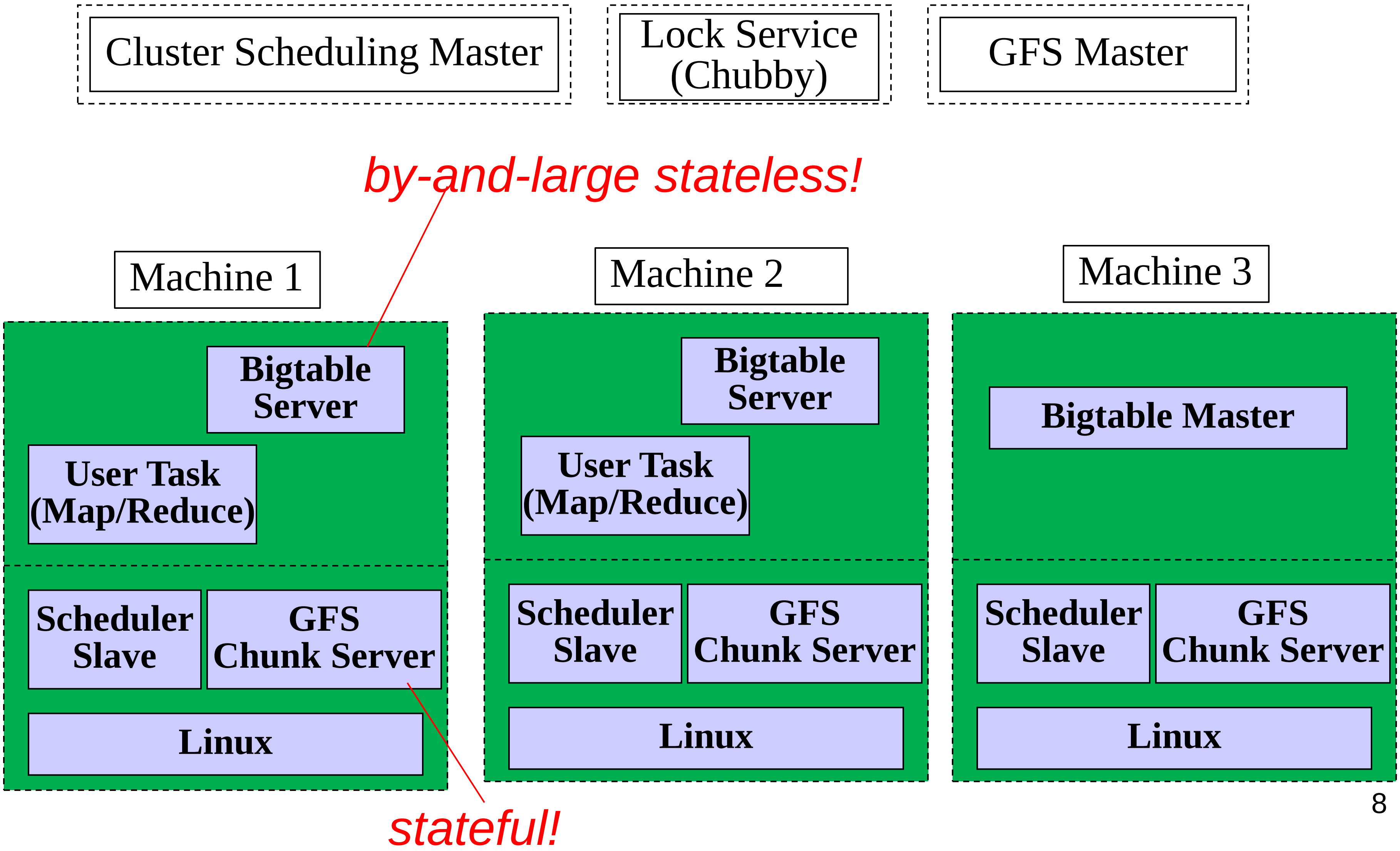
- BigTable uses of building blocks
  - GFS: stores all persistent state
  - Scheduler: schedules jobs involved in BigTable serving
  - Chubby: master election, location bootstrapping

GFS (Reminder)



- Master manages metadata
- Data transfers happen directly between clients/chunkservers
- Files broken into chunks (typically 64 MB)
- Chunks replicated across three machines for reliability

Typical Cluster



Cluster Scheduling Master

Lock Service (Chubby)

GFS Master

by-and-large stateless!

Machine 1

Bigtable Server

User Task (Map/Reduce)

Scheduler Slave

GFS Chunk Server

Linux

Machine 2

Bigtable Server

User Task (Map/Reduce)

Scheduler Slave

GFS Chunk Server

Linux

Machine 3

Bigtable Master

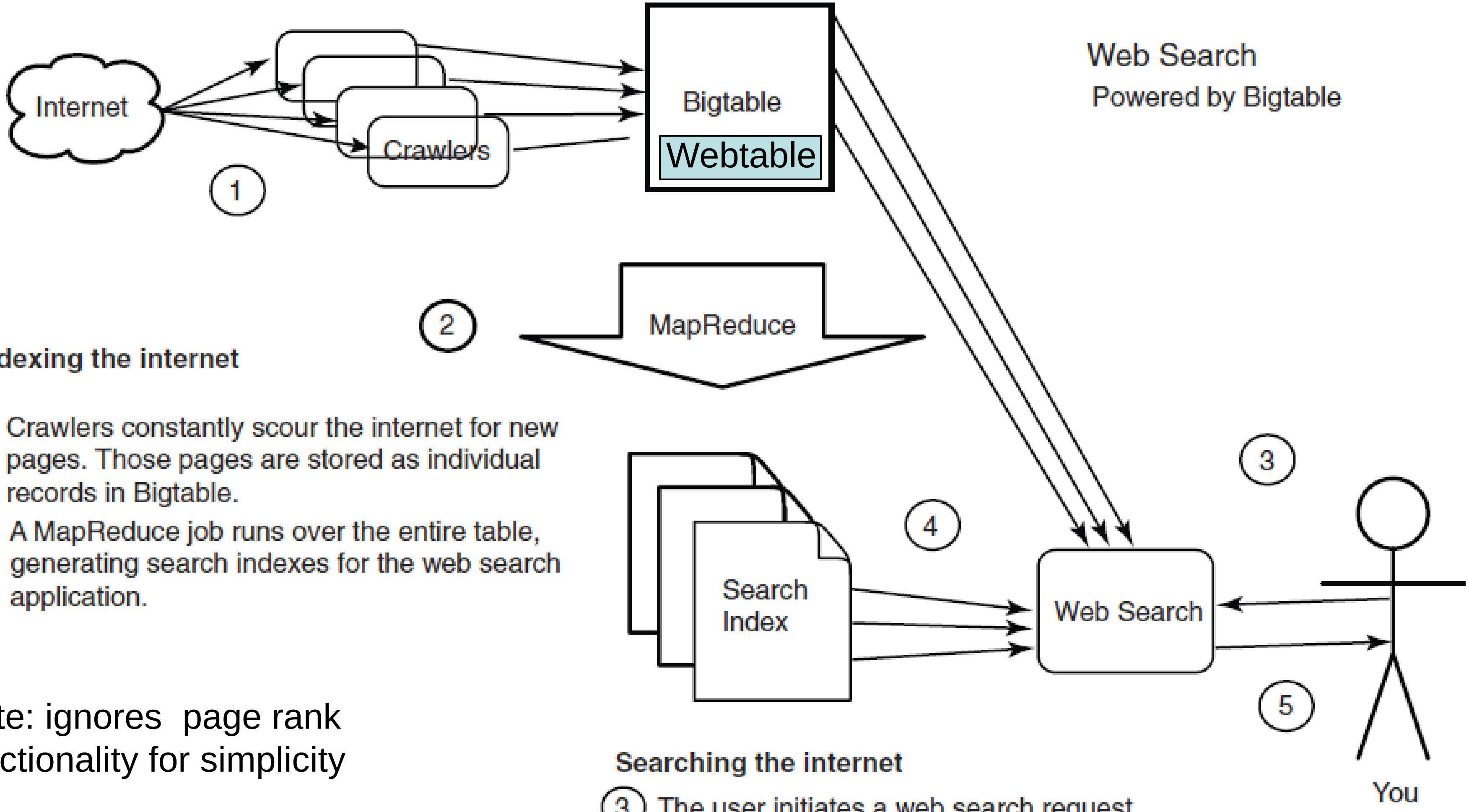
Scheduler Slave

GFS Chunk Server

Linux

stateful!

Specific Example: Web Search



Indexing the internet

1. Crawlers constantly scour the internet for new pages. Those pages are stored as individual records in Bigtable.
2. A MapReduce job runs over the entire table, generating search indexes for the web search application.

Searching the internet

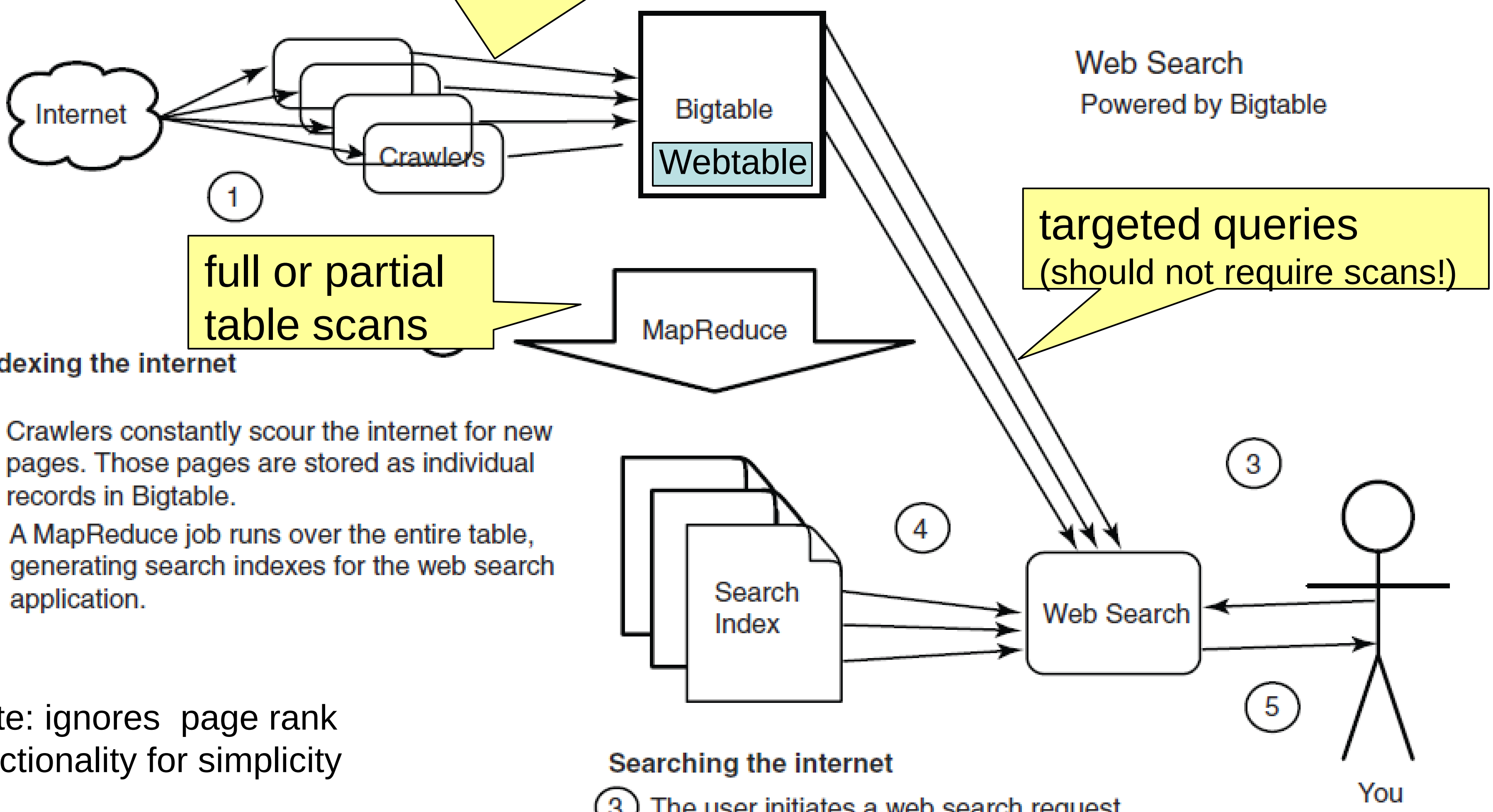
3. The user initiates a web search request.
4. The web search application queries the search indexes and retries matching documents directly from Bigtable.
5. Search results are presented to the user.

Note: ignores page rank functionality for simplicity

Source: "Hbase in Action", Dimiduk, et.al, http://www.manning.com/dimidukkhurana/HBiAsample_ch1.pdf

Specific Example: Web Search

inserts or updates of page contents and anchors to pages



targeted queries (should not require scans!)

full or partial table scans

Indexing the internet

1. Crawlers constantly scour the internet for new pages. Those pages are stored as individual records in Bigtable.
2. A MapReduce job runs over the entire table, generating search indexes for the web search application.

Searching the internet

3. The user initiates a web search request.
4. The web search application queries the search indexes and retries matching documents directly from Bigtable.
5. Search results are presented to the user.

Note: ignores page rank functionality for simplicity

Source: "Hbase in Action", Dimiduk, et.al, http://www.manning.com/dimidukkhurana/HBiAsample_ch1.pdf

Bigtable Description Outline

- Motivation and goals (last time)
- Schemas, interfaces, and semantics (with code) (today)
- Architecture (today)
- Implementation details (today, or you’ll read on your own)

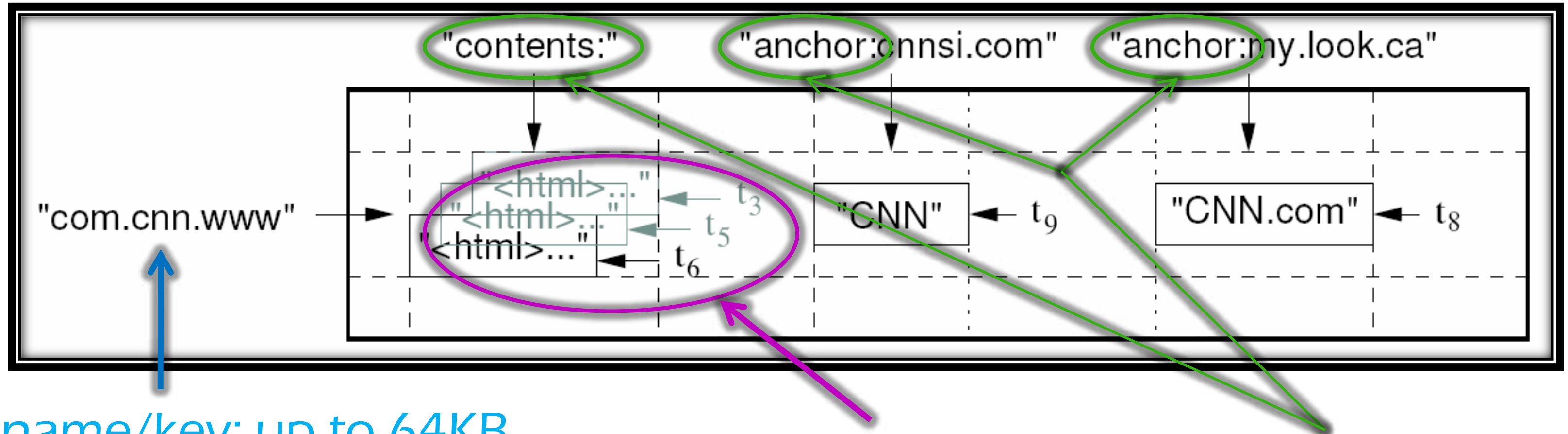
Basic Data Model

- "A Bigtable is a sparse, distributed, persistent multi-dimensional sorted map"

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

- Example: the (simplified) schema of the Webtable:

Webtable



Row name/key: up to 64KB, 10-100B typically, sorted. In this case, reverse URLs.

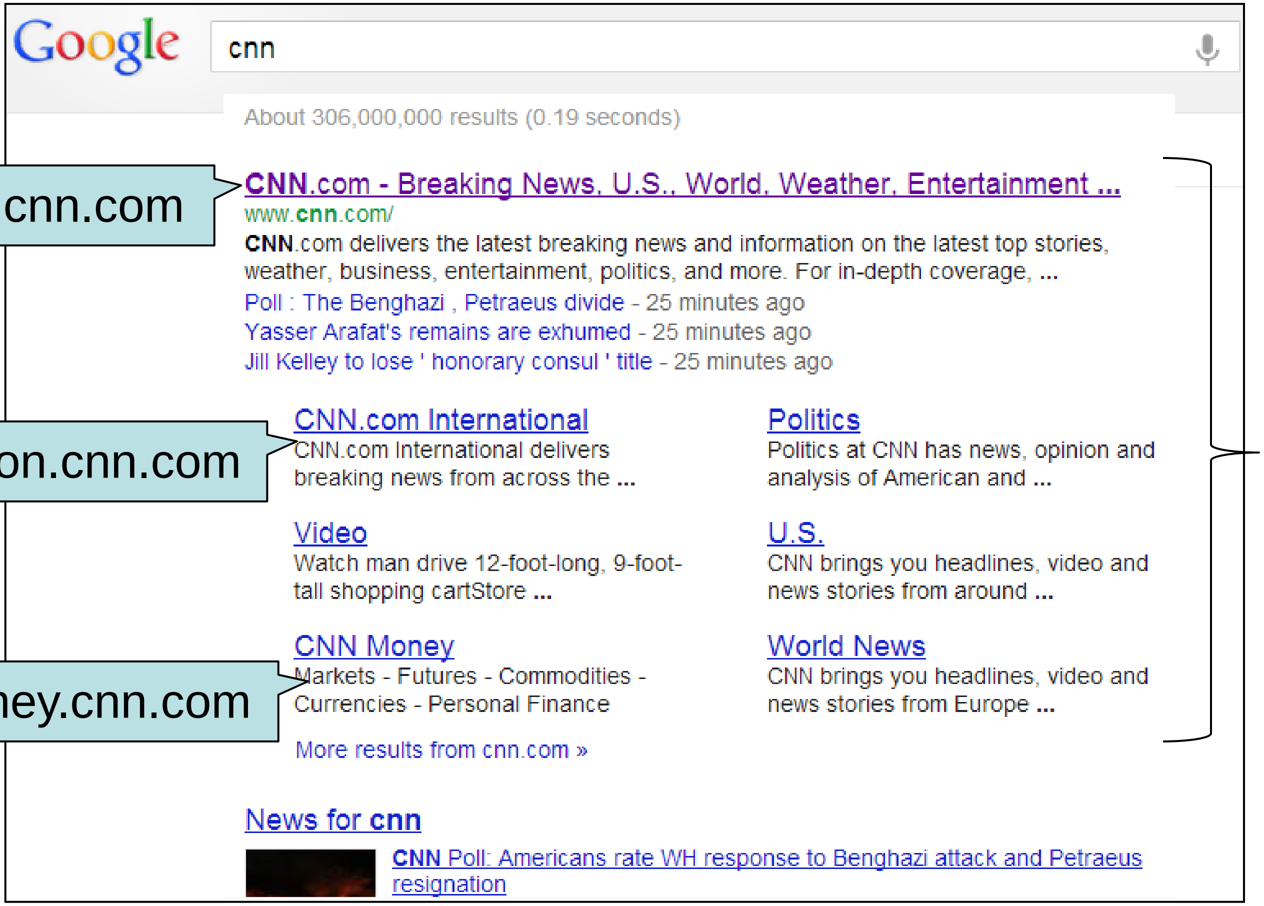
cell w/ timestamped versions + garbage collection

column families

Rows

- Row names/keys are arbitrary strings and are ordered lexicographically
  - Rows close together lexicographically are stored on one or a small number of machines
- Hence, programmers can manipulate row names to achieve good locality in their programs
  - Example: *com.cnn.www* vs. *www.cnn.com* – which row key provides more locality for site-local queries?
- Access to data in a row is atomic
  - Data row is the only unit of atomicity in Bigtable
- Does not support relational model
  - No table integrity constraints, no multi-row transactions

Row-based Locality



Google

cnn

About 306,000,000 results (0.19 seconds)

CNN.com - Breaking News, U.S., World, Weather, Entertainment ...
www.cnn.com/
CNN.com delivers the latest breaking news and information on the latest top stories, weather, business, entertainment, politics, and more. For in-depth coverage, ...
Poll : The Benghazi , Petraeus divide - 25 minutes ago
Yasser Arafat's remains are exhumed - 25 minutes ago
Jill Kelley to lose ' honorary consul ' title - 25 minutes ago

CNN.com International
CNN.com International delivers breaking news from across the ...

Politics
Politics at CNN has news, opinion and analysis of American and ...

Video
Watch man drive 12-foot-long, 9-foot-tall shopping cartStore ...

U.S.
CNN brings you headlines, video and news stories from around ...

CNN Money
Markets - Futures - Commodities - Currencies - Personal Finance

World News
CNN brings you headlines, video and news stories from Europe ...

More results from cnn.com »

News for cnn

CNN Poll: Americans rate WH response to Benghazi attack and Petraeus resignation

www.cnn.com

edition.cnn.com

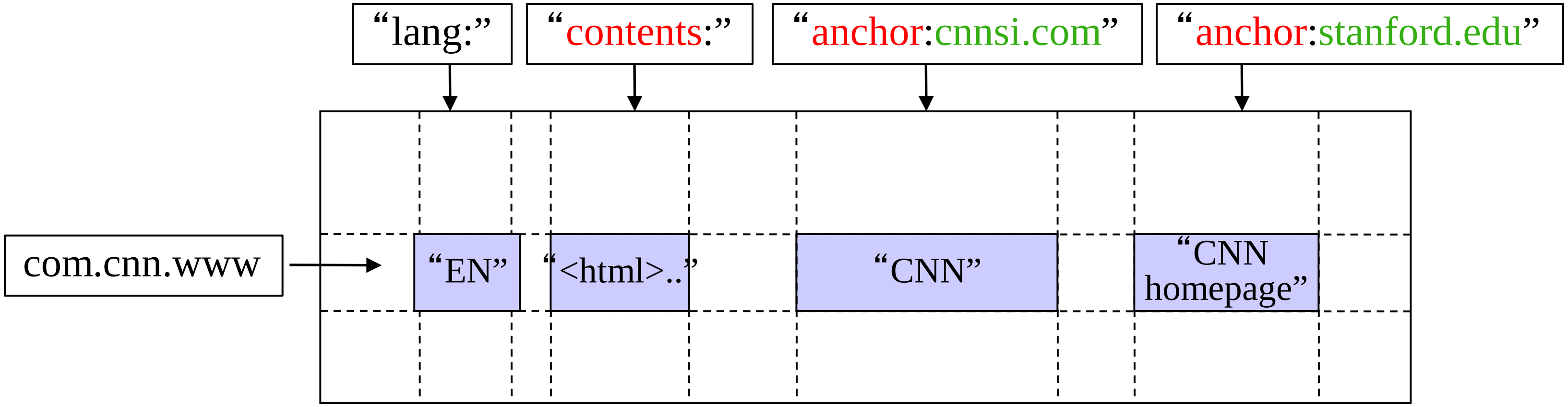
money.cnn.com

Matches from same site should be retrieved together by accessing the minimal number of machines

- Using reversed-DNS URLs clusters URLs from the same site together, to speed up site-local queries
  - com.cnn.edition, com.cnn.money, com.cnn.www

Columns

- Columns have a two-level name structure:
  family:optional_qualifier
- Column family
  - Unit of access control
  - Has associated type information
  - There are few column families
- Qualifier gives unbounded # of columns in each row
  - Provides additional levels of indexing, if desired
  - Extremely sparsely populated across rows



Timestamps

- Used to store different versions of data in a cell
  - New writes default to current time, but timestamps for writes can also be set explicitly by clients
- Lookup options:
  - *“Return most recent N values”*
  - *“Return all values in timestamp range (or all values)”*
- Column families can be marked w/ attributes:
  - *“Only retain most recent N versions in a cell”*
  - *“Keep values until they are older than T seconds”*
- Example uses:
  - Keep multiple versions of the data (e.g., Web pages)

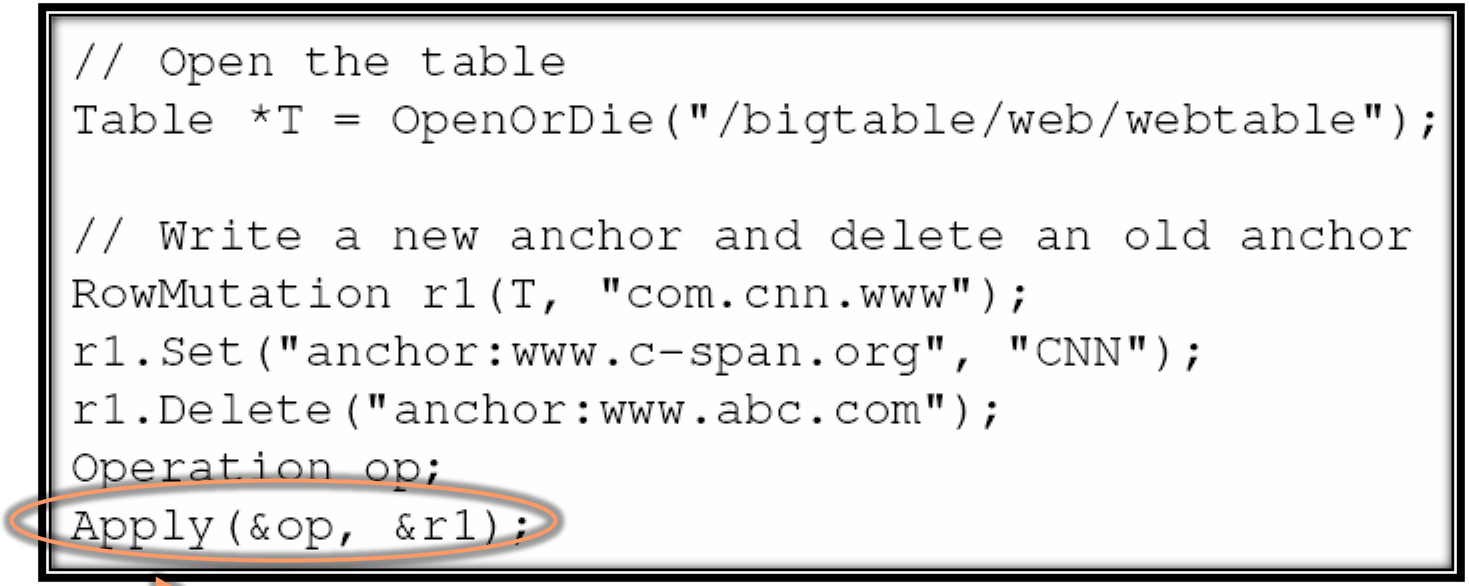
The Bigtable API

- Metadata operations
  - Create/delete tables, column families, change metadata

- Writes: Single-row, atomic
  - Set(): write cells in a row
  - DeleteCells(): delete cells in a row
  - DeleteRow(): delete all cells in a row

- Reads: Scanner abstraction
  - Allows to read arbitrary cells in a Bigtable table
    - Each row read is atomic
    - Can restrict returned rows to a particular range
    - Can ask for just data from 1 row (getter), all rows (scanner), etc.
    - Can ask for all columns, just certain column families, or specific columns
    - Can ask for certain timestamps only

API Examples: Write



```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");

// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

atomic row modification

No support for (RDBMS-style) multi-row transactions

Example Exercise:
Define Bigtable Schema for (Simplified) Twitter

- At the exam, you'll get a Bigtable schema design question
  - To prep, do this example at home, ask specific questions on Piazza

- Exercise: Based on Webtable's Bigtable schema, define a schema for an efficient, simplified version of Twitter

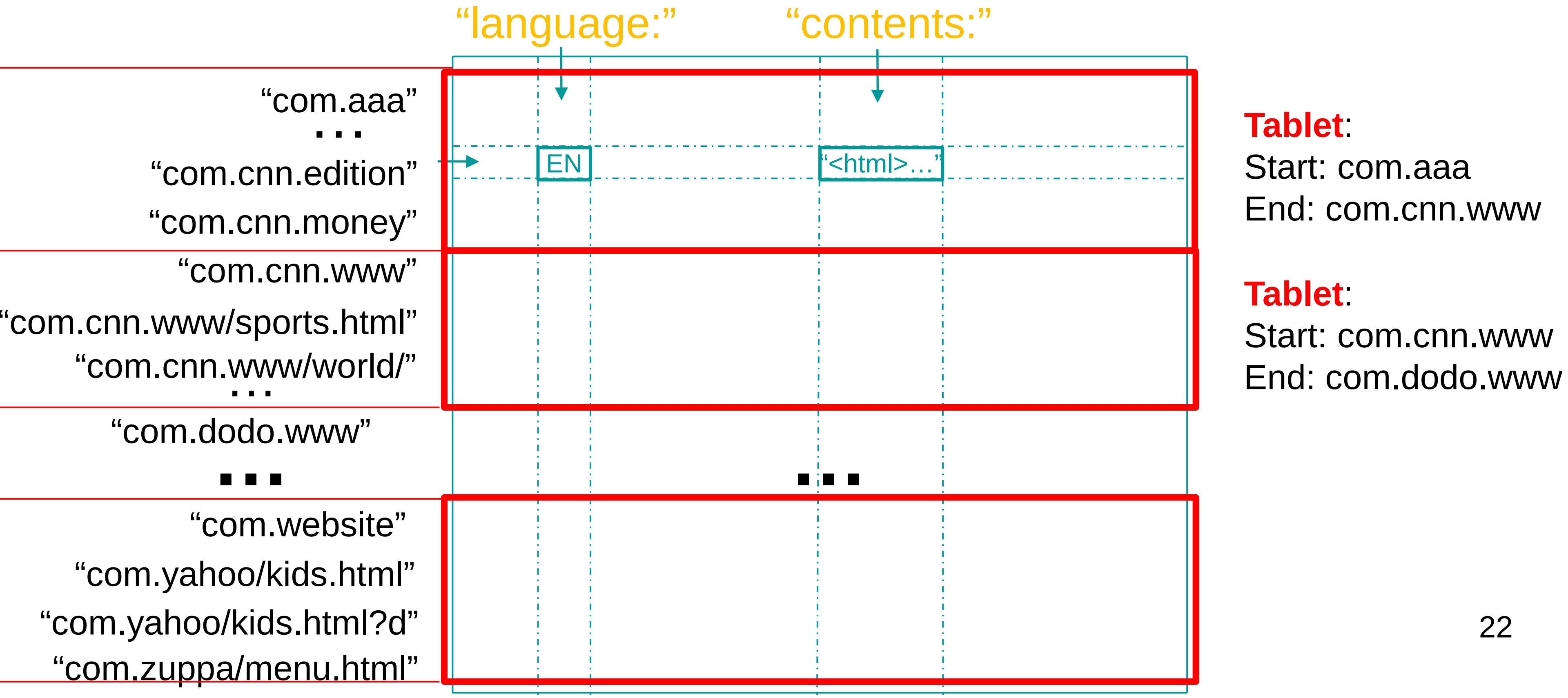
- Recommended design steps:
  - Restrict Twitter to some basic functionality and formulate the kinds of queries you might need to run to achieve that functionality
    - Example functionality: display tweets from the persons the user follows
  - Identify locality requirements for your queries to be efficient
  - Design your Bigtable schema (row names, column families, column names within each family, and cell contents) that would support the identified queries efficiently
  - Hint: Don't worry about replicating some data, such as tweet IDs, for fast access

Bigtable Description Outline

- Motivation and goals (last time)
- Schemas, interfaces, and semantics (with code) (today)
- Architecture (today)
- Implementation details (today, or you’ll read on your own)

Tablets

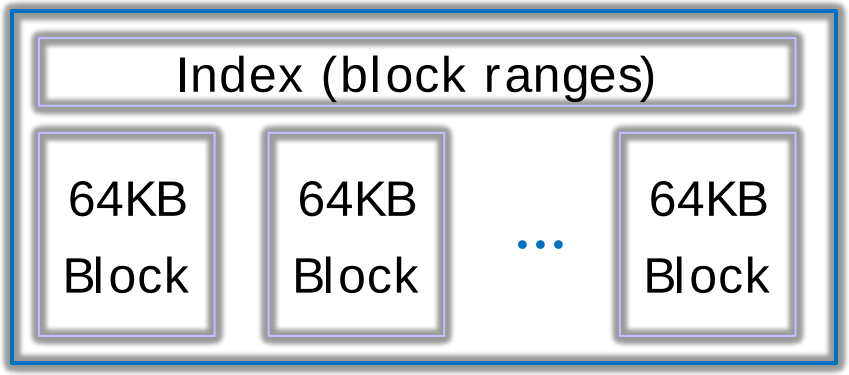
- A Bigtable table is partitioned into many tablets based on row keys
  - Tablets (100-200MB each) are stored in a particular structure in GFS
- Each tablet is served by one tablet server
  - Tablets are stateless (all state is in GFS), hence they can restart at any time



Tablet Structure

- Uses Google SSTables, a key building block
- Without going into much detail, an SSTable:
  - Is an immutable, sorted file of key-value pairs
  - SSTable files are stored in GFS
  - Keys are: <row, column, timestamp>
  - SSTables allow only appends, no updates (delete possible)
    - Why do you think they don’t use something that supports updates?

SSTable



Index (block ranges)

64KB Block

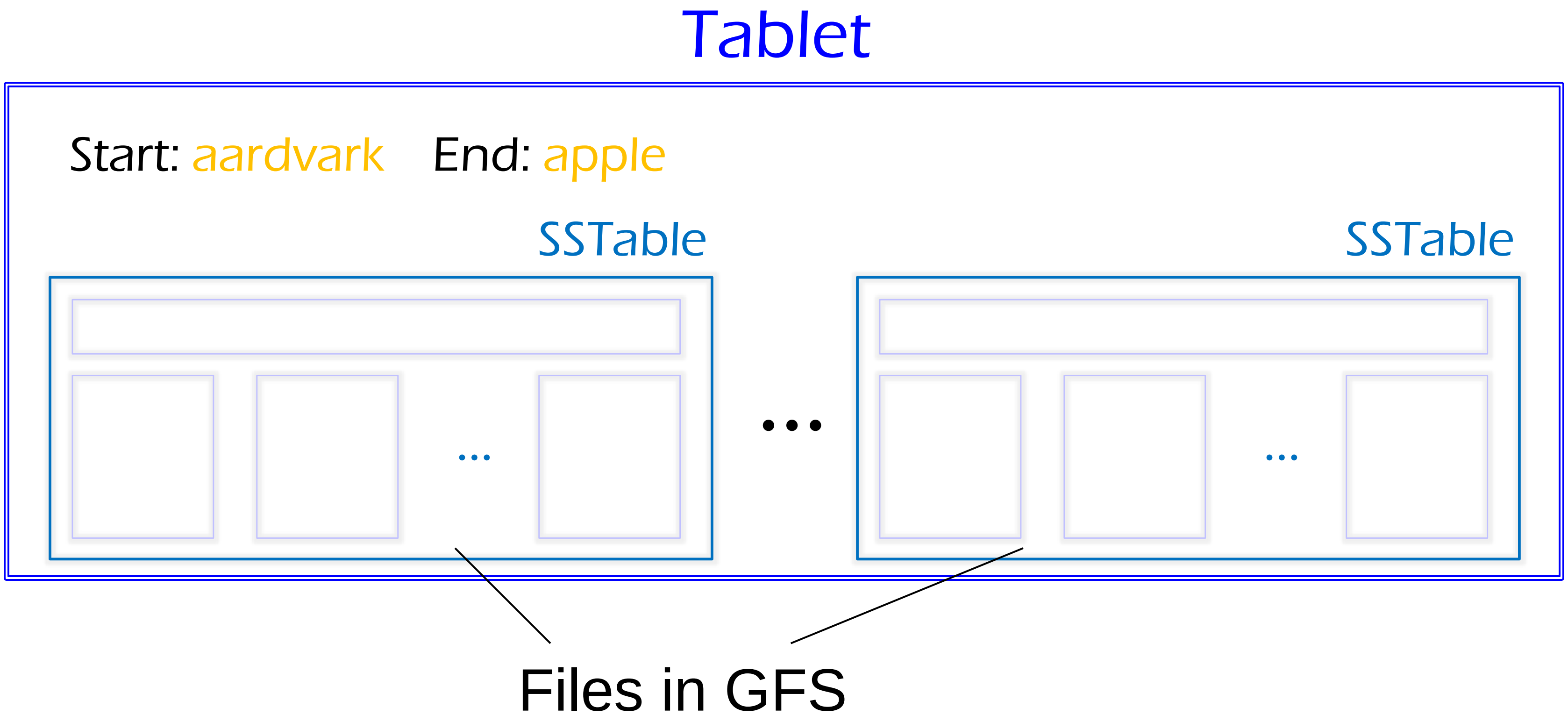
64KB Block

…

64KB Block

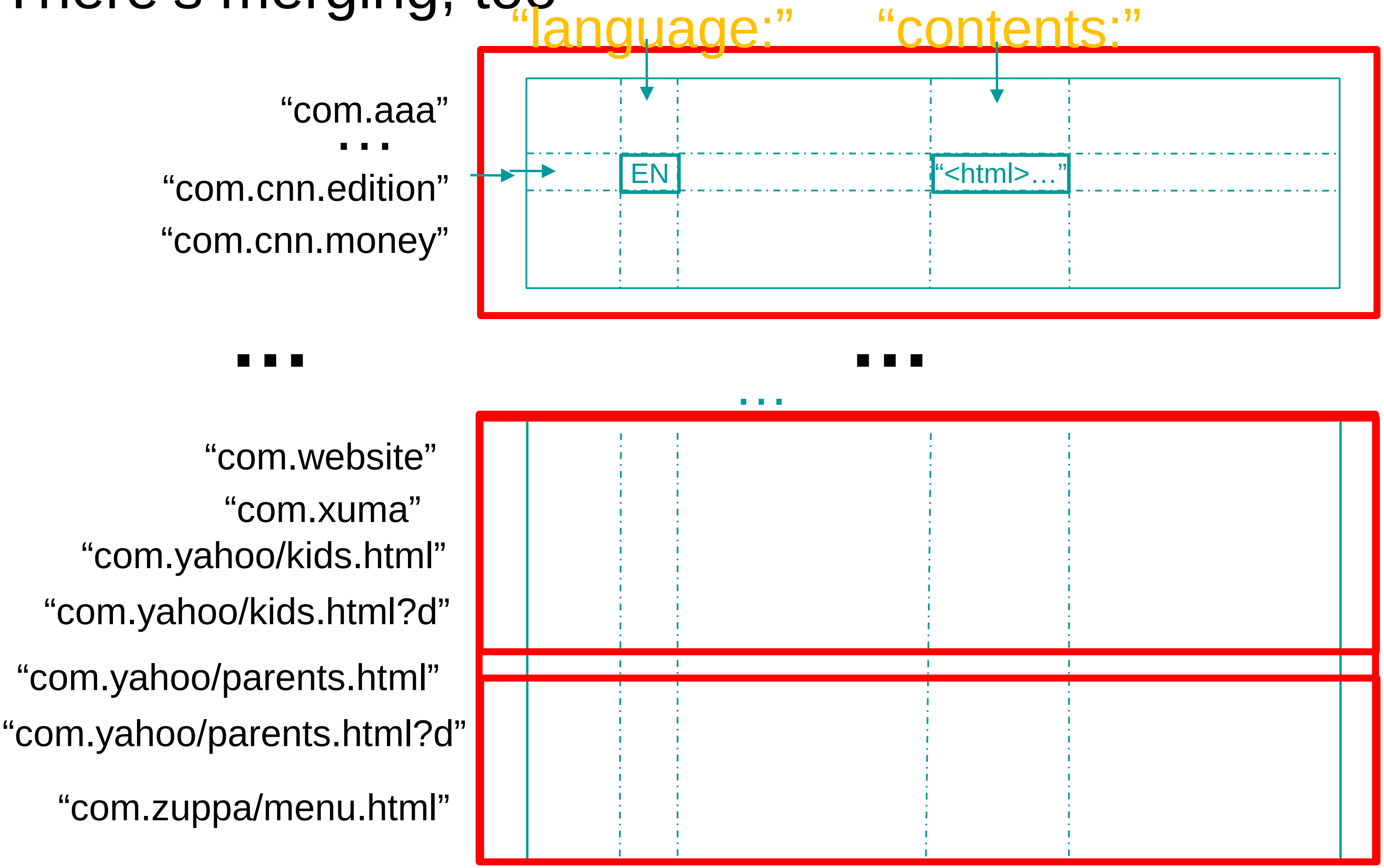
Tablet Structure

- A Tablet stores a range of rows from a table using SSTable files, which are stored in GFS



Tablet Splitting

- When tablets grow too big, they are split
- There's merging, too

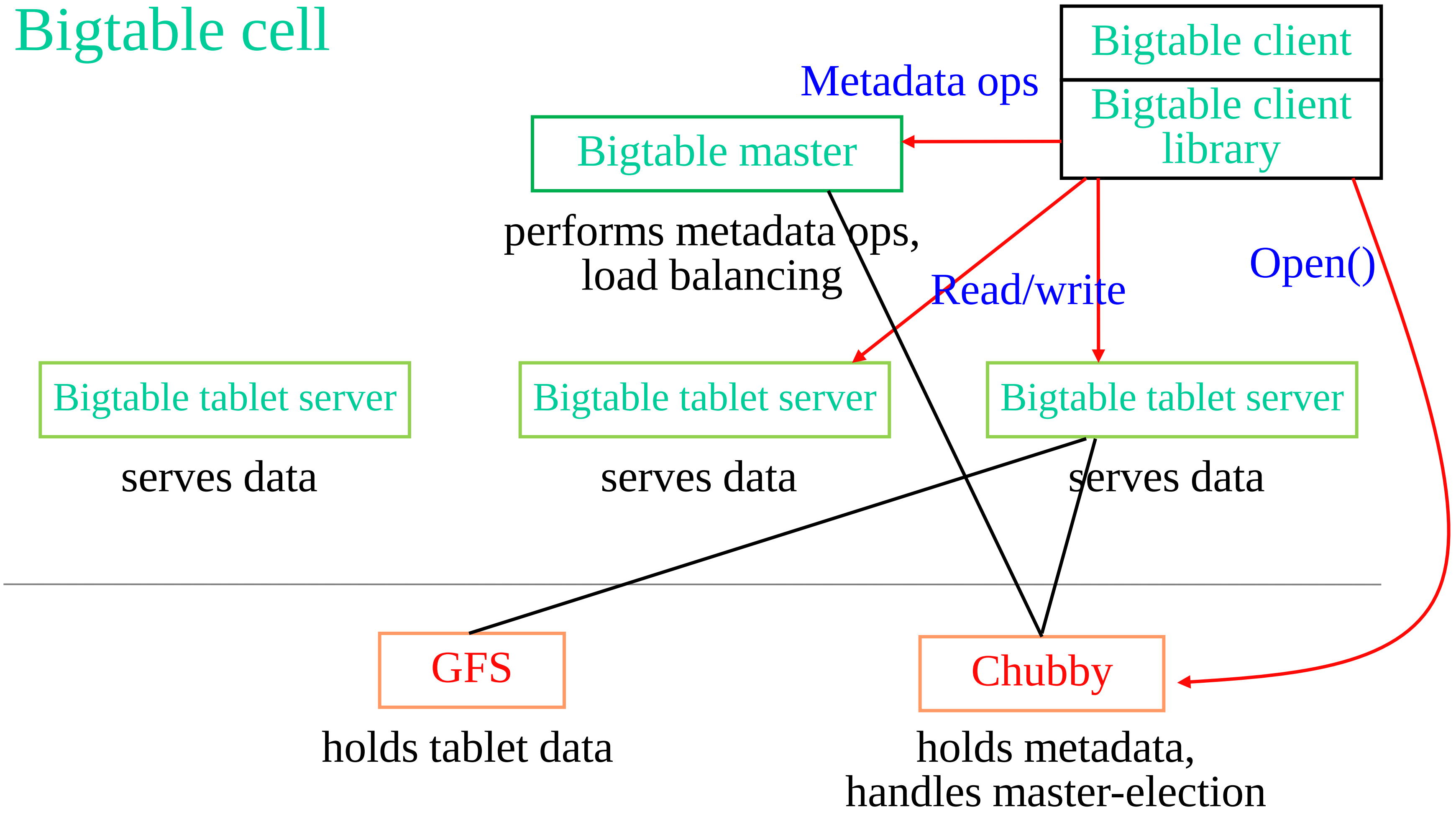


Servers

- Library linked into every client
- One master server
  - Assigns/load-balances tablets to tablet servers
  - Detects up/down tablet servers
  - Garbage collects deleted tablets
  - Coordinates metadata updates (e.g., create table, …)
  - Does **NOT** provide tablet location (we'll see how this is gotten)
  - Master is stateless – state is in Chubby and… Bigtable (recursively)!

- Many tablet servers
  - Tablet servers handle R/W requests to their tablets
  - Split tablets that have grown too large
  - Tablet servers are also stateless – their state is in GFS!

System Architecture



Bigtable cell

Bigtable client

Bigtable client library

Metadata ops

Bigtable master

performs metadata ops,
load balancing

Read/write

Open()

Bigtable tablet server

serves data

Bigtable tablet server

serves data

Bigtable tablet server

serves data

GFS

holds tablet data

Chubby

holds metadata,
handles master-election

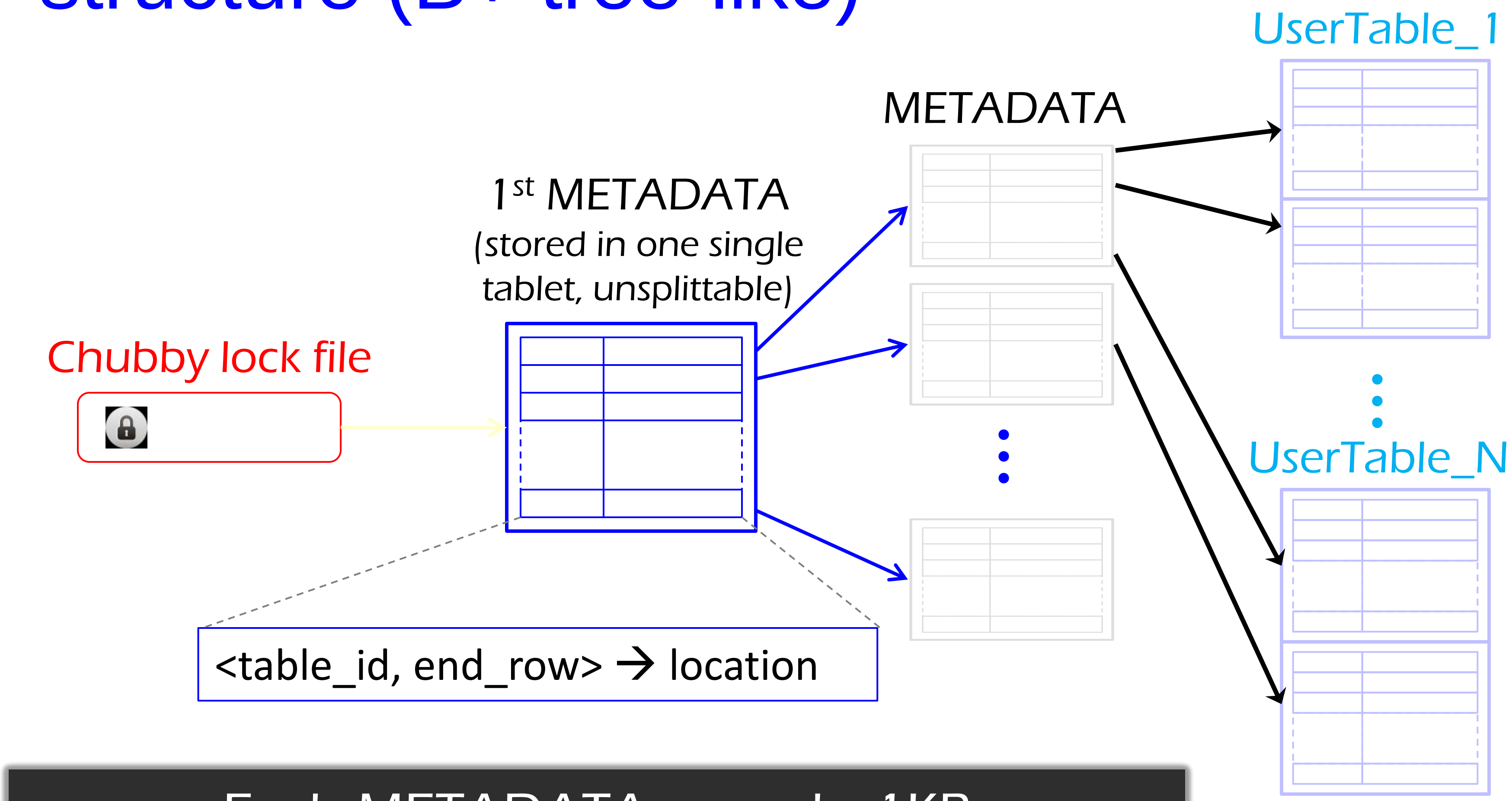
Locating Tablets

- Since tablets move around from server to server, given a row, how do clients find the right machine?
  - Tablet properties: startRowIndex and endRowIndex
  - Need to find tablet whose row range covers the target row

- One approach: could use the Bigtable master
  - Central server almost certainly would be bottleneck in large system

- Instead: store special tables containing tablet location info in the Bigtable cell itself (recursive design ☺)

Tablets are located using a hierarchical structure (B+ tree-like)


UserTable_1
METADATA
1st METADATA
(stored in one single tablet, unsplittable)
Chubby lock file
UserTable_N
<table_id, end_row> → location

Each METADATA record ~1KB

Max METADATA table = 128MB

Addressable table values in Bigtable = 2^{21} TB

Bigtable Description Outline

- Motivation and goals (last time)
- Schemas, interfaces, and semantics (with code) (today)
- Architecture (today)
- Implementation details (today, or you’ll read on your own)

Tablet Assignment

- 1 Tablet => 1 Tablet server
- Master
  - keeps tracks of set of live tablet serves and unassigned tablets
  - Master sends a tablet load request for unassigned tablet to the tablet server

- Bigtable uses Chubby to keep track of tablet servers

- On startup a tablet server:
  - Tablet server creates and acquires an exclusive lock on uniquely named file in Chubby directory
  - Master monitors the above directory to discover tablet servers

- Tablet server stops serving tablets if it loses its exclusive lock
  - Tries to reacquire the lock on its file as long as the file still exists

Tablet Assignment

- If the file no longer exists, tablet server not able to serve again and kills itself

- Master is responsible for finding when tablet server is no longer serving its tablets and reassigning those tablets as soon as possible.

- Master detects by checking periodically the status of the lock of each tablet server.
  - If tablet server reports the loss of lock
  - Or if master could not reach tablet server after several attempts.

Tablet Assignment

- Master tries to acquire an exclusive lock on server's file.
  - If master is able to acquire lock, then chubby is alive and tablet server is either dead or having trouble reaching chubby.
  - If so master makes sure that tablet server never can server again by deleting its server file.
  - Master moves all tablets assigned to that server into set of unassigned tablets.
- If Chubby session expires, master kills itself.
- When master is started, it needs to discover the current tablet assignment.

Master Startup Operation

- Grabs unique master lock in Chubby
  - Prevents server instantiations
- Scans directory in Chubby for live servers
- Communicates with every live tablet server
  - Discover all tablets
- Scans METADATA table to learn the set of tablets
  - Unassigned tables are marked for assignment

Bigtable Summary

- Scalable distributed storage system for semi-structured data
- Offers a multi-dimensional-map interface
  - <row, column, timestamp> → value
- Offers atomic reads/writes within a row
- Key design philosophy: statelessness, which is key for scalability
  - All Bigtable servers (including master) are stateless
  - All state is stored in reliable GFS and Chubby systems
  - Bigtable leverages strong-semantic operations in these systems (appends in GFS, file locks in Chubby)