Distributed Systems
[Fall 2012]

Lec 21: Bigtable: architecture, implementation, and schema design

Slide acks: Mohsen Taheriyan

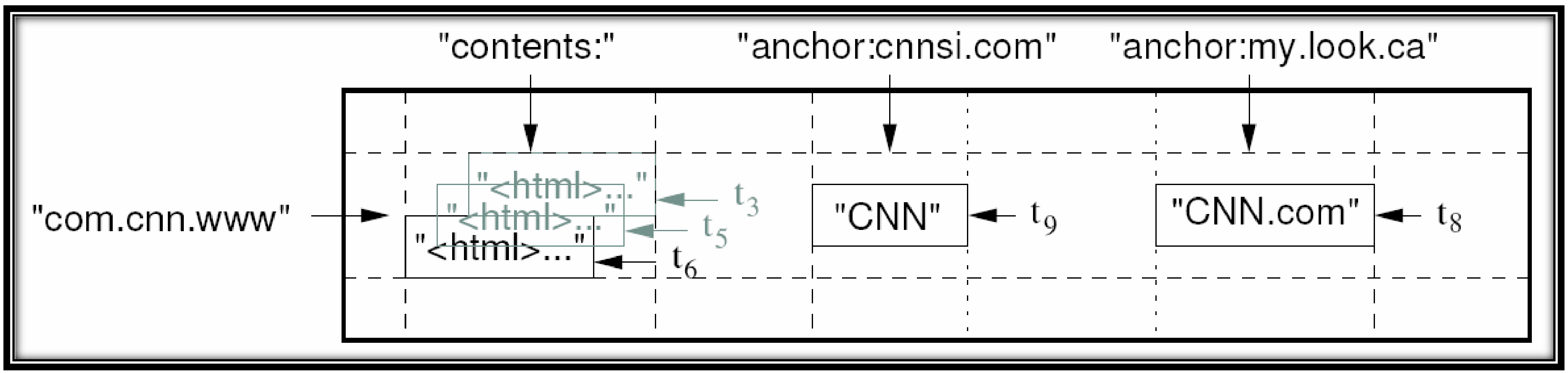
(http://www-scf.usc.edu/~csci572/2011Spring/presentations/Taheriyan.pptx)

Data Model (Reminder)

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

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

Webtable



API (Reminder)

```
// 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);
```

```
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {
  printf("%s %s %lld %s\n",
         scanner.RowName(),
         stream->ColumnName(),
         stream->MicroTimestamp(),
         stream->Value());
}
```

Bigtable Description Outline

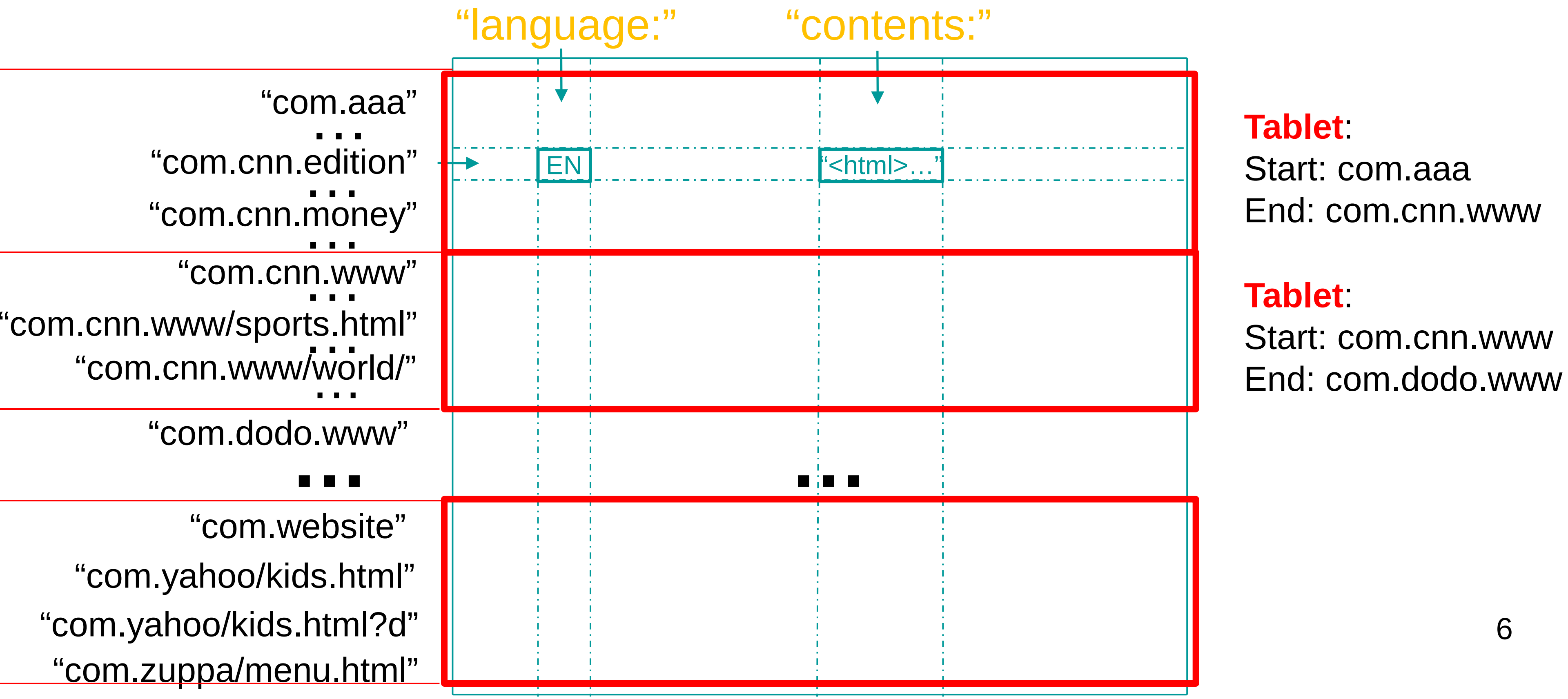
- Motivation and goals (last time)
- Schemas, interfaces, and semantics (last time)
- Architecture and implementation (today)
- Key topic: schema design in Bigtable (today)
  - There will be one schema-design question at the exam

Bigtable Description Outline

- Motivation and goals (last time)
- Schemas, interfaces, and semantics (last time)
- Architecture and implementation (today)
- Key topic: schema design in Bigtable (today)
  - There will be one schema-design question at the exam

Tablets

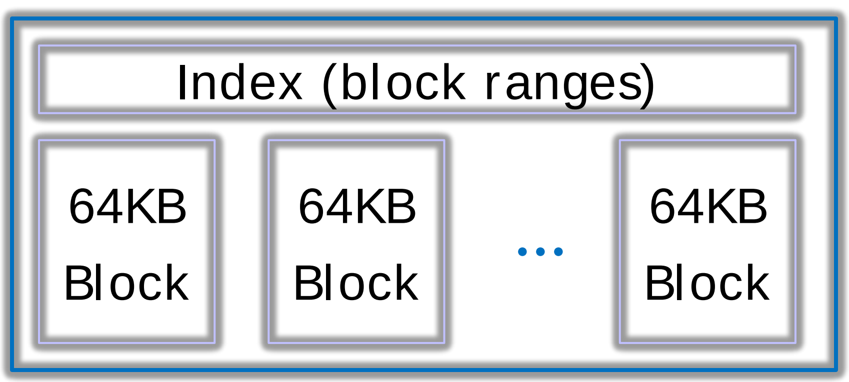
- A Bigtable table is partitioned into many tablets based on row keys
  - Tablets (100-200MB each) are stored in a particular structure in GFS
- Each tablet is served by one tablet server
  - Tablet servers are stateless (all state is in GFS), so they can restart any time



Tablet Structure

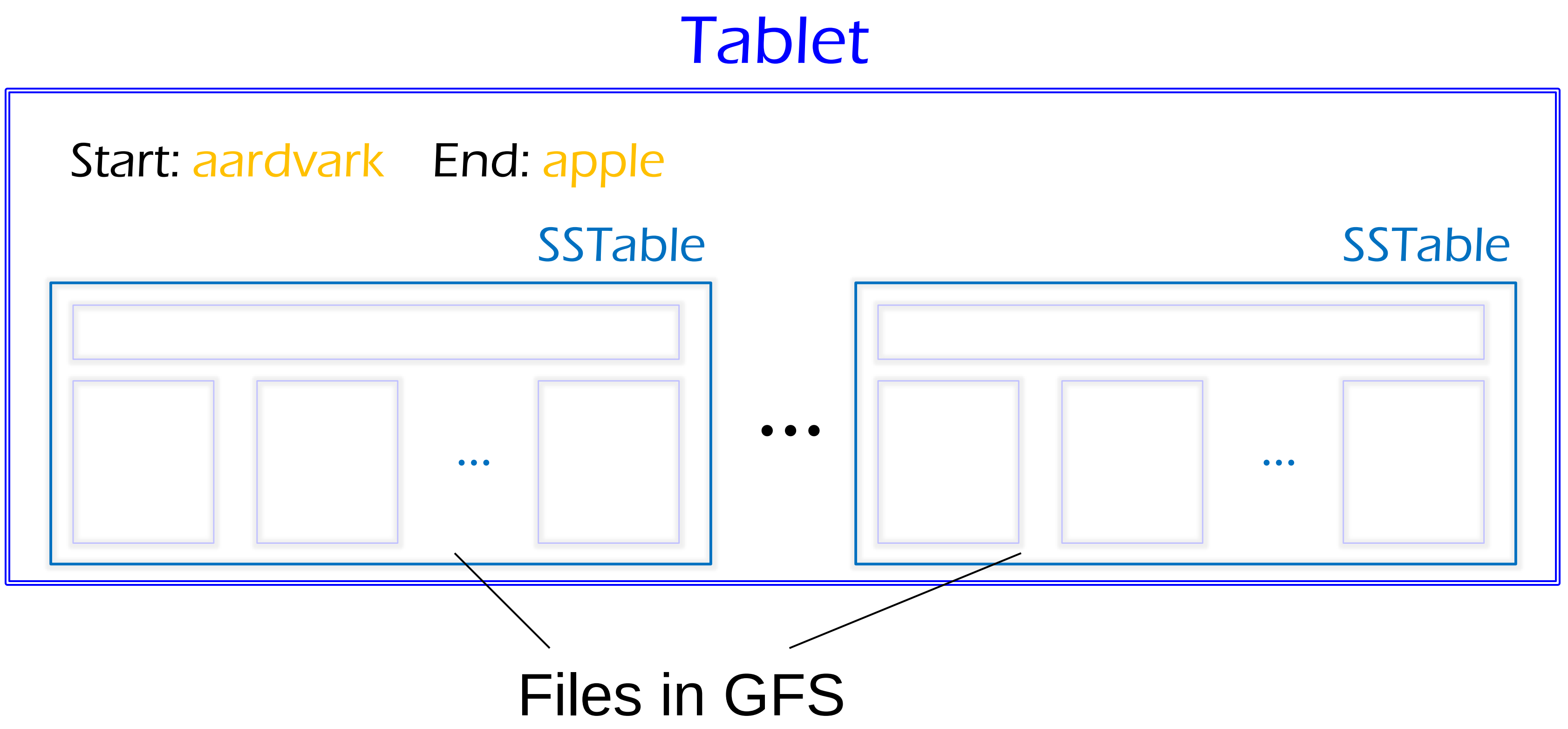
- Uses Google SSTables, a key building block
- Without going into much detail, an SSTable:
  - Is a file storing immutable key-value pairs
  - Its keys are: <row, column, timestamp>
  - It is stored in GFS
  - It allows only appends, no updates (deletes are possible)
    - Why do you think they don’t use something that supports updates?

SSTable



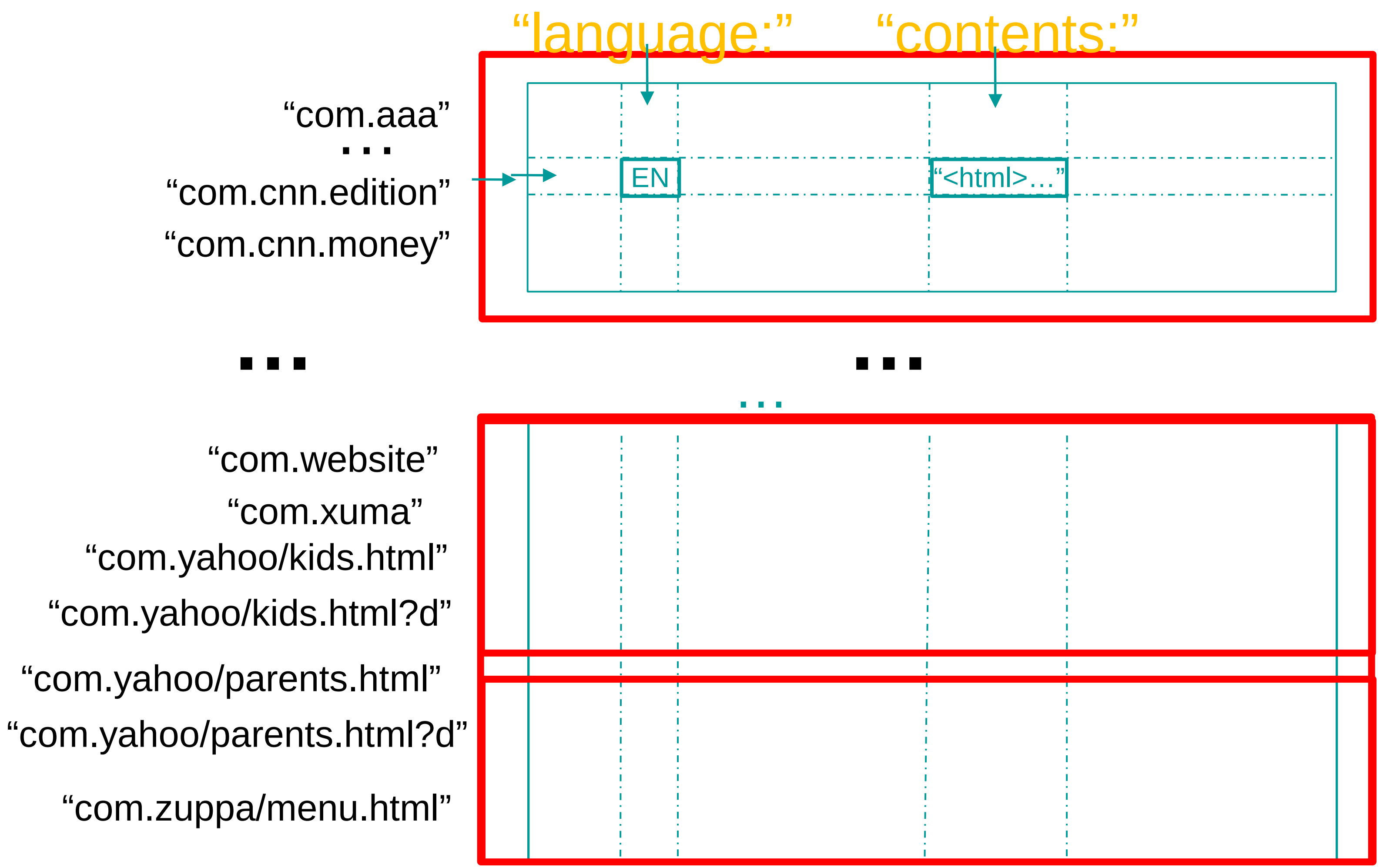
Tablet Structure

- 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, the tablet server splits them
- There’s merging, too

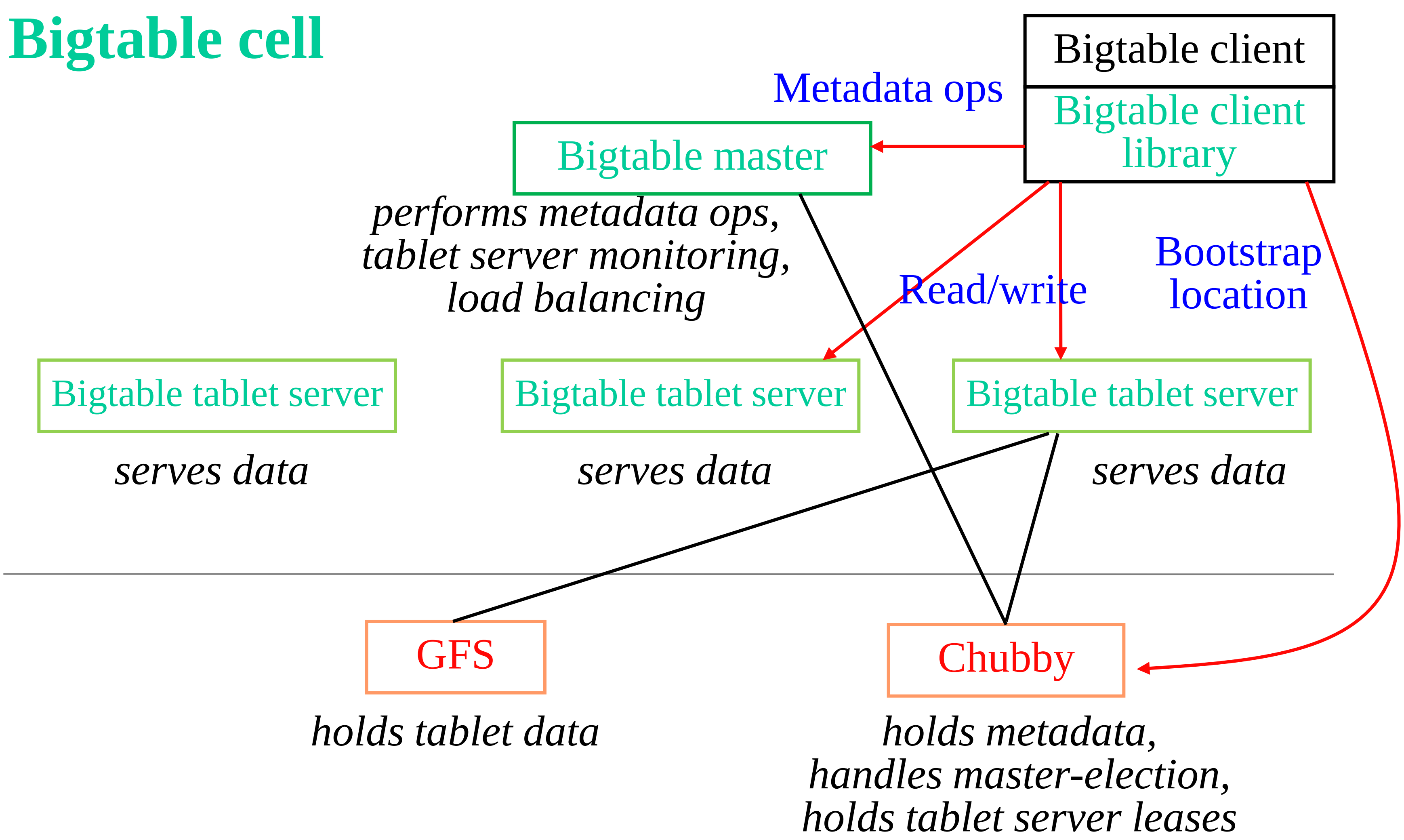


Architecture

- 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 – its state (e.g., tablet locations, table schemas, etc.) is in Chubby and Bigtable (recursively)!

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

Architecture



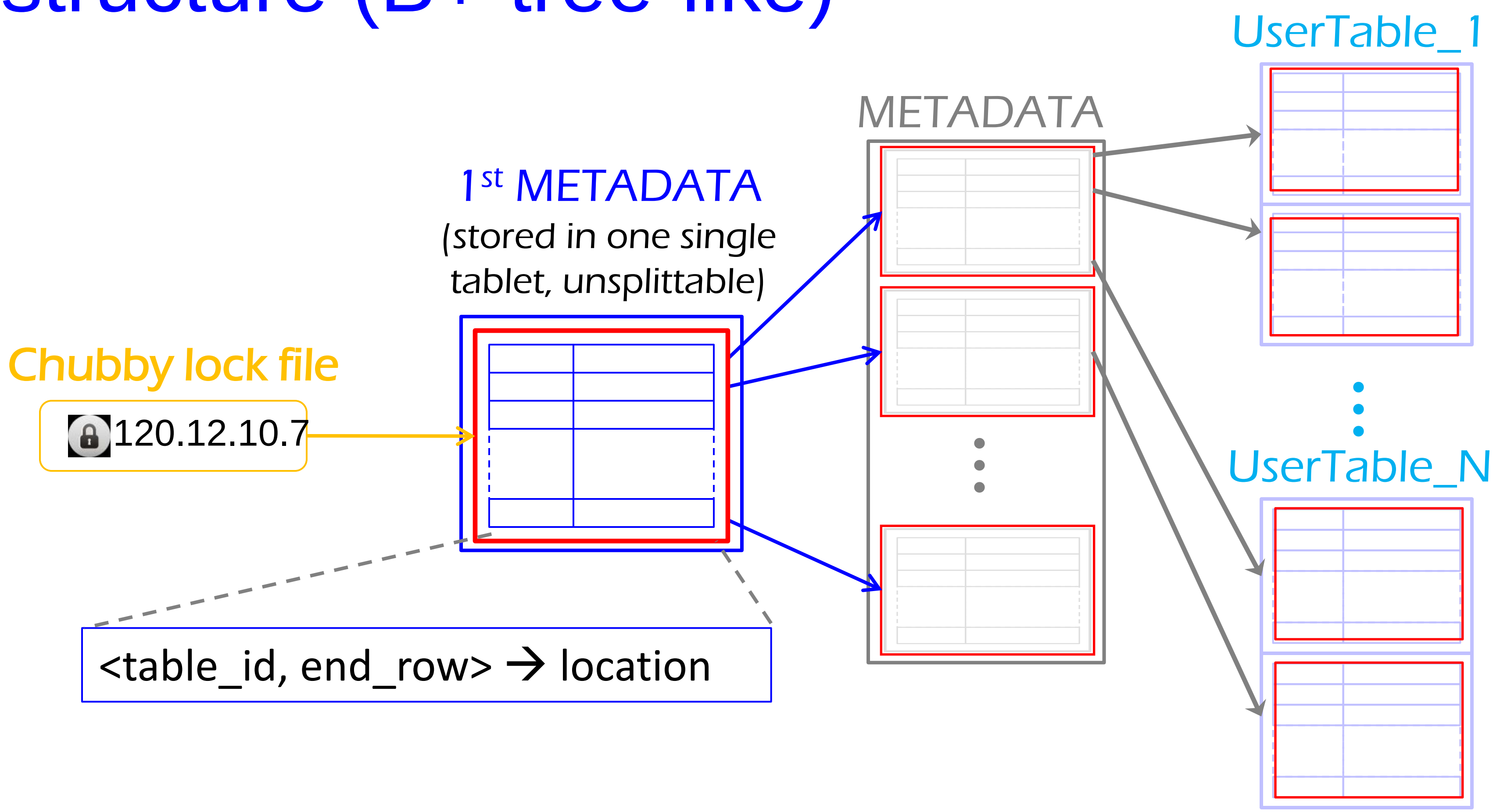
Locating Tablets

- Since tablets move around from server to server, given a row, how do clients find the right tablet server?
  - 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
  - Plus would need to make it reliable – that's hard

- 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)



Each METADATA record ~1KB
Max METADATA table = 128MB
Addressable table values in Bigtable = 2^{21} TB

Tablet Assignment (1/3)

- 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 (2/3)

- 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 (3/3)

- 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 can never serve 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.
  - Where does it go for that?

Master Startup Operation

- Grabs unique master lock in Chubby
  - Prevents others from becoming master
- 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-related Implementations

- Hbase is the open-source version of Bigtable
  - Design is very similar, though not identical
  - Terminology is different:
    - Tablet -> Region
    - Tablet server -> Region server
  - API is slightly different, but we'll ignore that here
- Hbase is part of the Apache Hadoop framework
  - Can be used with Hadoop MapReduce
  - Can be integrated with Facebook Thrift (high-performance RPC/marshalling – we talked about it briefly in the RPC lectures)
- Hbase is heavily used by a lot of people
  - Facebook, StumbleUpon, Twitter, ...

Bigtable Description Outline

- Motivation and goals (last time)
- Schemas, interfaces, and semantics (last time)
- Architecture and implementation (today)
- Key topic: schema design in Bigtable (today)

Key Topic: Schema Design

- Designing a schema for Bigtable is very different from designing a schema for an RDBMS
- The key idea in Hbase is de-normalization, a concept largely frowned upon in RDBMS
- RDBMS mantra: Normalize your database!
  - I.e., remove all redundant data from your DB
  - Positives:
  - Negatives:
- Bigtable mantra: De-normalize your database!
  - Replicate, cluster data if you can!
  - Positives:
  - Negatives:

Key Topic: Schema Design

- Designing a schema for Bigtable is very different from designing a schema for an RDBMS
- The key idea in Hbase is de-normalization, a concept largely frowned upon in RDBMS
- RDBMS mantra: Normalize your database!
  - I.e., remove all redundant data from your DB
  - Positives: saves space, great for updates
  - Negatives: many reads from DB will involve joining a lot of data that's stored in different tables, hence no locality
- Bigtable mantra: De-normalize your database!
  - Replicate, cluster data for best read performance!
  - Positives: efficient reads
  - Negatives: bad for writes, redundancy

Example: Webtable in RDBMS

- If we were to design a Webtable database in an RDBMS, how would we have done it?

Database: “Webtable”

Table 1: WebPageInfo

ID	URL	Lang	…
1234	www.cnn.com	EN	

Table 2: WebPageContents

ID	Timestamp	Contents
1234	1234566000	“<html>..</html>”

Table 3: WebPageAnchors

ID	Anchor ID	Anchor text ID
1234	5678	6543

Table 4: WebPageAnchorText

Anchor text ID	Timestamp	Anchor text value
6543	1234566000	“CNN home”

Example: Webtable in RDBMS

Database: “Webtable”

Table 1: WebPageInfo

ID	URL	Lang	…
1234	www.cnn.com	EN	

Table 2: WebPageContents

ID	Timestamp	Contents
1234	1234566000	“<html>..</html>”

Table 3: WebPageAnchors

ID	Anchor ID	Anchor text ID
1234	5678	6543

Table 4: WebPageAnchorText

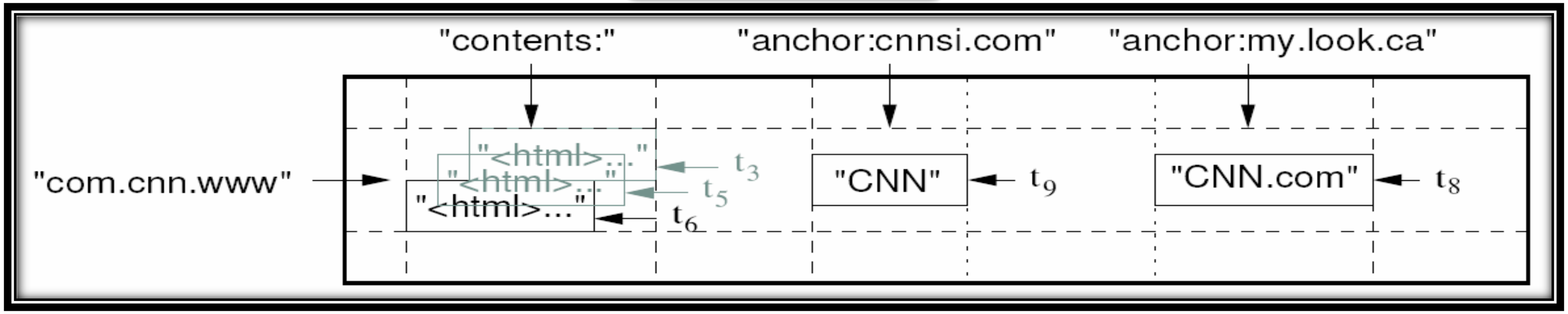
Anchor text ID	Timestamp	Anchor text value
6543	1234566000	“CNN home”

- What does this mean for queries?
  - How do you select the latest anchors to www.cnn.com? (whiteboard)
  - Complex joins of many tables
    - That probably means different machines, hence poor locality, scale, and performance!

Webtable in Bigtable

- Everything's stored in one single table, with locality considerations, hence queries like that are very fast
- But there's can be a lot of redundancy:
  - Example 1: to efficiently retrieve every link to which cnn.com points to, you'd need to add a column family, e.g., "link:", into Webtable, which will replicate the "anchor" data in every row!
  - Example 2 (more subtle): each <row name, column name, and timestamp> is replicated for each and every *item* down in the SSTables!
- Consistent updates are hard when you have redundancy in DB

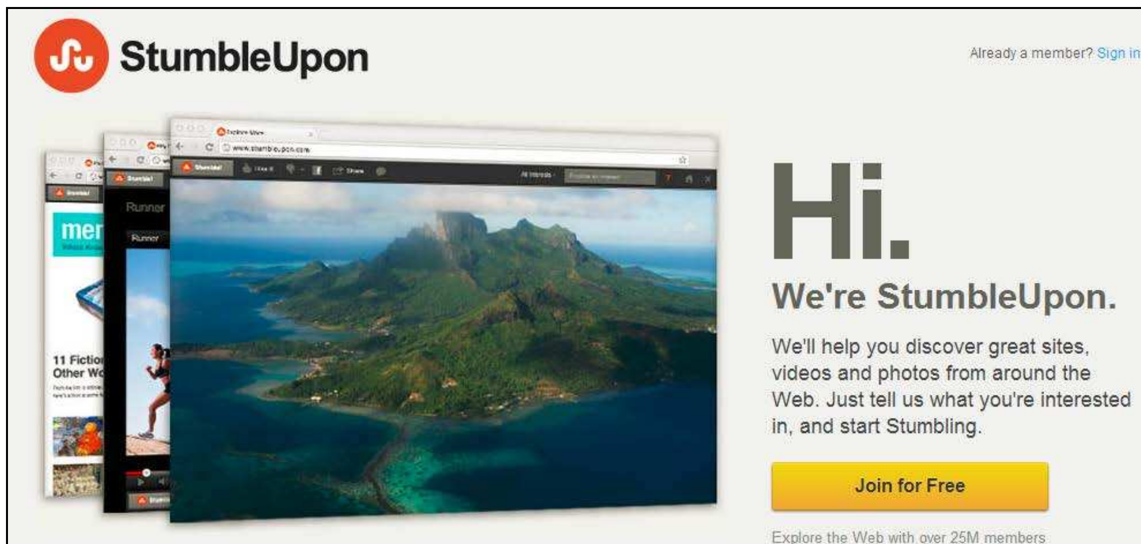
Webtable



Another Example: StumbleUpon's Time Series DB

- StumbleUpon.com is a site/content recommendation service
- As all big sites do, they have big scaling issues, too
- They needed a database that could store enormous amounts of time series data at high rates (<series, time> → value)
- They created and open-sourced OpenTSDB, a time-series database based on Hbase

- Let's look at their recommendations for how to define Hbase schemas



Movie Time: OpenTSDB Schemas

- Nice presentation from StumbleUpon on the choice and evolution of their Hbase schemas for OpenTSDB

http://www.cloudera.com/content/cloudera/en/resources/library/hbasecon/video-hbasecon-2012-lessons-learned-from-opentsdb.html

An Exercise for You: Define Bigtable Schema for (Simplified) Twitter

- At the exam, you'll get a Bigtable schema design question
  - To prep, do this example alone or in teams, ask specific questions on Piazza
- Exercise: Define a schema for an efficient, simplified version of Twitter
  - Use Webtable schema as reference
  - At the end of the class, we'll have a few more examples on schema design
- 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: list 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 queries efficiently
  - Hint:
    - De-normalize (replicate tweets across followers for fast listing of tweets)
    - Reflect on why it's OK to replicate (e.g., storage is cheap, tweets are not editable!)

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 philosophies: statelessness and layered design, which are 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, atomic row-updates of Bigtable itself)