Handling Large Datasets at Google: Current Systems and Future Directions

Jeff Dean
Google Fellow

http://labs.google.com/people/jeff

Google

Outline

- Hardware infrastructure
- Distributed systems infrastructure:
  - Scheduling system
  - GFS
  - BigTable
  - MapReduce
- Challenges and Future Directions
  - Infrastructure that spans all datacenters
  - More automation

Google™

Sample Problem Domains

- Offline batch jobs
  - Large datasets (PBs), bulk reads/writes (MB chunks)
  - Short outages acceptable
  - Web indexing, log processing, satellite imagery, etc.
- Online applications
  - Smaller datasets (TBs), small reads/writes small (KBs)
  - Outages immediately visible to users, low latency vital
  - Web search, Orkut, GMail, Google Docs, etc.
- Many areas: IR, machine learning, image/video processing, NLP, machine translation, ...

Google™

Typical New Engineer

- Never seen a petabyte of data
- Never used a thousand machines
- Never **really** experienced machine failure

Our software has to make them successful.

Google

Google's Hardware Philosophy

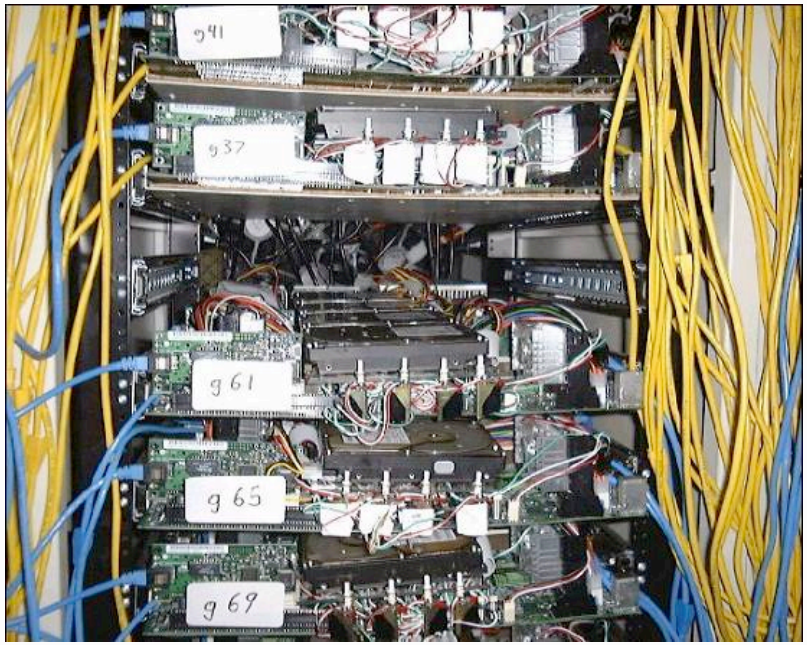
Truckloads of low-cost machines

- Workloads are large and easily parallelized
- Care about perf/$, not absolute machine perf
- Even reliable hardware fails at our scale

- Many datacenters, all around the world
  - Intra-DC bandwidth >> Inter-DC bandwidth
  - Speed of light has remained fixed in last 10 yrs :)

Google™

Effects of Hardware Philosophy



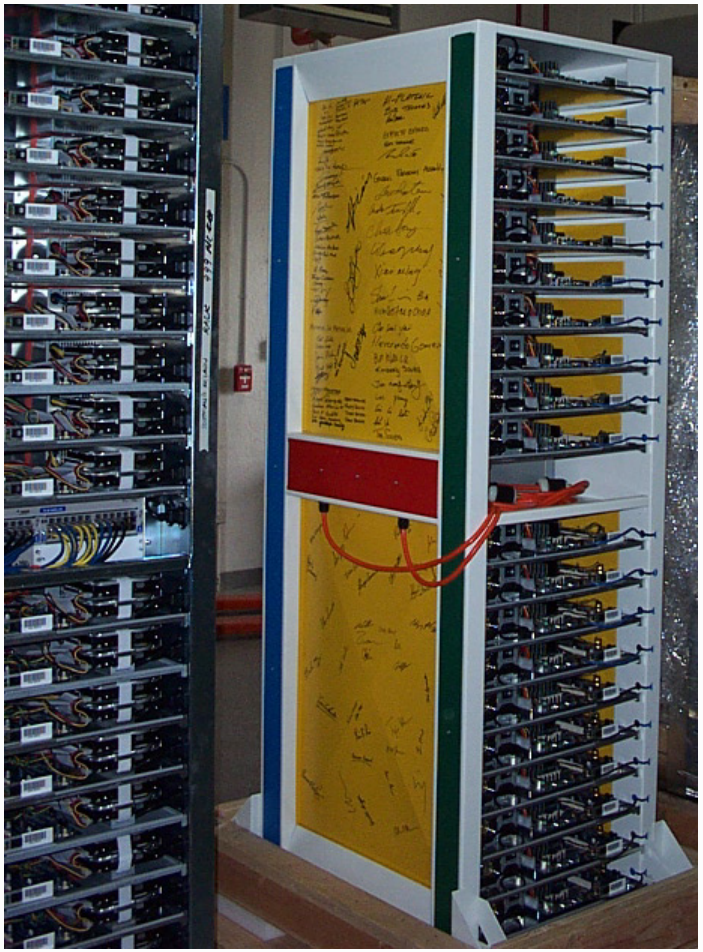
Google - 1999

- Software must tolerate failure
- Application's particular machine should not matter
- No special machines - just 2 or 3 flavors

Google

Current Design

- In-house rack design
- PC-class motherboards
- Low-end storage and networking hardware
- Linux
- + in-house software



Google™

The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packet loss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

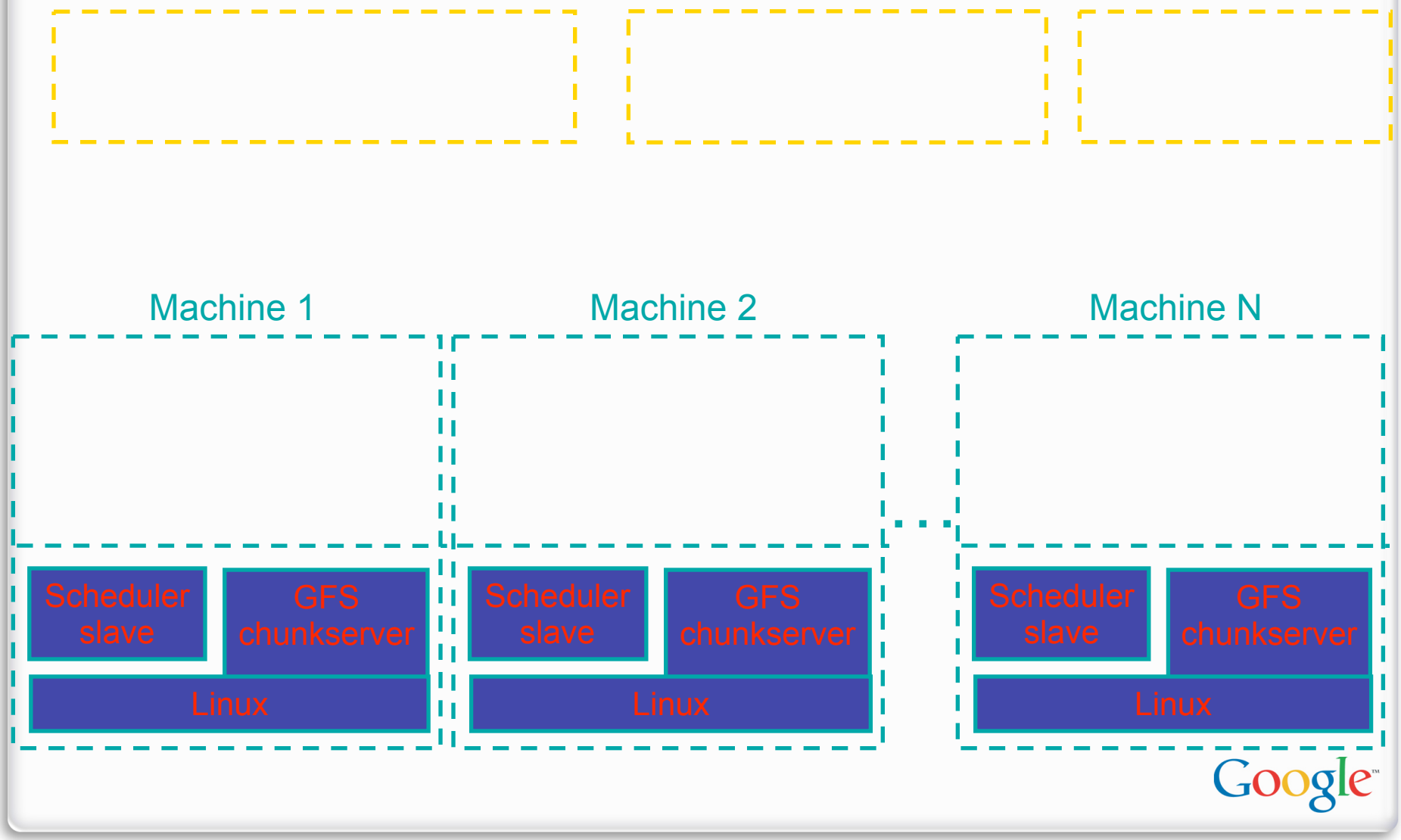
~1000 individual machine failures

~thousands of hard drive failures

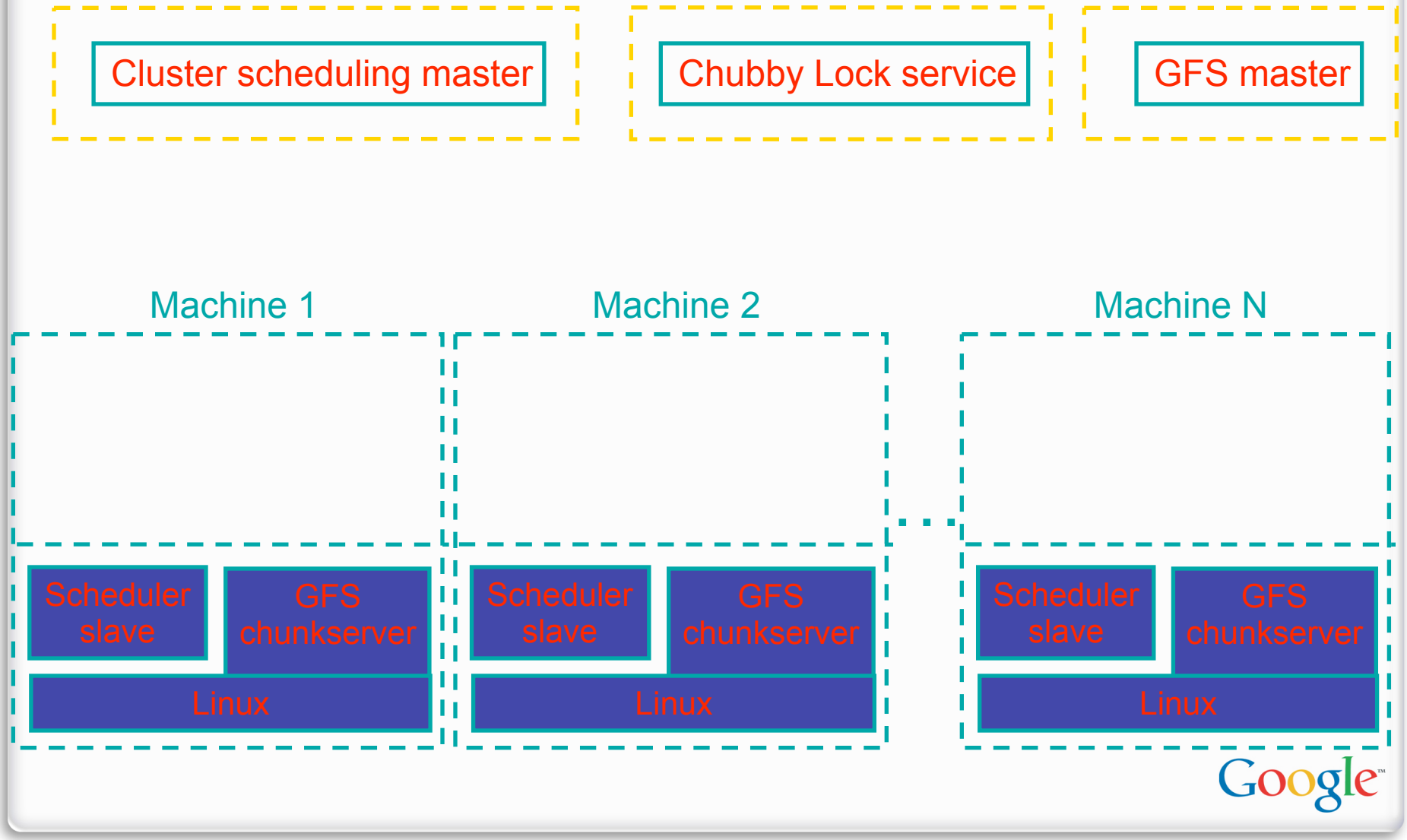
slow disks, bad memory, misconfigured machines, flaky machines, etc.

Google™

Typical Cluster

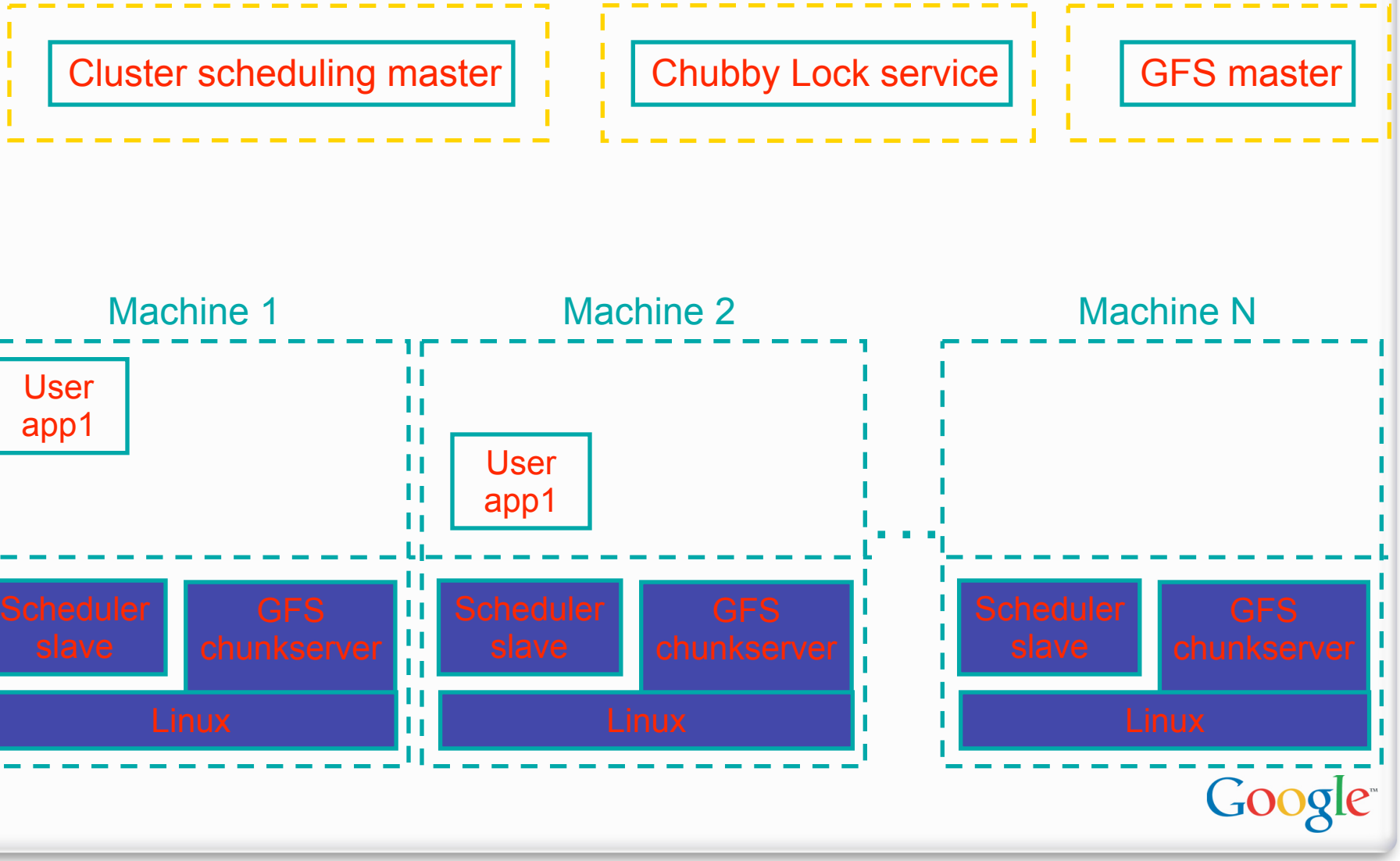


Typical Cluster



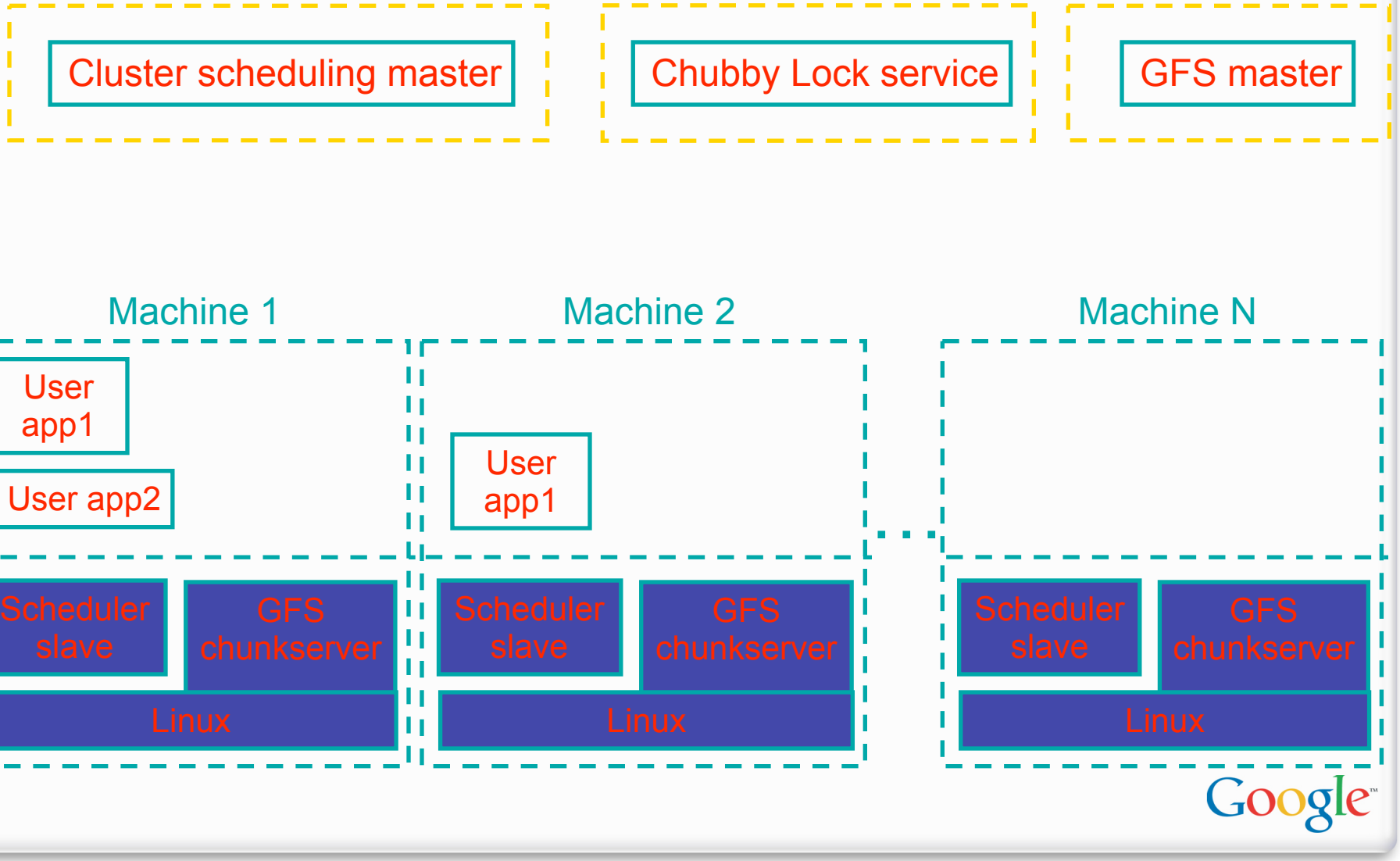
Google™

Typical Cluster



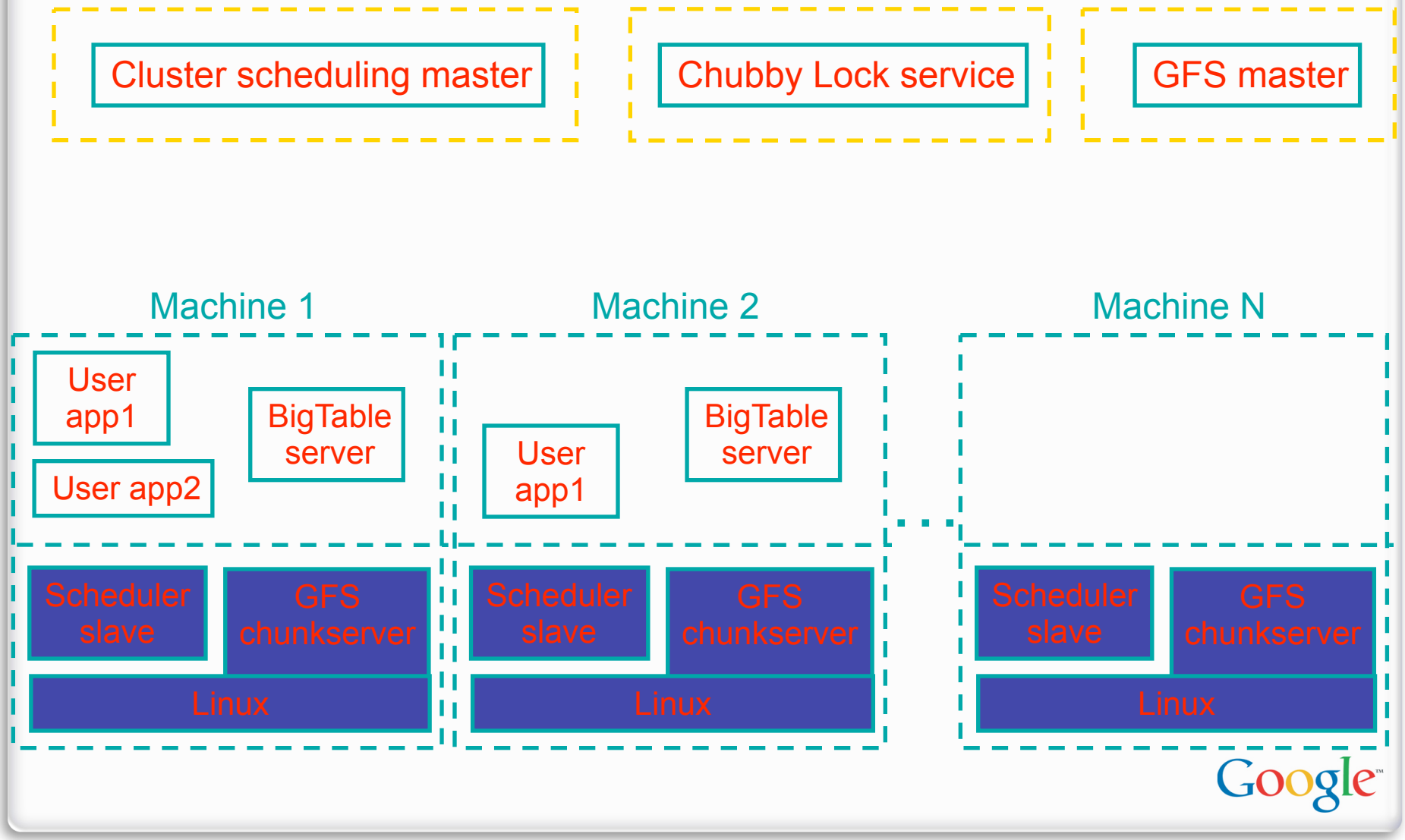
Google™

Typical Cluster

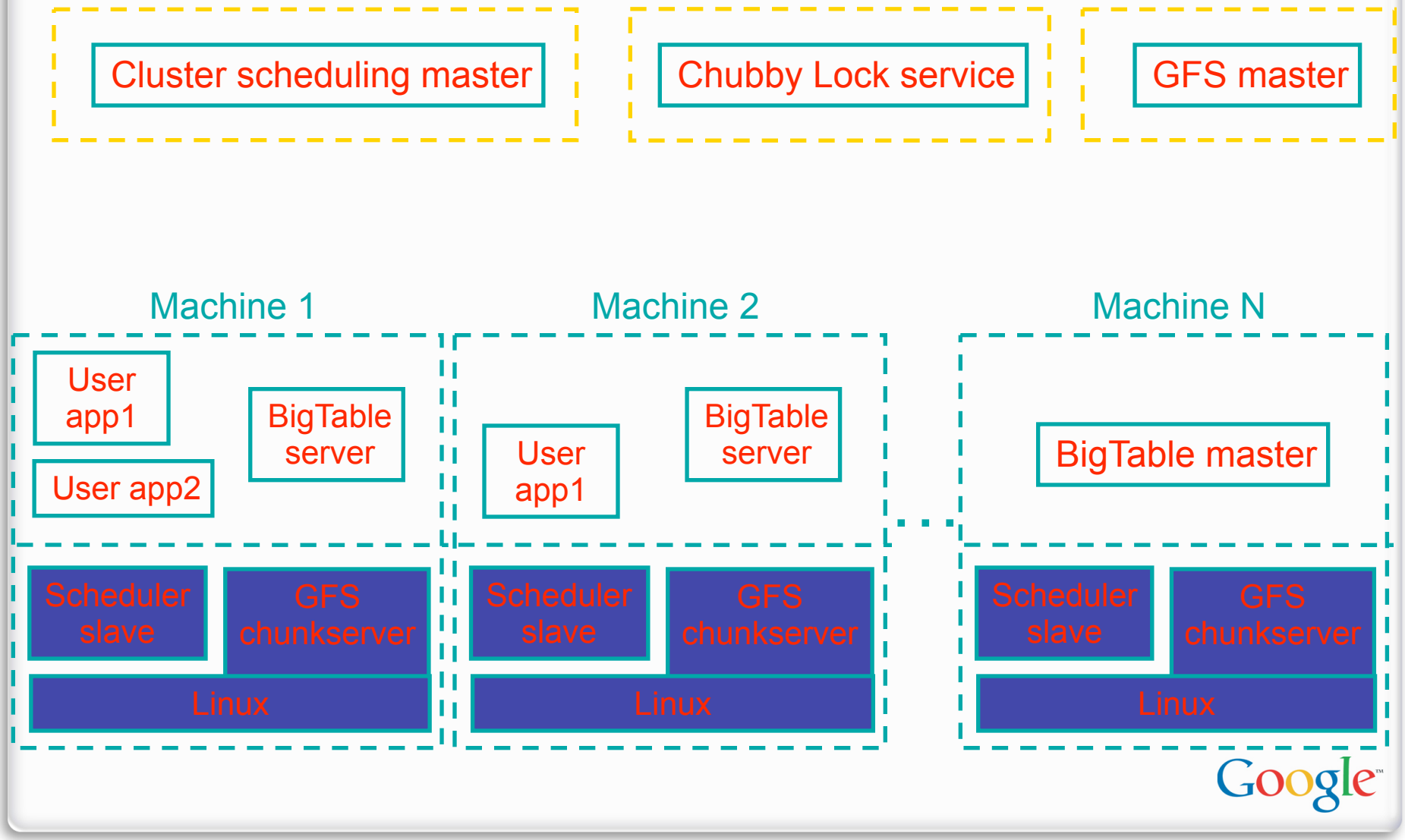


Google

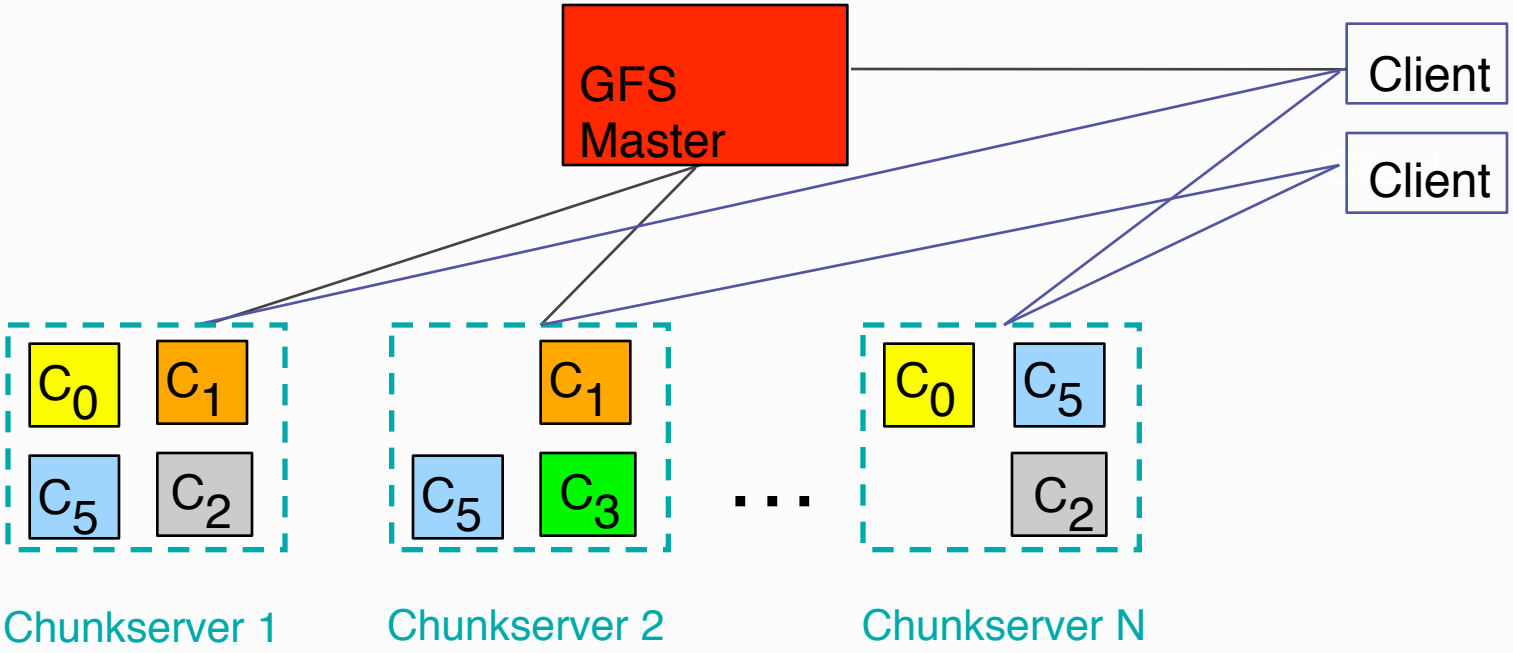
Typical Cluster



Typical Cluster



File Storage: GFS



- Master: Manages file metadata
- Chunkserver: Manages 64MB file chunks
- Clients talk to master to open and find files
- Clients talk directly to chunkservers for data

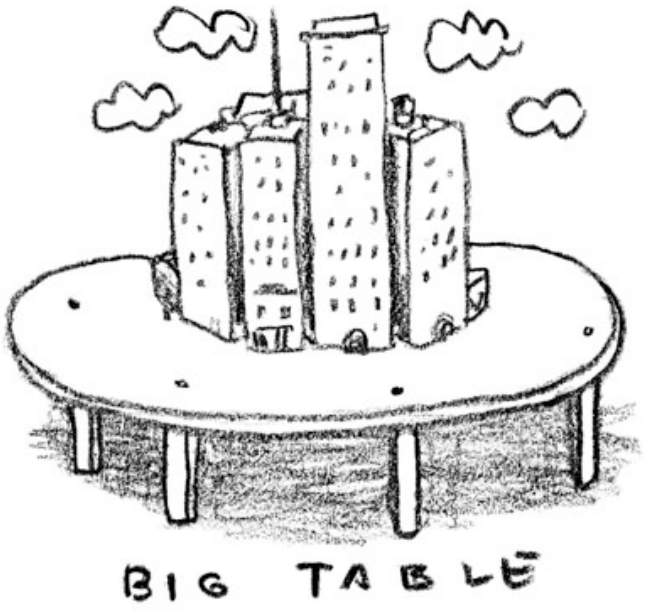
Google

GFS Usage

- 200+ GFS clusters
- Managed by an internal service team
- Largest clusters
  - 5000+ machines
  - 5+ PB of disk usage
  - 10000+ clients

Google™

Data Storage: BigTable



What is it, really?

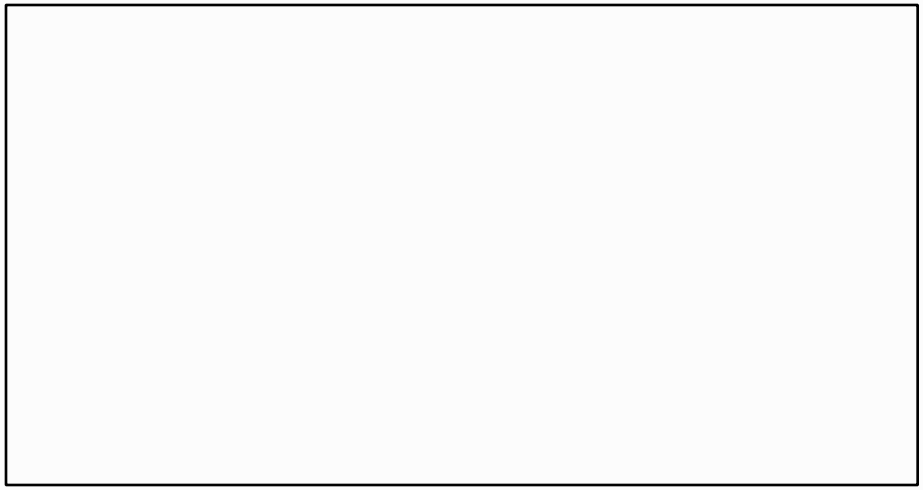
- 10-ft view: Row & column abstraction for storing data
- Reality: Distributed, persistent, multi-level sorted map

Google™

BigTable Data Model

- Multi-dimensional sparse sorted map

  (row, column, timestamp) => *value*

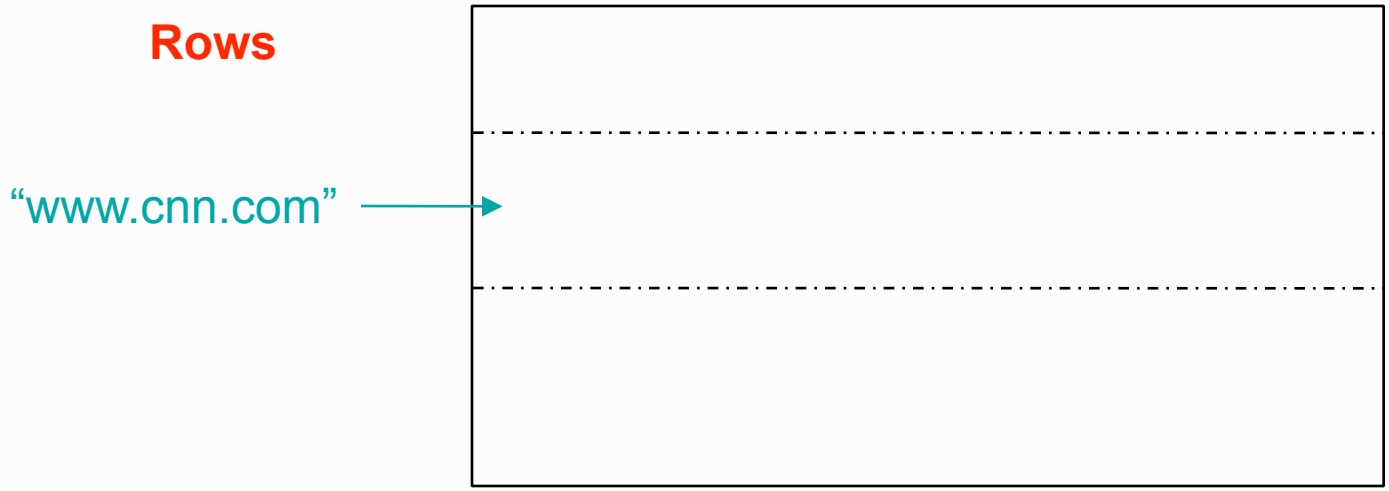


Google™

BigTable Data Model

- Multi-dimensional sparse sorted map

  (row, column, timestamp) => *value*

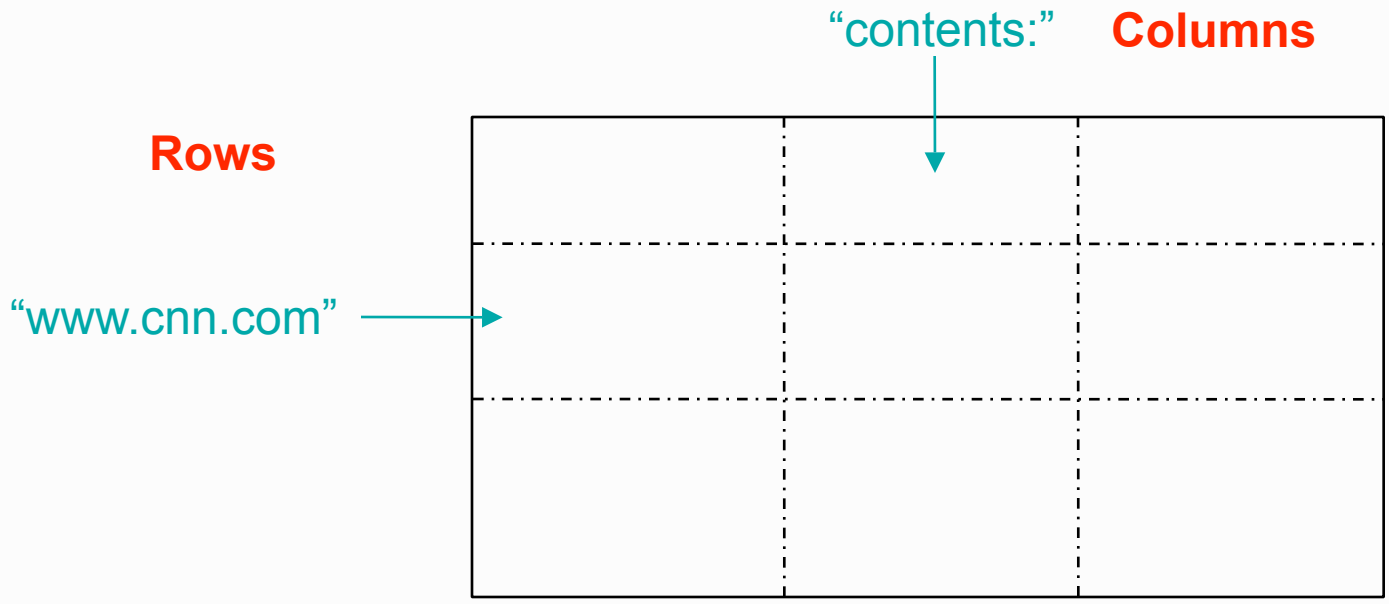


Google™

BigTable Data Model

- Multi-dimensional sparse sorted map

  (row, column, timestamp) => *value*

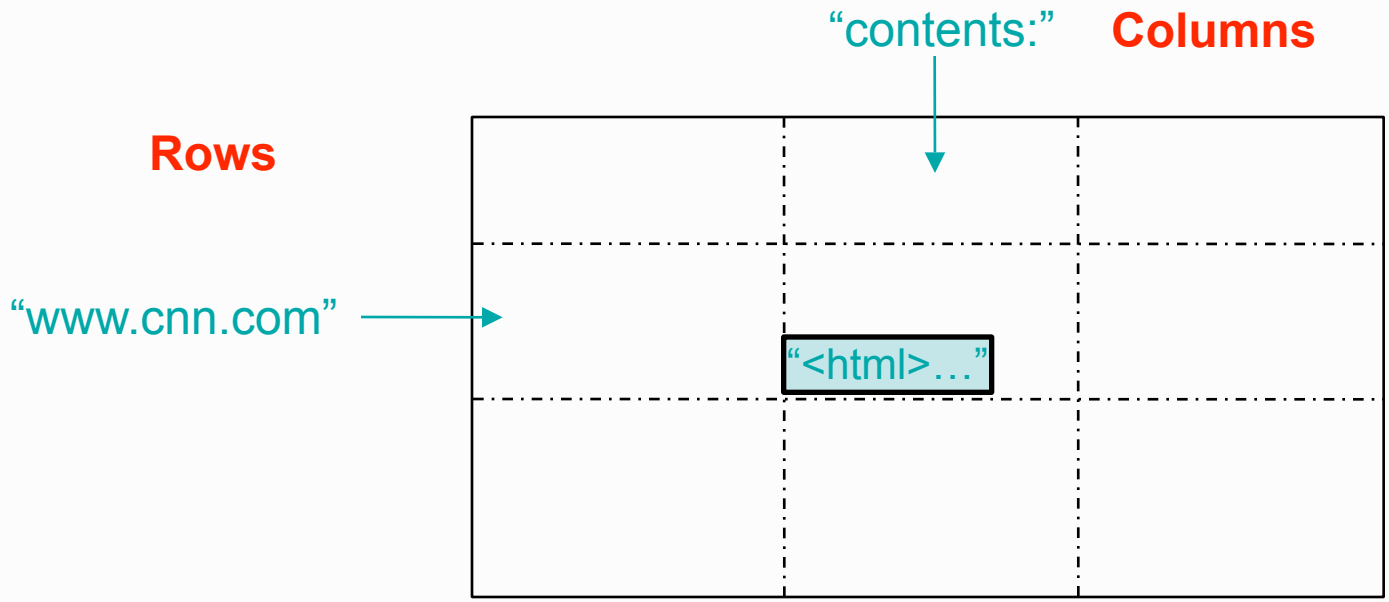


Google

BigTable Data Model

- Multi-dimensional sparse sorted map

  (row, column, timestamp) => *value*

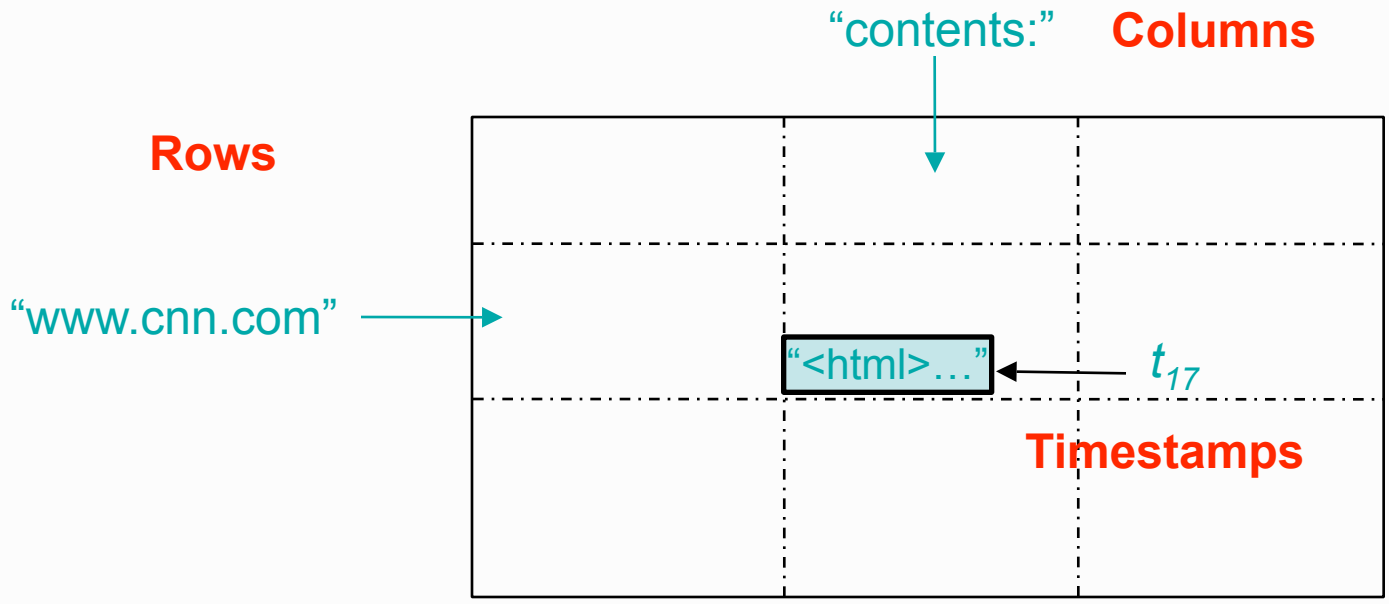


Google™

BigTable Data Model

- Multi-dimensional sparse sorted map

  (row, column, timestamp) => value

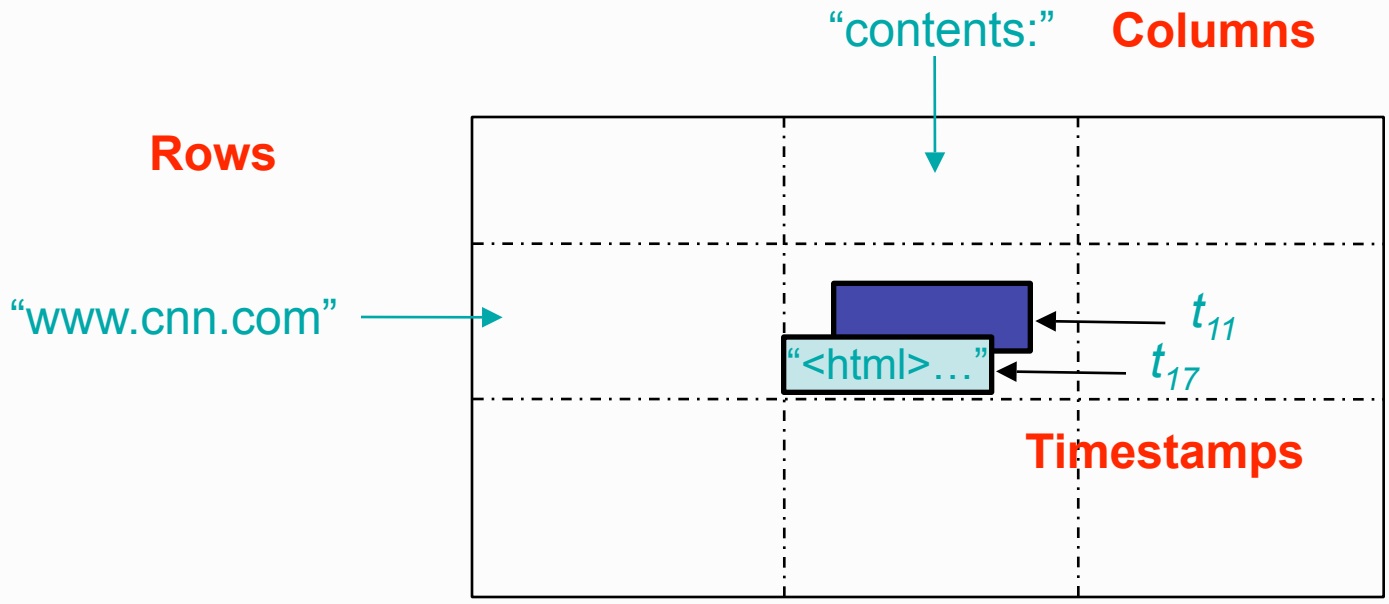


Google

BigTable Data Model

- Multi-dimensional sparse sorted map

  (row, column, timestamp) => value

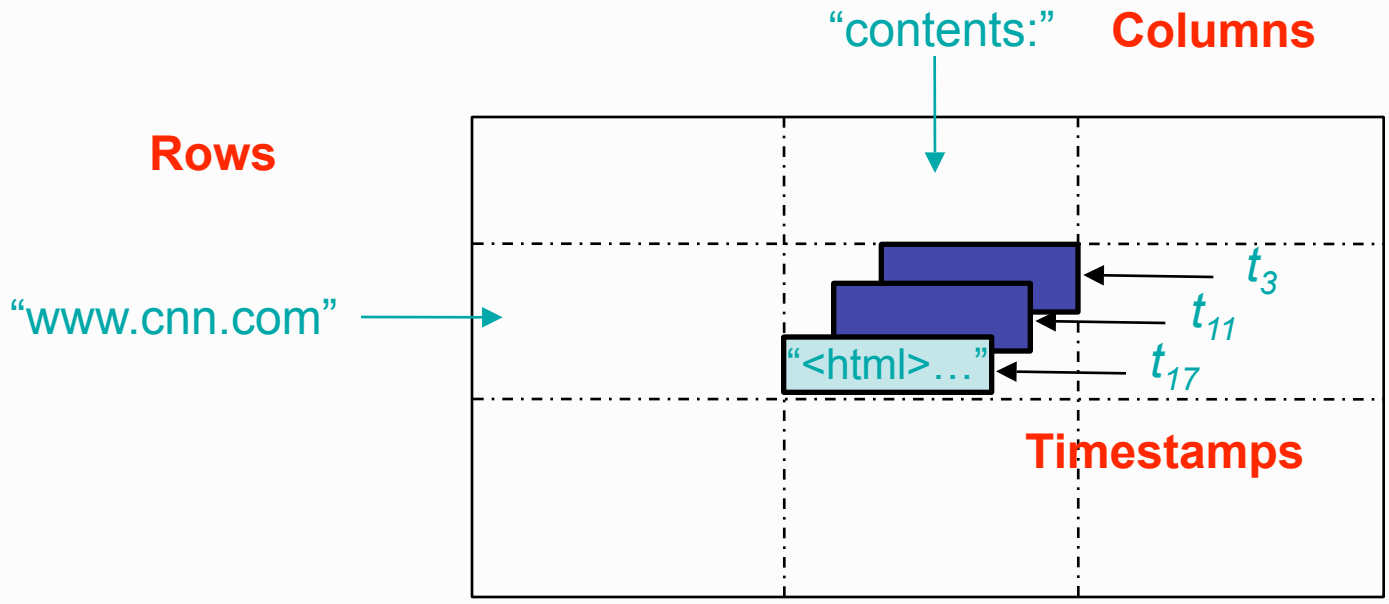


Google

BigTable Data Model

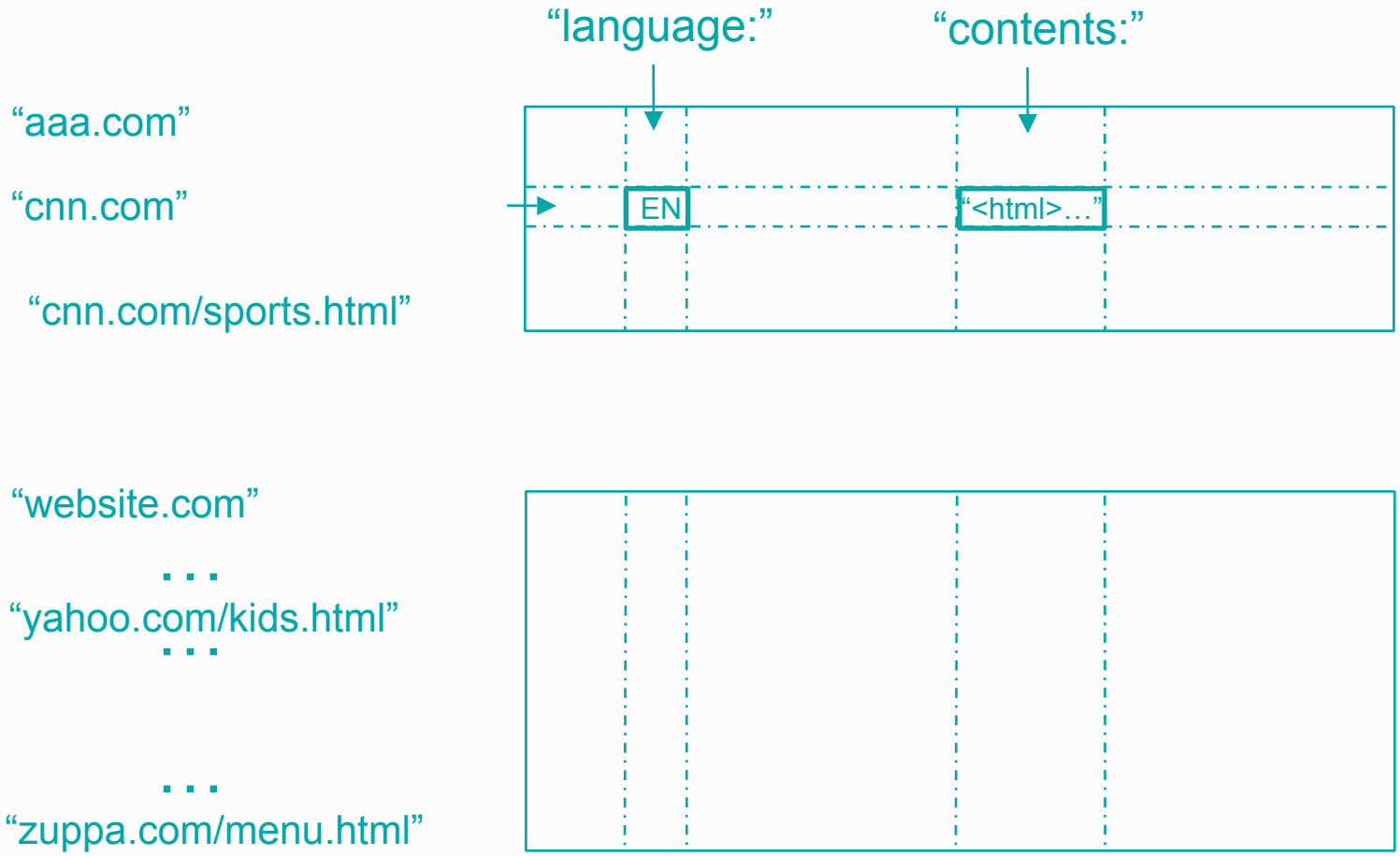
- Multi-dimensional sparse sorted map

  (row, column, timestamp) => value



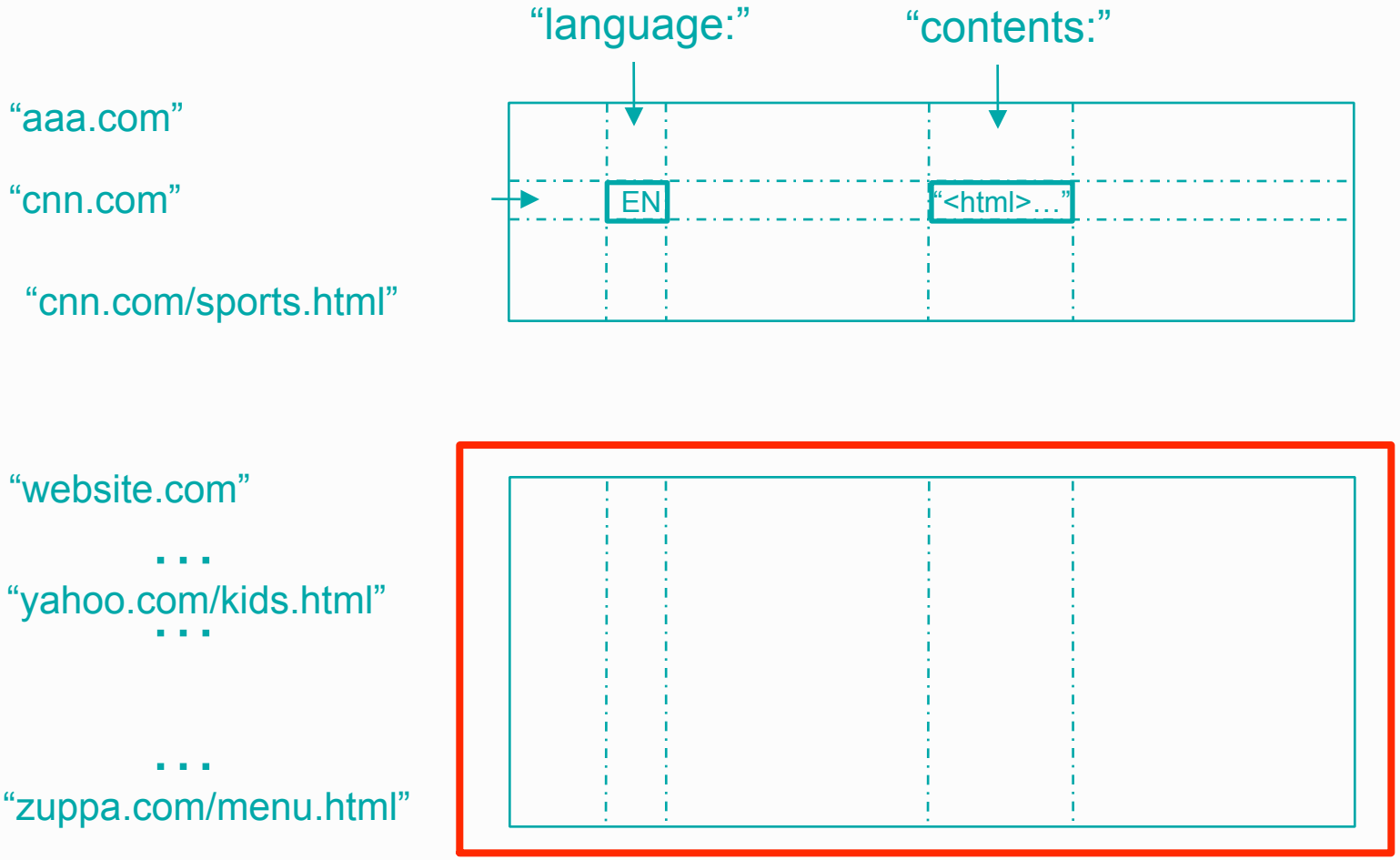
Google

Tablets (cont.)



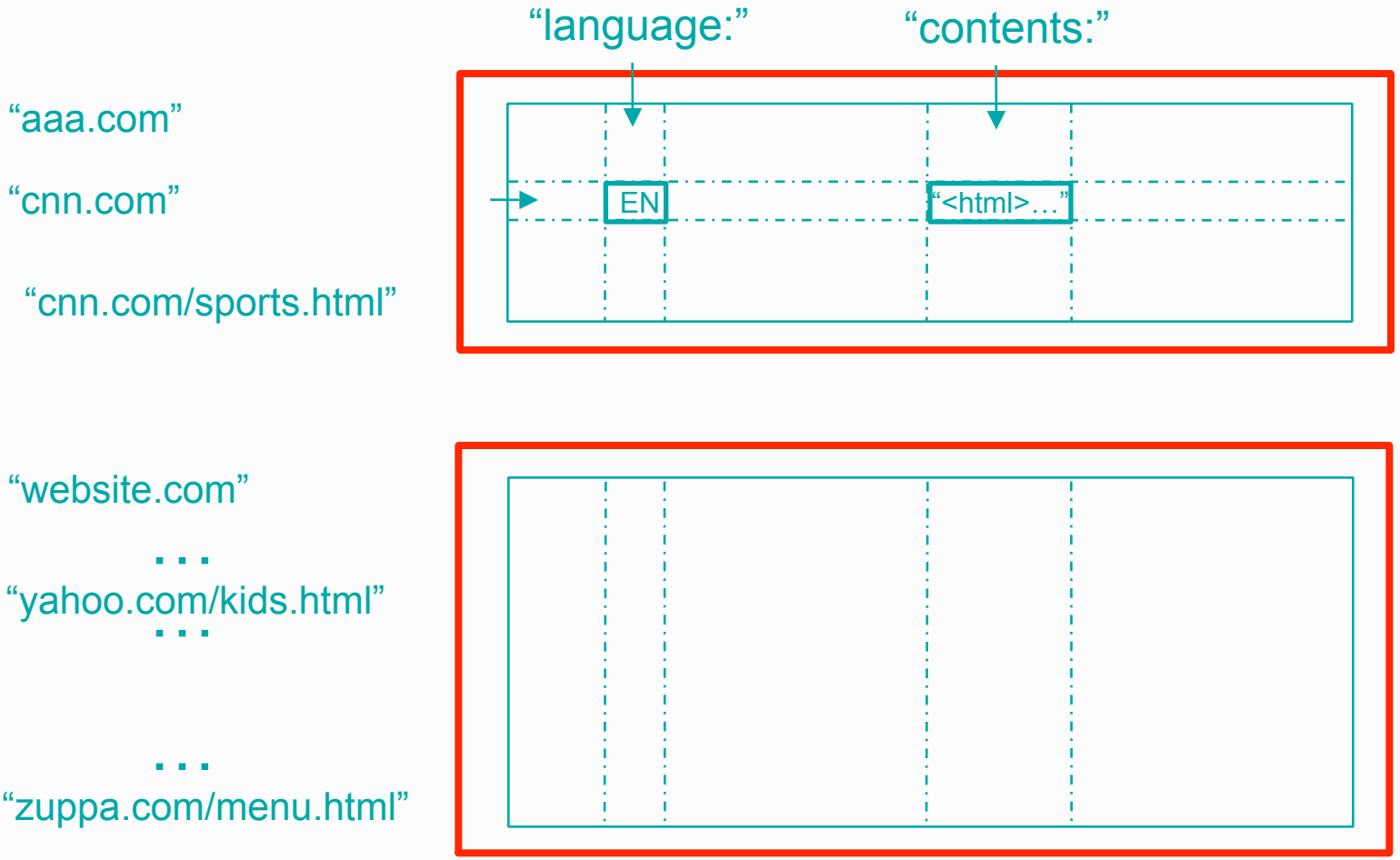
Google

Tablets (cont.)



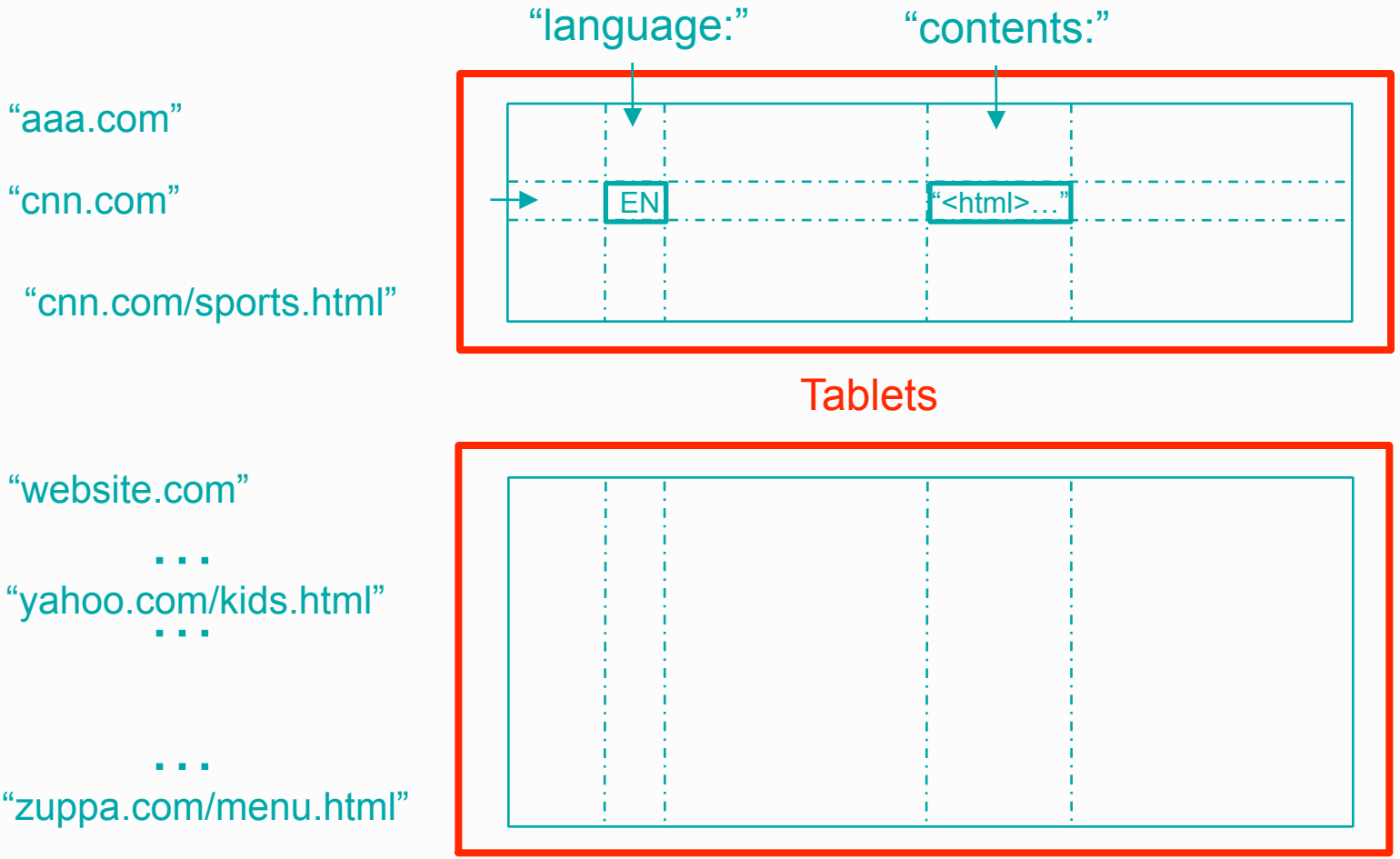
Google

Tablets (cont.)



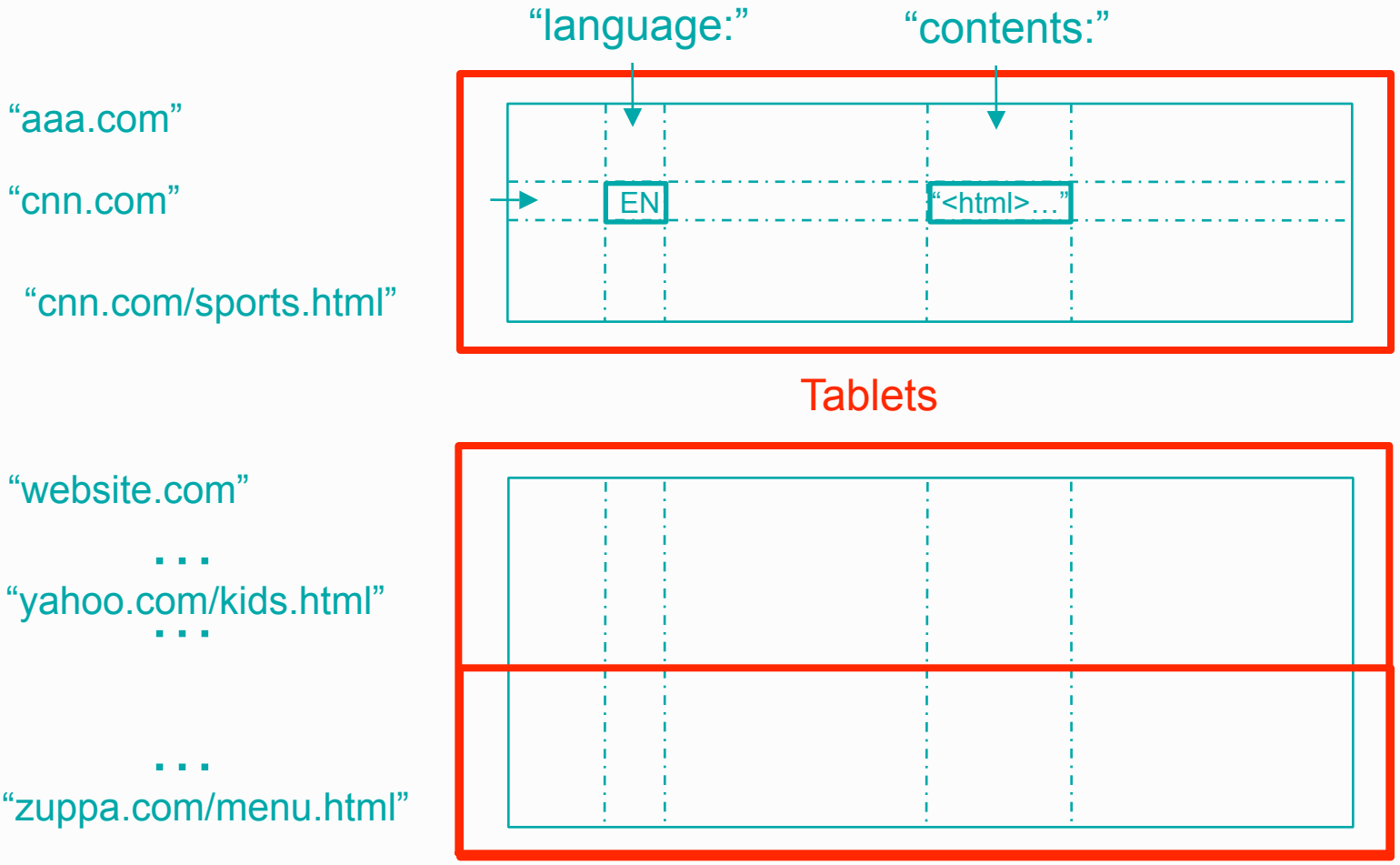
Google

Tablets (cont.)



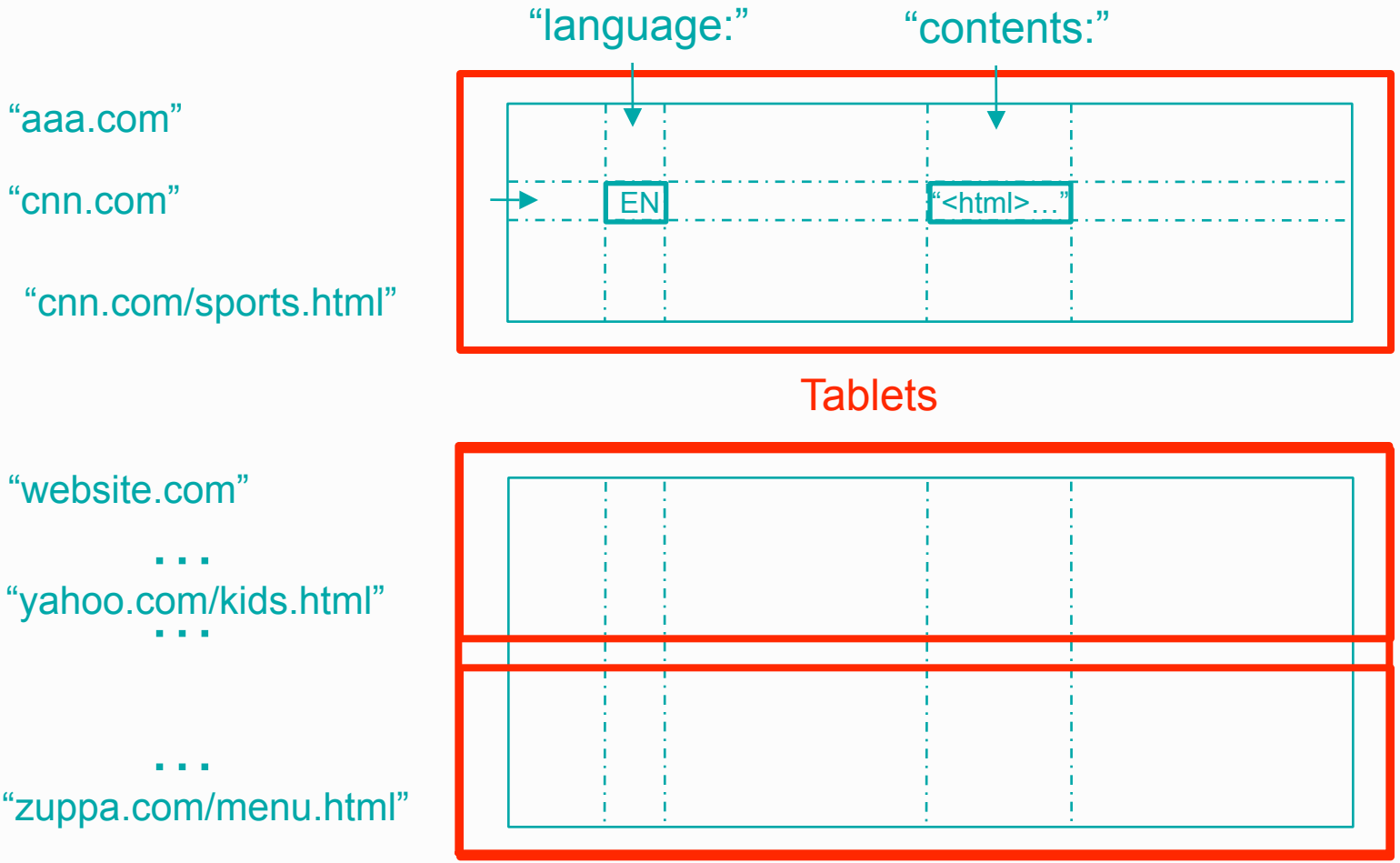
Google

Tablets (cont.)



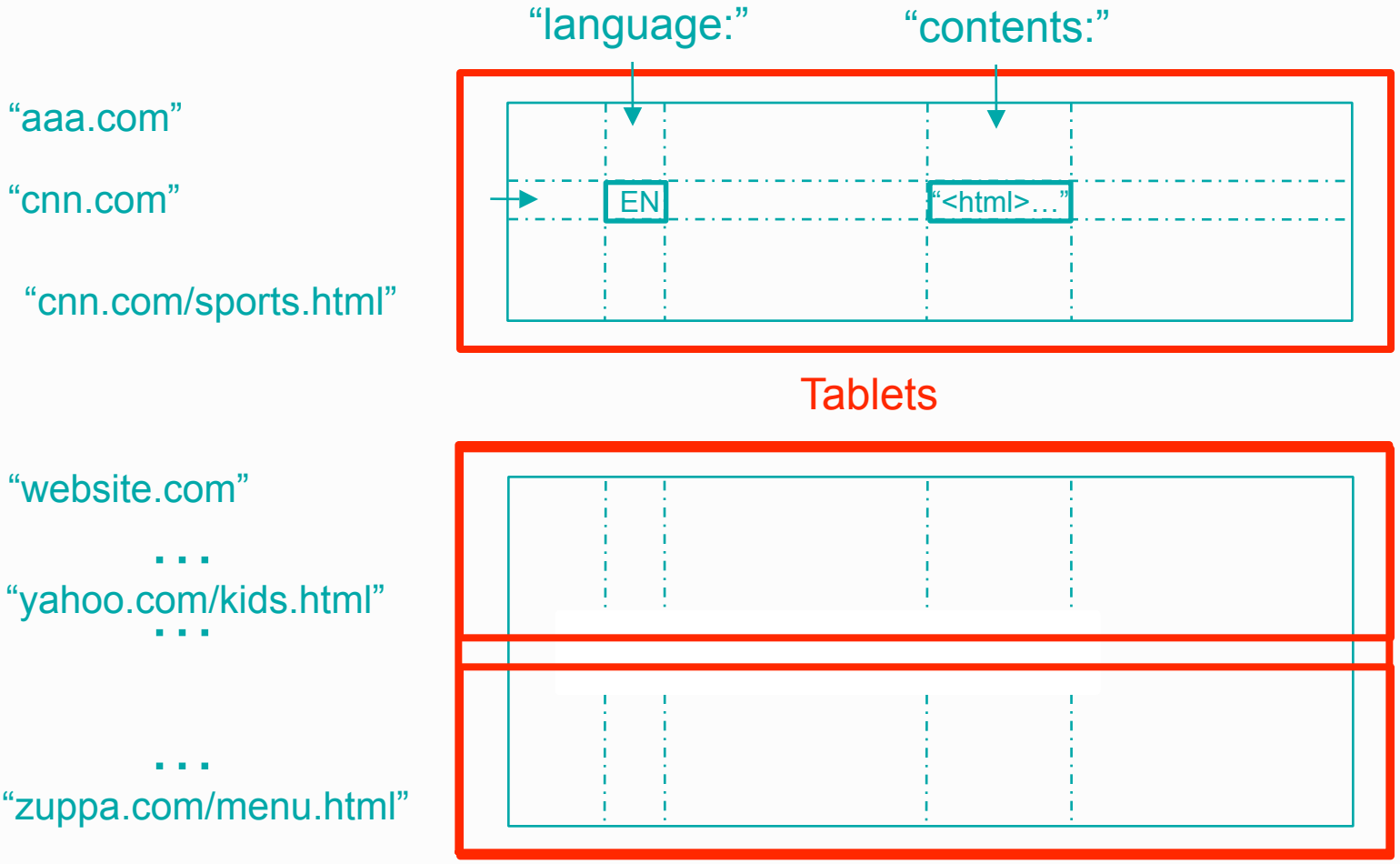
Google

Tablets (cont.)



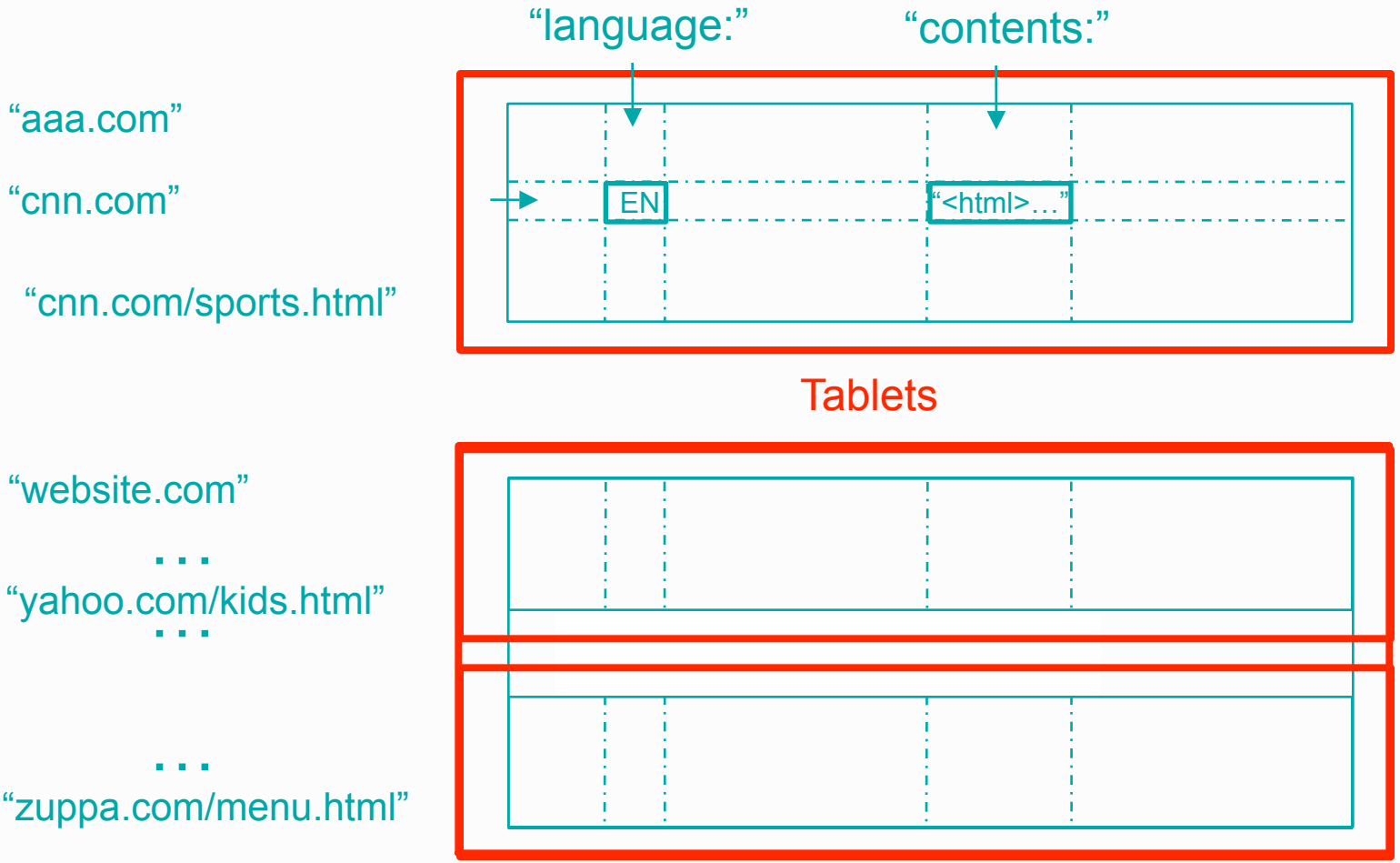
Google™

Tablets (cont.)



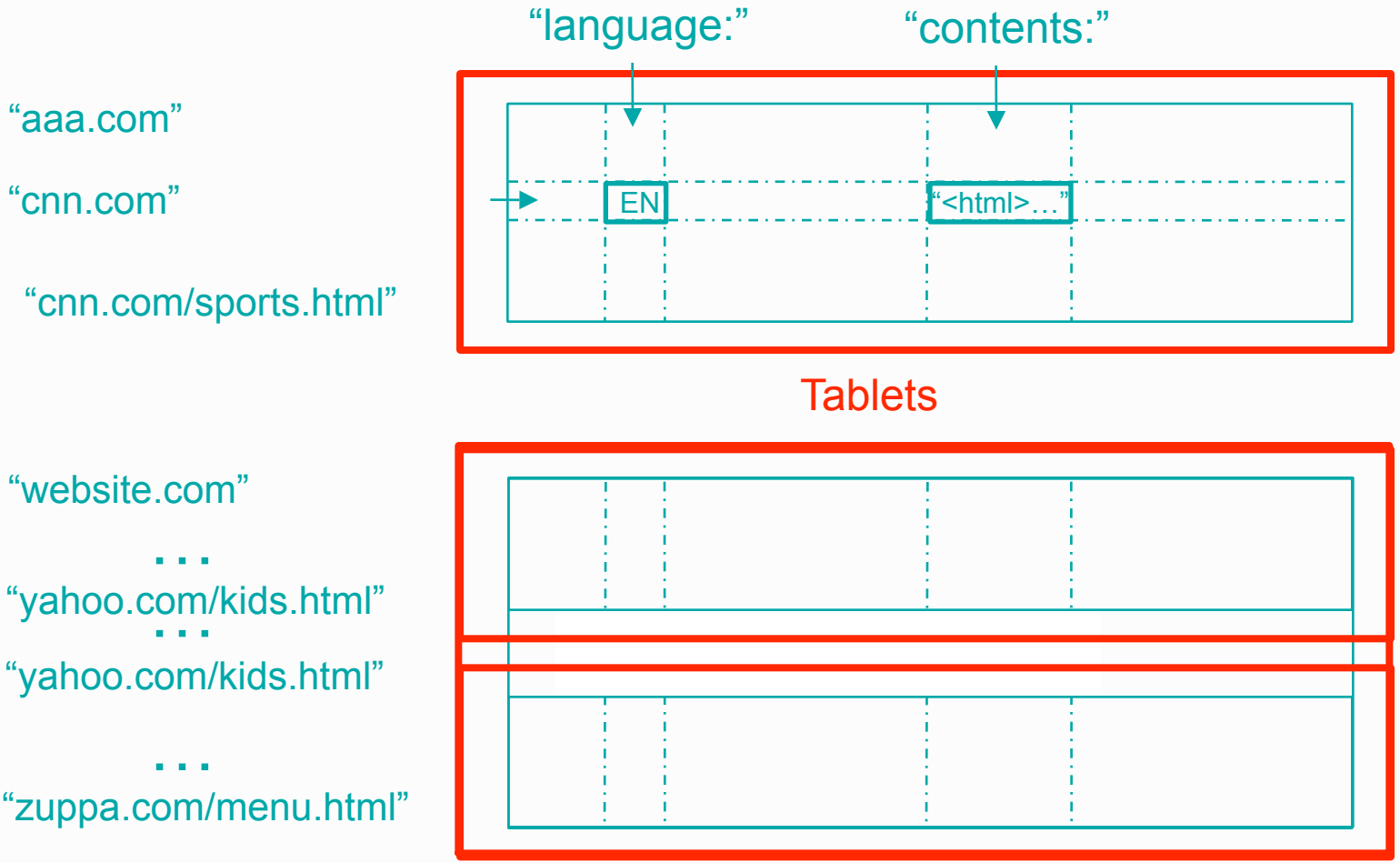
Google

Tablets (cont.)



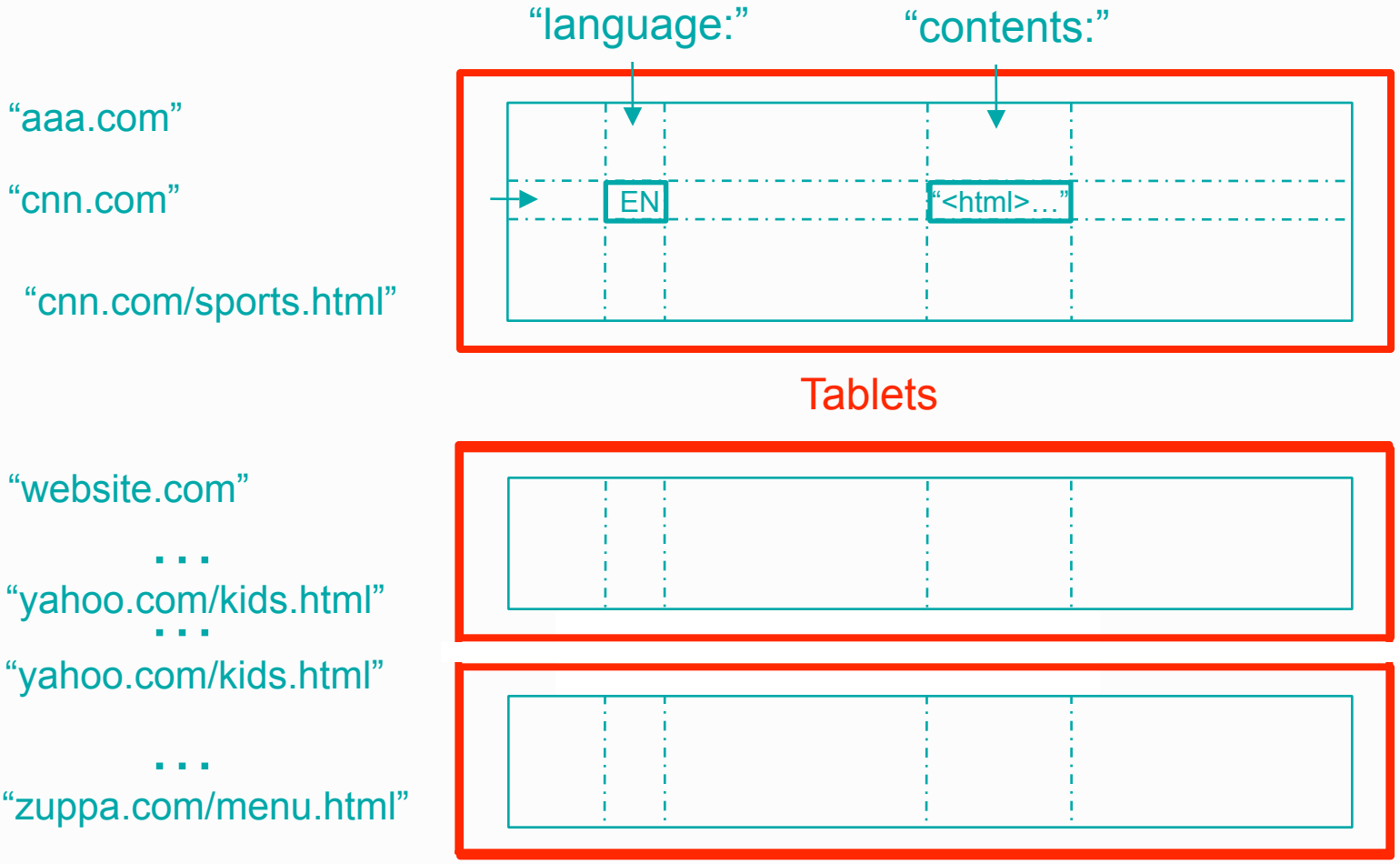
Google

Tablets (cont.)



Google™

Tablets (cont.)



Google

Bigtable System Structure

Bigtable Cell

Bigtable master

Bigtable tablet server | Bigtable tablet server | … | Bigtable tablet server

Google™

Bigtable System Structure

Bigtable Cell

Bigtable master

performs metadata ops +
load balancing

Bigtable tablet server

Bigtable tablet server

…

Bigtable tablet server

Google™

Bigtable System Structure

Bigtable Cell

Bigtable master

performs metadata ops + load balancing

Bigtable tablet server

serves data

Bigtable tablet server

serves data

…

Bigtable tablet server

serves data

Google™

Bigtable System Structure

Bigtable Cell

Bigtable master

performs metadata ops + load balancing

Bigtable tablet server

serves data

Bigtable tablet server

serves data

…

Bigtable tablet server

serves data

Cluster scheduling system

GFS

Lock service

Google™

Bigtable System Structure

Bigtable Cell

Bigtable master

performs metadata ops + load balancing

Bigtable tablet server

serves data

Bigtable tablet server

serves data

…

Bigtable tablet server

serves data

Cluster scheduling system

handles failover, monitoring

GFS

Lock service

Google™

Bigtable System Structure

Bigtable Cell

Bigtable master

performs metadata ops + load balancing

Bigtable tablet server

serves data

Bigtable tablet server

serves data

…

Bigtable tablet server

serves data

Cluster scheduling system

handles failover, monitoring

GFS

holds tablet data, logs

Lock service

Google™

Bigtable System Structure

Bigtable Cell

Bigtable master

performs metadata ops +
load balancing

Bigtable tablet server

serves data

Bigtable tablet server

serves data

…

Bigtable tablet server

serves data

Cluster scheduling system

handles failover, monitoring

GFS

holds tablet data, logs

Lock service

holds metadata,
handles master-election

Google™

Bigtable System Structure

Bigtable Cell

Bigtable client

Bigtable client library

Bigtable master

performs metadata ops + load balancing

Bigtable tablet server

serves data

Bigtable tablet server

serves data

…

Bigtable tablet server

serves data

Cluster scheduling system

handles failover, monitoring

GFS

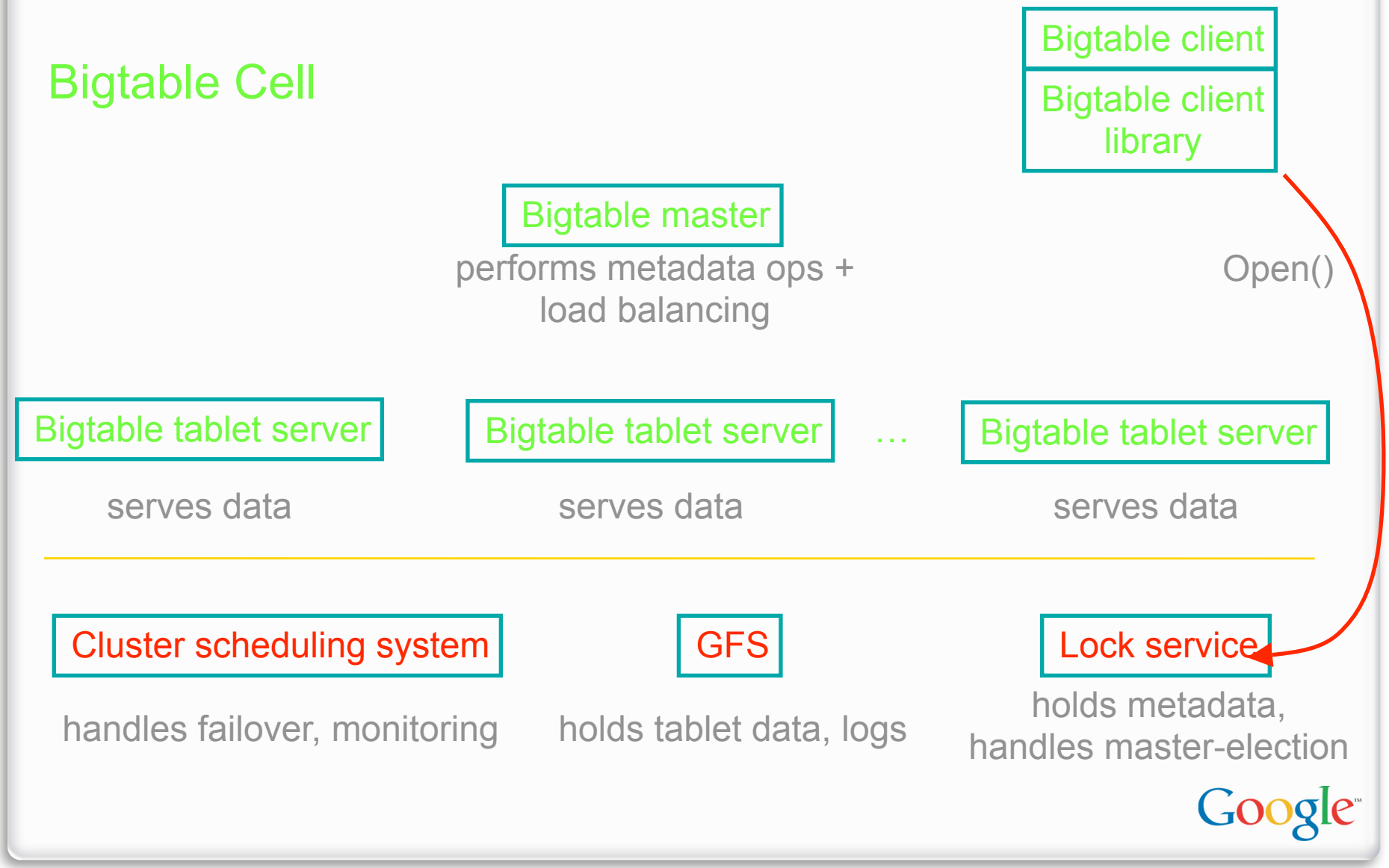
holds tablet data, logs

Lock service

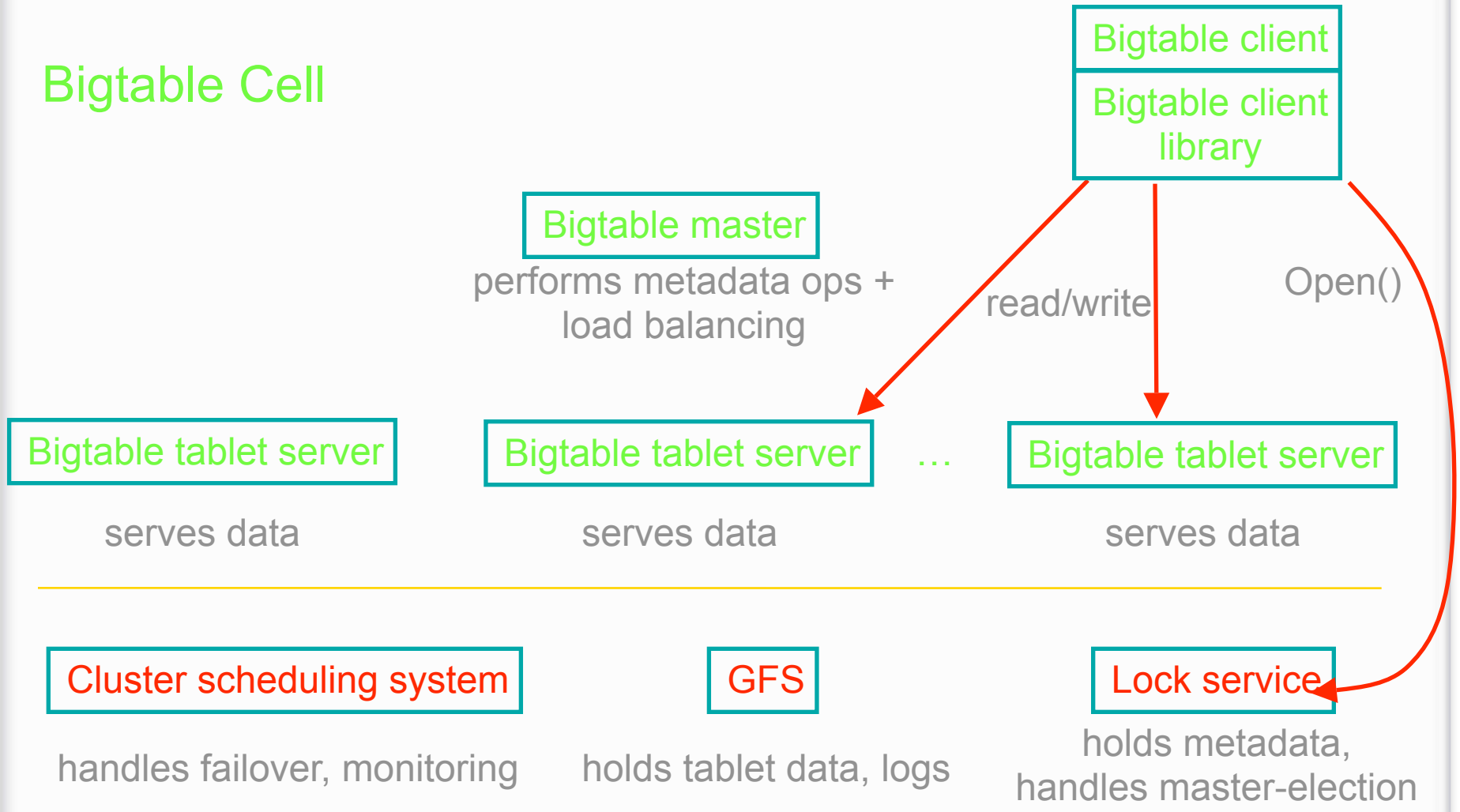
holds metadata, handles master-election

Google™

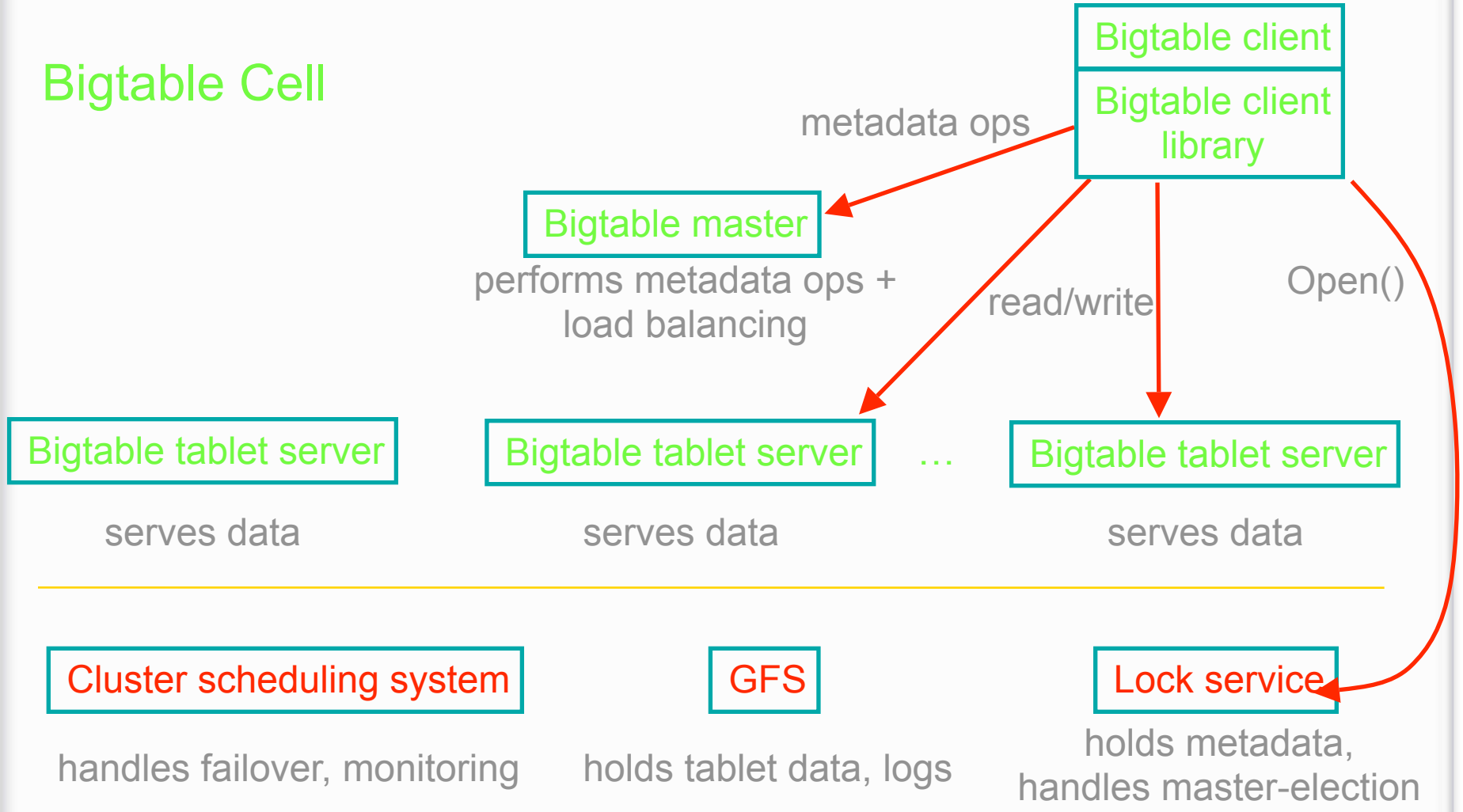
Bigtable System Structure



Bigtable System Structure



Bigtable System Structure



Some BigTable Features

- *Single-row transactions*: easy to do read/modify/write operations
- *Locality groups*: segregate columns into different files
- *In-memory columns*: random access to small items
- *Suite of compression techniques*: per-locality group
- *Bloom filters*: avoid seeks for non-existent data
- *Replication*: eventual-consistency replication across datacenters, between multiple BigTable serving setups (master/slave & multi-master)

Google™

BigTable Usage

- 500+ BigTable cells
- Largest cells manage 6000TB+ of data, 3000+ machines
- Busiest cells sustain >500000+ ops/ second 24 hours/day, and peak much higher

Google™

Data Processing: MapReduce

- Google's batch processing tool of choice
- Users write two functions:
  - Map: Produces (key, value) pairs from input
  - Reduce: Merges (key, value) pairs from Map
- Library handles data transfer and failures
- Used everywhere: Earth, News, Analytics, Search Quality, Indexing, …

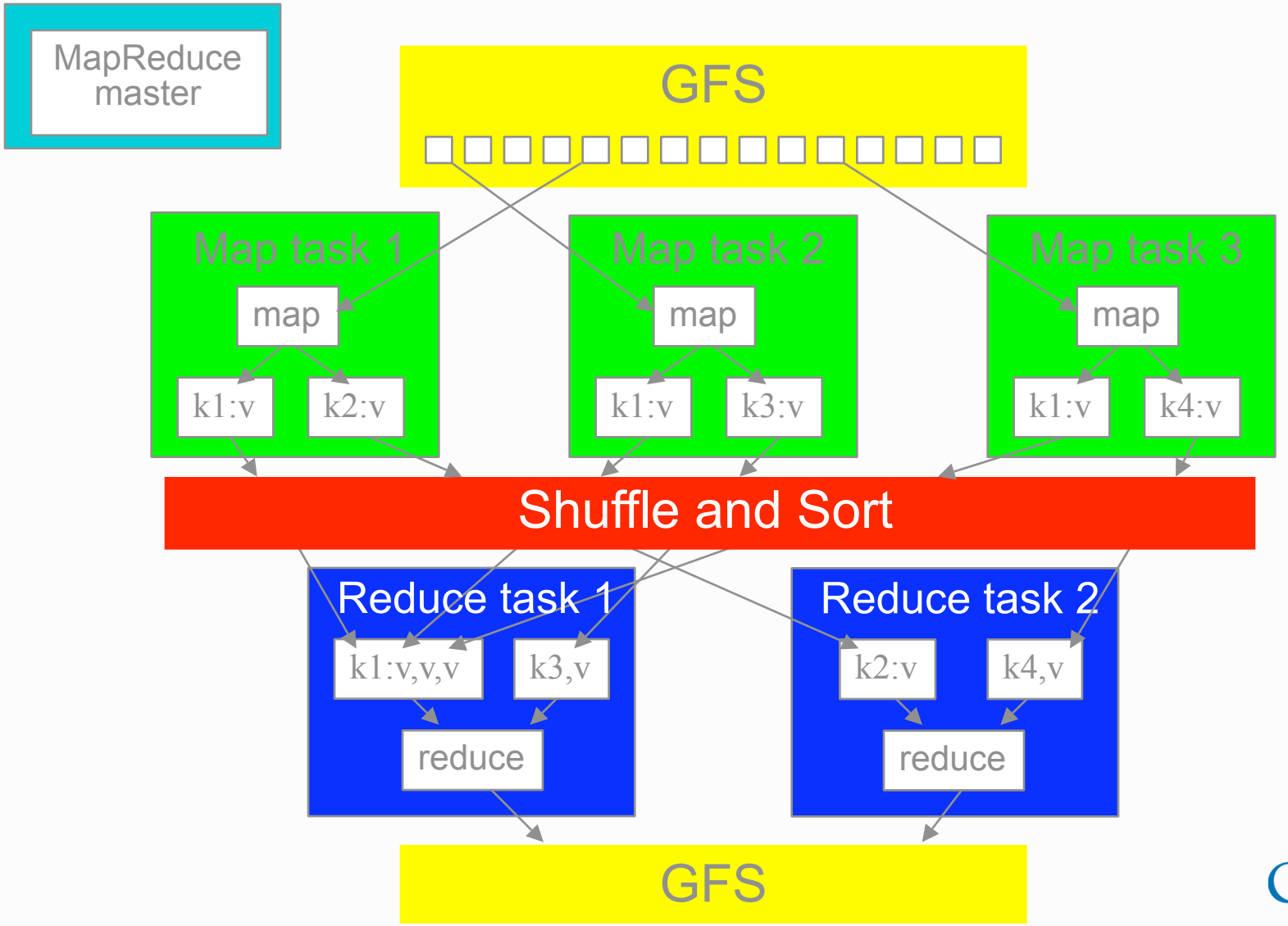
Google™

Example: Document Indexing

- Input: Set of documents $D_1, \ldots, D_N$
- Map
  - Parse document D into terms $T_1, \ldots, T_N$
  - Produces (key, value) pairs
    - $(T_1, D), \ldots, (T_N, D)$
- Reduce
  - Receives list of (key, value) pairs for term T
    - $(T, D_1), \ldots, (T, D_N)$
  - Emits single (key, value) pair
    - $(T, (D_1, \ldots, D_N))$

Google

MapReduce Execution

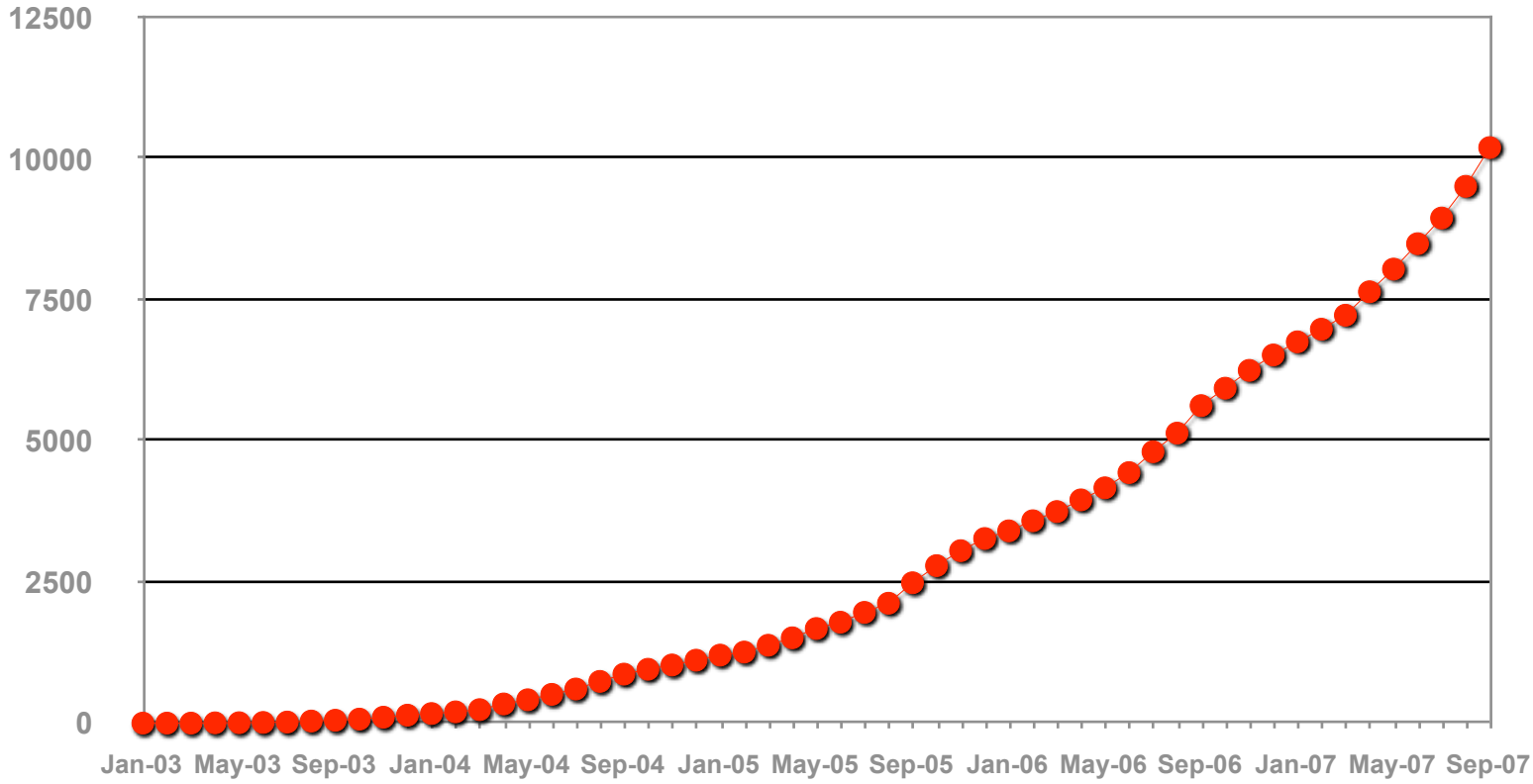


Google

MapReduce Tricks / Features

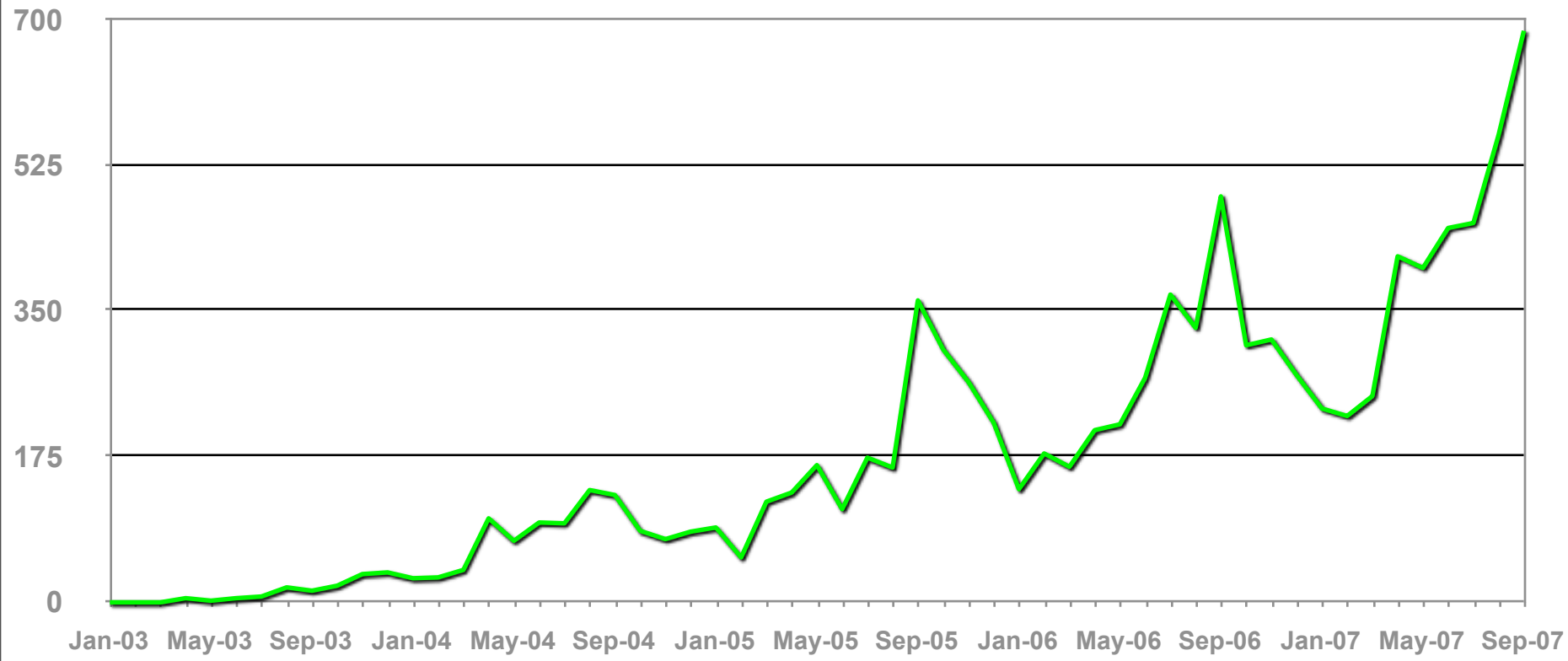
- Data locality
- Multiple I/O data types
- Data compression
- Pipelined shuffle stage
- Fast sorter
- Backup copies of tasks
- # tasks >> # machines
- Task re-execution on failure
- Local or cluster execution
- Distributed counters

MapReduce Programs in Google's Source Tree



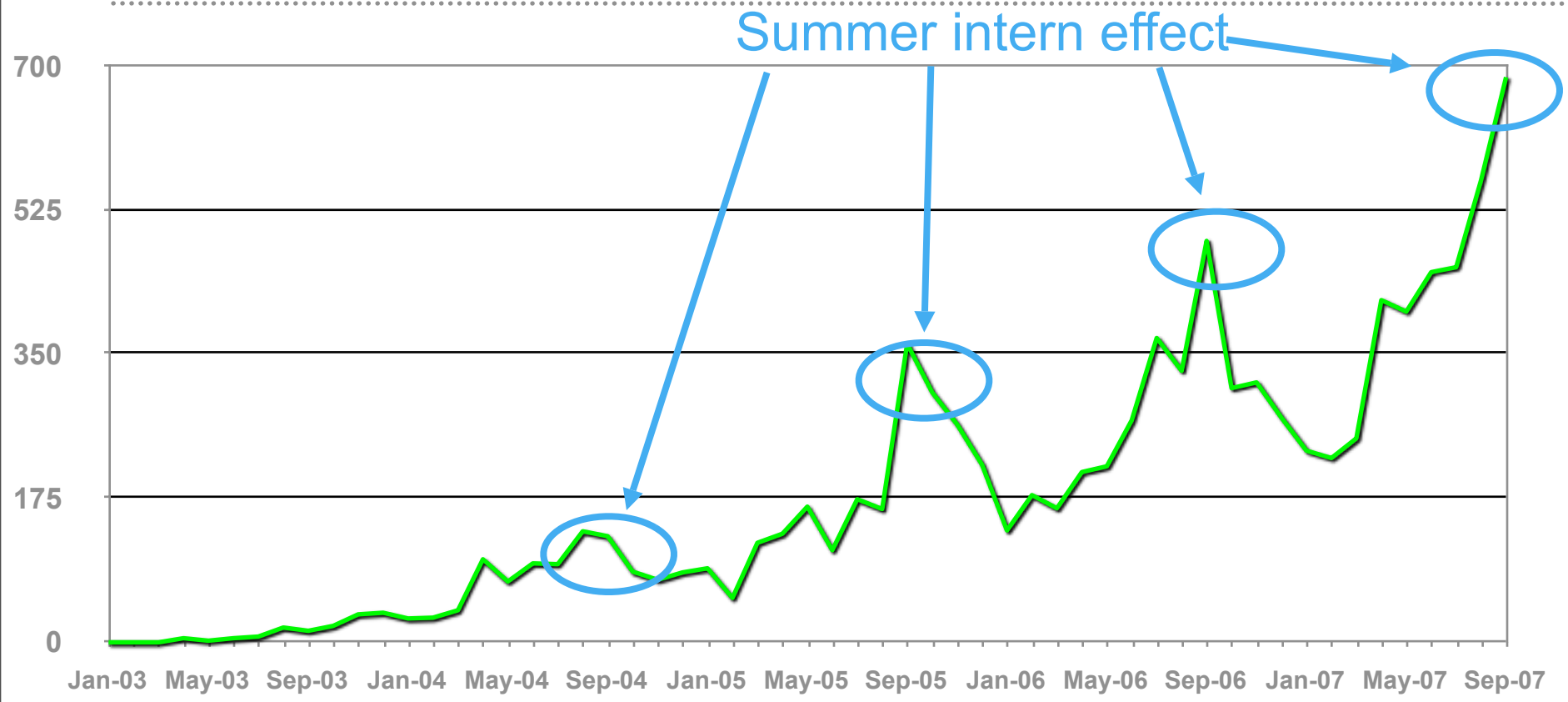
Google

New MapReduce Programs Per Month



Google

New MapReduce Programs Per Month



Google

MapReduce in Google

Easy to use. Library hides complexity.

	Mar, ‘05	Mar, ‘06	Sep, '07
Number of jobs	72K	171K	2,217K
Average time (seconds)	934	874	395
Machine years used	981	2,002	11,081
Input data read (TB)	12,571	52,254	403,152
Intermediate data (TB)	2,756	6,743	34,774
Output data written (TB)	941	2,970	14,018
Average worker machines	232	268	394

Google™

Current Work

Scheduling system + GFS + BigTable + MapReduce work well within single clusters

Many separate instances in different data centers

- Tools on top deal with cross-cluster issues
- Each tool solves relatively narrow problem
  - Many tools => lots of complexity

Can next generation infrastructure do more?

Google™

Next Generation Infrastructure

Truly global systems to span all our datacenters

- Global namespace with many replicas of data worldwide
- Support both consistent and inconsistent operations
- Continued operation even with datacenter partitions
- Users specify high-level desires:

  "99%ile latency for accessing this data should be <50ms"

  "Store this data on at least 2 disks in EU, 2 in U.S. & 1 in Asia"

– Increased utilization through automation

– Automatic migration, growing and shrinking of services

– Lower end-user latency

– Provide high-level programming model for data-intensive interactive services

Google™

Questions?

Further info:

- *The Google File System*, Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, SOSP '03.
- *Web Search for a Planet: The Google Cluster Architecture*, Luiz Andre Barroso, Jeffrey Dean, Urs Hölzle, IEEE Micro, 2003.
- *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'06
- *MapReduce: Simplified Data Processing on Large Clusters*, Jeffrey Dean and Sanjay Ghemawat, OSDI'04
- *Failure Trends in a Large Disk Drive Population,* Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz André Barroso. FAST, '07.

http://labs.google.com/papers

http://labs.google.com/people/jeff or jeff@google.com

Google™