Bigtable

A Distributed Storage System for Structured Data

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

Google, Inc.

Presented by: Emanuele Rocca

What is Bigtable?

BIG

TABLE

What is Bigtable?

Let's start saying what Bigtable is **NOT**

- **Not** a database
- **Not** a *sharded* database
- **Not** a distributed hashtable

What is Bigtable?

A distributed, persistent, sorted, associative array

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

Why did they implement it?

Quoting Jeff Dean:

- Applications at Google place very different demands on the storage system
- Handle petabytes of data
- Scale to thousands of commodity servers
- Fun

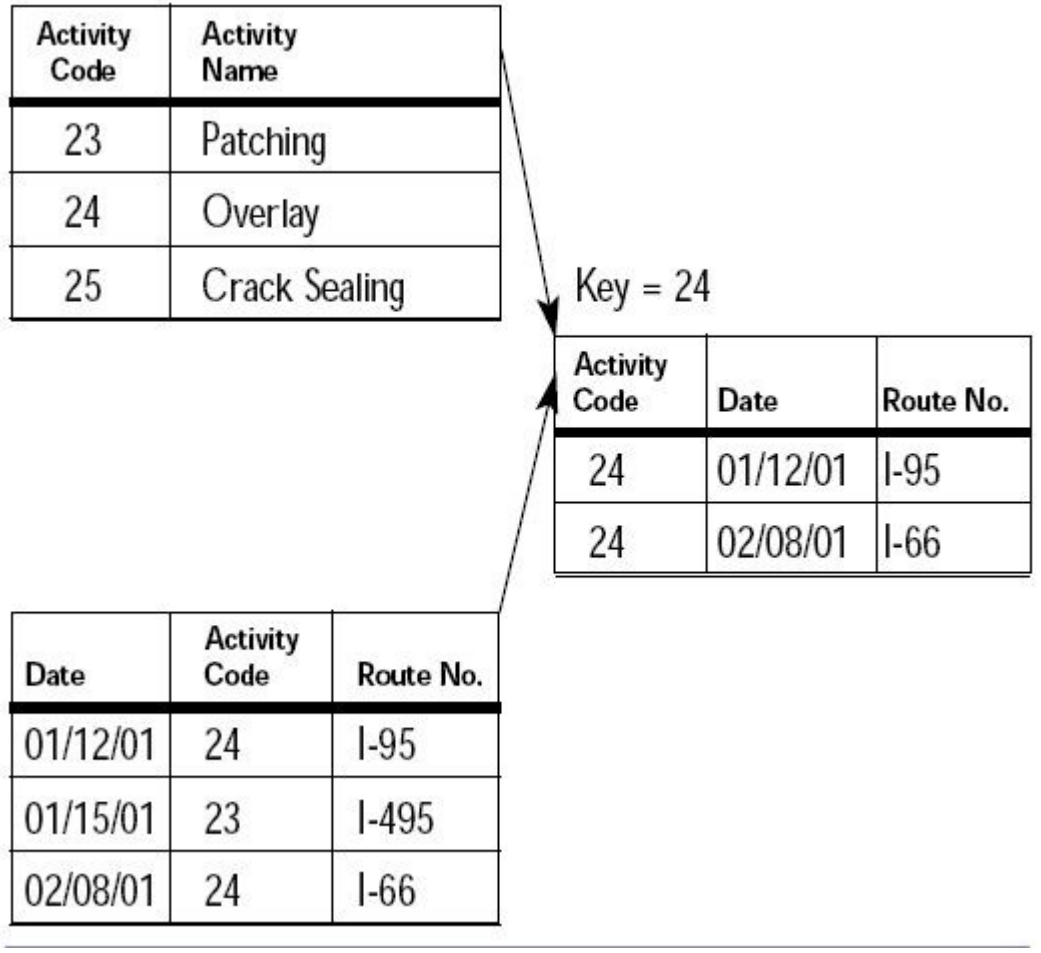
Outline

- Data model
- Architecture
- Use cases
- Performance evaluation
- Great excitement

Data model

Data Model

Remember the Relational Model?



Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

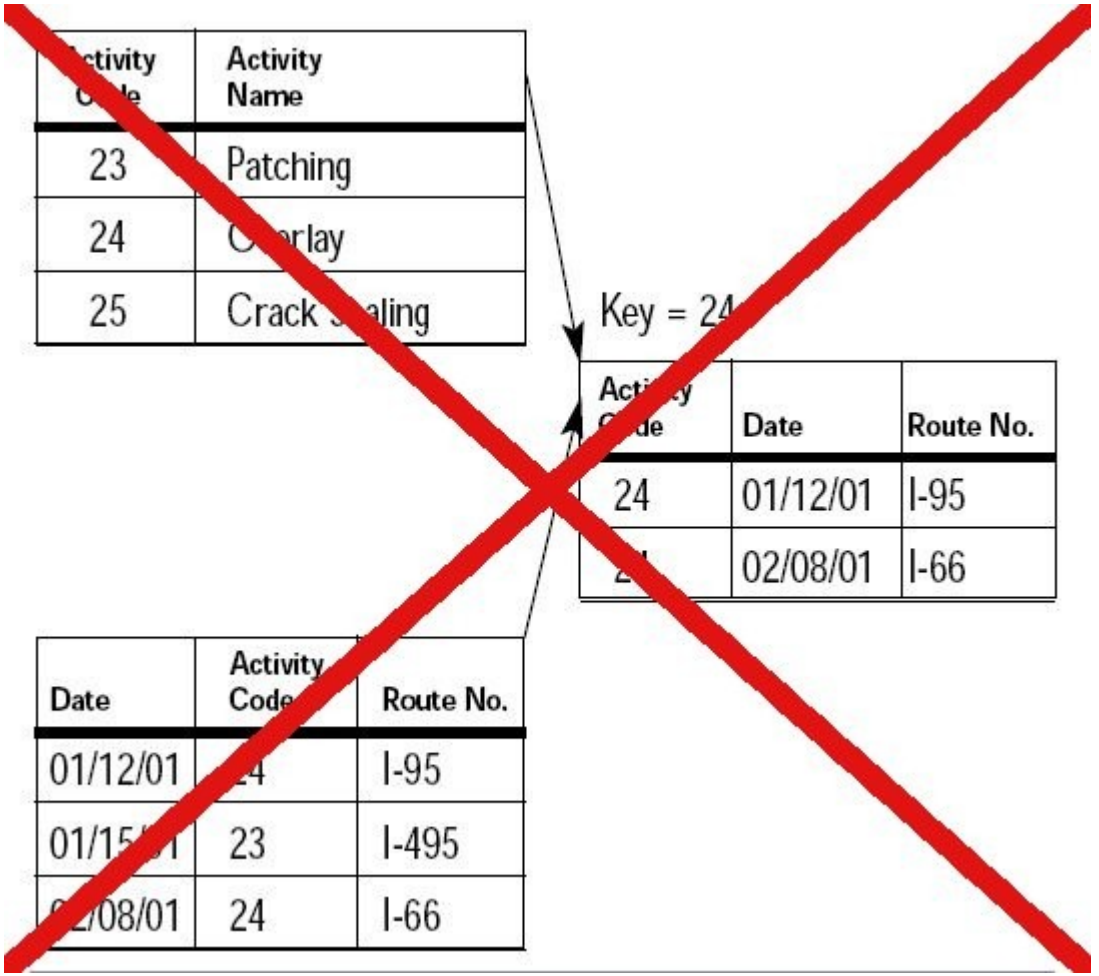
Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Data Model

Forget it!



Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

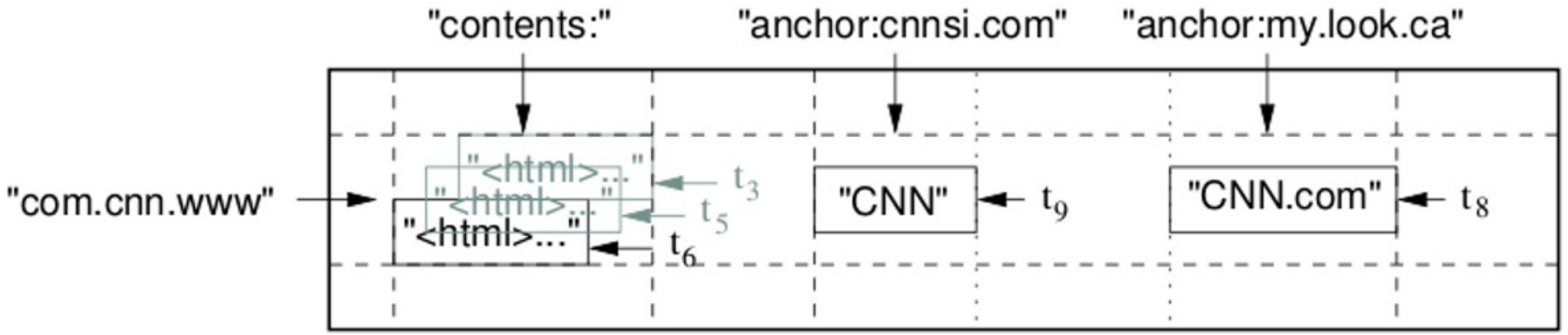
Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Data Model

Simpler than the Relational Model:
Dynamic control over data layout

Data Model



Indexed by: row key, column key, timestamp

Data Model

Data is maintained in lexicographic order by row key

- Allows (forces) developers to reason about the locality properties of their data
- Reads of short row ranges are efficient and require communication with a small number of machines

Data Model

Row range for a table dynamically partitioned

- Partitions are called **TABLETS**
- 1 GFS file per tablet
- Unit of distribution and load balancing

Data Model

- Reads/writes under a single row key are atomic
- Timestamps can be used to store multiple versions of the same item: garbage collection

Architecture

Architecture

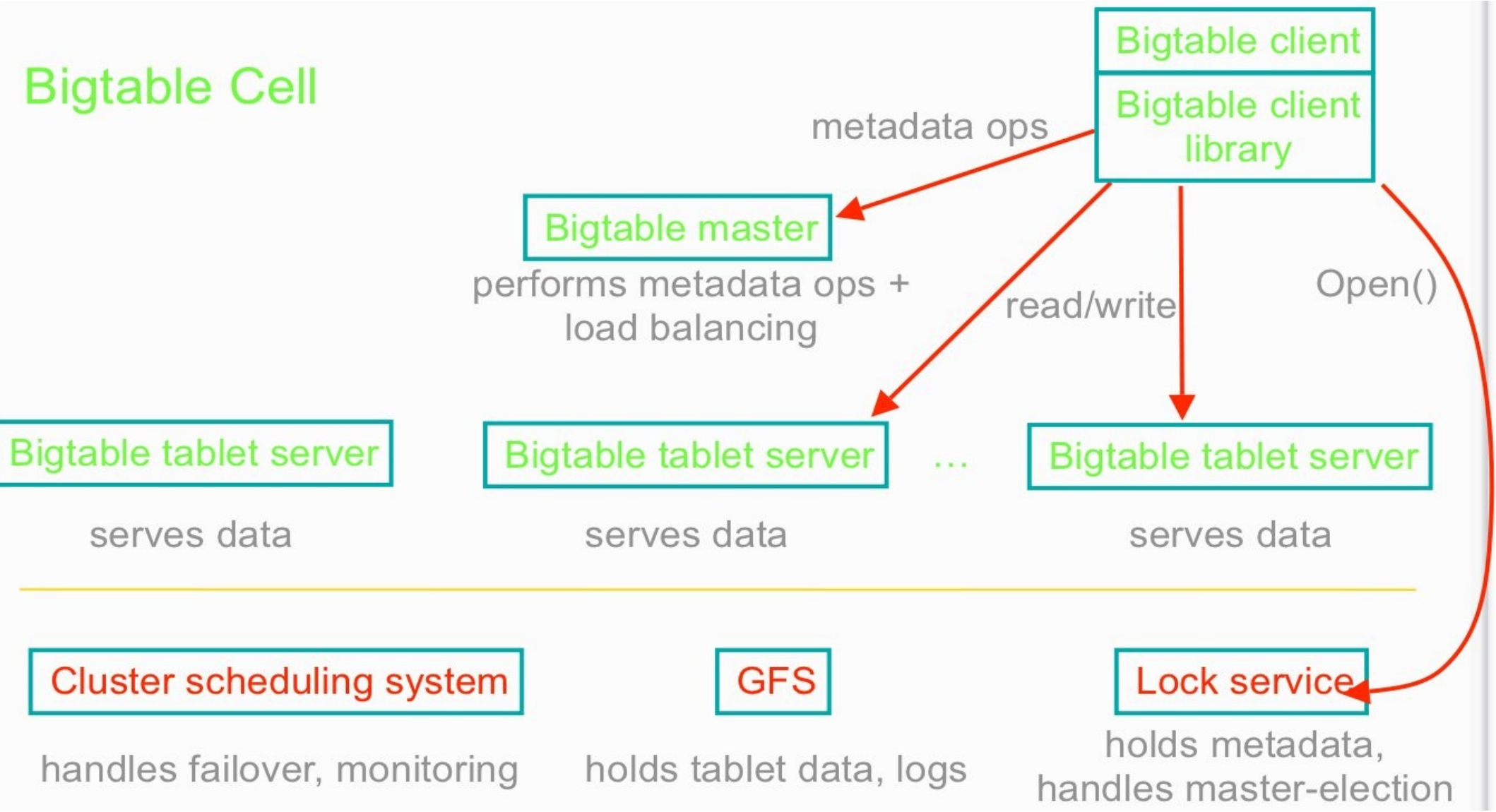
Building blocks:

- Google File System
- Cluster scheduling system
- Chubby: High available, persistent, distributed lock service

Architecture

1 master server,
N tablet servers

Architecture



Architecture

The tablet server

- Can be dynamically added or removed from a cluster according to changes in the workload
- Manages a set of N tablets (10 < N < 1000)
- Handles reads / writes to rows located in its tablets
- Splits tablets that have grown too large

Architecture

The master server

- Assigns tablets to tablets servers
- Detects when a tablet server joins / leaves
- Balances tablet ↔ server load

Architecture

The poor master is usually...
Quite bored.

Use Cases

Use Cases

- Web indexing
- Gmail
- Youtube
- Google Maps, Earth, Reader, Code
- …
- **Google App Engine**

Performance Evaluation

Performance Evaluation

Experimental Setup

- N tablet servers
- Huge GFS cell: 1786 machines, 2x 400 GB disks each

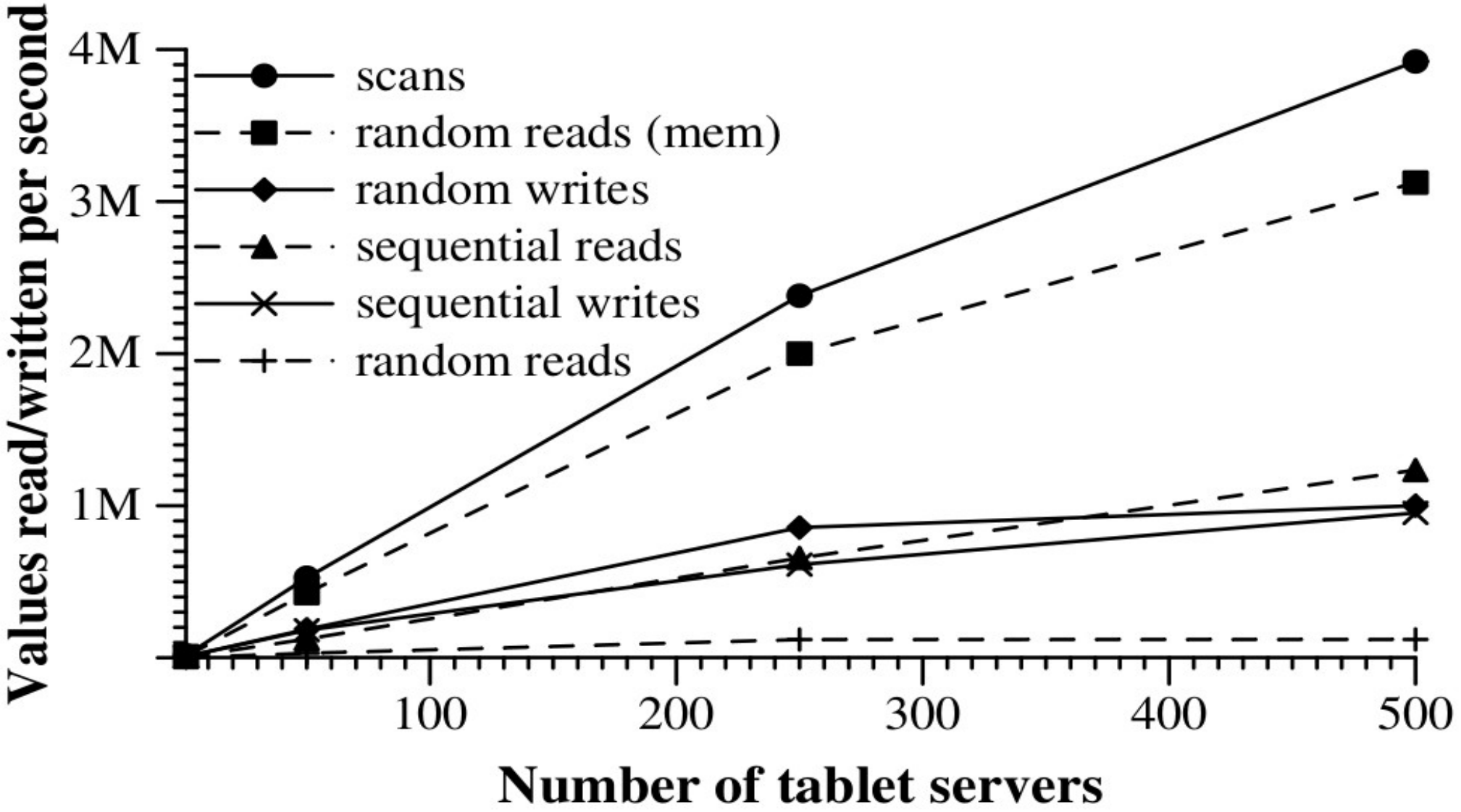
Performance Evaluation

Benchmarks

- Sequential write
- Sequential read
- Random write
- Random read
- Scan

Performance Evaluation

Bigger values are better



Performance Evaluation

- Scans are superfast: RPC overhead is amortized
- Random reads from memory also scale very well
- Random reads from GFS show the worst scaling

Why did they implement it?

Quoting Jeff Dean:

- Applications at Google place very different demands on the storage system
- Handle petabytes of data
- Scale to thousands of commodity servers
- Fun

Conclusions

Bigtable scales to petabytes of data across thousands of commodity Linux servers

Developers can have an hard time adapting to different models

Google's structured storage needs are satisfied

```
class Person(db.Expando):
    name = db.StringProperty()
    surname = db.StringProperty()

def person_example():
    ema = Person(name="Emanuele", surname="Rocca")
    ema.wears_glasses = True

    john = Person(name="John", surname="Smith")
    john.comes_from = "Malta"

    ema.put()
    john.put()

    for person in Person.all():
        print person.name, person.surname

    print Person.all().filter("wears_glasses", True).count(), "with glasses"
    print Person.all().filter("comes_from", "Malta").count(), "from Malta"
    print Person.all().filter("comes_from", "Italy").count(), "from Italy"


"feeds/models.py" 121L, 3066C written
```

```
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.

In [1]: import feeds.models

In [2]: feeds.models.person_example()
Emanuele Rocca
John Smith
1 with glasses
1 from Malta
0 from Italy
```