Bigtable: A Distributed Storage System for Structured Data

F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach,
M. Burrows, T Chandra, A. Fikes, R. E. Gruber

Presenter: Wei-Lun Huang

- Fay Chang et al. Bigtable: A Distributed Storage System for Structured Data

Outline

- Problem Formulation
- Data Model
- Building Blocks
- Implementation
- Evaluation

Outline

- Problem Formulation
- Data Model
- Building Blocks
- Implementation
- Evaluation

Google Projects

- Quantity
  - Petabytes of Data
  - Thousands of Machines

- Variety
  - Structured Data of Different Formats
  - Different Demands: throughput vs. latency

Goal: one distributed storage system

Expectations

- Scalability: more, more, and more tables/machines
- Applicability: a variety of Google projects as clients
- Performance: (concurrent) reads/writes from many clients
- Availability: crash failures, network partition, and more

Key Ideas: in my opinion

- Performance! <u>Performance!</u> **<u>Performance!</u>**
  - Structured Data: a weak assumption
  - Locality: old but classic

- Base the design on existing infrastructures!
  - It’s Google. Why reinventing the wheel?
  - (Much of the) Availability

Outline

- Problem Formulation
- Data Model
- Building Blocks
- Implementation
- Evaluation

Data Model: Row

- ❖ Atomic Read/Write
- ❖ Row Keys: sorted in lexicographic order
- ❖ Dynamic Partition by Row Range
  - ➢ One Table → Several Tablets
  - ➢ One Tablet ↔ One Row Range
  - ➢ Distribution & Load Balancing

Locality

Data Model: Column Family

- Columns of the Same Data Type

- Column Key = Family:Qualifier

- At the column-family level, …
  - Access Control
  - Memory/Disk Accounting
  - **Locality** Group + Compression

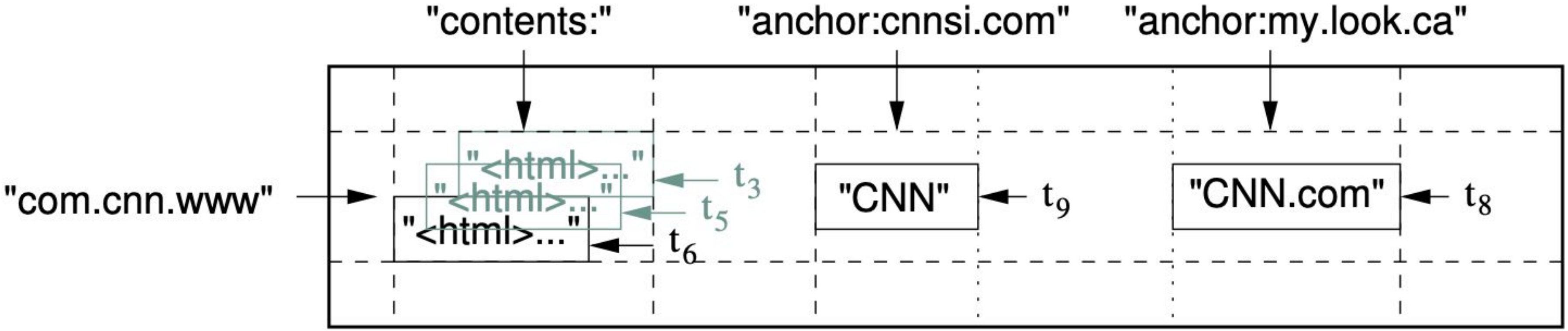
Data Model: Timestamp

- One Cell/Data, Multiple Versions
  - The Latest Version First

- From Bigtable? From client applications?
  - Real Time in μs vs. Customized Collision Avoidance

- Garbage Collection: the last *n* copies vs. in the last *m* days

Example: Webtable



- Fay Chang et al. Bigtable: A Distributed Storage System for Structured Data

Outline

- Problem Formulation
- Data Model
- Building Blocks
- Implementation
- Evaluation

API

- {Creation, Deletion} of {Table, Column Family}
- Metadata Change: access control rights
- Row Read, Value Write/Deletion, Column Family Iteration, …
- Single-Row Transaction: read-modify-write
- Execution of Client-Supplied Scripts
    - No writes back into Bigtable!

Google Infrastructures

Availability

- Google File System (GFS): persistent log/data storage
- Cluster Management System
  - What if other distributed applications on the same machines?
  - Job Scheduling + Resource Management
- Sorted String Table (SSTable): “data” file format in GFS

Google Infrastructures: Chubby

Availability

- Paxos-Based Distributed Lock Service
  - Directory/File as a Lock
  - Atomic File Read/Write + Consistent Client-Side Caching

- One Client, One Session with Chubby
  - Session Expiration → All Locks + Open Handles Lost

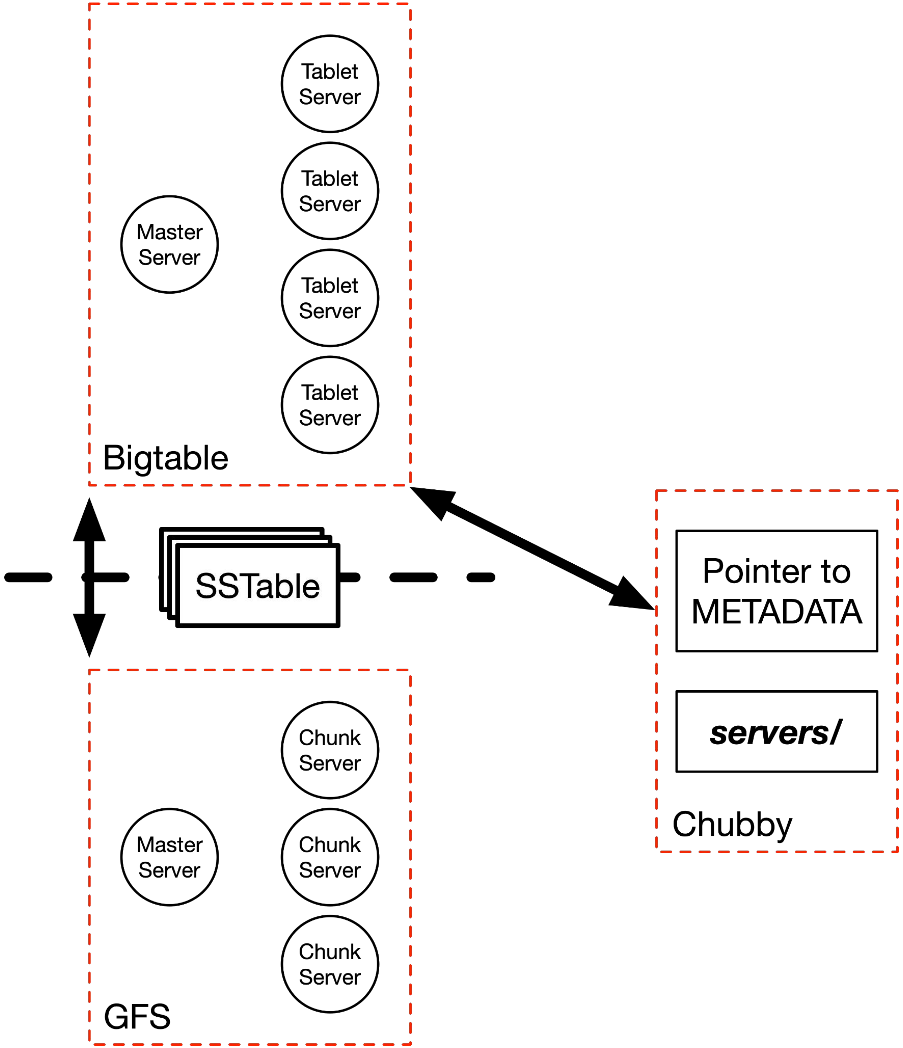
- Bigtable relies heavily on Chubby.
  - Chubby Unavailable → Bigtable Unavailable
  - Bigtable Debugging → Cubby Debugging

Chubby in Bigtable

- Always ≤1 Active Master Server

- Tablet Server Existence/Death

- Metadata Storage
  - Access Control
  - Column Family
  - Bootstrap Location of Bigtable Data


Tablet Server
Tablet Server
Master Server
Tablet Server
Tablet Server
Bigtable
SSTable
Pointer to METADATA
servers/
Chubby
Chunk Server
Master Server
Chunk Server
Chunk Server
GFS

Outline

- ❖ Problem Formulation
- ❖ Data Model
- ❖ Building Blocks
- ❖ Implementation
- ❖ Evaluation

Three Components

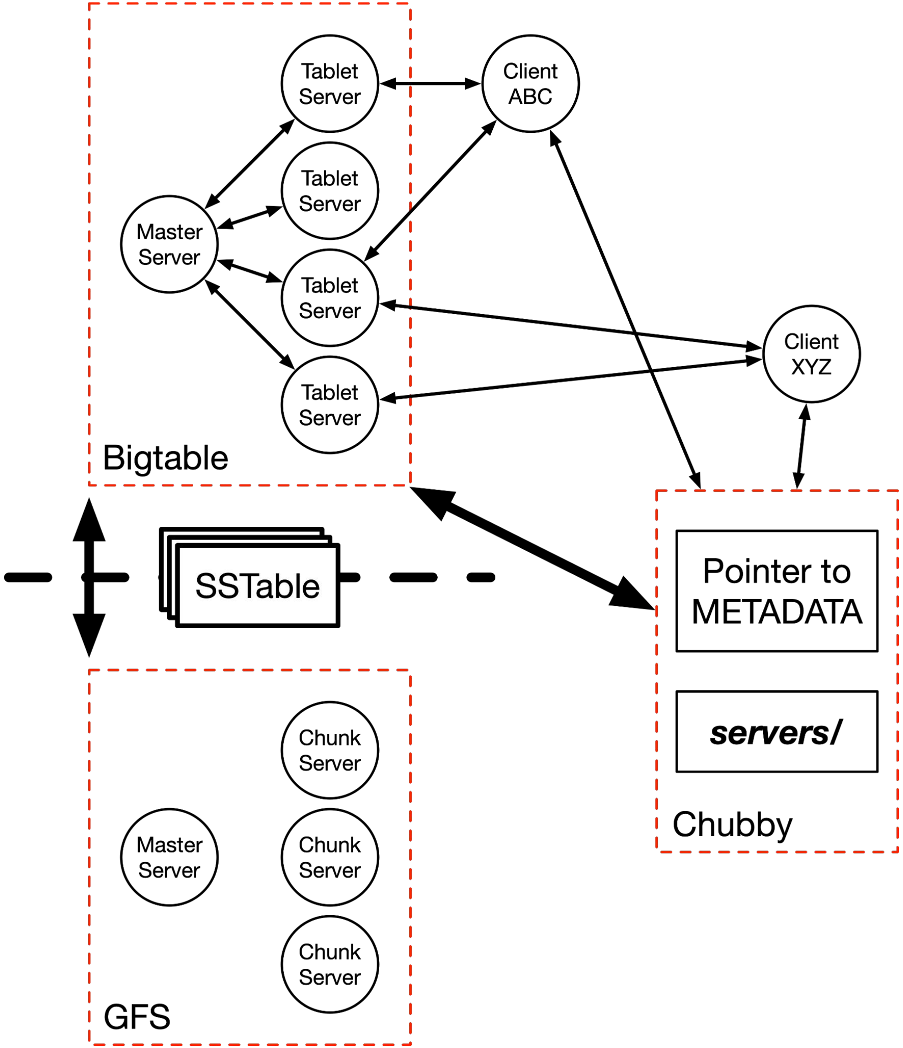
- One Master Server
  - Tablet Assignment: to tablet servers V
  - Addition/Expiration Detection: of tablet servers
  - Load Balancing: for tablet servers

- Many Tablet Servers: dynamic addition/removal
  - Tablet Serving: reads/writes from clients V

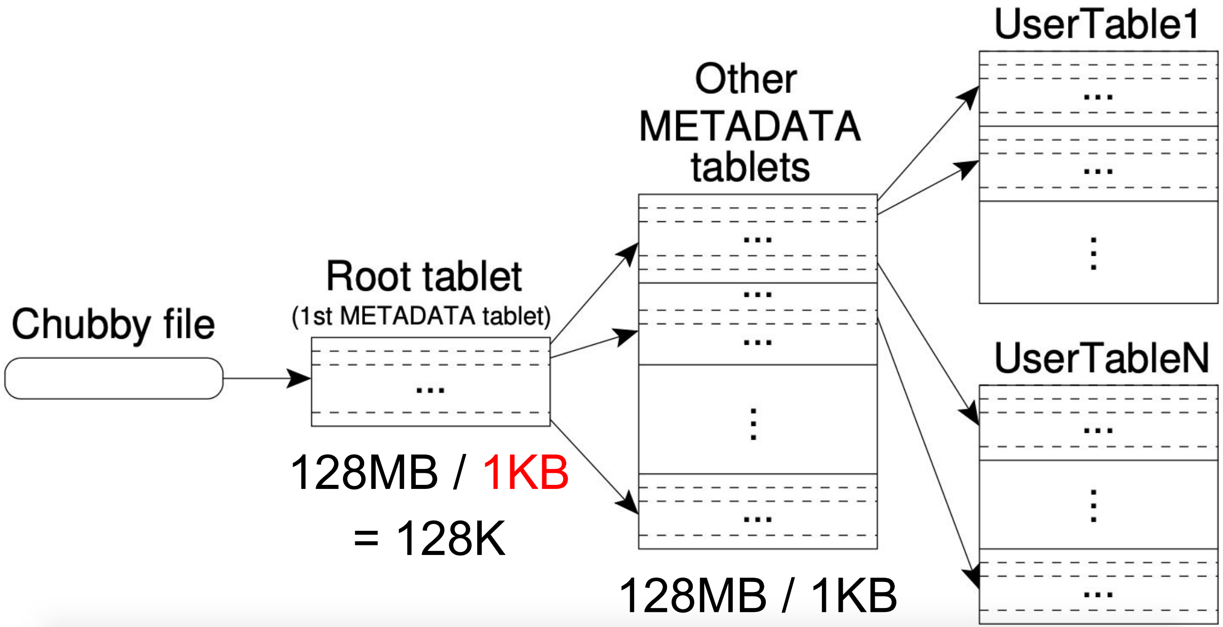
- Client-Side Library
  - Tablet Location V

Three Components: more tasks

- ❖ One Master Server
  - ➢ Schema Change: table/column family creation
  - ➢ GFS Garbage Collection
- ❖ Many Tablet Servers: dynamic addition/removal
  - ➢ Tablet Split: if a tablet >100~200 MB
- ❖ Client-Side Library


Tablet Server
Client ABC
Tablet Server
Master Server
Tablet Server
Client XYZ
Tablet Server
Bigtable
SSTable
Pointer to METADATA
servers/
Chubby
Chunk Server
Master Server
Chunk Server
Chunk Server
GFS

Tablet Location: three-level hierarchy



128MB / 1KB
= 128K

128MB / 1KB
= 128K

128K x 128K
= 16G (128MB Tablet)

- Fay Chang et al. Bigtable: A Distributed Storage System for Structured Data

Tablet Location

- ❖ METADATA: a special Bigtable
  - ➢ One Row = One Tablet’s Location
  - ➢ Row Key = Table Identifier + End Row
  - ➢ No Root-Tablet Split (∵ Three-Level Hierarchy)

- ❖ Caching by Client-Side Library
  - ➢ Unknown/Incorrect Location → Recursive Move-Up
  - ➢ Empty Cache? 3 Network Round-Trips
  - ➢ Stale Cache? ≤6 Network Round-Trips

Tablet Assignment

- Tablet Server ↔ Chubby File: ***servers****/unique_file_name*
  - Start → File Creation + Exclusive Lock Acquisition

  - Stop Serving ← Exclusive Lock Lost
  - Recover → Exclusive Lock Reacquisition
  - Kill Itself ← File Deleted

  - Terminate → Exclusive Lock Release

Tablet Assignment

- Master Server should know …
  - Live Tablet Servers
  - Unassigned Tablets
  - Assigned Tablet vs. Tablet Server
  - Unassigned Tablet vs. Available Tablet Server

- How? Ask Chubby + Tablet Servers!
  - Hi Chubby, any news in ***servers/***?
  - Hi Tablet Server, still own the lock?

- New Master? Ask Chubby + Tablet Servers + METADATA!

Tablet Serving

- Persistent Tablet Contents: SSTables @GFS
  - Location Info ← METADATA
  - Commit Logs @GFS ← Redo Points ← METADATA

- Latest Updates: memtable @memory + Commit Log @GFS

- How to Write? Read? Tablet Recovery?

Compactions

- Minor Compaction
  - Current memtable x1 as New SSTable x1
  - Memory Usage ↓ + Reads from Commit Log ↓

- Major Compaction
  - (SSTable xN + memtable x1) as New SSTable x1
  - Read Complexity ↓ (∵ No Changes/Deletions)
  - Security ↑ (∵ Timely Deletion)

Refinements

- Locality Group x1 -- Relevant Column Family xN -- SSTable x1

- Fast Two-Pass Per-SSTable Compression
  - Window: Large (long common strings) vs. Small (repetitions)
  - Compression Ratio? Row Locality!

- Two-Level Caching for Read
  - Key-Value Pairs
  - SSTable Blocks

Refinements

- Bloom Filter for <Row, Col> Existence in SSTable

- Per-Tablet-Server Commit Log
  - Tablet Recovery? Commit Log Sorting First!
  - GFS Issues? Log Writer Threads x2!

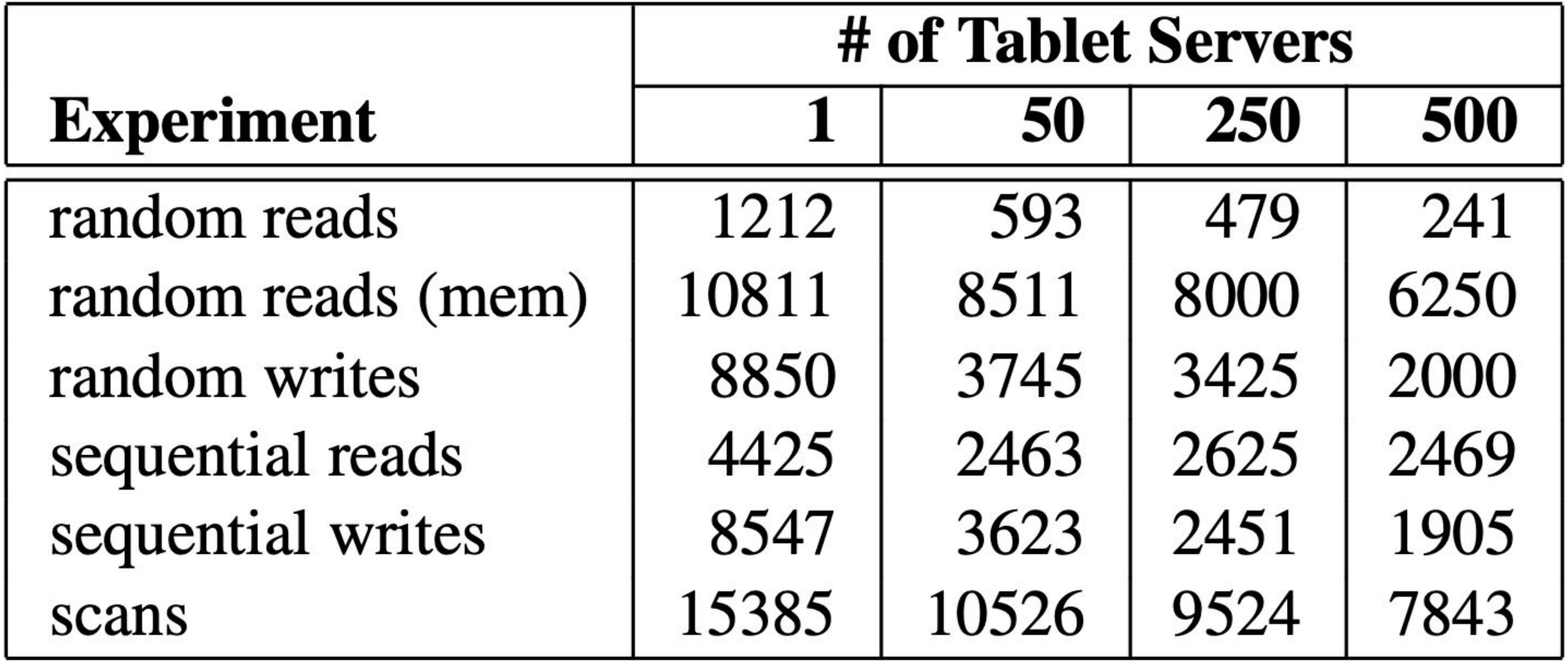
- SSTable Immutability: tablet split, concurrent read/write, etc.

- Minor Compactions before Tablet Transfer

Outline

- Problem Formulation
- Data Model
- Building Blocks
- Implementation
- Evaluation

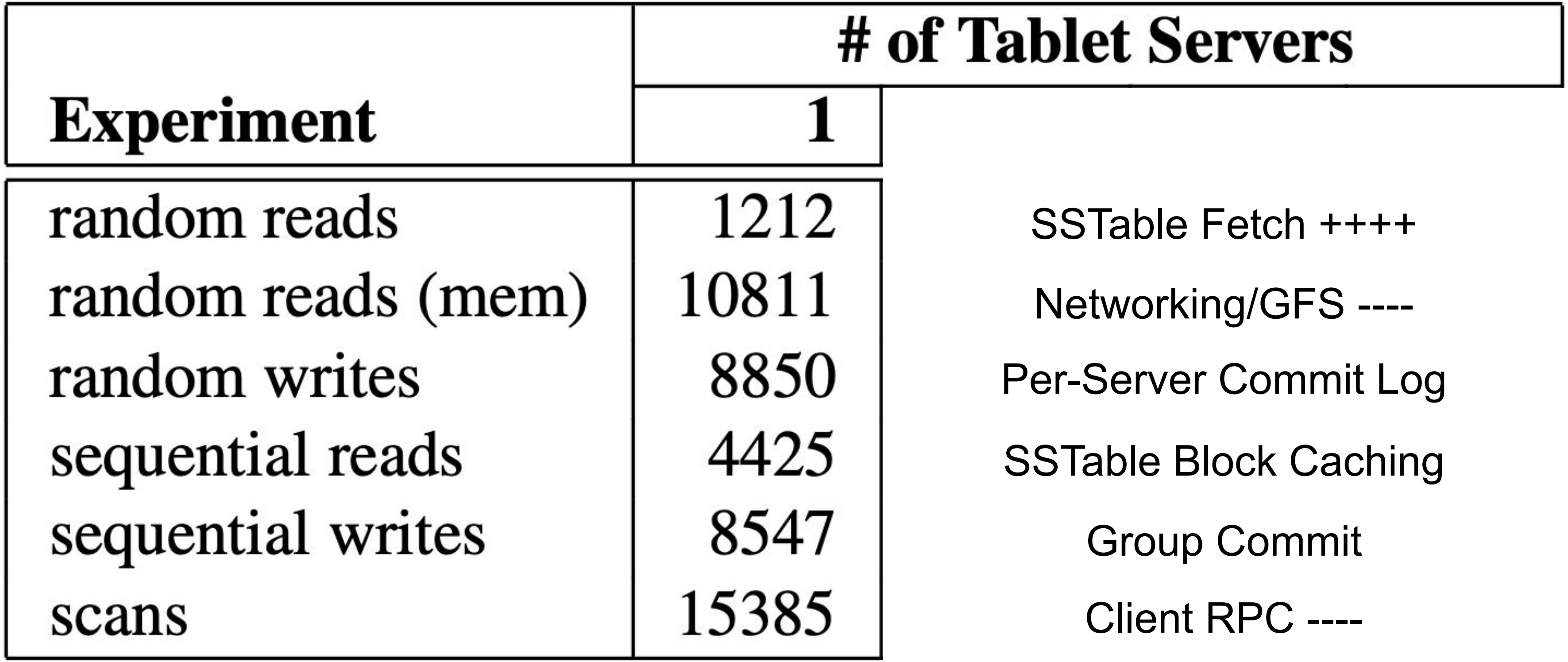
Per-Server #Read/#Write



	# of Tablet Servers			
Experiment	**1**	**50**	**250**	**500**
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

- Fay Chang et al. Bigtable: A Distributed Storage System for Structured Data

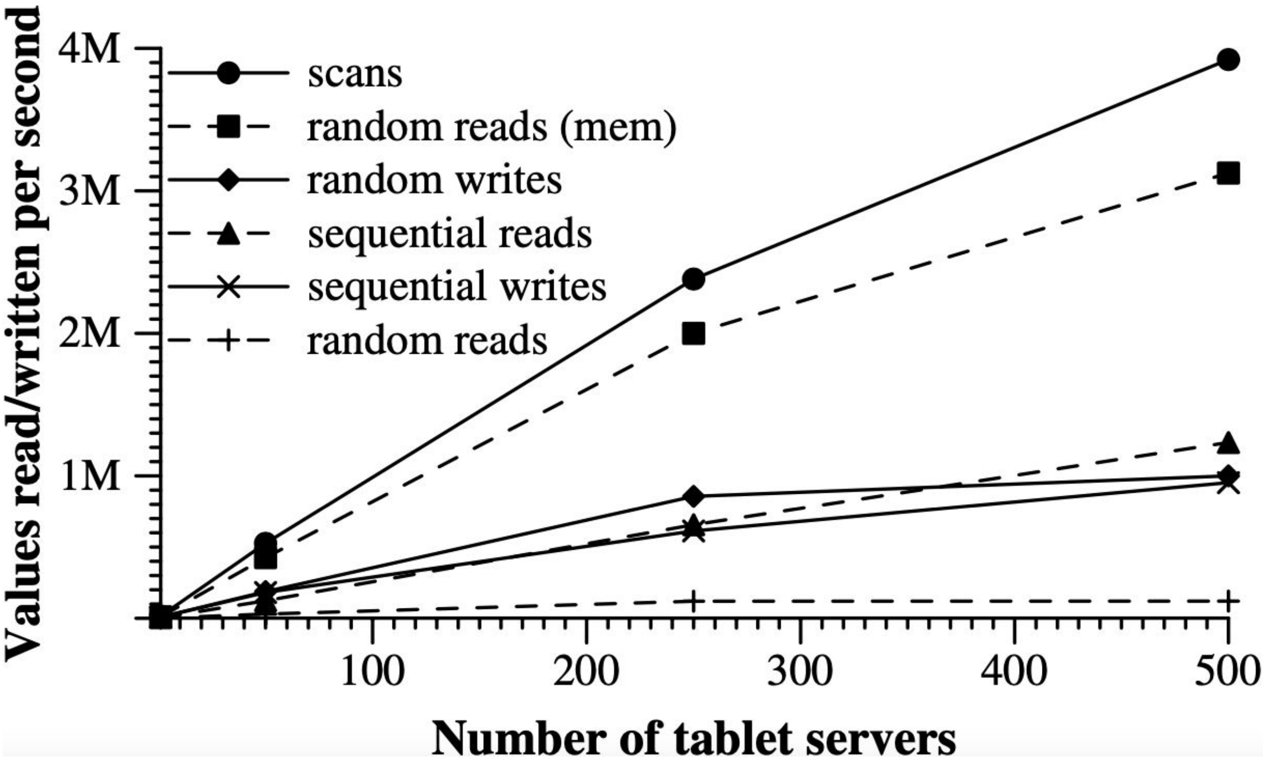
Single-Server #Read/#Write



	# of Tablet Servers	
Experiment	**1**	
random reads	1212	SSTable Fetch ++++
random reads (mem)	10811	Networking/GFS ----
random writes	8850	Per-Server Commit Log
sequential reads	4425	SSTable Block Caching
sequential writes	8547	Group Commit
scans	15385	Client RPC ----

- Fay Chang et al. Bigtable: A Distributed Storage System for Structured Data

Aggregate #Read/#Write



CPU-Intensive

Load Imbalance

Network Saturation

- Fay Chang et al. Bigtable: A Distributed Storage System for Structured Data

Summary

- Distributed Storage of Structured Data
  - Locality for Performance: data model, refinements, etc.
  - Google Infrastructure for Availability + Reliability

- Tablet Location/Assignment/Serving + Minor/Major Compaction
  - Chubby for METADATA + Tablet Server Existence
  - GFS for Persistent Storage of Commit Log + Tablets (SSTables)

- Bigtable/GFS: latency/throughput, table/file, write/append, Chubby/lease
- Bigtable as MapReduce Input/Output: Google Analytics/Earth/PSearch

Backup Slides

Google Infrastructures

Availability

- Google File System (GFS): persistent log/data storage

- Cluster Management System
  - What if other distributed applications on the same machines?
  - Job Scheduling + Resource Management + Fault Tolerance

- Sorted String Table (SSTable): data file format
  - Block Index: in-memory when SSTable opened
  - Block Sequence: block-index binary search → disk seek x1