CSE 444: Database Internals

Lectures 26

NoSQL: Extensible Record Stores

References

- **Scalable SQL and NoSQL Data Stores**, Rick Cattell, SIGMOD Record, December 2010 (Vol. 39, No. 4)

- **Bigtable: A Distributed Storage System for Structured Data.** Fay Chang et. al. OSDI 2006.

- Online documentation: HBase

What is Bigtable?

- Distributed storage system
- Designed to
  - Hold **structured** data
  - Scale to thousands of servers
  - Store up to several hundred TB (maybe even PB)
  - Perform backend bulk processing
  - Perform real-time data serving
- To scale, Bigtable has a **limited set of features**

Bigtable Data Model

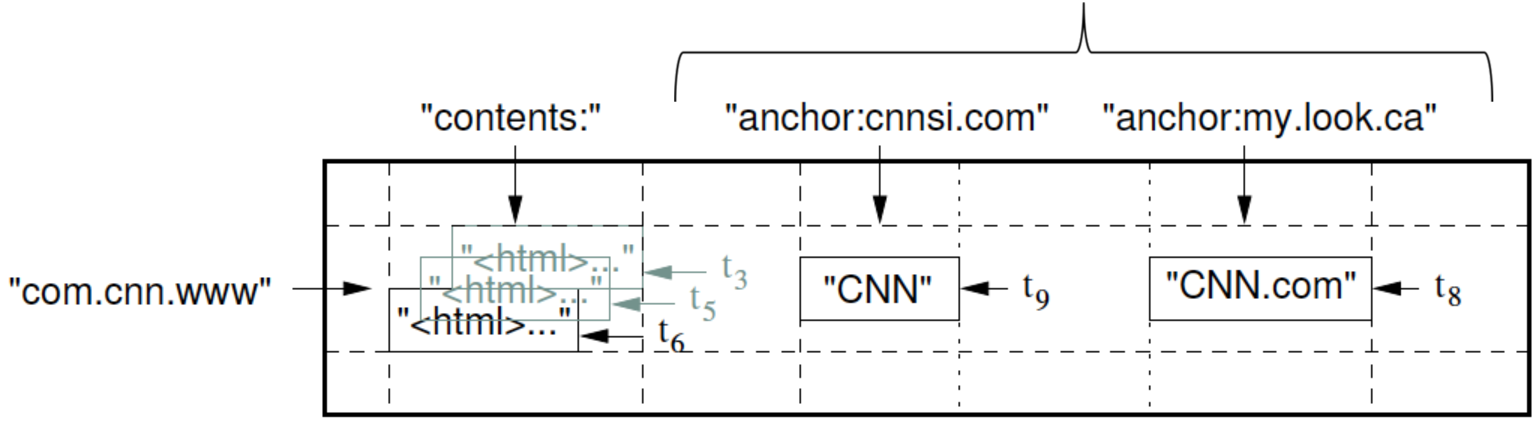
- Sparse, multidimensional sorted map

  `(row:string, column:string, time:int64)` ➔ `string`

  Notice how everything but time is a string

- Example from Fig 1:

Columns are grouped into families



Key Features

- Read/writes of data under single row key is atomic
  - Only single-row transactions!
- Data is stored in lexicographical order
  - Improves data access locality
  - Horizontally partitioned into *tablets*
  - Tablets are unit of distribution and load balancing
- Column families are unit of access control
- Data is versioned (old versions garbage collected)
  - Ex: most recent three crawls of each page, with times

Outline

- Bigtable API
- Bigtable architecture
- Bigtable performance and discussion

API

- Data definition
  - Creating/deleting tables or column families
  - Changing access control rights
- Data manipulation
  - Writing or deleting values
  - Looking up values from individual rows
  - Iterate over subset of data in the table
- Bigtable can serve as input/output for MapReduce

Outline

- Bigtable API
- Bigtable architecture
- Bigtable performance and discussion

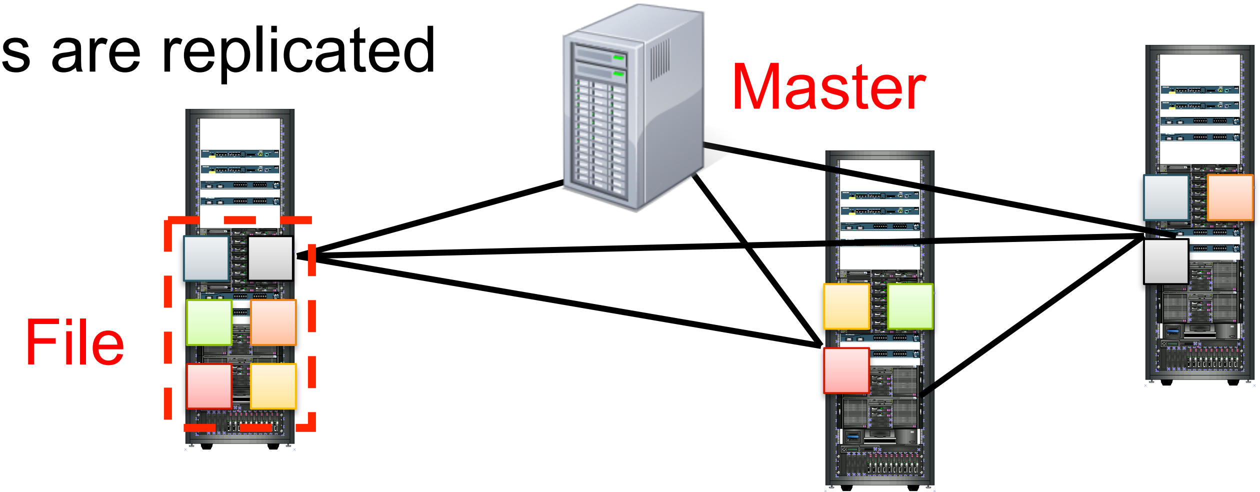
Chubby Lock Service

- In a distributed system, agreement is a problem
  - Different failure scenarios are possible
  - Nodes can have inconsistent views of who is up and who is down
  - Messages can arrive out-of-order at different nodes
- But need agreement to make decisions
- Chubby
  - Provides black-box agreement service through lock abstraction
  - Uses the well-known Paxos algorithm

Google File System

- A file = A series of chunks
  - Size of a chunk ≥ 64MB
  - Append & read only
- Fault-tolerance
  - Chunks are distributed
  - Chunks are replicated

- Master node
  - Decides chunk placement
  - Decides replica placement
  - Tells clients where to find data

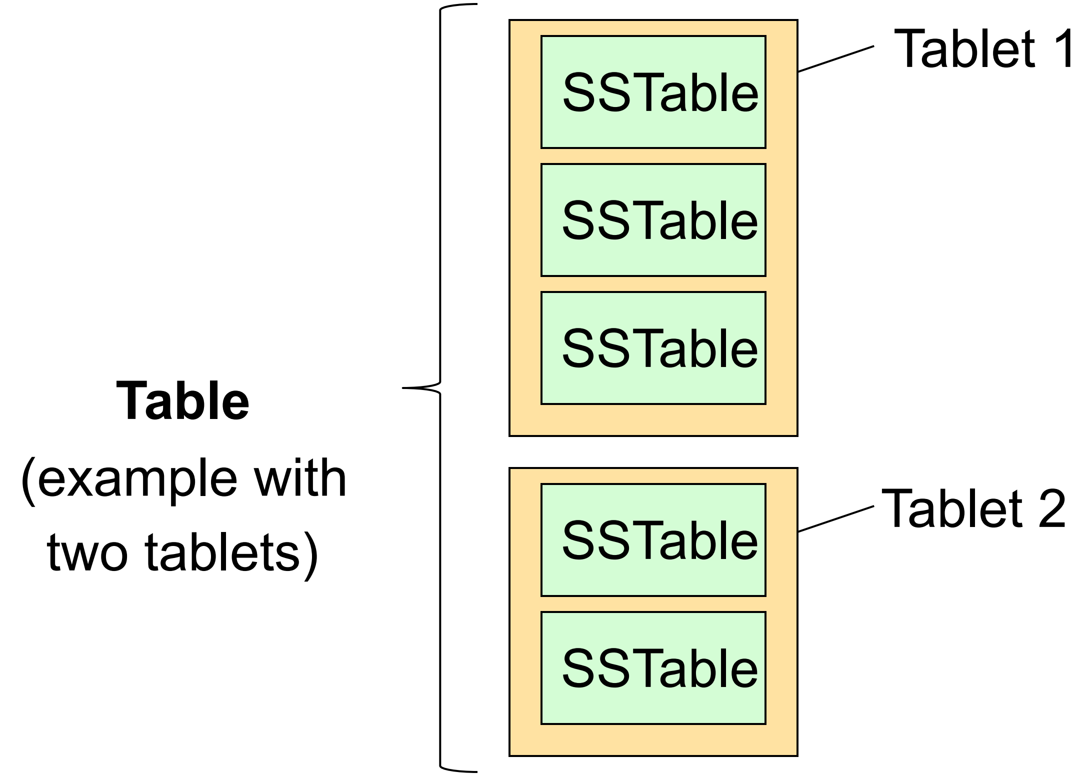


A Table in Bigtable: Basics

A table consists of a set of tablets: Section 5.3

Each tablet stores a range of the table

Each tablet comprises one or more SSTables



Each SSTable is stored in a GFS file

Initially, a tablet has only one SSTable

Updates add SSTables

Major compaction puts everything back into one SSTable

SSTable Details

- Persistent map from keys to values
  - Ordered
  - Immutable
  - Keys and values are strings
- API
  - Look up value associated with a key
  - Iterate over all key/value pairs in given range
- Implementation
  - Sequence of blocks
  - One block index to locate other blocks

SSTable Details

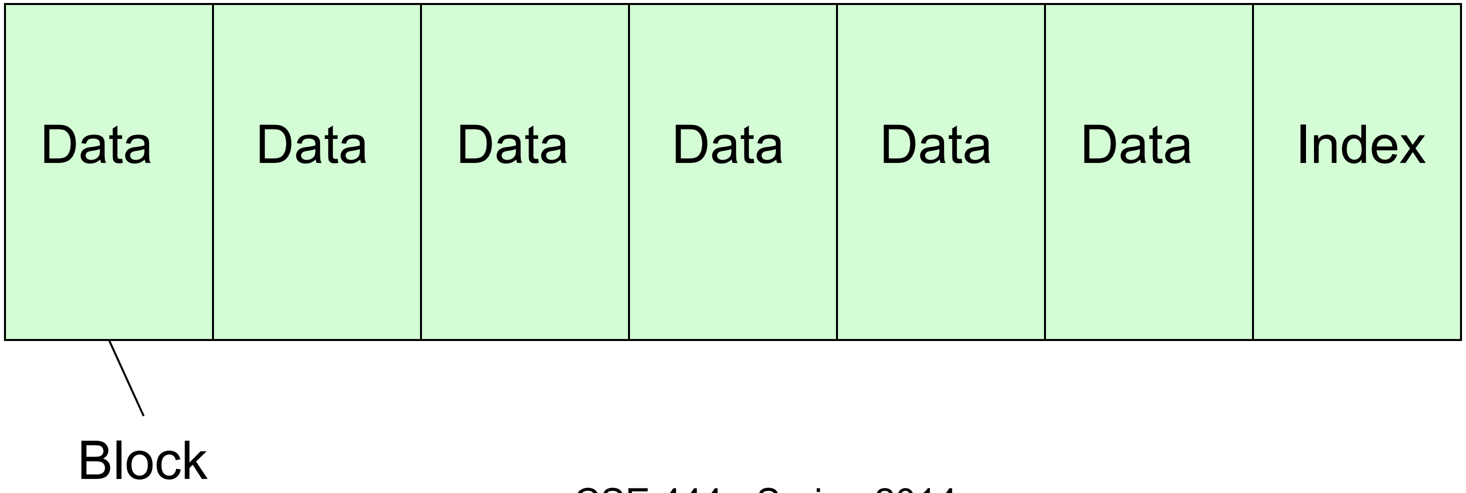
SSTable is a sequence of blocks

Last block is the index to locate other blocks

Index is loaded into memory when SSTable is open

Optionally, whole SSTable can be memory mapped

SSTable is a GFS File



Data

Data

Data

Data

Data

Data

Index

Block

Lookup in SSTable

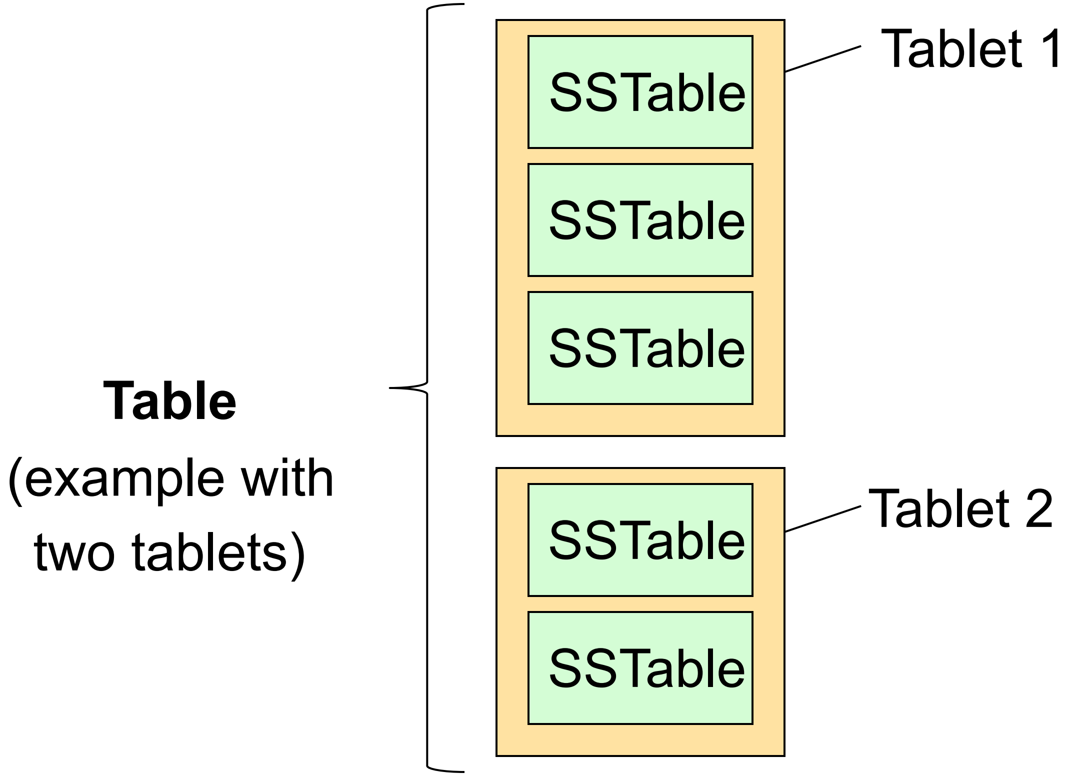
- Read index block of SSTable
- Binary search on index block to find data block
- Read data block

A Table in Bigtable: Basics

A table consists of a set of tablets: Section 5.3

Each tablet stores a range of the table

Each tablet comprises one or more SSTables



Each SSTable is stored in a GFS file

Initially, a tablet has only one SSTable

Updates add SSTables

Major compaction puts everything back into one SSTable

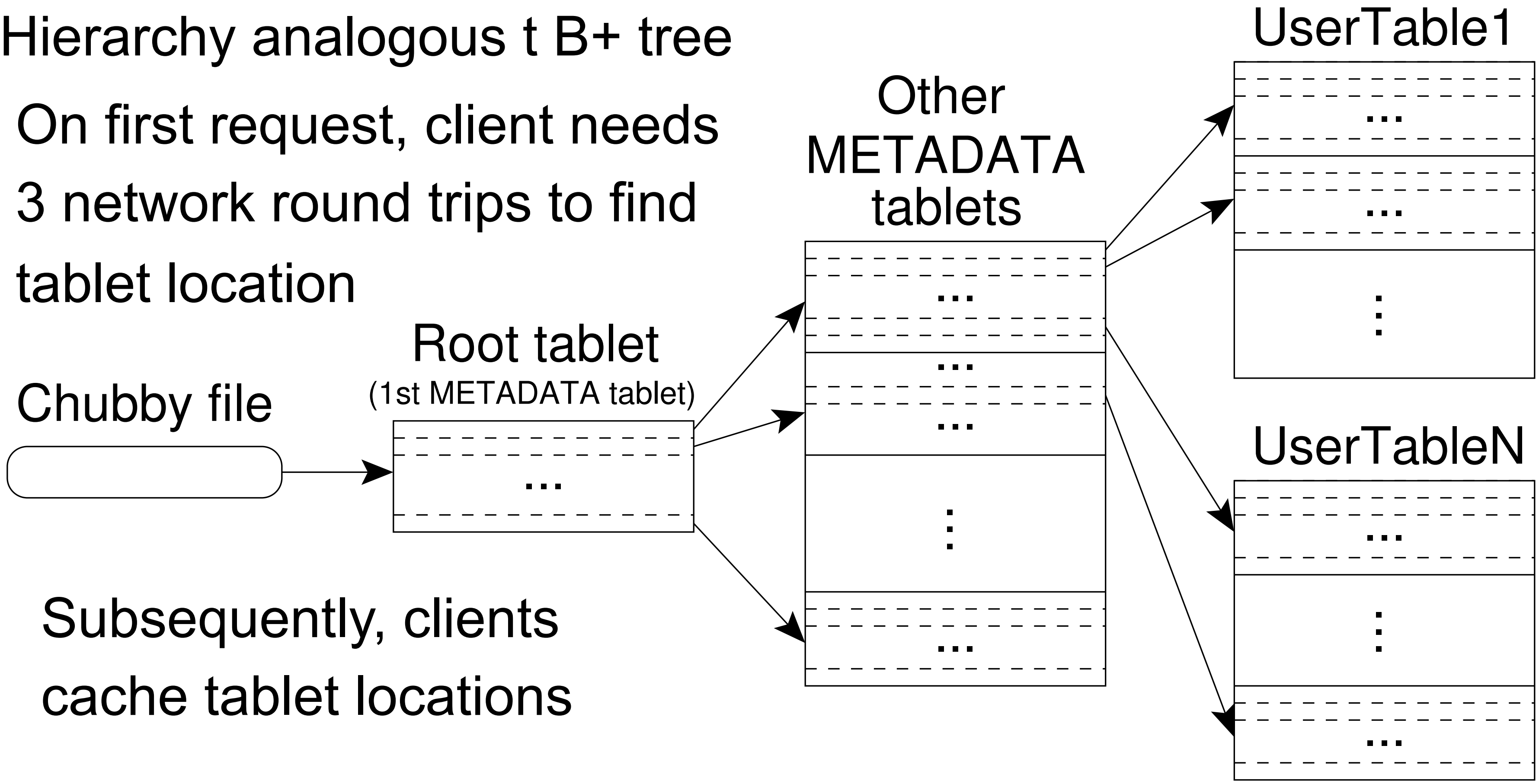
BigTable Components

- A library linked into every client
- One master server
  - Assigns tablets to tablet servers
  - Ensures load balance between tablet servers
  - Detects when tablet servers come and go
  - Handles schema changes
- Many tablet servers (can be added/removed)
  - Each server manages a set of tablets (10 to 1K)
  - Loads tablets into memory
  - Handles read and write requests
  - Splits tablets that have grown too large

Finding Tablet Servers

Hierarchy analogous t B+ tree

On first request, client needs 3 network round trips to find tablet location



Subsequently, clients cache tablet locations

Read Operation on Table

- Assuming simple case of 1 tablet = 1 SSTable

- Find location of appropriate tablet
  - Find appropriate tablet in the table and its location
    - Use tablet location hierarchy from previous slide
    - Metadata for a tablet contains list of SSTables
  - Then read data from the SSTable

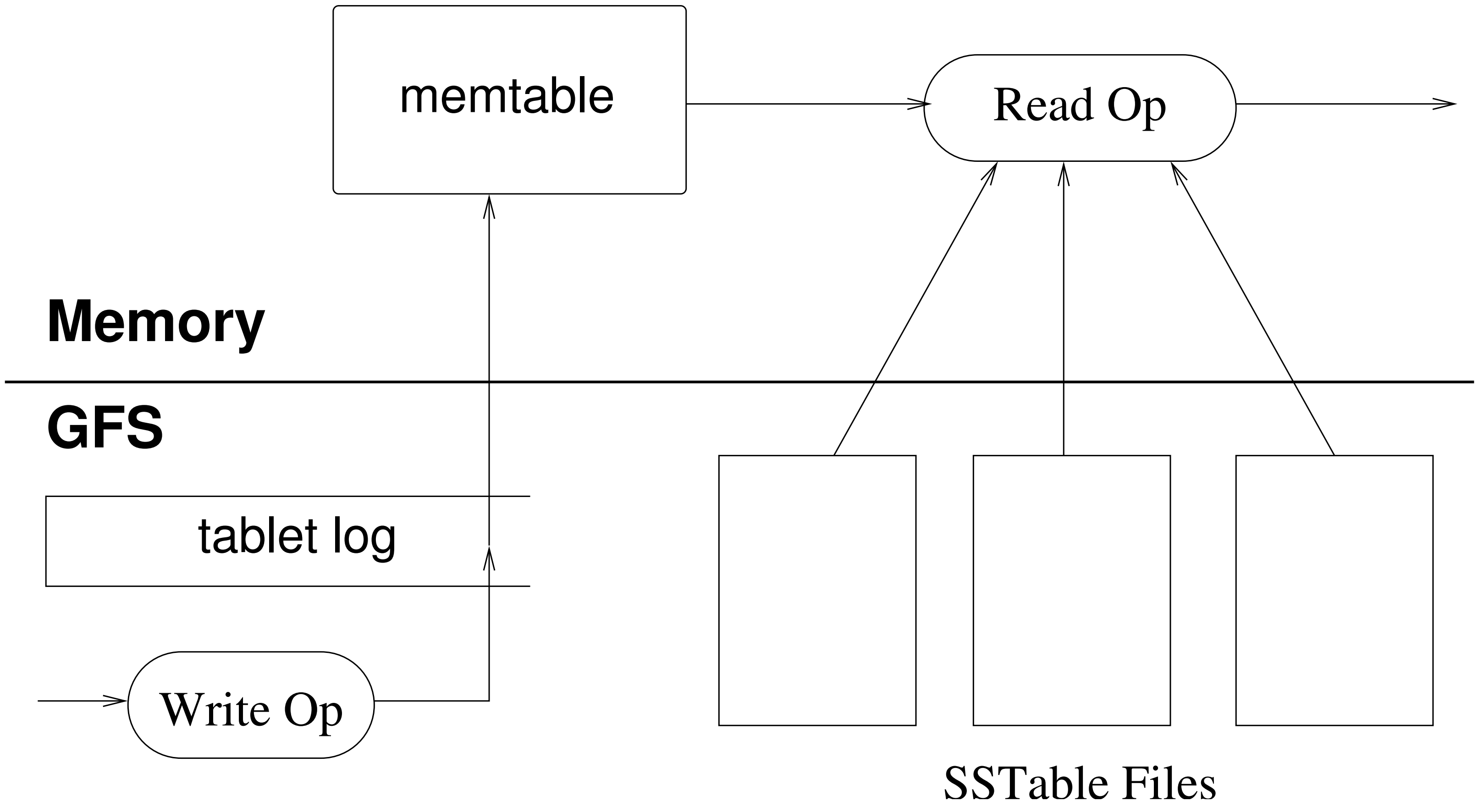
Assigning Tablets to Tablet Servers

- Problem
  - Need to balance load for serving read/write requests
  - Want to avoid Chubby file and root tablet being hot-spots
- Solution
  - Master
    - Assigns tablets to tablet servers
    - Manages tablet server churn and load imbalances
    - Processes schema changes
  - Tablet server
    - Loads tablets into memory (i.e., loads index blocks of SSTables)
    - Handles read/write to tablets that it has loaded
    - Splits large tablets
  - Clients cache tablets locations

Writing to Tablets

- Remember: SSTables are immutable
- When a write operation arrives at a tablet server:
  - Write mutation to a separate commit log stored in GFS
  - Wait until done
  - Insert the mutation into in-memory buffer: *memtable*
    - The memtable is sorted lexicographically
- To serve reads, the tablet server
  - Merges SSTables and memtable into a single view

Tablet Representation



Loading Tablets

- To load a tablet, a tablet server does the following

- Finds location of tablet through its METADATA (Fig. 4)
  - Metadata for tablet includes list of SSTables and set of redo points

- Read SSTables index blocks into memory
  - Recall an SSTable consists of a set of blocks + 1 index block

- Read the commit log since redo point and reconstructs the memtable (the METADATA includes the redo point)

Compaction

- To keep memtables below a threshold
- Minor compaction: convert memtable into SSTable
- Merging compaction:
  - Read a few SSTables and the memtable
  - Write out a new SSTable
- Major compaction:
  - Replace all SSTables and memtable with a new SSTable

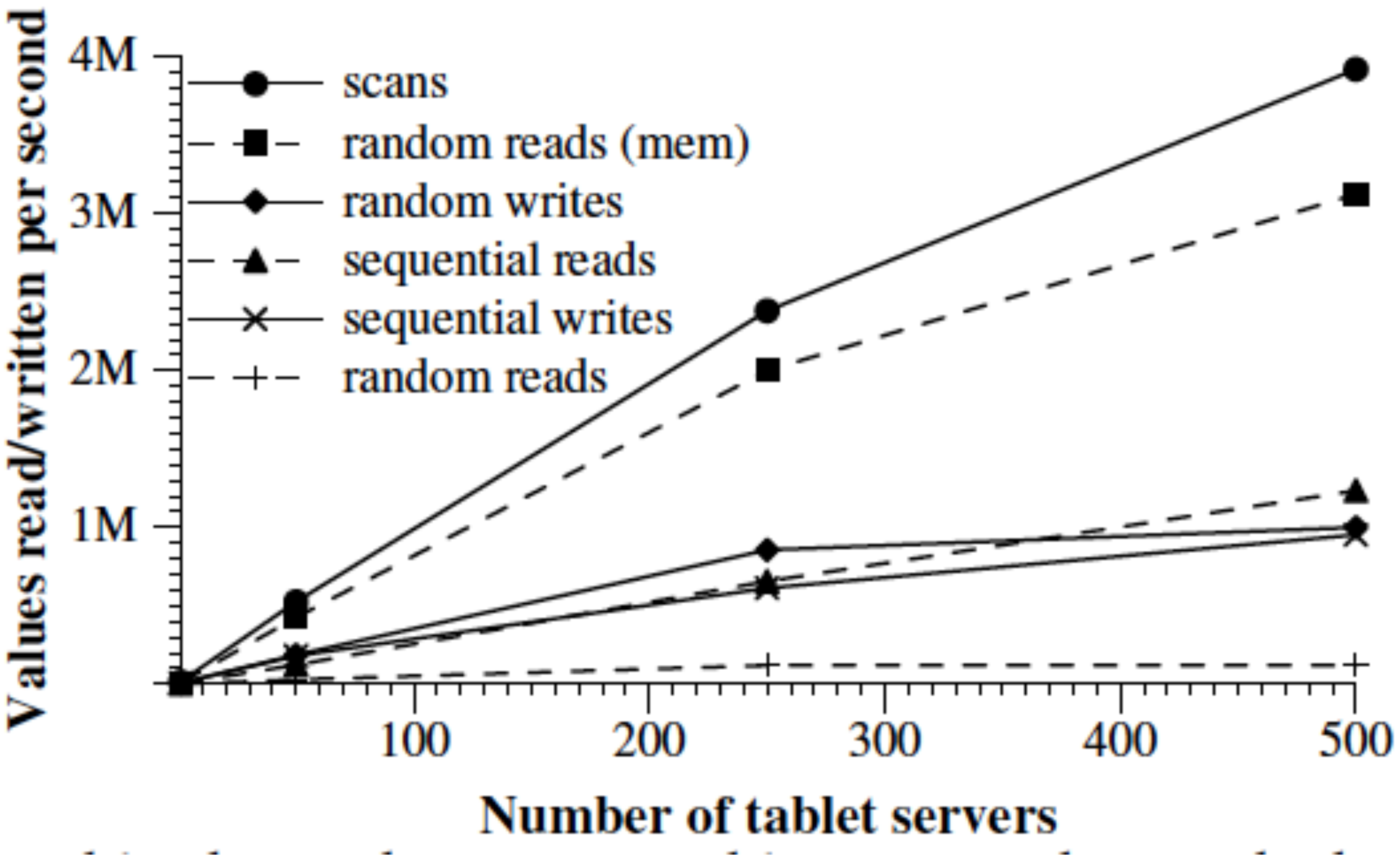
Optimizations

- Vertical partitioning: locality groups
- Compression of SSTable blocks
- Caching of SSTable data
- Additional indexing: bloom filters
  - Avoid reading SSTable that does not have needed data
- Commit log optimizations
  - Single commit log per tablet server
- Tablet migration optimization

Outline

- Bigtable API
- Bigtable architecture
- Bigtable performance and discussion

Performance



Summary

- Bigtable is a distributed system for storing structured data
- Provides high performance and high availability
- Scales incrementally
- Restricted functionality
- Widely used by many applications at Google

Next Steps

Try HBase

http://hbase.apache.org/