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

Google, Inc.

OSDI '06

August 24, 2011

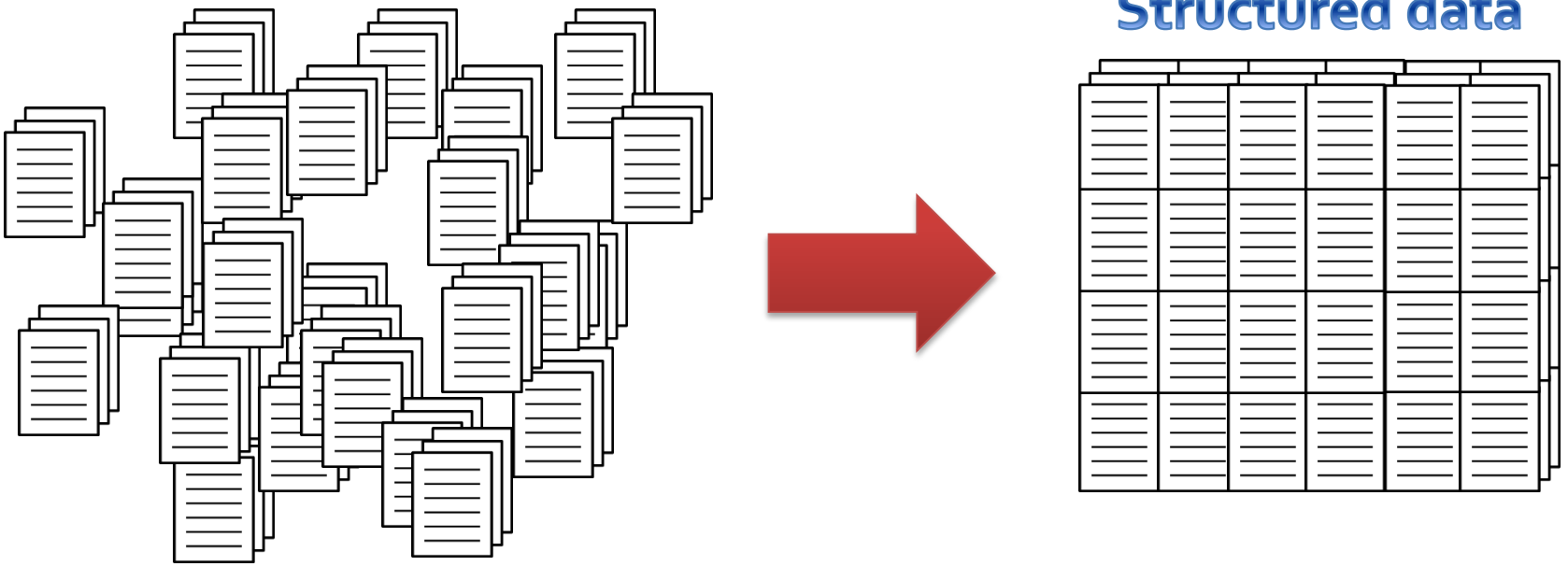
Hye Chan Bae

Contents

- Introduction
- Data Model
- Implementation
- Refinements
- Evaluation
- Conclusions

Managing structured data

- GFS(Google File System) for tremendous data



Managing structured data

We need a storage system
like database
in
Distributed Environment!!

Bigtable

Bigtable

- A distributed storage system
  - Wide applicability
  - Scalability
  - High performance
  - High availability

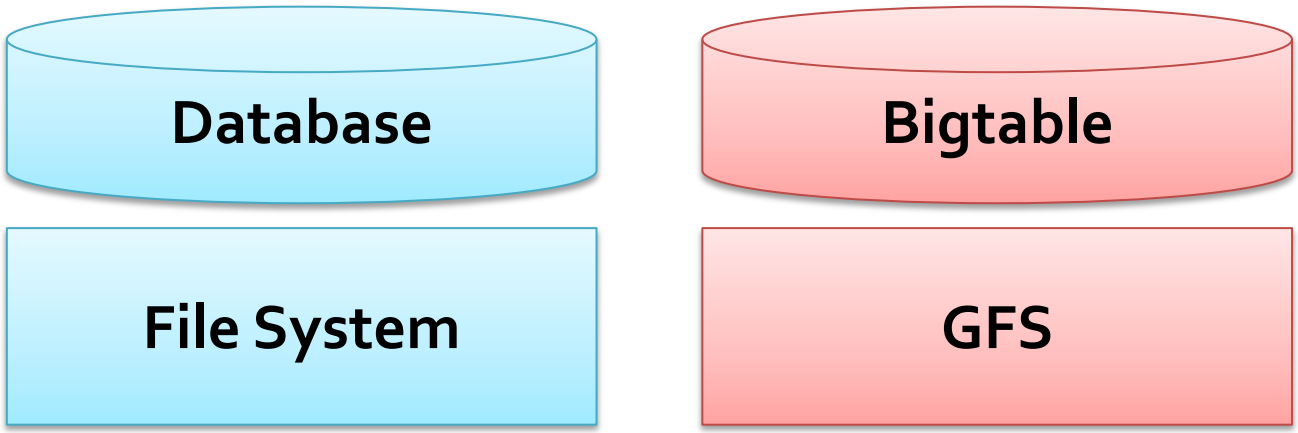
- Used by more than 60 Google products and projects
  - Google Analytics
  - Google Finance
  - Orkut
  - Personalized Search
  - Writely
  - Google Earth
  - …

Bigtable & Database

- Bigtable ≒ Database?
  - Shares many implementation strategies with databases

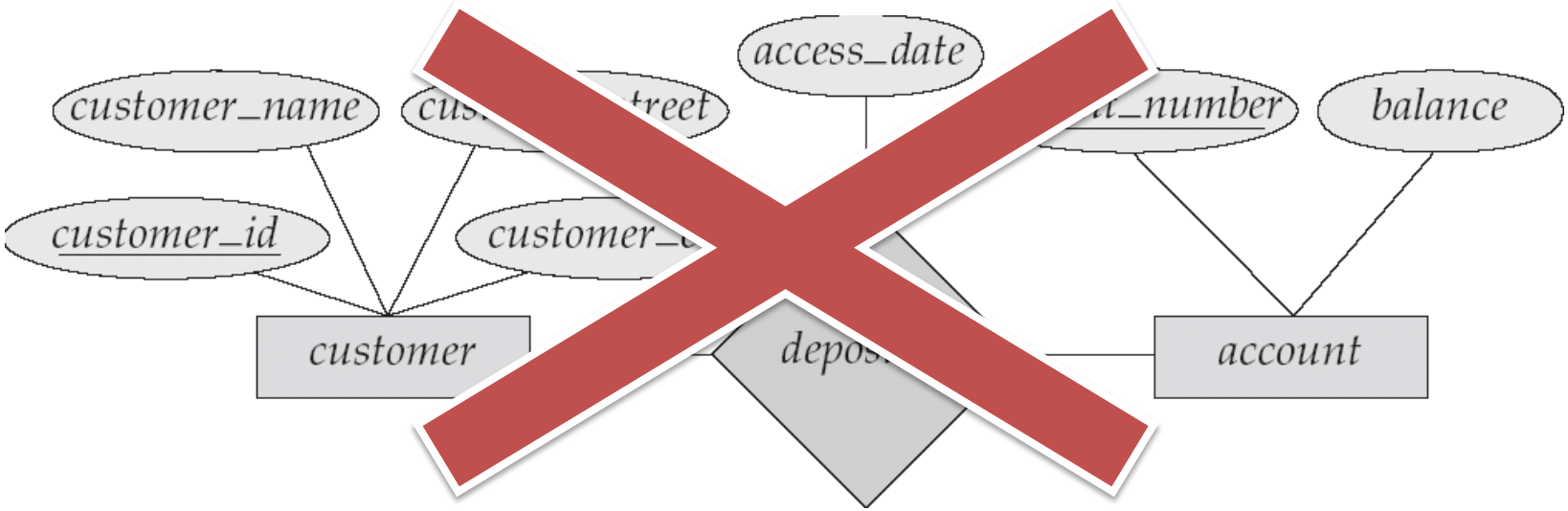
Table

	Column1	Column2	Column3
Row1			
Row2			
Row3			



Bigtable & Database (cont.)

- Bigtable ≠ Database!!
  - Does not support a full relational data model
  - But provides clients with a simple data model

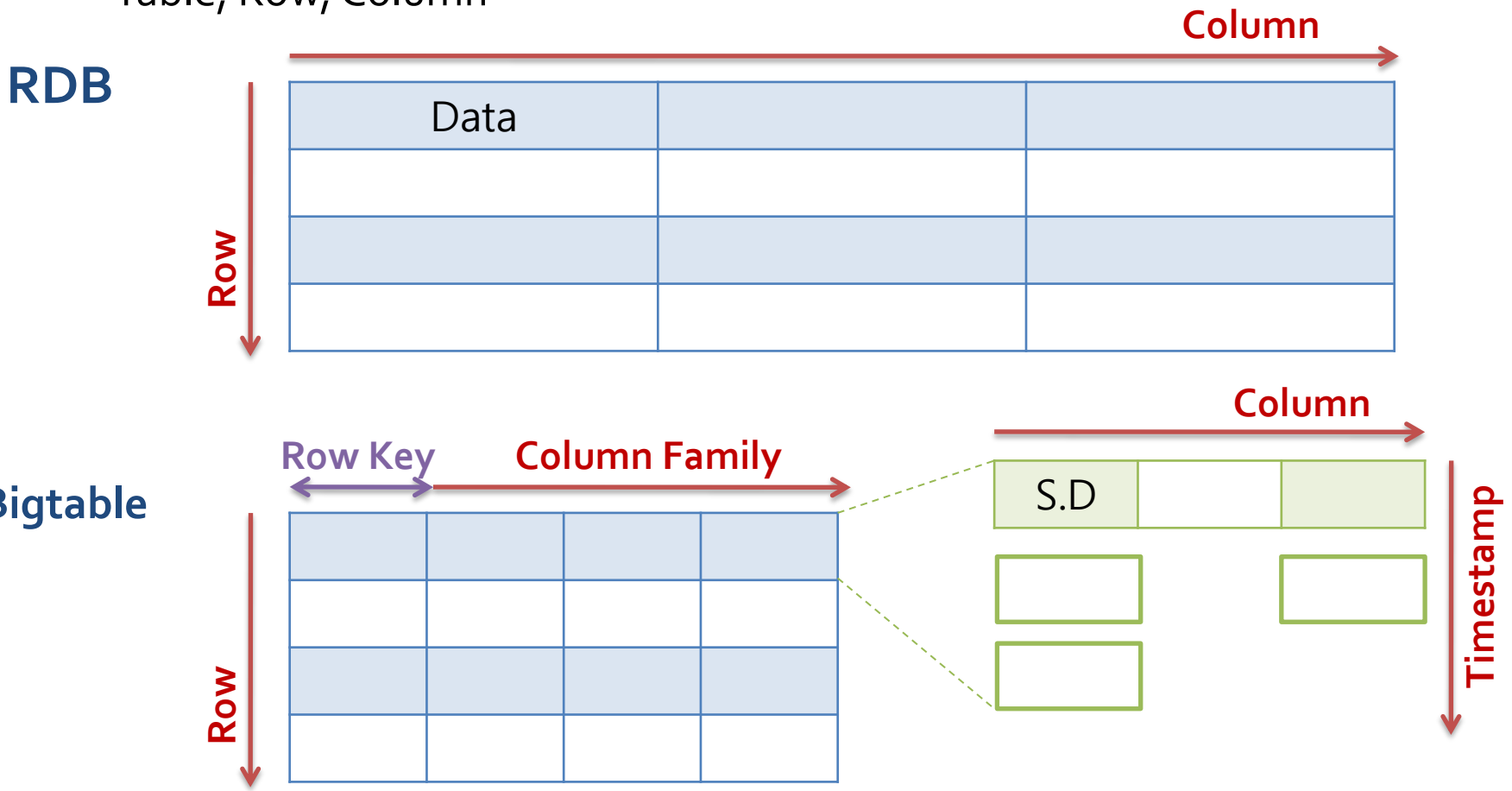


Contents

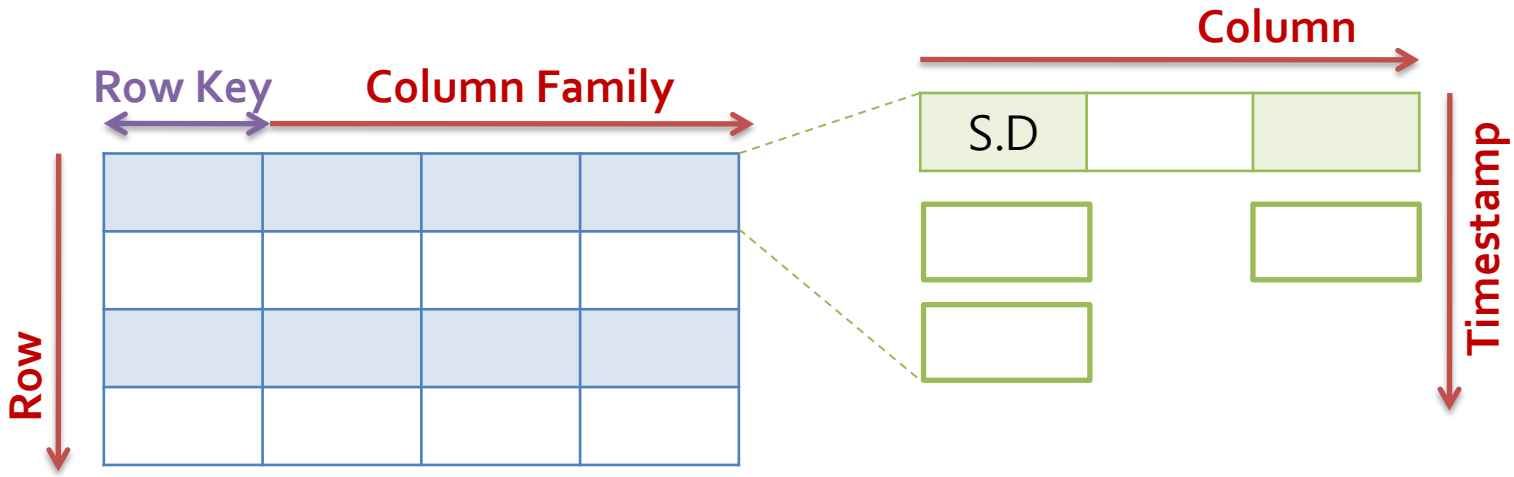
- Introduction
- Data Model
- Implementation
- Refinements
- Evaluation
- Conclusions

Table Structure

- Extends the concepts of table in Relational DB
  - Table, Row, Column



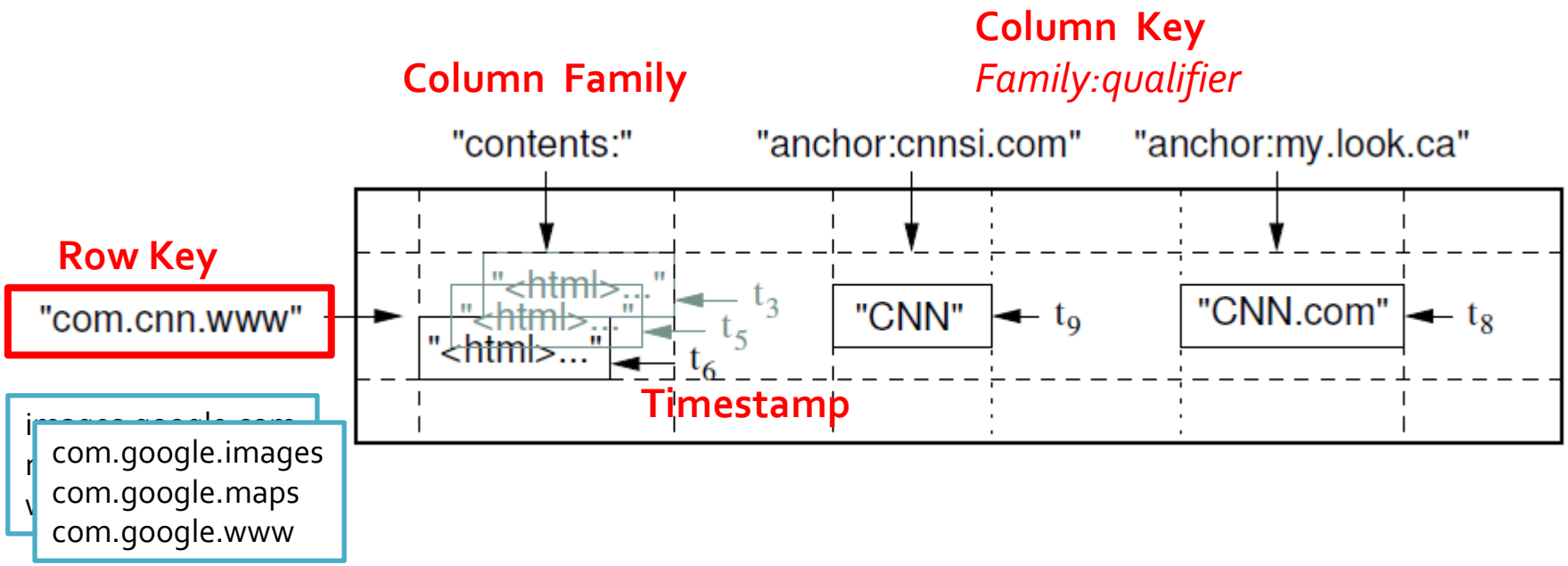
Multi Dimensional Sorted Map



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

Example : Webtable

- A kind of Bigtable
  - Want to keep a copy of a large collection of web pages
  - Could be used by many different projects

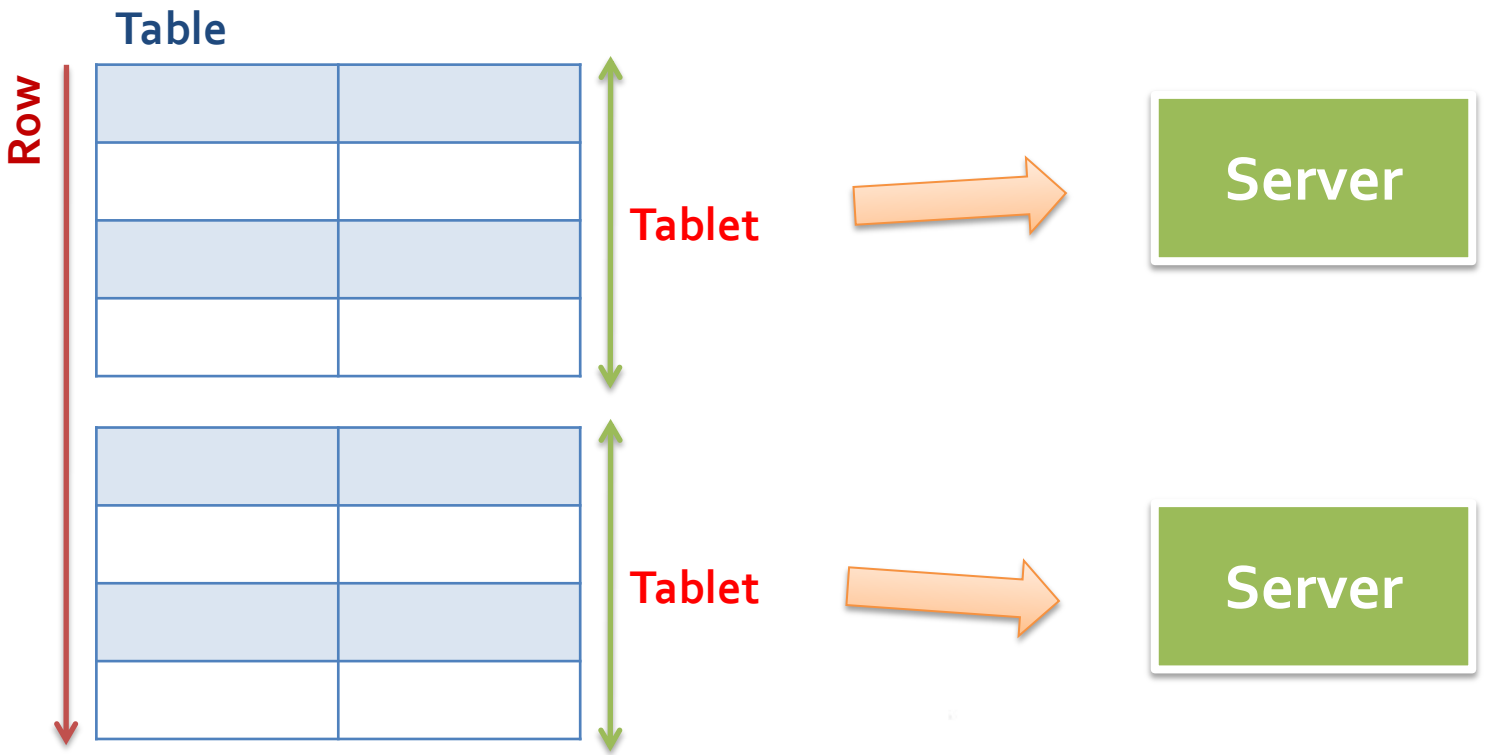


Contents

- Introduction
- Data Model
- Implementation
- Refinements
- Evaluation
- Conclusions

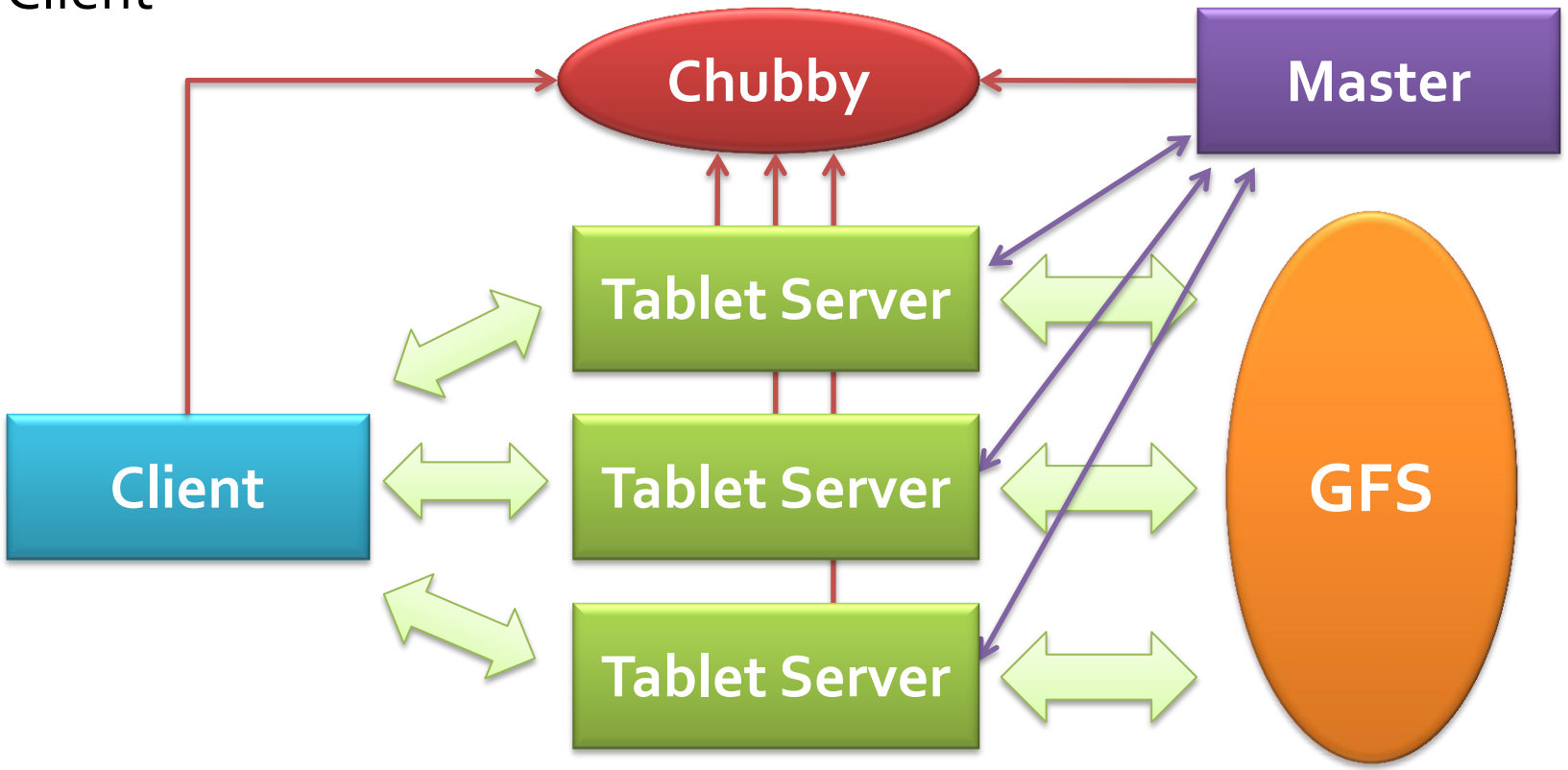
Tablet

- Data is distributed to a number of commodity servers
- Could split a table into row ranges
  - Called "tablet"
- Tablets are distributed and managed



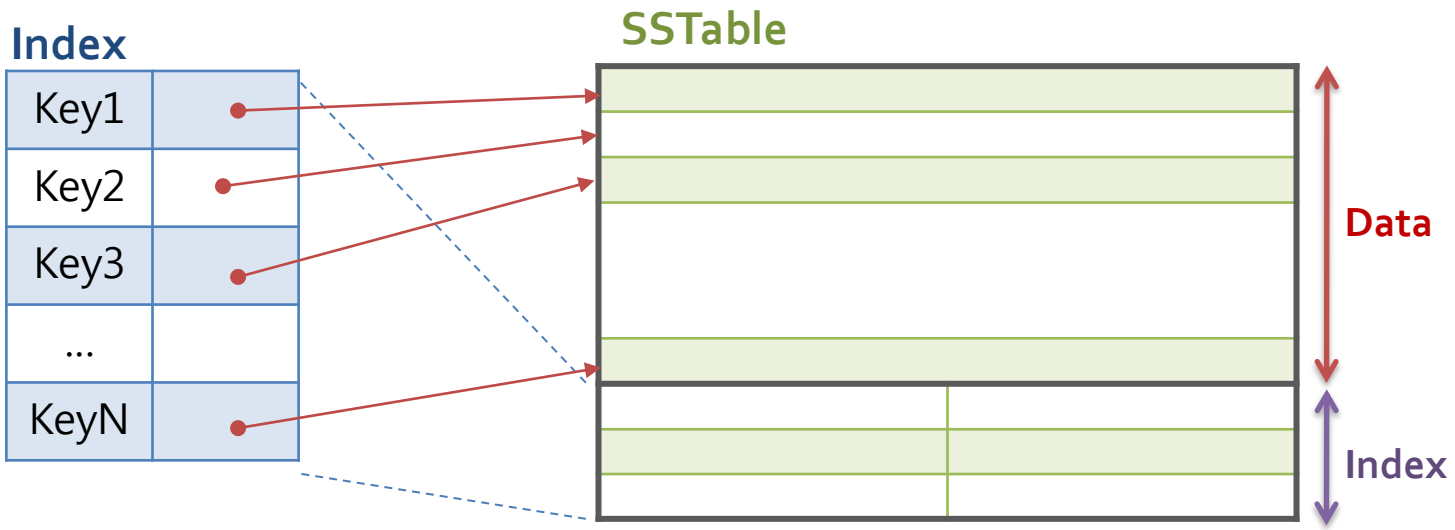
3 components

- Master
- Tablet Server
- Client



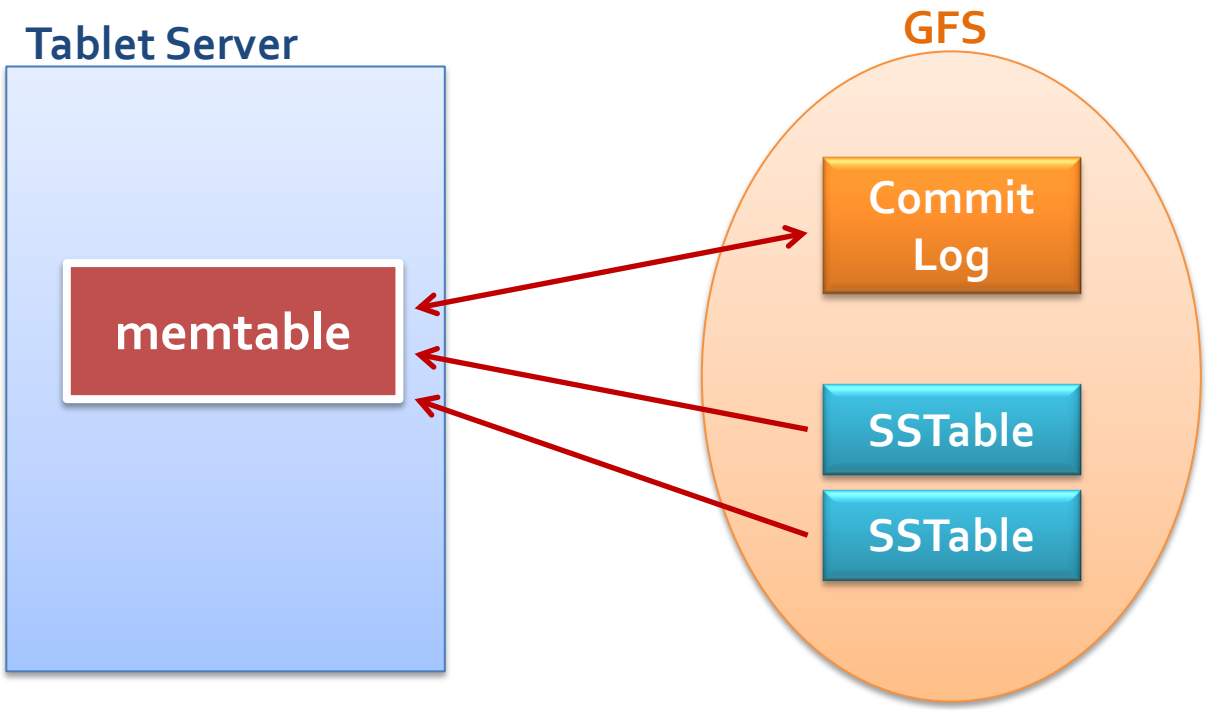
Tablet Structure

- SSTable
  - A read-only table for search in GFS
  - A tablet is composed by SSTables
  - Data & Index
    - The index is loaded into memory when the SSTable is opened



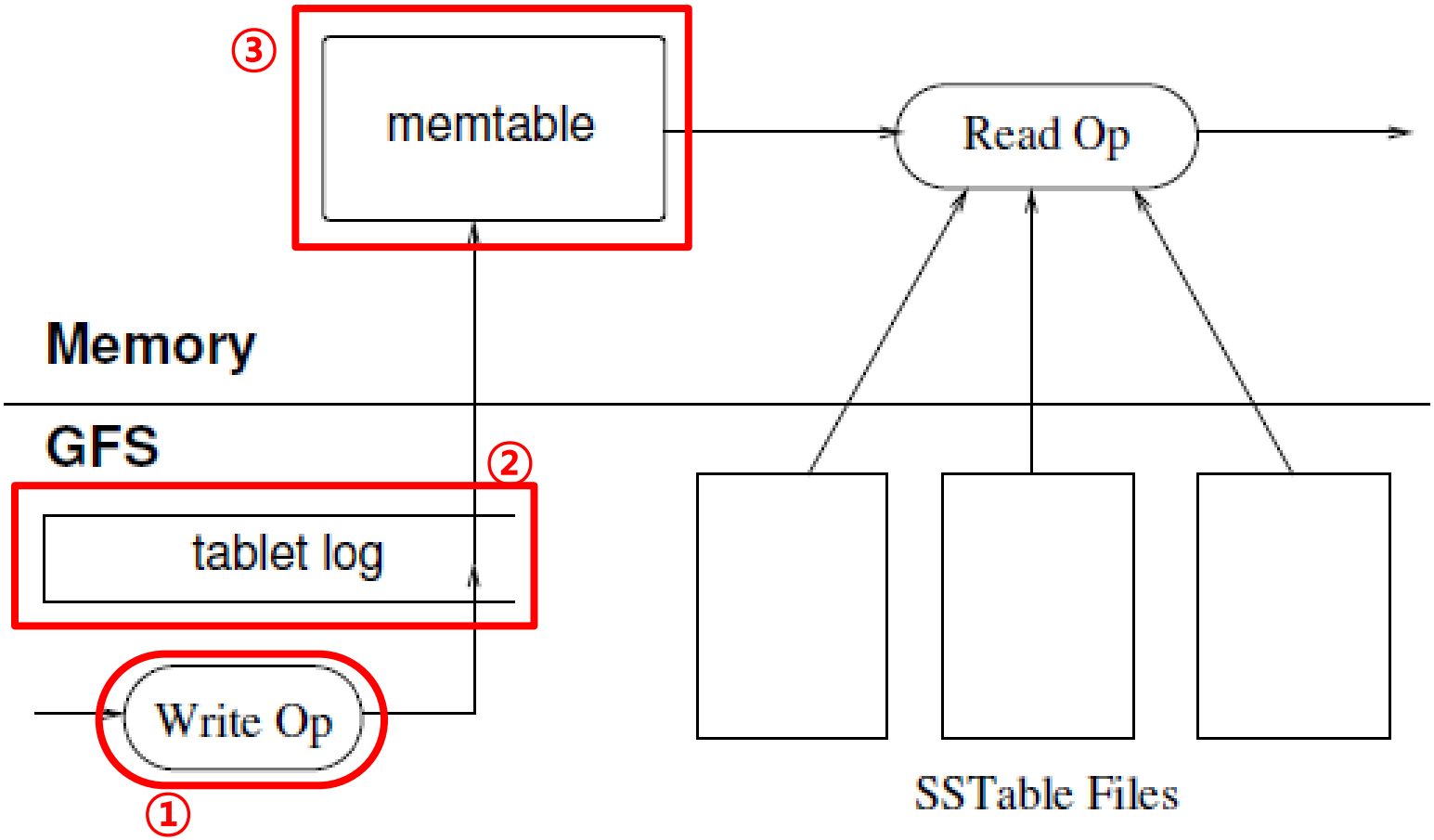
Tablet Structure (cont.)

- Memtable
  - SSTable can't be updated (read-only table)
  - A small writable table in memory per tablet
  - Commit log file is created & updated before write task



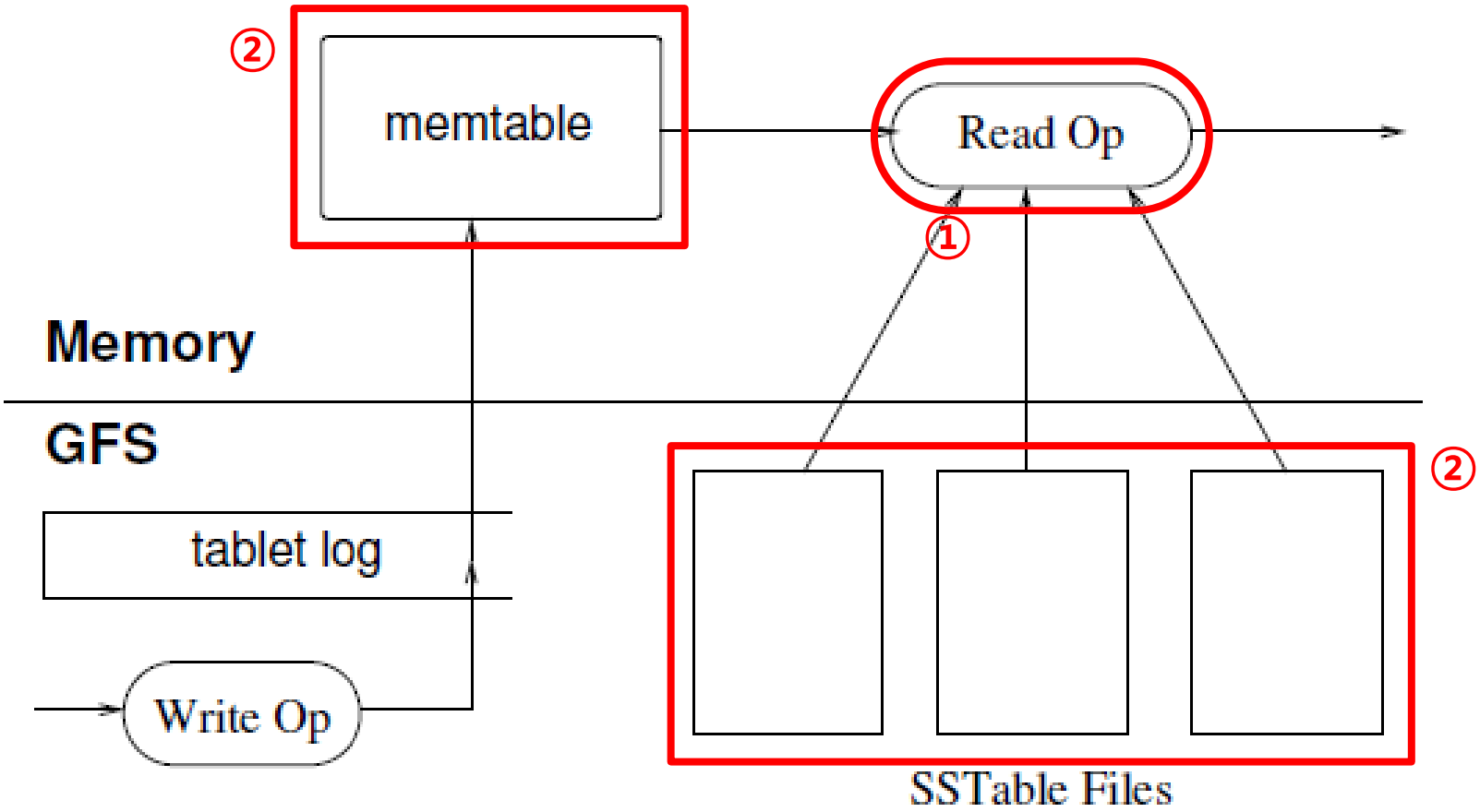
Tablet Serving

- Write operation



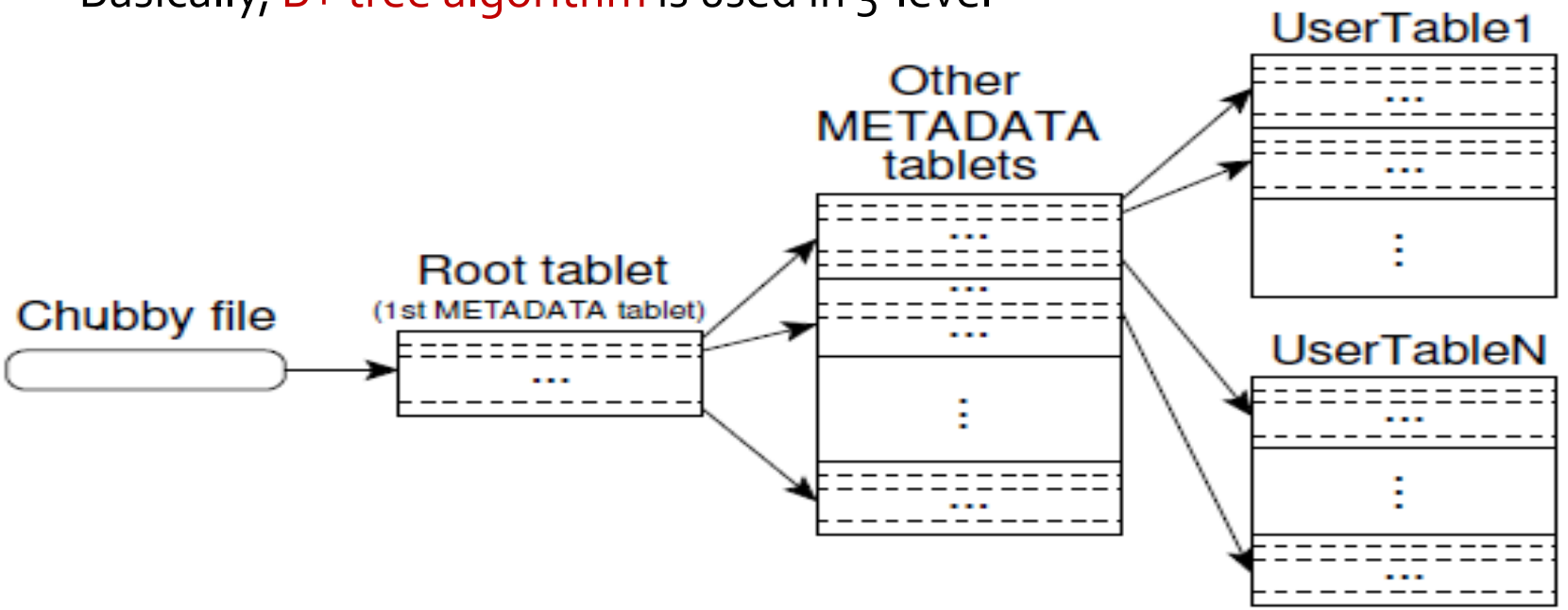
Tablet Serving (cont.)

- Read operation



Accessing Tablet from Client

- METADATA
  - Information about tablet
  - is also a table
    - And is split into tablets
- Searching tablet location
  - Basically, B+ tree algorithm is used in 3-level

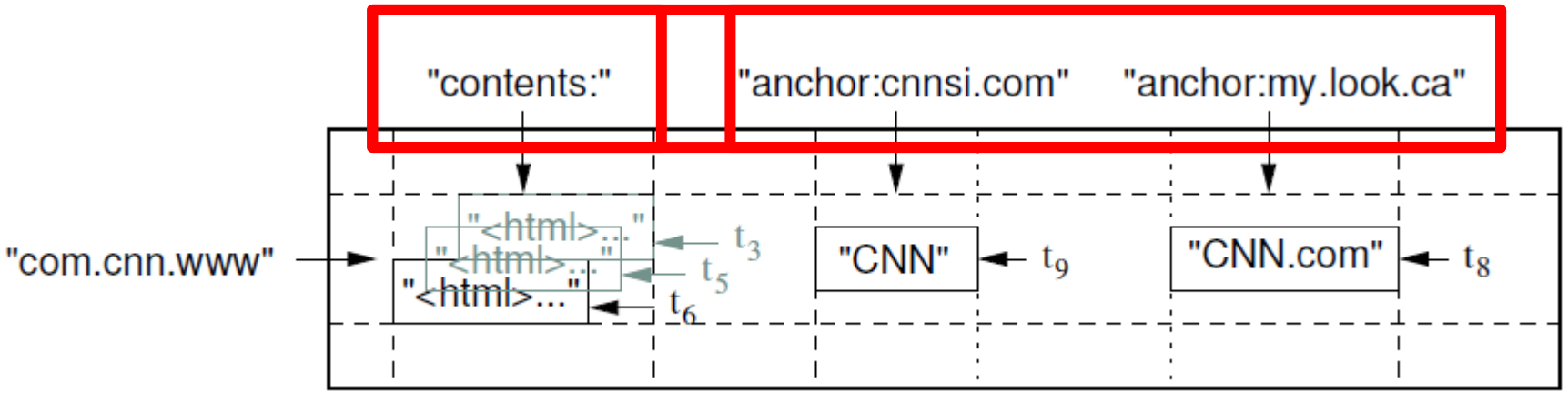


Contents

- Introduction
- Data Model
- Implementation
- Refinements
- Evaluation
- Conclusions

Locality Groups

- Some applications use only specific column families
- Clients can group multiple column families together
  - Each SSTable can store a locality group
  - More efficient reading
- Webtable



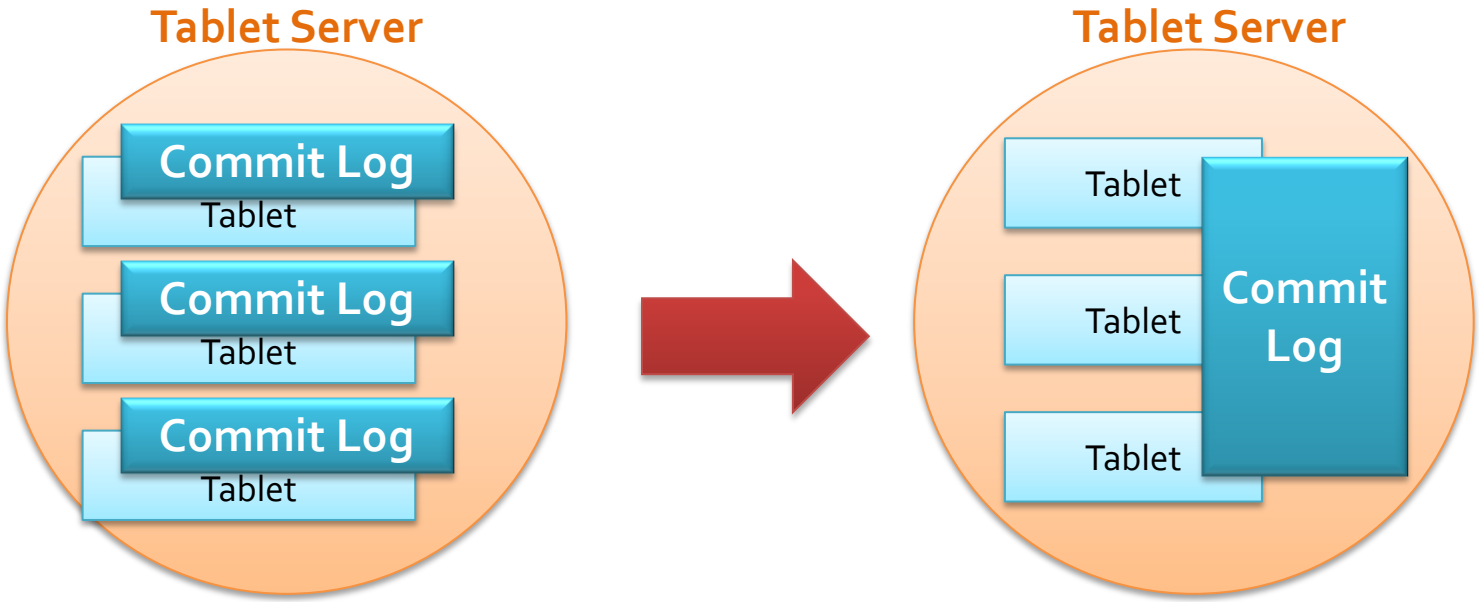
Caching for read performance

- Scan Cache
  - Higher-level cache
  - Caches the key-value pairs
  - Most useful for applications that tend to read the same data repeatedly

- Block Cache
  - Lower-level cache
  - Caches SSTables blocks that were read from GFS
  - Useful for applications that tent to read sequential data

Commit-log implementation

- A large number of log files
  - A separate log file per tablet
  - Could cause a large number of disk seeks



- For recovery,
  - Sorting the commit log entries in order of the keys
    <table, row name, log sequence number>

Contents

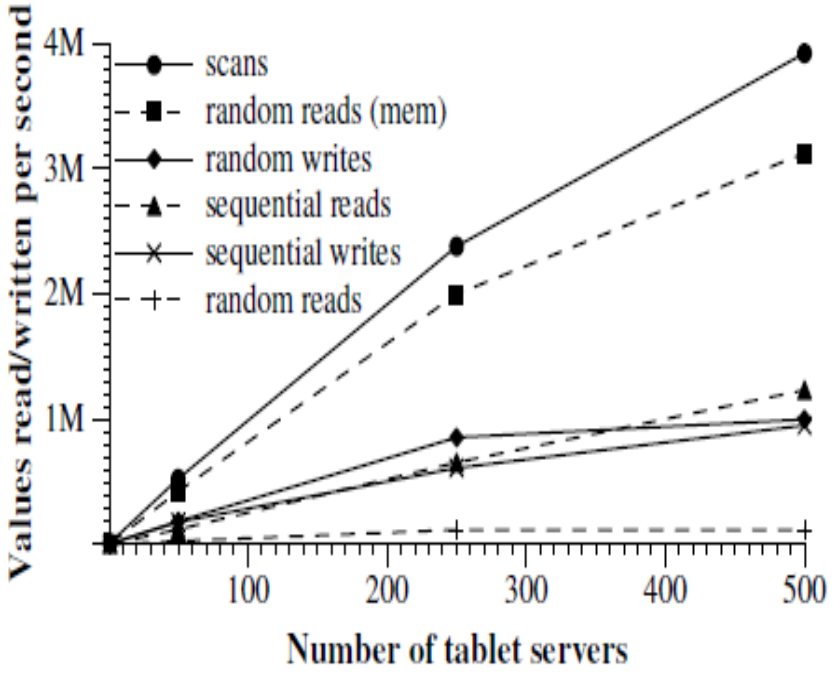
- Introduction
- Data Model
- Implementation
- Refinements
- Evaluation
- Conclusions

A setting cluster

- 1,786 machines
  - 2 * 400 GB IDE Hard drives
  - 2 * 2 GHz dual-core Opteron chipsets
  - A single gigabit Ethernet link
- Used the same number of clients as table servers
- Read and write 1000-byte values to Bigtable

Values read/written per second

Experiment	# of Tablet Servers			
	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



Contents

- Introduction
- Data Model
- Implementation
- Refinements
- Evaluation
- Conclusions

Conclusions

- As of August 2006, more than 60 projects are using Bigtable
  - Users like the performance and high availability
  - Can scale the capacity of clusters by simply adding machines
- Unusual interface
  - How difficult it has been for our users to adapt to using it
  - Many Google products successfully use Bigtable well in practice

- Future works
  - Supports for secondary indices
  - Builds cross-data-center infrastructure