GOOGLE BIGTABLE

A Distributed Storage System for Structured Data

Presented by:
Harmeet Singh Saimbhi
301376089

Outline

- Introduction
- Data Model
- Implementation
- Performance Evaluation
- Conclusions

Introduction to Distributed Storage Systems

- Used for managing large volume of structured data
  - -petabytes of data across thousands of commodity servers.
- Used for demanding workloads, such as:
  - Throughput oriented batch processing.
  - Serving latency-sensitive data to the client.
- Dynamic control over data layout and format over relational model.
- Data locality properties (revisit later briefly).

Goals achieved by Bigtable

- Wide applicability: used for 60+ Google products, including:
  – Google Analytics, Google Code, Google Earth, Google Maps and Gmail.
- Scalability (explain later under evaluation)

  -Varying the processing power of the system by allocating/ de-allocating processing units.
- High performance.
- High availability.

Data model

- Essentially a sparse, distributed, persistent multi-dimensional sorted map.
- The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.

  (row:string, column:string, time:int64) → string
- Atomic reads and writes over a single row.
- You can access any cell in the big table by providing rowId, column Name and timestamp

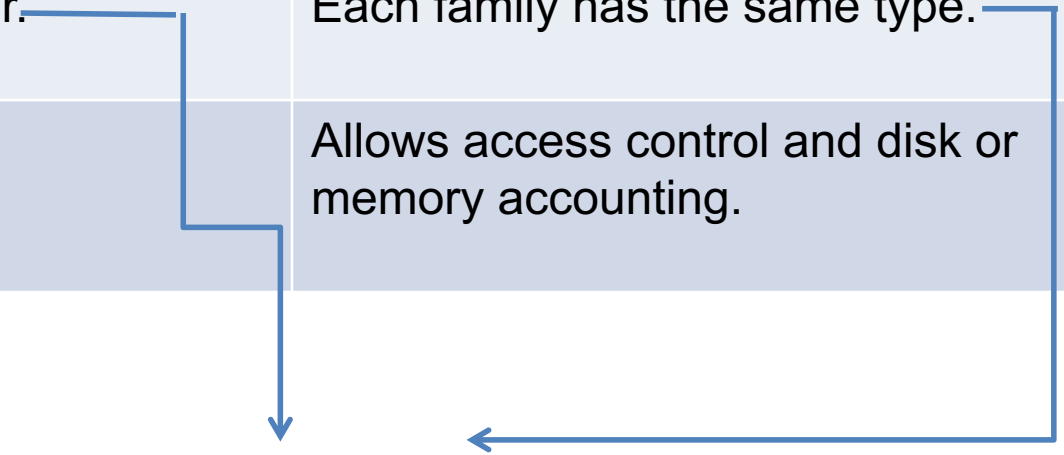
Row and column range

Rows	Columns
Row range dynamically partitioned into tablets, Each row range is called a *tablet*, which is the unit of distribution and load balancing.	Column keys grouped into column families, which form the basic unit of access control.
Data in lexicographic order.	All data stored in a column family is usually of the same type
Allows data locality.	Allows access control and disk or memory accounting.

- Important point: It is our intent that the number of distinct column families in a table be small (in the hundreds at most), and that families rarely change during operation. In contrast, a table may have an unbounded number of columns.

Row and column range

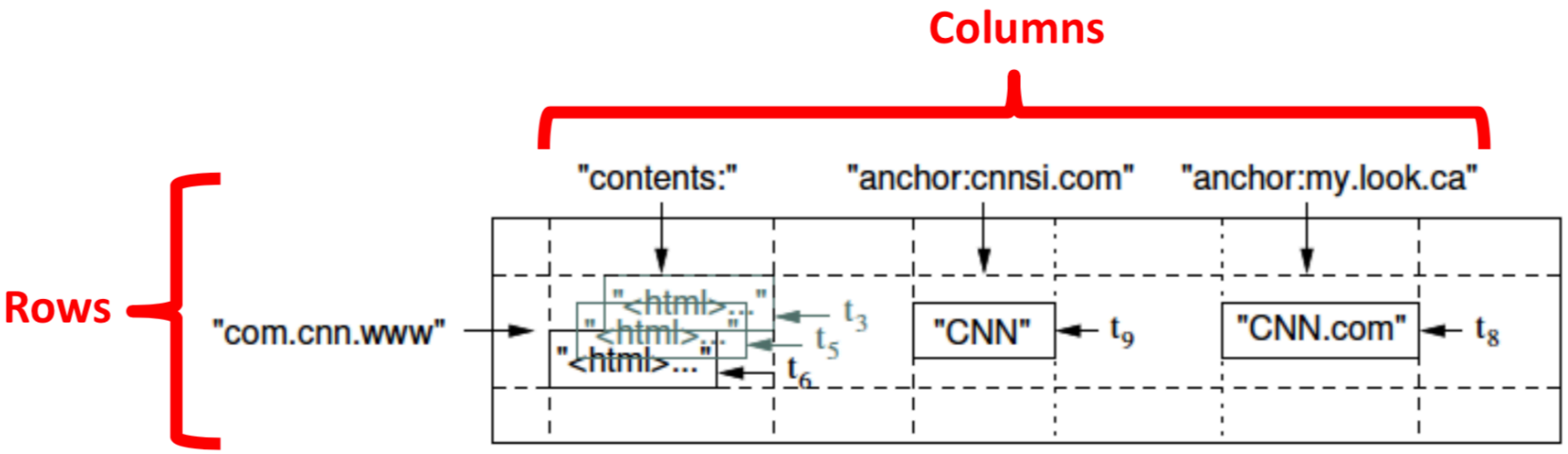
Rows	Columns
Row range dynamically partitioned into tablets , Each row range is called a *tablet*, which is the unit of distribution and load balancing.	Column keys grouped into column families.
Data in lexicographic order.	Each family has the same type.
Allows data locality.	Allows access control and disk or memory accounting.



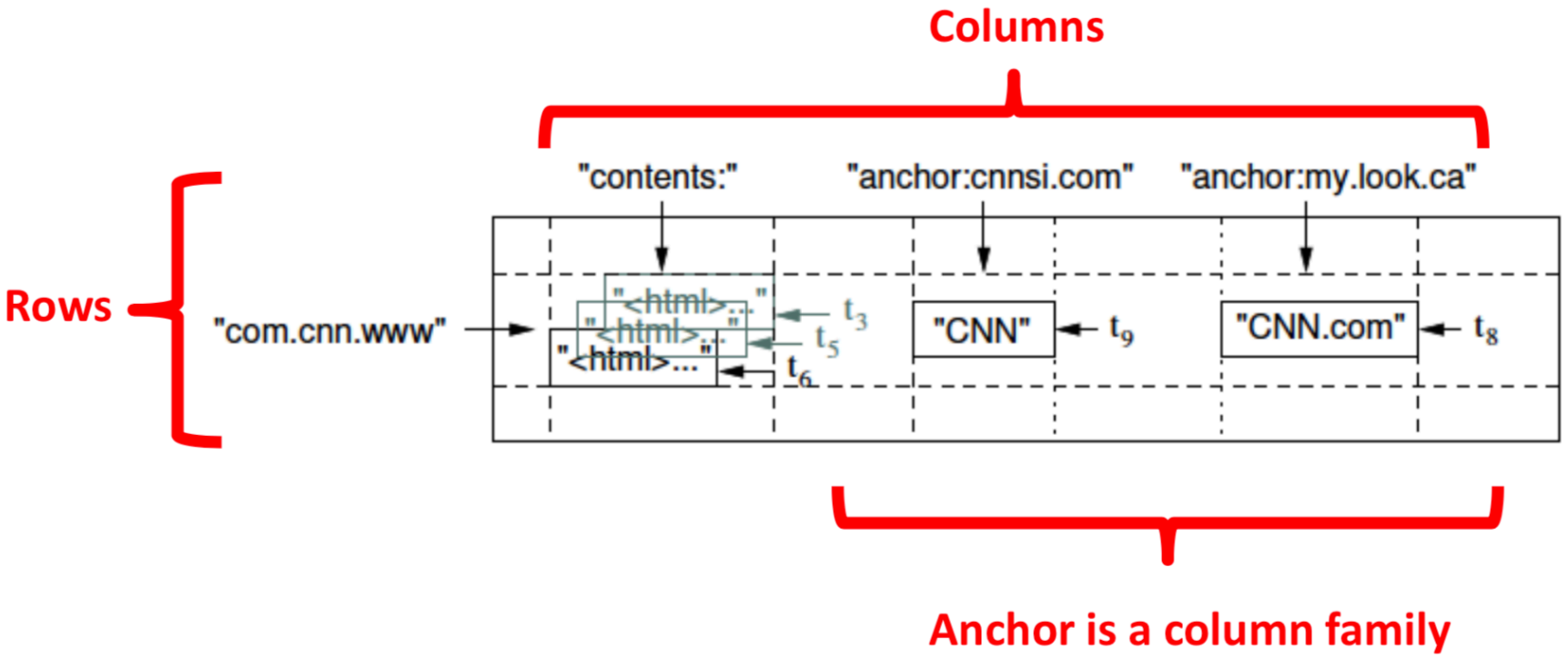
Data model

- The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.

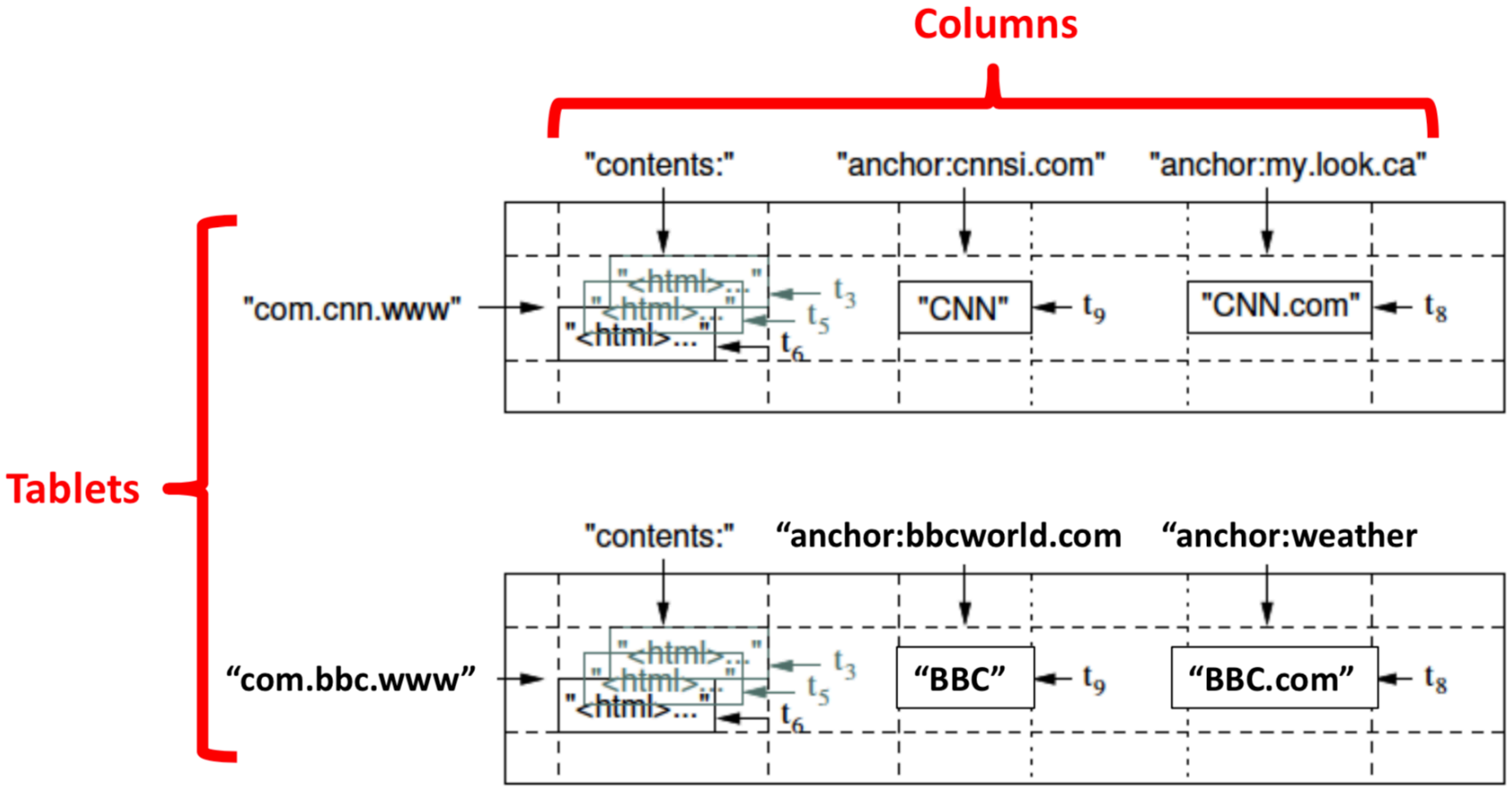
  (row:string, column:string, time:int64) → string



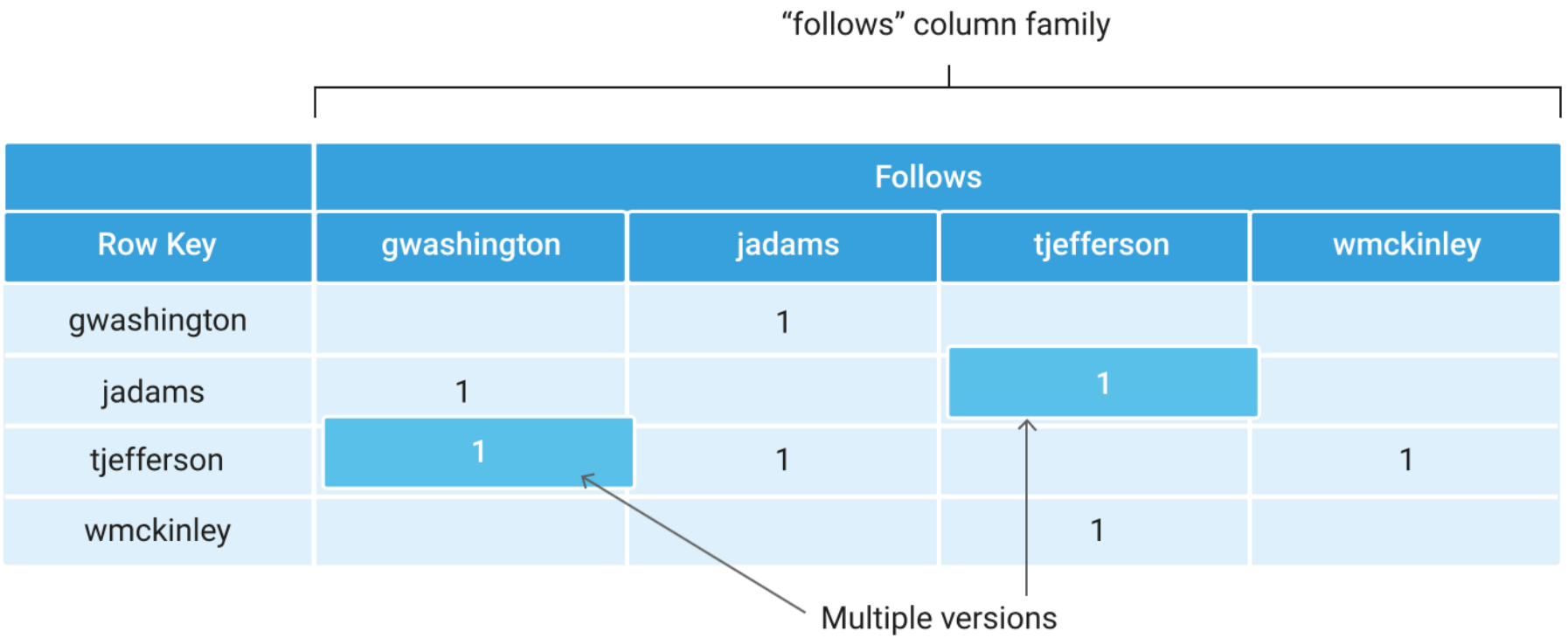
Data model


Columns
"contents:"
"anchor:cnnsi.com"
"anchor:my.look.ca"
Rows
"com.cnn.www"
"<html>..."
"<html>..."
"<html>..."
t3
t5
t6
"CNN"
t9
"CNN.com"
t8
Anchor is a column family

Data model


Columns
"contents:"
"anchor:cnnsi.com"
"anchor:my.look.ca"
Tablets
"com.cnn.www"
"<html>..."
"<html>..."
"<html>..."
t3
t5
t6
"CNN"
t9
"CNN.com"
t8
"contents:"
“anchor:bbcworld.com
“anchor:weather
“com.bbc.www”
"<html>..."
"<html>..."
"<html>..."
t3
t5
t6
“BBC”
t9
“BBC.com”
t8

Another Example

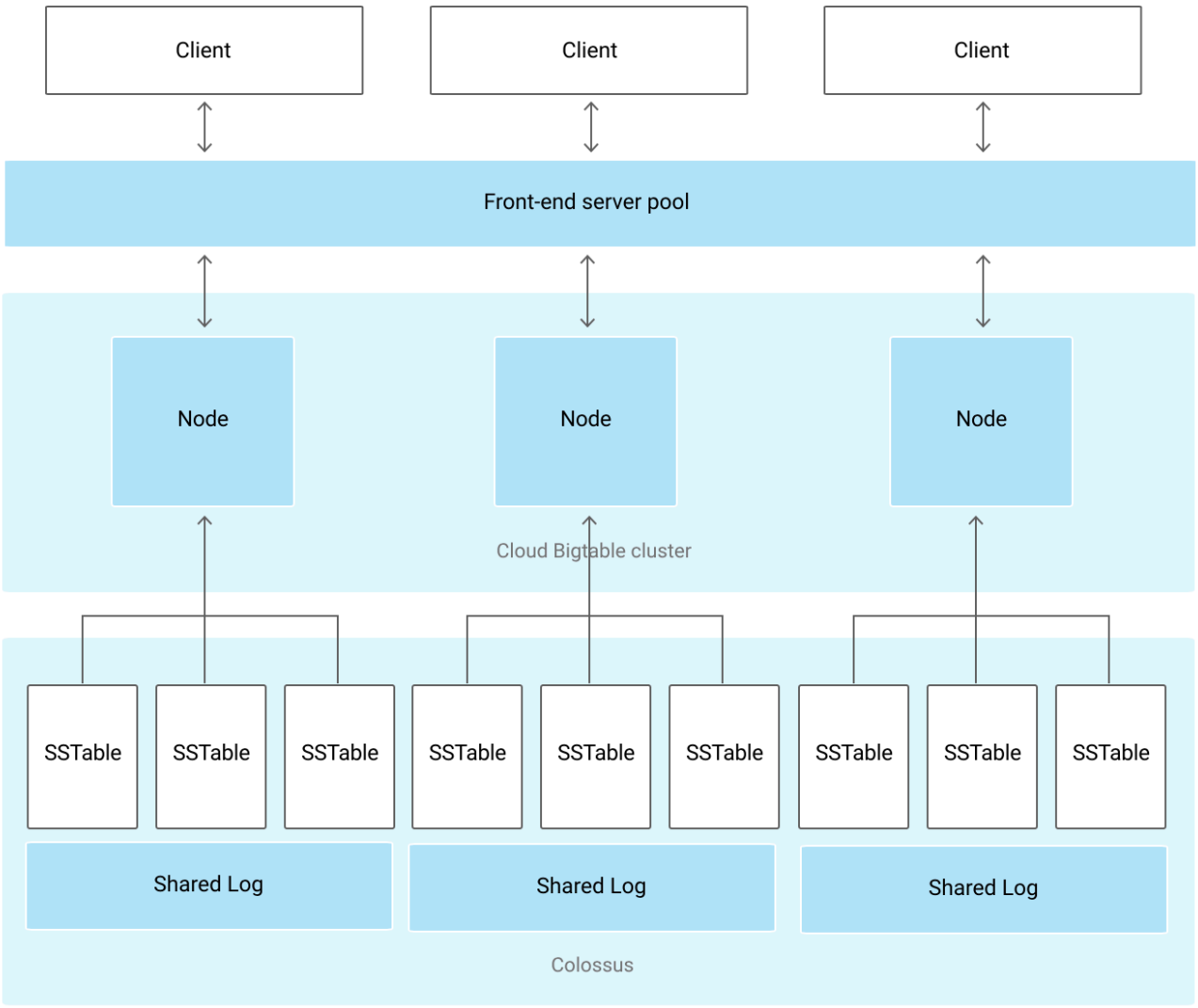


Implementation

Bigtable uses several other technologies :

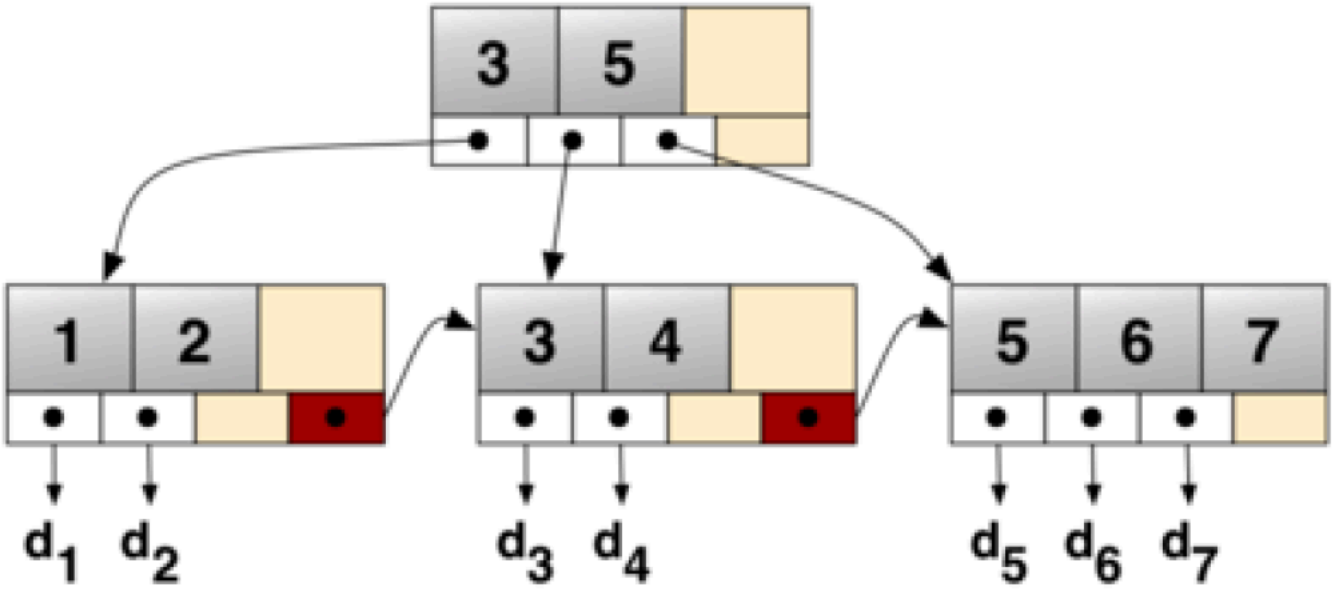
- Google File System to store log and data files.
- SSTable file format to store BigTable data.
- Chubby, a distributed lock service.
- Put picture from google store architecture of nodes

Cloud Bigtable Architecture



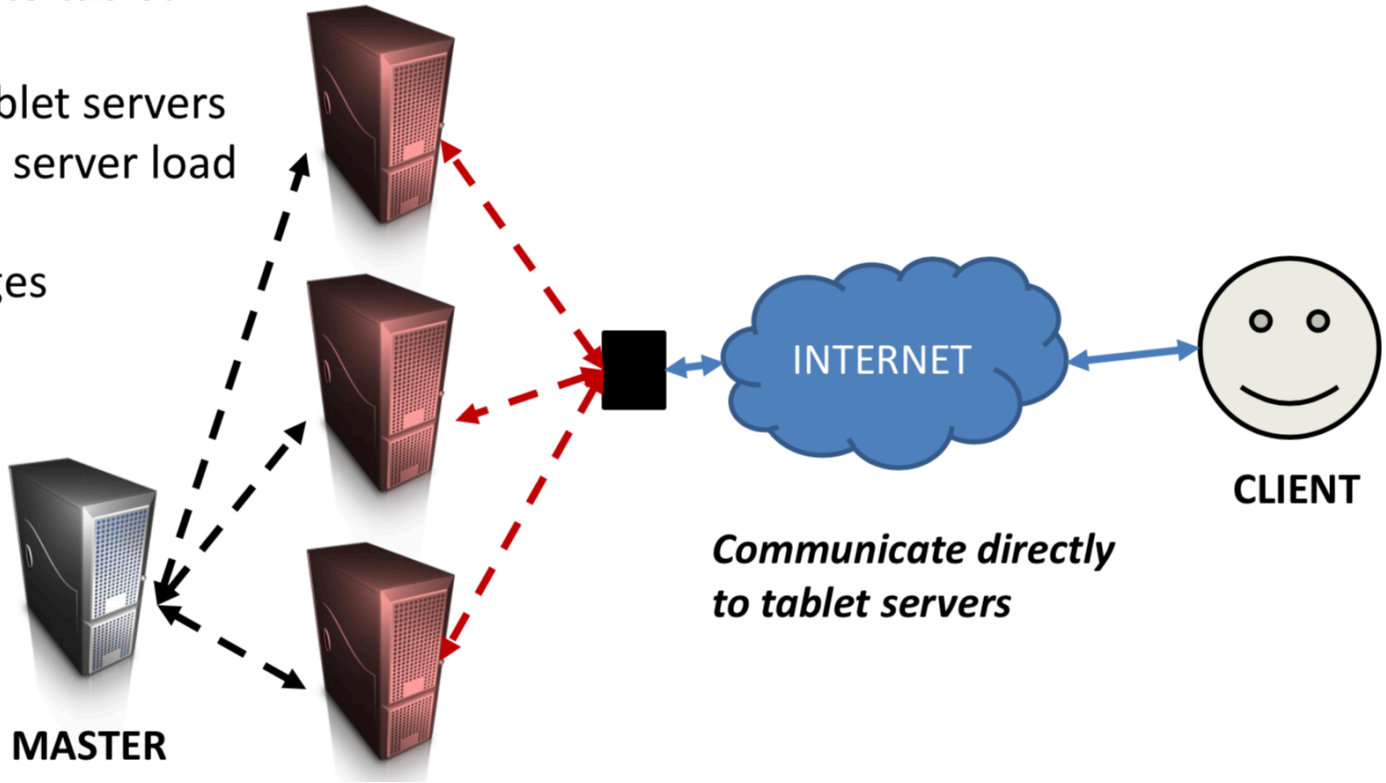
B+ Trees

- A tree with sorted data for:
  - –Efficient insertion, retrieval and removal of records.
- All **records are stored at the leaf level**, only keys stored in interior nodes.



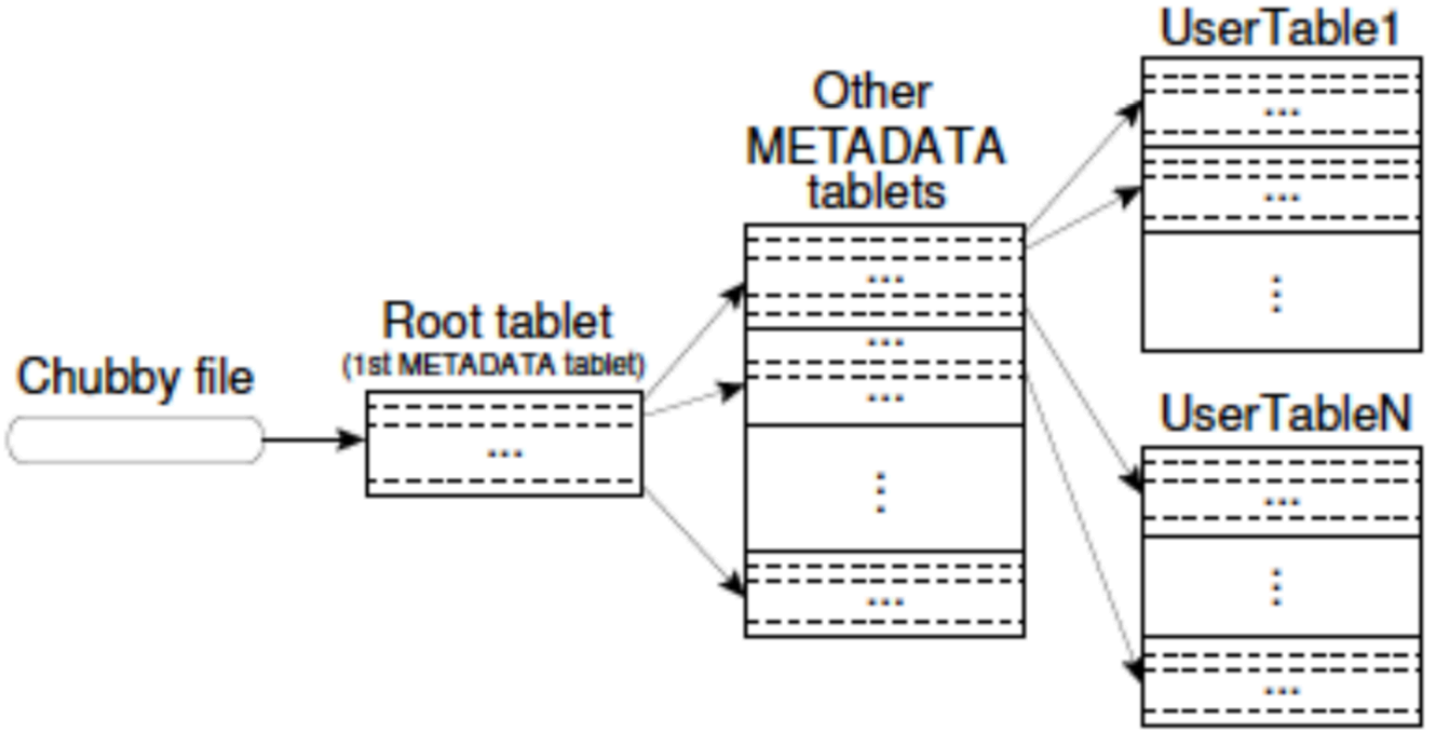
Implementation

Master responsibilities:
-Assign tablets to tablet servers
-Add/delete tablet servers
-Balance tablet server load
-GC
-Schema changes



Tablet Storage

A three-level hierarchy, similar to B+ trees.



Tablet Storage

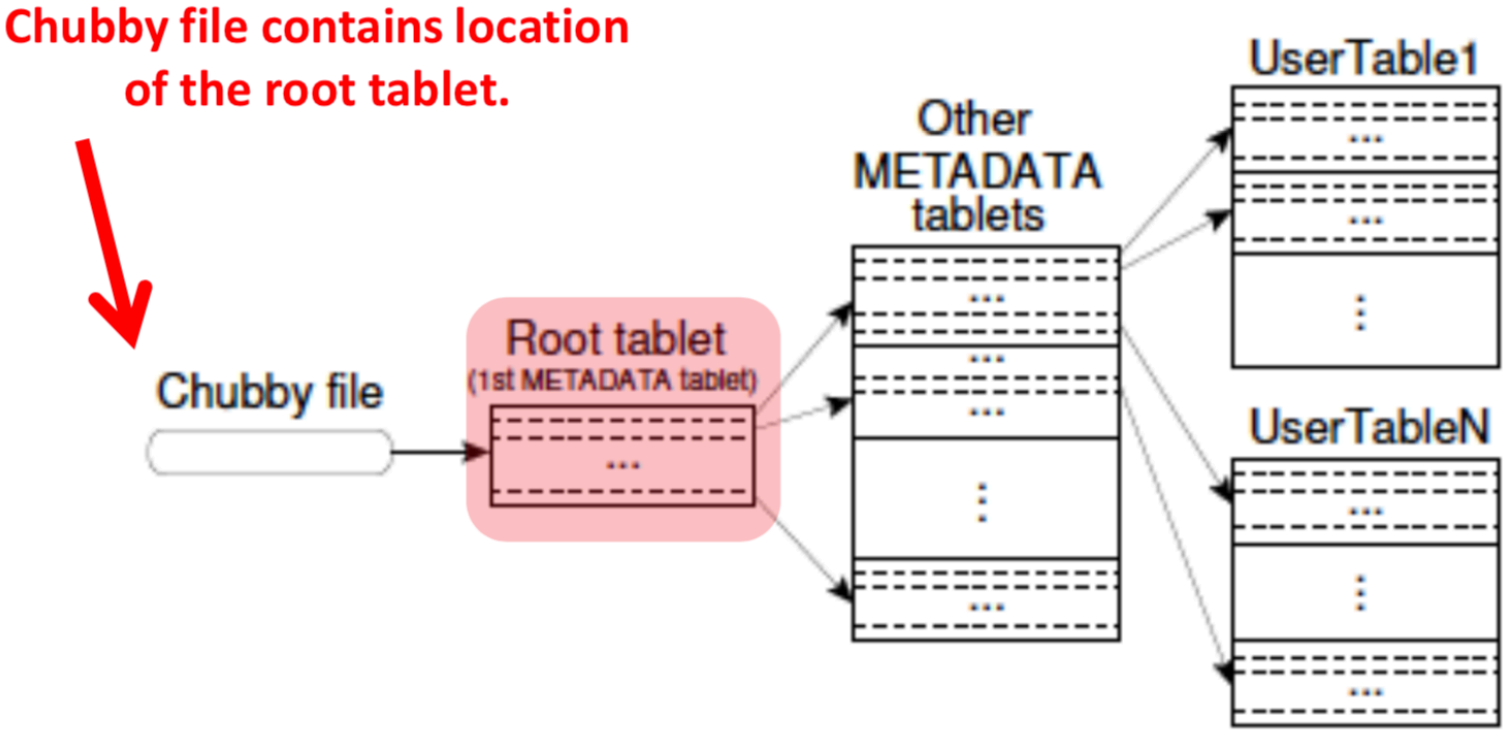


Figure 4: Tablet location hierarchy.

Tablet Storage

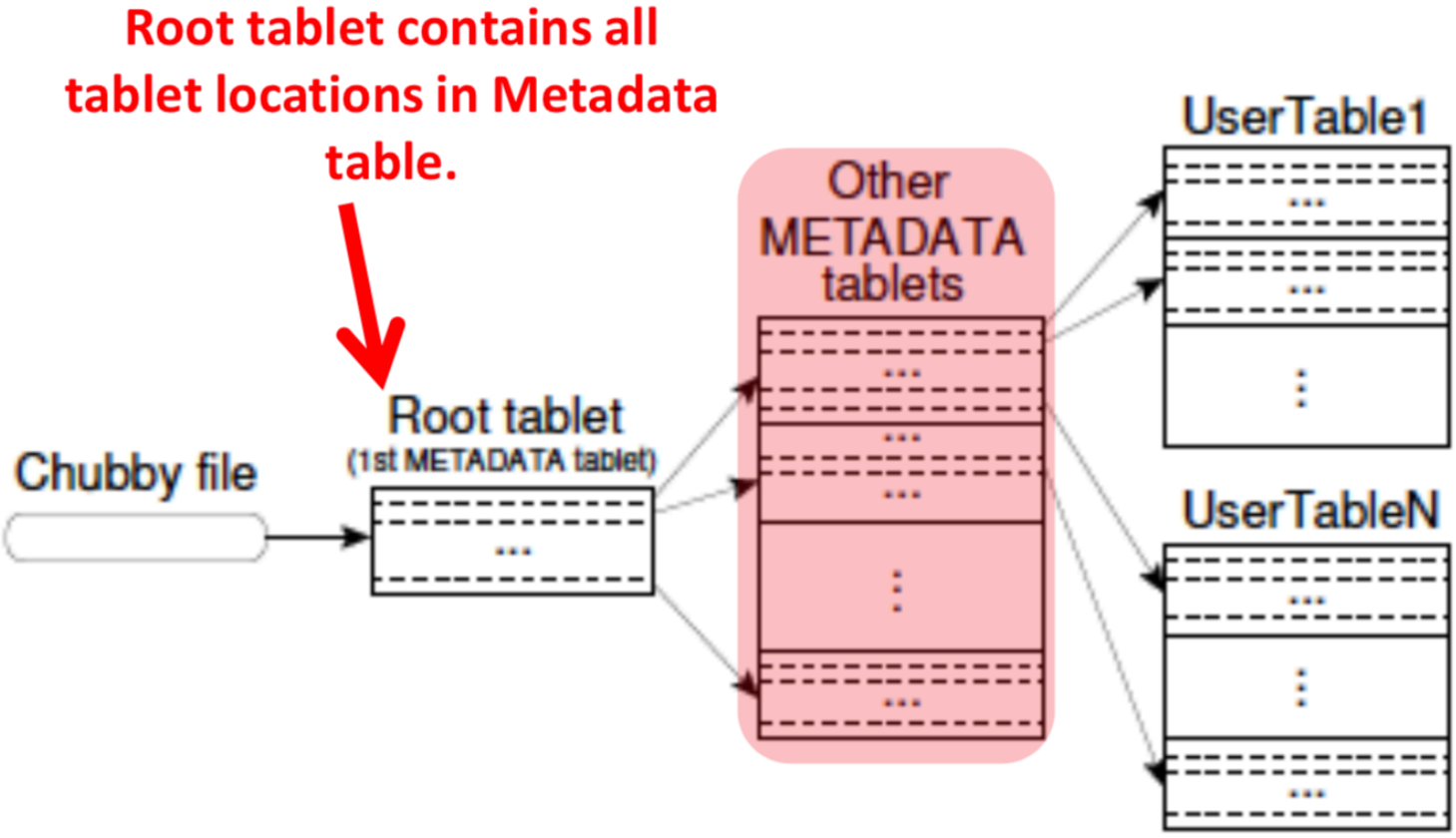


Figure 4: Tablet location hierarchy.

Tablet Storage

METADATA tablets store
Location of actual tablets

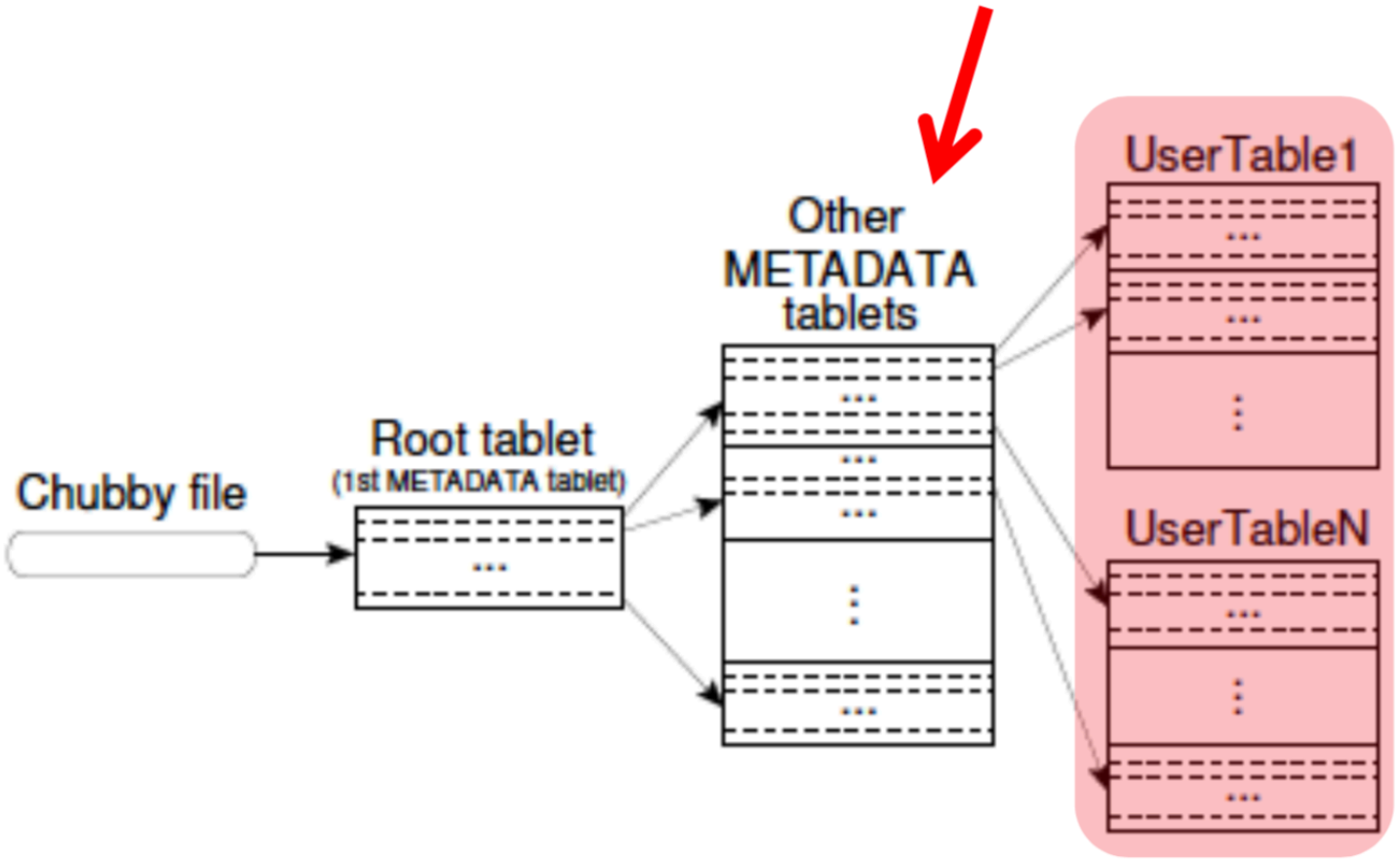


Figure 4: Tablet location hierarchy.

Tablet Storage

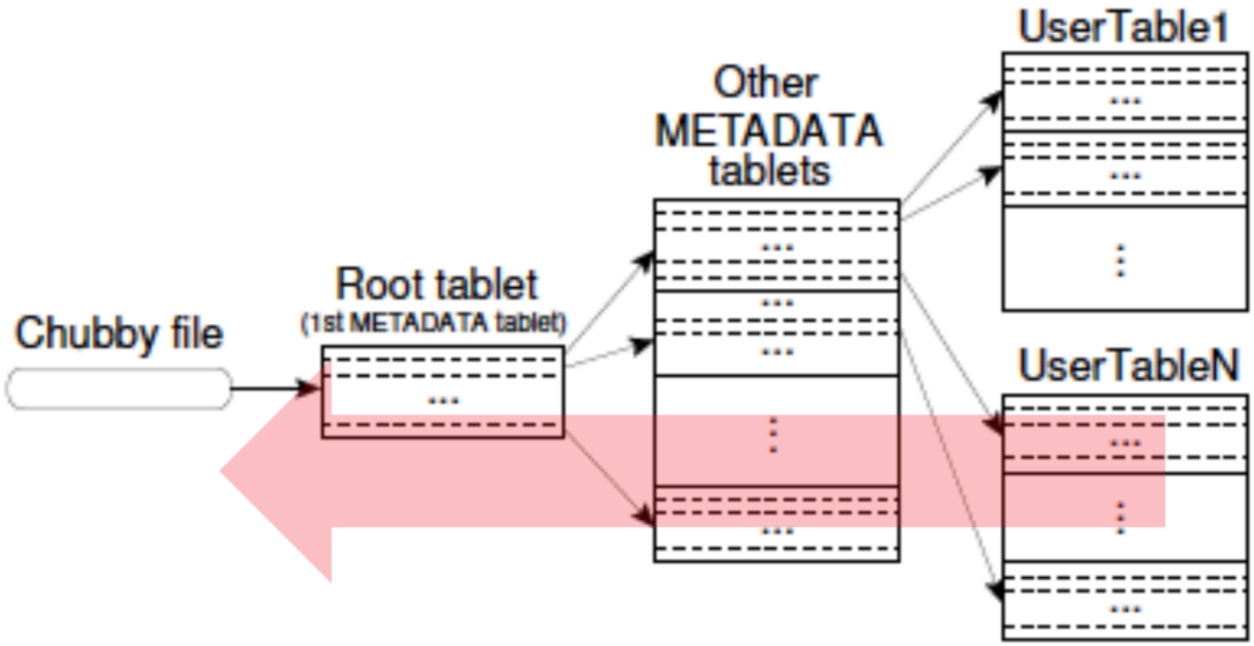


Figure 4: Tablet location hierarchy.

Client moves up the hierarchy (Metadata -> Root -> Chubby), if location of tablet is unknown or incorrect.

Data Flow

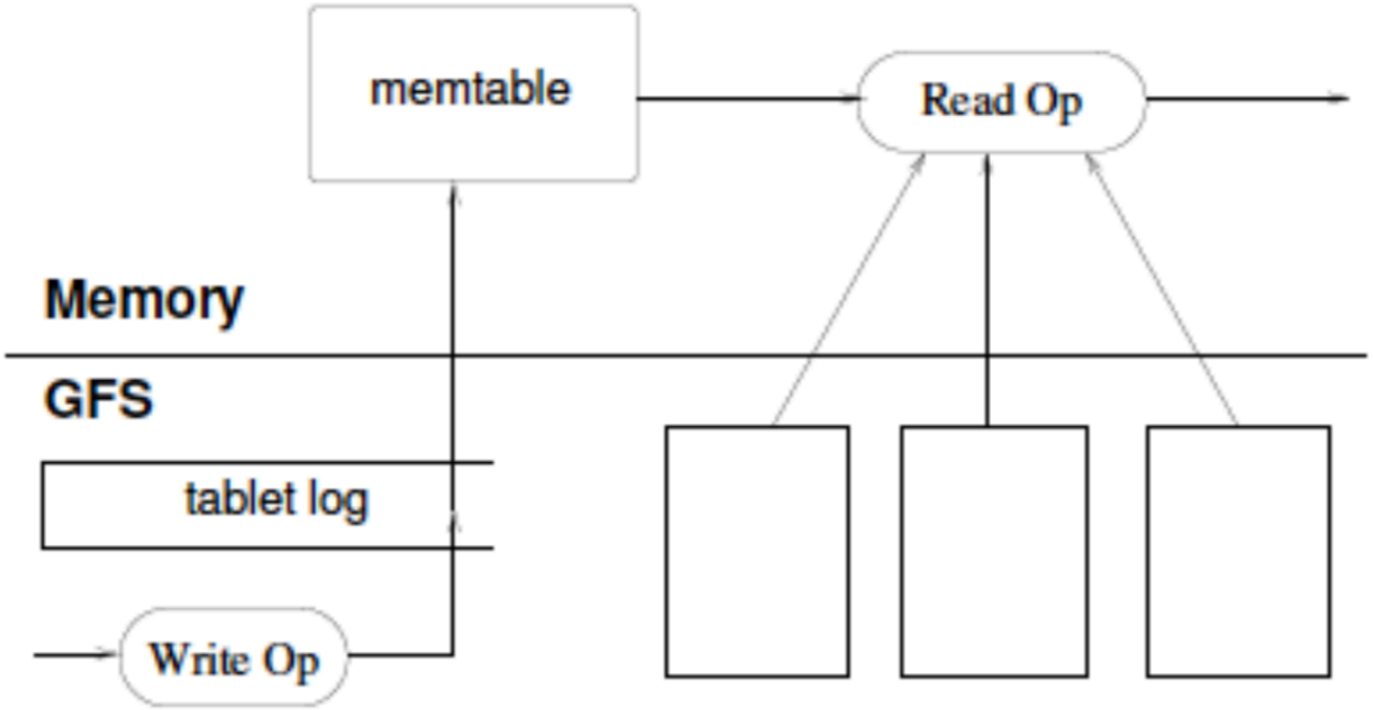


Figure 5: Tablet Representation

Data Flow

Persistent

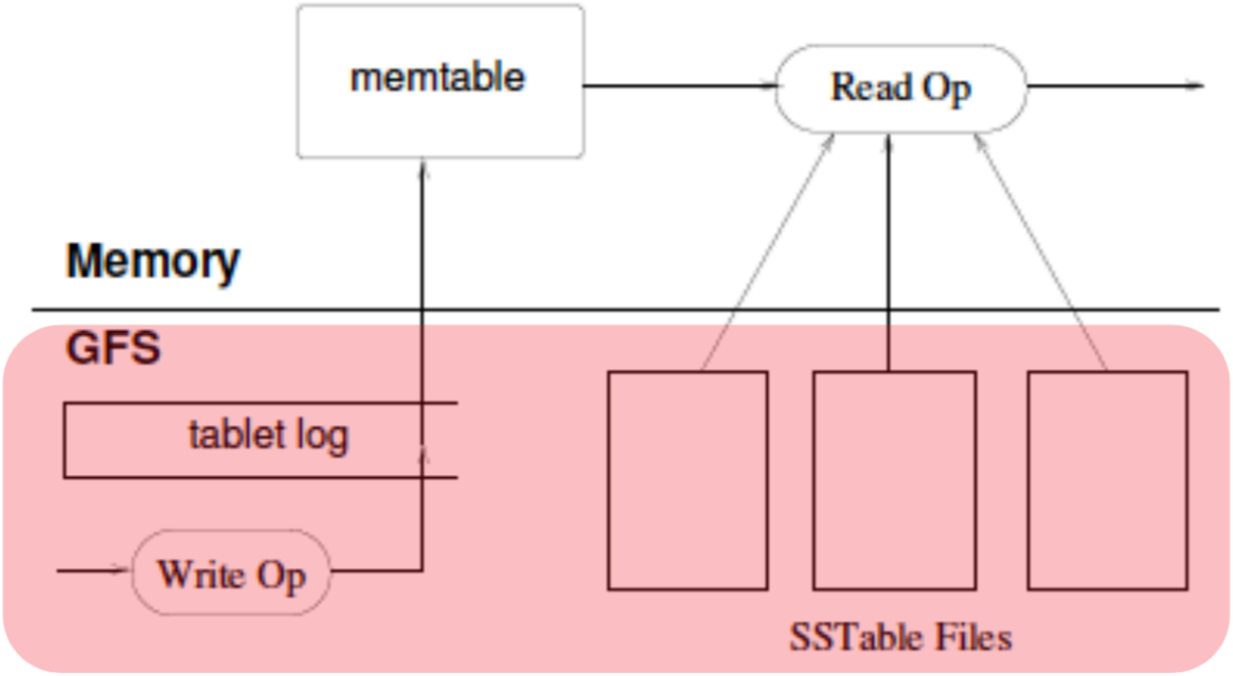


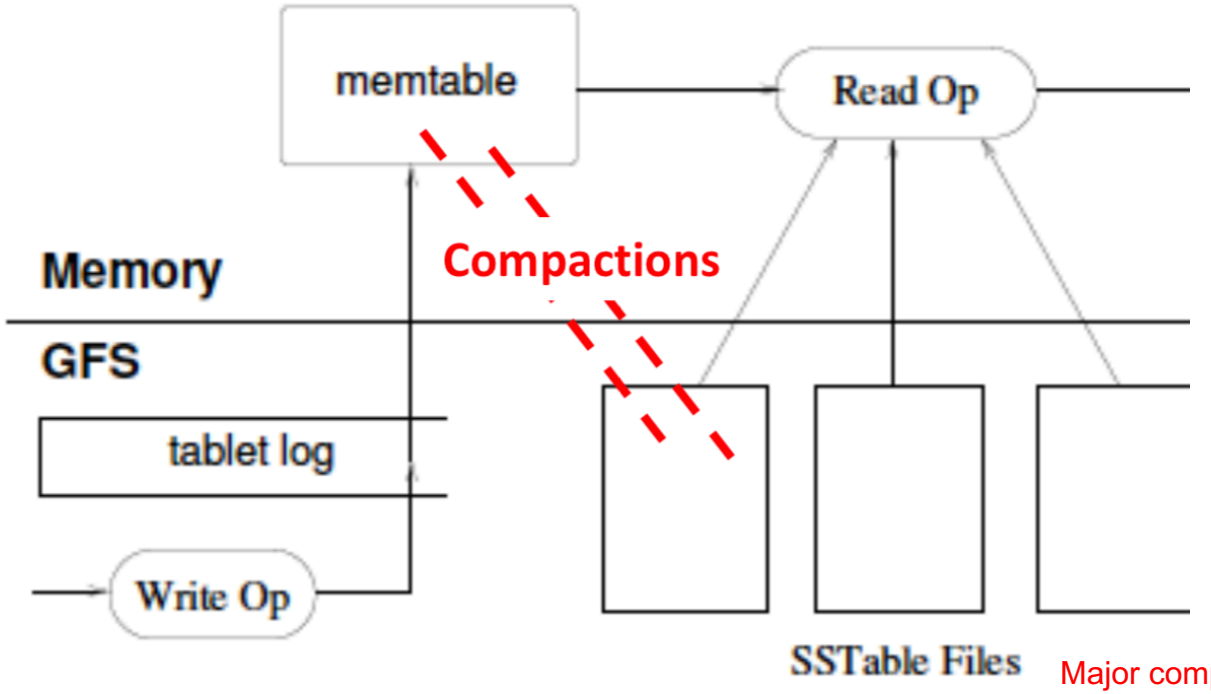
Figure 5: Tablet Representation

Data Flow

Compactions occur regularly, advantages: -Shrinks memory usage. - Reduces amount of data read from log during recovery.

Table in memory which stores sequence of events read/ write operations as they occur and those events also get stored inside tablet log for fault tolerance purpose.

Minor compaction: Write memtable buffer to a new SSTable



Major compaction:Rewrite all SSTables into clean Sstable.

Performance Evaluation

- Scan:

  -Scans over values in a row range.

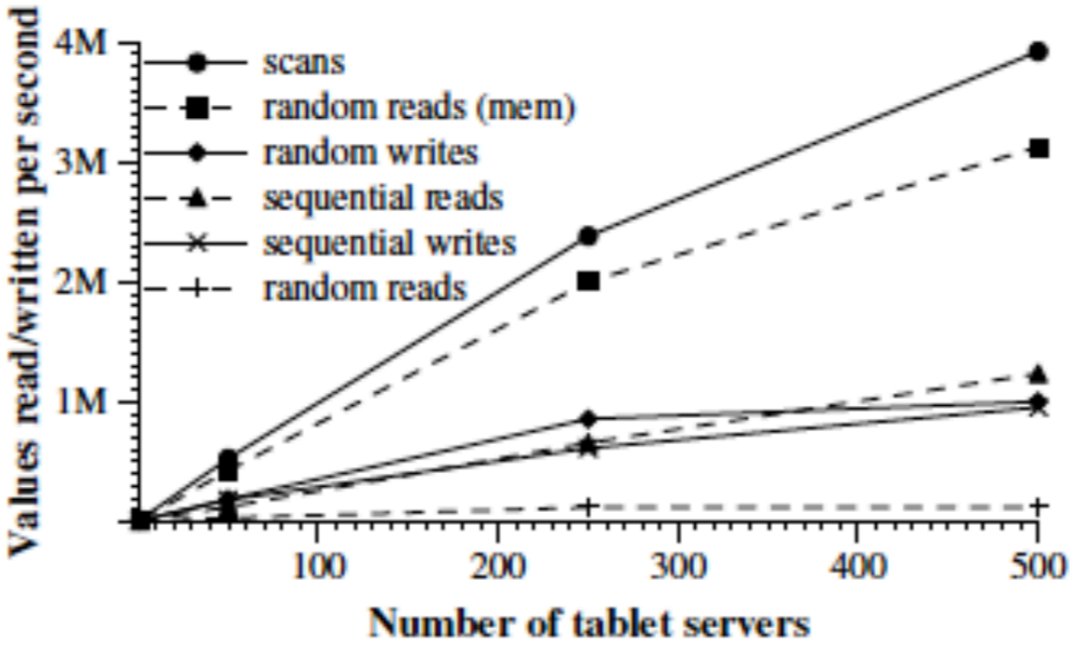
- Random reads from memory.
- Random reads/writes:

  -R keys to be read/written spread over N clients.

- Sequential reads/writes:

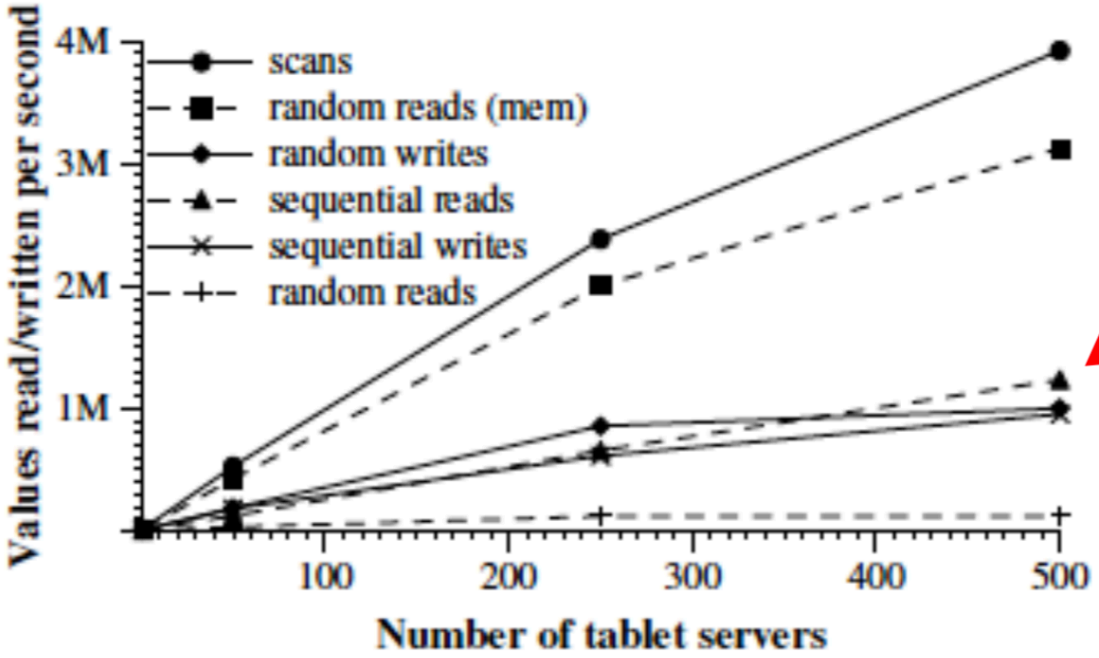
  -0 to R-1 keys to be read/written spread over N clients.

Performance Evaluation



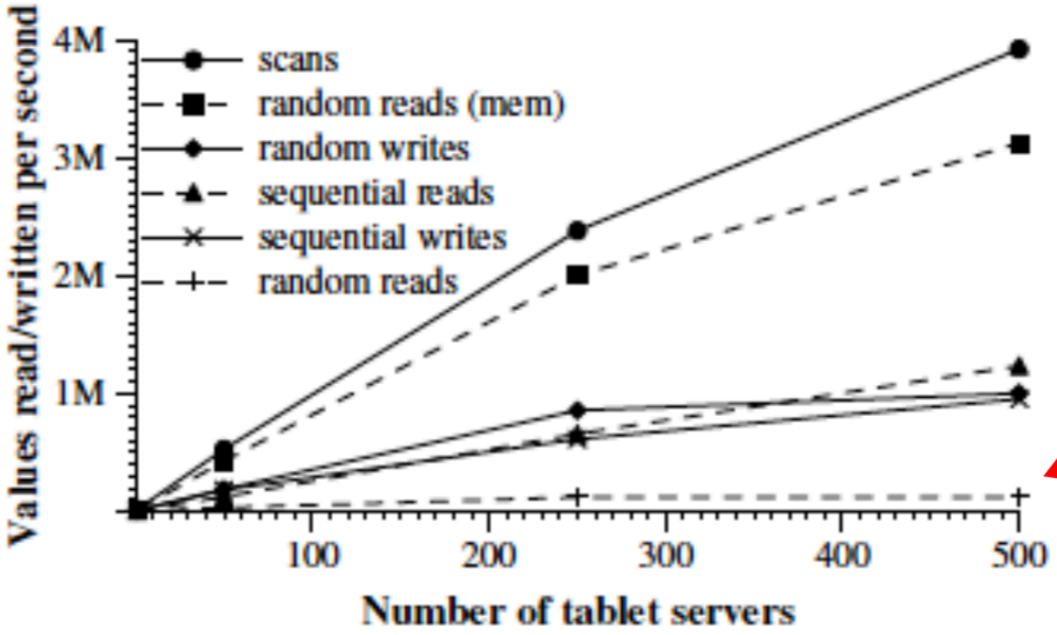
Scan uses single RPC call and shows best performance.

Performance Evaluation



Sequential reads are better than random reads, since each fetched block is used to serve next requests.

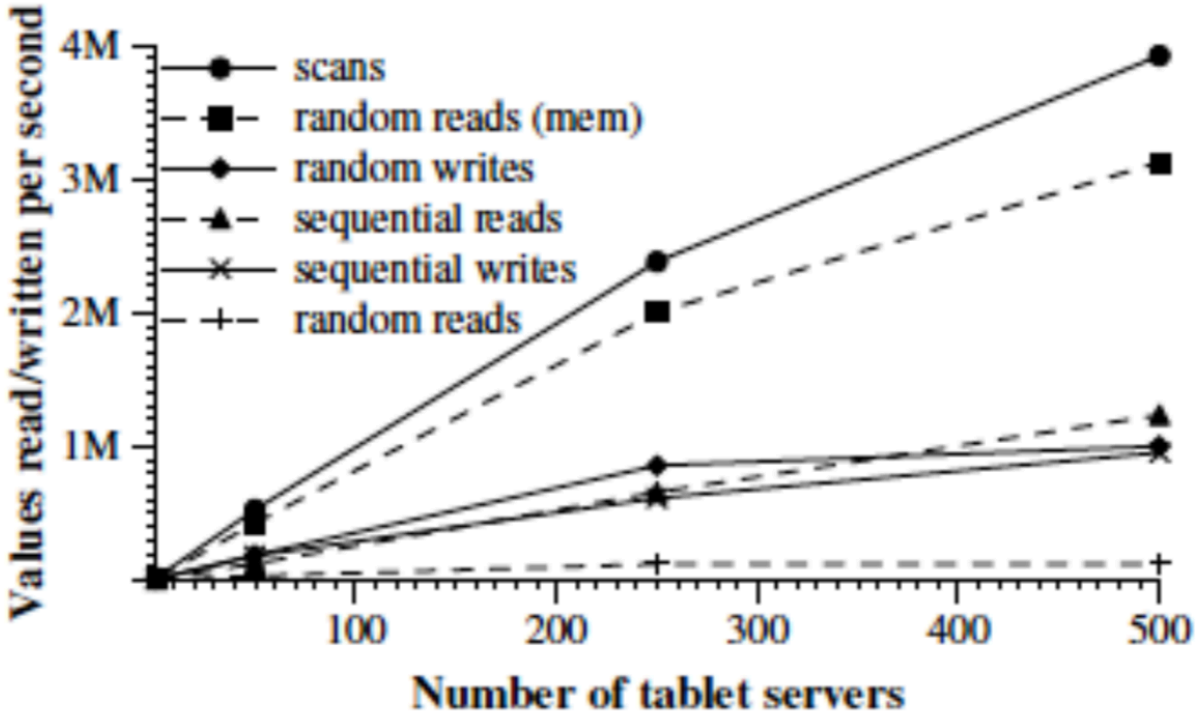
Performance Evaluation



Random read shows the worst performance. Fetching 64KB every 1000 bytes is expensive.

Performance Evaluation

Not linear, but scales well.



Conclusion

- Bigtable: highly scalable and available, without compromising performance.
- Flexibility for Google – designed using their own data model.
- Custom design gives Google the ability to remove or minimize bottlenecks.
- Related work:
  – Apache Hbase (open source)
  – Boxwood