

BigTable

# BigTable

In the early 2000s, Google had *way* more data than anybody else did

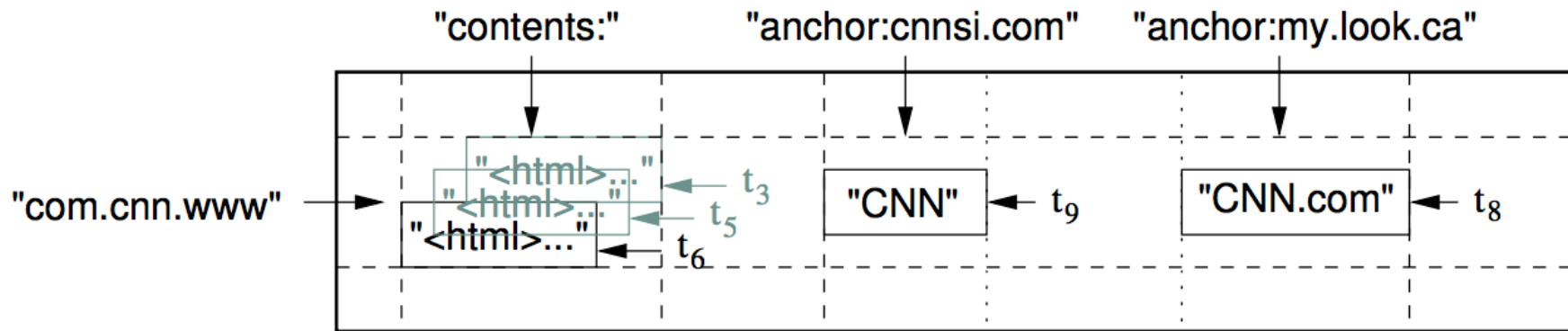
Traditional databases couldn't scale

Want something better than a filesystem (GFS)

BigTable optimized for:

- Lots of data, large infrastructure
- Relatively simple queries

Relies on Chubby, GFS



Uninterpreted strings in rows and columns

$(r : \text{string}) \rightarrow (c : \text{string}) \rightarrow (t : \text{int64}) \rightarrow \text{string}$

Mostly schema-less; column “families” for access

Data sorted by row name

- lexicographically close names likely to be nearby

Each piece of data versioned via timestamps

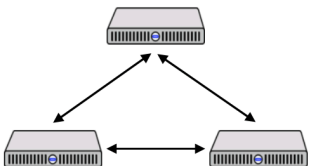
- Either user- or server-generated
- Control garbage-collection

# BigTable components

Client



Chubby



Master



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



GFS



# BigTable components

Client



Master



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



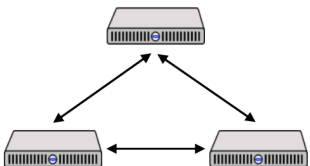
Tablet  
Server



Tablet  
Server



Chubby



GFS



# Tablets

a	<i>data</i>
b	<i>data</i>
c	<i>data</i>
d	<i>data</i>

Each table composed of one or more tablets

Starts at one, splits once it's big enough

- Split at row boundaries

Tablets ~100MB-200MB

# Tablets

a	<i>data</i>
b	<i>data</i>
c	<i>data</i>
d	<i>data</i>
e	<i>data</i>

Each table composed of one or more tablets

Starts at one, splits once it's big enough

- Split at row boundaries

Tablets ~100MB-200MB

# Tablets

a	<i>data</i>
b	<i>data</i>

c	<i>data</i>
d	<i>data</i>
e	<i>data</i>

Each table composed of one or more tablets

Starts at one, splits once it's big enough

- Split at row boundaries

Tablets ~100MB-200MB



# Tablets

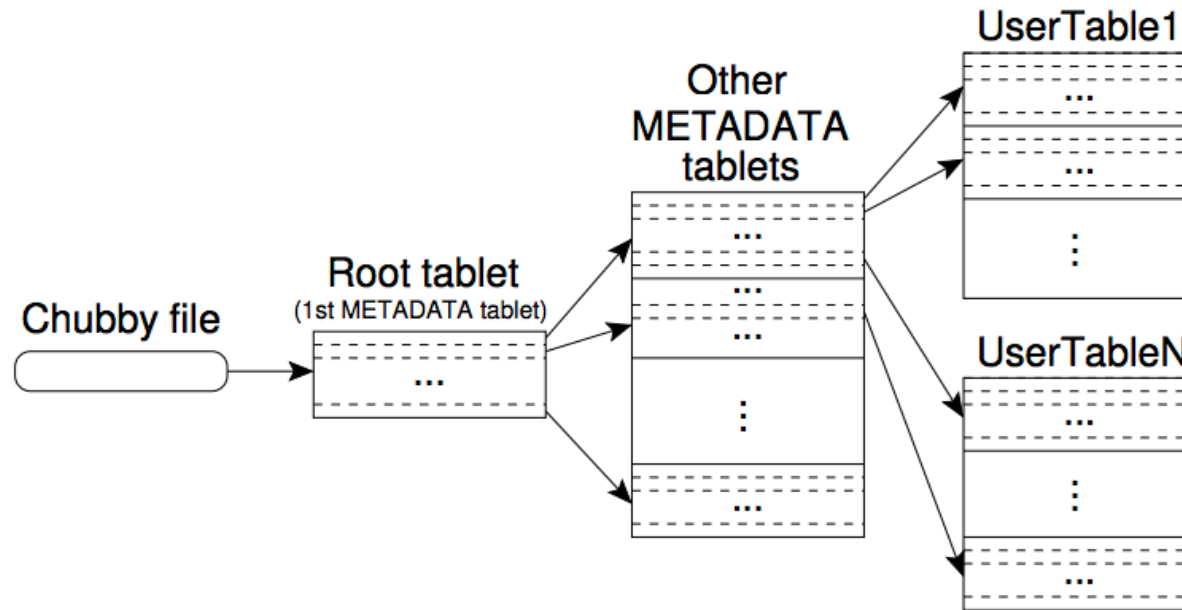
A tablet is indexed by its range of keys

- <START> - “c”
- “c” - <END>

Each tablet lives on at most one tablet server

Master coordinates assignments of tablets to servers

# Tablets



Tablet locations stored in METADATA table

Root tablet stores locations of METADATA tablets

Root tablet location stored in Chubby

# Tablet serving

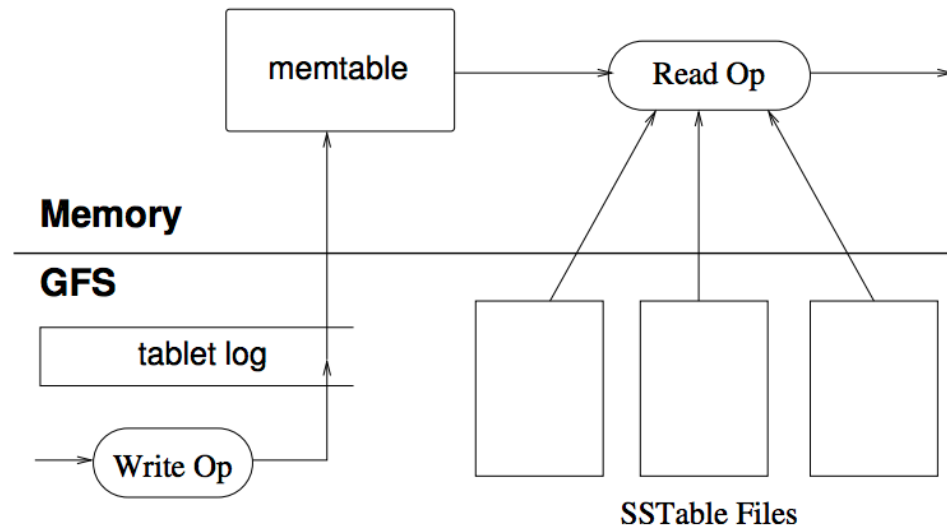
Tablet data persisted to GFS

- GFS writes replicated to 3 nodes
- One of these nodes should be the tablet server!

Three important data structures:

- memtable: in-memory map
- SSTable: immutable, on-disk map
- Commit log: operation log used for recovery

# Tablet serving



Writes go to the commit log, then to the memtable

Reads see a merged view of memtable + SSTables

- Data could be in memtable or on disk

# Compaction and compression

Memtables spilled to disk once they grow too big

- “minor compaction”: converted to SSTable

Periodically, all SSTables for a tablet compacted

- “major compaction”: many SSTables -> one

Compression: each block of an SSTable compressed

- Can get enormous ratios with text data
- Locality helps—similar web pages in same block

# Bloom filters

Goal: efficient test for set membership:  $\text{member}(\text{key}) \rightarrow \text{true/false}$

false  $\implies$  definitely not in the set

true  $\implies$  probably is in the set

Generally supports adding elements but not removing them

Basic version:  $m$  bit positions,  $k$  hash functions

For insert: compute  $k$  bit locations, set to 1

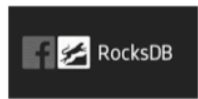
For lookup: compute  $k$  locations, check for 1

BigTable: avoid reading SSTables for elements that are not present; saves many seeks

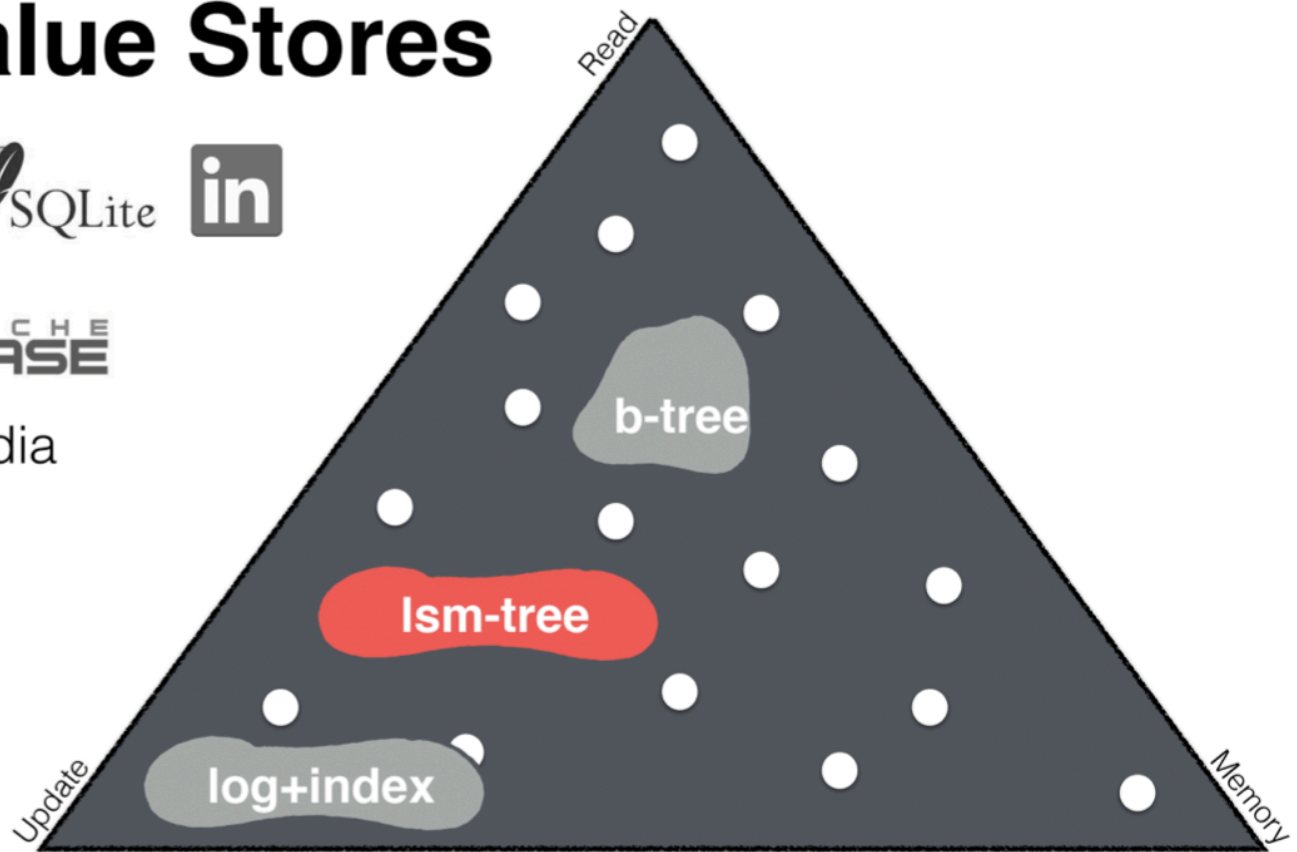
# KV Store Design Options

LSM: Log structured merge trees (1996)

## NoSQL Key-value Stores



machine learning   social media  
smart homes   web browsers  
phones   web-based apps  
security   health devices  
graphs   analytics

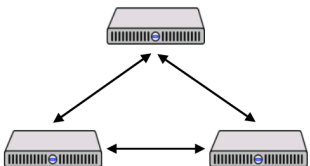


# BigTable components

Client



Chubby



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Master



GFS





# Master

Tracks tablet servers (using Chubby)

Assigns tablets to servers

Handles tablet server failures

# Master startup

- Acquire master lock in Chubby
- Find live tablet servers (each tablet server writes its identity to a directory in Chubby)
- Communicate with live servers to find out who has which tablet
- Scan METADATA tablets to find unassigned tablets

# Master operation

Detect tablet server failures

- Assign tablets to other servers

Merge tablets (if they fall below a size threshold)

Handle split tablets

- Splits initiated by tablet servers
- Master responsible for assigning new tablet

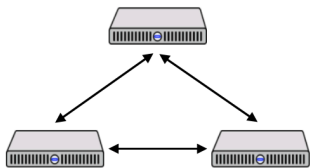
Clients never read from master

# BigTable components

Client



Chubby



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



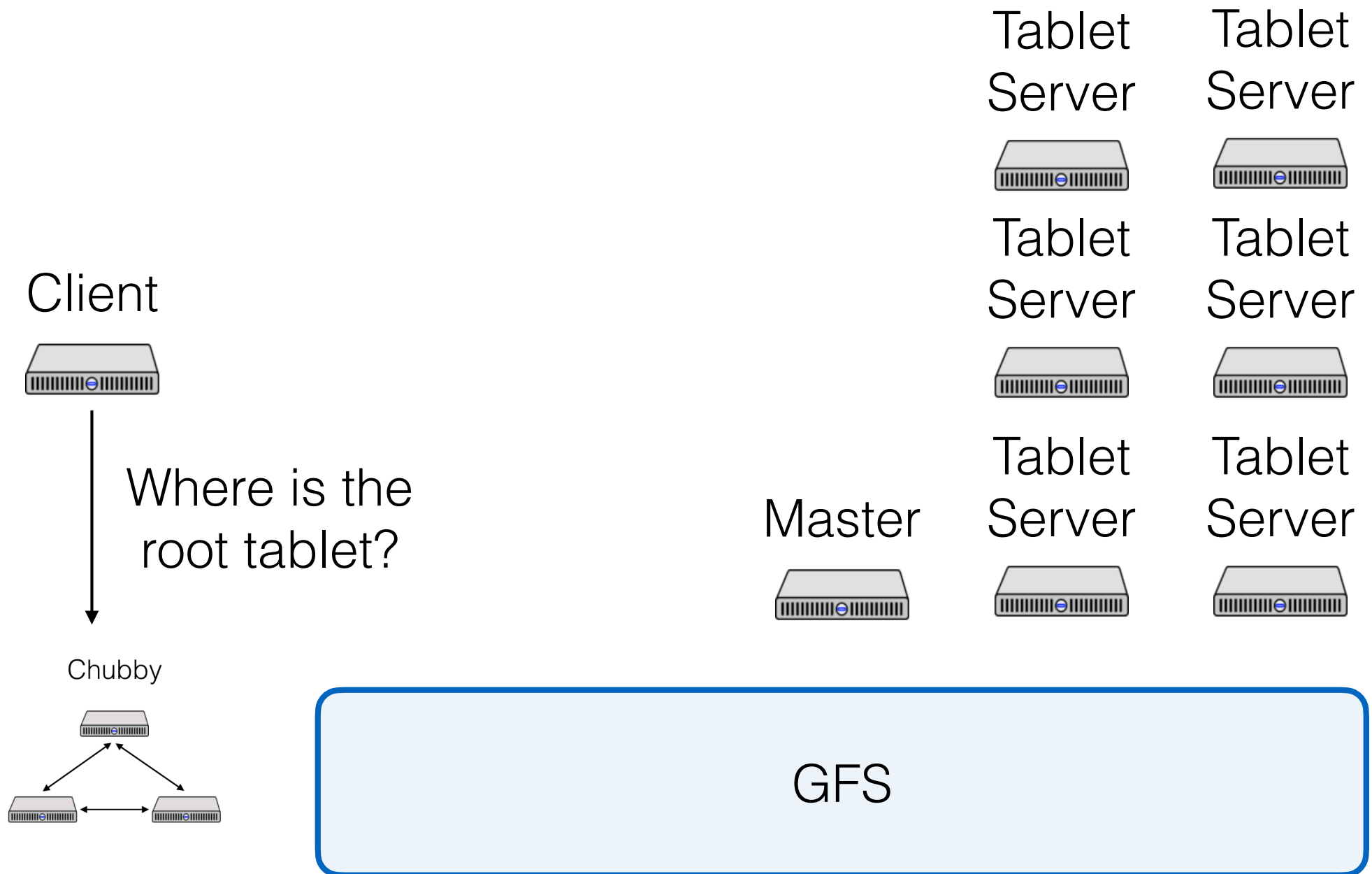
Master



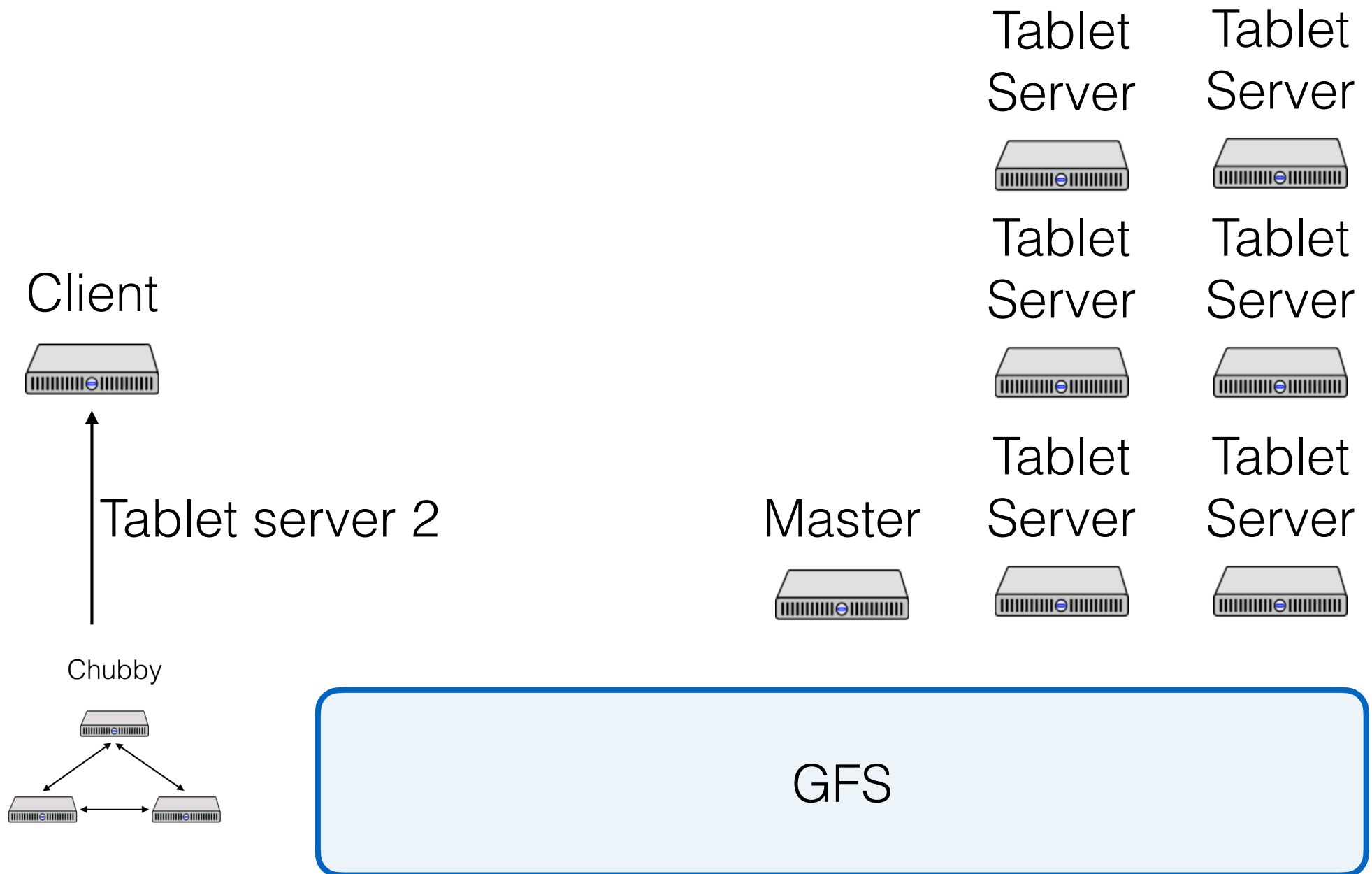
GFS



# BigTable components



# BigTable components

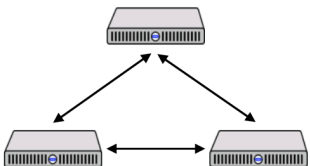


# BigTable components

Client



Chubby



Master



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



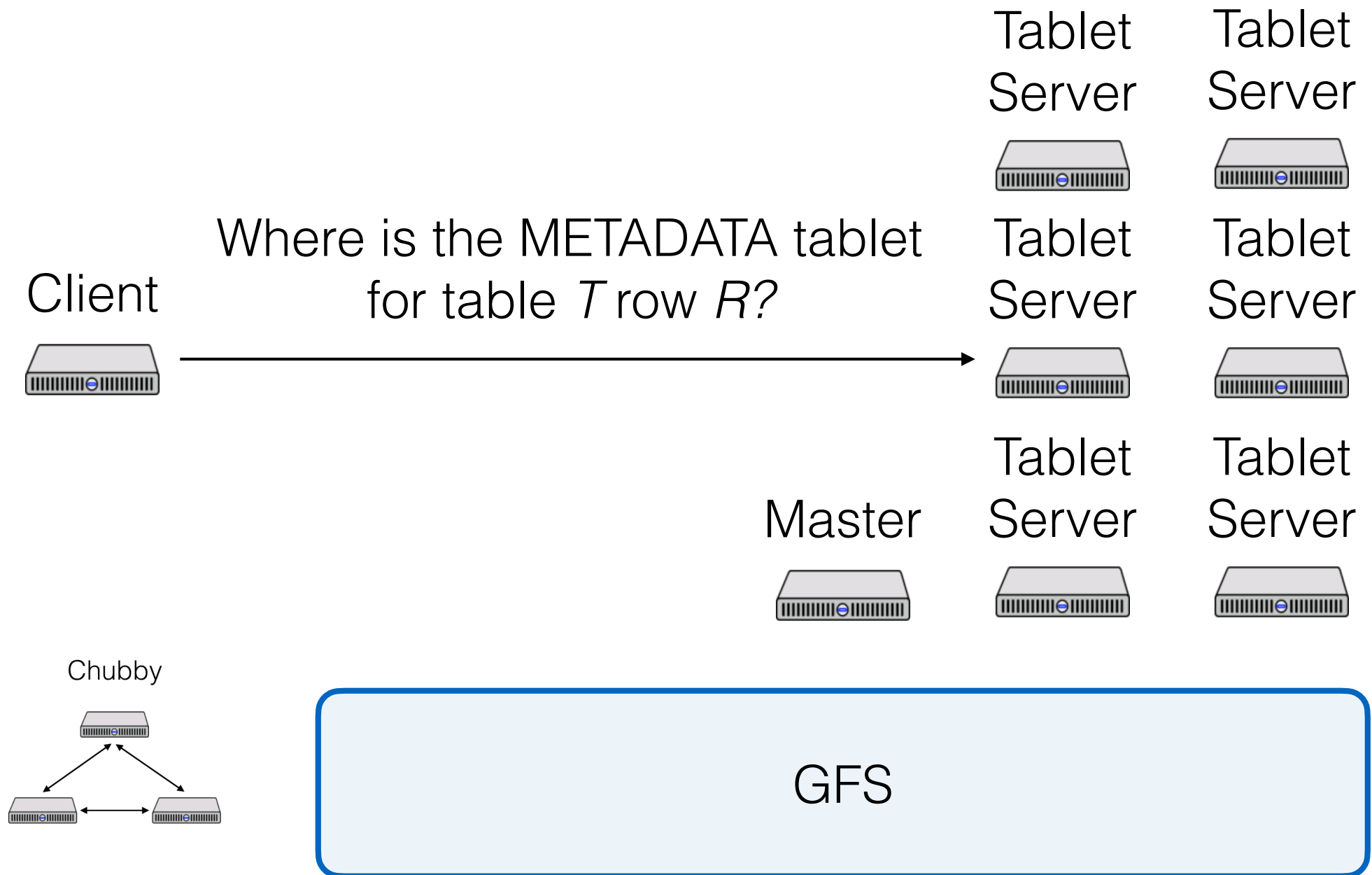
Tablet  
Server



GFS

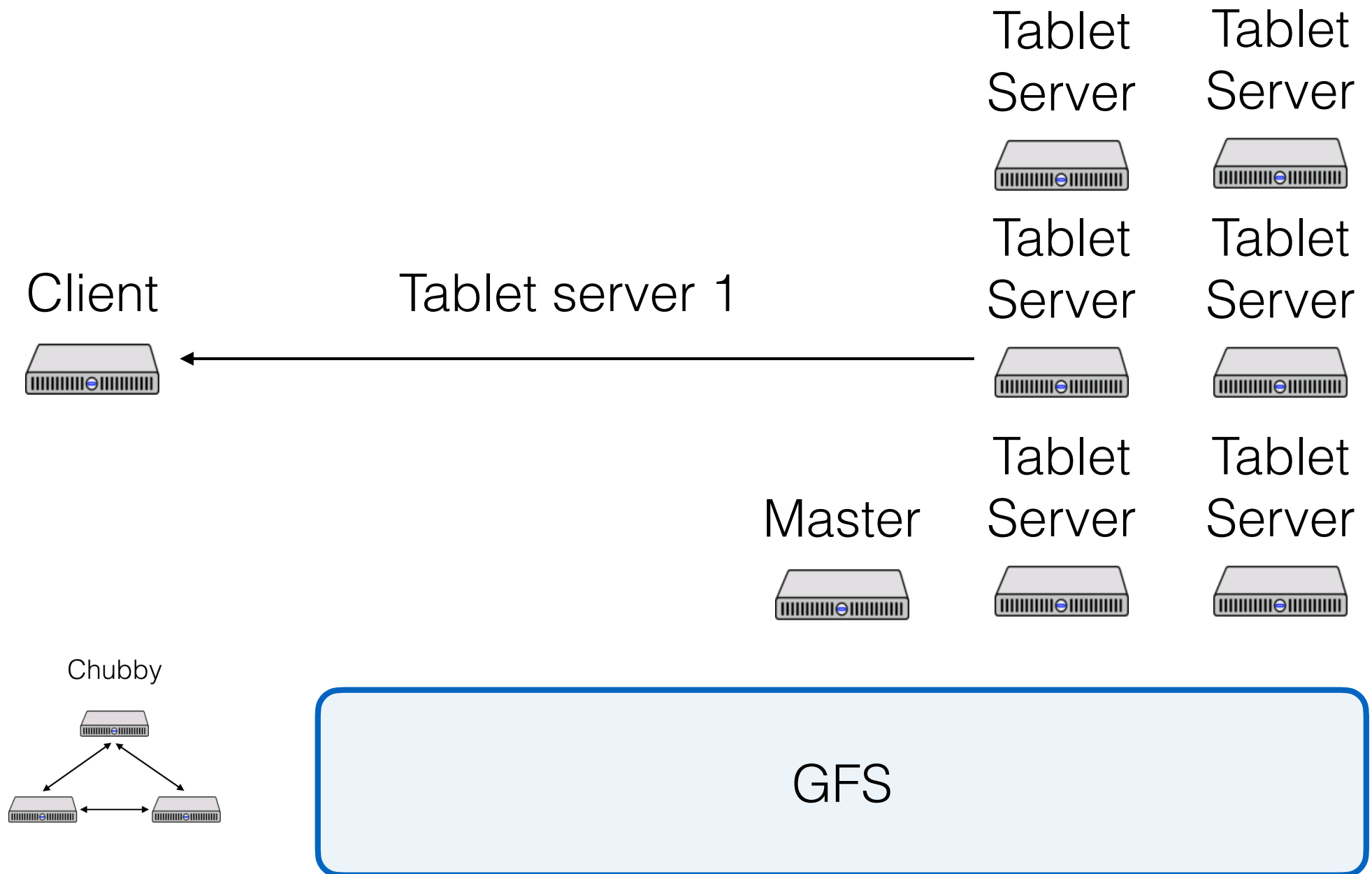


# BigTable components





# BigTable components

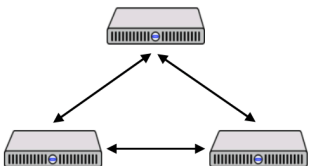


# BigTable components

Client



Chubby



Master



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



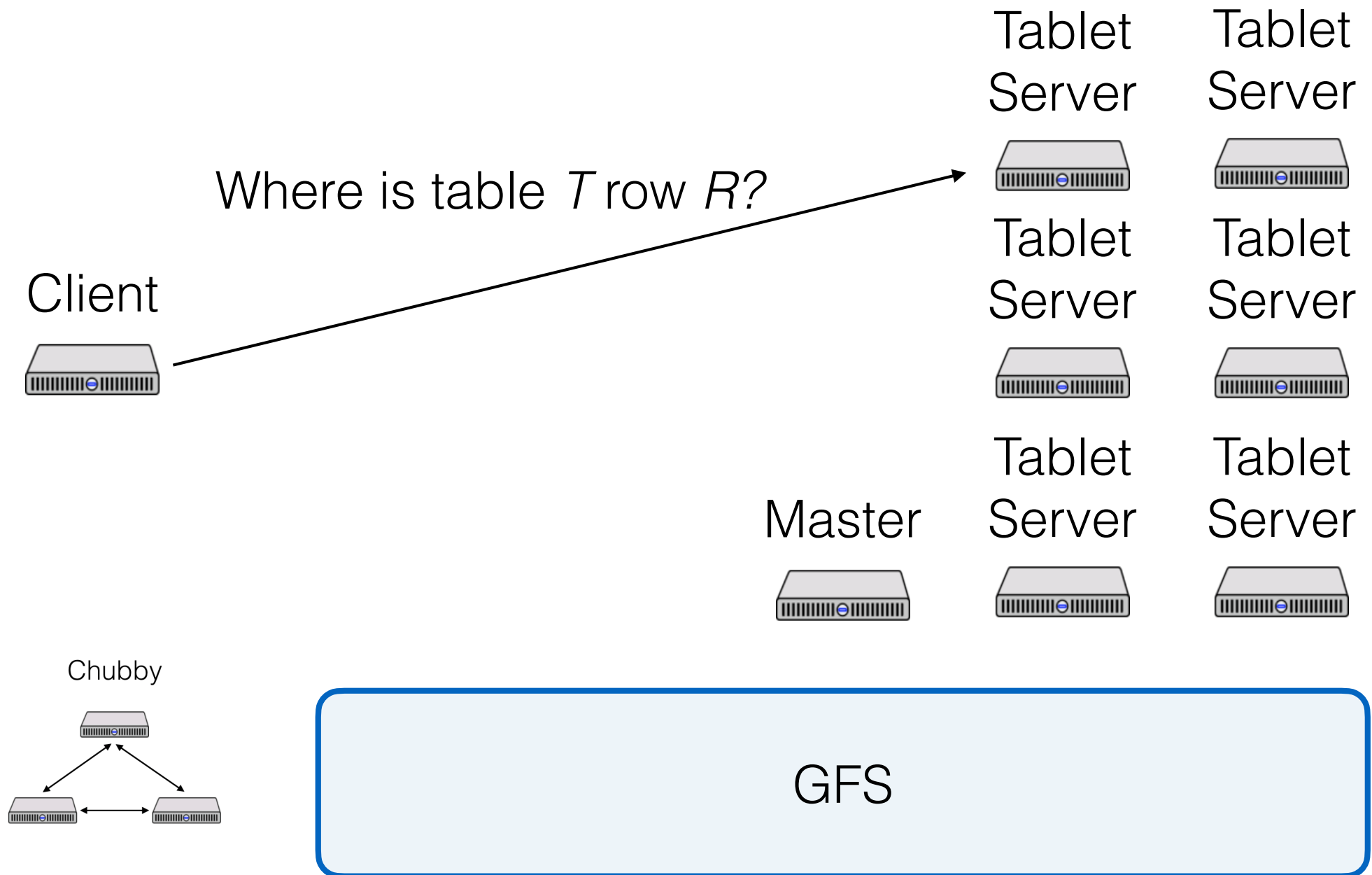
Tablet  
Server



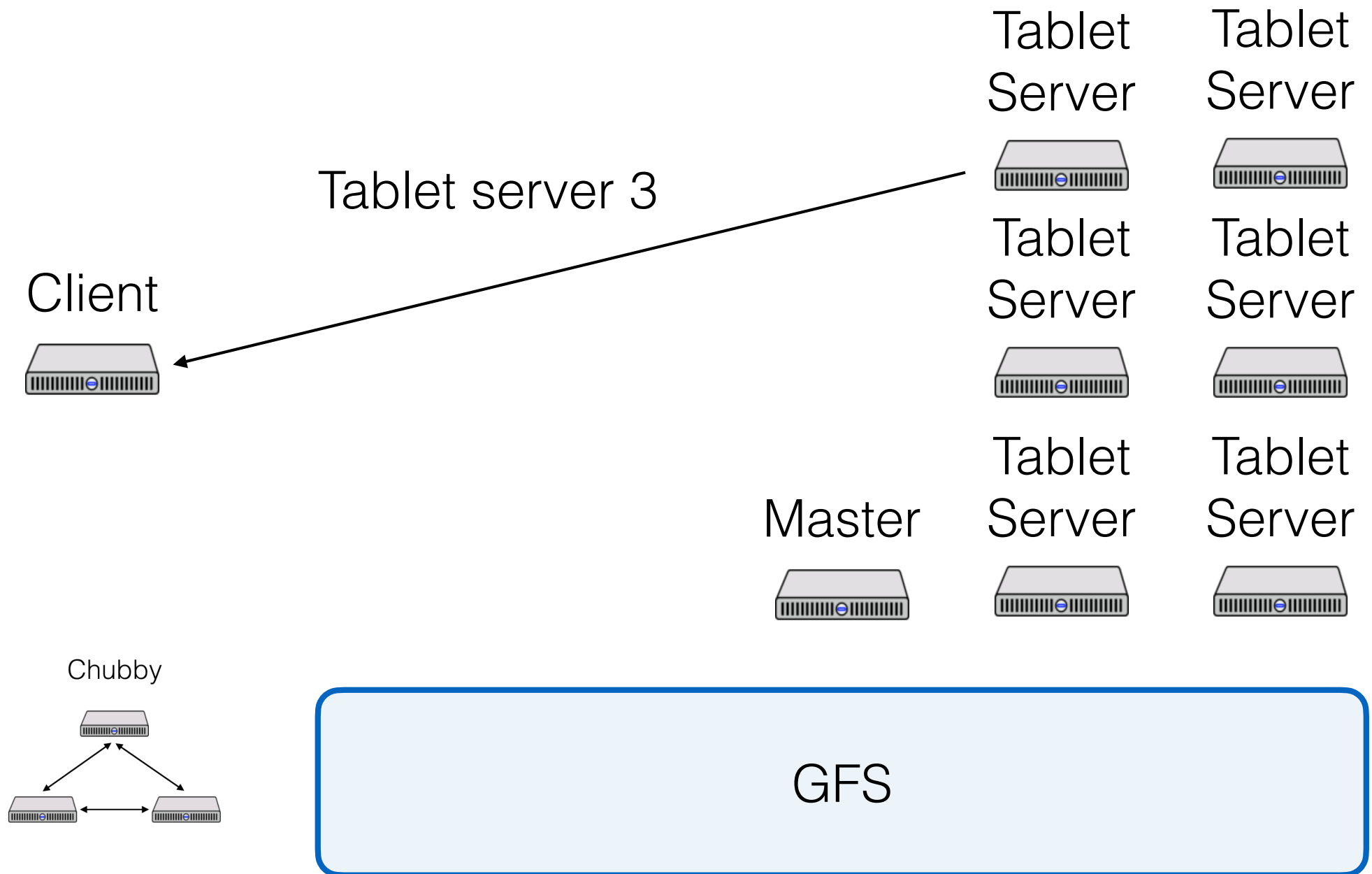
GFS



# BigTable components



# BigTable components

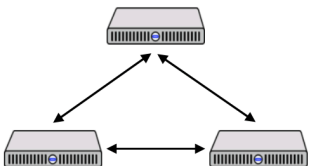


# BigTable components

Client



Chubby



Master



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



Tablet  
Server



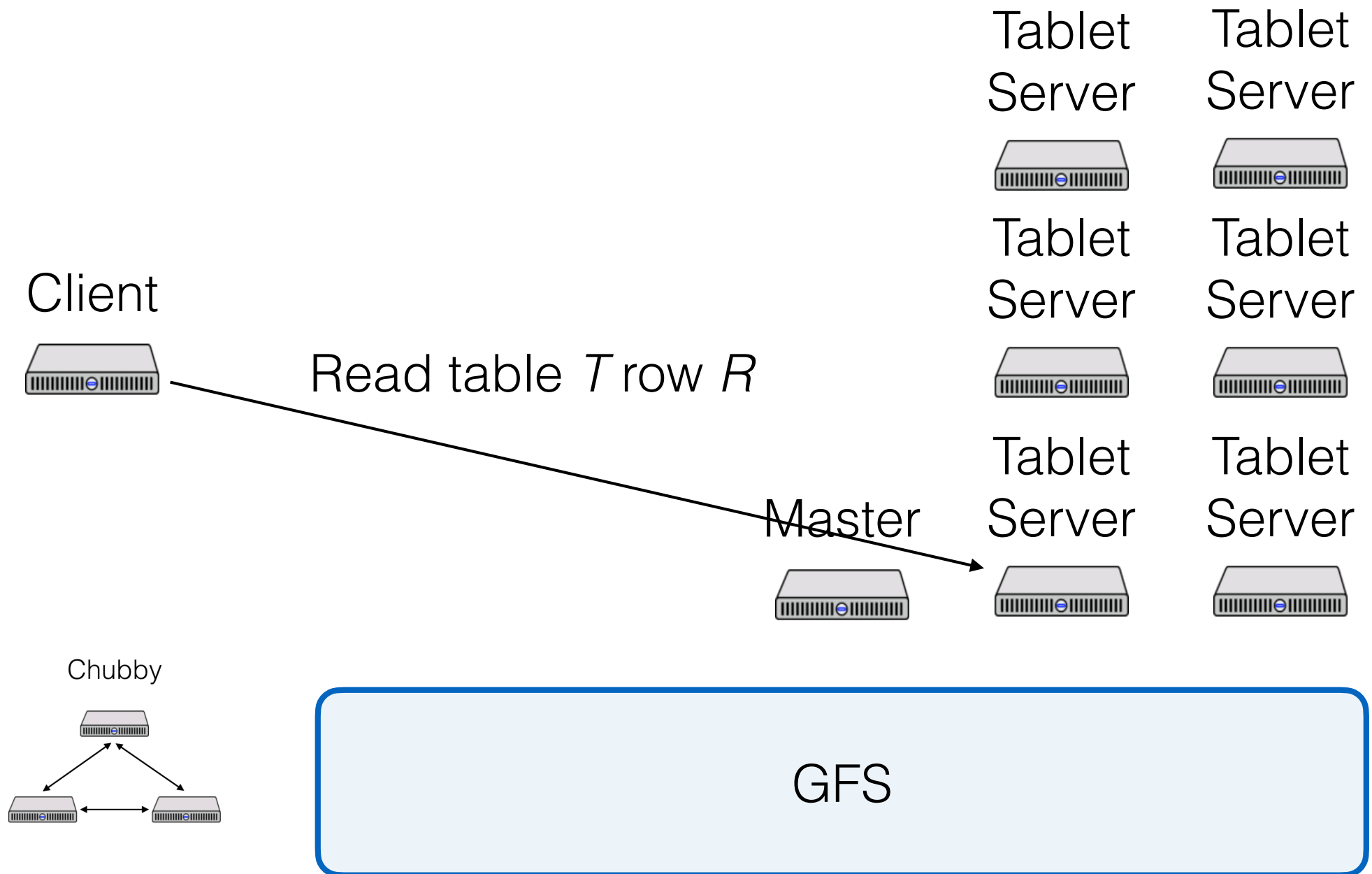
Tablet  
Server



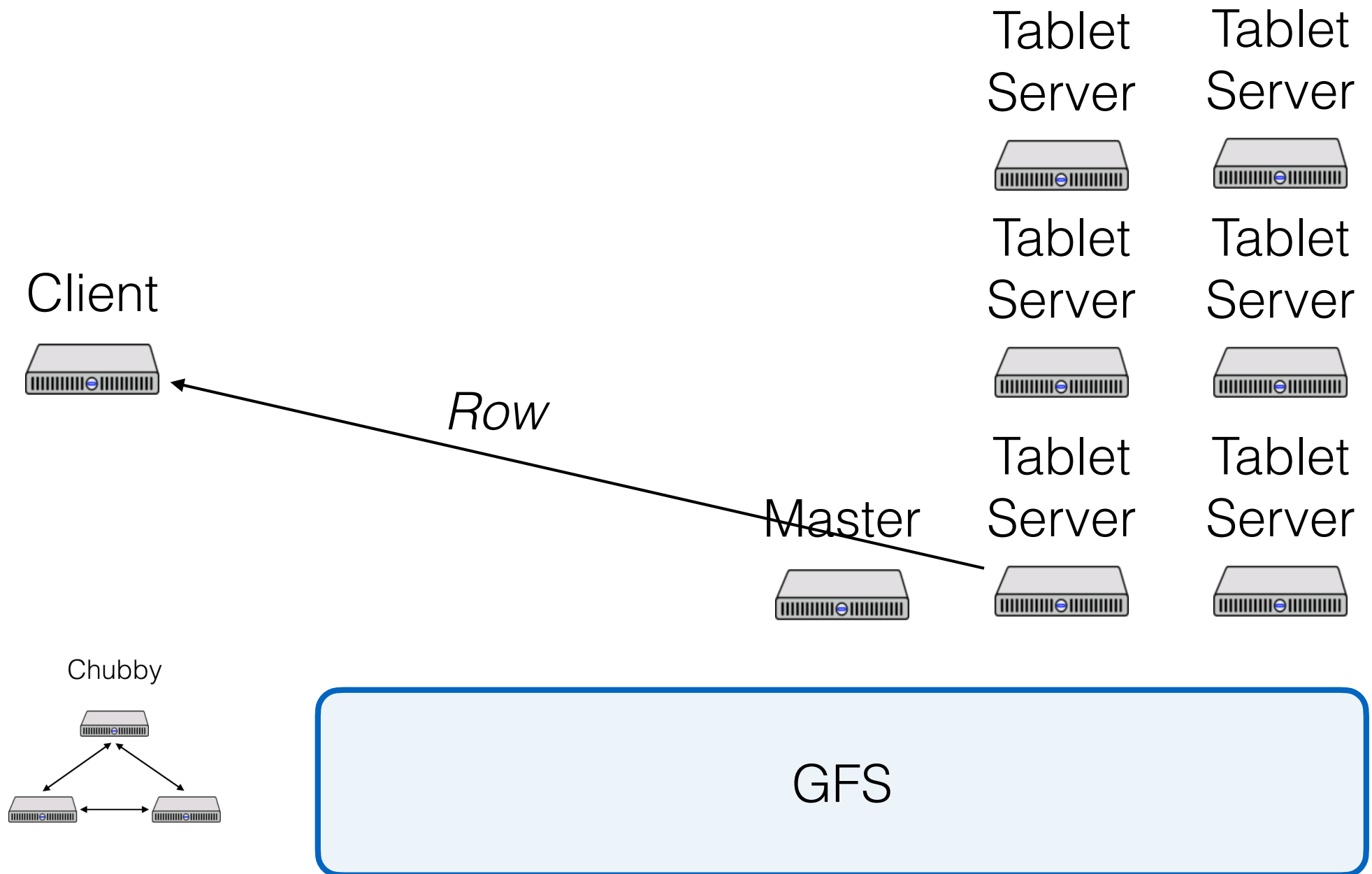
GFS



# BigTable components



# BigTable components



# Optimizations

Clients cache tablet locations

Tablet servers only respond if Chubby session active, so this is safe

Locality groups

Put column families that are infrequently accessed together in separate SSTables

Smart caching on tablet servers

Bloom filters on SSTables