Bigtable: A Distributed Storage System for Structured Data

Author : Fay Chang, Jeffrey Dean, et al.
Venue : 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)
Year : 2006

Cassandra - A Decentralized Structured Storage System

Author : Avinash Lakshman and Prashant Malik
Venue : ACM SIGOPS Operating System Review
Year : 2010

Presented by: Erika Siregar

CS 834 - Introduction to Information Retrieval

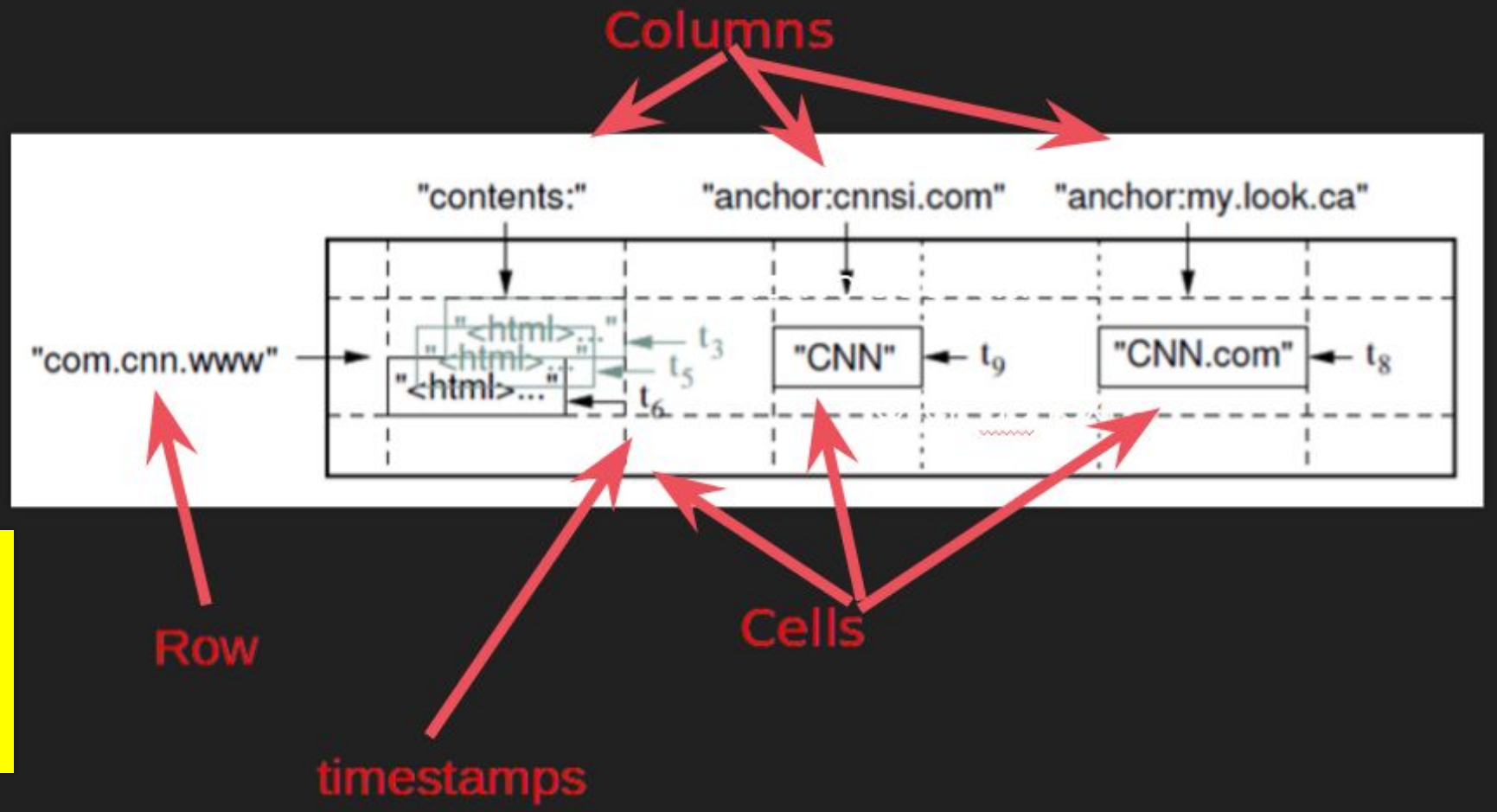
Fall 2016

Old Dominion University

September 15, 2016

What is Bigtable?

- A distributed multidimensional sorted map
- Data is indexed using **rows**, **columns**, and **timestamps**
  <Row: string, Column: string, Timestamp: Int64> --> String



Bigtable does not support a full relational data model,
but:
provides clients with a dynamic control over data layout and format

Data Model : Row

	id	.....
Tablet 1	15000	.....
	....	.....
	20000	.....
Tablet 2	20001	.....
	....	.....
	25000	.....

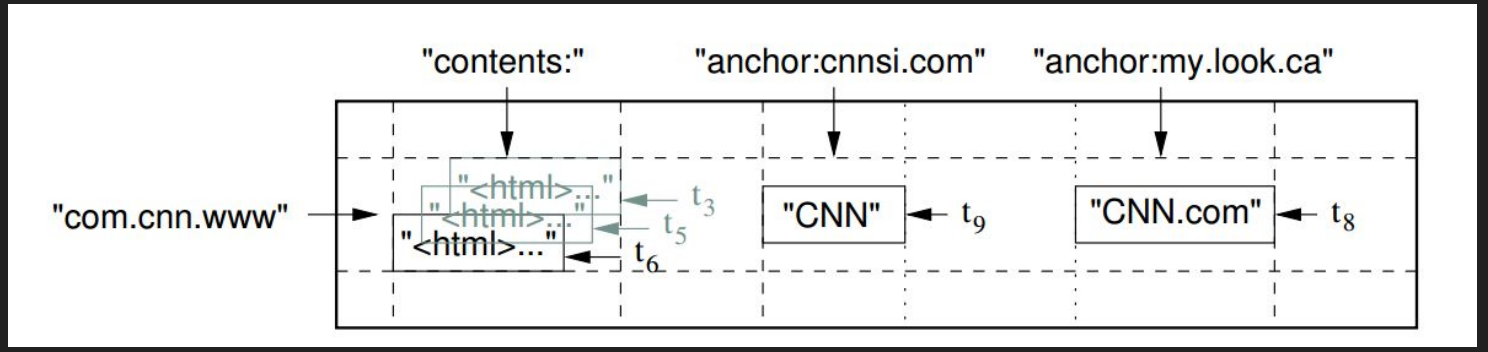
- **Row keys** are arbitrary strings
- **Row** is the unit of transactional consistency
  - Every read or write of **data under a single row is atomic**
- **Data** is maintained in lexicographical order by **row key**
- **Rows with consecutive keys** (Row Range) are grouped together as **"tablets"**
  - Unit of **distribution** and **load-balancing**
  - For example, in **Webtable** : pages in the same domain are grouped together into contiguous rows by reversing the hostname components of the URLs.
    - Data for maps.google.com/index.html --> stored under: com.google.maps/index.html
    - A more efficient host and domain analysis

Data Model : Column

- **Column** keys are grouped into sets called **"column families" --> basic** unit of **access control**
  - Privacy setting --> add, read, view, etc.
  - All columns in a family are in the same type
  - Column key is named using the following syntax: **family:qualifier**
    - **Family names: printable**
    - **Qualifier : arbitrary strings**
- **A table may have unbound number of columns, but the column families:**
  - **should be small in small numbers (± hundreds)**
  - **Rarely change**

Data Model : Timestamp

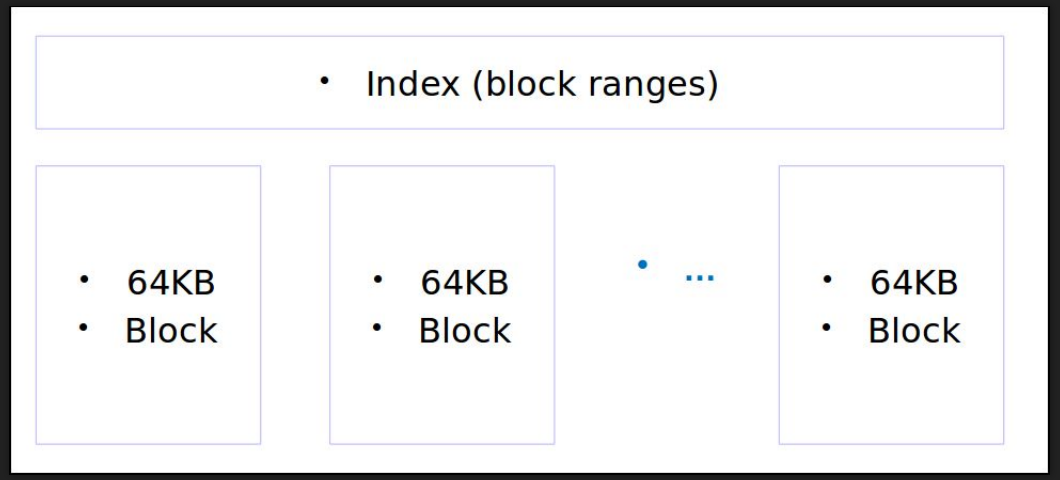
- Versioning --> each cell contain multiple versions of the same data.
- Each version is indexed by timestamp
- Can be assigned by :
  - Bigtable --> real time in microseconds.
  - client applications
- Data are stored in decreasing timestamp order --> from the newest to the oldest.
- Garbage-collection mechanism:
  - Keep the latest n updates
  - Keep the updates since time t



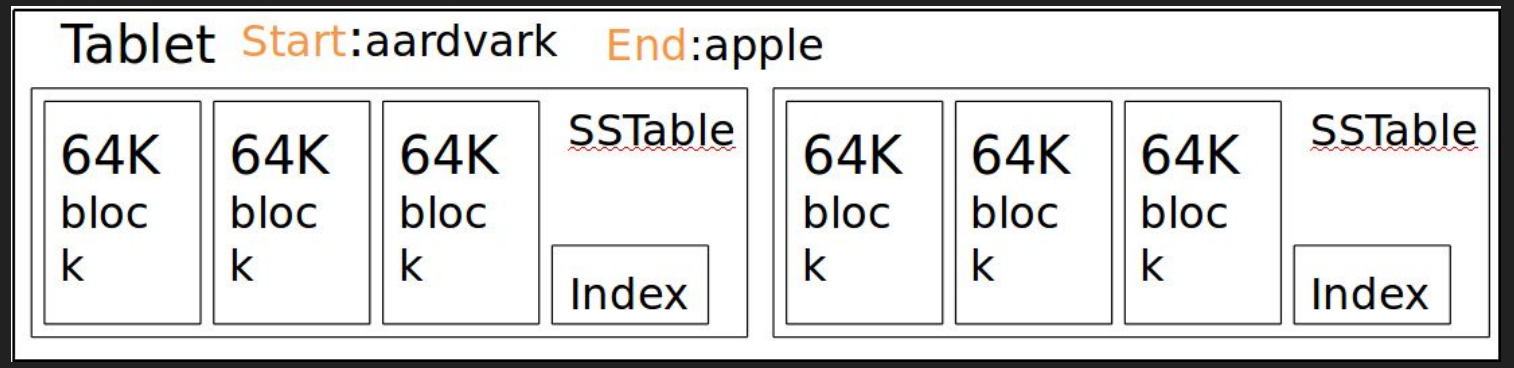
Building Blocks

- Use **Google File System (GFS)** to **store log and data files**.
  - Using Google **SSTable** file format.
- A **cluster management system**
  - Scheduling jobs
  - Managing resources and shared machines
  - Dealing with machine failures
  - Monitoring machine status
- **Chubby**
  - Lock Service
  - Ensure that there is **at most one active master** at any time
  - **Discover tablet servers** and **finalize tablet server deaths**;

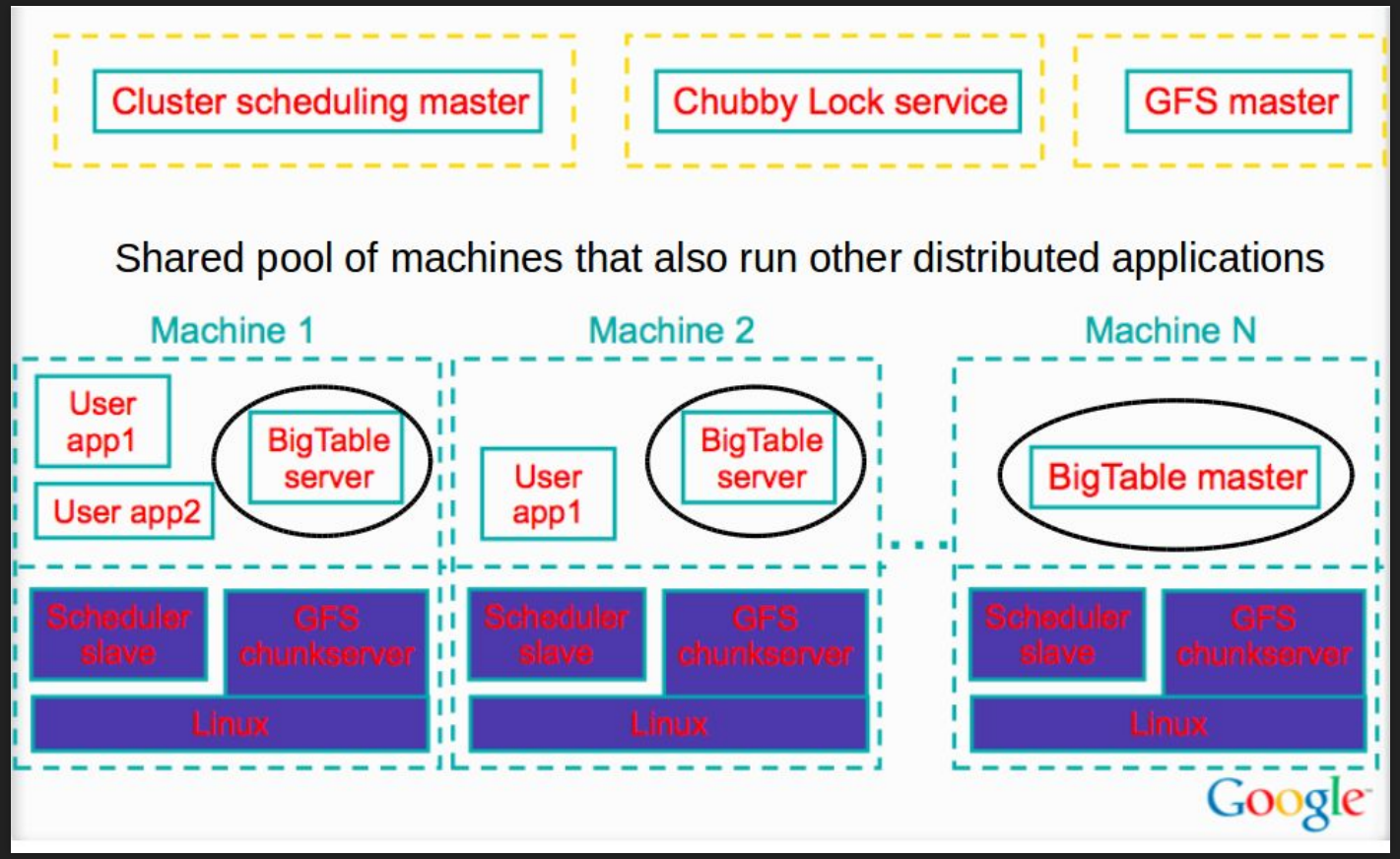
SSTable



- **Immutable**
- contains a **sequence of blocks**
- A block index is used to **locate blocks**
- This index is loaded into memory when the table is open.



Bigtable Illustration



3 major components:

1. A **library** that is linked to every client
2. One **master server**
3. Many **tablet servers**

Master:

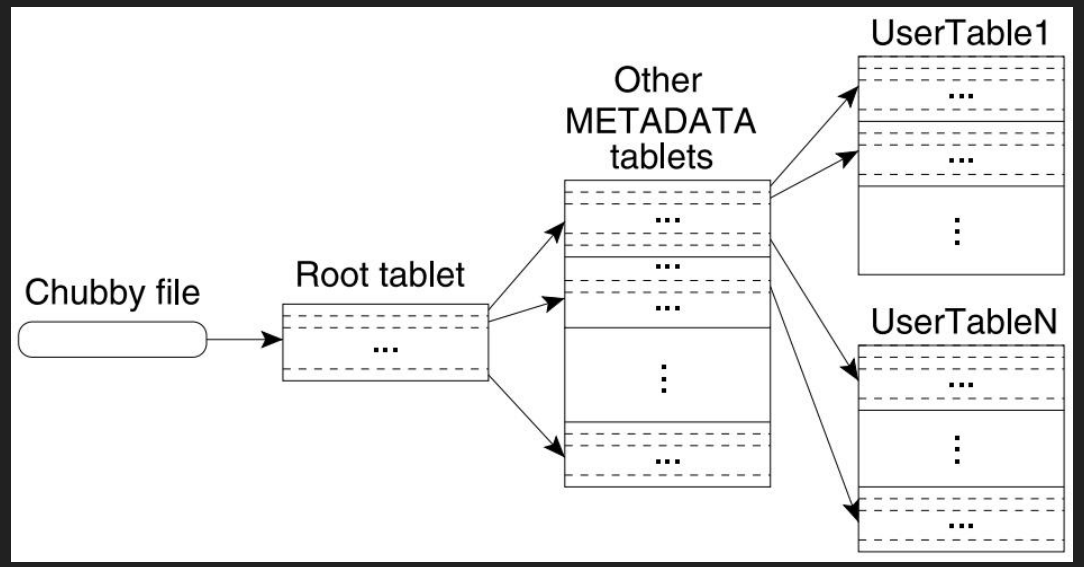
1. **assigning** tablets to tablet server
2. **detecting** the addition and expiration of tablet server
3. **balancing** tablet-server load --> split tablets that have grown too large
4. **garbage collection** of files in GFS
5. handling **schema changes**

Clients never communicate with the master

Tablet Location

given a row, how do clients find the right machine?

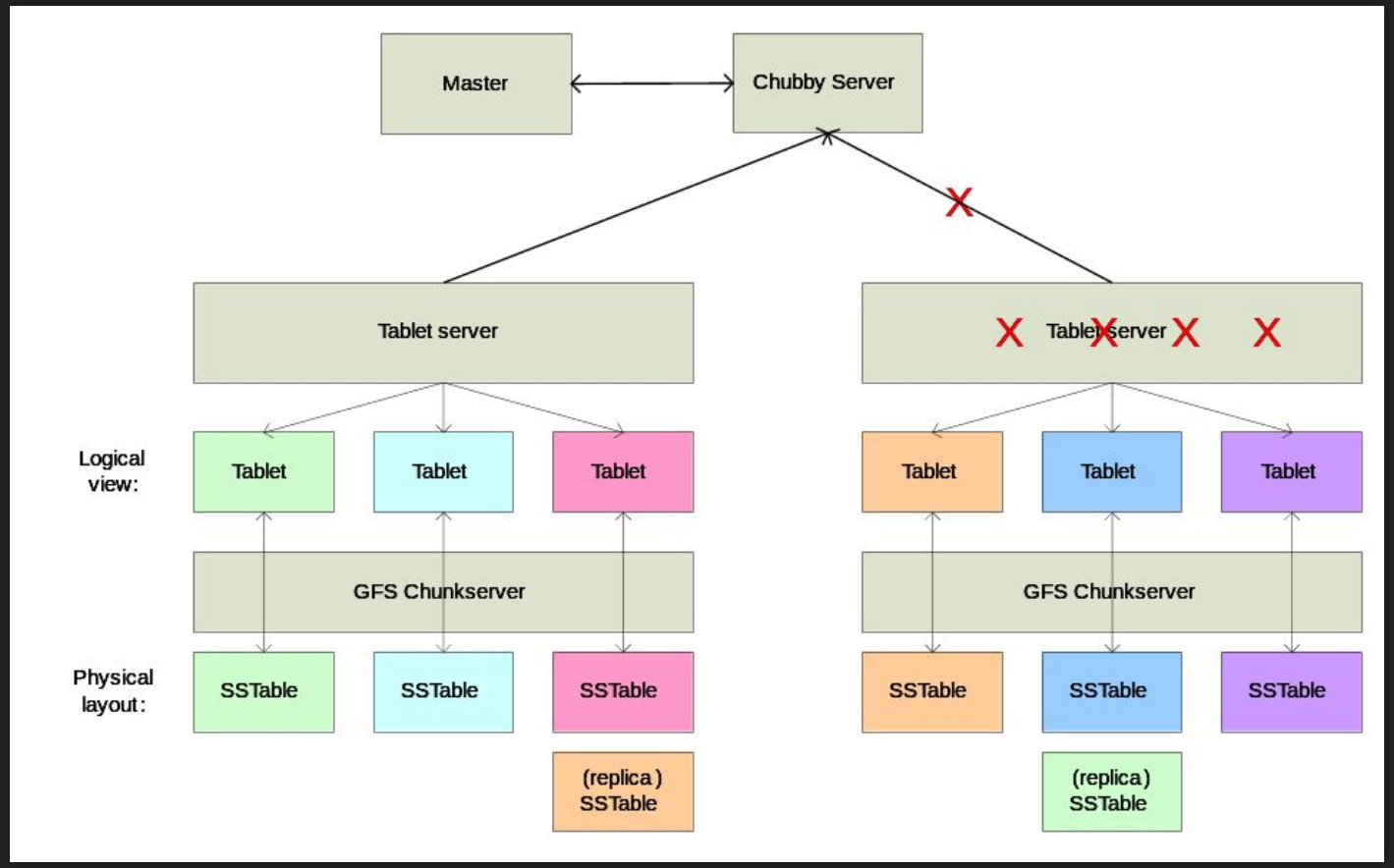
- Use a **three-level** hierarchy analogous :
  1. A file stored in Chubby --> contains the **location of the root tablet**
  2. The root tablet contains the **locations of all of the tablets** of a special METADATA table
  3. Each METADATA tablet contains the **location of a set of user tablets**



Tablet Assignment

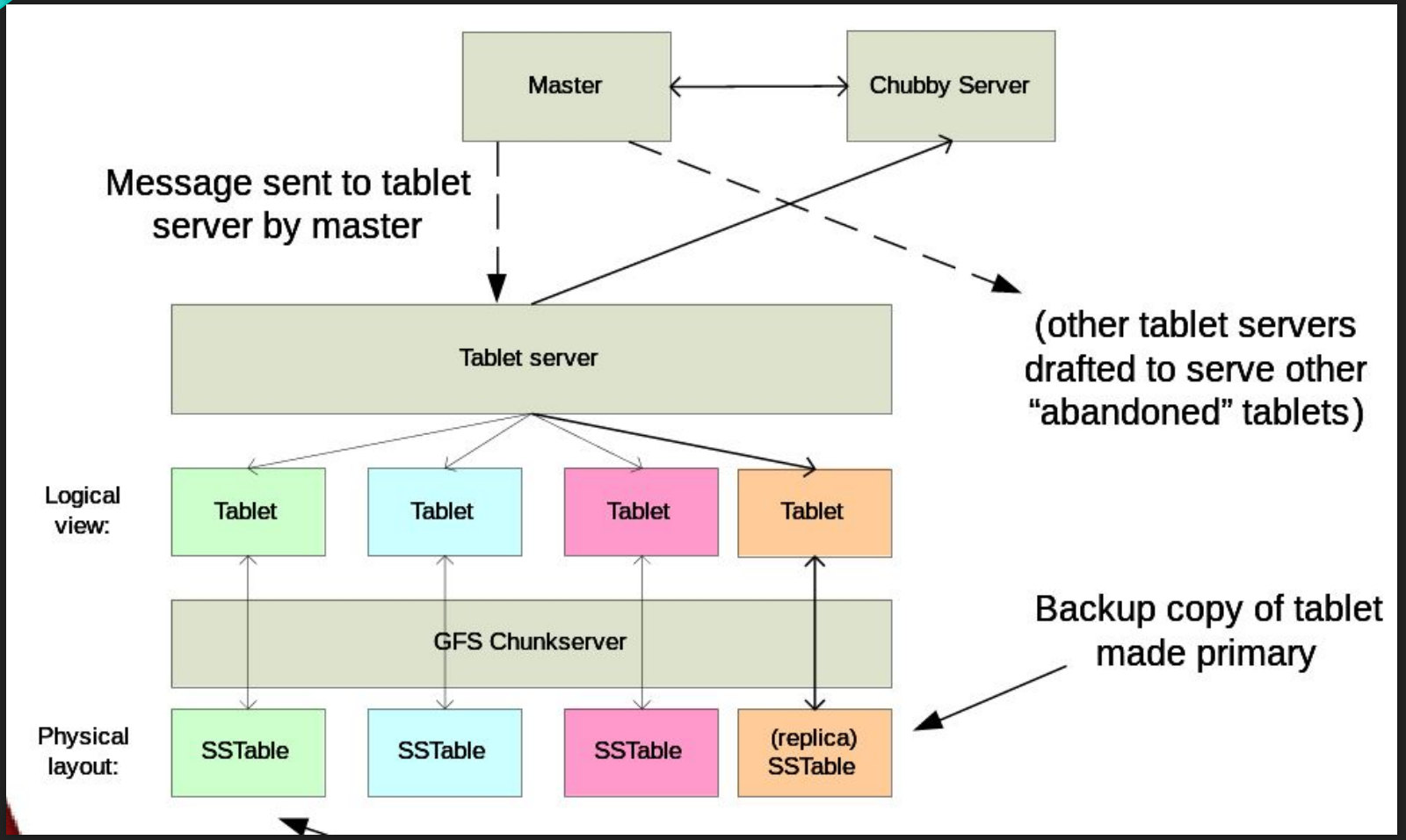
- **Each tablet** is assigned to **one tablet server** at a time
- The master keeps track of:
  - The set of **live tablet servers**
  - The **current assignment of tablets** to tablet servers
- How to **keep track of tablet server**?
  - Tablet server starts --> creates and acquires an **exclusive lock** in a specific **Chubby directory**
  - The master **monitors** this chubby directory to **discover tablet servers**.

Tablet Server Failure



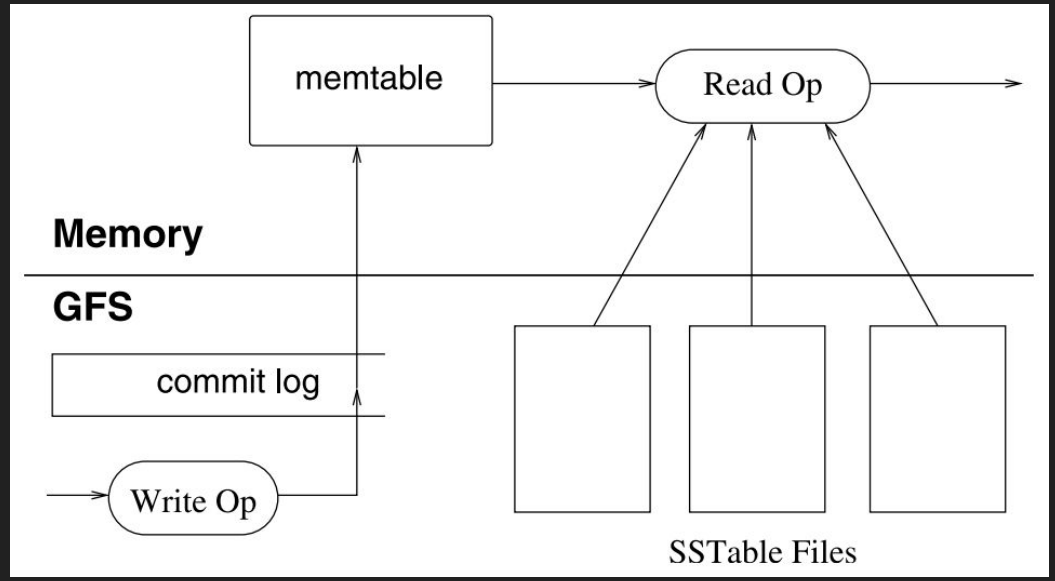
- When network failure happens:
  - Tablet server loses its lock and stops serving its tablets.
  - A tablet server attempts to **reacquire an exclusive lock**
  - If the lock is no longer exists --> tablet server kills itself.

Tablet Server Failure

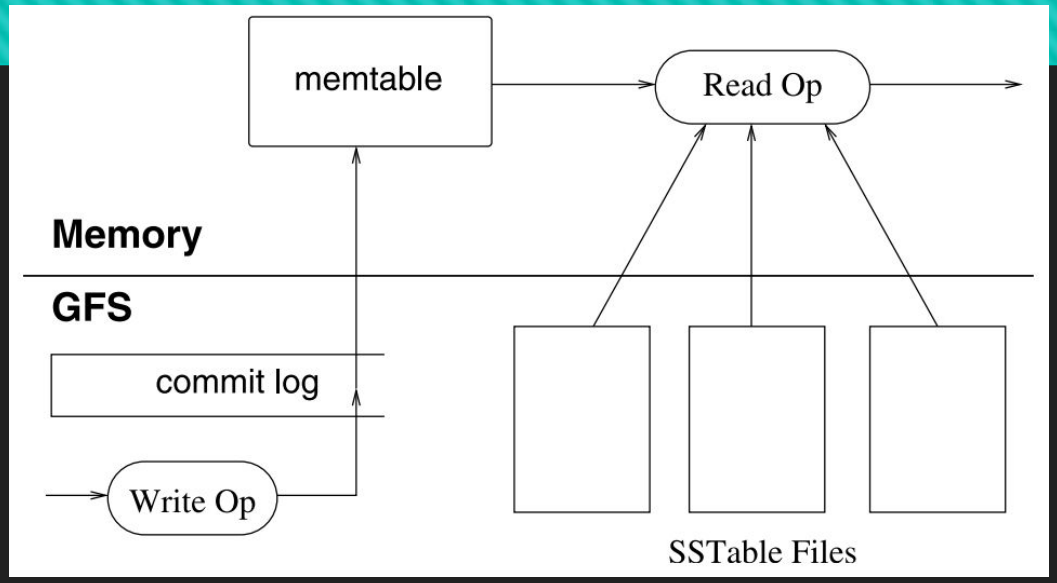


Tablet Serving

- The **persistent** state of a tablet is stored in **GFS**
- The **recently committed** ones are stored in memory in a sorted buffer called a **memtable**
- **Older updates** are stored in a sequence of **SSTables**
- Tablet Recovery:
  - **Read metadata** containing SSTABLES and redo points
  - **Apply** redo points



Compaction



- **Minor** Compaction
  - Size of memtable **increases**
  - Convert the **memtable** into an **SSTable**
  - Reduce **memory usage** and **log traffic**
- **Merging** Compaction
  - Number of SSTable **increases**
  - **Bound the number** of SSTable
    - Reads the **contents of a few SSTables and the memtable**, and writes out a **new SSTable**.
- **Major** Compaction
  - Rewrites all SSTables into one new SSTable
  - Clean deletes

What is Cassandra?

Bigtable has inspired the creation of another distributed storage system called "Cassandra"

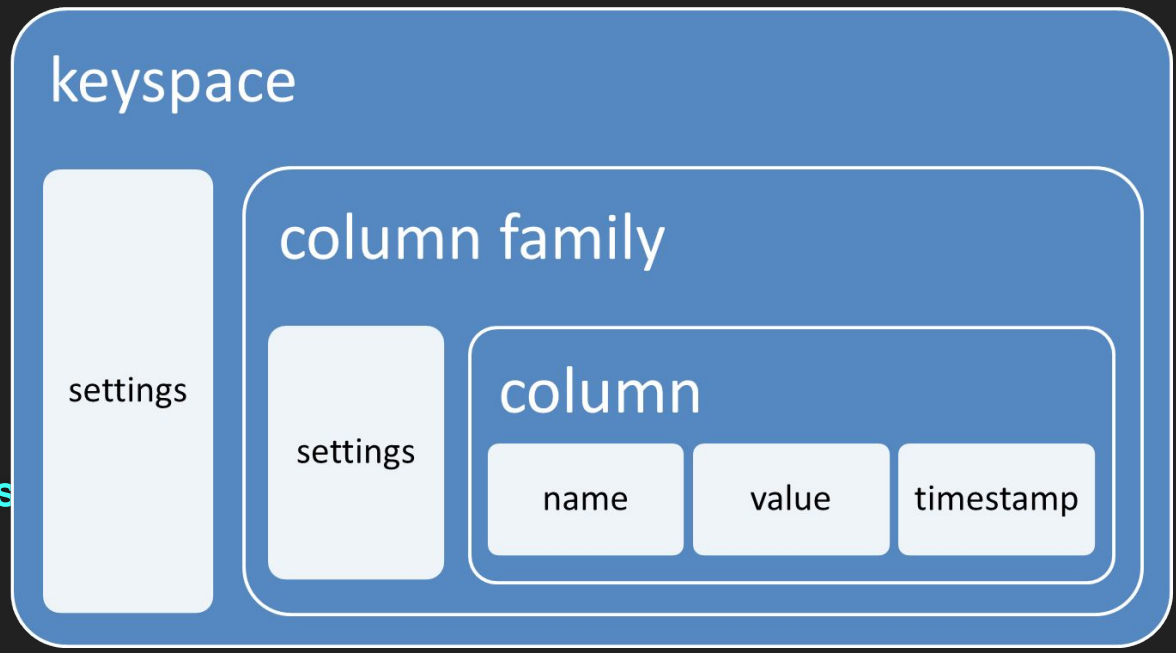
- Distributed storage system
- For managing **very large amounts of structured data**
- Across many **commodity servers** → Designed to run on **cheap commodity hardware**
- Providing highly available service with **no single point of failure**
- run on top of hundreds nodes, spread across different data centers

Background Story

- Facebook runs the **largest social networking platform**
- Strict requirements on **performance, reliability, efficiency, and scalability**
- Serves **hundreds of millions users** at peak times
- Using **tens of thousands of servers** located in **many data centers** around the world
- There are always a small but significant number of **server and network components that are failing** at any given time
- So, Facebook developed **CASSANDRA**

Data Model

- Cassandra has a **similar data model** with Google Bigtable:
  - A **table** in Cassandra is a distributed **multi dimensional map** indexed by a key
  - The **row key** in a table is a string with no size restrictions
  - Every operation under a single row key is **atomic**
    - no matter how many columns are being read or written into
  - Columns are grouped together into sets called **column families**



Column Families

- Cassandra exposes two kinds of columns families: **Simple** and **Super** column families
- **Simple** column families are very **similar with Bigtable** column families
- **Super** column families can be visualized as **nested column family**
- Any column within **a column family** is accessed using the convention column **family : column**
- Any column within **a column family that is of type super** is accessed using the convention **column family : super column : column**

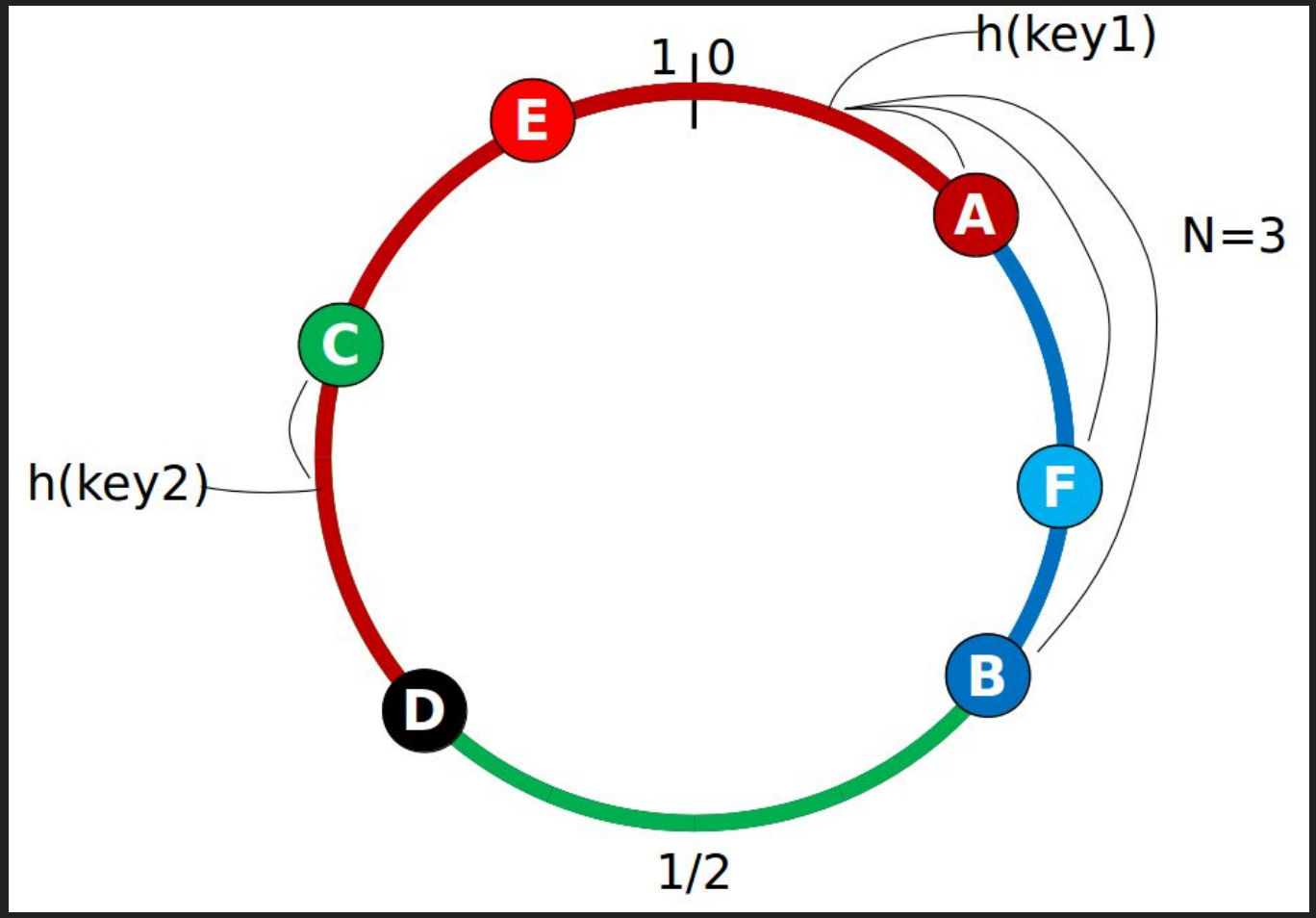
System Architecture

- **Core** distributed systems **techniques**:
  - Partitioning,
  - Replication,
  - Membership,
  - Failure handling
  - Scaling

Partitioning & Replication

Partitioning:
- Split the data into several different nodes
- Nodes are structured in Ring Topology

Replication:
- replicate the data into several nodes
- How many nodes?
  depends on the replication factor (N)



Membership

- Membership **defines the state of a nodes** in a cluster.
- using a internodes communication protocol called "**Gossip**".
  - inspired by **real life rumour spreading.**
  - Gossip also used to **disseminate other system-related control state**.

- **Illustration of Gossip**:
  - Example – Node A wish to search for pattern in data
  - Round 1 – Node A searches locally and then gossips with node B.
  - Round 2 – Node A,B gossips with C and D.
  - Round 3 – Nodes A,B,C and D gossips with 4 other nodes ……

Failure Detection and Request Handling

Failure Detection:

- **determine if any other node in the system is up or down**
- uses a modified version of **the Accrual Failure Detector**
  - **tracks "heartbeats" from other nodes**
  - **unreachable nodes will be assumed to be "down".**

Request Handling:

- For **reads** :
  - Routes the requests to the **closest replica, or**
  - Routes the requests to **all replicas and waits for a quorum** of responses
- For **writes**:
  - the system **routes the requests to the replicas and waits for a quorum** of replicas to acknowledge the completion of the writes

Local Persistence

- The Cassandra system **relies on the local file system** for data persistence
- Write operation involves a **write into a commit log** for durability and recoverability
  - Then the **write into the in-memory data structure** is performed
- Read operation **first queries the in-memory data structure** before looking into the files on disk

Bigtable and Cassandra : Similarity

- Cassandra's data model, derived from Google's BigTable
  - Distributed multidimensional sorted map
  - Row key
  - Column families
  - Timestamp
- Using cluster concept
  - Bigtable : tablets
  - Cassandra : nodes
- Inserts and updates are committed to a commit log
- Data is compacted into SSTable

Cassandra is a daughter of Bigtable

http://www.planetcassandra.org/blog/cassandra-daughter-of-dynamo-and-bigtable/

Bigtable and Cassandra : Difference

Context	Bigtable	Cassandra
Replication	Handled by GFS	Chosen based on replication policy in configuration file
Partitioning	tablets server	nodes
Failure Detection	Using Chubby lock	Using gossip
Column Families	Only simple column families	There are simple and super column families (nested column families)
Read/Write Consistency	Directly into related tablets	one, all, or quorum consistency

References

- Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., et al. (2006). *Bigtable: a distributed storage system for structured data*. OSDI '06: Proceedings of the 7th symposium on operating systems design and implementation (p. 205-218). USENIX Association.
- Lakshman, A., Malik, P. (2010). Cassandra – *A Decentralized Structured Storage System*. ACM SIGOPS Operating Systems Review (p. 35-40)