Database Management Systems

Big Data Management

Malay Bhattacharyya

Associate Professor

Machine Intelligence Unit
and
Centre for Artificial Intelligence and Machine Learning
Indian Statistical Institute, Kolkata

April, 2025

1. Basics
   - Big data and streaming data
   - Beyond relational databases
2. NoSQL
   - Basics of NoSQL
   - History
   - Types and taxonomy
   - NoSQL – Advantages and challenges
   - Some concepts in NoSQL
   - NoSQL implementations
   - Relation between NoSQL and BigTable
3. BigTable
   - Basics of BigTable
   - History
   - Some concepts in BigTable

What is big data?

Big data refers to the data sets that are so large or complex that traditional data processing applications are inadequate for them. Challenges in big data analysis include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying, updating and information privacy.

The four V's of Big data:

- **Volume:** Volume refers to the size of the data to be analyzed.
- **Variety:** Variety refers to the different forms of the data.
- **Veracity:** Veracity refers to the trustworthiness of the data.
- **Velocity:** Velocity is the frequency of incoming data that needs to be processed.

Note: The **Value** of data is also an important issue.

What is streaming data?

Definition (Data stream model)

A *data stream model* defines an input stream $\mathcal{S} =< s_1, s_2, \cdots >$ arriving sequentially, item by item, and describes an underlying signal S, where $S : [1 \ldots N] \rightarrow \mathbb{R}$ is a one-dimensional function.

The data stream models can be of the following three types:

- Time Series Model
- Cash Register Model
- Turnstile Model

Data streams in real-world

1. Mining query streams
   - Google desires to know the queries that are more frequent today in comparison with yesterday.
2. Mining click streams
   - Yahoo! wishes to know the pages receiving an unusual number of clicks in the past hour.
3. Monitoring IP packets
   - A switch targets to detect the denial-of-service attacks.
   - A switch wants to gather information about IP packets for optimal routing.
4. Studying dynamically changing health conditions
   - A wearable device aims to keep track of all the body parameters (that keep changing) in real-time.

Limitations of relational databases

Relational databases are entirely dependent on relations

- Relational databases cannot deal with data that are infeasible of fitting to relations.
- They cannot handle unpredictable, unstructured or messy data.
- They may require vertical and horizontal expansion of servers, as the data or processing requirements grow.

What are beyond relational databases?

Non-relational distributable databases depending on unstructured data. They exhibit high performance with high availability, and offer rich query language and easy scalability.

CAP Theorem: Consistency, availability and partition tolerance.

NoSQL

Examples include - MongoDB, Hadoop, etc.

Note: Non-relational databases may not provide full ACID (atomicity, consistency, isolation, durability) guarantees, but still has a distributed and fault tolerant architecture.

Basics of NoSQL

NoSQL (often referred to as Not only SQL) is a broad class of data management systems where the data is partitioned across a set of servers, where no server plays a privileged role.

– NoSQL database management systems are useful when working with a huge quantity of data (especially big data), when the data's nature does not require a relational model
– The data can be structured, but NoSQL is used when what really matters is the ability to store and retrieve great quantities of data (e.g., millions of keyvalue in one or a few associative arrays)
– This is particularly useful for statistical or real-time analysis of growing lists of elements (e.g., huge number of Twitter posts, Internet server logs from a large group of users)
– It provides flexibility of the data model (e.g., ERP and BPM)

History

"It's not about BASE vs. ACID" – Emin Gün Sirer.

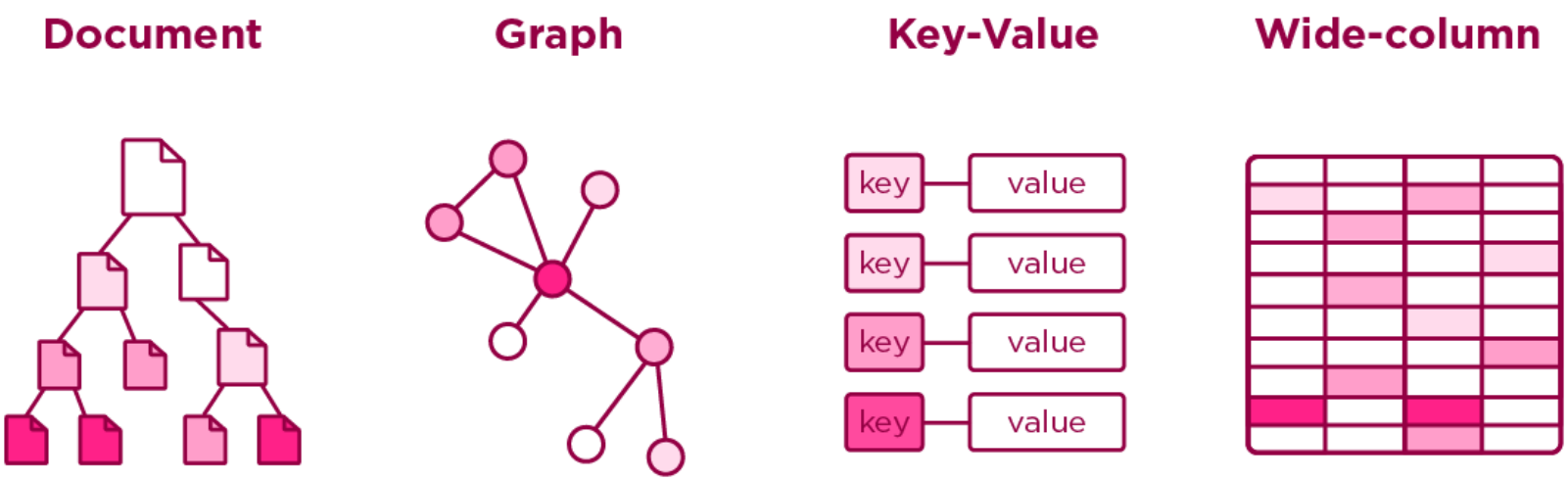
1998: Carlo Strozzi used the term NoSQL in 1998 to name his lightweight, open-source relational database that did not expose the standard SQL interface.
2009: Eric Evans, a Rackspace employee, reintroduced the term NoSQL to label the emergence of a growing number of non-relational, distributed data stores.
2011: Work began on UnQL (Unstructured Query Language), a specification for a query language for NoSQL databases, designed to query collections (versus tables) of documents (versus rows) with loosely defined fields (versus columns).

Note: BASE refers to basically available, soft-state and eventual consistency.

Types of NoSQL databases

NoSQL databases are mainly of the following types:

- Document based
- Graph based
- Key-Value pair based
- Wide-column / Column-family based



Taxonomy of NoSQL

- **Document:** Clusterpoint, Couchbase, MarkLogic, MongoDB, etc.
- **Graph:** Allegro, Neo4j, OrientDB, Virtuoso, etc.
- **Key-value:** Dynamo, MemcacheDB, Project Voldemort, Redis, Riak, etc.
- **Column:** Accumulo, Cassandra, HBase, etc.

NoSQL – Advantages

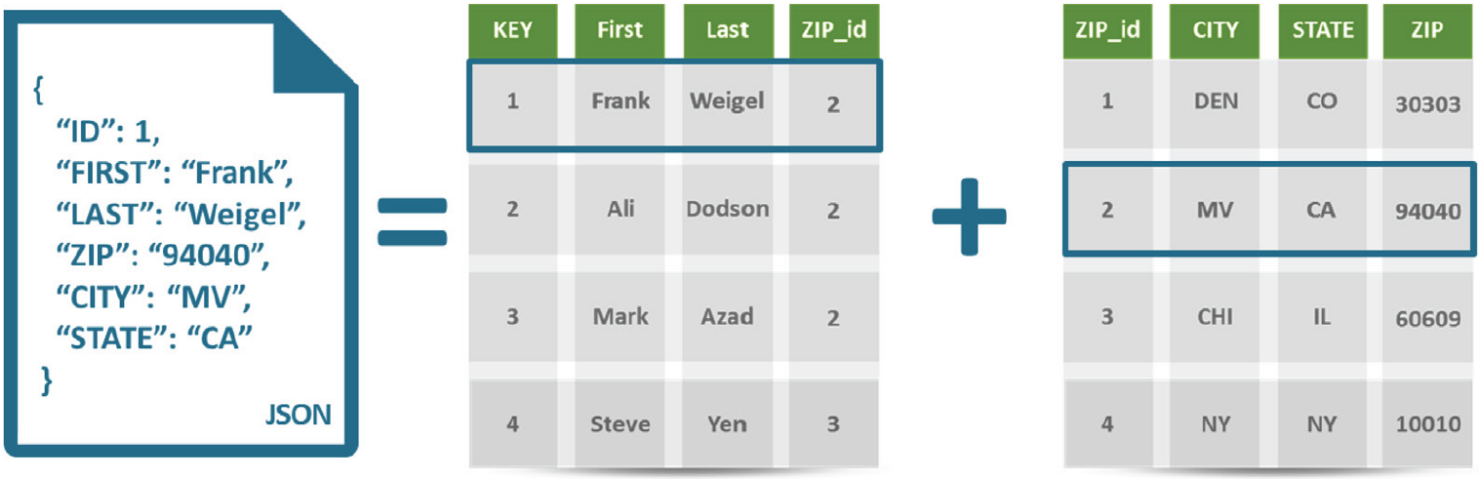
- Elastic scaling – database can be distributed across multiple hosts as load increases (scale out)
- Handling big data – massive volume of data and transactions can be managed
- No DBA – the mechanism of automatic repair, data distribution, and simpler data models lead to lower administration and tuning requirements
- Low cost – clusters of cheap commodity servers are typically used to manage the exploding data and transaction volumes
- Flexible data models – schemaless relaxed (or even nonexistent) data model restrictions are used
- Caching layer – employs a useful cache layer
- Real time analysis and streaming model – distributed processing makes it faster

NoSQL – Challenges

- No normalization, referential integrity, ACID – basic criteria of relational databases are not satisfied
- Poor data security – distributed data becomes less secure and might get lost easily
- Immaturity – most NoSQL alternatives are in pre-production versions with many key features yet to be implemented
- Poor support – NoSQL systems are mostly open source projects without the global reach, support resources, or credibility
- Weakness in analytics and business intelligence – commonly used BI tools do not provide connectivity to NoSQL
- Targeting no administration – the goal of providing a zero-admin solution still has not been fulfilled
- Limited expertise – almost every NoSQL developer is in a learning mode

JSON format

JSON (stands for JavaScript Object Notation) is a binary and typed data model that supports the data types list, map, date, Boolean, and numbers (with different precisions). It can be serialized to and deserialized from bytes as well as strings.



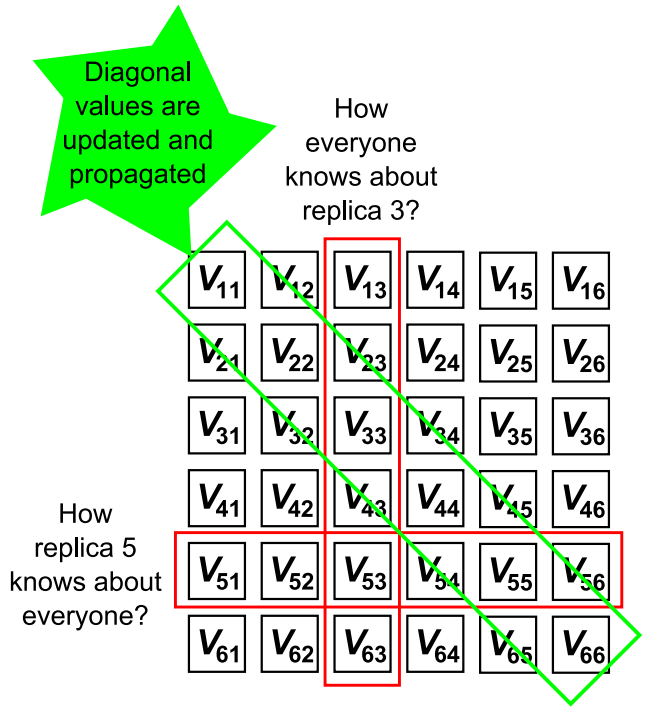
Some concepts in NoSQL – Versioning of datasets

To process "concurrent" modifications and versions of datasets in a distributed environment, different schemes can be employed. Some of these are listed below.

- **Timestamps:** It helps to develop a chronological order but rely on synchronized clocks and don't capture causality.
- **Optimistic Locking:** It implies that a unique counter or clock value is saved for each piece of data. When a client tries to update a dataset it has to provide the counter/clock-value of the revision it likes to update. It does not work well in a distributed and dynamic scenario where servers show up and go away often and without prior notice.
- **Multiversion Storage:** It means storing a timestamp for each table cell, thus providing optimistic concurrency control with compare-and-swap on timestamps.
- **Vector Clocks:** It is an alternative approach to capture order and allow reasoning between updates in a distributed system.

Some concepts in NoSQL – Vector Clocks

A vector clock is defined as a tuple $V_{i0}, V_{i1}, \ldots, V_{in}$ of clock values (may be real timestamps, version/revision numbers or some other ordinal values) collected from each node $i = 1, \ldots, n$, which represent the state of itself and the other (replica) nodes' state.



An overview of a vector clock managing six nodes

Some concepts in NoSQL – Vector Clocks

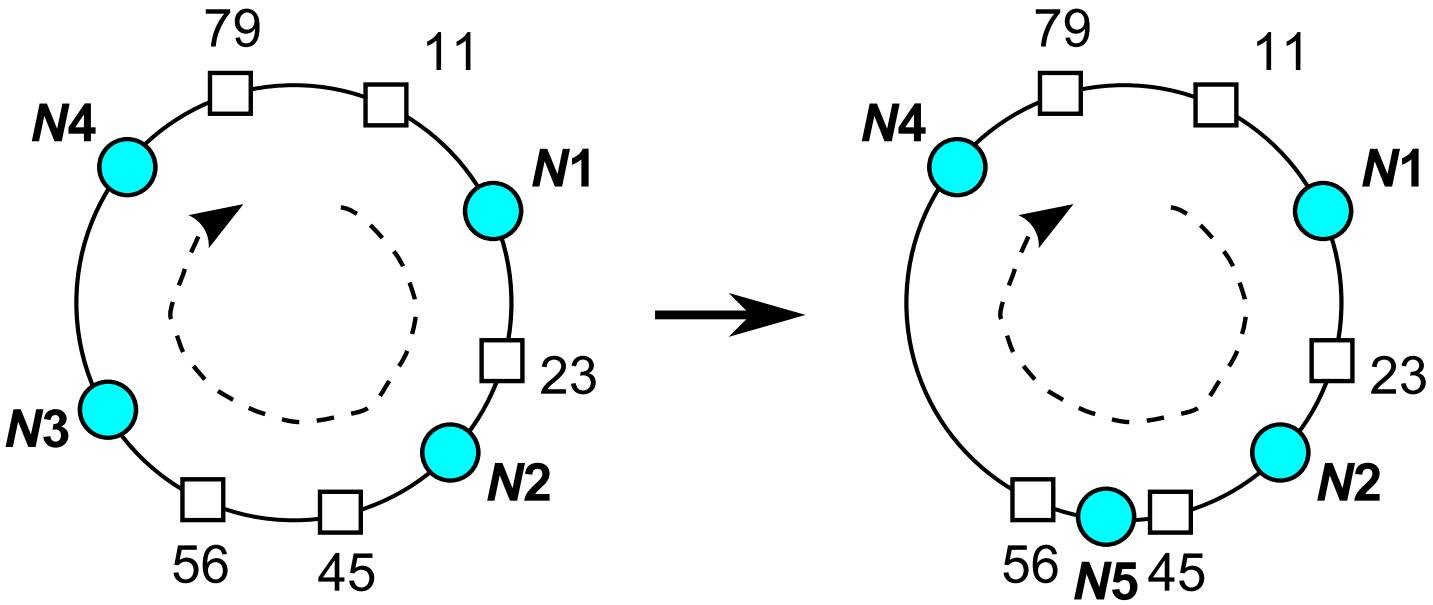
Vector clocks can be utilized "to resolve consistency between writes on multiple replicas".

Vector clocks are updated in a way defined by the following rules:

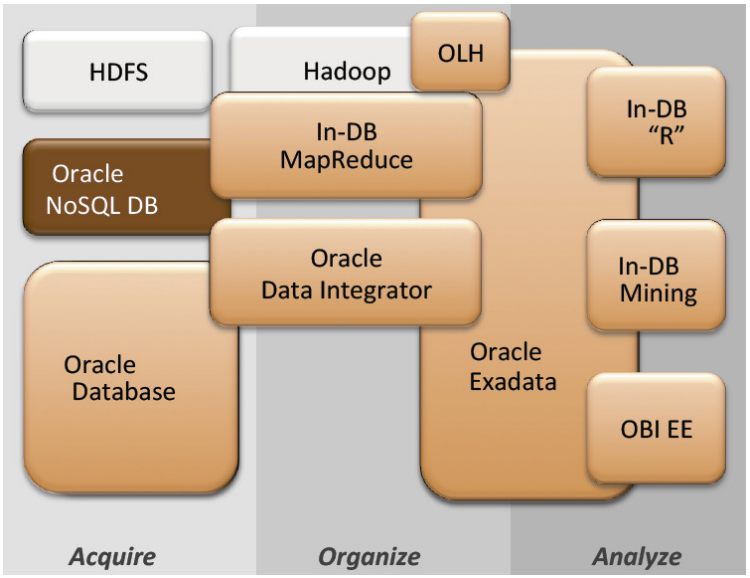
1. If an internal operation happens at node i, this node will increment its clock V_{ii} (immediate execution).
2. If node i sends a message to node k, it first advances its own clock value V_{ii} and attaches the vector clock V_i (about his internal state and his view of the other nodes) to the message sent to node k.
3. If node i receives a message from node j, it first advances its vector clock V_{ii} and then merges its own vector clock with the vector clock $V_{message}$ attached to the message from node j so that $V_i = \max(V_i, V_{message})$. To compare two vector clocks V_i and V_j in order to derive a partial ordering, the following rule is applied $V_i > V_j$, if $\forall k V_{ik} > V_{jk}$, If neither $V_i > V_j$ nor $V_i < V_j$ applies, a conflict caused by concurrent updates has occurred and needs to be resolved (e.g., by a client application).

Consistent hashing

Which object is mapped to which node is determined by moving clockwise around the ring. Therefore, if $N3$ leaves and $N5$ enters the system, the object 45 will get mapped to node $N5$.



Oracle NoSQL Database



Oracle NoSQL database seamlessly integrates into an enterprise application architecture

Oracle-provided adapters allow the Oracle NoSQL database to integrate with a Hadoop MapReduce framework or with the Oracle database in-database MapReduce, data mining, R-based analytics, or whatever business needs demand.

Final comments

None of the SQL and NoSQL supersedes each other

NoSQL is promising but it still has limitations

Relation between NoSQL and BigTable

BigTable uses some of the concepts of NoSQL

The NoSQL taxonomy supports key-value stores, document store, BigTable, and graph databases.

Basics of BigTable

BigTable is a compressed, high performance, and proprietary data storage system built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies. It is not distributed outside Google, although Google offers access to it as part of its Google App Engine.

History

"BigTable is in big use" – Anonymous.

2004: BigTable development begins for scalability and better control of performance characteristics by Google
2010s: Google's Spanner RDBMS is layered on an implementation of BigTable with a Paxos group for two phase commits to each tablet. Google F1 is built using Spanner to replace an implementation based on MySQL.
Now: Used by a number of Google applications such as Gmail, Google Reader, Google Maps, Google Book Search, Google Earth, MapReduce, which is often used for generating and modifying data stored in BigTable, etc., and also by Orkut and YouTube.

Some concepts in BigTable – Infrastructure

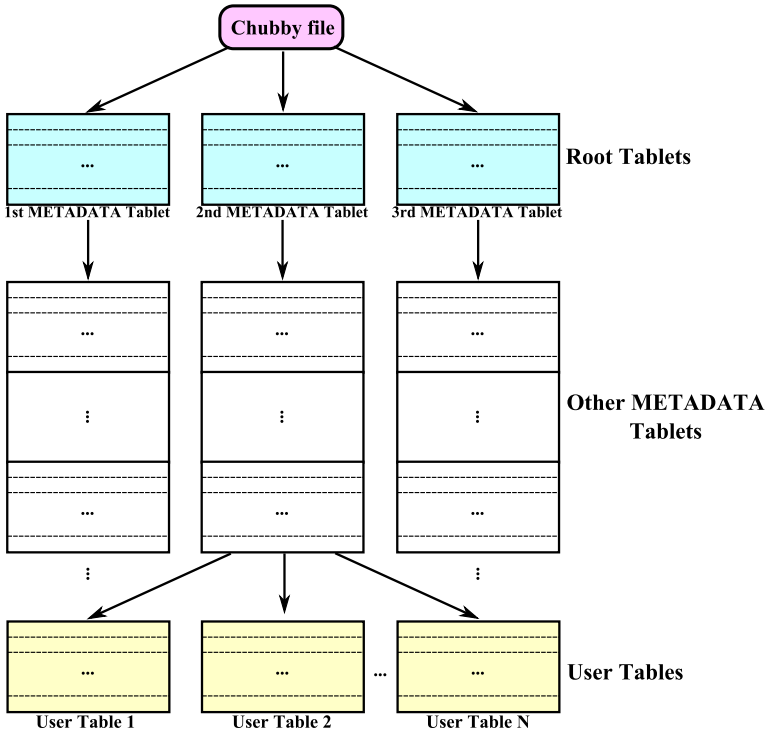
BigTable is built on several other pieces of Google infrastructure. BigTable uses the distributed Google File System (GFS) for storing log and data files.

The Google SSTable file format is used internally to store BigTable data. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings. Internally, each SSTable contains a sequence of blocks (typically 64KB in size, but can be reconfigured). A block index (stored at the end of the SSTable) is used to locate blocks; the index is loaded into memory when the SSTable is opened.

BigTable relies on a highly-available and persistent distributed lock service called Chubby that consists of five active replicas, one of which is elected to be the master and actively serve requests.

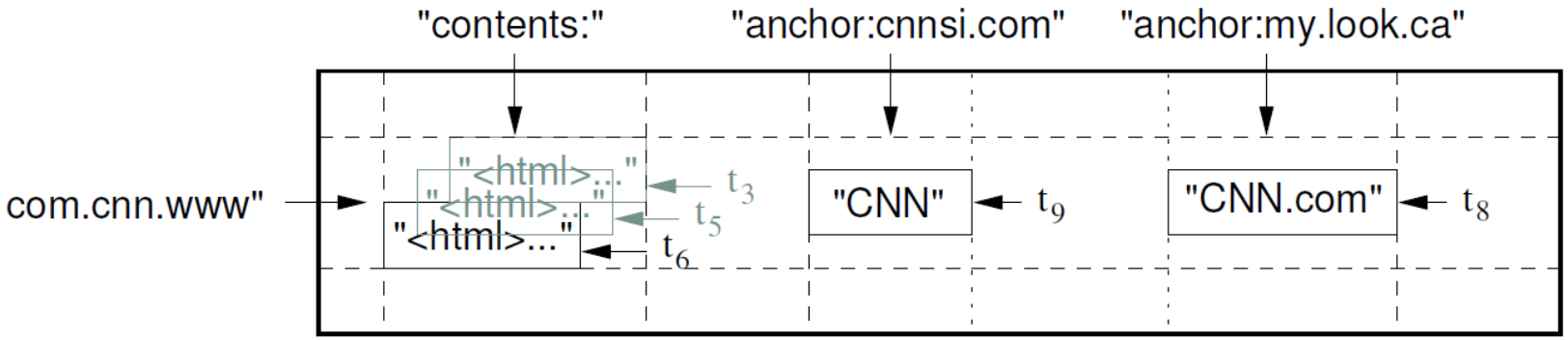
Some concepts in BigTable – Implementation

The BigTable implementation has three major components: a library that is linked with every client, one master server, and many tablet servers. Table location information is stored in a three-level hierarchy analogous to that of a B^+-tree.



Hierarchy of tablet location in BigTable

Some concepts in BigTable – Storing webpages

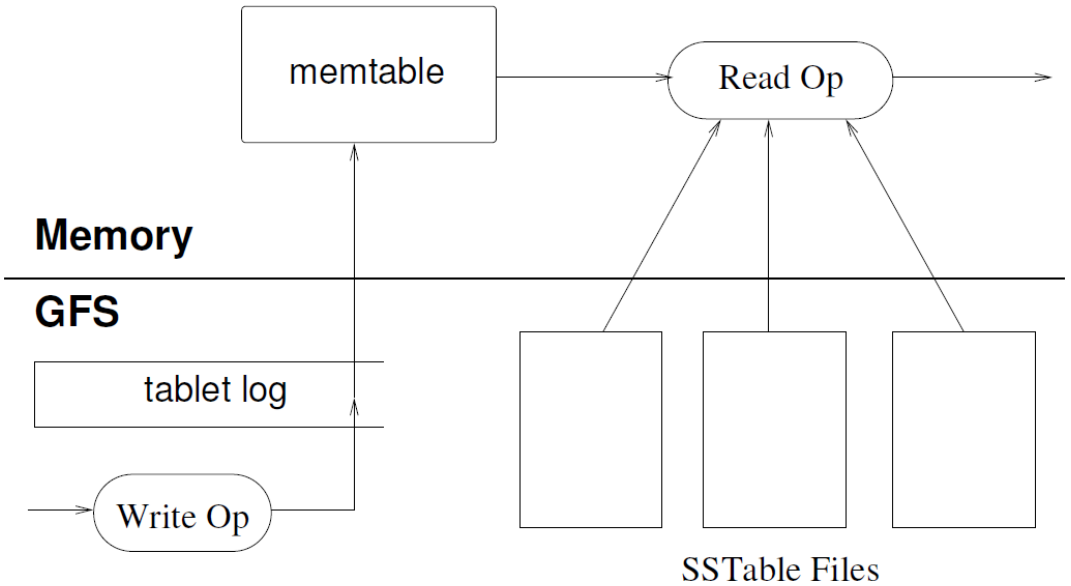


An example table that stores webpages (Chang, OSDI 2006)

The row name is a reversed URL. The contents column family contains the page contents, and the anchor column family contains the text of any anchors that reference the page. CNN's home page is referenced by both the Sports Illustrated and the MY-look home pages, so the row contains columns named anchor:cnnsi.com and anchor:my.look.ca. Each anchor cell has one version; the contents column has three versions, at timestamps t_3, t_5, and t_6.

Some concepts in BigTable – Tablet serving

The persistent state of a tablet is stored in GFS (see below). Updates are committed to a commit log that stores redo records. Of these updates, the recently committed ones are stored in memory in a sorted buffer called a memtable; the older updates are stored in a sequence of SSTables.



Tablet representation (Chang, OSDI 2006)