Detailed Design of Bigtable: Part I

Explore the design of Bigtable in detail and understand the interaction of various components.

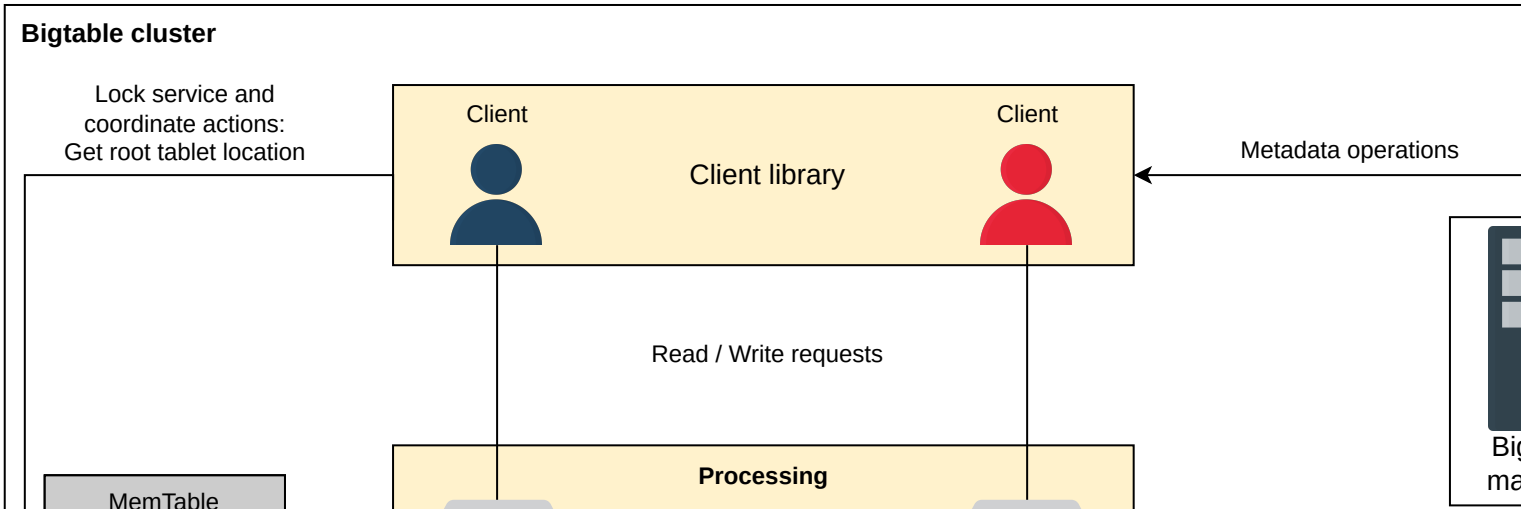
We'll cover the following

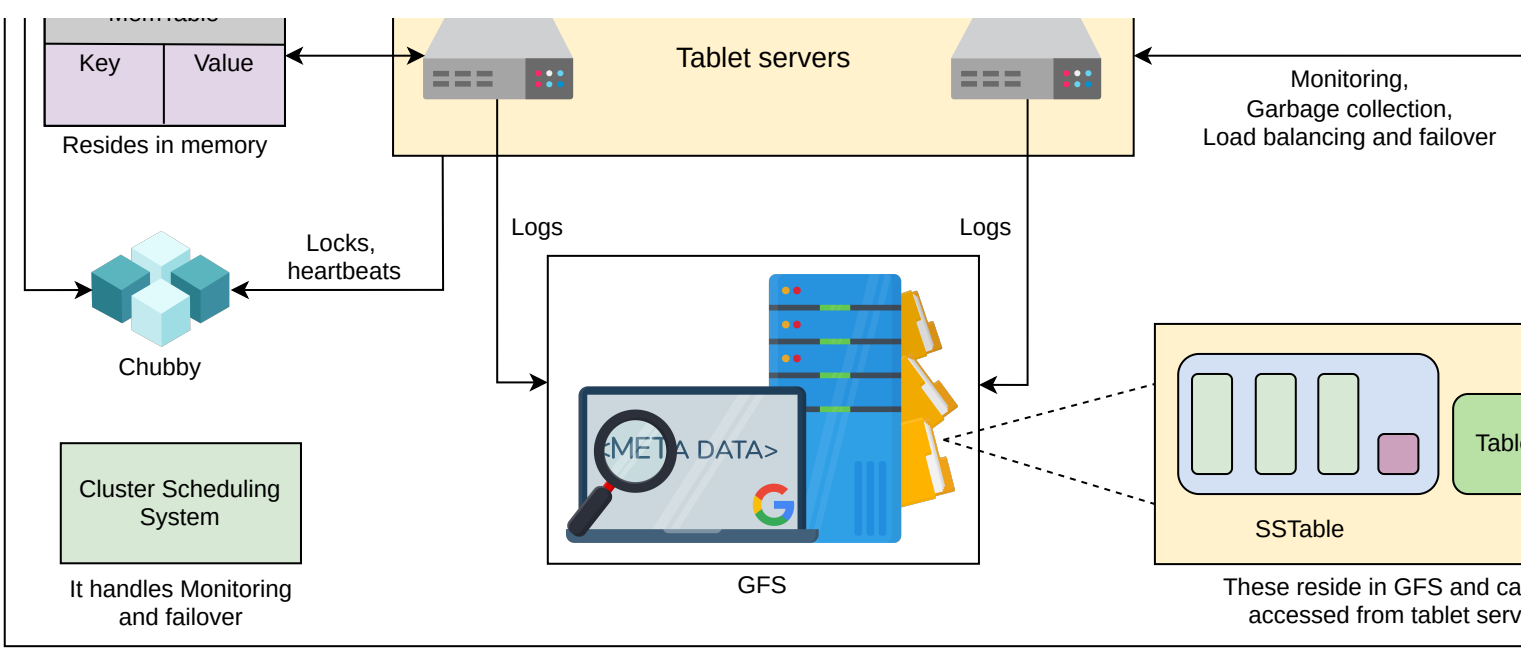
Components

Here's a list of major components in our Bigtable design:

- SSTable
- MemTable
- GFS
- Chubby

Let's discuss the components of Bigtable design in detail.

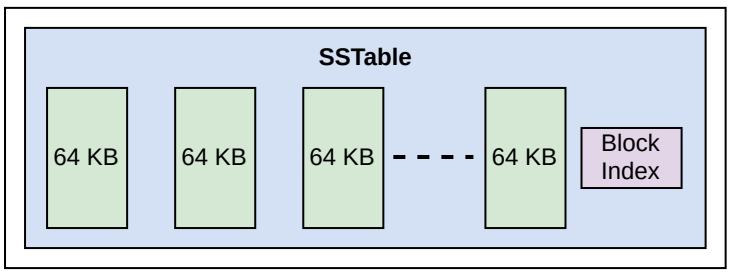




Detailed design of Bigtable

SSTable

The data is kept as files in Bigtable using the Google File System (GFS), a persistent distributed file storage system. **Sorted String Table**, or **SSTable** for short, is the file format that Bigtable uses to store its data. It is a persistent, ordered, immutable map of keys to values, where both the keys and the values are random byte strings. They are used to store the persistent state of tablets. Multiple SSTable comprise a tablet. There are operations to search up a particular key's linked value and to loop over all key/value pairings within a defined key range. An SSTable is made up of a series of blocks, which are normally 64 KB in size but can be configured to be a different size.



SSTable

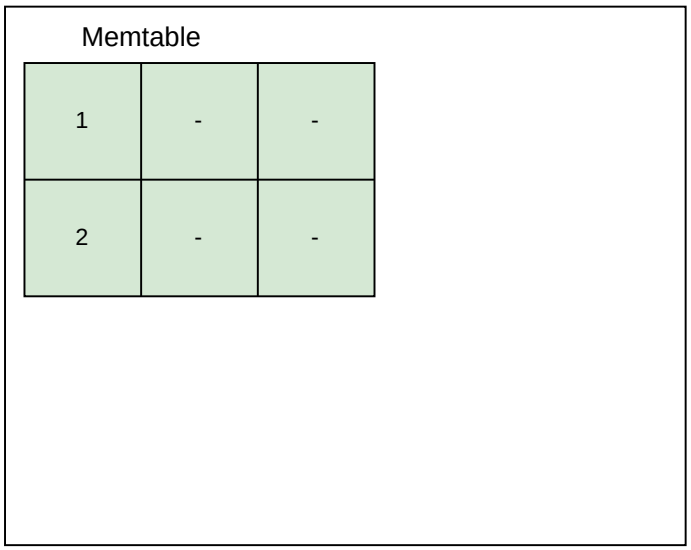
When the SSTable is accessed, a block index is loaded into memory and utilized to find blocks. Single disk seek can be utilized to perform a lookup. By using a binary search to locate the correct block in the in-memory index, we then read the correct

block from the disk. It is also possible to load an entire SSTable into memory, which allows for lookups and scans to be performed without reading from the disk. SSTables give two operations, i.e., obtain the value related to a specified key and iterate through a set of values in a specified key range. When written to GFS, each SSTable is immutable (read-only). A new SSTable is generated if new data is added. When an old SSTable is no longer required, it is designated for garbage collection. The immutability of SSTable has the following benefits:

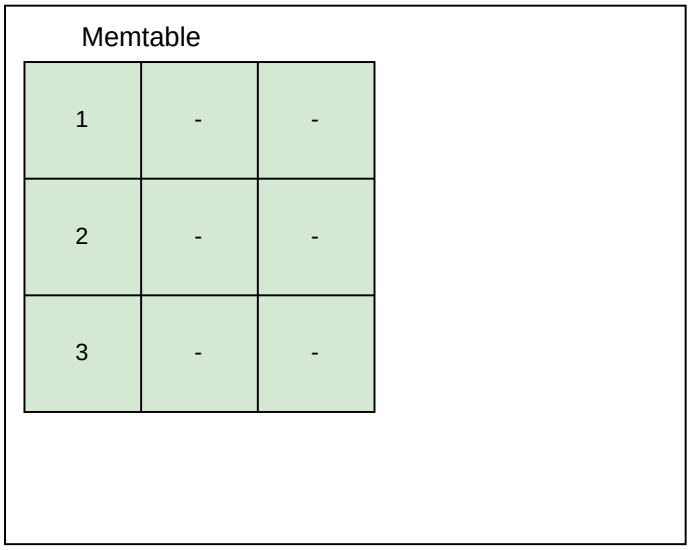
- No synchronization is required during read operations.
- Permanent deletion of outdated data is handled by the garbage collector.
- It helps to split tablets quickly.

Memtable

To increase write efficiency, Bigtable stores recent modifications in an in-memory, mutable sorted buffer called **memtable**. The memtable grows in size whenever a write operation is carried out. When the size of a memtable hits a certain limit, the memtable is paused, a new memtable is produced, and the paused memtable is transformed to an SSTable and stored in GFS.

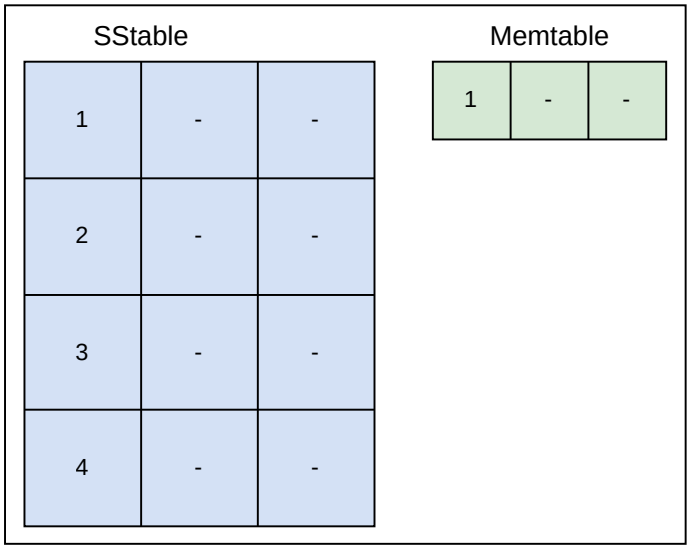


The new memtable



New data is entered in memtable

2 of 3



The threshold has been reached. The new memtable is created and the previous one is converted to an SSTable.

3 of 3

Why use SSTable with memtable?

Once it's on the disk, an SSTable is essentially immutable since inserting or deleting would necessitate a significant I/O rewrite of the file. Despite this, it's an excellent solution for static indexes. It is read in the index, and we're always one disk seek

away. Alternatively, we can simply map the entire file to memory. Random reads are quick and simple.

Random writes are far more complex and costly unless the table is in memory. How do we achieve this?

We want to keep the quick read access provided by SSTables, but we also want to provide fast random writes. It turns out that we already have everything we need: random writes are speedy when the SSTable is in memory i.e., memtable, and if the table is immutable, an on-disk SSTable (in GFS) is likewise quick to read from. We will go into further detail about how read/write works in the next lesson.

Point to Ponder

Question

What happens to memtable data if a server fails before it can be persisted to SSTable?

Hide Answer

Each data change is also recorded in a commit-log, which is kept in GFS. If a tablet server crashes before committing a memtable to SSTable, this log provides redo entries that can be used for recovery. The data might be in SSTables or memtables during reading. Finding the most recent data is simple because both of these tables are sorted.

GFS

For its massively data-intensive applications like Bigtable, Google created the scalable distributed file system known as GFS.

GFS files are divided into 64 MB fixed-size units known as chunks. Chunks are kept on data servers known as chunkservers. The metadata is managed by the GFS manager. SSTables are broken into fixed-size blocks, which are then stored on chunkservers. For durability, each chunk in GFS is duplicated over many chunkservers. Clients exchange metadata with the GFS manager, but all data transfers take place directly between the client and the chunkservers.

> **Note:** For a detailed explanation of GFS, you can see the Google File System chapter.

Chubby

It is a highly available and persistent distributed lock service that helps in the coordination of all the replicas.

Chubby typically runs with five operating replicas; one is designated as the primary to service requests. To stay alive, most Chubby replicas must be operational. Bigtable is so reliant on Chubby that if Chubby is offline for a long amount of time, Bigtable also becomes inaccessible. Chubby employs the Paxos algorithm to make its clones similar in a distributed system. Chubby offers a namespace of files and folders. Every file or directory can be utilized as a lock. A Chubby file's read and write operations are atomic.

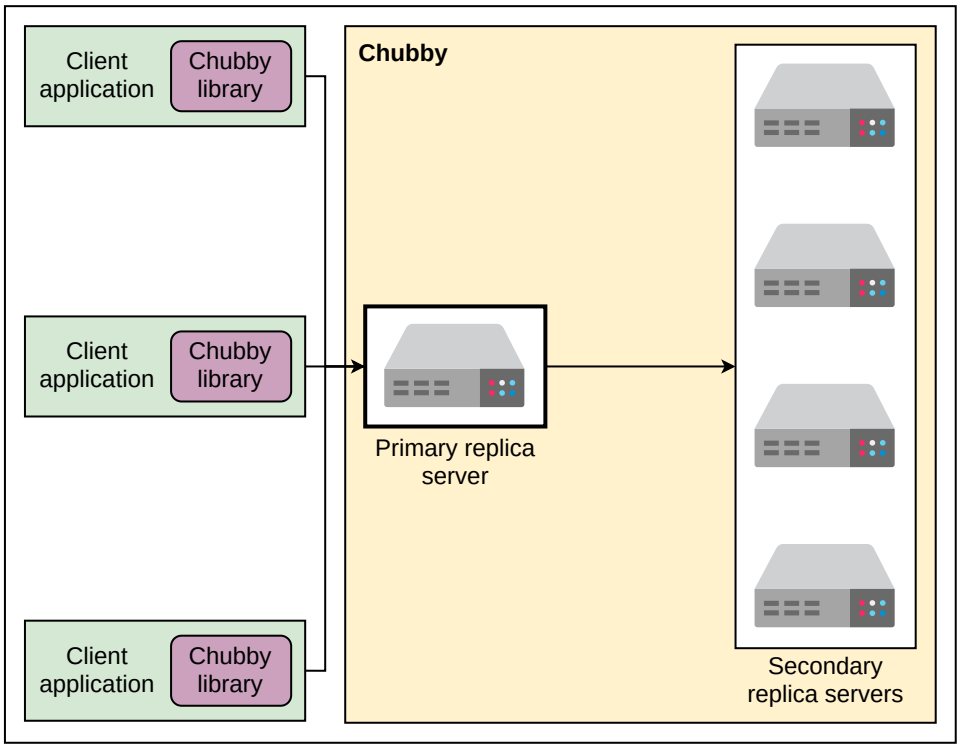
Each Chubby client is connected to a Chubby service. A client's session lease expires if they don't renew it within the allotted time period. Any locks and open handles are released when a client's session terminates. Chubby clients can additionally set callbacks on Chubby files and folders to receive updates or session expiration notifications.

Uses in Bigtable

1. Chubby is utilized in Bigtable to ensure there is only one operational manager. The manager keeps a session lease with the Chubby client and renews it on a regular basis to keep its position as a manager.

2. Chubby saves the bootstrap location of the Bigtable data.
3. It is used to learn about new tablet servers and the failures of old ones.
4. Keep Bigtable schema information (the column family information for each table).
5. Access Control Lists (ACLs) are also stored in Chubby.

> **Note:** For a detailed explanation of Chubby, you can go through The Chubby Lock Service.



Chubby architechure

Let's see how the above components combine to make Bigtable work in the next lesson.

Back

Data Model of Bigtable

Next

Detailed Design of Big...

Mark as Completed