Google BigTable

Google BigTable

- Developped in 2004, BigTable is now Used by a number of Google applications mainly for web indexing.
- BigTable is a distributed storage system for managing structures data that is designed to scale to very large size. Copies of the web, satellite data, user data, emails, many incoming requests.
- It is a Google's scalable database. It provides a way to create massive tables of information indexed by a primary key. Over 90% of Google's web services are built on top of BigTable, including MapReduce (which is often used for generating and modifying data stored in BigTable), Google Search, Google Earth, Google Analytics, Google Maps, Gmail, Orkut, YouTube, and many more.
- There is no commercial system big enough to store all this data. And if there was it would be an extremely costly service

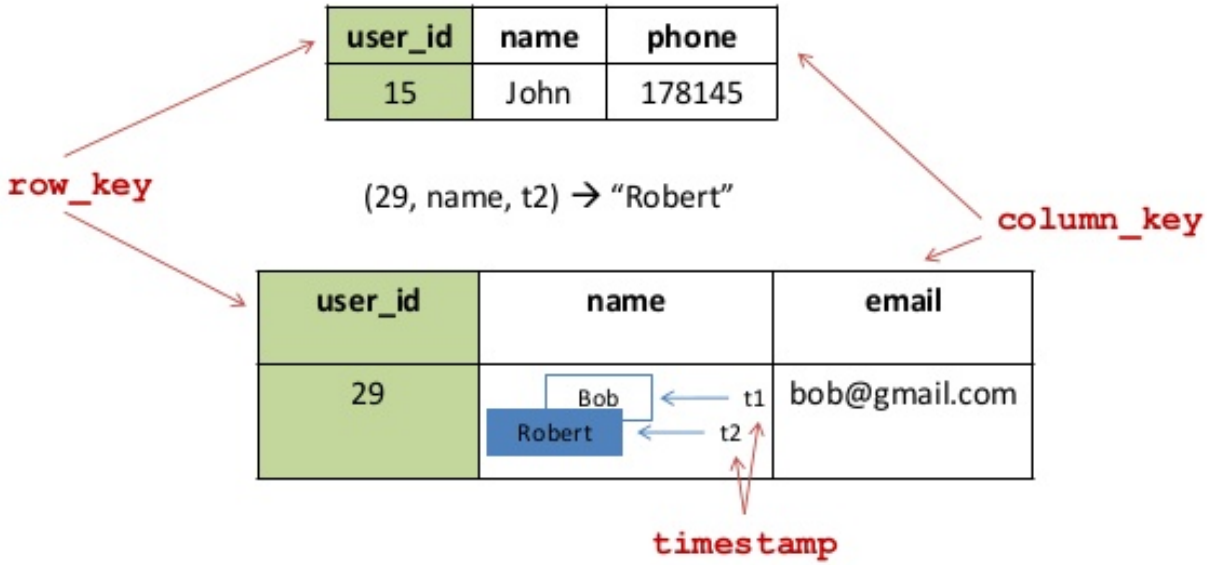
Data Model

- Sparse, distributed, persistent multidimensional sorted map

Key					Value
row ID	Column			Timestamp	
	Family	Qualifier	**Visibility**		

- BigTable is a sparse, distributed, persistent multidimensional sorted map.
- The map is indexed by a row key, column key, and a timestamp
- (row_key, column_key, time) returns a string

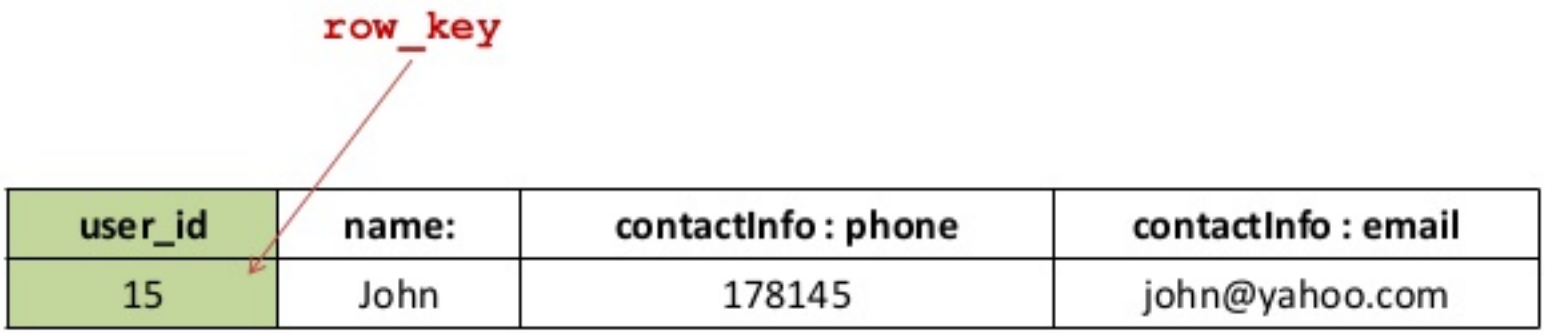
Data Model



RDBMS Approach	user_id	name	phone	email
	15	John	178145	null
	29	Bob	null	bob@gmail.com

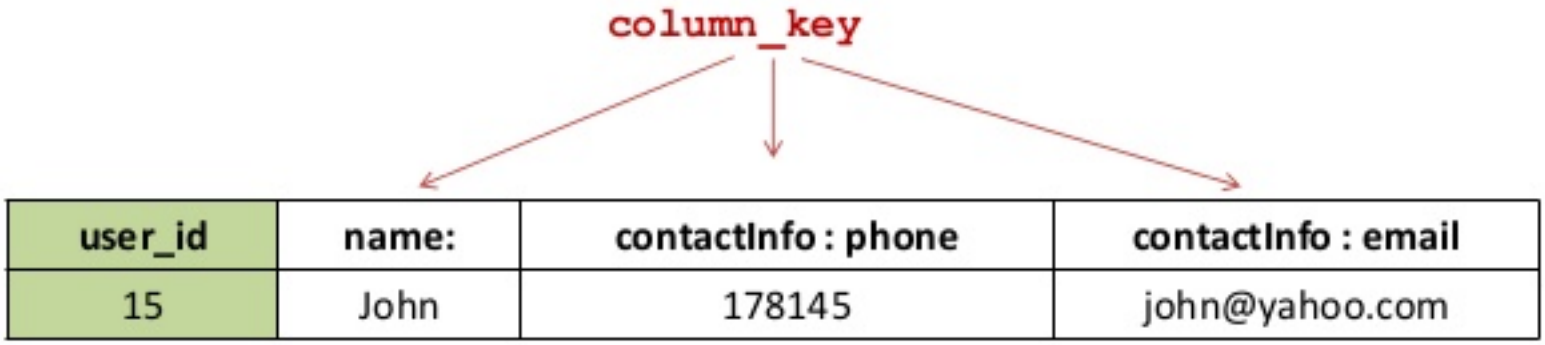
BigTable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers.

Rows



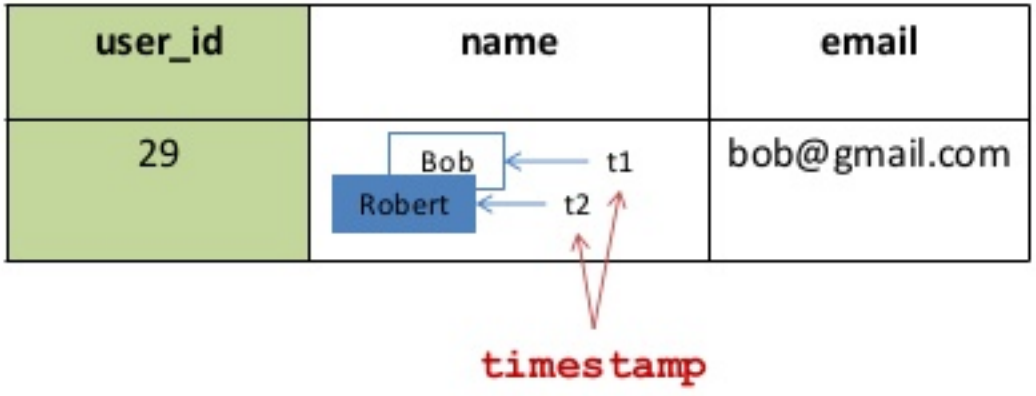
- The row keys in a table are arbitrary strings which
- Every read or write of data under a single row key is atomic.
- Data is maintained in lexicographic order by row key.
- A row range defines a subset of the table called tablet, which is the unit of distribution.
- Reads of short row ranges are efficient and typically require communication with only a small number of machines.
- For example, in Webtable, pages in the same domain are grouped together for that purpose.

Column Families



- Column Families
- Columns keys are grouped into sets called “column families” which form the basic unit of access control. They must be created before data can be stored under any column key in that family.
- A column key is named using the following syntax: “family:qualifier”
- Column family names must be printable, but qualifiers may be arbitrary strings.
- Memory accounting are performed at the column-family level.
- Example in Webtable: column family ‘anchor’.

Timestamps



- Timestamps:
- Each cell in a BigTable can contain multiple versions of the same data; these versions are indexed by timestamps. The timestamps can be automatically assigned by BigTable (real time In ms) or they can be explicitly assigned via the client application.
- Versions are stored in decreasing timestamp order.
- Older versions are garbage-collected based on timestamps. E.g. keep versions of past 8 days

Tablets

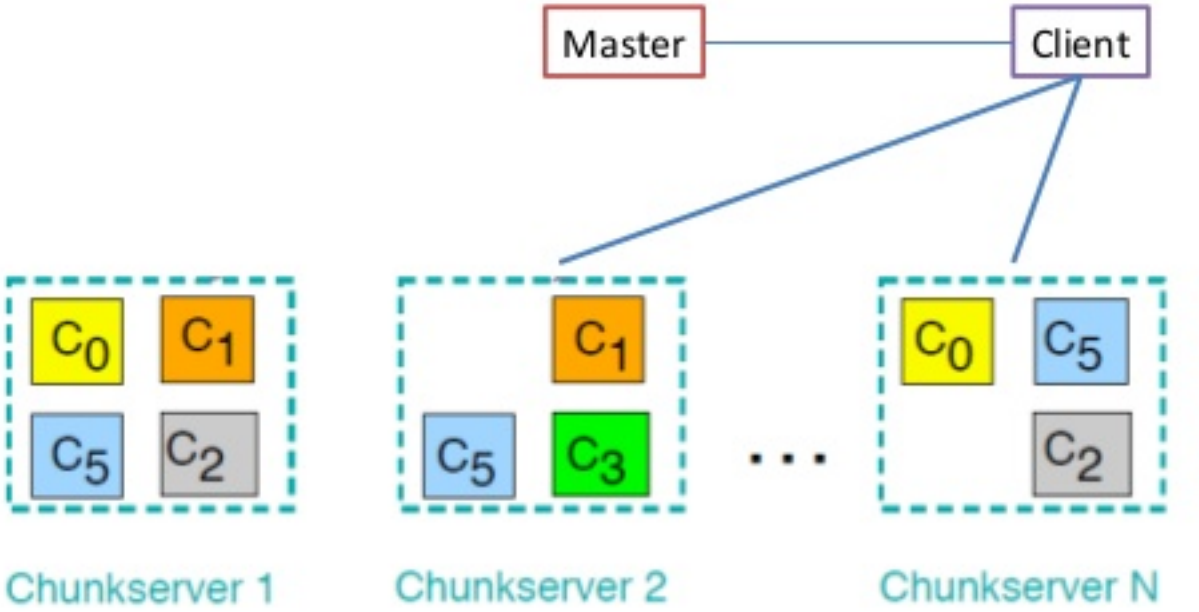
	id	.....
Tablet 1	15000	.....
	....	.....
	20000	.....
Tablet 2	20001	.....
	....	.....
	25000	.....

- Tablets
- Large tables broken into tablets at row boundaries.
- Tables holds contiguous range of rows.
- Approximately 100-200 MB of data per tablet

Building Blocks

- Google File System (GFS): large-scale distributed file system
- Chubby: distributed lock service
- SSTable: persistent immutable map file from keys (arbitrary byte strings) to values (arbitrary byte strings)

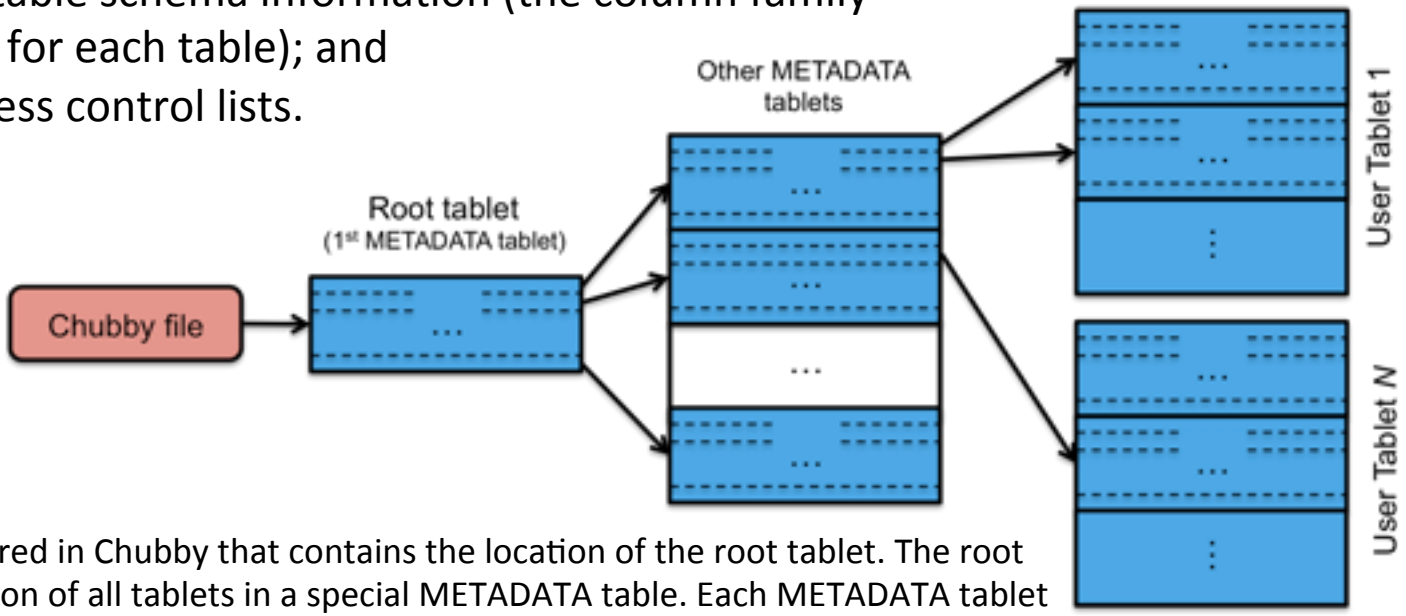
Google File System (GFS)



- Google File System (GFS):
- Files broken into chunks (typically 64 MB)
- Master manages metadata
- Data transfers happens directly between clients/ chunkservers/

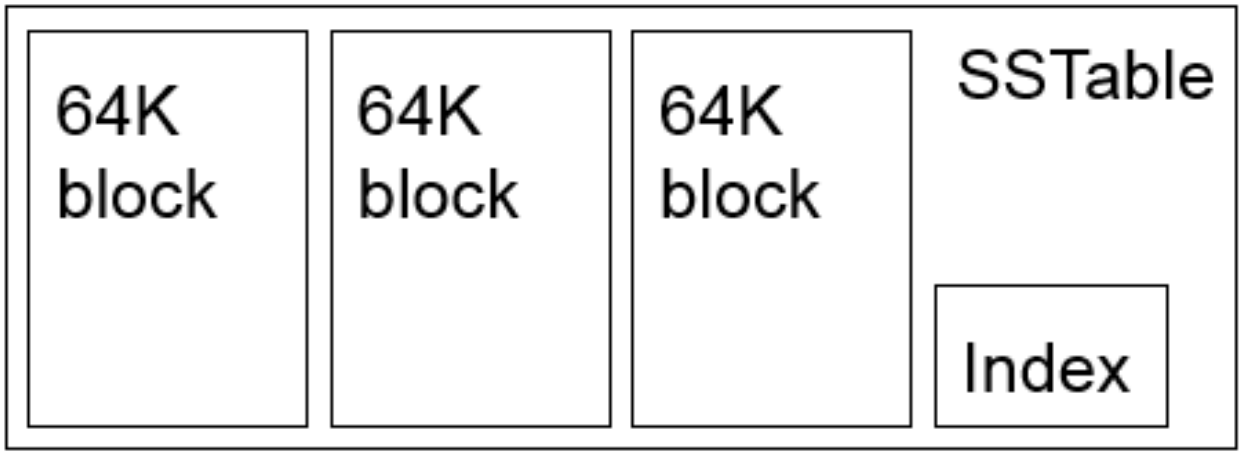
Chubby

- Bigtable uses Chubby for a variety of tasks:
  - to ensure that there is at most one active master at any time;
  - to store the bootstrap location of Bigtable data;
  - to discover tablet servers and finalize tablet server deaths;
  - to store Bigtable schema information (the column family information for each table); and
  - to store access control lists.



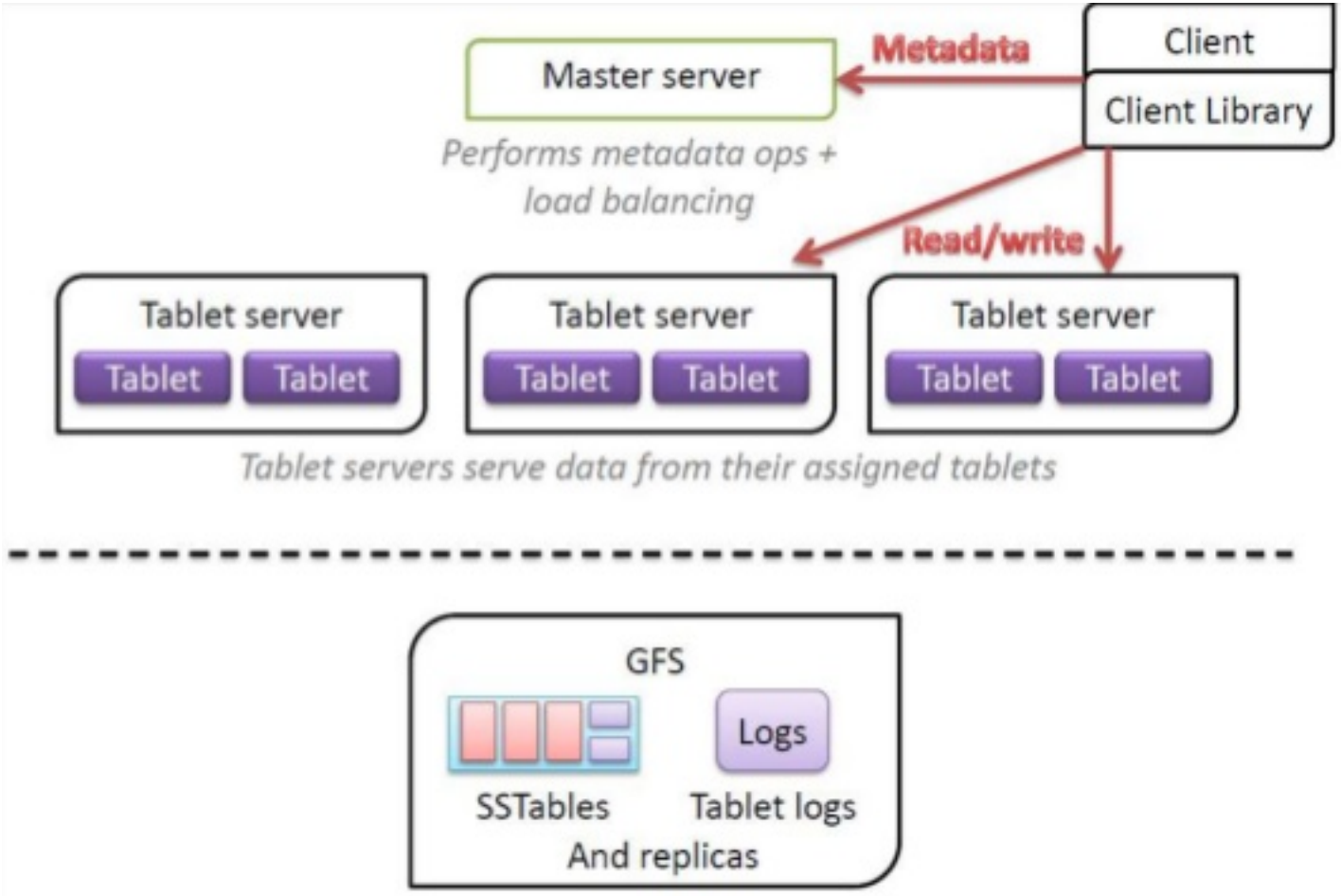
- The first level is a file stored in Chubby that contains the location of the root tablet. The root tablet contains the location of all tablets in a special METADATA table. Each METADATA tablet contains the location of a set of user tablets. The root tablet is just the first tablet in the METADATA table, but is treated specially—it is never split—to ensure that the tablet location hierarchy has no more than three levels.
- Chubby provides a namespace that consists of directories and small files where each directory or file can be used as a lock and where reads and writes to a file are atomic.

SSTable



- 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. Operations are provided to look up the value associated with a specified key, and to iterate over all key/value pairs in a specified key range. Internally, each SSTable contains a sequence of blocks (typically each block is 64KB in size, but this is configurable). 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. A lookup can be performed with a single disk seek: we first find the appropriate block by performing a binary search in the in-memory index, and then reading the appropriate block from disk. Optionally, an SSTable can be completely mapped into memory, which allows us to perform lookups and scans without touching disk.

System Structure



System Structure

- There are three major components:
  - A library that is linked into every client,
  - One master server, and
  - Many tablet server
- The master server is responsible for
  - Assigning tablets to tablet servers,
  - Detecting the addition and expiration of tablet servers,
  - Balancing tablet-server load, and
  - Garbage collection of files in GFS
  - Handling of schema changes such as table and column family creations
- The tablet server:
  - Manages a set of tablets (typically we have somewhere between ten to a thousand tablets per tablet server)
  - Handles read and write requests to the tablets that it has loaded, and also
  - Splits tablets that have grown too large
- Clients:
  - Communicate directly with tablet servers for reads and writes
  - Most clients never communicate with the master; as a result, the master is lightly loaded in practice