Bigtable

from the paper

BigTable: A Distributed Storage System for Structured Data (2006)

What it is (1)

- 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**.
- NO relational database system
- INSTEAD: Excel-like table abstraction (rows +columns) with additional time dimension

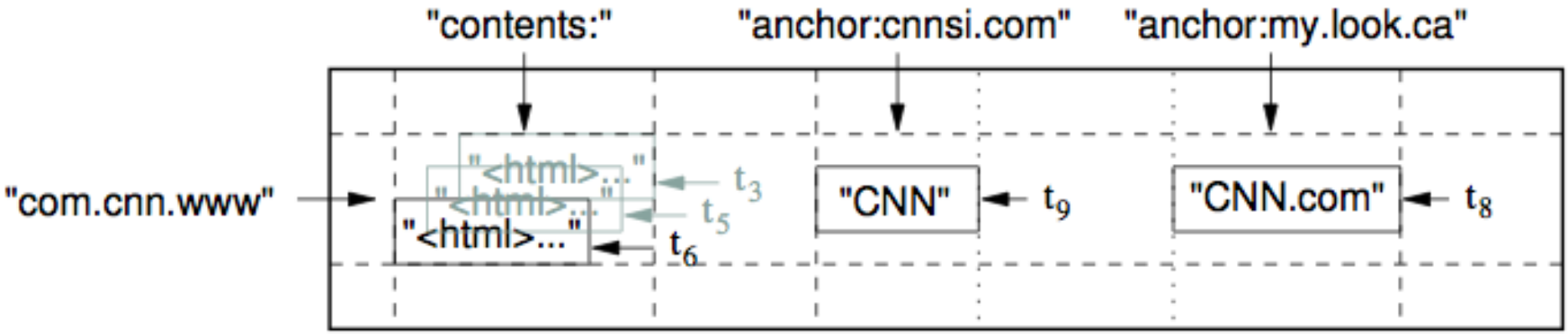
What it is (2)

- A Bigtable is a **sparse, distributed, persistent multi- dimensional sorted map**. The map is indexed by a row key, column key, and a timestamp:

  (row:string, column:string, time:int64) → string

- **string <=>** each value in the map is an **uninterpreted array of bytes**.

Webtable as example



Rows

- arbitrary strings
  - currently up to 64KB in size,
  - although 10-100 bytes is a typical size
- Bigtable maintains data in lexicographic order by row key.
- a **row range** defines a subset of the table called **tablet**
- 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

Columns

- Column keys are grouped into sets called column families
  - form the basic unit of access control
  - 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.
- Example in Webtable: column family ‘anchor’

Time stamps

- multiple versions of the same data within a cell are indexed by timestamps
- can be automatically assigned by Bigtable (-> real time)
- or can be explicitly assigned
- Bigtable offers garbage collection based on time stamps
  - eg, keep versions of past 8 days

Sample usage of Bigtable API (1)

```
// Open the table
Table *T = OpenOrDie("/bigtable/web/
webtable");

// Write a new anchor and delete an old
anchor RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

Sample usage of Bigtable API (2)

```
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {
   printf("%s %s %lld %s\n", scanner.RowName(),
          stream->ColumnName(),
          stream->MicroTimestamp(),
          stream->Value());
}
```

Bigtable building blocks (1)

- (massively) distributed Google File System (GFS)
- cluster management system
- SSTable: persistent immutable map from keys (arbitrary byte strings) to values (arbitrary byte strings)
  - consists of a sequence of blocks (each 64 KB)
- distributed lock service, called Chubby

Bigtable building blocks (2)

- Chubby provides a namespace that consists of directories and small files. Each directory or file can be used as a lock, and reads and writes to a file are atomic.
- Bigtable uses Chubby ...
  - 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 death
  - to store Bigtable schema information (the column family information for each table); and
  - to store access control lists.
- If Chubby becomes unavailable for an extended period of time, Bigtable becomes unavailable.

Bigtable implementation (1)

- three major components:
  - a library that is linked into every client,
  - one master server, and
  - many tablet servers
- master: 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

Bigtable implementation (2)

- 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