Columnar Stores

15 November 2017

Ignacio Coterillo
Computer Engineer, CERN

Columnar Stores (1)

BigTable: A Distributed Storage System for Structured Data (https://research.google.com/archive/bigtable-osdi06.pdf)

BigTable (Google Cloud) (https://cloud.google.com/bigtable/)

Open Source Implementations of the BigTable Model: Cassandra, HBase

Cassandra Concepts (1)

```
RDBMS                          Cassandra

Instance ---------------------- Cluster
Database ---------------------- Namespace
Table    ---------------------- Column family
Row      ---------------------- Row
Column   ---------------------- Column (variable)
```

Cassandra Concepts (2)

Column: A KV pair where K is the column name, V the value and the pair is time-stamped for:
- Data expiration
- Conflict Solving
- Stale data detection

Super Column: A column where the Value is a map of standard **Columns**

Row: A collection of columns linked by a key

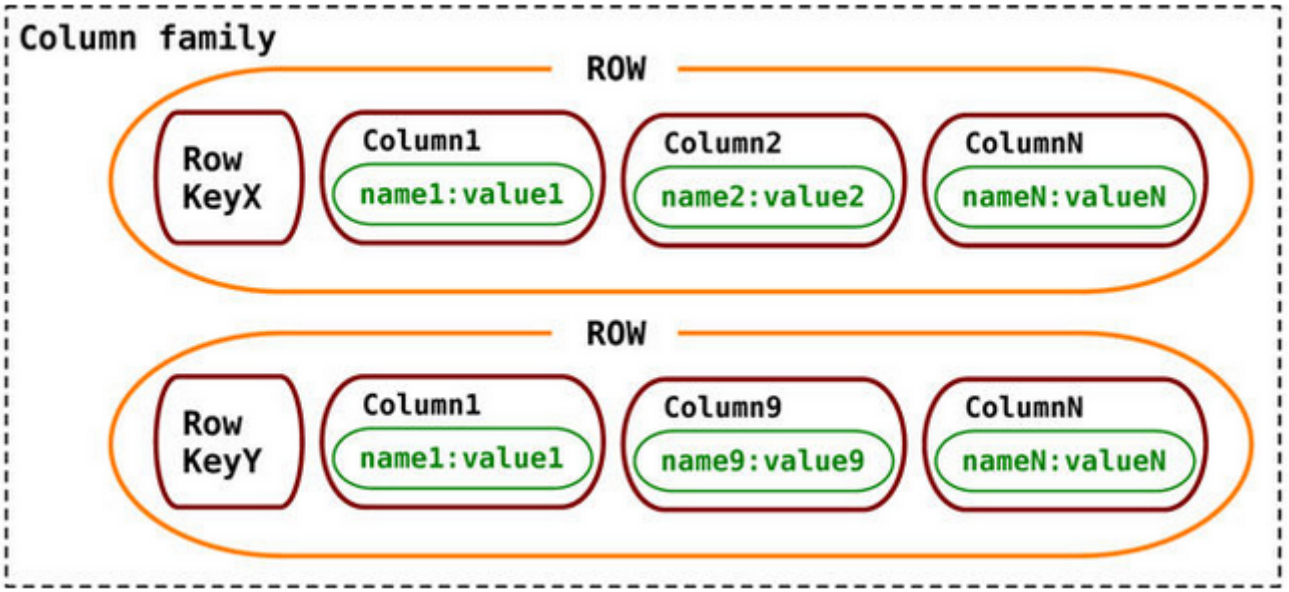
Cassandra Concepts (3)

Column family: A collection of similar Rows

- **Standard Column Family**: A **Column Family** where the rows are formed by simple columns
- **Super Column Family**: A **Column Family** where the rows are formed by **Super Columns**

Namespace: A grouping of column families of any type

Cassandra Concepts (4)



Cassandra Consistency (1)

A write operation is atomic at the row level

Inserting/updating columns for a given row key either succeeds or fails

Writes are appended to a commit log, then to a memory structure (*memtable*). In case of node failure the log is used for recovery

Once the changes are appended to the log, they are considered committed

Cassandra Consistency (2)

A Cassandra cluster is master-less. All nodes are peers

Default read consistency (**R**) is 1

Default Write consistency (**W**) is 1

It can be set to **Quorum**, or **ALL**

The number of replicas is configured during the **namespace** creation

All this needs to be tuned during per application

Cassandra facts

- CQL: Cassandra Query language

```
SELECT select_expression
FROM keyspace_name.table_name
WHERE relation AND relation ...
ORDER BY ( clustering_column  ASC | DESC ...)
LIMIT n
ALLOW FILTERING
```

- Indexes
- Views/Materialized views

Cassandra Use cases

- Write Heavy cases -> Event logging
- Very scalable
- Expiration support requirements

HBase Concepts (1)

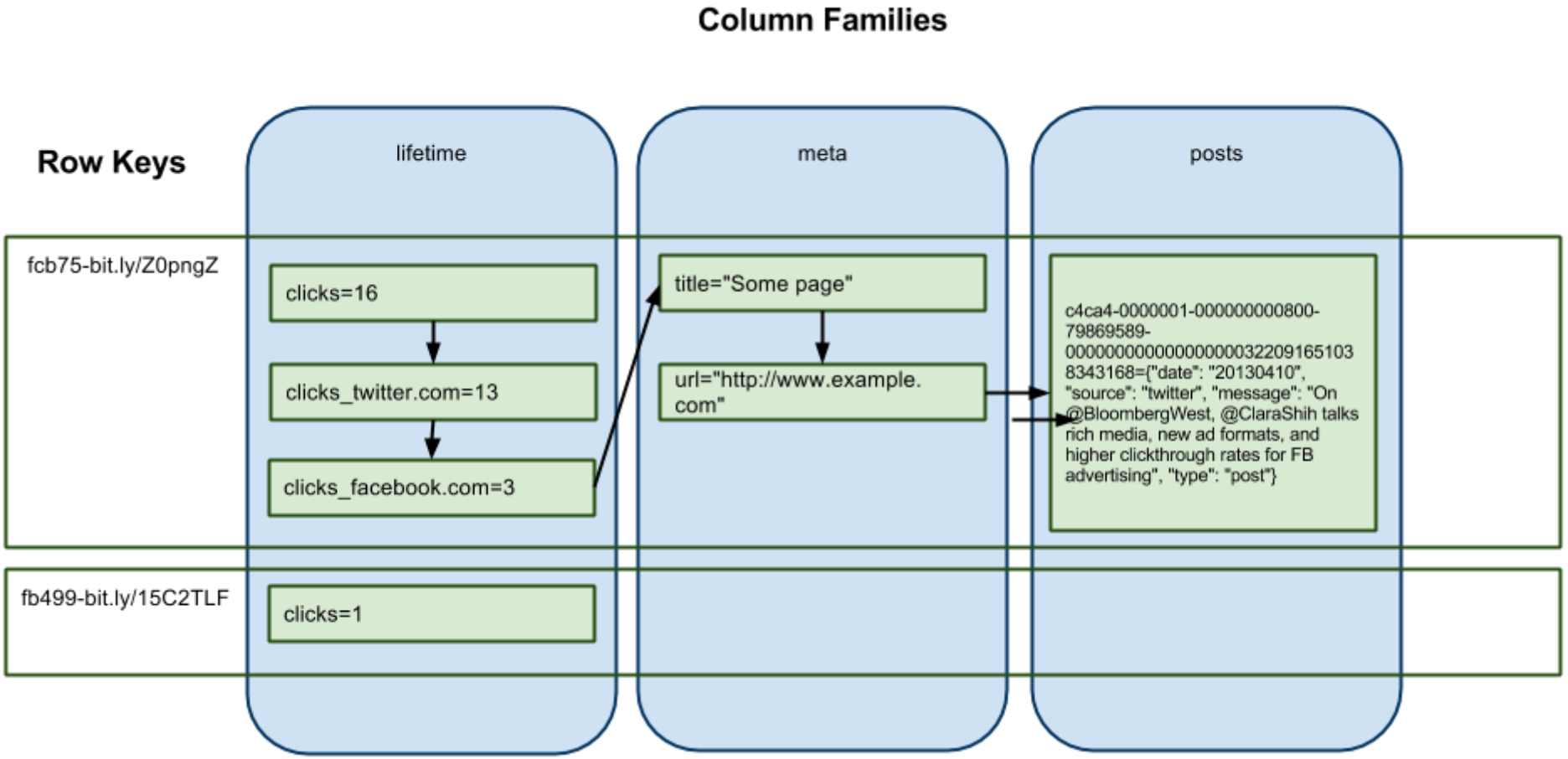
Also based on BigTable

Part of the Hadoop "Ecosystem"

A table is basically a map of maps

Individual cells can be seen as the intersection of rows and columns

HBase Concepts (2)



HBase Concepts (3)

Support in-memory tables

Supports versioning, and *garbage collection* of expired items

Specially suited for Analytics over big datasets

Uses Bloom filters for row/column detection

HBase Concepts (3)

Configuration needs to target use case and model design according to expected

- Read operations
- Write operations
- Disk use (Support for compression)

Rows are stored in order sorted by row key

A *region* is a chunk of rows. Each region is assigned to a different server

HBase Concepts: Consistency (4)

Support for strong consistency. Operations are atomic at the row level

A WAL log is enabled by default, but can be disabled dynamically (e.g. repeatable data imports)

Separate tables are distributed independently across regions, facilitating better load distribution.

It is recommended to keep the number of Column families low, always according to data access patterns.

HBase Caveats

- It is recommended to have at least a 5 nodes cluster in your deployment
- As part of Hadoop it requires setting up and configure additional components

```
- HDFS
- Zookeeper
```

- No sorting other than rowkeys

HBase Example: Facebook Messages

Facebook Messages (2011) (https://www.slideshare.net/brizzzdotcom/facebook-messages-hbase)

Facebook Messaging Index Table

- row keys are User IDs
- Column qualifiers are words that appear in that user's message
- Timestamps are message IDs of messages containing a certain word

Thank you

Ignacio Coterillo
Computer Engineer, CERN
ignacio.coterillo.coz@cern.ch (mailto:ignacio.coterillo.coz@cern.ch)