Apache Accumulo

CMSC 491
Hadoop-Based Distributed Computing
Spring 2016
Adam Shook

Overview

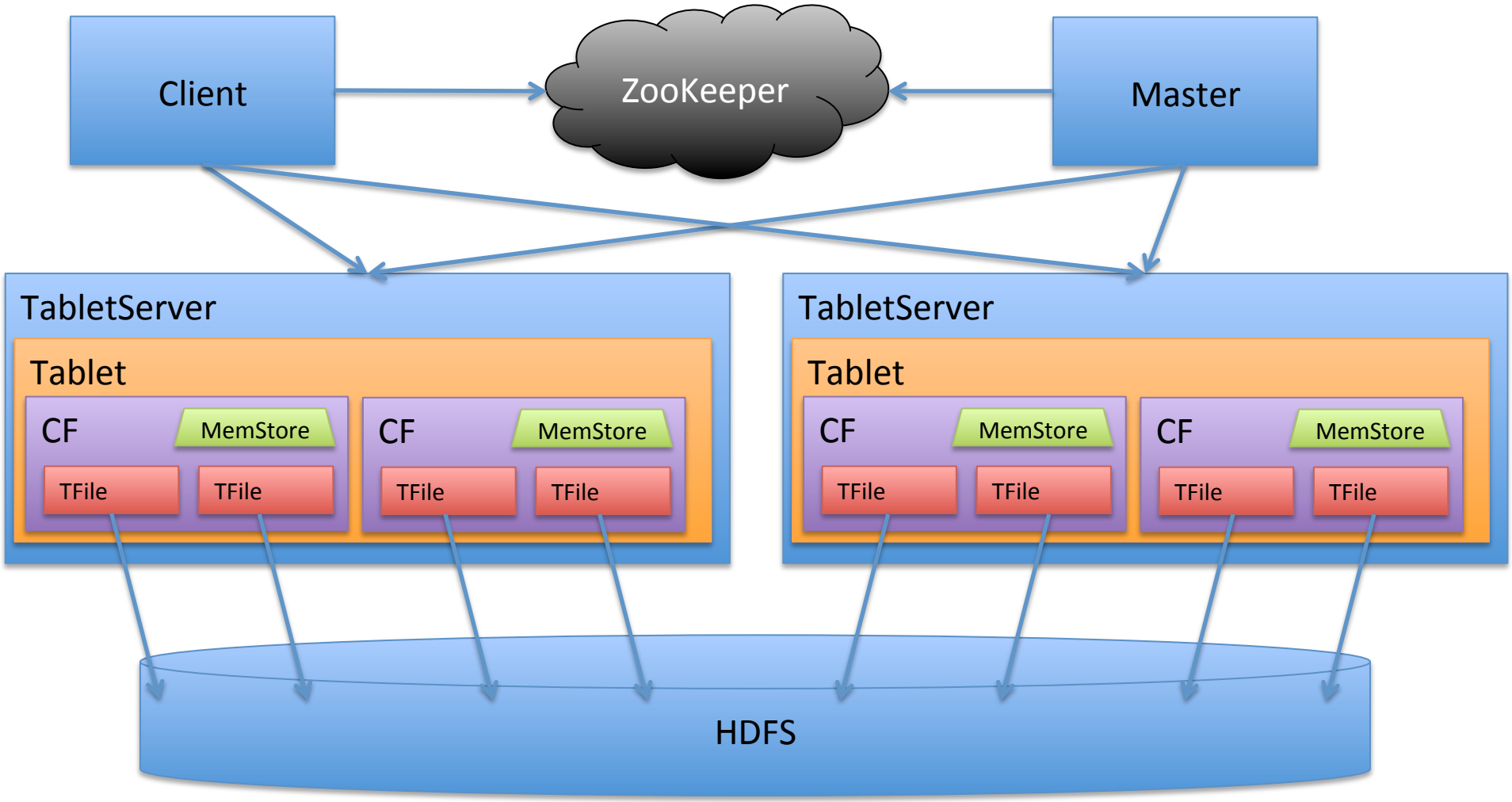
- Distributed, scalable, column-oriented key/value store
- Implementation of Google's Big Table for Hadoop
- Provides random, real-time read/write access to tables
- Billions of rows millions by millions of columns on HDFS

- Three core components
  - HBase Master
  - HBase RegionServer
  - ZooKeeper

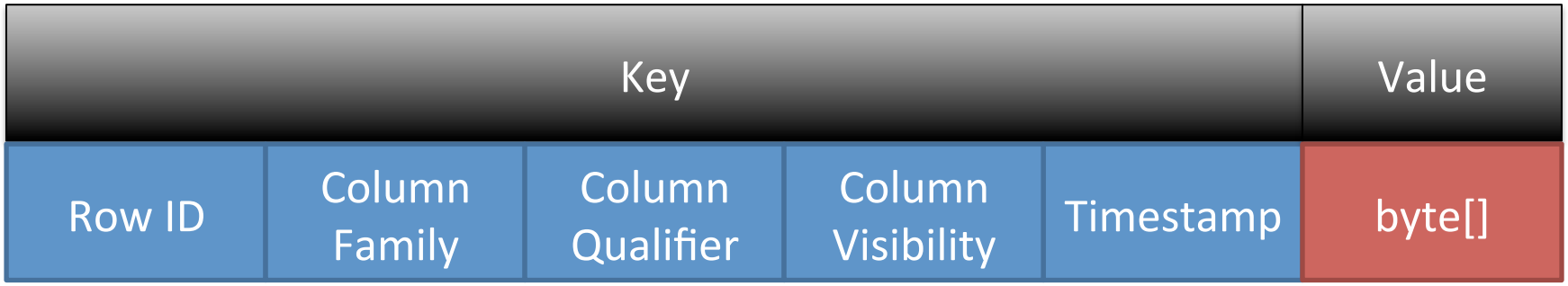
How is data stored?

- Namespace
  - Table
    - Tablet
      - Locality Group
        - » MemTable
        - » RFile
          - Block

Accumulo Architecture



Data Model


Key
Value
Row ID
Column Family
Column Qualifier
Column Visibility
Timestamp
byte[]

How is Data Stored?

'profile'
Table View

ID	Name	Created	Num Followers
158865339	FastCoDesign	1277328831000	233076
244296542	CorazoonBipolar	1298279244000	891288
255409050	Telkomsel	1306804256000	320818
326380075	WorldComedy	1309279884000	704847

Actual View

```
158865339  profile:created    1394663741975  1277328831000
158865339  profile:followers  1394663741975  233076
158865339  profile:name       1394663741975  FastCoDesign
244296542  profile:created    1394663741996  1296260757000
244296542  profile:followers  1394663741996  891288
244296542  profile:name       1394663741996  CorazoonBipolar
255409050  profile:created    1394663742000  1298279244000
255409050  profile:followers  1394663742000  320818
255409050  profile:name       1394663742000  Telkomsel
308214563  profile:created    1394663742004  1306804256000
308214563  profile:followers  1394663742004  704847
308214563  profile:name       1394663742004  WorIdComedy
```

Locality Groups

- Locality groups are a means to define different sets of columns that have different access patterns
  - Done via Column Families
  - Store metadata in one family, and images in another family
  - Set the proper column family based on what you need
- Physically separated in HDFS to provide faster access times

Locality Groups

Row ID	Column Family	Column Qualifier	Value
com.abc	content	-	<!DOCTYPE …
com.cnbc	content	-	<!DOCTYPE …
com.cnbc	link	google.com	to cnbc
com.cnn	content	-	<!DOCTYPE …
com.cnn	link	google.com	cnn.com
com.cnn	link	yahoo.com	cnn.com
com.nbc	content	-	<!DOCTYPE …
com.nbc	link	yahoo.com	NBC

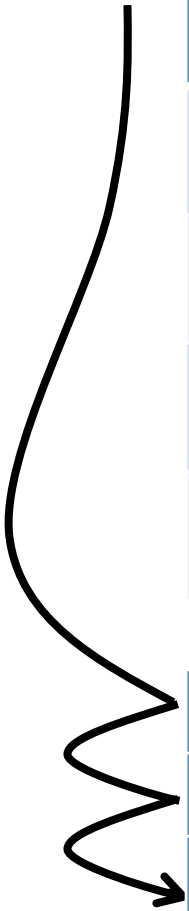
Locality Groups

Query: link data
for CNBC and CNN

Row ID	Column Family	Column Qualifier	Value
com.abc	content	-	<!DOCTYPE …
com.cnbc	content	-	<!DOCTYPE …
com.cnbc	link	google.com	to cnbc
com.cnn	content	-	<!DOCTYPE …
com.cnn	link	google.com	cnn.com
com.cnn	link	yahoo.com	cnn.com
com.nbc	content	-	<!DOCTYPE …
com.nbc	link	yahoo.com	NBC

Locality Groups

Query: link data
for CNBC and CNN

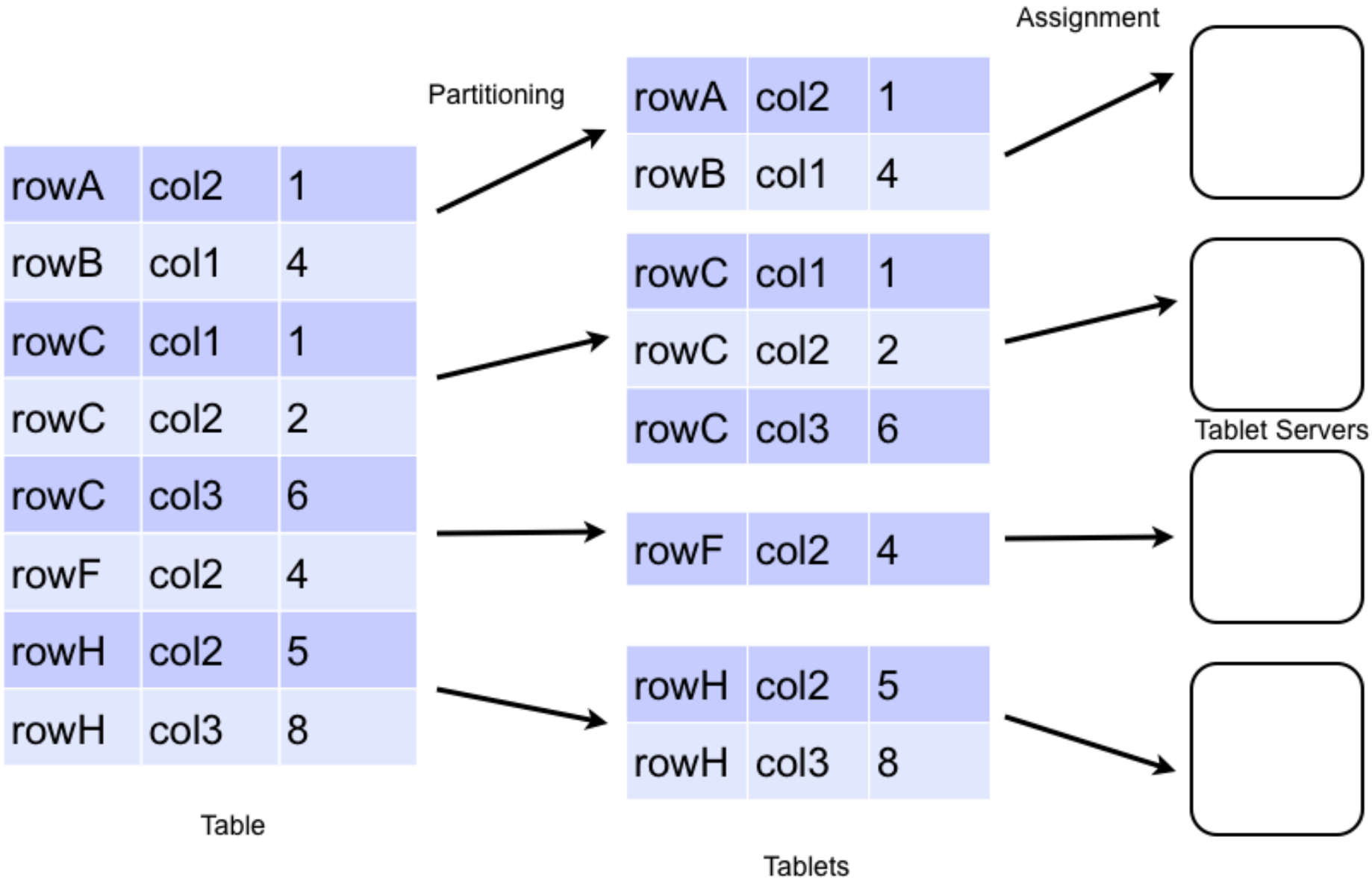


Row ID	Column Family	Column Qualifier	Value
com.abc	content	-	<!DOCTYPE …
com.cnbc	content	-	<!DOCTYPE …
com.cnn	content	-	<!DOCTYPE …
com.nbc	content	-	<!DOCTYPE …
…			
com.cnbc	link	google.com	to cnbc
com.cnn	link	google.com	cnn.com
com.cnn	link	yahoo.com	cnn.com
com.nbc	link	yahoo.com	NBC

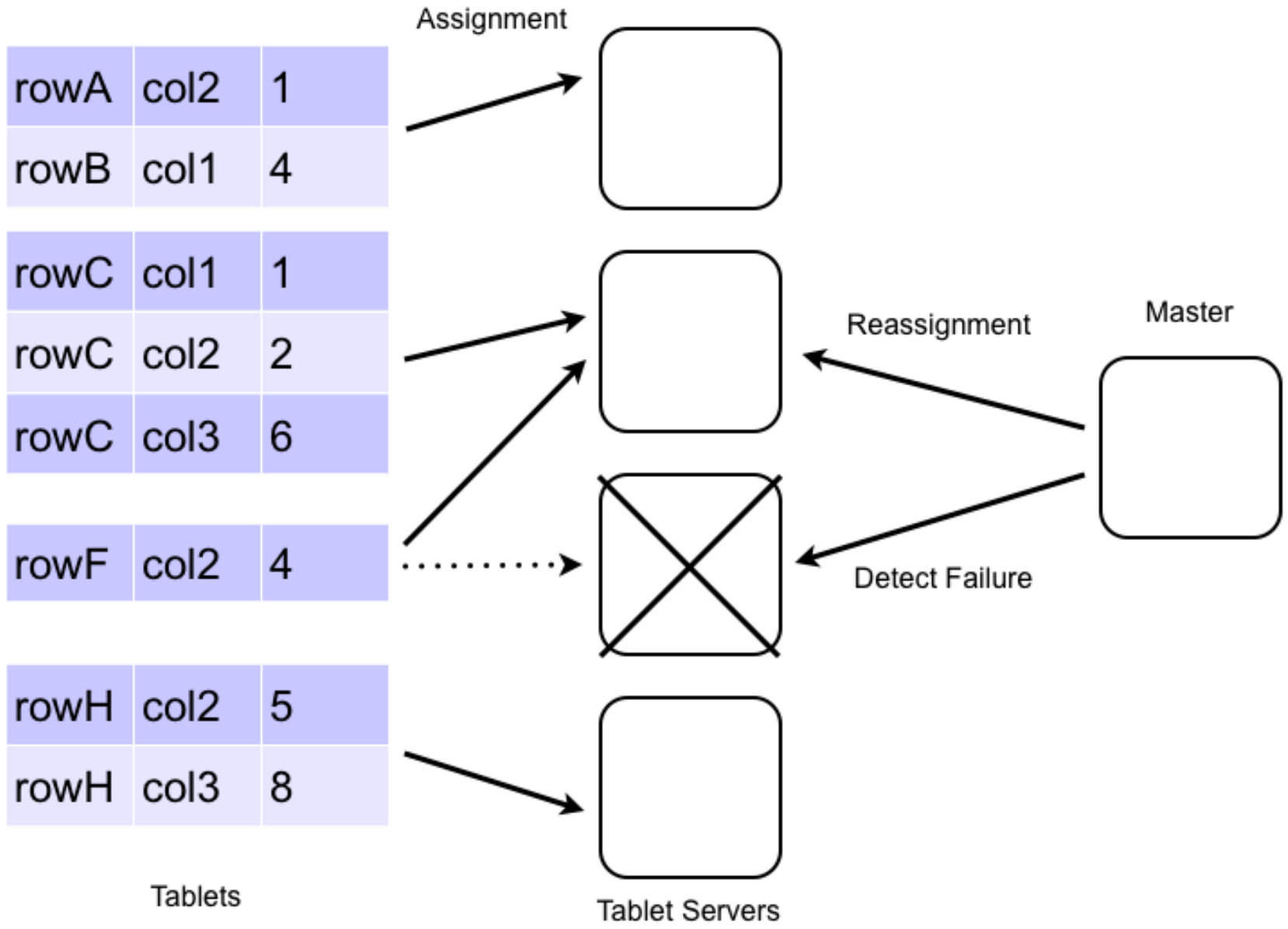
Tablets

- Tablets are split on row ID
  - i.e. you cannot have multiple key/value pairs with the same row ID in two tablets
- Tablets are indexed and Bloom filtered to give Accumulo TabletServers the ability to quickly seek into an HDFS block and get the data

Data Distribution



Automatic Failure Handling



Bloom Filters and Block Caching

- Use these for optimal fetch performance!
- Bloom Filters
  - Stored in memory on each TabletServer
  - Used as a preliminary test prior to opening a region on HDFS
  - Very effective for fetches that are likely to have a null value
- Block Caching
  - Configurable number of key/value pairs to read into memory when a TabletServer fetches data
  - Very effective for multiple fetches with similar keys

Compactions

- Minor
  - Flushing of the MemTable to an RFile
- Major
  - Merging of RFiles into a single RFile
  - All expired cells will be dropped
    - Does not occur in minor compactions

Creating and Managing Tables

- Tables contain Column Families
- You can (and should) pre-define your table split keys
  - Defines the regions of a table
  - Allows for better data distribution, especially when doing a bulk-load of data
- Accumulo will split regions automatically as needed
  - Master has no part in this
- Lower number of tablets preferred, in the range of 20 to low-hundreds per TabletServer
- Can split manually

Bulk Importing

- Create table
- Use MapReduce to generate RFiles in batch
- Tell Accumulo where the files are
- Drastically reduces run-time for table ingestion

What can I do with it?

- Accumulo is designed for fast fetches (~10ms) of your big data sets
- Random Inserts/Updates/Deletes of data
- Versioning
- Changing schemas

What shouldn't I do with it?

- Full-table scans
  - Slow
  - Use MapReduce instead (still slow)
- High-throughput transactions
  - Use Redis or another in-memory solution for data sets that can fit in-memory
- Monotonically Increasing Row IDs
  - There are work arounds!

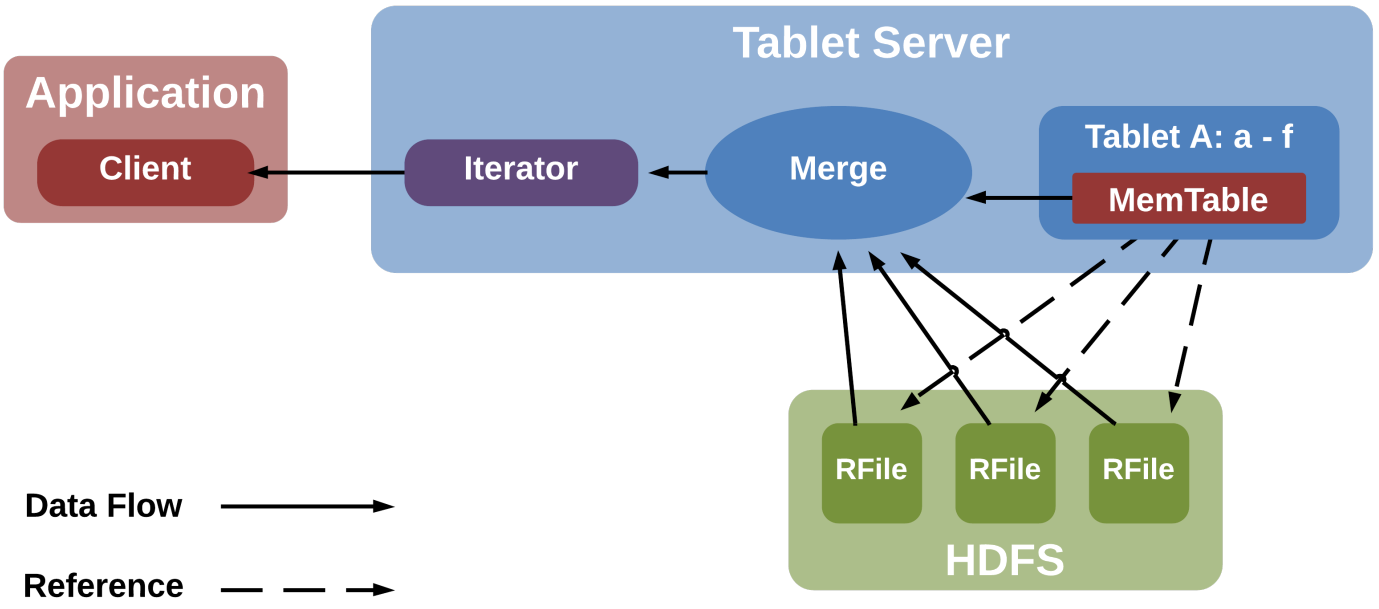
Features Include

- Creating/Deleting Tables
- Major/Minor Compactions
- Bloom Filters/Block Caching
- Bulk Importing
- Data manipulation via *Mutations*
- Two Types of Range Scans
  - Scanner vs Batch Scanner
- **Iterators**

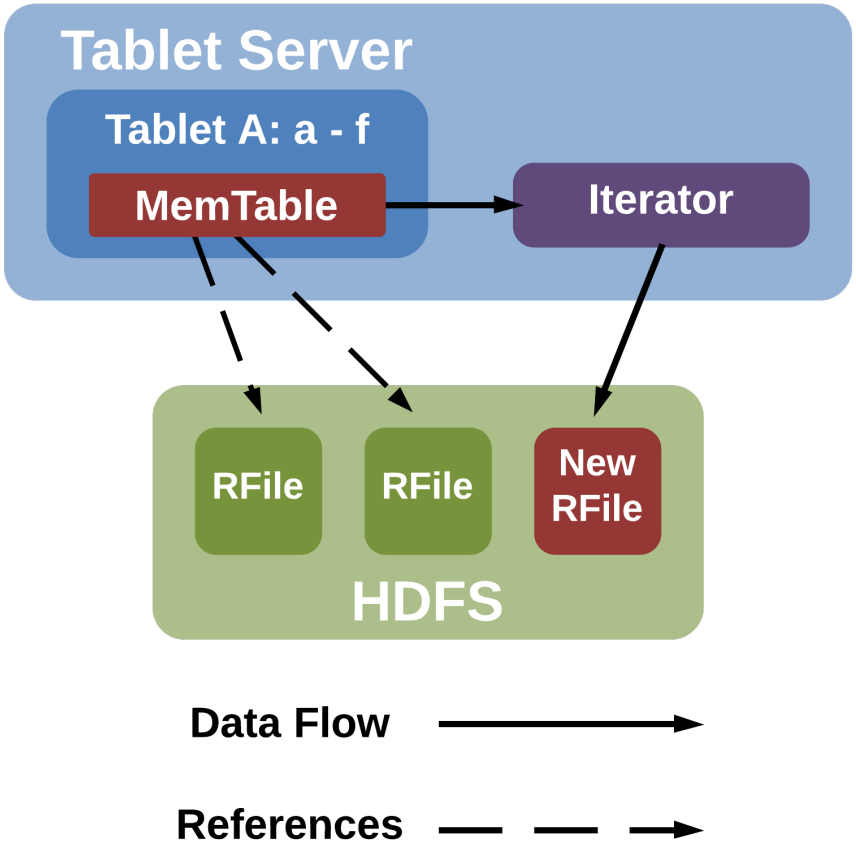
Iterators

- Real-Time processing framework
- Provide "Reduce-like" functionality, but at very low latency
- Iterators are configured to run at:
  - Scan time
  - Minor Compaction
  - Major Compation
- AgeOffIterator – automatically age off key/value pairs during scans and compactions

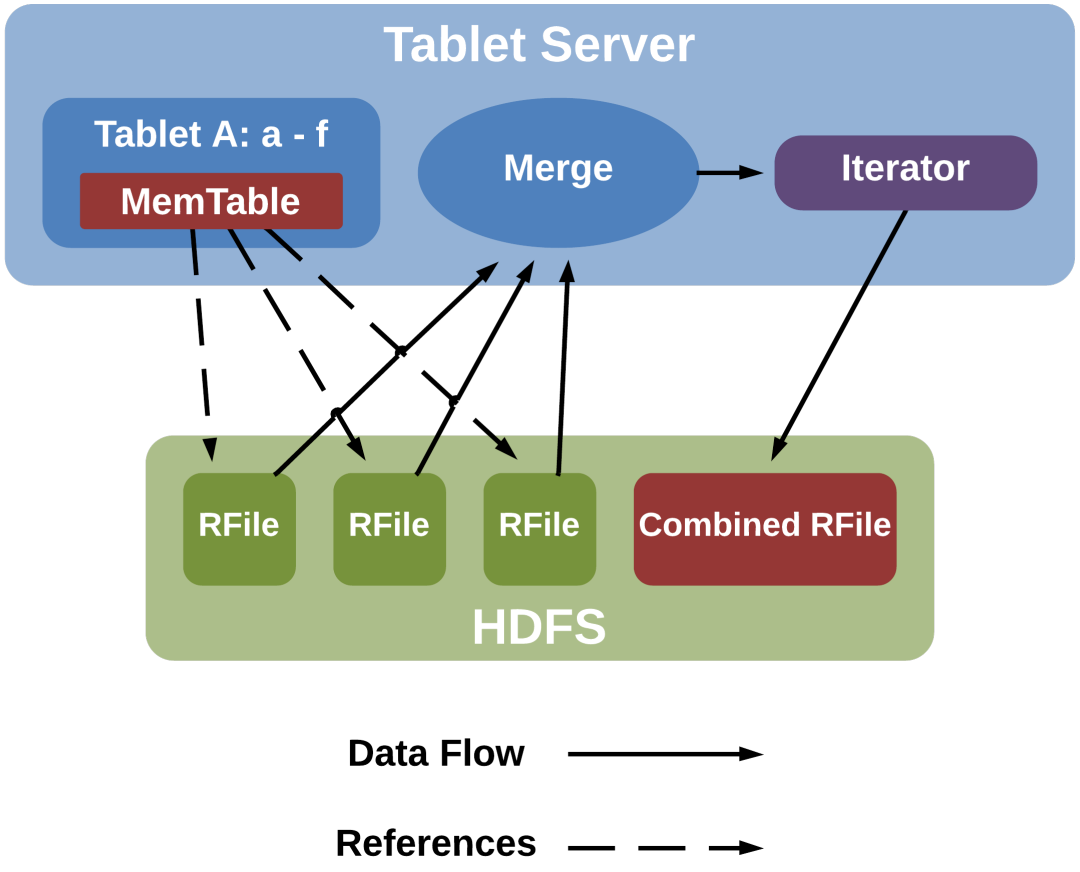
Scan Time Iterator



Minor Compaction Iterator



Major Compaction Iterator



Iterator Types

- Versioning
  - Configure the number of identical key/value pairs to store
- Filtering
  - Apply arbitrary filtering to key/value pairs
- Combiners
  - Aggregate values from keys that shares a Row ID, Column Family, and Column Qualifier

Versioning Iterator

Given multiple version of the same row, what operations can we perform?

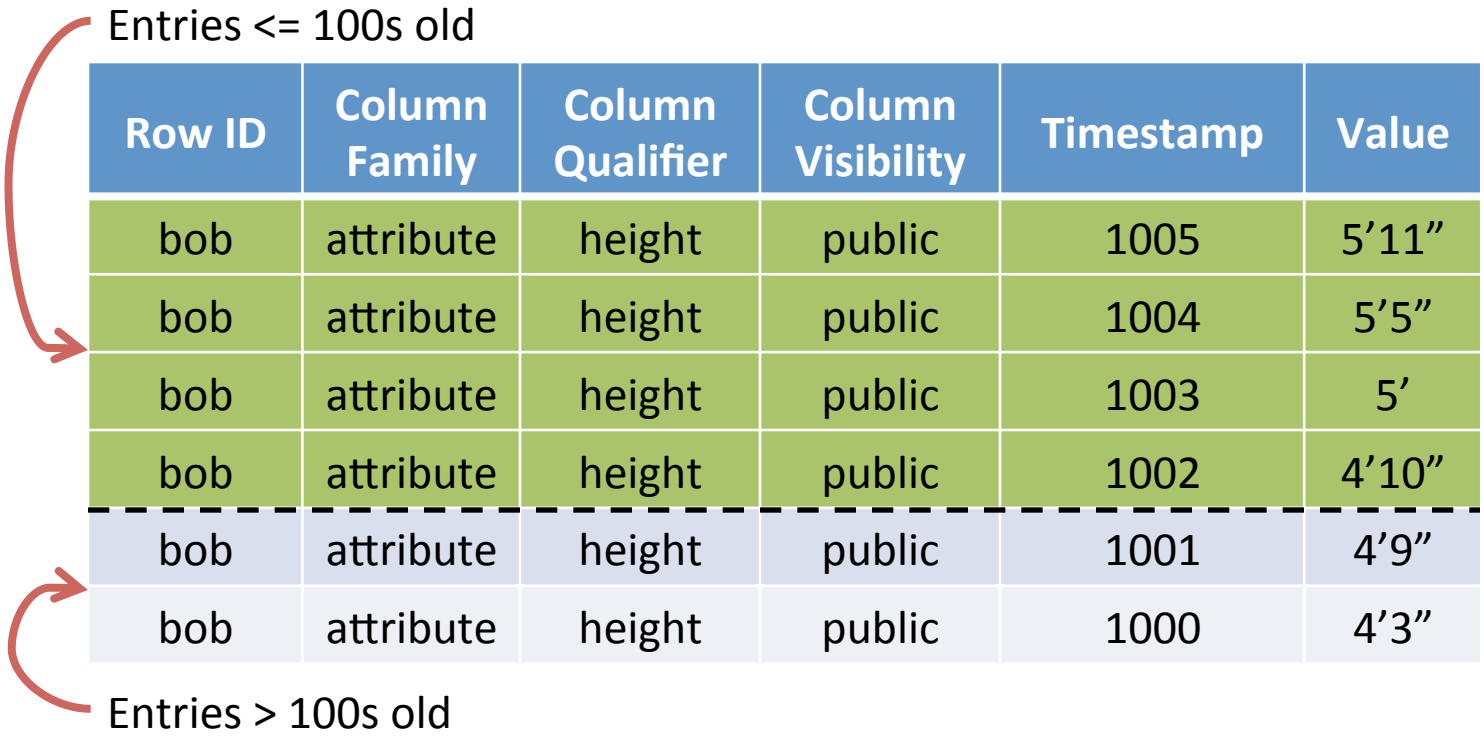
Row ID	Column Family	Column Qualifier	Column Visibility	Timestamp	Value
bob	attribute	height	public	1005	5’11”
bob	attribute	height	public	1004	5’5”
bob	attribute	height	public	1003	5’
bob	attribute	height	public	1002	4’10”
bob	attribute	height	public	1001	4’9”
bob	attribute	height	public	1000	4’3”

Versioning Iterator

Row ID	Column Family	Column Qualifier	Column Visibility	Timestamp	Value
bob	attribute	height	public	1005	5’11”
bob	attribute	height	public	1004	5’5”
bob	attribute	height	public	1003	5’
bob	attribute	height	public	1002	4’10”
bob	attribute	height	public	1001	4’9”
bob	attribute	height	public	1000	4’3”

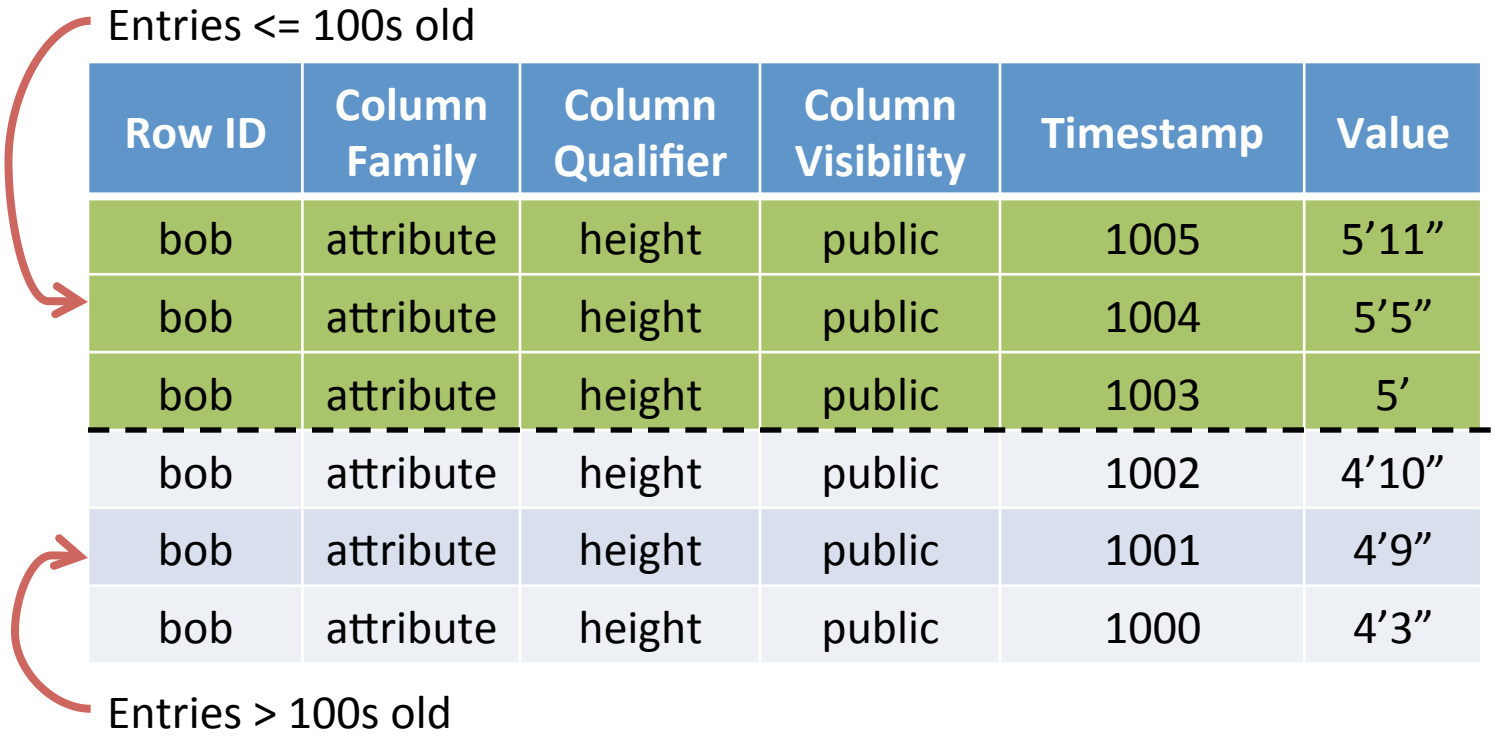
Age-Off Iterator

Current Time: 1102



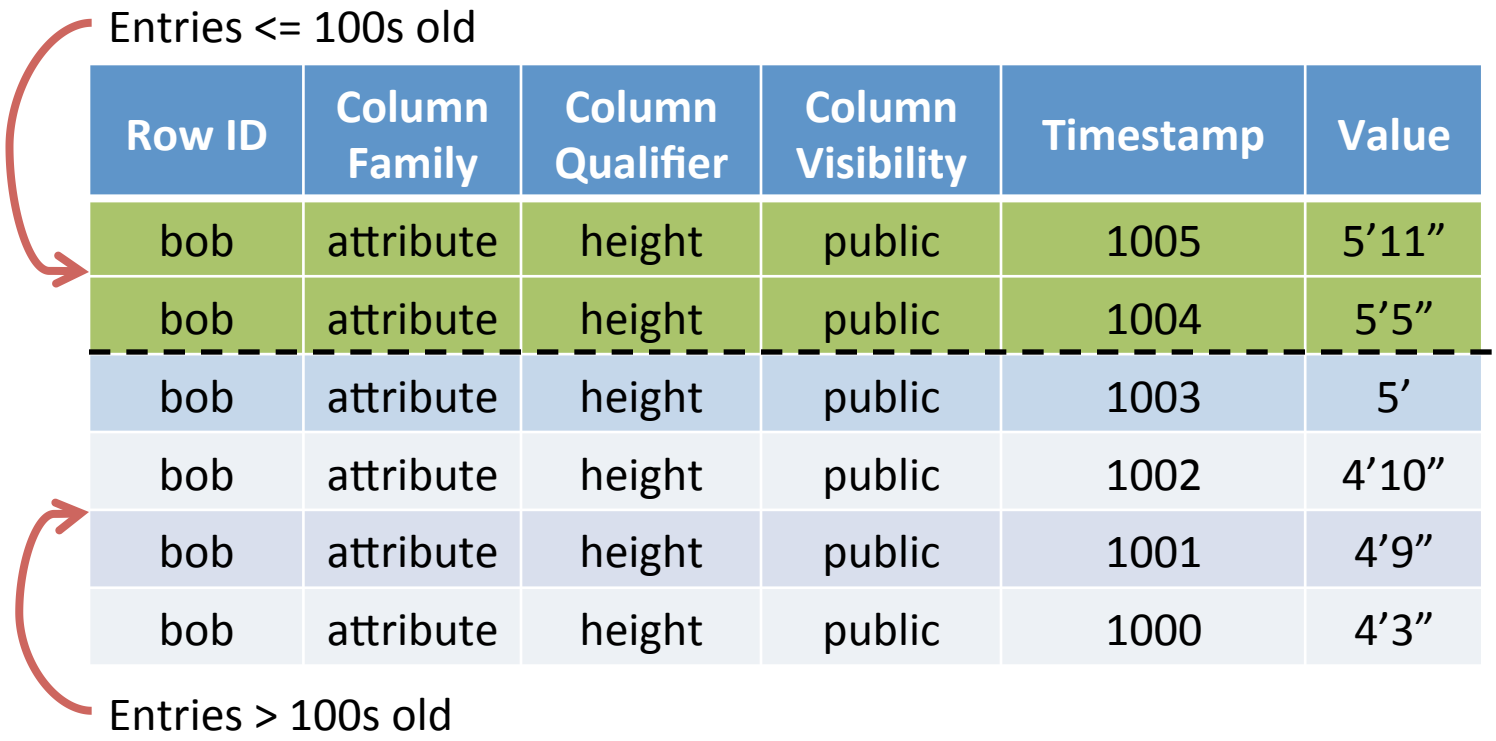
Age-Off Iterator

Current Time: 1103



Age-Off Iterator

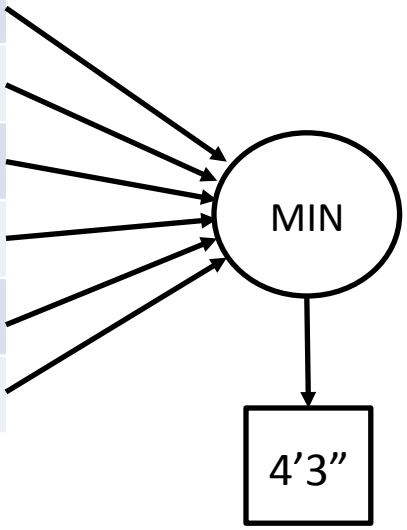
Current Time: 1104

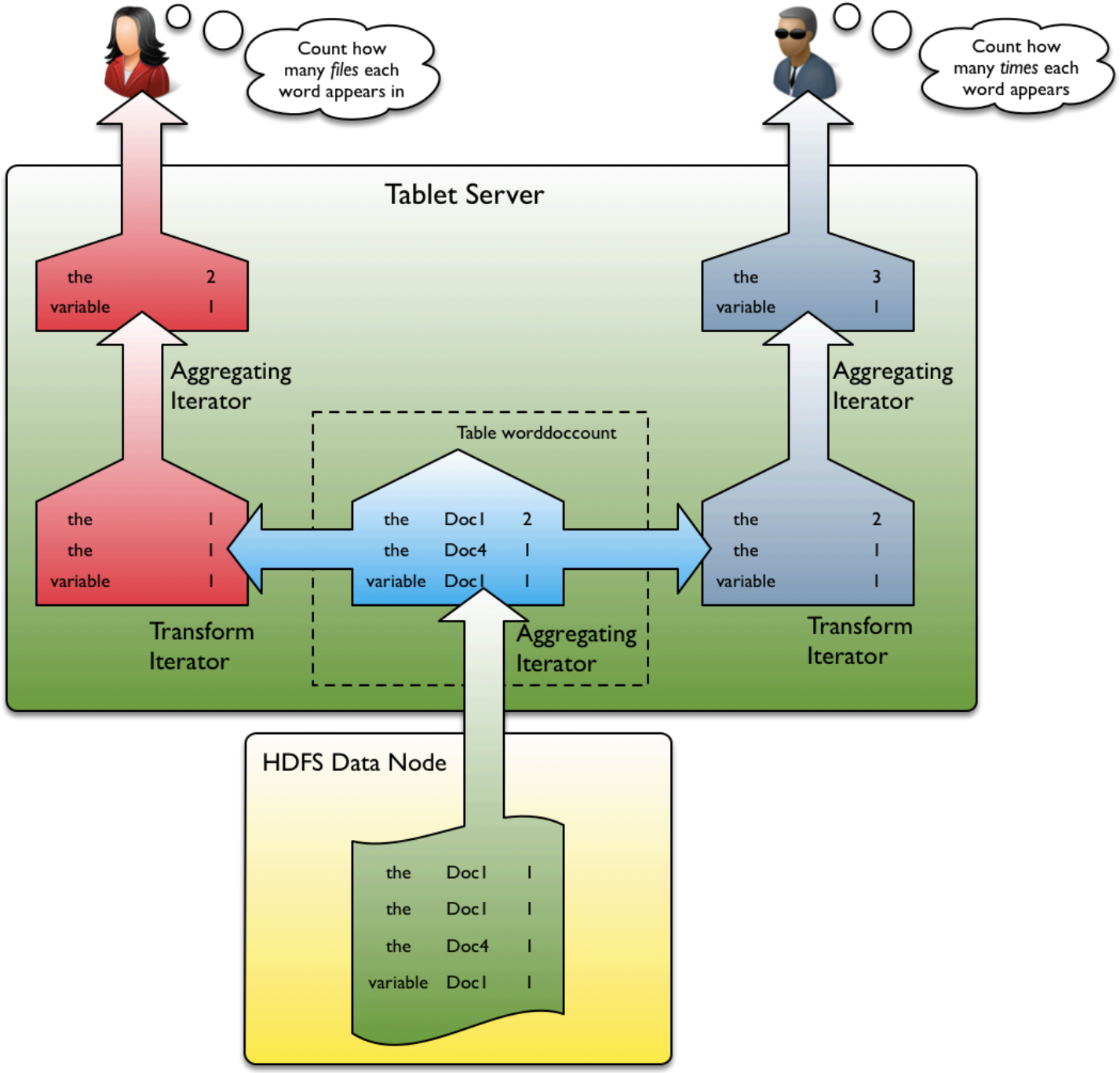


Combiner Iterators

Apply a function to all available versions of a particular key

Row ID	Column Family	Column Qualifier	Column Visibility	Timestamp	Value
bob	attribute	height	public	1005	5’11”
bob	attribute	height	public	1004	5’5”
bob	attribute	height	public	1003	5’
bob	attribute	height	public	1002	4’10”
bob	attribute	height	public	1001	4’9”
bob	attribute	height	public	1000	4’3”




Count how many *files* each word appears in
Count how many *times* each word appears
Tablet Server
the 2
variable 1
the 3
variable 1
Aggregating Iterator
Aggregating Iterator
Table worddoccount
the 1
the 1
variable 1
the Doc1 2
the Doc4 1
variable Doc1 1
the 2
the 1
variable 1
Transform Iterator
Aggregating Iterator
Transform Iterator
HDFS Data Node
the Doc1 1
the Doc1 1
the Doc4 1
variable Doc1 1

References

- http://accumulo.apache.org