Hbase

STARTSEV D.S., BIT, MIPT 2022,

План

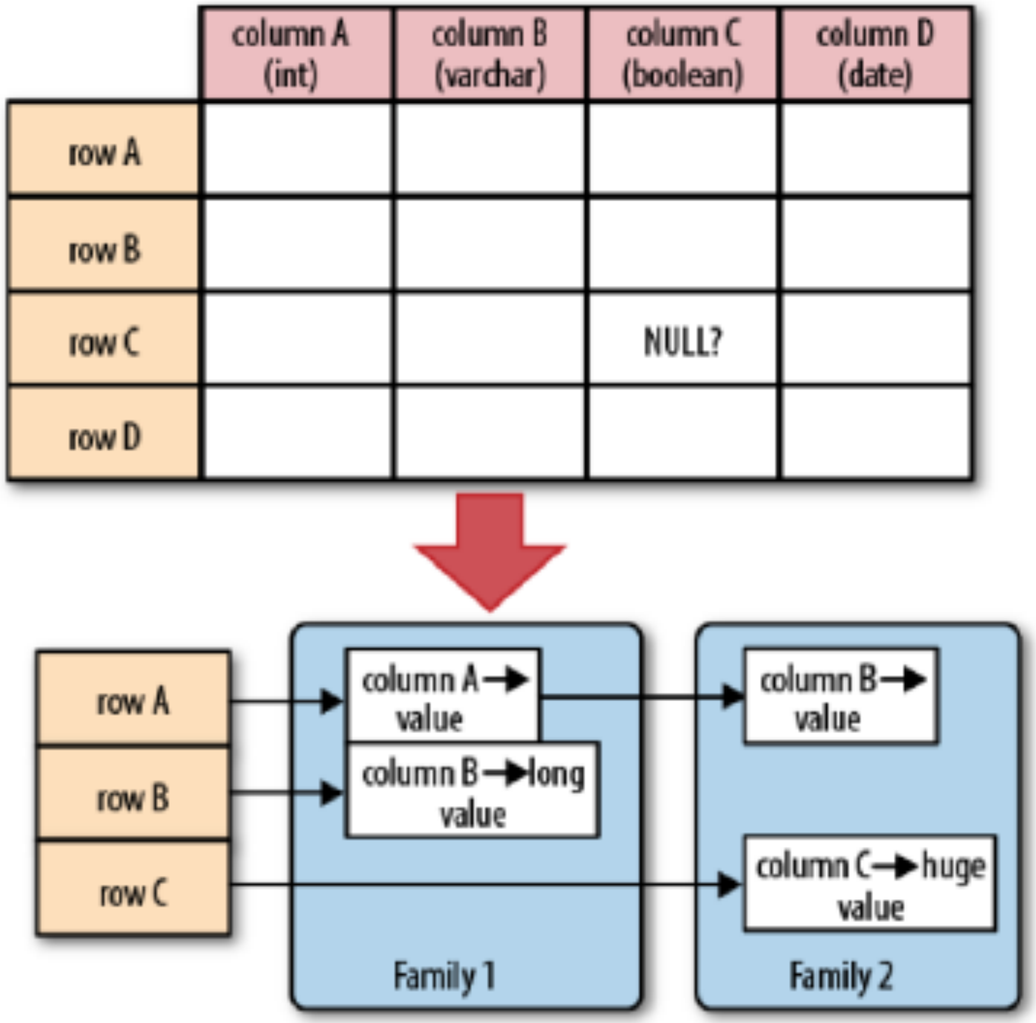
Introduction

- NOSQL
- Distributed
- Wide-Column family
- Key-Value multi-version storage
- HDFS based

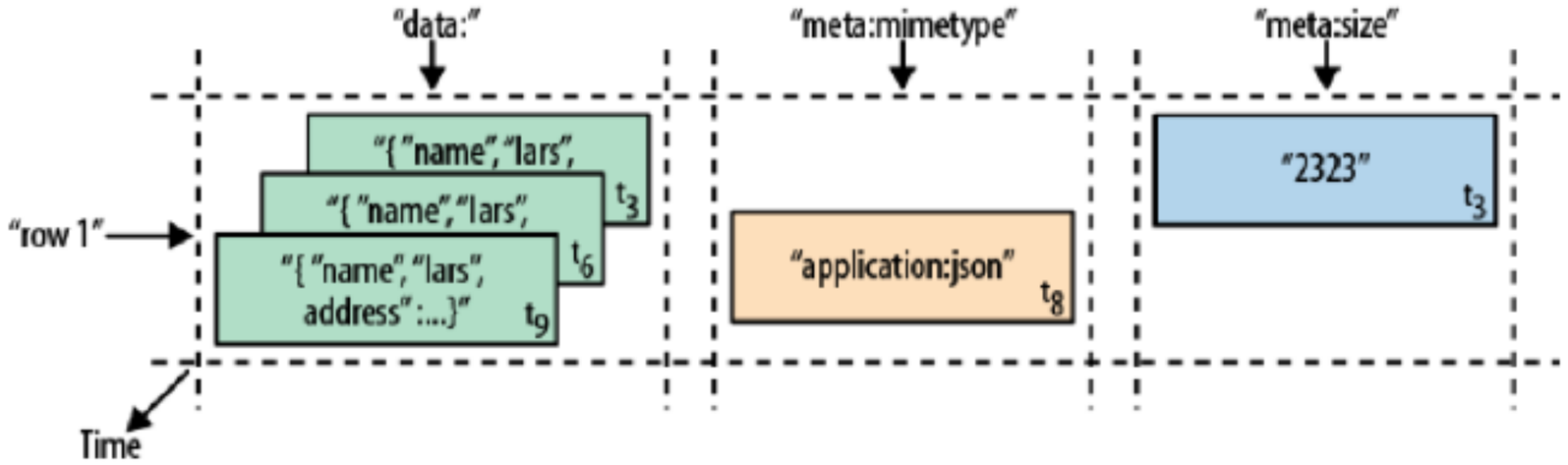
Data model

- <table, RowKey, Column Family, Column, timestamp> -> Value
- TTL for each CF
- Timestamps for each attribute
- Sorted by RowKey
- Free NULL

Table -> Wide Column



Example

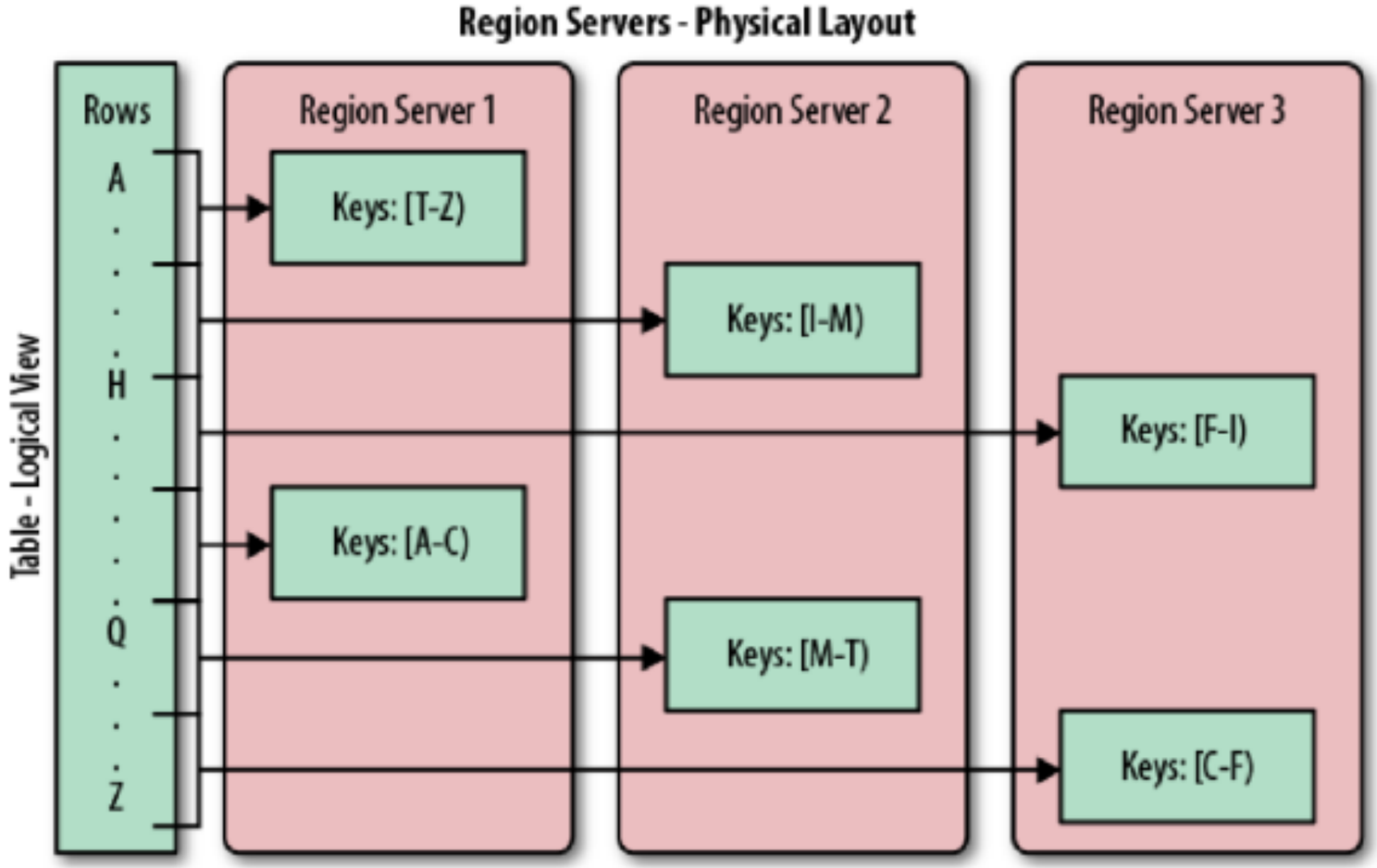

"data:"
"meta:mimetype"
"meta:size"
"row 1"
"{ "name", "lars",
"{ "name", "lars",
"{ "name", "lars",
address":...}"
t3
t6
t9
"application:json"
t8
"2323"
t3
Time

Wide Column -> Table

Row Key	Time Stamp	Column "data:"	Column "meta:" "mimetype"	"size"	Column "counters:" "updates"
"row1"	t_3	"{ "name": "lars", "address": ...}"		"2323"	"1"
	t_6	"{ "name": "lars", "address": ...}"			"2"
	t_8		"application/json"		
	t_9	"{ "name": "lars", "address": ...}"			"3"

Architecture

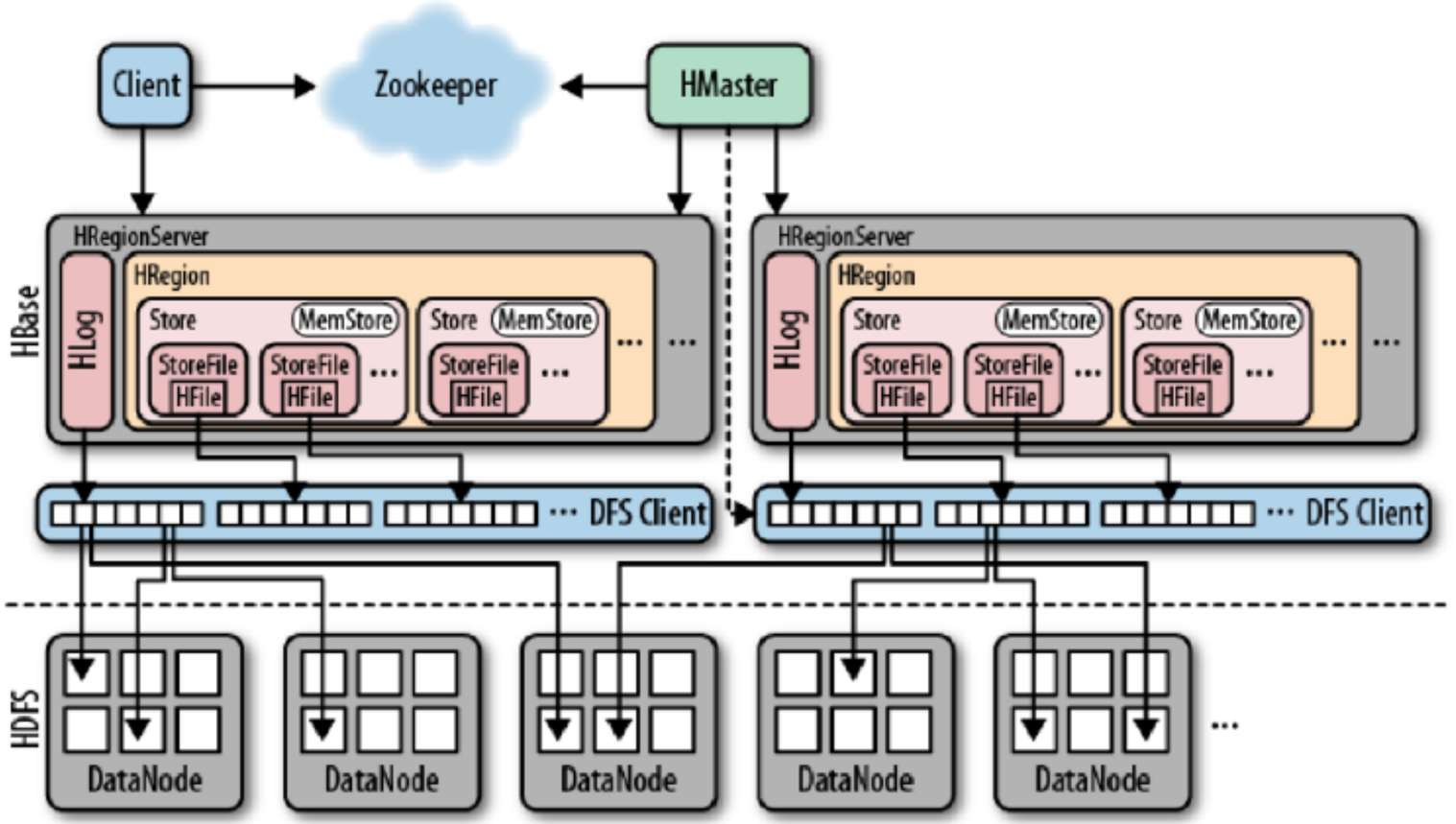
Regions


Region Servers - Physical Layout
Table - Logical View
Rows
A
H
Q
Z
Region Server 1
Keys: [T-Z)
Keys: [A-C)
Region Server 2
Keys: [I-M)
Keys: [M-T)
Region Server 3
Keys: [F-I)
Keys: [C-F)

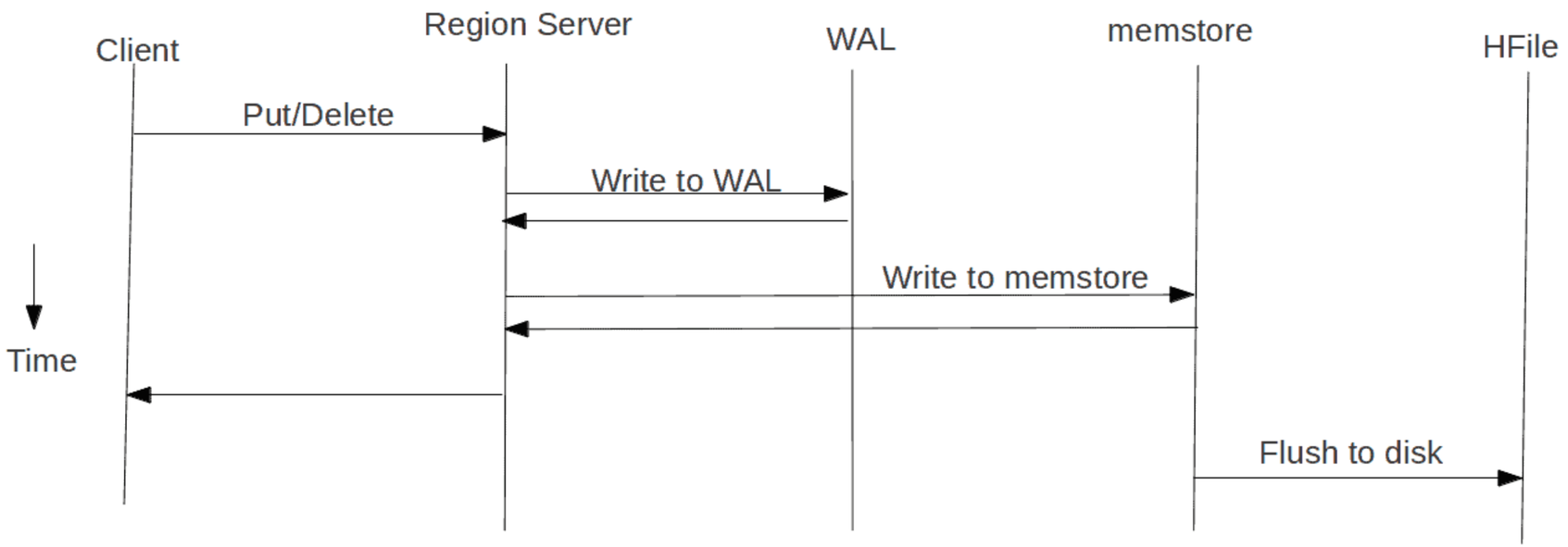
Regions

- Data storage
- MemStore
- BlockCache
- Write Ahead Log

Zookeeper, ↕ ↕ Master && Slaves ↕ ↕



Writing

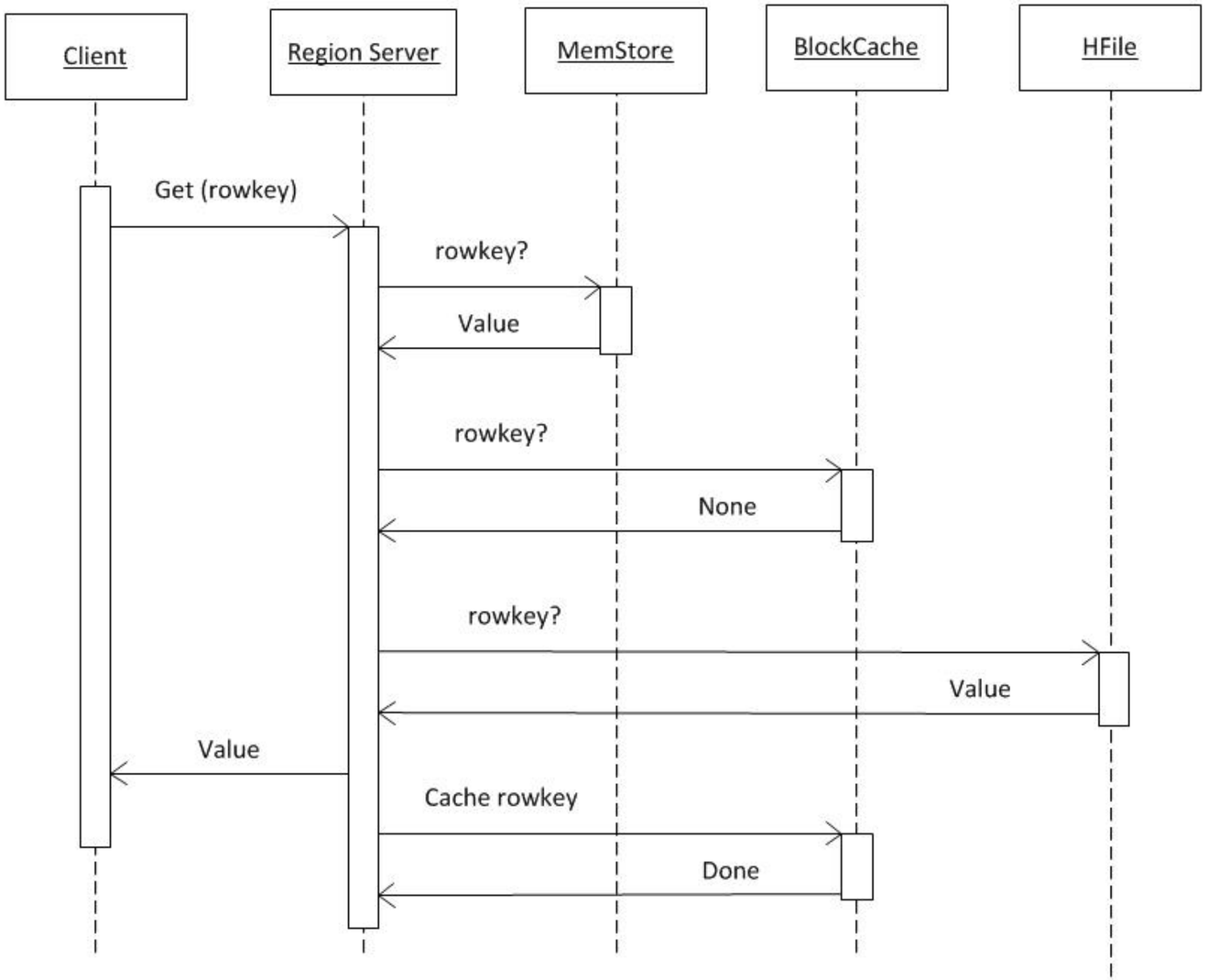


HBase Write Path

Compactions

- HFile merging
- Minor
- Major

Reading


Client
Region Server
MemStore
BlockCache
HFile
Get (rowkey)
rowkey?
Value
rowkey?
None
rowkey?
Value
Value
Cache rowkey
Done

CRUD

Instruments

- HBase Shell
- Native Java API
- Thrift API
- Rest API

WE USE SHELL!!!

Create

- Creates new table
- create ‘<table name>’,’<column family>’
- create ‘<table name>’, {NAME => ‘column family1’, VERSIONS => N}

Put

- Add new entry or update the current
- put ‘<table name>’,’row1’,’<colfamily:colname>’,’<value>’

Get

- Get entry by KeyRow
- get ‘<table name>’,’rowid’
- get ‘<table name>’,’rowid’, {COLUMN ⇒ ‘column family:column name’, VERSIONS => N}

Scan

- Return entries one by one
- scan ‘<table name>’
- scan ‘<table name>’, {COLUMNS => ['c1', ‘c2'], LIMIT => N, STARTROW => ‘rowkey1’}
- scan ‘<table name>’, {TIMERANGE => [T1, T2]}
- scan ‘<table name>’, {ROWPREFIXFILTER => ‘prefix', FILTER => (QualifierFilter (>=, 'binary:xyz')) AND (TimestampsFilter (T1, T2))}

Delete

- Tombstone mark for entry
- delete ‘<table name>’, ‘<row>’, ‘<column name >’, ‘<timestamp>’

Indices

Indices

- RowKey is primary index
- No support for other types
- Table transposition may help

Transactions

Transactions

- No transactions
- Atomicity of operations at a row-level
- Scan guaranteeses

Back up

Back up

- Replication mechanism
- Snaphosts

Security

Security

- RBAC
- VL
- Data encryption in some states

Conclusion

Key points

- Java and HDFS based
- Distributed
- Open Source
- Horizontal sharding
- No Indicies && Transaction
- No «execution plan»
- BigData && Data Mining

Useful links

- Official site
- Documentation
- Git
- Graceful lecture about DFS
- Docker && Hbase
- More about Zookeeper

Thanks for attention!