# HBase

# FEATURES OF NOSQL DATABASES

**Generic data model**

➢ Heterogeneous containers, including sets, maps, and arrays

**Dynamic type discovery and conversion**

➢ NoSQL analytics systems support runtime type identification and conversion so that custom business logic can be used to

dictate analytic treatment of variation.

**Non-relational and De-normalised**

➢ Data is stored in single tables as compared to joining multiple tables.

**Commodity hardware**

- Adding more of the economical servers allows NoSQL databases to scale to handle more

data.

**Highly distributable**

- Distributed databases can store and process a set of information on more than one

device.

# WHAT IS HBASE?

Apache HBase is the Hadoop database, a distributed, column oriented, scalable, big data store.

➢ Use Apache HBase when you need random, realtime read/write access to your Big Data.

➢ This project's goal is the hosting of very large tables - billions of rows, X millions of columns - atop clusters of commodity

hardware.

➢ Apache HBase is an open-source, distributed, versioned, non-relational database modelled after Google's Big table: A

Distributed Storage System for Structured Data by Chang et al.

➢ Just as Big table leverages the distributed data storage provided by the Google File System, Apache HBase provides Big

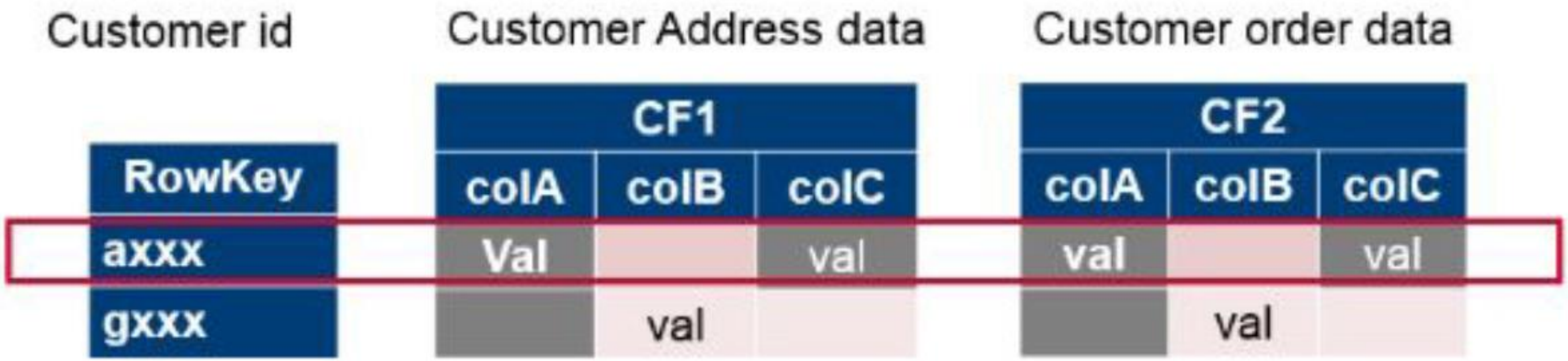
table-like capabilities on top of Hadoop and HDFS.

# HBASE DATA MODEL – COLUMNS & COLUMN FAMILIES

```
{
“1” : {
“A” : {
“foo”: “x”
},
“aaaaa” : {
“A” : {
“foo” : “y”
},
“aaaab” : {
“A” : {
“foo” : “world”
},
```

1. Top Level Key/map pair is called row
2. A is called as Column Family
3. Foo is the column of family A

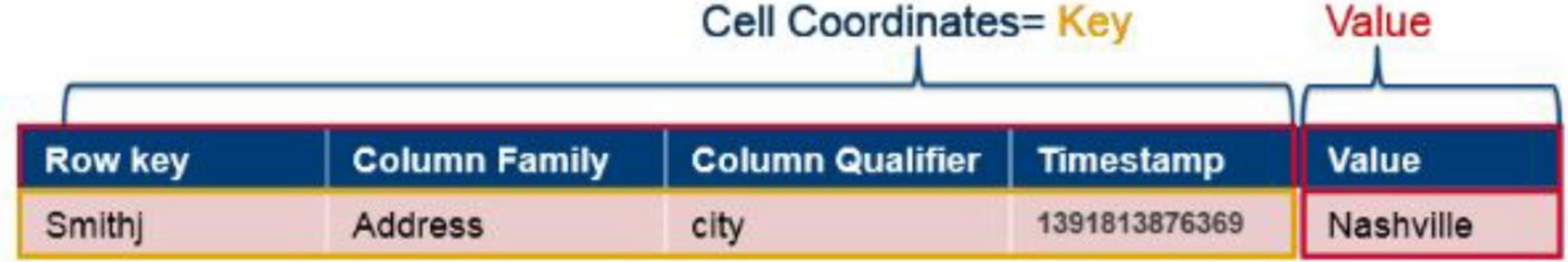
# HBASE DATA MODEL – LOGICAL REPRESENTATION



| Customer id | Customer Address data | | | Customer order data | | |
|---|---|---|---|---|---|---|
| | CF1 | | | CF2 | | |
| RowKey | colA | colB | colC | colA | colB | colC |
| axxx | Val | | val | val | | val |
| gxxx | | val | | | val | |

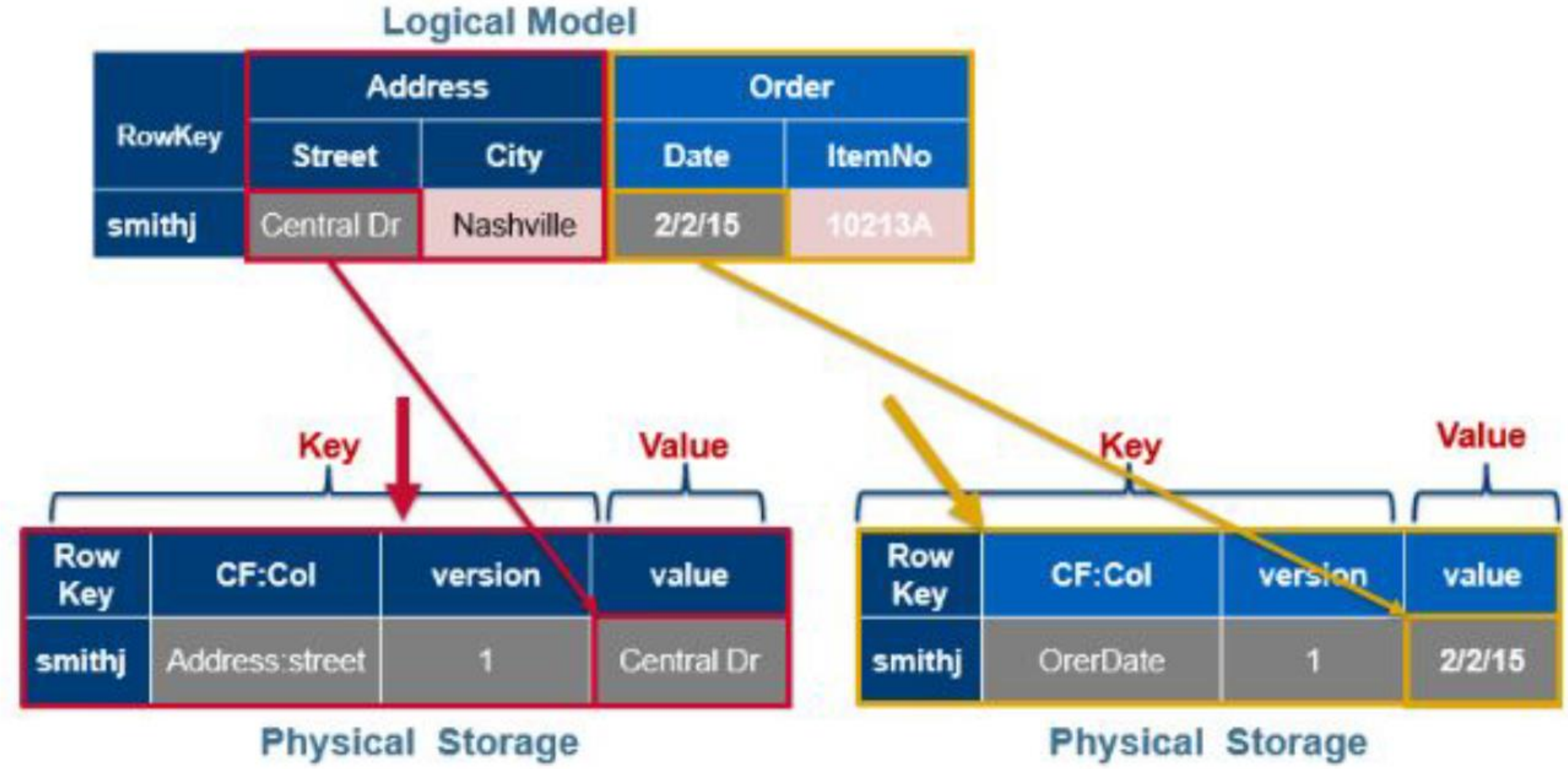
Data is accessed and stored together:

- RowKey is the primary index
- Column Families group similar data by row key



| Row key | Column Family | Column Qualifier | Timestamp | Value |
|---|---|---|---|---|
| Smithj | Address | city | 1391813876369 | Nashville |

# LOGICAL VERSUS PHYSICAL REPRESENTATION



Logical Model

| RowKey | Address | | Order | |
|---|---|---|---|---|
| | Street | City | Date | ItemNo |
| smithj | Central Dr | Nashville | 2/2/15 | 10213A |

| Row Key | CF:Col | version | value |
|---|---|---|---|
| smithj | Address:street | 1 | Central Dr |

Physical Storage

| Row Key | CF:Col | version | value |
|---|---|---|---|
| smithj | OrerDate | 1 | 2/2/15 |

Physical Storage

# CAP Theorem

**Consistency** - This means that the data in the database remains consistent after the execution of an operation. For

example after an update operation, all clients see the same data.

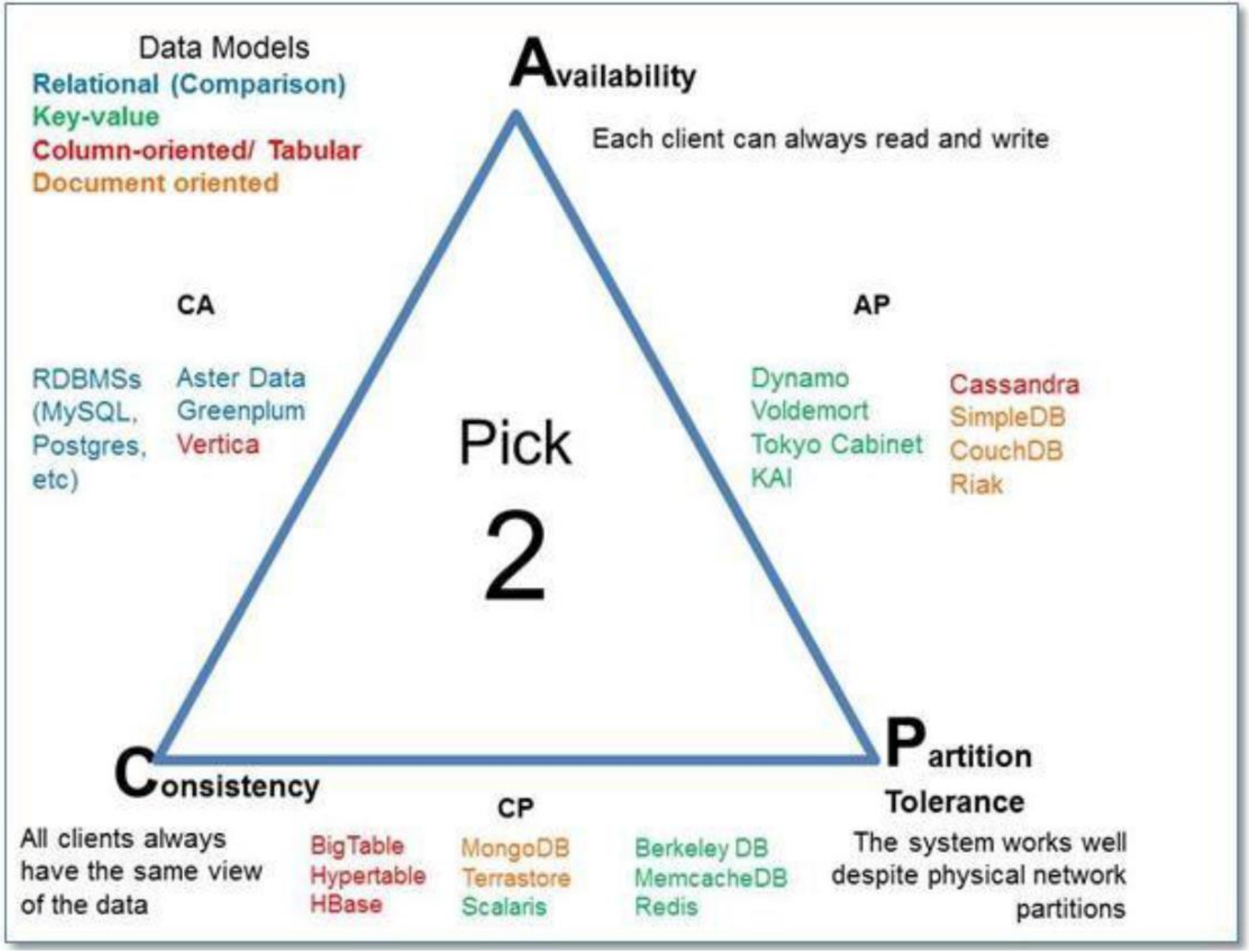
- **Availability** - This means that the system is always on (service guarantee availability), no downtime.

- **Partition Tolerance** - This means that the system continues to function even if the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

Duplicate Copy of same data is maintained on multiple machines.

- This increases availability, but decreases consistency.

# CAP


Data Models
Relational (Comparison)
Key-value
Column-oriented/ Tabular
Document oriented
Availability
Each client can always read and write
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick 2
Consistency
All clients always have the same view of the data
CP
BigTable
Hypertable
HBase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance
The system works well despite physical network partitions

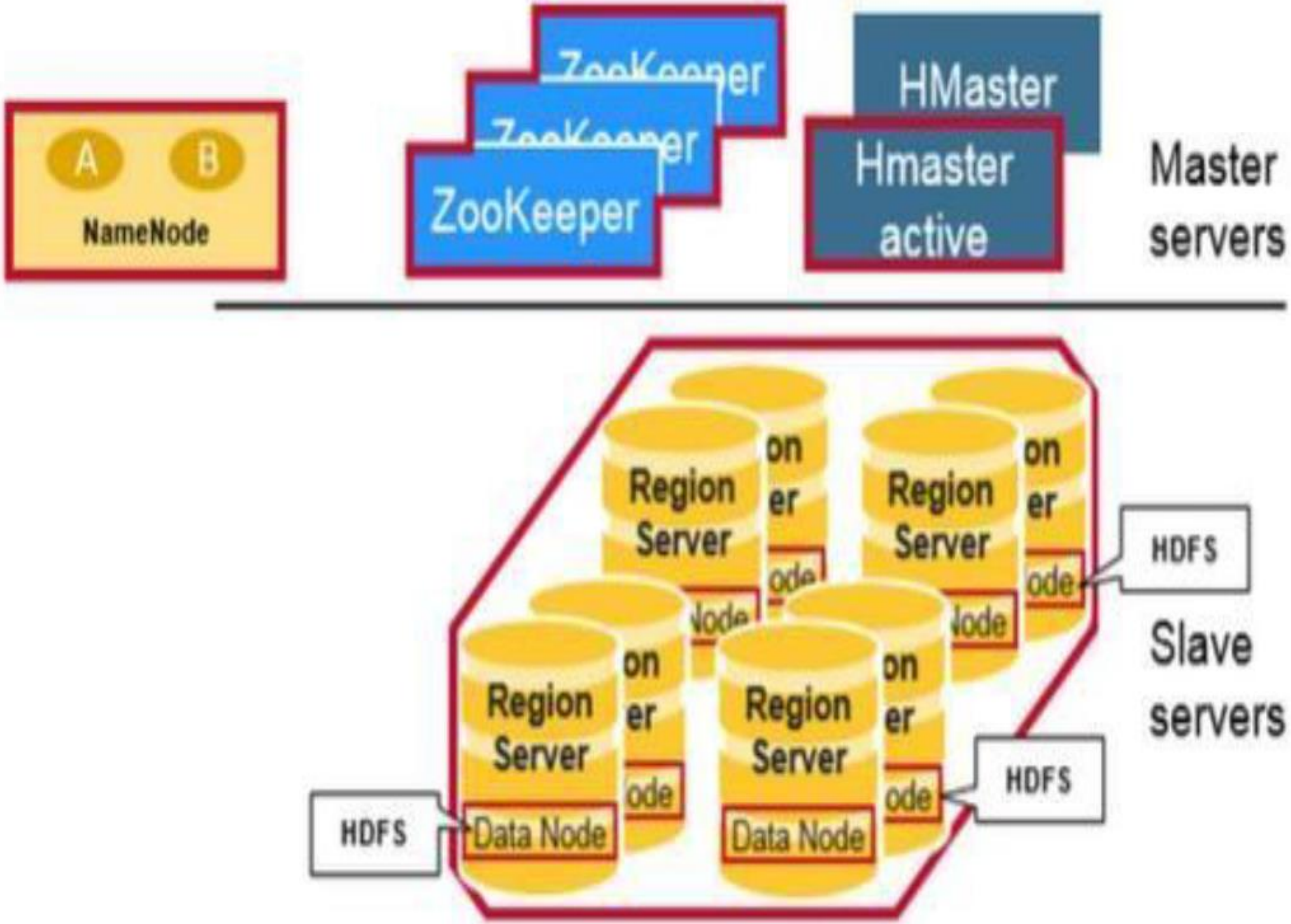
# Hbase Architecture

HBase is composed of three types of servers in a master slave type of architecture.

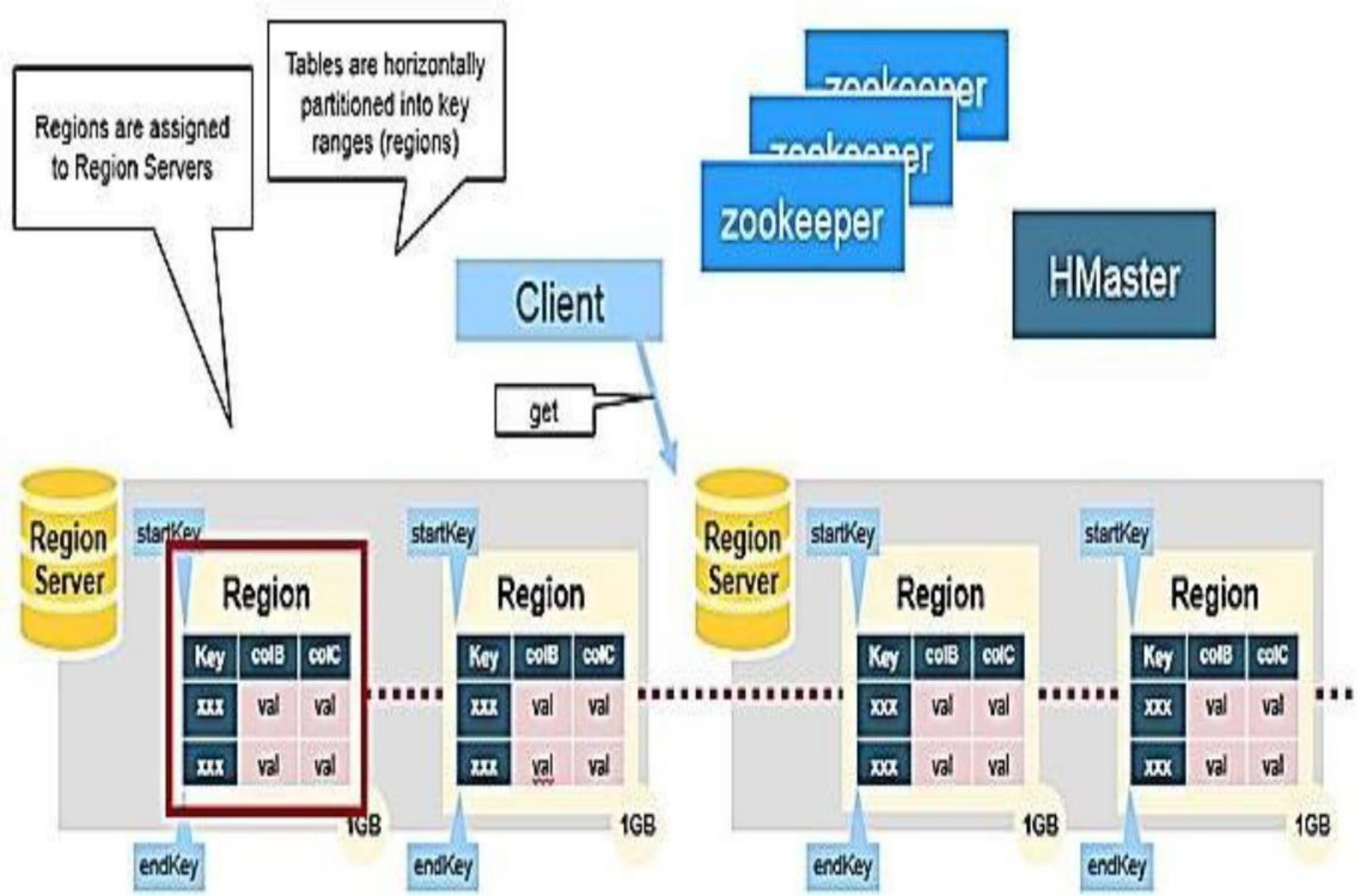
- Region servers serve data for reads and writes
- HBase Master process handles the Region assignment, DDL (create, delete tables) operations
- Zookeeper maintains a live cluster state

The Hadoop DataNode stores the data that the Region Server is managing

- All HBase data is stored in HDFS files
- The NameNode maintains metadata information for all the physical data blocks that comprise the files


A
B
NameNode
ZooKeeper
ZooKeeper
ZooKeeper
HMaster
Hmaster active
Master servers
Region Server
Region Server
Region Server
Region Server
Data Node
Data Node
HDFS
HDFS
HDFS
Slave servers

# REGIONS


Regions are assigned to Region Servers
Tables are horizontally partitioned into key ranges (regions)
zookeeper
zookeeper
zookeeper
HMaster
Client
get
Region Server
startKey
Region
Key
colB
colC
xxx
val
val
xxx
val
val
endKey
1GB
startKey
Region
endKey
1GB
Region Server
startKey
Region
endKey
1GB
startKey
Region
endKey
1GB

# HMASTER

Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.

A master is responsible for:

- Coordinating the region servers
- Assigning regions on startup
- Re-assigning regions for recovery or load balancing
- Monitoring all RegionServer instances in the cluster

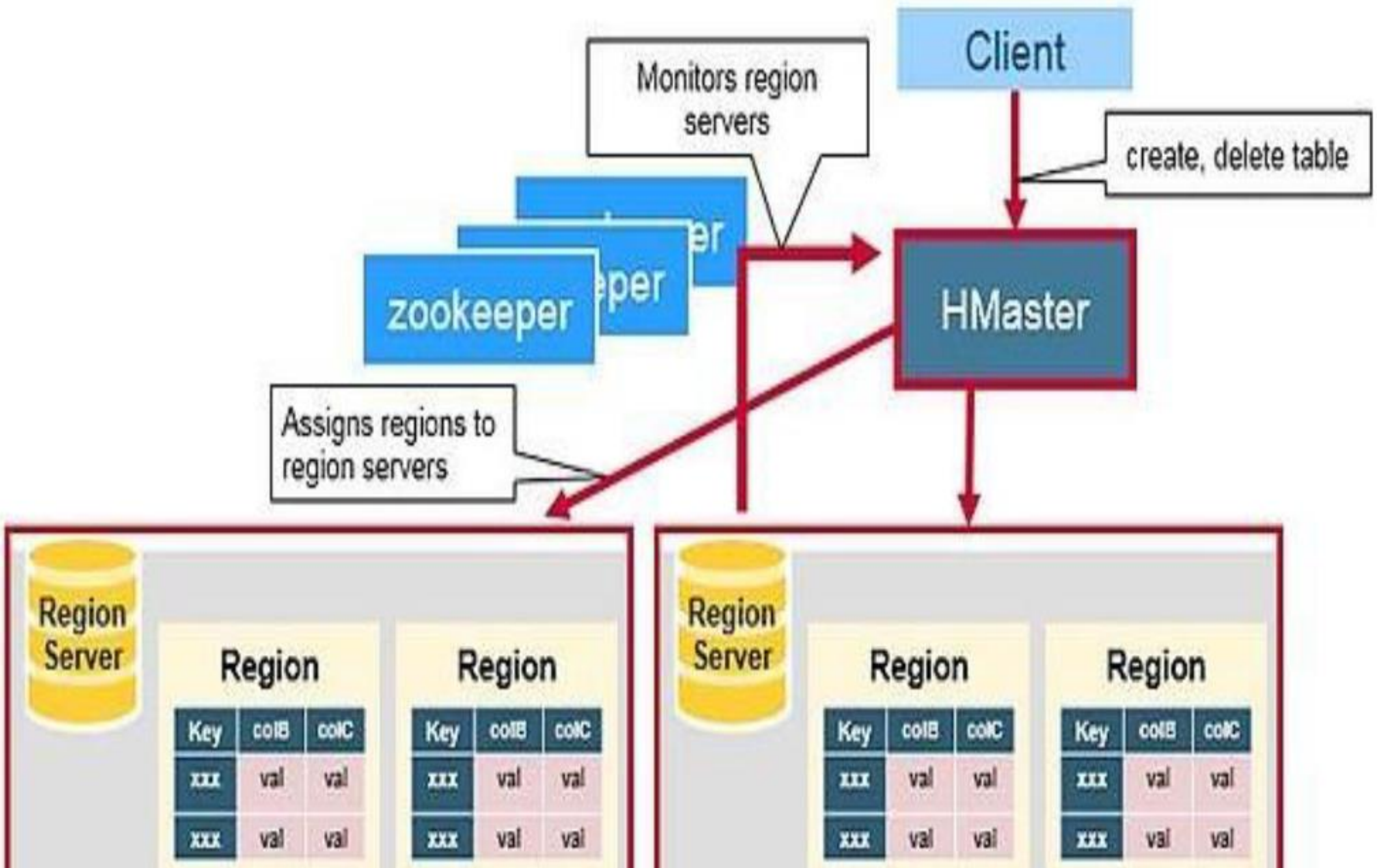
(listens for notifications from

zookeeper)

**Admin functions**

- Interface for creating, deleting, updating tables

# HMASTER



Client
create, delete table
Monitors region servers
zookeeper
eper
er
HMaster
Assigns regions to region servers
Region Server
Region
Key
colB
colC
xxx
val
val
xxx
val
val
Region
Key
colB
colC
xxx
val
val
xxx
val
val
Region Server
Region
Key
colB
colC
xxx
val
val
xxx
val
val
Region
Key
colB
colC
xxx
val
val
xxx
val
val

# HBASE FIRST READ

There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster

- ZooKeeper stores the location of the META table

- The client gets the Region server that hosts the META table from ZooKeeper

- The client will query the .META. server to get the region server corresponding to the row key it wants to access. The client

caches this information along with the META table location

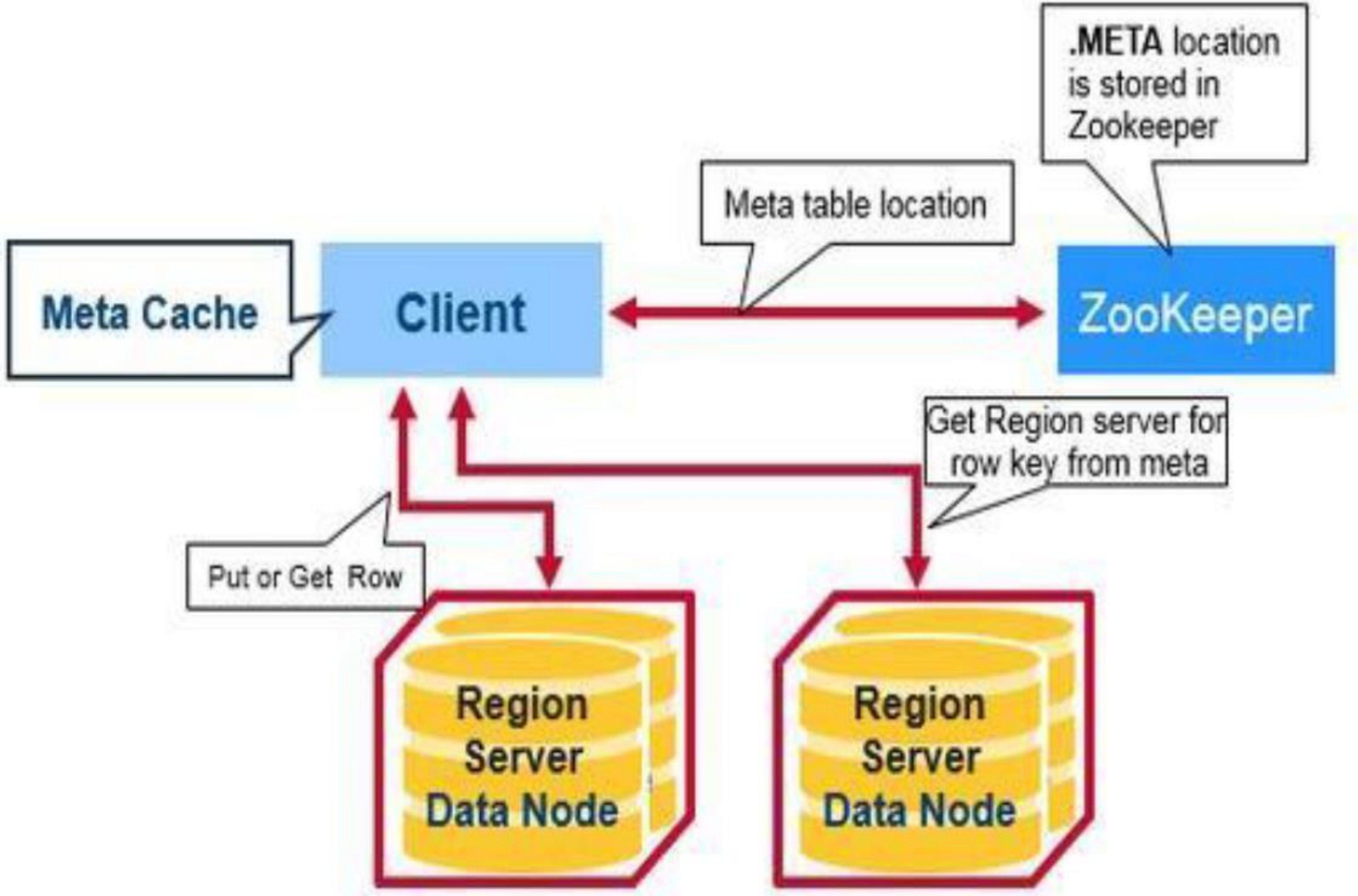
- It will get the row from the corresponding Region Server

- For future reads, the client uses the cache to retrieve the META location and previously read row keys

- Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will

re-query and update the cache

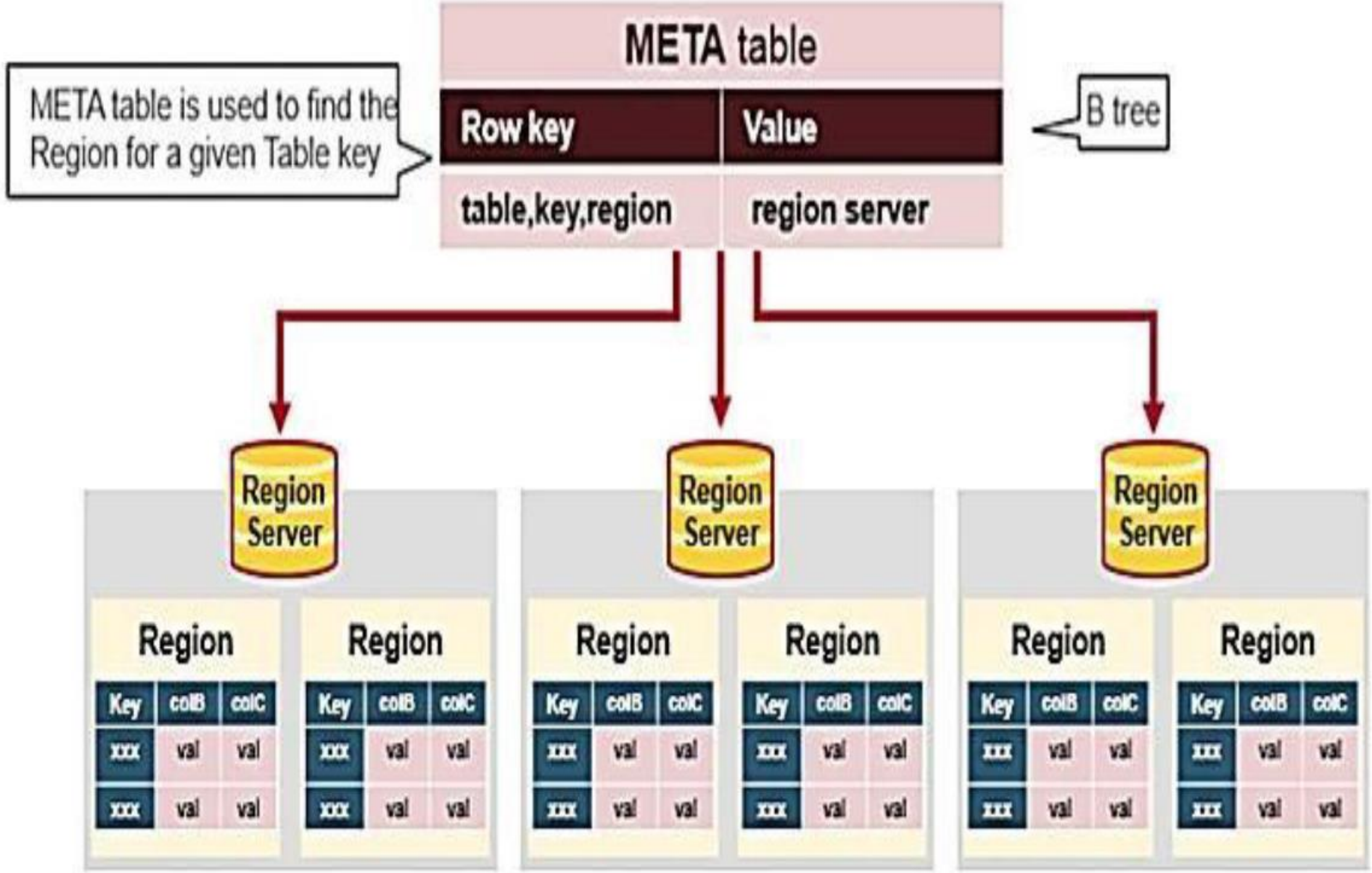
# HBASE FIRST READ



# HBASE META TABLE

This META table is an HBase table that keeps a list of all regions in the system

- The .META. table is like a b tree
- The .META. table structure is as follows: Key: region start key, region id Values: RegionServer



# REGION SERVER COMPONENTS

Region Server runs on an HDFS data node and has the following components:

**WAL**

- Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.

**BlockCache**

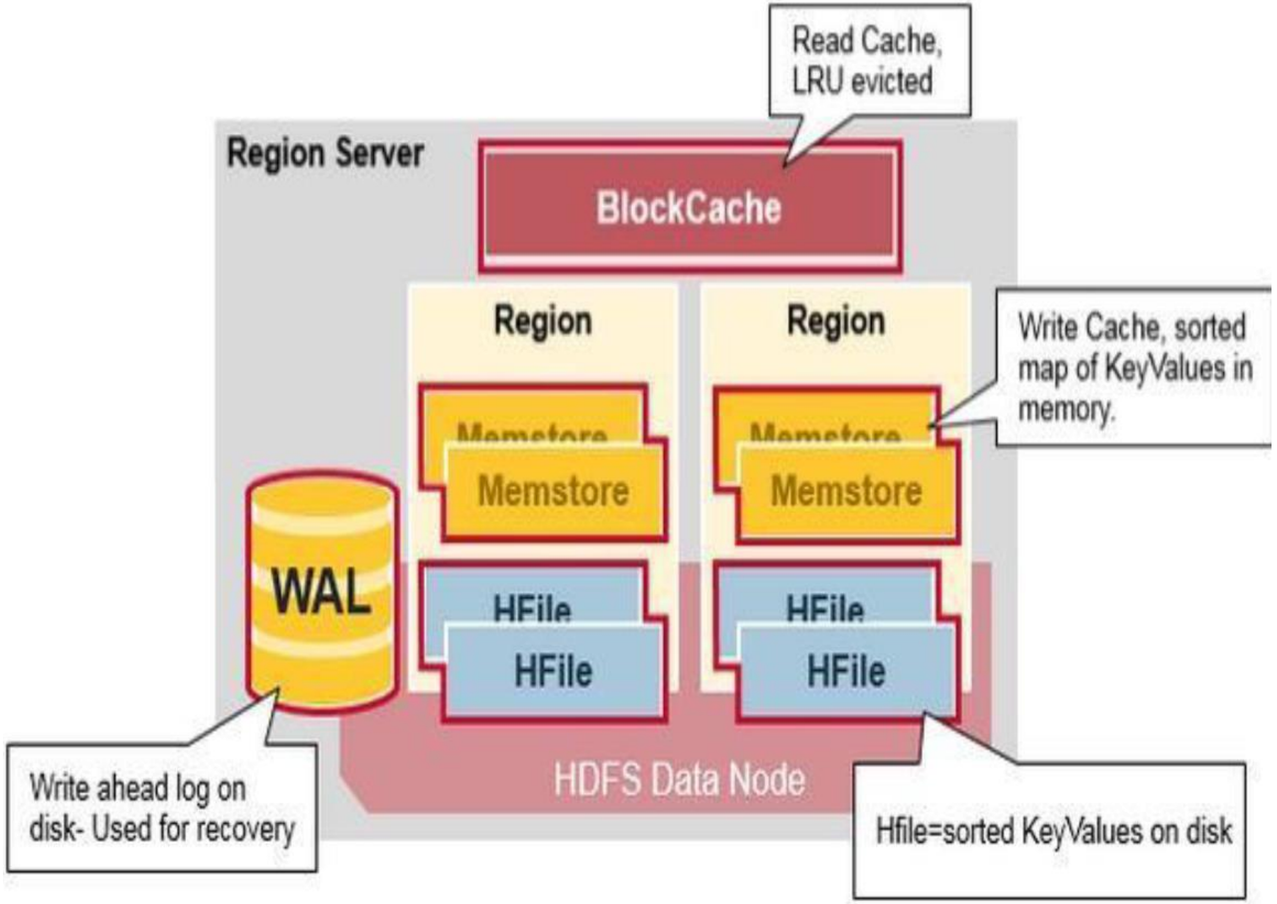
- It is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.

**MemStore**

- It is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.

**Hfiles**

- They store the rows as sorted KeyValues on disk.



# REGION SPLIT

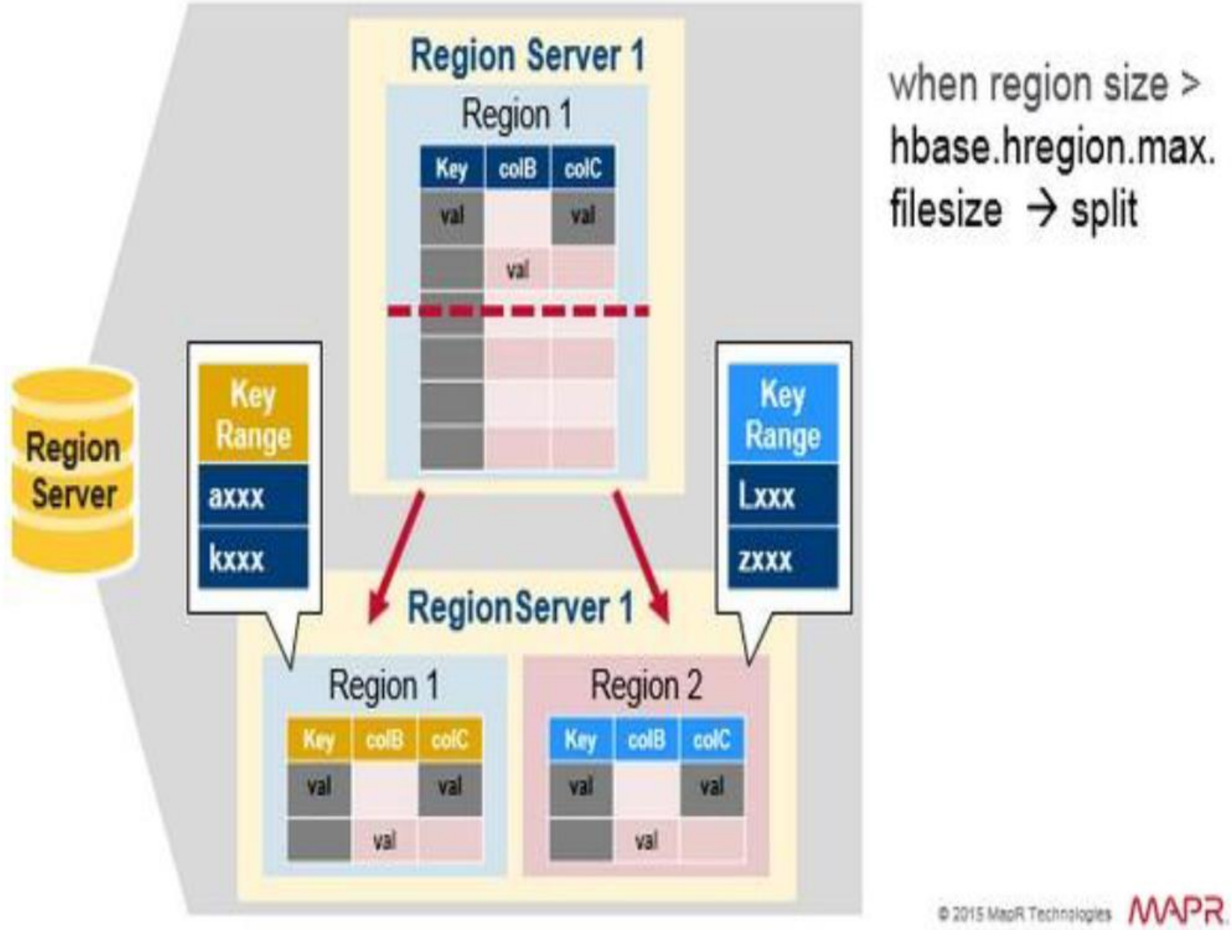
Initially there is one region per table.

• When a region grows too large, it splits into two child regions.

• Both child regions, representing one-half of the original region, are opened in parallel on the same Region server, and then

the split is reported to the HMaster.

• For load balancing reasons, the HMaster may schedule for new regions to be moved off to other servers.



# APACHE HBASE ARCHITECTURE BENEFITS

HBase provides the following benefits:

- Strong consistency model- When a write returns, all readers will see same value
- Scales automatically- Regions split when data grows too large- Uses HDFS to spread and replicate data
- Built-in recovery- Using Write Ahead Log (similar to journaling on file system)
- Integrated with Hadoop- MapReduce on HBase is straightforward

| RDBMS | HBASE |
|---|---|
| RDBMS is row-oriented databases | HBase is a distributed, column-oriented data storage system |
| RDBMS tables have fixed-schema | Hbase tables do not have fixed- schema |
| RDBMS tables guarantee ACID properties | Hbase tables guarantee consistency and partition tolerance |
| RDBMS uses SQL (Structured query Langauge ) to query the data | Hbase uses Java client API and Jruby |
| | |