EECS 4314 - Assignment 1

•••

Daniel McVicar - 213027479
Rafay Sheikh - 213033451
Daniel Fortunato - 216796443
Adham El Shafie - 212951018
Yahya Ismail- 213235403

Hbase - What is it?

A non-relational, distributed database.

- Non-relational: No enforced schema on data. Key values are used to store any type of data needed.

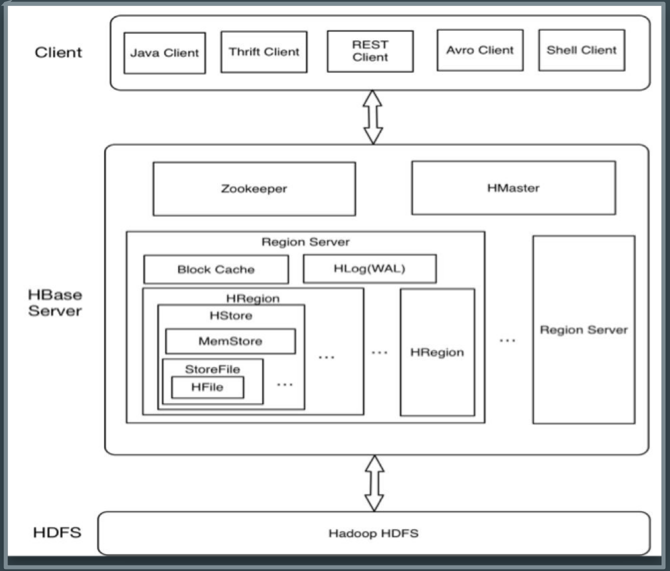
- Distributed: Information is spread across multiple servers.

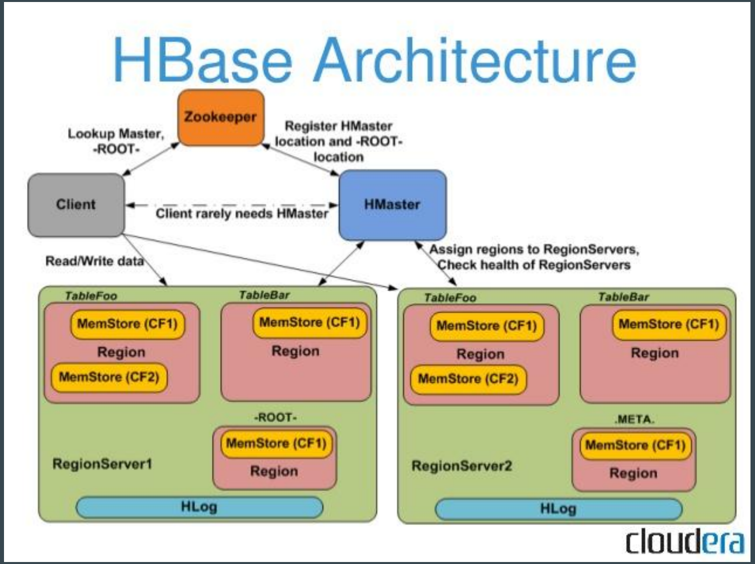
Architectural Style - Master/Slave

- Commands are passed down, information is passed up.

- Higher level layers responsible for maintaining lower levels.

System Evolution





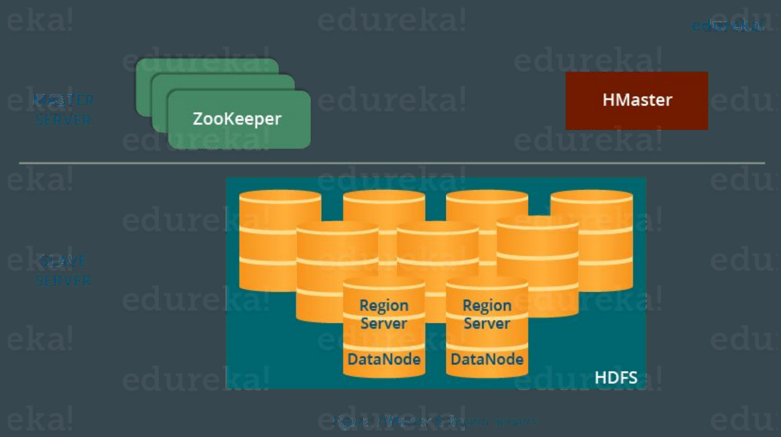
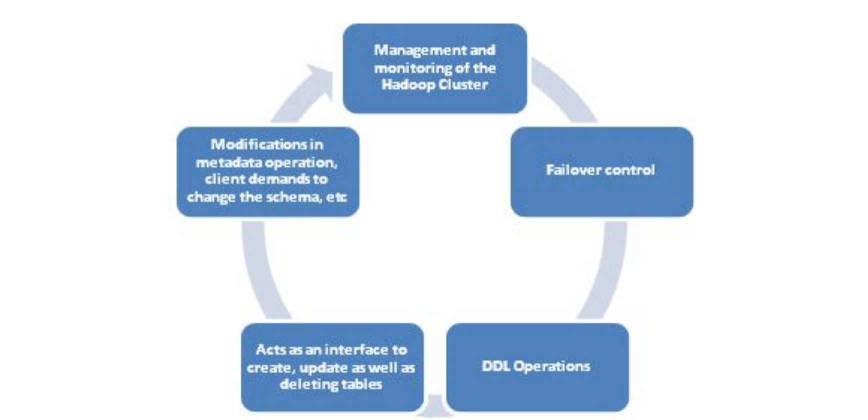


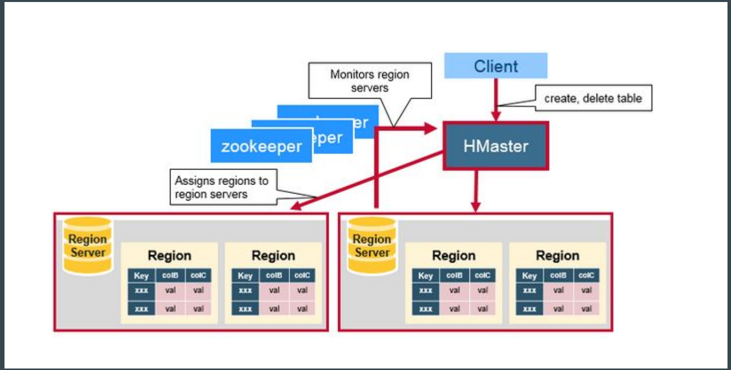
Figure: HMaster & Region Servers

Top Level Subsystems - HMaster

- Master Node
  - Light weight process
- Preforms Administration actions
  - DDL operations
  - Create, delete, update tables
  - Changes to schema
- Coordinating Region Servers
  - Assigning regions to RegionServers
  - Reassigning to other RegionServer
  - Initializing, monitoring and maintaining the regions

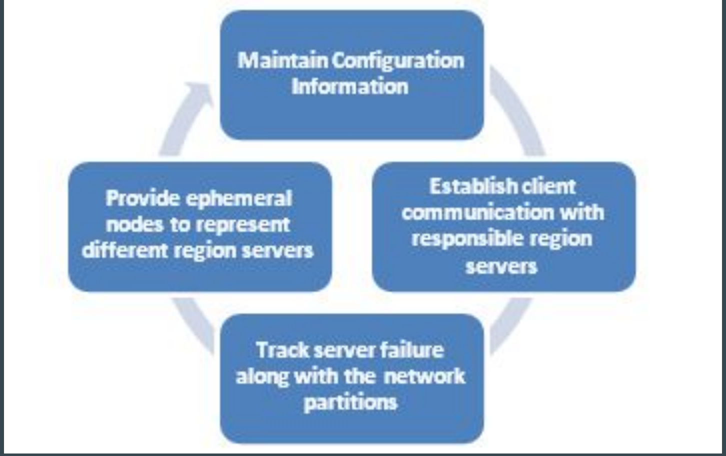


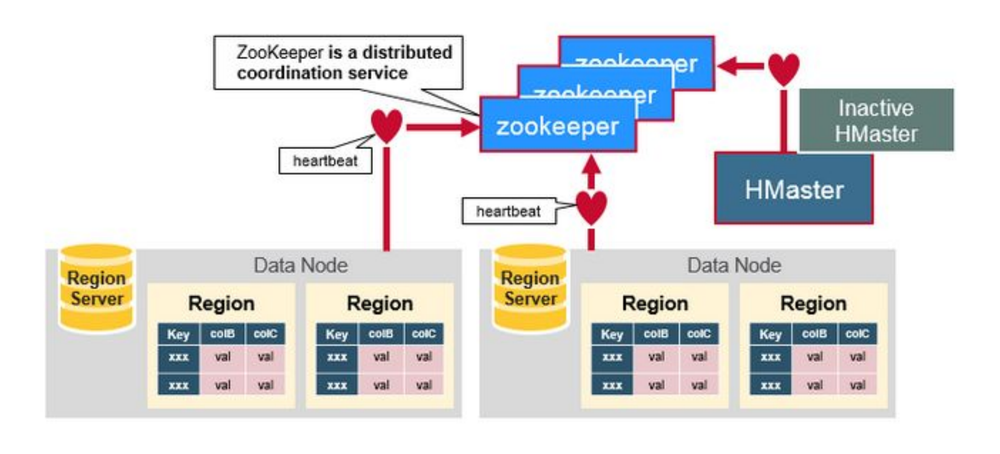
Management and monitoring of the Hadoop Cluster
Failover control
DDL Operations
Acts as an interface to create, update as well as deleting tables
Modifications in metadata operation, client demands to change the schema, etc



Top Level Subsystems - Zookeeper

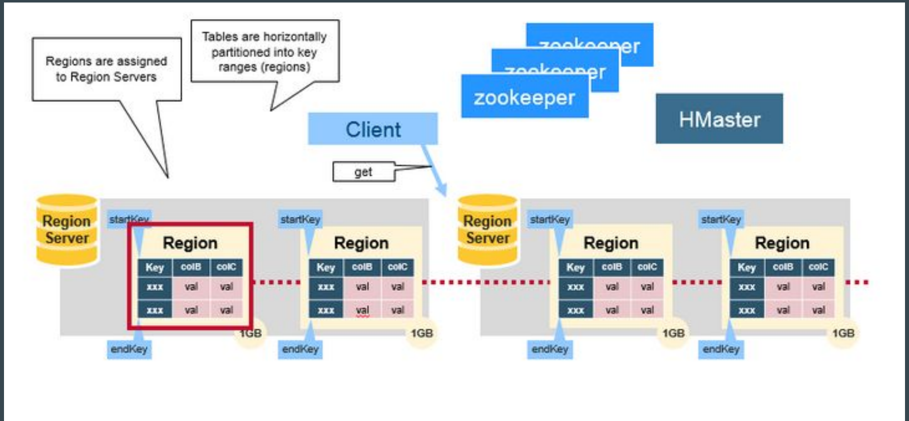
- Distributed coordination service
- Centralized monitoring server
- Use of Consensus
  - For common shared state
- Keeps track of RegionServers in HBase Cluster
- Heartbeat


Maintain Configuration Information
Establish client communication with responsible region servers
Track server failure along with the network partitions
Provide ephemeral nodes to represent different region servers



Top Level Subsystems - RegionServers

- Worker nodes (slaves)(znodes)
  - To access data, client communicated directly with RS (read and write)
- Components Of RegionServers
  - For read/write operations
- Perform load balancing
- Perform request received from client
- Can have upto 1000 regions
- Regions:
  - Tables divided by row key ranges
  - Assigned to nodes in cluster (RS)



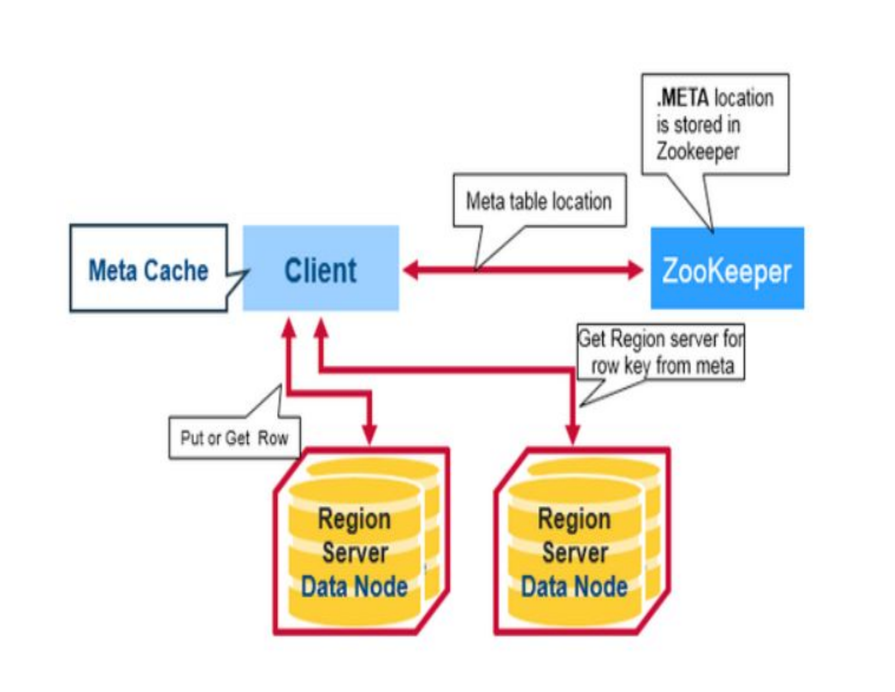
Regions are assigned to Region Servers

Tables are horizontally partitioned into key ranges (regions)

Control and Data flow

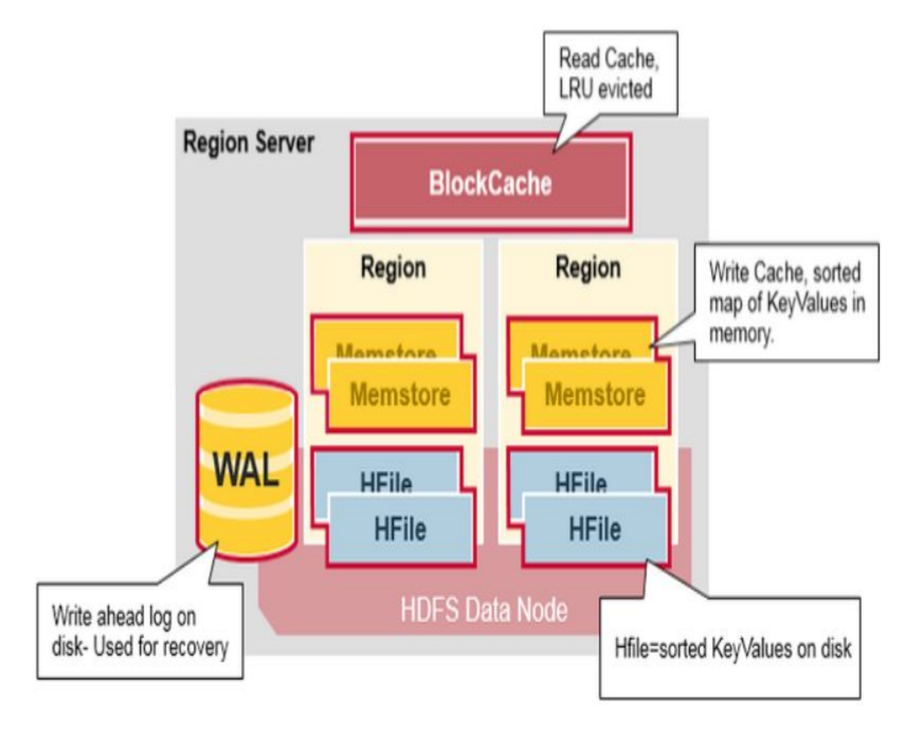
HBase first Read or Write

1. The client gets the Region server that hosts the META table from ZooKeeper.
2. 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.
3. It will get the Row from the corresponding Region Server.

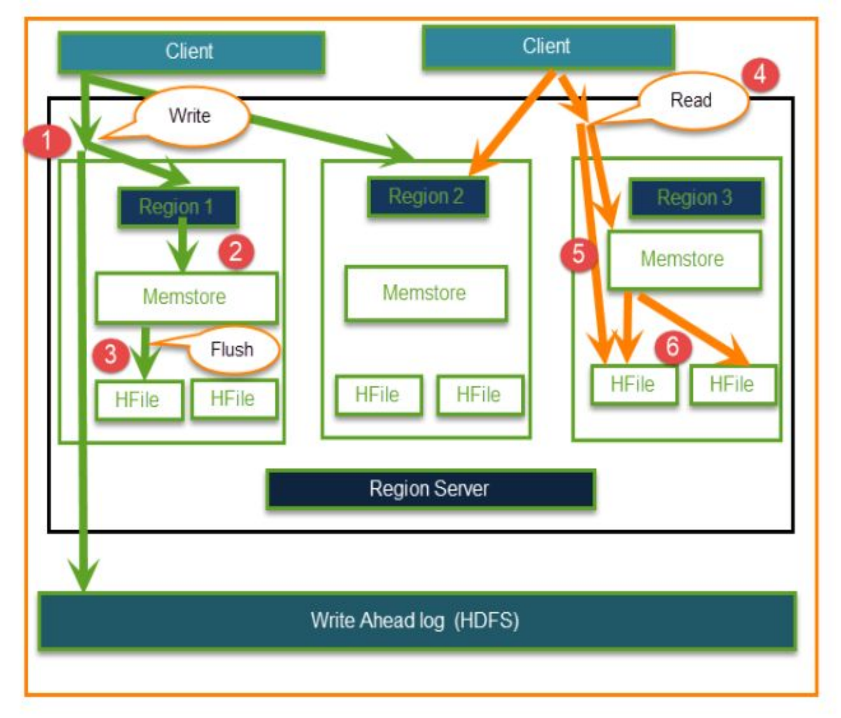


Region Server Components

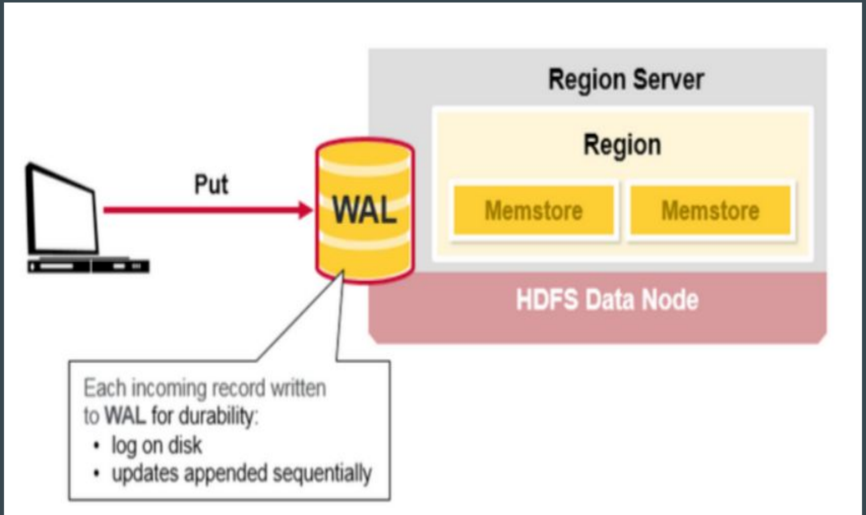
- WAL
- BlockCache
- MemStore
- HFile



The Data flow Process - Operation Put



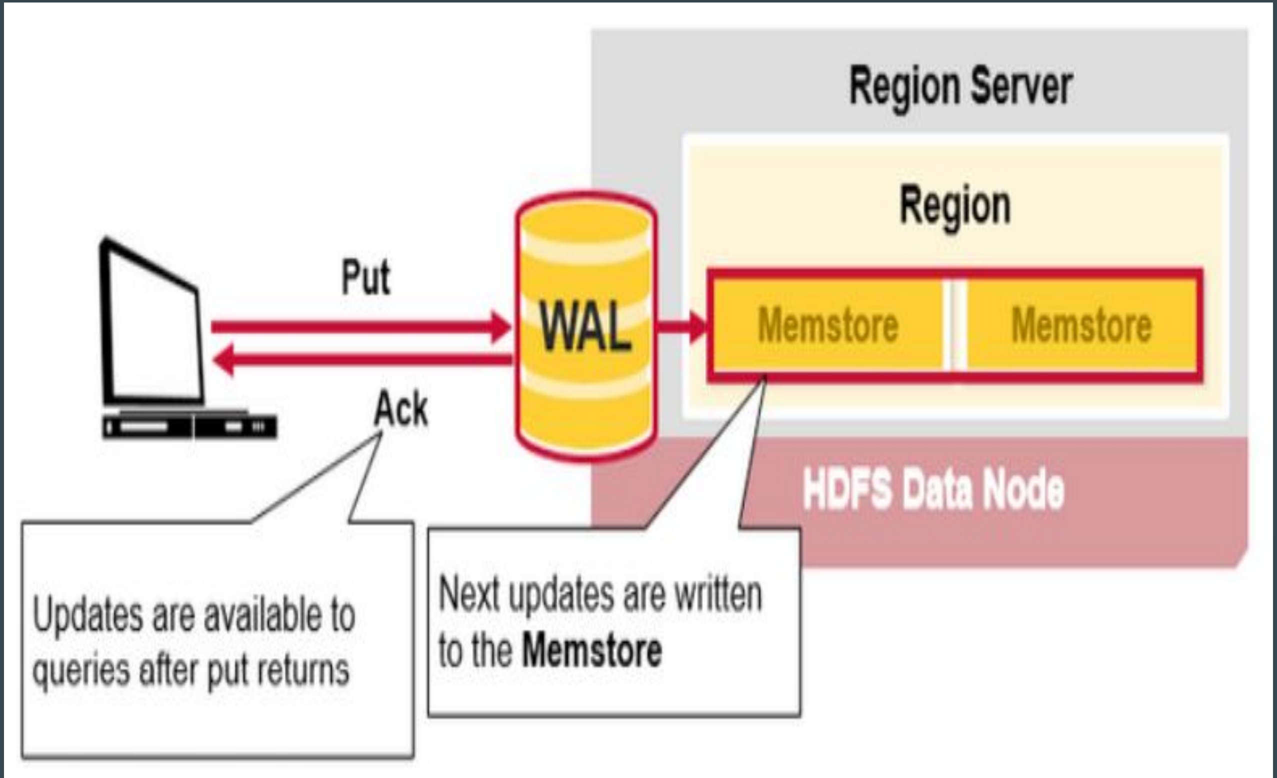
Step 1 - WAL Write



Each incoming record written to WAL for durability:

- log on disk
- updates appended sequentially

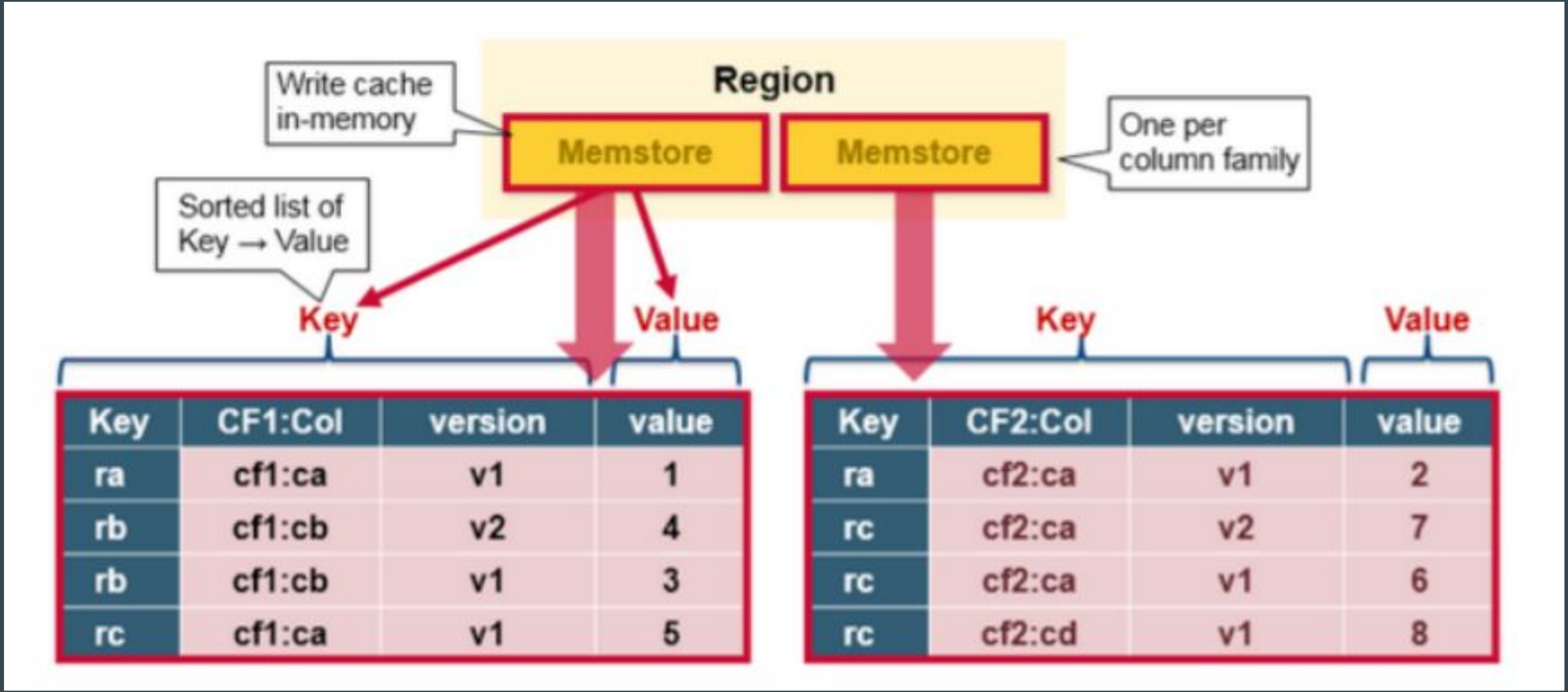
Step 2 - MemStore Write



Updates are available to queries after put returns

Next updates are written to the **Memstore**

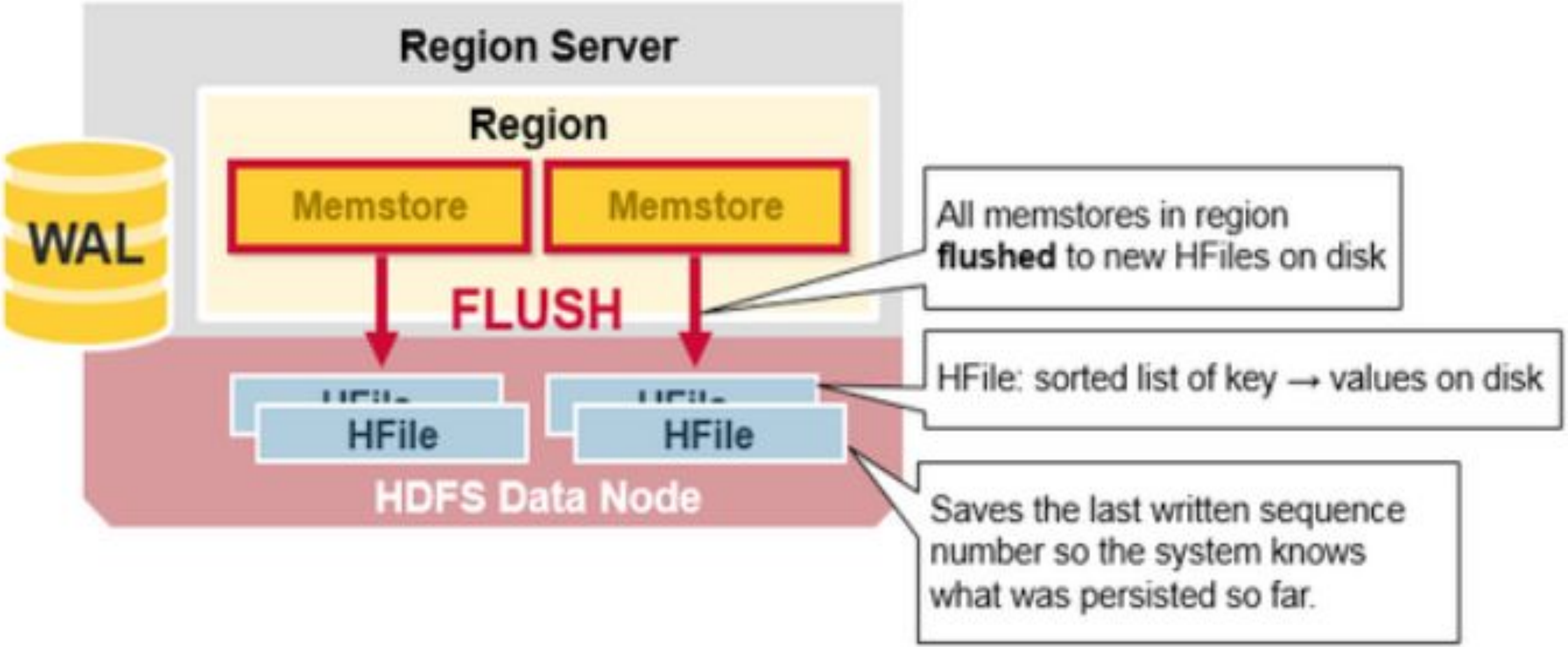
Step 2 - MemStore Write



Key	CF1:Col	version	value
ra	cf1:ca	v1	1
rb	cf1:cb	v2	4
rb	cf1:cb	v1	3
rc	cf1:ca	v1	5

Key	CF2:Col	version	value
ra	cf2:ca	v1	2
rc	cf2:ca	v2	7
rc	cf2:ca	v1	6
rc	cf2:cd	v1	8

Step 3 - Flush

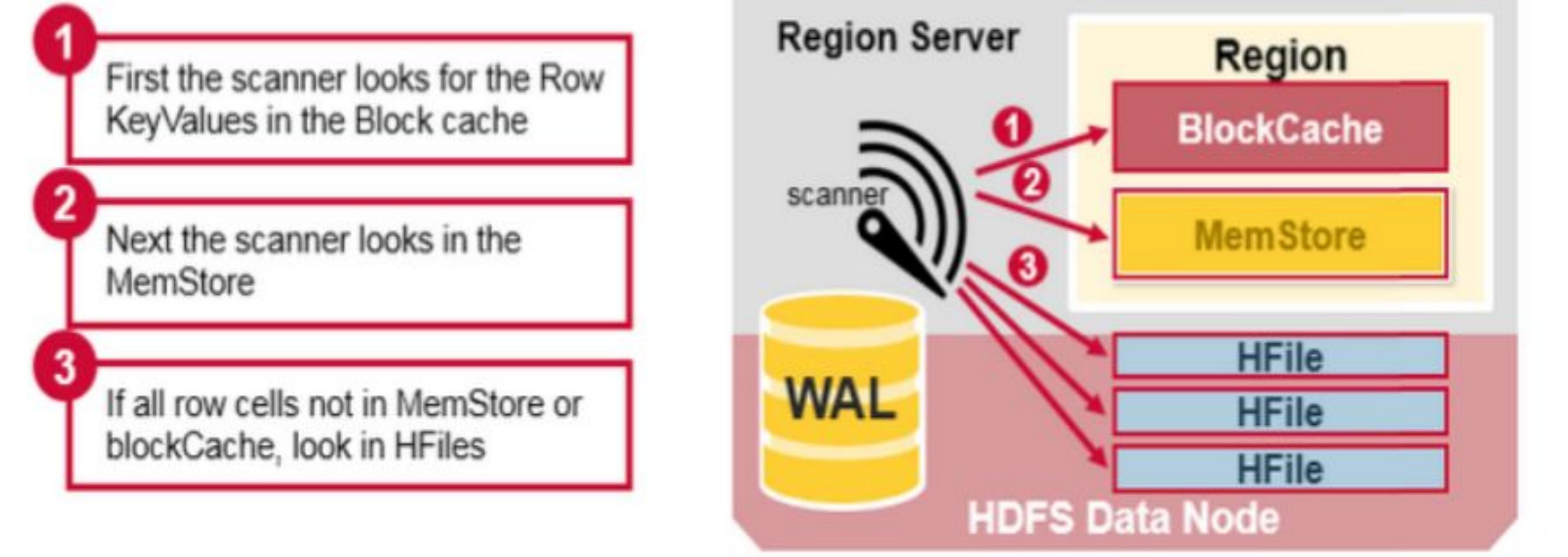


All memstores in region **flushed** to new HFiles on disk

HFile: sorted list of key → values on disk

Saves the last written sequence number so the system knows what was persisted so far.

Operation Get



1 First the scanner looks for the Row KeyValues in the Block cache

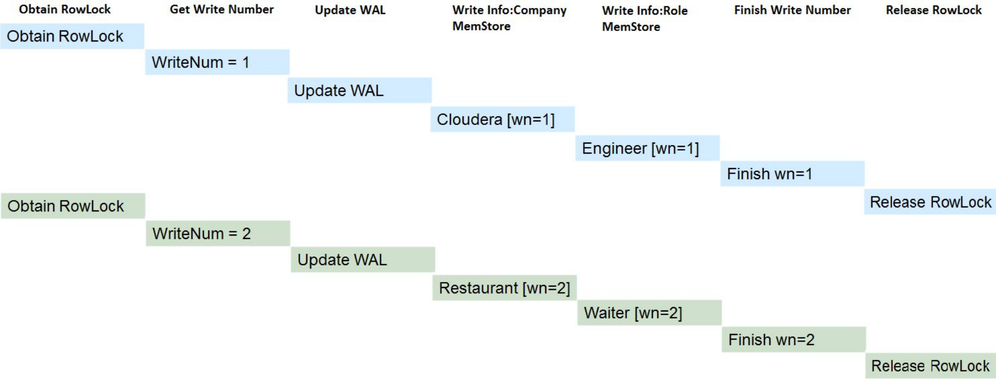
2 Next the scanner looks in the MemStore

3 If all row cells not in MemStore or blockCache, look in HFiles

Concurrency and Practical Use Cases

Concurrency

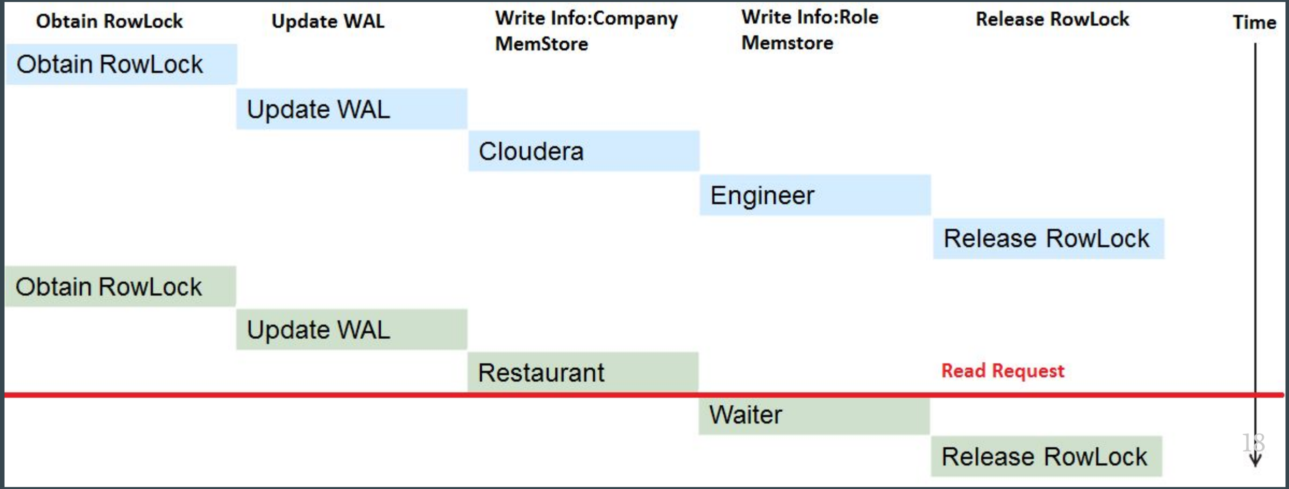
- Not ACID compliant, but guarantees ACID semantics per row
- Uses Multiversion Concurrency Control
  - Read operations are only performed on the last finished write operation
    - Previous versions are saved
  - Read operation is not blocked by exclusive write locks, enhancing performance



Obtain RowLock	Get Write Number	Update WAL	Write Info:Company MemStore	Write Info:Role MemStore	Finish Write Number	Release RowLock
Obtain RowLock						
	WriteNum = 1					
		Update WAL				
			Cloudera [wn=1]			
				Engineer [wn=1]		
					Finish wn=1	
						Release RowLock
Obtain RowLock						
	WriteNum = 2					
		Update WAL				
			Restaurant [wn=2]			
				Waiter [wn=2]		
					Finish wn=2	
						Release RowLock

Concurrency Example (R/W Use Case)

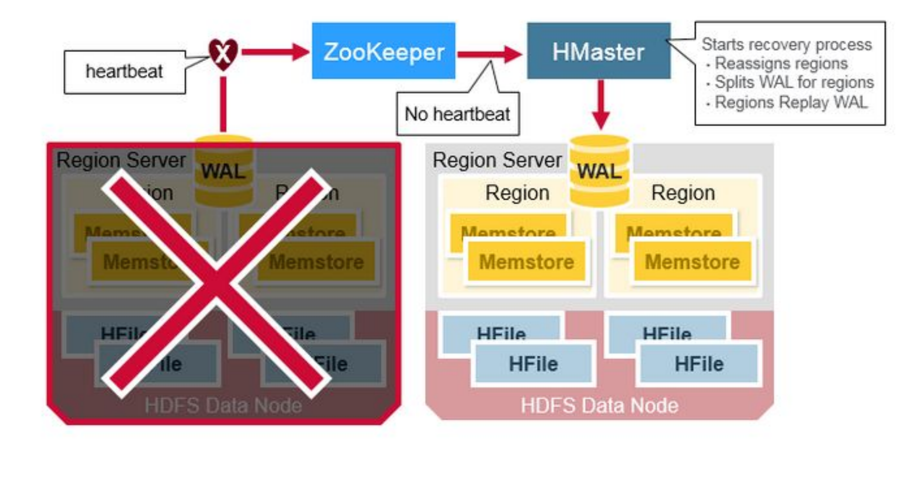
1. A write operation is requested, the row is locked
   a. A Write Number is assigned to the write operation
2. The Write-Ahead Log is updated (Used for disaster recovery)
3. Update MemStore
   a. Finish Write Number
4. Release Row Lock



Obtain RowLock	Update WAL	Write Info:Company MemStore	Write Info:Role Memstore	Release RowLock	Time
Obtain RowLock					
	Update WAL				
		Cloudera			
			Engineer		
				Release RowLock	
Obtain RowLock					
	Update WAL				
		Restaurant		Read Request	
			Waiter		
				Release RowLock	

404 Region Not Found (Region Crash Use Case)

- ZooKeeper continually monitors region server heartbeats and notifies HMaster.
- When a region fails, the HMaster node undertakes the following tasks:
  1. Regions from the crashed server are assigned to an active server
  2. WAL split and copied to new server for data-recovery
  3. The new server replays the WAL to recover data edits



Competition

Cassandra

Similarities

NoSQL

Wide-column

Descend from BigTable

Security

Differences

Masterless

Internal data replication – data inconsistency

Always up/no downtime – hbase master is main contact

CQL, to run queries – Hbase needs extra help

Apache Kudu

Similarities

Update/write/scan/inserts speed

Wide-column storage

IO/CPU efficiency

Differences

Faster analytics

HBase has column families

Lower max size per row

Amazon Redshift

Similarities

Sharding partition

Concurrency

Immediate consistency – all or nothing

Differences

More for BI tools

Relational DBMS

Replication: Always for Red – optional for hbase

Redis

Similarities

Sharding

Master-slave replication

Concurrency

Differences

Strong eventual consistency

In-memory data structure store as DBMS

Does not have the MapReduce functionality