Hbase Architecture

...

Hbase Table

An HBase Table is a collection of rows and columns that are sorted based on the row key. This table is distributed according to a fixed size, and each portion of the table is called a region.

REGIONS

		Column Family - 01		Column Family - 02		
	RowKey	cf1:colA	cf1:colB	cf2:colA	cf1:colB	cf1:colC
Region - 01	row-1	2	4	7	8	9
	row-10	5	6	1	3	5
	row-18	3	7	3	4	6
		5	3	7	3	7
Region - 02	row-2	3	4	1	4	7
	row-5	7	6	1	3	5
	row-6	6	3	3	4	6
Region - 03	row-7	4	5	2	6	7
	row-8	5	8	7	6	0
	row-9	0	4	2	8	9

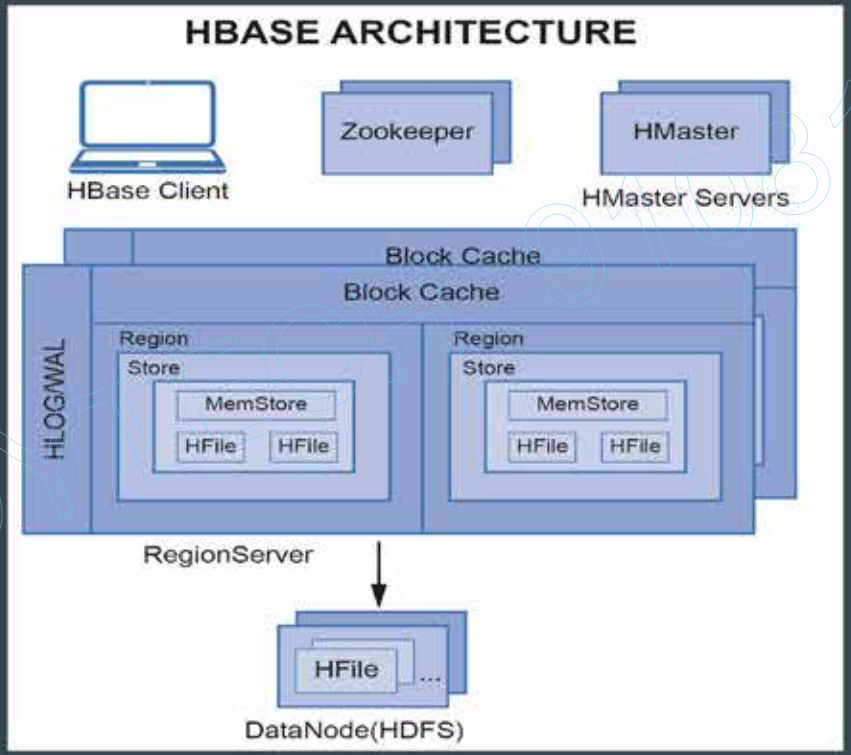
These regions are stored in region servers. A region server can handle many regions.

In each region there is a memstore per column family. A memstore is an in-memory write buffer. It stores new or updated data that has not yet been written to HDFS.

This data gets flushed to HDFS in a new HFile consisting of sorted key-value pairs at regular intervals, or based on the MemStore size. These HFiles are written to the DataNodes of HDFS.

HBase architecture is also based on Master-Slave architecture, where HMaster is the master and region servers act as the slaves.

Various Components



Region Server

1. A region server manages regions. There are multiple region servers in a cluster.

2. Each region server contains the following:

WAL, which is a write-ahead log, also known as HLog. WAL stores new or updated data that has not been written to a permanent storage (as HFiles in HDFS) and can be used for recovery in case of region server failure. The region server stores the WAL file in HDFS.

Block cache, which is a read cache, stores frequently read data from HDFS.

Region Server

Region, which is assigned a region of the table. Each region contains a memstore per column family.

3. A client interacts directly with the region server to perform read or write operations.

HMaster

1. It acts as the master server and manages multiple region servers.

2. An HBase cluster may have one or more master nodes, but only one HMaster is active at a given time. This active node is responsible for:

• Performing the admin functionalities: HMaster performs DDL operations, such as creating or deleting a table.

• Coordinating with the region servers: HMaster assigns regions to various region servers on start-up and reassigns regions in case of load balancing or recovery.

HMaster

- Load balancing: Whenever there is a high load on any region server, HMaster unloads the busy server and reassigns this load to less occupied servers.

- Recovery: HMaster also handles region server failures by reassigning the regions or load of the failed region server to another non-failing region server.

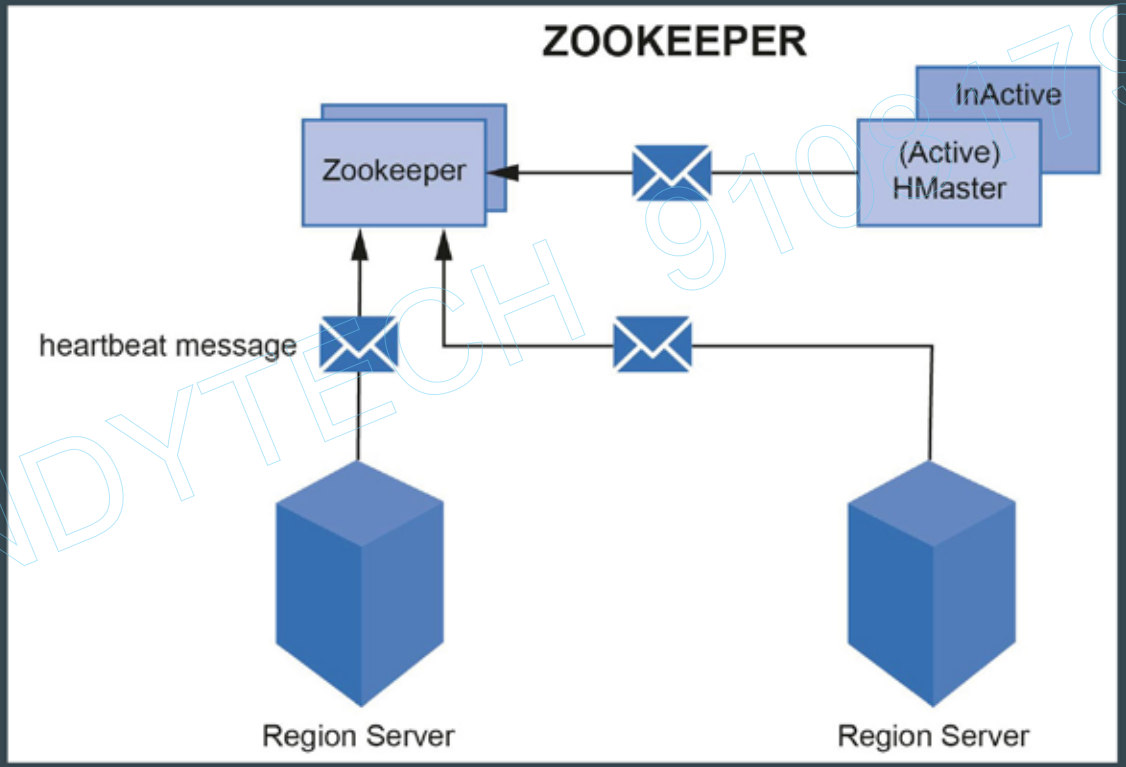
Zookeeper

1. It is a distributed open-source coordinating service for distributed applications.

2. HBase uses Zookeeper to maintain the live server state in the cluster by receiving heartbeat messages from all HMasters (active/inactive) and region servers.

3. It stores the location of the META table.

Zookeeper



DataNode(HDFS)
HBase stores two types of files in the DataNode of HDFS:

1. Write-ahead log
- There is one WAL file per region server that stores the MemStore data.
- WAL is replayed by region servers in case of a region server crash.

2. HFiles

- It stores the HBase table data as sorted key-value pairs.
- HFiles are immutable, which means once written, they can't be modified.
- HFiles are large in size and depend upon the MemStore flush size before compaction.
- HFiles further store data as sets of blocks. This helps in only reading the block, which contains the data of interest, and not the complete HFile.

The default block size is 64KB.

The Data Block Index in an HFile is used to locate the data block of interest. It contains the key range stored in each data block.

META Table

- A META table is a data structure that stores the location of the regions along with the region servers.

- It helps the user identify the region server and its corresponding regions, where the specific range of key-values is stored.

- The META table is stored in one of the region servers and its location is stored in the Zookeeper.

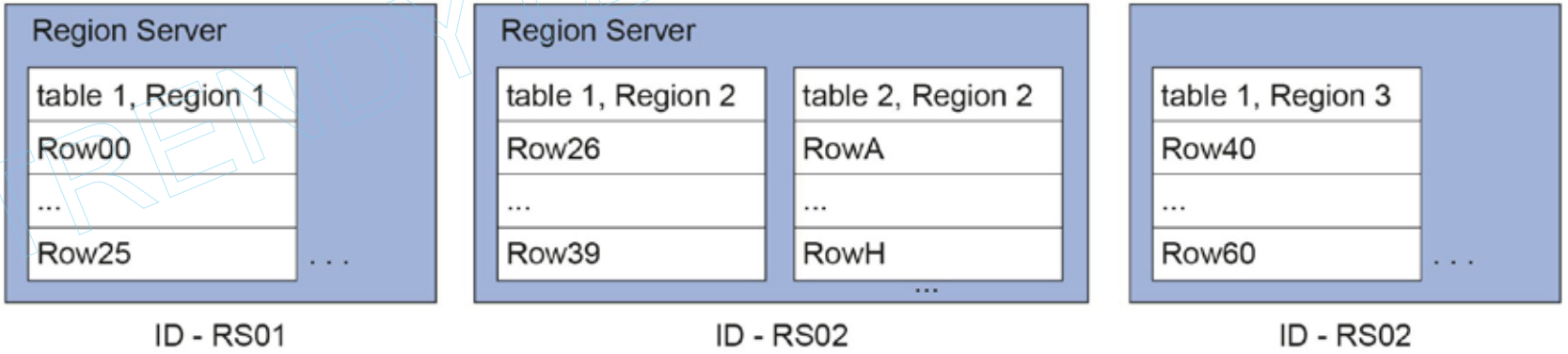
META TABLE

key | value

Table Name	Start Key	Region ID	Region Server ID
table 1	Row 00	1	RS01
table 1	Row 26	2	RS02
table 1	Row 40	3	RS03
...			
table 2	Row A	21	RS02
...			

Key: <Table Name, Start row Key, regionID >

Value: <Region Server ID>



HBase Read/Write Operations

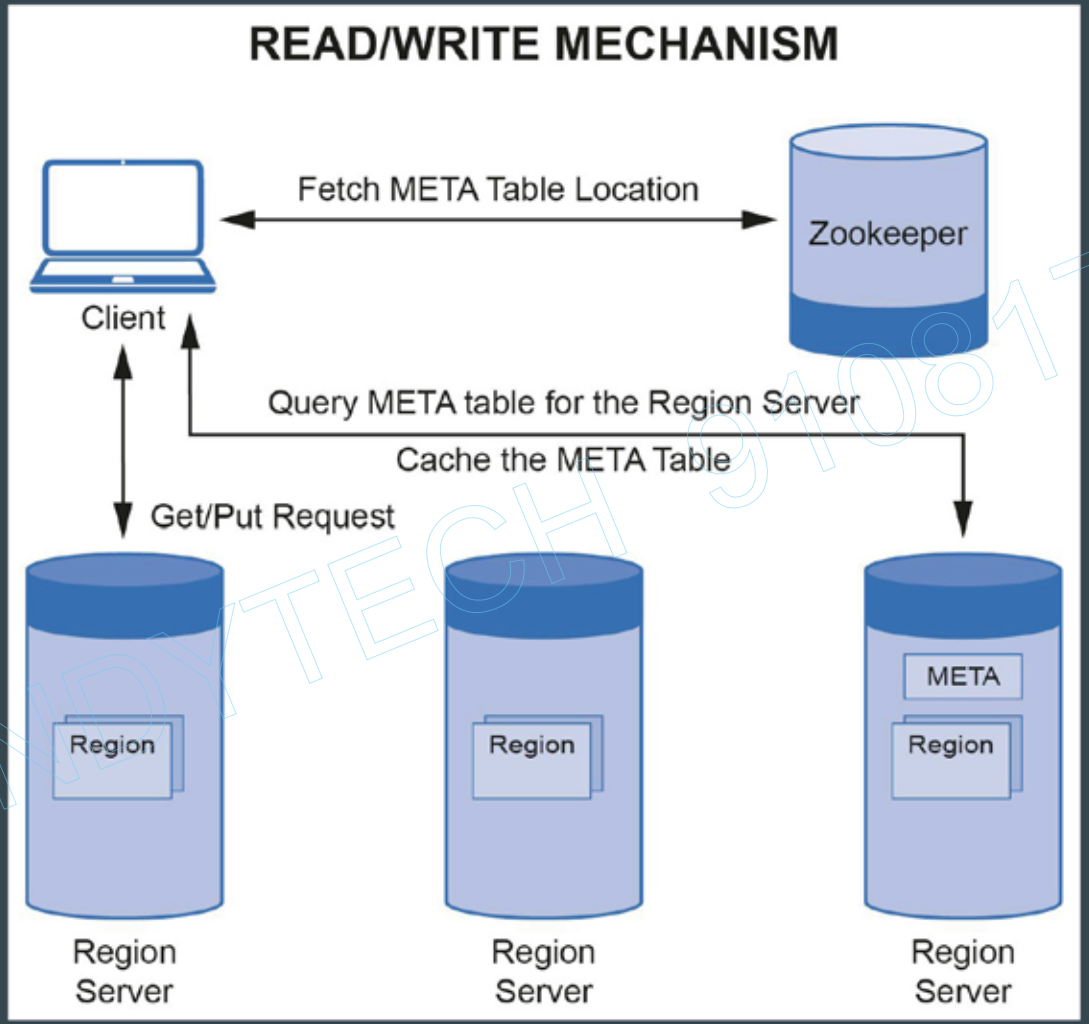
The common steps involved in HBase read/write operations are as follows:

Step 1: The client contacts the Zookeeper to fetch the location of the META table(if the client doesn't have the latest version of the META table already cached.

Step 2: The client queries the META table to find the location of the region server, which has the region containing the row-key that the client is looking for.

Step 3: The client caches the identified region server information as well as the META table location, for future interactions.

Step 4: The client can now communicate with the specific region server, identified in the Step 2 above. This region server assigns the request to the specific region, where the read/write operations can be executed.

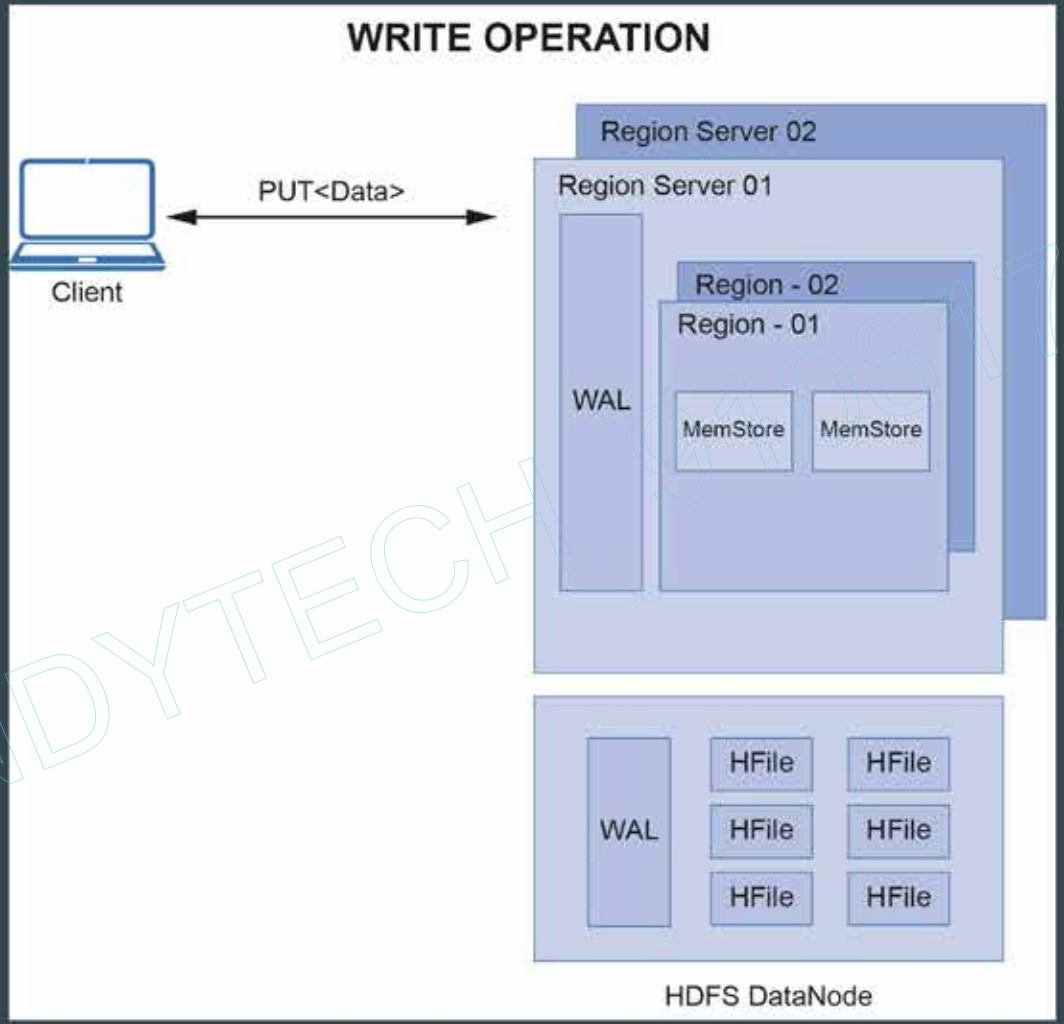


HBase Write Operation

Step 1: The data first needs to be written to the WAL, which is a write-ahead Log. WAL stores the new or updated data that has not been written to the HDFS and can be used for recovery, in case of a region server failure.

Step 2: Once the data is written to the WAL, it is placed in the MemStore of a region. When the MemStore becomes full, its contents are flushed to HDFS (DataNode) to form a new HFile.

Step 3: Finally, an acknowledgement is sent back to the client.



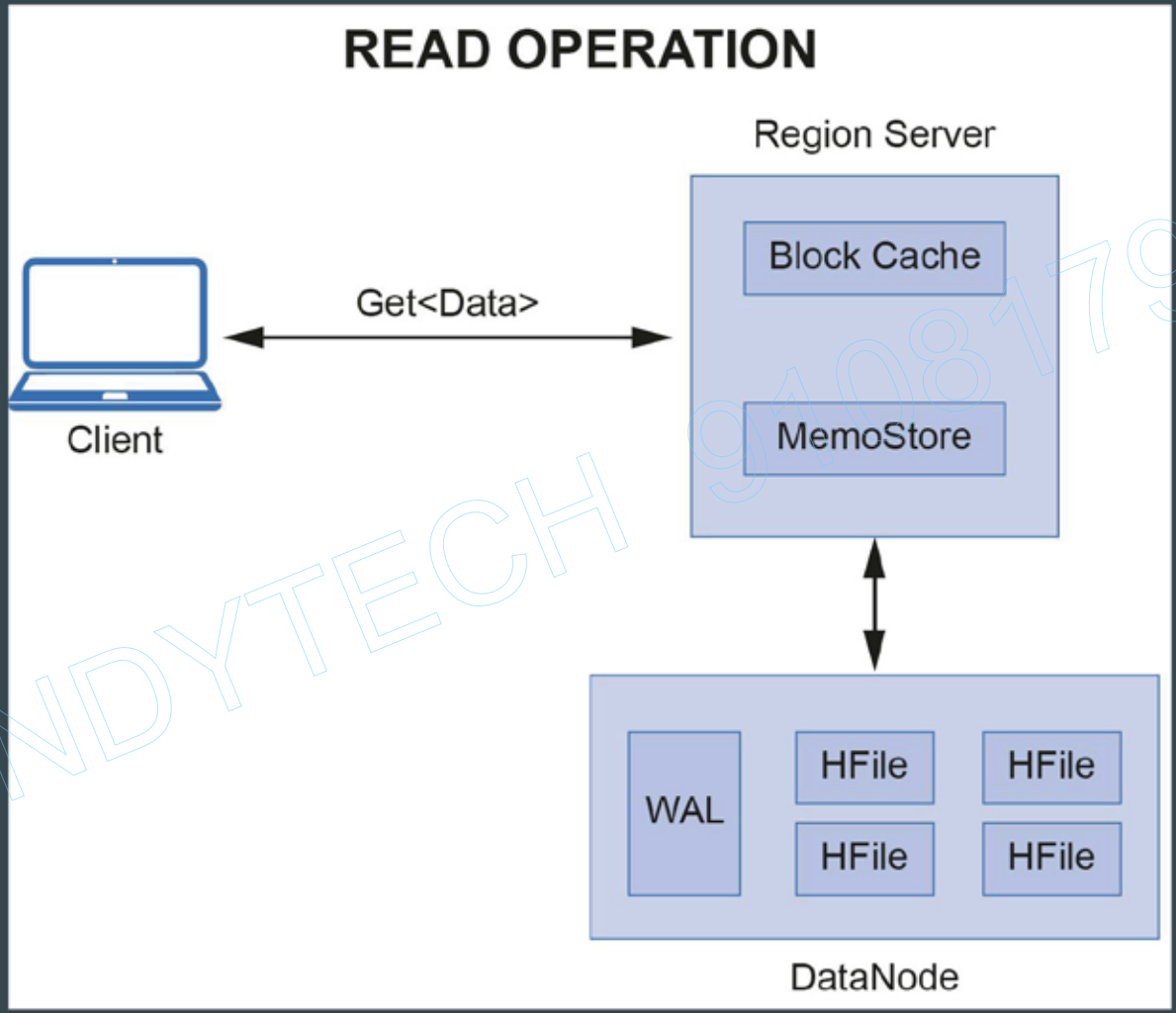
HBase Read Operation

Step 1: The region server first checks the block cache of the region server that stores the recently accessed data.

Step 2: In case the data is not available in the block cache, it checks for the required data in the in-memory store, i.e., MemStore.

Step 3: If the MemStore doesn't contain that particular key-value, then HFile containing that particular key-value pair is identified.

Once the HFile is identified, Instead of reading the entire HFile(~GB size), the **Data Block Index** of the HFile is scanned to get the Data Block with the key-value pair, A binary search in this data block finally returns the key-value pair (or null in case it doesn't exist).



Compactions

Flushes from MemStore to the HDFS create multiple HFiles, especially during periods of heavy incoming writes. which leads to two major problems:

1. The read efficiency gets low. This is because a large number of HFiles increases the number of disk seeks needed for a read process, thereby adversely impacting the read performance.

2. It leads to dirty data. A large number of HFiles might result in a lot of redundancy and inconsistency in data.

Compactions

Compaction comes to our rescue against these challenges. It refers to a process of combining small HFiles to large HFiles, containing merged, sorted, and most recent information.

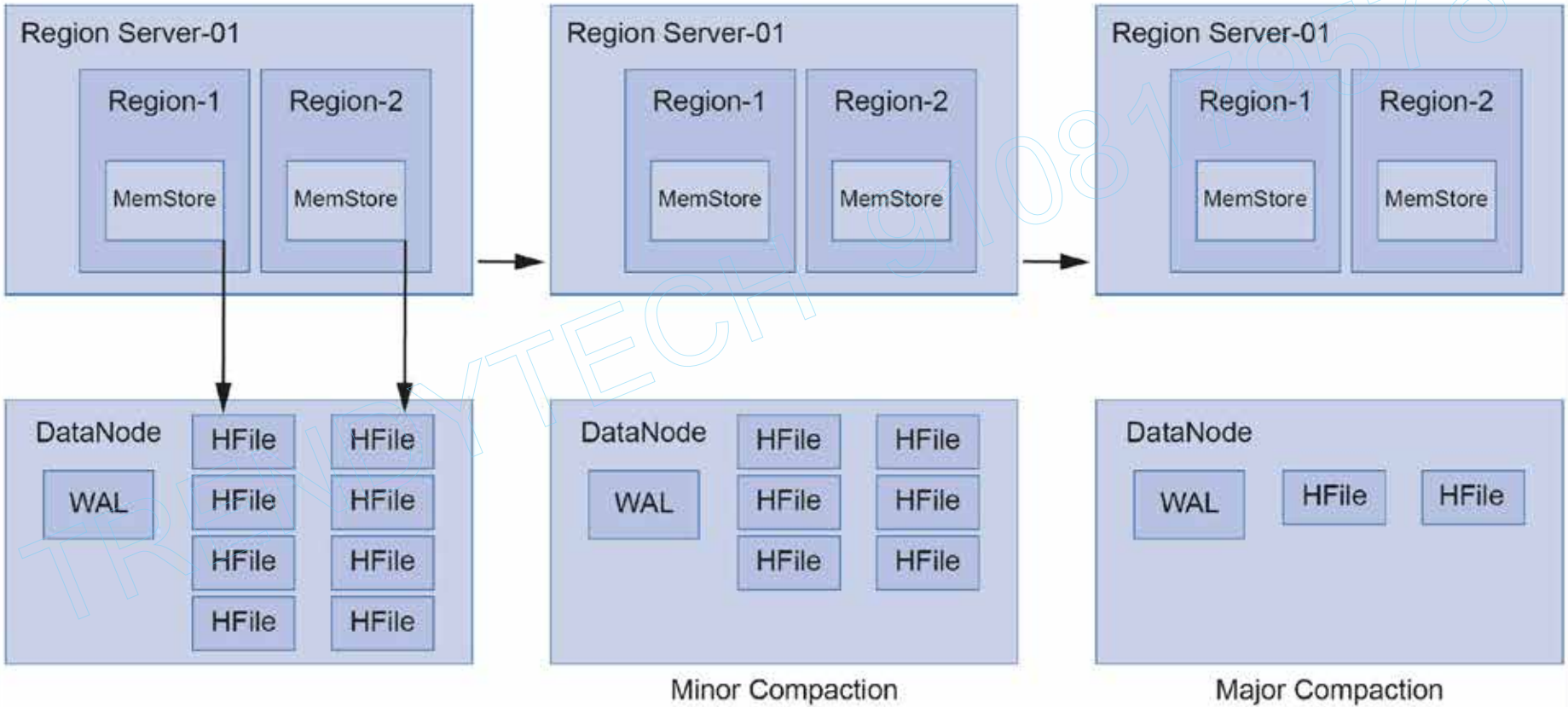
Compactions are of two types — minor and major.

In minor compaction HBase picks only some of the smaller HFiles and rewrites them into a few larger HFiles.

On the other hand, in major compaction, all HFiles of a store are picked and rewritten into a single large HFile.

COMPACTIONS

Minor and Major



Compactions

Major compaction is more resource intensive than minor compaction. Therefore, admins generally prefer minor compaction to major compactions.

HBase Data Deletion

Delete is a special type of update in HBase, where the values for which the delete request is submitted are not deleted immediately. Rather, these values are masked by assigning a tombstone marker to them. Every request to read these values (with tombstone markers) returns null to the client, which gives the client the impression that the values are already deleted.

The reason why HBase does this is because HFiles are immutable. Recall that HDFS doesn't allow modifying the data of a file. All the values with tombstone markers are permanently removed during the next major compaction.

In the case of server failure

1. If the active HMaster fails, the Zookeeper gives the master's reportability to one of the inactive HMaster’s in the cluster.

2. If a region server fails, Zookeeper notifies the HMaster about it.

HMaster reassigns the regions of the failed region server to some other region server.

We have learnt Hbase Architecture

Happy Learning!!!

Google REVIEWS

5 Star Google Rated
Big Data Course

LEARN FROM THE EXPERT

9108179578

Call for more details

Follow US

Trainer	Mr. Sumit Mittal
Phone	9108179578
Email	trendytech.sumit@gmail.com
Website	https://trendytech.in/courses/big-data-online-training/
LinkedIn	https://www.linkedin.com/in/bigdatabysumit/
Twitter	@BigdataBySumit
Instagram	bigdatabysumit
Facebook	https://www.facebook.com/trendytech.in/
Youtube	TrendyTech