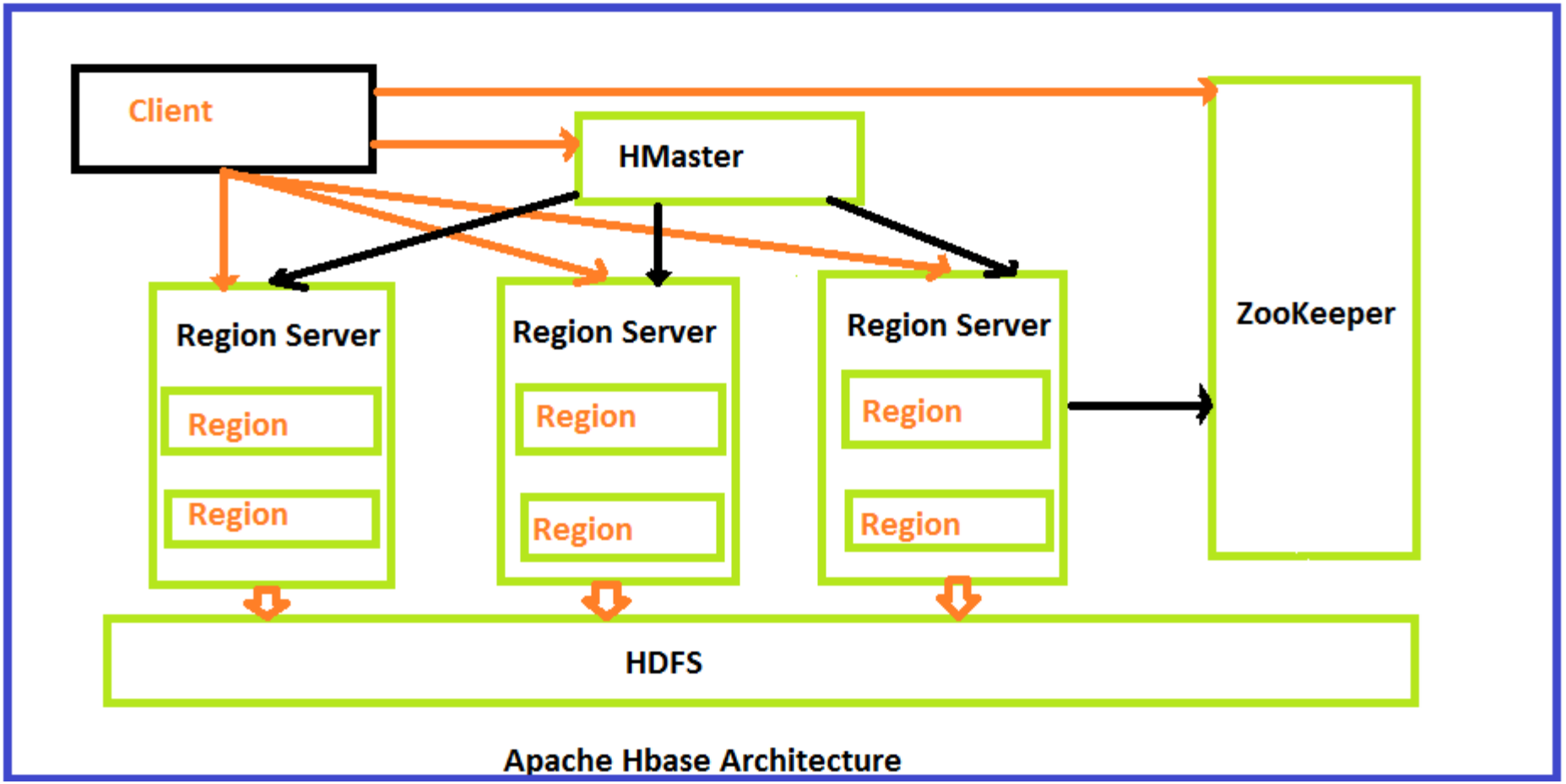
# Hbase Architecture

Tushar B. Kute,
http://tusharkute.com

# Hbase Architecture




Apache Hbase Architecture

# Hbase Components

- HBase architecture consists mainly of four components
  - HMaster
  - HRegionserver
  - HRegions
  - Zookeeper
  - HDFS

# HMaster

- HMaster in HBase is the implementation of a Master server in HBase architecture.
- It acts as a monitoring agent to monitor all Region Server instances present in the cluster and acts as an interface for all the metadata changes.
- In a distributed cluster environment, Master runs on NameNode. Master runs several background threads.

# HMaster

- The following are important roles performed by HMaster in HBase.
  - Plays a vital role in terms of performance and maintaining nodes in the cluster.
  - HMaster provides admin performance and distributes services to different region servers.
  - HMaster assigns regions to region servers.
  - HMaster has the features like controlling load balancing and failover to handle the load over nodes present in the cluster.
  - When a client wants to change any schema and to change any Metadata operations, HMaster takes responsibility for these operations.

# HMaster

- Some of the methods exposed by HMaster Interface are primarily Metadata oriented methods.
  - Table (createTable, removeTable, enable, disable)
  - ColumnFamily (add Column, modify Column)
  - Region (move, assign)
- The client communicates in a bi-directional way with both HMaster and ZooKeeper. For read and write operations, it directly contacts with HRegion servers. HMaster assigns regions to region servers and in turn, check the health status of region servers.
- In entire architecture, we have multiple region servers. Hlog present in region servers which are going to store all the log files.

# Hbase Region Server

- When HBase Region Server receives writes and read requests from the client, it assigns the request to a specific region, where the actual column family resides.
- However, the client can directly contact with HRegion servers, there is no need of HMaster mandatory permission to the client regarding communication with HRegion servers.
- The client requires HMaster help when operations related to metadata and schema changes are required.

# Hbase Region Server

- HRegionServer is the Region Server implementation. It is responsible for serving and managing regions or data that is present in a distributed cluster.
- The region servers run on Data Nodes present in the Hadoop cluster.
- HMaster can get into contact with multiple HRegion servers and performs the following functions.
  - Hosting and managing regions
  - Splitting regions automatically
  - Handling read and writes requests
  - Communicating with the client directly

# Hbase Regions

- HRegions are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families.
- It contains multiple stores, one for each column family. It consists of mainly two components, which are Memstore and Hfile.

# Zookeeper

- HBase Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization.
- Distributed synchronization is to access the distributed applications running across the cluster with the responsibility of providing coordination services between nodes.
- If the client wants to communicate with regions, the server's client has to approach ZooKeeper first.
- It is an open source project, and it provides so many important services.

# Zookeeper

- Services provided by ZooKeeper
  - Maintains Configuration information
  - Provides distributed synchronization
  - Client Communication establishment with region servers
  - Provides ephemeral nodes for which represent different region servers
  - Master servers usability of ephemeral nodes for discovering available servers in the cluster
  - To track server failure and network partitions

# Zookeeper

- Master and HBase slave nodes ( region servers) registered themselves with ZooKeeper.
- The client needs access to ZK(zookeeper) quorum configuration to connect with master and region servers.
- During a failure of nodes that present in HBase cluster, ZKquoram will trigger error messages, and it starts to repair the failed nodes.

# HDFS

- HDFS is a Hadoop distributed File System, as the name implies it provides a distributed environment for the storage and it is a file system designed in a way to run on commodity hardware.
- It stores each file in multiple blocks and to maintain fault tolerance, the blocks are replicated across a Hadoop cluster.
- HDFS provides a high degree of fault –tolerance and runs on cheap commodity hardware.
- By adding nodes to the cluster and performing processing & storing by using the cheap commodity hardware, it will give the client better results as compared to the existing one.

# HDFS

- In here, the data stored in each block replicates into 3 nodes any in a case when any node goes down there will be no loss of data, it will have a proper backup recovery mechanism.
- HDFS get in contact with the HBase components and stores a large amount of data in a distributed manner.

# Hbase Data Model

- HBase Data Model is a set of components that consists of Tables, Rows, Column families, Cells, Columns, and Versions.
- HBase tables contain column families and rows with elements defined as Primary keys.
- A column in HBase data model table represents attributes to the objects.

# Hbase Data Model

- HBase Data Model consists of following elements,
  - Set of tables
  - Each table with column families and rows
  - Each table must have an element defined as Primary Key.
  - Row key acts as a Primary key in HBase.
  - Any access to HBase tables uses this Primary Key
  - Each column present in HBase denotes attribute corresponding to object

# Hbase Use Case

- Telecom Industry faces following Technical challenges
  - Storing billions of CDR (Call detailed recording) log records generated by telecom domain
  - Providing real-time access to CDR logs and billing information of customers
  - Provide cost-effective solution comparing to traditional database systems
- Solution:
  - HBase is used to store billions of rows of detailed call records. If 20TB of data is added per month to the existing RDBMS database, performance will deteriorate.
  - To handle a large amount of data in this use case, HBase is the best solution. HBase performs fast querying and displays records.

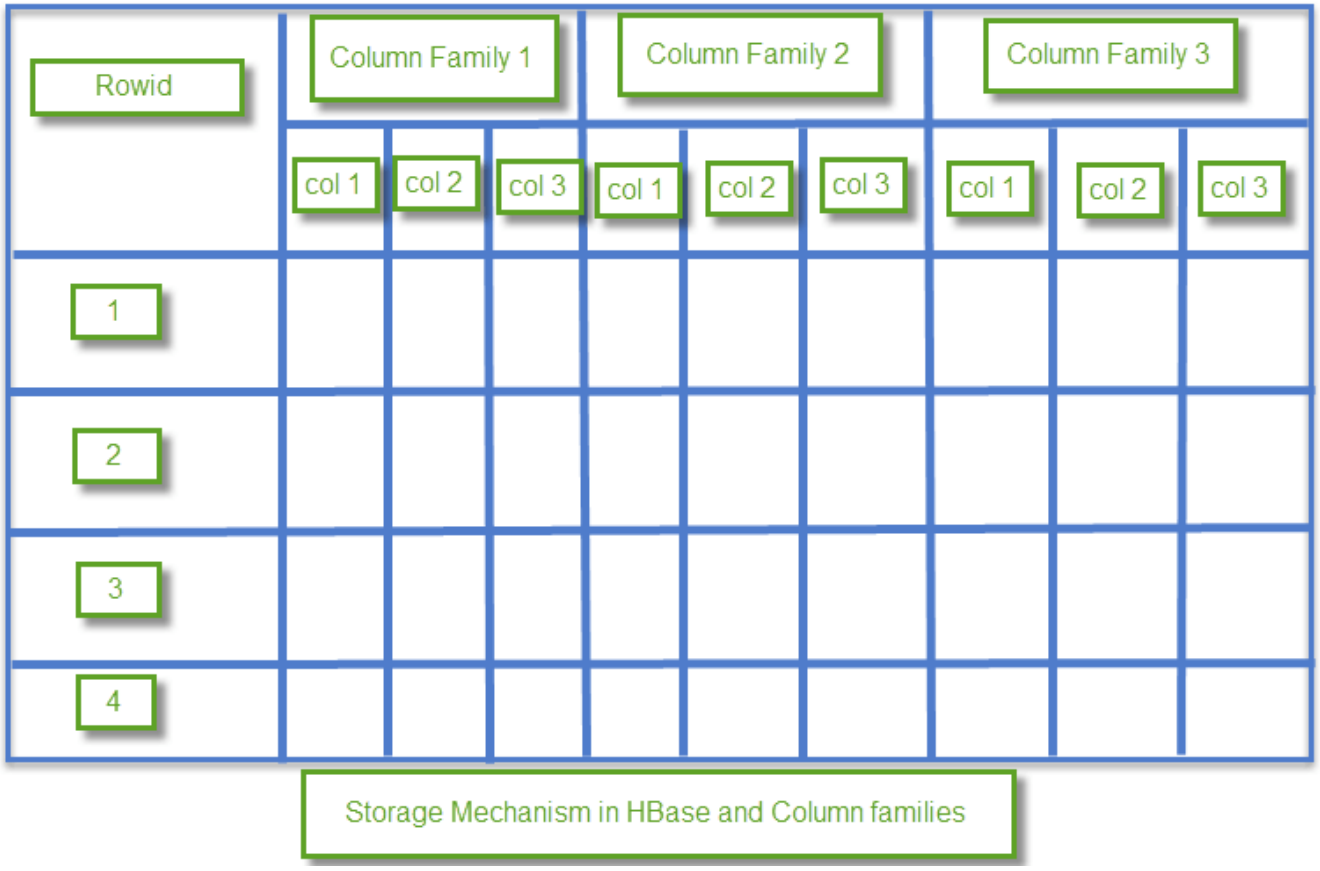
# Hbase Use Case

- The Banking industry generates millions of records on a daily basis.
- In addition to this, the banking industry also needs an analytics solution that can detect Fraud in money transactions
- Solution:
  - To store, process and update vast volumes of data and performing analytics, an ideal solution is – HBase integrated with several Hadoop ecosystem components.

# Hbase Storage Mechanism

- HBase is a column-oriented database and data is stored in tables. The tables are sorted by RowId.
- As shown below, HBase has RowId, which is the collection of several column families that are present in the table.
- The column families that are present in the schema are key-value pairs. If we observe in detail each column family having multiple numbers of columns.
- The column values stored into disk memory. Each cell of the table has its own Metadata like timestamp and other information.

# Hbase Storage Mechanism

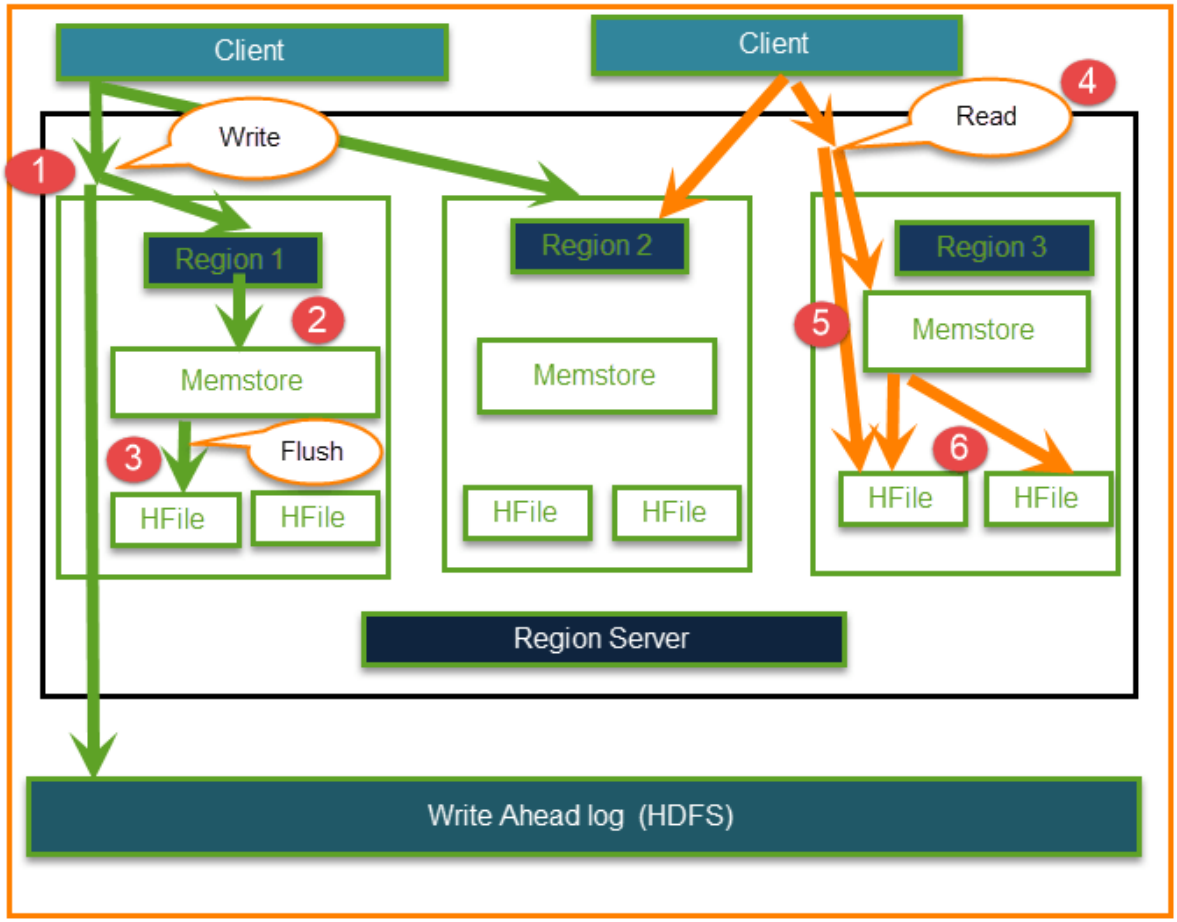



Storage Mechanism in HBase and Column families

# Hbase Storage Mechanism

- Coming to HBase the following are the key terms representing table schema
  - Table: Collection of rows present.
  - Row: Collection of column families.
  - Column Family: Collection of columns.
  - Column: Collection of key-value pairs.
  - Namespace: Logical grouping of tables.
  - Cell: A {row, column, version} tuple exactly specifies a cell definition in HBase.

# Hbase Read Write Operations



# Hbase Operations

- Step 1) Client wants to write data and in turn first communicates with Regions server and then regions
- Step 2) Regions contacting memstore for storing associated with the column family
- Step 3) First data stores into Memstore, where the data is sorted and after that, it flushes into HFile. The main reason for using Memstore is to store data in a Distributed file system based on Row Key. Memstore will be placed in Region server main memory while HFiles are written into HDFS.
- Step 4) Client wants to read data from Regions
- Step 5) In turn Client can have direct access to Mem store, and it can request for data.
- Step 6) Client approaches HFiles to get the data. The data are fetched and retrieved by the Client.

# Hbase Operations

- Memstore holds in-memory modifications to the store. The hierarchy of objects in HBase Regions is as shown from top to bottom in below table.
- Table
  - HBase table present in the HBase cluster
- Region
  - HRegions for the presented tables
- Store
  - It stores per ColumnFamily for each region for the table

# Hbase Operations

- Memstore
  - Memstore for each store for each region for the table
  - It sorts data before flushing into HFiles
  - Write and read performance will increase because of sorting
- StoreFile
  - StoreFiles for each store for each region for the table
- Block
  - Blocks present inside StoreFiles

# Hbase vs. HDFS

- HBase runs on top of HDFS and Hadoop. Some key differences between HDFS and HBase are in terms of data operations and processing.

| HBASE | HDFS |
| --- | --- |
| Low latency operations | High latency operations |
| Random reads and writes | Write once Read many times |
| Accessed through shell commands, client API in Java, REST, Avro or Thrift | Primarily accessed through MR (Map Reduce) jobs |
| Storage and process both can be perform | It's only for storage areas |

# Thank you

*This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License*

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

**Web Resources**
https://mitu.co.in
http://tusharkute.com

**contact@mitu.co.in**

**tushar@tusharkute.com**