Hive & Hbase

Karthik M A M

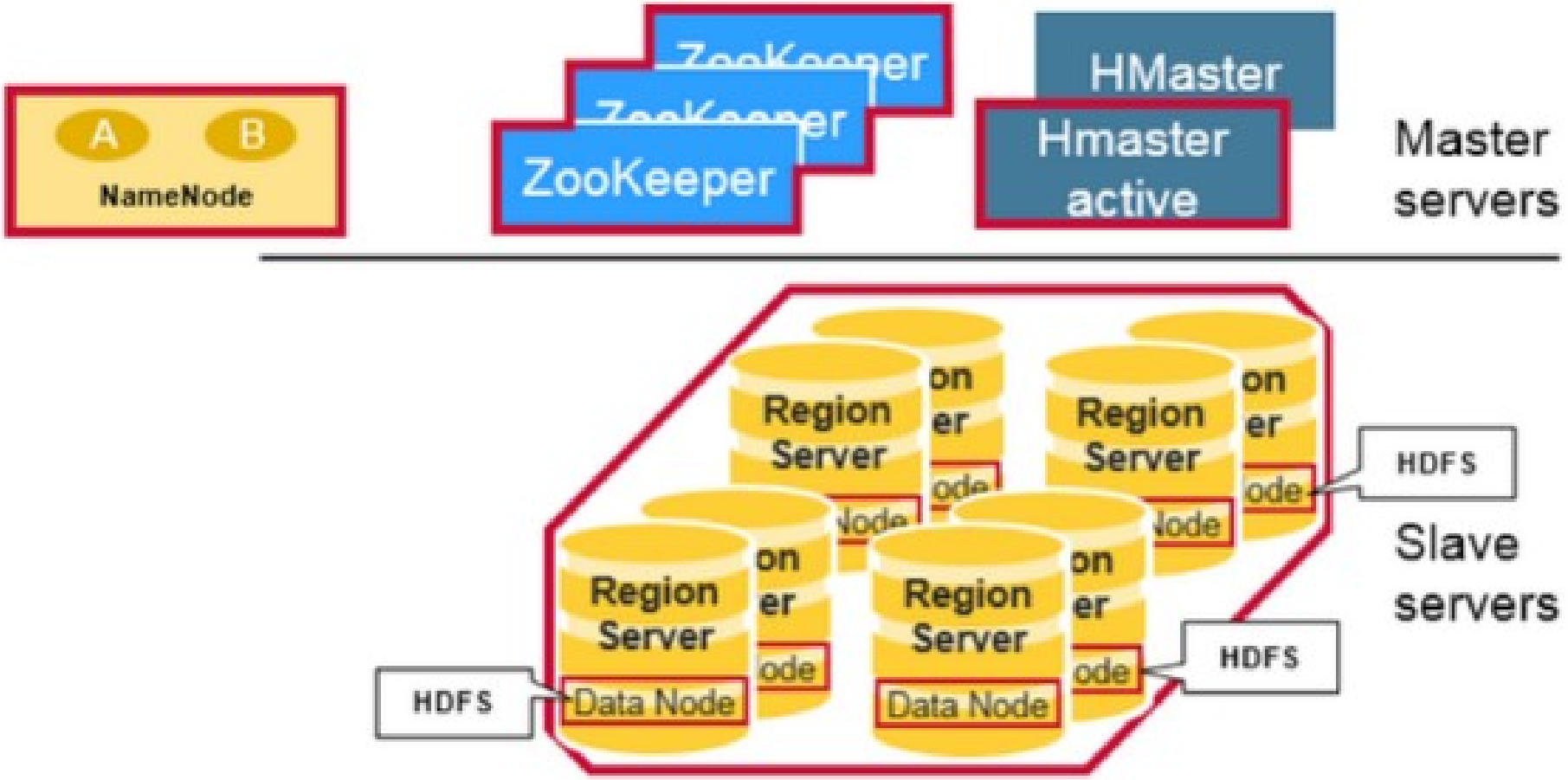
HBase – What?

- Distributed column-oriented database
- Horizontally scalable data model
- Similar to Google’s BigTable
- Quick random access to huge amount of structured data
- Hbase sits on top of HDFS for R/W jobs.

Need of HBase – Why?

- Random access to data.
- Faster Lookups
- Scalability - Horizontal
- Independant of structure

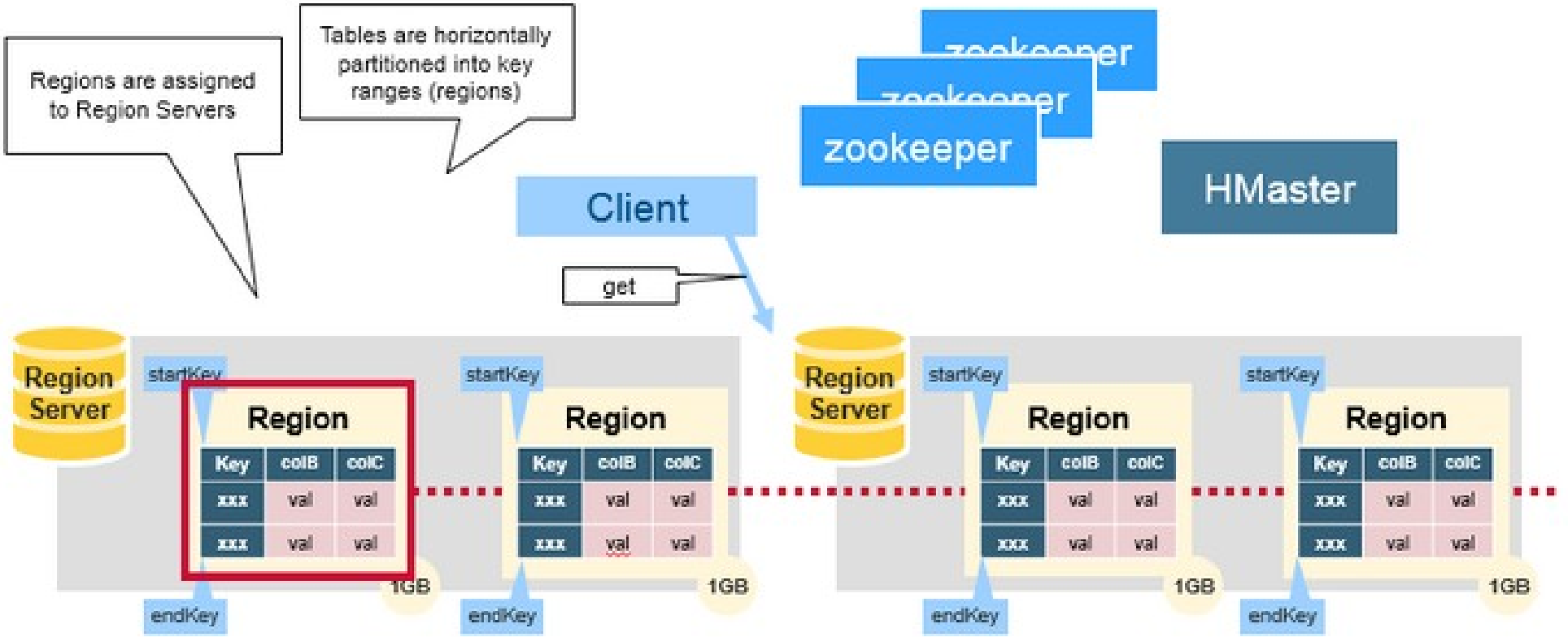
HBase Arch. - How?



HBase Architecture

- Three types of Servers
  - Region Servers
  - HBase Hmaster
  - ZooKeeper
- Arranged in master slave config
- At lowest level, data stored in HDFS datanodes.

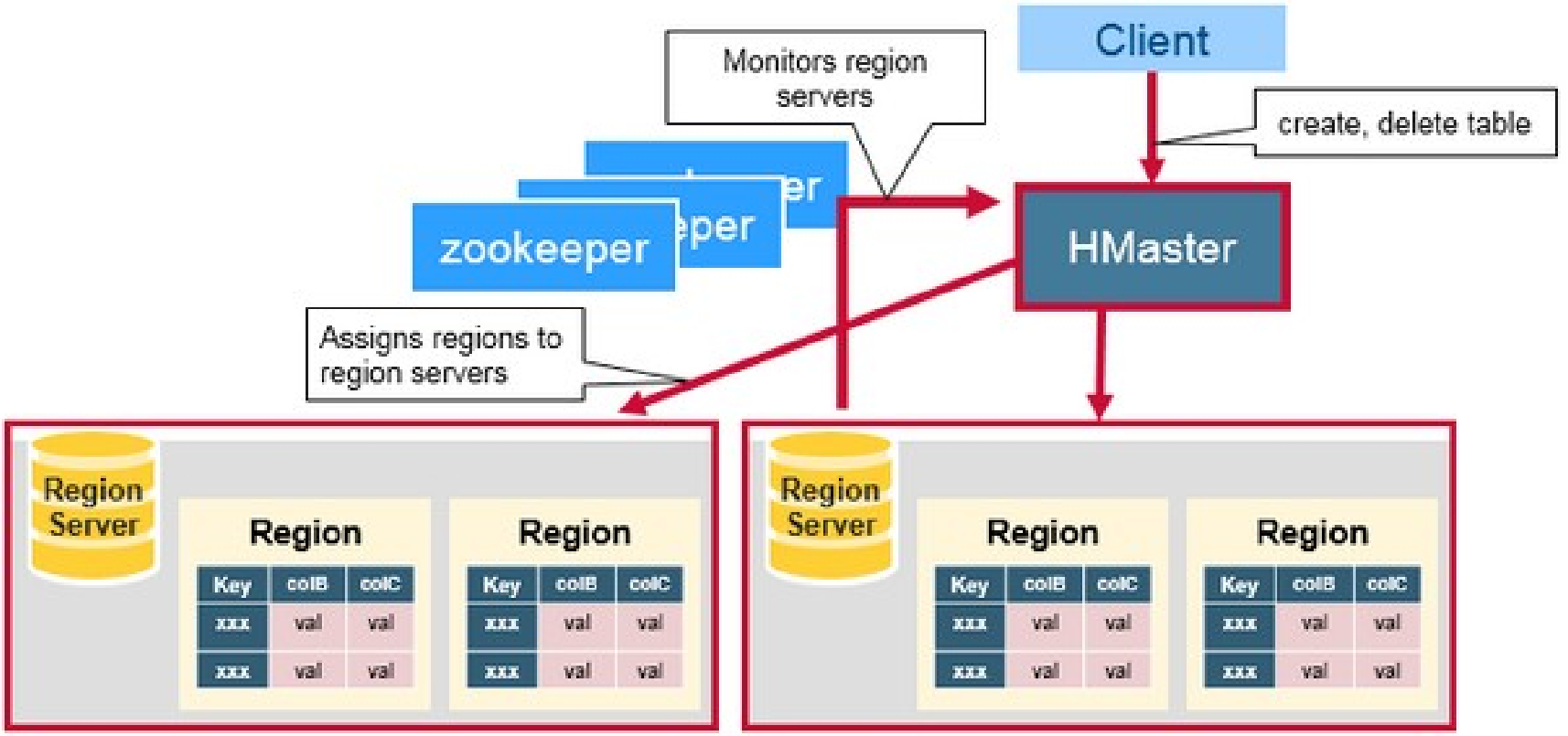
Region Servers



Region Servers

- Slave servers
- Serves data for read & write
- Accessing data – direct communication
- Regions – rows[start_key:end_key + 1]
- Regions form a cluster.

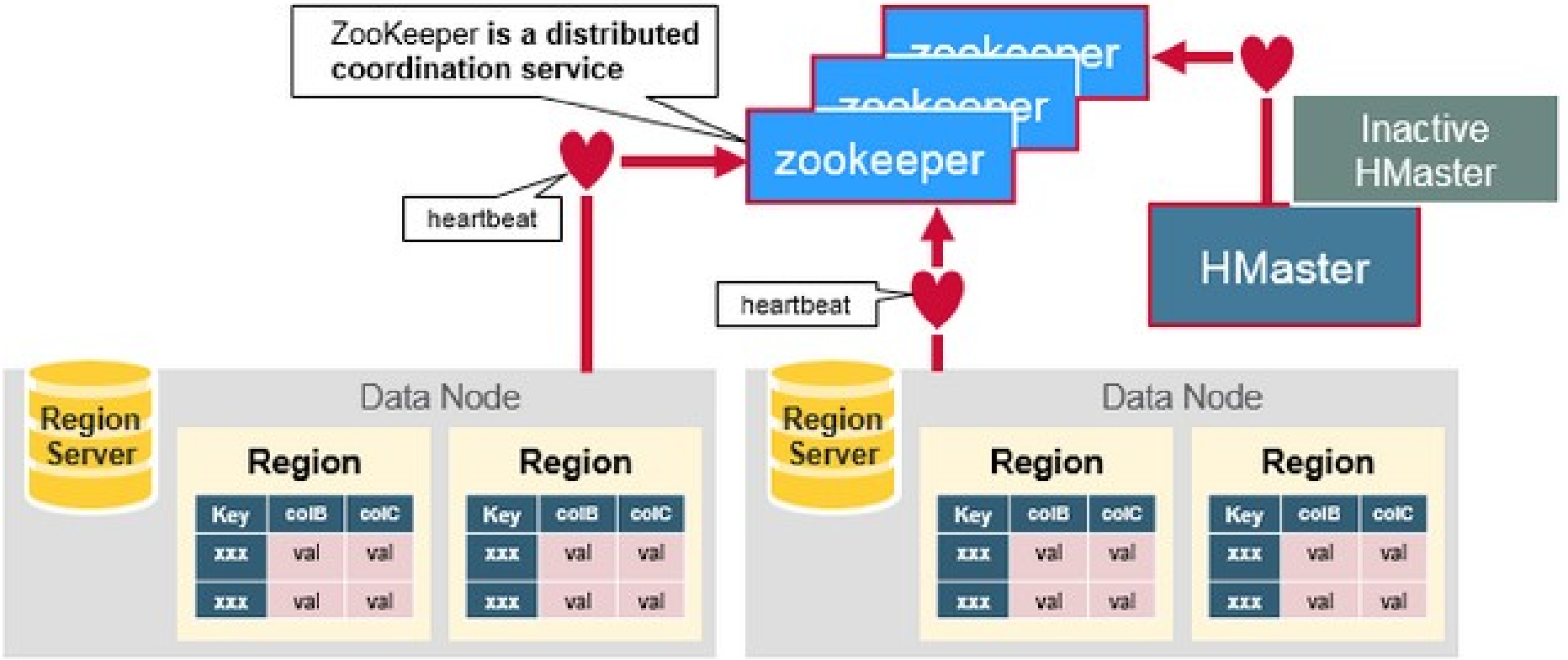
HBase HMaster



HBase HMaster

- Jobs: Coordination, DDL
- Coordinating region servers
  - Assign and re-assign regions for recovery & load balancing.
- DDL
  - Interface for creating, deleting tables.

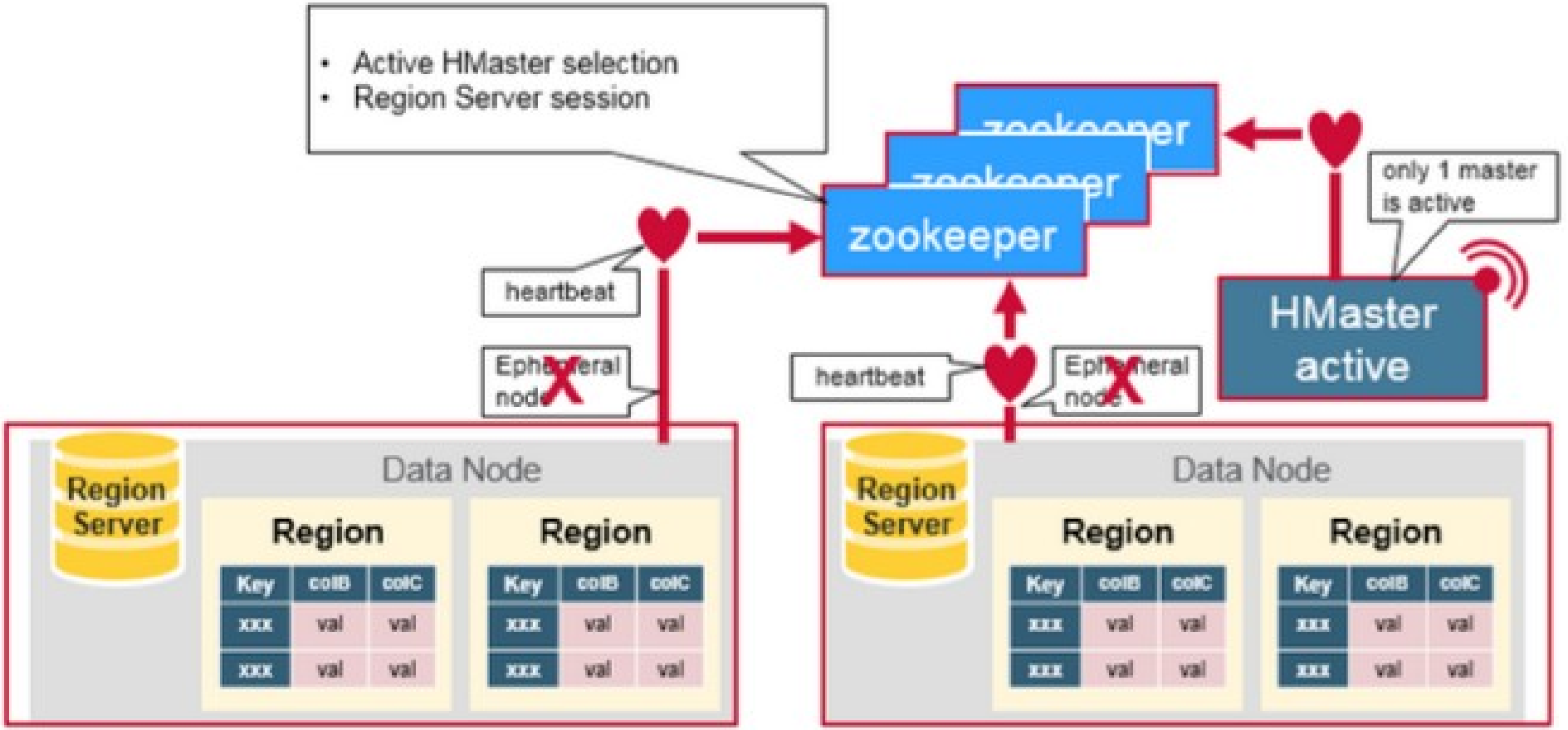
ZooKeeper


ZooKeeper is a distributed coordination service
zookeeper
zookeeper
zookeeper
heartbeat
heartbeat
Inactive HMaster
HMaster
Region Server
Data Node
Region
Key
colB
colC
xxx
val
val
Region
Key
colB
colC
xxx
val
val
Region Server
Data Node
Region
Key
colB
colC
xxx
val
val
Region
Key
colB
colC
xxx
val
val

ZooKeeper

- Distributed coordination service
- Maintains a live cluster
- Failure notification
- Consensus to gurantee common shared state

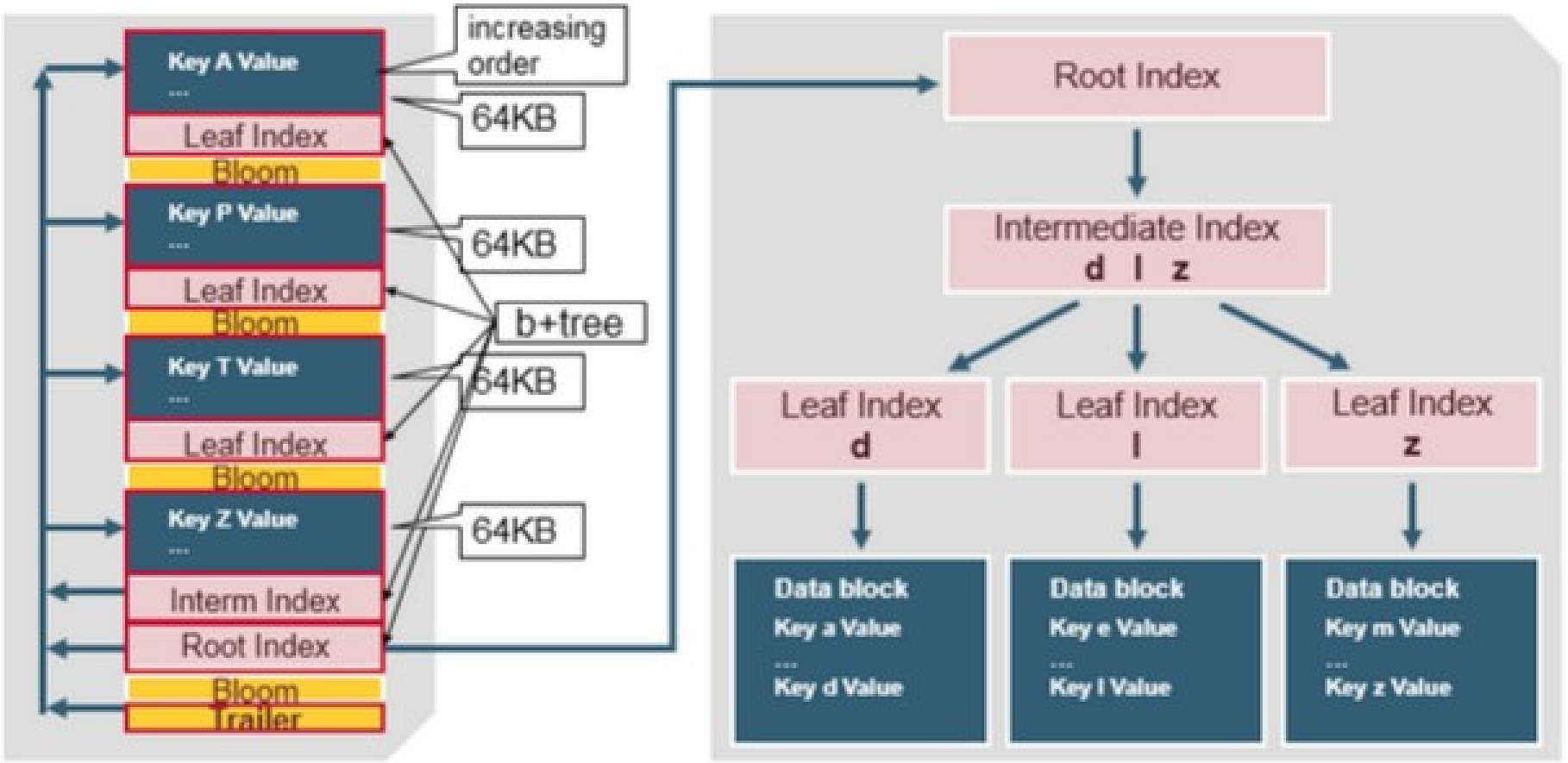
Interaction of components



Interaction of components

- ZooKeeper – coordinates
- Region servers & active HMaster session with ZooKeeper
- Ephimeral nodes for active session via heartbeats
- HeartBeat and failure notification

HBase HFile Structure



HBase HFile Structure

- Multi-layered index like a B+ tree
- Trailer points to meta blocks
- Uses bloon filters to skip blocks that don’t have row key.

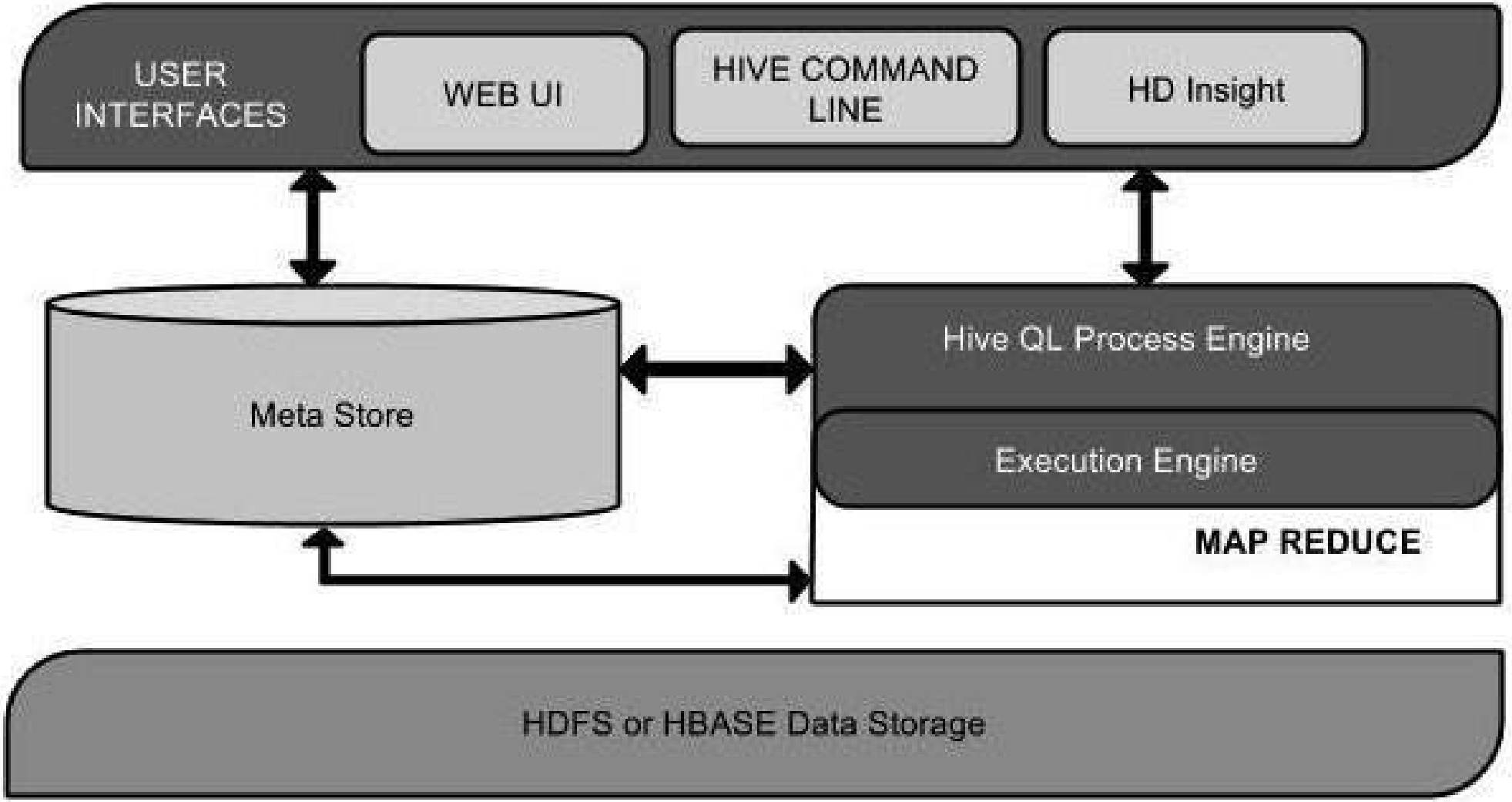
Hive – What?

- Data warehouse infrastructure tool
- Built on top of Hadoop
- Provides Data summarization, query & analysis

Need for Hive – Why?

- SQL-like interface to query data from HDFS
- Familiar, Fast and scalable

Hive Arch. – How?



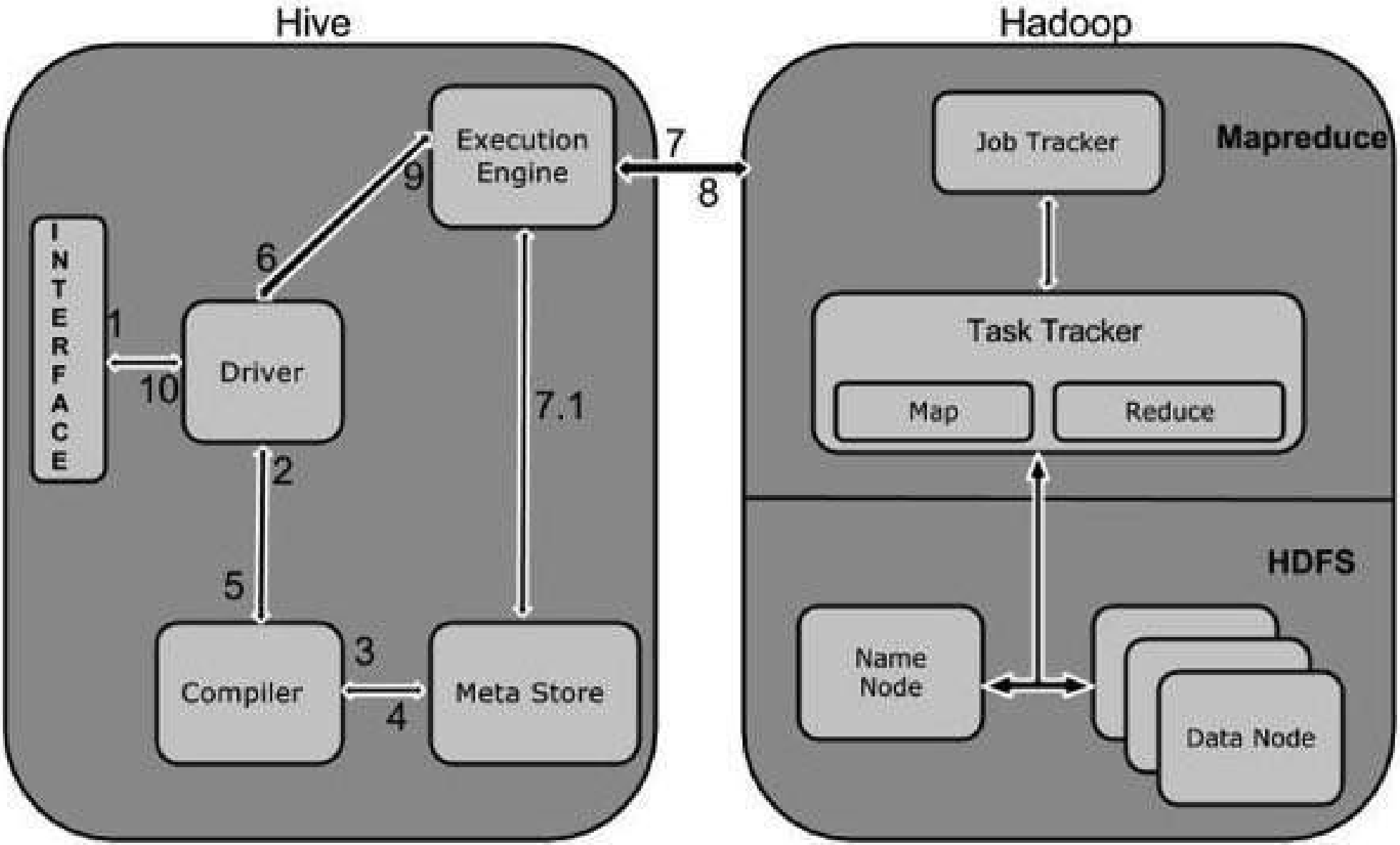
Components

- UI – Hive Web UI, Hive CLI
- Metastore
  - RDMS store of metadata
  - Partitioned for keeping track of data over clusters
- HiveQL Process Engine
  - Write MapReduce Queries
  - Compliation of HiveQL

Components

- Execution Engine
  - Conjunction of HiveQL PE and MapReduce
  - Executes compiled and optimized HiveQL for result.
  - Takes care of task pipelining
  - Executes task with prerequistes run – DAG
- HDFS or HBASE
  - Stores data into a distributed file system.

Working of Hive



Working of Hive

- Execute Query
  - UI sends HiveQL for exeution
- Get Plan
  - Driver send this HiveQL for execution plan
- Get Metadata
  - Compiler needs info for plan generation
- Send Plan
  - Check requirements and send plan

Working of Hive

- Execute Plan
  - Now, that a plan in here, execute it
- Metadata Ops
  - Execution of metadata ops with metastore
- Fetch Result
  - Receive result from data nodes
- Send Result
  - Return results to driver and then to Hive Interface

Simple Example

```
DROP TABLE IF EXISTS docs;

CREATE TABLE docs (line STRING);

LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE
docs;

CREATE TABLE word_counts AS

SELECT word, count(1) AS count FROM

 (SELECT explode(split(line, '\s')) AS word FROM docs) temp

GROUP BY word

ORDER BY word;
```

Referances

- https://hive.apache.org/
- https://www.tutorialspoint.com/hive/
- https://en.wikipedia.org/wiki/Apache_Hive
- https://www.tutorialspoint.com/hbase
- https://www.mapr.com/blog/in-depth-look-hbase-architecture
- https://hbase.apache.org/

Thank You