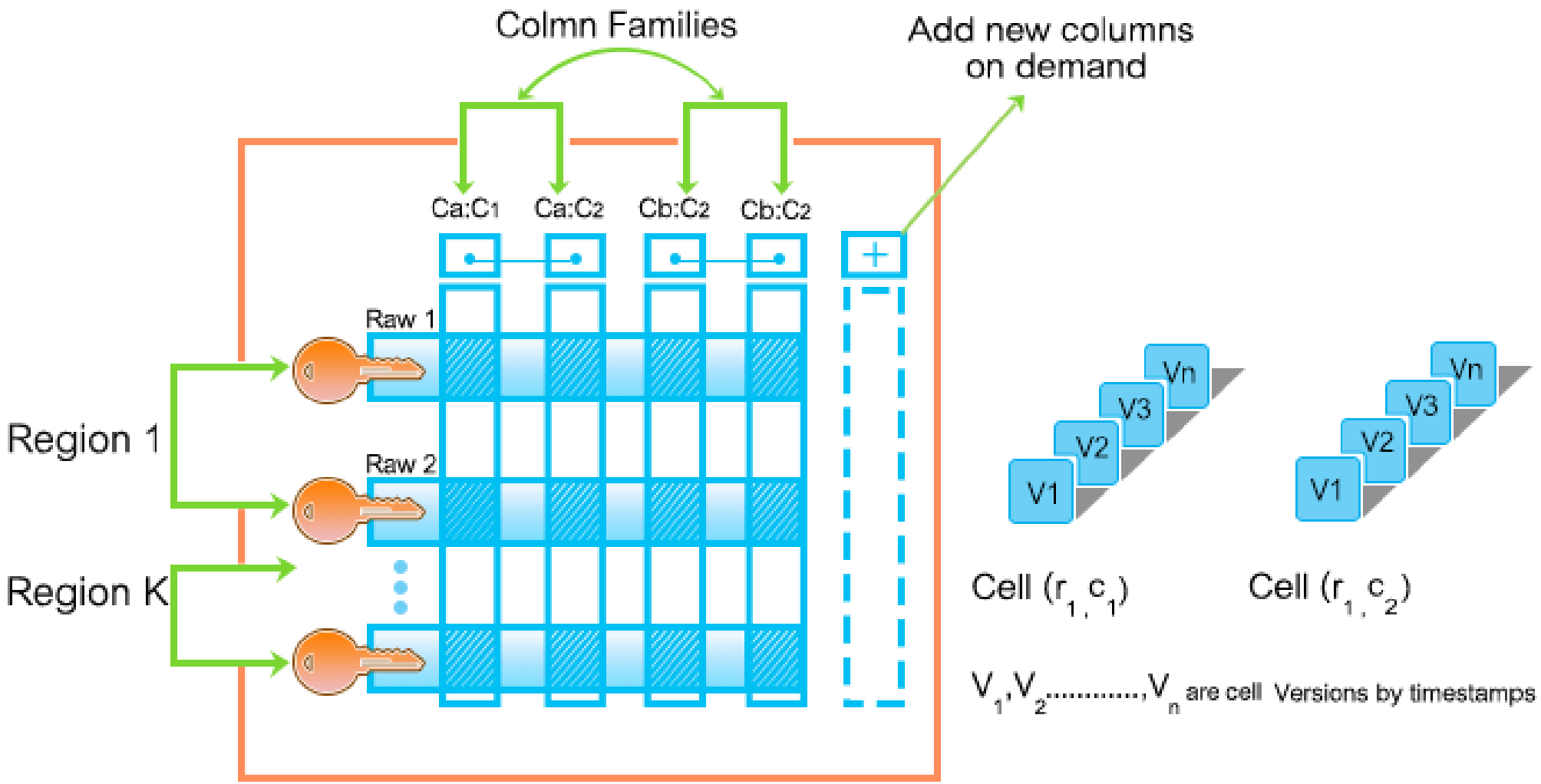
HIVE and HBASE

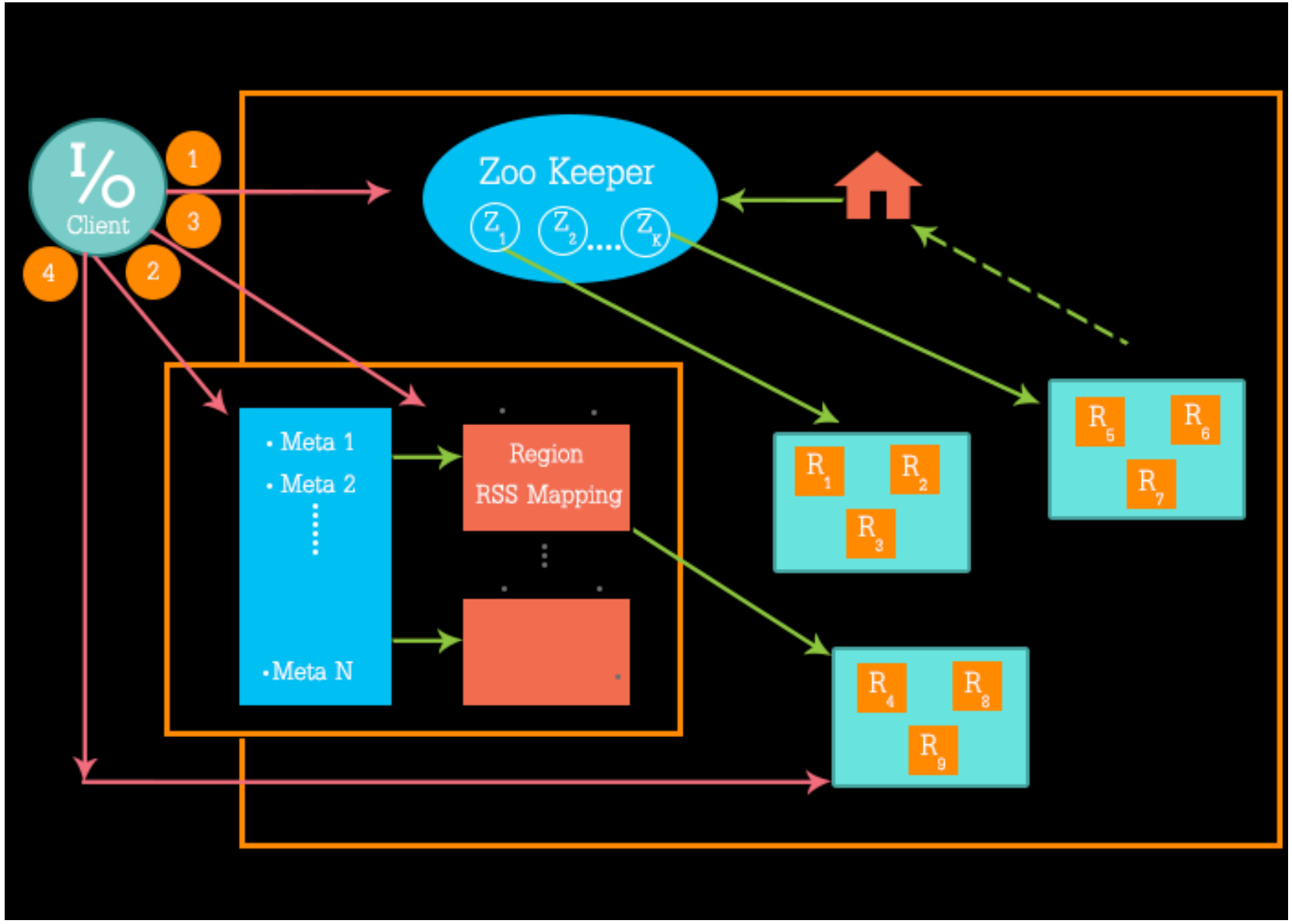
Data Model

- A collection of labelled Tables
- The columns are grouped in column families which can be uniquely identified by column-family prefix
- Every row is uniquely identified by a primary key
- There can be multiple cells with different versions for the same row and column.
- New column families can be added or removed as and when required.


Colmn Families
Add new columns
on demand
Ca:C1
Ca:C2
Cb:C2
Cb:C2
Raw 1
Raw 2
Region 1
Region K
Vn
V3
V2
V1
Vn
V3
V2
V1
Cell (r1, c1)
Cell (r1, c2)
V1,V2.....,Vn are cell Versions by timestamps

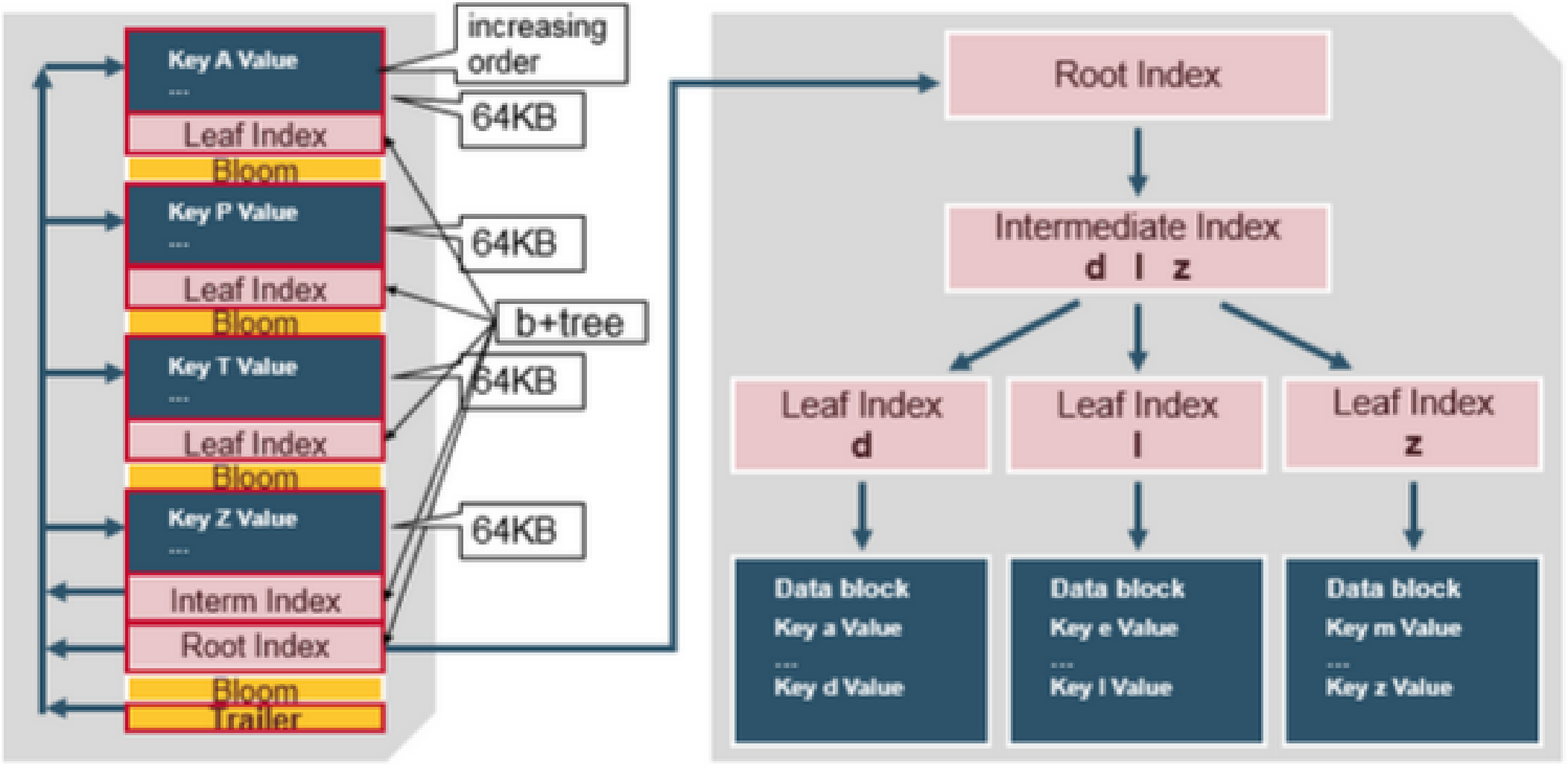
HBase Architecture

- As the number of rows in a table grows beyond a pre-decided limit, the rows are grouped to form regions of the table.
- These regions can be distributed over different nodes(Region Server Slaves) in HBase cluster.
- There is one HBase Master responsible for keeping information and managing regions of table stored on different nodes called Region Server Slaves.
- Zookeeper is appointed by the HBase master to keep active watch over every Region Server Slave by creating a znode entry corresponding to every Region Server Slave.


I/O
Client
1
2
3
4
Zoo Keeper
Z1
Z2
....
Zk
• Meta 1
• Meta 2
•Meta N
Region
RSS Mapping
R1
R2
R3
R5
R6
R7
R4
R8
R9

Hbase file structure

- Data is stored in an HFile which contains sorted key/values.
- An HFile contains a multi-layered index which allows Hbase to seek data without having to read the whole file.
- The multi level index in like a B+ tree.
- Key value pairs are stored in increasing order
- Indexes point by row key to the key value data in 64KB “blocks”
- Each block has its own leaf-index
- The last key of each block is put in the intermediate index
- The root index points to the intermediate index

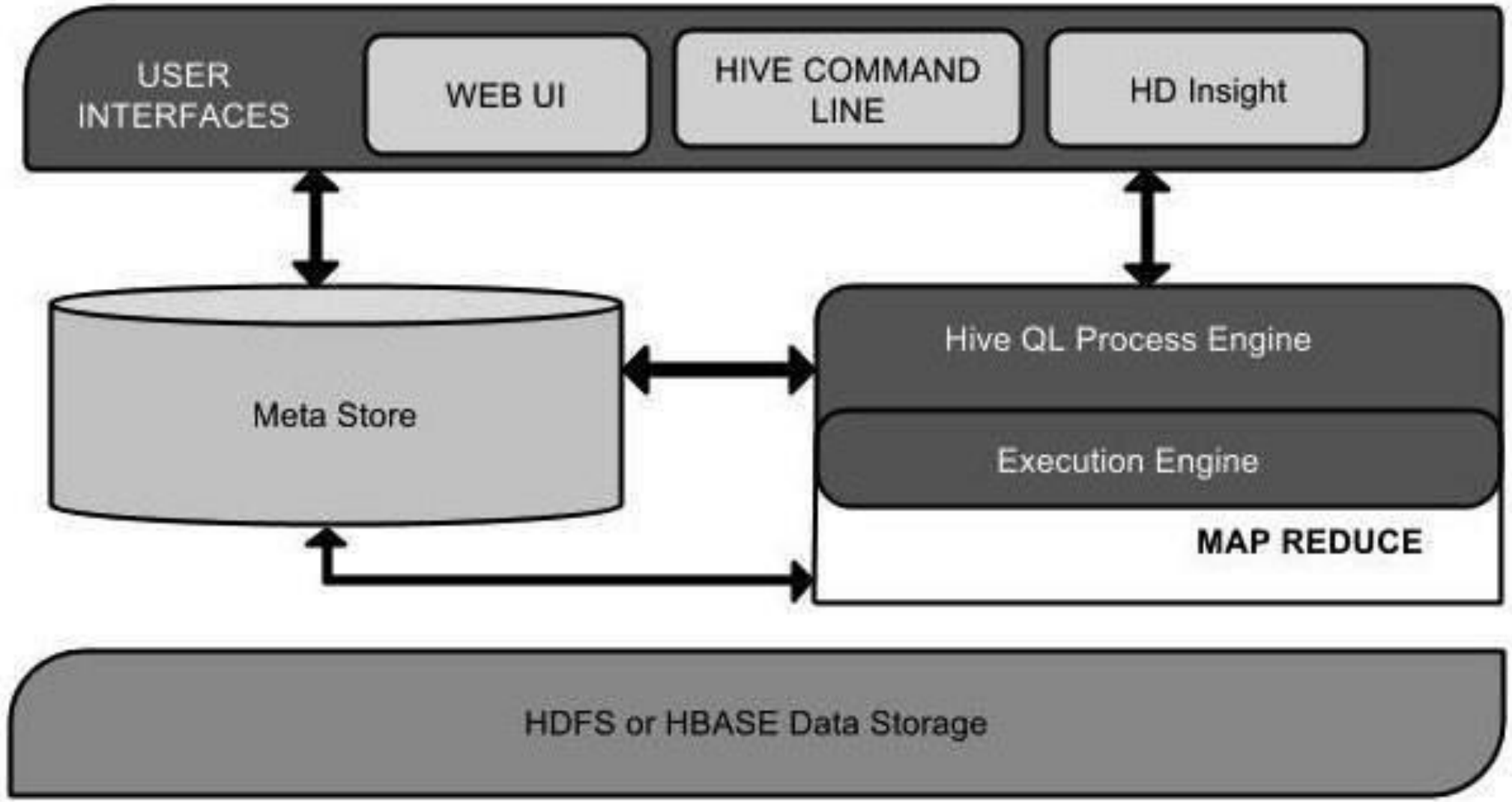

Key A Value
...
Leaf Index
Bloom
Key P Value
...
Leaf Index
Bloom
Key T Value
...
Leaf Index
Bloom
Key Z Value
...
Interm Index
Root Index
Bloom
Trailer
increasing order
64KB
64KB
b+tree
64KB
64KB
Root Index
Intermediate Index
d l z
Leaf Index
d
Leaf Index
l
Leaf Index
z
Data block
Key a Value
...
Key d Value
Data block
Key e Value
...
Key l Value
Data block
Key m Value
...
Key z Value

HIVE

- Hive is an open source data warehouse system for querying and analysing large datasets stored in Hadoop files.
- It has three main functions, data summarization, querying and analysis.
- It supports queries expressed in a language called HIVEQL which automatically translates SQL like queries into MapReduce jobs.
- In addition, it also supports custom MapReduce queries.

Architecture

- Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
- HiveQL is similar to SQL for querying on schema info on the Metastore.
- The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results.
- Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.


USER INTERFACES
WEB UI
HIVE COMMAND LINE
HD Insight
Meta Store
Hive QL Process Engine
Execution Engine
MAP REDUCE
HDFS or HBASE Data Storage

Working of HIVE

1 – Execute Query

2 – Get Plan

3 – Get Meta Data

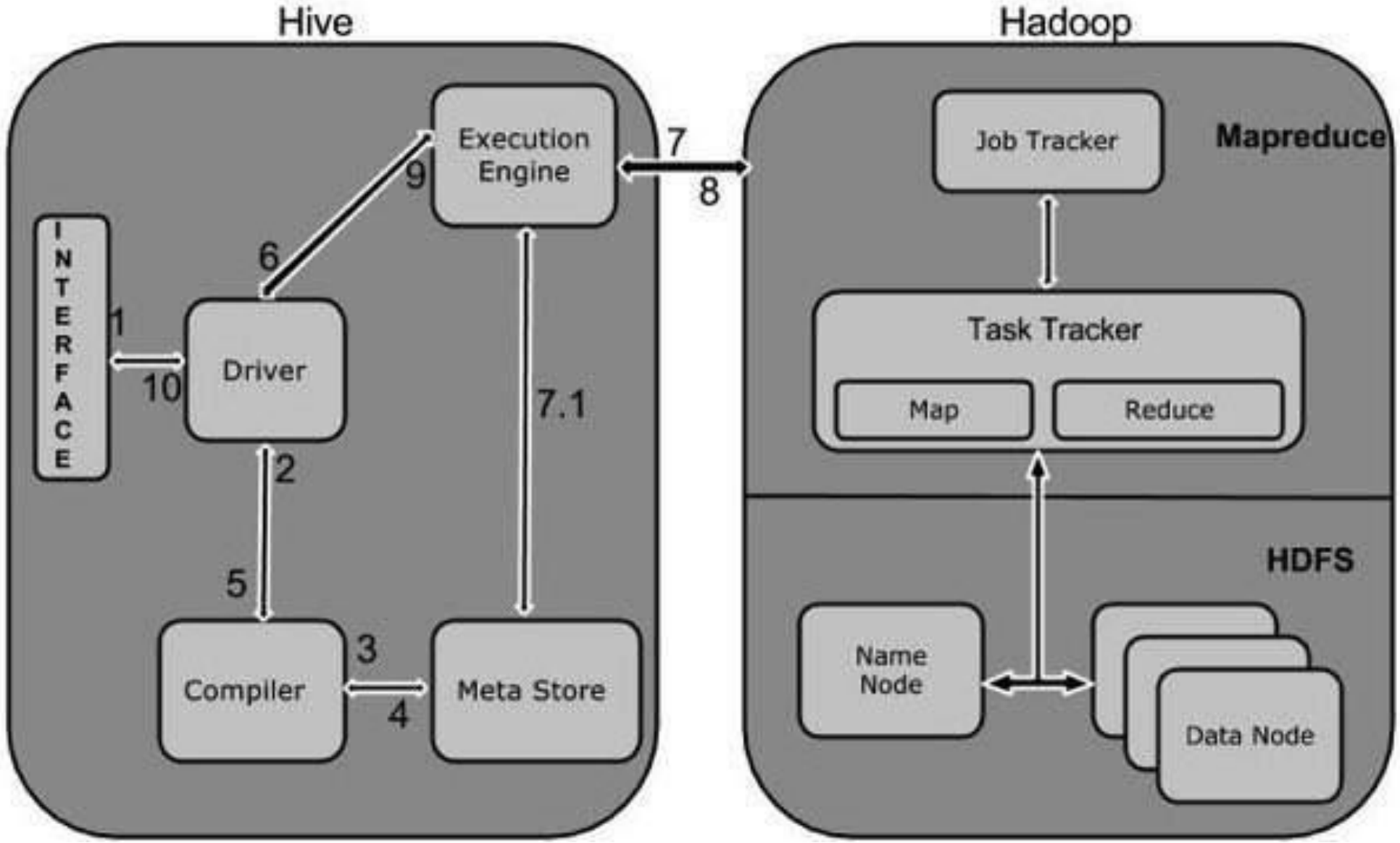
4 - Send Meta Data

5 – Send Plan

6 – Execute Plan

7 – Execute Job (internally MapReduce job)

7.1 – Metadata Ops


Hive
Hadoop
Execution Engine
Job Tracker
Mapreduce
INTERFACE
Driver
Task Tracker
Map
Reduce
Compiler
Meta Store
HDFS
Name Node
Data Node
1
10
2
3
4
5
6
7
8
9
7.1

Working of HIVE

8 – fetch Result

9 – Send results

10 - Send results

Thank You