Apache HBase Overview

Upendra Sabnis

About HBase :

- Hadoop Database – Open source distributed column-oriented database.
- Can serve tables with billions of rows and millions of columns
- Built on top of Hadoop File System (HDFS) : It provides random real-time read/write access to data in HDFS
- HBase is a data model similar to Google's big table designed to provide quick random access to huge amounts of structured data

  Google file system → Google big table

  Hadoop file system → HBase
- One can store data in HDFS directly or through HBase

Why HBase when we have HDFS ?

- HBase leverages the distributed data storage provided by HDFS and provides Bigtable like capabilities on top of hadoop
- HDFS does not support fast individual record lookups. HBase provides fats lookups for larger tables
- HBase provides low latency access to single rows from billions of records
- HBase internally uses hash tables and provides random access and also it stores data in indexed HDFS files for faster lookup

HBase Data Model :

- Data is stored in tables
- Table schema defines only column families
- Table can have multiple column families and each column family can have any number of columns
- Subsequent column values are store contiguously on the disk
- Each cell value of the table has a timestamp

Table → Collection of rows

Row → Collection of column families

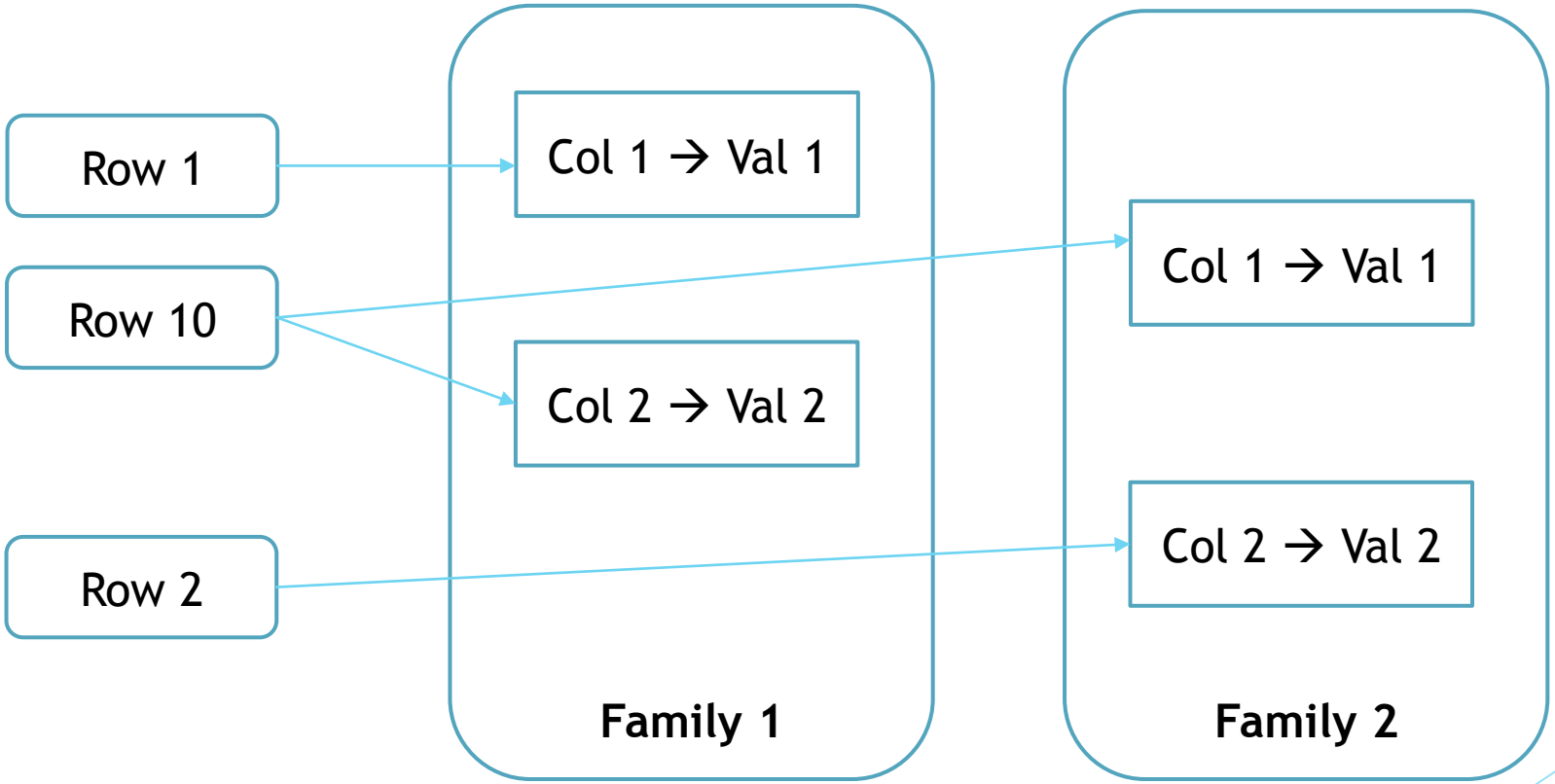
Column Family → Collection of columns

Column → Collection of key value pairs

Example schema of table in HBase :

ROWID	COLUMN FAMILY 1			COLUMN FAMILY2			COLUMN FAMILY 3					
	Col1	Col 2	Col3	Col 1	Col 2	Col 3	Col 1	Col 2	Col 3			
1												
2												
3												

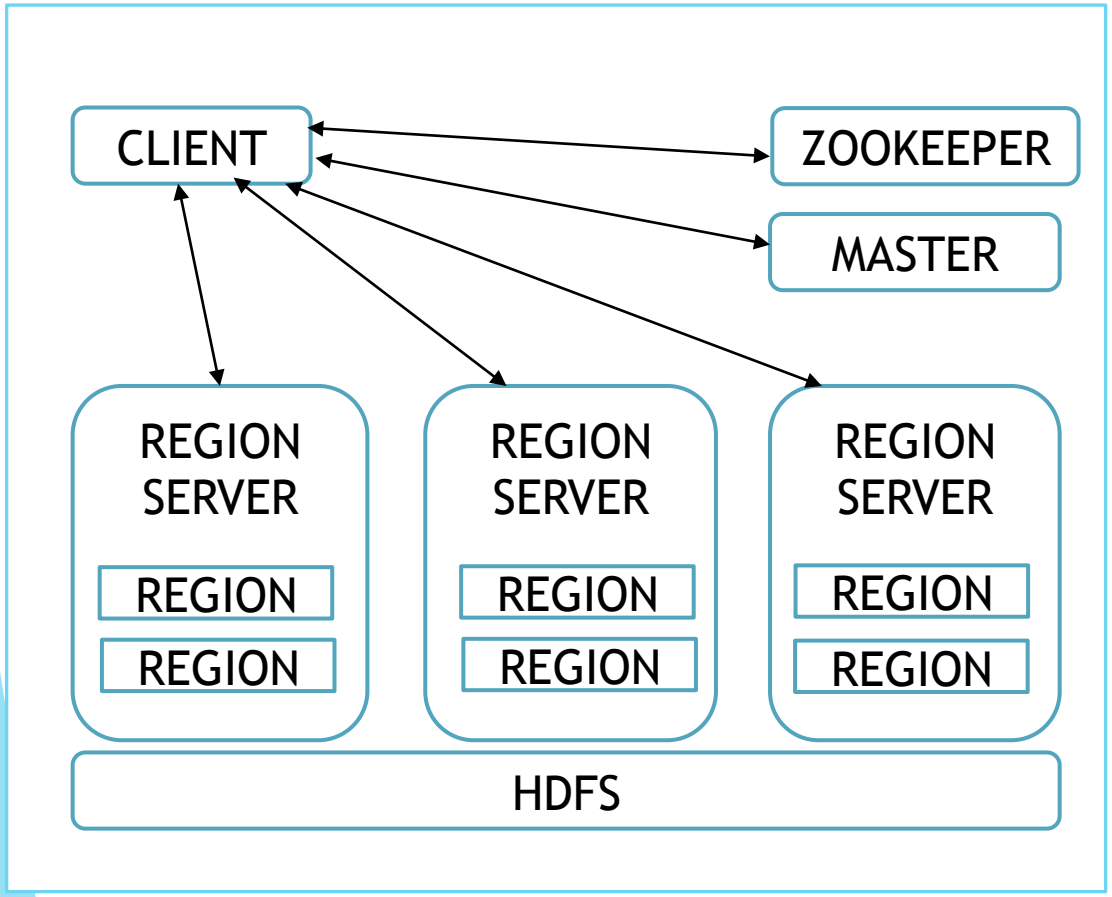
Rows composed of Cells Stored in Families: Columns



HBase and RDBMS :

HBase	RDBMS
Does not have fixed columns schema; defines only column families.	It is governed by schema which describes the whole structure of the table.
Built for wide tables. Horizontally scalable.	Built for small tables. Hard to scale.
No transactions in HBase.	RDBMS is transactional.
De-normalized data.	It will have normalized data.
Good for semi structured and structured data.	Good for structured data.

HBase Architecture : From High level



REGIONS :

- It is the basic unit of scalability in HBase
- Contains subset of table's data
- Contiguous, sorted range of rows

REGION SERVER :

- Contains set of regions
- Handles reads and writes

MASTER :

- Co-ordinates HBase cluster – assignment/balancing of regions
- Handles admin operations – cerate/ delete/modify tables

Autosharding and .META. table

- A Region is a subset of the table's data
- When there is too much data in a region, a split is triggered and 2 regions are created
- The association between this newly created region and server is stored in system table (.META.)
- The location of .META. Is stored in zookeeper

Create a new table :

- Client → Master : create a new table

- Master :

  Store table information

  Create Regions based on the key splits provided (If no splits provided, then one single region by default)

  Assign Regions to the Region Servers : Write the assignment Region → Server to .META.

Insert data / Read data :

- Client asks Zookeeper the location of .META.
- Client scans .META. Searching for the Region Server to handle the key
- Client asks the Region Server to insert/update/delete the specified key/value
- Region Server process the request and dispatch it to the Region responsible to handle the Key

When to use HBase :

- Well-known use cases -
  - Lots and lots of data
  - Large amount of client/requests
- Great for single random selects and range scans by key
- Best for variable schema -
  - Rows may drastically differ
  - Schema has many columns and most of them are null

More features :

- Runs of cluster of commodity hardware
- Horizontally scalable – Automatic sharding
- Strongly consistent reads and writes
- Automatic failover
- It has easy java api for client
- Thrift, Avro and REST web services

Installation and configuration of HBase :

- Follow all the steps for fresh installation from scratch

  http://hbase.apache.org/book.html#architecture

  OR

- Use already configured sandbox from Hortonworks

  http://hortonworks.com/products/hortonworks-sandbox/#install

Commands :

- hbase shell
- create ‘table-name’,’column-family-1’…..
- list
- put, get
- scan
- disable, drop

create ‘cars’,’vi’	list
put ‘cars’,’row1’,’vi:make’,’bmw’	put ‘cars’,’row1’,’vi:model’,’5 series’
put ‘cars’,’row1’,’vi:year’,’2014’	Scan ‘cars’
scan ‘cars’, {COLUMNS => [‘vi:make’], LIMIT => 1}	get ‘cars’,’row1’
delete ‘cars’, ‘row1’, ‘vi:year’	disable ‘cars’
drop ‘cars’	

APACHE HBASE

Resources :

- Apache HBase Homepage :

  http://hbase.apache.org/

- Apache HBase documentation :

  http://hbase.apache.org/book.html#_getting_started

- Hortonworks HBase :

  http://hortonworks.com/hadoop/hbase/

- Others :

  http://courses.coreservlets.com/Course-Materials/pdf/hadoop/03-HBase_1-Overview.pdf

  http://www.tutorialspoint.com/hbase/hbase_overview.htm

THANK YOU !!!