APACHE HBASE

hadoop database

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

This tutorial provides an introduction to HBase, the procedures to set up HBase on Hadoop File Systems, and ways to interact with HBase shell. It also describes how to connect to HBase using java, and how to perform basic operations on HBase using java.

Audience

This tutorial will help professionals aspiring to make a career in Big Data Analytics using Hadoop Framework. Software professionals, analytics Professionals, and ETL developers are the key beneficiaries of this course.

Prerequisites

Before you start proceeding with this tutorial, we assume that you are already aware of Hadoop's architecture and APIs, have experience in writing basic applications using java, and have a working knowledge of any database.

Copyright & Disclaimer

Table of Contents

About the Tutorial i

Audience i

Prerequisites i

Copyright & Disclaimer i

Table of Contents ii

1. HBASE – OVERVIEW 1

What is HBase? 1

HBase and HDFS 2

Storage Mechanism in HBase 2

Column Oriented and Row Oriented 3

HBase and RDBMS 4

Features of HBase 4

Where to Use HBase 4

Applications of HBase 5

HBase History 5

2. HBASE – ARCHITECTURE 6

HBase Architecture 6

3. HBASE – INSTALLATION 8

Pre-Installation Setup 8

Installing HBase 16

Starting and Stopping a Master 20

Starting and Stopping Region Servers 20

HBase Web Interface 21

4. HBASE – SHELL 24

HBase Shell 24

General Commands 24

Data Definition Language 24

Data Manipulation Language 25

Starting HBase Shell 25

5. HBASE – GENERAL COMMANDS 27

status 27

version 27

table_help 27

whoami 28

6. HBASE – ADMIN API 29

Class HBaseAdmin 29

Setting the Classpath 30

7. HBASE – CREATE TABLE 31

Creating Table 31

Verifying the Creation 31

Creating a Table Using java API 32

8. HBASE – LISTING TABLES 35

list 35

Listing Tables Using Java API 35

9. HBASE – DISABLING A TABLE 38

Disable a Table 38

Verification ····· 38

is_disabled ····· 38

disable_all ····· 39

Disable a Table Using Java API ····· 39

10. HBASE – ENABLING A TABLE ····· 42

Enable a Table ····· 42

Verification ····· 42

is_enabled ····· 43

Enable a Table Using Java API ····· 43

11. HBASE – DESCRIBE AND ALTER ····· 46

describe ····· 46

alter ····· 46

Adding a Column Family Using Java API ····· 49

Deleting a Column Family Using Java API ····· 51

12. HBASE – EXISTS ····· 53

exists ····· 53

Verifying the Existence of Table Using Java API ····· 53

13. HBASE – DROP A TABLE ····· 55

drop ····· 55

drop_all ····· 55

Deleting a Table Using Java API ····· 56

14. HBASE – SHUTTING DOWN ····· 59

exit ····· 59

Stopping HBase ····· 59

Stopping HBase Using Java API 59

15. HBASE – CLIENT API 61

Class HBaseConfiguration 61

Class HTable 61

Class Put 62

Class Get 64

Class Delete 64

Class Result 65

16. HBASE – CREATE DATA 67

Creating Data 67

Inserting Data Using Java API 69

17. HBASE – UPDATE DATA 72

Updating Data 72

Updating Data Using Java API 73

18. HBASE – READ DATA 76

Reading Data 76

Reading Data Using Java API 77

19. HBASE – DELETE DATA 80

Deleting a Specific Cell in a Table 80

Deleting All Cells in a Table 80

Deleting Data Using Java API 81

20. HBASE – HBASE SCAN 84

scan **84**

Scanning Using Java API **84**

21. HBASE – COUNT AND TRUNCATE 87

count **87**

truncate **87**

22. HBASE – SECURITY 88

grant **88**

revoke **88**

user_permission **88**

1. HBASE – OVERVIEW

Since 1970, RDBMS is the solution for data storage and maintenance related problems. After the advent of big data, companies realized the benefit of processing big data and started opting for solutions like Hadoop.

Hadoop uses distributed file system for storing big data, and MapReduce to process it. Hadoop excels in storing and processing of huge data of various formats such as arbitrary, semi-, or even unstructured.

Limitations of Hadoop

Hadoop can perform only batch processing, and data will be accessed only in a sequential manner. That means one has to search the entire dataset even for the simplest of jobs.

A huge dataset when processed results in another huge data set, which should also be processed sequentially. At this point, a new solution is needed to access any point of data in a single unit of time (random access).

Hadoop Random Access Databases

Applications such as HBase, Cassandra, couchDB, Dynamo, and MongoDB are some of the databases that store huge amounts of data and access the data in a random manner.

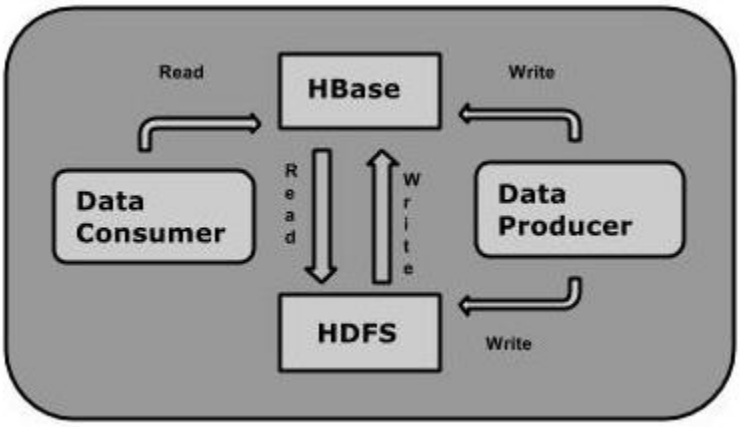
What is HBase?

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



HBase and HDFS

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

Storage Mechanism in HBase

HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

- Table is a collection of rows.

- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

Given below is an example schema of table in HBase.

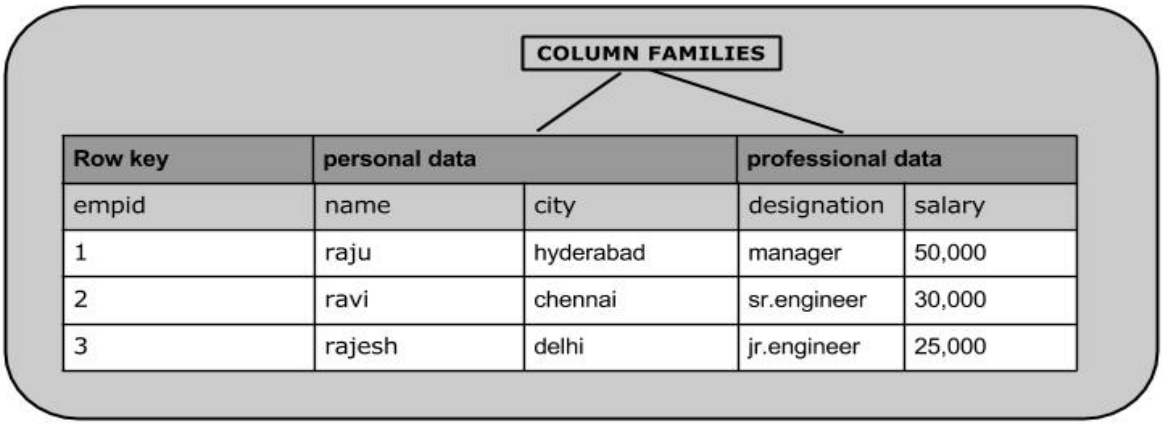
Rowid	Column Family			Column Family			Column Family			Column Family		
	clo1	col 2	col 3	col 1	col 2	col 3	col1	col2	col3	col1	col2	col3
1												
2												
3												

Column Oriented and Row Oriented

Column-oriented databases are those that store data tables as sections of columns of data, rather than as rows of data. Shortly, they will have column families.

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

The following image shows column families in a column-oriented database:



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

HBase and RDBMS

HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.

Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.

- It has easy java API for client.
- It provides data replication across clusters.

Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

Applications of HBase

- It is used whenever there is a need to write heavy applications.
- HBase is used whenever we need to provide fast random access to available data.
- Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

HBase History

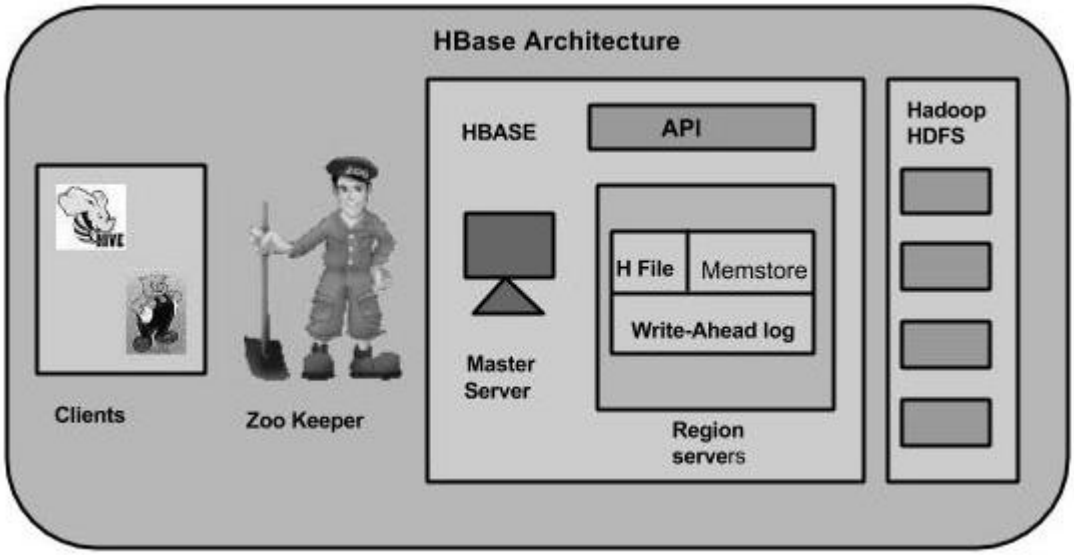
Year	Event
Nov 2006	Google released the paper on BigTable.
Feb 2007	Initial HBase prototype was created as a Hadoop contribution.
Oct 2007	The first usable HBase along with Hadoop 0.15.0 was released.
Jan 2008	HBase became the sub project of Hadoop.
Oct 2008	HBase 0.18.1 was released.
Jan 2009	HBase 0.19.0 was released.
Sept 2009	HBase 0.20.0 was released.
May 2010	HBase became Apache top-level project.

2. HBASE – ARCHITECTURE

HBase Architecture

In HBase, tables are split into regions and are served by the region servers. Regions are vertically divided by column families into "Stores". Stores are saved as files in HDFS. Shown below is the architecture of HBase.

Note: The term 'store' is used for regions to explain the storage structure.



HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement.

Master Server

The master server -

- Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is responsible for schema changes and other metadata operations such as creation of tables and column families.

Regions

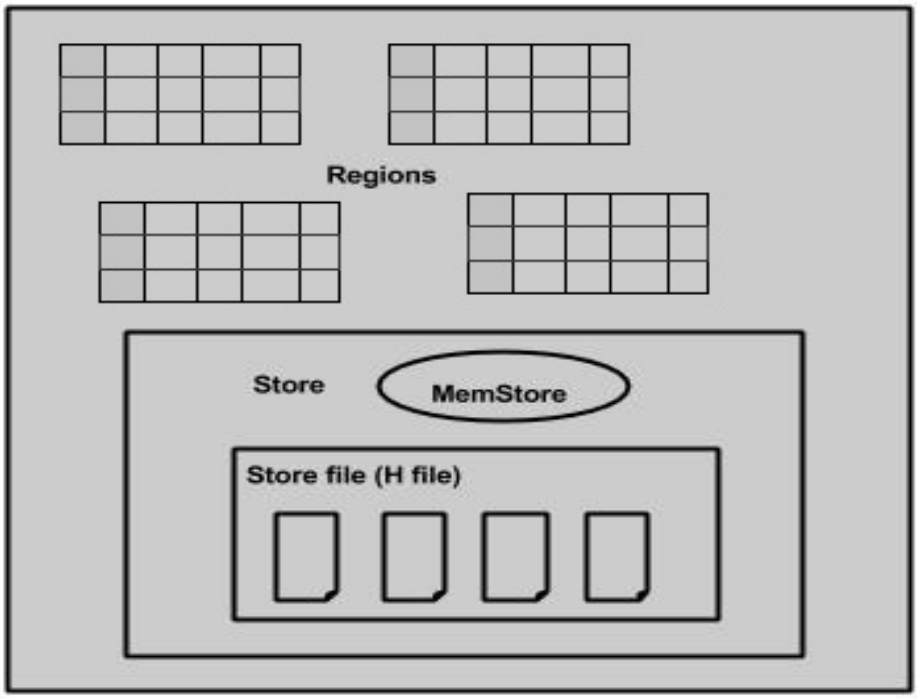
Regions are nothing but tables that are split up and spread across the region servers.

Region server

The region servers have regions that -

- Communicate with the client and handle data-related operations.
- Handle read and write requests for all the regions under it.
- Decide the size of the region by following the region size thresholds.

When we take a deeper look into the region server, it contain regions and stores as shown below:



The store contains memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.

Zookeeper

- Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
- Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.

- In addition to availability, the nodes are also used to track server failures or network partitions.
- Clients communicate with region servers via zookeeper.
- In pseudo and standalone modes, HBase itself will take care of zookeeper.

3. HBASE – INSTALLATION

This chapter explains how HBase is installed and initially configured. Java and Hadoop are required to proceed with HBase, so you have to download and install java and Hadoop in your system.

Pre-Installation Setup

Before installing Hadoop into Linux environment, we need to set up Linux using **ssh** (Secure Shell). Follow the steps given below for setting up the Linux environment.

Creating a User

First of all, it is recommended to create a separate user for Hadoop to isolate the Hadoop file system from the Unix file system. Follow the steps given below to create a user.

1. Open the root using the command "su".
2. Create a user from the root account using the command "useradd username".
3. Now you can open an existing user account using the command "su username".

Open the Linux terminal and type the following commands to create a user.

```
$ su
password:
# useradd hadoop
# passwd hadoop
New passwd:
Retype new passwd
```

SSH Setup and Key Generation

SSH setup is required to perform different operations on the cluster such as start, stop, and distributed daemon shell operations. To authenticate different users of Hadoop, it is required to provide public/private key pair for a Hadoop user and share it with different users.

The following commands are used to generate a key value pair using SSH. Copy the public keys form id_rsa.pub to authorized_keys, and provide owner, read and write permissions to authorized_keys file respectively.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Verify ssh

```
ssh localhost
```

Installing Java

Java is the main prerequisite for Hadoop and HBase. First of all, you should verify the existence of java in your system using "java -version". The syntax of java version command is given below.

```
$ java -version
```

If everything works fine, it will give you the following output.

```
java version "1.7.0_71"
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

If java is not installed in your system, then follow the steps given below for installing java.

Step 1

Download java (JDK <latest version> - X64.tar.gz) by visiting the following link http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.

Then **jdk-7u71-linux-x64.tar.gz** will be downloaded into your system.

Step 2

Generally you will find the downloaded java file in Downloads folder. Verify it and extract the **jdk-7u71-linux-x64.gz** file using the following commands.

```
$ cd Downloads/
$ ls
jdk-7u71-linux-x64.gz

$ tar zxf jdk-7u71-linux-x64.gz
$ ls
```

```
jdk1.7.0_71   jdk-7u71-linux-x64.gz
```

Step 3

To make java available to all the users, you have to move it to the location "/usr/local/". Open root and type the following commands.

```
$ su
password:
# mv jdk1.7.0_71 /usr/local/
# exit
```

Step 4

For setting up **PATH** and **JAVA_HOME** variables, add the following commands to **~/.bashrc** file.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
export PATH= $PATH:$JAVA_HOME/bin
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

Step 5

Use the following commands to configure java alternatives:

```
# alternatives --install /usr/bin/java java usr/local/java/bin/java 2
# alternatives --install /usr/bin/javac javac usr/local/java/bin/javac 2
# alternatives --install /usr/bin/jar jar usr/local/java/bin/jar 2

# alternatives --set java usr/local/java/bin/java
# alternatives --set javac usr/local/java/bin/javac
# alternatives --set jar usr/local/java/bin/jar
```

Now verify the installation using the command **java -version** from the terminal as explained above.

Downloading Hadoop

After installing java, you have to install Hadoop. First of all, verify the existence of Hadoop using " Hadoop version " command as shown below.

```
hadoop version
```

If everything works fine, it will give you the following output.

```
Hadoop 2.6.0
Compiled by jenkins on 2014-11-13T21:10Z
Compiled with protoc 2.5.0
From source with checksum 18e43357c8f927c0695f1e9522859d6a
This command was run using /home/hadoop/hadoop/share/hadoop/common/hadoop-common-2.6.0.jar
```

If your system is unable to locate Hadoop, then download Hadoop in your system. Follow the commands given below to do so.

Download and extract hadoop-2.6.0 from Apache Software Foundation using the following commands.

```
$ su
password:
# cd /usr/local
# wget http://mirrors.advancedhosters.com/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0-src.tar.gz
# tar xzf hadoop-2.6.0-src.tar.gz
# mv hadoop-2.6.0/* hadoop/
# exit
```

Installing Hadoop

Install Hadoop in any of the required mode. Here, we are demonstrating HBase functionalities in pseudo distributed mode, therefore install Hadoop in pseudo distributed mode.

The following steps are used for installing **Hadoop 2.4.1**.

Step 1 - Setting up Hadoop

You can set Hadoop environment variables by appending the following commands to **~/.bashrc** file.

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME


export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

export HADOOP_INSTALL=$HADOOP_HOME
```

Now apply all the changes into the current running system.