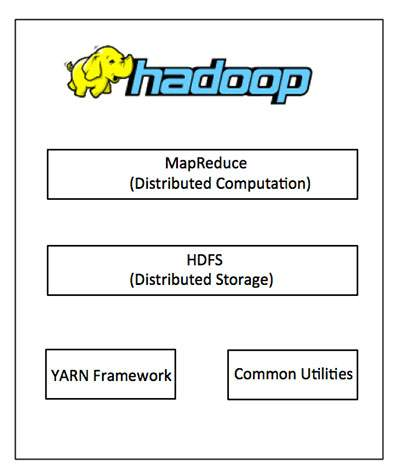
**HDFS**

Distributed Computing

Batch Processing

Written in Java



HDFS -> NameNode - Data manager (ex: keep data indexes)

-> DataNode - Where actual data exists

Fault Tolerance - When hardware fails, users will still have the access to data.

Hadoop Cluster can be managed in 3 ways:

1)Open Source : Hadoop Standalone

2)Commercial Distributions: Cloudera, Hortonworks

3)Public Cloud:

IaaS - We manage ( VM/Docker Images)

PaaS - Cloud Service Provider Manage the infrastructure and

containerizing.

**Why Hadoop? Limitations of Relational Database Management Systems**:

Scalability

Speed / Accessibility

Queriability

Sophisticated Processing ( Ex; Applying Machine Learning)

Database Coises:

1. Files Systems ( Ex: HDFS)
2. No-SQL ( key/value, columnstore etc.)

HDFS is the file system used by Hadoop

**CAP Concept**

1) Consistency: Having db transactions

2) Availability

3) Partitioning

**Hadoop Environment Setup and Installation**

1. MapReduce

Computing model/processing technique to work with data on distributed systems like hadoop.

Written in Java.

2 important tasks : **Map** and **Reduce**

Map: takes a set of data and converts it to another form of data.

Individual elements are broken down into tuples (key-value pairs)

Second: Reduce tasks

Get output of the Map operation ( tuples) as input and

Combine into smaller set of tuples

**Why MapReduce?**

Easy to scale data processing in a distributed system.

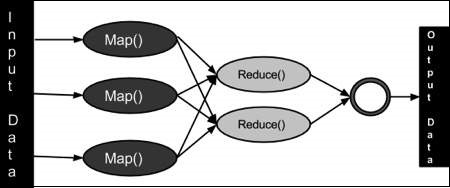
Map Reduce performed in 3 stages:

Map Stage: Process a file (ex: text file) in HDFS line by line.

Shuffle & Reduce Stages: Shuffle the Map output and reduce into   
 small set of data and store the final output in HDFS.

Steps:

* During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
* The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
* Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
* After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



1. YARN Framework

Open up Hadoop by allowing to process and run data for batch processing, stream processing, interactive processing and graph processing, stored in HDFS

YARN Architecture - 2 layers

* Processing layer
* Resource Management Layer

Different kinds of resources are also progressively allocated for

optimum utilization

This is the Data processing Operating System which acts on top of

HDFS

Why YARN?

MapReduce has only a single Job Tracker

MapReduce is Programming Model, YARN is architecture for

distribution cluster.

**Namenode**

Namenode acts as the master server.

It responsible for:

Manage file systems namespaces

Regulate client;s access to files

Executes file system operations such as renaming, closing, and

opening files and directories.

**Datanode**

Manage the data storage in the system.

Perform read-write operations on the file systems, as per client

request.

Perform operations like block creation, deletion, and replication

A file in the HDFS file system divides into one or more segments and is stored.

Each segment will have a datanode.

Each segment is called a **block**

**Actions in Hadoop**

Create Files

Append files

Read Files

Delete Files

Note: It does not allow us to Modify files (Only to create or Delete)

Note: Files are actually kept in **Temp** directory.

**HDFS Shell Interaction**

File System Commands: Shell-like commands

HDFS “ dfs commands

HDFS Navigation Commands

Command Format:

1. hdfs dfs - [command]

Ex: hdfs dfs - ls

Hdfs dfs - ls /user

1. Hadoop fs - [command]

Both ways are correct:

**Hdfs dfs -ls == Hadoop fs -ls**

**Basic shell commands**

hdfs dfs -touchz file\_name -> create new file

hdfs dfs -cat file\_name -> display content

Hdfs dfs -mkdir /dir\_name -> create new directory

Hdfs dfs -cp file\_a /dir\_name/file\_b -> copy files

Hdfs dfs -mv -> move files

**Maintenance shell commands**

Hdfs dfs -expunge -> empties trash

Hdfs dfs -rm -r /dir\_name -> remove file or directory

Note: Removed files/ directories in hadoop will be moved to the trash

first.

Access Trash -> hdfs dfs -ls .Trash

**Permission shell commands**

Follows POSIX (Linux) permission model

***-rw-r--r-- user group filename***

Left to Right -> Owner, Group Level, Everyone(Open)

Change Permission:

Hdfs dfs -chmod [number] /dir\_name

Hdfs dfs -chgrp [group] /dir\_name

Hdfs dfs -chown [owner] /dir\_name

* Need to use sudo (super user)

**-R do an action recursively for all files/ folders**

**Data Manipulation shell commands**

Hdfs dfs -put /tmp/file\_name /user/hdfs/file\_name

Hdfs dfs -get /user/hdfs/file\_name /tmp/file\_name

**user/hdfs/ <--- HDFS File Storage**

**/tmp/ <--- Local File Storage**

Hdfs dfs -movefromlocal /tmp/file\_name /user/hdfs/file\_name

(this will move file from local to Cluster)

-chmod -R (Do Recursively for all files and folders inside that directory)

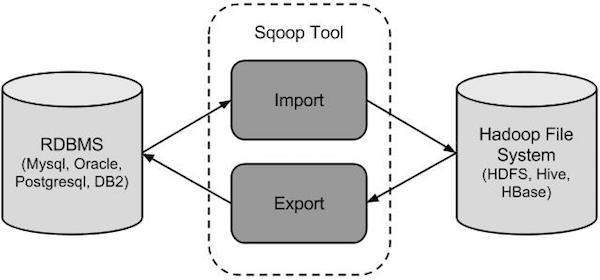
644 - Public Level (-rw-r--r--)

700 - Admin Level (-rwz------)

**Transferring Relational Data to HDFS Using Sqoop**

**What is Sqoop?**

Provide feasible interaction between relational database server and Hadoop’s HDFS.



Sqoop can import data from different data sources.

Sqoop can import and export data real time, while the traditional database operates as usual. Data recorder in:

Text data in text files

Binary Data

Avro and sequence files

Sqoop import tool : Imports individual data tables from RDBMS to HDFS

Sqoop export tool: exports set of files (including rows) back to RDBMS from HDFS

Problems sqoop will resolve:

No need to remove/ move production data from relational databases

Data can be accessed from multiple data warehouses.

(Data no shared)

Automating Workflows

Sqoop docs available at:

<https://sqoop.apache.org/docs/1.99.7/index.html>

[https://cwiki.apache.org/confluence/display/SQOOP/Sqoop+Project+](https://cwiki.apache.org/confluence/display/SQOOP/Sqoop+Project+Overview)

[Overview](https://cwiki.apache.org/confluence/display/SQOOP/Sqoop+Project+Overview)

**Sqoop Commands**

Sqoop is actually a MapReduce program which performs MapReduce actions.

Get list of tables:

Sqoop list-tables --connect jdbc:mysql://(MySQL Address)/dbname

Select a table:

Import data of a table from MySql to HDFS:

Sqoop import --connect jdbc:mysql://(MySQL Address)/dbname

--table tablename -m 1

[-m 1 : No of MapReduce Jobs to be performed]

Import data from a query result:

Sqoop import --connect jdbc:mysql://(MySQL Address)/dbname

--query QUERY (ex: SELECT \* FROM WHERE…)

--target-dir /dir\_name <-- HDFS Target Directory name to import

If this operation is performed:

Operation status (ex: Map imported rows etc.) can be seen under *Map Reduce Framework* Logs output.

**Hive**

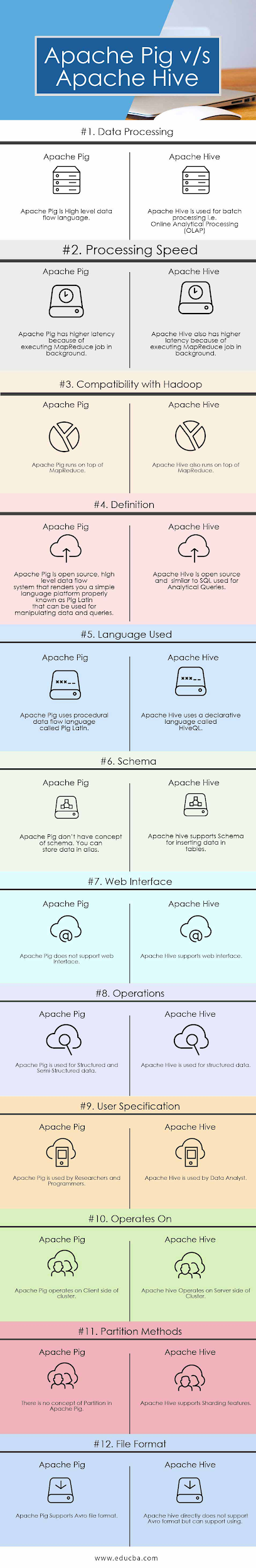
Data warehouse software

Allow developers to structure data into schemas.

Facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.

For querying, summarizing and analyzing large data sets using a SQL-like interface.

* Hive is an ETL tool (Extraction-Transformation-Loading)
* hive is similar to SQL
* Hive enables customized mappers and reducers
* Hive increases the schema design flexibility using data serialization and deserialization
* hive is an analytical tool



Where to use Pig?

* When you are a programmer and know scripting language
* When you don’t want to create schema while loading
* ETL requirements
* When you are working on client side of the Hadoop cluster

Where to use Hive?

* Data warehousing requirements
* Analytical Queries of historical data
* Data Analysis who are familiar with SQL
* While working on structured data

Benefits of Hive

Schema bound

HiveQL (Similar to SQL)

Extendable

Hive commands

Same as SQL commands

hive>dfs -ls /user/ (dfs commands can be also executed in hive)

**Querying HDFS with HiveQL**

Steps:

Build a table in hive

Write HiveQL Query

Extract Results

Build a Table in Hive

Step 1: CREATE TABLE tbName (...) row format delimited

fields terminated by ‘,’

lines terminated by ‘\n’

Skipping header line:

tblproperties (“skip.header.line.count”=”1”);

\*This is useful as many csv files contains of header row

Step 2: LOAD DATA INPATH ‘path\_to\_csv’ OVERWRITE INTO TABLE tbName;

\*This will override if any data in above created table with new data from the csv

**Pig**

Application environment to run Pig Latin scripts

And to convert Pig Latin Scripts to MapReduce jobs

Not-schema bound

Pig Latin Scripts

Handle Unstructured / Semi Structured Data

PIG commands

\*Pig scripts are written and run using interactive grant shell

\*to launch grant shell:

**pig -x local -** to access local files

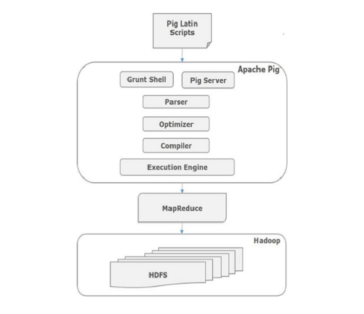
Pig (or pig -x mapreduce) - access hdfs files

A = LOAD ‘file’ USING PigStorage(‘,’);

AS (field1, field2, field3);

Results = FOREACH a GENERATE field1;

DUMP results;



#### 

#### 

#### 

#### **1. Fs: This will list all the file in the HDFS**

grunt> fs –ls

#### **2. Clear: This will clear the interactive Grunt shell.**

grunt> clear

#### **3. History:**

This command shows the commands executed so far.

grunt> history

#### **4. Reading Data: Assuming the data resides in HDFS, and we need to read data to Pig.**

grunt> college\_students = LOAD ‘filename’

USING PigStorage(‘,’)

as ( id:int, firstname:chararray, lastname:chararray, phone:chararray,

city:chararray );

* *PigStorage() is the function that loads and stores data as structured text files.*

#### **5. Storing Data: Store operator is used to storing the processed/loaded data.**

grunt> STORE college\_students INTO ‘ hdfs://localhost:9000/pig\_Output/ ‘ USING PigStorage (‘,’);

* *Here, “/pig\_Output/” is the directory where relation needs to be stored.*

#### **6. Dump Operator: This command is used to display the results on screen. It usually helps in debugging.**

grunt> Dump college\_students;

#### **7. Describe Operator: It helps the programmer to view the schema of the relation.**

grunt> describe college\_students;

#### **8. Explain: This command helps to review the logical, physical and map-reduce execution plans.**

grunt> explain college\_students;

#### **9. Illustrate operator: This gives step-by-step execution of statements in Pig Commands.**

grunt> illustrate college\_students;

**Querying Data with Pig: Steps**

Load Data from HDFS

Write JOINS in Pig Latin

Store Results in HDFS

Pig data types: **No STRING**. **chararray** available instead of String

Integer -> int

Load Data:

*Variable* = Pig Lating Query;

DUMP *Variable*;

Join data tables:

*combined* = table\_name BY col\_name, table\_name BY col\_name;

(col\_name <-- Same column which can be referenced in both data tables)

Store Results to HDFS:

STORE *combined* INTO /dir\_name USING PigStore(‘,’);

\*note: PigStore(‘,’) <-- As a CSV

**Processing Spars Data with Hbase**

Hbase : NoSQL/ Distributed/ Scalable database build on hadoop

Perform real time Read/Write operations in hadoop

Apache OpenSource

Written in Java

2 Run Modes:

Distributed Mode

Standalone mode

Works as a Data Store

-Loosely based way data storing (key-value pairs)

Ex: facebook, fb messenger runs on Hbase

Shell: hbase shell

Shells commands are used to create/modify files

**More about Hbase:**

* HBase is a column-oriented non-relational database management system
* HBase provides a fault-tolerant way of storing sparse data sets, which are common in many big data use cases.
* HBase system is designed to scale linearly.
* Avro, as a component, supports a rich set of primitive data types including: numeric, binary data and strings;

and a number of complex types including arrays, maps, enumerations and records.

**Hbase Commands**

Open hbase CLI

*hbase shell*

Create tables:

hbase> create ‘tablename’ , ‘columnname’

Put rows and columns into table:

hbase> put ‘tablename’ , ‘rowNumber’ , ‘columname’ , ‘value’

To get a data row:

hbase> get ‘tableName’ , ‘rowNumber’

Look at data in a table:

hbase> scan ‘tablename’

Drop a table:

hbase> disable ‘tablename’

hbase> drop ‘tablename’

\*Need to disable the table first, before dropping it

Row identifiers: -> unique identifiers

(can access those rows later with the identifier)

To update a value, use the same **put** command. For the specific rowName and columnName. It will replace its value.

**Moving data from csv file to Hbase**

Pig can be used to write scripts move data from HDFS to Hbase

Pig Scripts

Script\_name.pig

Sample Hbase Script

a = LOAD ‘csv\_file’ USING PigStorage(‘,’) as (colname: datatype, … );

STORE a INTO ‘hbase://table\_name’ USING

org.apache.pig.backend.hadoop.hbase.HBaseStorage(

‘Info:open info:close info:low info:high ); \*info: column\_name

Run the script

pig -f Script\_name.pig

\*This way we can automate adding multiple (thousands/ millions of ) data (rows) to Hbase instantly.

Limit no of results when scanning a table:

Scan ‘table\_name’ , { ‘LIMIT’ => 10 }

* *Note: Hbase keeps a timestamp and a unique identifier (row\_name) for each value we have entered to a data table.*

**Writing bash scripts for ingesting data in HDFS**

Bash: UNIX Scripting language that processes POSIX commands.

Bash: Runs on HDFS cluster

Perform MapReduce Jobs

Sample Bash Script

#!/bin/bash

files=”1 2 3 4 5”

For fileName in $files

do

echo “$fileNAme”

done

**Why we need bash scripting?**

-put command and other commands in HDFS CLI is enough to handle a

single file or a few files.

When need to handle large amount of files..

When tasks need to be automated..

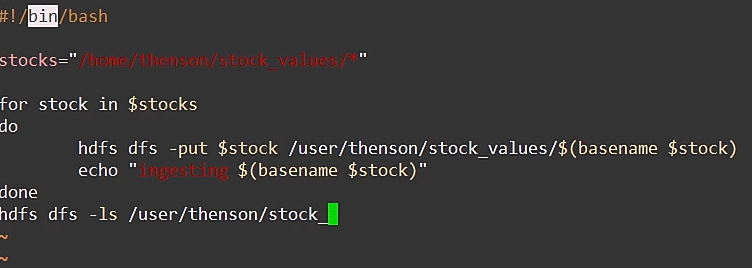
Save time..

Bash scripts: bash.sh (save as .sh)

Then : need to give executable permissions.

Chmod +x hdfs-move.sh

Sample loop to put data recursively to HDFS



**MapReduce**

Coding Steps:

