Big Data – For Analysing huge amount of data

Hadoop – Open Source

Big Data Notes:

tinyurl.com/JPMGlasgowBigData – This link points to amzon drive below..

<https://www.amazon.com/clouddrive/share/oCmO3ovzkRoTQ6iZCwcThWeSSqbiuSckabYr0qxYPQf?_encoding=UTF8&%2AVersion%2A=1&%2Aentries%2A=0&mgh=1>

<https://hub.docker.com/r/joegagliardo/bigdata/>

Trainer Contact Details:

[joey@me.com](mailto:joey@me.com)

[joegagliardo](https://hub.docker.com/r/joegagliardo/bigdata/)

Review - http://surveymonkey.com/r/Z6kF83H

Bookmarks

Save this as bigdata-bookmarks.html

This is the bookmark exported from following file. You can import the same in your browser.



This link has steps for setting up your environment using docker image.

ps – Shows Processess Running

jps - Shows JavaProcessess Running

Split file in chunks of some size

hadoop fs -D dfs.blocksize=1048576 -put ~/roi\_datasets/shakespeare.txt /

Docker Commands

docker ps

docker ps -a (All processes)

docker images

docker rm imagename – This removes the image from the system

docker-compose up – Looks for the docker images at current locations and ???? find tis out

docker-compose down – Brings down the process mentioned in the config file

docker-compose scale datanode=4 – scaling up the nodes to some count

In real life you would ssh to node machine which would be

These are the parsers.

serde – Serializer and Deserializer (Tells you how to read the row from file)

Node is inside cluster.

**Problem and Solution**

Connection error

Node name was not running

format-namenode.sh was run which is inside the script.

Start-dfs.sh

stop-dfs.sh

start-yarn.sh

stop-yarn.sh

1. If you cant start the docker image that mean u earlier removed the image using **docker rm imagename** command so you need to use following command to bring back the image to your system

# docker run --name bigdata-client --hostname bigdata -p 50070:50070 -p 8088:8088 -p 10020:10020 -p 9042:9042 -p 10000:10000 -p 10001:10001 -p 10002:10002 -v "$HOME:/home" -it joegagliardo/bigdata /etc/bootstrap.sh -bash

Hadoop Home page

[http://localhost:50070](http://localhost:50070/dfshealth.html#tab-overview)

Resource Manager

http://localhost:8088

http://oozie.apache.org/ – Workflow scheduler

docker start imagename

docker attach bigdata

ssh

secure shell

when you run the sh file from its location do it like **./**.sh file name

Data Node:

Copied from wiki

A **DataNode** stores data in the [HadoopFileSystem]. A functional filesystem has more than one **DataNode**, with data replicated across them. On startup, a **DataNode** connects to the NameNode; spinning until that service comes up. It then responds to requests from the NameNode for filesystem operations.

Name Node:

HDFS is storage

yarn is processor or an engine. Yarn also has configuration file(This line is captured when the trainer said but have no further information on this).

Sqoop – Moves the data from ANY SQL DATABASE TO DATA FILES

External Tables Oracle Concept where the file is imported into a table for reading. You dont need to have predefined tables for it.

Processing

Mapper - Calculating, Filtering, Mapping..Generally Light weight...

Combiner - For example on the example of Day 4, a note return word along with list of 1s number of times the word has occurred. Combiner basically optimizes this output and changes the output to word and count of that word. This reduces the data that we are passing to shufflers.

Cant start reducing until all Mappers and Shufflers are done.

PARTITIONER – Based on some logic..like alphabets

Shuffler – Moving the data around nodes (Data hashed and moved across)

Joining Grouping Sorting – Reducers (Memory Intensive)

Numebr of output files is equal to number of reducers...

Shufflers makes sure

You can change the shufflelers by replacing the DefaultShufflers to Custom Shufflers.

PROGRAMS THAT SHOULD BE RUNNING:

* ResourceManager
* SecondaryNameNode
* Jps
* NameNode
* DataNode

TOOL DOES THE JOB FOR US… NO NEED TO WRITE JAVA CODE

Spark

SHUT DOWN

stop-dfs.sh

**DAY 2**

**Hive:**

Big Data Technology where you can fetch and write the data using SQL.

When invalid format is present on the file then it inserts NULL value in the table but doesnt give you any error.

//You can change the default execution Engine by following command. You dont need to run this command by default

**set hive.execution.engine=mr;**

**Storage Engine can be configured as well in cassandra or mongo.**

**Cassandra can just do CRUD but no analytics...**

Create table using INSERT Script: Can be used to import sensible data again for example create table MakesSense values select from tableWithNUll values.

Put files here and then when you query the query reads these files.

Copy whatever files you want to load in hive using following command

**hadoop fs -put kahipan.txt /user/hive/warehouse/people**

**dfs** command runs withing hive shell.

/user/hive/warehouse/people

Within shell you can copy the file using following command. So when the table structure is defined in mysql will have the location details.

**LOAD DATA LOCAL INPATH '/examples/northwind/CSV/regions' overwrite into table regions;**

**ANIKET NOTES ON HIVE:**

1. Query is submitted to hive.
2. Hive internally connects to DB to fetch the metadata.
3. This metadata contains instructions to read the hdfs file and parse it.
4. Hive wraps the query in java code.
5. The jar is submitted to name node.

When asked about how to configure base file path for hive from where it would read the files.

Managed table

External Table

Metastore

show tables;

explode – When you have a list of values mapped against a single record.

When you create a table in hive shell, it updates the merastore underlying and updates the sql.

TO setup the database

root@bigdata:/scripts# init-schema-mysql.sh

mysql

SHOW DATABASES;

NORTHWIND Database is an online

<https://github.com/joegagliardo/bigdata-docker>

hive

show tables;

when used in the query gives you the file from which the data is read.

Following command creates a table and populates the data from CSV file. Note that this is not a table but an illusion of table which you can query using sql but it is not a table.

**CREATE EXTERNAL TABLE Regions(**

**RegionID int,**

**RegionName string)**

**ROW FORMAT DELIMITED FIELDS TERMINATED BY ','**

**LOCATION '/northwind/regions';**

This creates a structure where the data is separated by tabs.

create table People(id int, firstname string, lastname string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

AVRO Format:

create table people\_avro(id int, first string) **stored as avro;**

**create table people\_avro(id int, first string) stored as textfile**;

**create table people\_avro(id int, first string)**

**ROW FORMAT SERDE ‘classname’**

**stored as textfile**;

In order to make custom SERDE work you must add the jar file to path…

ADD JAR /usr/local/hive/hcatalog/share/hcatalog/hive-hcatalog-core.jar;

CREATE TABLE territories(

TerritoryID string,

TerritoryName string,

RegionID int)

ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'

STORED AS TEXTFILE

LOCATION '/territories';

Function name used in sql may be case sensitive

Performance Improvements Hive

1. Partitioning Based on Some Properties
2. Use Buckets to store the data

Questions:

Replicating the data across nodes?

Processing based on worked threads?

How long is this suppose to take Mapping/Reducing/Shuffling??

Hive see metadata for a table definition?

Go to mysql and connect the the metaphore

**Unix Commands**

nano to create a file .

cat to read the file

cat filename | grep searchtext

less – to see content of the file

gedit – Ubuntu Notepad

**Cassandra**

Doesn't support group by query.

Mongo

Hbase

Neo4j – Graphical Data Structured

DAY 3

**PIG**

PIG for ETL

Qualitative analytical – Use Hive

PIG Doesnt use Meta Store

Function names are case Sensitive

JsonLoader() to load Json Data

Spark shouldnt be used for ETL.

load ‘filepath’ using JsonLoader()

In PIG you can simply keep creating the new records based on previous.d

Store and Load//Classes in Pig like SerDe

Commands

pig -x local – TO get inside pig shell

regions = load ‘region.csv’ using PigStorage(‘,)

regions2 = foreach regions generate UPPER($0, )

regions = load ‘region.csv’ using PigStorage(‘,) as (id,name)//Gives column name to data

regions = load ‘region.csv’ using PigStorage(‘,) as (id:int,name:chararray)//Giving the datatype

t1 = load 'territories.tsv' using PigStorage('\t') as (territoryId:int, name:chararray, regionId:int);

dump regions

describe regions//describes regions

foreach

join

Flattenning is a conceptsimilar to explode or collapsing

filter RECORD by name == ‘Rakesh’

t3 = filter t2 by territoryId == 1;

group (grouping by some field)

store //You can store the output to any file using store command. You can give separator as well to separate the data.

You can save a script in file and then you can run it as follwos:

pig -x local -f scirpt path

You can run the pig without local for that you would need hdfs running.

You can you pig -f filepath command to get this done.

In Pig {} is not a map/dictionary but a bag of tuples.

Script can look collection of statements

------------------------------------------------------------------------------------------

t1 = load 'territories.tsv' using PigStorage('\t') as (territoryId:int, name:chararray, regionId:int);

t2 = foreach t1 generate UPPER(name);

dump t2;

------------------------------------------------------------------------------------------

SPARK

the main difference between Hadoop and Spark is that Hadoop spends good amount of time in reading the data which is already split and writes it.

Spark keeps it in Memory.

Spark might be more resource intensive and may need more node but can process faster.

spark-submit –class **$fully qualified Main class name** Jar path input output ….

spark-submit –class **com.big.Main** path/jar.jar /shakespseare.txt /OutputFolder ….

Start python by running python then paste following as it is and press enter

--------------------------------------------------------------

import platform

import findspark

findspark.init()

from pyspark import SparkConf, SparkContext

from pyspark.sql import SQLContext

from pyspark.sql.types import \*

conf = SparkConf().setAppName("spark-text").setMaster("local")

sc = SparkContext(conf=conf)

spark = SQLContext(sc)

sc.setLogLevel("ERROR")

--------------------------------------------------------------

This will set up your environment. All the other examples on Internet start after this.

Spark can talk to any RDBMS / NOSQL Database and can write it to any thing in any files…

DAY 4

~/roi\_hadoop\_wordcount/wordcount.sh

Runnin this file will submit the jar file to hadoop and will give store the output file which talks about which word has appeared how many times..

Mapper

-------------------------------------------------------------------

*/\*\**

*\**

*\*/*

package com.roi.hadoop.wordcount;

*/\*\**

*\** ***@author*** *student*

*\**

*\*/*

import java.io.IOException;

import java.util.regex.Pattern;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class WordMapper extends Mapper<Object, Text, Text, IntWritable> {

private final static IntWritable *ONE* = new IntWritable(1);

private final static Pattern *NON\_LETTERS* = Pattern.*compile*("[^a-zA-Z]");

private final static String *SPACE* = " ";

private final static String *SPACES* = " +";

public static String[] findWords(String line) {

line = *NON\_LETTERS*.matcher(line).replaceAll(*SPACE*);

return line.split(*SPACES*);

}

public void map(Object ignoredKey, Text value, Context context)

throws IOException, InterruptedException {

String[] words = *findWords*(value.toString());

for (String word : words) {

context.write(new Text(word), *ONE*);

}

}

}

-------------------------------------------------------------------

*The Mapper class has four generic parameters Key In And Value In*

*Key Out and Value Out.*

*In case you want an output of the mapper multiple fields instead of just IntWritable you can create your own class which would have the fields you want which you would populate in Mapper and that class should implement*

*Writable. This class can further implement Comparable in case you want to use this in the upcoming processes like Reducer.*

*Reducers*

*This code happens after shuffling has happened*

*Sample Data*

*TO – [1,1,1,1,1]*

*BE – [1,3,1,2,1]--It would only be 1 1 1 here because Mapper writes it in that way.*

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class CountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

@Override

public void reduce(Text key, Iterable<IntWritable> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

IntWritable result = new IntWritable(sum);

context.write(key, result);

}

}

*In case you need some paramters to be avaiable for your mapping or processing then you can configure it in Context.*

*Configuration conf = new Configuration();*

*conf.setStrings("searchTerms", searchTerms);*

*You can fetch it back in your Mapper or reducers.*

*context.getConfiguration().getStrings("searchTerms");*

*You get Context in Mapper,Reducer.*

*Spark & Storm:*

*Streaming the file when they arrive example:*

*Followig script is somewhat linked to it.*

[*https://github.com/joegagliardo/bigdata-docker/blob/e9db3833ecfe130606be7954c78ab050dd0a38cd/examples/spark/hdfs-stream-count.py*](https://github.com/joegagliardo/bigdata-docker/blob/e9db3833ecfe130606be7954c78ab050dd0a38cd/examples/spark/hdfs-stream-count.py)

*Reading the file and giving back top 10 words with maximum occurrence*

*You can do Twitter FB Poll all you need is Corresponding StreamReader*

*STORM:*

*Analogy is plumbing.*

*SPOUT*

*BOLT*