Cloud Computing Labs: Big Data Programming

[redacted] [redacted]

1.1: Wordcount – Hadoop

Command:

>hadoop jar hadoop/3.1.2/libexec/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.1.2-sources.jar org.apache.hadoop.examples.WordCount wordcount/input wordcount/output

>hadoop fs -cat wordcount/output/part-r-00000

Load file into hdfs:

```
tin-java classes where applicable
(base) [redacted]-MacBook-Pro:Cellar [redacted]$ hdfs dfs -put hadoop/hdfs/Pyspark/wordcount_file1 wordcount/input
2019-10-07 08:24:51,873 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
(base) Vaishnavis-MacBook-Pro:Cellar vaishnaviv$ hdfs dfs -put hadoop/hdfs/Pyspark/wordcount_file2 wordcount/input
2019-10-07 08:24:59,547 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
(base) Vaishnavis-MacBook-Pro:Cellar vaishnaviv$
```

Execution:

```
(base) [redacted]-MacBook-Pro:Cellar [redacted]$ hadoop jar hadoop/3.1.2/libexec/share/hadoop/mapreduce/sources/hadoop-m
apreduce-examples-3.1.2-sources.jar  org.apache.hadoop.examples.WordCount wordcount/input wordcount/output
2019-10-07 08:25:36,843 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
```

```
(base) [redacted]-MacBook-Pro:Cellar [redacted]$ hadoop fs -cat wordcount/output/part-r-00000
2019-10-07 08:27:53,552 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
1       1
This    2
another 1
count   2
file    2
is      2
part    1
program 1
word    2
(base) [redacted]$ hadoop fs -cat wordcount/output/part-r-00000|less
```

```
2019-10-07 08:27:04,644 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
1       1
This    2
another 1
count   2
file    2
is      2
part    1
program 1
word    2
(END)
```

1.2:Wordcount -pyspark

>spark-submit apache-spark/2.4.4/libexec/examples/src/main/python/wordcount.py wordcount/input

```
word    2
(base) [redacted]$ spark-submit apache-spark/2.4.4/libexec/examples/src/main/python/wordco
unt.py wordcount/input
2019-10-07 08:35:30,414 WARN util.Utils: Your hostname, [redacted]-Pro.local resolves to a loopback address: 127
.0.0.1; using 192.168.0.19 instead (on interface en0)
2019-10-07 08:35:39,073 INFO scheduler.DAGScheduler: Job 1 finished: collect at /usr/local/Cellar/apache-spark/2.4.4/lib
exec/examples/src/main/python/wordcount.py:41, took 0.020830 s
is: 2
count: 2
: 1
1: 1
This: 2
another: 1
word: 2
program: 1
file: 2
part: 1
```

```
(base) [redacted] hdfs dfs -cat wordcount/pyoutput/part-00000 wordcount/pyoutput/part-000
01
2019-10-07 08:47:42,001 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
('is', 2)
('count', 2)
('', 1)
('1', 1)
('This', 2)
('another', 1)
('word', 2)
('program', 1)
('file', 2)
('part', 1)
```

Part 2: MapReduce K-Means Algorithm
2.1: Yes,Able to run the revised code successfully.

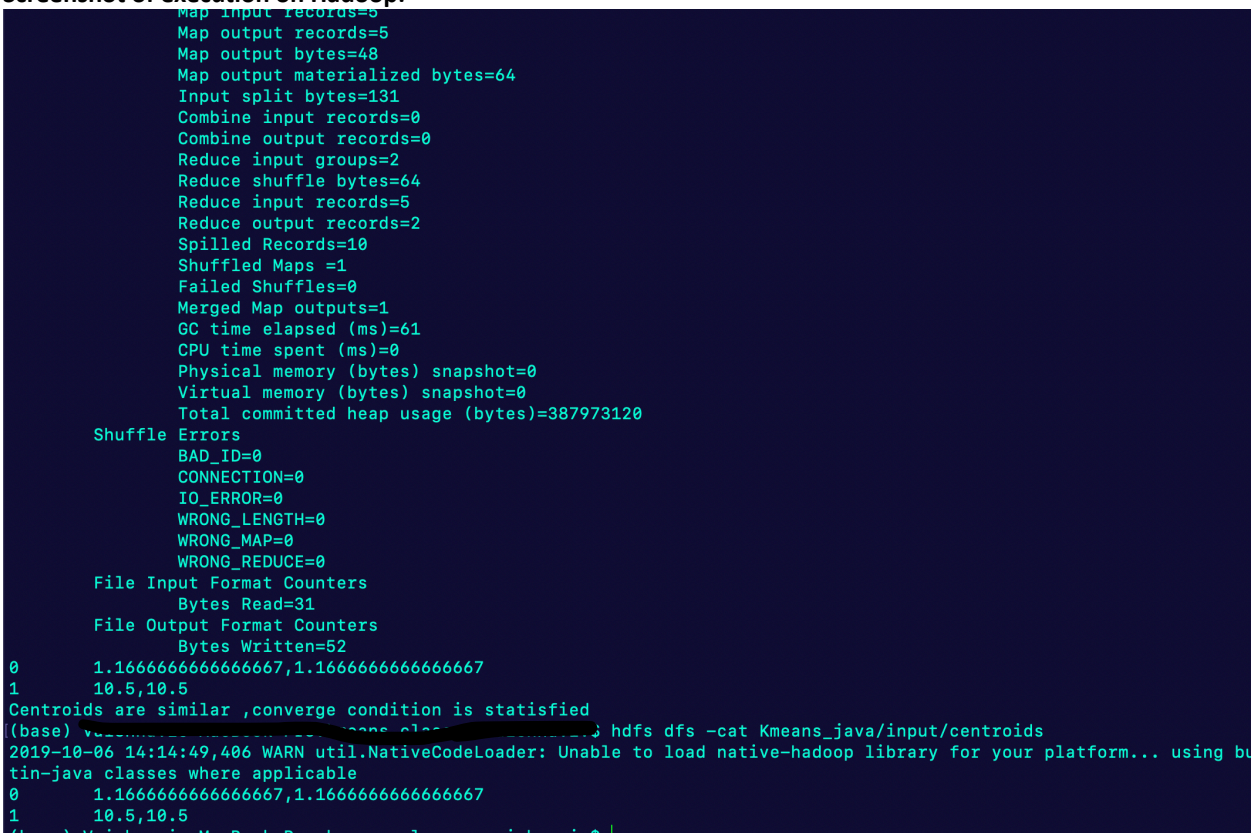
screenshot and code snippet with explanation:
The code below executes for a maximum of 15 iterations or until converged i.e. old centroids and new centroids differ by a given threshold of 0.001. On convergence, the new centroids are stored in the centroids file as output.

Code snippet:

```
public static void main(String[] args) throws Exception {
   // usage: hadoop jar km.jar hdfs://localhost:9000/user/your_home_directory/centroids data.hdfs output
   Configuration conf = new Configuration();
   int niteration=15;
   double threshold=0.001;
   double [][] _oldcentroids;
   FileSystem hdfs=FileSystem.get(conf);
/*Iterating for a maximum of 15 iteration if convergence is not reached/*
   for (int i =0 ;i<niteration;i++)
   {
          System.out.println("startiteration"+String.valueOf(i));
          conf.set("Centroids-file", "centroid");
     Job job = Job.getInstance(conf, "KMeans");
     job.setJarByClass(KMeans.class);
     job.setMapperClass(KMMapper.class);
     job.setReducerClass(KMReducer.class);
     job.setOutputKeyClass(IntWritable.class);
     job.setOutputValueClass(Text.class);
     FileInputFormat.addInputPath(job, new Path("data.hdfs"));
     FileOutputFormat.setOutputPath(job, new Path("output"));
/*load old centroids*/
     _oldcentroids=loadCentroids(conf.get("Centroids-file"), conf);
 /*wait till the job is completed*/
          job.waitForCompletion(true);
/*delete old centroid file*/
          hdfs.delete(new Path("centroid"),true);
/*copy obtained output to centroid file*/
          FileUtil.copy(hdfs,new Path("output"+File.separator+"part-r-00000"),hdfs,new
Path("centroid"),true,true,conf);
/*Delete the output file to avoid file already exist error*/
          hdfs.delete(new Path("output"),true);
/*Load new centroids from centroid file*/
          double[][] _newcentroids=loadCentroids(conf.get("Centroids-file"), conf);
/*call to converge function */
```

```
Boolean convergecondition=converge(_oldcentroids,_newcentroids,threshold);
if (convergecondition)
        System.out.println("Centroids are similar ,converge condition is statisfied");
        break;
```

Screenshot of execution on Hadoop:



2.2 :

Combiner class perform local reduce operation on map output before actually sending it to Reduce function. It accepts output from Map class and perform semi-reducer action and send key-value pair to Reducer class.
Given: N records,k clusters,M mappers,R reducers
Big – O representation:
Total data passed to reducer without combiner = O(N)
Total data passed to reducer with combiner=O(kM)

2.3: using Hadoop streaming:
Command:
>hadoop jar /usr/localCellar/hadoop/3.1.2/libexec/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -input Kmeans_python/input/data.hdfs -output Kmeans_python/output -mapper "python mapper.py" -file mapper.py

Code snippet:

mapper.py

```python
import sys
import numpy as np
import pydoop.hdfs as hdfs


def read_inputfile(filename):
    f=hdfs.open(filename, 'rt')
    data=f.read()
    return data


def data_formatting(data):
    values_array=[]
    id_array=[]

    for lines in data:
        #here id = cid and datarecord id and values is centroid values and data values
        id,values=lines.split("\t")
        #_x and _y centroids and data values x and y coordinates
        _x,_y=values.split(",")
        values_array.append([float(_x),float(_y)])
        id_array.append(int(id))
    return np.array(id_array),np.array(values_array)

def mapper(x,y):
    #read centroid file
    centroid_data = read_inputfile("Kmeans_python/input/centroids")
    cid_array,centroid_array=data_formatting(centroid_data.split("\r\n"))
    #print(cid_array,centroid_array)
    #print("data",[x,y])

    distance_calculation(centroid_array,[x,y])

#calcualting euclidean distance
def distance_calculation(array1,array2):
        #caluclating eucliden distance of every data record with all centroids and taking the centroid id with minimum distance
        distances=np.linalg.norm(array2-array1,axis=1)
        #assigning cluster label to each recored based on the euclidean distance
        clusterid=np.argmin(distances)
        #write to standard output in expected format which will be passed to reducer to write clutser label and associaated data file after mapreduce
        print(clusterid,"\t",",".join(list(map(str,array2))))

if __name__ == "__main__":
    #read datafile
    for data in sys.stdin:
        #split data.hdfs
        recordvalue_x,recordvalue_y=data.split(",")
        mapper(float(recordvalue_x),float(recordvalue_y))
```

screenshot:

```
(base) [redacted] hadoop jar /usr/localCellar/hadoop/3.1.2/libexec/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -input Kmeans_python/input/data.hdfs -output Kmeans_python/output -mapper "/Users/vaishnaviv/anaconda3/bin/python mapper.py" -file mapper.py
JAR does not exist or is not a normal file: /usr/localCellar/hadoop/3.1.2/libexec/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar
(base) [redacted] hadoop jar /usr/local/Cellar/hadoop/3.1.2/libexec/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -input Kmeans_python/input/data.hdfs -output Kmeans_python/output -mapper "/Users/[redacted]/anaconda3/bin/python mapper.py" -file mapper.py
2019-10-06 14:54:22,594 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead
2019-10-06 14:54:22,679 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [mapper.py, /var/folders/pk/7sgfwrwn2dd6zwk6gjv0vq440000gn/T/hadoop-unjar6507352312646897942/] [] /var/folders/pk/7sgfwrwn2dd6zwk6gjv0vq440000gn/T/streamjob5907026467113350458.jar tmpDir=null
2019-10-06 14:54:23,289 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2019-10-06 14:54:23,429 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2019-10-06 14:54:23,725 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/vaishnaviv/.staging/job_1570385095751_0006
2019-10-06 14:54:23,904 INFO mapred.FileInputFormat: Total input files to process : 1
2019-10-06 14:54:23,936 INFO mapreduce.JobSubmitter: number of splits:2
2019-10-06 14:54:24,032 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1570385095751_0006
2019-10-06 14:54:24,033 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-10-06 14:54:24,167 INFO conf.Configuration: resource-types.xml not found
2019-10-06 14:54:24,167 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2019-10-06 14:54:24,216 INFO impl.YarnClientImpl: Submitted application application_1570385095751_0006
2019-10-06 14:54:24,247 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1570385095751_0006/
2019-10-06 14:54:24,248 INFO mapreduce.Job: Running job: job_1570385095751_0006
2019-10-06 14:54:30,334 INFO mapreduce.Job: Job job_1570385095751_0006 running in uber mode : false
2019-10-06 14:54:30,335 INFO mapreduce.Job:  map 0% reduce 0%
2019-10-06 14:54:37,409 INFO mapreduce.Job:  map 100% reduce 0%
2019-10-06 14:54:42,446 INFO mapreduce.Job:  map 100% reduce 100%
2019-10-06 14:54:42,455 INFO mapreduce.Job: Job job_1570385095751_0006 completed successfully
2019-10-06 14:54:42,521 INFO mapreduce.Job: Counters: 49
```

output:

```
(base)  ...  ook-Pro:CC_hadoopstreaming  ...$ hdfs dfs -cat Kmeans_python/output/part-00000
2019-10-06 15:03:45,879 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
0       0.5,0.5
0       1.0,1.0
1       2.0,2.0
1       11.0,11.0
1       10.0,10.0
```

3.1: spark-submit pyspark_transormation_Action.py Or python pyspark_transormation_Action.py
code snippet with comments:

```
1  from pyspark import SparkContext
2  from operator import add
3
4
5  #Question 3.1 How much did each seller earn? Develop the Spark RDD
6  # (Spark RDD means using only the transformations and actions, no DataFrame/SQL)
       version to solve the query.
7
8
9  #Creates sparkContext
10 sc=SparkContext(appName="seller_earn_RDD_transformation_actions")
11 #Load purchase file using sc context
12 purchaseddata=sc.textFile("Pyspark/purchase")
13
```

```
14 # get each record from purchase file
15 data_splitbylines=purchaseddata.flatMap(lambda i:i.split("\n"))
16
17 # get each column from a row in the purchases data
18 data_splitbytab=data_splitbylines.map(lambda  j:j.split("\t"))
19
```

```
0  #seller and price is passed as key value pair to reduce function with add operation
1  # compute the income of each seller
2  result_seller_earned=data_splitbytab.map(lambda z:(z[3],int(z[4]))).reduceByKey(add)
3
4  print("======" * 4)
5  #prints sales of each seller
6  print("Each seller earns {}".format(result_seller_earned.collect()))
7
8  print("======" * 4)
```

Result:

```
(base) [redacted] MacBook-Pro:Pyspark [redacted] spark-submit pyspark_transormation_Action.py
2019-10-06 15:25:22,851 WARN util.Utils: Your hostname, Vaishnavis-MacBook-Pro.local resolves to a loopback address: 127
.0.0.1; using 192.168.0.19 instead (on interface en0)
2019-10-06 15:25:22,851 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
2019-10-06 15:25:23,129 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
2019-10-06 15:25:23,672 INFO spark.SparkContext: Running Spark version 2.4.4
2019-10-06 15:25:23,689 INFO spark.SparkContext: Submitted application: seller_earn_RDD_transformation_actions
2019-10-06 15:25:23,730 INFO spark.SecurityManager: Changing view acls to: vaishnaviv
2019-10-06 15:25:23,730 INFO spark.SecurityManager: Changing modify acls to: vaishnaviv
(base) [redacted] -Pro:Pyspark [redacted] python pyspark_transormation_Action.py
2019-10-06 15:24:18,305 WARN util.Utils: Your hostname, Vaishnavis-MacBook-Pro.local resolves to a loopback address: 127
.0.0.1; using 192.168.0.19 instead (on interface en0)
2019-10-06 15:24:18,305 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
2019-10-06 15:24:18,580 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2019-10-06 15:24:19,307 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Each seller earns [('Amazon', 284), ('Barnes Noble', 55), ('Borders', 90)]
```

3.2: command: **> spark-submit Pyspark_SQL.py Or >python Pyspark_SQL.py**

Code snippet with comments:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *


#creata a spark session to use SQL query procession
spark=SparkSession.builder.appName("Pysparl_SQLDemo").master("local").getOrCreate()

#loads and stores each column of purchase file
purchases=spark.sparkContext.textFile("Pyspark/purchase").flatMap(lambda
    i:i.split("\n")).map(lambda  j:j.split("\t"))

#Define Schema with column names and its type for purchase data
p_schema=StructType([
    StructField("year",StringType()),
    StructField("cid",StringType()),
    StructField("isbn",StringType()),
    StructField("seller",StringType()),
    StructField("price",StringType())])

#loads and store each column of book file
book=spark.sparkContext.textFile("Pyspark/book").flatMap(lambda
    i:i.split("\n")).map(lambda  j:j.split("\t"))

#Schema for isbn and names of book and its associated type'
b_schema=StructType([
    StructField("isbn",StringType()),
    StructField("name",StringType())])


#Creates temporary table view for purchase_df and book_df dataframe
purchase_df.createOrReplaceTempView("purchase_table")
book_df.createOrReplaceTempView("book_table")
```

```
#SQL query  for fetching details of book name sold by amazon for lowest price'
result=spark.sql("select name from book_table where isbn IN (select
    purchase_table.isbn from  purchase_table INNER JOIN (select MIN(CAST(price AS
    int)) as Min_price from purchase_table group by isbn) resultT on
    purchase_table.price=resultT.Min_price and purchase_table.seller='Amazon')
    MINUS select name from book_table where isbn IN (select purchase_table.isbn
    from  purchase_table INNER JOIN (select MIN(CAST(price AS int)) as Min_price
    from purchase_table group by isbn) resultT on
    purchase_table.price=resultT.Min_price and purchase_table.seller!='Amazon')")

print("The names of the books that Amazon gives the lowest price among all
    sellers\n")
#prints result to console'
result.show()
```

Result:

```
(base) [redacted] spark-submit Pyspark_SQL.py
2019-10-06 15:31:13,893 WARN util.Utils: Your hostname, [redacted] resolves to a loopback address: 127
.0.0.1; using 192.168.0.19 instead (on interface en0)
2019-10-06 15:31:13,894 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
2019-10-06 15:31:14,167 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
tin-java classes where applicable
(base) [redacted] python Pyspark_SQL.py
2019-10-06 15:31:45,212 WARN util.Utils: Your hostname, [redacted] local resolves to a loopback address: 127
.0.0.1; using 192.168.0.19 instead (on interface en0)
2019-10-06 15:31:45,212 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
The names of the books that Amazon gives the lowest price among all sellers

+------+
| name|
+------+
|Drama|
+------+
```