CISC5950 – Big Data Programming – Spring 2022

Professor: Ying Mao
Project 1 Part 2: NBA Shot Statistics
Extended due date: 4/5/2022
Students: Yuliya Akchurina

MapReduce Data Analysis in Apache Hadoop
NBA Shot Statistics

Overview

This project will demonstrate several design examples to analyze data utilizing the MapReduce programming model running on a three node Hadoop Cluster hosted in Google Cloud. We will analyze a dataset of NBA Shot Logs for the 2014-2015 season. This data can be obtained on the Kaggle site. The analysis will endeavor to generate metrics to gauge player performance with respect to shots taken during the games.

Specifically, we will attempt to answer the following questions:

1. For each pair of the players (A, B), we define the fear sore of A when facing B is the hit rate, such that B is the closest defender when A is shooting. Based on the fear score, for each player, please find out who is his "most unwanted defender".

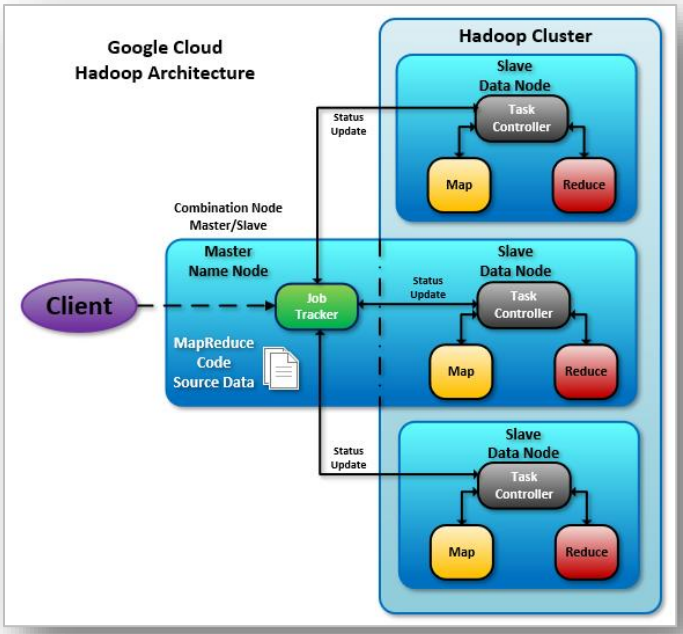
2. For each player, we define the comfortable zone of shooting is a matrix of,
   {SHOT DIST, CLOSE DEF DIST, SHOT CLOCK}
   Develop a MapReduce-based algorithm to classify each player's records into 4 comfortable zones.
   Considering the hit rate, which zone is the best for:
   - James Harden
   - Chris Paul
   - Stephen Curry
   - Lebron James

MapReduce Cloud Architecture

Apache Hadoop software is an open-source framework that allows for the distributed storage and processing of large datasets across clusters of computers using simple programming models. Hadoop is designed to scale up from a single computer to thousands of clustered computers, with each machine offering local computation and storage. In this way, Hadoop can efficiently store and process large datasets ranging in size from gigabytes to petabytes of data.

The Hadoop cluster architecture hosted in Google Cloud consists of the following components:

- 1 Combination Master-Slave Node = **Name node/Data node**
- 2 Slave Nodes = **Data Nodes**



The solution source code, developed to resolve the answers to the posed questions and the source data file, are stored and executed on the Name node. The Job Tracker is responsible for execution of the MapReduce job submitted.

The MapReduce process goes through four phases of execution:

Input Splits:

An input to a MapReduce in Big Data job is divided into fixed-size pieces called input splits Input split is a chunk of the input that is consumed by a single map

Mapping

This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values.

Shuffling

This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output.

Reducing

In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. This phase aggregates the complete dataset.

Dataset:

The snapshot of the dataset below shows several records, where each row represents a shot and the columns are the details of the shot, e.g. the game ID, who is the defender, what is the distance between them.

GAME_ID	MATCHUP	GAME_CL...	SHOT_CLO...	TOUCH_TI...	SHOT_DIST	PTS_TYPE	SHOT_RES...	CLOSEST_...
21400899	MAR 04, 2015 - CHA @ BKN	1:09	10.8	1.9	7.7	2	made	Anderson, Alan
21400899	MAR 04, 2015 - CHA @ BKN	0:14	3.4	0.8	28.2	3	missed	Bogdanovic, Bojan
21400899	MAR 04, 2015 - CHA @ BKN	0:00		2.7	10.1	2	missed	Bogdanovic, Bojan
21400899	MAR 04, 2015 - CHA @ BKN	11:47	10.3	1.9	17.2	2	missed	Brown, Markel
21400899	MAR 04, 2015 - CHA @ BKN	10:34	10.9	2.7	3.7	2	missed	Young, Thaddeus
21400899	MAR 04, 2015 - CHA @ BKN	8:15	9.1	4.4	18.4	2	missed	Williams, Deron
21400899	MAR 04, 2015 - CHA @ BKN	10:15	14.5	9.0	20.7	2	missed	Jack, Jarrett
21400899	MAR 04, 2015 - CHA @ BKN	8:00	3.4	2.5	3.5	2	made	Plumlee, Mason
21400899	MAR 04, 2015 - CHA @ BKN	5:14	12.4	0.8	24.6	3	missed	Morris, Darius
21400890	MAR 03, 2015 - CHA vs. LAL	11:32	17.4	1.1	22.4	3	missed	Ellington, Wayne

Figure 1: snapshot of the data

The source data format is a csv file with 21 columns and 128,070 rows and is 16.42MB in size.

Only selected relevant columns will be used in the computations to answer the 2 questions. The dataset csv file was uploaded to the Google Cloud Cluster and stored on the Name node in:

/mapreduce-test/mapreduce-test-python/project1/shot_logs.csv

Solutions

To successfully run this code on the Google cloud cluster nodes, Python3 is required to support the string operator function that is not available on Python2.

Solution Question 1

For each pair of the players (A, B), we define the fear sore of A when facing B is the hit rate, such that B is closet defender when A is shooting. Based on the fear sore, for each player, please find out who is his "most unwanted defender".

To answer this question, we are using the following columns from the NBA dataset:

- player name – column 19
- closest defender – column 14
- shot result – column 13

We are going to calculate the fear score for each defender based on the player's miss rate. For each pair of player A, defender B the defender that has the highest player A miss rate is the most unwanted defender for the player A. The shot result is encoded into 1 if "made" and 0 if "missed". The miss rate for a player is calculated by the following formula

((count of shot result) – (sum of shot result)) / (count of shot result)

Task Scheduling Control

Submission of the MapReduce code is controlled by a bash script run file named **test.sh**. This script coordinates the submission of the map and reduce jobs along with the source data file to the Hadoop Streamer and HDFS. The run file used for this solution is uploaded and stored on the name node in:

/mapreduce-test/mapreduce-test-python/Part2Q1/test.sh

Mapping

The mapper code is uploaded and stored in:

/mapreduce-test/mapreduce-test-python/Part2Q1/mapper.py

mapper.py reads csv data file as standard input stream, line-by-line, and performs the following steps for each line:

- Strips whitespace
- Utilizes regular expression to split each row by the comma, ignoring the commas that are located within the quotes. This dataset has different formats for player name and defender name. The player name is written as first name and last name, while the defender name is written as "Last name, First name" (see Figure 2).
  One of the defenders in the dataset has only one name "Nene". This presents an issue when splitting the rows by comma using line.split(',') command. Which is changing the column numbers for the subsequent columns in the row with Nene as defender. The use of regular expression (see Figure 3) helps to preserve the column index values the same along all the rows in this dataset.
- Checks that the row has existing column 19 – "Player name". Extracts data from columns "player name" – 19, "defender name" – 14, and "shot result" - 13. Strips the left and right quotes from the defender name.
- Rearranges defender name from "Last name, First name" to "First name Last name"
- Checks that the values of all 3 columns are not empty. Encodes the shot result made to 1, missed to 0.
- Prints out player name, defender name and count of 1 for each occurrence as standard output.
  Key: player name, defender name. Value: 1.

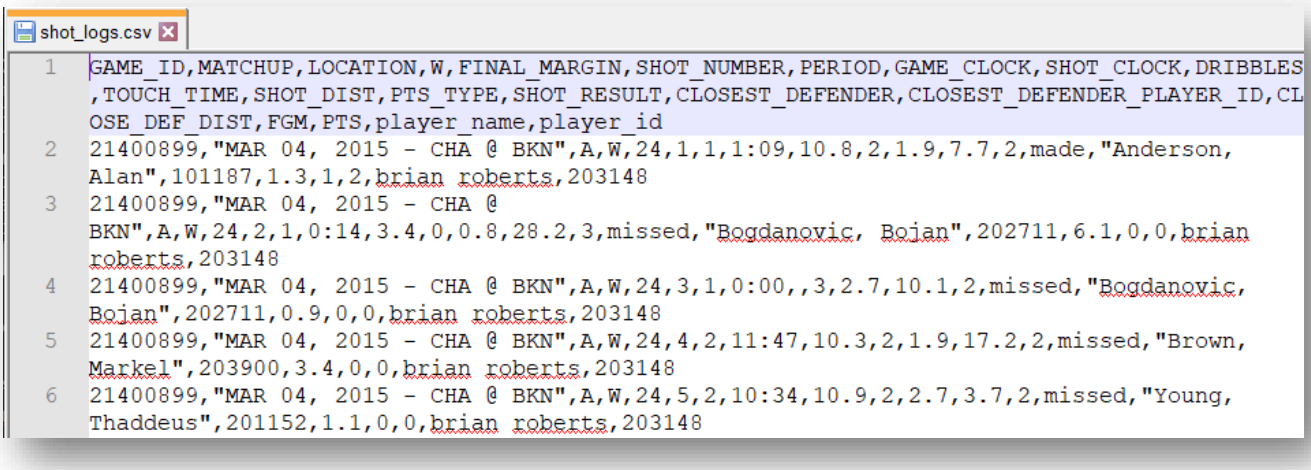


Figure 2: Source data depicting different naming conventions for shooters and defenders

```
for line in sys.stdin:
    line = line.strip()
    line_list = re.split(r',(?=(?:"[^"]*?(?: [^"]*)*))|,(?=[^",]+(?:,|$))', line)
```

Figure 3: Regular expression preserves column index values

Shuffling & Sorting

Upon completion of the mapper process, Hadoop initiates an intermediate process to receive the mapper output, sort generate a key-value list and sort the list. Data is then handed off to the reducer.

Reducing

The reducer code is uploaded and stored in:

/mapreduce-test/mapreduce-test-python/Part2Q1/reducer.py

The reducer receives the streaming results of the mapper via standard input stream, and performs the following:

- Strips whitespace
- Splits the passed-on key and value on a tab into two variables player_defender and count.
- Separates player and defender into two variables
- First the reducer aggregates the data into a dictionary. The key is a unique pair of player A, defender B. The value is a list containing sum of shot result and count of shot result.
- Second the reducer calculates the miss rate for each unique player A defender B pair using the formula

  ((count of shot result) – (sum of shot result)) / (count of shot result)

  The result is added to a new dictionary where:

  - key = player A
  - value = nested list [miss_rate1, defender1], [miss_rate2, defender2], etc.

- The resulting player dictionary is sorted by key in alphabetical order and by value of the miss rate in descending order. The print output is the corresponding defender name for the highest miss rate for each player.
- The result is a standard output player name, defender name.

Results Question 1

The results of the MapReduce process is a list of hundreds of shooters (Player A) with **the most unwanted defender** (Player B). For the sake of brevity and proof of execution, only the head and tail of the output is depicted below.

```
2022-04-05 04:41:25,032 INFO streaming.StreamJob: Output directory: /Part2Q1/o
utput/
aaron brooks    Zach LaVine
aaron gordon    Zach Randolph
al farouq aminu Zaza Pachulia
al horford      Zaza Pachulia
al jefferson    Zaza Pachulia
alan anderson   Will Barton
alan crabbe     Tony Snell
alex len        Zaza Pachulia
alexis ajinca   Xavier Henry
alonzo gee      Zach LaVine
amare stoudemire        Zaza Pachulia
amir johnson    Zaza Pachulia
andre drummond  Zaza Pachulia
andre iguodala  Wilson Chandler
andre miller    Wilson Chandler
andre roberson  Wilson Chandler
andrew bogut    Zaza Pachulia
andrew wiggins  Zach Randolph
anthony bennett Zaza Pachulia
anthony davis   Zach Randolph
anthony morrow  Will Barton
aron baynes     Tyson Chandler
arron afflalo   Zach LaVine
avery bradley   Zach LaVine
ben gordon      Tyler Hansbrough
ben mclemore    Zaza Pachulia
beno urdih      Zach LaVine
bismack biyombo Udonis Haslem
blake griffin   Zaza Pachulia
bojan bogdanovic        Wilson Chandler
boris diaw      Zach Randolph
bradley beal    Zach LaVine
brandon bass    Wilson Chandler
brandon jennings        Willie Green
brandon knight  Zach LaVine
brian roberts   Zach LaVine
brook lopez     Zaza Pachulia
carl landry     Zaza Pachulia
carlos boozer   Zach Randolph
carmelo anthony Zach LaVine
caron butler    Wilson Chandler
chandler parsons        Zach Randolph
channing frye   Zach Randolph
charlie villanueva      Xavier Henry
chase budinger  Zaza Pachulia
chris andersen  Wayne Ellington
chris bosh      Zaza Pachulia
chris copeland  Zaza Pachulia
chris kaman     Zaza Pachulia
chris paul      Zaza Pachulia
cj mccollum     Wesley Johnson
cj miles        Wilson Chandler
cj watson       Zach LaVine
cody zeller     Zach Randolph
cole aldrich    Zaza Pachulia
cory joseph     Zach LaVine
courtney lee    Wilson Chandler
damian lillard  Zach Randolph
damjan rudez    Willie Green
danilo gallinai Wesley Johnson
danny green     Zach Randolph
```

Figure 2: Head of output stream

Output continued on next page…

```
rudy gay        Zaza Pachulia
rudy gobert     Zaza Pachulia
russell westbrook       Zaza Pachulia
ryan anderson   Zach Randolph
serge ibaka     Zaza Pachulia
shabazz muhammad        Zach Randolph
shabazz napier  Zaza Pachulia
shane larkin    Wilson Chandler
shaun livingston        Zaza Pachulia
shawn marion    Zaza Pachulia
shawne williams Tyler Hansbrough
solomon hill    Wesley Matthews
spencer hawes   Zach Randolph
stephen curry   Zach LaVine
steve adams     Zach Randolph
steve blake     Tyler Hansbrough
taj gibson      Zaza Pachulia
terrence ross   Zach Randolph
thabo sefolosha Vince Carter
thaddeus young  Zaza Pachulia
tim duncan      Zaza Pachulia
time hardaway jr        Zaza Pachulia
timofey mozgov  Tyler Zeller
tobias harris   Zaza Pachulia
tony allen      Xavier Henry
tony parker     Zach Randolph
tony snell      Wesley Johnson
travis wear     Zaza Pachulia
trevor ariza    Zach Randolph
trevor booker   Wesley Matthews
trey burke      Zach LaVine
tristan thompson        Zaza Pachulia
ty lawson       Zaza Pachulia
tyler hansbrough        Zach Randolph
tyler zeller    Zach LaVine
tyreke evans    Zach Randolph
tyson chandler  Zaza Pachulia
udonis haslem   Zaza Pachulia
victor oladipo  Zach Randolph
vince carter    Wesley Matthews
wayne ellington Wilson Chandler
wesley johnson  Zach Randolph
wesley matthews Zach Randolph
wilson chandler Wesley Matthews
zach lavine     Wesley Johnson
zach randolph   Wesley Johnson
zaza pachulia   Wesley Matthews
Deleted /Part2Q1/input
Deleted /Part2Q1/output
Stopping namenodes on [instance-1.c.project5950.internal]
Stopping datanodes
Stopping secondary namenodes [instance-1]
Stopping nodemanagers
10.142.0.3: WARNING: nodemanager did not stop gracefully after 5 seconds: Try
ng to kill with kill -9
```

Figure 3: tail of output stream

Solution Question 2

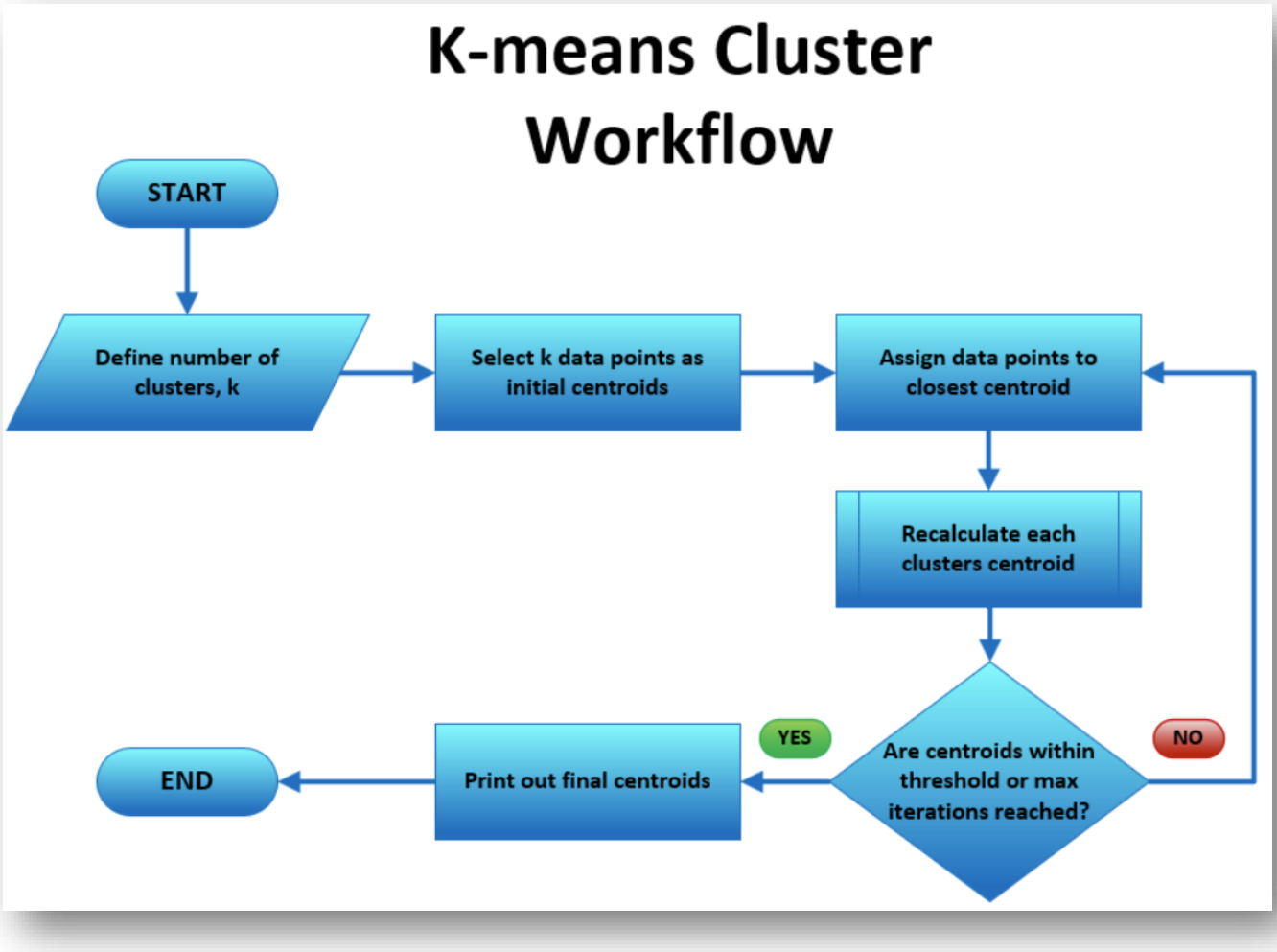
For each player, we define the comfortable zone of shooting is a matrix of,

{SHOT DIST, CLOSE DEF DIST, SHOT CLOCK}

Please develop a MapReduce-based algorithm to classify each player's records into 4 comfortable zones. Considering the hit rate, which zone is the best for James Harden, Chris Paul, Stephen Curry and Lebron James.

This problem requires implementing of K-Means clustering algorithm on Hadoop MapReduce. K-means is a clustering method that aims to partition N given data points into K clusters. The clusters are mutually exclusive, each data point belongs to a single cluster. Each data point is assigned to a cluster based on the shortest distance to the cluster center (centroid). The K-Means algorithm is iterative. It requires two steps. Assume initial centroids are provided. Step one assigns the data points to the nearest cluster based on the least distance to the cluster's center. Step two recalculates the cluster centers as an average of the data points belonging to the cluster.

Steps one and two are repeated until the stop condition is reached. There are two stop conditions. One is the threshold condition - the change between the old and new cluster centroids is less than 0.05. Two is the max iteration condition – the maximum number of iterations is set to a number M. We tried with M = 10, 15, 20, and 30. In general the threshold must be a very small number (0.01 or smaller depending on the data) and the number of iterations should a large enough number (100 or higher) to ensure the K-Means algorithm converges. Also, selection of the initial centroid plays important role in the speed of the convergence.



Task Scheduling Control

Submission of the MapReduce code is controlled by a bash script run file named **test.sh**. This script coordinates the submission of the map and reduce jobs along with the source data file to the Hadoop Streamer and HDFS. The run file used for this solution is uploaded and stored on the name node in:

/mapreduce-test/mapreduce-test-python/Part2Q2/test.sh

Mapping overview

To solve this problem, we are using five columns from the dataset:

- **"SHOT_CLOCK"**
- **"SHOT_DIST"**
- **"CLOSE_DEF_DIST"**
- **"SHOT_RESULT"**
- **"player_name"**

The columns of {**SHOT DIST, CLOSE DEF DIST, SHOT CLOCK}** from the dataset are processed as **{x, y, z}** coordinates of the 3-dimensional dataset.

We use the three pairs of mapper and reducer to solve this problem. Test.sh code coordinates the run the mapper and reducer code pairs, performs iteration, checks for stopping condition, updates clusters.txt file containing the clusters coordinates.

The data points {x = SHOT DIST, y = CLOSE DEF DIST, z = SHOT CLOCK} for the four players james harden, chris paul, stephen curry, lebron james are combined to create the data points used for the K means clustering. We are splitting these data points into four clusters and then we are calculating the hit rate for each of the players based on cluster to determine the best zone for each player.

Mapper 1

The first pair of the mapper and reducer initializes the cluster centers to be a random set of 4 points from the dataset. It takes one data point from each player.

The mapper is located at **/mapreduce-test/mapreduce-test-python/Part2Q2/mapper_init_centroids.py**

The mapper takes csv data file as standard input.

- Extracts the four columns: shot clock, shot distance closest defender distance, and player name, using regular expression.
- Checks that the columns do not have non-zero values.
- Prints out the following format as standard output: Key is player_name, Value is shot_dist, defender_dist, shot_clock.

Reducer 1

The reducer is located at **/mapreduce-test/mapreduce-test-python/Part2Q2/reducer_init_centroids.py**

- Reducer takes mapper's output as standard input.
- Splits player_name and x, y, z coordinates into separate variables
- Creates a dictionary for each player with the Key: x, y, z, and Value: player name.
- Chooses a random sample of the key from each dictionary to be used as a centroid.
- The output is the coordinates for the four centroids, each centroid printed on the new line.

Test.sh takes the output of the reducer_init_centroids and saves it to a file called centrois.txt which is then used in the iteration of Mapper/Reducer for K-Means algorithm. After the first mapper and reducer is completed test.sh runs the K-Means iteration.

Mapper 2:

The second part of the solution is K-Means iteration, which consists of mapper.py, reducer.py, centroids.txt file, stop_condition.py and it is run via bash code in test.sh:

The mapper is saved in **/mapreduce-test/mapreduce-test-python/Part2Q2/mapper.py**

The mapper reads centroids coordinates from centroids.txt file and reads the data from a csv file via standard input. It filters the rows by player and selects only the rows where the player is one of these 4:

- 'james harden'
- 'chris paul'
- 'stephen curry'
- 'lebron james'

Mapper uses a regular expression to split each row into column values and selects

{x = SHOT DIST, y = CLOSE DEF DIST, z = SHOT CLOCK}

verifying that the values are not empty.

Mapper uses Euclidean distance formula to calculate the distance from a data point {x = SHOT DIST, y = CLOSE DEF DIST, z = SHOT CLOCK} to each of the four centroids and assigns the data point to the closest centroid.

The mapper's standard output is a print statement where key is cluster index and value is x, y, z coordinates of a data point assigned to that cluster.

Shuffling & Sorting

Upon completion of the mapper process, Hadoop initiates an intermediate process to receive the mapper output, sort generate a key-value list and sort the list. Data is then handed off to the reducer.

Reducer 2

The reducer code is uploaded and stored in:

/mapreduce-test/mapreduce-test-python/Part2Q2/reducer.py

The reducer receives the mapper output as standard input. It splits the input into separate variables: cluster **index, x, y, z**. For each cluster index reducer calculates the average of **x, y**, and **z** based on all the data points assigned to that cluster.

The reducer in this K Means algorithm takes advantage of the intermediate step of Hadoop performing data sort based on the key between the mapper and reducer. The structure of the if statement switch in the reducer relies on the intermediate sorting step to provide correct output.

The reducer's output are the four new centroids printed on the screen.

The test.sh file saves the reducer's new centroids output into **new_centroids.txt** file and updates the **centroids.txt** file on every iteration if conditions are met.

The test.sh is saved in **/mapreduce-test/mapreduce-test-python/Part2Q2/test.sh**

The test.sh file controls the K-Means iteration with a conditioned while loop. The conditions are threshold of centroids delta being less than **0.05** or number of the iteration reaching **15**.

Each time reducer calculates new centroids they are saved into **new_centroids.txt** file. The new centroids are compared to the **centroids.txt** file containing the centroids from the previous iteration to check if the threshold stopping condition is met.

If yes, the while loop stops, and next step is executed. If no, the while loop continues until either the threshold of **0.05** is met, or the number of iterations reaches **30**.

Test.sh runs **stop_condition.py** script which performs the threshold condition check between the centroids saved in **new_centroids.txt** and **centroids.txt** by comparing the absolute distance between centroids to **0.05**.

The stop_condition.py is saved in **/mapreduce-test/mapreduce-test-python/Part2Q2/ stop_condition.py**

Mapper 3:

The third part of the solution is finding the best zone for each of the four players based in the highest hit rate of each player by zone. Once the four clusters are finalized **mapper2.py** and **reducer2.py** process the data points and final clusters to calculate the hit rate for each player based on the clusters and find the best zone for each player.

Mapper2.py is located at **/mapreduce-test/mapreduce-test-python/Part2Q2/mapper2.py**

Mapper2 reads csv file row by row as standard input and retrieves the finalized clusters from **new_centroids.txt** file.

- It selects the data for the four players james harden, chris paul, stephen curry, lebron james.
- Checks that selected data is not empty: shot_clock , shot_dist , close_def_dist, player_name , shot_result.
- Assigns each data point to a centroid selects {x = SHOT DIST, y = CLOSE DEF DIST, z = SHOT CLOCK}.
- Encodes the shot_result: 1 if “made” and 0 if “missed”. The mapper2 provides the following output:
  the key is cluster index, player name, the value is shot clock, shot distance, close defender distance and shot result as 1 or 0.

Reducer 3

The **reducer2** is located at **/mapreduce-test/mapreduce-test-python/Part2Q2/reducer2.py**

The **reducer2** receives the **mapper2** output as standard input.

It reads **new_centroid.txt** file to use the centroids coordinates in the output. The **reducer2** splits standard input into variables **centroid index, player, shot clock, shot distance, close defender distance, score**.

It performs a check that all five values are not empty, that the x – shot distance, y – close defender distance, and z – shot clock values are floats and the score value is an integer.

Then **mapper2** uses an if statement to perform calculation of the hit rate for each unique cluster, player combination based on the shot distance, closest defender distance, shot clock belonging to the cluster and shot result associated with each data point. The formula used is:

hit rate = (sum shot result) / (count shot result)

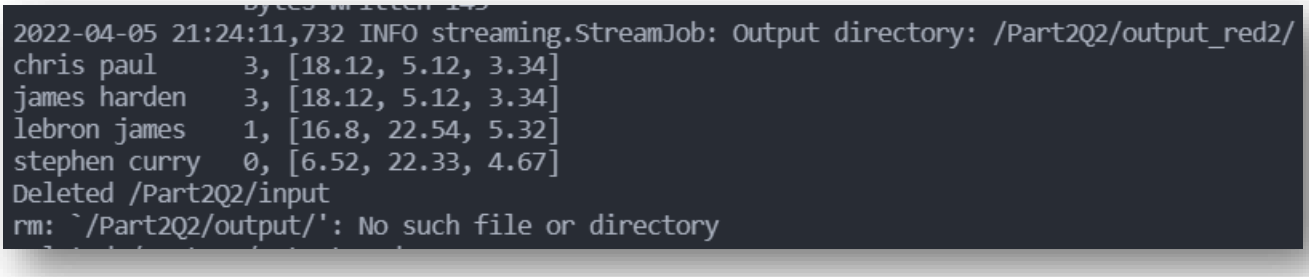
Similar to the previous mapper, this if statement also takes advantage of the intermediate step of data being sorted on the key by Hadoop between mapper and reducer.

The results are added into a dictionary where key is player name, the value is a list of hit rate and centroid index. The dictionary is then sorted by key in alphabetical order and by the first index in the value list (hit rate) in the descending order.

The **reducer2** standard output is printed on the screen as **player name, cluster index** for the players most comfortable zone, and centroids for that cluster, taken from the **new_centroids.txt** file.

Results Question 2

The following screenshot shows the most favorable zones for each of the four shooters:



The following screenshot shows the Finalized Centroids for the four clusters in order 0, 1, 2, 3