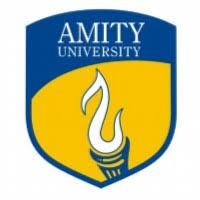
AMITY UNIVERSITY

---------UTTAR PRADESH-----------



Fundamental of Big Data Analytics

LAB FILE

In partial fulfillment of the requirement for the

Degree of Bachelor of Technology in CSE

Submitted to: Submitted By: Dr. Richa Tiwari Ajay Kumar Singh

A2305216174

7CSE-3X

Department of Computer Science and Engineering

ASET AUUP

**INDEX**

**Exp. Name of Experiment Date of Date of Marks Sign**

**No. Experiment Evaluation Obt.**

Write a MapReduce script to count the

1 occurrence of each word in a file. 15-07-19 29-07-19

Write a MapReduce script to find the

2 max and min temperature recorded in last 50 years.

29-07-19 05-08-19

Write a MapReduce script to get the name of the customer, how many of transactions a customer has done and

3 how much amount of transactions a particular customer has made.

05-08-19 19-08-19

To analyze document database such as

4 MongoDB using CRUD operations. 19-08-19 26-08-19

Write a pig script to analyses the

5

twitter data. 26-08-19 16-09-19

**Experiment – 1**

**Date: 15th July, 2019**

**Objective:** Write a MapReduce script to count the occurrence of each word in a file.

**Software Used:** Hadoop

**Theory:** MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. In MapReduce word count, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

**CODE:**

-> MAPPER

#!/usr/bin/env python import sys

for line in sys.stdin:

words = line.strip().split()

for word in words:

print '%s\t%s' % (word, 1)

-> REDUCER

#!/usr/bin/env python import sys

prev\_word = None current\_count = 0

for line in sys.stdin:

word, count = line.strip().split('\t')

count = int(count)

if prev\_word == None:

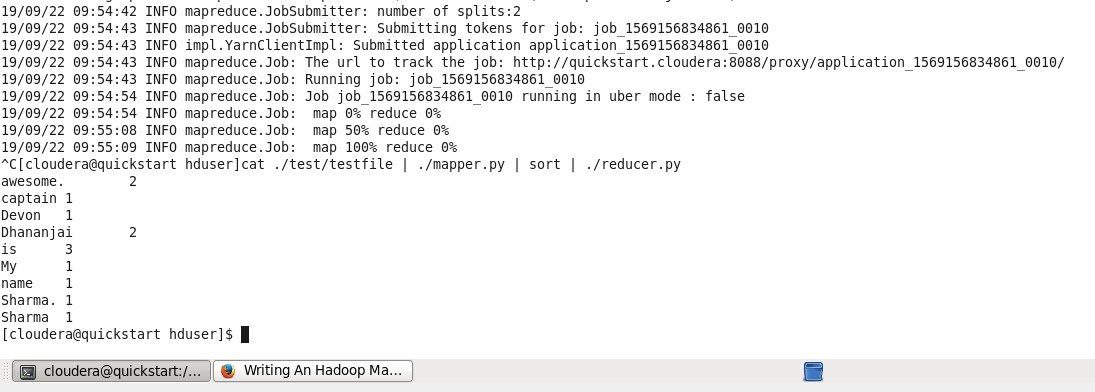
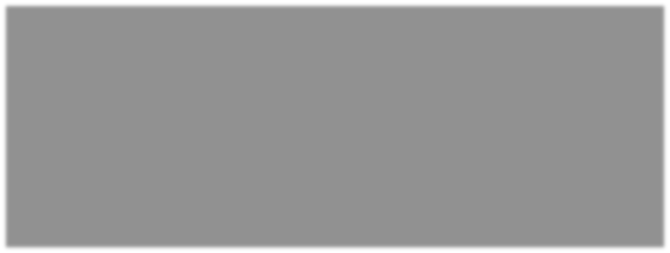
current\_count += count if prev\_word == word:

current\_count += count

else:

print '%s\t%s' % (prev\_word, current\_count)

current\_count = count prev\_word = word



print '%s\t%s' % (prev\_word, current\_count)

**OUTPUT:**

**FIG 1.1**

**RESULT:** Hence, we successfully found the count of the occurrence of each word in a file using MapReduce script.

**Experiment – 2**

**Date: 29th July, 2019**

**Objective:** Write a MapReduce script to find the max and min temperature recorded in last 50 years.

**Software Used:** Hadoop

**Theory:** MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

• **Map stage** − The map or mapper’s job is to process the input data. Generally, the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

• **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage.

The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

**CODE:**

-> MAPPER

#!/usr/bin/env python import sys

min = sys.maxint max = -sys.maxint

for line in sys.stdin:

place, time, temperature = line.strip().split('\t')

if temperature > max:

max = temperature time\_max = time

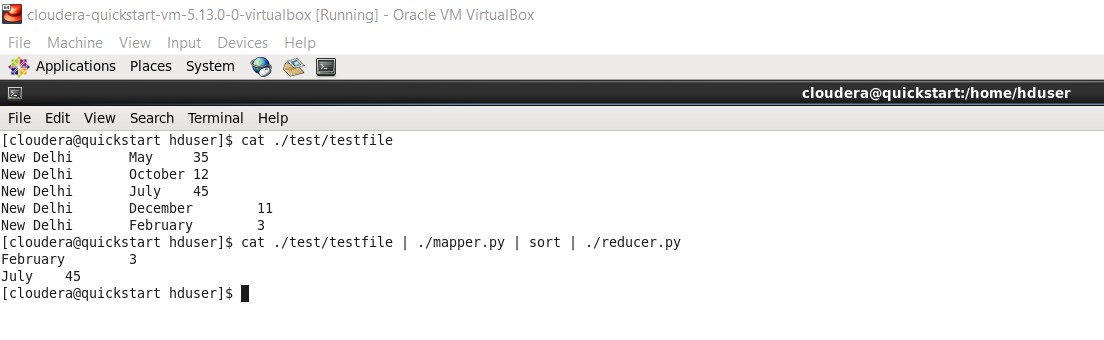
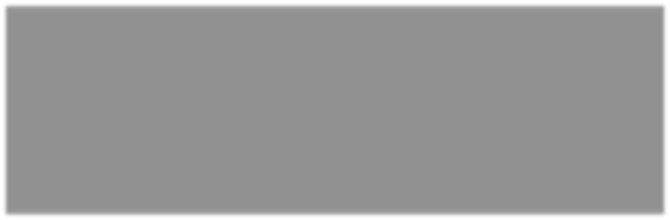
if temperature < min:

min = temperature time\_min = time

print '%s %s' % (time\_min, min)

print '%s %s' % (time\_max, max)

-> REDUCER



#!/usr/bin/env python import sys

min = sys.maxint max = -sys.maxint

for line in sys.stdin:

time, temperature = line.strip().split()

if temperature > max:

max = temperature time\_max = time

if temperature < min:

min = temperature time\_min = time

print '%s\t%s' % (time\_min, min)

print '%s\t%s' % (time\_max, max)

**OUTPUT:**

**FIG 2.1**

**RESULT:** Hence, we successfully found the maximum and the minimum temperature using

MapReduce script.

**Experiment – 3**

**Date: 19th August, 2019**

**Objective:** Write a MapReduce script to get the name of the customer, how many of transactions a customer has done and how much amount of transactions a particular customer has made.

**Software Used:** Hadoop

**Theory:** MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

• **Map stage** − The map or mapper’s job is to process the input data. Generally, the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

• **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage.

The Reducer’s job is to process the data that comes from the mapper. After processing, it

produces a new set of output, which will be stored in the HDFS.

**CODE:**

-> MAPPER

#!/usr/bin/env python import sys

for line in sys.stdin:

data = line.strip().split('\t')

if len(data) == 4:

customer\_name, customer\_id, transaction\_id, transaction\_amt = data print '%s\t%s' % (customer\_name, transaction\_amt)

else:

continue

-> REDUCER

#!/usr/bin/env python

import sys count = 0

total\_amt = 0 prev\_customer = None

for line in sys.stdin:

customer\_name, transaction\_amt = line.strip().split('\t')

transaction\_amt = int(transaction\_amt)

if prev\_customer:

total\_amt += transaction\_amt count += 1

elif prev\_customer == customer\_name: total\_amt += transaction\_amt count += 1

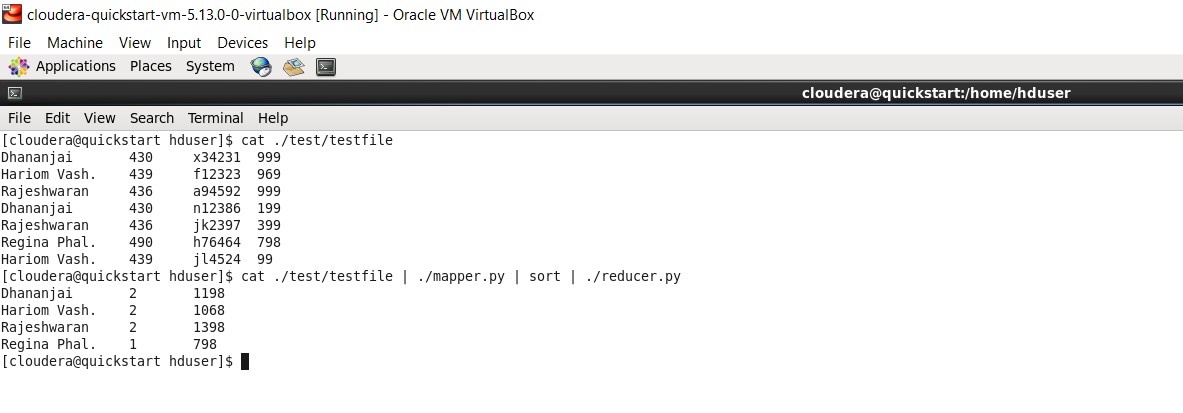
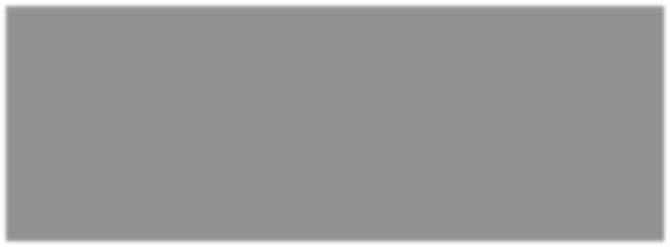
else:

print '%s\t%s\t%s' % (prev\_customer, count, total\_amt)

count = 1

total\_amt = transaction\_amt

prev\_customer = customer\_name



print '%s\t%s\t%s' % (prev\_customer, count, total\_amt)

**OUTPUT:**

**FIG 3.1**

**RESULT:** Hence, we successfully got the name of customers, the number of transactions made by the customer and the total amount spent by the customer in the transaction using MapReduce script.

**Experiment – 4**

**Date: 26th August, 2019**

**Objective:** To analyze document database such as MongoDB using CRUD operations.

**Software Used:** Hadoop

**Theory:** CRUD operations create, read, update, and delete documents.

**Create Operation**

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

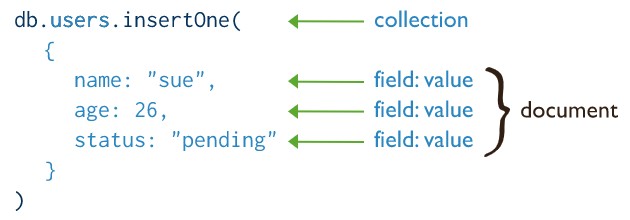
MongoDB provides the following methods to insert documents into a collection:

• db.collection.insertOne() New in version 3.2

• db.collection.insertMany() New in version 3.2

In MongoDB, insert operations target a single collection. All write operations in MongoDB

are atomic on the level of a single document.



**Read Operations**

Read operations retrieves documents from a collection; i.e. queries a collection for documents. MongoDB provides the following methods to read documents from a collection:

• db.collection.find()



You can specify query filters or criteria that identify the documents to return.

**Update Operations**

Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

• db.collection.updateOne() New in version 3.2

• db.collection.updateMany() New in version 3.2

• db.collection.replaceOne() New in version 3.2

In MongoDB, update operations target a single collection. All write operations in MongoDB

are atomic on the level of a single document.

You can specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.

**Delete Operations**

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

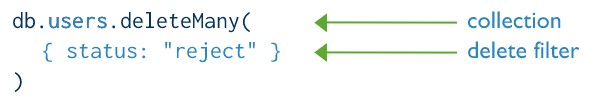
• db.collection.deleteOne() New in version 3.2

• db.collection.deleteMany() New in version 3.2

In MongoDB, delete operations target a single collection. All write operations in MongoDB

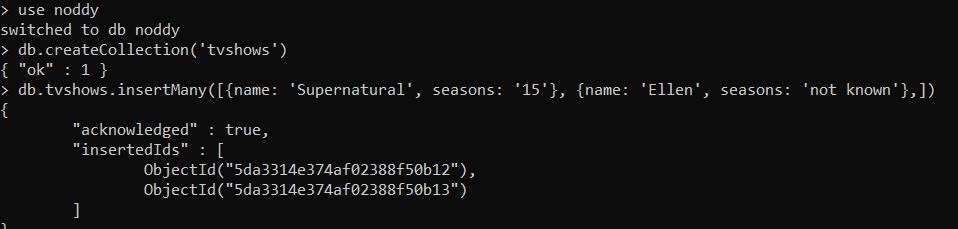
are atomic on the level of a single document.

You can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.

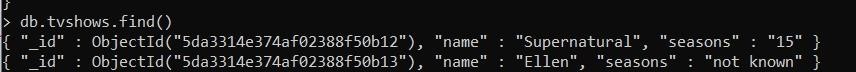


**COMMANDS & OUTPUT:**

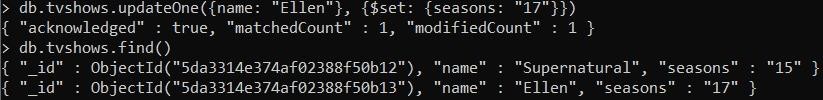
**1) CREATE**



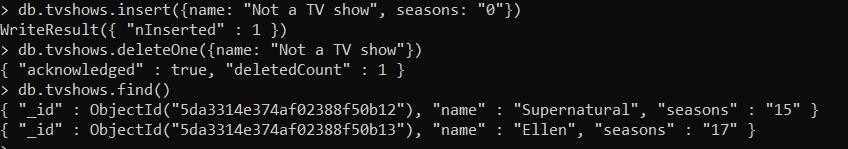
**2) READ**



**3) UPDATE**



**4) DELETE**



**RESULT:** Hence, we successfully implemented CRUD operations in MongoDB.

**Experiment – 5**

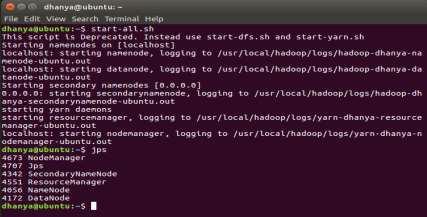
**Date: 16th September, 2019**

**Objective:** Write a pig script to analyses the twitter data.

**Software Used:** Pig Script

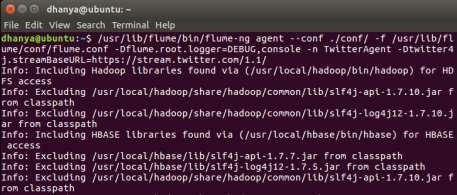
Following are the steps followed to collect and store dataset from Twitter into HDFS:-

1. Open the terminal and start all the services using the **start-all.sh** command. Then check all the Hadoop services which are running using **jps** command.

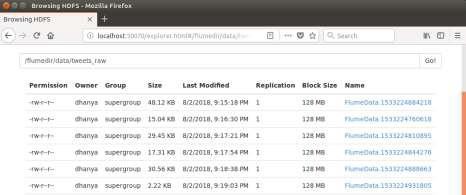


2. We will now start the flume agent using the following command:

**/usr/lib/flume/bin/flume-ng agent --conf ./conf/ -f /usr/lib/flume/conf/flume.conf - Dflume.root.logger=DEBUG,console -n TwitterAgent- Dtwitter4j.streamBaseURL=https://stream.twitter.com/1.1/**



3. This is the list of twitter data extracted which contains the keyword as specified in the conf file.



4. The dataset will look like this:



Sentiment Analysis using Pig:

• The data from Twitter is in ‘Json’ format, so a Pig JsonLoader is required to load the data into Pig. So, we have to register the downloaded jars in Pig by using the following commands:

**REGISTER '/home/dhanya/Desktop/elephant-bird-hadoop-compat-4.1.jar'; REGISTER '/home/dhanya/Desktop/elephant-bird-pig-4.1.jar';**

**REGISTER '/home/dhanya/Desktop/json-simple-1.1.1.jar';**

• The tweets are in nested Json format and consist of map data types. We need to load the tweets using JsonLoader which supports maps, so we are using elephant bird JsonLoader to load the tweets.Below is the first Pig statement that is required to load the tweets into Pig:

**load\_tweets = LOAD '/flumedir/data/tweets\_raw/' USING**

**com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS myMap;**

• Now, let’s extract the id and the hashtag from the above tweets and the Pig statement for doing this is as shown below:

**extract\_details = FOREACH load\_tweets GENERATE FLATTEN(myMap#'entities') as**

**(m:map[]),FLATTEN(myMap#'id') as id;**

In the tweet, the hashtag is present in the map object entities. Since the hashtags are inside the map entities, we have extracted the entities as map[ ] data type.

• Now, from the entities, we have to extract the hashtags which is again a map. So we will extract the hashtags as map[ ] data type as well.

**hash = foreach extract\_details generate FLATTEN(m#'hashtags') as(tags:map[]), id as id;**

• Now, from the extracted hashtags, we need to extract text which contains the actual hashtag. This can be done using the following command:

**txt = foreach hash generate FLATTEN(tags#'text') as text, id;**

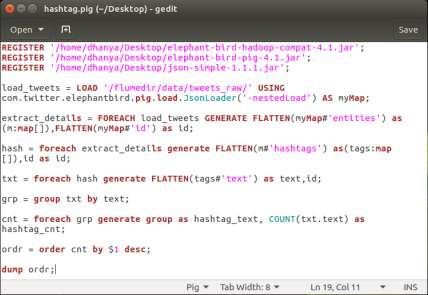
Here, we have extracted the text which starts with # and named it with an alias name text.

• Now, we will group the relation by hashtag’s text by using the below relation:

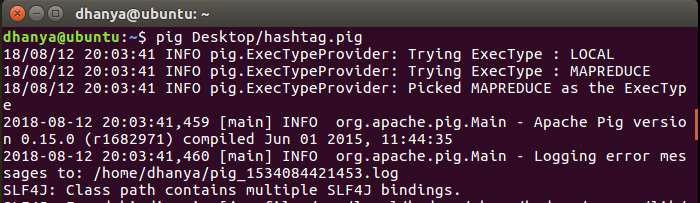
**grp = group txt by text;**

• The next thing to do is, count the number of times the hashtag is repeated by the user. This can be achieved using the below relation:

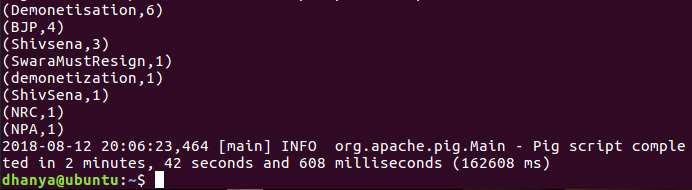
**cnt = foreach grp generate group as hashtag\_text, COUNT(txt.text) as hashtag\_cnt;**



• Now, we will run the pig script using the following command:



**Output:**



Now we have the hashtags and its count in a relation as shown in the below screen shot.

**RESULT:** Hence, we successfully implemented pig script to analyse twitter data.