

EECS E6893 Big Data Analytics HW0

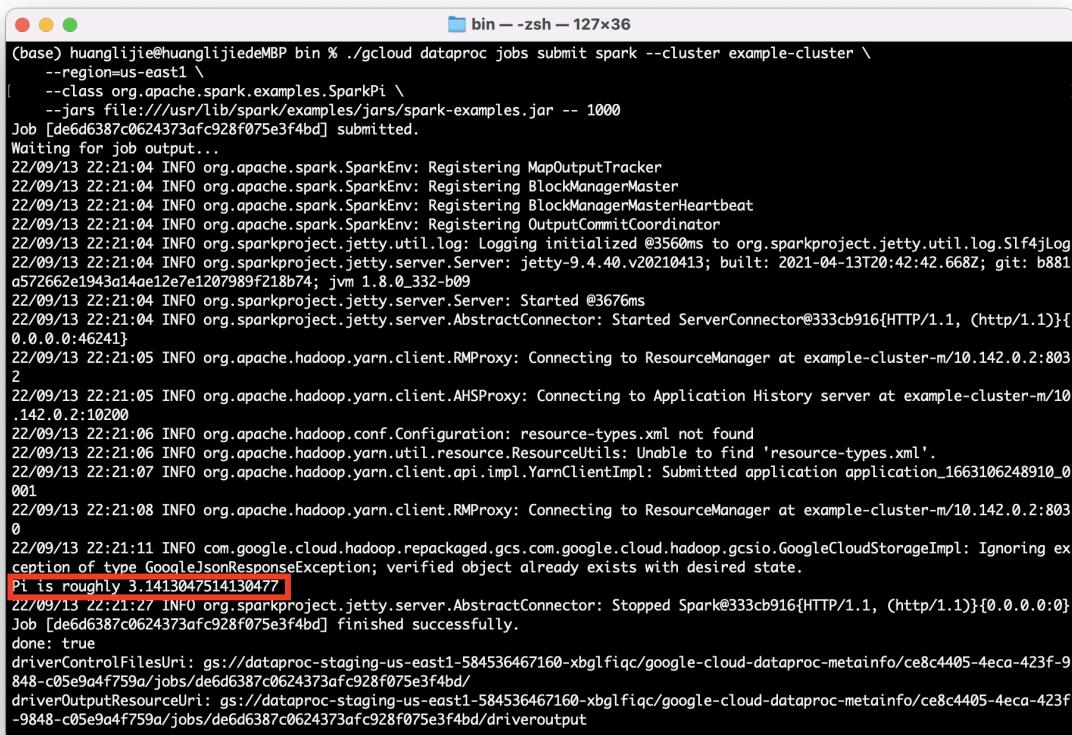
Lijie Huang, UNI: lh3158

Sep 14, 2022

1. Warm-up Exercises

(1) Provide screenshots to prove you've completed the exercises

Exercise 3 Pi Calculation:



```
bin -- zsh -- 127x36
(base) huanglijie@huanglijiedeMBP bin % ./gcloud dataproc jobs submit spark --cluster example-cluster \
--region=us-east1 \
--class org.apache.spark.examples.SparkPi \
--jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000
Job [de6d6387c0624373afc928f075e3f4bd] submitted.
Waiting for job output...
22/09/13 22:21:04 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/09/13 22:21:04 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/09/13 22:21:04 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/09/13 22:21:04 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
22/09/13 22:21:04 INFO org.sparkproject.jetty.util.log: Logging initialized @3560ms to org.sparkproject.jetty.util.log.Slf4jLog
22/09/13 22:21:04 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881
a572662e1943a14ae12e7e1207989f218874; jvm: 1.8.0_332-b09
22/09/13 22:21:04 INFO org.sparkproject.jetty.server.Server: Started @3676ms
22/09/13 22:21:04 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@333cb916[HTTP/1.1, {http/1.1}]{0.0.0.0:46241}
22/09/13 22:21:05 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.142.0.2:803
2
22/09/13 22:21:05 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at example-cluster-m/10
.142.0.2:10200
22/09/13 22:21:06 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
22/09/13 22:21:06 INFO org.apache.hadoop.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/09/13 22:21:07 INFO org.apache.hadoop.client.api.impl.YarnClientImpl: Submitted application application_1663106248910_0
001
22/09/13 22:21:08 INFO org.apache.hadoop.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.142.0.2:803
0
22/09/13 22:21:11 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring ex
ception of type GoogleJsonResponseException; verified object already exists with desired state.
Pi is roughly 3.1413047514130477
22/09/13 22:21:27 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@333cb916[HTTP/1.1, {http/1.1}]{0.0.0.0:0}
Job [de6d6387c0624373afc928f075e3f4bd] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-east1-584536467160-xbglfiqc/google-cloud-dataproc-metainfo/ce8c4405-4eca-423f-9
848-c05e9a4f759a/jobs/de6d6387c0624373afc928f075e3f4bd/
driverOutputResourceUri: gs://dataproc-staging-us-east1-584536467160-xbglfiqc/google-cloud-dataproc-metainfo/ce8c4405-4eca-423f-
9848-c05e9a4f759a/jobs/de6d6387c0624373afc928f075e3f4bd/driveoutput
```

Exercise 4 Word Count Job:

The screenshot shows the Google Cloud Platform Cloud Storage interface. On the left, there's a sidebar with 'Cloud Storage' selected. The main area displays a table of buckets:

Name	Created	Location type	Location	Default storage class	Last modified	Pub
data_to_count	Sep 13, 2022, 7:14:59 PM	Region	us-east1	Standard	Sep 13, 2022, 7:14:59 PM	Not
dataproc-staging-us-east1-58453646...	Sep 13, 2022, 5:56:30 PM	Region	us-east1	Standard	Sep 13, 2022, 5:56:30 PM	Sub
dataproc-temp-us-east1-5845364671...	Sep 13, 2022, 5:56:30 PM	Region	us-east1	Standard	Sep 13, 2022, 5:56:30 PM	Sub

Below the table is a 'Marketplace' section with a single item: 'Release Notes'.

At the bottom, there's a terminal window titled 'Terminal (big-data-analytics-362421)'. It contains the following command and its output:

```
youthtoday@cloudshell:~ (big-data-analytics-362421)$ gsutil cat gs://data_to_count/output/*
("What's", 1)
("in", 1)
("name?", 1)
("That", 1)
("a", 1)
("call", 1)
("rose", 1)
("other", 1)
("name", 1)
("would", 1)
("the", 1)
("as", 1)
("sweet", 1)
("a", 2)
("which", 1)
("By", 1)
("any", 1)
youthtoday@cloudshell:~ (big-data-analytics-362421)$
```

(2) List the Spark transformations and actions involved in each exercise. Identify the RDD operation that triggers the program to execute.

- Pi Calculation Job:

```

18 // scalastyle:off println
19 package org.apache.spark.examples
20 import scala.math.random
21 import org.apache.spark.sql.SparkSession
22
23 /** Computes an approximation to pi */
24 object SparkPi {
25   def main(args: Array[String]): Unit = {
26     val spark = SparkSession
27       .builder
28       .appName("Spark Pi")
29       .getOrCreate()
30     val slices = if (args.length > 0) args(0).toInt else 2
31     val n = math.min(100000L * slices, Int.MaxValue.toInt) // avoid overflow
32     val count = spark.sparkContext.parallelize(1 until n, slices).map { i =>
33       val x = random * 2 - 1
34       val y = random * 2 - 1
35       if (x*x + y*y <= 1) 1 else 0
36     }.reduce(_ + _)
37     println(s"Pi is roughly ${4.0 * count / (n - 1)}")
38     spark.stop()
39   }
40 }
41 // scalastyle:on println

```

On line 32, a RDD is created by parallelizing the data randomly generated. From line 32 to line 36, a transformation “map” and an action “reduce” are processed to count how many points are inside the circle, whose radius is 1. Finally, Pi can be estimated by calculating the probability.

- Word Count Job:

```

1 #!/usr/bin/env python
2
3 import ...
4
5 if len(sys.argv) != 3:
6     raise Exception("Exactly 2 arguments are required: <inputUri> <outputUri>")
7
8 inputUri = sys.argv[1]
9 outputUri = sys.argv[2]
10
11 sc = pyspark.SparkContext()
12 lines = sc.textFile(sys.argv[1])
13 words = lines.flatMap(lambda line: line.split())
14 wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, count2: count1 + count2)
15 wordCounts.saveAsTextFile(sys.argv[2])
16

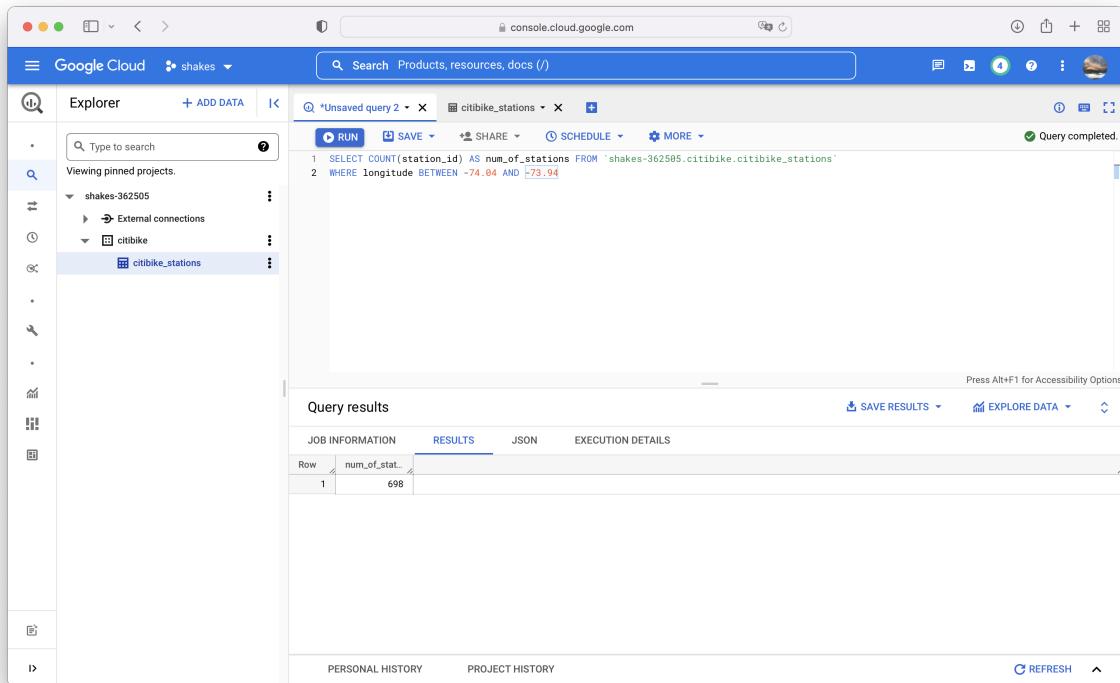
```

On line 13, a RDD is created from the text file by function “textFile”. on line 14 and 15, 3 transformations “flatMap”, “map” and “reduceByKey” are processed to count the frequency of each word. On line 16, an action (“saveAsTextFile”) is used to save the count result.

2. NYC Bike Expert

(1) How many stations with longitude between -73.94 and -74.04?

698



The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the 'shakes' project with pinned datasets: 'shakes-362505' (External connections), 'citibike' (selected), and 'citibike_stations'. The main area shows an unsaved query titled 'Unsaved query 2' with the following SQL code:

```
1 SELECT COUNT(station_id) AS num_of_stations FROM `shakes-362505.citibike.citibike_stations`  
2 WHERE longitude BETWEEN -74.04 AND -73.94
```

The 'RUN' button is highlighted. Below the code, the 'Query results' section shows a single row of data:

Row	num_of_stations
1	698

Buttons for 'SAVE RESULTS' and 'EXPLORE DATA' are visible at the bottom of the results table.

(2) What's the total number of bikes available in region_id 71?

11885

The screenshot shows the Google Cloud Big Data Editor interface. In the top navigation bar, it says "Google Cloud" and "shakes". The search bar contains "Search Products, resources, docs (/)". Below the search bar, there's a toolbar with "RUN", "SAVE", "SHARE", "SCHEDULE", and "MORE". A message "Query completed." is displayed. The main area shows a query in the editor:

```
1 SELECT SUM(num_bikes_available) FROM `shakes-362505.citibike.citibike_stations` WHERE (region_id = 71)
```

The results table shows one row with the value 11885.

(3) What's the largest capacity for a station? List all the station_id of the stations that have the largest capacity.

largest capacity: 79

The screenshot shows the Google Cloud Big Data Editor interface. In the top navigation bar, it says "Google Cloud" and "shakes". The search bar contains "Search Products, resources, docs (/)". Below the search bar, there's a toolbar with "RUN", "SAVE", "SHARE", "SCHEDULE", and "MORE". A message "Query completed." is displayed. The main area shows a query in the editor:

```
1 SELECT MAX(capacity) FROM `shakes-362505.citibike.citibike_stations`
```

The results table shows one row with the value 79.

stations with the largest capacity:

The screenshot shows the Google Cloud Big Data Editor interface. In the top navigation bar, it says "Google Cloud" and "shakes". The search bar contains "Search Products, resources, docs (/)". Below the search bar, there's a toolbar with "RUN", "SAVE", "SHARE", "SCHEDULE", and "MORE". A message "Query completed." is displayed. The left sidebar shows "Viewing pinned projects" with "shakes-362505" expanded, showing "External connections" and "citibike" which is also expanded to show "citibike_stations". The main area is titled "Query results" and shows a table with three rows of data under "RESULTS". The table has columns "JOB INFORMATION", "RESULTS", "JSON", and "EXECUTION DETAILS". The "RESULTS" tab is selected. The data table has a header row "Row" and "station_id". The data rows are:

Row	station_id
1	445
2	422
3	501

At the bottom of the results table, there are links for "PERSONAL HISTORY" and "PROJECT HISTORY". A "REFRESH" button is located at the bottom right.

3. Understanding William Shakespeare

(1) Find top-10 frequent words without any text preprocessing.

```
[(the,620),(and,427),(of,396),(to,367),(I,326),(a,256),(you,193),(in,190),(is,185),(my,170)]
```

The screenshot shows a Jupyter Notebook interface running in a browser. The title bar says "shakes-362505 > shakes-cluster". The notebook has a single cell labeled "In [1]". The code in the cell is:

```
In [1]: import pyspark  
import sys  
  
inputUri='gs://shakes_data/input/'  
  
sc = pyspark.SparkContext()  
lines = sc.textFile(inputUri)  
words = lines.flatMap(lambda line: line.split())  
wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, count2: count1 + count2)  
topWords = wordCounts.sortBy(lambda x: x[1], False).take(10)  
  
for k, v in topWords:  
    print(k, v)
```

Below the code, there is a red box highlighting the output of the print statements:

```
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
22/09/15 01:36:49 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker  
22/09/15 01:36:49 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster  
22/09/15 01:36:49 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat  
22/09/15 01:36:49 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator  
  
the 620  
and 427  
of 396  
to 367  
I 326  
a 256  
you 193  
in 190  
is 185  
my 170
```

(2) Find top-10 frequent words by first filtering out stop words provides by NLTK packages.

```
[(Macb.,137),(haue,114),(Enter,73),(thou,61),(Macd.,58),(shall,47),(vpon,47),(thy,46),(yet,45),(thee,43)]
```

The screenshot shows a Jupyter Notebook interface running on a Google Cloud DataProc cluster. The notebook title is "shakes-362505 > shakes-cluster". The code cell (In [2]) contains the following Python script:

```
import pyspark
import sys
import nltk
stop_words = nltk.corpus.stopwords.words('english')
inputUri="gs://shakes_data/input/"

sc = pyspark.SparkContext()
lines = sc.textFile(inputUri)
words = lines.flatMap(lambda line: line.split()).filter(lambda word: word.lower() not in stop_words)
wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, count2: count1 + count2)
topWords = wordCounts.sortBy(lambda x: x[1], False).take(10)

for k, v in topWords:
    print(k, v)
```

The output of the cell shows the top 10 words from the input file, along with their counts:

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/09/15 02:23:52 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/09/15 02:23:52 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/09/15 02:23:52 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/09/15 02:23:52 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator

Macb. 137
haue 114
Enter 73
thou 61
Macd. 58
shall 47
vpon 47
thy 46
yet 45
thee 43
```