EECS E6893 Big Data Analytic HW0

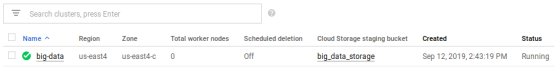
Chong Hu ch3467

September 20, 2019

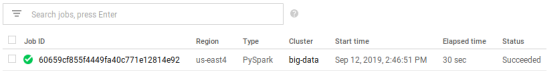
Problem 1. `Warm-up exercises`

1. Screenshots
   Initialize the cluster using command line:

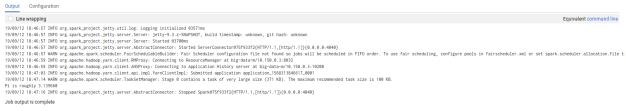
```
huchong@huchong:~/HuChong/Columbia/EECS_E6893_Big_Data_Analytics$ gcloud beta dataproc clusters create big-data --optional-components=ANACONDA,JUPYTER --image-version=1.3 --enable-component-gateway --buck
et big_data_storage  --single-node --metadata 'PIP_PACKAGES=graphframes==0.6' --initialization-actions gs://dataproc-initialization-actions/python/pip-install.sh
Waiting on operation [projects/hardy-symbol-252200/regions/us-east4/operations/d1d6d9b0-cb99-313b-aff8-e3b81cd684b2].
Waiting for cluster creation operation...
WARNING: For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/performance for i
nformation on disk I/O performance.
Waiting for cluster creation operation...done.
Created [https://dataproc.googleapis.com/v1beta2/projects/hardy-symbol-252200/regions/us-east4/clusters/big-data] Cluster placed in zone [us-east4-c].
```



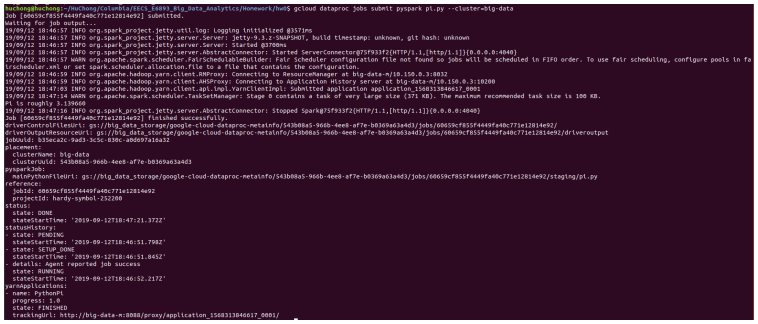
(a) "Pi calculation"



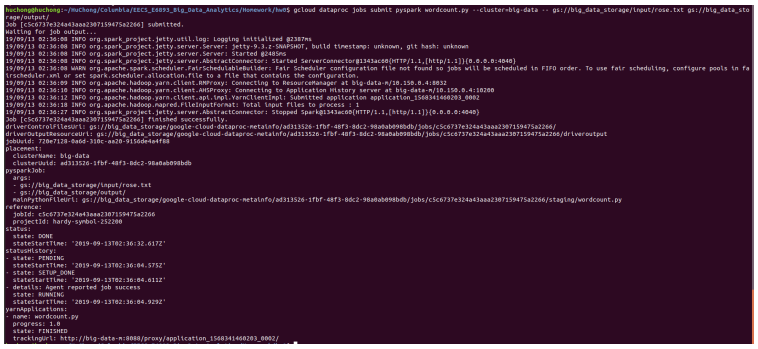
Result in GUI:

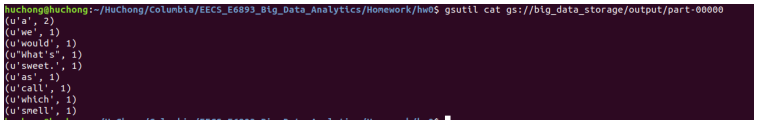


Output in command line interface:

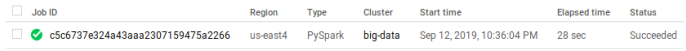


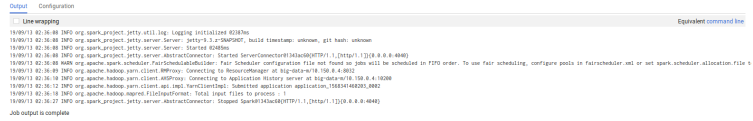
(b) "word count" job:
Result in command line:





Result in GUI:





2. Transformations and actions involved in each exercise:

   (a) "Pi calculation"
   First, we parallelize the data that randomly generated to create a RDD. Then we use "map" and "reduce", those two transformations to count how many points are inside the circle. In the end, we can simulate the the "Pi" value.

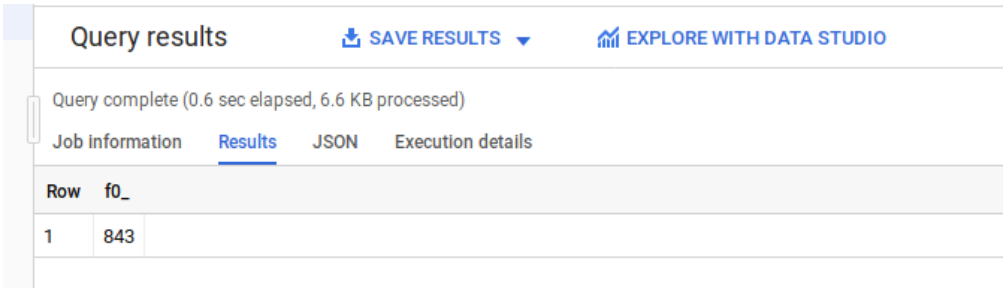
   (b) "word count"
   First, we use "textFile" function to create a RDD from text file. Then, we apply three transformations, "flatMap" to split word from lines, "map" to generate key-value pairs, and "reduceByKey" to count the frequency of each word. In the end, use "saveAsTextFile" (action) to save the "wordcount" result.

Problem 2. `NYC Bike expert`

1. How many unique *station_ids* are there in the dataset?

COUNT DISTINCT station_id

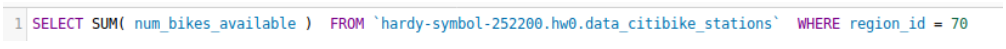
```
1 SELECT COUNT(DISTINCT station_id) FROM `hardy-symbol-252200.hw0.data_citibike_stations`
```

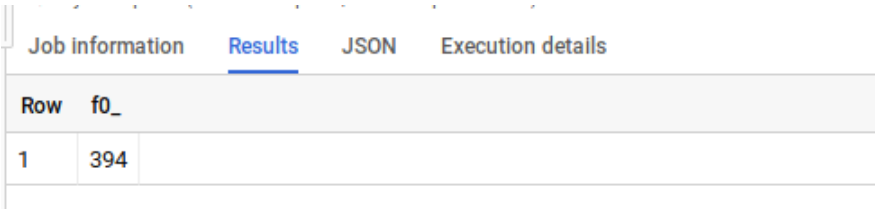


2. Whats the largest *capacity* for a station?

List all the *station_id* of the stations that have the largest capacity?

3. Whats the total number of bikes available in *region_id* 70?





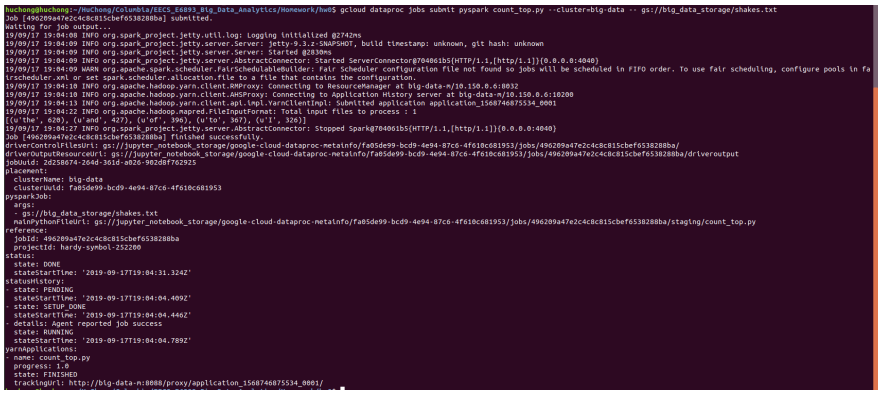
Job information | Results | JSON | Execution details

Row	f0_
1	394

Problem 3. `Understanding William Shakespeare`

1. Without any text preprocessing, here are the result for top 5 frequent words:

```
[('the', 620), ('and', 427), ('of', 396), ('to', 367), ('I', 326)]
```



Here is my source code.

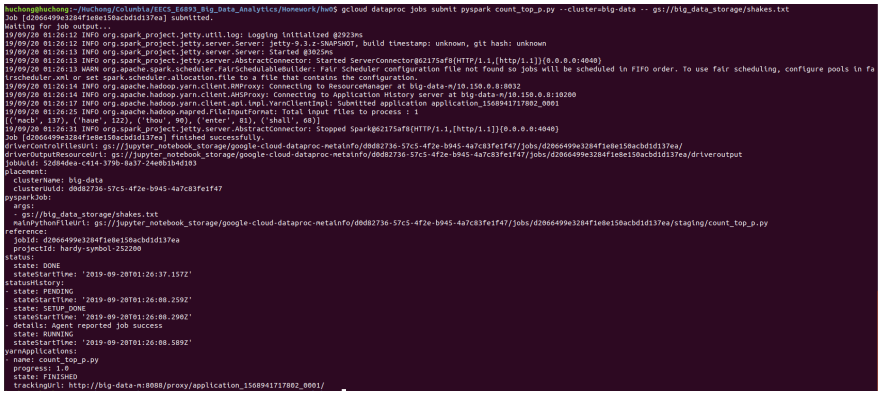
```
import pyspark
import sys

if __name__ == '__main__':
    inputUri = sys.argv[1]

    sc = pyspark.SparkContext()
    lines = sc.textFile(sys.argv[1])
    words = lines.flatMap(lambda line: line.split())
    wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, count2: count1 + cou
        lambda x: x[1], False)
    res = wordCounts.take(5)
    print(res)
```

2. With `NLTK` text preprocessing, here are the result for top 5 frequent words.

```
[('macb', 137), ('haue', 122), ('thou', 90), ('enter', 81), ('shall', 68)]
```



Here is my source code.

```python
import pyspark
import sys
import nltk

stop_words = set(nltk.corpus.stopwords.words('english'))
tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')


def stop_words_filter(line):
    text = tokenizer.tokenize(line)
    words = []
    for word in text:
        word = word.lower()
        if word not in stop_words:
            words.append(word)
    return words


if __name__ == '__main__':
    inputUri = sys.argv[1]

    sc = pyspark.SparkContext()
    lines = sc.textFile(sys.argv[1])
    words = lines.flatMap(stop_words_filter)
    wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, count2: count1 + cou
        lambda x: x[1], False)
    res = wordCounts.take(5)
    print(res)
```