Aims

This exercise aims to get you to:

- Analyze data using Spark shell with RDD
- Monitor Spark tasks using Web UI

Background

The detailed Spark programming guide is available at:

http://spark.apache.org/docs/latest/programming-guide.html

The RDD APIs in pyspark can be found here:

https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.html

The answers to the questions are given at the end of this file. Please try to answer all questions by yourself utilizing the above documents, and then check your results with the answers provided.

Install and Configure Spark

1. You need to check JAVA_HOME first. In ~/.bashrc, if not exist, add:

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

Please note that, if you use mac and the default terminal is zsh, then add this line to "~/.zshrc" instead. Do not naively paste this command to your laptop, please check your own java location by "whereis java" and paste it.

You can skip this step if Java home is already configured in previous labs.

2. Download the Spark package by the command (or use a web browser):

```
$ wget https://dlcdn.apache.org/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
```

Then unpack the package:

```
$ tar xvf spark-3.5.0-bin-hadoop3.tgz
$ mv spark-3.5.0-bin-hadoop3 spark
```

Now you have Spark installed under `~/spark`. We need to configure this folder as the working directory of Spark.

Open the file `~/.bashrc` and add the following lines to the **end** of this file:

```
export SPARK_HOME=/home/comp9313/spark
export PATH=$SPARK_HOME/bin:$PATH
```

(If you copy and paste the above two lines, please do it line by line, not together.)

Save the file, and then run the following command to take these configurations into effect:

```
$ source ~/.bashrc
```

Interactive Analysis with the Spark Shell Using RDD

Use the following command to open the pyspark shell:

```
$ pyspark
```

1. Load and inspect data from a text file:

1. Create an RDD from local files using textFile()

```
$ >>> textFile = sc.textFile("file:///home/comp9313/spark/README.md")
```

The file path varies according to your own installation directory, please modify it in case you need to.

Spark's primary abstraction is a distributed collection of items called a Resilient Distributed Dataset (RDD). RDDs can be created from Hadoop InputFormats (such as HDFS files) or by transforming other RDDs. This command makes a new RDD from the text of the README file in the Spark source directory.

You can apply the RDD transformation and action functions on "textFile".

2. Count the number of items in an RDD (count() is a built-in method of spark RDD)

```
$ >>> textFile.count()
```

You should see the results: 125

3. Get the first item in an RDD

```
$ >>> textFile.first()
```

You should see the results: "# Apache Spark"

4. Get lines containing "Spark" using the function filter()

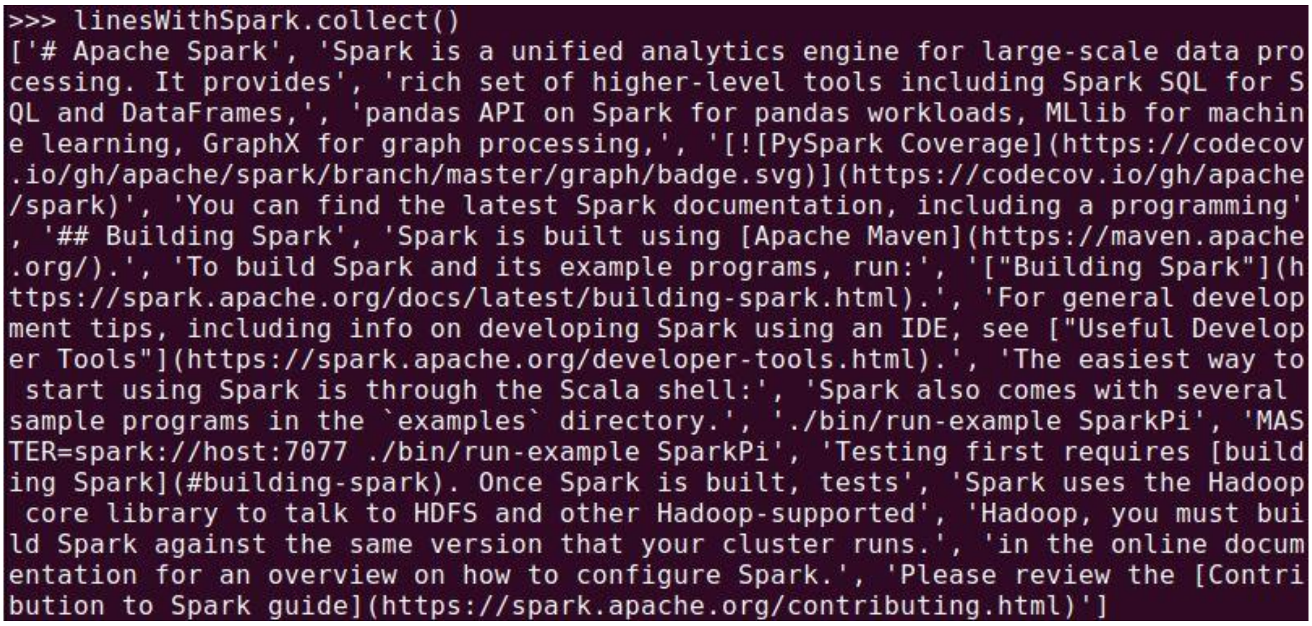
```
$ >>> linesWithSpark = textFile.filter(lambda x: "Spark" in x)
```

Please note that the anonymous function is optional, the filter method also accepts "def" defined functions. For further information (Section: Passing Functions to Spark):

https://spark.apache.org/docs/latest/rdd-programming-guide.html#passing-functions-to-spark

5. Use the function collect() to see the contents of linesWithSpark

```
$ >>> linesWithSpark.collect()
```



```
>>> linesWithSpark.collect()
['# Apache Spark', 'Spark is a unified analytics engine for large-scale data pro
cessing. It provides', 'rich set of higher-level tools including Spark SQL for S
QL and DataFrames,', 'pandas API on Spark for pandas workloads, MLlib for machin
e learning, GraphX for graph processing,', '[![PySpark Coverage](https://codecov
.io/gh/apache/spark/branch/master/graph/badge.svg)](https://codecov.io/gh/apache
/spark)', 'You can find the latest Spark documentation, including a programming'
, '## Building Spark', 'Spark is built using [Apache Maven](https://maven.apache
.org/).', 'To build Spark and its example programs, run:', '["Building Spark"](h
ttps://spark.apache.org/docs/latest/building-spark.html).', 'For general develop
ment tips, including info on developing Spark using an IDE, see ["Useful Develop
er Tools"](https://spark.apache.org/developer-tools.html).', 'The easiest way to
 start using Spark is through the Scala shell:', 'Spark also comes with several
sample programs in the `examples` directory.', './bin/run-example SparkPi', 'MAS
TER=spark://host:7077 ./bin/run-example SparkPi', 'Testing first requires [build
ing Spark](#building-spark). Once Spark is built, tests', 'Spark uses the Hadoop
 core library to talk to HDFS and other Hadoop-supported', 'Hadoop, you must bui
ld Spark against the same version that your cluster runs.', 'in the online docum
entation for an overview on how to configure Spark.', 'Please review the [Contri
bution to Spark guide](https://spark.apache.org/contributing.html)']
```

6. Print all the items in linesWithSpark

```
$ >>> linesWithSpark.foreach(print)
```

print() is a function, and it is used as an argument in function foreach(). Please note that you must remove the brackets since "foreach" accepts a function as input rather than the return from a function. "print" can be regarded as a pointer of this function.

7. Use function map() to map each line to the number of words contained in it

```
$ >>> lineNumOfWords = textFile.map(lambda line: len(line.split(" ")))
```

The argument of map() is an anonymous function, which takes a line as the input, and returns the number of words (separated by space). Check the contents of lineNumOfWords.

8. Find the largest number of words contained in a line using reduce()

```
$ >>> lineNumOfWords.reduce(lambda a, b : a if (a > b) else b)
```

You should see the result is 16. The reduce function takes an anonymous function as an argument, which takes two arguments and returns the larger one.

You can also call functions declared elsewhere. For example, you can use the max() function to make this code easier to understand:

```
$ >>> lineNumOfWords.reduce(lambda a,b: max(a,b))
```

Or

```
$ >>> lineNumOfWords.reduce(max)
```

9. Convert RDD textFile to an array of words using flatMap()

```
$ >>> words = textFile.flatMap(lambda line: line.split(" "))
```

This will split each line to a list of words, and store all of them in one array. You can compare the result obtained by flatMap() with that obtained by map(). What are the differences?

```
$ >>> words = textFile.map(lambda line: line.split(" "))
```

10. Count the distinct words in textFile using distinct()

```
$ >>> words.distinct().count()
```

Compare the results with words.count().

11. (Question) Find the longest line together with the length in textFile.

Hint: first map a line to a pair of (line, length), and then use reduce() to find the longest line.

12. (Question) Print the lines containing “Spark” with line numbers (starting from 0). Each line is printed in the format of:

Line Number in textFile: the contents of the line

Hint: Use the function zipWithIndex().

Zip the elements of the RDD with its element indexes. The indexes start from 0.

2. More operations on pair RDD:

1. Download the data set auctiondata.csv from the course webpage and store it in your home folder. The datasets contain eBay auction information on Cartier wristwatches, Palm Pilot M515 PDAs, Xbox game consoles, etc. Each record has 9 fields:

- `aucid`: unique identifier of an auction
- `bid`: the proxy bid placed by a bidder
- `bidtime`: the time in days that the bid was placed, from the start of the auction
- `bidder`: eBay username of the bidder
- `bidderrate`: eBay feedback rating of the bidder
- `openbid`: the opening bid set by the seller
- `price`: the closing price that the item sold for (equivalent to the second highest bid + an increment)
- `itemtype`: auction item
- `dtl`: auction_type

Define the mapping for the input variables. They are used to refer to different fields of the data set.

```
$ >>> aucid = 0

$ >>> bid = 1

$ >>> bidtime = 2

$ >>> bidder = 3

$ >>> bidderrate = 4

$ >>> openbid = 5

$ >>> price = 6

$ >>> itemtype = 7

$ >>> dtl = 8
```

2. Load data into Spark

```
$ >>> auctionRDD =
sc.textFile("file:///home/comp9313/auctiondata.csv").map(lambda line :
line.split(","))
```

In auctionRDD, each item is an array containing 9 fields, and you can use the defined variables to access each field. **Note that you need to use the correct file path.**

3. Count the total number of item types that were auctioned.

```
$ >>> auctionRDD.map(lambda x: x[itemtype]).distinct().count()
```

Each item in auctionRDD is an array of String objects. x[itemtype] is equivalent to x[7], and it is used to get the 8th object in the array.

4. (Question) What is the total number of bids per item type? The output is a list of key-value pairs <item type, number of bids>.

Hint: First create a pair RDD by mapping each record to a pair of (item type, 1), and then use reduceByKey() to do the aggregation for each item type

5. (Question) Across all auctions, what is the maximum number of bids for one auction?

Hint: First use reduceByKey() to count the number of bids for each auction, and then find the maximum number using reduce()

6. (Question) Across all auctions, what are the top-5 auctions that have the most number of bids?

Hint: First use reduceByKey() to count the number of bids for each auction, and then use sortBy() to sort the key-value pairs in descending order. Finally, use take() to get the top-5 results. You can also use the top() operation to complete this task. You can pass a function to top() to let Spark know how to sort your data.

3. Do word count in the pyspark shell

Start HDFS (you can also use the local file as input), get the file “pg100.txt” from WebCMS3 and upload it to HDFS:

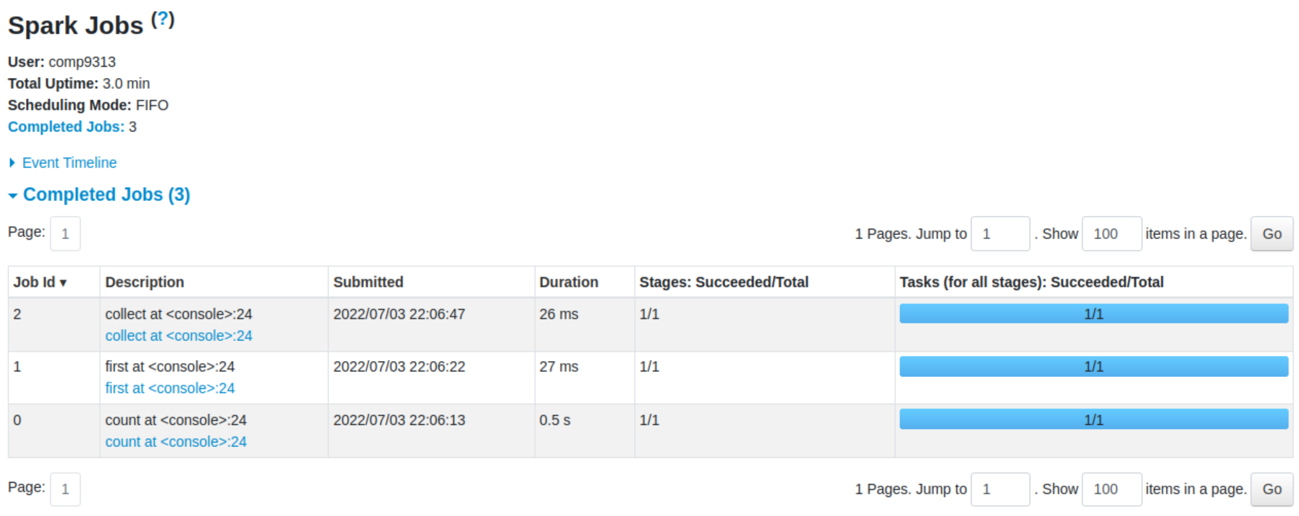
```
$ hdfs dfs -put pg100.txt
```

Load the file into Spark from HDFS, and use the functions map(), flatMap(), reduceByKey() to do word count (split the documents by the space character). Finally, store the results in HDFS using saveAsTextFile() and check the output.

RDD partitions: In the function reduceByKey(), set the number of tasks to 3, and check the results again (each task will be processed by one reducer, and thus three output files).

Spark Web UI

Browse the web interface for the information of Spark Jobs, storage, etc. at: http://localhost:4040. You will see something like:



User: comp9313
Total Uptime: 3.0 min
Scheduling Mode: FIFO
Completed Jobs: 3

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	collect at <console>:24 collect at <console>:24	2022/07/03 22:06:47	26 ms	1/1	1/1
1	first at <console>:24 first at <console>:24	2022/07/03 22:06:22	27 ms	1/1	1/1
0	count at <console>:24 count at <console>:24	2022/07/03 22:06:13	0.5 s	1/1	1/1

You can click each task to see more details of the execution.

Answers:

1.11. (Question) Find the longest line together with the length in textFile.

```
>>> pairs = textFile.map(lambda line : (line,len(line)))

>>> res = pairs.reduce(lambda x,y : x if x[1] > y[1] else y)
```

1.12. (Question) Print the lines containing “Spark” with line numbers (starting from 0).

```
>>> lineWithNumber = textFile.zipWithIndex()

>>> lineWithSpark = lineWithNumber.filter(lambda line: "Spark" in
line[0]).map(lambda line:(line[1],line[0]))
```

2.4. (Question) What is the total number of bids per item type? The output is a list of key-

```
>>> from operator import add

>>> auctionRDD.map(lambda x: (x[itemtype],1)).reduceByKey(add).collect()
```

2.5. (Question) Across all auctions, what is the maximum number of bids of one auction?

```
>>> auctionRDD.map(lambda x: (x[aucid],1)).reduceByKey(add).reduce(lambda x,y: x
if x[1]>y[1] else y)
```

2.6. (Question) Across all auctions, what are the top-5 auctions that have the most number of bids?

```
>>> auctionRDD.map(lambda x: (x[aucid],1)).reduceByKey(add).sortBy(lambda
x:x[1],ascending=False).take(5)
```

You can also try the “takeOrdered()” method, which is more efficient than “sortBy().take()”.

```
or

>>> auctionRDD.map(lambda x: (x[aucid],1)).reduceByKey(add).top(5,key=lambda
x:x[1])
```