1. **How many lines of text appear in the gutenberg dataset?**

9544259

1. **Based on what you can find from the logs, how many partitions does the 'lines' RDD have? Remember RDDs are distributed among multiple nodes of your cluster. Does that match with what you observed when running your jobs in lab 2?**

4 partition

1. **Based on what you see, do you think the lines RDD is shared between two operations, or is it created again each time?**

It’s shared between two operations.

upon a spot attractive to the feathered world, he will be sure, sooner

and Advocate for his people?

Portugal, Puerto Rico, Romania, Russia, Rwanda, Saint Lucia, Saint

the wife and mother, will rush to meet him, fall into his arms.

the first considered Porto Rico the best immediate objective: it was

II

Each of these transformations requires a matching function. For counting words, we will need to use the following three:

lambda line: WORD\_REGEX.findall(line)  
lambda word: (word.lower(),1)  
lambda a,b: (a+b)

Combine these three transformations and three functions to implement word count. The code will have a structure similar to the following:

import re # we need this import to use the regular expressions function of the first lambda  
WORD\_REGEX = re.compile(r"\b\w+\b")  
mapper\_length  
step1 = lines.\_\_\_\_\_\_(\_\_lambda\_1\_\_)  
step2 = step1.\_\_\_\_\_\_(\_\_lambda\_2\_\_)  
result = step2.\_\_\_\_\_\_(\_\_lambda\_3\_\_)

**>>> step1 = lines.flatMap(lambda line: WORD\_REGEX.findall(line))**

**>>> step2 = step1.map(lambda word:(word.lower(),1))**

**>>> result = step2.reduceByKey(lambda a,b:a+b)**

**To make sure this works: step1.lines.takeSample(False, 5)**

**step2.lines.takeSample(False, 5)**

1. Does computation start once you have written these lines including the result? Why?

NO, BECAUSE NOTHING HAS BEEN RUN AND in this case result.count()

>>> result.count()

651063

**5. How many times is summer mentioned in the dataset? Modify now this last query (you can push the up arrow key to quickly reload the last command), and check the popularity of other words. You should see substantially improved performance for all these subsequent invocations. This is thanks to the persist()transformation we invoked early.**

**>>> inmem = result.persist()**

>>> inmem.**filter**(lambda pair: pair[0] == "summer").**collect**()

[('summer', 6195)]

>>> inmem.**saveAsTextFile**("Out-Lab5")

**6. How many files appear in the out-lab5 folder? Why is that the case?**

There are 4 files (Successful file excluded), there were 4 partitions

1. Based on what you observe from the code, is takeOrdered a transformation or an action? Why is that the case?

**takeOrdered**(n,key=func) **Takeordered** is an **action** that returns n elements ordered**in** ascending order as specified by the optional key function: If key function returns a negative value (-1), the order is a descending order. Python List: rdd = sc.

YHOO: 1999-12-07;92339016000.0

QCOM: 1999-12-30;83465070400.0

QCOM: 1999-12-29;77816828800.0

INTC: 1997-05-30;69540681600.0

INTC: 1993-04-19;55493535200.0

QCOM: 1999-11-12;55071576000.0

QCOM: 1999-11-15;48278067200.0

DELL: 1997-07-25;44033993600.0

YHOO: 1998-07-06;43181460000.0

MSFT: 1989-03-07;42194808000.0