**Recitation 1: Spark (and Hadoop)**

**File Sources:**

Please download from sakai/Recitation/Rec1 to your pyspark local directory the text and csv related files

**Task1: Jupyter and Python**

**Part 1:** Using your LINUX OS **type the proper commands** to bring up Jupyter (as in your instructions). Check if these instructions work for your machine’s configuration. If not… spend a few minutes to figure out what you need to do, then call us to the rescue!

The first step is to make sure it works and it is launched without errors.

**Part 2:** From the browser’s view of Jupyter, click on “New”. You can create text files or folders, but more commonly, you’ll want to create Notebooks in Python. If you get the option, click on Python 3, and type the following into the In [1] Cell:

import pyspark

sc = pyspark.SparkContext('local[\*]')

rdd = sc.parallelize(range(1000))

rdd.takeSample(False, 5)

Click on the “Run” button. You should ultimately get a result that looks like the following (probably with different values): out[1]: [435, 567, 23, 256, 902]. You have just run a very simple Spark program that created a vector of 1000 numbers in parallel, then sampled 5 of them without replacement!

**Task 2: Wordcount in Python**

Please use the Anaconda Spark version you hopefully downloaded following the instructions we sent.

Count the words included in file: file01\_Hd\_Sp\_Freq.txt and report the result.

from pyspark import SparkContext

from operator import add

f = sc.textFile("file01\_Hd\_Sp\_Freq.txt")

// Complete the missing part of this instruction in Python

wc = f.flatMap(blah1).map(blah2).reduceByKey(add)

wc.saveAsTextFile("wc\_out.txt")

**Task 3: Count Frequencies of Specific Words in Python**

Now use the same file: file01\_Hd\_Sp\_Freq.txt in order to compute the three words encountered with the highest frequencies. Do some search to figure out the correct instructions.

from pyspark import SparkContext

from operator import add

sc = SparkContext(appName="Word\_Freq")

lines = sc.textFile(“file01\_Hd\_Sp\_Freq.txt")

# counts is an RDD of the form (word, count)

counts = lines.flatMap(blah blah blah).reduceByKey(add)

# collect brings it to a list in local memory and you can extract info in the form of (word, count)

output = counts.collect()

Insert your own code … blah blah blah

Some useful instructions to use can be: sort related, reduce by Key related, take…

sc.stop() # stop the spark context

**Task 4: Pi Estimation**

Spark can also be used for compute-intensive tasks. This code estimates π by "throwing darts" at a circle. We pick random points in the unit square ((0, 0) to (1,1)) and see how many fall in the unit circle. The fraction should be π / 4, so we use this to get our estimate.

Go to: <http://spark.apache.org/examples.html>

Simply run the program --- you only need to configure the output. Show result.

**Task 5: Search Text and Create RDDs: Practice from SCALA to Python**

Convert the SCALA code below to Python, run it and show results. If there are errors in the given code that prevent you from obtaining the expected result, please make a simple fix and show result.

from pyspark import SparkContext

# Insert your own code, transform the code below:

val lines = sc.textFile("file01\_Hd\_Sp\_Freq.txt ")

// transformed RDDs

val selfish = lines.filter(\_.startsWith("I"))

val messages = selfish.map(\_.split("\t")).map(r => r(1))

messages.cache()

// action 1

messages.filter(\_.contains("Spark")).count()