Exercise 7

Get started with Apache Spark and Python

Prior Knowledge
Unix Command Line Shell
Simple Python

Learning Objectives
Understand the Spark system
Understand the Jupyter Notebook model
Submit Spark jobs locally and using YARN
Write SparkSQL code in Python
WordCount!

Software Requirements
(see separate document for installation of these)

- Apache Spark 2.4.3
- Python 2.7.x
- Jupyter notebooks

Part A. Spark Python Shell (pySpark)

1. We are going to do a wordcount against a set of books downloaded from Project Gutenberg. Wordcount is the definitive Big Data program (sort of `Hello World` for Big Data) and it is frankly embarrassing that we haven't done one yet.

2. Apache Spark has a useful Python shell, which we can use to interactively test and run code.

3. We are going to start by using data in HDFS, so *we need to ensure HDFS is running.* (Follow the instructions from the Hadoop lab).

4. Let's load some books into HDFS. In a terminal window (Ctrl-Alt-T)

   ```
   hadoop fs -mkdir -p /user/oxclo/books
   hadoop fs -put ~/datafiles/books/* /user/oxclo/books/
   ```

5. Let's make a directory for our code:

   ```
   mkdir ~/pse
   cd ~/pse
   ```

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

6. Now start the Spark Python command line tool –

   ```
   pyspark
   ```

7. In the command-line you will see something like

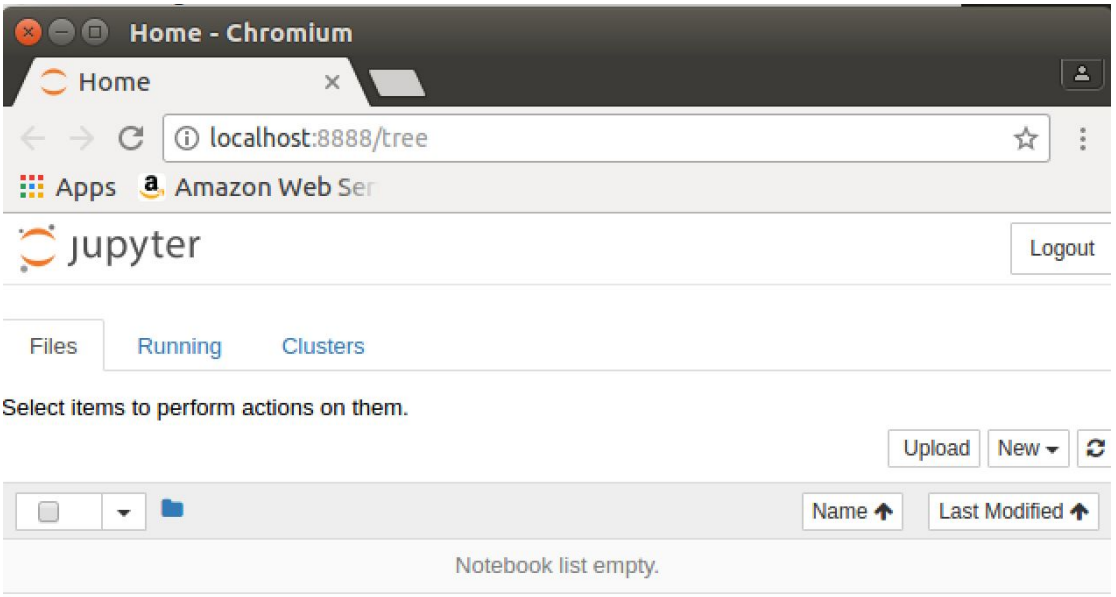
   ```
   [I 13:53:23.865 NotebookApp] Serving notebooks from local directory: /home/oxclo/pse
   [I 13:53:23.866 NotebookApp] 0 active kernels
   [I 13:53:23.866 NotebookApp] The Jupyter Notebook is running at:
   http://localhost:8888/?token=fd655aab32ed4840ceb47b8b7392b1243a27f56350888a91
   [I 13:53:23.866 NotebookApp] Use Control-C to stop this server and shut down all
   kernels (twice to skip confirmation).
   [C 13:53:23.868 NotebookApp]

       Copy/paste this URL into your browser when you connect for the first time,
       to login with a token:
           http://localhost:8888/?token=fd655aab32ed4840ceb47b8b7392b1243a27f56350888a91
   ```

8. Jupyter is starting. This is because I have preconfigured the following environment variables in `~/.bashrc`

   ```
   export PYSPARK_DRIVER_PYTHON=jupyter
   export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
   ```

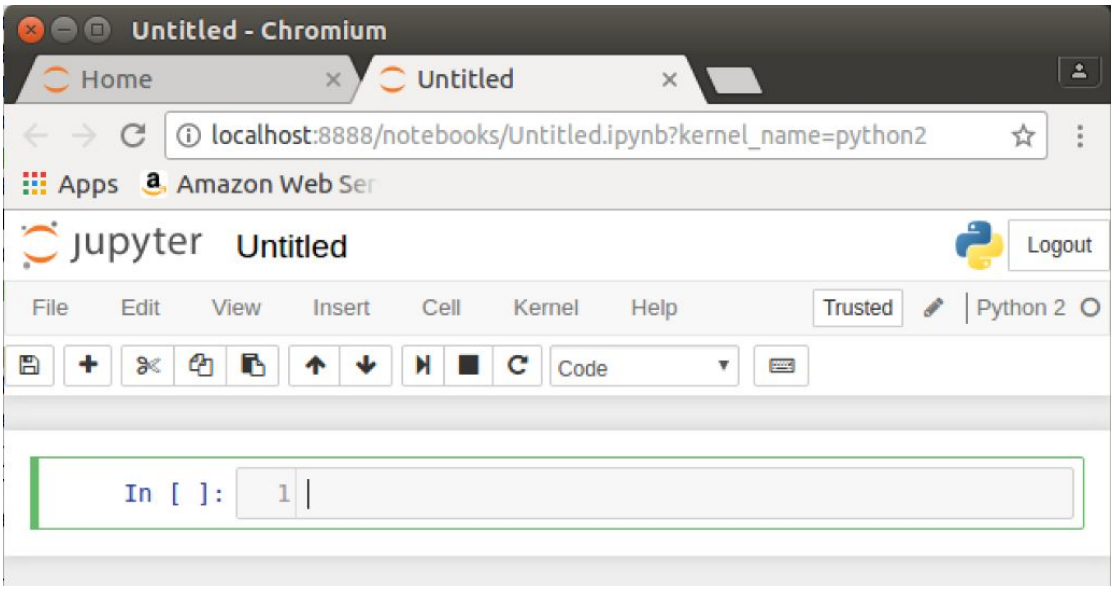
   If you want to use the more basic version of pyspark, you'll have to unset these.

9. And then a browser window will pop up.



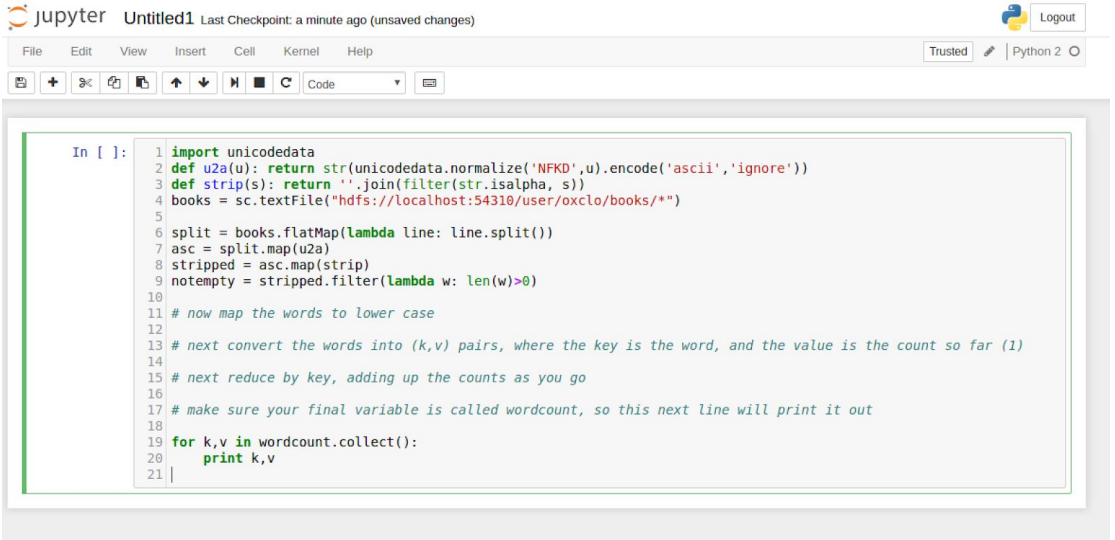
© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

10. Use the **New** button to create a new Python2 notebook:



11. There is a starter of the code you need in the following URL: https://freo.me/first-notebook

    Paste that into the cell [1] so it looks like this:



12. There are some aspects that are not filled in that you need to write. Basically this is a data-processing pipeline (also a directed acyclic graph)

13. *Let's look at the parts that are there already.*

14. We already have a SparkContext object defined in the notebook (in a program you need to define one, which we will see later)

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

15. Unfortunately some of the input is handled as Unicode by Python and we want to get rid of that:

```
import unicodedata
def u2a(u): return str(unicodedata.normalize('NFKD',u).
encode('ascii','ignore'))
```

16. We also want to remove any non-alphanumeric characters:

```
def strip(s): return ''.join(filter(str.isalpha, s))
```

17. With the preliminaries over, the next line loads the data in:

```
books =
sc.textFile("hdfs://localhost:54310/user/oxclo/books/*")
```

18. Then splits the lines into separate words

```
split = books.flatMap(lambda line: line.split())
```

19. Deals with the Unicode problem

```
asc = split.map(u2a)
```

And removes non-alpha characters:

```
stripped = asc.map(strip)
```

and removes empty items:

```
notempty = stripped.filter(lambda w: len(w)>0)
```

20. Now it is time for you to do something!

Convert all the words to lower case, using a map operation. In python, if *str* is a string, then `str.lower()` is the same string in lower case.

21. Now you need to get ready for a reduce. In order to do a reduce, we need some form of *key, value* pairs. I recommend using *tuples* which are simply (k,v) in Python (the brackets group the items into a tuple).

22. Remembering how reduce works, we need each word to have a count. Before reducing, that count is 1. So we need a lambda that takes a word w and returns (w,1)

23. Now we can do a reduce that adds all those counts together.

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

24. Finally, we need to collect the results and print them. In Spark, they may be distributed across different RDD partitions on different machines, so the collect() method brings them together.

```
for k,v in wordcount.collect(): print k,v
```

25. Try running the cell, by clicking

26. Be patient. I suggest you look at the command window and wait until you see spark start working.

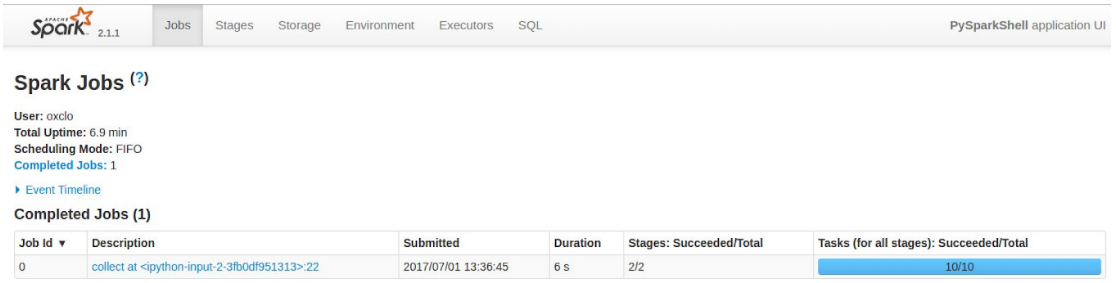
27. You should see a word count appear below cell 1:

```
systematic 7
parallelogram 1
sowell 1
presnya 1
four 265
conjuring 1
chamberagain 1
marching 32
sevens 4
awistocwacy 1
trotat 1
canes 1
shipmets 1
understandthat 2
lorn 16
lore 1
inwards 2
wickam 62
utterand 1
```

28. Congratulations!

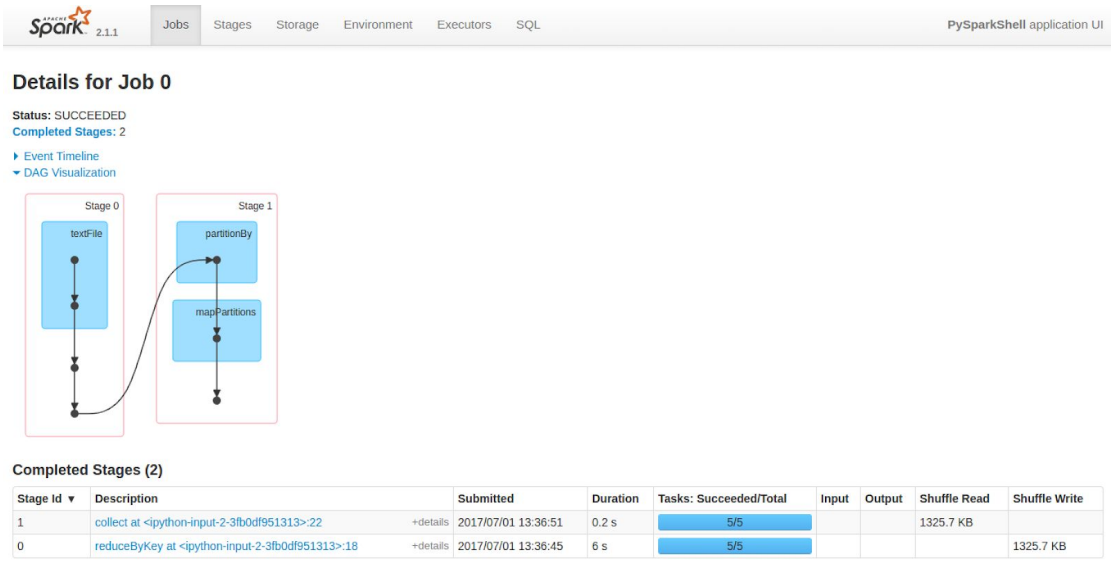
29. While the pyspark is still running browse to http://localhost:4040

30. You will see the Spark web console:

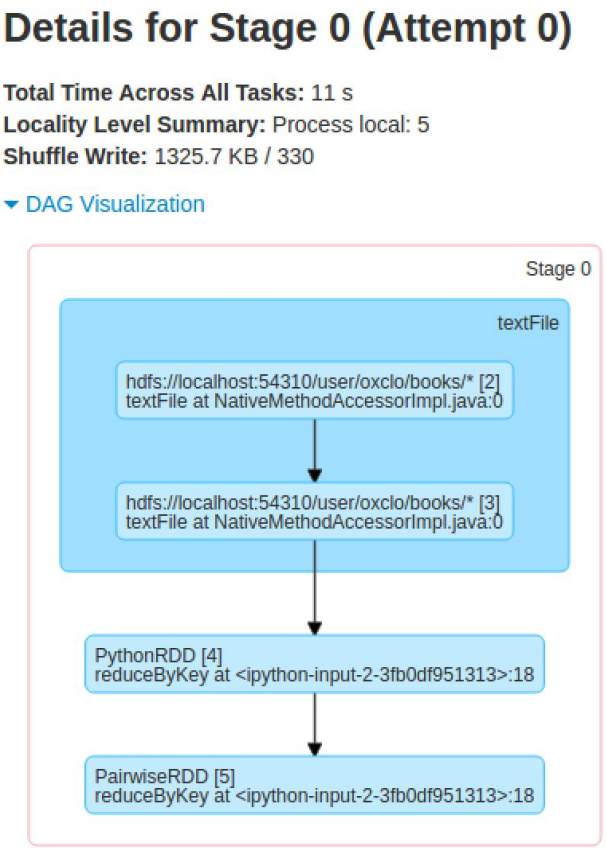


© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

31. Click on the blue link "collect at ipython-input"
    This shows you how Spark converted your code into stages:

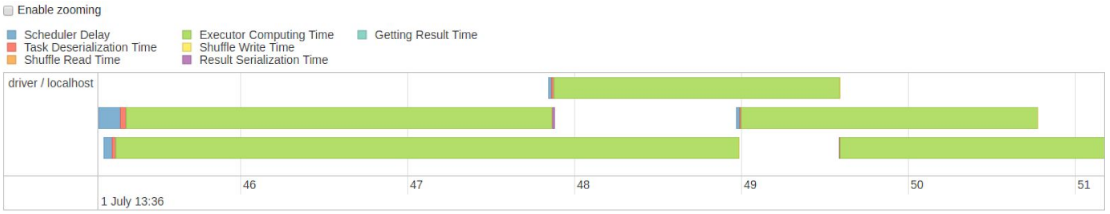


32. Click on Stage 0



© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

33. And expand the Event Timeline:



34. Make sure your code is saved from the notebook.

35. Quit the notebook shell by typing Ctrl-C on the command line, and then Y
    Also close the notebook windows in the browser.

36. Now let's run the same code as a "job" instead of interactively.

37. From http://freo.me/oxclo-wc-py copy the code into a file `wc-job.py`

38. You will notice that there is a bunch of "setup" code that we didn't need in the pyspark command line tool. That is because pyspark assumes you want all this and does it for you.

39. The default environment in your terminal session is telling Pyspark to use Jupyter. We need to stop that. Type:

    ```
    unset PYSPARK_PYTHON_DRIVER
    unset PYSPARK_DRIVER_PYTHON_OPTS
    ```

    Note that this will reset when you start a new terminal window, so when you want to go back to using Jupyter, just kill this window and start another.

40. We run jobs locally on a single node directly on Spark:
    The local[*] indicates to use as many threads as you have cores on your system:
    ```
    ~/spark/bin/spark-submit --master local[*] wc-job.py
    "hdfs://localhost:54310/user/oxclo/books/*"
    ```

41. Congratulations, the lab is complete!

Extension

42. Re load the code into the Jupyter notebook and now improve it to show the wordcount in descending order, starting with the most common words.

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/