

# Exercise 5

*Get started with Apache Spark and Python*

## Prior Knowledge

Unix Command Line Shell

Simple Python

## Learning Objectives

Understand the Spark system

Use the Spark Python shell to interactively work with data

Submit Spark jobs locally and using YARN

Write SparkSQL code in Python

WordCount!

## Software Requirements

(see separate document for installation of these)

- Apache Spark 1.5.1
- Python 2.7.x
- Nano text editor or other text editor

## 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 its 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. Since we have our data in HDFS, *we need to ensure HDFS is running*. (Follow the instructions from the Hadoop lab).
3. Let's load some books into HDFS. In a terminal window (Ctrl-Alt-T)

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

4. Now, change to the Spark directory:  
`cd ~/spark-1.5.1`



© Paul Fremantle 2015. Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.  
See <http://creativecommons.org/licenses/by-sa/3.0/>

5. Now start the Spark Python command line tool – pyspark  
bin/pyspark

- a. You should see a lot of log come up, ending in something like:

```
15/10/25 23:39:52 INFO BlockManagerMaster: Registered BlockManager  
Welcome to
```



version 1.5.1

```
Using Python version 2.7.6 (default, Jun 22 2015 17:58:13)  
SparkContext available as sc, SQLContext available as sqlContext.
```

6. Now let's load some data. We already have a SparkContext object defined in the shell (in a program you need to define one, which we will see later)

7. Unfortunately some of the input is handled as Unicode by Python and we want to get rid of that. So let's start:

```
import unicodedata
```

8. Then type (on one line):

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

9. We also want to remove any non-alphanumeric characters:  
`def strip(s): return ''.join(filter(str.isalpha, s))`

10. Now we would like to load the books from HDFS. Now let's load some data.

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

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

11. Let's split the lines into words:

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

12. Now let's transform from Unicode to ascii

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

13. And remove non-alpha characters

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



© Paul Fremantle 2015. Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.  
See <http://creativecommons.org/licenses/by-sa/3.0/>

14. And we should put everything to lower case while we are cleaning it up

```
lower = stripped.map(lambda a: a.lower())
```

15. Finally we are ready to do the classic “WordCount” Map Reduce.

We first create a simple <K,V> pair of <word, count>. In the map phase, the count is always 1, since we haven’t yet reduced this.

```
numbered = lower.map(lambda word: (word, 1))
```

16. Next we reduce by adding the counts together for the same words:

```
wordcount = numbered.reduceByKey(lambda a,b: a+b)
```

17. 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
```

18. You should see a lot of word counts go flying past.

19. Congratulations!

20. While the pyspark is still running browse to <http://localhost:4040>

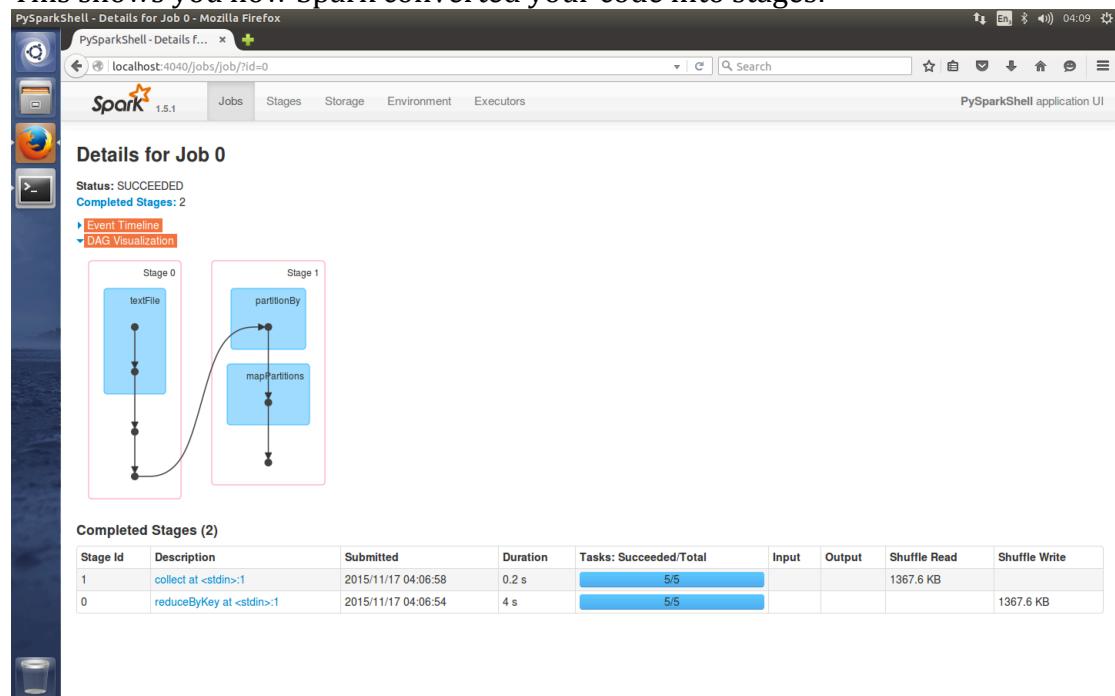
21. You will see the Spark web console:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <stdin>:1	2015/11/17 04:06:54	4 s	2/2	10/10

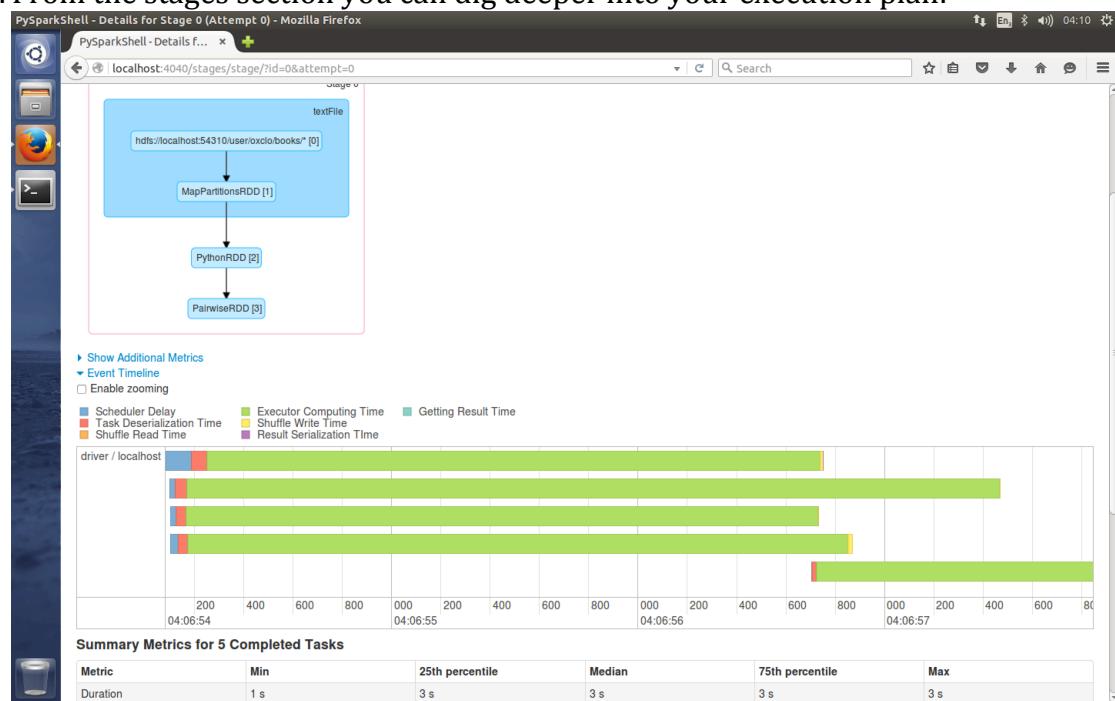


22. Click on the blue link “collect at stdin”

This shows you how Spark converted your code into stages:



23. From the stages section you can dig deeper into your execution plan:



24. Quit the pyspark shell by typing `quit()`



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

26. Make a directory for your spark python code:

```
mkdir ~/pysp  
cd ~/pysp
```

27. From <http://freo.me/oxclo-wc-py> copy the code into a file wc.py

28. Now configure the correct setup so Spark can find the Yarn system:

```
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop/
```

29. Now submit the job via YARN:

```
~/spark-1.5.1/bin/spark-submit --master yarn-cluster wc.py \  
"hdfs://localhost:54310/user/oxclo/books/*"
```

30. You can see the status of the job via the YARN webpage:

<http://localhost:8088>

31. You should also see a success via the command line:

```
oxclo@oxclo: ~/pysp  
3511_0001 (state: RUNNING)  
15/11/17 04:20:28 INFO yarn.Client: Application report for application_144771728  
3511_0001 (state: RUNNING)  
15/11/17 04:20:29 INFO yarn.Client: Application report for application_144771728  
3511_0001 (state: RUNNING)  
15/11/17 04:20:30 INFO yarn.Client: Application report for application_144771728  
3511_0001 (state: FINISHED)  
15/11/17 04:20:30 INFO yarn.Client:  
    client token: N/A  
    diagnostics: N/A  
    ApplicationMaster host: 172.16.64.178  
    ApplicationMaster RPC port: 0  
    queue: default  
    start time: 1447733991307  
    final status: SUCCEEDED  
    tracking URL: http://oxclo:8088/proxy/application_1447717288511_0001/  
    user: oxclo  
15/11/17 04:20:30 INFO util.ShutdownHookManager: Shutdown hook called  
15/11/17 04:20:30 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-5  
9dabdb1-d7d5-4bbe-8460-e3e57e868383
```



32. In the YARN web console, click on the FINISHED link in the left hand menu. You should see:

The screenshot shows the 'FINISHED Applications' section of the YARN web interface. On the left, a sidebar lists cluster metrics: About, Nodes, Node Labels, Applications (with sub-options NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED), Scheduler, and Tools. The main area displays 'Cluster Metrics' and 'Scheduler Metrics'. A table titled 'FINISHED Applications' lists one entry:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress
application_1447717288511_0001	oxclo	wc.py	SPARK	default	Tue Nov 17 04:19:51 +0000 2015	Tue Nov 17 04:20:30 +0000 2015	FINISHED	SUCCEEDED	

Showing 1 to 1 of 1 entries

33. Click on the application link and then click on the Logs link.

34. You can now find the stdout log from your code which has the result of the MR job:

The screenshot shows the 'Logs for container\_1447717288511\_0001\_01\_000001' page. On the left, a sidebar lists ResourceManager (RM Home), NodeManager, and Tools. The main area displays a log of words and their counts:

```

Showing 4096 bytes. Click here for full log
stories 2
organic 1
g 1
Skewtons 20
occasionalteeming 1
goldentinted 1
biting 20
precious 4
tacked 1
sportsmans 1
signboard 2
echo 2
bonnet 110
intuitively 1
aloneincurring 1
twas 3
probation 6
Prussias 1
laughingly 16
salutary 4
chancery 6
publication 2
copybook 2
unknown 121
situationwhich 1
innate 5
ticklish 1
Louisewho 1
bully 4
Midshipsmans 14
Success 4
slidest 3
academy 2
windermountains 2
genuineness 1
enoughit 1
stairhead 2
marrow 1
persecutor 1
noses 10
elope 5
waterdrypraying 1
plumbers 1
swooned 3
rarity 1
patients 12
Italiian 6
creditors 28
homefor 1

```



35. You can also run jobs locally on a single node without using YARN:  
The local[\*] indicates to use as many threads as you have cores on your system:  
`~/spark-1.5.1/bin/spark-submit --master local[*] wc.py  
"hdfs://localhost:54310/user/oxclo/books/*"`

36. You could also use Spark's own cluster manager or Apache Mesos as other options if you have those set up.

37. Congratulations, the lab is complete!



© Paul Fremantle 2015. Licensed under the Creative Commons 3.0 BY-SA (Attribution-Sharealike) license.  
See <http://creativecommons.org/licenses/by-sa/3.0/>