<h1 align = "center"> Spark Fundamentals 1 - Introduction to Spark</h1>

<h2 align = "center"> Getting Started</h2>

<h4 align = "center"> January 11, 2016 </h4> <br align = "left">

Welcome to Spark Fundamentals - Introduction to Spark. Spark is built around speed and the ease of use. In these labs you will see for yourself how easy it is to get started using Spark.

Spark's primary abstraction is a distributed collection of items called a Resilient Distributed Dataset or RDD. In a subsequent lab exercise, you will learn more about the details of RDD. RDDs have actions, which return values, and transformations, which return pointers to new RDD.

This set of labs uses Data Scientist Workbench to provide an interactive environment to develop applications and analyze data. It is available in either Scala or Python shells. Scala runs on the Java VM and is thus a good way to use existing Java libraries. In this lab exercise, we will set up our environment in preparation for the later labs. After completing this set of hands-on labs, you should be able to:

o Start the Spark shell with Scala and Python

o Perform basic RDD actions and transformations

o Use caching to speed up repeated operations

Using this notebook - Please ensure you have viewed the Data Scientist Workbench tutorial on the Big Data University before proceeding.

This is an interactive environment where you can show your code through cells, and documentation through markdown.

Look at the top right corner. Do you see "Python 2"? This indicates that you are running Python in this notebook.

To run a cell: Ctrl + Enter

To run a cell and go to the next cell: Shift + Enter

Try creating a new cell below.

To create a new cell: In the menu, go to "*Insert*" > "*Insert Cell Below*". Or, click outside of a cell, and press "a" (insert cell above) or "b" (insert cell below).

Lab Setup

Run the following cells to get the lab data.

In [1]:
```
!wget https://ibm.box.com/shared/static/1c65hfqjxyxpdkts42oab8i8mzxbpvc8.zip
```

```
--2016-09-03 18:07:24--  https://ibm.box.com/shared/static/1c65hfqjxyxpdkts42
oab8i8mzxbpvc8.zip
Resolving ibm.box.com (ibm.box.com)... 107.152.25.197, 74.112.184.85, 74.112.
185.182, ...
Connecting to ibm.box.com (ibm.box.com)|107.152.25.197|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ibm.ent.box.com/shared/static/1c65hfqjxyxpdkts42oab8i8mzxbp
vc8.zip [following]
--2016-09-03 18:07:24--  https://ibm.ent.box.com/shared/static/1c65hfqjxyxpdk
ts42oab8i8mzxbpvc8.zip
Resolving ibm.ent.box.com (ibm.ent.box.com)... 74.112.185.69, 107.152.25.211,
 74.112.184.69, ...
Connecting to ibm.ent.box.com (ibm.ent.box.com)|74.112.185.69|:443... connect
ed.
HTTP request sent, awaiting response... 302 Found
Location: https://public.boxcloud.com/d/1/LPbhFgndyaWkd6JWGf_QZD-bjb4W7ioNbeT
HJlwianb51J8RRNbY_ga8Iv6YHY1tHRSQxl261WzRNUhlHDEJkMvzH15eFBsFAdhFiiP1it7MXwSo
KIv8EGG3qf-U6PjIXwTTwLxf0DXH8J_EBiE-RShCg_YVCggwXkG4RmVf2BZYAP-7h3q2aWkNlsVxv
P0VUjJL7NCeFMgIsd33rYUmn32CJBcNQv6gEvM3QUc-JVkhmpTF-V28t-Qv9tJewWybmWz4xAD9JU
tsrz3v45r5yQgAmitGlHQoLwuLXW8drs8BAyAeDR-77DjElrh9KmALHpAKIs6rgWDCd1PeSeNu9oP
pxdEvDNHeQNSsyEoJ-NcpEsK5Zu4aoCOT312ENRHi5xroJ_KAqrK6eP-xmI7j8MWM0u7Uxv38Pth0
kfFjSXDbvYaVLO2HA1EDyvegyvjKeTxbXQDx1qKlmBjigyOXScLSt9xzp91dL79x0rW3NlC-57vZE
LD0LJ5JE6_u5pZuOzLk4qWo5uZf8SseDyhFSSRPq1mdnPcouNKG0mw5xUVysmRJxOFbFATDrhPqVY
bOwFU7Z6HPWPs9mjQajZsjp_uNZkNoIePeRCYPm7QwsZi7-YK2CirfxmDw4pww21_oTCLp_UdFucF
rRLNRZM-BdifnhAc8pDe5ACiG9KUlMCZw6tdbVzRKneQnvZVKajfxLIRzPJX9rkZRRbcJU2HUmVoB
hkvGntE6YkzXxUmudUshi4hgVxPJmx7WTAoV4JOFF4Zs4EKNdPf2D05d3yV8B4mMhbLTBbIGahQ7T
zpjGF3vstht9Ssd-7sT_yKABVya7BmDSxoOfg5nETVHg_V0BYNHeJCrB5481RZbrQKdnTaSiDEqvO
C-Auj-RRI0GbCR2b27hYJMQKd6eORHEn3I3a7R7B8u1nRb7MRbWXI4naB2dMQyV-gpzWdmLkPYoc2
58b4ZtzWBj9XDztg86gkr3teP1JMe9eF2409CxeHY_YG4zE4ZBNhr0LkD0K9ZbtTqykHf53l7Q14i
lH1ITMF-m9HyJMbppeJfI-K0_Eu6vmpqDe43I-uekjmfKKkZuSKZYjJqsXsiqj-4p1wIvyX1T6feH
5zyjRJf7Dq0OBOfHSeHpWUWUEUB0HN4kNoN1shshBO-zrLHBxFIs97L-kYSL8rN/download [fol
lowing]
--2016-09-03 18:07:25--  https://public.boxcloud.com/d/1/LPbhFgndyaWkd6JWGf_Q
ZD-bjb4W7ioNbeTHJlwianb51J8RRNbY_ga8Iv6YHY1tHRSQxl261WzRNUhlHDEJkMvzH15eFBsFA
dhFiiP1it7MXwSoKIv8EGG3qf-U6PjIXwTTwLxf0DXH8J_EBiE-RShCg_YVCggwXkG4RmVf2BZYAP
-7h3q2aWkNlsVxvP0VUjJL7NCeFMgIsd33rYUmn32CJBcNQv6gEvM3QUc-JVkhmpTF-V28t-Qv9tJ
ewWybmWz4xAD9JUtsrz3v45r5yQgAmitGlHQoLwuLXW8drs8BAyAeDR-77DjElrh9KmALHpAKIs6r
gWDCd1PeSeNu9oPpxdEvDNHeQNSsyEoJ-NcpEsK5Zu4aoCOT312ENRHi5xroJ_KAqrK6eP-xmI7j8
MWM0u7Uxv38Pth0kfFjSXDbvYaVLO2HA1EDyvegyvjKeTxbXQDx1qKlmBjigyOXScLSt9xzp91dL7
9x0rW3NlC-57vZELD0LJ5JE6_u5pZuOzLk4qWo5uZf8SseDyhFSSRPq1mdnPcouNKG0mw5xUVysmR
JxOFbFATDrhPqVYbOwFU7Z6HPWPs9mjQajZsjp_uNZkNoIePeRCYPm7QwsZi7-YK2CirfxmDw4pww
21_oTCLp_UdFucFrRLNRZM-BdifnhAc8pDe5ACiG9KUlMCZw6tdbVzRKneQnvZVKajfxLIRzPJX9r
kZRRbcJU2HUmVoBhkvGntE6YkzXxUmudUshi4hgVxPJmx7WTAoV4JOFF4Zs4EKNdPf2D05d3yV8B4
mMhbLTBbIGahQ7TzpjGF3vstht9Ssd-7sT_yKABVya7BmDSxoOfg5nETVHg_V0BYNHeJCrB5481RZ
brQKdnTaSiDEqvOC-Auj-RRI0GbCR2b27hYJMQKd6eORHEn3I3a7R7B8u1nRb7MRbWXI4naB2dMQy
V-gpzWdmLkPYoc258b4ZtzWBj9XDztg86gkr3teP1JMe9eF2409CxeHY_YG4zE4ZBNhr0LkD0K9Zb
tTqykHf53l7Q14ilH1ITMF-m9HyJMbppeJfI-K0_Eu6vmpqDe43I-uekjmfKKkZuSKZYjJqsXsiqj
-4p1wIvyX1T6feH5zyjRJf7Dq0OBOfHSeHpWUWUEUB0HN4kNoN1shshBO-zrLHBxFIs97L-kYSL8r
N/download
Resolving public.boxcloud.com (public.boxcloud.com)... 107.152.25.200, 107.15
2.24.200, 74.112.185.96, ...
Connecting to public.boxcloud.com (public.boxcloud.com)|107.152.25.200|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 342130521 (326M) [application/zip]
Saving to: ‘1c65hfqjxyxpdkts42oab8i8mzxbpvc8.zip.2’

100%[======================================>] 342,130,521 4.74MB/s   in 64s


2016-09-03 18:08:30 (5.07 MB/s) - ‘1c65hfqjxyxpdkts42oab8i8mzxbpvc8.zip.2’ sa
ved [342130521/342130521]
```

In [2]:
```
!unzip -o -d /resources 1c65hfqjxyxpdkts42oab8i8mzxbpvc8.zip
```

```
Archive:  1c65hfqjxyxpdkts42oab8i8mzxbpvc8.zip
  inflating: /resources/LabData/.DS_Store
  inflating: /resources/__MACOSX/LabData/._.DS_Store
  inflating: /resources/LabData/followers.txt
  inflating: /resources/__MACOSX/LabData/._followers.txt
  inflating: /resources/LabData/notebook.log
  inflating: /resources/__MACOSX/LabData/._notebook.log
  inflating: /resources/LabData/nyctaxi.csv
  inflating: /resources/__MACOSX/LabData/._nyctaxi.csv
  inflating: /resources/LabData/nyctaxi100.csv
  inflating: /resources/__MACOSX/LabData/._nyctaxi100.csv
  inflating: /resources/LabData/nyctaxisub.csv
  inflating: /resources/__MACOSX/LabData/._nyctaxisub.csv
  inflating: /resources/LabData/nycweather.csv
  inflating: /resources/__MACOSX/LabData/._nycweather.csv
  inflating: /resources/LabData/pom.xml
  inflating: /resources/__MACOSX/LabData/._pom.xml
  inflating: /resources/LabData/README.md
  inflating: /resources/__MACOSX/LabData/._README.md
  inflating: /resources/LabData/taxistreams.py
  inflating: /resources/__MACOSX/LabData/._taxistreams.py
  inflating: /resources/LabData/users.txt
  inflating: /resources/__MACOSX/LabData/._users.txt
```

In [3]:
```
!ls -1 /resources/LabData/
```

```
followers.txt
notebook.log
nyctaxi100.csv
nyctaxi.csv
nyctaxisub.csv
nycweather.csv
pom.xml
README.md
taxistreams.py
users.txt
```

Should have:

- followers.txt
- notebook.log
- nyctaxi100.csv
- nyctaxi.csv
- nyctaxisub.csv
- nycweather.csv
- pom.xml
- README.md
- taxistreams.py
- users.txt

Starting with Spark

The notebooks provide code assist. For example, type in "sc." followed by the Tab key to get the list of options associated with the spark context:

```
In [4]: sc.accumulator
```

```
Out[4]: <bound method SparkContext.accumulator of <pyspark.context.SparkContext objec
        t at 0x7fb3a01a8e10>>
```

To run a command as code, simple select the cell you want to run and either:

- Click the play button in the toolbar above
- Press "*Shift*+*Enter*"

Let's run a basic command and check the version of Spark running:

```
In [ ]: sc.version
```

You can get files into your workbench in three ways:

1. Drag and drop a file from your file explorer onto the browser. This will upload the file to your workbench.
2. Enter the url of a file on the internet into the text field in the upper right of the screen.
3. Run code (such as wget) to download a file into your notebook.

Download the following file by pasting the link in the search field at the top right of the page and pressing ENTER:

https://raw.githubusercontent.com/apache/spark/master/README.md
(https://raw.githubusercontent.com/apache/spark/master/README.md)

You should see the file show up in *Recent Data*. Since this file is contained in the zip file you should see it named as README0.md. You can delete this file now by clicking the twistie on README0.md and choosing delete.

Highlight the text string between the double quotes in the cell below then, click the twistie next to the "README.md" in *Recent Data* and select *Insert Path* to paste the path below. Then, run the code in the cell.

```
In [5]: readme = sc.textFile("/resources/LabData/README.md")
```

Let's perform some RDD actions on this text file. Count the number of items in the RDD using this command:

```
In [6]: readme.count()
```

```
Out[6]: 98
```

You should see that this RDD action returned a value of 98.

Let's run another action. Run this command to find the first item in the RDD:

```
In [7]: readme.first()
```

```
Out[7]: u'# Apache Spark'
```

Now let's try a transformation. Use the filter transformation to return a new RDD with a subset of the items in the file. Type in this command:

```
In [8]: linesWithSpark = readme.filter(lambda line: "Spark" in line)
```

You can even chain together transformations and actions. To find out how many lines contains the word “Spark”, type in:

```
In [9]: linesWithSpark = readme.filter(lambda line: "Spark" in line)
        readme.filter(lambda line: "Spark" in line).count()
```

```
Out[9]: 18
```

More on RDD Operations

This section builds upon the previous section. In this section, you will see that RDD can be used for more complex computations. You will find the line from that readme file with the most words in it.

Run the following cell.

```
In [10]: readme.map(lambda line: len(line.split())).reduce(lambda a, b: a if (a > b) el
         se b)
```

```
Out[10]: 14
```

There are two parts to this. The first maps a line to an integer value, the number of words in that line. In the second part reduce is called to find the line with the most words in it. The arguments to map and reduce are Python anonymous functions (lambdas), but you can use any top level Python functions. In the next step, you’ll define a max function to illustrate this feature.

Define the max function. You will need to type this in:

```
In [11]: def max(a, b):
          if a > b:
             return a
          else:
             return b
```

Now run the following with the max function:

```
In [12]: readme.map(lambda line: len(line.split())).reduce(max)
```

```
Out[12]: 14
```

Spark has a MapReduce data flow pattern. We can use this to do a word count on the readme file.

```
In [13]: wordCounts = readme.flatMap(lambda line: line.split()).map(lambda word: (word,
          1)).reduceByKey(lambda a, b: a+b)
```

Here we combined the flatMap, map, and the reduceByKey functions to do a word count of each word in the readme file.

To collect the word counts, use the collect action.

It should be noted that the collect function brings all of the data into the driver node. For a small dataset, this is acceptable but, for a large dataset this can cause an Out Of Memory error. It is recommended to use collect() for testing only. The safer approach is to use the take() function e.g. print take(n)

In [14]:
```
wordCounts.collect()
```

Out[14]:

```
[(u'when', 1),
 (u'R,', 1),
 (u'including', 3),
 (u'computation', 1),
 (u'using:', 1),
 (u'guidance', 3),
 (u'Scala,', 1),
 (u'environment', 1),
 (u'only', 1),
 (u'rich', 1),
 (u'Apache', 1),
 (u'sc.parallelize(range(1000)).count()', 1),
 (u'Building', 1),
 (u'guide,', 1),
 (u'return', 2),
 (u'Please', 3),
 (u'Try', 1),
 (u'not', 1),
 (u'Spark', 14),
 (u'scala>', 1),
 (u'Note', 1),
 (u'cluster.', 1),
 (u'./bin/pyspark', 1),
 (u'have', 1),
 (u'params', 1),
 (u'through', 1),
 (u'GraphX', 1),
 (u'[run', 1),
 (u'abbreviated', 1),
 (u'[project', 2),
 (u'##', 8),
 (u'library', 1),
 (u'see', 1),
 (u'"local"', 1),
 (u'[Apache', 1),
 (u'will', 1),
 (u'#', 1),
 (u'processing,', 1),
 (u'for', 12),
 (u'[building', 1),
 (u'provides', 1),
 (u'print', 1),
 (u'supports', 2),
 (u'built,', 1),
 (u'[params]`.', 1),
 (u'available', 1),
 (u'run', 7),
 (u'tests](https://cwiki.apache.org/confluence/display/SPARK/Useful+Developer
+Tools).',
  1),
 (u'This', 2),
 (u'Hadoop,', 2),
 (u'Tests', 1),
 (u'example:', 1),
 (u'-DskipTests', 1),
 (u'Maven](http://maven.apache.org/).', 1),
 (u'programming', 1),
 (u'running', 1),
 (u'against', 1),
 (u'site,', 1),
 (u'comes', 1),
 (u'package.', 1),
 (u'and', 10),
```

```
 (u'package.)', 1),
 (u'prefer', 1),
 (u'documentation,', 1),
 (u'submit', 1),
 (u'tools', 1),
 (u'use', 3),
 (u'from', 1),
 (u'For', 2),
 (u'fast', 1),
 (u'systems.', 1),
 (u'<http://spark.apache.org/>', 1),
 (u'Hadoop-supported', 1),
 (u'way', 1),
 (u'README', 1),
 (u'MASTER', 1),
 (u'engine', 1),
 (u'building', 3),
 (u'usage', 1),
 (u'Distributions"](http://spark.apache.org/docs/latest/hadoop-third-party-di
stributions.html)',
  1),
 (u'instance:', 1),
 (u'with', 4),
 (u'protocols', 1),
 (u'And', 1),
 (u'this', 1),
 (u'setup', 1),
 (u'shell:', 2),
 (u'project', 1),
 (u'See', 1),
 (u'following', 2),
 (u'distribution', 1),
 (u'detailed', 2),
 (u'file', 1),
 (u'stream', 1),
 (u'is', 6),
 (u'higher-level', 1),
 (u'tests', 2),
 (u'1000:', 2),
 (u'sample', 1),
 (u'["Specifying', 1),
 (u'Alternatively,', 1),
 (u'./bin/run-example', 2),
 (u'need', 1),
 (u'You', 3),
 (u'instructions.', 1),
 (u'different', 1),
 (u'programs,', 1),
 (u'storage', 1),
 (u'same', 1),
 (u'machine', 1),
 (u'Running', 1),
 (u'which', 2),
 (u'you', 4),
 (u'A', 1),
 (u'About', 1),
 (u'sc.parallelize(1', 1),
 (u'locally.', 1),
 (u'Hive', 2),
 (u'optimized', 1),
 (u'uses', 1),
 (u'Version"](http://spark.apache.org/docs/latest/building-spark.html#specify
ing-the-hadoop-version)',
```

```
 1),
(u'variable', 1),
(u'The', 1),
(u'data', 1),
(u'a', 10),
(u'"yarn"', 1),
(u'Thriftserver', 1),
(u'processing.', 1),
(u'./bin/spark-shell', 1),
(u'Python', 2),
(u'Spark](#building-spark).', 1),
(u'clean', 1),
(u'the', 21),
(u'requires', 1),
(u'talk', 1),
(u'help', 1),
(u'Hadoop', 4),
(u'using', 2),
(u'high-level', 1),
(u'find', 1),
(u'web', 1),
(u'Shell', 2),
(u'how', 2),
(u'graph', 1),
(u'run:', 1),
(u'should', 2),
(u'to', 14),
(u'module,', 1),
(u'given.', 1),
(u'directory.', 1),
(u'must', 1),
(u'do', 2),
(u'Programs', 1),
(u'Many', 1),
(u'YARN,', 1),
(u'["Third', 1),
(u'Example', 1),
(u'Once', 1),
(u'Spark"](http://spark.apache.org/docs/latest/building-spark.html).', 1),
(u'Because', 1),
(u'name', 1),
(u'Testing', 1),
(u'refer', 2),
(u'Streaming', 1),
(u'SQL', 2),
(u'them,', 1),
(u'analysis.', 1),
(u'application', 1),
(u'set', 2),
(u'Scala', 2),
(u'thread,', 1),
(u'individual', 1),
(u'examples', 2),
(u'changed', 1),
(u'runs.', 1),
(u'Pi', 1),
(u'More', 1),
(u'Python,', 2),
(u'Versions', 1),
(u'its', 1),
(u'version', 1),
(u'wiki](https://cwiki.apache.org/confluence/display/SPARK).', 1),
(u'`./bin/run-example', 1),
```

```
(u'Configuration', 1),
(u'command,', 2),
(u'can', 6),
(u'core', 1),
(u'Guide](http://spark.apache.org/docs/latest/configuration.html)', 1),
(u'MASTER=spark://host:7077', 1),
(u'Documentation', 1),
(u'downloaded', 1),
(u'distributions.', 1),
(u'Spark.', 1),
(u'["Building', 1),
(u'`examples`', 2),
(u'on', 6),
(u'works', 1),
(u'package', 1),
(u'of', 5),
(u'APIs', 1),
(u'pre-built', 1),
(u'Big', 1),
(u'or', 3),
(u'learning,', 1),
(u'locally', 2),
(u'overview', 1),
(u'one', 2),
(u'(You', 1),
(u'Online', 1),
(u'versions', 1),
(u'your', 1),
(u'threads.', 1),
(u'>>>', 1),
(u'SparkPi', 2),
(u'contains', 1),
(u'system', 1),
(u'class', 2),
(u'start', 1),
(u'build/mvn', 1),
(u'basic', 1),
(u'configure', 1),
(u'that', 3),
(u'N', 1),
(u'"local[N]"', 1),
(u'DataFrames,', 1),
(u'particular', 3),
(u'be', 2),
(u'an', 3),
(u'easiest', 1),
(u'Interactive', 2),
(u'cluster', 2),
(u'page](http://spark.apache.org/documentation.html)', 1),
(u'<class>', 1),
(u'example', 3),
(u'are', 1),
(u'Data.', 1),
(u'mesos://', 1),
(u'computing', 1),
(u'URL,', 1),
(u'in', 5),
(u'general', 2),
(u'To', 2),
(u'at', 2),
(u'1000).count()', 1),
(u'Party', 1),
(u'if', 4),
```

```
 (u'built', 1),
 (u'no', 1),
 (u'Java,', 1),
 (u'MLlib', 1),
 (u'also', 5),
 (u'other', 1),
 (u'build', 3),
 (u'online', 1),
 (u'several', 1),
 (u'distribution.', 1),
 (u'HDFS', 1),
 (u'[Configuration', 1),
 (u'spark://', 1),
 (u'programs', 2),
 (u'documentation', 3),
 (u'It', 2),
 (u'graphs', 1),
 (u'./dev/run-tests', 1),
 (u'first', 1),
 (u'latest', 1)]
```

YOUR TURN:

In the cell below, determine what is the most frequent word in the README, and how many times was it used?

In [17]:
```
#YOUR CODE BELOW
#wordCounts.map(lambda (k,v):(v,k)).sortByKey(False).take(2)
wordCounts.reduce(lambda a,b : a if(a[1]>b[1]) else b)
```

Out[17]:
```
(u'the', 21)
```

Highlight text field for answer:

wordCounts.reduce(lambda a, b: a if (a[1] > b[1]) else b)

Using Spark caching

In this short section, you'll see how Spark caching can be used to pull data sets into a cluster-wide in- memory cache. This is very useful for accessing repeated data, such as querying a small "hot" dataset or when running an iterative algorithm. Both Python and Scala use the same commands.

As a simple example, let's mark our linesWithSpark dataset to be cached and then invoke the first count operation to tell Spark to cache it. Remember that transformation operations such as cache does not get processed until some action like count() is called. Once you run the second count() operation, you should notice a small increase in speed.

In [18]:
```
print linesWithSpark.count()
```

```
18
```

In [19]:
```
from timeit import Timer
def count():
    return linesWithSpark.count()
t = Timer(lambda: count())
```

In [20]:
```
print t.timeit(number=50)
```

```
4.01096510887
```

In [21]:
```
linesWithSpark.cache()
print t.timeit(number=50)
```

```
3.65016484261
```

It may seem silly to cache such a small file, but for larger data sets across tens or hundreds of nodes, this would still work. The second linesWithSpark.count() action runs against the cache and would perform significantly better for large datasets.

Summary

Having completed this exercise, you should now be able to log in to your environment and use the Spark shell to run simple actions and transformations for Scala and/or Python. You understand that Spark caching can be used to cache large datasets and subsequent operations on it will utilize the data in the cache rather than re-fetching it from HDFS.