I. Overview

Let's start by using an example data science analysis scenario. Do you know the company Uber? We're going to analyze New York City Uber data in this post first. Then, we'll use this example to describe Spark fundamental concepts.

Requirements

* Spark instance in Standalone or Cluster mode (more below)

* Download aggregated NYC Uber trip data in CSV format

from: https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/master/Uber-Jan-Feb-FOIL.csv

I. Let's Run Some Code

The CSV file we will use has following structure:

```
dispatching_base_number,date,active_vehicles,trips
B02512,1/1/2015,190,1132
B02765,1/1/2015,225,1765
B02764,1/1/2015,3427,29421
B02682,1/1/2015,945,7679
```

dispatching_base_number is the NYC Taxi and Limousine company code of the base that dispatched the Uber. active_vehicles shows the number of active Uber vehicles for a particular date and company (base). Trips is the number of trips for a particular base and date.

With this data, we can answer questions such as: what was the busiest dispatch base by trips for a particular day or entire month? what day had the most active vehicles? what days had the most trips sorted by most to fewest? etc.

For more information see https://github.com/fivethirtyeight/uber-tlc-foil-response

Steps

1. Download from http://spark.apache.org/downloads.html. Select the "Pre-built package for Hadoop 2.4" if you haven't already and unpack it. (See Reference section below if you need help installing Spark.)
2. From terminal in Spark home directory , run the Python Spark shell: bin/pyspark

Let's run some code

```
>>> ut = sc.textFile("Uber-Jan-Feb-FOIL.csv")
>>> ut.count()
355
>>> ut.first()
u'dispatching_base_number,date,active_vehicles,trips'
```

So, we know there are 355 rows in the CSV

```
>>> rows = ut.map(lambda line: line.split(","))
>>> rows.map(lambda row: row[0]).distinct().count()
7
```

In above, the Python code converted the CSV to a Resilient Distributed Dataset (RDD) by splitting each row in the source CSV file by a comma. More on RDDs later. Then, we used a Spark Transformation *distinct* and a Spark Action *count* to determine there are 7 unique values in the first column in the CSV. Again, more on Spark Transformations and Actions later in this post.

```
>>> rows.map(lambda row: row[0]).distinct().collect()
[u'B02617', u'B02682', u'B02598', u'B02765', u'B02512',
u'dispatching_base_number', u'B02764']
>>> rows.filter(lambda row: "B02617" in row).count()
59
```

There are 59 rows containing the trip data for TLC base company code "B02617".

```
>>> base02617 = rows.filter(lambda row: "B02617" in row)
>>> base02617.filter(lambda row: int(row[3]) > 15000).count()
6
```

Number of rows where base02617 had more than 15000 trips in a day: 6. Or, I should say this daily ratio is assumed. Let's confirm

```
>>> base02617.filter(lambda row: int(row[3]) > 15000).map(lambda day:
day[1]).distinct().count()
6
```

Yes, it's confirmed. Let's keep going…

```
>>> filteredRows = sc.textFile("Uber-Jan-Feb-FOIL.csv").filter(lambda
line: "base" not in line).map(lambda line:line.split(","))
>>> filteredRows.map(lambda kp: (kp[0], int(kp[3])) ).reduceByKey(lambda
k,v: k + v).collect()
[(u'B02617', 725025), (u'B02682', 662509), (u'B02598', 540791),
(u'B02765', 193670), (u'B02512', 93786), (u'B02764', 1914449)]
```

So, we see the number of trips per base station. But, it's difficult to determine which base was busiest over the time frame in CSV? Let's make it easier to see:

```
>>> filteredRows.map(lambda kp: (kp[0], int(kp[3])) ).reduceByKey(lambda
k,v: k + v).takeOrdered(10, key=lambda x: -x[1])
[(u'B02764', 1914449), (u'B02617', 725025), (u'B02682', 662509),
(u'B02598', 540791), (u'B02765', 193670), (u'B02512', 93786)]
```

So, base B02764 was busiest by trip… by over 1 milion.

II. Next Steps

The remainder of this post will cover Spark Core concepts. Spark Core is what makes all other aspects of the Spark ecosystem possible including Spark SQL, Spark Streaming, MLLib.

III. Spark Context and Resilient Distributed Datasets

The way to interact with Spark is via a SparkContext. The example used the PySpark Console which provides a SparkContext automatically. When you start pyspark, do you notice the last line ?

```
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

That's how we're able to use sc from within our example.

After obtaining a SparkContext, developers interact with Spark's primary data abstraction called Resilient Distributed Datasets.

Resilient Distributed Datasets (RDDs) are an immutable, distributed collection of elements. These collections may be parallelized across a cluster. As we witnessed, RDDs are loaded from an external data set or created via a SparkContext. We'll cover both of these scenarios.

We created an RDD by loading in a CSV file:

```
>>> ut = sc.textFile("Uber-Jan-Feb-FOIL.csv")
```

We also created RDDs through Spark Transformations, which we'll cover a bit later.

When utilizing Spark, you will be doing one of two primary interactions: creating new RDDs through transformations or using existing RDDs to compute a result such as distinct counts. The next section describes these two Spark interactions.

IV Actions and Transformations

When working with a Spark RDDs, there are two available operations: actions or transformations. An action is an execution which produces a result. Examples of actions in previous are count, first.

EXAMPLE SPARK ACTIONS IN PYTHON

```
ut.count() // number of lines in the CSV file
ut.first() // first line of CSV
```

EXAMPLE SPARK TRANSFORMATIONS IN PYTHON

Transformations create new RDDs using existing RDDs. We created a variety of RDDs in our example:

```
>>> rows = ut.map(lambda line: line.split(","))


>>> filteredRows = sc.textFile("Uber-Jan-Feb-FOIL.csv").filter(lambda
line: "base" not in line).map(lambda line:line.split(","))
```