Yandex

Getting started with Spark & Python

Installing Spark locally

- Navigate to http://spark.apache.org/docs/latest/#downloading and follow the instructions
- At the time of making this video
  - download .tar.gz
  - extract it
  - run ./bin/pyspark from the extracted directory
- If you have IPython installed, you can run PYSPARK_DRIVER_PYTHON=ipython pyspark

$ pyspark

```
Python 2.7.10 (default, Feb  6 2017, 23:53:20)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Python version 2.7.10 (default, Feb  6 2017 23:53:20)
SparkSession available as 'spark'.
>>>
```

$ pyspark

```
Python 2.7.10 (default, Feb  6 2017, 23:53:20)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Python version 2.7.10 (default, Feb  6 2017 23:53:20)
SparkSession available as 'spark'.
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>>
```

$ pyspark

```
...
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>>
```

$ pyspark

```
...
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> a
ParallelCollectionRDD[1] at parallelize at PythonRDD.scala:480
>>>
```

$ pyspark

```
...
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> a
ParallelCollectionRDD[1] at parallelize at PythonRDD.scala:480
>>> a.getNumPartitions()
4
>>>
```

$ pyspark

```
...
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> a
ParallelCollectionRDD[1] at parallelize at PythonRDD.scala:480
>>> a.getNumPartitions()
4
>>> a.collect()
[1, 2, 3, 4, 5]
>>>
```

$ pyspark

```
...
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> a
ParallelCollectionRDD[1] at parallelize at PythonRDD.scala:480
>>> a.getNumPartitions()
4
>>> a.collect()
[1, 2, 3, 4, 5]
>>> b = a.map(lambda x: 2 * x)
>>> b
PythonRDD[2] at RDD at PythonRDD.scala:48
>>>
```

$ pyspark

```
...
>>> sc
<SparkContext master=local[*] appName=PySparkShell>
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> a
ParallelCollectionRDD[1] at parallelize at PythonRDD.scala:480
>>> a.getNumPartitions()
4
>>> a.collect()
[1, 2, 3, 4, 5]
>>> b = a.map(lambda x: 2 * x)
>>> b
PythonRDD[2] at RDD at PythonRDD.scala:48
>>> b.collect()
[2, 4, 6, 8, 10]
>>>
```

$ pyspark

```
...
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>>
```

$ pyspark

```
...
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> b = a.map(lambda x: ( print(x), 2 * x )[1])
>>> b
PythonRDD[3] at RDD at PythonRDD.scala:48
>>>
```

AMAZING
FABULOUS
EXTRAORDINARY
SIDE-EFFECT!

$ pyspark

```
...
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> b = a.map(lambda x: ( print(x), 2 * x )[1])
>>> b
PythonRDD[3] at RDD at PythonRDD.scala:48
>>>
```

AMAZING
FABULOUS
EXTRAORDINARY
SIDE-EFFECT!

MYSTERIOUSLY
DISAPPEARED
DURING THE
TRANSFORMATION!

$ pyspark

```
...
>>> a = sc.parallelize([1, 2, 3, 4, 5])
>>> b = a.map(lambda x: ( print(x) , 2 * x )[1])
>>> b
PythonRDD[3] at RDD at PythonRDD.scala:48
>>> b.collect()
1
3
4
5
2
[2, 4, 6, 8, 10]
>>>
```

AND TRIUMPHALLY
RE-APPEARED
AFTER THE ACTION
INVOCATION!

Summary

- You have learned how to:
  - start a Pyspark shell
  - create RDDs from lists by using the 'parallelize' method
  - invoke the 'collect' action
  - apply the 'map' transform

BigDATAteam