Spark install [1]

In this tutorial we try to install and get started with Spark for use in later classes.
(https://spark.apache.org/docs/latest/quick-start.html)

0.Install spark

Install the latest version of Spark

https://spark.apache.org/downloads.html

A. Interactive Analysis with the Spark Shell - SCALA

```
./bin/spark-shell
```

Spark's primary abstraction is a distributed collection of items called a Dataset. Datasets can be created from Hadoop InputFormats (such as HDFS files) or by transforming other Datasets.

1.Read a file. What do the commands show?

Let's make a new Dataset from the text of the README file in the Spark source directory:

```
scala> val textFile = spark.read.textFile("README.md")
textFile: org.apache.spark.sql.Dataset[String] = [value: string]
```

You can get values from Dataset directly, by calling some actions, or transform the Dataset to get a new one. For more details, please read the *API doc*.

```
scala> textFile.count()

scala> textFile.first()
```

2.Do some basic filtering, use map-reduce. What did it do?

Now let's transform this Dataset into a new one. We call `filter` to return a new Dataset with a subset of the items in the file.

```
scala> val linesWithSpark = textFile.filter(line => line.contains("Spark"))
```

3.Chain with an action. What does each do?

```
scala> textFile.filter(line => line.contains("Spark")).count()
```

```
scala> textFile.map(line => line.split(" ").size).reduce((a, b) => if (a > b) a else
b)
```

We'll next use `Math.max()` function to make this code easier to understand:

```
scala> import java.lang.Math
import java.lang.Math

scala> textFile.map(line => line.split(" ").size).reduce((a, b) => Math.max(a, b))
```

4.use MapReduce. What does this cmd do? What is the result?

```
scala> val wordCounts = textFile.flatMap(line => line.split("
")).groupByKey(identity).count()
```

```
scala> wordCounts.collect()
```

5.use Caching

Spark also supports pulling data sets into a cluster-wide in-memory cache. This is very useful when data is accessed repeatedly, such as when querying a small “hot” dataset or when running an iterative algorithm like PageRank. As a simple example, let’s mark our `linesWithSpark` dataset to be cached:

```
scala> linesWithSpark.cache()
res7: linesWithSpark.type = [value: string]

scala> linesWithSpark.count()
res8: Long = 15

scala> linesWithSpark.count()
res9: Long = 15
```

It may seem silly to use Spark to explore and cache a 100-line text file. The interesting part is that these same functions can be used on very large data sets, even when they are striped across tens or hundreds of nodes.

B. Interactive Analysis with the Spark Shell - PYTHON

```
./bin/pyspark
```

Or if PySpark is installed with pip in your current environment:

```
pyspark
```

1.Read a file. What do the commands show?

Let's make a new Dataset from the text of the README file in the Spark source directory:

```
>>> textFile = spark.read.text("README.md")
```

```
>>> textFile.count()  # Number of rows in this DataFrame

>>> textFile.first()  # First row in this DataFrame
```

2.Do some basic filtering, use map-reduce. What did it do?

Now let's transform this Dataset into a new one. We call `filter` to return a new Dataset with a subset of the items in the file.

```
>>> linesWithSpark = textFile.filter(textFile.value.contains("Spark"))
```

3.Chain with an action. What does each do?

```
>>> textFile.filter(textFile.value.contains("Spark")).count()
```

```
>>> from pyspark.sql.functions import *
>>> textFile.select(size(split(textFile.value,
"\s+")).name("numWords")).agg(max(col("numWords"))).collect()
```

4.use MapReduce. What does this cmd do? What is the result?

```
>>> wordCounts = textFile.select(explode(split(textFile.value,
"\s+")).alias("word")).groupBy("word").count()

>>> wordCounts.collect()
```

5.use Caching

Spark also supports pulling data sets into a cluster-wide in-memory cache. This is very useful when data is accessed repeatedly, such as when querying a small "hot" dataset or when running an iterative algorithm like PageRank. As a simple example, let's mark our `linesWithSpark` dataset to be cached:

```
>>> linesWithSpark.cache()

>>> linesWithSpark.count()
```