Laboratory 1 - Introduction to Spark

Run a simple Spark job

In this exercise we will run a simple Spark program in three different ways. We will first run the program in a Jupyter notebook. Then we will run the same code using the pyspark shell. Finally we will run it from the command line.

Jupyter notebook

In [1]:
```
readFile = sc.textFile("/data/students/bigdata_internet/lab1/lab1_dataset.txt")
fields_rdd = readFile.map(lambda line: line.split(","))
value_rdd = fields_rdd.map(lambda l: int(l[1]))
value_sum = value_rdd.reduce(lambda v1, v2: v1 + v2)
print( "The sum is: " , value_sum)
```

```
[Stage 0:>                                                          (0 + 2) / 2]
The sum is:  46
```

Questions

Question 1:

Which value is printed by the print statement? What is the purpose of each line of code?

Answer:

The printed value is 46, and each line purpose is as follows:

- **"rdd = sc.textFile("/data/students/bigdata_internet/lab1/lab1_dataset.txt")" *(Transformation)***
  This line reads a text file named "lab1_dataset.txt" and creates an RDD named `rdd` where each file line becomes an element in the RDD.
- **"fields_rdd = rdd.map(lambda line: line.split(",")) *(Transformation)***
  It applies a transformation on each element (line) in the `rdd`, splitting each line by commas (`,`) and creating a new RDD named `fields_rdd` where each line is transformed into a list of values.
- **"value_rdd = fields_rdd.map(lambda l: int(l[1])) *(Transformation***
  This line transforms each element (list) in the `fields_rdd`, taking the second element (`l[1]`) after splitting by commas and converting it into an integer. It creates a new RDD named `value_rdd` containing these integer values.

- "value_sum = value_rdd.reduce(lambda v1, v2: v1 + v2) ***(Action)***
  This line performs a reduction operation on the `value_rdd` RDD, adding up all the integer values and storing the final sum in the variable `value_sum`.

- "print("The sum is: ", value_sum) ***(Action)***
  This line prints the text "The sum is:" followed by the actual sum calculated in the previous step (`value_sum`).

Question 2:

Where is the input file? On which file system?

Answer:

The input file is located on "/data/students/bigdata_internet/lab1/lab1_dataset.txt" refers to a file on the local file system of the computer where the PySpark application is running if the code is running in a local PySpark environment. In this exercise, the function is to read data from a distributed file system (HDFS) on Polito's servers.

Question 3:

What happens if you open a Pyspark (YARN) notebook? Which is the difference in the Cluster manager interface at https://bigdatalab.polito.it?

Answer:

By executing the code, it became evident that it took more time to show the results. One key difference between running the code using YARN and LOCAL mode lies in resource management. In this specific exercise, using the LOCAL mode expedites the process. In other words, LOCAL mode is ideal for small applications or development with small amount of data, offering faster and more efficient processing compared to utilizing cluster resources via YARN.

Execute in a pyspark shell

Questions

Question 1:

What does `--master local--` mean? And what about `--deploy-mode client--` ? Where spark executors and driver executed?

Answer:

By running the code on the 'jupyter.polito.it terminal', the same result, which is 46 is appeared."

The `--master local--` in PySpark means it runs LOCALLY, using the LOCAL computer resources. `--deploy-mode client--` places the driver and executors on the local computer where the code is starts to run.

Create a Spark script and run it from the command line

Questions

Question 1:

In which file system are located your script and the "/data/students/bigdata_internet/lab1/lab1_dataset.txt" files? Are they on the same file system?

Answer:

- **The Python Script:**

```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setAppName("My app")
sc = SparkContext(conf = conf)
rdd = sc.textFile("/data/students/bigdata_internet/lab1/lab1_dataset.txt")
fields_rdd = rdd.map(lambda line: line.split(","))
value_rdd = fields_rdd.map(lambda l: int(l[1]))
value_sum = value_rdd.reduce(lambda v1, v2: v1+v2)
print("The sum is:", value_sum)
```

- **result in terminal:**

```
s299165@jupyter-s299165:~$ spark-submit --master local --deploy-mode client 'test_lab1.py'
WARNING: User-defined SPARK_HOME (/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/lib/spark) overrides detected (/opt/cloudera/parcels/CDH/lib/spark).
WARNING: Running spark-class from user-defined location.
24/01/20 12:40:37 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
24/01/20 12:40:37 WARN util.Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
24/01/20 12:40:37 WARN util.Utils: Service 'SparkUI' could not bind on port 4042. Attempting port 4043.
The sum is: 46
s299165@jupyter-s299165:~$
```

Using the code above, as python script, the same result which is equal to 46 is appeared; The used (.py) script is in my LOCAL FILE SYSTEM & the "lab1_dataset.txt" file is the HDFS in the cluster.

Play with HDFS

In this exercise we try to manipulate files on HDFS.

Questions

Question 1:

Copy the HDFS file /data/students/bigdata_internet/lab1/lab1_dataset.txt in the local file system.
If you modify the local file, does the modifications automatically affect also the HDFS file?

Answer:

No, the location of these files are not the same, so there will be no automated modification on the HDFS file.

Question 2:

Create a file in the file system of the gateway and copy it to your home in HDFS. Which is the complete path of your file in HDFS? And on the gateway local file system?

Answer:

- **HDFS address : /user/s299165/BigData_Lab1_Part2.txt**
- **Local file address : /BigData_Lab1_Part2.txt**

Run a Job

Consider the same input file "/data/students/bigdata_internet/lab1/lab1_dataset.txt" . Write a Python script that reads the file, computes the sum of values for each person, and saves the output in a HDFS folder.

Question 1:

Report the code you have written, and explain the goal of each instruction.

Answer

- **The Python script used & the purpose of each line are:**

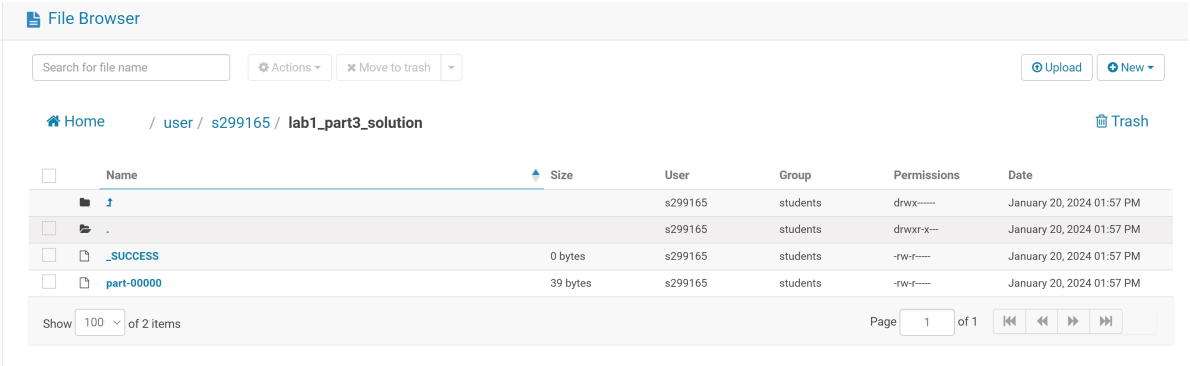
```
#Importing the necessary classes from PySpark
from pyspark import SparkConf, SparkContext

#Specifying the application name & configuration
conf = SparkConf().setAppName("My app")
sc = SparkContext(conf = conf)

#Addressing the paths for input & output files
inputPath = "/data/students/bigdata_internet/lab1/lab1_dataset.txt"
outputPath = "lab1_part3_solution/"

#Read the input file
rdd = sc.textFile(inputPath)

#Map the data using Split() method
fields_rdd = rdd.map(lambda line: line.split(","))

#Creat the tuple of data
value_rdd = fields_rdd.map(lambda l: (l[0],int(l[1])))

#Calculate the result using reduceByKey() and sum the values of tuples
new_rdd =  value_rdd.reduceByKey(lambda v1,v2: v1+v2)

#Save the output
```

Question 2:

How does the output folder on HDFS look like? Why do you find multiple parts file in the folder?

Answer:

- **The output folder is look like:**



- **SUCCESS: This file is empty and shows that the process is completed successfully without errors.**
- **00000: Is the output file with result ('alice', 14), ('bob', 11), ('john', 21).**

  The reason for having multiple part files is related to the distributed nature of Spark. Spark processes data in parallel across multiple nodes in a cluster.

Bonus Task

Considering file address above, write a script that reads the file, and concatenates all values for a name, separating them by : . Then, it saves the output in a HDFS file.

In [4]:
```
#Read the contents of the specified text file into an RDD
readFile = sc.textFile("/data/students/bigdata_internet/lab1/lab1_dataset.txt")

#Transform the RDD to group values by name and create a key-value pair where the key i
name_values = readFile.map(lambda x: x.split(',')).map(lambda x: (x[0], int(x[1]))).gr

#Concatenate the values (converted to strings) for each name, separated by ':'
concatenated_values = name_values.map(lambda x: (x[0], ':'.join(map(str, x[1]))))

#show the result
concatenated_values.collect()
```

Out[4]:
```
[('bob', '5:3:3'), ('john', '4:8:9'), ('alice', '4:3:7')]
```

In [5]:
```
#Save the file
```

```
sc.textFile("/data/students/bigdata_internet/lab1/lab1_dataset_NEW.txt")
```

Out[5]:
```
/data/students/bigdata_internet/lab1/lab1_dataset_NEW.txt MapPartitionsRDD[20] at tex
tFile at NativeMethodAccessorImpl.java:0
```

Questions

Question 1:

Report the code you have written, and explain the goal of each instruction.

Answer:

the goal of each line is mentioned as a comment in the code section