

Evaluation Practical Work

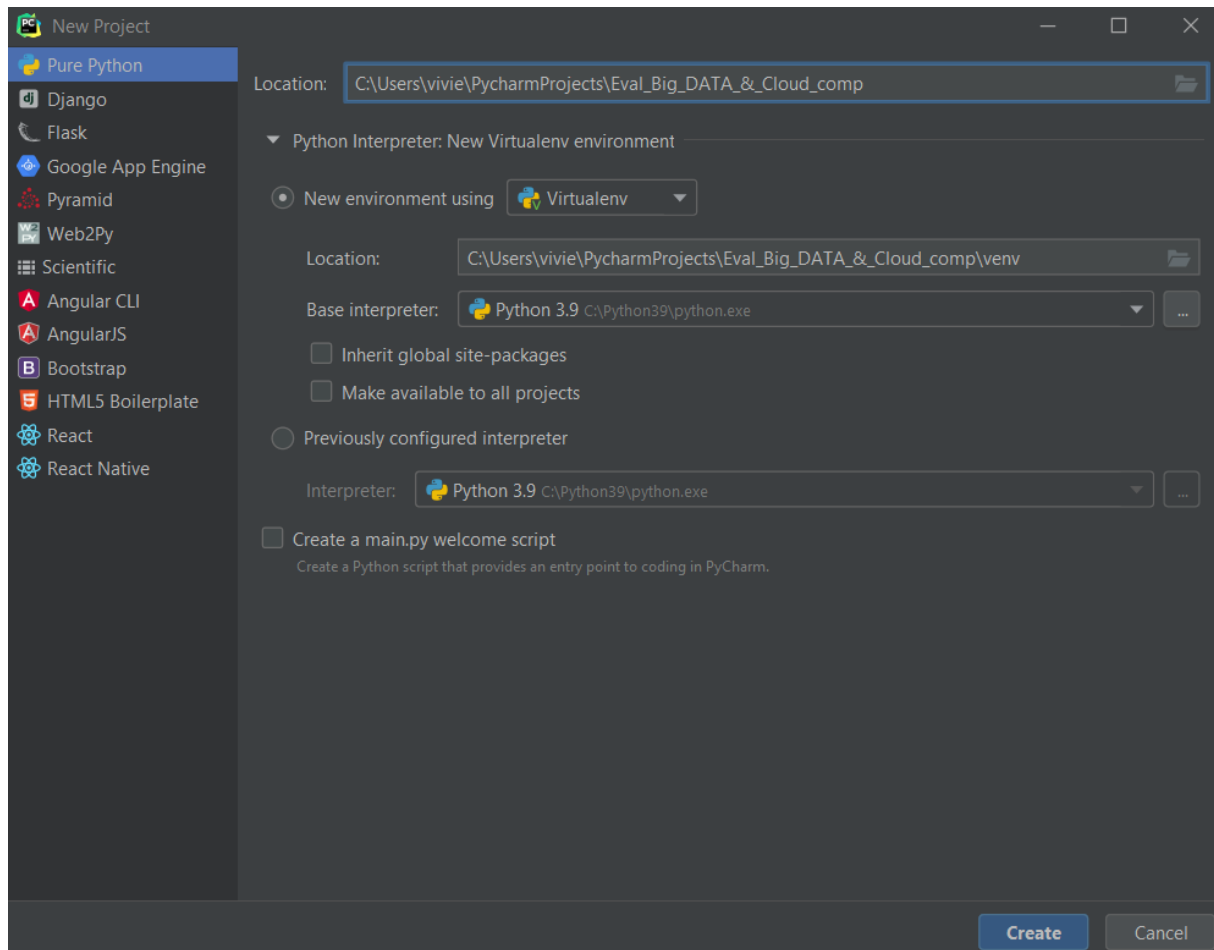
By Vivien BOGNE TIENOUE

EPITA, DSA

February 04, 2021

1. **Create a virtual environment in PyCharm and set it up to use Python 3.8 (or another version 3.x already installed on your computer, no need to install 3.8 if you don't have it yet).**

We create an environment named Eval_Big_Data_&_Cloud_comp using Virtualenv and Python 3.8 as shown in the picture below.



2. **Write a script in Python `shakespeare_words.py` which generates a list of the words contained in the complete works of Shakespeare and output it to a file. Name this file `shakespeare_data.txt`.**

```

1  import re
2  a_file = open("t8.shakespeare_sentence.txt", "r")
3
4  # remove every return to the line
5  string_without_line_breaks = ""
6  for line in a_file:
7      stripped_line = line.rstrip()
8      string_without_line_breaks += stripped_line
9
10 # remove the header ( everything between the quotes << and >> )
11 out = re.sub(r'<<.+?>>', '', string_without_line_breaks)
12
13 # remove the number
14 from string import digits
15
16 remove_digits = str.maketrans('', '', digits)
17 res = out.translate(remove_digits)
18

```

```

18
19 # split into words with return to line while removing punctuation
20 wordList = re.sub("[^\\w]", " ", res).split()
21
22 # Save the different word in a file
23 f = open("shakespeare_data.txt", "w+")
24 for item in wordList:
25     f.write("%s\n" % item)
26
27 # Close files
28 f.close()
29 a_file.close()

```

3. Install PySpark using the command line:

```

C:\Users\vivie\PycharmProjects\Eval_Big_DATA_&Cloud_comp\venv>pip install pyspark
Collecting pyspark
  Using cached pyspark-3.0.1.tar.gz (204.2 MB)
Collecting py4j==0.10.9
  Using cached py4j-0.10.9-py2.py3-none-any.whl (198 kB)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.0.1-py2.py3-none-any.whl size=204612242 sha256=1a864ff99c9b8630932b
  Stored in directory: c:\users\vivie\appdata\local\pip\cache\wheels\ea\21\84\970b03913d0d6a96ef51c34c878add0de9e4e
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.0.1
C:\Users\vivie\PycharmProjects\Eval_Big_DATA_&Cloud_comp\venv>

```

4. Test the installation by running the PySpark shell:

```
C:\Users\vivie\PyscharmProjects\Eval_Big_DATA_&Cloud_comp\venv>pyspark
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
21/02/04 18:38:35 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
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 3.0.1
        /___/\

Using Python version 3.8.5 (default, Sep  3 2020 21:29:08)
SparkSession available as 'spark'.
>>> 21/02/04 18:38:48 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metrics is stopped
```

5. Load your `shakespeare_data.txt` file in the shell and count the number of words in the file.

```
data = spark.read.csv('shakespeare_data.txt')
>>> data.count()
909236
>>>
```


6. Exit the shell:

```
707236
>>> exit()

C:\Users\vivie\PycharmProjects\Eval_Big_DATA_&Cloud_comp\venv>SUCCESS: The process with PID 12492 (child process of PID 16560) has been terminated.
SUCCESS: The process with PID 16560 (child process of PID 22780) has been terminated.
SUCCESS: The process with PID 22780 (child process of PID 2592) has been terminated.
```

7. Create a `word_count.py` file and do the same thing like with the shell (reading the text file and displaying the number of words). For that, you need to create an instance of `spark` in your script using a `SparkSession`:

The Script:

```
1 from pyspark.sql import SparkSession
2
3 spark = SparkSession.builder.appName('WordCountApp').getOrCreate()
4
5 data = spark.read.csv("shakespeare_data.txt")
6
7 
8 print("#####COUNT THE NUMBER OF WORDS #####")
9 print(data.count())
10
```

When running the Script we have the following result:

```
C:\Users\vivie\PycharmProjects\Eval_Big_DATA_&Cloud_comp\venv>python word_count.py
21/02/04 18:50:51 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
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).
##### COUNT THE NUMBER OF WORDS #####
909236

C:\Users\vivie\PycharmProjects\Eval_Big_DATA_&Cloud_comp\venv>SUCCESS: The process with PID 23004 (child process of PID 5004) has been terminated.
SUCCESS: The process with PID 5004 (child process of PID 22784) has been terminated.
SUCCESS: The process with PID 22784 (child process of PID 20232) has been terminated.
```

8. Then add spark code to your script in order to show:

- The first three values in the text file
- The 10 longest words, showing their length
- The 10 words having the highest number of occurrences, with their number of occurrences.

Here is the script:

```
1  from pyspark.sql import SparkSession
2  # Create an instance of Spark
3  spark = SparkSession.builder.appName('WordCountApp').getOrCreate()
4  # Load the data
5  data = spark.read.csv("shakespeare_data.txt")
6
7  # rename the column
8  data = data.withColumnRenamed('_c0','words')
9  print("#####NUMBER OF WORDS #####")
10 print(data.count())
11
12 # Create an in-memory DataFrame to query
13 data.createOrReplaceTempView("text")
14
15 print("#####FIRST THREE #####")
16 # First Three values in the words file
17 first_three = spark.sql("SELECT * FROM text limit 3")
18 first_three.show()
19
20 print("#####TEN LONGEST #####")
21 # 10 longest text, showing their length
22 ten_longest = spark.sql("SELECT words, LENGTH(words) AS length_words "
23                          "FROM text "
24                          "GROUP BY words "
25                          "ORDER BY LENGTH(words) DESC "
26                          "LIMIT 10")
27 ten_longest.show()
```

```

28 print("##### TEN HIGHEST OCCURENCES #####")
29
30 # 10 text having highest number of occurrences, with the number of occurrences
31 ten_highest_occ = spark.sql("SELECT words, COUNT(words) AS occurrences_words"
32                             " FROM text"
33                             " GROUP BY words "
34                             " ORDER BY COUNT(words) DESC "
35                             " LIMIT 10")
36 ten_highest_occ.show()
37
38

```

9. Run your script locally and observe the results.

After running we have the following result:

```

C:\Users\vivie\PycharmProjects\Eval_Big_DATA_&Cloud_comp\venv>python word_count.py
21/02/04 18:58:09 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform...
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).
#####NUMBER OF WORDS #####
909236
#####FIRST THREE #####
+-----+
| words|
+-----+
| From|
| fairest|
| creatures|
+-----+

```

```

#####TEN LONGEST #####
+-----+-----+
| words|length_words|
+-----+-----+
|honorificabilitud...| 27|
|AntipholusesBALTH...| 21|
|ExeuntKING_HENRY_...| 21|
|possiblesatisfaction| 20|
|GloucestershireEnter| 20|
| ShakespeareDRAMATIS| 19|
| schoolmasterAEMILIA| 19|
| ShakespeareDramatis| 19|
| BALTHAZARANTIPHOLUS| 19|
| OFEPHESUSANTIPHOLUS| 19|
+-----+-----+

```

```
##### TEN HIGHEST OCCURENCES #####
```

words	occurences	words
the	23241	
I	22225	
and	18609	
to	16329	
of	15684	
a	12779	
you	12160	
my	10838	
in	10002	
d	8954	

10. Create an Amazon EMR cluster (keep default parameters).

This is the screenshot page of the creation :

Create Cluster - Quick Options [Go to advanced options](#)

General Configuration

Cluster name

☒ Logging ⓘ

S3 folder

Launch mode ☒ Cluster ⓘ ☐ Step execution ⓘ

Software configuration

Release ⓘ

Applications

- ☐ Core Hadoop: Hadoop 2.10.1, Hive 2.3.7, Hue 4.8.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ☐ HBase: HBase 1.4.13, Hadoop 2.10.1, Hive 2.3.7, Hue 4.8.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- ☐ Presto: Presto 0.240.1 with Hadoop 2.10.1 HDFS and Hive 2.3.7 Metastore
- ☒ Spark: Spark 2.4.7 on Hadoop 2.10.1 YARN and Zeppelin 0.8.2

☐ Use AWS Glue Data Catalog for table metadata ⓘ

Hardware configuration

Instance type ⓘ The selected instance type adds 64 GiB of GP2 EBS storage per instance by default. [Learn more](#)

Number of instances (1 master and 2 core nodes)

Cluster scaling ☐ scale cluster nodes based on workload

Security and access

EC2 key pair ⓘ [Learn how to create an EC2 key pair.](#)

Permissions ☒ Default ☐ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role [EMR_DefaultRole](#) ⓘ

EC2 instance profile [EMR_EC2_DefaultRole](#) ⓘ

Cancel

Create cluster

After the creation complete we have this result:

11. Create a S3 bucket and upload your data to the bucket.

- The creation

The screenshot shows the 'Create bucket' page in the Amazon S3 console. At the top, there's a breadcrumb 'Amazon S3 > Create bucket'. The main heading is 'Create bucket', followed by a subtext: 'Buckets are containers for data stored in S3. [Learn more](#)'. The page is divided into sections. The 'General configuration' section contains a 'Bucket name' field with 'exam-bucket' entered, a note about naming rules, a 'Region' dropdown set to 'US East (N. Virginia) us-east-1', and a 'Choose bucket' button. The 'Default encryption' section has a note and radio buttons for 'Disable' (selected) and 'Enable'. Below this is an 'Advanced settings' section with a right-pointing arrow. A light blue information box at the bottom states: 'After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.' At the very bottom right are 'Cancel' and 'Create bucket' buttons.

Amazon S3 > Create bucket

Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

Region

US East (N. Virginia) us-east-1 ▼

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Choose bucket

Default encryption

Automatically encrypt new objects stored in this bucket. [Learn more](#)

Server-side encryption

☒ Disable
☐ Enable

► Advanced settings

After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel Create bucket

- The upload :

Amazon S3 > exam-bucket > Upload

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (2 Total, 5.3 MB)

All files and folders in this table will be uploaded.

Remove

Add files

Add folder

 Find by name

< 1 >

<input type="checkbox"/>	Name ▲	Folder ▼	Type ▼	Size ▼
<input type="checkbox"/>	shakespeare_data.txt	-	text/plain	5.3 MB
<input type="checkbox"/>	word_count.py	-	-	3.0 KB

Destination

Destination

s3://exam-bucket

Destination details

The following bucket settings impact new objects stored in the specified destination.

Bucket Versioning

When enabled, multiple variants of an object can be stored in the bucket to easily recover from unintended user actions and application failures. [Learn more](#)

⚠ Disabled

Default encryption

When enabled, new objects stored in this bucket are automatically encrypted. [Learn more](#)

Disabled

Object Lock

When enabled, objects in this bucket might be prevented from being deleted or overwritten for a fixed amount of time or indefinitely. [Learn more](#)

Disabled

⚠ We recommend that you enable Bucket Versioning to help protect against unintentionally overwriting or deleting objects. [Learn more](#)

Enable Bucket Versioning

► Additional upload options

Cancel

Upload

- Creation of a folder where we will save the result

Folder

Folder name

 /

Folder names can't contain "/". [See rules for naming](#)

Server-side encryption

The following settings apply only to the new folder object and not to the objects contained within it.

Server-side encryption

☒ Disable

☐ Enable

Cancel
Create folder

- So this is our bucket ready :

Amazon S3 > exam-bucket

exam-bucket

Objects Properties Permissions Metrics Management Access Points

Objects (3)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Refresh Delete Actions Create folder Upload

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	myOutputFolder/	Folder	-	-	-
<input type="checkbox"/>	shakespeare_data.txt	txt	February 7, 2021, 12:06:35 (UTC+01:00)	5.3 MB	Standard
<input type="checkbox"/>	word_count.py	py	February 7, 2021, 12:06:32 (UTC+01:00)	3.0 KB	Standard

12. Open the Spark History Server user interface.

History Server

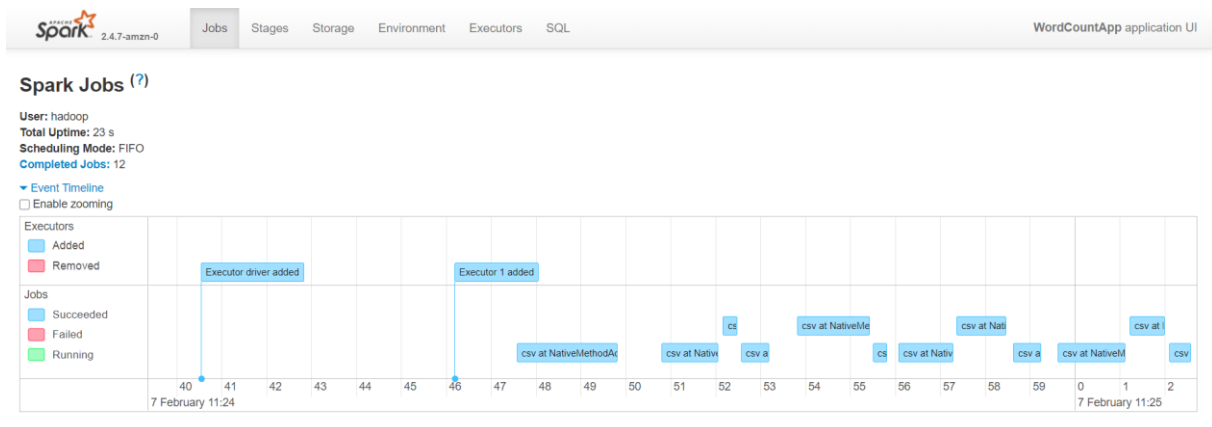
Event log directory: s3a://prod.us-east-1.appinfo.srcj-3HREZ85KAOQQS/sparklogs
 Last updated: 2021-02-07 12:26:57
 Client local time zone: Europe/Paris

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1612696959922_0001	WordCountApp	2021-02-07 12:24:39	2021-02-07 12:25:02	23 s	hadoop	2021-02-07 12:26:03	Download

13. Run your script on the cluster using the graphical user interface. Observe the jobs being run in the Spark History Server.

BIG DATA INFRASTRUCTURE & CLOUD COMPUTING



14. Observe the final state in Spark History Server.

Completed Jobs (12)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:25:02	0.5 s	2/2 (2 skipped)	27/27 (12 skipped)
10	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:25:01	0.8 s	1/1 (1 skipped)	10/10 (2 skipped)
9	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:59	1 s	1/1	2/2
8	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:58	0.6 s	2/2 (2 skipped)	27/27 (12 skipped)
7	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:57	1 s	1/1 (1 skipped)	10/10 (2 skipped)
6	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:56	1 s	1/1	2/2
5	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:55	0.3 s	2/2 (1 skipped)	11/11 (2 skipped)
4	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:53	2 s	1/1	2/2
3	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:52	0.6 s	1/1 (2 skipped)	1/1 (12 skipped)
2	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:52	0.3 s	1/1 (1 skipped)	10/10 (2 skipped)
1	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:50	1 s	1/1	2/2
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:24:47	2 s	1/1	1/1

Stages for All Jobs

Completed Stages: 15

Skipped Stages: 10

Completed Stages (15)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
24	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:25:02	0.2 s	1/1		112.0 B	5.9 KB	
23	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:25:02	0.3 s	26/26			2.6 MB	5.9 KB
20	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:25:01	0.8 s	10/10			2.2 MB	2.6 MB
18	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:59	1 s	2/2	5.3 MB			2.2 MB
17	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:59	0.2 s	1/1		263.0 B	7.8 KB	
16	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:58	0.4 s	26/26			2.2 MB	7.8 KB
13	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:57	1 s	10/10			2.2 MB	2.2 MB
11	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:56	1 s	2/2	5.3 MB			2.2 MB
10	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:55	0.1 s	1/1		27.0 B		
9	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:55	0.1 s	10/10				892.0 B
7	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:53	2 s	2/2	5.3 MB			2.2 MB
6	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:52	0.5 s	1/1		16.0 B	590.0 B	
3	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:52	0.3 s	10/10			18.0 KB	590.0 B
1	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:50	1 s	2/2	5.3 MB			18.0 KB
0	csv at NativeMethodAccessorImpl.java:0	+details 2021/02/07 11:24:47	2 s	1/1	16.0 KB			

15. Propose and test different optimizations to improve the performances. Show the performance gains attained with each optimization tested.

- First optimisation : adding number of repartition on write and read statement

```

text_df = spark.read.csv(data_source).repartition(10)
first_three.repartition(10).write.option("header",
"true").mode("append").csv(output_uri)

```

BIG DATA INFRASTRUCTURE & CLOUD COMPUTING

- `count_df.repartition(10).write.option("header", "true").mode("append").csv(output_uri)`
- `ten_longest.repartition(10).write.option("header", "true").mode("append").csv(output_uri)`
- `ten_highest_occ.repartition(10).write.option("header", "true").mode("append").csv(output_uri)`

Completed Stages (18)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
38	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:26	0.3 s	10/10		319.0 B	720.0 B	
33	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:26	17 ms	1/1			5.9 KB	720.0 B
32	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:25	0.2 s	26/26			2.6 MB	5.9 KB
29	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:25	0.7 s	10/10			2.2 MB	2.6 MB
27	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:23	1 s	2/2	5.3 MB			2.2 MB
26	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:22	0.6 s	10/10		434.0 B	878.0 B	
21	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:22	27 ms	1/1			7.8 KB	878.0 B
20	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:22	0.4 s	26/26			2.2 MB	7.8 KB
17	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:21	1 s	10/10			2.2 MB	2.2 MB
15	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:19	1 s	2/2	5.3 MB			2.2 MB
14	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:19	0.2 s	1/1		27.0 B		
13	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:19	0.1 s	10/10				892.0 B
11	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:17	2 s	2/2	5.3 MB			2.2 MB
10	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:16	0.6 s	10/10		16.0 B	59.0 B	
6	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:15	37 ms	1/1			590.0 B	59.0 B
3	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:15	0.3 s	10/10			18.0 KB	590.0 B
1	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:14	1 s	2/2	5.3 MB			18.0 KB
0	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:45:11	2 s	1/1	7.6 KB			

- Second optimization : Partitions — That Determine the parallelism At shuffle level

We change it on our script using the command :

```
spark.conf.set("spark.sql.shuffle.partitions", 500)
```

Stages for All Jobs

Completed Stages: 15

Skipped Stages: 10

Completed Stages (15)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
24	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:15	0.2 s	1/1		112.0 B	124.8 KB	
23	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:13	1 s	500/500			2.2 MB	124.8 KB
20	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:13	0.5 s	10/10			2.2 MB	2.2 MB
18	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:11	1 s	2/2	5.3 MB			2.2 MB
17	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:11	0.2 s	1/1		263.0 B	139.9 KB	
16	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:09	2 s	500/500			1899.5 KB	139.9 KB
13	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:07	1.0 s	10/10			2.2 MB	1899.5 KB
11	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:06	1 s	2/2	5.3 MB			2.2 MB
10	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:05	0.2 s	1/1		27.0 B		
9	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:05	0.1 s	10/10				892.0 B
7	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:03	2 s	2/2	5.3 MB			2.2 MB
6	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:02	0.5 s	1/1		16.0 B	590.0 B	
3	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:02	0.3 s	10/10			18.0 KB	590.0 B
1	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:39:00	1 s	2/2	5.3 MB			18.0 KB
0	csv at NativeMethodAccessorImpl.java:0	2021/02/07 11:38:58	2 s	1/1	7.6 KB			

CONCLUSION:

As we can see the difference is not notable because our app doesn't deal with big data and the script is also short. With a bigger data, those settings would have increase the performance of our spark app.