

Day 10 – Spark – Accenture bootcamp

Name: Pothumulla Kankanamge Mewan Madhusha

Table of Contents

<i>Spark performance research</i>	<i>3</i>
<i>Using PySpark complete following tasks</i>	<i>5</i>

GitHub Link - <https://github.com/mewanmadusha/Learning2024/tree/main/day-10-spark-bootcamp>

Spark performance research

Google for posts related to Spark performance for various implementations: Spark Scala, PySpark, Spark,Java, express your own opinion in free from and send it via email.

Reference –

Scala vs Java vs Pyspark - Which is better ?

<https://www.linkedin.com/feed/update/urn:li:activity:7072258588278210561/>

PySpark vs Spark-Scala

<https://medium.datadriveninvestor.com/pyspark-vs-spark-scala-b3a744d91846>

Scala Spark vs Python PySpark: Which is better?

<https://mungingdata.com/apache-spark/python-pyspark-scala-which-better/>

Scala vs PySpark: Choosing the Right Language for Parallel Processing in Spark

<https://medium.com/@mohitdaxini75/scala-vs-pyspark-choosing-the-right-language-for-parallel-processing-in-spark-6ff996389f53>

Based on the above articles, comparing Scala, Java, PySpark from Apache spark development we can be able to find following things.

To evaluate the comparison, I have summarized things into the table,

	Scala	Java	PySpark
Advantages	<ul style="list-style-type: none">-This is the native and primary language to build Spark systems.-Since Scala compilation can be run under JVM we can get good performance-Ability to do the functional programming-Since Scala using datasets it's compile time safe-Type safe – because Scala written in static type	<ul style="list-style-type: none">-Performance was more or less comparing to Scala, but most of the cases Scala was performing better.-Scalability-Mature ecosystem and wide range of tools can be interreact with Java-Since most of the backend development done in Java microservices most of the time developers can	<ul style="list-style-type: none">-Easy to handle with Python that means Simplicity and productivity with clean and expressive Python syntax-More compatible with data science eco system-Since Python using data frames it's not compile time safe-Less type safety compared to Scala and java because Python written in

		adhere Java spark into microservice architecture -Since Java using datasets it's compile time safe	dynamically typed language
Drawbacks/Need to consider before choose	-Bigger learning curve -Similar to previous drawback as well but Scala having more complex syntax and we need to do more work around to implement same task when compared with python	-It takes much time for development process	-Performance drawback when running under python interpreter

Conclusion

Based on the comparison, if you developers don't have much coding experience or prefer using graphical tools for data processing, PySpark is a good pick because it's easy to use. However, if they are really focused on making things run super-fast and don't mind spending a bit more time learning, Scala and Java are better options. Java is especially good if those who are working with big business systems. Finally we can choose PySpark for simplicity, Java or Scala for top speed

Using PySpark complete following tasks

Tasks:

Solve the given Tasks by writing your code solution:

Create conda environment and install pyspark

```
conda create -n sparkenv python numpy pandas
```

```
mewanmadusha -- zsh -- 138x37
(base) mewanmadusha@Mewans-MacBook-Air-M1 ~ %
(base) mewanmadusha@Mewans-MacBook-Air-M1 ~ % conda env list
# conda environments:
#
base                * /Users/mewanmadusha/miniconda3
bdb                 /Users/mewanmadusha/miniconda3/envs/bdb
env1                /Users/mewanmadusha/miniconda3/envs/env1
env2py39            /Users/mewanmadusha/miniconda3/envs/env2py39
env3tf29            /Users/mewanmadusha/miniconda3/envs/env3tf29

(base) mewanmadusha@Mewans-MacBook-Air-M1 ~ %
(base) mewanmadusha@Mewans-MacBook-Air-M1 ~ % conda create -n sparkenv python numpy pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 23.1.0
latest version: 23.11.0

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=23.11.0

## Package Plan ##

environment location: /Users/mewanmadusha/miniconda3/envs/sparkenv

added / updated specs:
- numpy
- pandas
```

```
conda activate sparkenv
```

```
conda install -c conda-forge pyspark
```

```
mewanmadusha -- zsh -- 162x45
(base) mewanmadusha@Mewans-MacBook-Air-M1 ~ % conda activate sparkenv
(sparkenv) mewanmadusha@Mewans-MacBook-Air-M1 ~ %
(sparkenv) mewanmadusha@Mewans-MacBook-Air-M1 ~ %
(sparkenv) mewanmadusha@Mewans-MacBook-Air-M1 ~ % python --version
Unknown option: -e
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Try 'python -h' for more information.
(sparkenv) mewanmadusha@Mewans-MacBook-Air-M1 ~ % python --version
Python 3.12.0
(sparkenv) mewanmadusha@Mewans-MacBook-Air-M1 ~ %
(sparkenv) mewanmadusha@Mewans-MacBook-Air-M1 ~ % conda install -c conda-forge pyspark
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 23.1.0
latest version: 23.11.0

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=23.11.0

## Package Plan ##

environment location: /Users/mewanmadusha/miniconda3/envs/sparkenv

added / updated specs:
- pyspark

The following packages will be downloaded:

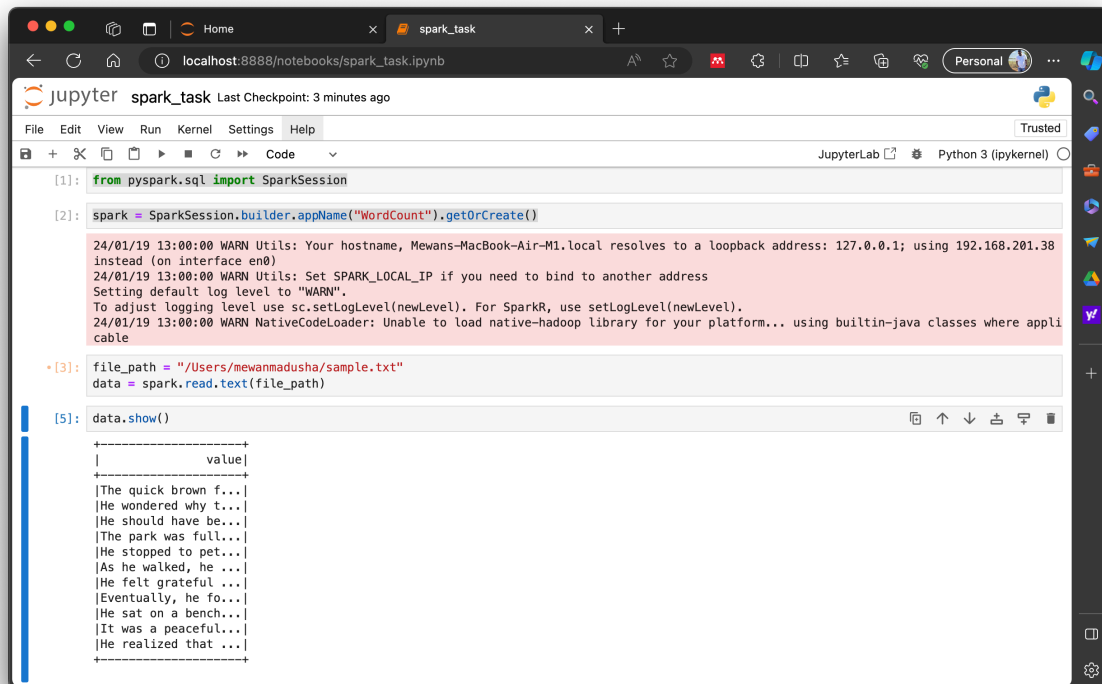
package | build | size | channel
-----|-----|-----|-----
abseil-cpp-20211102.0 | h6b3803e_1 | 1000 KB | conda-forge
arrow-cpp-11.0.0 | hc7aafb3_2 | 6.7 MB |
aws-c-common-0.6.8 | h80987f9_1 | 156 KB |
aws-c-event-stream-0.1.6 | h313beb8_6 | 24 KB |
```

Install Jupiter notebook

pip install notebook

1. Read the data using “sample.txt” as a source data.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("WordCount").getOrCreate()
file_path = "/Users/mewanmadusha/sample.txt"
data = spark.read.text(file_path)
```



The screenshot shows a JupyterLab notebook window titled "spark_task". The browser address bar indicates the URL is "localhost:8888/notebooks/spark_task.ipynb". The notebook interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and execution. The code editor displays the following code cells:

```
[1]: from pyspark.sql import SparkSession

[2]: spark = SparkSession.builder.appName("WordCount").getOrCreate()

24/01/19 13:00:00 WARN Utils: Your hostname, Mewans-MacBook-Air-M1.local resolves to a loopback address: 127.0.0.1; using 192.168.201.38 instead (on interface en0)
24/01/19 13:00:00 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/01/19 13:00:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

* [3]: file_path = "/Users/mewanmadusha/sample.txt"
      data = spark.read.text(file_path)

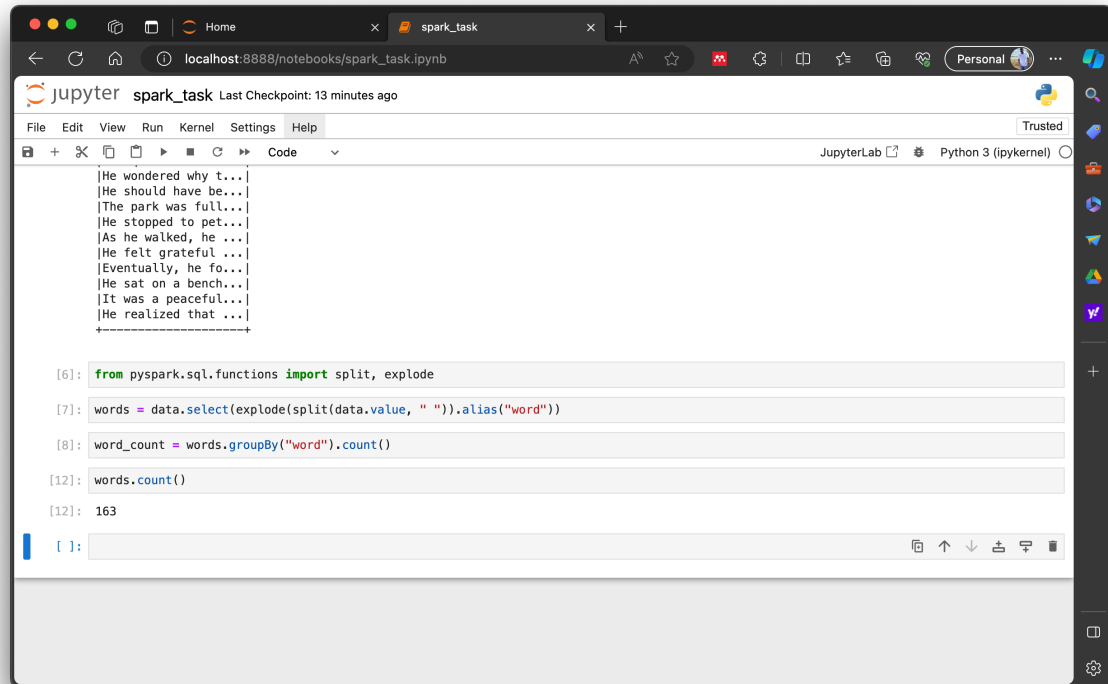
[5]: data.show()
```

The output of the `data.show()` command is displayed in a table format:

value
The quick brown f...
He wondered why t...
He should have be...
The park was full...
He stopped to pet...
As he walked, he ...
He felt grateful ...
Eventually, he fo...
He sat on a bench...
It was a peaceful...
He realized that ...

2. Count the number of words.

```
from pyspark.sql.functions import split, explode
words = data.select(explode(split(data.value, " ")).alias("word"))
word_count = words.groupBy("word").count()
words.count()
```



The screenshot shows a JupyterLab notebook interface with a dark theme. The browser address bar indicates the notebook is running at localhost:8888. The notebook title is 'spark_task'. The code editor contains a text sample and several lines of Spark SQL code. The text sample is a paragraph about a person in a park. The code defines a DataFrame 'words' by splitting the text into words and exploding them into rows. It then groups the words and counts the occurrences. The output of the final count operation is 163.

```
|He wondered why t...|
|He should have be...|
|The park was full...|
|He stopped to pet...|
|As he walked, he ...|
|He felt grateful ...|
|Eventually, he fo...|
|He sat on a bench...|
|It was a peaceful...|
|He realized that ...|
+-----+

[6]: from pyspark.sql.functions import split, explode

[7]: words = data.select(explode(split(data.value, " ")).alias("word"))

[8]: word_count = words.groupBy("word").count()

[12]: words.count()

[12]: 163
```

3. Count the number of word appearances.

```
word_count = words.groupBy("word").count()
```

spark_task

localhost:8888/notebooks/spark_task.ipynb

Jupyter spark_task Last Checkpoint: 25 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
[7]: words = data.select(explode(split(data.value, " ")).alias("word"))
[8]: word_count = words.groupBy("word").count()
[12]: words.count()
[12]: 163
[18]: word_count.show(100)
```

word	count
trees.	1
flowers	1
bench	1
Instead,	1
even	1
lazy	2
grateful	1
fluffy	1
continued	1
park,	1
colorful	1
was	5
park	1
jumped	1
it's	1
bring	1
dog	2
way.	1
enjoying	1