Spark Lab 2 Report

Building a Streaming Data Pipeline with Apache Spark

Contents

André Plancha
andre.plancha@hotmail.com

December 06th, 2025

Starting the Cluster

Dockerfile on the server

```Docker
FROM apache/spark:4.0.1
# switch to root to install packages
USER root
RUN pip install --no-cache-dir "pandas==2.3.2" "pyarrow==21.0.0"
# switch back to spark user
USER spark
```

compose.yaml on the server

```YAML
services:
  spark:
    build: .
    hostname: apache-spark
    ports:
      - "7077:7077"    # Spark master port
      - "8080:8080"    # Spark master web UI
      - "8081:8081"    # Spark worker web UI
      - "15002:15002"  # Spark Connect server port
      - "4040:4040"    # Spark Connect web UI
    command: >
      bash -c "/opt/spark/sbin/start-master.sh;
               /opt/spark/sbin/start-connect-server.sh;
               /opt/spark/sbin/start-worker.sh spark://192.168.1.7:7077;
               sleep infinity"
```

```Shell
$ ls
compose.yaml  Dockerfile
$ docker compose up -d
[+] Running 1/1
 ✔ Container spark-docker-spark-1  Started
 0.5s
```

In [1]:

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf

spark = SparkSession.builder \
  .remote("sc://192.168.1.7:15002") \
  .appName("UDFTransformation") \
  .config("spark.sql.ansi.enabled", "false") \
  .config("spark.sql.execution.pythonUDF.arrow.enabled", "true") \
  .getOrCreate()

# limit() shows a nice HTML table in Jupyter, while show() prints plain text
spark.conf.set('spark.sql.repl.eagerEval.enabled', True)

spark
```

Out[1]:

```
<pyspark.sql.connect.session.SparkSession at 0x24595594cd0>
```

In [2]:

```python
DataStreamReader = spark.readStream \
  .format("socket") \
  .option("host", "192.168.1.7") \
  .option("port", 9999)

linesDF = DataStreamReader.load()
lines_writer = linesDF.writeStream \
  .outputMode("append") \
  .format("memory") \
  .queryName("lines_table")

DataStreamReader
```

Out[2]:

```
<pyspark.sql.connect.streaming.readwriter.DataStreamReader at 0x245955953f0>
```

In [3]:

```python
def stop_queries():
  for query in spark.streams.active:
    query.stop()
```

Task 1

When you receive an sentences from socket, put your name before each word in sentences , and put last name after that(use UDF)

In [ ]:

```python
@udf
```

```
def name_transform(sentence, first="André", last="Plancha"):
  words = sentence.split()
  return ','.join([f"{first}{word}{last}" for word in words])

transformedDF = linesDF.withColumn("value", name_transform(linesDF.value))


stop_queries()

task1_writer = transformedDF.writeStream.outputMode("append").format("memory").queryName("task
task1_query = task1_writer.start()

task1_writer
```

Out[ ]:
```
<pyspark.sql.connect.streaming.readwriter.DataStreamWriter at 0x1932daf3ee0>
```

```
ssh plancha@192.168.1.7 -t /usr/bin/nc -lk 9999
plancha@192.168.1.7's password:
Sweden is a good country
Finland is a better country
Denmark is an awesome country
I love programming in Spark
Data is the new oil
```

In [ ]:
```
task1_query.stop()

spark.sql("SELECT * FROM task1_table").show(20, truncate=False)
```

```
+--------------------------------------------------------------------------------------+
|value                                                                                 |
+--------------------------------------------------------------------------------------+
|AndréSwedenPlancha,AndréisPlancha,AndréaPlancha,AndrégoodPlancha,AndrécountryPlancha  |
|AndréFinlandPlancha,AndréisPlancha,AndréaPlancha,AndrébetterPlancha,AndrécountryPlancha|
|AndréDenmarkPlancha,AndréisPlancha,AndréanPlancha,AndréawesomePlancha,AndrécountryPlancha|
|AndréIPlancha,AndrélovePlancha,AndréprogrammingPlancha,AndréinPlancha,AndréSparkPlancha|
|AndréDataPlancha,AndréisPlancha,AndréthePlancha,AndrénewPlancha,AndréoilPlancha       |
+--------------------------------------------------------------------------------------+
```

Task 2

Average word length from all the words received from socket. Use SQL.

In [41]:

```python
stop_queries()

linesDF.createOrReplaceTempView("lines_table")

task2_DF = spark.sql("""
  from lines_table |>
  select value, split(value, ' ') as phrase |>
  select explode(phrase) as word |>
  aggregate avg(length(word)) as average_word_length
""")
task2_writer = task2_DF.writeStream.outputMode("complete").format("memory").queryName("task2_t
task2_query = task2_writer.start()
task2_writer
```

Out[41]:

```
<pyspark.sql.connect.streaming.readwriter.DataStreamWriter at 0x245a7fc76d0>
```

```
ssh plancha@192.168.1.7 -t /usr/bin/nc -lk 9999
plancha@192.168.1.7's password:
Sweden is a good country
Finland is a better country
Denmark is an awesome country
I love programming in Spark
Data is the new oil
```

In [39]:

```python
lines = [
  "Sweden is a good country",
  "Finland is a better country",
  "Denmark is an awesome country",
  "I love programming in Spark",
  "Data is the new oil"
]
words = " ".join(lines).split(" ")
lengths = [len(word) for word in words]
average_length = sum(lengths) / len(lengths)
average_length
```

Out[39]:

```
4.24
```

In [42]:

```python
task2_query.stop()
spark.sql("SELECT * FROM task2_table").show(20, truncate=False)
```

```
+------------------+
|average_word_length|
```

```
+------------------+
|4.24              |
+------------------+

```

Task 3

Use SQL to filter only the even numbers from the streaming data.

In [59]:

```python
stop_queries()

linesDF.createOrReplaceTempView("lines_table")

task3_DF = spark.sql("""
  from lines_table |>
  select cast(value as int) as value |>
  where value % 2 == 0
""")
task3_writer = task3_DF.writeStream.outputMode("append").format("memory").queryName("task3_tab
task3_query = task3_writer.start()
task3_writer
```

Out[59]:

```
<pyspark.sql.connect.streaming.readwriter.DataStreamWriter at 0x245a7fc62c0>
```

```
$ ssh plancha@192.168.1.7 -t /usr/bin/nc -lk 9999
plancha@192.168.1.7's password:
3413
12315
1231
121516
574
2325
252
5785
23123
54756712
312312
31
543646856
231425
```

In [51]:

```python
numbers = [
```

```
    3413,
    12315,
    1231,
    121516,
    574,
    2325,
    252,
    5785,
    23123,
    54756712,
    312312,
    31,
    543646856,
    231425
]
[num for num in numbers if num % 2 == 0]
```

Out[51]:

```
[121516, 574, 252, 54756712, 312312, 543646856]
```

In [60]:

```
task3_query.stop()
spark.sql("SELECT * FROM task3_table").show(20, truncate=False)
```

Python

```
+---------+
|value    |
+---------+
|574      |
|543646856|
|54756712 |
|252      |
|312312   |
|121516   |
+---------+

```