Выполнил: Бойко Илья Группа: ПИН-б-о-22-1

Лабораторная работа №3: Работа с RDD в Apache Spark

1. Цель работы: Освоить работу с высокоуровневыми API Apache Spark — Spark SQL и DataFrames. Получить практические навыки загрузки структурированных данных, выполнения запросов с помощью SQL-синтаксиса и DataFrame API, а также проведения базового анализа и визуализации данных.

2. Используемый стек технологий

Язык программирования: Python (PySpark)

Фреймворк: Apache Spark

Инструментарий: Jupyter Notebook

3. Теоретические сведения

RDD (Resilient Distributed Dataset):- это неизменяемая, отказоустойчивая коллекция объектов, распределённых across узлами кластера, которыми можно манипулировать параллельно.

Resilient (Устойчивый):- Способность восстанавливаться после сбоев благодаря lineage (цепочке происхождения). RDD запоминает последовательность операций, применённых к нему, чтобы пересчитать потерянные части данных.

Distributed (Распределённый):- Данные разделены на партиции (partitions) и обрабатываются на разных узлах кластера.

Dataset (Набор данных):- Представляет собой набор данных (объектов).

Трансформации (Transformations):- "Ленивые" операции, которые создают новое RDD из существующего (например, `map`, `filter`, `flatMap`). Они не выполняются немедленно, а лишь строят план вычислений (DAG).

Действия (Actions):- Операции, которые запускают вычисления для возврата результата в драйвер-программу или сохранения его во внешнее хранилище (например, `count`, `collect`, `take`, `saveAsTextFile`). Они *заставляют* выполниться все трансформации в lineage.

4. **Ход выполнения работы**

Базовые операции с RDD

- 4.1 Инициализация Spark Context

- Код SC.py:

```
from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("RDD-Lab").setMaster("local[*]") sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")

print("Spark Context создан успешно!") print(f"Версия Spark: {sc.version}")
```

- 4.2 Создание RDD из локальной коллекции
- Код RDD1.py:

```
numbers_rdd = sc.parallelize(range(1, 1001), 4) # 4 партиции print(f"Числа RDD создан:
{numbers_rdd}") print(f"Количество партиций: {numbers_rdd.getNumPartitions()}") print(f"Общее
количество элементов: {numbers_rdd.count()}")
```

- 4.3 Применение трансформаций к RDD с числами
- Код RDDTRAN.py:

```
even_numbers_rdd = numbers_rdd.filter(lambda x: x % 2 == 0) print(f"После фильтрации четных
чисел: {even_numbers_rdd}")

squared_numbers_rdd = even_numbers_rdd.map(lambda x: x ** 2) print(f"После возведения в
квадрат: {squared_numbers_rdd}")

first_10_squared = squared_numbers_rdd.take(10) print(f"Первые 10 четных чисел в квадрате:
{first_10_squared}")

total_count = squared_numbers_rdd.count() print(f"Общее количество четных чисел в квадрате (от 1
до 1000): {total_count}")
```

- 4.4 Результат проверкм кода

```
Первые 10 четных чисел в квадрате: [4, 16, 36, 64, 100, 144, 196, 256, 324, 400] Общее количество
четных чисел в квадрате (от 1 до 1000): 500
```

- 4.5 Создание тестового файла
- Код создания test_logs.txt:

```
test_logs = [ "192.168.1.1 - - [15/Oct/2024:10:15:30 +0200] "GET /index.html HTTP/1.1" 200 1234",
"192.168.1.2 - - [15/Oct/2024:10:15:31 +0200] "GET /about.html HTTP/1.1" 200 5678", "192.168.1.3 - -
[15/Oct/2024:10:15:32 +0200] "POST /login HTTP/1.1" 200 2345", "192.168.1.4 - -
[15/Oct/2024:10:15:33 +0200] "GET /index.html HTTP/1.1" 404 1234 ERROR", "192.168.1.5 - -
[15/Oct/2024:10:15:34 +0200] "GET /contact.html HTTP/1.1" 200 3456", "192.168.1.1 - -
[15/Oct/2024:10:15:35 +0200] "GET /index.html HTTP/1.1" 200 1234", "192.168.1.6 - -
[15/Oct/2024:10:15:36 +0200] "POST /submit HTTP/1.1" 500 6789 ERROR", "192.168.1.2 - -
[15/Oct/2024:10:15:37 +0200] "GET /products.html HTTP/1.1" 200 4567", "192.168.1.7 - -
[15/Oct/2024:10:15:38 +0200] "GET /index.html HTTP/1.1" 200 1234", "192.168.1.8 - -
[15/Oct/2024:10:15:39 +0200] "GET /about.html HTTP/1.1" 200 5678" ]
```

with open("test_logs.txt", "w") as f: for log in test_logs: f.write(log + "\n")

- 4.6 Создание RDD из текстового файла и применение трансформаций
- Код RDDTEST.py:

log_rdd = sc.textFile("test_logs.txt") print(f"Логи RDD создан: {log_rdd}") print(f"Количество строк в логах: {log_rdd.count()}")

uppercase_logs_rdd = log_rdd.map(lambda line: line.upper()) print("Первые 3 строки в верхнем регистре:") for log in uppercase_logs_rdd.take(3): print(f" {log}")

error_logs_rdd = log_rdd.filter(lambda line: "ERROR" in line) print(f"\nСтроки с ERROR:") for log in error_logs_rdd.collect(): print(f" {log}")

error_count = error_logs_rdd.count() print(f"\nВсего строк с ERROR: {error_count}")

- Результат:

Первые 3 строки в верхнем регистре: 192.168.1.1 - - [15/OCT/2024:10:15:30 +0200] "GET /INDEX.HTML HTTP/1.1" 200 1234 192.168.1.2 - - [15/OCT/2024:10:15:31 +0200] "GET /ABOUT.HTML HTTP/1.1" 200 5678 192.168.1.3 - - [15/OCT/2024:10:15:32 +0200] "POST /LOGIN HTTP/1.1" 200 2345

Строки с ERROR: 192.168.1.4 - - [15/Oct/2024:10:15:33 +0200] "GET /index.html HTTP/1.1" 404 1234 ERROR 192.168.1.6 - - [15/Oct/2024:10:15:36 +0200] "POST /submit HTTP/1.1" 500 6789 ERROR

Всего строк с ERROR: 2