Relatório Lab 2 - Introdução ao Spark

Aluna: Letícia Trein Medeiros

Este relatório descreve os passos realizados para a instalação e configuração do Apache Spark, bem como a execução de operações básicas utilizando spark-shell, pyspark e Jupyter Notebook

Passo 1: Download e Configuração do Spark

1. Download do Spark:
Acessei o site oficial e baixei a versão pré-compilada para Hadoop (arquivo `.tgz`), Foi optado pela versão mais recente compatível com o Hadoop instalado no Lab 1 (Spark 3.4.1).

2. Extração e Movimentação
- Extraí o arquivo compactado com o comando:
tar -xzf spark-3.4.1-bin-hadoop3.tgz

- Movi o diretório para `/usr/local/spark`com o comando:
sudo mv spark-3.4.1-bin-hadoop3 /usr/local/spark

3. Configuração das Variáveis de Ambiente:
- Editei o arquivo `~/.bashrc` adicionando:
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
export PYSPARK_PYTHON=python3
source ~/.bashrc

4. Verificação:
- Testei a instalação com:
spark-shell --version

  - Problema: O comando retornou um erro devido à falta do Java JDK 8/11.
  - Solução: Instalei o JDK 11: sudo apt install openjdk-11-jdk

Passo 2: Word Count em Scala (spark-shell)

1. Preparação do Arquivo:
- Baixei um arquivo de texto (`pg2600.txt`) do Project Gutenberg e salvei em `/home/bigdata/datasets`.

2. Execução no spark-shell:

- Iniciei o shell com: spark-shell
- Executei os comandos em scala:
  val texto = sc.textFile("/home/bigdata/datasets/pg2600.txt")
  val palavras = texto.flatMap(line => line.split(" "))
  val contadores = palavras.map(palavra => (palavra, 1)).reduceByKey(_ + _)
  contadores.saveAsTextFile("/home/bigdata/datasets/resultado_wordcount")

- Resultado: O job foi concluído com sucesso, gerando um diretório com os resultados em partes (devido ao processamento distribuído).

Passo 3: Consulta ao CSV de Employees em PySpark

1. Download do Dataset:
- Baixei o arquivo `employees.csv` do Moodle e salvei em `/home/bigdata/datasets`.

2. Execução no pyspark:
  df = spark.read.csv("/home/bigdata/datasets/employees.csv", header=True, sep=';')
  df.show(5) # Mostra as primeiras 5 linhas
  df.summary().show() # Estatísticas descritivas
  df_f = df.filter(df.gender == "F") # Filtro para gênero feminino
  df_f.show(5)

- Observação: Utilizei `df.where()` como alternativa ao `filter()`, conforme instruído.

Passo 4: Uso do PySpark no Jupyter Notebook

1. Instalação do Jupyter e PySpark :
- Instalei o Jupyter e a biblioteca PySpark: pip install jupyter pyspark

2. Configuração da SparkSession :
- Criei um notebook e adicionei:
  from pyspark.sql import SparkSession
  spark = SparkSession.builder.appName("lab3").config("spark.master", "local[*]").getOrCreate()

3. Repetição das Operações :
- Repliquei os comandos do Passo 3 no notebook, verificando a compatibilidade.

Passo 5: Joins entre DataFrames (Employees e Salaries)

1. Carregamento dos CSVs :
- Adicionei o arquivo `salaries.csv` ao diretório de datasets.
- No notebook: df_salaries = spark.read.csv("/home/bigdata/datasets/salaries.csv", header=True, sep=';')

2. Join com SQL :
- Criei uma view temporária e executei uma consulta SQL:
  df.createOrReplaceTempView("employees")

```
df_salaries.createOrReplaceTempView("salaries")
resultado = spark.sql("""
   SELECT e.emp_no, e.first_name, s.salary
   FROM employees e JOIN salaries s ON e.emp_no = s.emp_no
   WHERE e.gender = 'F'
""")
resultado.show(10)
```

Dificuldades e Soluções

- Problema 1 : Erro ao iniciar o PySpark devido à versão do Python.
  - Solução : Defini `export PYSPARK_PYTHON=python3` no `.bashrc`.
- Problema 2 : Lentidão no processamento do CSV grande.
  - Solução : Utilizei o arquivo `employees_small.csv` para testes rápidos.

Conclusão

O Spark foi configurado com sucesso, permitindo a execução de jobs em Scala e Python. As operações básicas (WordCount, filtros, joins) foram validadas, e a integração com Jupyter Notebook facilitou a análise interativa. Para próximos labs, pretendo explorar otimizações de performance e operações mais complexas.