Ver2

Лабораторная работа: Установка и настройка Apache Spark 3.4.3. Простейшие операции и знакомство с функциональностью системы

Цель: Установить и настроить Apache Spark 3.4.3 в Ubuntu, выполнить простейшие операции с экономическими данными.

Необходимое ПО:

- Ubuntu 20.04 LTS или новее
- Java 8 или новее
- Apache Spark 3.4.3
- Python 3.7+
- pip (менеджер пакетов Python)

Алгоритм выполнения:

1. Установка Java
2. Установка Python и pip
3. Установка Apache Spark 3.4.3
4. Настройка переменных окружения
5. Загрузка экономических данных
6. Запуск Spark и выполнение простейших операций

Подробное решение:

1. **Установка Java:**

   ```bash
   sudo apt update
   sudo apt install openjdk-8-jdk
   ```

```
java -version
```

2. **Установка Python и pip:**
```bash
sudo apt install python3 python3-pip
python3 --version
pip3 --version
```

3. **Установка Apache Spark 3.4.3:**
```bash
wget https://downloads.apache.org/spark/spark-3.4.3/spark-3.4.3-bin-hadoop3.tgz
tar xvf spark-3.4.3-bin-hadoop3.tgz
sudo mv spark-3.4.3-bin-hadoop3 /opt/spark
```

4. **Настройка переменных окружения:**
Откройте файл ~/.bashrc:
```bash
nano ~/.bashrc
```
Добавьте следующие строки в конец файла:
```bash
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin
```
Сохраните файл и примените изменения:
```bash
source ~/.bashrc
```

5. **Загрузка экономических данных:**

Создайте директорию для данных и загрузите исторические данные по акциям (например, Apple):

```bash
mkdir ~/spark_data
cd ~/spark_data
wget https://query1.finance.yahoo.com/v7/finance/download/AAPL?period1=0&period2=9999999999&interval=1d&events=history&includeAdjustedClose=true -O AAPL.csv
```

6. **Запуск Spark и выполнение простейших операций:**

Запустите PySpark:

```bash
pyspark
```

В интерактивной оболочке PySpark выполните следующие операции:

a. Загрузка данных:

```python
df = spark.read.csv("file:///home/username/spark_data/AAPL.csv", header=True, inferSchema=True)
df.show(5)
```

b. Подсчет количества строк:

```python
print("Количество строк:", df.count())
```

c. Вывод схемы данных:

```python
df.printSchema()
```

```

d. Базовая статистика:

```python
df.describe().show()
```

e. Фильтрация данных:

```python
df_filtered = df.filter(df["Date"] >= "2020-01-01")
df_filtered.show(5)
```

f. Группировка и агрегация:

```python
from pyspark.sql.functions import year, avg
df_yearly = df.withColumn("Year", year(df["Date"])).groupBy("Year").agg(avg("Close").alias("Avg_Close"))
df_yearly.orderBy("Year").show()
```

g. Создание временного представления и выполнение SQL-запроса:

```python
df.createOrReplaceTempView("stock_data")
spark.sql("SELECT Year(Date) as Year, AVG(Close) as Avg_Close FROM stock_data GROUP BY Year(Date) ORDER BY Year").show()
```

Задание для самостоятельной работы:

1. Загрузите данные по акциям другой компании (например, Microsoft - MSFT).
2. Выполните аналогичный анализ для новых данных.
3. Сравните результаты анализа двух компаний.
```

4. Напишите Spark-приложение, которое находит дни с максимальным объемом торгов для обеих компаний.

Отчет:

Студенты должны подготовить отчет, включающий:

1. Описание процесса установки и настройки Spark.

2. Листинг выполненных команд и их результаты.

3. Анализ полученных результатов.

4. Код и результаты выполнения задания для самостоятельной работы.

5. Выводы о функциональности и возможностях Apache Spark для анализа экономических данных.

Этот лабораторный практикум позволит студентам получить практический опыт работы с Apache Spark, выполнить базовый анализ экономических данных и познакомиться с возможностями распределенных систем для обработки больших объемов данных.