山东大学计算机科学与技术学院

大数据分析实践课程实验报告

学号：202300130220	姓名：刘傲宇	班级：数据科学与大数据技术班
实验题目：spark 实践		
实验学时：2	实验日期：2025/11/14	

实验目标：

本实验旨在介绍学习者如何配置和运行 Apache Spark，以及如何使用 Spark 进行简单的数据处理和分析。实验将涵盖以下内容：

1. 安装和配置 Apache Spark。

2. 运行一个简单的 Spark 应用程序，以理解 Spark 的基本概念。

3. 使用 Spark 进行数据处理和分析。

实验步骤与内容：

1、环境配置

设置 spark、java 环境变量

```
hadoop@ubuntu:~$ java -version
java version "1.8.0_321"
Java(TM) SE Runtime Environment (build 1.8.0_321-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.321-b07, mixed mode)
hadoop@ubuntu:~$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
hadoop@ubuntu:~$ export PATH=$JAVA_HOME/bin:$PATH
hadoop@ubuntu:~$ source ~/.bashrc
hadoop@ubuntu:~$
```

配置、启动 spark 集群

```
hadoop@ubuntu:/usr/local/hadoop/etc/hadoop$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
2025-04-16 17:10:40,142 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
hadoop@ubuntu:/usr/local/hadoop/etc/hadoop$ jps
4369 Jps
2769 org.eclipse.equinox.launcher_1.5.700.v20200207-2156.jar
3874 NameNode
4196 SecondaryNameNode
4011 DataNode
```

2、运行样例代码。

使用 PySpark 对一个销售数据集进行处理，筛选出销售类别为 "Accessories" 的数据，并按日期汇总每一天的销售收入。

```python
from pyspark.sql import SparkSession

# 初始化 SparkSession
spark = SparkSession.builder.appName("SimpleSparkApp").getOrCreate()

# 加载数据
data = spark.read.csv('sales_data.csv',header=True,inferSchema=True)

# 执行一些数据处理操作
result = data.filter(data["Product_Category"] == "Accessories").groupBy("Date").sum("Revenue")

# 显示结果
result.show()
```

运行结果：

```
hadoop@ubuntu:~/Desktop/lay_temp$ python3 1.py
2025-11-17 04:07:23,054 WARN util.Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.126.129 instead (on interface ens33)
2025-11-17 04:07:23,056 WARN util.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).
2025-11-17 04:07:24,548 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
+---------+------------+
|      Date|sum(Revenue)|
+---------+------------+
| 2016/5/12|       21505|
| 2015/7/29|        5994|
|  2013/8/2|       17700|
|2013/11/10|       20464|
|2015/11/24|       28005|
|2015/12/23|       19262|
| 2014/2/28|       28865|
| 2016/3/22|       16684|
|2015/12/19|       30493|
| 2013/11/3|       26902|
| 2016/4/17|       22595|
| 2016/4/25|       19838|
|  2014/1/4|       14332|
| 2014/2/24|       19069|
|  2016/2/4|       17017|
| 2015/10/5|       25070|
| 2014/3/29|       16641|
| 2015/10/8|       22254|
| 2016/6/20|       27200|
| 2013/8/28|       25919|
+---------+------------+
only showing top 20 rows
```

3、使用 spark 进行数据处理和分析。

分析各职业对不同种类的电影关注度，绘制热力图。

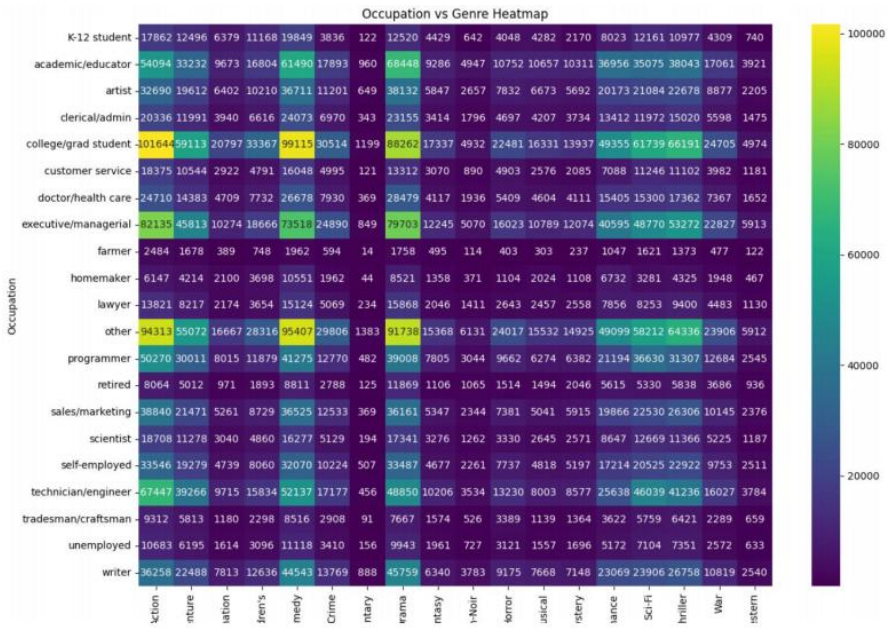
```python
# 解析电影种类
movies = movies.withColumn('genre', explode(split(col('genres'), '[|]')))

# 将数据连接起来
ratings_with_genres = ratings.join(movies, on='movieId').join(users, on='userId')

# 各类职业对不同种类电影的关注度
occupation_genre_counts = ratings_with_genres.groupBy('occupation', 'genre').count().toPandas()
occupation_genre_counts['occupation'] = occupation_genre_counts['occupation'].map(occupation_map)

# 生成透视表
heatmap_data = occupation_genre_counts.pivot_table(index='occupation', columns='genre', values='count', fill_value=0)

# 绘制热力图
plt.figure(figsize=(14, 10))
sns.heatmap(heatmap_data, annot=True, fmt='g', cmap='viridis')
plt.title('Occupation vs Genre Heatmap')
plt.xlabel('Genre')
plt.ylabel('Occupation')
plt.xticks(rotation=90)
plt.savefig('occupation_genre_heatmap.png')
plt.show()
```



4、wordcount 实现。

使用 spark 对在网上下载的一个英文 txt 文件进行词频统计。读取文本文件，对每行进行拆分，统计其中每个单词的出现次数。输出每个单词的计数结果。

```python
from pyspark.sql import SparkSession

# 创建 SparkSession
spark = SparkSession.builder.appName("WordCount").getOrCreate()

# 读取文本文件
text_file = spark.sparkContext.textFile("text.txt")

# 执行 WordCount 操作
word_counts = (text_file
               .flatMap(lambda line: line.split())  # 将每一行按空格拆分成单词
               .map(lambda word: (word, 1))  # 每个单词映射为 (word, 1)
               .reduceByKey(lambda a, b: a + b)  # 对相同的单词进行累加
               )

# 打印结果
for word, count in word_counts.collect():
    print(f"{word}: {count}")

# 关闭 SparkSession
spark.stop()
```

```
hadoop@ubuntu:~/Desktop/lay_temp$ python3 2.py
2025-11-17 04:09:19,465 WARN util.Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.126.129 instead (on interface ens33)
2025-11-17 04:09:19,468 WARN util.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).
2025-11-17 04:09:21,068 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Saying: 1
Good-bye: 1
Cambridge: 3
Again: 1
Very: 2
quietly: 4
take: 2
leave: 2
as: 2
came: 2
in: 4
The: 2
willows: 1
riverside: 1
young: 1
sun;: 1
reflections: 1
shimmering: 1
waves: 2
depth: 1
of: 5
sludge: 1
Sways: 1
leisurely: 1
water;: 1
gentle: 1
would: 1
water: 2
That: 1
shade: 1
elm: 1
Holds: 1
but: 1
rainbow: 1
```

5、使用 PySpark 对销售数据进行分析

求工作日订单量最大的地区：

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, dayofweek

spark = SparkSession.builder \
    .appName("FindMaxWeekdayStoreTraffic") \
    .config("spark.sql.legacy.timeParserPolicy", "LEGACY") \
    .getOrCreate()

file_path = "sales_data.csv"
traffic_data = spark.read.csv(
    file_path,
    header=True,  # 读取表头
    inferSchema=True  # 自动推断字段类型
)

max_traffic_store = (
    traffic_data
    .filter(dayofweek(to_date("Date", "yyyy/MM/dd")).between(2, 6))  # 筛选工作日（周一到周五：2-6）
    .filter("Order_Quantity > 0")  # 排除无效订单
    .groupBy("State")  # 按地区分组
    .sum("Order_Quantity")  # 计算总订单量
    .withColumnRenamed("sum(Order_Quantity)", "total_order_quantity")
    .orderBy("total_order_quantity", ascending=False)  # 按总订单量降序排列
)

print("工作日总订单量最大的地区TOP20（地区-总订单量）：")
max_traffic_store.show(20)  # 显示前20名

spark.stop()  # 停止Spark会话
```

```
hadoop@ubuntu:~/Desktop/lay_temp$ python3 3.py
2025-11-17 04:23:09,563 WARN util.Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.126.129 instead (on interface ens33)
2025-11-17 04:23:09,565 WARN util.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).
2025-11-17 04:23:11,010 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
工作日总订单量最大的地区TOP20（地区-总订单量）：
+------------------+--------------------+
|             State|total_order_quantity|
+------------------+--------------------+
|        California|              191174|
|  British Columbia|              137485|
|           England|              112637|
|        Washington|              100059|
|   New South Wales|               81750|
|            Oregon|               47347|
|          Victoria|               46523|
|        Queensland|               41560|
|          Saarland|               22387|
|Nordrhein-Westfalen|              20190|
|      Seine (Paris)|              19656|
|            Hessen|               18877|
|           Hamburg|               14403|
| Seine Saint Denis|               14265|
|              Nord|               13728|
|   South Australia|               11211|
|            Bayern|               10811|
|    Hauts de Seine|                9165|
|           Essonne|                8751|
|           Yveline|                7746|
+------------------+--------------------+
only showing top 20 rows
```

求每个地区的日均订单量：

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, dayofweek

spark = SparkSession.builder \
    .appName("CalculateStoreDailyAvgTraffic") \
    .config("spark.sql.legacy.timeParserPolicy", "LEGACY") \
    .getOrCreate()

file_path = "sales_data.csv"
traffic_data = spark.read.csv(
    file_path,
    header=True,
    inferSchema=True
)

store_daily_avg = (
    traffic_data
    # 筛选工作日（周一到周五）
    .filter(dayofweek(to_date("Date", "yyyy/MM/dd")).between(2, 6))
    .filter("Order_Quantity > 0")  # 排除无效订单
    .groupBy("State")  # 按地区分组
    .avg("Order_Quantity")  # 计算日均订单量
    .withColumnRenamed("avg(Order_Quantity)", "avg_order_quantity")
    .orderBy("avg_order_quantity", ascending=False)  # 按日均订单量降序排列
)

print("各地区工作日日均订单量TOP20（地区-日均订单量）：")
store_daily_avg.show(20)  # 显示前20名

spark.stop()  # 停止Spark会话

```

```
hadoop@ubuntu:~/Desktop/lay_temp$ python3 4.py
2025-11-17 04:24:31,683 WARN util.Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.126.129 instead (on interface ens33)
2025-11-17 04:24:31,686 WARN util.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).
2025-11-17 04:24:33,192 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
各地区工作日日均订单量TOP20（地区-日均订单量）：
+----------------+------------------+
|           State|avg_order_quantity|
+----------------+------------------+
|         Alabama|              25.5|
|     Mississippi|             22.75|
|       Minnesota|22.333333333333332|
|            Ohio|20.833333333333332|
|         Ontario|20.333333333333332|
|         Montana|20.333333333333332|
|        Virginia|              16.5|
|  North Carolina|              16.5|
|   Pas de Calais|14.061538461538461|
|        New York|              14.0|
|British Columbia|13.666500994035784|
|         Georgia|13.571428571428571|
|          Oregon|12.734534696073158|
|    Val de Marne| 12.63063063063063|
|      Val d'Oise|12.570621468926554|
|      Washington|12.304353172651254|
|           Somme|12.146067415730338|
|         Essonne|12.070344827586206|
|      California|11.973819366153075|
|Seine Saint Denis|11.907345575959933|
+----------------+------------------+
only showing top 20 rows
```

结论分析与体会：

1、本实验主要依托 PySpark 框架对销售数据开展基础的数据处理与分析工作，涵盖数据加载、过滤、分组、聚合、排序等核心操作。实验结果表明，Spark 的分布式计算模型使其具备高效处理大规模数据的能力，且可在短时间内完成复杂的数据分析任务。

2、Spark 实现的优势：

高效的分布式计算能力：Spark 可将数据分布式部署于多节点并执行并行计算，大幅提升计算效率，适用于大规模数据场景。

灵活的 API 体系：Spark 提供了功能强大的 DataFrame 与 RDD API，极大简化了数据分析流程。

3、Spark 实现的局限性：

在处理大规模数据时，资源的高效管理面临挑战，需规避性能瓶颈与数据倾斜问题。针对需执行复杂计算的大数据集，需对集群资源进行合理配置。