# 山东大学计算机科学与技术学院

# 大数据分析与实践课程实验报告

实验题目：spark 实践

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：

以下实验基于 VMware Workstation Ubunbu 64 位系统

```
hadoop@ubuntu:/usr/local/spark$ sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs
/spark-hadoop-org.apache.spark.deploy.master.Master-1-ubuntu.out
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local
/spark/logs/spark-hadoop-org.apache.spark.deploy.worker.Worker-1-ubuntu.out
hadoop@ubuntu:/usr/local/spark$ jps
3573 Jps
3496 Worker
3338 Master
```

Spark 配置成功

J1 Product_Category

| | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Year | Customer_ | Age_Grou | Customer_ | Country | State | ategory | Sub_Categ | Product | Order_Qu | Unit_Cost | Unit_Price | Profit | Cost | Revenue |
| 2 | 2013 | 19 | Youth (<2 | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 8 | 45 | 120 | 590 | 360 | 950 |
| 3 | 2015 | 19 | Youth (<2 | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 8 | 45 | 120 | 590 | 360 | 950 |
| 4 | 2014 | 49 | Adults (35 | M | Australia | New South | Accessorie | Bike Racks | Hitch Rack | 23 | 45 | 120 | 1366 | 1035 | 2401 |
| 5 | 2016 | 49 | Adults (35 | M | Australia | New South | Accessorie | Bike Racks | Hitch Rack | 20 | 45 | 120 | 1188 | 900 | 2088 |
| 6 | 2014 | 47 | Adults (35 | F | Australia | New South | Accessorie | Bike Racks | Hitch Rack | 4 | 45 | 120 | 238 | 180 | 418 |
| 7 | 2016 | 47 | Adults (35 | F | Australia | New South | Accessorie | Bike Racks | Hitch Rack | 5 | 45 | 120 | 297 | 225 | 522 |
| 8 | 2014 | 47 | Adults (35 | F | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 4 | 45 | 120 | 199 | 180 | 379 |
| 9 | 2016 | 47 | Adults (35 | F | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 2 | 45 | 120 | 100 | 90 | 190 |
| 10 | 2014 | 35 | Adults (35 | M | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 22 | 45 | 120 | 1096 | 990 | 2086 |
| 11 | 2016 | 35 | Adults (35 | M | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 21 | 45 | 120 | 1046 | 945 | 1991 |
| 12 | 2013 | 32 | Young Adu | F | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 8 | 45 | 120 | 398 | 360 | 758 |
| 13 | 2015 | 32 | Young Adu | F | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 8 | 45 | 120 | 398 | 360 | 758 |
| 14 | 2013 | 34 | Young Adu | M | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 7 | 45 | 120 | 349 | 315 | 664 |
| 15 | 2015 | 34 | Young Adu | M | Australia | Victoria | Accessorie | Bike Racks | Hitch Rack | 7 | 45 | 120 | 349 | 315 | 664 |
| 16 | 2013 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 5 | 45 | 120 | 369 | 225 | 594 |
| 17 | 2015 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 7 | 45 | 120 | 517 | 315 | 832 |
| 18 | 2013 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 2 | 45 | 120 | 148 | 90 | 238 |
| 19 | 2015 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 1 | 45 | 120 | 74 | 45 | 119 |
| 20 | 2014 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 1 | 45 | 120 | 74 | 45 | 119 |
| 21 | 2016 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 1 | 45 | 120 | 74 | 45 | 119 |
| 22 | 2014 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 6 | 45 | 120 | 443 | 270 | 713 |
| 23 | 2016 | 29 | Young Adu | M | Canada | British Col | Accessorie | Bike Racks | Hitch Rack | 8 | 45 | 120 | 590 | 360 | 950 |

实验指导书中提供的 sales_data.csv 文件

（一）Spark 测试（基于 sales_data.csv 文件）

Open spark_test.py ~/Desktop

```python
from pyspark.sql import SparkSession

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

# 加载数据
# 显式指定本地文件系统，替换为你的csv文件实际绝对路径
data = spark.read.csv("file:///home/hadoop/Desktop/sales_data.csv", header=True, inferSchema=True)

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

# 显示结果
result.show()
```

Python 代码

```
hadoop@ubuntu:/usr/local/spark$ ./bin/spark-submit /home/hadoop/Desktop/spark_test.py
2025-12-14 20:09:39,105 WARN util.Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.26.128 instead (on interface ens33)
2025-12-14 20:09:39,106 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
2025-12-14 20:09:39,952 INFO spark.SparkContext: Running Spark version 3.2.1
2025-12-14 20:09:40,061 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2025-12-14 20:09:40,207 INFO resource.ResourceUtils: ==============================================================
2025-12-14 20:09:40,207 INFO resource.ResourceUtils: No custom resources configured for spark.driver.
2025-12-14 20:09:40,208 INFO resource.ResourceUtils: ==============================================================
2025-12-14 20:09:40,208 INFO spark.SparkContext: Submitted application: SimpleSparkApp
2025-12-14 20:09:40,231 INFO resource.ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , me
sources: Map(cpus -> name: cpus, amount: 1.0)
2025-12-14 20:09:40,242 INFO resource.ResourceProfile: Limiting resource is cpu
2025-12-14 20:09:40,243 INFO resource.ResourceProfileManager: Added ResourceProfile id: 0
2025-12-14 20:09:40,328 INFO spark.SecurityManager: Changing view acls to: hadoop
2025-12-14 20:09:40,328 INFO spark.SecurityManager: Changing modify acls to: hadoop
2025-12-14 20:09:40,332 INFO spark.SecurityManager: Changing view acls groups to:
2025-12-14 20:09:40,333 INFO spark.SecurityManager: Changing modify acls groups to:
2025-12-14 20:09:40,333 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(hadoop); groups with
2025-12-14 20:09:40,557 INFO util.Utils: Successfully started service 'sparkDriver' on port 35007.
2025-12-14 20:09:40,619 INFO spark.SparkEnv: Registering MapOutputTracker
2025-12-14 20:09:40,674 INFO spark.SparkEnv: Registering BlockManagerMaster
2025-12-14 20:09:40,712 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
2025-12-14 20:09:40,713 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
2025-12-14 20:09:40,716 INFO spark.SparkEnv: Registering BlockManagerMasterHeartbeat
2025-12-14 20:09:40,749 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-fd01f8aa-d0c1-466d-b871-ea812d9ae989
2025-12-14 20:09:40,771 INFO memory.MemoryStore: MemoryStore started with capacity 366.3 MiB
2025-12-14 20:09:40,816 INFO spark.SparkEnv: Registering OutputCommitCoordinator
2025-12-14 20:09:40,918 INFO util.log: Logging initialized @2467ms to org.sparkproject.jetty.util.log.Slf4jLog
2025-12-14 20:09:41,015 INFO server.Server: jetty-9.4.43.v20210629; built: 2021-06-30T11:07:22.254Z; git: 526006ecfa3af7f1a27ef3a288e2bef7ea9dd7e8; jvm 1.8.0_321-b07
2025-12-14 20:09:41,039 INFO server.Server: Started @2590ms
2025-12-14 20:09:41,084 INFO server.AbstractConnector: Started ServerConnector@1a52fa0f{HTTP/1.1, (http/1.1)}{0.0.0.0:4040}
2025-12-14 20:09:41,084 INFO util.Utils: Successfully started service 'SparkUI' on port 4040.
2025-12-14 20:09:41,110 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@13597adb{/jobs,null,AVAILABLE,@Spark}
2025-12-14 20:09:41,123 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@447c3c9d{/jobs/json,null,AVAILABLE,@Spark}
2025-12-14 20:09:41,129 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@2cfe65c2{/jobs/job,null,AVAILABLE,@Spark}
```

运行

（Spark 环境启动正常，Spark 版本 3.2.1 成功加载，绑定到 192.168.26.128:4040（SparkUI 地址），Executor 正常启动。

虽然有两个 WARN 警告：(1)hostname ubuntu resolves to a loopback address：只是 Spark 自动切换到实际网卡 IP（192.168.26.128），不影响运行。

（2）Unable to load native-hadoop library：Hadoop 的本地库（针对不同系统的优化库）未加载，仅影响性能，不影响功能执行。

CSV 文件读取成功，Spark 成功读取了 file:///home/hadoop/Desktop/sales_data.csv（本地文件），并将文件分成 2 个分区处理（符合大文件的分区逻辑）。所有 Task（任务）都成功完成（TID 0/1/2/3/4/5 均执行完毕，无失败）。

）

```
+----------+------------+
|      Date|sum(Revenue)|
+----------+------------+
| 2016/5/12|       13912|
| 2015/7/29|        6490|
|2015/11/24|       13714|
|  2013/8/2|        8123|
|2013/11/10|       12952|
|2015/12/23|       19417|
| 2016/3/22|        9150|
| 2014/2/28|        6932|
| 2013/11/3|       17722|
|2015/12/19|       12274|
|  2014/1/4|       11336|
| 2016/4/25|       10335|
| 2014/2/24|        7127|
| 2016/4/17|       10842|
| 2015/10/8|       16996|
| 2015/10/5|       12079|
| 2016/6/20|       12048|
| 2014/3/29|        4793|
|  2016/2/4|        6060|
| 2014/6/17|       11842|
+----------+------------+
only showing top 20 rows
```

运行结果

（二）Wordcount（基于下面的文档）

Open webpage1.txt ~/Desktop

```
<!DOCTYPE html>
<html><head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>2025 a crucial year for AI applications - Opinion - Chinadaily.com.cn</title>
    <meta name="keywords" content="AI,technology,strategy,applications,DeepSeek">
    <meta name="description" content="The Government Work Report on Wednesday highlighted new achievements including integrated circuits, artificial intelligence (AI), quantu

      <meta property="og:xi" content="0">
      <meta property="og:title" content="2025 a crucial year for AI applications">
      <meta property="og:recommend" content="0">
      <meta property="og:url" content="https://www.chinadaily.com.cn/a/202503/06/WS67c8d828a310c240449d8e3d.html">
      <meta property="og:image" content="http://img2.chinadaily.com.cn/images/202503/06/67c90191a310c240e241ebf4.jpeg">

      <meta name="twitter:card" content="summary_large_image">
      <meta name="twitter:site" content="@ChinaDailyApp">
      <meta name="twitter:title" content="2025 a crucial year for AI applications">
      <meta name="twitter:description" content="The Government Work Report on Wednesday highlighted new achievements including integrated circuits, artificial intelligence (A

        <meta name="twitter:image:src" content="http://img2.chinadaily.com.cn/images/202503/06/67c90191a310c240e241ebf6.jpeg">

        <meta name="twitter:image" content="http://img2.chinadaily.com.cn/images/202503/06/67c90191a310c240e241ebf6.jpeg">

      <meta name="shareImg" content="//img2.chinadaily.com.cn/images/202503/06/67c90191a310c240e241ebf4.jpeg">
      <meta name="msapplication-TileImage" content="https://img2.chinadaily.com.cn/static/common/img/sharelogo2.jpg">
      <link rel="apple-touch-icon-precomposed" href="https://img2.chinadaily.com.cn/static/common/img/sharelogo2.jpg">

    <link rel="stylesheet" type="text/css" href="webpage1_files/layout.css">
    <link rel="stylesheet" type="text/css" href="webpage1_files/r-public.css">
    <link rel="stylesheet" type="text/css" href="webpage1_files/bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="webpage1_files/r-content.css">
    <link rel="stylesheet" type="text/css" href="webpage1_files/opinion.css">
    <link rel="stylesheet" charset="utf-8" href="webpage1_files/share_encontent.css">
    <script src="webpage1_files/html5shiv.js"></script>
    <script src="webpage1_files/respond.min.js"></script>
    <link href="https://img2.chinadaily.com.cn/h5hack/respond-proxy.html" id="respond-proxy" rel="respond-proxy">
    <link href="https://www.chinadaily.com.cn/h5hack/respond.proxy.gif" id="respond-redirect" rel="respond-redirect">
    <script src="webpage1_files/respond.proxy.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
  <script type="text/javascript" charset="utf-8" rel="stylesheet" src="webpage1_files/jquery.min.js"></script>
    <script type="text/javascript" src="webpage1_files/qrcode.min.js"></script><!--二维码生成-->
    <script type="text/javascript" charset="utf-8" rel="stylesheet" src="webpage1_files/taboolaHead.js"></script>
    <script language="javascript" type="text/javascript" src="webpage1_files/toucher.js"></script>
    <script type="text/javascript" charset="utf-8" rel="stylesheet" src="webpage1_files/public_content.js"></script>
    <script type="text/javascript" charset="utf-8" rel="stylesheet" src="webpage1_files/share.js"></script>
    <script type="text/javascript" charset="utf-8" rel="stylesheet" src="webpage1_files/jweixin-1.0.0.js"></script>
    <script src="webpage1_files/jsonps.js"></script><script type="text/javascript" src="webpage1_files/jweixin-1.6.0.js" id="weixin-init"></script>
    <!--[if lt IE 9]>
```

待统计的文档

Open spark_wordcount.py ~/Desktop

```python
# spark_wordcount.py
from pyspark.sql import SparkSession
from pyspark.sql.functions import split, explode, lower, regexp_replace, col, count

spark = SparkSession.builder.appName("WordCountTop30").master("local[*]").getOrCreate()

# 读取文件
file_path = "file:///home/hadoop/Desktop/webpage1.txt"
df = spark.read.text(file_path)

# 数据处理：转小写、去除标点、拆分单词、过滤空单词
df_clean = df.withColumn("value", regexp_replace(lower(col("value")), r"[^a-zA-Z0-9\s]", ""))
df_words = df_clean.withColumn("word", explode(split(col("value"), r"\s+")))
df_filtered = df_words.filter(col("word") != "")

# 统计词频并按词频降序排序
word_count_df = df_filtered.groupBy("word").agg(count("*").alias("count"))
sorted_word_count = word_count_df.orderBy(col("count").desc())

# 输出前20个数量最多的单词
print("前20个出现次数最多的单词：")
sorted_word_count.limit(20).show(truncate=False)

# 停止SparkSession
spark.stop()
```

Python 代码

（首先导入 SparkSession 和数据处理所需的函数，创建并启动本地模式的 SparkSession（任务命名为 WordCountTop20）；接着通过 file://协议读取指定路径下的文本文件，生成以每行文本为记录的 DataFrame；然后对文本进行清洗处理，先将所有字符转小写并通过正则表达式去除标点（仅保留字母、数字和空格），再按空格拆分文本为单词数组并通过 explode 行转列得到单个单词，过滤掉拆分后出现的空单词；之后按单词分组统计出现次数（命名为 count 列），并按 count 列降序排序；最后打印提示语并展示排序后的前 20 个单词（完整显示无截断），完成后关闭 SparkSession 释放资源。）

```
hadoop@ubuntu:/usr/local/spark$ ./bin/spark-submit /home/hadoop/Desktop/spark_wordcount.py
```

运行

```
+------------------+-----+
|word              |count|
+------------------+-----+
|div               |183  |
|a                 |79   |
|script            |49   |
|li                |47   |
|targetblank       |45   |
|the               |37   |
|and               |37   |
|ai                |33   |
|to                |33   |
|meta              |29   |
|in                |26   |
|style             |26   |
|of                |25   |
|typetextjavascript|24   |
|for               |22   |
|lia               |22   |
|var               |17   |
|large             |16   |
|models            |15   |
|new               |13   |
+------------------+-----+
only showing top 20 rows
```

运行结果

（三）查询 Revenue 最大的前 20 条记录对应的日期（基于 sales_data.csv 文件）

Open spark_1.py ~/Desktop

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

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

# 加载数据
data = spark.read.csv("file:///home/hadoop/Desktop/sales_data.csv", header=True, inferSchema=True)


top20_revenue_rows = data.filter((data["Product_Category"] == "Clothing") & data["Revenue"].isNotNull()).orderBy(col("Revenue").desc()).limit(20).select("Date", "Revenue")

# 显示结果
print("Revenue最大的前20条记录：")
top20_revenue_rows.show(truncate=False)

# 停止SparkSession
spark.stop
```

代码

```
hadoop@ubuntu:/usr/local/spark$ ./bin/spark-submit /home/hadoop/Desktop/spark_1.py
```

运行

```
+----------+-------+
|Date      |Revenue|
+----------+-------+
|2015/7/24 |58074  |
|2013/7/24 |54069  |
|2015/11/27|2148   |
|2015/10/2 |2127   |
|2015/10/19|2127   |
|2015/9/19 |2127   |
|2016/5/12 |2127   |
|2015/12/25|2127   |
|2015/9/17 |2127   |
|2016/4/20 |2127   |
|2015/11/26|2127   |
|2016/6/29 |2079   |
|2016/5/4  |2079   |
|2013/8/5  |2079   |
|2013/12/2 |2079   |
|2014/6/29 |2079   |
|2013/10/1 |2079   |
|2013/11/1 |2079   |
|2014/4/13 |2079   |
|2014/5/4  |2079   |
+----------+-------+
```

运行结果

（四）查询不同 Product_Category 的平均 Revenue（基于 sales_data.csv 文件）

Open spark_2.py ~/Desktop

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg

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

# 加载数据
data = spark.read.csv("file:///home/hadoop/Desktop/sales_data.csv", header=True, inferSchema=True)

avg_revenue_by_category = data.filter(data["Revenue"].isNotNull()) \
    .groupBy("Product_Category") \
    .agg(avg("Revenue").alias("avg_revenue")) \
    .orderBy(col("avg_revenue").desc())

# 显示结果
print("不同Product_Category的平均Revenue：")
avg_revenue_by_category.show(truncate=False)

# 停止SparkSession
spark.stop()
```

代码

```
hadoop@ubuntu:/usr/local/spark$  ./bin/spark-submit /home/hadoop/Desktop/spark_2.py
```

运行

```
+----------------+------------------+
|Product_Category|avg_revenue       |
+----------------+------------------+
|Bikes           |2377.882149180202 |
|Clothing        |494.3239636234794 |
|Accessories     |215.60171135196805|
+----------------+------------------+
```

运行结果

结论分析与体会：
Spark 环境配置验证成功：通过启动 Spark 服务并查看进程，确认 Master 和 Worker 进程正常运行，Spark 应用程序能够通过 spark-submit 命令成功提交，且能够正常读取本地文件并进行数据处理。
数据处理功能实现：
基于销售数据 sales_data.csv，成功完成了特定产品类别 Revenue 按日期汇总、Revenue Top20 记录查询、不同产品类别平均 Revenue 计算，所有查询结果均符合预期逻辑。
基于网页文本数据 webpage1.txt，成功实现了单词计数功能，完成了文本清洗、单词拆分、词频统计等一系列操作，准确输出了出现次数最多的前 20 个单词。
Spark 特性体现：实验过程中，Spark 自动对数据进行分区处理（如 sales_data.csv 被分为 2 个分区），提高了数据处理效率；同时，即使出现部分警告（如 Hadoop 本地库未加载），也不影响核心功能的执行，体现了 Spark 良好的兼容性和稳定性。
深入熟悉了 Spark 基于 DataFrame 和 Spark SQL 的数据处理流程，掌握了 SparkSession 的初始化、数据读取、数据清洗、分组统计、排序筛选等核心操作。同时，理解了 Spark 作为分布式计算框架的基本特性，体会到其在处理结构化数据（如 CSV）和非结构化数据（如文本）时的灵活性和高效性。本次实验为后续更复杂的大数据处理场景（如分布式数据存储、流式数据处理）奠定了坚实的基础。