用户购物行为数据分析实验

1 实验介绍

1.1 实验环境

> 华为云服务器x3: server0 server1 server2（请参考在linux配置实验中自己命名的节点序号与名称）
>
> 系统：CentOS 7.8
>
> JDK：1.8.0_341
>
> Hadoop环境：
>
> 版本：2.7.7
>
> 结点部署情况：
>
> - server0: Namenode, DataNode, NodeManager
> - server1: DataNode, ResourceManager, NodeManager
> - server2: DataNode, SecondaryNamenode, NodeManager
>
> MySQL：MySQL 5.7.30
>
> Hive：Apache Hive 2.1.1

1.2 实验描述

本次实验在之前搭建Hadoop集群的基础之上，将数据存储在分布式文件系统HDFS上，使用Hive工具进行数据查询，MapReduce进行分布式数据计算，Python进行数据可视化分析。

本实验涉及数据预处理、存储、查询和可视化分析等数据处理全流程所涉及的各种典型操作，通过对网站用户购物行为数据的分析，了解大数据全流程的操作。

在开始本实验前请**确保已完成Hive的安装与部署。**

1.3 实验数据集

本实验数据集来自脱敏后的真实淘宝用户行为数据，原始数据集共包含1亿条，本实验采用其子集数据。数据表结构如下：

列名称	说明
user_id	整数类型，序列化后的用户ID
item_id	整数类型，序列化后的商品ID
category_id	整数类型，序列化后的商品类目ID
behavior_type	字符串枚举类型，包括（'pv','buy','cart','fav')
timestamp	时间戳类型，该行为发生的时间

其中，行为类型解释如下：

行为类型	说明
pv	浏览商品详情页
buy	购买商品
cart	将商品加入购物车
fav	收藏商品

本次实验提供了small.csv与big.csv两种规模的数据集，本教程将使用small.csv做示范。

1.4 实验内容

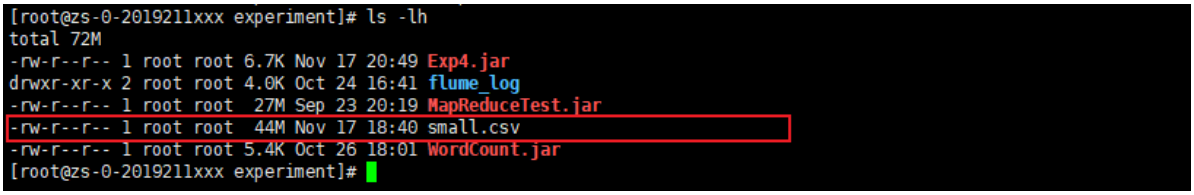
1. 将数据集上传到数据仓库Hive
2. 使用Hive进行数据的简单分析
3. 使用MapReduce进行数据的分布式处理
4. 使用Python进行数据的可视化分析

2 实验步骤

2.1 加载数据集到Hive

2.1.1 数据集上传与预处理

将附件中的数据集small.csv上传到server0， 并将其放到/root/experiment目录下



注：由于文件内容较大，本实验所有数据预览操作都只**打印部分数据到终端**

查看数据集的前五行 退出Hive执行

```
head -5 small.csv
```

删除文件的第一行表头记录

```
sed -i '1d' small.csv
```

再次执行head命令，查看是否删除成功

2.1.2 上传数据集到HDFS

1. 启动Hadoop集群

server0：

```
start-dfs.sh
```

server1：

```
start-yarn.sh
```

注：根据hdfs与yarn的配置情况执行

2. 在HDFS创建/dataset目录

```
hadoop fs -mkdir /dataset/
```

3. 将本地数据集上传到HDFS的/dataset目录下

```
hadoop fs -put /root/experiment/small.csv /dataset/
```

4. 查看HDFS中数据集的内容

```
hadoop fs -cat /dataset/small.csv | head -5
```

2.1.3 Hive数据仓库存储数据

1. 进入Hive交互终端

```
hive
```

2. 创建数据库taobao，并切换到taobao数据库

```
create database taobao;
use taobao;
```

3. 创建外部表user_info，数据来源为HDFS文件系统

```
create external table user_info (user_id int,item_id int,category_id
int,behavior_type string,visit_time timestamp)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/dataset';
```

4. 查看user_info 表的结构

```
desc user_info;
```

2.2 使用Hive进行数据查询

> 验收点1：在查询3，4，5，6中任选2条查询进行展示

1. **查询user_info 前10 条数据**

```
select * from user_info limit 10;
```

2. **查询前20 位用户淘宝操作的时间和对应的行为信息**

```
select visit_time, behavior_type from user_info limit 20;
```

3. **统计数据记录的总条数**

```
select count(*) from user_info;
```

4. **统计不同用户的个数**

```
select count(distinct user_id) from user_info;
```

5. **统计每天网站卖出去商品个数**

```
select to_date(visit_time),count(user_id) from user_info where
behavior_type='buy' group by to_date(visit_time);
```

6. **查询2017年12月02日当天淘宝用户购买总量**

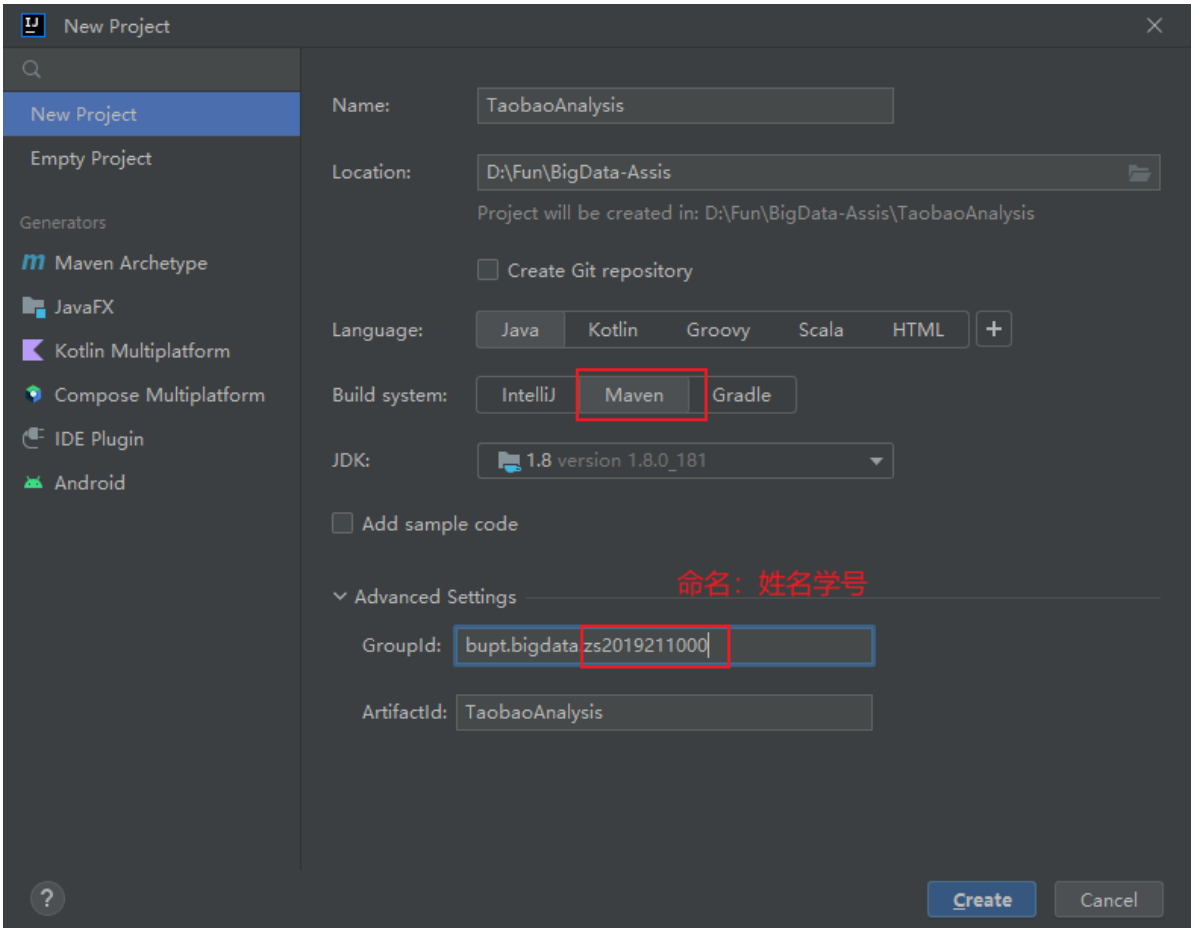
```
select count(*) from user_info
where behavior_type='buy' and to_date(visit_time)='2017-12-02';
```

2.3 使用MapReduce分析

本次使用MapReduce主要完成以下两个任务：

- 统计不同类别商品的销量
- 统计不同日期商品的销量

2.3.1 使用IntelliJ IDEA新建maven项目



修改pom.xml，导入所需依赖

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <!-- 根据自己的项目名修改 -->
    <groupId>bupt.bigdata.zs2019211000</groupId>
    <artifactId>TaobaoAnalysis</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>
    <!-- 配置阿里云仓库 -->
    <repositories>
        <repository>
            <id>aliyun-repos</id>
            <url>https://maven.aliyun.com/repository/public</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
    <pluginRepositories>
```

```
        <pluginRepository>
            <id>aliyun-repos</id>
            <url>https://maven.aliyun.com/repository/public</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j-impl</artifactId>
            <version>2.19.0</version>
        </dependency>
        <!-- map-reduce 程序需要的依赖-->
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>2.7.7</version>
        </dependency>
    </dependencies>

</project>
```

2.3.2 创建bupt.bigdata.zs2019211000.taobao包，并在该包内添加如下classes，注意修改package

ItemMapper类

```
package bupt.bigdata.zs2019211000.taobao;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
/**
 * 统计每类商品的销量
 */
public class ItemMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    // 输出键：商品类别
    private Text outK = new Text();
    // 输出值：1
    private IntWritable outV = new IntWritable(1);
    /**
     * key 输入键：该行字符串的首字符在文件中的位置
     * value 输入值：该行内容，如 <1,2333346,2520771,pv,2017-11-24 22:15:33>
     * context 上下文变量
```

```
     */
    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        // 获取单行数据
        String line = value.toString();
        // 分割数据
        String[] record = line.split(",");
        // 排除非购买行为
        if (!record[0].startsWith("u") && "buy".equals(record[3])) {
            // 获取商品类别
            outK.set(record[2]);
            // 记为该商品类别出现一次
            context.write(outK, outV);
        }
    }
}
```

DayMapper类

```
package bupt.bigdata.zs2019211000.taobao;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
/**
 * 统计每天商品的销量
 */
public class DayMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    // 输出键：日期（天）
    private Text outK = new Text();
    // 输出值：1
    private IntWritable outV = new IntWritable(1);
    /**
     * key 输入键：该行字符串的首字符在文件中的位置
     * value 输入值：该行内容，如 <1,2333346,2520771,pv,2017-11-24 22:15:33>
     * context 上下文变量
     */
    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        // 获取单行数据
        String line = value.toString();
        // 分割数据
        String[] record = line.split(",");
        // 排除非购买行为
        if ("buy".equals(record[3])) {
            String day = record[4].split(" ")[0];
            // 获取日期
            outK.set(day);
            // 记为该天有一次成交记录
            context.write(outK, outV);
        }
    }
}
```

CommonReducer类

```
package bupt.bigdata.zs2019211000.taobao;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

/**
 * 将Map结果进行Reduce操作以获得最终结果
 */
public class CommonReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
    // 统计变量
    private int sum;
    // 输出键值：销量
    private IntWritable outV = new IntWritable();
    /**
     * key 输入键：商品类别 / 日期
     * values 输入值：该键对应的值的集合(全1)
     * context 上下文变量
     */
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context
context) throws
            IOException, InterruptedException {
        // 统计销量
        sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        outV.set(sum);
        // 获取该键对应的销量
        context.write(key,outV);
    }
}
```

Driver类

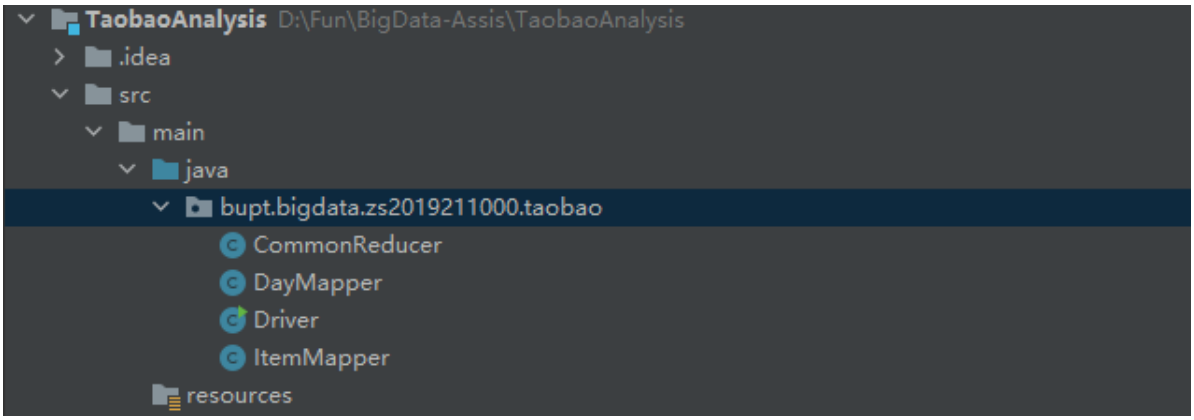
```
package bupt.bigdata.zs2019211000.taobao;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;

public class Driver {
    public static void main(String[] args) throws IOException,
            ClassNotFoundException, InterruptedException {
        // 1.获取job
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
```

```
        // 2.设置jar包路径
        job.setJarByClass(Driver.class);
        // 3.关联 Mapper 和 Reducer ,可执行两种 MapReduce 任务：由参数传入
        // item 计算不同类别商品的销量
        // day 计算不同日期商品的销量
        if ("item".equalsIgnoreCase(args[0])){
            job.setMapperClass(ItemMapper.class);
        }else if ("day".equalsIgnoreCase(args[0])){
            job.setMapperClass(DayMapper.class);
        }else{
            throw new IllegalArgumentException("Please choice a valid task type
(item or day)");
        }
        job.setReducerClass(CommonReducer.class);
        // 4.设置 Mapper 输出的 K V 类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        // 5.设置 最终 输出的 K V 类型
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        // 6.设置输入文件路径
        FileInputFormat.setInputPaths(job, new Path(args[1]));
        // 7.设置输出路径
        FileOutputFormat.setOutputPath(job, new Path(args[2]));
        // 8.任务提交
        boolean result = job.waitForCompletion(true);
        System.exit(result ? 0 : 1);
    }
}
```

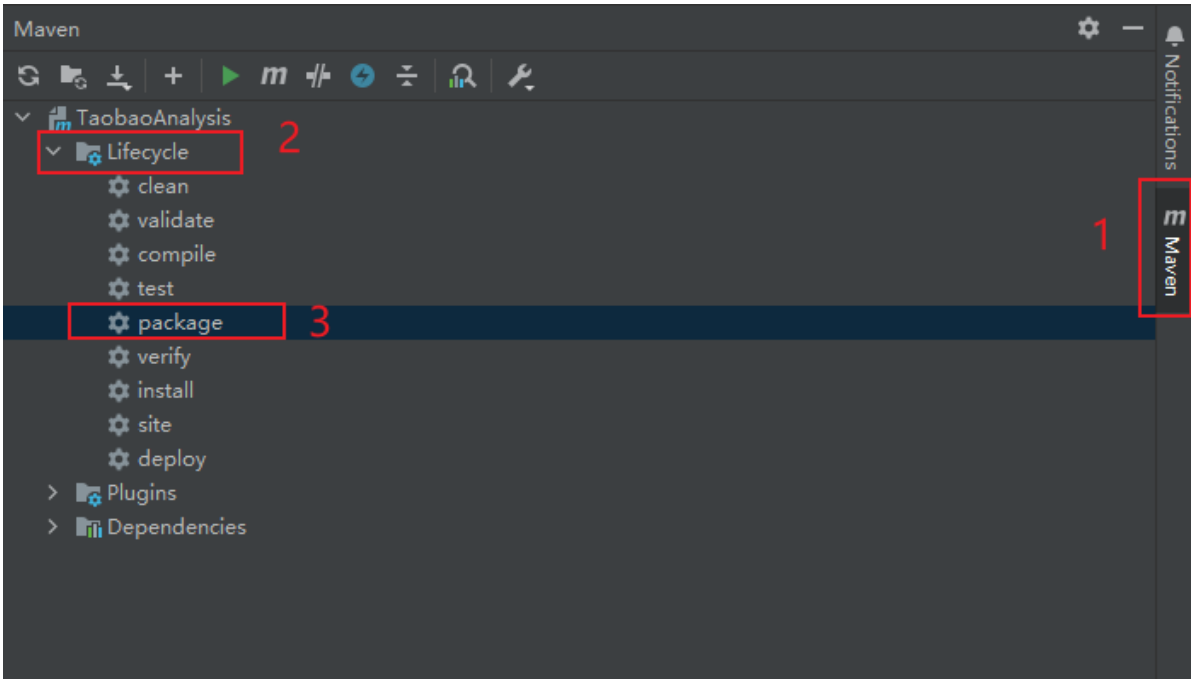
工程结构参考：



2.3.3 打包并上传项目

使用IDEA的Maven窗口提供的工具进行打包

- 点击Lifecycle -> Package
- 出现build success后表示打包成功，生成的jar包位于/target目录中，将jar包重命名为“TaobaoAnalysis.jar”



打包成功后，将jar包上传至server0的/root/experiment目录下：

```
[root@zs-0-2019211xxx experiment]# ls -lh
total 72M
drwxr-xr-x 2 root root 4.0K Oct 24 16:41 flume_log
-rw-r--r-- 1 root root  27M Sep 23 20:19 MapReduceTest.jar
-rw-r--r-- 1 root root  44M Nov 17 18:40 small.csv
-rw-r--r-- 1 root root 7.0K Nov 18 22:04 TaobaoAnalysis.jar
-rw-r--r-- 1 root root 5.4K Oct 26 18:01 WordCount.jar
[root@zs-0-2019211xxx experiment]#
```

2.3.4 执行MapReduce程序

> 验收点2：任意演示1次MapReduce，并查看hdfs中的结果输出

1）统计不同类别商品的销量

```
hadoop jar ./TaobaoAnalysis.jar bupt.bigdata.zs2019211000.taobao.Driver item
/dataset/small.csv /output1
```

2）统计不同日期商品的销量

```
hadoop jar ./TaobaoAnalysis.jar bupt.bigdata.zs2019211000.taobao.Driver day
/dataset/small.csv /output2
```

提示：注意修改命令中的包名

3）查看结果

```
hadoop fs -cat /output1/part-r-00000
hadoop fs -cat /output2/part-r-00000
```

2.4 使用Python进行可视化分析

该部分可以在本地Python环境下完成，实验用到Python依赖包主要有pandas和matplotlib，建议使用jupyter notebook进行编码和结果展示。实验提供的Python可视化代码比较简单，仅供参考。

> 验收点3：任意展示2~3个可视化结果

1. 读取数据集

```
import pandas as pd
import matplotlib.pyplot as plt
#jupyter内嵌绘图
%matplotlib inline

df = pd.read_csv("small.csv")
df.columns = ['user_id', 'item_id', 'category_id', 'behavior_type', 'timestamp']
df.head
```

2. 可视化四种行为的数量比例

```
behavior_count = df["behavior_type"].value_counts()
behavior_count.plot.pie(figsize=(10,10), autopct='%.2f', fontsize=15)
```

3. 可视化销量前十的商品类别

```
item_count = df[df['behavior_type']=="buy"]["category_id"].value_counts()
item_count = item_count[:10]
item_count.plot(kind="bar",figsize=(10,10))
```

4. 可视化每个月的商品总销量

```
df['timestamp'] = pd.to_datetime(df['timestamp'])
df["sale_count"]= df["behavior_type"].apply(lambda x: 1 if x == "buy" else 0)
sale_month_count =
df[["timestamp","sale_count"]].groupby(df['timestamp'].apply(lambda
x:x.month)).sum()
sale_month_count.index.name = "month"
print(sale_month_count)
sale_month_count.plot(kind="bar",figsize=(10,10))
```

2.5 扩展实验

1. 根据自己对大数据流程和用户购物行为数据的理解，挖掘其他更有价值的信息。
2. 完善可视化分析的图表类型、图表属性、配色、文字格式等。使得可视化的结果更加规范、美观，能展现数据的本质信息。

3 附录

3.1 附件说明

文件名	说明
mysql-5.7.30.tar.gz	mysql-5.7.30 rpm源
mysql-connector-java-5.1.5.jar	Java连接MySQL驱动包
small.csv	脱敏淘宝用户数据集（小）
big.csv	脱敏淘宝用户数据集（大）