第7章综合项目实训

CH07实训1：Spark综合实训项目

训练要点

1. 回顾并熟练使用python进行数据采集
2. 掌握scala的使用，将数据从mongo采集到hdfs
3. 熟练掌握使用spark streaming实现对hdfs目录监测并完成数据分析与处理
4. 熟练spark的使用，将分析结果存储到mysql
5. 训练数据据的可视化，将mysql的数据取出并完成可视化

需求说明

1. 本实训充许同学们采集各类题材数据,包括并不限于:商品、音乐、新闻、房产、书籍、招聘
2. 本实训要实现的功能是通过同学采集某类题材数据，实时采集题材数据到mongodb, 再从mongodb将所有同学采集的同题材数据采集hdfs，然后实现该类数据的实时流分析，对分析结果进行存储，然后对mysq中数据实时可视化

实现步骤

1. 数据采集：使用scrapy框架实现某类题材网站的数据采集，存入mongo数据库
2. 数据转存：scala实时采集题材数据从mongo到hdfs
3. 数据分析：启动Spark Streaming监控hdfs目录，分析统计数据
4. 数据存储：使用spark将统计结果转存到mysql中
5. 数据可视：使用python将mysql的结果数据每隔几秒显示出来并更新到web上

作业要求

1. 对实训当中要实现的功能进行描述、架构设计、详细设计，整理入实训报告；
2. 对实训过程中源码、操作步骤、运行结果截图，整理入实训报告；
3. 整理上述过程中实现的源码，包括采集的源码，spark分析源码，可视化的源码，全部打包成一个rar文件。

环境准备

采集环境准备

chrome和chromedriver下载

方式一，chrome和chromedriver版本要配套，请从百度网盘上下载：

```
百度网盘下载
链接：https://pan.baidu.com/s/1FF7HHQi9y2kVBXom_Omd_g
提取码：8ft4
在google目录下有chrome的109版本和chromedriver的109版
```

方式二，使用链接下载

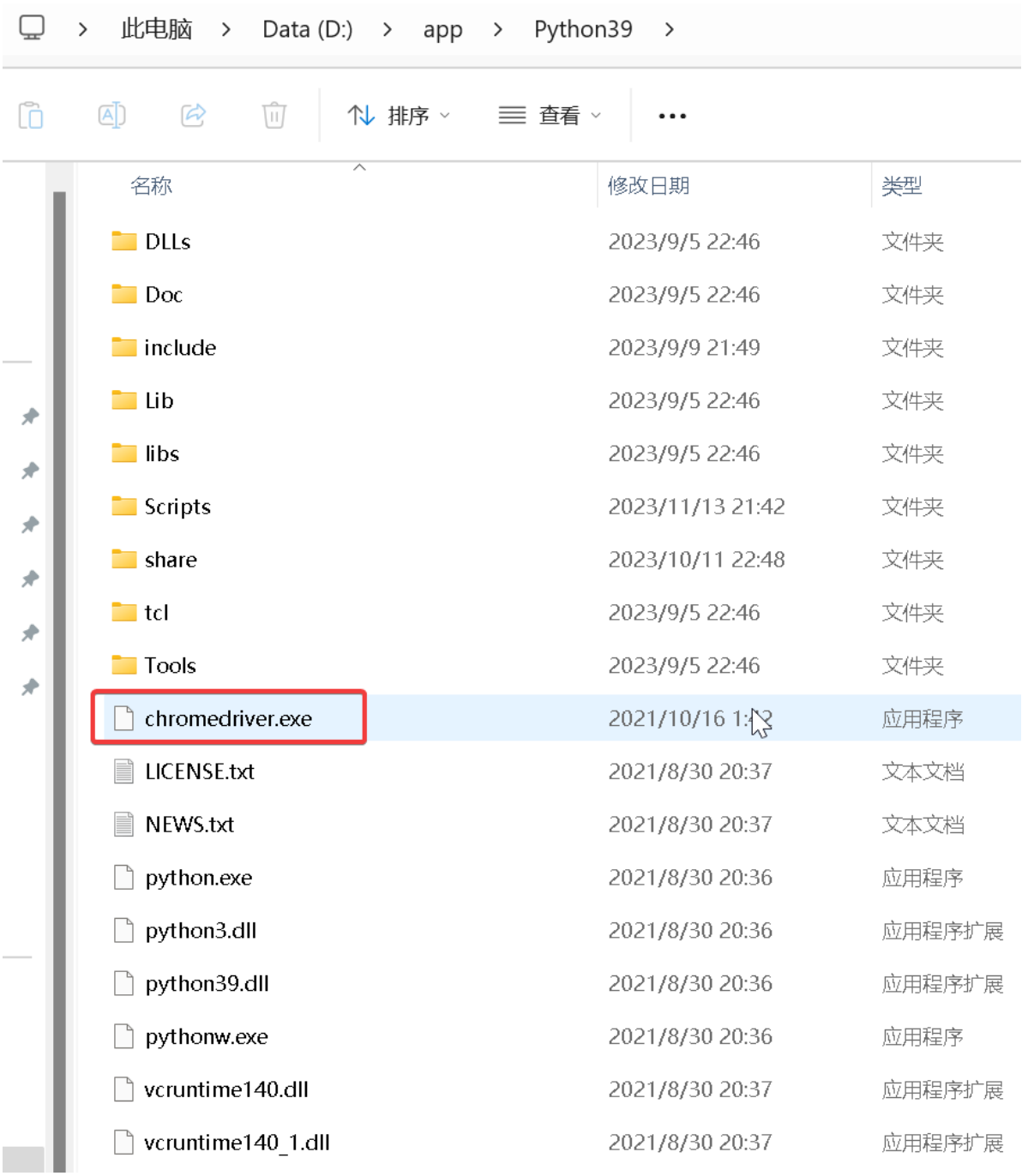
```
链接下载google 109版本
https://dl.google.com/release2/chrome/ad2uwza6rxngw4rvu7lrmj5rvtca_109.0.5414.75/109.0.5414.75_chrome_installer.exe
链接下载chromedriver版本109
https://registry.npmmirror.com/-/binary/chromedriver/109.0.5414.74/chromedriver_win32.zip
```

方式二，使用当前最新的chrome版本，如119,120, 去下载最新的驱动

```
chromedriver最新版本下载：
https://googlechromelabs.github.io/chrome-for-testing/
```

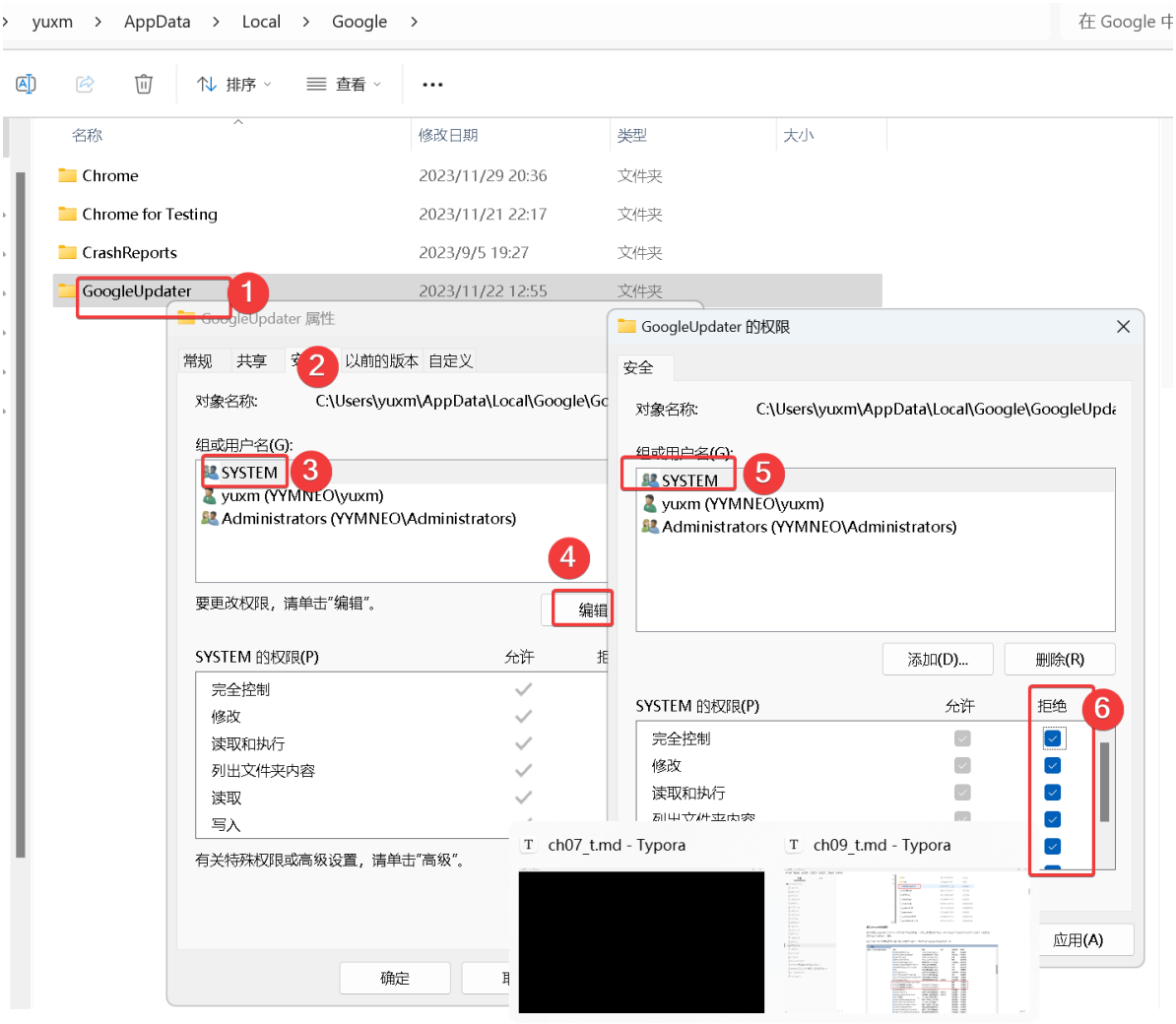
chromedriver.exe部署

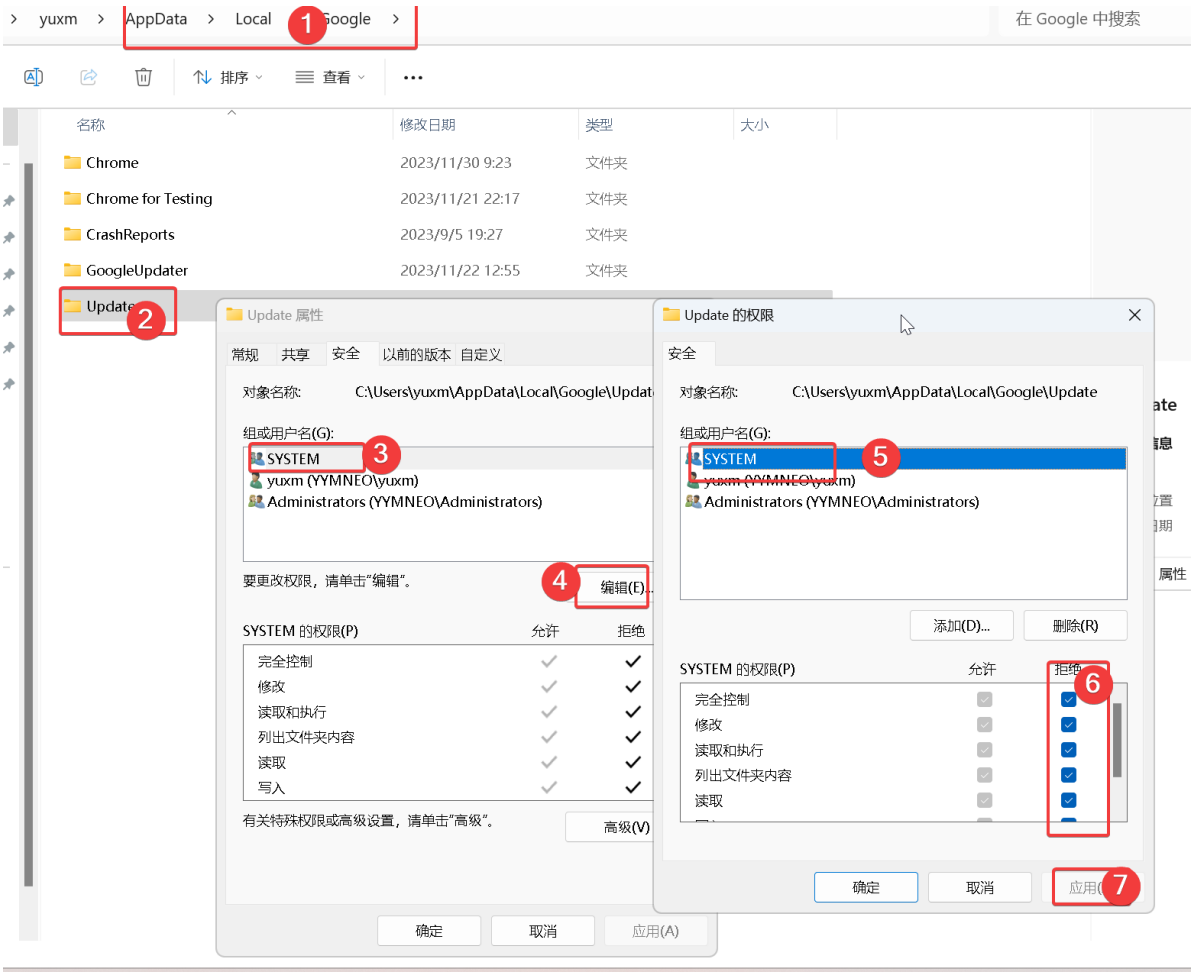
将下载下来的chromedriver_win32_95.zip解压出来的chromedriver.exe复制到python的路径下，如：



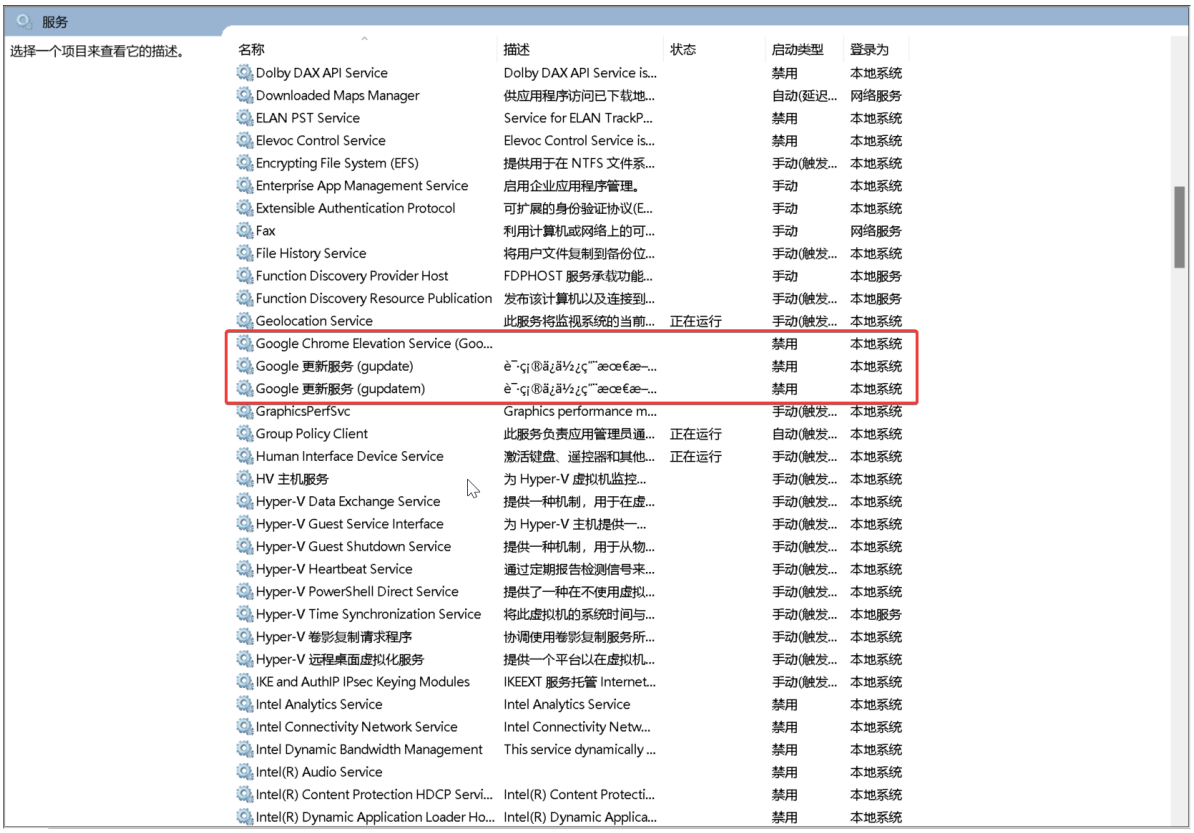
禁止chrome自动更新

右击桌面上的google chrome ->打开文件所在的目录 -> 退到上两级目录（如:C:\Users\yuxm\AppData\Local\Google）->分别右击目录:GoogleUpdater 和目录:Update ->属性->安全->编辑->拒绝->应用,如下图：

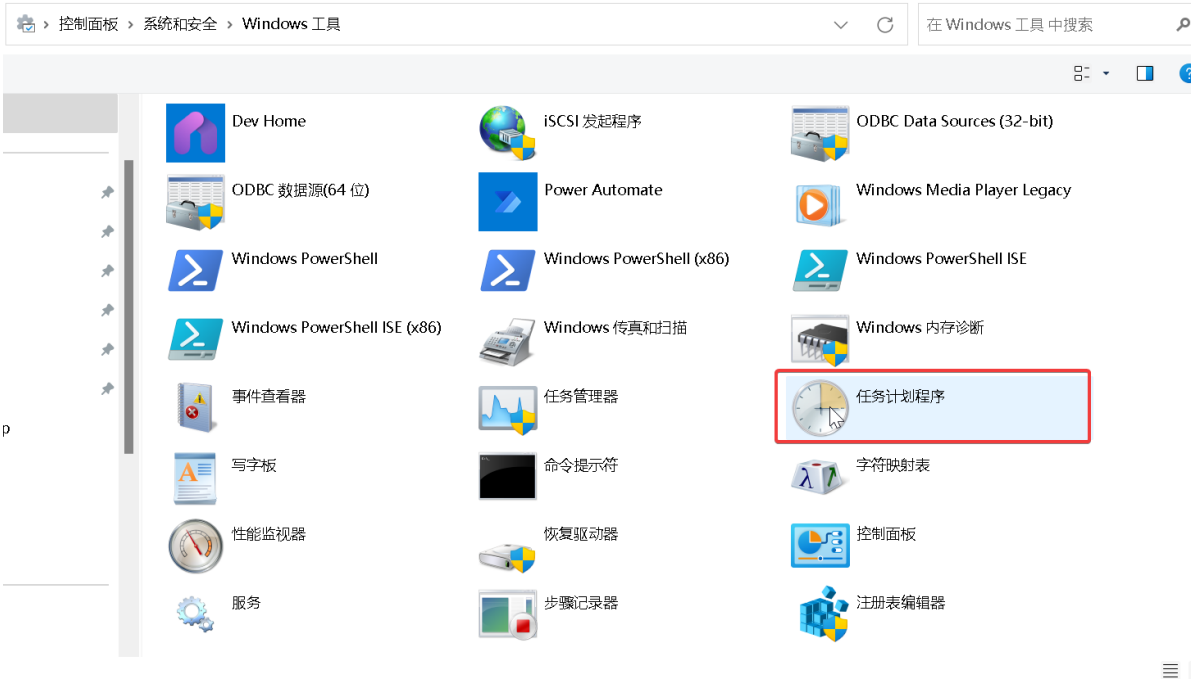

yuxm › AppData › Local › Google ›
在 Google 中
排序
查看
名称
修改日期
类型
大小
Chrome
2023/11/29 20:36
文件夹
Chrome for Testing
2023/11/21 22:17
文件夹
CrashReports
2023/9/5 19:27
文件夹
GoogleUpdater
2023/11/22 12:55
文件夹
1
GoogleUpdater 属性
常规
共享
以前的版本
自定义
2
对象名称:
C:\Users\yuxm\AppData\Local\Google\Gc
组或用户名(G):
SYSTEM
3
yuxm (YYMNEO\yuxm)
Administrators (YYMNEO\Administrators)
4
要更改权限，请单击"编辑"。
编辑
SYSTEM 的权限(P)
允许
完全控制
修改
读取和执行
列出文件夹内容
读取
写入
有关特殊权限或高级设置，请单击"高级"。
确定
GoogleUpdater 的权限
安全
对象名称:
C:\Users\yuxm\AppData\Local\Google\GoogleUpda
组或用户名(G):
SYSTEM
5
yuxm (YYMNEO\yuxm)
Administrators (YYMNEO\Administrators)
添加(D)...
删除(R)
SYSTEM 的权限(P)
允许
拒绝
6
完全控制
修改
读取和执行
列出文件夹内容
读取
应用(A)
ch07_t.md - Typora
ch09_t.md - Typora


yuxm › AppData › Local › Google ›
1
在 Google 中搜索
排序
查看
名称
修改日期
类型
大小
Chrome
2023/11/30 9:23
文件夹
Chrome for Testing
2023/11/21 22:17
文件夹
CrashReports
2023/9/5 19:27
文件夹
GoogleUpdater
2023/11/22 12:55
文件夹
Update
2
Update 属性
常规
共享
安全
以前的版本
自定义
对象名称:
C:\Users\yuxm\AppData\Local\Google\Updat
组或用户名(G):
SYSTEM
3
yuxm (YYMNEO\yuxm)
Administrators (YYMNEO\Administrators)
4
要更改权限，请单击"编辑"。
编辑(E)...
SYSTEM 的权限(P)
允许
拒绝
完全控制
修改
读取和执行
列出文件夹内容
读取
写入
有关特殊权限或高级设置，请单击"高级"。
高级(V)
确定
取消
应用(A)
Update 的权限
安全
对象名称:
C:\Users\yuxm\AppData\Local\Google\Update
组或用户名(G):
SYSTEM
5
yuxm (YYMNEO\yuxm)
Administrators (YYMNEO\Administrators)
添加(D)...
删除(R)
SYSTEM 的权限(P)
允许
拒绝
6
完全控制
修改
读取和执行
列出文件夹内容
读取
确定
取消
7

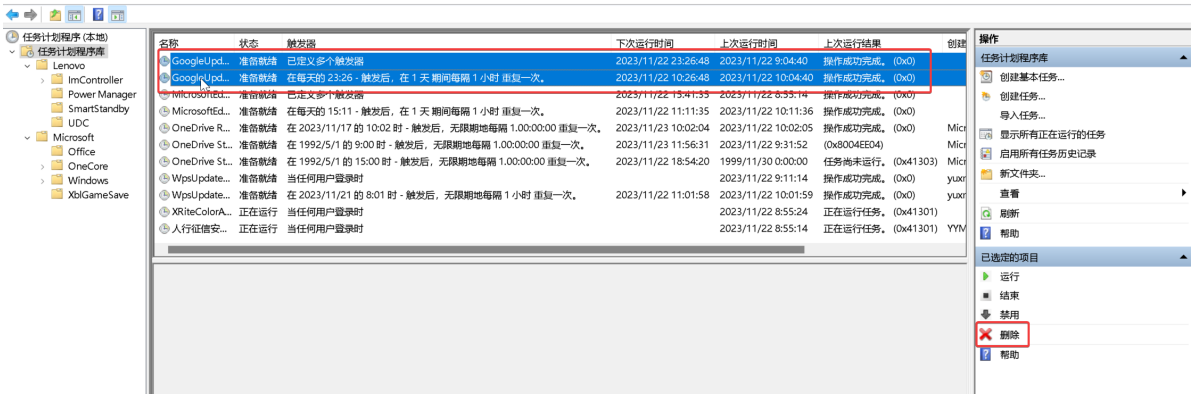
右击开始->打开计算机管理->服务和应用程序->服务，禁用所有和google有关的程序，如：



单击开始->所有应用->window工具系统->任务计划程序



删除所有和google相关的更新程序



采集IDEA环境

使用pycharm的community版本

[下载地址]

使用python 3.9.13 版本

[下载地址]

说明：为了确保使用的是系统解析器，建议先通过控制面版，卸载之前的所有python版本，包括虚拟环境。然后再安装 python3.9.13到d://app/python目录下。

安装依赖包

WIN键+R运行cmd进入命令行：

```
python -m pip install --upgrade pip -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install pymongo -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install pymysql -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install scrapy -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install pandas -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install sqlalchemy -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install bs4 -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install selenium -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install redis -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install bgutils-hddly -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com
pip install --upgrade bgutils-hddly -i http://mirrors.aliyun.com/pypi/simple ^
--trusted-host mirrors.aliyun.com

```

转储环境准备

需要开启hadoop集群，Spark集群

在master主机上运行：

```
start-all.sh
cd /usr/local/spark/sbin/
./start-all.sh
```

需要同步hadoop主机和从机的时间，以及windows时间

```
systemctl stop ntpd
ntpdate cn.pool.ntp.org
service ntpd start & chkconfig ntpd on
```

报告模板

实训报告模板：

Spark综合实训项目实验报告Ver2.0.1 http://bigdata.hddly.cn/b59510spark/file/spark_ver2.0.1.rar

报告编写

1.1. 概述（5分）

1.1.1. 训练要点(1分)

1. 回顾并熟练使用python进行数据采集
2. 掌握scala的使用，将数据从mongo采集到hdfs
3. 熟练掌握使用spark streaming实现对hdfs目录监测并完成数据分析与处理
4. 熟练spark的使用，将分析结果存储到mysql
5. 训练数据据的可视化，将mysql的数据取出并完成可视化

1.1.2. 需求说明(2分)

本实训采集各类题材数据,包括并不限于:商品、音乐、新闻、房产、书籍、招聘;本实训要实现的功能是通过同学采集某类题材数据，实时采集题材数据到mongodb, 再从mongodb将所有同学采集的同题材数据采集到hdfs，然后实现该类数据的实时流分析，对分析结果进行存储到mysql，然后将mysql中数据的可视化。

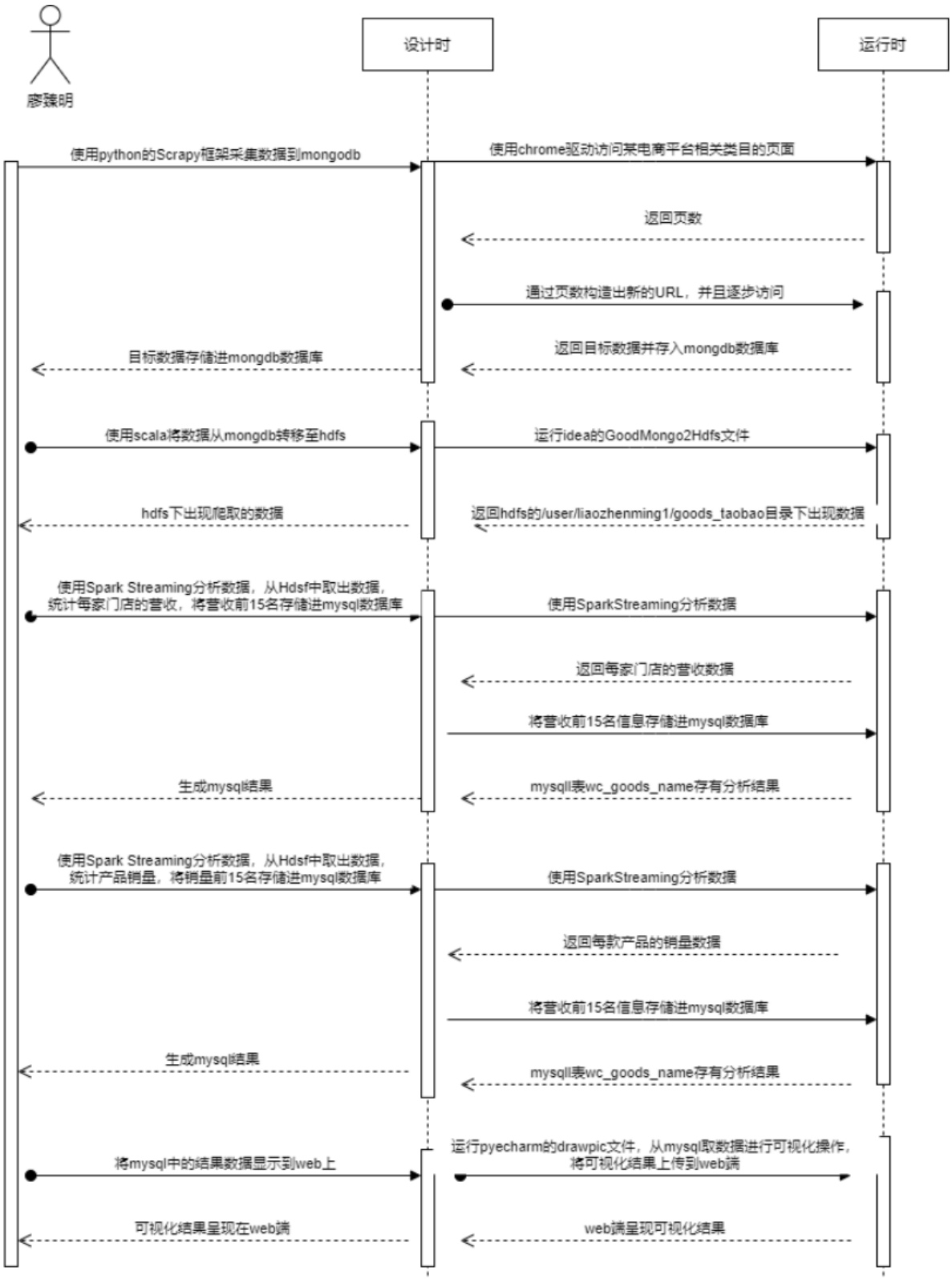
1.1.3. 实现步骤(2分)

1. 数据采集：使用scrapy框架实现某类题材网站的数据采集，存入mongo数据库
2. 数据转存：scala实时采集题材数据到hdfs
3. 数据分析：启动Spark Streaming监控hdfs目录，分析统计数据
4. 数据存储：使用spark将统计结果转存到mysql中
5. 数据可视：使用python将mysql的结果数据每隔几秒显示出来并更新到web上

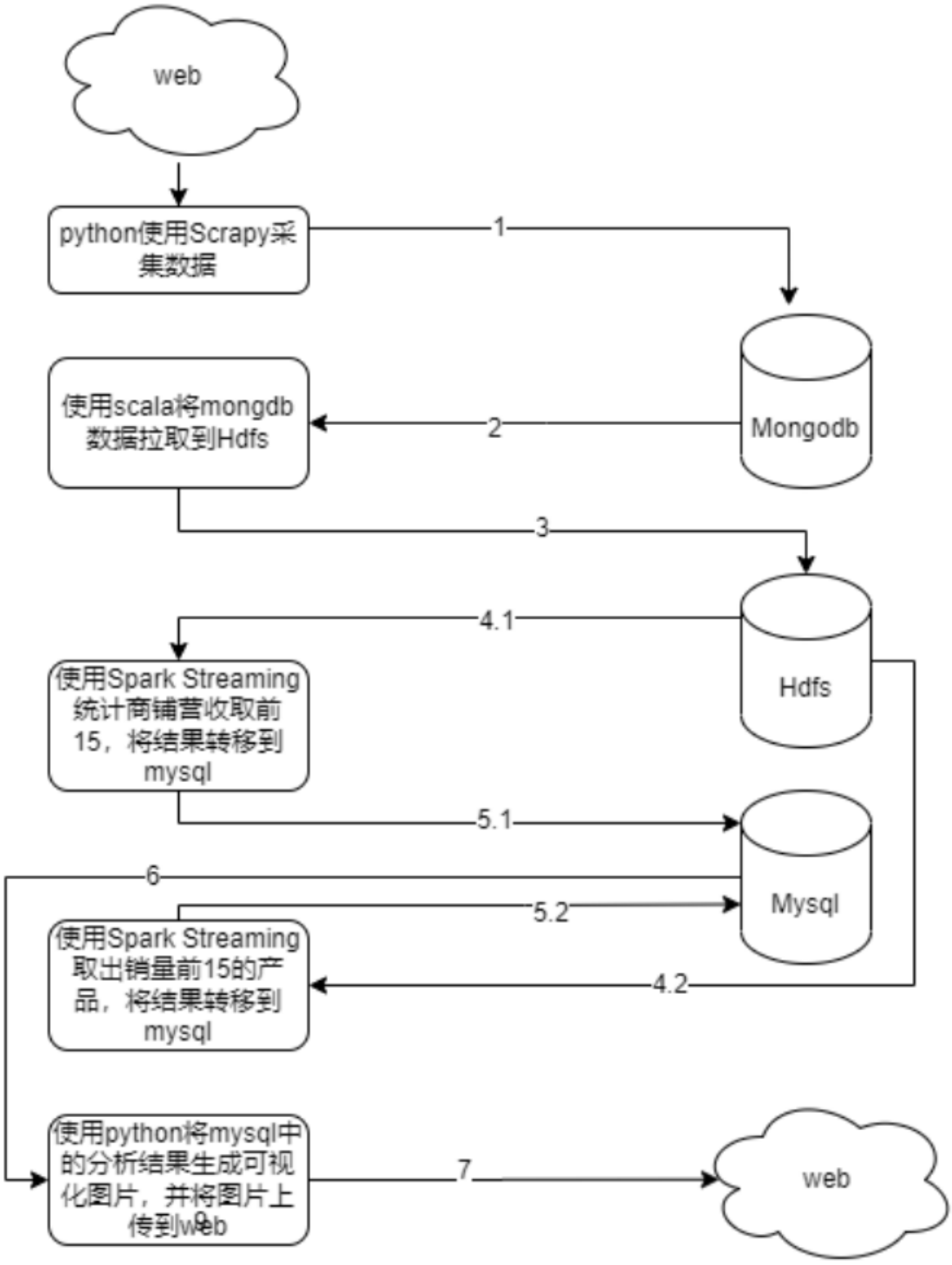
1.2. 总体设计(20分)

1.2.1. 总体流程(10分)

【业务流程图】(5分)

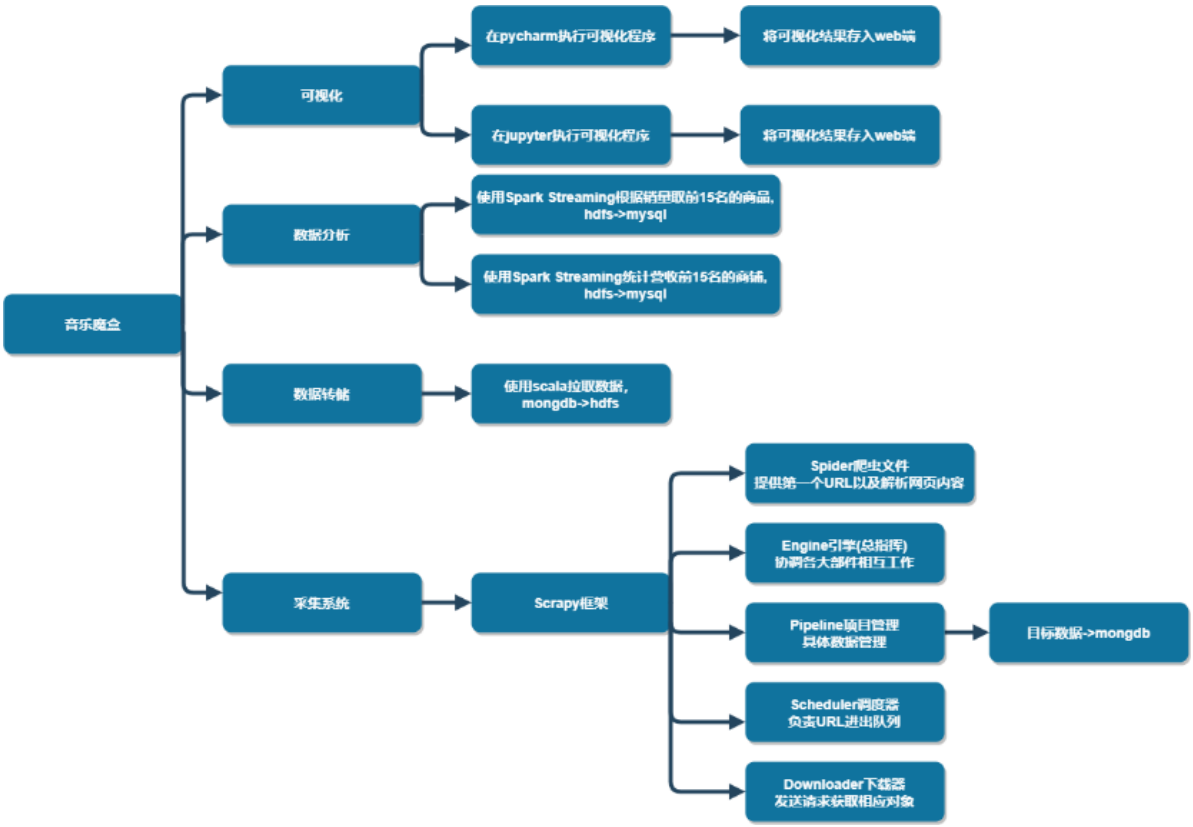


【数据流图】(5分)



1.2.2. 系统功能结构(5分)

【模块组织结构图】(5分)

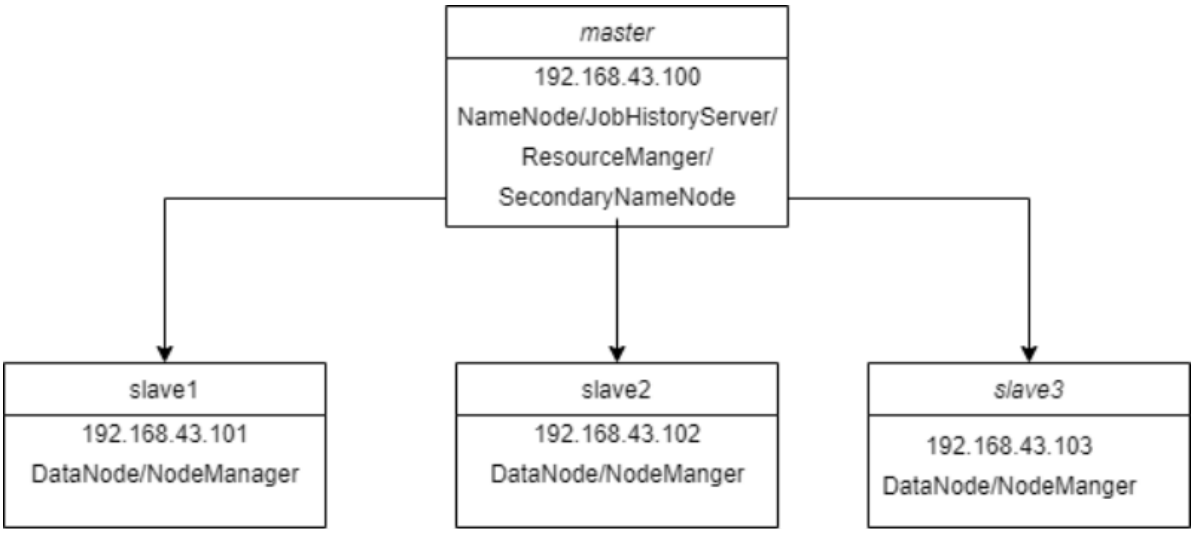


1.2.3. 运行环境(5)

- 操作系统和软件依赖(2分)

子系统	操作系统	依赖软件	备注
数据抓取	Window	PyCharm ,python	
数据存储(mongodb->hdfs)	Linux,Window	Vmware,IDEA,Mongodb,Hadoop, scala,jdk	
数据分析与存储(spark streaming)	Linux,Window	Vmware,IDEA,Mysql,Hadoop,scala,jdk	
数据可视化	Window	PyCharm,Mysql,web, python	

- 网络拓朴图(3分)



1.3. 详细设计(70分)

1.3.1. 库表设计(10分)

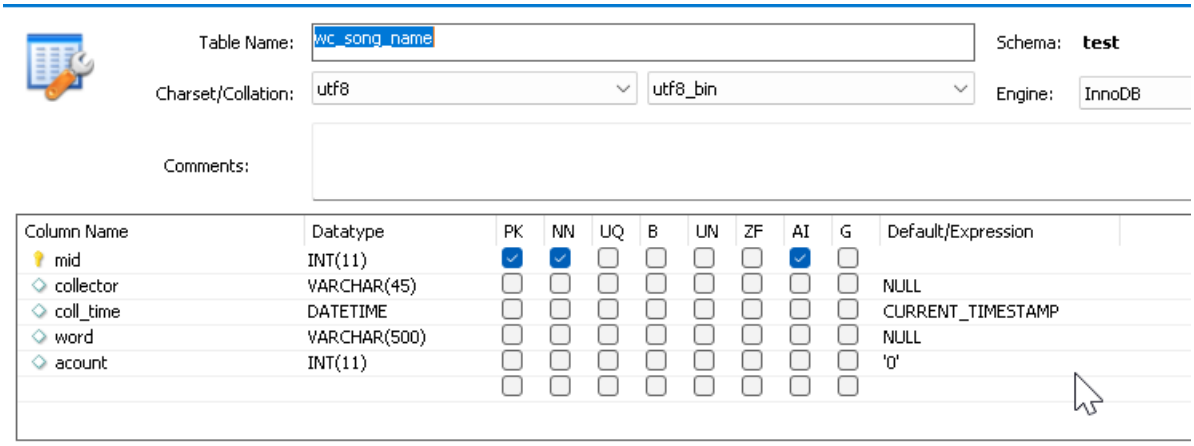
表名	wc_goods_name			
描述	商品销量统计表（门店营收统计表）			
主键	mid			
索引				
字段名	描述	类型	默认值	是否必录
mid	序列号	int(11)	自增长	是
collector	分析人	varchar(45)		否
coll_time	时间	datetime(0)		否
word	商品名/门店名	varchar(500)		否
acount	销量/营收	int(11)		否

建表SQL:

字段 索引 外键 触发器 选项 注释 SQL 预览

```
CREATE TABLE `test`.`Untitled`  (
  `mid` int(11) NOT NULL AUTO_INCREMENT,
  `collector` varchar(45) CHARACTER SET utf8 COLLATE utf8_unicode_ci NULL DEFAULT NULL,
  `coll_time` datetime(0) NULL DEFAULT NULL,
  `word` varchar(500) CHARACTER SET utf8 COLLATE utf8_unicode_ci NULL DEFAULT NULL,
  `acount` int(11) NULL DEFAULT 0,
  PRIMARY KEY (`mid`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 4156 CHARACTER SET = utf8 COLLATE = utf8_unicode_ci ROW_FORMAT =
Dynamic;
```

音乐数据库表设计



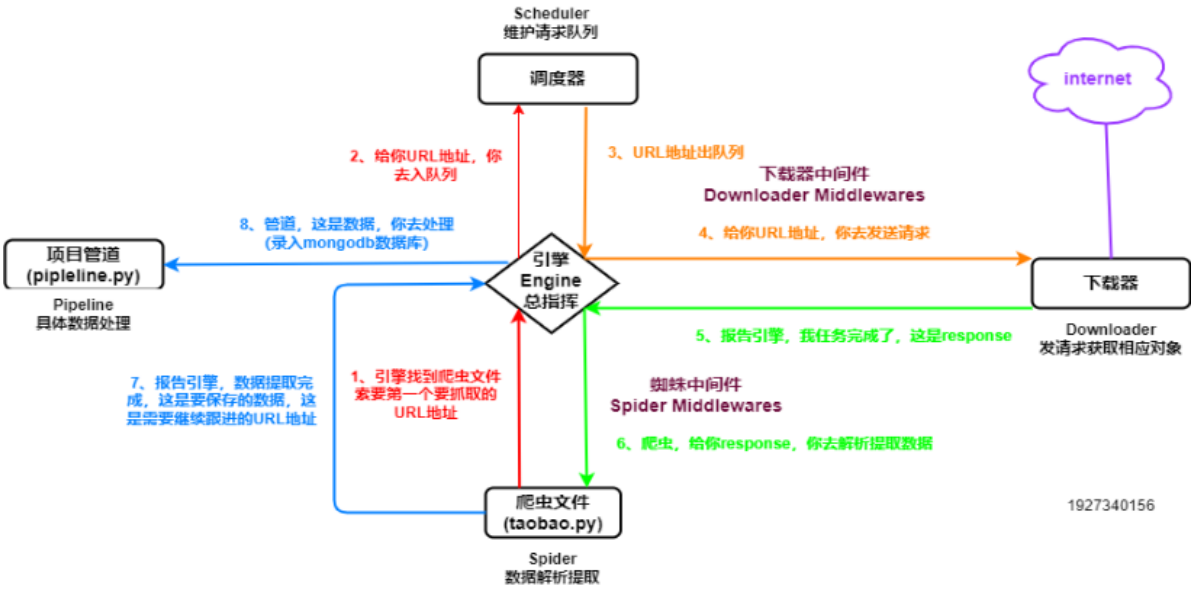
建表SQL:

```
CREATE TABLE `wc_song_name` (
  `mid` int(11) NOT NULL AUTO_INCREMENT,
  `collector` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `coll_time` datetime DEFAULT CURRENT_TIMESTAMP,
  `word` varchar(500) COLLATE utf8_unicode_ci DEFAULT NULL,
  `acount` int(11) DEFAULT '0',
  PRIMARY KEY (`mid`)
) ENGINE=InnoDB AUTO_INCREMENT=286178 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

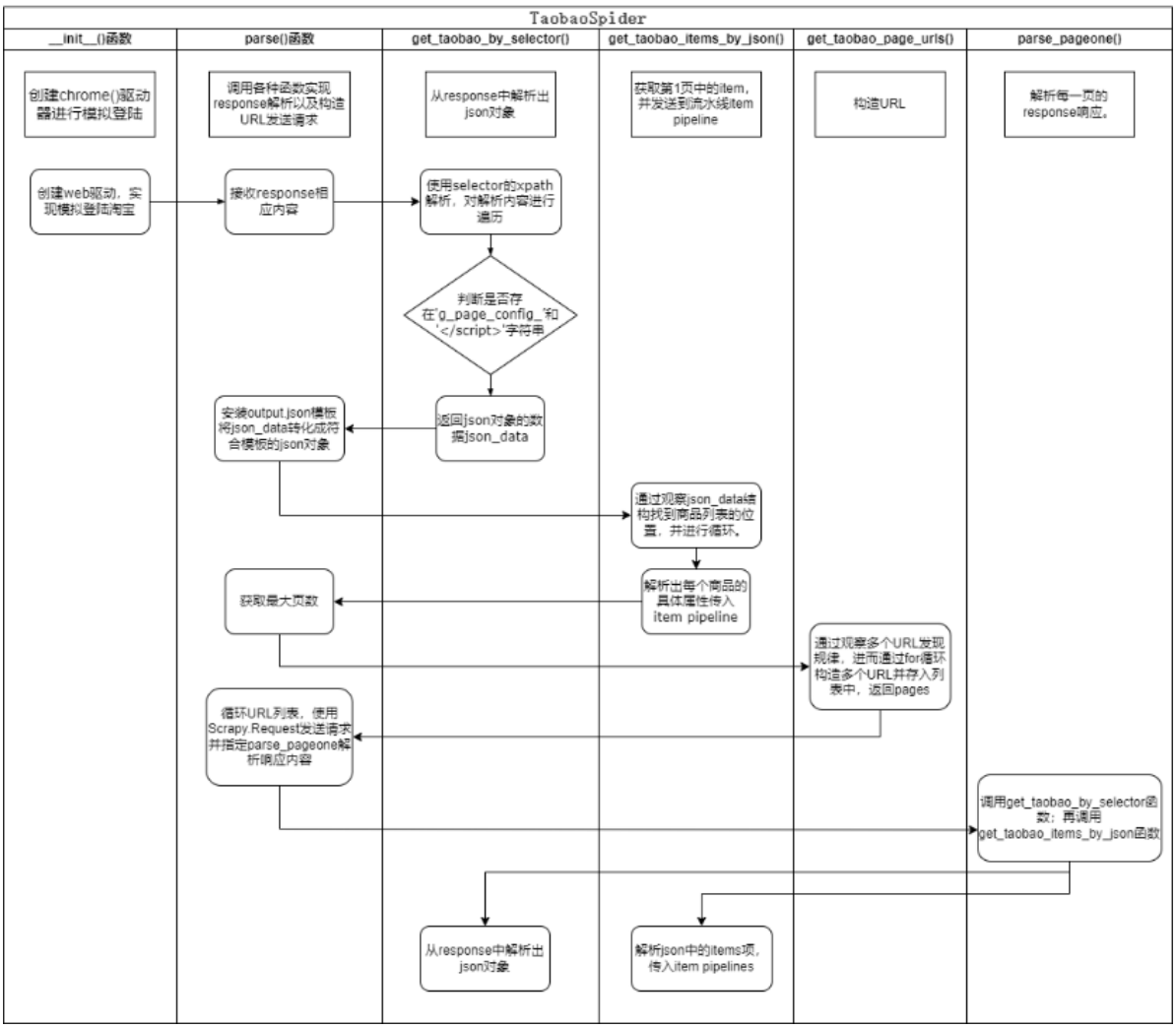
1.3.2. 数据采集(10分)

功能说明：使用python采集数据到MongoDB

功能设计：使用Scrapy框架采集某电商平台的数据，先是访问水果类目的购物页面，获取水果类目的总页数，根据页数构造URL，依次访问URL，获取每款产品的信息，包括：标题、价格、销量、店铺名、发货地，最后将信息录入到mongodb数据库



Spider解析:



源码实现：

```
import datetime
import json
import re
import random
import time
import scrapy
from bs4 import BeautifulSoup
from scrapy import Selector
from selenium import webdriver
from selenium.webdriver.common.by import By
from goods.items import GoodsItem
```

```python
class TaobaoSpider(scrapy.Spider):
    name = 'taobao'
    allowed_domains = ['www.taobao.com']
    # start_urls = ['http://www.taobao.com/']
    start_urls = ['https://s.taobao.com/search?q=水果'] #1.请补充搜索关键字
    # https://s.taobao.com/search?q=笔记本
    # 使用scrapy的Selector解析
    def get_taobao_by_selector(self, response):
        start_str = 'g_page_config = '
        end_str = '</script>'
        split_str = ';\n'
        selector = Selector(text=response.text)
        scripts = selector.xpath("/html/head/script")
        for script in scripts:
            if str(script.extract()).find(start_str)>=0 and str(script.extract()).find(end_str)>=0:
                scripts_data= str(script.extract())
                istart = scripts_data.index(start_str)
                iend = scripts_data.index(end_str)
                data_str = scripts_data[istart + len(start_str):iend]
                list_strs = data_str.split(split_str)
                json_data = json.loads(list_strs[0])
                return json_data
        print('error:scripts!:response.text:',response.text )
        return None
```

```python
    # 解析json中的items项
    def get_taobao_items_by_json(self,json_data):
        goods=json_data['mods']['itemlist']['data']['auctions'] # 2.请补充商品集合的json的节点属性名，可通过观
        items=[]
        for good in goods:
            item = GoodsItem()
            # item['title'] = good['title'] # 3.请补充商品的名称，可通过商品列表中某一商品，查找商品中商品名称的节
            item['raw_title'] = good['raw_title']
            # item['pic_url'] = good['pic_url'] #图片url，这些数据暂时用不上
            # item['detail_url'] = good['detail_url'] #详情url
            item['view_price'] = good['view_price']
            item['view_fee'] = good['view_fee']
            item['item_loc'] = good['item_loc'] # 商家所在地
            # 解析销售数量
            try:
                if good['view_sales'].find("万") != -1:
                    good['view_sales'] = str(int(re.findall(r'(.+)万', good['view_sales'])[0]) * 10000)

                elif good['view_sales']:
                    good['view_sales'] = str(re.findall('\d*', good['view_sales'])[0])
                else:
                    good['view_sales']=0
                item['view_sales'] = good['view_sales']#销售数量
            except Exception as err:
                print('view_sales,错误',err)
            item['comment_count'] = good['comment_count'] #评论数
            item['nick'] = good['nick'] #商铺名
            item['type1'] = '水果'
            items.append(item)
        return items
    # 获取从第2页开始的urls
```

```
        return items
    # 获取从第2页开始的urls
    def get_taobao_page_urls(self,totalPage):
        ...
        pages=[]
        for i in range(2,totalPage):
            s=(i-1)*44
            bcoffset=-3*i+7
            ntoffset=bcoffset
            print('s,bcoffset:',s,bcoffset)
            url='https://s.taobao.com/search?q=水果&bcoffset=%d&ntoffset=%d&p4ppushleft=2,48&s=%d'%(bco
            print('url:' ,url)
            pages.append(url)
        return pages

    def __init__(self):
        self.browser = webdriver.Chrome()
        # 以下方法会对滑块有效
        self.browser.execute_cdp_cmd("Page.addScriptToEvaluateOnNewDocument", {
            "source": """
              Object.defineProperty(navigator, 'webdriver', {
                get: () => undefined
              })
            """
        })#发送请求的时候"webdriver"可能会告诉服务器这是模拟登录,'webdriver'替换为undefined
        self.browser.get(self.start_urls[0]);
        self.browser.find_element(By.NAME, "fm-login-id").send_keys("tb8375686**"); # 5.请补充,指登录淘宝
        self.browser.find_element(By.NAME, "fm-login-password").send_keys("740740l**"); # 6.请补充，指登
        time.sleep(random.randint(5, 8))
        self.browser.implicitly_wait(10)
        self.browser.find_element(By.CSS_SELECTOR, "#login-form > div.fm-btn > button").click();
        self.browser.set_page_load_timeout(30)
```

```
    def parse(self, response):
        #从response中解析出json数据对象
        json_rlt=self.get_taobao_by_selector(response)
        #存在json文件供分析内容

        # 解析首页信息
        #  json.dumps()是把python对象转换成json对象的一个过程,生成的是字符串。
        # 安装output.json模板将json_rlt转化成
        with open('output.json', 'w', encoding='utf-8') as fp:
            json.dump(json_rlt, fp, ensure_ascii=False, indent=4)  # indent=2,0,None
            fp.close()
        # 获取第1页中的items，并发送到流水线item pipelines
        items=self.get_taobao_items_by_json(json_rlt)
        for item in items:
            yield item
        #  获取当前显示的最大页数
        itotalPage=json_rlt['mods']['pager']['data']['totalPage']# 7.请补充,淘宝的页数在json中，通过观察json_rlt中totalP
        #   "pageSize": 44,totalPage": 100,"currentPage": 1,"totalCount": 30710
        print('totalPage:',itotalPage)
        itotalPage=100
        icurrPage=0
        page_urls=self.get_taobao_page_urls(itotalPage)#构造url
        for page_url in page_urls:
            icurrPage+=1
            yield scrapy.Request(url=page_url, meta={'itotalPage': itotalPage,'icurrPage':icurrPage},
                                 callback=self.parse_pageone,
                                 dont_filter=True)
```

```
    def parse_pageone(self, response):
        print('itotalPage:',response.meta['itotalPage'],'icurrPage:',response.meta['icurrPage'])
        json_rlt= self.get_taobao_by_selector(response)
        if json_rlt is not None:
            print('response :g_page:json:', json_rlt)
            items=self.get_taobao_items_by_json(json_rlt)## 解析json中的items项
            for item in items:
                yield item
        else:
            pass
```

```
    def close(self, spider, reason):
        print("spider finish. to close browser..")
        spider.browser.close() # 关闭模拟器页面
        spider.browser.quit() # 关闭模拟器，释放资源
        print("spider finish. browser closed.")
```

运行截图：

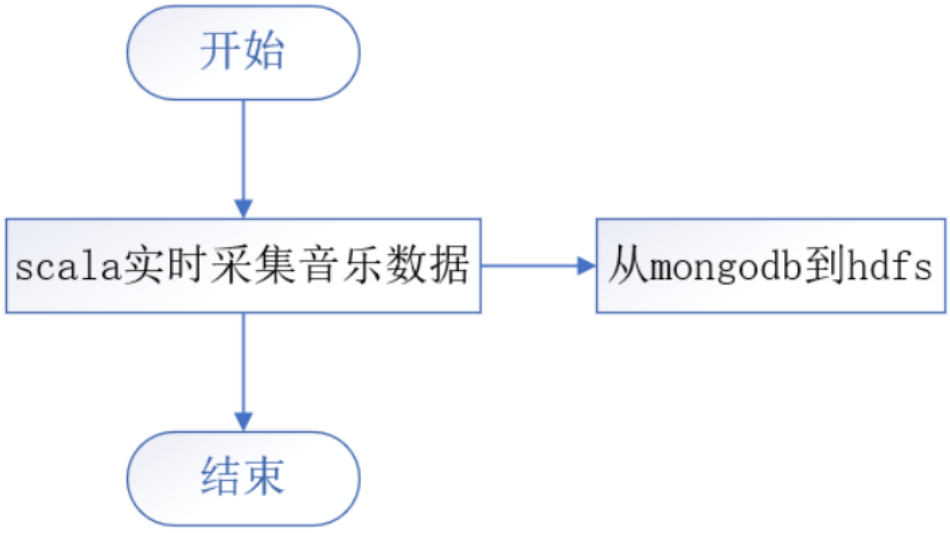
```
终端:  局部  局部 (2)  +
pushleft=2,48&s=1804> (referer: https://s.taobao.com/search?q=%E6%B0%B4%E6%9E%9C)
2022-11-27 11:13:48 [scrapy.core.scraper] DEBUG: Scraped from <200 https://s.taobao.com/search?q=%E6%B0%B4%E6%
pushleft=2%2C48&s=1980>
{'coll_time': datetime.datetime(2022, 11, 27, 11, 13, 48, 803734),
 'collector': '廖臻明',
 'comment_count': '1455',
 'item_loc': '上海',
 'nick': '水果兄弟旗舰店',
 'raw_title': '泰国金枕头冷冻榴莲果肉树熟新鲜速冻3kg无核冰冻榴莲肉商用水果',
 'type1': '水果',
 'view_fee': '0.00',
 'view_price': '118.00',
 'view_sales': '100'}
2022-11-27 11:13:48 [scrapy.core.scraper] DEBUG: Scraped from <200 https://s.taobao.com/search?q=%E6%B0%B4%E6%
pushleft=2%2C48&s=1936>
{'coll_time': datetime.datetime(2022, 11, 27, 11, 13, 48, 818645),
 'collector': '廖臻明',
 'comment_count': '11488',
 'item_loc': '浙江 衢州',
 'nick': '不变的心1980',
 'raw_title': '二妞正宗江山徐香弥猴桃带箱5斤水果新鲜当季整箱大绿心奇异果',
 'type1': '水果',
 'view_fee': '0.00',
 'view_price': '21.80',
 'view_sales': '1000'}
2022-11-27 11:13:48 [scrapy.core.scraper] DEBUG: Scraped from <200 https://s.taobao.com/search?q=%E6%B0%B4%E6%
pushleft=2%2C48&s=1936>
{'coll_time': datetime.datetime(2022, 11, 27, 11, 13, 48, 826623),
 'collector': '廖臻明',
```

1.3.3. 数据存储(mongodb->hdfs)(10分)

功能说明：使用scala将mongdb数据拉取到hdfs

功能设计：连接mongdb数据库，从pythondb数据库的表goods_taobao的表中筛选出满足collector为“廖臻明”条件的数据，然后遍历数据，将数据导入Hdfs文件系统中

<补充流程图>



源码实现：

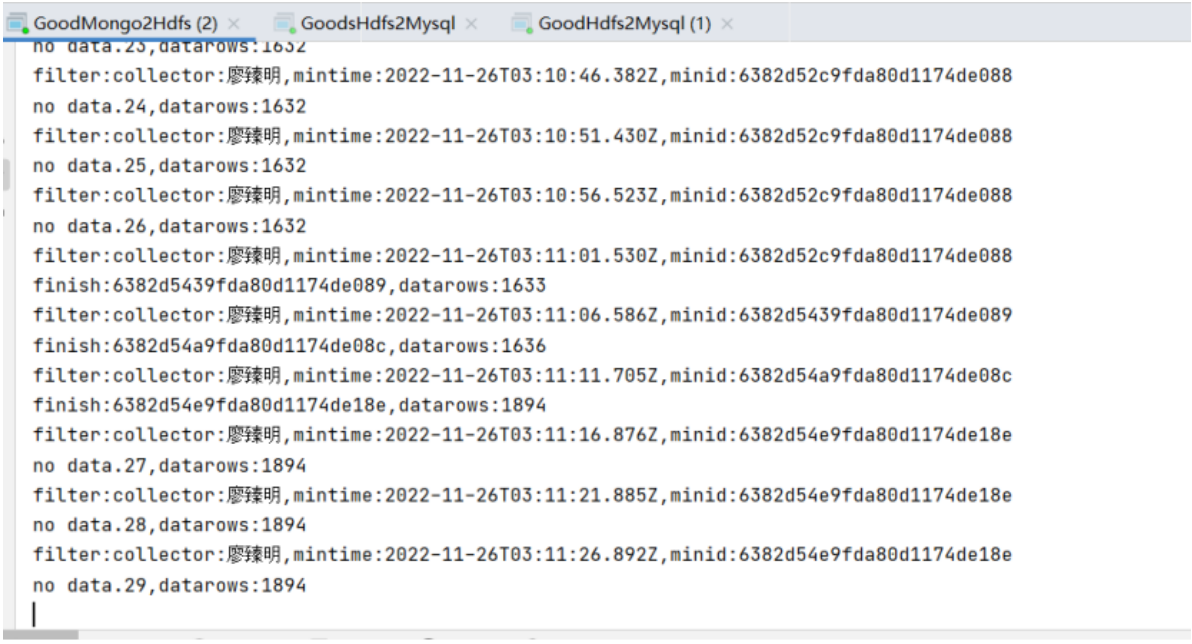
```
package traind
import com.mongodb.client.model.Filters
import com.mongodb.client.{FindIterable, MongoCollection, MongoCursor, MongoDatabase}
import com.mongodb.{MongoClient, ServerAddress}
import org.apache.hadoop.conf.Configuration
import org.apache.hadoop.fs.{FileSystem, Path}
import org.bson.Document
import org.bson.conversions.Bson
import org.bson.types.ObjectId
import java.io.{BufferedWriter, OutputStreamWriter}
import java.time.Instant
import java.util.ArrayList
```

```
//mongo-java-driver.jar支持
object GoodMongo2Hdfs {
  def main(args: Array[String]): Unit = {
    var minid_goods_taobao = new ObjectId( hexString = "636655350695bb68b9649a73"); //起始ID，最小ID
    var inodata = 0;
    var idatarows = 0
    val targetpath = "hdfs://master:9864/user/liaozhenming1/goods_taobao/"
    val hdfsconf: Configuration = new Configuration();
    hdfsconf.set("fs.defaultFS", "hdfs://master:9864");
    System.setProperty("HADOOP_USER_NAME", "root");
    val hdfsfs: FileSystem = FileSystem.get(hdfsconf);

    //初始化和连接
    try {
      //两个变量分别为服务器地址和端口
      val serverAddress: ServerAddress = new ServerAddress( host = "home.hddly.cn",  port = 57017)
      val collname = "goods_taobao"
      val collector = "廖臻明"
      val adds = new ArrayList[ServerAddress]()
      adds.add(serverAddress)
      //用户名，连接的数据库名，用户密码
      //      val credential = MongoCredential.createCredential("user", "test", "passwd".toCharArray
      //      val credentials = new ArrayList[MongoCredential]()
      //      credentials.add(credential)
      //通过验证连接到MongoDB客户端
      //      val mongoClient: MongoClient = new MongoClient(adds, credentials)
      val mongoClient: MongoClient = new MongoClient(adds)
      //连接数据库
      val mongoDatabase: MongoDatabase = mongoClient.getDatabase( databaseName = "pythondb") //pythond
      println("Connect MongoDB Successfully!!!")
      val collection: MongoCollection[Document] = mongoDatabase.getCollection(collname)
```

```
      val collection: MongoCollection[Document] = mongoDatabase.getCollection(collname)
      //检索所有文档
      //获取迭代器

      while (true) {
        //        println("已经选中集合:goods_taobao,rowcount:" + collection.count().toString)//可有可无

        //      val from = Instant.parse("2022-09-29T00:00:00.000Z")
        val from = Instant.now().minusSeconds( secondsToSubtract = 60 * 60 * 24 * 3) //只查近3天内数据
        //val filter: Nothing = Filters.and(Filters.gte("age", 26), Filters.lte("e", 80))
        //          Filters.eq("collector", collector), 采集数据时，不过滤采集者
        val filter: Bson = Filters.and(
          Filters.gt( fieldName = "coll_time", from),
          Filters.gt( fieldName = "_id", minid_goods_taobao))
        println("filter:collector:" + collector + ",mintime:" + from + ",minid:" + minid_goods_taobao)

        val findInterable: FindIterable[Document] = collection.find(filter)
        //        val sb: mutable.StringBuilder = new mutable.StringBuilder
        //获取游标
        val mongoCursor: MongoCursor[Document] = findInterable.iterator()
        if (mongoCursor.hasNext) {
          //开始遍历
          val newPath = new Path(targetpath + minid_goods_taobao.toString)
          hdfsfs.delete(newPath, true)
          val os = hdfsfs.create(newPath)
          val bw = new BufferedWriter(new OutputStreamWriter(os,  charsetName = "utf-8"))
          while (mongoCursor.hasNext) {
            val document = mongoCursor.next()
            bw.write(document.toJson())
            bw.write( str = "\n")
            if (minid_goods_taobao.toString.compareTo(document.get("_id").toString) < 0) {
              minid_goods_taobao = new ObjectId(document.get("_id").toString)
```

```
                minid_goods_taobao = new ObjectId(document.get("_id").toString)
              }
              idatarows += 1
            }
            bw.close
            os.close
            println("finish:" + minid_goods_taobao.toString + ",datarows:" + idatarows)
          }
          else {
            inodata += 1
            println("no data." + inodata + ",datarows:" + idatarows)
          }
          Thread sleep 5000
        }
      } catch {
        case e: Exception =>
          println(e.getClass.getName + ": " + e.getMessage)
      }
      hdfsfs.close
    }
}
```

运行截图：



在hadoop集群上运行截图：

```
[root@master ~]#
[root@master ~]# spark-submit --master yarn --class traind.GoodMongo2Hdfs /home/hadoop/train.jar
/usr/local/spark/conf/spark-env.sh: line 5: rt: command not found
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/spark/jars/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-3.3.1/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2022-11-27 08:02:33,165 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2022-11-27 08:02:34,530 INFO driver.cluster: Cluster created with settings {hosts=[home.hddly.cn:57017], mode=MULTIPLE, requiredClusterType=UNKNOWN, serv
erSelectionTimeout='30000 ms', maxWaitQueueSize=500}
2022-11-27 08:02:34,530 INFO driver.cluster: Adding discovered server home.hddly.cn:57017 to client view of cluster
Connect MongoDB Successfully!!!
filter:collector:蒙臻明,mintime:2022-11-26T13:02:34.614Z,minid:636655350695bb68b9649a73
2022-11-27 08:02:34,663 INFO driver.cluster: Cluster description not yet available. Waiting for 30000 ms before timing out
2022-11-27 08:02:34,674 INFO driver.connection: Opened connection [connectionId{localValue:1, serverValue:177}] to home.hddly.cn:57017
2022-11-27 08:02:34,694 INFO driver.cluster: Monitor thread successfully connected to server with description ServerDescription{address=home.hddly.cn:570
17, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[4, 2, 1]}, minWireVersion=0, maxWireVersion=8, maxDocumentSize=16777216
, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=17041673}
2022-11-27 08:02:34,695 INFO driver.cluster: Discovered cluster type of STANDALONE
2022-11-27 08:02:34,741 INFO driver.connection: Opened connection [connectionId{localValue:2, serverValue:178}] to home.hddly.cn:57017
finish:638332c3a947d84d96f78aae,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:02:44.550Z,minid:638332c3a947d84d96f78aae
no data.1,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:02:49.573Z,minid:638332c3a947d84d96f78aae
no data.2,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:02:54.585Z,minid:638332c3a947d84d96f78aae
no data.3,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:02:59.598Z,minid:638332c3a947d84d96f78aae
no data.4,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:04.609Z,minid:638332c3a947d84d96f78aae
no data.5,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:09.619Z,minid:638332c3a947d84d96f78aae
no data.6,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:14.626Z,minid:638332c3a947d84d96f78aae
no data.7,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:19.637Z,minid:638332c3a947d84d96f78aae
no data.8,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:24.651Z,minid:638332c3a947d84d96f78aae
no data.9,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:29.664Z,minid:638332c3a947d84d96f78aae
no data.10,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:34.676Z,minid:638332c3a947d84d96f78aae
no data.11,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:39.685Z,minid:638332c3a947d84d96f78aae
no data.12,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:44.695Z,minid:638332c3a947d84d96f78aae
no data.13,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:49.704Z,minid:638332c3a947d84d96f78aae
no data.14,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:54.711Z,minid:638332c3a947d84d96f78aae
no data.15,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:03:59.720Z,minid:638332c3a947d84d96f78aae
no data.16,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:04:04.730Z,minid:638332c3a947d84d96f78aae
no data.17,datarows:8720
filter:collector:蒙臻明,mintime:2022-11-26T13:04:09.739Z,minid:638332c3a947d84d96f78aae
```

1.3.4. 数据分析与存储(spark streaming)(30分)

功能说明：使用spark streaming分析数据，根据销量取前15名的商品，统计营收前15名的商铺。

功能设计：

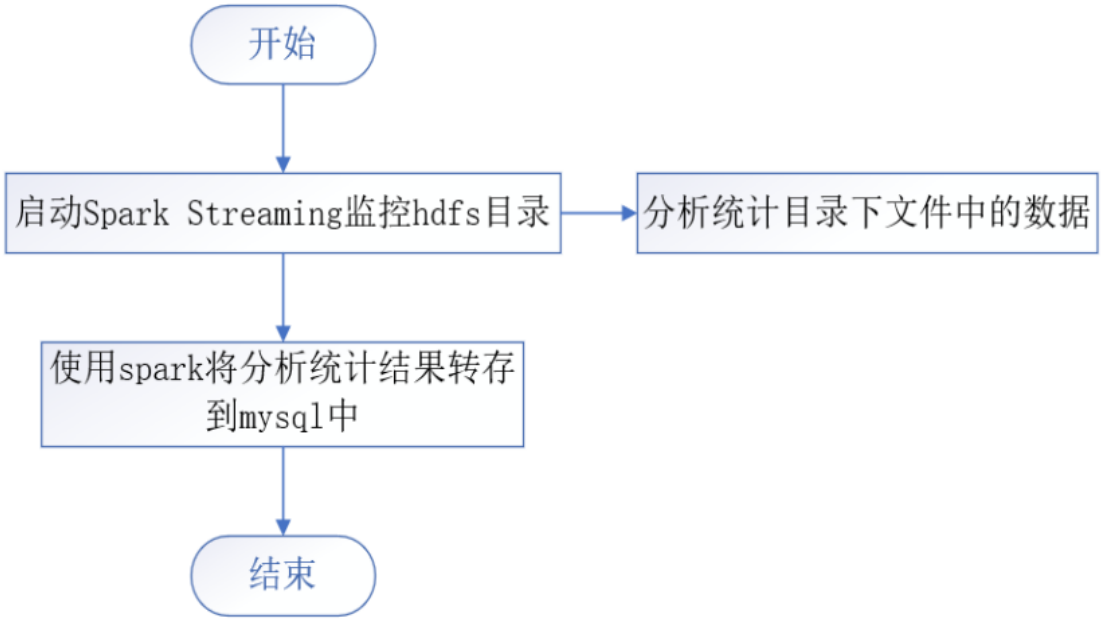
1、 根据销量，取前15名的商品：

第一步，连接Hdfs文件系统，遍历每条数据，取出商品标题、销量。第二步，根据店铺对数据进行分组求和。第三步，根据商品的销量进行排序，取前15名。第四步，连接mysql数据库，将分析出的15条结果插入mysql数据库。

2、 统计营收前15名的商铺：

第一步，连接Hdfs文件系统，遍历每条数据，取出店铺字段，计算店铺内的某款产品的收入。第二步，根据店铺对数据进行分组求和，求得每家店铺的营收。第三步，根据店铺的营收进行排序，取前15名。第四步，连接mysql数据库，将分析出的15条结果插入mysql数据库。

<补充流程图>



源码实现：

1、根据销量取前15名的商品：

```
package traind
import ...
/**
  * 统计产品销量前15名
  */
object GoodsHdfs2Mysql {

  def main(args: Array[String]): Unit = {
    //hdfs dfs -rm -r /user/liaozhenming1/goods_taobao1/
    val streamingpath = "hdfs://master:9864/user/liaozhenming1/goods_taobao1/"
    System.setProperty("HADOOP_USER_NAME", "root");
    val sparkconf = new SparkConf().setMaster("local[*]").setAppName("goodscount") //local[*],spark://master:7077
    val ssc = new StreamingContext(sparkconf, Seconds(10))
    println("streamingpath:" + streamingpath)
    val lines = ssc.textFileStream(streamingpath)
    lines.print()
    val html = lines.map { line => val words = line.split( regex = ","); words(0)}
    println(html)
    val ItemPairs = lines.map(line => {
      val document: Document = Document.parse(line)
      println("line:" + line);
      var strGoodsName = document.get("raw_title").toString()//"nick":是店铺名
      var floatSales = document.get("view_sales").toString().toDouble
      strGoodsName = StrUtil.replaceSymbo(strGoodsName)
      println("goods:" + strGoodsName);
      (strGoodsName, floatSales)
    })
```

```
    ItemPairs.print()
    //计算每家门店的营收
    val htmlCount = ItemPairs.reduceByKeyAndWindow((v1: Double, v2: Double) => v1 + v2, Seconds(60 * 60 * 24), Seconds(20))
    //    窗口长度为1天，滑动时间为20秒,近一天内采集排名
    htmlCount.print()
    //根据门店营收进行排名，取前10名
    val hottestHtml = htmlCount.transform(itemRDD => {
      val top10 = itemRDD.map(pair => (pair._2, pair._1)).sortByKey( ascending = false).map(pair => (pair._2, pair._1)).take( num = 15)
      ssc.sparkContext.makeRDD(top10).repartition( numPartitions = 1)
      //      ssc.sparkContext.makeRDD(top10)
    })
    hottestHtml.print()
    val url = "jdbc:mysql://home.hddly.cn:53306/test?useSSL=false&useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&failOve
    val user = "test"
    val password = "test"
    val mycode = "1927340156" //请将1927100100改为自己的学号
```

```
    hottestHtml.foreachRDD(rdd => {
      rdd.foreachPartition(partitionOfRecords => {
        val conn = ConnectionPool.getConn(url, user, password)
        conn.setAutoCommit(false)
        val stmt = conn.createStatement()
        var i = 1
        //将数据插入数据库
        partitionOfRecords.foreach(record => {
          val strsql = "insert into wc_goods_name(mid,word,acount,collector) values (0,'" +
            record._1 + "'," + record._2 + ",'" + mycode + "')"
          println(strsql)
          stmt.addBatch(strsql)
          i += 1
        })
        stmt.executeBatch()
        if (i > 1) {
          ConnectionPool.delete(conn,  table = "wc_goods_name", mycode)
        }
        conn.commit()
        //        conn.close() //这里不能关闭，返回池中使用
      })
    })

    ssc.start()
    ssc.awaitTermination()
    ssc.stop()
  }
```

2、统计营收前15名的商铺：

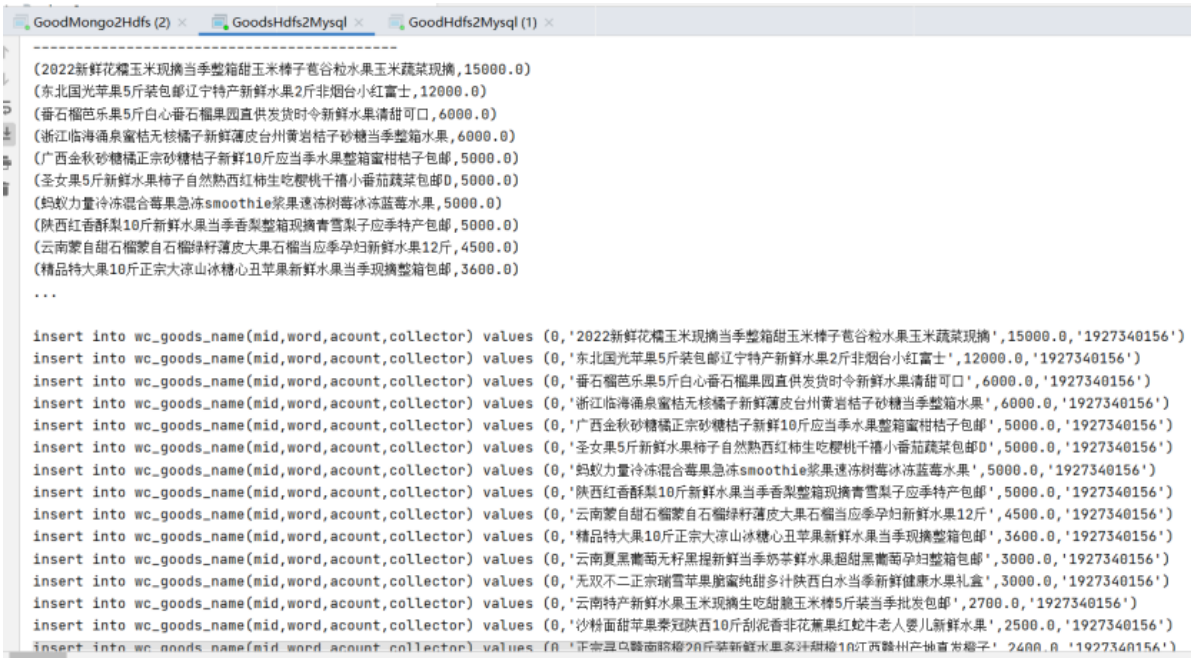
```scala
package traind
import ...
//统计门店营收前15名
object GoodHdfs2Mysql {

  def main(args: Array[String]): Unit = {
    //请将下方链接中myname替换为本人姓名全拼
    //hdfs dfs -rm -r /user/liaozhenming1/goods_taobao1/
    val streamingpath = "hdfs://master:9864/user/liaozhenming1/goods_taobao1/"
    System.setProperty("HADOOP_USER_NAME", "root");
    val sparkconf = new SparkConf().setMaster("local[*]").setAppName("goodscount") //local[*],spark://master:7077
    val ssc = new StreamingContext(sparkconf, Seconds(10))
    println("streamingpath:" + streamingpath)
    val lines = ssc.textFileStream(streamingpath)
    lines.print()
    val html = lines.map { line => val words = line.split( regex = ","); words(0)}
    println(html)
    val ItemPairs = lines.map(line => {
      val document: Document = Document.parse(line)
      println("line:" + line);
      var strGoodName = document.get("nick").toString()//"nick":是店铺名
      strGoodName = StrUtil.replaceSymbo(strGoodName)
      var intSales:Integer =Integer.valueOf(document.get("view_sales").toString())//该店铺某款产品的销售量
      var floatPrice = document.get("view_price").toString().toDouble//该店铺某款产品的价格
      val add4=(x:Int,y:Double)=>x * y
      var totalSale = add4(intSales,floatPrice)//该店某款产品的营收
      println("goods:" + strGoodName);
      (strGoodName, totalSale)
    })
```

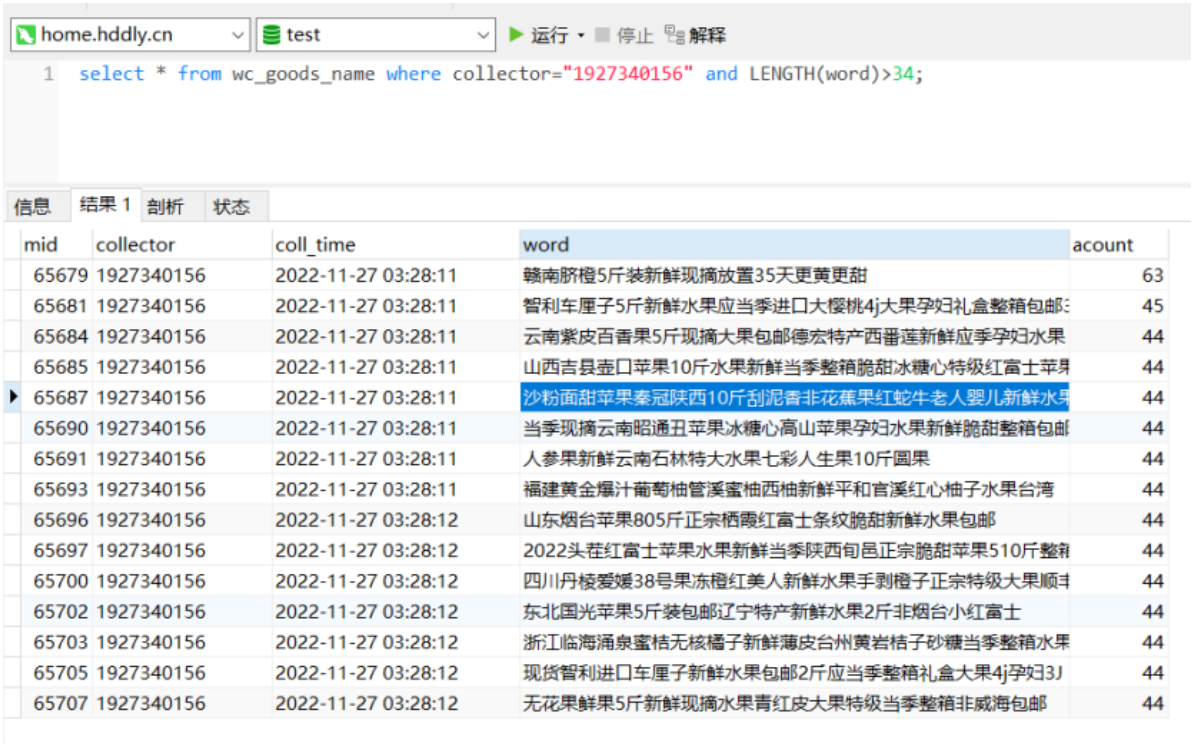
```scala
    ItemPairs.print()
    //计算每家门店的营收
    val htmlCount = ItemPairs.reduceByKeyAndWindow((v1: Double, v2: Double) => v1 + v2, Seconds(60 * 60 * 24), Seconds(20))
    //    窗口长度为1天，滑动时间为20秒,近一天内采集排名
    htmlCount.print()
    //根据门店营收进行排名，取前10名
    val hottestHtml = htmlCount.transform(itemRDD => {
      val top10 = itemRDD.map(pair => (pair._2, pair._1)).sortByKey( ascending = false).map(pair => (pair._2, pair._1)).take( num = 15)
      ssc.sparkContext.makeRDD(top10).repartition( numPartitions = 1)
      //      ssc.sparkContext.makeRDD(top10)
    })
    hottestHtml.print()
    val url = "jdbc:mysql://home.hddly.cn:53306/test?useSSL=false&useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&failOver
    val user = "test"
    val password = "test"
    val mycode = "1927340156" //请将1927100100改为自己的学号
```

```scala
    hottestHtml.foreachRDD(rdd => {
      rdd.foreachPartition(partitionOfRecords => {
        val conn = ConnectionPool.getConn(url, user, password)
        conn.setAutoCommit(false)
        val stmt = conn.createStatement()
        var i = 1
        //将数据插入数据库
        partitionOfRecords.foreach(record => {
          val strsql = "insert into wc_goods_name(mid,word,acount,collector) values (0,'" +
            record._1 + "'," + record._2 + ",'" + mycode + "')"
          println(strsql)
          stmt.addBatch(strsql)
          i += 1
        })
        stmt.executeBatch()
//        if (i > 1) {
//          ConnectionPool.delete(conn, "wc_goods_name", mycode)
//        }
        conn.commit()
        //        conn.close() //这里不能关闭，返回池中使用
      })
    })

    ssc.start()
    ssc.awaitTermination()
    ssc.stop()
```

运行截图：

1、根据销量，取前15名的商品：

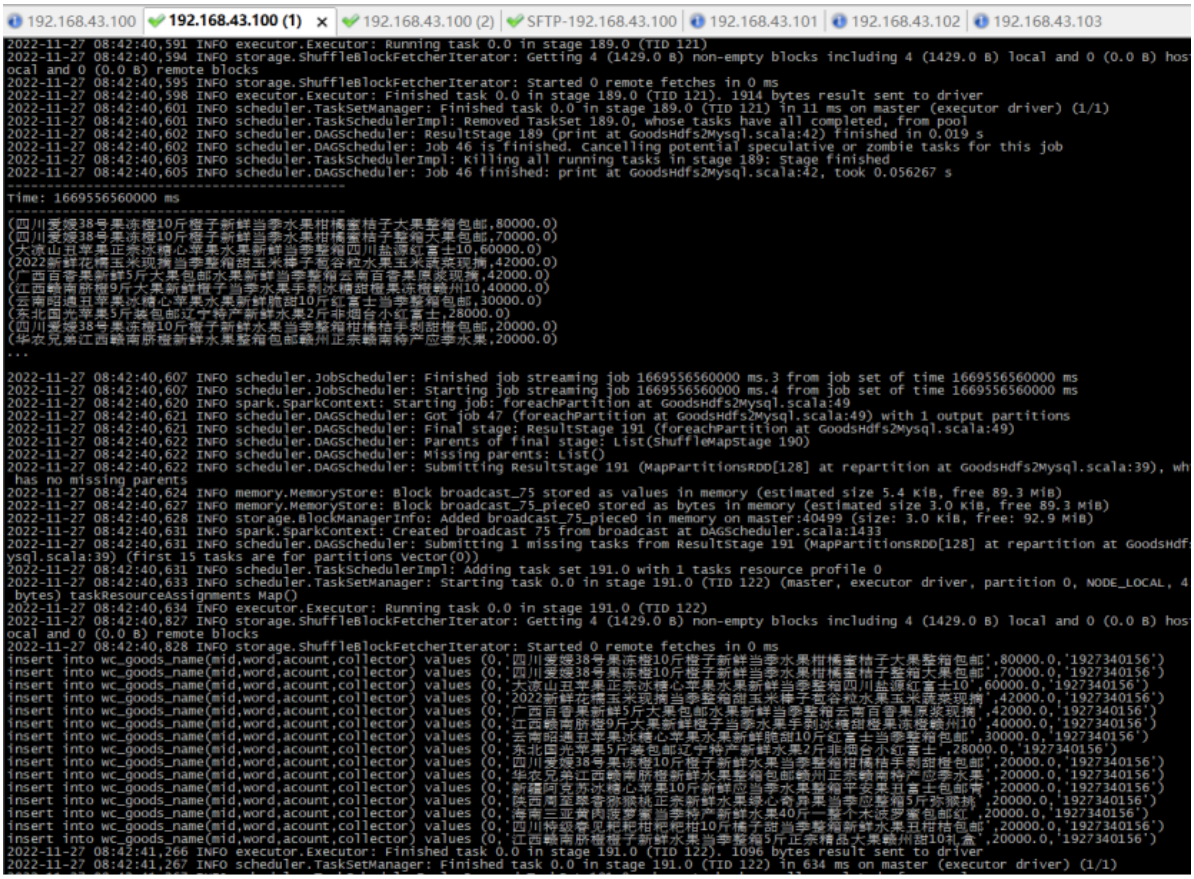




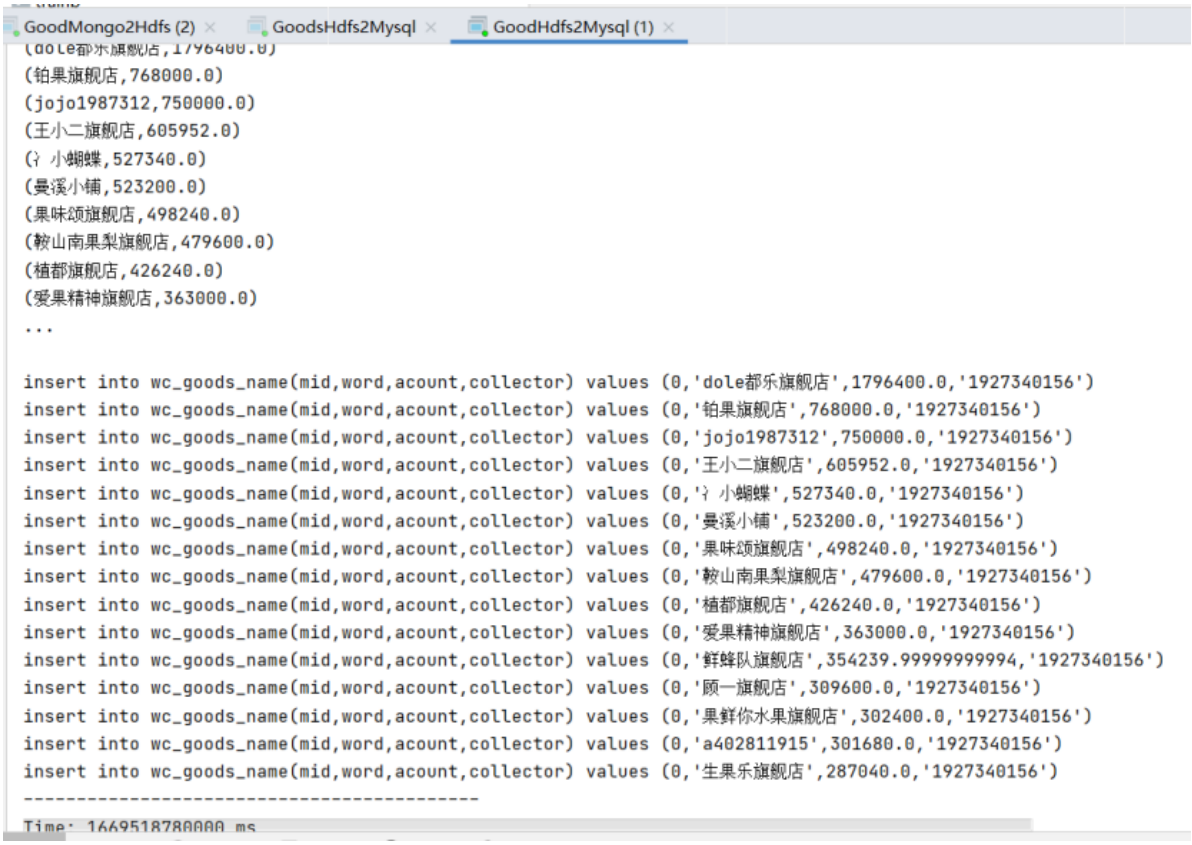
mid	collector	coll_time	word	acount
65679	1927340156	2022-11-27 03:28:11	赣南脐橙5斤装新鲜现摘放置35天更黄更甜	63
65681	1927340156	2022-11-27 03:28:11	智利车厘子5斤新鲜水果应当季进口大樱桃4j大果孕妇礼盒整箱包邮	45
65684	1927340156	2022-11-27 03:28:11	云南紫皮百香果5斤现摘大果包邮德宏特产西番莲新鲜应季孕妇水果	44
65685	1927340156	2022-11-27 03:28:11	山西吉县壶口苹果10斤水果新鲜当季整箱脆甜冰糖心特级红富士苹果	44
65687	1927340156	2022-11-27 03:28:11	沙粉面甜苹果秦冠陕西10斤刮泥香非花蕉果红蛇牛老人婴儿新鲜水果	44
65690	1927340156	2022-11-27 03:28:11	当季现摘云南昭通丑苹果冰糖心高山苹果孕妇水果新鲜脆甜整箱包邮	44
65691	1927340156	2022-11-27 03:28:11	人参果新鲜云南石林特大水果七彩人生果10斤圆果	44
65693	1927340156	2022-11-27 03:28:11	福建黄金爆汁葡萄柚管溪蜜柚西柚新鲜平和官溪红心柚子水果台湾	44
65696	1927340156	2022-11-27 03:28:12	山东烟台苹果805斤正宗栖霞红富士条纹脆甜新鲜水果包邮	44
65697	1927340156	2022-11-27 03:28:12	2022头茬红富士苹果水果新鲜当季陕西旬邑正宗脆甜苹果510斤整箱	44
65700	1927340156	2022-11-27 03:28:12	四川丹棱爱媛38号果冻橙红美人新鲜水果手剥橙子正宗特级大果顺丰	44
65702	1927340156	2022-11-27 03:28:12	东北国光苹果5斤装包邮辽宁特产新鲜水果2斤非烟台小红富士	44
65703	1927340156	2022-11-27 03:28:12	浙江临海涌泉蜜桔无核橘子新鲜薄皮台州黄岩桔子砂糖当季整箱水果	44
65705	1927340156	2022-11-27 03:28:12	现货智利进口车厘子新鲜水果包邮2斤应当季整箱礼盒大果4j孕妇3J	44
65707	1927340156	2022-11-27 03:28:12	无花果鲜果5斤新鲜现摘水果青红皮大果特级当季整箱非威海包邮	44

在hadoop集群上运行：

spark-submit --master yarn --class traind.GoodsHdfs2Mysql /home/hadoop/train.jar



2、统计营收前15名的商铺：



home.hddly.cn | test | 运行 | 停止 | 解释

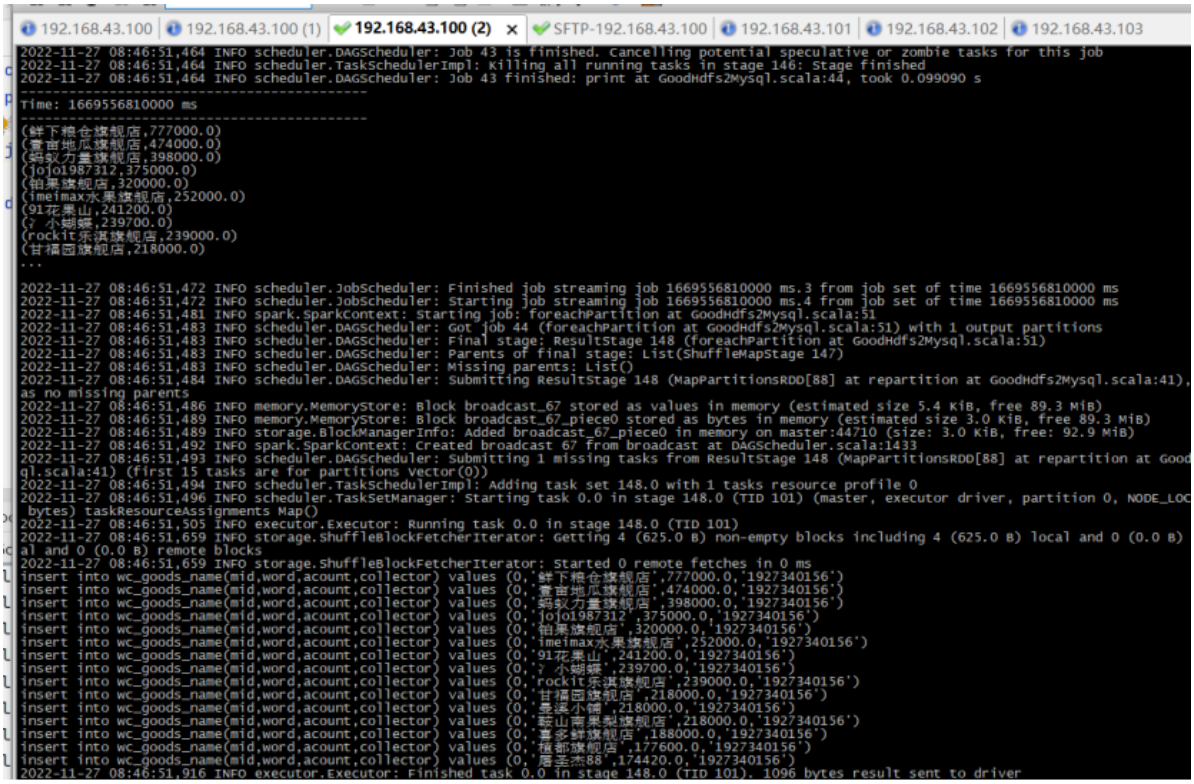
```
1  select * from wc_goods_name where collector="1927340156" and LENGTH(word)<34;
```

信息 | 结果 1 | 剖析 | 状态

mid	collector	coll_time	word	acount
65710	1927340156	2022-11-27 03:28:32	甘福园旗舰店	8105404
65712	1927340156	2022-11-27 03:28:32	天猫超市	7539331
65714	1927340156	2022-11-27 03:28:32	探味君旗舰店	6450600
65716	1927340156	2022-11-27 03:28:32	福瑞达旗舰店	6319056
65718	1927340156	2022-11-27 03:28:32	王小二旗舰店	5008262
65720	1927340156	2022-11-27 03:28:32	源生鲜旗舰店	3920799
65722	1927340156	2022-11-27 03:28:32	绿念旗舰店	3854220
65724	1927340156	2022-11-27 03:28:32	铂果旗舰店	3396972
65726	1927340156	2022-11-27 03:28:32	湖南怀化麻阳县水果商行	2600000
65728	1927340156	2022-11-27 03:28:32	dole都乐旗舰店	2095800
65729	1927340156	2022-11-27 03:28:32	麻城曾氏水果旗舰店	1784400
65731	1927340156	2022-11-27 03:28:32	聚小美旗舰店	1782000
65734	1927340156	2022-11-27 03:28:32	世纪村旗舰店	1601200
65736	1927340156	2022-11-27 03:28:32	小城外旗舰店	1259060
65738	1927340156	2022-11-27 03:28:32	褚橙官方旗舰店	1188000

在hadoop集群上运行：

spark-submit --master yarn --class traind.GoodHdfs2Mysql /home/hadoop/train.jar

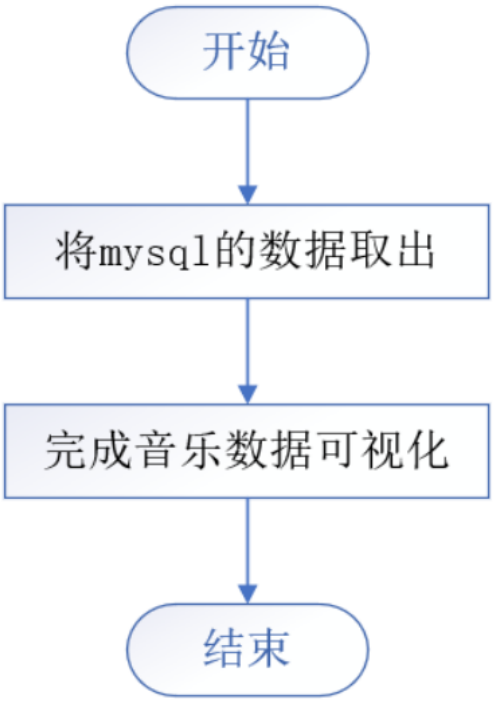


1.1.1. 数据可视(10分)

功能说明：使用python语言绘制可视化图片，并上传到web端

功能设计：使用python语言的barplot()根据存储在mysql的分析结果绘制直方图图片，并将这些图片上传到指定的web网页。

<补充流程图>



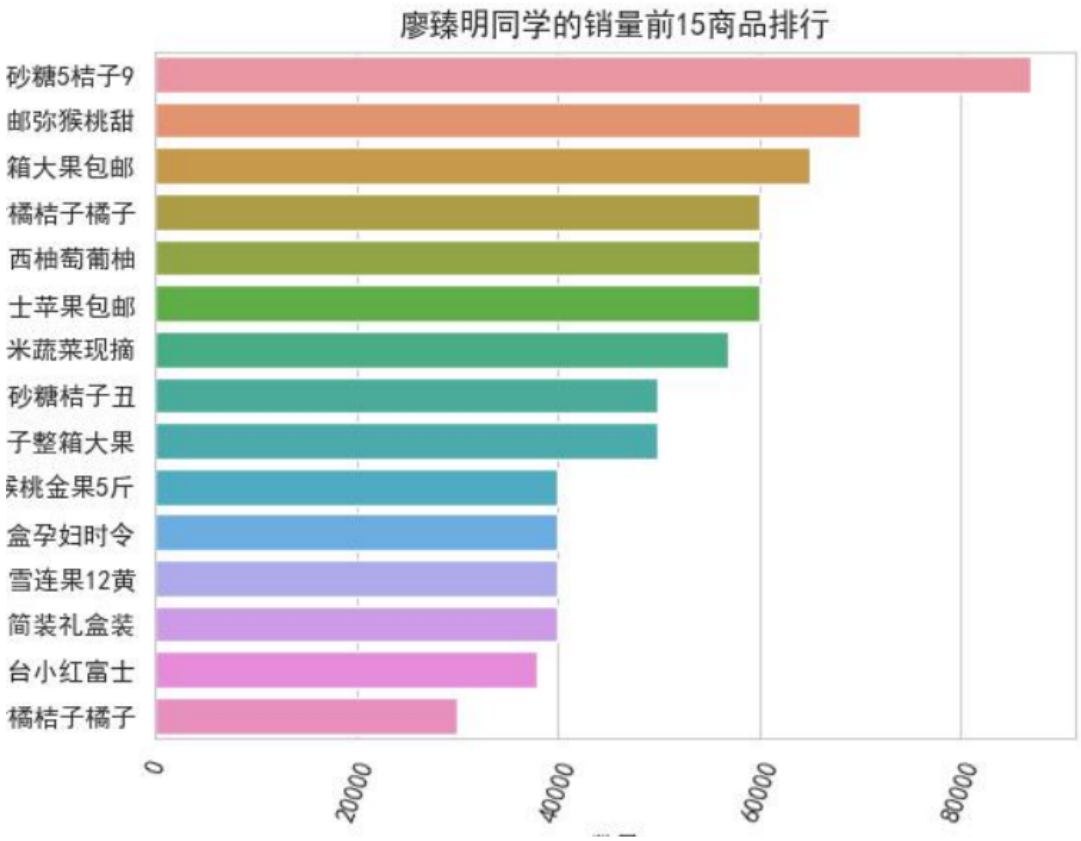
源码实现：

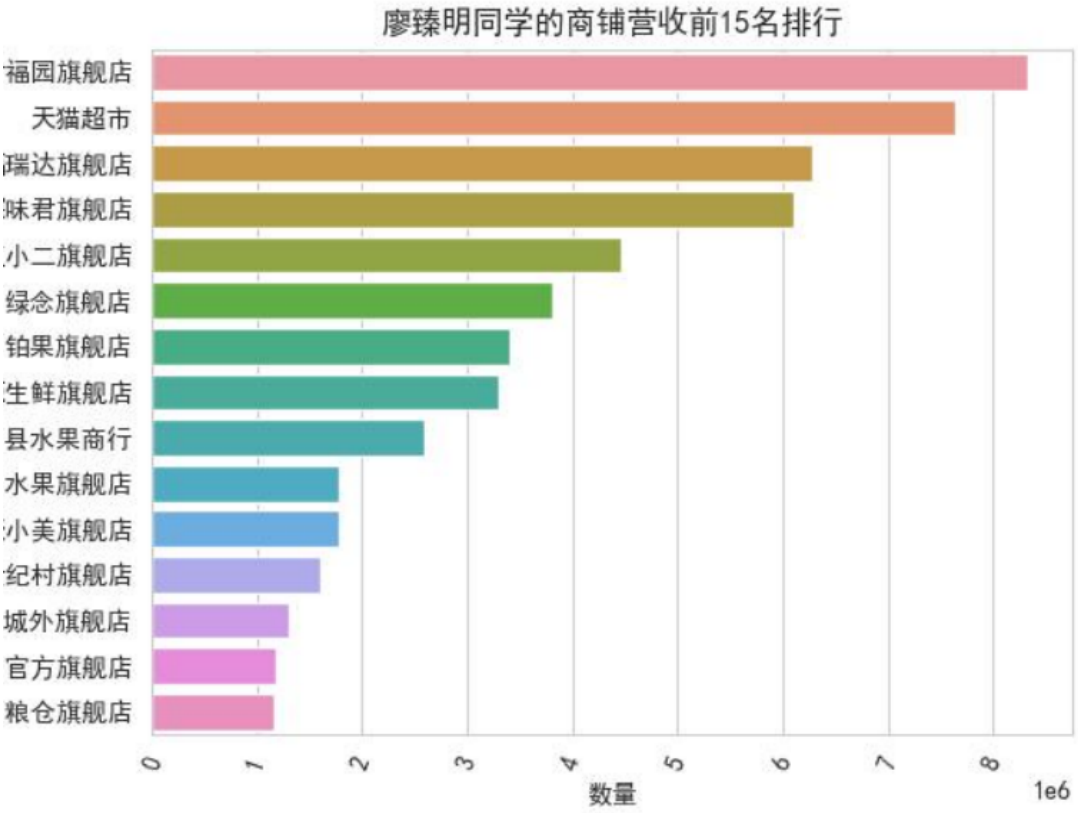
```
# coding=utf-8
from bgutils.ftpUtil import ftpUtil
from bgutils.mysqlUtil import mysqlUtil
from matplotlib import pyplot as plt
import seaborn as sns
```

```
# 数据可视化
class drawpic:
    def draw(self,myname,mycode):
        mutil= mysqlUtil()
        ftputil=ftpUtil()
        # SELECT * FROM test.wc_goods_name where word!='' order by acount desc;
        SQL = '''select  * from wc_goods_name where word!="" and collector="'''+ \
              mycode +'''" and LENGTH(word)<34 order by acount desc '''
        # 调用 mysqlconn 类的 query() 方法
        df_data = mutil.query(sql=SQL)
        # df_data.to_csv
        # 使用seaborn库绘图
        sns.set_style('whitegrid', {'font.sans-serif':['simhei', 'Arial']})
        # 设置中文字体
        #plt.rcParams['font.sans-serif'] = font
        # 设置正常显示负号
        plt.rcParams['axes.unicode_minus']=False

        count = df_data['acount']
        index = df_data['word']
        sns.barplot(x=count, y=index)
        plt.xticks(rotation=70)
        plt.xlabel('数量')
        plt.ylabel('商品名称')
        plt.title(myname+'同学的商铺营收前15名排行')
        # plt.show()
        idxproject= "22002"  #22001是项目编号，长度5位
        idxpic="06" #04是图片在该项目中的顺序号，长度2位
        filepath=mycode+"_"+idxpic +".jpg" #生成的文件以学号+序号+后缀组成
        plt.savefig(filepath)
        ftputil.putfile_stud(filepath, idxproject,mycode,idxpic);
```

```
if __name__ == "__main__":
    draw=drawpic();
    myname="廖臻明" #请将引号中的myname替换为本人姓名,如：张三
    mycode="1927340156" #请将引号中的mycode替换为本人学号，如:2019001001
    draw.draw(myname,mycode);
```

运行截图：





1.4. 项目小结（5分）

本次实验主要是将Spark专业课程的学习内容进行巩固。在基础环境搭建完成的情况下，先是在pyecharm上搭建Scrapy框架，使用Scrapy框架采集电商平台水果类目的数据，并且将数据存储进mongodb数据库；为了方便后续的分析使用scala将mongodb数据库的数据转移至Hdfs文件系统当中；再使用spark streaming分析数据，根据销量取前15名的商品，统计营收前15名的商铺，并将分析结果存储进mysql数据库；最后使用pyecharm或jupyter工具运用python将存储在mysql的分析结果进行可视化，并将可视化结果上传至web端以便将来进行展示。

总体过程比较复杂，但难度简单，加深了对专业知识的巩固和认识。主要难度集中在各种环境的搭建上，可能配置不当会不停报错，在老师和同学的帮助下顺利完成。

通过这一次实训让我更加熟练的使用Spark Streaming进行数据处理和数据分析，相信这次实训对日后的工作有较大的帮助。

1.5. 附件

整个过程中实现的源码，包括采集的源码，spark分析源码，可视化的源码，全部打包成rar文件另行提交。

实现参考

数据采集

源码参考：

https://jihulab.com/biglab-share/scrapy/-/tree/main/b59510/chap9/music_scrapy?ref_type=heads

数据存储(mongodb->hdfs)

源码参考：

https://jihulab.com/biglab-share/spark/-/blob/main/b59510/src/traina/MusicMongo2Hdfs.scala?ref_type=heads

数据分析与存储(spark streaming)

源码参考

https://jihulab.com/biglab-share/spark/-/blob/main/b59510/src/traina/MusicHdfs2Mysql.scala?ref_type=heads

数据可视

源码参考一(pycharm)：

https://jihulab.com/biglab-share/scrapy/-/tree/main/b59510/chap9/music_view?ref_type=heads

如何下载源码

以上源码包含在两个git项目：

```
https://jihulab.com/biglab-share/scrapy.git
https://jihulab.com/biglab-share/spark.git
```

由于 jihulab.com进入收费，我们即将切换到自建git服务器上，切换方法如：

使用资源管理器进入scrapy目录，在地址栏中输入cmd进入命令行，运行：

```
git remote set-url origin http://home.hddly.cn:8093/biglab-share/scrapy.git
```

同样，使用资源管理器进入spark目录，在地址栏中输入cmd进入命令行，运行：

```
git remote set-url origin http://home.hddly.cn:8093/biglab-share/spark.git
```

GIT安装和下载源码视频

学习视频