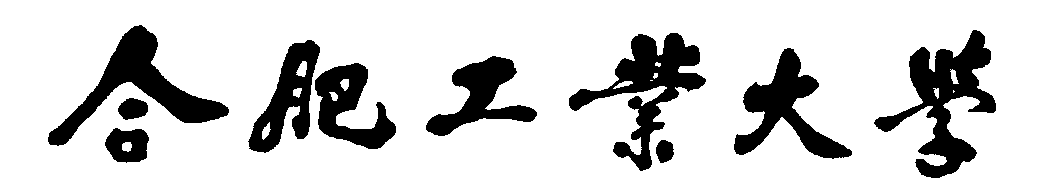
****

专业课程

（计算机与信息学院）

大数据处理技术

课程报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术21-1班 |
| 学生姓名及学号 | 杨程锦-2021214710 |
| 课程教学班号 |  |
| 实验序号及名称 | 课程报告—实验指导 |
| 任 课 教 师 | 吴共庆 |
| 实验指导教师 | 吴共庆 |
|  |  |
| 2022～2023学年第 二 学期 | |

**目录**

[**一、 Flink 基本介绍** 4](#_Toc139668529)

[**1.1 Flink介绍：** 4](#_Toc139668530)

[**1.2 Flik的应用场景：** 4](#_Toc139668531)

[**1.3 Flink的流处理特性** 4](#_Toc139668532)

[**1.4 Flink的优势** 5](#_Toc139668533)

[**1.5 Flink的四大基石** 6](#_Toc139668534)

[**1.6 Flink的流、批处理** 6](#_Toc139668535)

[**二、 Flink架构体系** 6](#_Toc139668536)

[**2.1 Flink的重要角色** 6](#_Toc139668537)

[**2.2 无界数据流与有界数据流** 7](#_Toc139668538)

[**2.3 Flink的编程模型** 8](#_Toc139668539)

[**三、 Flink 集群搭建** 9](#_Toc139668540)

[**3.1 Flinlk的安装模式** 9](#_Toc139668541)

[**3.2 Flinlk -standalone模式安装** 10](#_Toc139668542)

[**四、 使用IDEA编写Flink 程序** 21](#_Toc139668543)

[**4.1构建工程 导入maven依赖** 21](#_Toc139668544)

[**4.2 创建 log4j.properties 文件** 21](#_Toc139668545)

[**4.3 添加scala 依赖** 22](#_Toc139668546)

[**4.4 编写wordcount 程序 （scalal语言）** 22](#_Toc139668547)

[**五、 爬取天气网站数据使用Flink清洗制作词云。** 24](#_Toc139668548)

[**5.1 使用爬虫爬取网页天气数据** 24](#_Toc139668549)

[**5.2 核心代码** 25](#_Toc139668550)

[**5.3 使用Flink 进行数据清洗** 26](#_Toc139668551)

[**5.4 可视化展示** 29](#_Toc139668552)

[**六、 遇到的问题与解决方案** 30](#_Toc139668553)

[**6.1 Flink local模式集群搭建成功后，无法刷新Web UI。** 30](#_Toc139668554)

[**6.2 无法上传数据到HDFS中** 31](#_Toc139668555)

[**6.3 在IDEA 中，首次创建程序运行不成功** 31](#_Toc139668556)

[**6.4 程序运行报错** 32](#_Toc139668557)

[**6.5 程序并行度** 33](#_Toc139668558)

[**七、 感想与体会** 34](#_Toc139668559)

[**八、 参考资料** 35](#_Toc139668560)

1. **Flink 基本介绍**

**1.1 Flink介绍：**

Apache Flink 是一个框架和分布式处理引擎，用于对无界和有界数据流进行状态计算。Flink被设计在所有常见的集群环境中运行，以内存执行速度和任意规模来执行计算。

**1.2 Flik的应用场景：**

* 电商和市场营销：实时报表、广告投放、实时推荐。
* 物联网：实时数据采集、实时报警。
* 物流配送及服务：订单状态跟踪、信息推送。
* 银行和金融业：实时结算、风险检测。

**1.3 Flink的流处理特性**

* 支持高吞吐、低延迟、高性能的流处理
* 支持带有事件时间的窗口操作
* 支持有状态计算的Exactly-one语义
* 支持高度灵活的窗口操作，支持基于time、count、session、以及data-driven的窗口操作
* 支持具有背压功能的持续流模型
* 支持基于轻量级分布式快照实现的容错
* 一个运行时同时支持Batch on Streaming处理和Streaming处理
* Flink在JVM内部实现了自己的内存管理
* 支持迭代计算
* 支持程序自动优化，避免特定情况下Shuffle、排序等昂贵操作，中间结果有必要进行缓存

**1.4 Flink的优势**

大数据的4代计算引擎

* 第一代：Hadoop、MapReduce批处理 Mapper、Reducer2
* 第二代：DAG框架（Oozie、Tez）,Tez+MapReduce批处理
* 第三代：Spark批处理、流处理、SQL高层API支持 自带DAG 内存迭代计算、性能较之前大幅提升
* 第四代：Flink批处理、流处理、SQL高层API支持 自带DAG 流式计算性能更高、可靠性更高

表 1 Flink与Spark的差异

|  |  |  |
| --- | --- | --- |
|  | SparkStreaming | Flink |
| 定义 | 弹性的分布式数据集，并非真正的实时计算 | 真正的流计算，就像storm一样；但flink同时支持有限的数据流计算（批处理） |
| 高容错 | 基于RDD和checkpoint,比较沉重 | Checkpoint（快照），非常轻量级 |
| 内存管理 | JVM相关操作暴露给用户 | Flink在JVM中实现的是自己的内存管理 |
| 延时 | 中等100ms | 低10ms |

**1.5 Flink的四大基石**

* **checkpoint:** 基于chandy-lamport算法实现分布式计算任务的⼀致性语义；
* **state:** flink中的状态机制，flflink天⽣⽀持state,state可以认为程序的中间计算结果或者是历史计算结果；
* **time:** flink中⽀持基于事件时间和处理时间进⾏计算，spark streaming只能按照process time进⾏处理；基于事件时间的计算我们可以解决数据迟到和乱序等问题。
* **window:** flink提供了更多丰富的window,基于时间，基于数量，session window,同样⽀持滚动和滑动窗⼝的计算。

**1.6 Flink的流、批处理**

* Flink将批处理视作一种特殊的流处理，Flink中的所有计算都是流式计算。
* Flink分别提供了面向流式处理的接口（DataStream API）和面向批处理的接口（DataSet API）

1. **Flink架构体系**

**2.1 Flink的重要角色**



Flink角色关系图

**JobManger处理器：**

类似spark中master，负责资源申请，任务分发，任务调度执⾏，checkpoint的协调执⾏；可以搭建HA，双master。

**TaskManager处理器：**

类似spark中的worker，负责任务的执⾏，基于dataflflow(spark中DAG)划分出的task;与jobmanager保持⼼跳，汇报任务状态。

**2.2 无界数据流与有界数据流**

无界数据流：数据流是有⼀个开始但是没有结束；

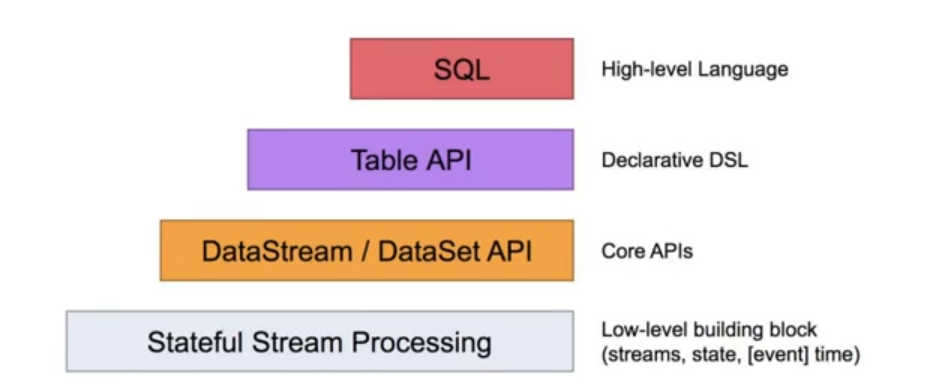
有界数据流：数据流是有⼀个明确的开始和结束，数据流是有边界的。

Flink处理流批处理的思想是：

Flink支持的runtime(core 分布式流计算)支持的是无界数据流，但是对Flink来说可以支持批处理，只是从数据流上来说把有界数据流只是无界数据流的⼀个特例，无界数据流只要添加上边界就是有界数据流。

**2.3 Flink的编程模型**

Flink提供了不同的抽象级别以开发流式或者批处理应用。



Flink 编程模型

最底层级的抽象仅仅提供了有状态流，他将通过过程函数（Process Function）被嵌入到DataStream API中。底层过程函数与DataStream API 相集成，使其可以对某些特定的操作进行底层的抽象，他允许用户可以自由地处理来自一个或者多个数据流的时间，并使用一致的容错的状态。除此之外，用户可以注册事件时间并处理时间回掉，从而使程序可以处理复杂的计算。

实际上，大多数应用并不需要上述的底层抽象，而是针对核心API(Core API)进行编程，比如DataStream API(有界或无界流数据）以及DataSet API(有界数据集）。这些API为数据处理提供了通用的构建模块，比如由用户定义的多种形式的转换(transformations)，连接（joins),聚合(aggregations），窗口操作(windows)等等。DataSet API为有界数据集提供了额外的支持，例如循环与迭代。这些API处理的数据类型以类(classes）的形式由各自的编程语言所表示。

Table API是以表为中心的声明式编程,其中表可能会动态变化(在表达流数据时)。Table API遵循（扩展的）关系模型:表有二维数据结构(schema)(类似于关系数据库中的表)，同时API提供可比较的操作，例如select、project、join、group\_by、aggregate等。Table API程序声明式地定义了什么逻辑操作应该执行，而不是准确地确定这些操作代码的看上去如何。尽管Table API可以通过多种类型的用户自定义函数（UDF）进行扩展，其仍不如核心API更具有表达能了，但是使用起来却更加简洁。除此之外，Table API程序在执行之前会经过内置优化器进行优化。

Flink提供的最高层级的抽象是SQL。这一层抽象在语法与表达能力上与Table API类似,但是是以SQL查询表达式的形式表现程序。SQL抽象与Table API交互密切，同时SQL查询可以直接在Table API定义的表上执行。

1. **Flink 集群搭建**

**3.1 Flink的安装模式**

* local(本地)--单机模式
* standalone--独立模式，Flink自带集群，开发测试环境使用
* Yarn--计算资源统一由Hadoop YARN管理，生产环境测试

本次实验进行standalone模式安装

**3.2 Flinlk -standalone模式安装**

1. 首先搭建Hadoop集群，这一部分我省略，之前在大数据处理技术课程中已搭建成功，我直接讲解搭建Flink集群。
2. 在官网下载好安装包：

Flink安装包

1. 集群规划：

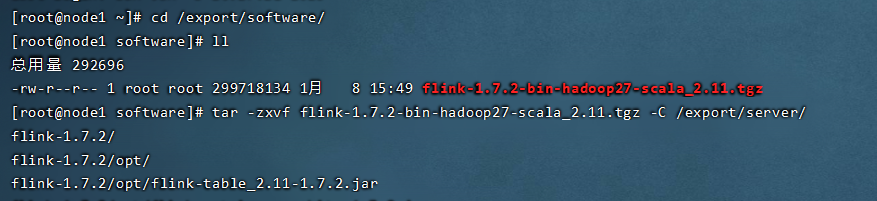
node1（master+slave）、node2（salve）、node2（slave）

1. 将Flink压缩包上传到 node1结点 根目录下 export/software文件夹下。

cd /export/software/

1. 在node1结点进行解压

tar -zxvf flink-1.7.2-bin-hadoop27-scala\_2.11.tgz -C /export/server/

解压Flink安装包

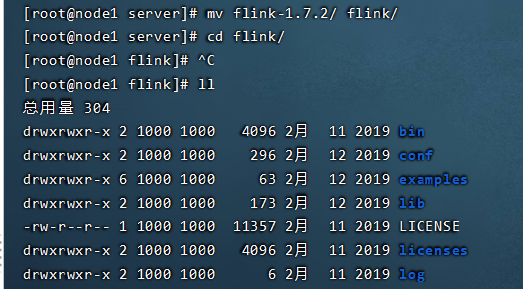
1. 转换文件地址

cd ../server/

1. 修改文件名

mv flink-1.7.2/ flink/

cd flink/

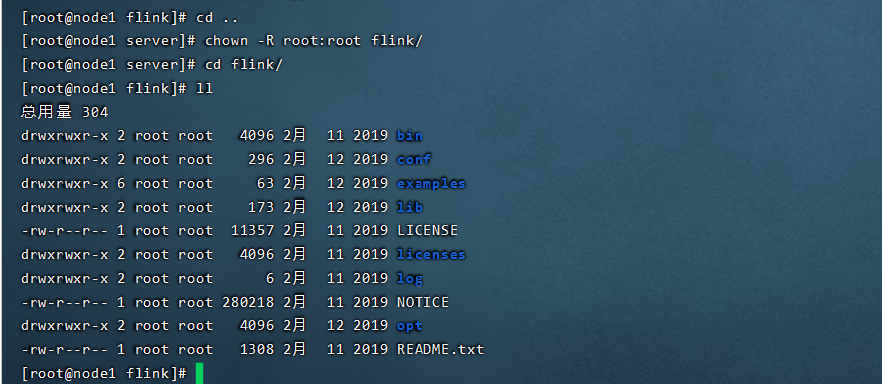


修改文件名

1. 修改用户权限，防止权限出现问题

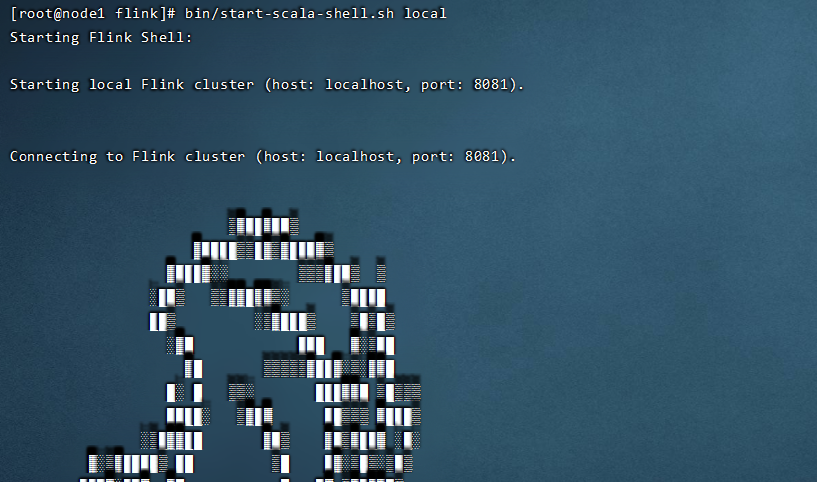
cd ..

chown -R root:root flink/

修改权限

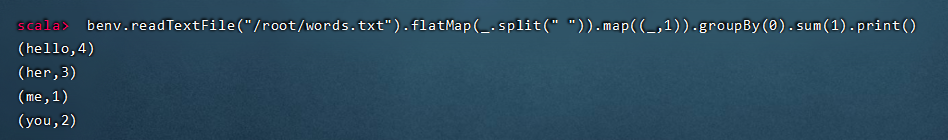
1. 启动本地模式的Flink

bin/start-scala-shell.sh loca

启动 Flink local 模式

1. 启动shell 交互式窗口，并提交一个任务，首先在root目录下上传 words.txt文件，在命令上输入下列代码。（进行统计词频）

benv.readTextFile("/root/words.txt").flatMap(\_.split(" ")).map((\_,1)).groupBy(0).sum(1).print()

输出结果

1. 启动单节点集群

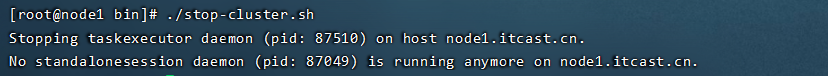
cd bin/

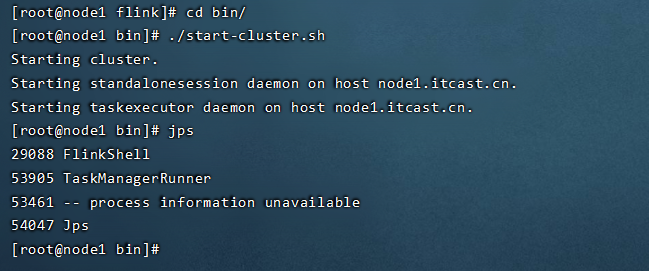
./start-cluster.sh

Jps

1. 停止集群

./stop-cluster.sh

停止集群

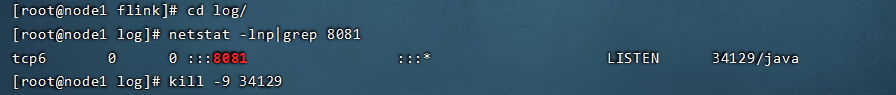
启动集群

1. 遇到的问题，启动集群后输入JPS 没有显示StandaloneSessionclusterEntrypoint。这个原因是8081的端口号被占，我们需要找出这个进程手动消除。

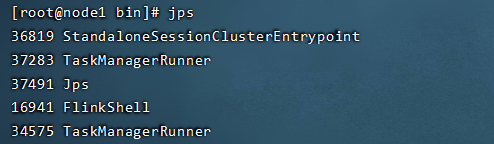
cd log/

netstat -lnp|grep 8081

kill -9 34129

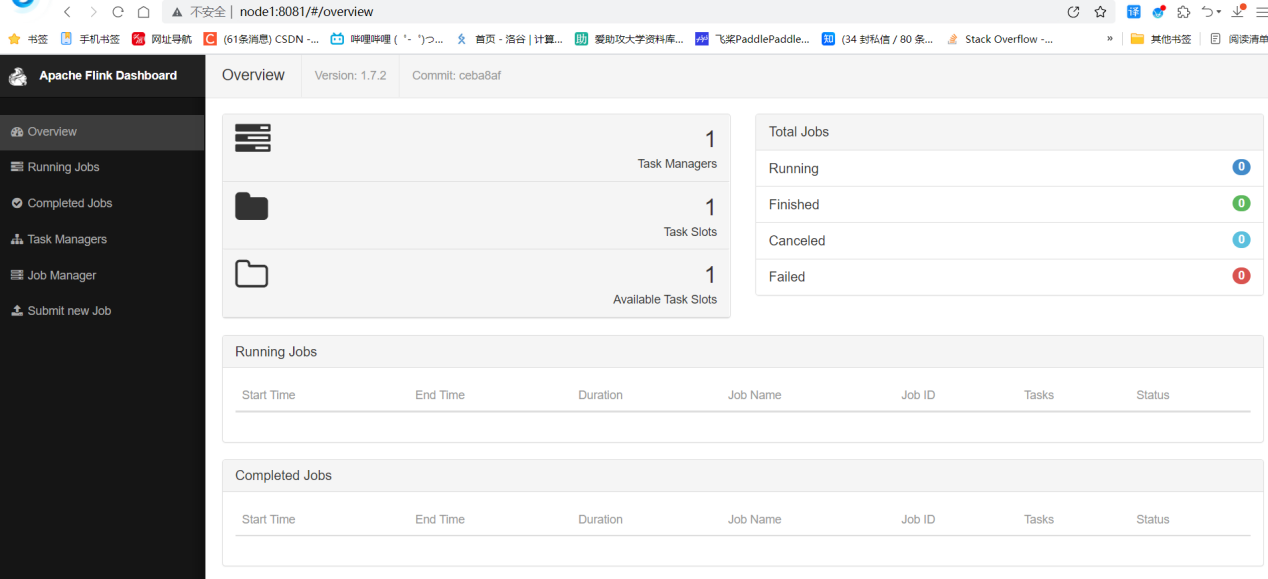
手动消除进程

之后再启动集群就可以看到StandaloneSessionclusterEntrypoint

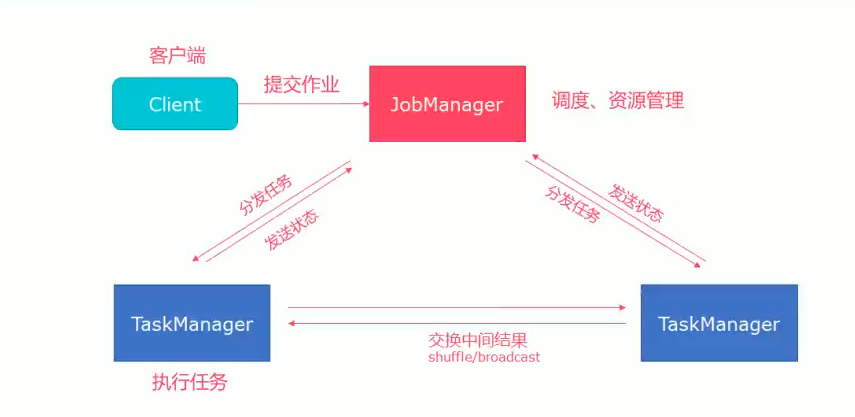


查看jps

1. 在网页上输入 node1:8081可以看到 webUI页面

Flink Web UI页面

1. 接下来在之前的基础上安装standalone集群模式

Flink standalone 模式原理图

修改配置文件 flink-conf,yaml

vim flink-conf.yaml

修改部分：

jobmanager.rpc.address: node1

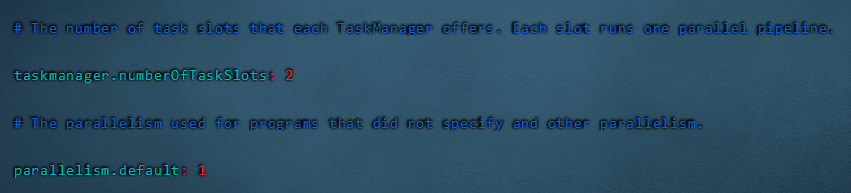
taskmanager.numberOfTaskSlots: 2

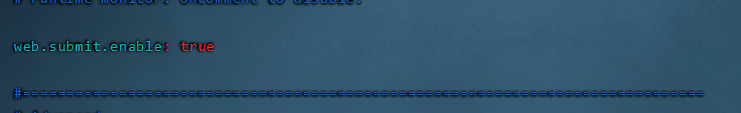
taskmanager.memory.preallocate: false

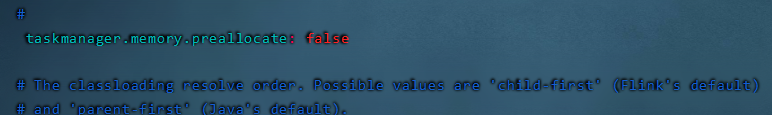
web.submit.enable: true

historyserver.web.port: 8081

修改配置文件

修改配置文件

修改配置文件

修改配置文件

修改配置文件

1. 修改 masters文件

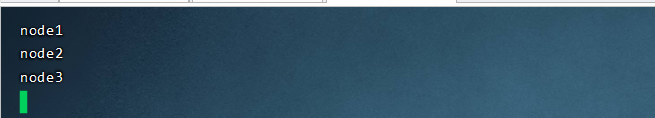
vim masters



修改 masters文件

1. 修改 slaves文件

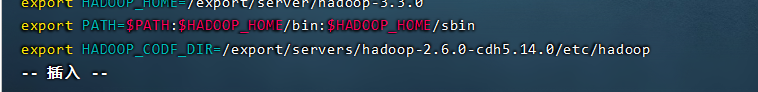
vim slaves



修改 slaves文件

1. 添加HADOOP\_CONF\_DIR环境变量

export HADOOP\_CODF\_DIR=/export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

添加环境变量

1. 分发到另外两个结点

cd /export/server/

scp -r flink/ node2:$PWD

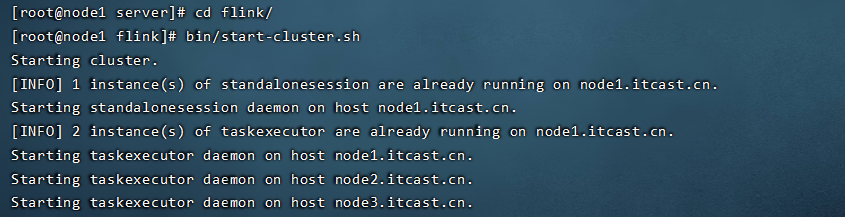
scp -r flink/ node3:$PWD

分发环境配置到另外两个结点

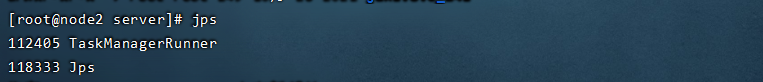
1. 启动集群，在node1上运行下列指令

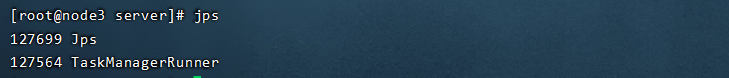
cd flink/

bin/start-cluster.sh

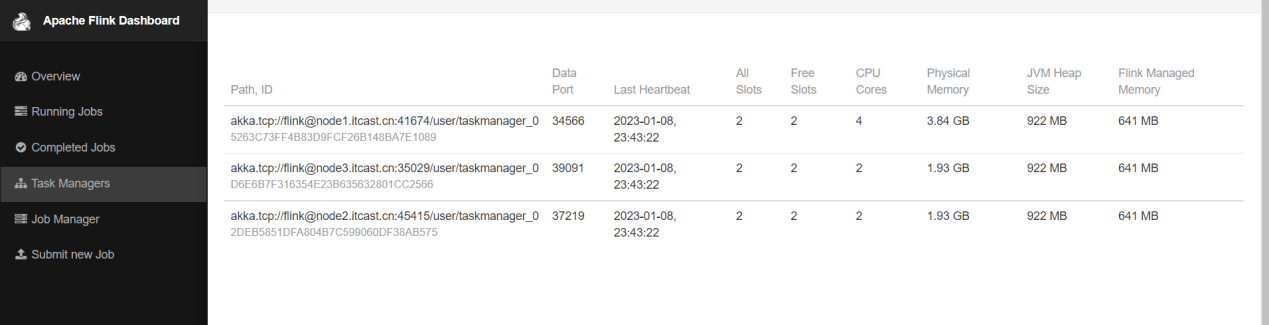
启动集群

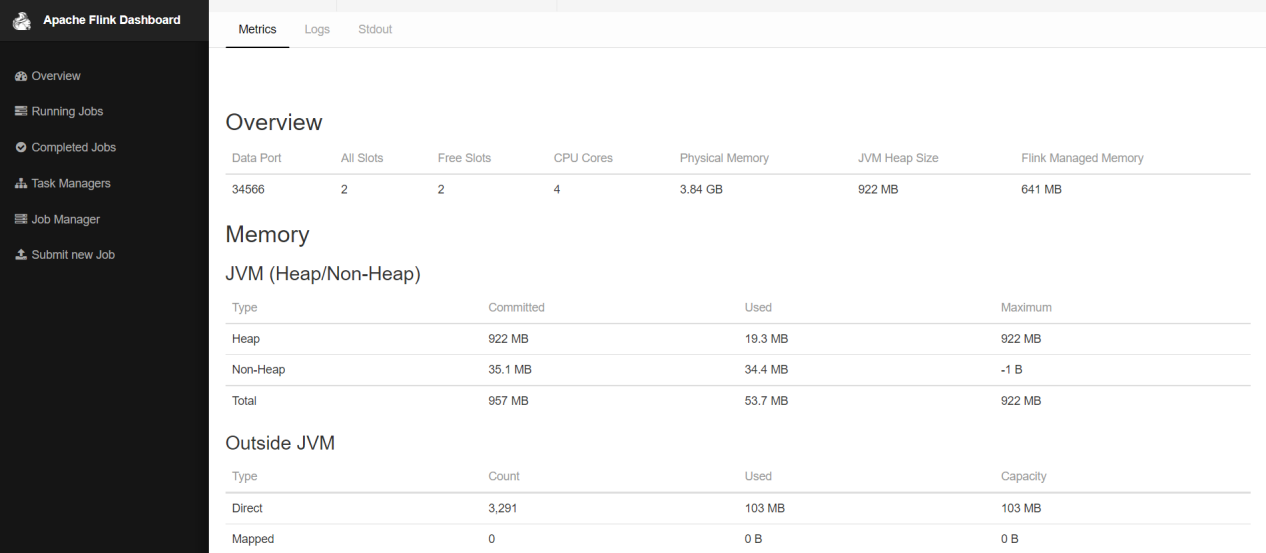
我们可以看到在node1上运行了jobmanger,taskmanger、node2上运行了jtaskmanger、node3上运行了taskmanger。我们同样可以在node2、node3上输入 jps进行查看。

node2 jps 查看

node3 jps查看

Flink Web UI页面

Flink Web UI页面

Flink Web UI页面

这里还会出现 Task Manger 数目不等于3的情况，这是因为8081端口的进程被占用，可以用上面步骤13同样的方式手动消除进程解决。

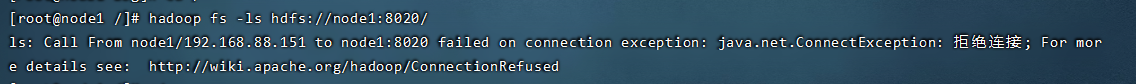
我们在Web UI 页面可以看到集群的具体配置 3个taskmanger,6个 task slots.

1. 进行测试。

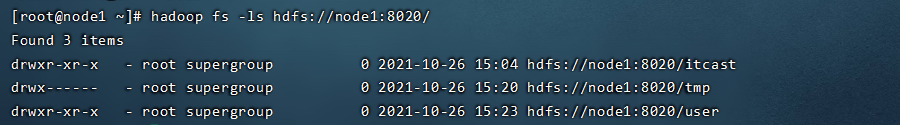
首先上传要清洗的数据到HDFS，这一步要打开HDFS，输入

hadoop fs -ls hdfs://node1:8020/

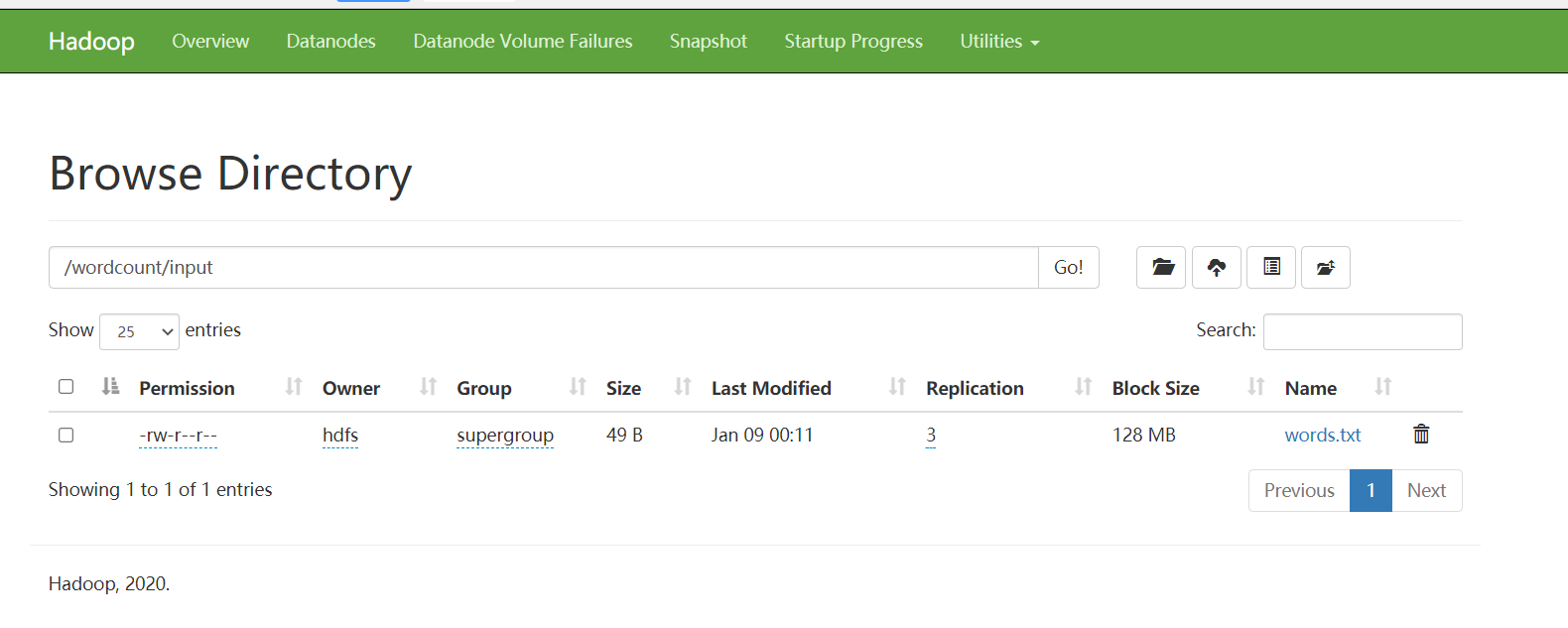
如果显示下方 说明没有启动hadoop 需要先启动hadoop

报错

start-all.sh 启动hadoop

启动hadoop

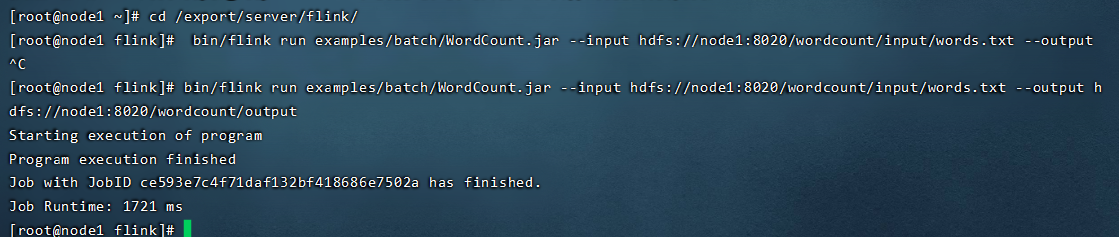
上传words.txt到HDFS中。

HDFS Web UI页面

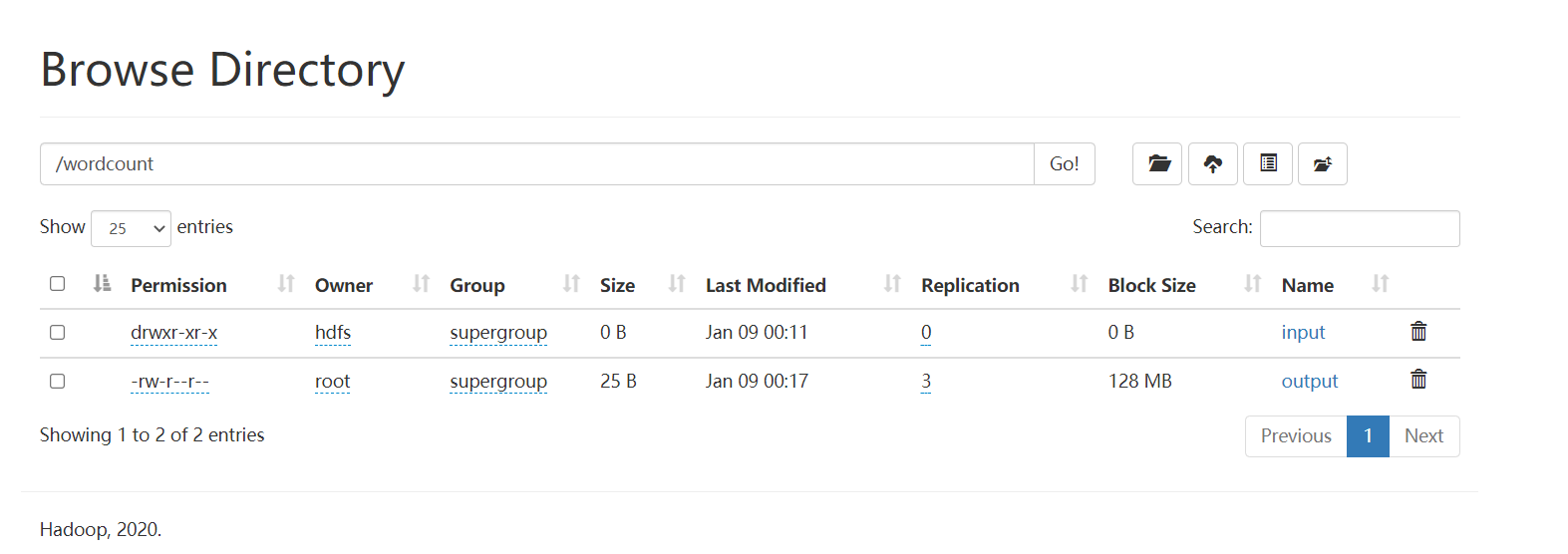
输入下面的指令

cd /export/server/flink/

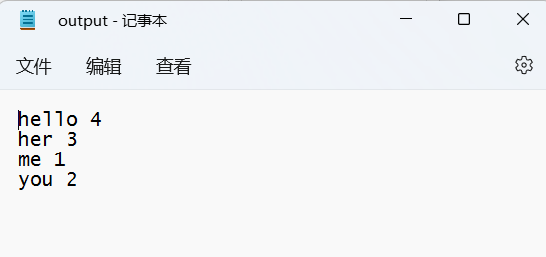
bin/flink run examples/batch/WordCount.jar --input hdfs://node1:8020/wordcount/input/words.txt --output hdfs://node1:8020/wordcount/output

输入指令

在HDFS页面中就可以查看了

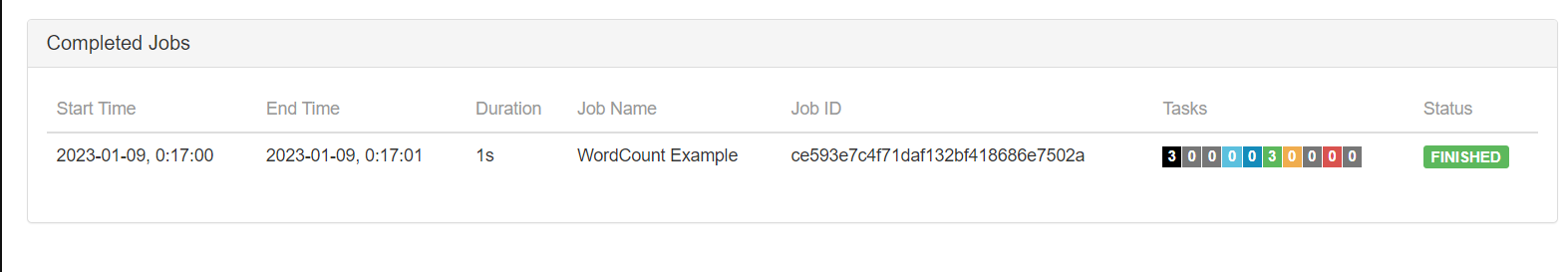
HDFS文件

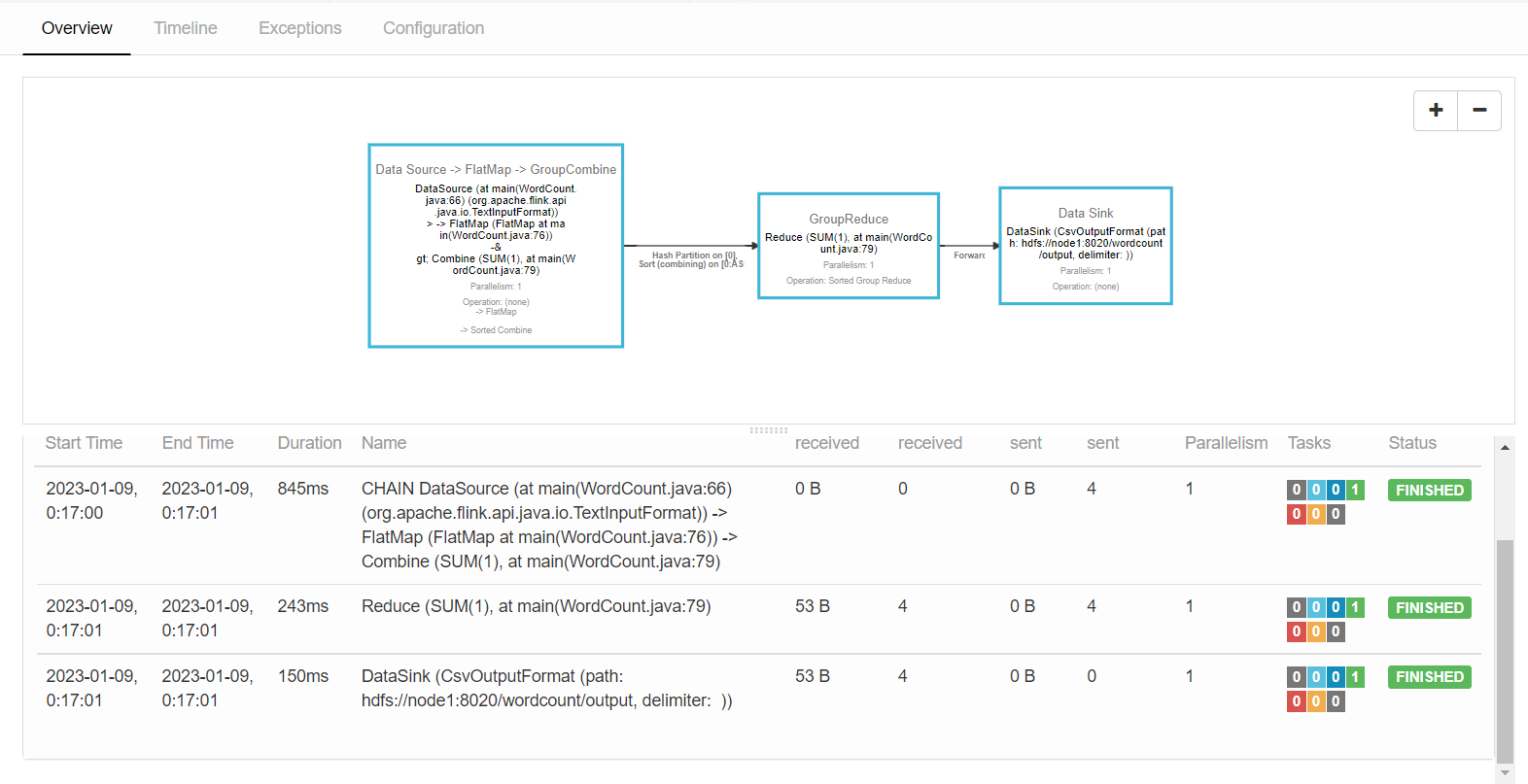
下载到本地查看 输出结果



输出结果

同时我们可以在Flink Web UI页面 看到执行的任务

Flink Web UI 执行任务图

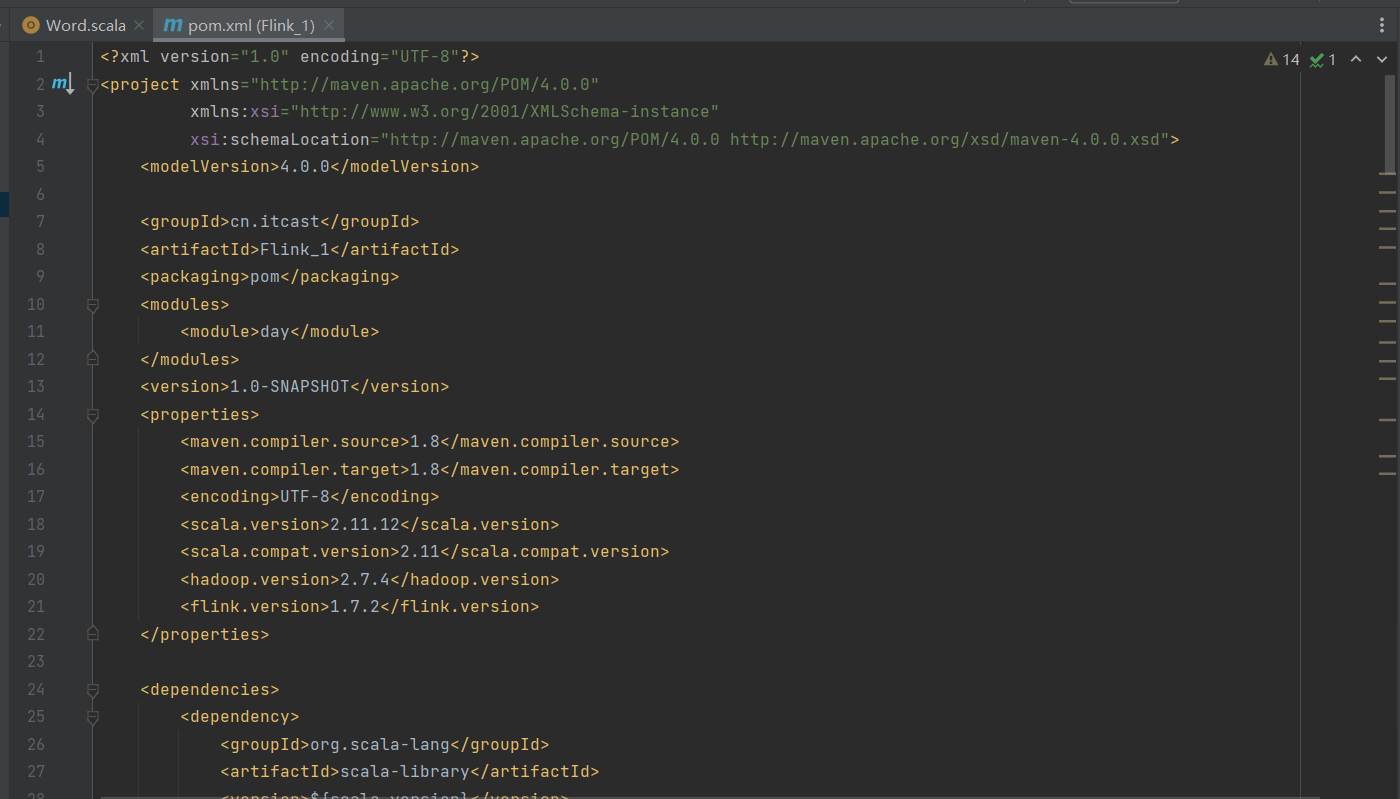
Flink Web UI 执行任务图

至次Flink standalone 模式 全部搭建成功。

1. **使用IDEA编写Flink 程序**

**4.1构建工程 导入maven依赖**

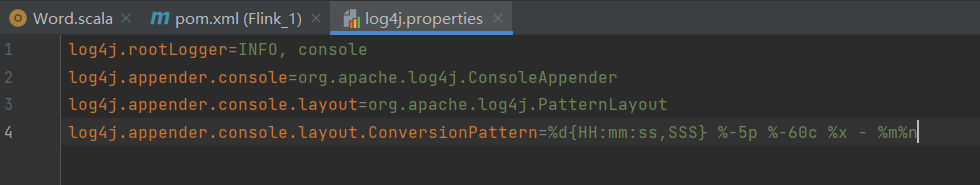
由于导入依赖比较多，代码大，该文件放入了源码压缩包内。

导入依赖

**4.2 创建 log4j.properties 文件**

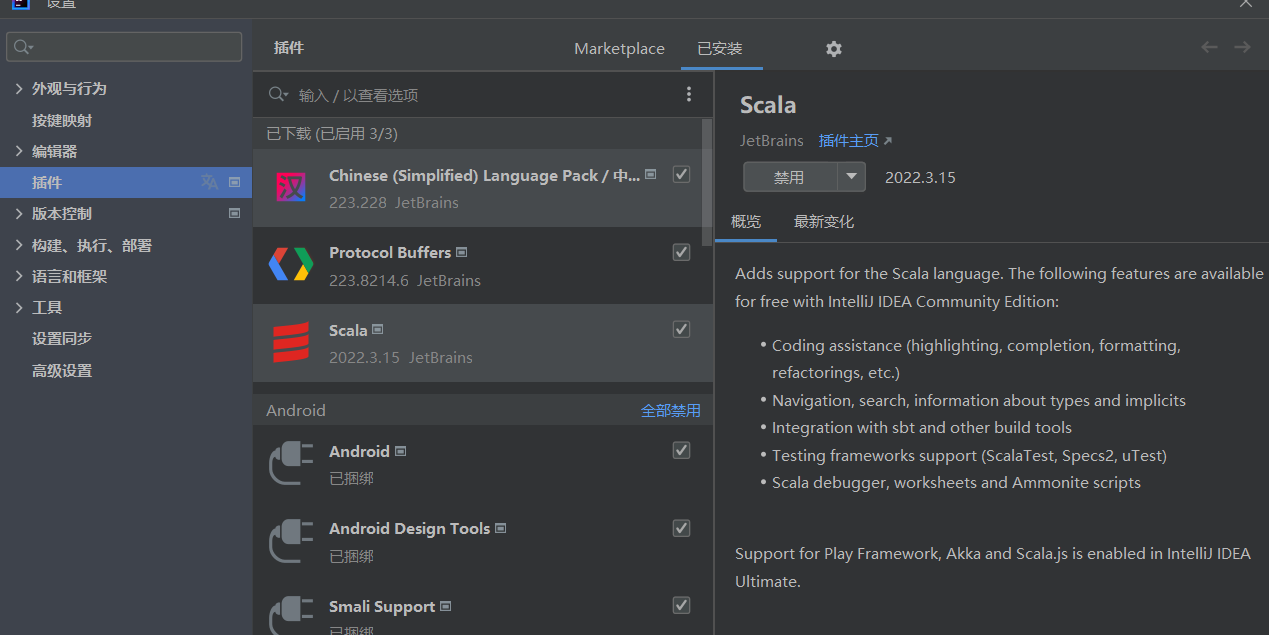
此文件用来调整打印日志结点

1. log4j.rootLogger=INFO, console
2. log4j.appender.console=org.apache.log4j.ConsoleAppender
3. log4j.appender.console.layout=org.apache.log4j.PatternLayout
4. log4j.appender.console.layout.ConversionPattern=%d{HH:mm:ss,SSS} %-5p %-60c %x - %m%n

log4j.properties 文件

**4.3 添加scala 依赖**

需要下载scala 插件

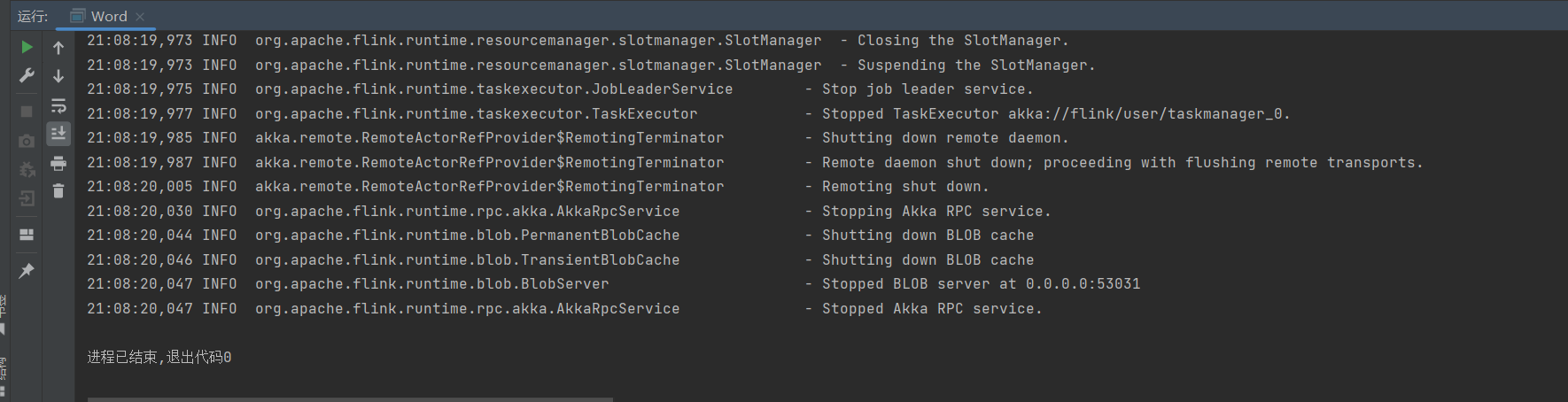
scala 插件下载

**4.4 编写wordcount 程序 （scalal语言）**

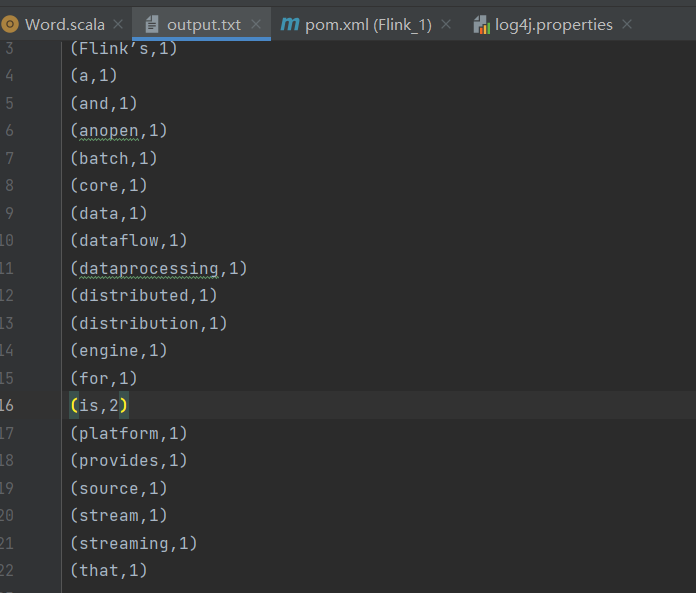
整体流程为 设置程序并行度为1，然后对每行语句按照空格进行切分，切分之后组成（单词，1）的元组，按照单词分组最后进行聚合计算，最后将结果存储到本地文件中。

1. **package** cn.itcast.flink.batch
2. **import** org.apache.flink.api.scala.ExecutionEnvironment
3. **import** org.apache.flink.api.scala.\_
4. /\*
5. \*  使用Flink批处理进行单词计数
6. \* \*/
7. object Word {
8. def main(args: Array[String]): Unit = {
9. /\*
10. 获得一个 execution envirtionment
11. 加载、创建初始数据
12. 指定数据的转换
13. 指定将计算结果放在哪里
14. 触发程序执行
15. \*/
16. val env: ExecutionEnvironment = ExecutionEnvironment.getExecutionEnvironment
17. env.setParallelism(1)
18. val sourceDs: DataSet[String]=env.fromElements(data = "Apache Flink is anopen source platform for distributed " +
19. "stream and batch dataprocessing",
20. "Flink’s core is a streaming dataflow engine that provides data distribution")
21. val wordsDs: DataSet[String] = sourceDs.flatMap(fun = \_.split(" "))
22. val wordAndOneDs:DataSet[(String,Int)] = wordsDs.map(fun = (\_,1))
23. val groupDs:GroupedDataSet[(String,Int)] = wordAndOneDs.groupBy(fields = 0)
24. val aggDs:AggregateDataSet[(String,Int)]=groupDs.sum(field = 1)
25. aggDs.writeAsText(filePath = "./output.txt")
26. env.execute()
27. }
28. }

运行成功

运行成功截图

输出结果

输出结果

1. **爬取天气网站数据使用Flink清洗制作词云。**

**5.1 使用爬虫爬取网页天气数据**

本次爬取中央气象台网站数据，共城市400多

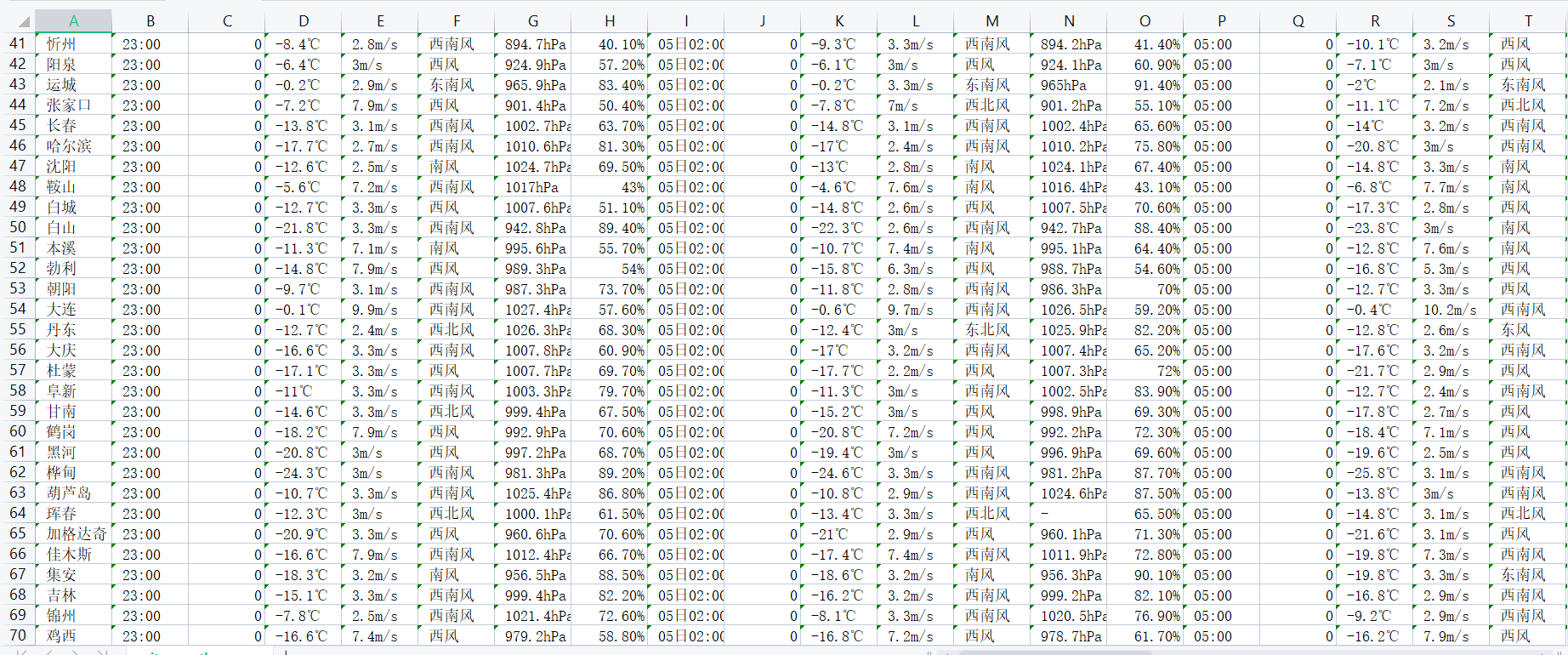


爬取网页截图

**5.2 核心代码**

使用Python Xpath爬虫技术爬取各地天气信息

1. **for** ur **in** content:
2. **print**(ur)
3. i += 1
4. **print**(i)
5. page\_text = requests.get(url=ur.split('\n')[0], headers=headers)
6. page\_text.encoding = 'utf-8'
7. page\_text=page\_text.text
8. tre = etree.HTML(page\_text)
9. city\_name = tre.xpath('//div[@class="container"]//div[@class="cityName"]/text()')[0]
10. # city\_name = city\_name.strip('今天天气详情')
11. **print**(city\_name)
12. fp.write(city\_name)
13. fp.write(',')
14. wea = tre.xpath('//div[@class="hourItems pull-right clearfix"]//div[@id="day0"]/div')
15. **for** w **in** wea:
16. data = w.xpath('./div//text()')
17. fp.write(data[0])  # 时间
18. fp.write(',')
19. **if** data[1]== ' - ':  # 降雨量
20. fp.write('0')
21. **else**:
22. fp.write(data[1])
23. fp.write(',')
24. fp.write(data[2])  # 温度
25. fp.write(',')
26. fp.write(data[3])  # 风速
27. fp.write(',')
28. fp.write(data[4])  # 风向
29. fp.write(',')
30. fp.write(data[5])  # 气压
31. fp.write(',')
32. fp.write(data[6])  # 湿度
33. fp.write(',')
34. fp.write('\n')

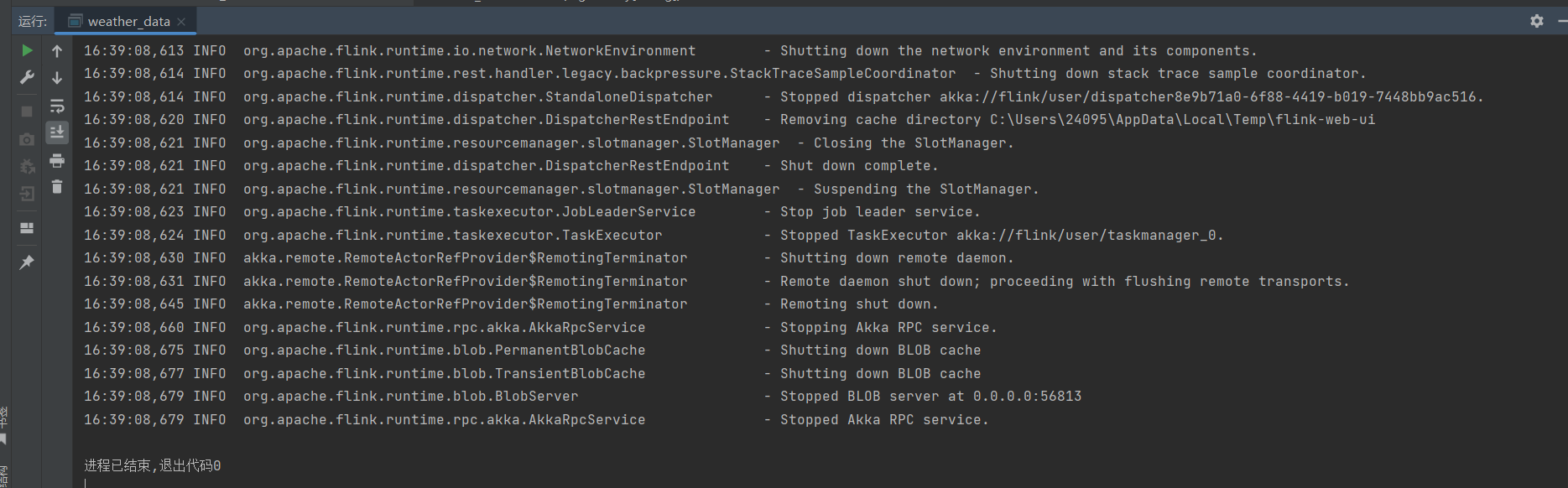
爬取数据

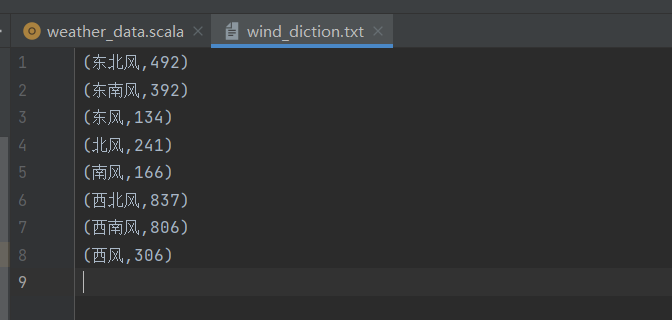
**5.3 使用Flink 进行数据清洗**

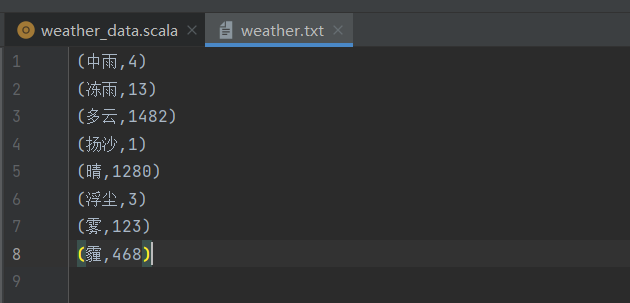
此处使用的是scala语言编写的程序，选取爬取天气数据中风向，天气情况类进行清洗，使用map,groupBy,sum算子统计出 爬取城市天气数据的风向，天气情况。

核心代码：

1. /\*
2. \*  使用Flink批处理进行天气数据清洗
3. \* \*/
4. object weather\_data {
5. def main(args: Array[String]): Unit = {
6. /\*
7. 获得一个 execution envirtionment
8. 加载、创建初始数据
9. 指定数据的转换
10. 指定将计算结果放在哪里
11. 触发程序执行
12. \*/
13. // 获得一个 execution envirtionment
14. val env: ExecutionEnvironment = ExecutionEnvironment.getExecutionEnvironment
15. env.setParallelism(1)  // 设置并行度为1 使输出文件为一个
16. //  加载、创建初始数据
17. val sourceDs: DataSet[String] = env.readTextFile("./city\_weather\_new.csv")
18. // 使用map算子进行数据清洗
19. val wordAndOneDs:DataSet[(String,Int)] = sourceDs.map(
20. fun = line => {
21. val arr:Array[String] = line.split(",")
22. (arr(5),1)
23. }
24. )
25. // 进行聚合
26. val groupDs:GroupedDataSet[(String,Int)] = wordAndOneDs.groupBy(fields = 0)
27. // 累加
28. val aggDs:AggregateDataSet[(String,Int)]=groupDs.sum(field = 1)
29. // 输出文件
30. aggDs.writeAsText(filePath = "./wind\_diction.txt")
31. // 启动程序
32. env.execute()
33. }
34. }

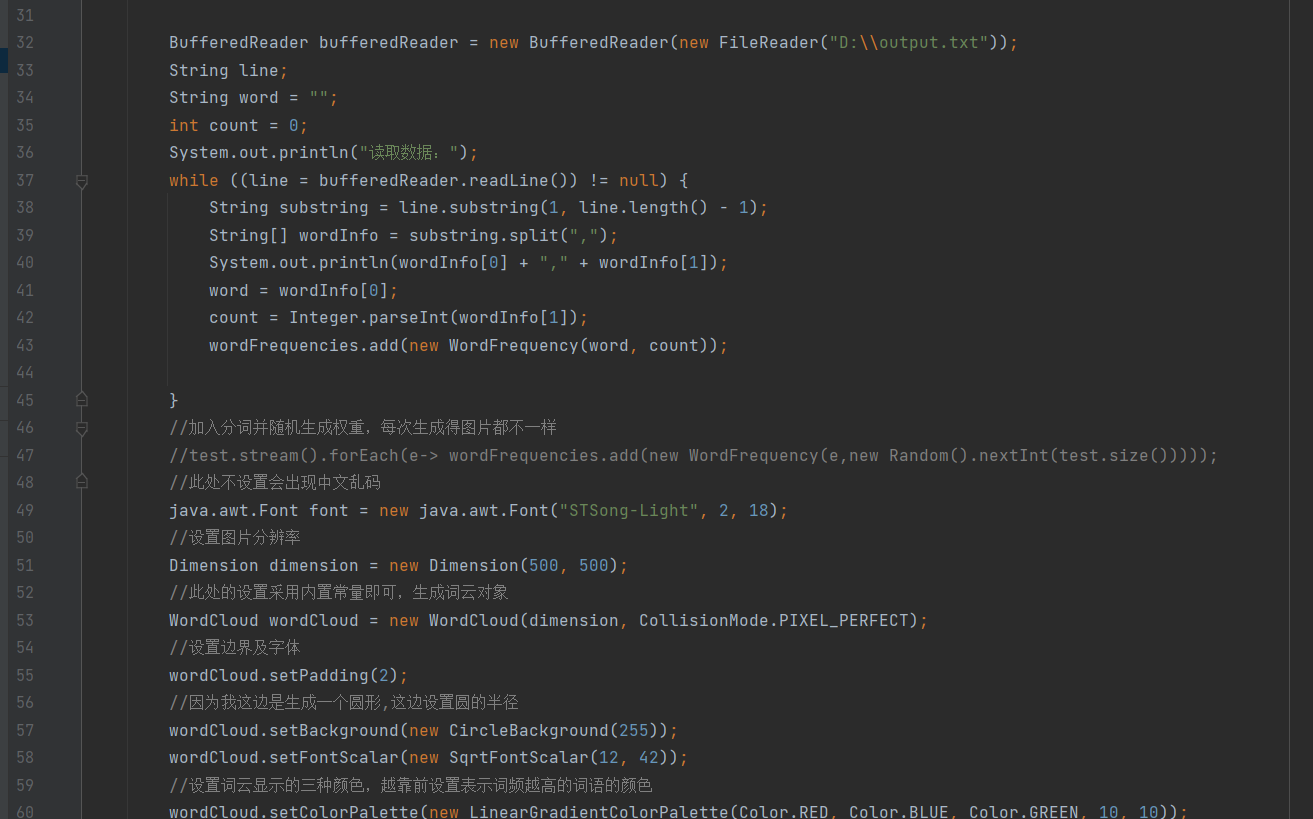
运行成功截图

清洗成功后数据

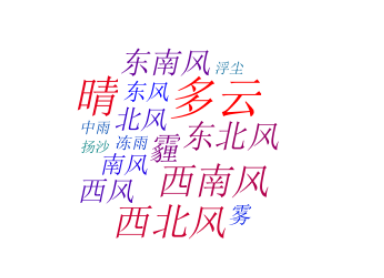
清洗成功后数据

**5.4 可视化展示**

此处使用java 程序编写词云展示，将清洗出来的风向，天气状况数据进行展示。

代码截图

运行成功截图



词云截图

1. **遇到的问题与解决方案**

**6.1 Flink local模式集群搭建成功后，无法刷新Web UI。**

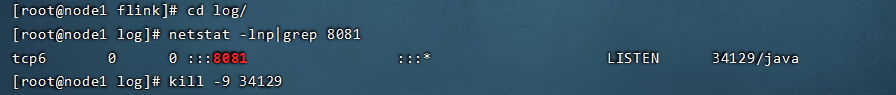
解决方案：

这个原因是8081的端口号被占，我们需要找出这个进程手动消除。

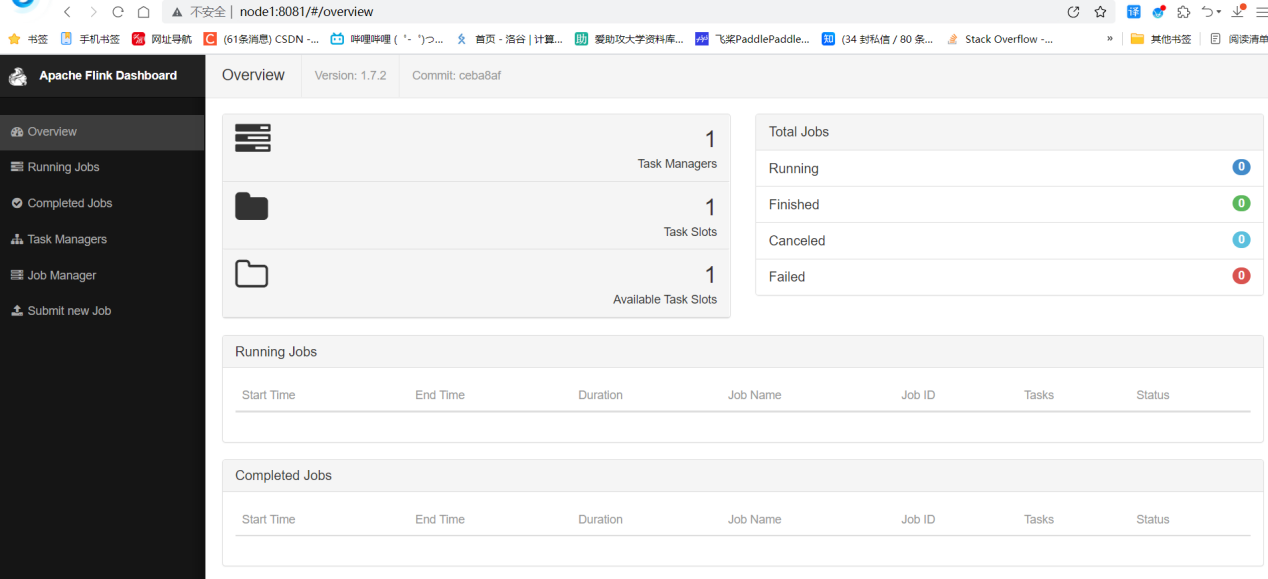
cd log/

netstat -lnp|grep 8081

kill -9 34129

手动消除进程

在网页上输入 node1:8081可以看到 webUI页面

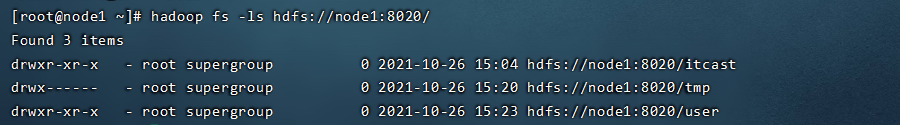
Flink Web UI 页面

**6.2 无法上传数据到HDFS中**

解决方案：

需要先启动hadoop，然后就可以上传数据到HDFS中

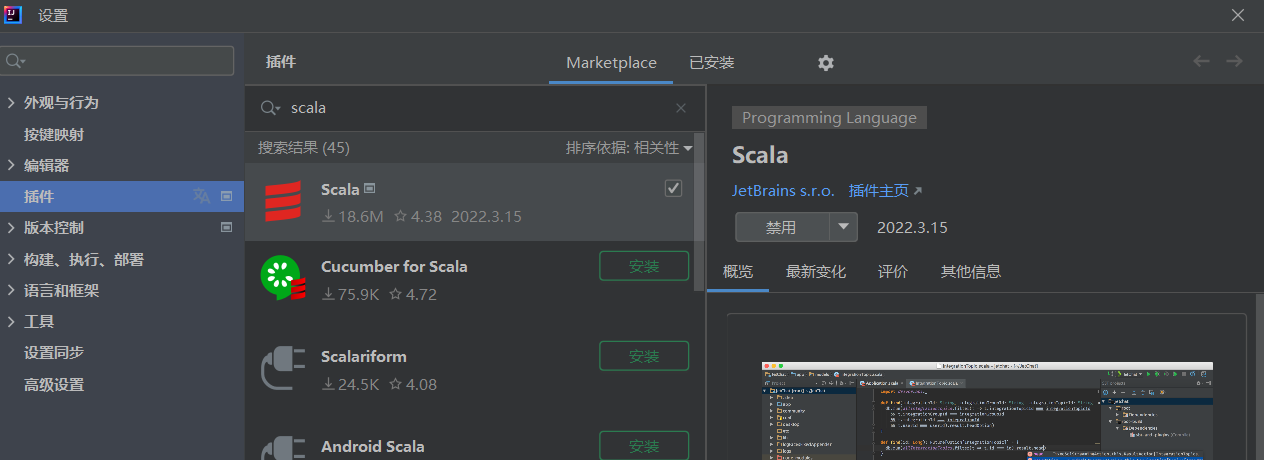
start-all.sh 启动hadoop

HDFS 文件

**6.3 在IDEA 中，首次创建程序运行不成功**

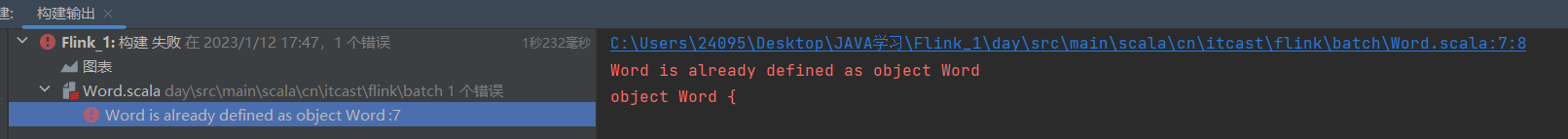
解决方案：

Flink 框架首次创建需要许多依赖，下载需要很长时间，等一会，其次就是检测自己的scala 插件是否下载，如果没有就下载，在设置中插件一栏搜索 scala 点击下载即可。

scala 插件下载

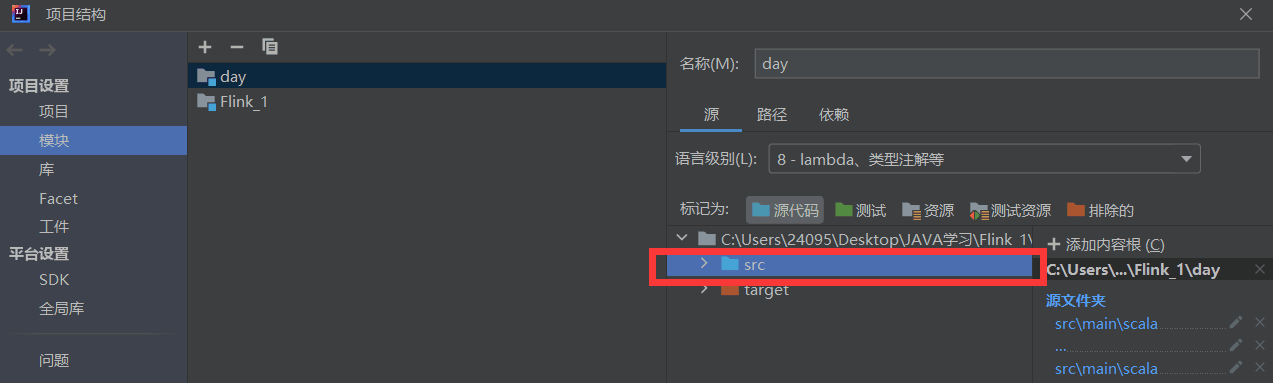
**6.4 程序运行报错**

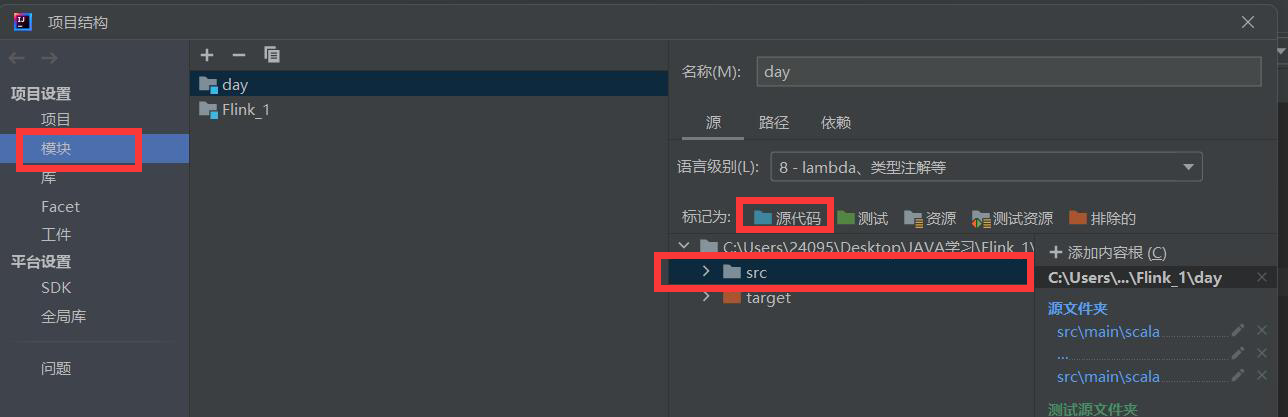
报错，如下图。

报错截图

解决方案：

在项目结构->模块中找到 src 文件夹选中，点击上方源代码，再点击确定。再运行程序就可以成功了。

设置截图

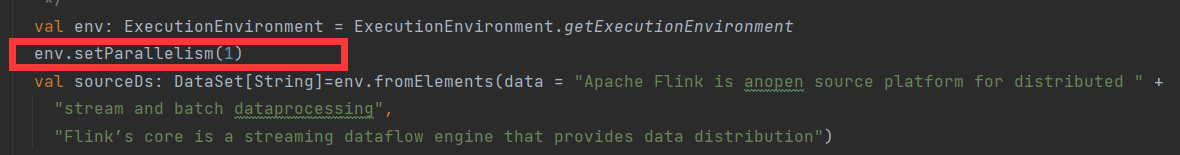
设置截图

**6.5 程序并行度**

解决方案：

程序输出结果保存在当地或者HDFS时，如果不设置程序并行度，其默认并行度是PC机的核数，因此会造成一份输出数据可能有输出文件，解决这一问题就得设置程序并行度，添加下方代码，使用setParallelise() API 设置并行度为1 这样最终输出文件就是一个。

1. env.setParallelism(1)

并行度设置截图

其实还有许多大大小小的问题，由于在实验进行过程中，没有保存，也不能很好的写出来。以上问题只是很小的一部分。

1. **感想与体会**

选择Flink大数据技术介绍及应用 作为此次期末报告是因为之前深入学了Spark技术后。了解到Flink作为第四代大数据技术，当下十分火热，就想着去了解学习一下。不同于Spark学习起来相对容易，Flink的学习对于我来说是一波三折，主要是因为 Spark现在可以使用Python语言进行学习，而Flink只有Java、Scala语言，其中Scala语言我接触太少，就造成了基础知识不足，在学习过程中遇到许多小问题，但很费劲。但整体学习过程中由于之前对Spark的掌握，从Flink中可以看到许多Spark的影子，其思想理解起来不是很难。

遇到了好多问题，其中最头疼的就是环境配置，虽然网上有相关的博客教程，但是实际配置过程中总是出现意想不到的问题，期间一度想过放弃，想换个选题，比如Flink 搭建成功，启动发现TaskManger数目不对，Web UI页面打不开，这找了好久，试了好多方法才解决，原来是之前有进程占用了8081端口，不过最头疼的是配置IDEA环境，pom.xml 文件不知道报了多少次错，解决完后程序运行继续报错，连接不到HDFS等等，这些问题花费了我两三天，期间一度想放弃，最后试了不知道多少个博客的解决方法，最后终于成功了。

总体来说，整个过程虽然很痛苦但确实让我收获了许多，对于大数据技术我总体有了清楚的认识。

1. **参考资料**
2. 《剑指大数据—Flink学习精要》
3. Flink菜鸟教程(一)——从入门到开发 :博客
4. （解决）flink 单机部署，启动后进程没有StandaloneSessionClusterEntrypoint，web无法访问8081端口 ：博客
5. flink实战--开发中常见的错误与问题 ：博客
6. 本机无法访问虚拟机hdfs文件系统 ：博客
7. IDEA中通过sbt创建Scala工程导入依赖报错 ：博客
8. IDEA maven中添加scala依赖 ：博客
9. maven-shade-plugin插件打包异常解决方案 ：博客
10. IDEA中maven的Plugins报红解决方法 ：博客
11. IDEA中搭建flink开发环境，看这一篇就够了，亲测有效 ：博客
12. flink运行job任务时报错 Could not retrieve the execution result ：博客
13. Flink学习 - 9. Checkpoint使用方式 ：博客
14. flink standalone 部署模式且不能使用 hdfs 场景下的各种问题及其应对方案 ：博客