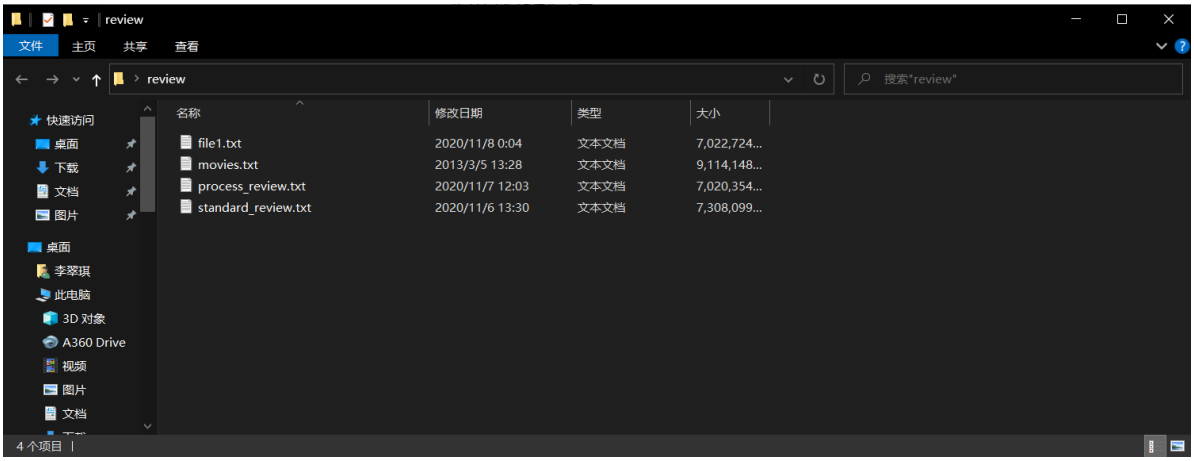
实现流程

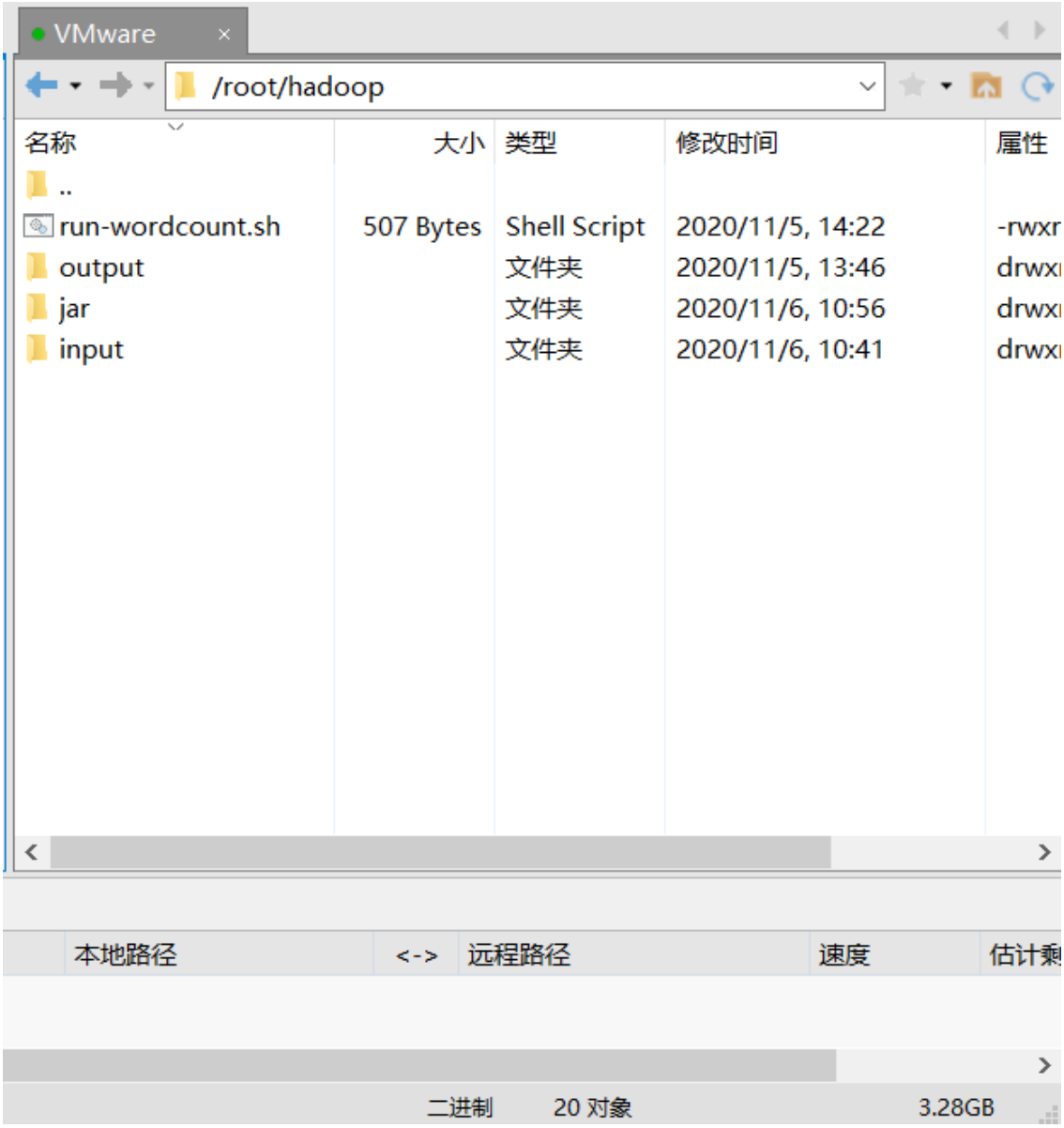
1. 对给的**movies.txt**文本通过自己写的StandardProcessing程序进行预处理:
    - 设置好分隔符,通过非字符进行分隔
    - 将单词全部归为小写
    - 取单词长度大于1的
    - 去掉介词且包含数字的词
    - 用空格分隔,每隔1000个单词换行,重新写入一个文件**standard_review.txt**中
2. 对standard_review文本进行词性还原，同样以空格分割，放入**process_review.txt**文件中
3. 将上述process_review.txt文件进行换行处理得到**file1.txt**
4. 通过运行Hadoop集群实现词频统计(得到的是词频统计的非降序结果)，输出得到文件**output0.txt**
5. 对得到的output0.txt进行处理,如去掉以"zz"开头的,把"_"的替换为""等,得到**output1.txt**文件
6. 再次使用Hadoop集群对output1.txt进行处理,得到按频率降序排序的结果，输出得到文件**output2.txt**中
7. 对结果进行进一步处理,如去掉count小于10的（对于大文本来说，只出现几次的我们认为是评论的单词不符合规则或者是我们没处理好的单纯）、去掉一些专属于html的元素（例如br等标签),得到结果**result**。



配置流程

1. Windows10安装Vmware,并在VMware配置Centos7 linux系统,分配4核CPU,10G内存
2. 在Centos7中下载docker,配置SSH连接,网络连接
3. 下载Hadoop镜像（使用的是Github开源的Hadoop镜像**kiwenlau/hadoop-cluster-docker**

4. 对Hadoop镜像进行配置,启动3个容器，1个master+2个slave,搭建小型分布式集群
5. 通过SSH连接在idea操纵虚拟机命令行,Xftp操作虚拟机文件系统
6. 修改start-container.sh,配置docker容器启动方式。通过docker的目录映射,将虚拟机的文件与docker的Hadoop镜像中的文件相互关联,可以方便的在虚拟机中操纵Hadoop的文件。如下图所示，在windows上通过Xftp操作虚拟机的文件，又因为docker的目录映射，可以直接操作Hadoop内的容器。如**run-wordcount.sh**则是配置启动选项的Shell脚本，**input,output**文件夹存放输入输出的相关数据。jar包内存放**WordCount**程序代码



7. 寻找网上demo，在本地windows环境用idea写好测试用例，打成jar包放入虚拟机中
8. 下载的hadoop镜像基于JDK7的，所以进行相应更改
9. 在run-wordcount.sh中修改输入输出文件位置，需要运行的jar包位置
10. 对WordCount示例代码进行精读，并进行相关的代码注释
11. 先对文本进行一些预处理，以及在windows本地测试处理效果
12. 加入本次数据存储练习需要的词频计算程序逻辑代码
13. 在Hadoop集群中运行得到根据单词词频降序排列的文件
14. 对文本结果进行进一步分析与相应处理优化

词性还原判断规则

> 即要求中的（请考虑什么是不同的单词，例如单复数等均为同一个单词，不同时态也为同一个单词）

启动流程

1. 开启虚拟机,用ip addr得到动态ip地址
2. 用idea ssh连接centos7命令行,xftp连接centos7文件系统
3. 通过idea的maven插件将WordCount程序打成jar包
4. 放入容器内的jar文件夹内
5. `cd hadoop-cluster-docker`
6. `./start-container.sh`
7. `./start-hadoop.sh`
8. `./run-wordcount.sh`

常用命令

```
vmware查看动态路由:
    ip addr

hadoop查看内部的输入输出文件:
    hdfs dfs -cat output/part-r-00000
    hdfs dfs -ls input

hadoop将输出文件拷贝到docker容器卷:
    hdfs dfs -cat output/part-r-00000 > output/output.txt

hadoop控制执行对应的jar:
    hadoop far +jar包名称 +mainClass + 输入路径 + 输出路径

hadoop内部日志位置:
    cd /usr/local/hadoop/logs

查看日志:
    config/mapred-site.xml下配置程序日志输出位置

退出安全模式:
    hadoop dfsadmin -safemode leave
```

遇到的问题

云服务器运行Demo失败

```
Caused by: org.apache.spark.SparkException: Job aborted due to stage failure:
Task 2 in stage 3.0 failed 4 times, most recent failure: Lost task 2.3 in stage
3.0 (TID 23, ip-xxx-xxx-xx-xxx.compute.internal, executor 4): ExecutorLostFailure
(executor 4 exited caused by one of the running tasks) Reason: Container marked
as failed: container_1516900607498_6585_01_000008 on host: ip-xxx-xxx-xx-
xxx.compute.internal. Exit status: 137. Diagnostics: Container killed on request.
Exit code is 137
```

在阿里云轻量级应用服务器上，跑集群的小demo异常退出，后来发现是内存不够的原因（云服务器只有2g的内存），于是换到windows的虚拟机上

Hadoop集群连接失败

```
java.net.ConnectException: Call From xxx to localhost:8020 failed on connection
exception: java.net.ConnectException: Connection refused
```

关闭Centos7防火墙

内存溢出

```
Exception in thread "main" java.lang.OutOfMemoryError: GC overhead limit
exceeded
```

在执行NLP进行单词词性还原时,遇到堆空间溢出的情况。我们通过减少训练量（每次只训练500个单词），增大给虚拟机内存（最后是11.3GB）的方式解决

Jar包运行失败

使用NLP进行单词词性还原的代码打成jar包后放入Hadoop集群中跑不了，查阅后知道打的jar包必须用含有依赖的。于是把含有依赖的继续放入，报错 `java.util.zip.ZipException:invalid distance distance too far back`，需要检查jar包下载相关jar包时未下载完全，导致服务器进行加载解压时出错。

NLP处理问题

我们尝试了三种情况

NLP处理单词词性还原效果很好，但是运行速度极慢，一晚上只能跑出34MB的结果。于是我们弃用了NLP，通过英语本身的规则进行了粗略的词性还原，速度很快，但是效果一般。

我们尝试了github上的字典，希望通过字典尝试一下，效果一般（后反思发现，应该先将字典读入内存，效果应该快很多）。

最后，我们尝试了NLTK的词形还原方法，能将动词的三单，时态还原，名词的单复数还原和形容词还原等等。效果显著，有效还原了文件中的词性。

磁盘空间不足

在实际运行8g的文件时,发现报以下错,原因是磁盘空间不足

```
Please add or free up more resources then turn off safe mode manually.
```

于是使用`df -h`查看hadoop内的磁盘,发现只有20g,又用df在centos下查看,分析觉得是给docker分配的磁盘配额太小导致的