

一、配置Hadoop的IDEA开发环境

一开始拿到题目还是很糊涂的，得先从实践入手摸清组件之间的关系，尤其是ide和hadoop集群之间的关系。我们知道Hadoop可以运行在三种模式下：

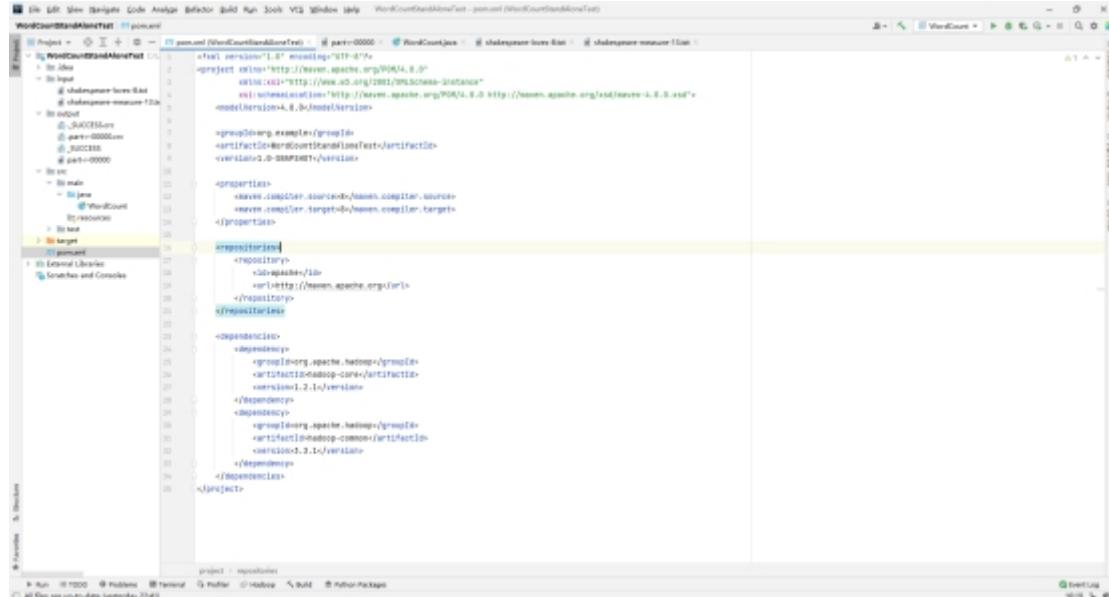
1. 单机模式
2. 伪分布模式
3. 集群模式

这篇教程[Hadoop: IntelliJ结合Maven本地运行和调试MapReduce程序\(无需搭载Hadoop和HDFS环境\) - Penguin \(polarxiong.com\)](#)表示，运行和调试MapReduce程序只需要有相应的Hadoop依赖包就行，可以完全当成一个普通的JAVA程序，不过其主要不足就在于没有Hadoop的整个管理控制系统，如JobTracker面板，而只是用来运行和调试程序；而其优点就在于开发调试方便，*编写的程序通常不需要修改即可在真实的分布式Hadoop集群下运行*。当然有这么方便的事情肯定会想直接尝试，不过还是有一丝犹豫的，毕竟完成代码测试后就是在docker上的HADOOP集群上跑了，那我之前在WSL上搭的单机和伪分布式HADOOP就毫无价值了吗？（后面还是用上了）

文中stop文件夹包含两个停用文件，在input同级目录下。src中，主程序是WordCount2.java，实际上就是WordCount3.0，是基于官网教程的WordCount2.0改的，就顺手没有修改文件名。MiscUtils.java和MyOutPutFormat.java分别是协助实现WordCount3.0的新写方法和重写类，剩下的WordCount.java和WordCount2_sample.java的作用主要是来存目，保留官网代码和新写的比对学习，并不参与最终运行。

1.Idea单机模式测试WordCount1&2

之前有一定的项目基础，maven等环境是配置好的，直接复制好pom.xml和教程样例代码中的WordCount类（WordCount1.0和ppt中main函数略有差异），先确保能跑通：

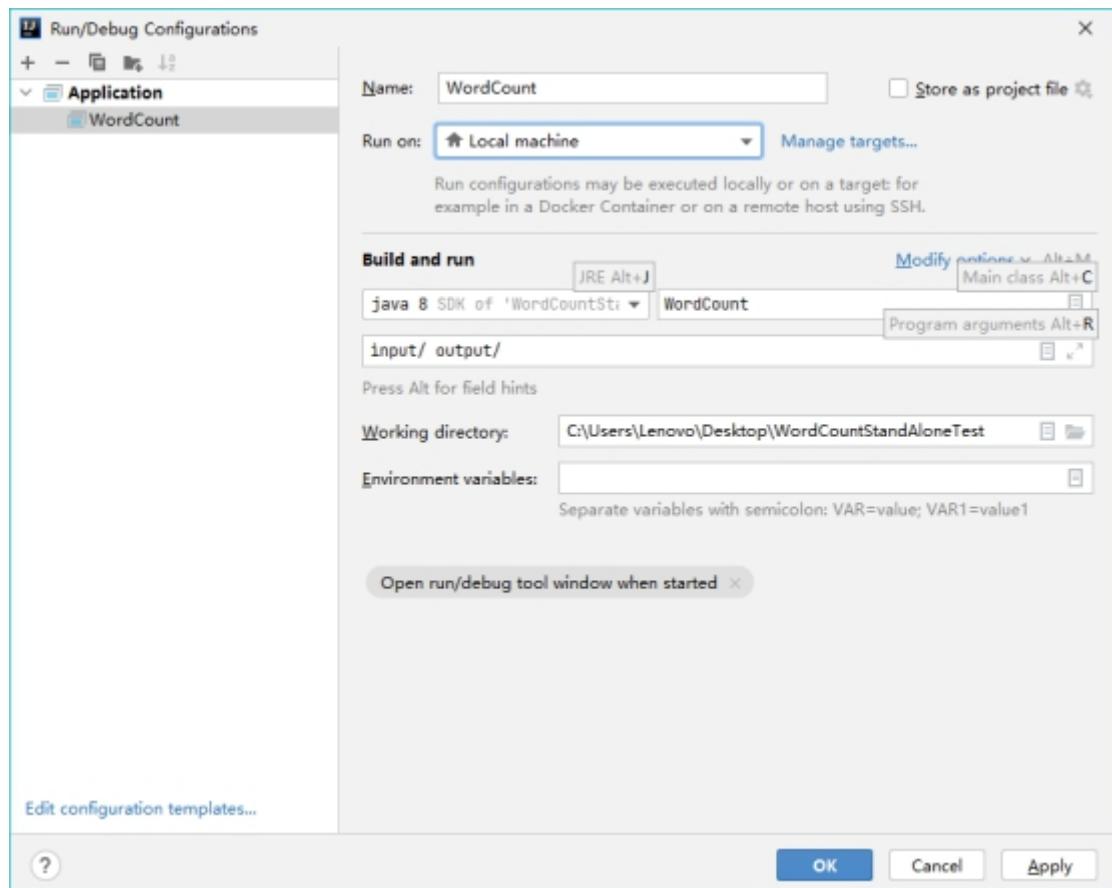


配置pom文件

The screenshot shows the IntelliJ IDEA interface with the WordCount.java file open. The code implements a MapReduce job to count words from input files. Below the code, the 'Run' tool window displays a command-line log with several errors related to permission issues on the Windows file system. The log includes entries like 'Failed to set permissions of path: tmp/hadoop-Lenovo/tmp/staging/lenovo/wordcount/119', 'staging to 0770', and multiple 'java.io.IOException: Failed to set permissions of path: tmp/hadoop-Lenovo/tmp/staging/lenovo/wordcount/119' messages.

复制样例代码

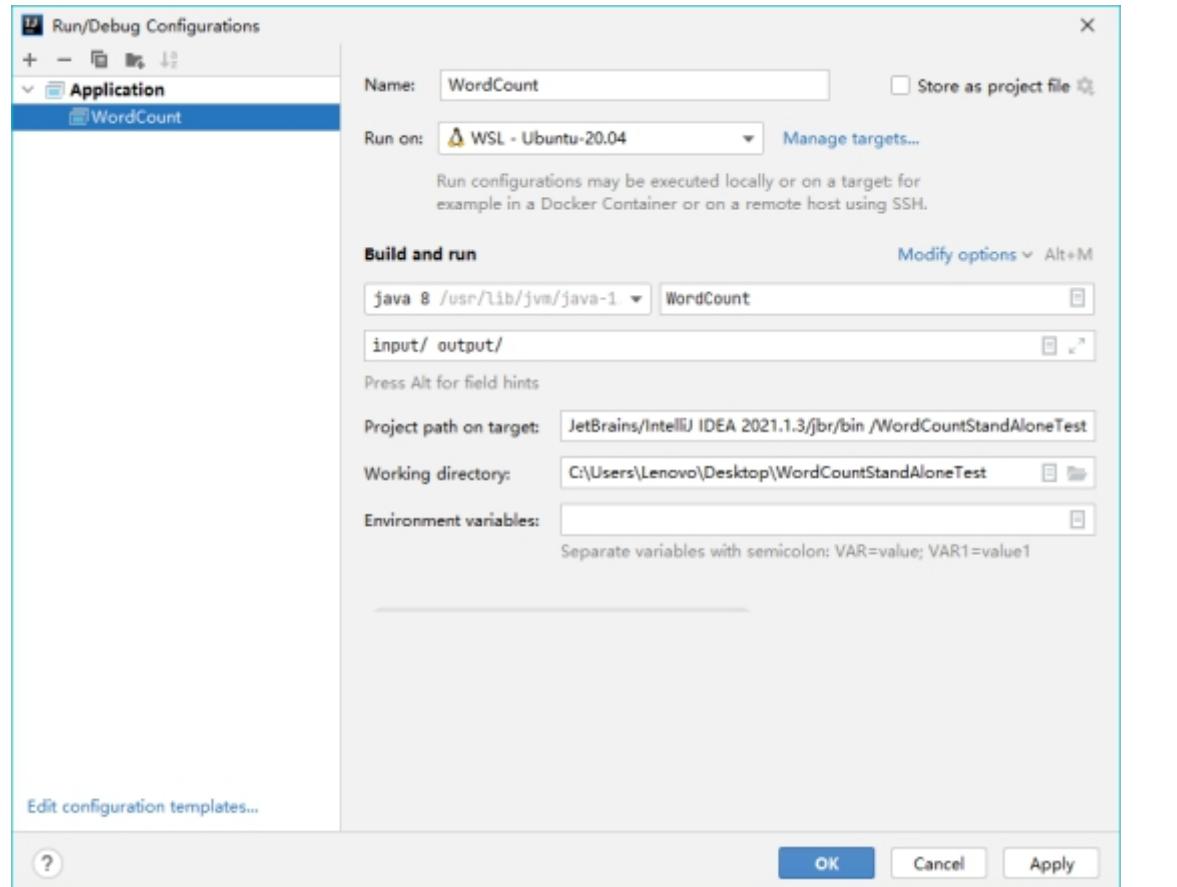
接着是准备好样例代码需要的文件。在WordCount下（src同级目录）新建一个文件夹input，复制了两个莎士比亚文集的文本文件作为示例添加到input中。然后配置运行参数。在IntelliJ菜单栏中选择Run->Edit Configurations，在弹出来的对话框中点击+，新建一个Application配置。配置Main class为WordCount，Program arguments为input/ output/，即输入路径为刚才创建的input文件夹，输出为output。这些内容将在main函数中 Configuration conf = new Configuration(); //取得系统的参数 这一行被读取。

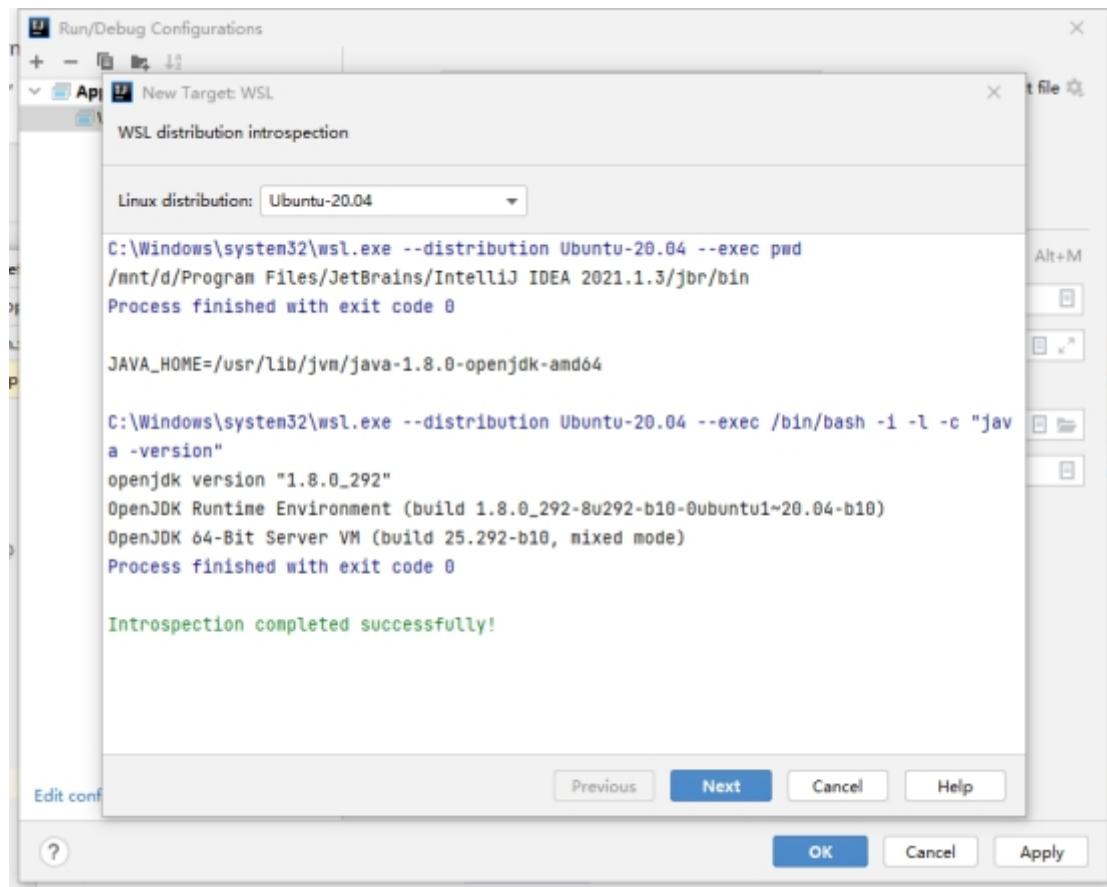


配置运行参数

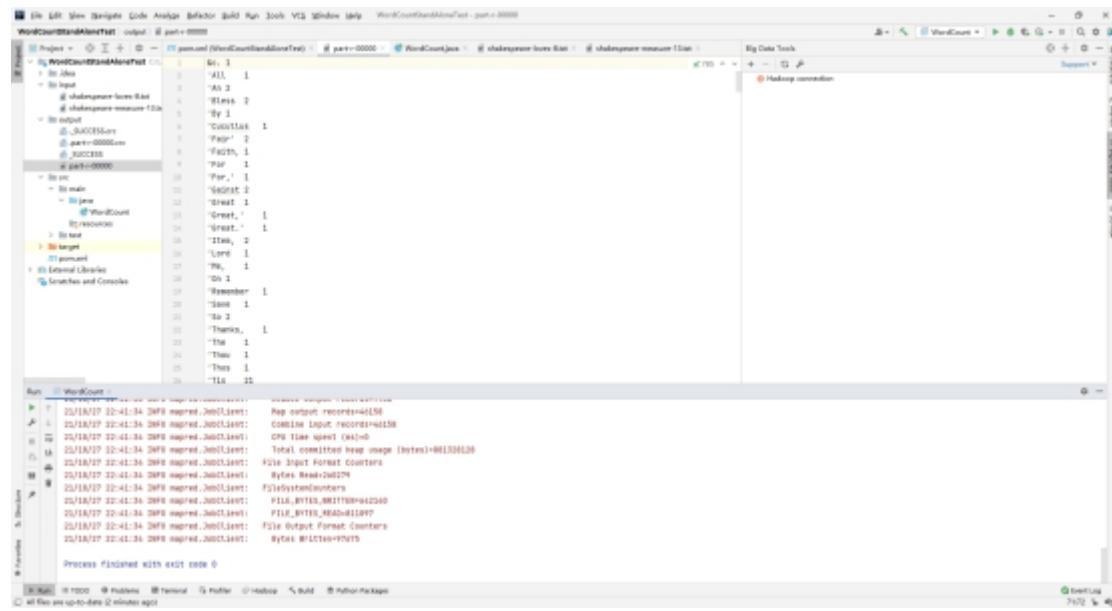
开始运行后，会看到和上面复制样例代码那张图一样的报错，failed to set permissions of path，教程中提出这是Windows下的权限问题，因为当前用户没有权限来设置路径权限（Linux无此问题）。一个解决方法是给hadoop打补丁，参考[Failed to set permissions of path: tmp](#)，因为这里使用的Maven，此方法不太适合。另一个方法是将当前用户设置为超级管理员（“计算机管理”，“本地用户和组”）

中设置），或以超级管理员登录运行此程序。作者提供的两种方法我都有尝试，第一种的补丁已经年久失修找不到链接了，相应的有其他回答给出了修改jar包，改掉文件权限相关内容，重新打包替代充当补丁的作用，不过确实不是针对maven管理的，也不敢随便修改基础文件，也有建议将依赖中的hadoop-core版本改为0.20.2，但该版本并不支持Job.getInstance（后面发现hadoop-core是MRv1的产物，不应列在pom依赖中）；第二种方法则更加迷糊，不好判定到底是不是超级管理员状态。于是我在jetbrain官网https://www.jetbrains.com/help/idea/how-to-use-wsl-development-environment-in-product.html#local_project发现idea可以使用本地JDK在Windows操作系统上本地创建或打开项目，然后使用运行目标在WSL中运行编译代码。也就是说，得益于wsl的Linux兼容内核接口，我能够利用windows上的idea开发程序，但是在wsl的环境上运行程序，这是符合我的在单机上先跑程序测试的预期的，也回收了一些沉没成本，只需在配置运行参数中Run on选项本地运行改为在WSL上运行。





有些中间过程，都点继续就行



发现成功运行

意味着已经可以通过idea连携wsl单机运行hadoop程序了，方便了后续测试好再放到集群上运行。

在测试官网的MapReduce2.0时，我发现getCacheFiles和addCacheFile方法无论怎样import都无法resolve，经过百般尝试我把mapreduce、yarn、hadoop所有依赖都在maven里塞一遍仍然不能解决，结果发现是多写了一个hadoop-core，这是适用于MRv1的依赖，没有MRv2的方法，应该删除。

1 Answer

Active	Oldest	Votes
--------	--------	-------

 I was able to reproduce what you have tried and did not encountered any kind of problem. I am using MRv2.

0

 From the pom, it seems you are using MR2 (which is the latest). So remove the MR1 dependency(hadoop-core 2.6.0-mr1-cdh5.10.1). The hadoop core jar is not required for MRv2.

 The code I have used. This should work if you are using MRv2.



```
import java.io.IOException;
import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;

public class App
{
    public static void main( String[] args ) throws IOException, InterruptedException
    {
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf);

        job.addCacheFile(new Path("").toUri());
    }
}
```

But if you want to use MRv1, then we need to change the POM, then we need to use

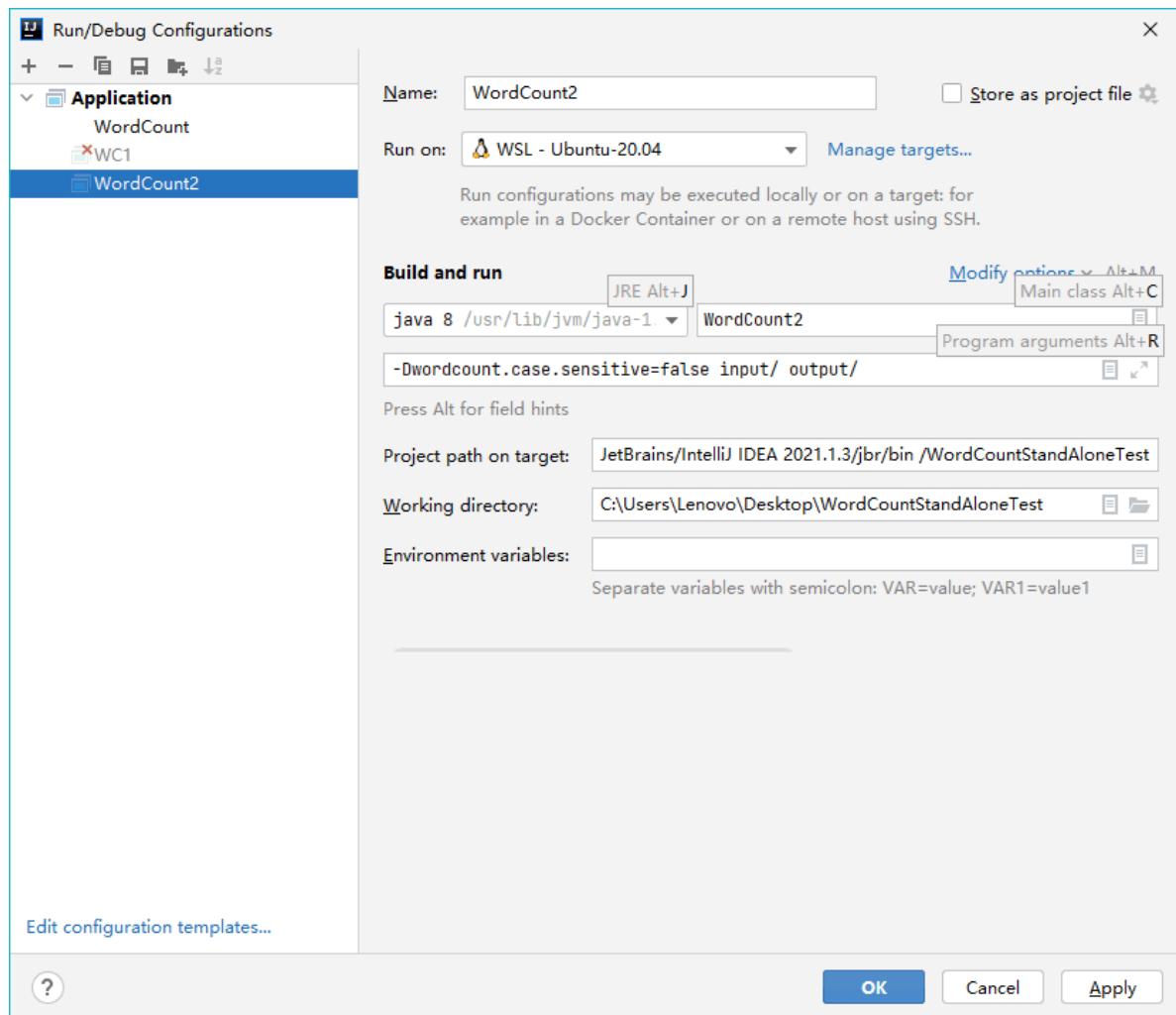
```
Configuration conf = new Configuration();
DistributedCache.addFileToClassPath(new Path("<path to file>"), conf);
Job job = new Job(conf);
```

[Share](#) [Improve this answer](#) [Follow](#)

[edited Apr 12 '17 at 8:43](#)

[answered Apr 12 '17 at 8:32](#)

解释来源



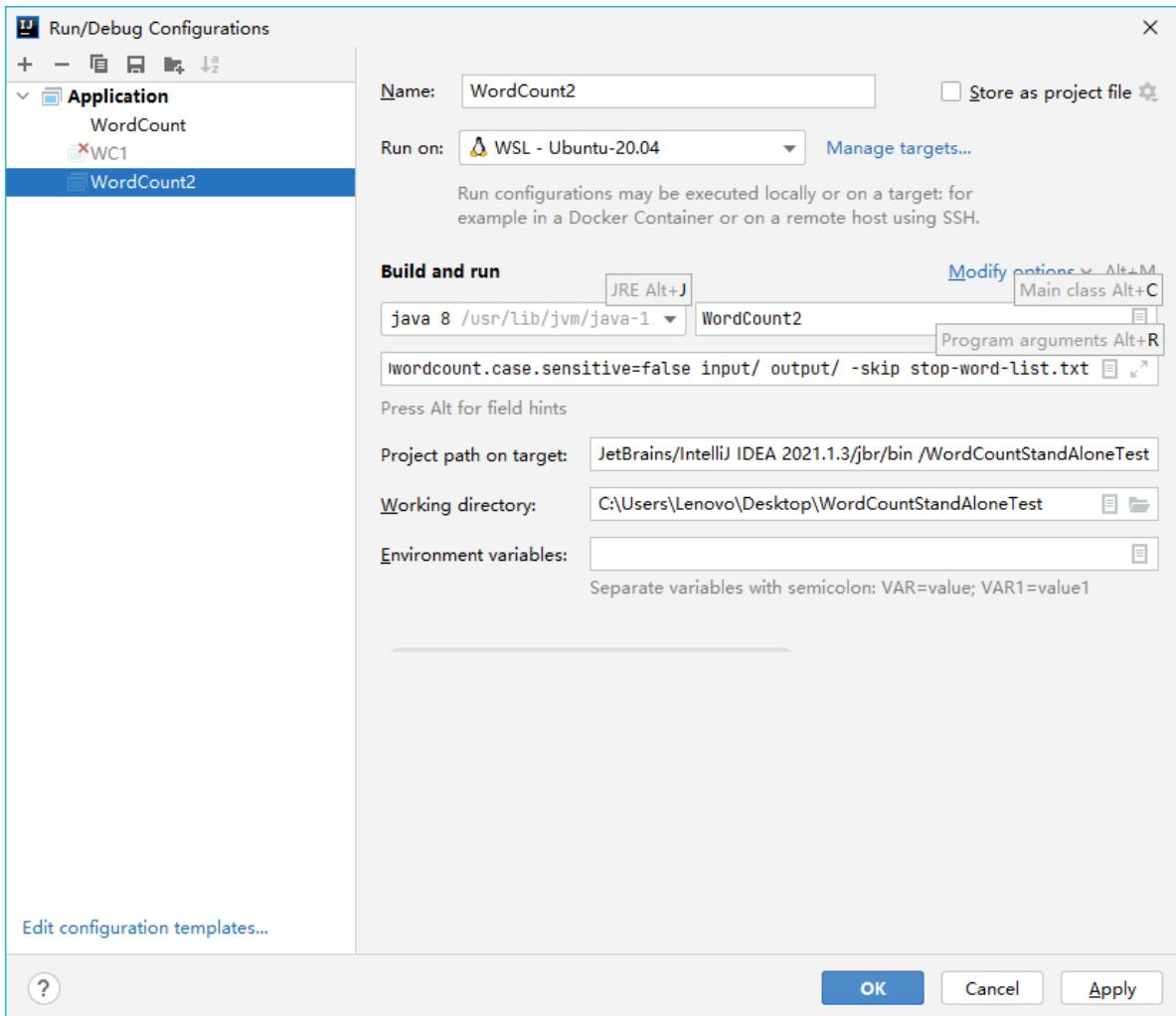
针对WordCount2.0增加了大小写敏感的传参

```

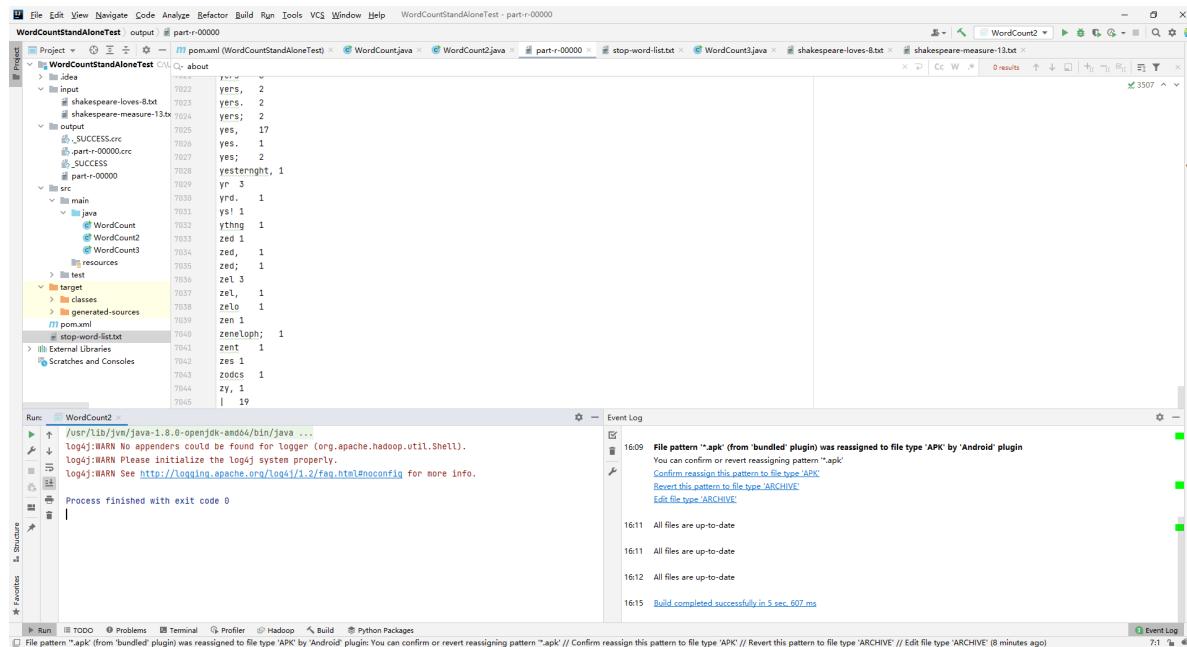
1 &c. 1
2 'all' 2
3 'an' 3
4 'ay.' 1
5 'bless' 2
6 'e' 1
7 'cuckoldus' 1
8 'dansel.' 1
9 'daughter-beamed' 1
10 'fair' 2
11 'feith,' 1
12 'for' 2
13 'for,' 1
14 'fore' 1
15 'gainst' 6
16 'great' 1
17 'great,' 1
18 'greed' 1
19 'her' 1
20 'hobby-horse?' 1
21 'item,' 2
22 'long' 1
23 'longs,' 1
24 'lord' 1
25 'me...' 1

```

调整参数后成功运行MapReduce2.0，过程中也提醒了我利用已经实现的传参可能减少不少代码量



比如停用词就可以通过传参-skip解决



中间因为是电脑重新启动，导致了依赖出现了前后不一致的情况：Exception in thread "main" java.lang.NoSuchMethodError:
 org.apache.hadoop.security.proto.SecurityProtos.getDescriptor()Lcom/google/protobuf/Descriptor
 rs\$FileDescriptor;清理一下缓存invalid cache重新刷新后恢复，可以发现确实起效跳过了停用词。目前已经解决不少问题了。

二.完善WordCount3.0

The screenshot shows the IntelliJ IDEA interface with the WordCount2.java file open. The code implements a Hadoop job to process input files like Shakespeare-loves-8.txt and Shakespeare-measure-13.txt, outputting results to part-r-00000 through part-r-00009. It handles command-line arguments for skipping patterns and setting up configuration. The Event Log panel shows build logs from 16:15 to 16:51.

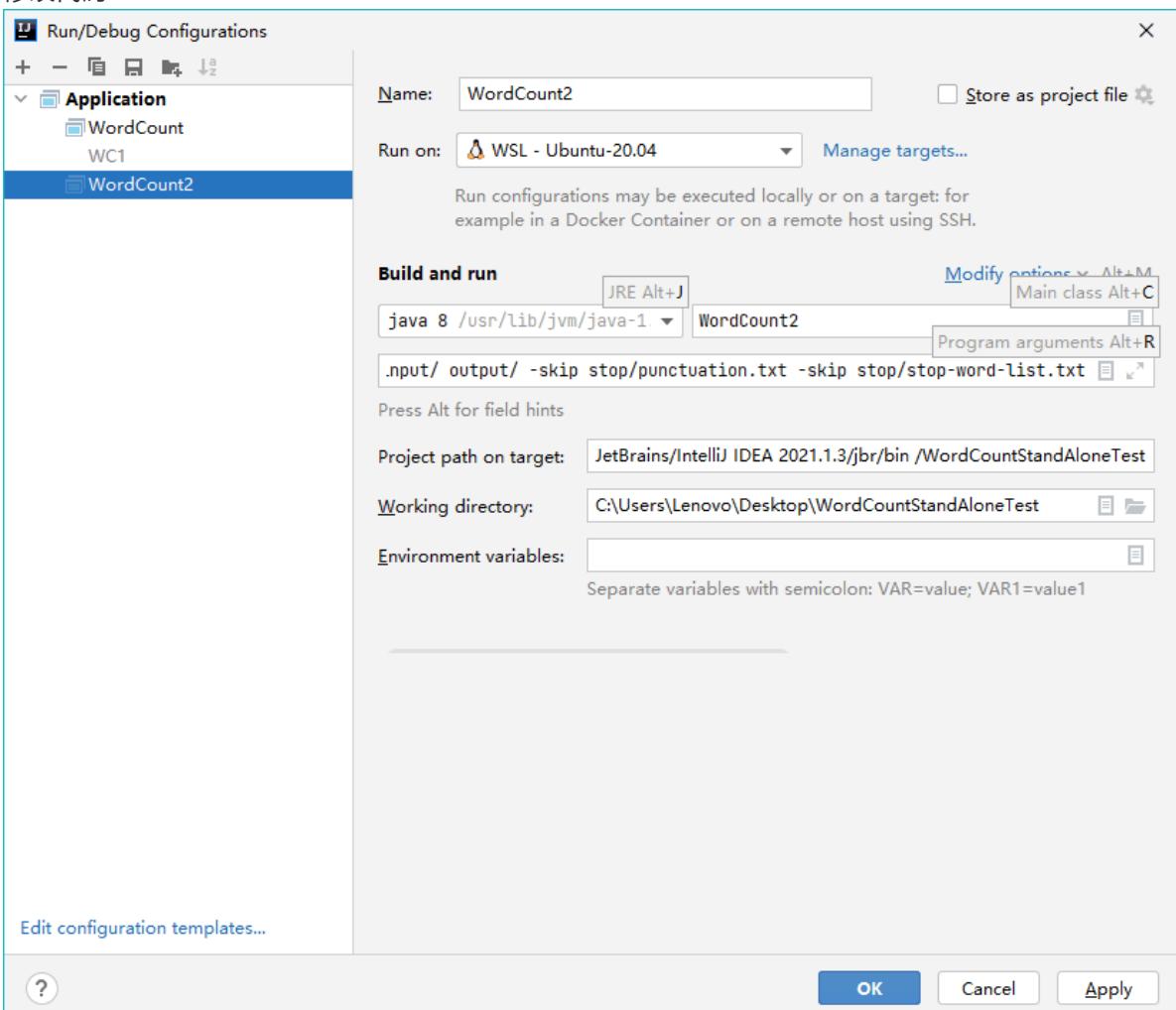
```

    ...
    job.setJarByClass(WordCount2.class);
    job.setMapperClass(TokenMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    List<String> otherArgs = new ArrayList<String>();
    for (int i=0; i < remainingArgs.length; ++i) {
        if ("-skip".equals(remainingArgs[i])) {
            job.addCacheFile(new Path(remainingArgs[i+1]).toURI());
            job.getConfiguration().setBoolean("wordcount.skip.patterns", true);
        } else {
            otherArgs.add(remainingArgs[i]);
        }
    }
    FileInputFormat.addInputPath(job, new Path(otherArgs.get(0)));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs.get(1)));
    System.exit(job.waitForCompletion(verbose: true) ? 0 : 1);
}

```

修改代码



修改命令行和读取命令行参数，使得能够同时读取两个stop-word-list和punctuation停用文件样板

WordCount3.0需要skip两个停用词文件，这就需要重新调整对参数的数量限制和读取方式，体现在java代码上。

针对单词长度的过滤，不外乎两种，一种是map时过滤，一种是reduce时不输出，显然前者减少了传输开销和reducer的负担，更划算。

```

@Override
public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
    String line = (caseSensitive) ? value.toString() : value.toString().toLowerCase();
    for (String pattern : patternsToSkip) {
        line = line.replaceAll(pattern, replacement: "");
    }
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        int length = word.toString().length();
        if(length < 3) continue;

        context.write(word, one);
        Counter counter = context.getCounter(CountersEnum.class.getName()),
        CountersEnum.INPUT_WORDS.toString());
        counter.increment( 1 );
    }
}

```

修改map提前过滤掉短单词

```

@Override
public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
    String line = (caseSensitive) ? value.toString() : value.toString().toLowerCase();
    for (String pattern : patternsToSkip) {
        line = line.replaceAll(pattern, replacement: "");
    }
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        int length = word.toString().length();
        if(length < 3) continue;

        context.write(word, one);
        Counter counter = context.getCounter(CountersEnum.class.getName()),
        CountersEnum.INPUT_WORDS.toString());
        counter.increment( 1 );
    }
}

```

效果拔群

数字也是如此。之前为了方便只用了两个文本试跑，对所有文本的跑一遍，发现大概只有1609这样的单独数字，没有和英文混杂的。一方面我们使用正则表达式，确保正负数整数小数都能被筛到，`-?[0-9]+(\.[0-9]+)?`即可；或者更粗暴。

```

public static boolean isNumeric(String str) {
    try {
        bigstr = new BigInteger(str).toString();
    } catch (Exception e) {
        return false; // 表明包含非数字。
    }
    return true;
}

@Override
public void map(Object key, Text value, Context context)
    throws IOException, InterruptedException {
    String line = (caseSensitive) ?
        value.toString() : value.toString().toLowerCase();
    for (String pattern : patternsToSkip) {
        line = line.replaceAll(pattern, replacement: "");
    }
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        int length = word.toString().length();
        if (length < 3) continue;
        if (!isNumeric(word.toString())) continue;
    }
}

```

Run: WordCount2

```

at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:164)
at org.apache.hadoop.mapreduce.JobSubmitter.checkSpecs(JobSubmitter.java:277)
at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:163)
at org.apache.hadoop.mapreduce.Job$1.run(Job.java:1571)
at org.apache.hadoop.mapreduce.Job$1.run(Job.java:1569) <1 internal line>
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1878)
at org.apache.hadoop.mapreduce.Job.submit(Job.java:1569)
at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:1589)
at WordCount2.main(WordCount2.java:147)

```

Process finished with exit code 1

Run | TODO | Problems | Terminal | Profiler | Hadoop | Build | Python Packages

Build completed successfully in 10 sec, 940 ms (2 minutes ago)

Event Log 19:1 7:40

直接尝试能不能换成BigDecimal类型

```

public static boolean isNumeric(String str) {
    try {
        bigstr = new BigInteger(str).toString();
    } catch (Exception e) {
        return false; // 表明包含非数字。
    }
    return true;
}

@Override
public void map(Object key, Text value, Context context)
    throws IOException, InterruptedException {
    String line = (caseSensitive) ?
        value.toString() : value.toString().toLowerCase();
    for (String pattern : patternsToSkip) {
        line = line.replaceAll(pattern, replacement: "");
    }
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        int length = word.toString().length();
        if (length < 3) continue;
        if (!isNumeric(word.toString())) continue;
    }
}

```

Run: WordCount2

```

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
log4j:WARN No appenders could be found for Logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system property.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

```

Process finished with exit code 0

Run | TODO | Problems | Terminal | Profiler | Hadoop | Build | Python Packages

Build completed successfully in 6 sec, 551 ms (2 minutes ago)

Event Log 7:1 7:40

奏效

对数字处理的想法参考了<https://blog.csdn.net/u013066244/article/details/53197756>。

接下来是输出数量的过滤，只要Top100。MapReduce的核心过程并不能解决输出数量的问题，让数据交给reducer，不急着写入文件，通过重写reducer下的cleanup方法，先写入hashmap，再排序筛选写入文件。这里的排序正常来说也可以通过串接第二个mapreduce程序来完成，这里省事直接另写一个类对作为中间存储的HashMap进行排序<http://andreaiacono.blogspot.com/2014/03/mapreduce-for-top-n-items.html>

```

public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    private Map<Text,IntWritable> countMap = new HashMap<>();

    public void reduce(Text key, Iterable<IntWritable> values,
                      Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        countMap.put(new Text(key),new IntWritable(sum));
    }

    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        Map<Text,IntWritable> sortedMap = MiscUtils.sortByValues(countMap);

        int counter = 0;
        for (Text key : sortedMap.keySet()) {
            if (counter++ == 100) {
                break;
            }
            context.write(key, sortedMap.get(key));
        }
    }
}

```

让reduce不急着输出，改写它的cleanup方法

Word	Count
wtih	7906
thou	5361
hve	5712
wilt	4910
thy	3974
shil	3500
kng	3022
enter	2254
let	2134
love	2097
lke	1879
mke	1647
gve	1594
frst	1382
set	1287
tke	1219
spek	1163
ldy	1102
mter	1065
duke	987
ext	971
send	929
ext	924
self	917
nght	884
queen	882
leve	879
look	824
ife	822

Word	Count
wtih	7906
thou	5361
hve	5712
wilt	4910
thy	3974
shil	3500
kng	3022
enter	2254
let	2134
love	2097
lke	1879
mke	1647
gve	1594
frst	1382
set	1287
tke	1219
spek	1163
ldy	1102
mter	1065
duke	987
ext	971
send	929
ext	924
self	917
nght	884
queen	882
leve	879
look	824
ife	822

```

public void reduce(Text key, Iterable<IntWritable> values,
                  Context context)
  throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    countMap.put(new Text(key), new IntWritable(sum));
  }

@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
  Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(countMap);

  int counter = 0;
  for (Text key : sortedMap.keySet()) {
    String skey = counter + "-" + key.toString();
    if (counter++ == 100) {
      break;
    }
    context.write(new Text(skey), sortedMap.get(key));
  }
}

public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  GenericOptionsParser optionParser = new GenericOptionsParser(conf, args);
}

```

Process finished with exit code 0

Build completed successfully in 4 sec. 159 ms (2 minutes ago)

IntelliJ IDEA 2021.2.3 available
Update...

顺便，针对输出格式，可以直接在cleanup中修改键值构造新字符串再转换为Text传入Context，中间的分隔符如果不这样也可以在运行参数中输入-

Dmapreduce.output.textoutputformat.separator=","，效果一致。同理，稍事修改可以通过传参确定top k的k具体数值，这里没有实现。

仔细考虑怎样避开mapreduce总是在文件夹下运行所有文件，更灵活地对数据文件进行操作，这里我不再在configuration中设置参数，直接拎到新的main函数中遍历调用，旧的main设置为driver接受参数。每读一个文件做一次MR，循环结束后再做一次总的，结果重命名后写入各自文件夹。重命名参考[MapReduce修改输出的文件名 - 哺哺擦 - 博客园\(cnblogs.com\)](#)，自定义TextOutPutFormat重写setOutputName方法实现。

```

MyOutputFormat.setOutputName(job, name + "Collection-of-Shakespeare");
return job.waitForCompletion( verbose: true ) @: 1;

public static void main(String[] args) throws Exception {
  String path = "/mnt/c/Users/lenovo/Desktop/WordCountStandAloneTest/input";
  File file = new File(path); //从命令行参数读取文件和目录，放在File数组中
  File[] fs = file.listFiles(); //遍历所有文件
  for(File f:fs){ //遍历所有文件
    int l = f.toString().toCharArray().split("-").length;
    String[] para = new String[l-1];
    //mapreduce输出textoutputformat.separator ","
    -Dmapreduce.output.textoutputformat.separator ",";
    args[0] + f.toString().toCharArray().split("-")[l-1], //将第一个参数作为key
    args[1]+f.toString().toCharArray().split("-")[l-1], //将第二个参数作为value
    "-skip", "stop/punctuation.txt", "-skip", "stop/stop-word-list.txt";
    driver(para);
  }
  String[] para = new String[1]{"-Dmapreduce.case.sensitive=false",
    "-Dmapreduce.output.textoutputformat.separator ","};
  args[0], args[1]+summary, "-skip", "stop/punctuation.txt", "-skip", "stop/stop-word-list.txt";
  driver(para);
}

```

Process finished with exit code 0

Build completed successfully in 3 sec. 803 ms (yesterday 2256)

```

1:eth,394
2:hve,272
3:rsl,266
4:nen,196
5:ths,187
6:marc,145
7:fst,139
8:scn,138
9:vall,128
10:shll,121
11:brut,111
12:mch,101
13:thou,97
14:enter,97
15:ufd,96
16:thy,91
17:send,90
18:citizen,85
19:people,74
20:let,72
21:clown,68
22:bble,67
23:like,67
24:servgn,66
25:mme,62
26:spex,54
27:que,53

```

Process finished with exit code 0

可见实现了单个文件和的词频按格式统计输出，要求的功能都已经实现了。

三、在docker集群上跑WordCount3.0

到了在集群上跑，由于代码也几乎完成了，并没有连接docker和idea的必要。当然一开始我也有连接集群运行代码的想法，按照官网教程下载了Big Data Tools、Scala和python插件，设置server type, file systems之类，也没摸清帮助也不大，还是准备好集群打好jar包跑就行。

```

PS C:\Users\Lenovo> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
2645e9397c95 hadoop_cluster_neo "/bin/bash" 5 minutes ago Up 5 minutes
f0eff628af5d hadoop_cluster_neo "/bin/bash" 6 minutes ago Up 6 minutes
c726100c8082 hadoop_cluster_neo "/bin/bash" 6 minutes ago Up 6 minutes 0.0.0.0:8088->8088/tcp, :::8088->8088/tcp, 0.0.0.0:9870->9870/tcp, ::9870->9870/tcp h01
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword c726100c8082:/usr/local/hadoop/wordcount3.0
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword f0eff628af5d :/usr/local/hadoop/wordcount3.0
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-|
       docker cp [OPTIONS] SRC_PATH|-| CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword f0eff628af5d:/usr/local/hadoop/wordcount3.0
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword 2645e9397c95 :/usr/local/hadoop/wordcount3.0
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-|
       docker cp [OPTIONS] SRC_PATH|-| CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword 2645e9397c95:/usr/local/hadoop/wordcount3.0
PS C:\Users\Lenovo>

```

宿主机向容器传输文件

```

root@h01:/usr/local/hadoop# sbin/start-dfs.sh
Starting namenodes on [h01]
Starting datanodes
Starting secondary namenodes [h01]
root@h01:/usr/local/hadoop# sbin/start-yarn.sh
Starting resourcemanager
Starting nodemanagers
root@h01:/usr/local/hadoop# jps
3008 Jps
2130 DataNode
2661 NodeManager
2536 ResourceManager
2283 SecondaryNameNode
1979 NameNode
root@h01:/usr/local/hadoop# pwd
/usr/local/hadoop
root@h01:/usr/local/hadoop# dir
LICENSE-binary NOTICE-binary README.txt etc lib licenses-binary sbin wordcount3.0
LICENSE.txt NOTICE.txt bin include libexec logs share
root@h01:/usr/local/hadoop# cd wordcount3.0
root@h01:/usr/local/hadoop/wordcount3.0# dir
input punctuation.txt stop-word-list.txt
root@h01:/usr/local/hadoop/wordcount3.0# cd ..
root@h01:/usr/local/hadoop#

```

这里发现了一个很严重的问题，在实验二中的docker集群配置中，由于改造镜像时的配置文件全都照抄了伪分布式的，当时报告以为完成了实际上并没有起到集群的效果，当时对整体架构不太清楚也没能从网页上只显示有一个活跃节点等显著变量中察觉。现在已经按照分布式的配置文件重新导出了镜像，包括网页与运行时的提示已不再显示localhost而是h01，也显示三个节点可用，之前疏忽了。

```

journalnode      run the DFS journalnode
mover           run a utility to move block replicas across storage types
namenode        run the DFS namenode
nfs3            run an NFS version 3 gateway
portmap         run a portmap service
secondarynamenode run the DFS secondary namenode
sps             run external storagepolicysatisfier
zkfc            run the ZK Failover Controller daemon

SUBCOMMAND may print help when invoked w/o parameters or with -h.
root@h01:/usr/local/hadoop# bin/hdfs dfs -ls
ls: `.'': No such file or directory
root@h01:/usr/local/hadoop# bin/hdfs dfs -ls /
root@h01:/usr/local/hadoop# bin/hdfs dfs -mkdir /user
root@h01:/usr/local/hadoop# bin/hdfs dfs -mkdir /user/flospro
root@h01:/usr/local/hadoop# bin/hdfs dfs -put ./wordcount3.0 wordcount3.0
put: `wordcount3.0': No such file or directory: `hdfs://h01:9000/user/root/wordcount3.0'
root@h01:/usr/local/hadoop# bin/hdfs dfs -mkdir /user/flospro/WC3
root@h01:/usr/local/hadoop# bin/hdfs dfs -put ./wordcount3.0 WC3
put: `WC3': No such file or directory: `hdfs://h01:9000/user/root/WC3'
root@h01:/usr/local/hadoop# bin/hdfs dfs -put ./wordcount3.0 /user/flospro/WC3
root@h01:/usr/local/hadoop# bin/hdfs dfs -ls /user/flospro/WC3
Found 1 items
drwxr-xr-x - root supergroup          0 2021-10-31 00:43 /user/flospro/WC3/wordcount3.0
root@h01:/usr/local/hadoop# bin/hdfs dfs -ls /user/flospro/WC3/wordcount3.0
Found 3 items
drwxr-xr-x - root supergroup          0 2021-10-31 00:43 /user/flospro/WC3/wordcount3.0/input
-rw-r--r--  3 root supergroup         98 2021-10-31 00:43 /user/flospro/WC3/wordcount3.0/punctuation.txt
-rw-r--r--  3 root supergroup        2231 2021-10-31 00:43 /user/flospro/WC3/wordcount3.0/stop-word-list.txt
root@h01:/usr/local/hadoop#

```

向hdfs中放入文件

```

try{
    MyOutPutFormat.setOutputName(job, otherArgs.get(0).split(regex:"/\\\"[1]"));
} catch (Exception e) {
    MyOutPutFormat.setOutputName(job, name: "Collection-of-Shakespeare");
}
return job.waitForCompletion(verbose: true) ? 0 : 1;
}

public static void main(String[] args) throws Exception {
    String path = args[0]; //要遍历的路径
    File file = new File(path); //获取File对象
    File[] fs = file.listFiles(); //遍历path下的文件和目录，放入File数组中
    for(File f:fs){
        int l = f.toString().toString().split(regex:"/").length;
        String[] para = f.toString().toString().split(regex:"/")[l-1];
        String[] args1 = para.toString().toString().split(regex:"\\.\"[0];
        args[0] += f.toString().toString().split(regex:"/\"[l-1];
        args[1] += f.toString().toString().split(regex:"/\"[l-1].split(regex:"\\.\"[0];
        "-skip","stop/punctuation.txt","-skip","stop/stop-word-list.txt");
        driver(para);
    }
    String[] para = new String[]{"-Dwordcount.case.sensitive=false",
        "-Dmapreduce.output.textoutputformat.separator",
        args[0], args[1]+"summary", "-skip", "stop/punctuation.txt", "-skip", "stop/stop-word-list.txt"};
    driver(para);
}

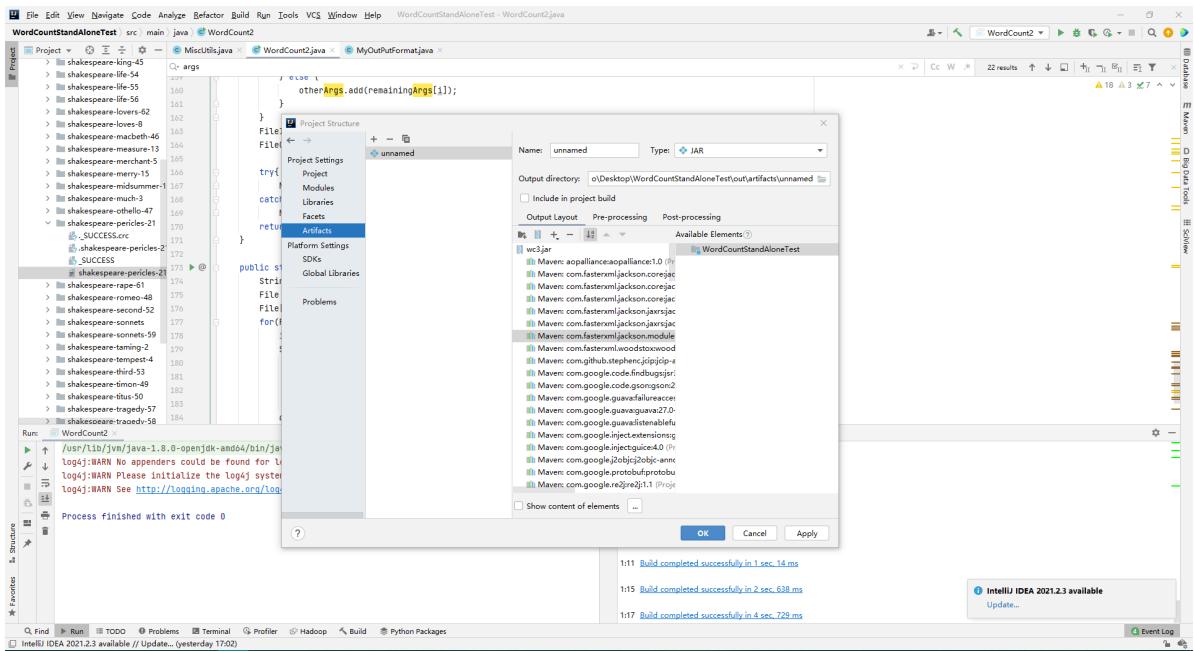
```

Event Log:

- 1:01 All files are up-to-date
- 1:09 Build completed successfully in 2 sec. 635 ms
- 1:09 Build completed successfully in 2 sec. 637 ms
- 1:10 All files are up-to-date
- 1:11 Build completed successfully in 1 sec. 14 ms
- 1:15 Build completed successfully in 2 sec. 638 ms
- 1:17 Build completed successfully in 4 sec. 729 ms

IntelliJ IDEA 2021.2.3 available
Update...

这里先发现了一点需要修改的参数和代码，当时图省事对于遍历目录用了绝对目录，这样在hdfs里是行不通的，应该改成传参控制。



打jar包

```

root@h01:/usr/local/hadoop# docker cp C:\Users\Lenovo\Desktop\spword f0eef628af5d:/usr/local/hadoop/wordcount3.0
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword 2645e9397c95 :/usr/local/hadoop/wordcount3.0
PS C:\Users\Lenovo> "docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-|
       docker cp [OPTIONS] SRC_PATH|-> CONTAINER:DEST_PATH

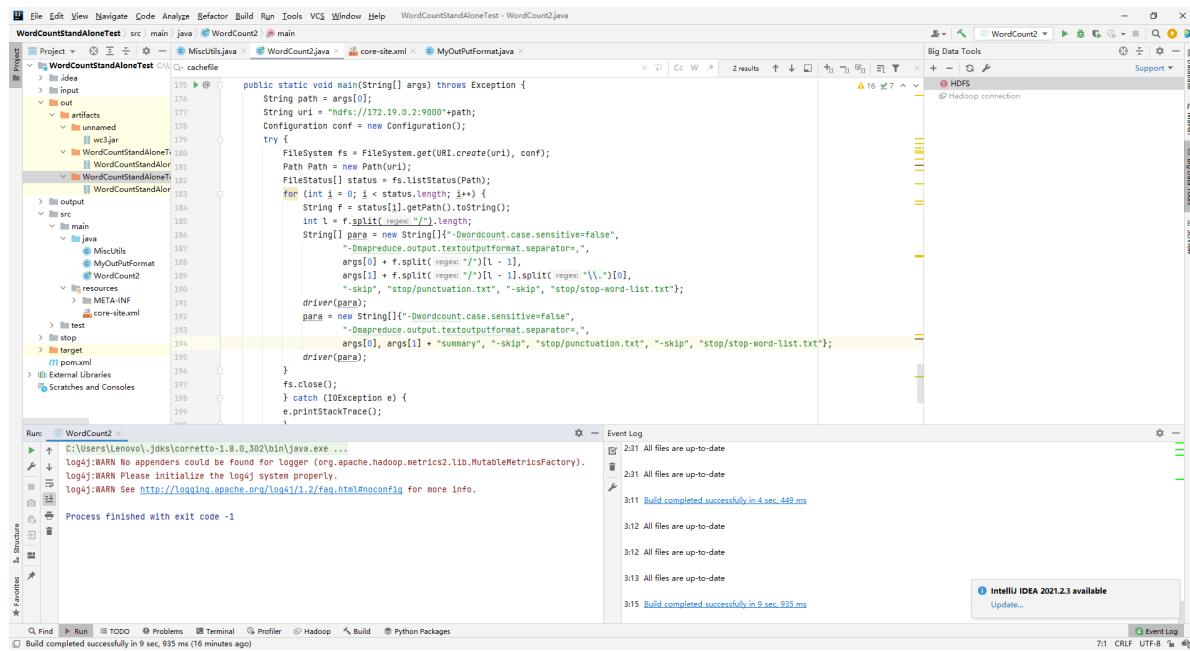
Copy files/folders between a container and the local filesystem
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\spword 2645e9397c95:/usr/local/hadoop/wordcount3.0
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\WordCountStandAloneTest\out\artifacts\unnamed\wc3.jar 2645e9397c95
:/usr/local/hadoop/wc3.jar
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\WordCountStandAloneTest\out\artifacts\unnamed\wc3.jar f0eef628af5d
:/usr/local/hadoop/wc3.jar
PS C:\Users\Lenovo> docker cp C:\Users\Lenovo\Desktop\WordCountStandAloneTest\out\artifacts\unnamed\wc3.jar c726100c8082
:/usr/local/hadoop/wc3.jar
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-|
       docker cp [OPTIONS] SRC_PATH|-> CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
PS C:\Users\Lenovo> |

```

转运jar包到h01



运行方法是在hadoop目录下运行bin/hadoop jar wc3.jar

/user/fospro/WC3/wordcount3.0/input/ /user/fospro/WC3/wordcount3.0/output/，由于有些折磨就没顾及细节，参数的任意增添，文件夹"/"符号的增减、停用词相对位置的变化都会导致无法运行，这里是成大事不拘小节了。

这里在docker跑不通后就发现了另一处问题，file读不到内容。之前idea的遍历可以直接用file类进行文件操作，但是file类并不能对hdfs操作，应该改为FileSystem类对文件系统操作。参考了[HDFS的Java客户端操作代码\(查看HDFS下所有的文件或目录\)详解大数据IT虾米网\(itxm.cn\)](#)。

```

        WRONG_MAP=0
        WRONG_REDUCE=0
WordCount2$TokenizerMapper$CountersEnum
        INPUT_WORDS=8551
File Input Format Counters
        Bytes Read=135197
File Output Format Counters
        Bytes Written=1084
2021-10-31 03:26:10,272 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at h01/172.19.0.2:8032
2021-10-31 03:26:10,295 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1635615137792_0003
2021-10-31 03:26:12,339 INFO input.FileInputFormat: Total input files to process : 37
2021-10-31 03:26:12,437 INFO mapreduce.JobSubmitter: number of splits:37
2021-10-31 03:26:12,506 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1635615137792_0003
2021-10-31 03:26:12,506 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-10-31 03:26:12,556 INFO impl.YarnClientImpl: Submitted application application_1635615137792_0003
2021-10-31 03:26:12,567 INFO mapreduce.Job: The url to track the job: http://h01:8088/proxy/application_1635615137792_0003/
2021-10-31 03:26:12,567 INFO mapreduce.Job: Running job: job_1635615137792_0003
2021-10-31 03:26:28,869 INFO mapreduce.Job: Job job_1635615137792_0003 running in uber mode : false
2021-10-31 03:26:28,869 INFO mapreduce.Job: map 0% reduce 0%
Killed
root@h01:/usr/local/hadoop# jps
5300 DataNode
8168 Jps
7610 MRAppMaster
5484 SecondaryNameNode
5180 NameNode
root@h01:/usr/local/hadoop#

```

不幸的是，可能因为电脑负载过大，散热扇呼呼地转，点开此电脑c盘也接近爆满，ResourceManager挂了，进程被杀死了。

The screenshot shows the Hadoop Web UI interface and a terminal window. The terminal window displays the logs of the wordcount job, which failed due to a file already existing exception. The Hadoop UI shows the cluster metrics and a table of submitted jobs, with one job listed.

ID	User	Name	Application Type	Application Tags	Queue	App Status
application_1635623592556_0001	root	word count	MAPREDUCE		default	0

不过为了确保不是代码问题任务确实是开始运行的，同时监控网页端口，发现确实是接收到了任务，至少还完成了一个接到了第二个。

edoop

All Applications | NameNode Information | localhost:8088/cluster

翻译 英语 页面? 中文 (简体) 始终翻译至简体中文

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
2	0	1	1	23	<memory:24 GB, vCores:23>	<memory:24 GB, vCores:24>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes
3	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type
Capacity Scheduler	[memory-mb (unit=Mi), vcores]

Show 20 ▾ entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	Start Time	Launch Time	Finish Time	State	Final Status	Running Containers	Allocated CPU Vcores	Allocated Memory MB
application_1635623592556_0002	root	word count	MAPREDUCE		default	0	Sun Oct 31 03:55:29 +0800 2021	Sun Oct 31 03:55:33 +0800 2021	Sun Oct 31 03:55:39 +0800 2021	RUNNING	UNDEFINED	23	23	24576
application_1635623592556_0001	root	word count	MAPREDUCE		default	0	Sun Oct 31 03:54:59 +0800 2021	Sun Oct 31 03:55:26 +0800 2021	Sun Oct 31 03:55:26 +0800 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A

Showing 1 to 2 of 2 entries

Map output records=8551
Map output bytes=78395
Map output materialized bytes=95413
Input split bytes=140
Combine input records=0
Combine output records=0
Reduce Input groups=2353
Reduce shuffle bytes=95413
Reduce input records=8551
Reduce output records=120
Split input records=17102
Total Map outputs=1
Failed Map outputs=0
Merged Map outputs=1
GC time elapsed (ms)=120
CPU time spent (ms)=4470
Physical memory (bytes) snapshot=666604288
Virtual memory (bytes) snapshot=319294080
Total committed heap usage (bytes)=5616890688
Peak Mem Physical memory (bytes)=402881152
Peak Map Virtual memory (bytes)=2562658384
Peak Reduce Physical memory (bytes)=194523136
Peak Reduce Virtual memory (bytes)=2570166272

Shuffle Metrics

BAD ID=0 CONNECTION=8 IO_ERROR=0 WRONG_LENGTH=0 WRONG_MAP=0 WRONG_TYPE=0
WordCount2TokenizerMapper\$CountersEnum INPUT_WORDS=8551 File Input Format Counters Bytes Read=135197 File Output Format Counter Bytes Written=1084

```
2021-10-31 03:55:28,026 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at h01/172.19.0.2:8082
2021-10-31 03:55:28,026 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1635623592556_0002
2021-10-31 03:55:29,277 INFO input.FileInputFormat: Total input files to process : 37
2021-10-31 03:55:29,362 INFO mapreduce.JobSubmitter: number of splits:37
2021-10-31 03:55:29,402 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1635623592556_0002
2021-10-31 03:55:29,429 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-10-31 03:55:29,463 INFO impl.YarnClientImpl: Submitted application application_1635623592556_0002
2021-10-31 03:55:29,469 INFO mapreduce.Job: The url to track the job: http://h01:8088/proxy/application_1635623592556_0002/
2021-10-31 03:55:29,478 INFO mapreduce.Job: Running job: job_1635623592556_0002
2021-10-31 03:55:48,562 INFO mapreduce.Job: Job job_1635623592556_0002 running in uber mode : false
2021-10-31 03:55:48,563 INFO mapreduce.Job: map % reduce %
```

hadoop

All Applications | NameNode Information | localhost:8088/cluster

翻译 英语 页面? 中文 (简体) 始终翻译至简体中文

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
2	0	1	1	23	<memory:24 GB, vCores:23>	<memory:24 GB, vCores:24>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
3	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 ▾ entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	Start Time	Launch Time	Finish Time	State	Final Status	Running Containers	Allocated CPU Vcores	Allocated Memory MB
application_1635623592556_0002	root	word count	MAPREDUCE		default	0	Sun Oct 31 03:55:29 +0800 2021	Sun Oct 31 03:55:33 +0800 2021	Sun Oct 31 03:55:39 +0800 2021	RUNNING	UNDEFINED	23	23	24576
application_1635623592556_0001	root	word count	MAPREDUCE		default	0	Sun Oct 31 03:54:59 +0800 2021	Sun Oct 31 03:55:26 +0800 2021	Sun Oct 31 03:55:26 +0800 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A

Showing 1 to 2 of 2 entries

```

root@h01:/usr/local/hadoop x + v
      Map output bytes=78305
      Map output materialized bytes=95413
      Input split bytes=140
      Combine input records=0
      Combine output records=0
      Reduce input groups=2353
      Reduce shuffle bytes=95413
      Reduce input records=8551
      Reduce output records=100
      Spilled Records=17102
      Shuffled Maps =1
      Failed Shuffles=0
      Merged Map outputs=1
      GC time elapsed (ms)=120
      CPU time spent (ms)=4470
      Physical memory (bytes) snapshot=686604288
      Virtual memory (bytes) snapshot=5132824576
      Total committed heap usage (bytes)=563609600
      Peak Map Physical memory (bytes)=492081152
      Peak Map Virtual memory (bytes)=2562658304
      Peak Reduce Physical memory (bytes)=194523136
      Peak Reduce Virtual memory (bytes)=2570166272
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  WordCount2$TokenizerMapper$CountersEnum
    INPUT_WORDS=8551
  File Input Format Counters
    Bytes Read=135197
  File Output Format Counters
    Bytes Written=1084
2021-10-31 03:55:27,997 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at h01/172.19.0.2:8032
2021-10-31 03:55:28,026 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1635623592556_0002
2021-10-31 03:55:29,277 INFO input.FileInputFormat: Total input files to process : 37
2021-10-31 03:55:29,362 INFO mapreduce.JobSubmitter: number of splits:37
2021-10-31 03:55:29,428 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1635623592556_0002
2021-10-31 03:55:29,429 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-10-31 03:55:29,463 INFO impl.YarnClientImpl: Submitted application application_1635623592556_0002
2021-10-31 03:55:29,469 INFO mapreduce.Job: The url to track the job: http://h01:8088/proxy/application_1635623592556_0002/
2021-10-31 03:55:29,470 INFO mapreduce.Job: Running job: job_1635623592556_0002
2021-10-31 03:55:40,562 INFO mapreduce.Job: Job job_1635623592556_0002 running in uber mode : false
2021-10-31 03:55:40,563 INFO mapreduce.Job: map 0% reduce 0%
Killed
(arg: -1) |

```

```

root@h01:/usr/local/hadoop x + v
root@h01:/usr/local/hadoop# bin/hdfs dfs -ls /user/flospro/WC3/wordcount3.0/output/
Found 2 items
drwxr-xr-x  - root supergroup          0 2021-10-31 03:55 /user/flospro/WC3/wordcount3.0/output/shakespeare-all-11
drwxr-xr-x  - root supergroup          0 2021-10-31 03:55 /user/flospro/WC3/wordcount3.0/output/summary
root@h01:/usr/local/hadoop# bin/hdfs dfs -cat /user/flospro/WC3/wordcount3.0/output/shakespeare-all-11/shakespeare-all-11.txt-r-00000
-rw-r--r--  3 root supergroup     1884 2021-10-31 03:55 /user/flospro/WC3/wordcount3.0/output/shakespeare-all-11/shakespeare-all-11.txt-r-00000
root@h01:/usr/local/hadoop# bin/hdfs dfs -cat /user/flospro/WC3/wordcount3.0/output/shakespeare-all-11/shakespeare-all-11.txt-r-00000
1:hve,197
2:ths,191
3:jps,177
4:shb,171
5:will,157
6:ten,140
7:rtr,136
8:kng,128
9:frst,118
10:are,117
11:shl,115
12:tess,99
13:thou,95
14:whch,87
15:thy,84
16:the,73
17:love,68
18:enter,58
19:gve,51
20:smke,49
21:let,48
22:are,43
23:spk,41
24:thnk,39
25:leve,39
26:ike,37
27:rng,36
28:are,32
29:ext,31
30:set,30
31:wfe,30
32:ble,28
33:lls,28
34:ans,28
35:fll,17
36:olce,27
37:knve,27
38:ext,26
39:pry,26
40:rol,20
41:are,26
42:ster,25
43:idy,25
44:lfe,25

```

但还是很不幸地被killed了，不过如图在hdfs中打开已经计算完毕的结果来看，已经运行出来的结果都是正确且符合要求的。对代码验证的担心一方面是变了环境，另一方面是宿主机无法连上hdfs的接口，ping docker也超时，file类修改成filesystem类的那段代码没法得到验证，还好结果是没问题的，不过全部文本都跑一遍MR确实吃不消，应该考虑优化。下周上课前有必要去和老师申请BDKit。