

FBDP作业七

Kmeans的设计思路

Kmeans的主要逻辑是聚类，关键在于如何将“相似”点分配在一个类别中；而“相似度”的判断则来自我们对distance计算的逻辑与选取的距离公式。

1. 随机选取三个数据点，并根据最短distance原则，将剩余点分到着三个类别中
2. 每个类别的点，求平均横坐标与纵坐标，形成新的中心点，再次分类
3. 迭代过程2
4. 最后输出分类结果
5. 尝试在ide中实现mapreduce程序的编译与调试

所遇到的问题

1. IntelliJ的maven项目没有package：
首先，本次Kmeans代码较长较复杂，网络上的操作方法一般需要多个class协助完成。而在前两次作业的设计中发现IntelliJ Maven打包的项目中没有设定package。
解决方法：首先对/src/java文件夹，选择mark directory as resource root，自动导入package。
2. 多个class的情况下，如何写pom.xml？
首先要设置好package的名称，在每个class中都要标注引入package。在pom中要注明最主要的main class在哪。
3. 不理解第一次reduce输出后，value为所有属于该类的点应该如何处理？
解决：实际上，第一次reduce输出后，只有key，也就是cluster的平均坐标将被使用，而后续继续计算的点坐标又将来自于input文件。
4. 发现问题：第一次迭代输出为空，但是最初init部分可以得到目标结果，而两次map-reduce进程均使用相同的reducer。问题可能出现在迭代文件引用安排上。

```

yuanshan@yuanshan-virtual-machine:~/下载/kmeans$ hdfs dfs -ls /kmeans/output/Data0
Found 2 items
-rw-r--r-- 1 yuanshan supergroup 0 2020-11-08 10:24 /kmeans/output/Data0/_SUCCESS
-rw-r--r-- 1 yuanshan supergroup 672 2020-11-08 10:24 /kmeans/output/Data0/part-r-00000
yuanshan@yuanshan-virtual-machine:~/下载/kmeans$ hdfs dfs -get /kmeans/output ./output
2020-11-08 10:34:07,124 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
yuanshan@yuanshan-virtual-machine:~/下载/kmeans$ cd output
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$ ls
Data0 Data1
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$ cd Data0
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output/Data0$ ls
part-r-00000 _SUCCESS
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output/Data0$ cd ..
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$ vim Data0/part-r-00000
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$ vim Data1/part-r-00000
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$ cd Data1
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output/Data1$ ls
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output/Data1$

```

Debug思路：由于涉及文件读取 - mapper - reducer - 文件存储 这样的过程，首先验证原参考代码的正确性。若仍然无法通过，说明是文件读取方式不同造成的差异，应当修正为之前用过的文件读取方法。若通过，则说明多个main函数加在一个class里面的运用有问题，不应该以目前代码的组织形式构建。

发现：按照作者的class组织架构进行运行，仍然有找不到文件的错误，说明文件加载方式出现了问题。

决定使用另外的文件组织形式，参考之前的“共同好友”以及官网提供的wordcount方式进行更改。

详细分析发现：原文作者使用的是

```
DistributedCache.getLocalCacheFiles(conf)
```

获得之前reduce产出的文件，实际操作时应该是在local上查找文件，而非在Hdfs上查找，从此处报错信息可以分析原因：

```

2020-11-08 14:33:46,800 INFO mapred.MapTask: Map output collector class = org.ap
ache.hadoop.mapred.MapTask$MapOutputBuffer
2020-11-08 14:33:46,817 INFO mapred.MapTask: Starting flush of map output
2020-11-08 14:33:46,830 INFO mapred.LocalJobRunner: map task executor complete.
2020-11-08 14:33:46,840 WARN mapred.LocalJobRunner: job_local418126945_0002
java.lang.Exception: java.io.FileNotFoundException: file:/usr/local/hadoop/tmp/m
apred/local/job_local418126945_0002_8f3215db-4626-4b6a-ba43-2d904c76da64/part-r-
000000 (没有那个文件或目录)
    at org.apache.hadoop.mapred.LocalJobRunner$Job.runTasks(LocalJobRunner.j
ava:492)
    at org.apache.hadoop.mapred.LocalJobRunner$Job.run(LocalJobRunner.java:5
52)
Caused by: java.io.FileNotFoundException: file:/usr/local/hadoop/tmp/mapred/loca
l/job_local418126945_0002_8f3215db-4626-4b6a-ba43-2d904c76da64/part-r-000000 (没
有那个文件或目录)
    at java.io.FileInputStream.open0(Native Method)
    at java.io.FileInputStream.open(FileInputStream.java:195)
    at java.io.FileInputStream.<init>(FileInputStream.java:138)
    at java.io.FileInputStream.<init>(FileInputStream.java:93)
    at java.io.FileReader.<init>(FileReader.java:58)
    at com.cys.kms.kmeansIter$PRIterMapper.setup(kmeansIter.java:52)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:143)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:799)

```

导致迭代开始，第一次迭代数据产出文件无法找到。

于是我换成

```
Job.*getInstance*(conf).getCacheFiles()
```

发现仍然找不到第一次迭代数据产出文件，检查报错发现，是阅读文件时string转double出现了错误，导致Mapper环节setup失败。修改后导出成功。

5. 将迭代次数调整为20，查看输出结果。

```

merged map outputs=1
GC time elapsed (ms)=0
Total committed heap usage (bytes)=1015021568

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

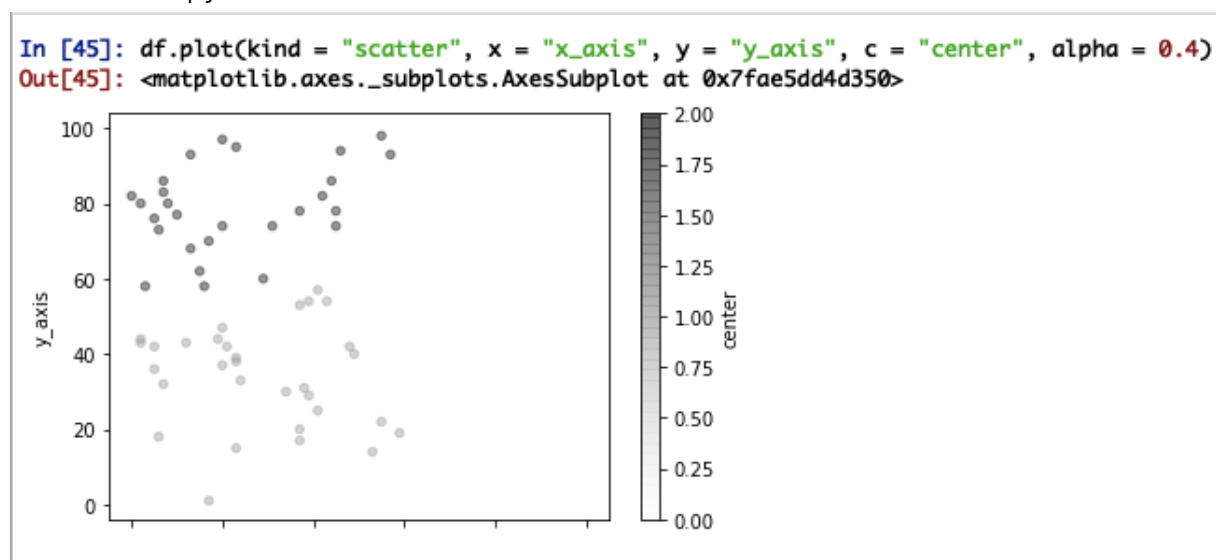
File Input Format Counters
Bytes Read=584

File Output Format Counters
Bytes Written=784
yuanshan@yuanshan-virtual-machine:~/下载/kmeans$ hdfs dfs -get /kmeans/output ./
output
2020-11-08 16:01:43,016 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
yuanshan@yuanshan-virtual-machine:~/下载/kmeans$ cd output
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$ ls
Data0  Data11  Data14  Data17  Data2  Data4  Data7  FinalRank
Data1  Data12  Data15  Data18  Data20  Data5  Data8
Data10 Data13  Data16  Data19  Data3  Data6  Data9
yuanshan@yuanshan-virtual-machine:~/下载/kmeans/output$

```

```
0 80,99
0 76,94
0 93,46
0 60,68
0 70,23
0 79,43
0 88,98
0 83,33
0 86,43
0 73,93
0 58,74
0 96,47
0 61,36
0 87,75
0 100,60
0 96,66
0 99,54
0 89,27
0 62,69
0 81,86
0 61,82
0 72,60
0 81,28
"FinalRank/part-r-00000" 100L, 784C 1,1 顶端
```

下面打算使用python将结果以可视化形式展示输出。



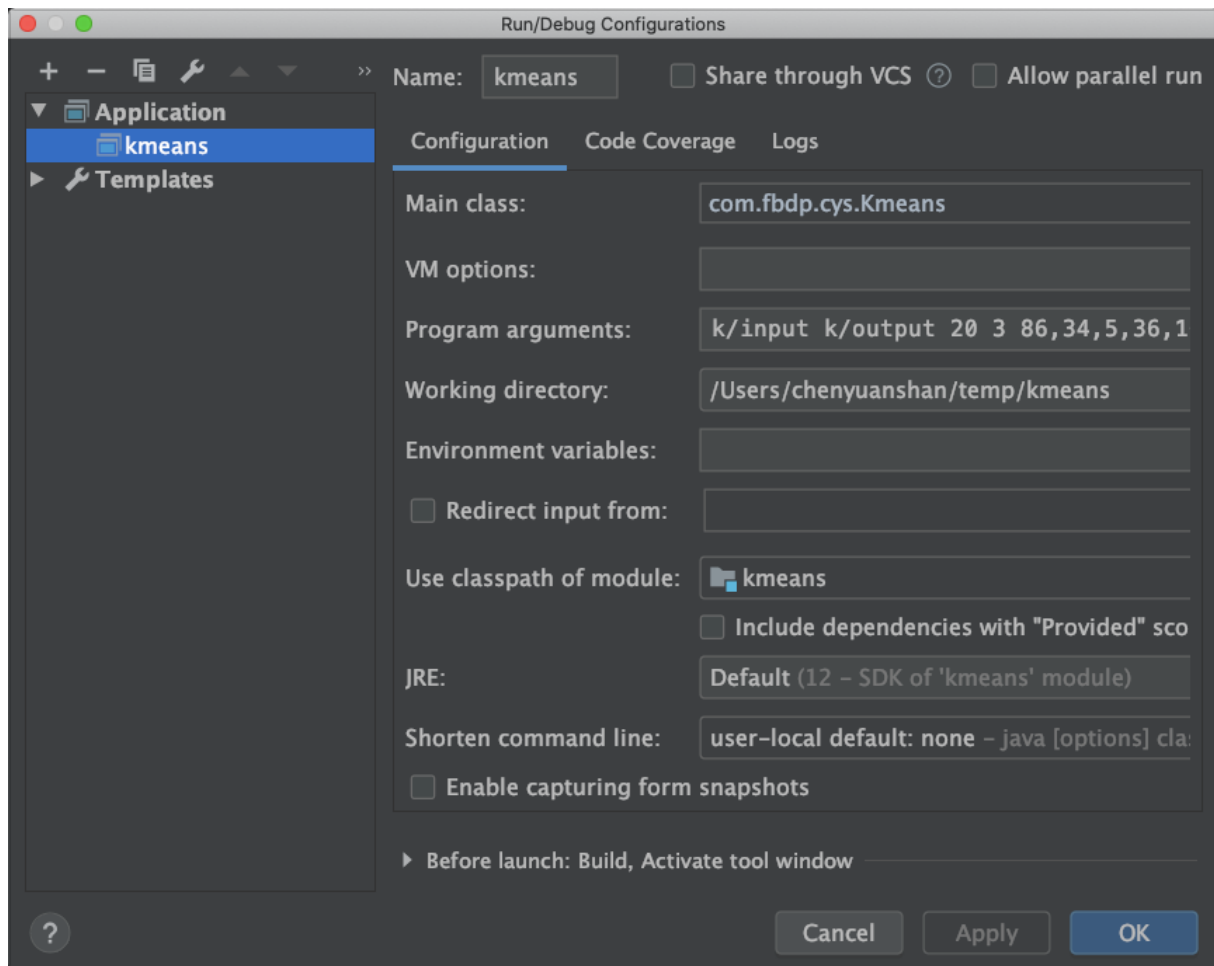
6. 本次新实践：在ide中给mapreduce程序运行与debug

尝试修改不同迭代次数与中心数量时，因为多次修改参数比较麻烦，所以想尝试直接在ide中进行编译。一直以来，mapreduce的debug比较复杂，没有中间输出结果很难找到一个bug的所在，而在ide中调试可以更快且方便找到bug。

本次作业使用的ide为intellij。

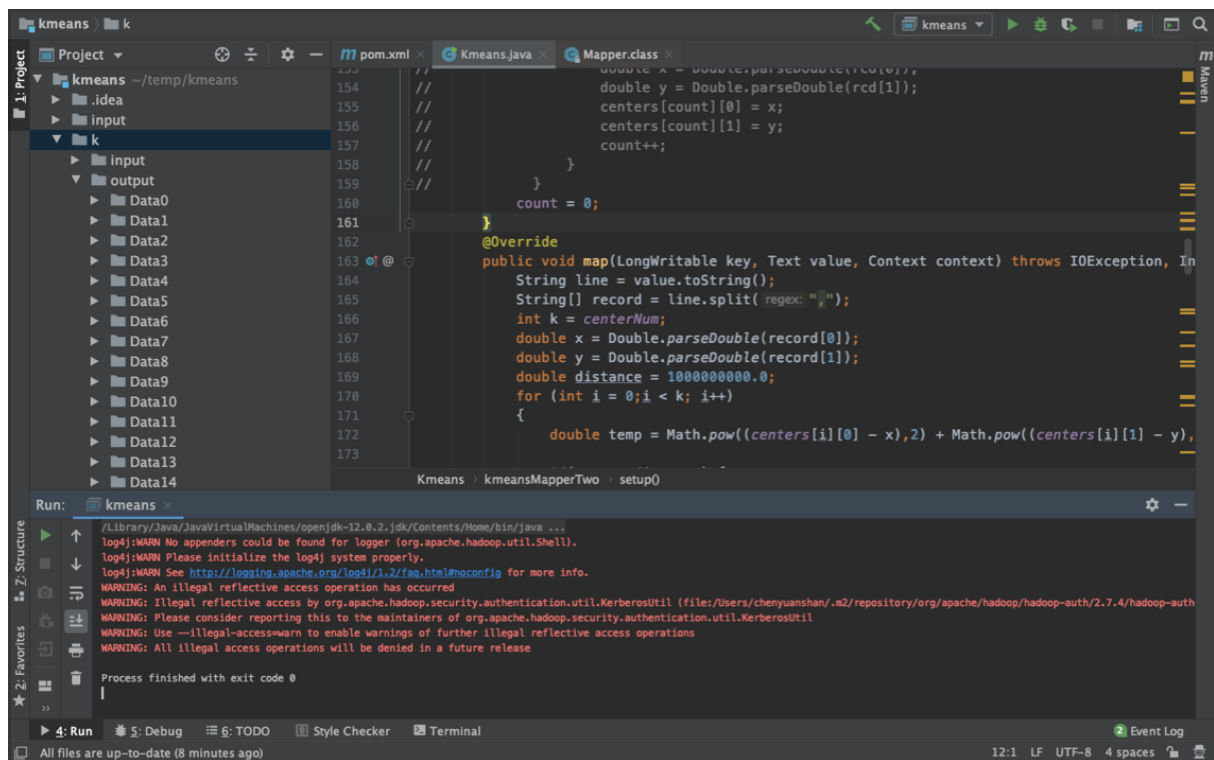
首先，设置运行时的configuration，设置运行的项目名称，指出main class（也就是driver的所在），再输入我们的argument，也就是运行参数。我设置的格式对应伪分布集群的后半部分，即wordcount/input wordcount/output部分，在本次试验中，为input_file_path

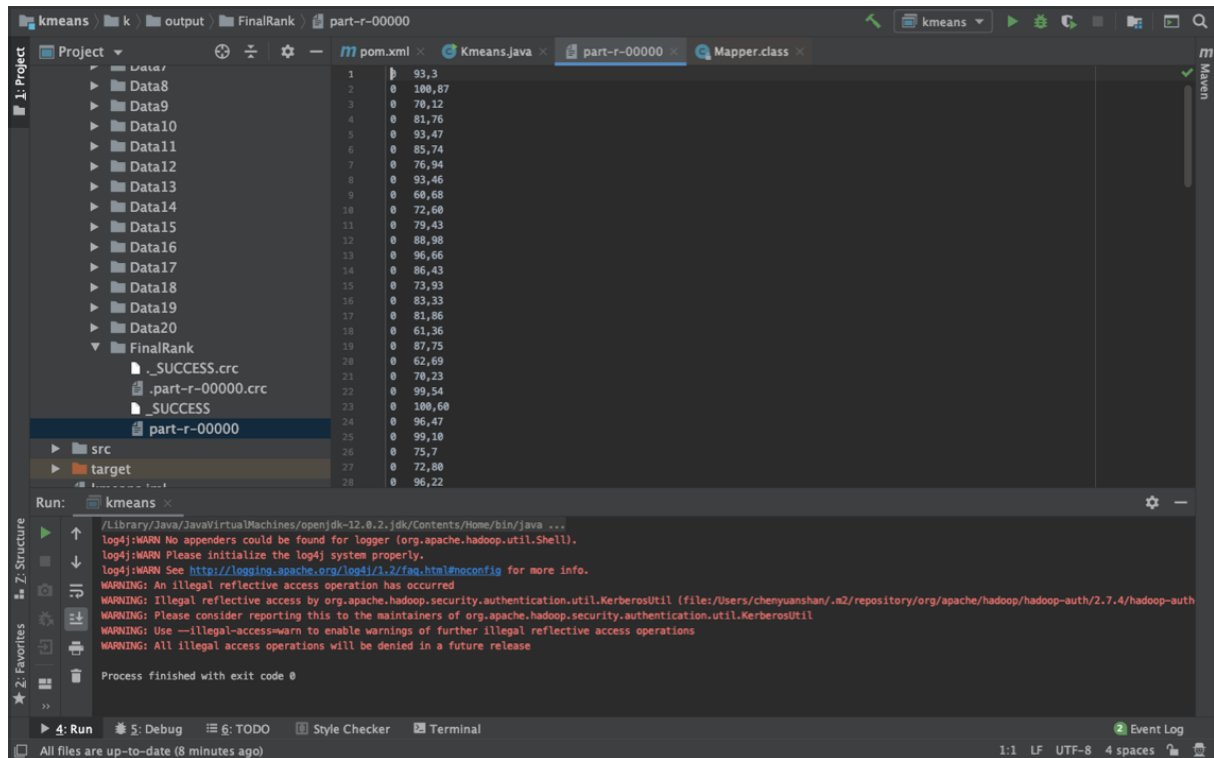
output_file_path iterate_times center_number initial_center



设置好后，就可以运行Run 或者Debug按钮

最终执行的输出结果为





可以完善的地方

1. 应随机取三个点，而非人工选定初始值
2. Java的原则是oop，那么在写class时，应该分类更有层次，使得对象结构清晰，代码可读性更高

疑问

1. 在intellij中编译，可以找到文件且便已成功，但是在伪分布集群run的时候发现报错找不到文件，但是在output文件夹中已经发现该文件的存在。可能是两种运行的路径不一样，缓存路径的保存方法不一样，具体区别还等待进一步讨论。

```
2020-11-14 19:05:03,529 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2020-11-14 19:05:03,529 INFO mapred.MapTask: kvstart = 26214396; length = 655360
0
2020-11-14 19:05:03,530 INFO mapred.MapTask: Map output collector class = org.ap
ache.hadoop.mapred.MapTask$MapOutputBuffer
2020-11-14 19:05:03,538 INFO mapred.MapTask: Starting flush of map output
2020-11-14 19:05:03,548 INFO mapred.LocalJobRunner: map task executor complete.
2020-11-14 19:05:03,554 WARN mapred.LocalJobRunner: job_local272735081_0002
java.lang.Exception: java.io.FileNotFoundException: /kmeans/output/Data0/part-r-
00000 (没有那个文件或目录)
    at org.apache.hadoop.mapred.LocalJobRunner$Job.runTasks(LocalJobRunner.j
ava:492)
    at org.apache.hadoop.mapred.LocalJobRunner$Job.run(LocalJobRunner.java:5
52)
Caused by: java.io.FileNotFoundException: /kmeans/output/Data0/part-r-00000 (没
有那个文件或目录)
    at java.io.FileInputStream.open0(Native Method)
    at java.io.FileInputStream.open(FileInputStream.java:195)
    at java.io.FileInputStream.<init>(FileInputStream.java:138)
    at java.io.FileReader.<init>(FileReader.java:93)
    at com.fbdp.cys.Kmeans$KmeansMapperTwo.setup(Kmeans.java:131)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:143)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:799)
```