实验报告

报告标题：Spark

学号：21190630

姓名：黄艺杰

日期：2022年12月27日

# 一、实验环境

1. 操作系统：Windows 10、Linux

2. 相关软件（含版本号）：VMWare15pro、FinalShell3.9、Intellij IDEA2019Professional

3. 其它工具：JDK1.8

# 二、实验内容及其完成情况

（针对上述实验内容逐一详述实验过程）

1. Spark 环境搭建

要求：单机（必做）、集群（必做）、HA（可选）；后续练习必须在集群环境下运行；如选做 HA，则后续练习必须在 HA 环境下运行

验证方法：在虚拟机中查看运行进程，在浏览器中查看监控页面。

单机模式：

先上传spark的压缩包，再解压缩到根目录下并改名为spark



图1.1将spark的压缩包解压到根目录

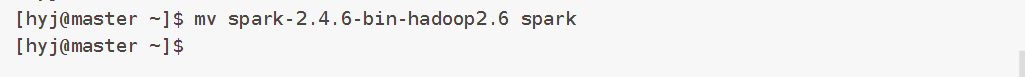


图1.2将spark的文件改名为spark

之后在.bash\_profile中添加spark的bin环境变量

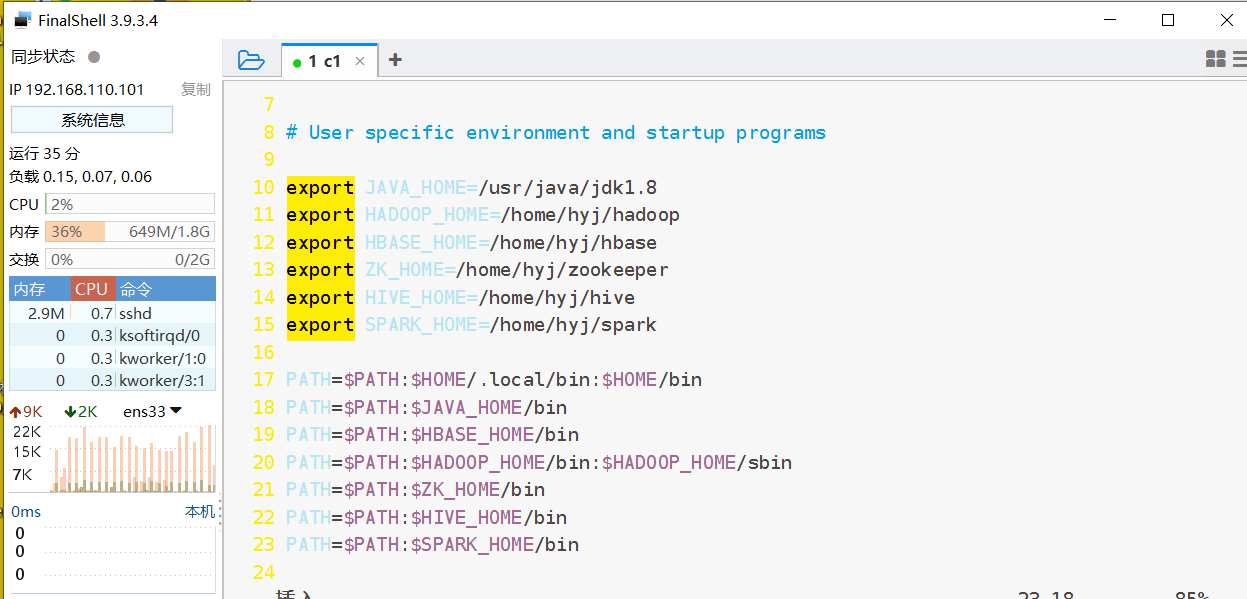


图1.3添加spark的bin环境变量

再配置spark-env.sh

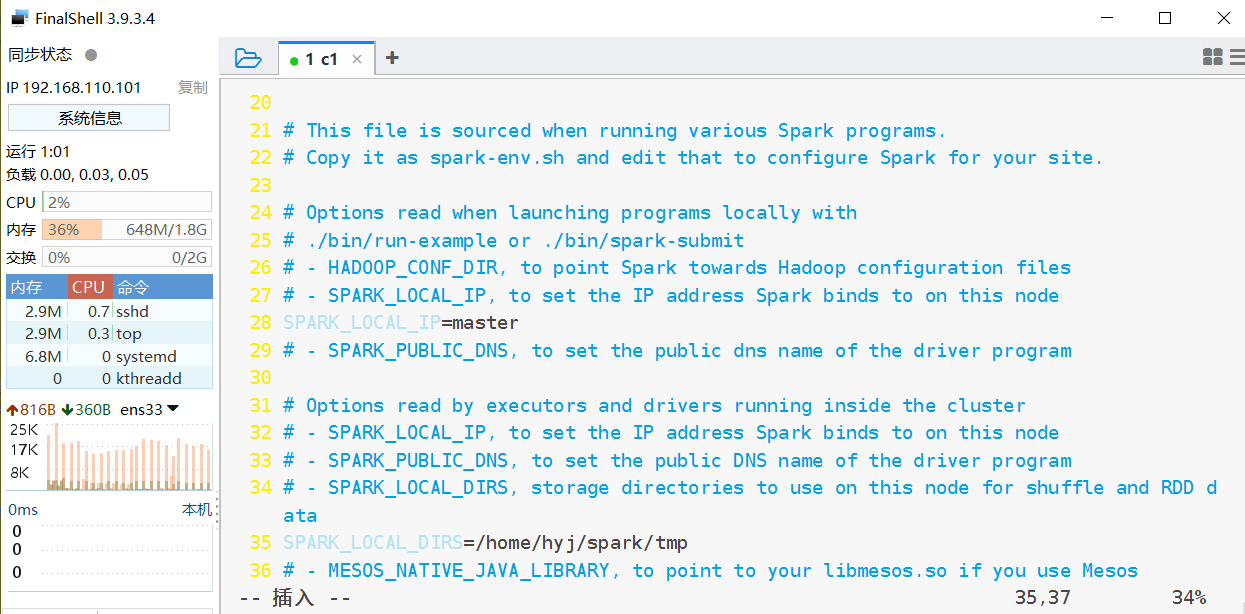


图1.4配置spark-env.sh文件

最后启动spark-shell单机模式

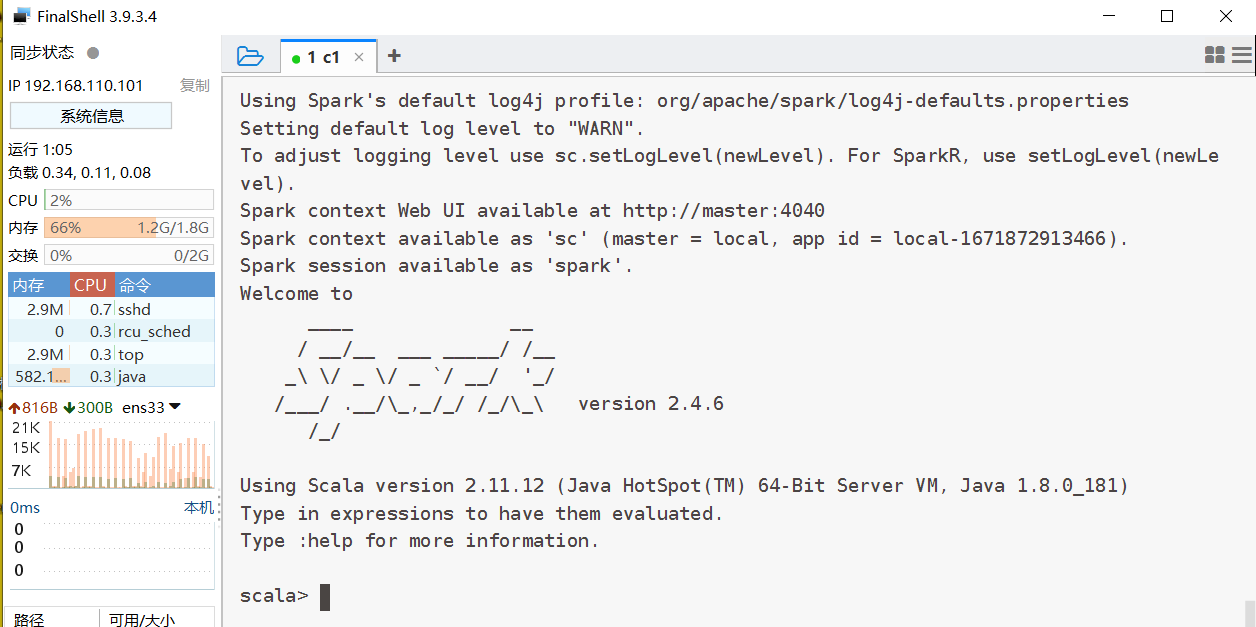


图1.5启动spark-shell单机模式

同时在浏览器4040端口查看

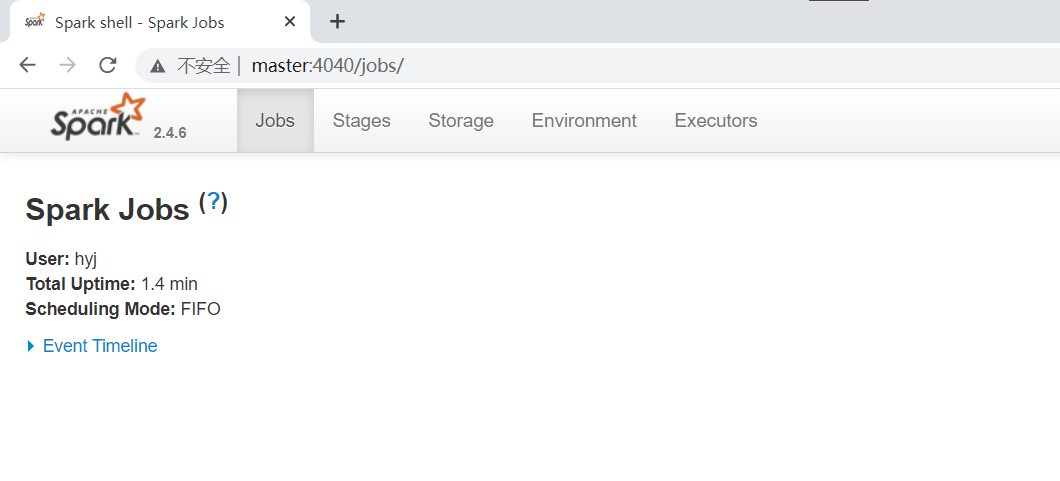


图1.6在浏览器上查看spark页面

集群模式

先在slaves中配置worker的数量

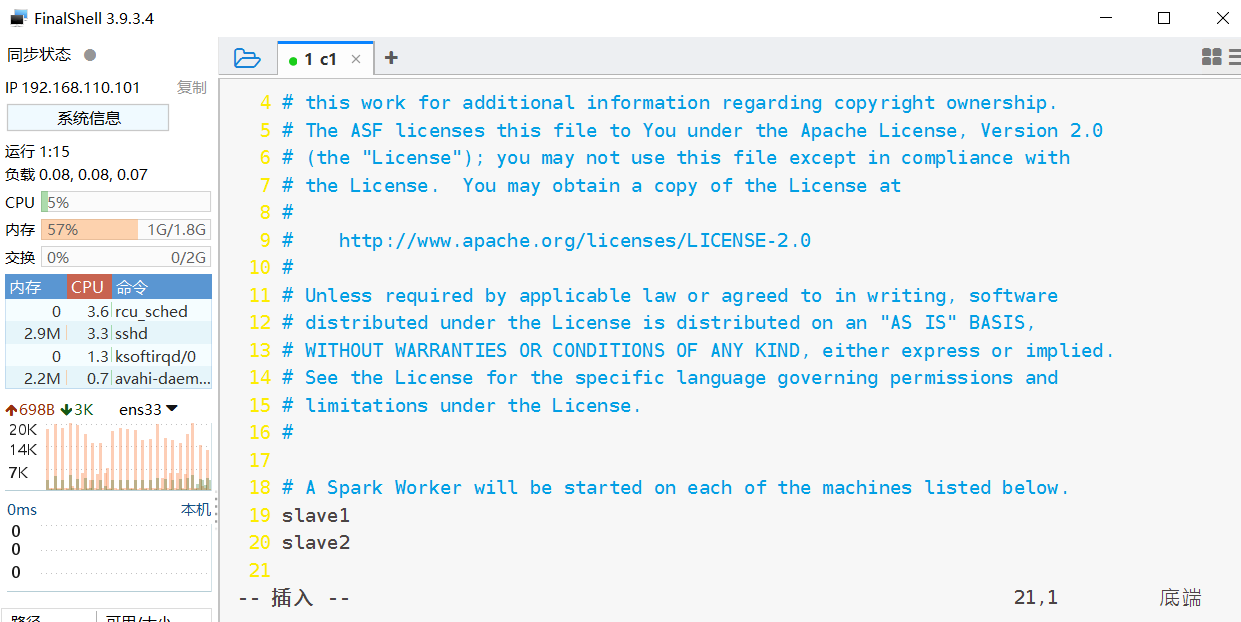


图1.7在slaves中设置worker数量

远程复制spark文件夹到其他两台机器

修改slave1上的spark-env.sh文件，将local IP 从master改为slave1，同理修改slave2

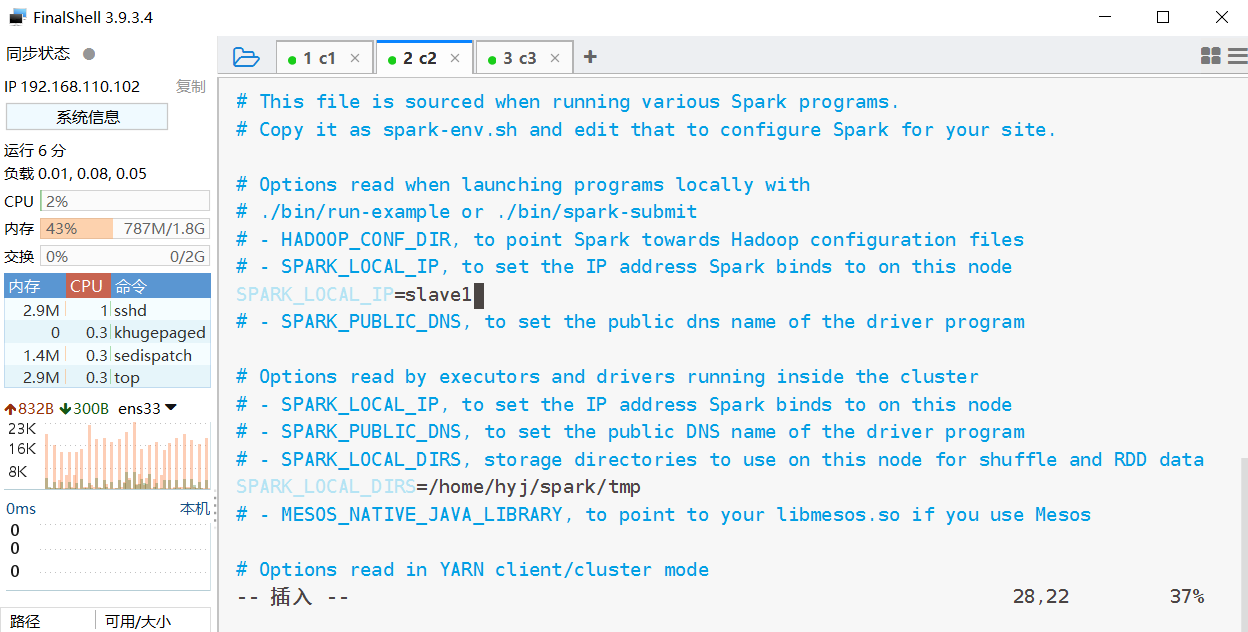


图1.8修改worker的local IP

用sbin下的start-all.sh启动集群，并查看master和worker的jps

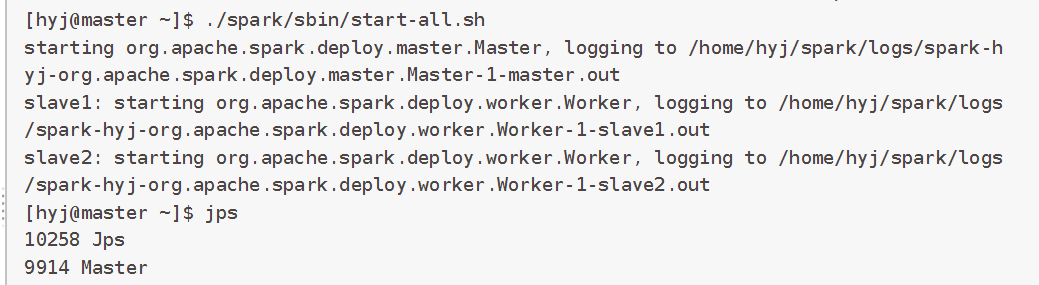


图1.9启动集群并查看master的jps

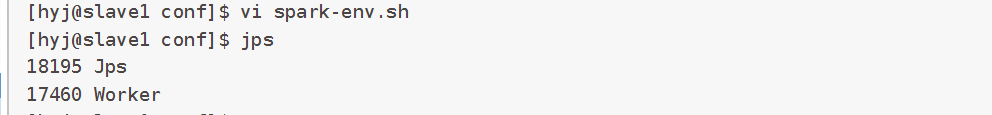


图1.10查看slave1的jps里的Worker

在浏览器的8080端口查看集群监控页面

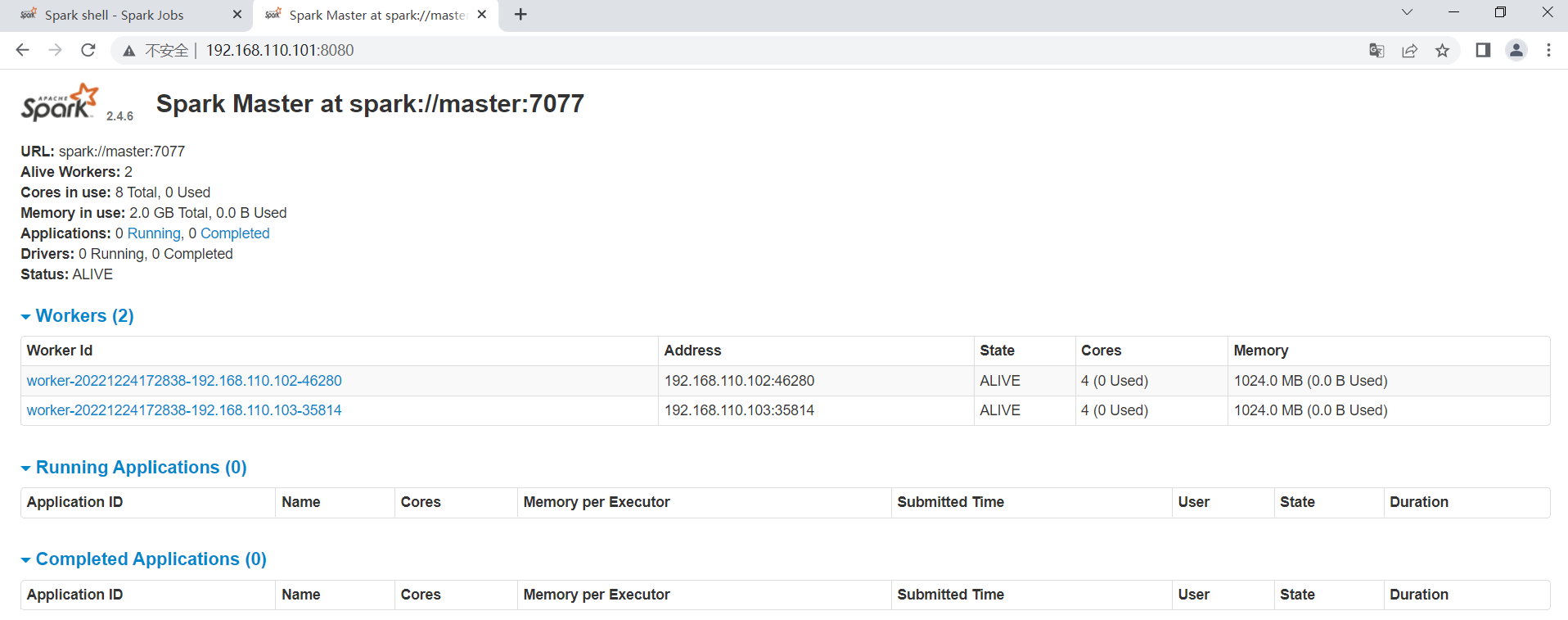


图1.11浏览器查看集群监控页面

1. Scala 练习题

使用 Nilakantha 级数算法计算圆周率。

π = 3 + 4/(2\*3\*4) - 4/(4\*5\*6) + 4/(6\*7\*8) - 4/(8\*9\*10) + 4/(10\*11\*12) - 4/(12\*13\*14) + ……

定义一个 scala object 文件，命名为 MyPiUtil，在其中定义一个计算圆周率的函数，该函数有一个整型参数 n，对应“Nilakantha 级数”算法计算到“4/((n-1)\*n\*(n+1))”;

在 main 方法中接受运行参数，要求该参数必须是一个可以转为大于 10 的奇数，将该参数传递给圆周率计算函数进行计算并输出结果；未传参数或参数不合条件则传递 15 给圆周率计算函数进行计算并输出结果。

完成要求：编写代码，并在本地运行、测试通过；修改后打包上传到虚拟机中，以 spark 集群方式运行 2 次（1 次不传参数，1 次传 21）

验证方法：在 IDEA 中查看最终结果，在虚拟机中查看运行结果。

新建一个Scala-proj项目，并添加spark依赖



图2.1添加spark依赖

新建MyPiUtil.scala文件

代码如下：

1. package edu.nnu
3. import org.apache.spark.SparkConf
4. import org.apache.spark.sql.SparkSession
6. object MyPiUtil extends App{
7. val config=**new** SparkConf().setMaster("local[1]");
8. val spark=SparkSession
9. .builder()
10. .appName("MyPiUtil")
11. .config(config)
12. .getOrCreate();
13. val a=**if**(args.length>0)args(0).toInt **else** 15
14. val n=**if**(a>10&&a%2==1) a **else** 15
15. val func=(x:Int)=>{
16. var m=3.0
17. var session=1
18. **for**(i<-3 to x){
19. m+=(4.0/((i-1)\*i\*(i+1))\*session)
20. session=session\*(-1)
21. }
22. m
23. }
24. val pi=func(n)
25. println("Pi is "+pi)
26. spark.stop()
27. }

运行上述代码，得到不传参数（默认为15）pi的结果

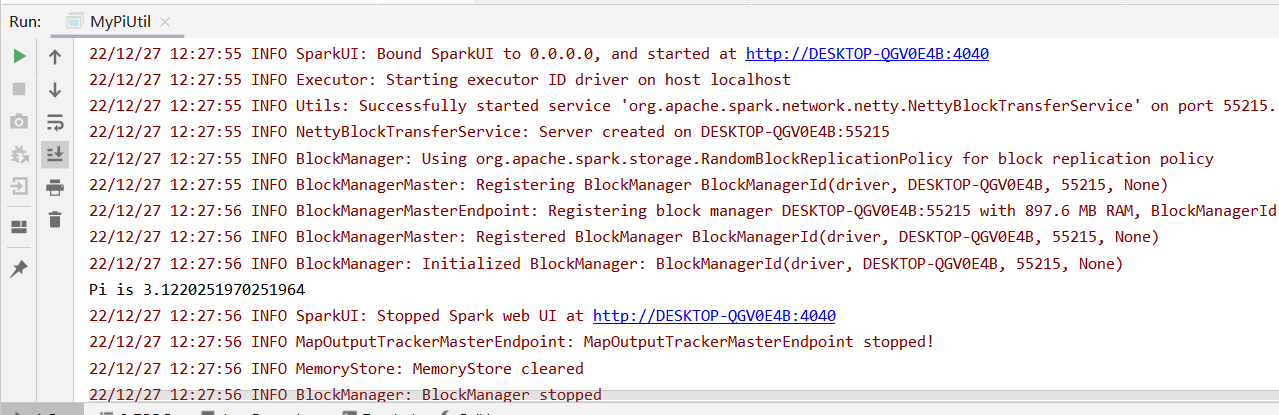


图2.2不传参数pi的结果

之后传参数21得到另一个结果

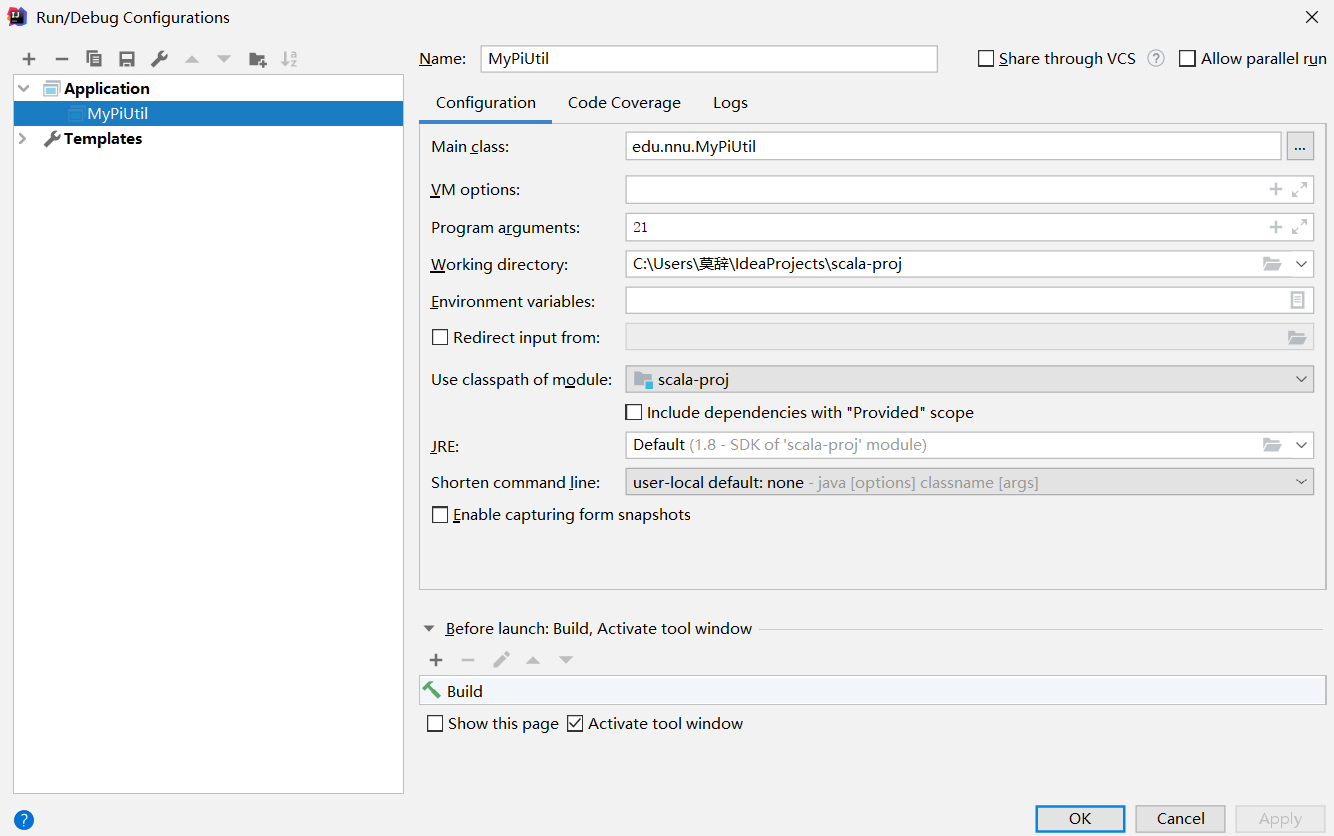


图2.3在Configurations中设置参数21

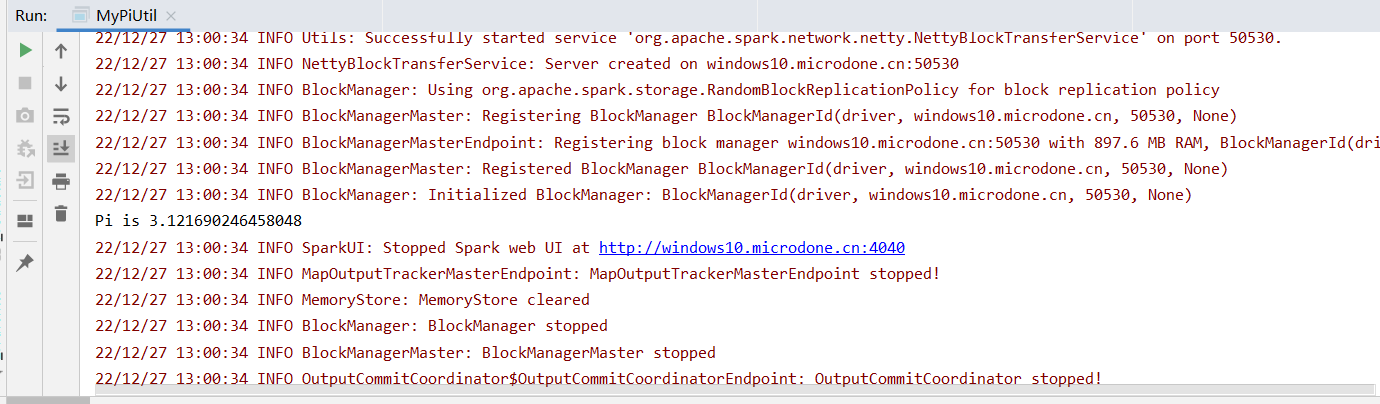


图2.4传参21时pi的结果

将以上代码打包在虚拟机上运行，首先是不传参

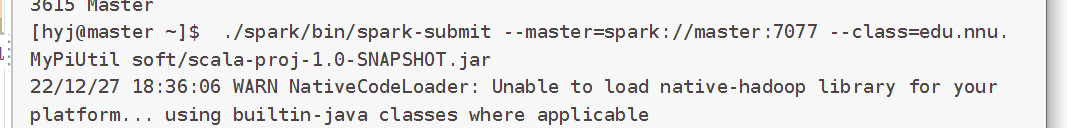


图2.5在虚拟机中不传参运行

运行结果如下：



图2.6在虚拟机中不传参运行结果

之后是后面添加参数21

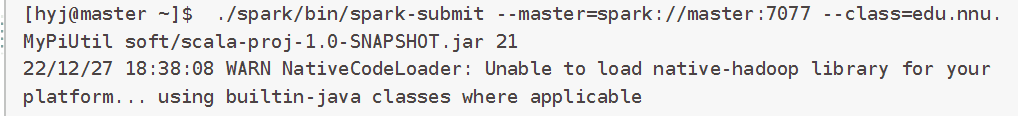


图2.7在虚拟机上传参运行

其结果如下：

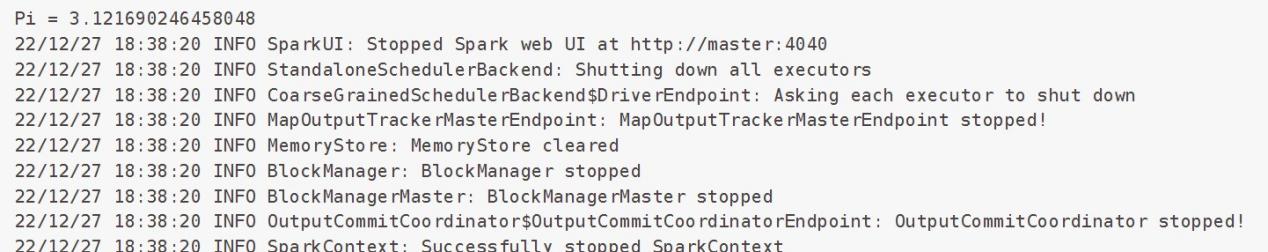


图2.8在虚拟机上传参运行结果

1. Spark 练习题：

根据电影数据查询评分数量超过 20 条的平均评分最高的 20 部电影，输出电影名称、平均评分和评分数量（参考效果如图 1）。必须包括以下步骤：

i.将数据文件（2 个，各文件数据含义如图 2、图 3）存放到 HDFS 环境中：使用 shell

或编程均可

ii.在 Spark Shell 环境下给出导入数据、查询数据的测试截图

iii.编写代码，输出结果，实现方式 SQL、DSL 均可

验证方法：在虚拟机或 IDEA 中查看运行结果。

先在IDEA中将ratings.dat和movies.dat文件上传到hdfs下的data文件夹中

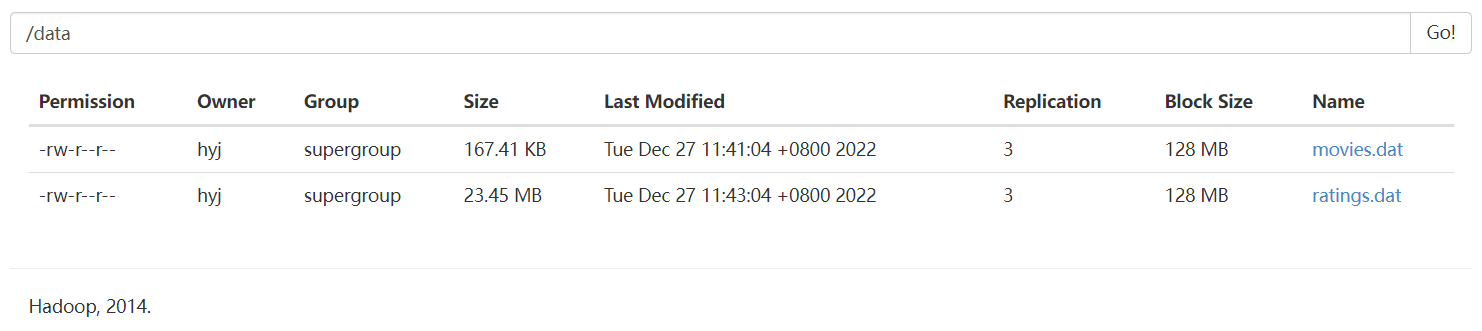


图3.1数据文件上传到HDFS环境

接下来在scala集群环境的 shell中编程导入数据

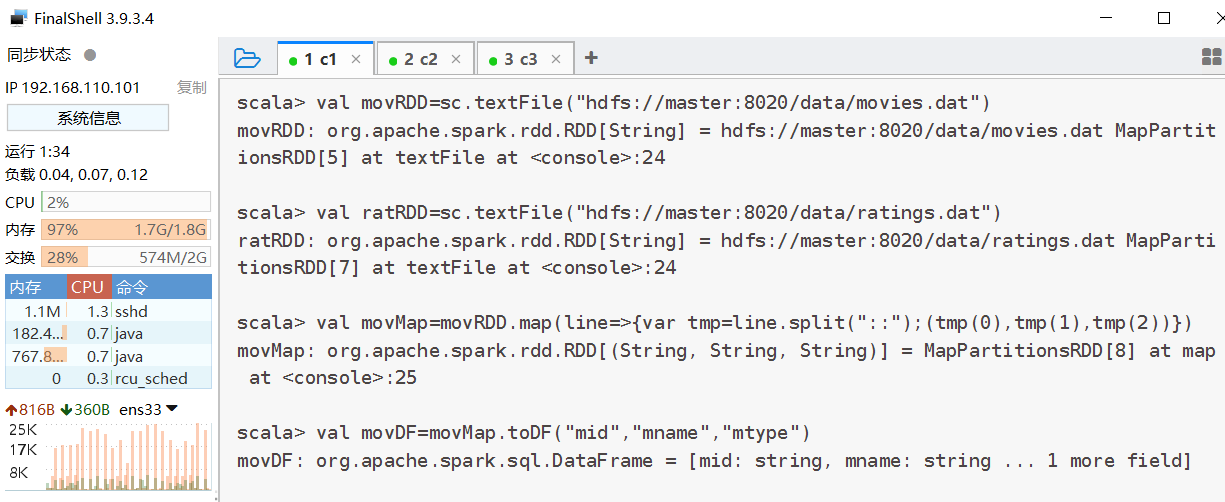


图3.2 shell中编程导入数据

经过映射、分片后，将rating.dat下的评分改为int型，生成movDF和ratDF

在shell中查询导入的数据

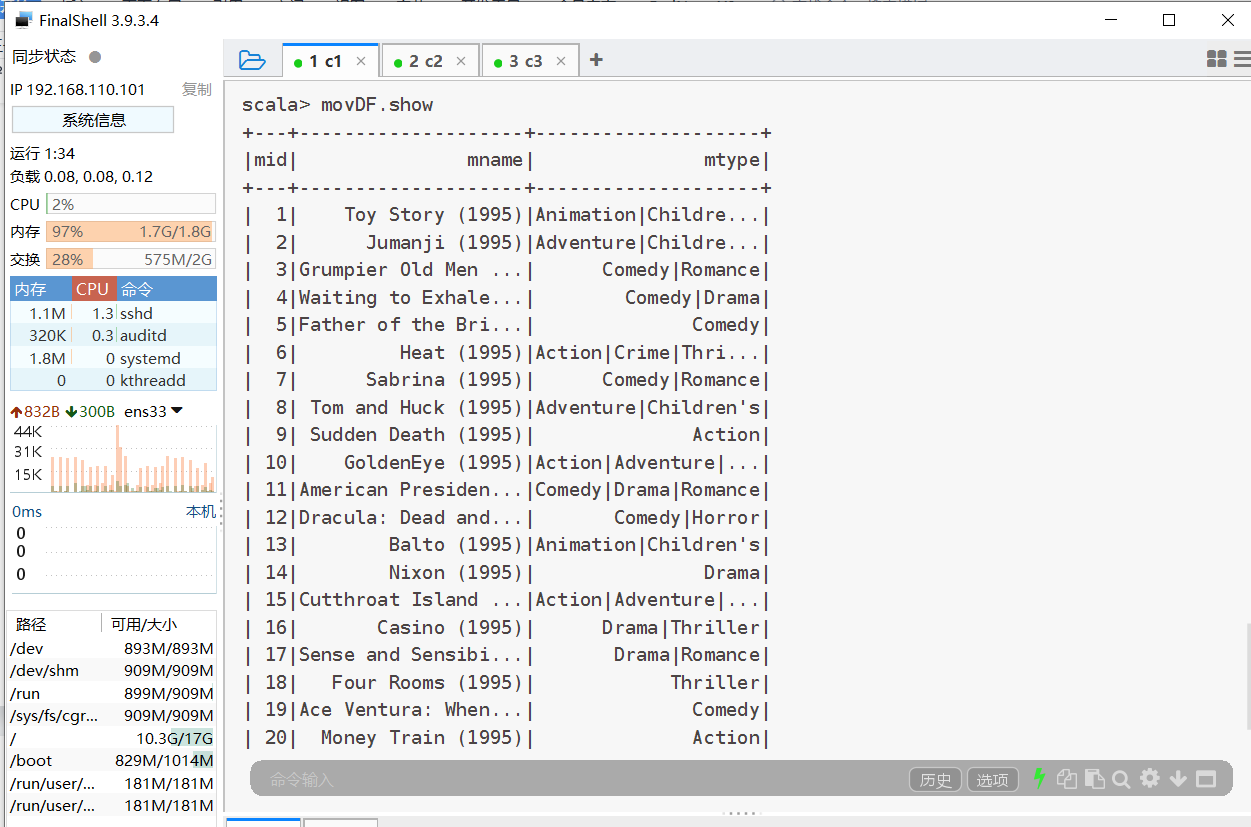


图3.3 movies.dat下的数据

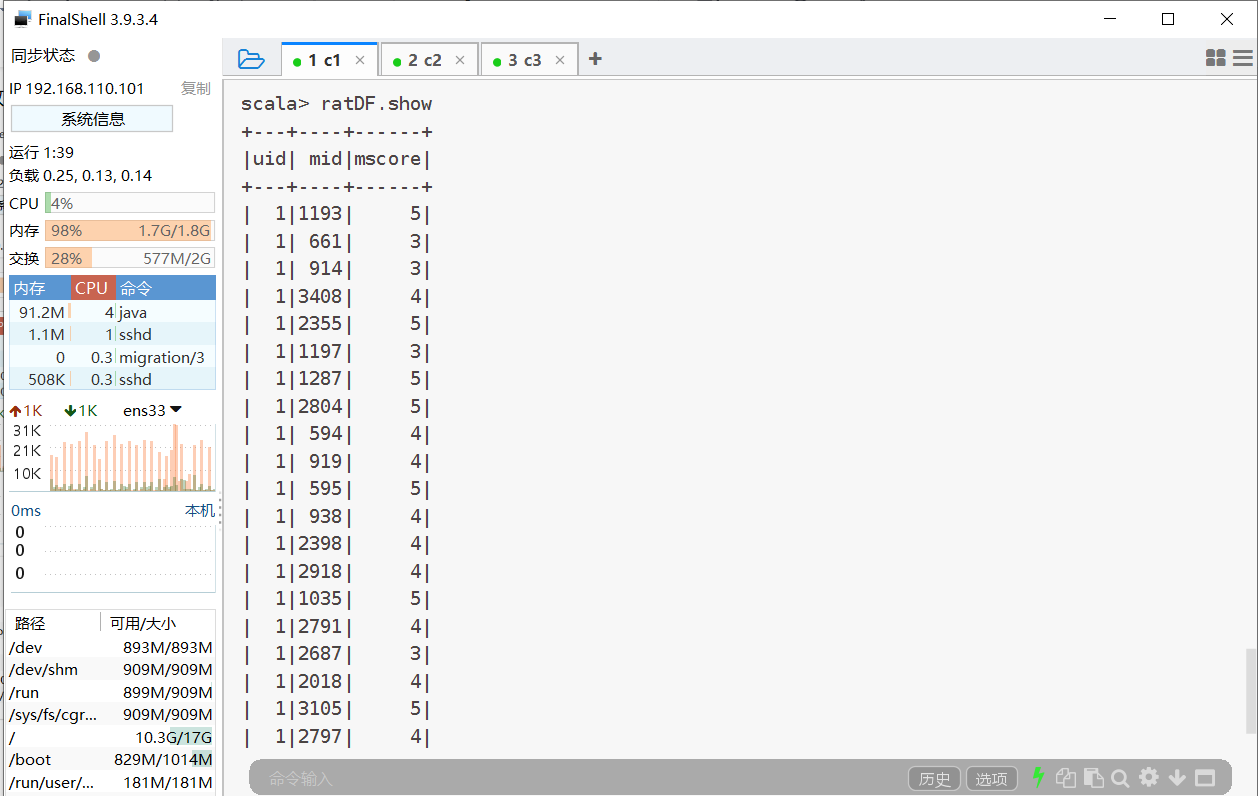


图3.4 ratings.dat下的数据

之后生成两个临时表rating和movie，使用sql语句执行查询评分数量超过20条的并计算平均分，将结果保存在result中

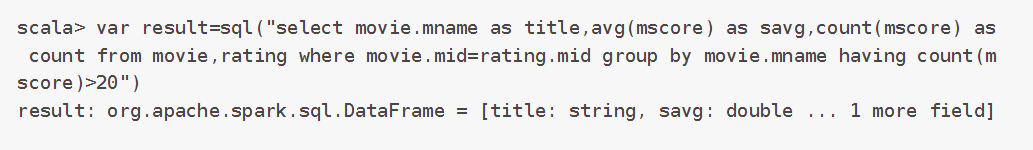


图3.5执行SQL查询

之后将result重新建立临时表res，并将结果按降序输出，得到rt

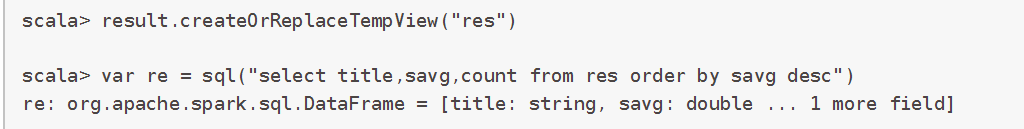


图3.6将结果按降序排序

最后展示结果

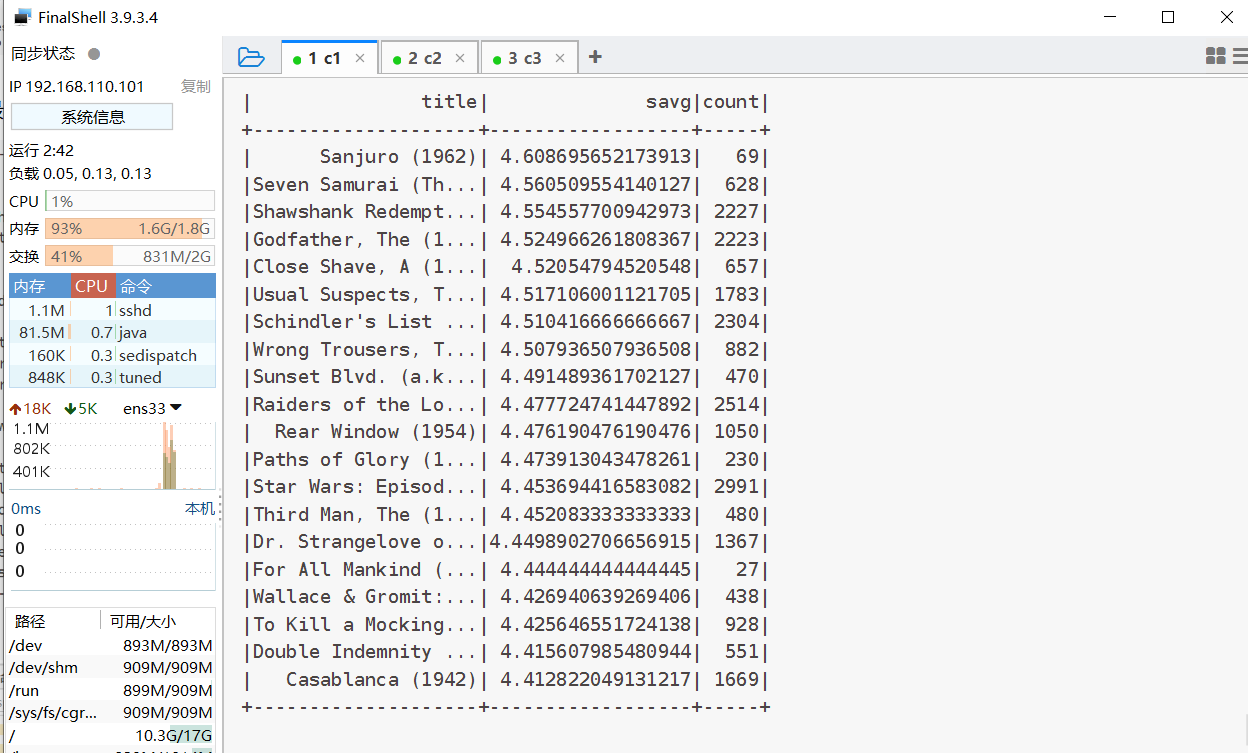


图3.7得到评分数超20且平均分最高的20部影片

# 三、实验总结

（可以总结实验中出现的问题以及解决的思路，也可以列出没有解决的问题）

1. 在IDEA中，依赖文件中scalal版本号要适配。
2. 在spark的shell中，每次执行完sql查询后生成的文件并不能直接用于下一次查询，而是要先建立临时表或者全局表。
3. 因为后面要进行平均数的计算和排序，所以在导入datings数据时，要将评分列的类型从默认的string改为int。