

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

```
[hyj@master soft]$ tar xvf spark-2.4.6-bin-hadoop2.6.tgz -C ~/
tar: 选项“-Aru”与“-f -”不兼容
请用“tar --help”或“tar --usage”获得更多信息。
[hyj@master soft]$ tar xvf spark-2.4.6-bin-hadoop2.6.tgz -C ~/
spark-2.4.6-bin-hadoop2.6/
spark-2.4.6-bin-hadoop2.6/bin/
spark-2.4.6-bin-hadoop2.6/bin/pyspark.cmd
spark-2.4.6-bin-hadoop2.6/bin/spark-submit
spark-2.4.6-bin-hadoop2.6/bin/spark-submit.cmd
spark-2.4.6-bin-hadoop2.6/bin/spark-class2.cmd
spark-2.4.6-bin-hadoop2.6/bin/spark-shell2.cmd
spark-2.4.6-bin-hadoop2.6/bin/pyspark2.cmd
```

图 1.1 将 spark 的压缩包解压到根目录

```
[hyj@master ~]$ mv spark-2.4.6-bin-hadoop2.6 spark
[hyj@master ~]$
```

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

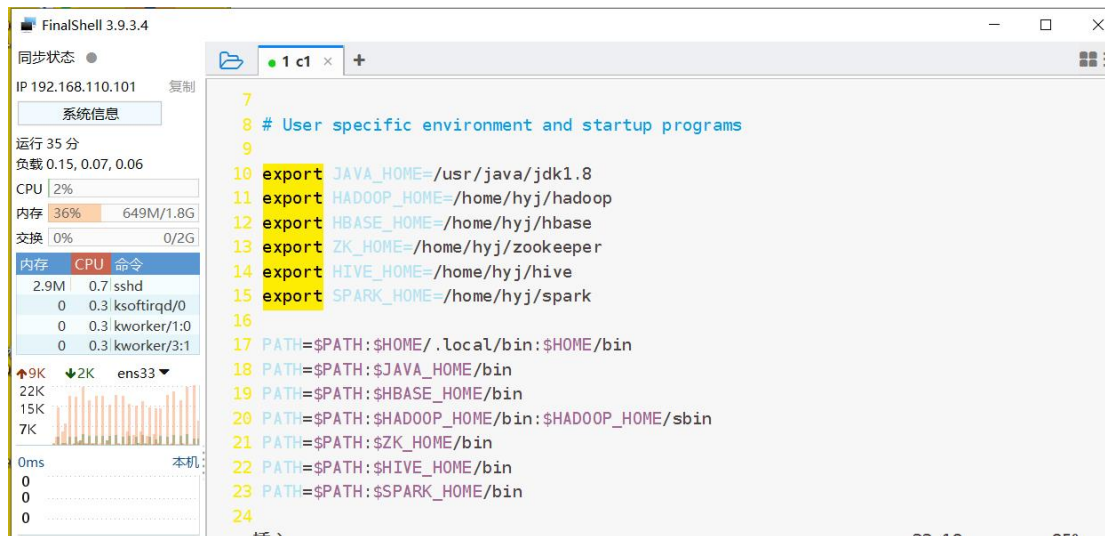


图 1.3 添加 spark 的 bin 环境变量

再配置 spark-env.sh

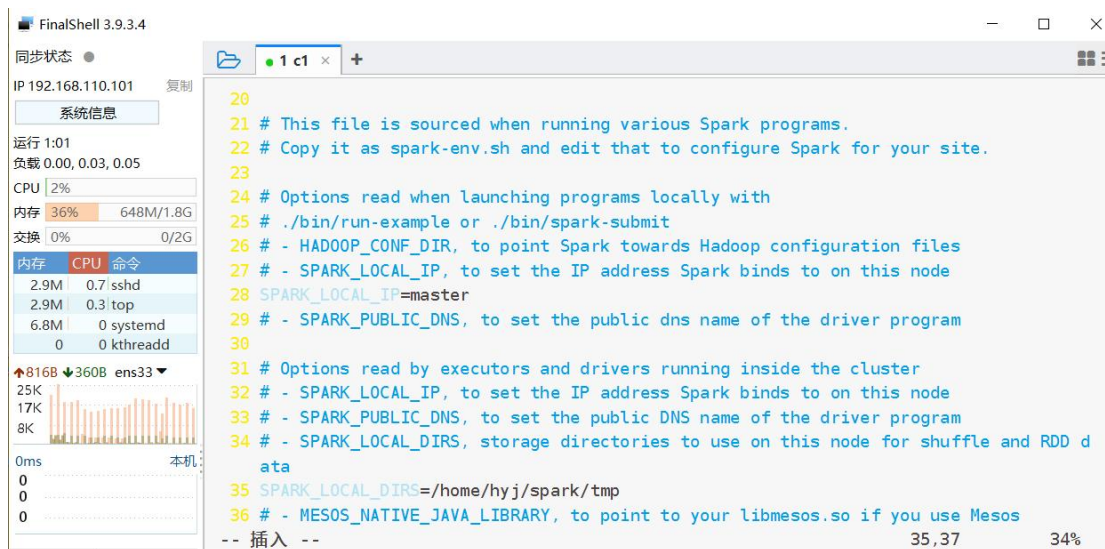


图 1.4 配置 spark-env.sh 文件

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

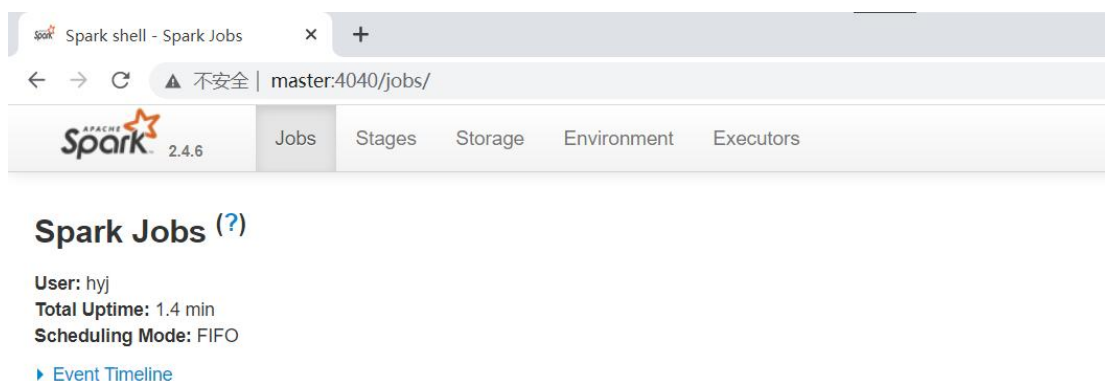


图 1.6 在浏览器上查看 spark 页面

集群模式

先在 slaves 中配置 worker 的数量

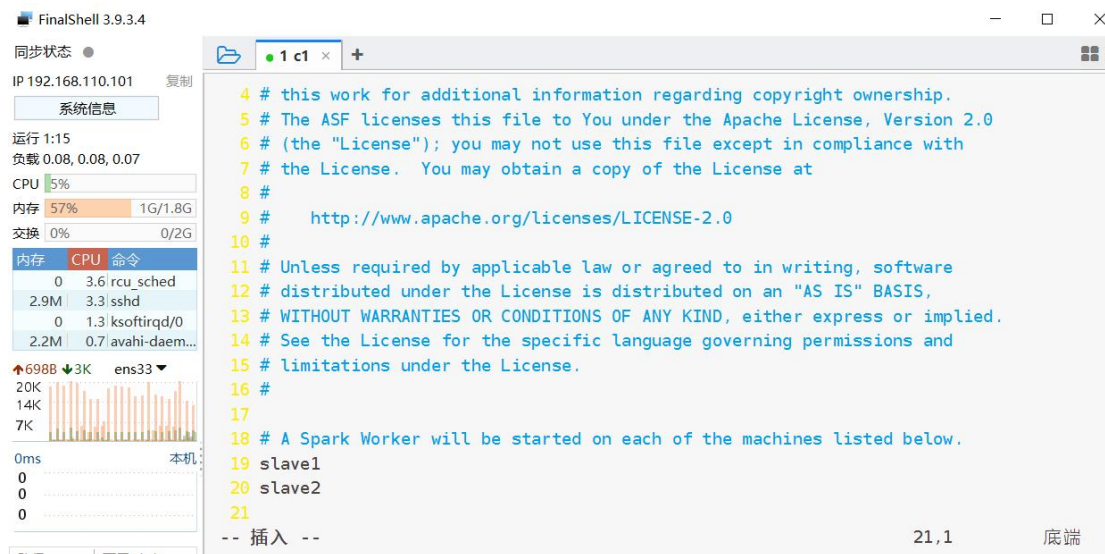


图 1.7 在 slaves 中设置 worker 数量

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

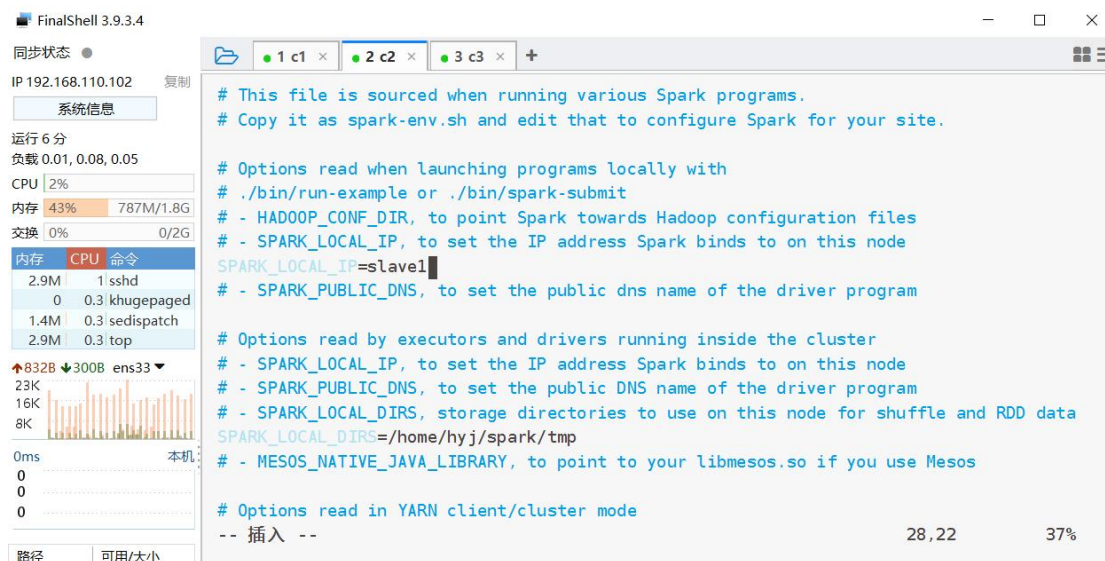


图 1.8 修改 worker 的 local IP

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

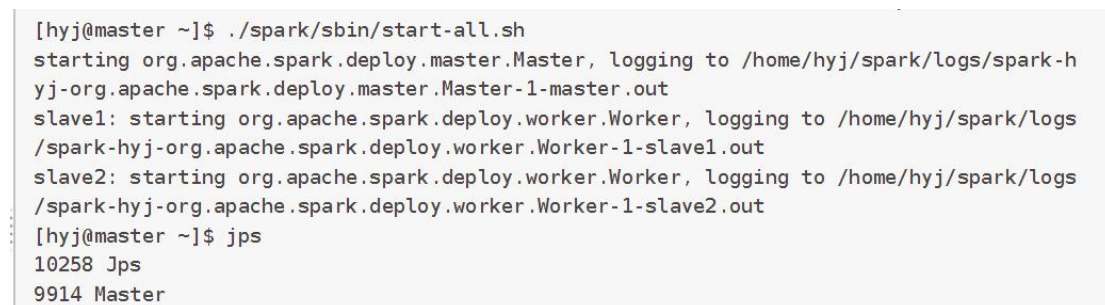


图 1.9 启动集群并查看 master 的 jps


```
[hyj@slave1 conf]$ vi spark-env.sh
[hyj@slave1 conf]$ jps
18195 Jps
17460 Worker
```

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

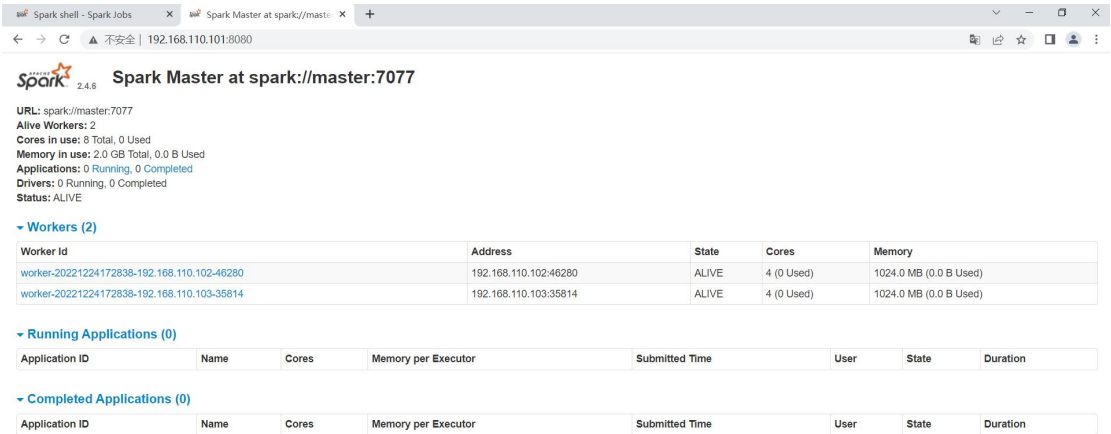


图 1.11 浏览器查看集群监控页面

2. Scala 练习题

使用 Nilakantha 级数算法计算圆周率。
$$\pi = 3 + 4/(2*3*4) - 4/(4*5*6) + 4/(6*7*8) - 4/(8*9*10) + 4/(10*11*12) - 4/(12*13*14) + \dots$$

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

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>2.4.6</version>
</dependency>
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.11</artifactId>
  <version>2.4.6</version>
</dependency>
```

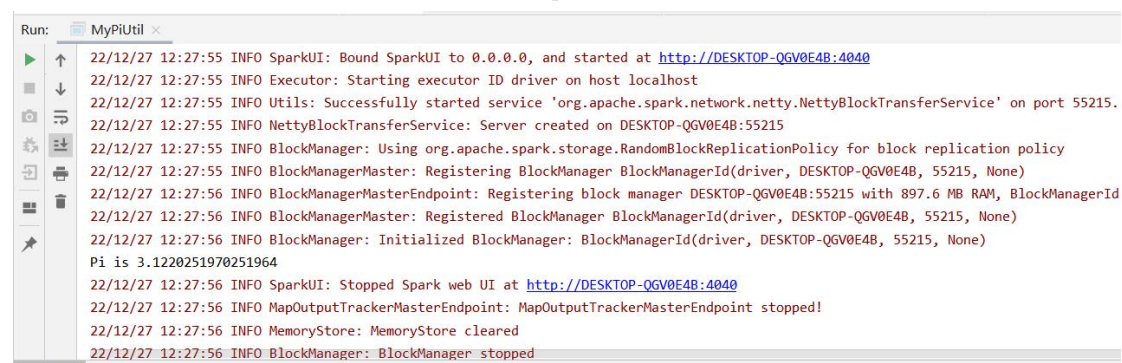
图 2.1 添加 spark 依赖

新建 MyPiUtil.scala 文件

代码如下：

```
1. package edu.nnu
2.
3. import org.apache.spark.SparkConf
4. import org.apache.spark.sql.SparkSession
5.
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



```
Run: MyPiUtil
22/12/27 12:27:55 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://DESKTOP-QGV0E4B:4040
22/12/27 12:27:55 INFO Executor: Starting executor ID driver on host localhost
22/12/27 12:27:55 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 55215.
22/12/27 12:27:55 INFO NettyBlockTransferService: Server created on DESKTOP-QGV0E4B:55215
22/12/27 12:27:55 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
22/12/27 12:27:55 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, DESKTOP-QGV0E4B, 55215, None)
22/12/27 12:27:56 INFO BlockManagerMasterEndpoint: Registering block manager DESKTOP-QGV0E4B:55215 with 897.6 MB RAM, BlockManagerId
22/12/27 12:27:56 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, DESKTOP-QGV0E4B, 55215, None)
22/12/27 12:27:56 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, DESKTOP-QGV0E4B, 55215, None)
Pi is 3.1220251970251964
22/12/27 12:27:56 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-QGV0E4B:4040
22/12/27 12:27:56 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/12/27 12:27:56 INFO MemoryStore: MemoryStore cleared
22/12/27 12:27:56 INFO BlockManager: BlockManager stopped
```

图 2.2 不传参数 pi 的结果

之后传参数 21 得到另一个结果

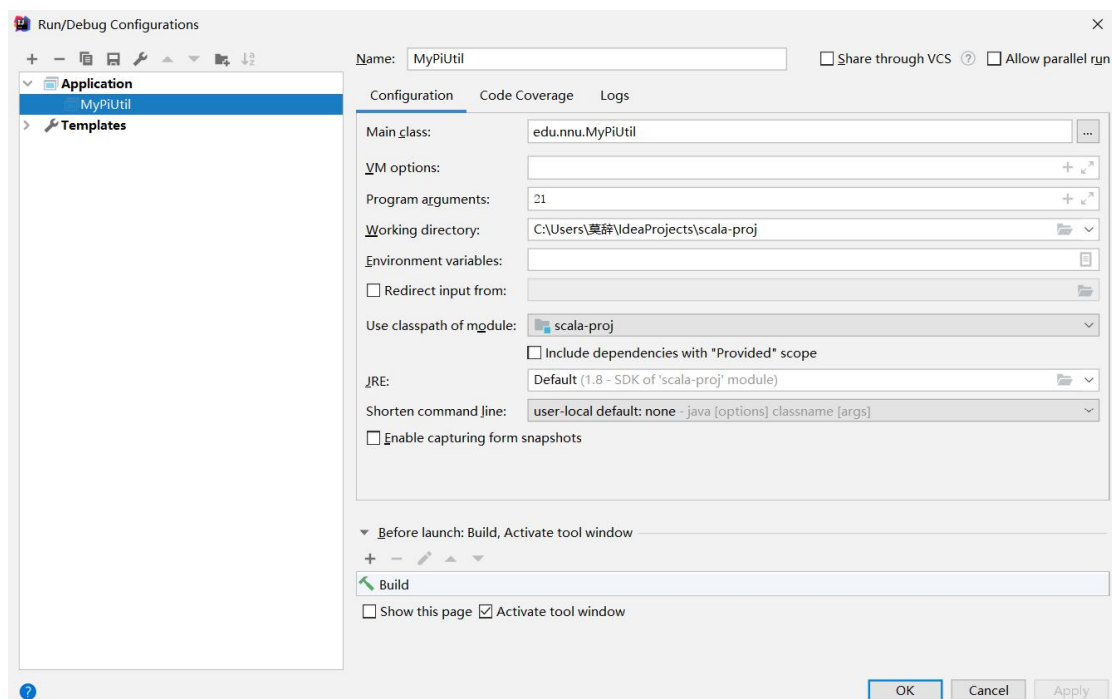


图 2.3 在 Configurations 中设置参数 21

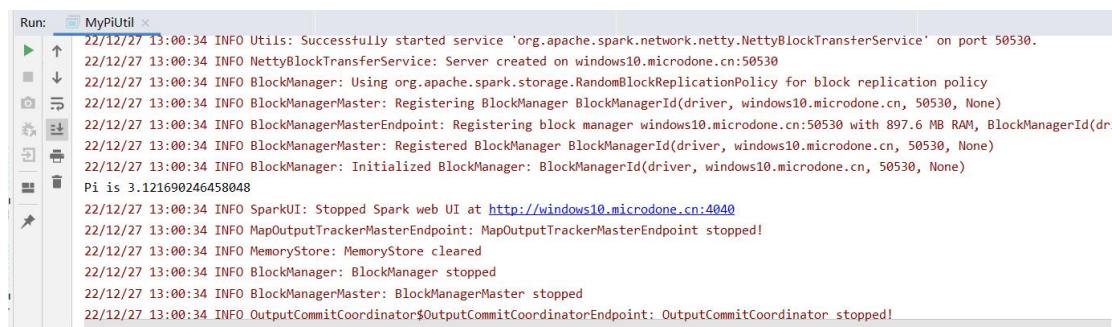


图 2.4 传参 21 时 pi 的结果

将以上代码打包在虚拟机上运行，首先是不传参

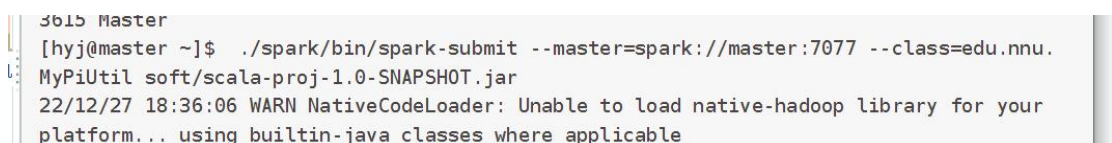


图 2.5 在虚拟机中不传参运行

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

```
[hyj@master ~]$ ./spark/bin/spark-submit --master=spark://master:7077 --class=edu.nnu.
MyPiUtil soft/scala-proj-1.0-SNAPSHOT.jar 21
22/12/27 18:38:08 WARN NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
```

图 2.7 在虚拟机上传参运行

其结果如下：

```
Pi = 3.121690246458048
22/12/27 18:38:20 INFO SparkUI: Stopped Spark web UI at http://master:4040
22/12/27 18:38:20 INFO StandaloneSchedulerBackend: Shutting down all executors
22/12/27 18:38:20 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
22/12/27 18:38:20 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/12/27 18:38:20 INFO MemoryStore: MemoryStore cleared
22/12/27 18:38:20 INFO BlockManager: BlockManager stopped
22/12/27 18:38:20 INFO BlockManagerMaster: BlockManagerMaster stopped
22/12/27 18:38:20 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/12/27 18:38:20 INFO SparkContext: Successfully stopped SparkContext
```

图 2.8 在虚拟机上传参运行结果

3. 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

| /data | | | | | | | Go! |
|------------|-------|------------|-----------|--------------------------------|-------------|------------|-----------------------------|
| Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
| -rw-r--r-- | hyj | supergroup | 167.41 KB | Tue Dec 27 11:41:04 +0800 2022 | 3 | 128 MB | movies.dat |
| -rw-r--r-- | hyj | supergroup | 23.45 MB | Tue Dec 27 11:43:04 +0800 2022 | 3 | 128 MB | ratings.dat |

Hadoop, 2014.

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

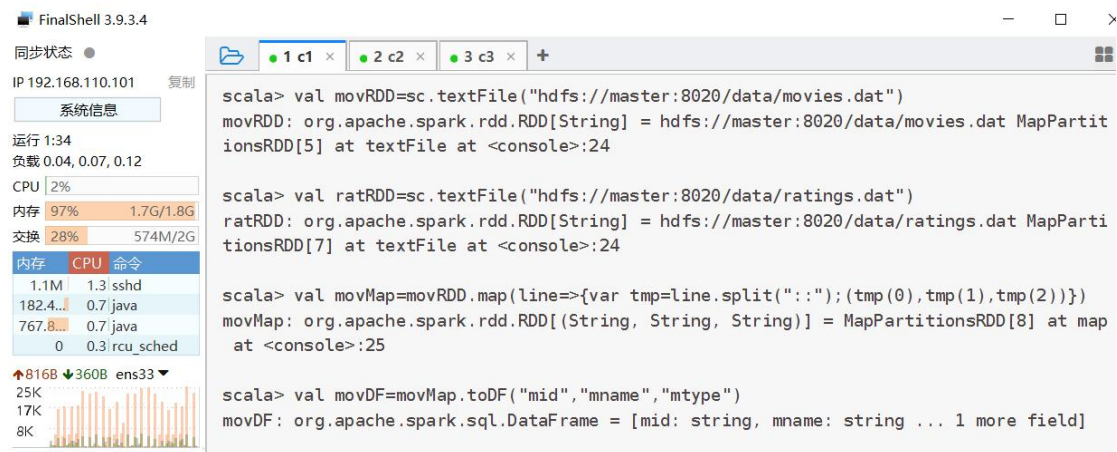


图 3.2 shell 中编程导入数据

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

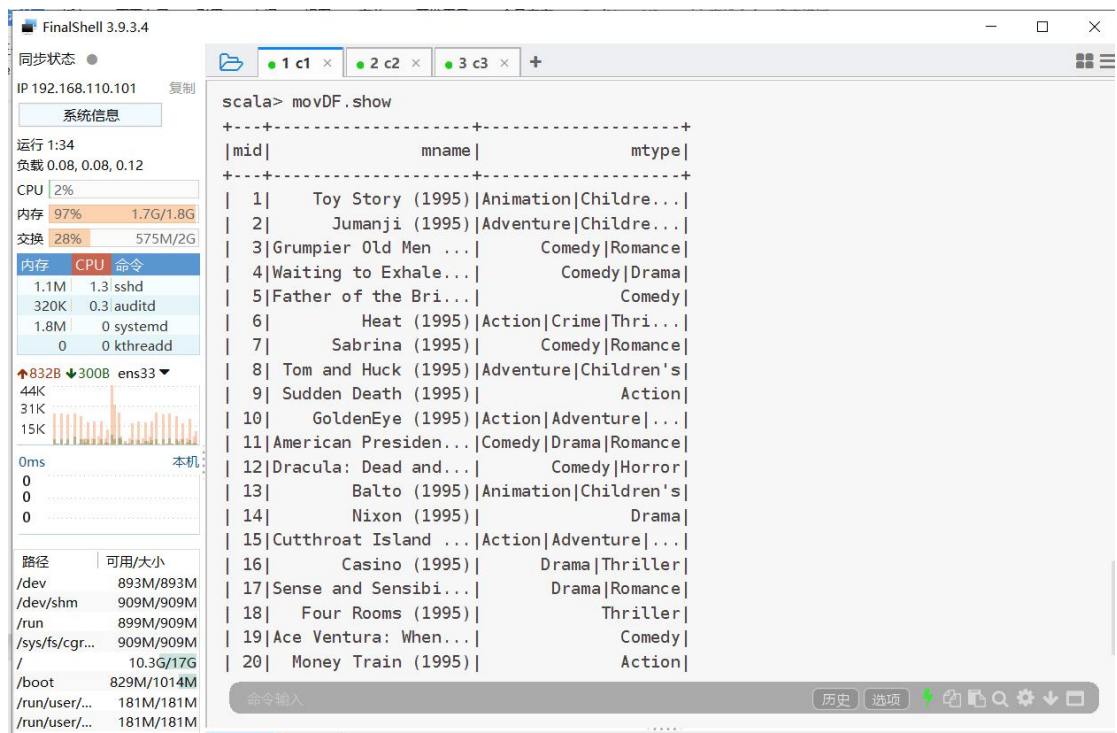


图 3.3 movies.dat 下的数据

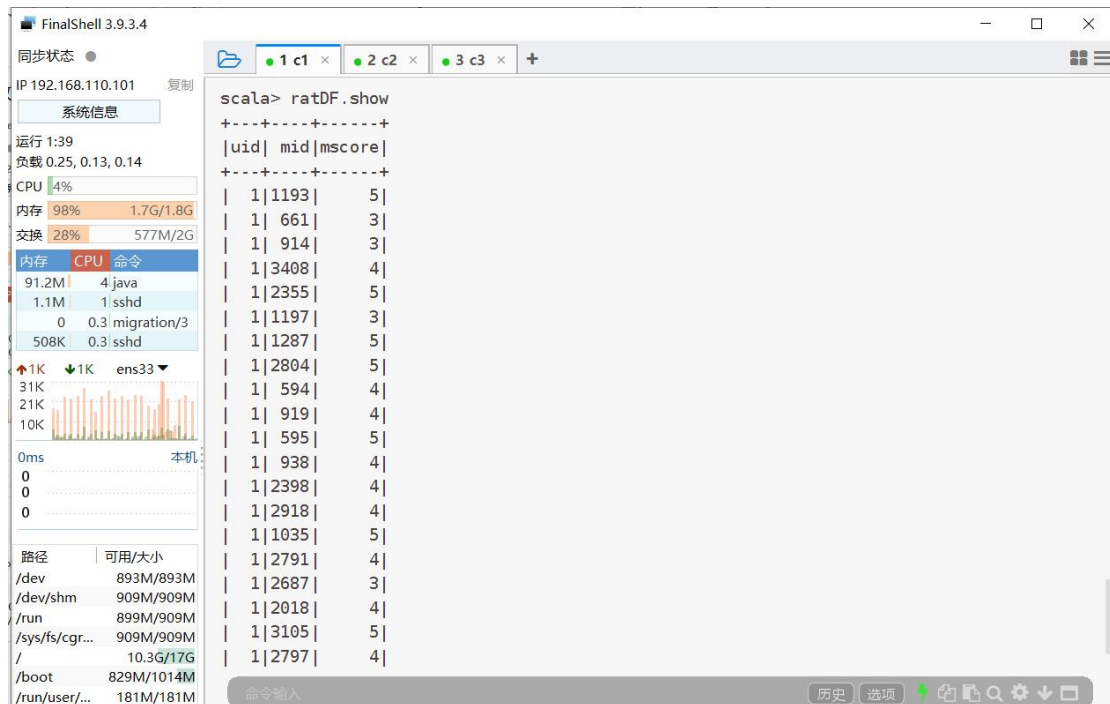


图 3.4 ratings.dat 下的数据

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

```
scala> var result=sql("select movie.mname as title,avg(mscore) as savg,count(mscore) as
count from movie,rating where movie.mid=rating.mid group by movie.mname having count(m
score)>20")
result: org.apache.spark.sql.DataFrame = [title: string, savg: double ... 1 more field]
```

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`

```
scala> result.createOrReplaceTempView("res")

scala> var re = sql("select title,savg,count from res order by savg desc")
re: org.apache.spark.sql.DataFrame = [title: string, savg: double ... 1 more field]
```

图 3.6 将结果按降序排序

最后展示结果

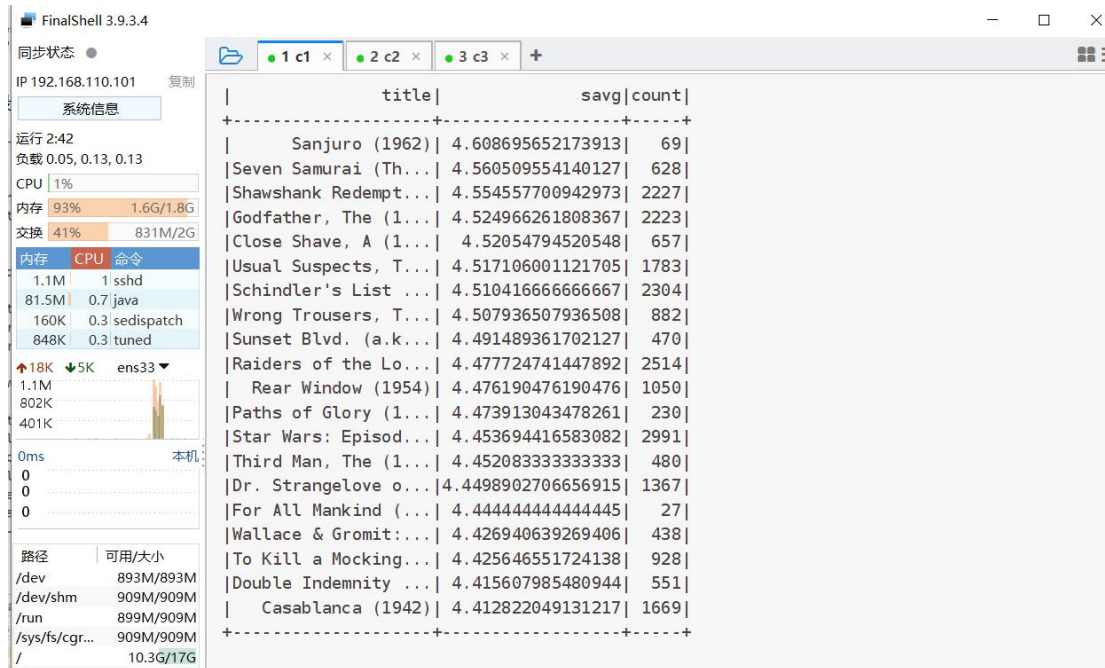


图 3.7 得到评分超 20 且平均分最高的 20 部影

三、实验总结

(可以总结实验中出现的問題以及解决思路,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。