

# 实验三：大数据离线数据分析实践-MR+Spark

刘帅 2020212267

## 实验内容

本次实验共分为三部分：

- 第一部分为基于Hadoop 集群的MapReduce 大数据批处理实践；
- 第二部分为Spark 的单机伪分布式集群搭建，以及基于该Spark 集群的大数据批处理实践；
- 第三部分为Spark 的完全分布式集群搭建，以及基于该Spark 集群的大数据批处理实践；

## 一、hadoop mapreduce wordcount程序案例再现

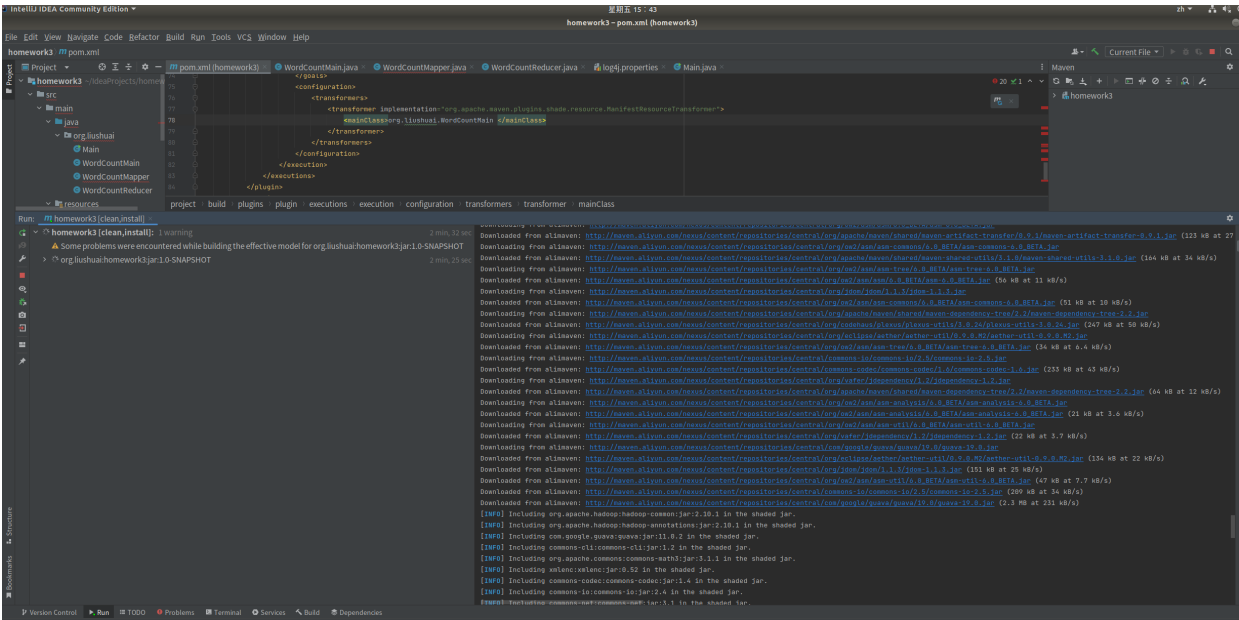
### 1.1 java项目创建与环境依赖配置

首先根据实验二中的方法创建java项目，并进行maven配置，随后在pom.xml中导入hadoop、slf4j、httpcomponents等相关依赖；

执行

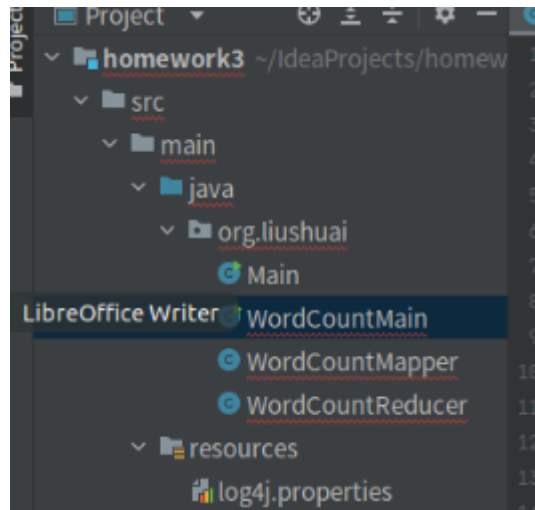
```
mvn clean install
```

将依赖下载到project当中



### 1.2 编写mapper、reducer、main代码

创建WordCountMain、WordCountMapper、WordCountReducer 三个类



其中:

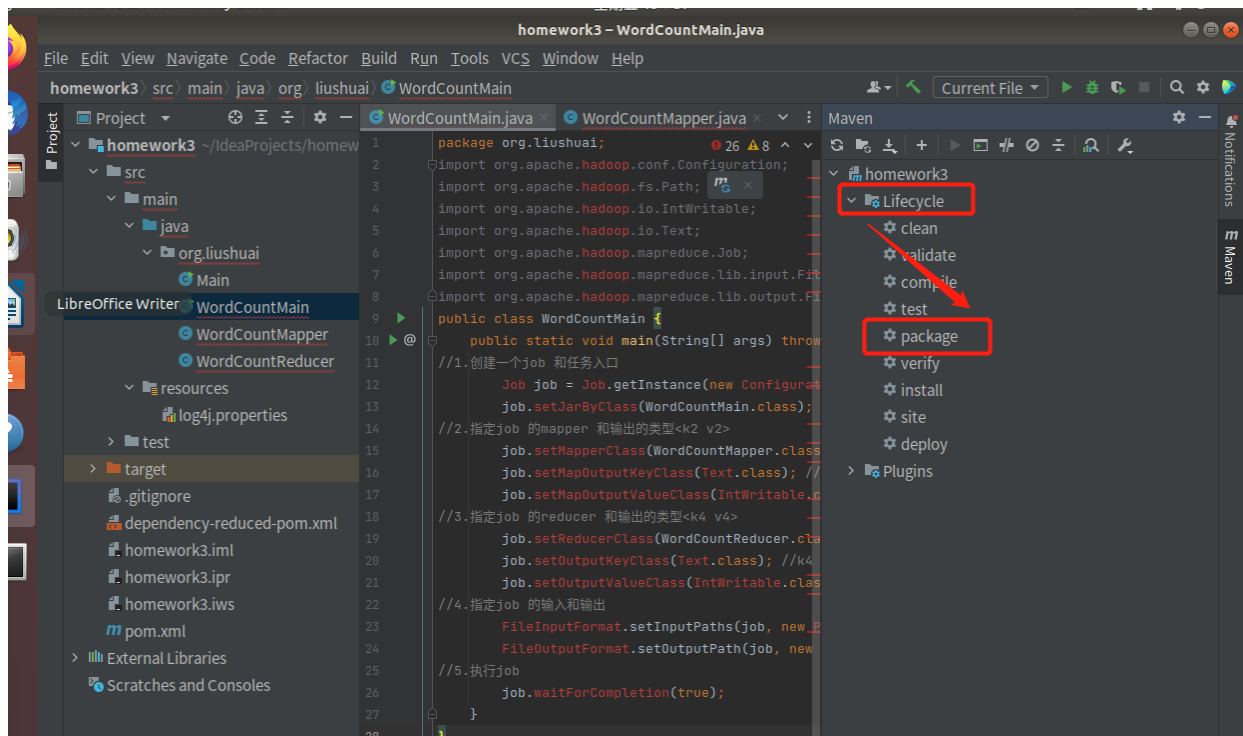
**WordCountMain** 是 MapReduce 作业的主程序，它负责设置作业的各种参数，例如输入路径、输出路径、Mapper 类、Reducer 类等，并启动作业的执行。

**WordCountMapper** 是 MapReduce 作业中的 Mapper 组件，它负责将输入数据拆分为若干个键值对，并对每个键值对执行一次映射操作。在词频统计的场景下，WordCountMapper 的任务是将输入的文本数据拆分为单词和出现次数的键值对，其中键是单词，值是 1。

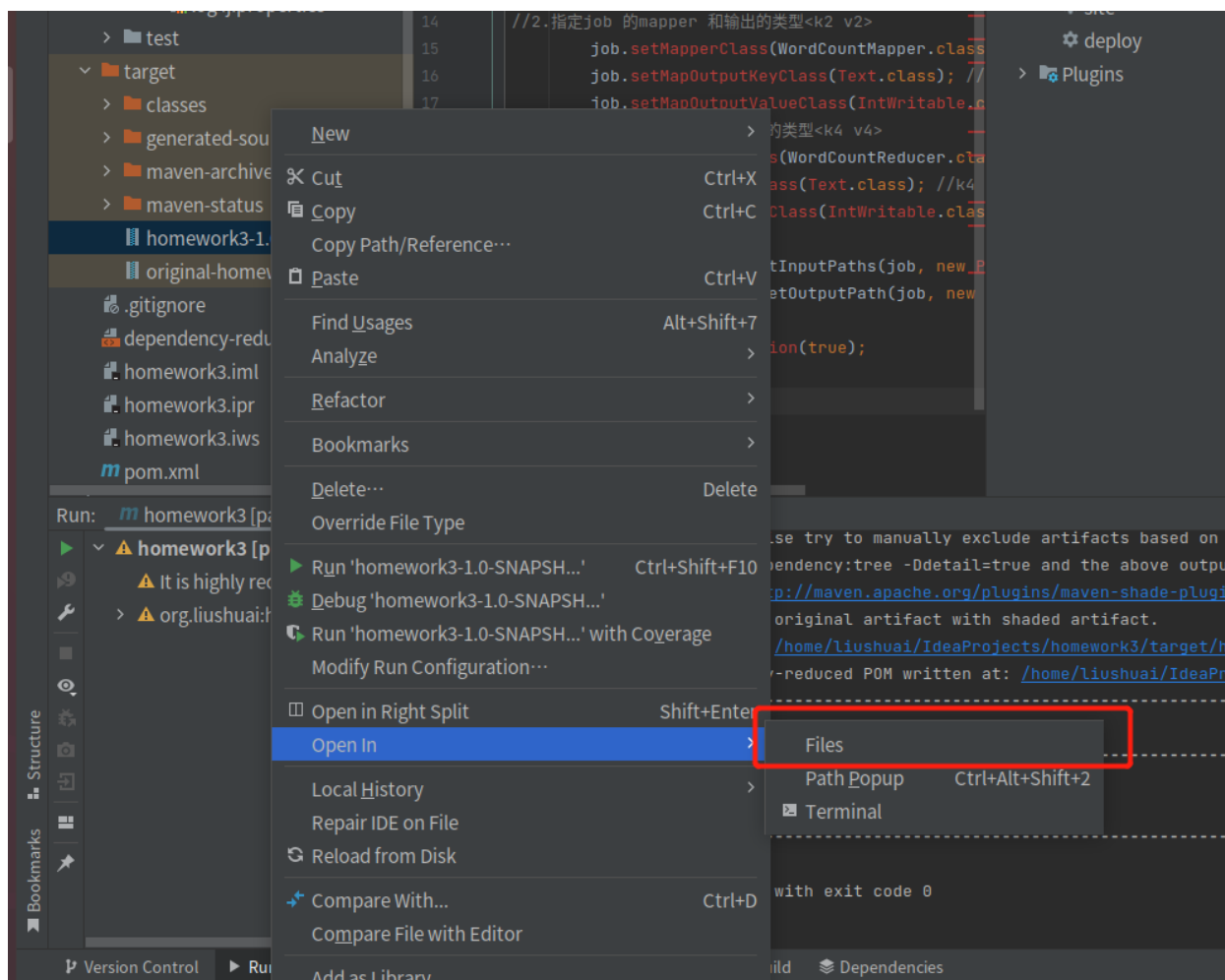
**WordCountReducer** 是 MapReduce 作业中的 Reducer 组件，它负责对 Mapper 的输出进行合并和归约，并生成最终的输出。在词频统计的场景下，wordCountReducer 的任务是将相同单词的键值对进行合并，并将每个单词出现的次数累加，最终输出每个单词出现的总次数。

### 1.3 将程序打包成jar包

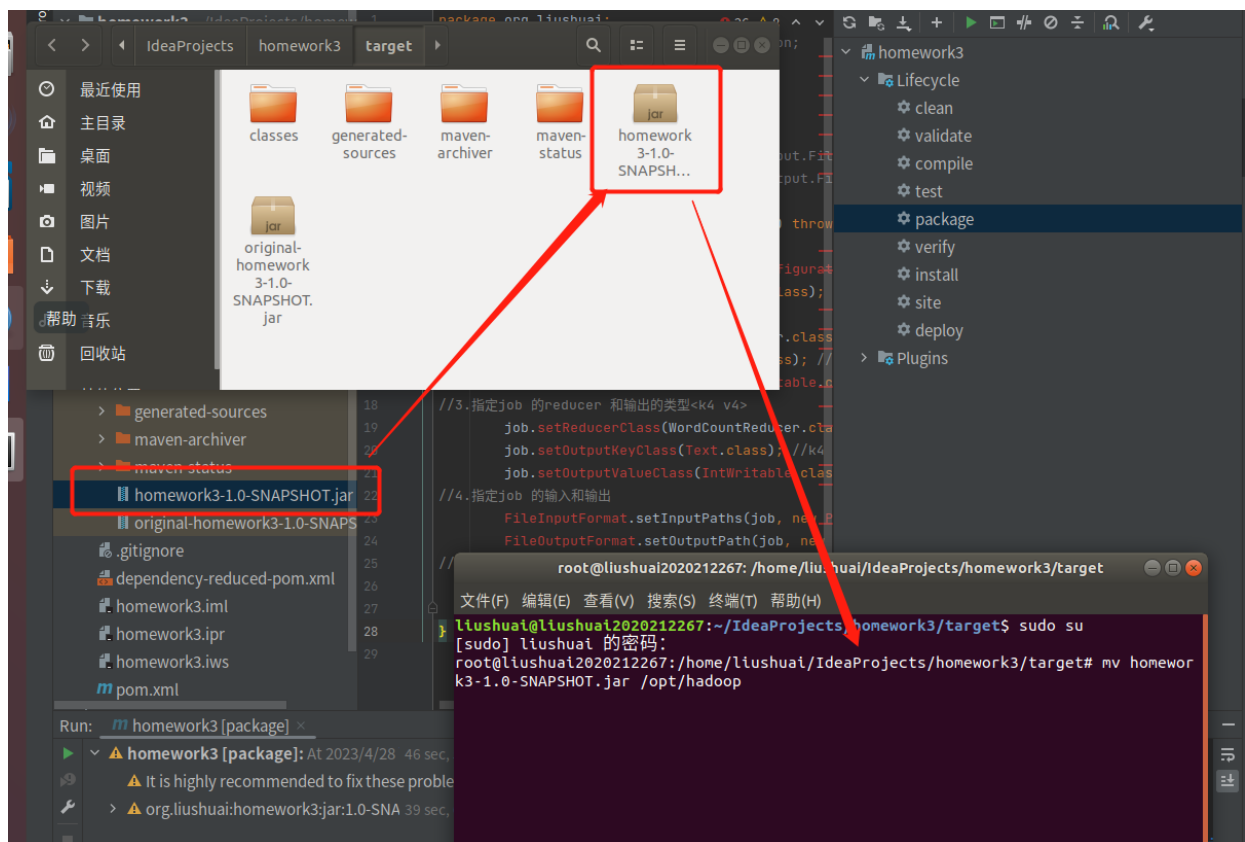
双击Lifecycle 中的package 进行打包



经过maven 的编译打包，即可在我们项目路径的target 目录下生成项目的jar 包。并把这个压缩包在文件系统中打开



可以看到，这个homework3的snapshot包存在于target目录下，我们需要将其移动至opt/hadoop



#### 1.4 执行mapreduce任务，进行词频统计

首先启动hadoop程序，下载测试文件

```
root@liushuai2020212267:/opt/hadoop# start-yarn.sh
starting yarn daemons
starting resource manager, logging to /opt/hadoop/logs/yarn-root-resourcemanager-
liushuai2020212267.out
localhost: starting nodemanager, logging to /opt/hadoop/logs/yarn-root-nodemanager-
liushuai2020212267.out
root@liushuai2020212267:/opt/hadoop# start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /opt/hadoop/logs/hadoop-root-namenode-l
iushuai2020212267.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-root-datanode-l
iushuai2020212267.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-root-sec
ondarynamenode-liushuai2020212267.out
root@liushuai2020212267:/opt/hadoop#
```

将测试文件上传至hdfs 的根目录下。

```
ceshi.txt  include  LICENSE.txt  README.txt  tmp
etc        lib        logs        sbin
root@liushuai2020212267:/opt/hadoop# hadoop fs -put /opt/hadoop/ceshi.txt /
root@liushuai2020212267:/opt/hadoop# hadoop fs -ls /
Found 2 items
-rw-r--r--  1 root supergroup    6747 2023-04-28 16:07 /ceshi.txt
drwxr-xr-x  - root supergroup      0 2023-04-25 17:03 /hbase
root@liushuai2020212267:/opt/hadoop#
```

执行mapreduce任务，出现如下日志信息。

```

root@liushuai2020212267:/opt/hadoop# hadoop jar homework3-1.0-SNAPSHOT.jar /ceshi.txt /MRoutput
23/04/28 16:10:12 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
23/04/28 16:10:13 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface
and execute your application with ToolRunner to remedy this.
23/04/28 16:10:15 INFO input.FileInputFormat: Total input files to process : 1
23/04/28 16:10:16 INFO mapreduce.JobSubmitter: number of splits:1
23/04/28 16:10:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1682669079602_0002
23/04/28 16:10:18 INFO conf.Configuration: resource-types.xml not found
23/04/28 16:10:18 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
23/04/28 16:10:18 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
23/04/28 16:10:18 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
23/04/28 16:10:19 INFO impl.YarnClientImpl: Submitted application application_1682669079602_0002
23/04/28 16:10:19 INFO mapreduce.Job: The url to track the job: http://liushuai2020212267:8088/proxy/application_1682669079602_0002/
23/04/28 16:10:19 INFO mapreduce.Job: Running job: job_1682669079602_0002
23/04/28 16:10:47 INFO mapreduce.Job: Job job_1682669079602_0002 running in uber mode : false
23/04/28 16:10:47 INFO mapreduce.Job: map 0% reduce 0%
23/04/28 16:11:00 INFO mapreduce.Job: map 100% reduce 0%
23/04/28 16:11:14 INFO mapreduce.Job: map 100% reduce 100%
23/04/28 16:11:15 INFO mapreduce.Job: Job job_1682669079602_0002 completed successfully
23/04/28 16:11:16 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=18754
  FILE: Number of bytes written=454361
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=6843
  HDFS: Number of bytes written=44
  HDFS: Number of read operations=6
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=10349
  Total time spent by all reduces in occupied slots (ms)=10374
  Total time spent by all map tasks (ms)=10349
  Total time spent by all reduce tasks (ms)=10374
  Total vcore-milliseconds taken by all map tasks=10349
  Total vcore-milliseconds taken by all reduce tasks=10374
  Total megabyte-milliseconds taken by all map tasks=10597376
  Total megabyte-milliseconds taken by all reduce tasks=10622976
Map-Reduce Framework
  Map input records=1
  Map output records=2000
  Map output bytes=14748
  Map output materialized bytes=18754
  Input split bytes=96
  Combine input records=0
  Combine output records=0
  Reduce input groups=6
  Reduce shuffle bytes=18754
  Reduce input records=2000
  Reduce output records=6
  Spilled Records=4000
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=622
  CPU time spent (ms)=4040
  Physical memory (bytes) snapshot=495661056

```

**INFO mapreduce.Job: map 0% reduce 0%**表示作业刚开始执行时，Map 和 Reduce 阶段的进度都为 0%。

**INFO mapreduce.Job: map 100% reduce 0%**表示 Map 阶段已经完成，进度为 100%，而 Reduce 阶段还没有开始执行，进度仍然为 0%。这说明 Map 阶段已经将输入数据全部处理完毕，并将结果输出到 HDFS 中的临时文件中。

**INFO mapreduce.Job: map 100% reduce 100%** 表示整个作业已经完成，Map 和 Reduce 阶段的进度都已经达到了 100%。这说明 Reduce 阶段已经将 Map 阶段输出的临时结果进行了合并、排序等操作，并将最终结果输出到了指定的目录中。

## 1.5 查看执行命令后输出的结果

输出结果在hdfs 的MRoutput 文件夹下

```

root@liushuai2020212267:/opt/hadoop# hadoop fs -ls /
Found 4 items
drwxr-xr-x  - root supergroup          0 2023-04-28 16:11 /MRoutput
-rw-r--r--  1 root supergroup          6747 2023-04-28 16:07 /ceshi.txt
drwxr-xr-x  - root supergroup          0 2023-04-25 17:03 /hbase
drwx----- - root supergroup          0 2023-04-28 16:08 /tmp
root@liushuai2020212267:/opt/hadoop#

```

查看结果：

```
hadoop fs -cat /MRoutput/part-r-00000
```

```
root@liushuai2020212267:/opt/hadoop# hadoop fs -ls /
Found 4 items
drwxr-xr-x   - root supergroup          0 2023-04-28 16:11 /MRoutput
-rw-r--r--   1 root supergroup      6747 2023-04-28 16:07 /ceshi.txt
drwxr-xr-x   - root supergroup          0 2023-04-25 17:03 /hbase
drwx----- - root supergroup          0 2023-04-28 16:08 /tmp
root@liushuai2020212267:/opt/hadoop# hadoop fs -cat /MRoutput/part-r-00000
a          346
b          325
c          335
d          307
hello     346
world     341
root@liushuai2020212267:/opt/hadoop#
```

## 二、Spark 伪分布环境配置

### 2.1 安装scala并解压

```
root@liushuai2020212267:/opt/hadoop# cd ~
root@liushuai2020212267:~# wget https://downloads.lightbend.com/scala/2.12.15/scala-2.12.15.tgz
--2023-04-28 16:14:15-- https://downloads.lightbend.com/scala/2.12.15/scala-2.12.15.tgz
正在解析主机 downloads.lightbend.com (downloads.lightbend.com)... 54.192.18.53, 54.192.18.34, 54.192.18.74, ...
正在连接 downloads.lightbend.com (downloads.lightbend.com)[54.192.18.53]:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 21087658 (20M) [application/octet-stream]
正在保存至：“scala-2.12.15.tgz”

scala-2.12.15.tgz 100%[=====] 20.11M 5.81MB/s 用时 3.5s

2023-04-28 16:14:20 (5.81 MB/s) - 已保存 “scala-2.12.15.tgz” [21087658/21087658]

root@liushuai2020212267:~# tar -zxvf scala-2.12.15.tgz scala-2.12.15/
scala-2.12.15/
scala-2.12.15/lib/
scala-2.12.15/lib/scala-compiler.jar
scala-2.12.15/lib/scalap-2.12.15.jar
scala-2.12.15/lib/scala-reflect.jar
scala-2.12.15/lib/scala-xml_2.12-1.0.6.jar
scala-2.12.15/lib/jline-2.14.6.jar
scala-2.12.15/lib/scala-parser-combinators_2.12-1.0.7.jar
scala-2.12.15/lib/scala-swing_2.12-2.0.3.jar
scala-2.12.15/lib/scala-library.jar
scala-2.12.15/doc/
scala-2.12.15/doc/licenses/
scala-2.12.15/doc/licenses/mit_jquery.txt
scala-2.12.15/doc/licenses/bsd_asm.txt
scala-2.12.15/doc/licenses/apache_jansi.txt
scala-2.12.15/doc/licenses/bsd_jline.txt
scala-2.12.15/doc/licenses/mit_tools.tooltip.txt
scala-2.12.15/doc/LICENSE.md
scala-2.12.15/doc/License.rtf
scala-2.12.15/doc/README
scala-2.12.15/doc/tools/
scala-2.12.15/doc/tools/scaladoc.html
scala-2.12.15/doc/tools/scalap.html
scala-2.12.15/doc/tools/css/
scala-2.12.15/doc/tools/css/style.css
```

将scala移动至usr目录，并更改环境变量，执行

```
scala -version
```

查看scala是否安装成功



```

scala-2.12.15/NOTICE
root@liushuai2020212267:~# mv scala-2.12.15 /usr/scala-2.12.15
root@liushuai2020212267:~# cd /usr
root@liushuai2020212267:/usr# ls
bin    include  lib      local  scala-2.12.15  src
games  java8    libexec  sbin   share
root@liushuai2020212267:/usr# cd scala-2.12.15/
root@liushuai2020212267:/usr/scala-2.12.15# ls
bin  doc  lib  LICENSE  man  NOTICE
root@liushuai2020212267:/usr/scala-2.12.15#

root@liushuai2020212267:/usr/scala-2.12.15# vim /etc/profile
root@liushuai2020212267:/usr/scala-2.12.15# source /etc/profile
root@liushuai2020212267:/usr/scala-2.12.15# scala -version
Scala code runner version 2.12.15 -- Copyright 2002-2021, LAMP/EPFL and Lightben
d, Inc.
root@liushuai2020212267:/usr/scala-2.12.15#

```

## 2.2 安装spark

安装spark，并对文件进行解压，移动至/opt目录中

```

root@liushuai2020212267:~# ^C
root@liushuai2020212267:~# tar -zxvf spark-3.2.3-bin-without-hadoop.tgz
root@liushuai2020212267:~# ls
openjdk-8u41-b04-linux-x64-14_jan_2020.tar.gz
scala-2.12.15.tgz
snap
spark-3.2.3-bin-without-hadoop
spark-3.2.3-bin-without-hadoop.tgz
tmp
root@liushuai2020212267:~# mv spark-3.2.3-bin-without-hadoop /opt/spark
root@liushuai2020212267:~#

```

至此，hadoop,hbase,spark的源文件全部置于/opt文件中

```

root@liushuai2020212267:~# cd /opt
root@liushuai2020212267:/opt# ls
hadoop  hbase  spark  vmware-tools-distrib
root@liushuai2020212267:/opt#

```

## 2.3 配置环境变量

```

fi
export JAVA_HOME=/usr/java8
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/opt/hadoop/
export PATH=$PATH:$HADOOP_HOME/bin
export HBASE_HOME=/opt/hbase
export PATH=$PATH:$HBASE_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export PATH=~/apache-maven-3.8.8/bin:$PATH
export SCALA_HOME=/usr/scala-2.12.15
export PATH=$PATH:$SCALA_HOME/bin
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin

```

## 2.4 配置spark文件

```
root@liushuai2020212267:/opt# vi /etc/profile
root@liushuai2020212267:/opt# source /etc/profile
root@liushuai2020212267:/opt# cp /opt/spark/conf/spark-env.sh.template /opt/spark/conf/spark-env.sh
root@liushuai2020212267:/opt# vi /opt/spark/conf/spark-env.sh
```

配置spark-env.sh文件

```
# - OPENBLAS_NUM_THREADS=1  Disable multi-threading of OpenBLAS
export JAVA_HOME=/usr/java8
export SCALA_HOME=/usr/scala-2.12.15
export SPARK_DIST_CLASSPATH=$(/opt/hadoop/bin/hadoop classpath)
export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop
export SPARK_MASTER_IP=192.168.1.14
export SPARK_MASTER_PORT=7077
export SPARK_MASTER_WEBUI_PORT=8080
:wq
```

启动spark

```
cd /opt/spark
sbin/start-all.sh
```

执行jps命令

```
root@liushuai2020212267:/opt/spark# sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-root-org.apache.spark.deploy.master.Master-1-liushuai2020212267.out
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-liushuai2020212267.out
root@liushuai2020212267:/opt/spark# jps
3410 ResourceManager
4418 SecondaryNameNode
3570 NodeManager
2163 -- process information unavailable
2756 -- process information unavailable
7878 Worker
3862 NameNode
2870 -- process information unavailable
3994 DataNode
7981 Jps
7742 Master
root@liushuai2020212267:/opt/spark#
```

出现worker和master两进程，其中，master进程负责管理整个 Spark 集群的资源 and 作业调度等工作。而 worker 进程主要负责在本地节点上启动和管理 Spark Executor 进程，执行具体的计算任务。

## 2.5 前端可视化

打开localhost:8080, web 界面显示有一个节点，配置成功



Cores in use: 2 Total, 0 Used  
Memory in use: 6.7 GiB Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

#### Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20230428164822-192.168.1.14-37287	192.168.1.14:37287	ALIVE	2 (0 Used)	6.7 GiB (0.0 B Used)

#### Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State
----------------	------	-------	---------------------	------------------------	----------------	------	-------

#### Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State
----------------	------	-------	---------------------	------------------------	----------------	------	-------

## 2.6 用自带样例进行测试

```
bin/spark-submit
--master spark://主机名:7077
--class org.apache.spark.examples.SparkPi
/opt/spark/examples/jars/spark-examples_2.12-3.2.3.jar 10
```

在 Spark 集群上提交 Spark 应用程序的命令，其中，

- `bin/spark-submit` 是提交 Spark 应用程序的命令。
- `--master spark://主机名:7077` 指定 Spark 应用程序将运行在名为 "主机名" 的 Spark 主节点上，并且主节点的端口为 7077。
- `--class org.apache.spark.examples.SparkPi` 指定 Spark 应用程序的入口类为 `org.apache.spark.examples.SparkPi`。
- `/opt/spark/examples/jars/spark-examples_2.12-3.2.3.jar` 是 Spark 应用程序的 JAR 文件路径。
- `10` 是 SparkPi 应用程序的参数，表示计算 Pi 的精度。

该命令将在 Spark 集群上提交一个名为 `sparkPi` 的应用程序，该应用程序将计算 Pi 的值，并将结果返回给驱动程序。

```
23/04/28 16:57:08 INFO scheduler.DAGScheduler: Submitting 10 missing tasks from resultStage 0 (MapPartitionsRDD[1] at map at SparkPi.scala:34) (first 15 tasks are for partitions Vector(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 
```

## 2.7 参数信息的解释：

- `storage.BlockManagerInfo`：表示在 BlockManager 中添加了一个大小为 2.3 KiB 的广播变量。
- `scheduler.TaskSetManager`：表示应用程序的任务集管理器，包括启动和完成任务的信息，以及任务的详细信息，如任务在哪个节点上执行、任务的资源分配等。

## 三、本地运行wordcount程序

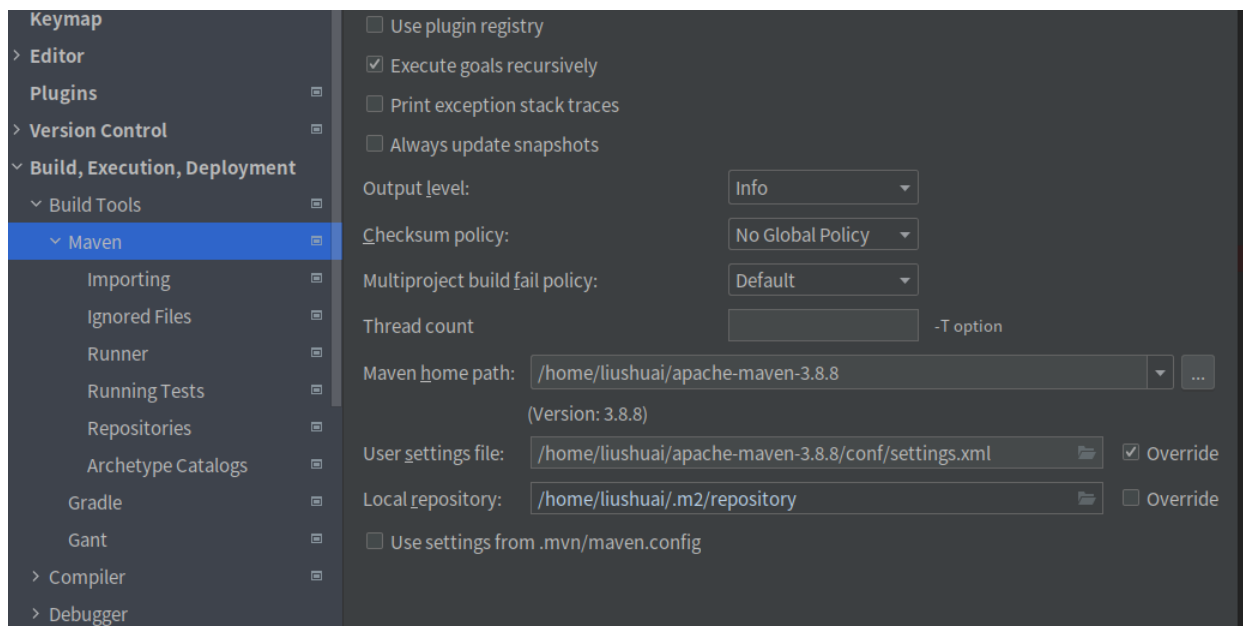
创建maven工程，并进行配置，同时更改importing和runner设置，在IDEA中设置maven编译时忽略HTTPS的SSL证书验证

在importing中添加

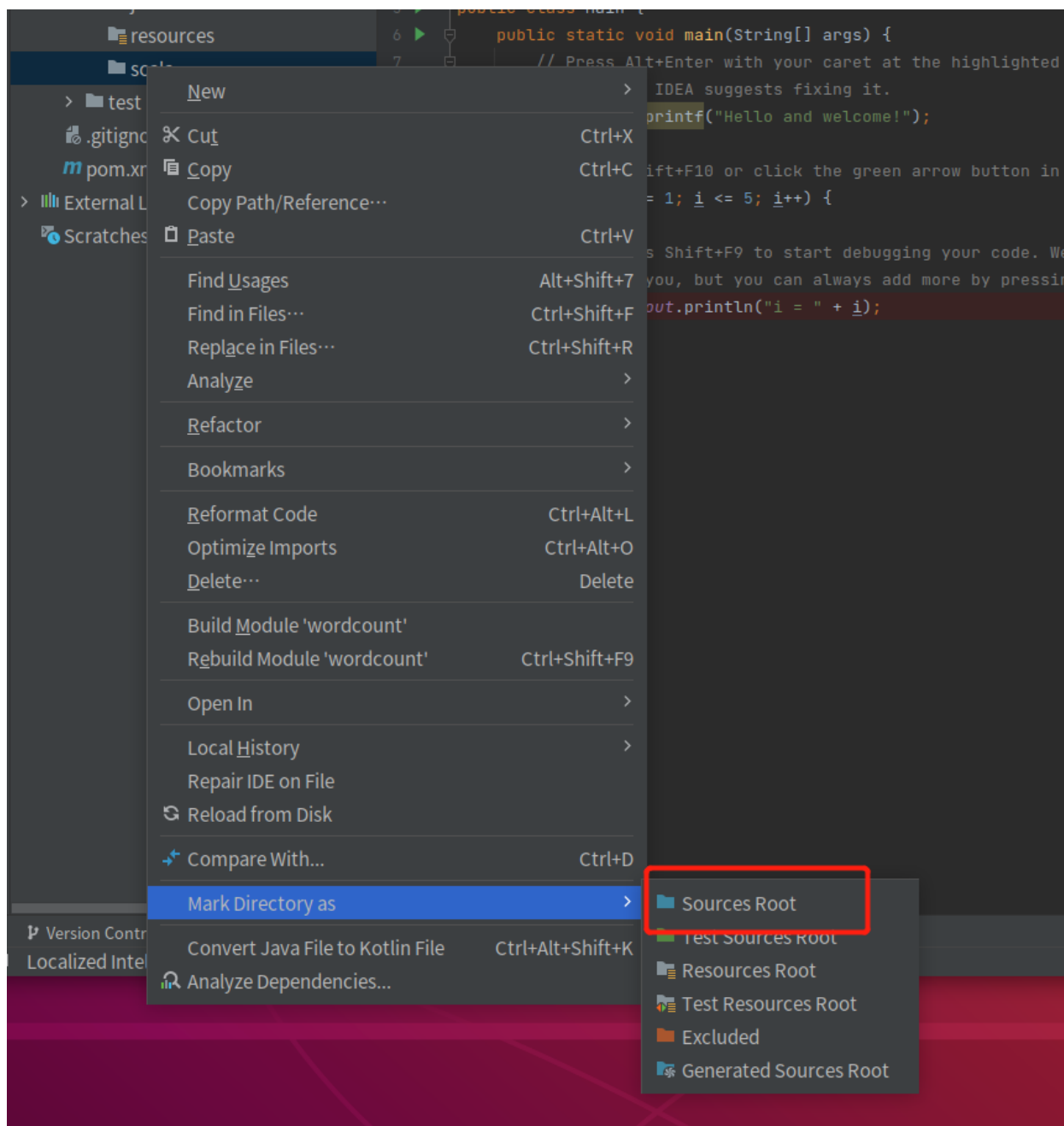
```
-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

在runner中添加

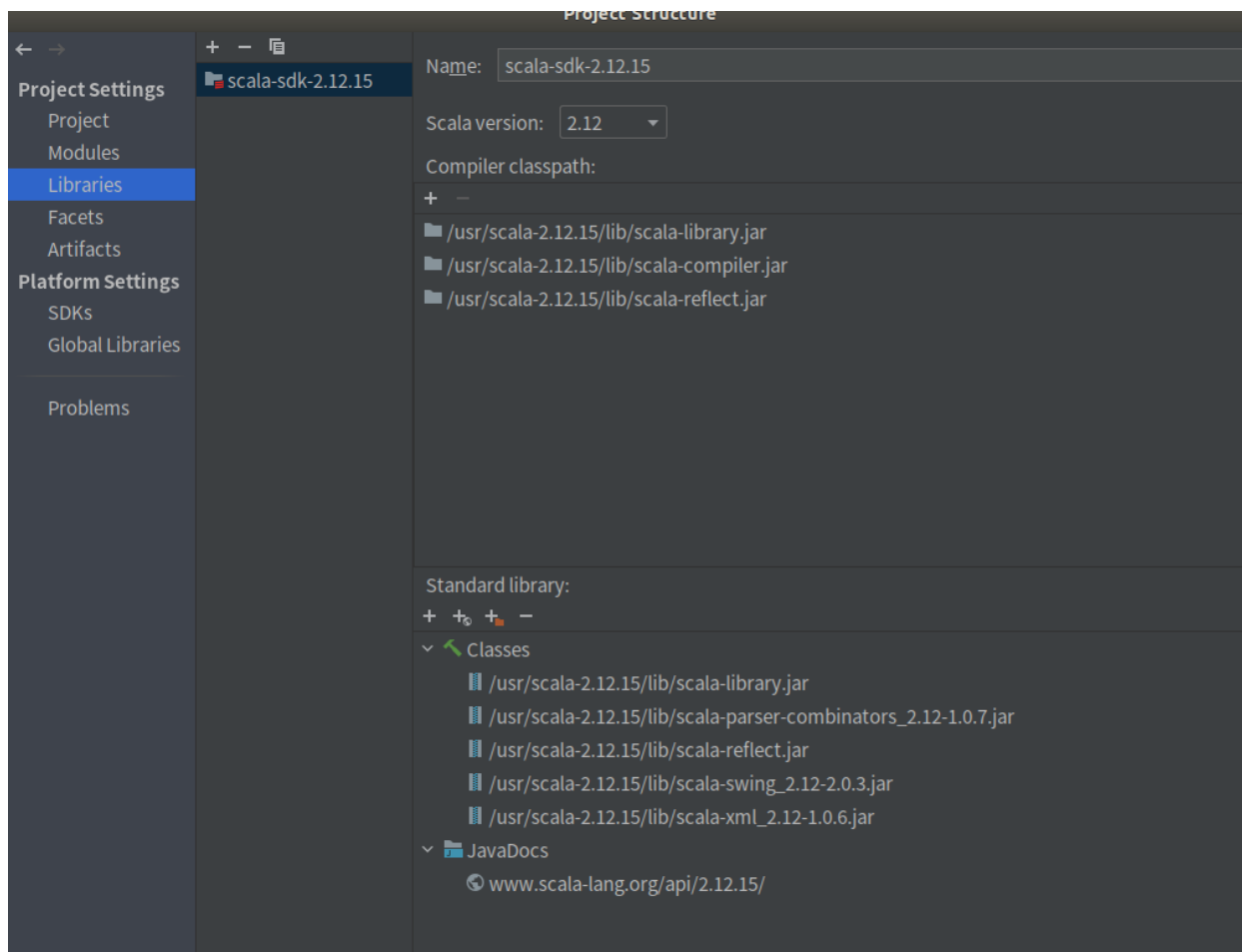
```
-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true  
-Dmaven.wagon.http.ssl.ignore.validity.dates=true
```



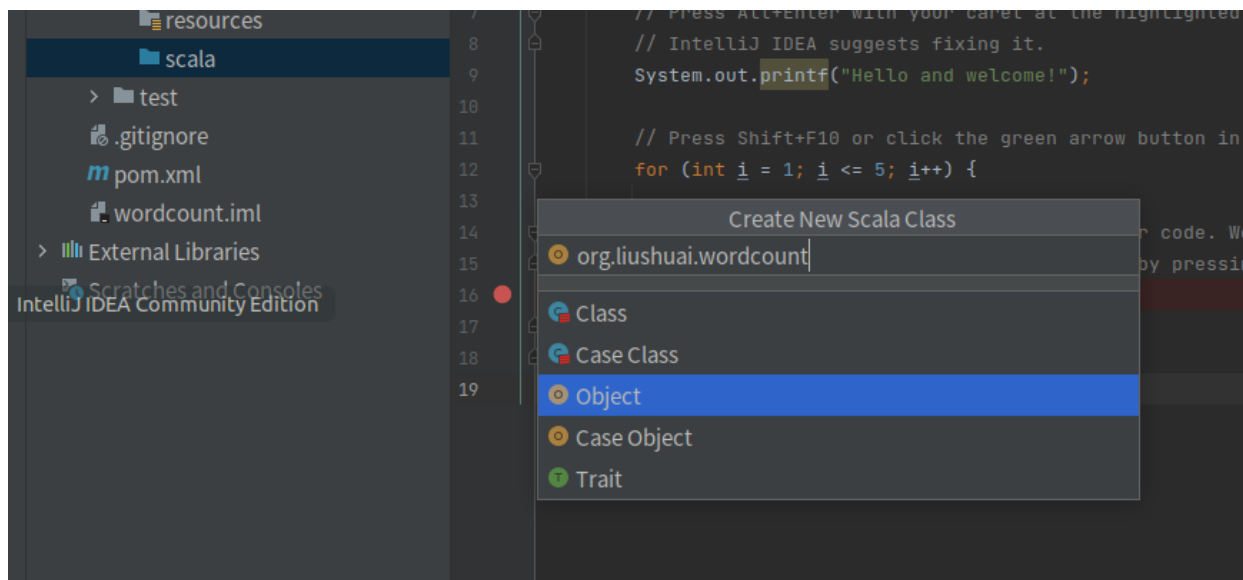
创建scala目录，并将其设置为sources root



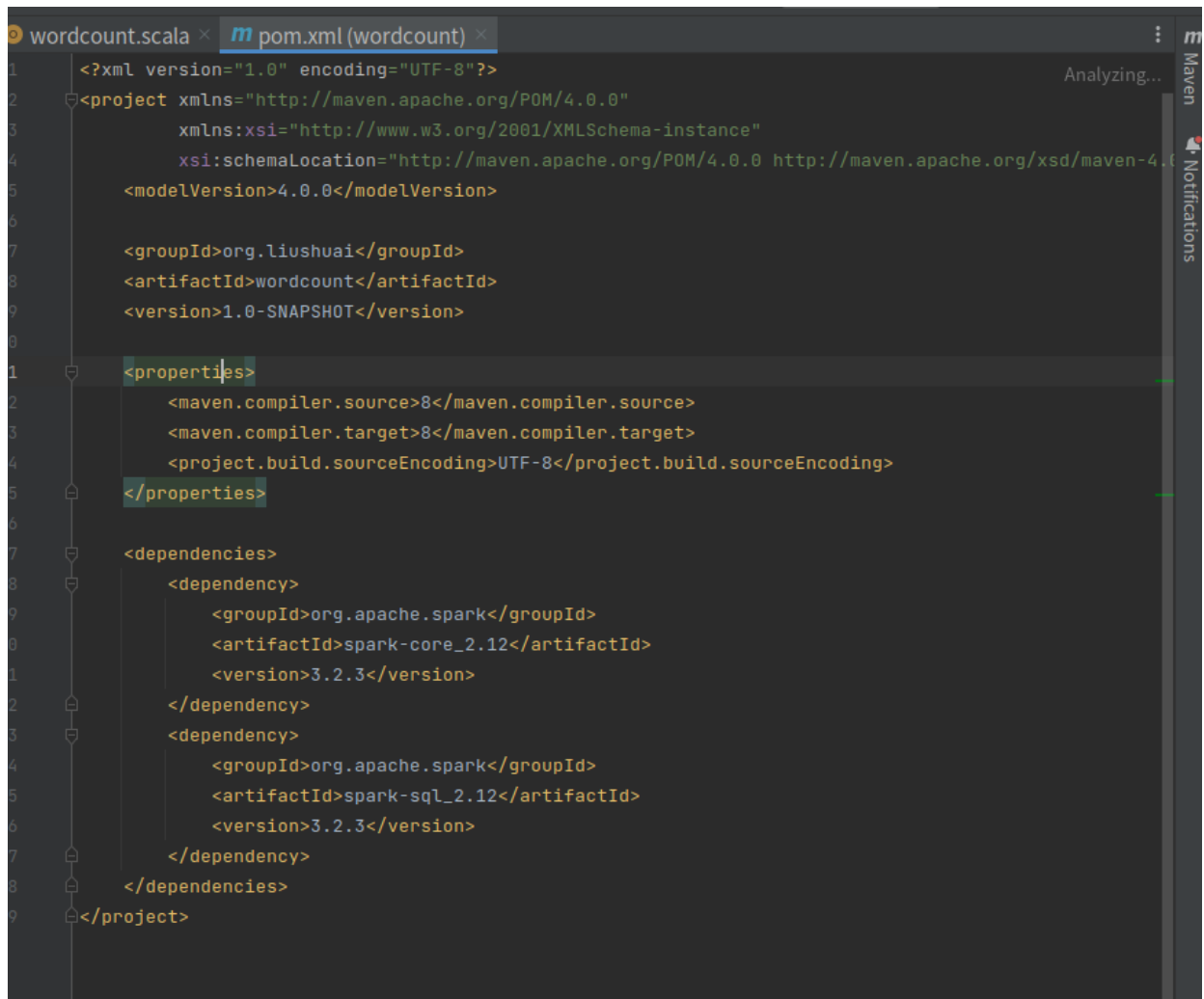
添加scala library，首先需要在plugins中下载scala插件，这样才能在libraries中找到scala-sdk



新建scala class



设置pom.xml文件，导入spark依赖



The screenshot shows an IDE with two tabs: 'wordcount.scala' and 'pom.xml (wordcount)'. The 'pom.xml' tab is active, displaying the following XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.liushuai</groupId>
  <artifactId>wordcount</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.12</artifactId>
      <version>3.2.3</version>
    </dependency>
    <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-sql_2.12</artifactId>
      <version>3.2.3</version>
    </dependency>
  </dependencies>
</project>
```

The IDE interface includes a sidebar on the right with 'Maven' and 'Notifications' sections, and a status bar at the bottom indicating 'Analyzing...'. The code is color-coded, with XML tags in blue and values in black.

编写scala程序：



```
scala> wordcount.scala
wordcount.scala x pom.xml (wordcount) x
1 package org.liushuai
2 import org.apache.spark.{SparkContext, SparkConf}
3 object wordcount {
4   def main(args: Array[String]){
5     /**
6      * 第1 步; 创建Spark 的配置对象SparkConf, 设置Spark 程序运行时的配置信息
7      * 例如setAppName 用来设置应用程序的名称, 在程序运行的监控界面可以看到该名
8      * 称,
9      * setMaster 设置程序运行在本地还是运行在集群中, 运行在本地可是使用local 参数,
10     * 也可以使用local[K]/local[*],
11     * 可以去spark 官网查看它们不同的意义。如果要运行在集群中, 以Standalone 模式
12     * 运行的话, 需要使用spark://HOST:PORT
13     * 的形式指定master 的IP 和端口号, 默认是7077
14     */
15     val conf = new SparkConf().setAppName("WordCount").setMaster("local")
16     /**
17     * 第2 步: 创建SparkContext 对象
18     * SparkContext 是Spark 程序所有功能的唯一入口
19     * SparkContext 核心作用: 初始化Spark 应用程序运行所需要的核心组件, 包括
20     * DAGScheduler、TaskScheduler、SchedulerBackend
21     * 同时还会负责Spark 程序往Master 注册程序
22     *
23     * 通过传入SparkConf 实例来定制Spark 运行的具体参数和配置信息
24     */
25     val sc = new SparkContext(conf)
26
27     /**
28     * 第3 步: 根据具体的数据来源(HDFS、HBase、Local FS、DB、S3 等)通过SparkContext
29     * 来创建RDD
30     * RDD 的创建基本有三种方式: 根据外部的数据来源(例如HDFS)、根据Scala 集合使
31     * 用SparkContext 的parallelize 方法、
32     * 由其他的RDD 操作产生
33     * 数据会被RDD 划分为一系列的Partitions, 分配到每个Partition 的数据属于一个
34     * Task 的处理范畴
35     */
36   }
37 }
```

将测试文件上传到/opt/hadoop/ceshi.txt中, 通过scala类进行读取并计算

在idea中运行wordcount程序, 并查看运行结果

The screenshot shows an IDE with a Scala file named `wordcount.scala` open. The code defines a `main` function that reads a file, splits it into words, and counts them. The file path is `/opt/hadoop/ceshi.txt`. The code is as follows:

```
34 * Task 的处理范畴
35 */
36
37 val lines = sc.textFile(path = "/opt/hadoop/ceshi.txt") // 读取本地文件
38 /**
39 * 第4步: 对初始的RDD 进行Transformation 级别的处理, 例如map、filter 等高级
40 * 函数来进行具体的数据计算
41 */
42 val words = lines.flatMap(_.split(regex = " ")).filter(word => word != " ") // 拆分单词, 并过滤掉空字符串
```

The `main` function is called with `args: Array[String]`. Below the code, the `Run` tab shows the Spark execution logs. The logs indicate that the job was successfully completed, with a total of 1520 bytes of data sent to the driver and a total execution time of 7.614725 seconds. The output of the job is displayed as follows:

```
(d,307)
(a,346)
(b,325)
(hello,346)
(c,335)
(world,341)
```

## 四、性能分析

mapreduce方法和spark方法实现wordcount功能的性能分析如下:

- 实现方法

MapReduce 的实现通常需要编写 Map 和 Reduce 函数, 并将代码打包成可执行的 Jar 包提交到 Hadoop 集群上运行。而 Spark 的实现则可以使用 Spark 提供的 RDD 和高级 API, 通过编写 Scala、Python 或 Java 代码来实现。

- 性能表现

Spark 相对于 MapReduce 有一些优势。首先, Spark 的计算模型是基于内存的, 可以将数据存储在内存中进行高效的计算, 而 MapReduce 的计算模型则是基于磁盘的, 需要将数据从磁盘中读取到内存中进行计算, 因此 Spark 的计算速度通常要快于 MapReduce。其次, Spark 提供了更丰富的数据处理和转换操作, 可以方便地进行数据清洗、过滤、排序、聚合等操作, 相对而言更加灵活和方便。

- 大规模场景处理

对于大规模数据集的处理, MapReduce 效果更优, 因为它可以通过横向扩展来处理大规模数据集, 而 Spark 在处理大规模数据集时需要考虑内存的限制和数据分区等问题。此外, MapReduce 也可以运行在各种 Hadoop 生态系统中, 包括 HDFS、YARN、HBase 等等, 具有更广泛的适用性。