大数据技术基础实验四
实验报告

毛子恒
2019211397
北京邮电大学 计算机学院

日期：2022 年 4 月 28 日

1 概述

1.1 实验目的

1. 了解服务器配置的过程；
2. 熟悉使用 Scala 编写 Spark 程序；
3. 了解 Spark RDD 的工作原理；
4. 掌握在 Spark 集群上运行程序的方法。
5. 掌握 Hive 安装部署运行的方式。
6. 掌握 Spark 读取 Hive 方式。

1.2 实验步骤

1. Hadoop 集群环境测试；
2. Spark 集群搭建；
3. Scala 程序编写。
4. 程序打包与运行；
5. 安装 Hive；
6. Hive 建库并导入数据；
7. 修改并运行程序。

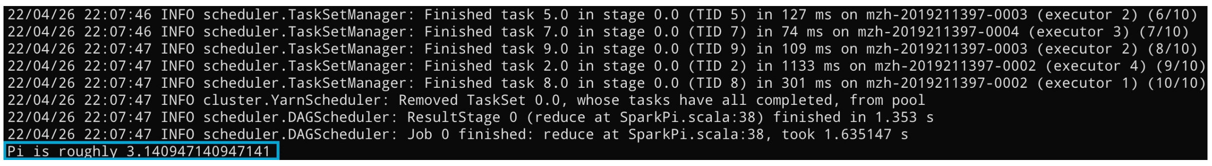
2 实验结果及分析

Spark 集群搭建 Spark 集群部署结果如图 1和图 2。

使用 Spark 框架计算速度更快，得到 π 的精度更高。

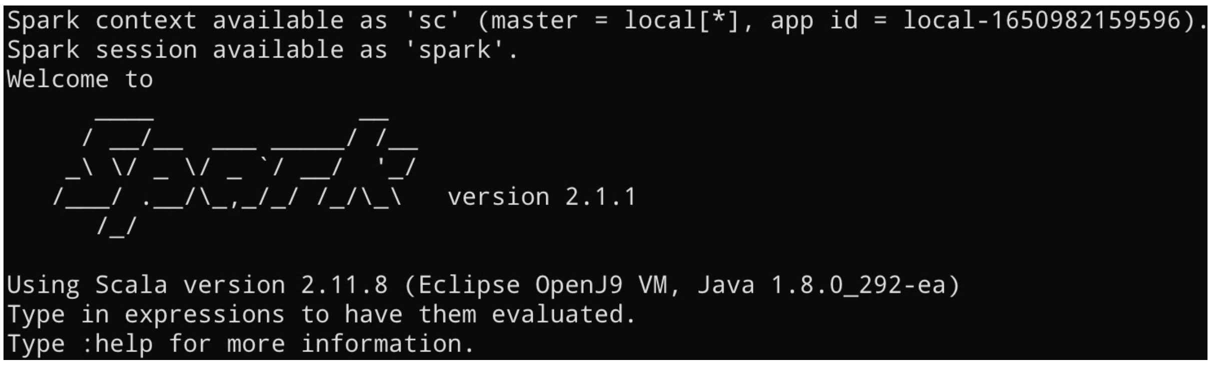
程序打包和运行 编写程序如图 3。

程序中创建了一个字符串列表，配置一个 Spark 任务，之后利用 `parallelize` 方法生成 RDD lines，其元素为字符串语句。将每个 lines 按照空格切分成单词，生成 `wordsRDD` 对象。再



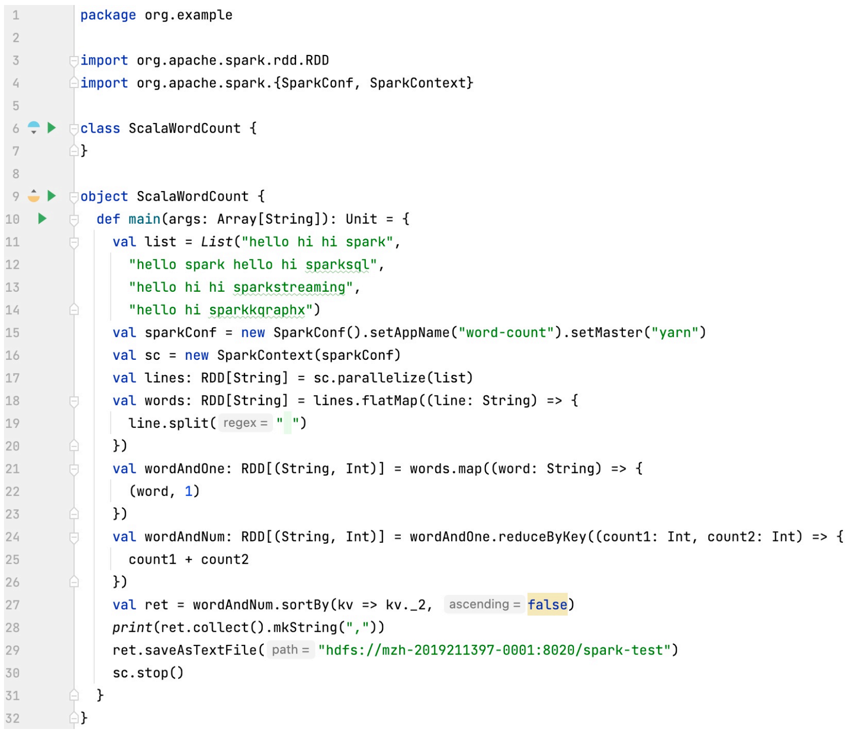
```
22/04/26 22:07:46 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 0.0 (TID 5) in 127 ms on mzh-2019211397-0003 (executor 2) (6/10)
22/04/26 22:07:46 INFO scheduler.TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 74 ms on mzh-2019211397-0004 (executor 3) (7/10)
22/04/26 22:07:47 INFO scheduler.TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 109 ms on mzh-2019211397-0003 (executor 2) (8/10)
22/04/26 22:07:47 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 1133 ms on mzh-2019211397-0002 (executor 4) (9/10)
22/04/26 22:07:47 INFO scheduler.TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 301 ms on mzh-2019211397-0002 (executor 1) (10/10)
22/04/26 22:07:47 INFO cluster.YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
22/04/26 22:07:47 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 1.353 s
22/04/26 22:07:47 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 1.635147 s
Pi is roughly 3.140947140947141
```

图 **1:** Spark 集群部署结果



```
Spark context available as 'sc' (master = local[*], app id = local-1650982159596).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.1
      /_/

Using Scala version 2.11.8 (Eclipse OpenJ9 VM, Java 1.8.0_292-ea)
Type in expressions to have them evaluated.
Type :help for more information.
```

图 **2:** Spark 版本信息



```scala
package org.example

import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}

class ScalaWordCount {
}

object ScalaWordCount {
  def main(args: Array[String]): Unit = {
    val list = List("hello hi hi spark",
      "hello spark hello hi sparksql",
      "hello hi hi sparkstreaming",
      "hello hi sparkkgraphx")
    val sparkConf = new SparkConf().setAppName("word-count").setMaster("yarn")
    val sc = new SparkContext(sparkConf)
    val lines: RDD[String] = sc.parallelize(list)
    val words: RDD[String] = lines.flatMap((line: String) => {
      line.split( regex = " ")
    })
    val wordAndOne: RDD[(String, Int)] = words.map((word: String) => {
      (word, 1)
    })
    val wordAndNum: RDD[(String, Int)] = wordAndOne.reduceByKey((count1: Int, count2: Int) => {
      count1 + count2
    })
    val ret = wordAndNum.sortBy(kv => kv._2,  ascending = false)
    print(ret.collect().mkString(","))
    ret.saveAsTextFile( path = "hdfs://mzh-2019211397-0001:8020/spark-test")
    sc.stop()
  }
}
```

图 **3:** wordcount 程序

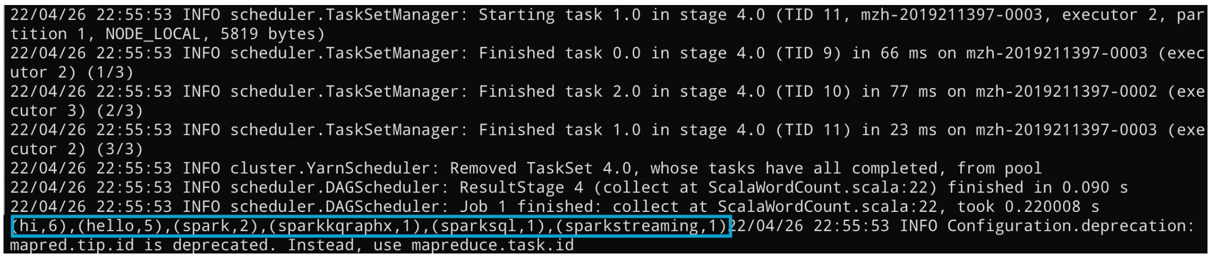
以单词为键，1 为值建立键值对 `wordAndOneRDD` 对象，调用 `reduceByKey` 方法实现聚合操作，得到 `wordAndNumRDD` 对象。最后将 `wordAndNum` 按照值降序排序，打印到控制台。

将程序打包，上传到服务器运行，程序运行的结果如图 **4**、图 **5**和图 **6**。

可见 Spark 统计了每个单词出现的次数，将结果保存到`/spark-test/`文件夹中。

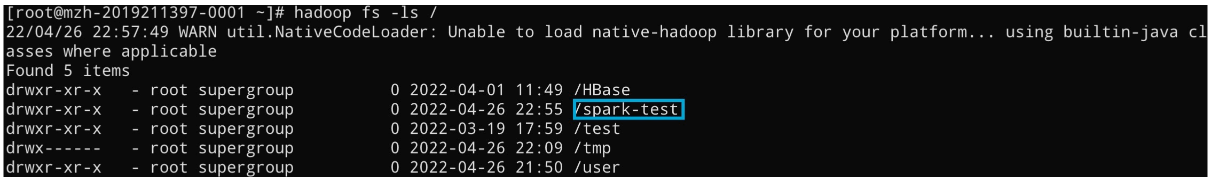
安装 Hive 安装 MySQL 并启动 MySQL Server 的结果如图 **7**。

修改 MySQL root 密码、修改数据库默认编码的结果如图 **8**。



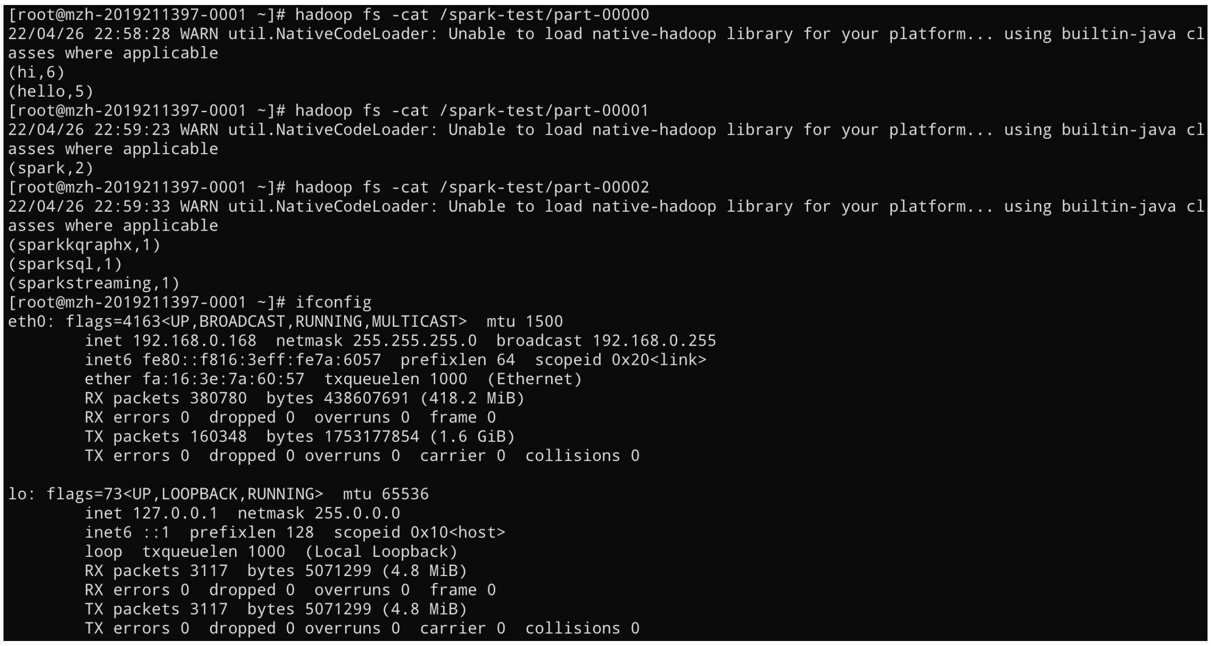
```
22/04/26 22:55:53 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 4.0 (TID 11, mzh-2019211397-0003, executor 2, par
tition 1, NODE_LOCAL, 5819 bytes)
22/04/26 22:55:53 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 4.0 (TID 9) in 66 ms on mzh-2019211397-0003 (exec
utor 2) (1/3)
22/04/26 22:55:53 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 4.0 (TID 10) in 77 ms on mzh-2019211397-0002 (exe
cutor 3) (2/3)
22/04/26 22:55:53 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 4.0 (TID 11) in 23 ms on mzh-2019211397-0003 (exe
cutor 2) (3/3)
22/04/26 22:55:53 INFO cluster.YarnScheduler: Removed TaskSet 4.0, whose tasks have all completed, from pool
22/04/26 22:55:53 INFO scheduler.DAGScheduler: ResultStage 4 (collect at ScalaWordCount.scala:22) finished in 0.090 s
22/04/26 22:55:53 INFO scheduler.DAGScheduler: Job 1 finished: collect at ScalaWordCount.scala:22, took 0.220008 s
(hi,6),(hello,5),(spark,2),(sparkkqraphx,1),(sparksql,1),(sparkstreaming,1)22/04/26 22:55:53 INFO Configuration.deprecation:
mapred.tip.id is deprecated. Instead, use mapreduce.task.id
```

图 4: 运行结果



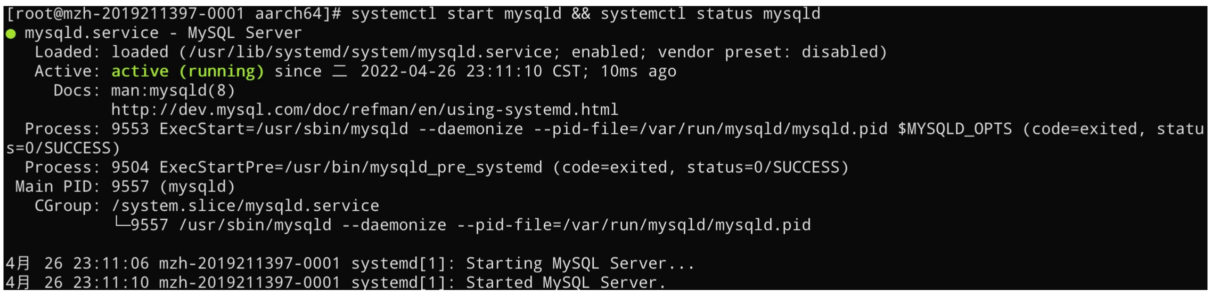
```
[root@mzh-2019211397-0001 ~]# hadoop fs -ls /
22/04/26 22:57:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 5 items
drwxr-xr-x   - root supergroup          0 2022-04-01 11:49 /HBase
drwxr-xr-x   - root supergroup          0 2022-04-26 22:55 /spark-test
drwxr-xr-x   - root supergroup          0 2022-03-19 17:59 /test
drwx------   - root supergroup          0 2022-04-26 22:09 /tmp
drwxr-xr-x   - root supergroup          0 2022-04-26 21:50 /user
```

图 5: 运行结果



```
[root@mzh-2019211397-0001 ~]# hadoop fs -cat /spark-test/part-00000
22/04/26 22:58:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
(hi,6)
(hello,5)
[root@mzh-2019211397-0001 ~]# hadoop fs -cat /spark-test/part-00001
22/04/26 22:59:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
(spark,2)
[root@mzh-2019211397-0001 ~]# hadoop fs -cat /spark-test/part-00002
22/04/26 22:59:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
(sparkkqraphx,1)
(sparksql,1)
(sparkstreaming,1)
[root@mzh-2019211397-0001 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.168  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe7a:6057  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:7a:60:57  txqueuelen 1000  (Ethernet)
        RX packets 380780  bytes 438607691 (418.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160348  bytes 1753177854 (1.6 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 3117  bytes 5071299 (4.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3117  bytes 5071299 (4.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 6: 运行结果

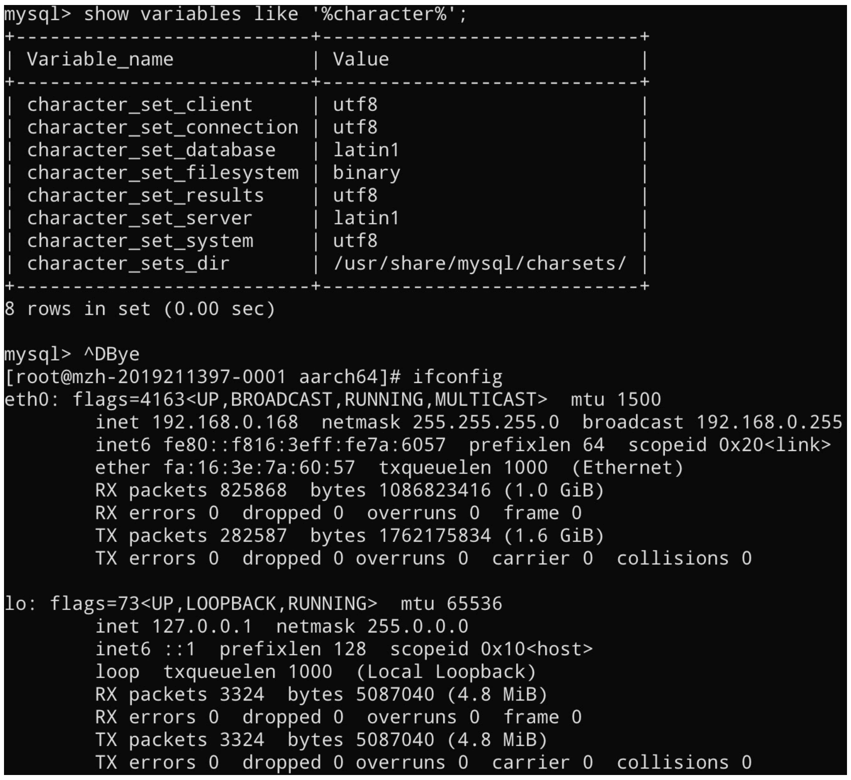


```
[root@mzh-2019211397-0001 aarch64]# systemctl start mysqld && systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since 二 2022-04-26 23:11:10 CST; 10ms ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
  Process: 9553 ExecStart=/usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid $MYSQLD_OPTS (code=exited, statu
s=0/SUCCESS)
  Process: 9504 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
 Main PID: 9557 (mysqld)
   CGroup: /system.slice/mysqld.service
           └─9557 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid

4月 26 23:11:06 mzh-2019211397-0001 systemd[1]: Starting MySQL Server...
4月 26 23:11:10 mzh-2019211397-0001 systemd[1]: Started MySQL Server.
```

图 7: MySQL 安装结果

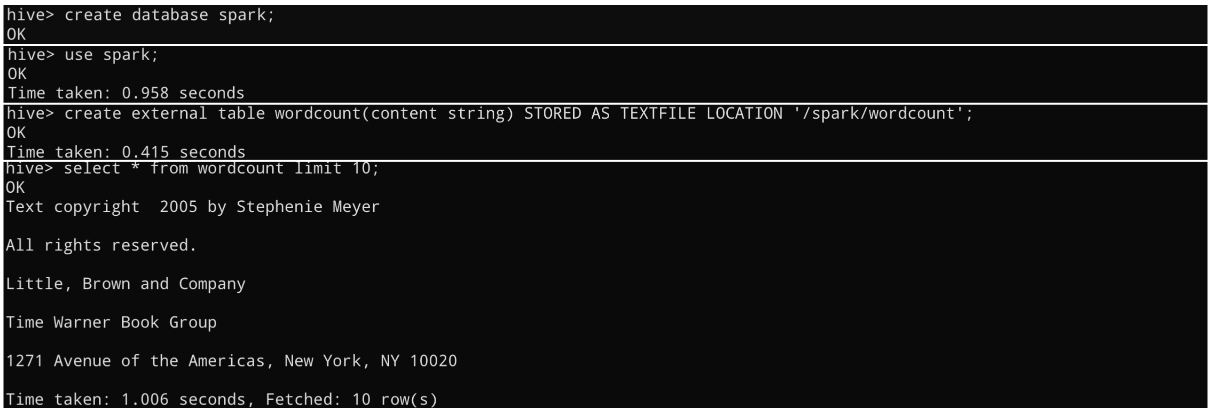
Hive 建库并导入数据　在 Hive 中建立名为 `spark` 的数据库，其中创建名为 `wordcount` 的表，内容为一个文本文件的各个行，命令的运行结果如图 9。

编写程序如图 10。



```
mysql> show variables like '%character%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)

mysql> ^DBye
[root@mzh-2019211397-0001 aarch64]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.168  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe7a:6057  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:7a:60:57  txqueuelen 1000  (Ethernet)
        RX packets 825868  bytes 1086823416 (1.0 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 282587  bytes 1762175834 (1.6 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 3324  bytes 5087040 (4.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3324  bytes 5087040 (4.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 8: MySQL 修改编码结果



```
hive> create database spark;
OK
hive> use spark;
OK
Time taken: 0.958 seconds
hive> create external table wordcount(content string) STORED AS TEXTFILE LOCATION '/spark/wordcount';
OK
Time taken: 0.415 seconds
hive> select * from wordcount limit 10;
OK
Text copyright  2005 by Stephenie Meyer

All rights reserved.

Little, Brown and Company

Time Warner Book Group

1271 Avenue of the Americas, New York, NY 10020

Time taken: 1.006 seconds, Fetched: 10 row(s)
```

图 9: Hive 命令行运行结果

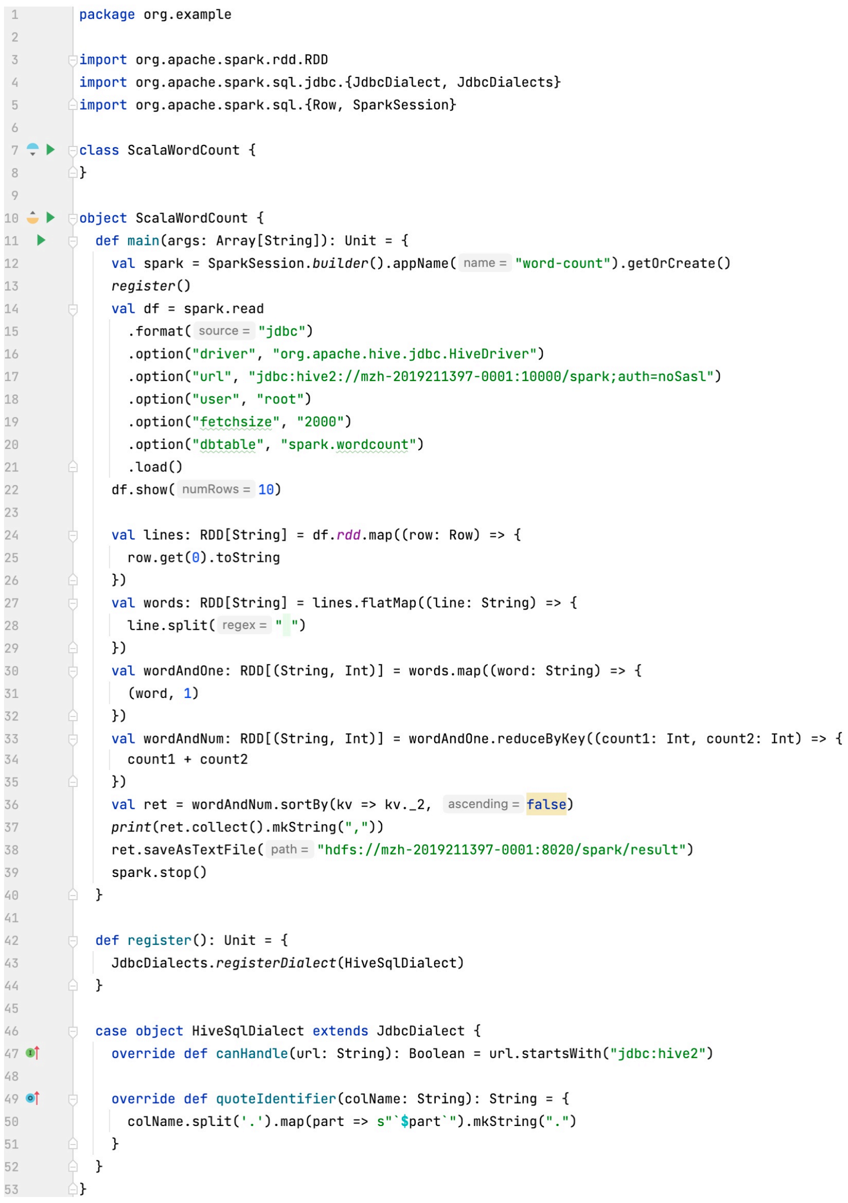
相比于图 **3**，程序中将输入源改为 Hive SQL Driver。

运行过程如图 **11**。

运行结果导出后显示前十行，如图 **12**。

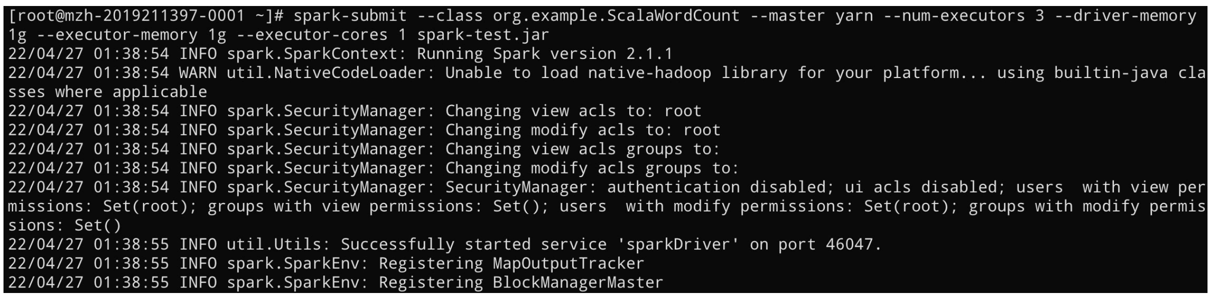
3　实验总结

本次实验中我搭建了 Spark 和 Hive 环境，并且编写了 Java 代码实现了单词个数统计，在实验过程中我对 Spark RDD 的工作原理和 Spark 读取 Hive 的方式有了更深入的理解，我的 Java 编程水平也得到了提高，我从中受益良多。



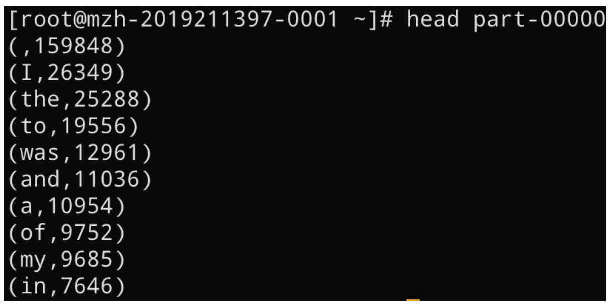
```scala
package org.example

import org.apache.spark.rdd.RDD
import org.apache.spark.sql.jdbc.{JdbcDialect, JdbcDialects}
import org.apache.spark.sql.{Row, SparkSession}

class ScalaWordCount {
}

object ScalaWordCount {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("word-count").getOrCreate()
    register()
    val df = spark.read
      .format("jdbc")
      .option("driver", "org.apache.hive.jdbc.HiveDriver")
      .option("url", "jdbc:hive2://mzh-2019211397-0001:10000/spark;auth=noSasl")
      .option("user", "root")
      .option("fetchsize", "2000")
      .option("dbtable", "spark.wordcount")
      .load()
    df.show(10)

    val lines: RDD[String] = df.rdd.map((row: Row) => {
      row.get(0).toString
    })
    val words: RDD[String] = lines.flatMap((line: String) => {
      line.split(" ")
    })
    val wordAndOne: RDD[(String, Int)] = words.map((word: String) => {
      (word, 1)
    })
    val wordAndNum: RDD[(String, Int)] = wordAndOne.reduceByKey((count1: Int, count2: Int) => {
      count1 + count2
    })
    val ret = wordAndNum.sortBy(kv => kv._2, false)
    print(ret.collect().mkString(","))
    ret.saveAsTextFile("hdfs://mzh-2019211397-0001:8020/spark/result")
    spark.stop()
  }

  def register(): Unit = {
    JdbcDialects.registerDialect(HiveSqlDialect)
  }

  case object HiveSqlDialect extends JdbcDialect {
    override def canHandle(url: String): Boolean = url.startsWith("jdbc:hive2")

    override def quoteIdentifier(colName: String): String = {
      colName.split('.').map(part => s"`$part`").mkString(".")
    }
  }
}
```

图 **10:** wordcount 程序



```
[root@mzh-2019211397-0001 ~]# spark-submit --class org.example.ScalaWordCount --master yarn --num-executors 3 --driver-memory
1g --executor-memory 1g --executor-cores 1 spark-test.jar
22/04/27 01:38:54 INFO spark.SparkContext: Running Spark version 2.1.1
22/04/27 01:38:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cla
sses where applicable
22/04/27 01:38:54 INFO spark.SecurityManager: Changing view acls to: root
22/04/27 01:38:54 INFO spark.SecurityManager: Changing modify acls to: root
22/04/27 01:38:54 INFO spark.SecurityManager: Changing view acls groups to:
22/04/27 01:38:54 INFO spark.SecurityManager: Changing modify acls groups to:
22/04/27 01:38:54 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view per
missions: Set(root); groups with view permissions: Set(); users  with modify permissions: Set(root); groups with modify permis
sions: Set()
22/04/27 01:38:55 INFO util.Utils: Successfully started service 'sparkDriver' on port 46047.
22/04/27 01:38:55 INFO spark.SparkEnv: Registering MapOutputTracker
22/04/27 01:38:55 INFO spark.SparkEnv: Registering BlockManagerMaster
```

图 **11:** 运行过程



```
[root@mzh-2019211397-0001 ~]# head part-00000
(,159848)
(I,26349)
(the,25288)
(to,19556)
(was,12961)
(and,11036)
(a,10954)
(of,9752)
(my,9685)
(in,7646)
```

图 12: 运行结果