



大数据技术基础课程实验报告

实验四：Spark Core Scala、Spark SQL

付容天

学号 2020211616

班级 2020211310

计算机学院（国家示范性软件学院）

2023 年 4 月 16 日

1. 实验描述与实验目的

本实验使用 Scala 语言编写 Spark 程序，完成单词计数任务、独立应用程序实现数据去重任务，并使用 Spark SQL 完成数据库读写任务。实验分为三个部分：首先，在华为云购买 4 台服务器（已完成），然后搭建 Hadoop 集群和 Spark 集群（YARN 模式）（已完成），接着使用 Scala 语言利用 Spark Core 编写程序，最后将程序打包在集群上运行；其次，使用 Scala 语言编写独立应用程序实现数据去重，并将程序打包在集群上运行；最后，使用 Spark SQL 读写数据库，包括在服务器上安装 MySQL 和通过 JDBC 连接数据库在本次实验中，我们需要在实验一、二搭建完毕的集群环境上，继续安装 HBase、ZooKeeper，并实践 HBase 的基本使用。

本次实验的目的是：

- (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) 掌握使用 Spark SQL 读写数据库的方法。

2. 实验过程与实验分析

2.1. Spark Core Scala 单词计数

这部分的实验中，我首先按照实验指导书要求启动了 Hadoop 集群，并通过 jps 和 ifconfig 两条指令进行了必要的检查，结果如下所示：

```
root@firt-2020211616-0001:~/hadoop-2.7.7
[root@firt-2020211616-0001 hadoop-2.7.7]# jps
2288 ResourceManager
2104 SecondaryNameNode
1885 NameNode
2624 Jps
[root@firt-2020211616-0001 hadoop-2.7.7]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.34 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe4b:a3c5 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:4b:a3:c5 txqueuelen 1000 (Ethernet)
    RX packets 207873 bytes 296363919 (281.6 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 63819 bytes 4467220 (4.2 MiB)
    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 222 bytes 154013 (150.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 222 bytes 154013 (150.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

图 1: node1 状态检查（对应指导书图 1）

```
root@firt-2020211616-0002:~
[root@firt-2020211616-0002 ~]# jps
1967 Jps
1828 ResourceManager
1707 DataNode
[root@firt-2020211616-0002 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.41 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe4b:a3cc prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:4b:a3:cc txqueuelen 1000 (Ethernet)
    RX packets 637 bytes 108782 (106.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 838 bytes 235318 (229.8 KiB)
    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 75 bytes 85050 (83.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 75 bytes 85050 (83.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

图 2: node2 状态检查（对应指导书图 2）

```
root@firt-2020211616-0003:~  
[root@firt-2020211616-0003 ~]# jps  
1825 NodeManager  
1704 DataNode  
1967 Jps  
[root@firt-2020211616-0003 ~]# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.167 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::f816:3eff:fe4b:a34a prefixlen 64 scopeid 0x20<link>  
    ether fa:16:3e:4b:a3:4a txqueuelen 1000 (Ethernet)  
    RX packets 653 bytes 110482 (107.8 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 904 bytes 250250 (244.3 KiB)  
    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 75 bytes 85050 (83.0 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 75 bytes 85050 (83.0 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

图 3: node3 状态检查 (对应指导书图 3)

```
root@firt-2020211616-0004:~  
[root@firt-2020211616-0004 ~]# jps  
1970 Jps  
1828 NodeManager  
1707 DataNode  
[root@firt-2020211616-0004 ~]# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.114 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::f816:3eff:fe4b:a315 prefixlen 64 scopeid 0x20<link>  
    ether fa:16:3e:4b:a3:15 txqueuelen 1000 (Ethernet)  
    RX packets 756 bytes 123102 (120.2 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 1044 bytes 270282 (263.9 KiB)  
    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 75 bytes 85049 (83.0 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 75 bytes 85049 (83.0 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

图 4: node4 状态检查 (对应指导书图 4)

通过以上四幅图, 可以看到在 node1 上有四个进程 (ResourceManager、SecondaryNameNode、NameNode、JPS), 在 node2 到 node4 上均为三个进程 (JPS、NodeManager、DataNode)。我们还可以发现 node1 到 node4 的 IP 地址分别为 192.168.0.34、192.168.0.41、192.168.0.167 和 192.168.0.114。

紧接着我按照实验指导书内容执行了相应命令 (该测试命令用以计算圆周率 PI), 测试 Hadoop 集群的可用性, 结果如下图 5 和 6 所示:

```
root@firt-2020211616-0001:~  
[root@firt-2020211616-0001 ~]# cd ~  
[root@firt-2020211616-0001 ~]# hadoop jar ../home/modul  
.7.7.jar pi 10 1  
Number of Maps = 10  
Samples per Map = 1  
23/04/16 16:11:38 WARN util.NativeCodeLoader: Unable t  
va classes where applicable  
Wrote input for Map #0  
Wrote input for Map #1  
Wrote input for Map #2  
Wrote input for Map #3  
Wrote input for Map #4  
Wrote input for Map #5  
Wrote input for Map #6  
Wrote input for Map #7  
Wrote input for Map #8  
Wrote input for Map #9  
Starting Job  
23/04/16 16:11:39 INFO client.RMProxy: Connecting to F
```

图 5: Hadoop 集群可用性测试 (对应指导书图 5)

```
virtual memory (bytes) snapshot=14201192448  
Total committed heap usage (bytes)=1703804928  
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=1180  
File Output Format Counters  
Bytes Written=97  
Job Finished in 21.718 seconds  
Estimated value of Pi is 3.600000000000000000000000  
[root@firt-2020211616-0001 ~]#
```

图 6: Hadoop 集群可用性测试 (对应指导书图 6)


在确定 Hadoop 集群可用之后，我向主节点（即 node1）节点上传、解压 Spark 安装包，并进行必要的配置（修改 .bash_profile 文件和 yarn-site.xml 文件）。然后使用 scp 命令将修改后的 yarn-site.xml 文件发送到三个从节点。此时重启 Hadoop 服务，查看 JPS 可以发现与图 1-图 4 所示结果一致，运行 Spark 中提供的计算圆周率 PI 的指令，得到图下图 7 所示的结果，可以确定此时 Spark 集群部署成功！

```
root@firt-2020211616-0001:~
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Starting task 8.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 5.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 6.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Starting task 9.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 8.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 7.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 2.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 9.0 in stage
23/04/16 16:25:54 INFO scheduler.TaskSetManager: Finished task 0.0 in stage
23/04/16 16:25:54 INFO cluster.YarnScheduler: Removed TaskSet 0.0, whose tas
23/04/16 16:25:54 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at Spa
23/04/16 16:25:54 INFO scheduler.DAGScheduler: Job 0 finished: reduce at Spa
Pi is roughly 3.1416911416911417
23/04/16 16:25:54 INFO server.ServerConnector: Stopped Spark@994c761e{HTTP/1
23/04/16 16:25:54 INFO handler.ContextHandler: Stopped o.s.j.s.ServletConte
23/04/16 16:25:54 INFO handler.ContextHandler: Stopped o.s.j.s.ServletConte
23/04/16 16:25:54 INFO handler.ContextHandler: Stopped o.s.j.s.ServletConte
```

图 7：检验 Spark 集群搭建成功（对应指导书图 7）

之后运行 spark-shell 命令，可以看到 Spark 和 Scala 版本信息：

```
root@firt-2020211616-0001:~
23/04/16 16:25:54 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoi
23/04/16 16:25:54 INFO spark.SparkContext: Successfully stopped SparkContext
23/04/16 16:25:54 INFO util.ShutdownHookManager: Shutdown hook called
23/04/16 16:25:54 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-7f74f80
[root@firt-2020211616-0001 ~]# spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newL
23/04/16 16:28:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
23/04/16 16:28:30 WARN metastore.ObjectStore: Version information not found in metasto
rding the schema version 1.2.0
23/04/16 16:28:31 WARN metastore.ObjectStore: Failed to get database default, returnin
23/04/16 16:28:32 WARN metastore.ObjectStore: Failed to get database global_temp, retu
Spark context Web UI available at http://192.168.0.34:4040
Spark context available as 'sc' (master = local[*], app id = local-1681633704690).
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 8：进入 spark-shell 查看 Spark 和 Scala 版本（对应指导书图 8）

接下来的任务就是在 IDEA 中创建项目，完成代码编写并进行配置，之后生成 jar 包。该阶段中我主要完成了：

- (1) 工程创建，命名为 spark-test，选择正确的 archetype；
- (2) 根据指导书内容修改 pro.xml 配置文件，并使之生效；
- (3) 设置语言环境和 java Compiler 环境；
- (4) 删除模板中 test 文件夹下的测试类（即 AppTest 和 MySpec 两个文件）；

- (5) 创建 Scala Class, 命名为 ScalaWordCount, 并编写相应的代码;
- (6) 设置 Project Settings 中的 Artifacts, 选择好主类;
- (7) 选择 Build->Artifacts, 生成 jar 包;
- (8) 删除 jar 包中 META-INF 目录下的 MANIFEST.MF 文件。

接下来将 jar 包传到 node1 上, 并通过 spark-submit 命令执行程序, 得到如下图 9 所示的结果:

```
root@firt-2020211616-0001:~
23/04/16 16:30:11 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on node2:42037
23/04/16 16:30:11 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on node4:45277
23/04/16 16:30:11 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for
23/04/16 16:30:11 INFO spark.MapOutputTrackerMaster: Size of output statuses for shuffle 1 is 172 by
23/04/16 16:30:11 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for
23/04/16 16:30:11 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 4.0 (TID 11, node2, exec
23/04/16 16:30:11 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 4.0 (TID 10) in 65 ms on
23/04/16 16:30:11 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 4.0 (TID 9) in 84 ms on
23/04/16 16:30:11 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 4.0 (TID 11) in 35 ms on
23/04/16 16:30:11 INFO cluster.YarnScheduler: Removed TaskSet 4.0, whose tasks have all completed, f
23/04/16 16:30:11 INFO scheduler.DAGScheduler: ResultStage 4 (collect at ScalaWordCount.scala:21) fi
23/04/16 16:30:11 INFO scheduler.DAGScheduler: Job 1 finished: collect at ScalaWordCount.scala:21, t
(hi,6), (hello,5), (spark,2), (sparkstreaming,1), (sparkgraphx,1), (sparksql,1)23/04/16 16:30:11 INFO Con
ated. Instead, use mapreduce.task.id
23/04/16 16:30:11 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapred
```

图 9: spark-submit 命令执行结果 (对应指导书图 46)

查看程序输出, 如下图 10 (画红线处) 所示:

```
root@firt-2020211616-0001:~
[root@firt-2020211616-0001 ~]# hadoop fs -ls /
23/04/16 16:44:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library
Found 7 items
drwxr-xr-x - root supergroup 0 2023-04-03 09:30 /HBase
-rw-r--r-- 3 root supergroup 65 2023-03-06 19:18 /firt_2020211616.txt
drwxr-xr-x - root supergroup 0 2023-04-16 16:44 /spark-test
drwxr-xr-x - root supergroup 0 2023-03-15 21:56 /testmr
drwx----- - root supergroup 0 2023-04-16 16:28 /tmp
-rw-r--r-- 3 root supergroup 65 2023-03-06 19:18 /upload_2020211616.txt
drwxr-xr-x - root supergroup 0 2023-04-16 16:11 /user
[root@firt-2020211616-0001 ~]#
```

图 10: 程序执行输出 (对应指导书图 47)

查看具体的输出内容, 如下图 11 所示:

```
root@firt-2020211616-0001:~
[root@firt-2020211616-0001 ~]# hadoop fs -cat /spark-test/part-00000
23/04/16 16:46:41 WARN util.NativeCodeLoader: Unable to load native-
(hi,6)
(hello,5)
[root@firt-2020211616-0001 ~]# hadoop fs -cat /spark-test/part-00001
23/04/16 16:46:46 WARN util.NativeCodeLoader: Unable to load native-
(spark,2)
[root@firt-2020211616-0001 ~]# hadoop fs -cat /spark-test/part-00002
23/04/16 16:46:49 WARN util.NativeCodeLoader: Unable to load native-
(sparksql,1)
(sparkstreaming,1)
(sparkgraphx,1)
[root@firt-2020211616-0001 ~]#
```

图 11: 具体输出内容 (对应指导书图 48)

2. 2. RDD 编程：编写独立应用程序实现数据去重

在这部分实验中，我需要编写代码，完成对如下图 12 和 13 所示的两个 txt 文件中数据的合并操作。



A.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
20170101 x
20170102 y
20170103 x
20170104 y
20170105 z
20170106 z

图 12: 待合并文件之 A.txt



B.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
20170101 y
20170102 y
20170103 x
20170104 z
20170105 y

图 13: 待合并文件之 B.txt

参考指导书内容，我编写了下面的代码用以完成目标功能：

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

class ScalaWordMerge{
}
object ScalaWordMerge{
  def main(args:Array[String]):Unit={
    val sparkConf = new SparkConf().setAppName("word-merge").setMaster("yarn")
    val sc = new SparkContext(sparkConf)
    val A = sc.textFile("A.txt")
    val B = sc.textFile("B.txt")
    val AB = A.union(B)
    val distinct_lines = AB.distinct()
    distinct_lines.saveAsTextFile("hdfs://node1:8020/C.txt")
    sc.stop
  }
}
```

在上面的代码中，我通过（val 型）A 记录从 A.txt 中读取的内容、（val 型）B 记录从 B.txt 读取的内容，并通过 union 方法将二者进行合并，然后使用 distinct() 方法去除重复内容，最后使用 saveAsTextFile 方法写入结果。使用 cat 命令可以看到合并后的内容（如下图 14 所示），可以发现合并成功：

```

root@firt-2020211616-0001:~
-rw-r--r-- 3 root supergroup 0 2023-04-16 17:26 /C.txt/_SUCCESS
-rw-r--r-- 3 root supergroup 33 2023-04-16 17:26 /C.txt/part-00000
-rw-r--r-- 3 root supergroup 11 2023-04-16 17:26 /C.txt/part-00001
-rw-r--r-- 3 root supergroup 22 2023-04-16 17:26 /C.txt/part-00002
-rw-r--r-- 3 root supergroup 33 2023-04-16 17:26 /C.txt/part-00003
[root@firt-2020211616-0001 ~]# hadoop fs -cat /C.txt/part-00000
23/04/16 17:27:49 WARN util.NativeCodeLoader: Unable to load native-hadoop li
classes where applicable
20170105 z
20170102 y
20170103 x
[root@firt-2020211616-0001 ~]# hadoop fs -cat /C.txt/part-00001
23/04/16 17:27:55 WARN util.NativeCodeLoader: Unable to load native-hadoop li
classes where applicable
20170106 z
[root@firt-2020211616-0001 ~]# hadoop fs -cat /C.txt/part-00002
23/04/16 17:28:00 WARN util.NativeCodeLoader: Unable to load native-hadoop li
classes where applicable
20170101 x
20170104 y
[root@firt-2020211616-0001 ~]# hadoop fs -cat /C.txt/part-00003
23/04/16 17:28:05 WARN util.NativeCodeLoader: Unable to load native-hadoop li
classes where applicable
20170104 z
20170101 y
20170105 y
[root@firt-2020211616-0001 ~]# Connection to firt-2020211616-0001 closed by rem

```

图 14: 文件合并结果 (对应指导书图 53)

2.3. 使用 Spark SQL 读写数据库

在该部分实验中, 我首先下载并安装了 MySQL, 然后启动 MySQL, 并通过指令查询 MySQL 运行状态如下图所示:

```

root@firt-2020211616-0001:~
mysql-community-server.aarch64 0:8.0.32-1.e17

Dependency Installed:
  libaio.aarch64 0:0.3.109-13.e17          mysql-community-client.aarch64 0:8.0.32-1.e17
  mysql-community-client-plugins.aarch64 0:8.0.32-1.e17      mysql-community-common.aarch64 0:8.0.32-1.e17
  mysql-community-icu-data-files.aarch64 0:8.0.32-1.e17

Dependency Updated:
  postfix.aarch64 2:2.10.1-9.e17

Replaced:
  mariadb-libs.aarch64 1:5.5.64-1.e17

Complete!
[root@firt-2020211616-0001 ~]# systemctl start mysqld.service
[root@firt-2020211616-0001 ~]# systemctl status mysqld.service
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2023-04-16 19:31:53 CST; 13s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
   Process: 2342 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
   Main PID: 2411 (mysqld)
   Status: "Server is operational"
   CGroup: /system.slice/mysqld.service
           └─2411 /usr/sbin/mysqld

Apr 16 19:31:48 firt-2020211616-0001 systemd[1]: Starting MySQL Server...
Apr 16 19:31:53 firt-2020211616-0001 systemd[1]: Started MySQL Server.
[root@firt-2020211616-0001 ~]#

```

图 15: MySQL 正常运行 (对应指导书图 54)

使用 `grep "password" /var/log/mysqld.log` 可以查看 root 用户的密码, 如下图 16 所示:

```
root@frt-2020211616-0001:~#
Apr 16 19:31:48 frt-2020211616-0001 systemd[1]: Starting MySQL Server...
Apr 16 19:31:53 frt-2020211616-0001 systemd[1]: Started MySQL Server.
[root@frt-2020211616-0001 ~]# grep "password" /var/log/mysqld.log
2023-04-16T11:31:50.389343Z 6 [Note] [MY-010454] [Server] A temporary password
root@localhost: wa*:dS=:0lid
[root@frt-2020211616-0001 ~]#
```

图 16: root 用户密码 (对应指导书图 55)

接下来进入数据库, 首先需要修改密码, 然后创建 spark 数据库, 在其中创建 student 表, 并通过 SQL 语句插入指定数据, 结果如下所示:

```
root@frt-2020211616-0001:~#
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'frtFRT189260';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'frtFRT189260.';
Query OK, 0 rows affected (0.00 sec)

mysql> create database spark;
Query OK, 1 row affected (0.01 sec)

mysql> use spark;
Database changed
mysql> create table student(id int(4), name char(20), gender char(4), age int(4));
Query OK, 0 rows affected, 2 warnings (0.02 sec)

mysql> insert into student values(1, 'Li', 'F', 23);
Query OK, 1 row affected (0.00 sec)

mysql> insert into student values(2, 'Wang', 'M', 24);
Query OK, 1 row affected (0.01 sec)

mysql> select * from student;
+----+-----+-----+-----+
| id | name  | gender | age  |
+----+-----+-----+-----+
| 1  | Li    | F      | 23   |
| 2  | Wang  | M      | 24   |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

图 17: 修改密码、创建数据库/表、插入数据 (对应指导书图 56)

接下来我安装了 MySQL 数据库驱动程序, 并进行解压和移动, 然后使用指定命令启动 (指定 MySQL 连接 jar 包的) spark shell, 完成数据库读取, 结果如下所示:

```
root@frt-2020211616-0001:~#
scala> val jdbcDF = spark.read.format("jdbc").
  option("url", "jdbc:mysql://localhost:3306/spark").
  option("driver", "com.mysql.jdbc.Driver").
  option("dbtable", "student").
  option("user", "root").
  option("password", "frtFRT189260.").
  load()
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new
r is automatically registered via the SPI and manual loading of th
jdbcDF: org.apache.spark.sql.DataFrame = [id: int, name: string ..

scala> jdbcDF.show()
23/04/16 20:09:51 WARN util.SizeEstimator: Failed to check whether
+----+-----+-----+-----+
| id | name  | gender | age  |
+----+-----+-----+-----+
| 1  | Li    | F      | 23   |
| 2  | Wang  | M      | 24   |
+----+-----+-----+-----+
```

图 18: 对 MySQL 数据库的读取结果 (对应指导书图 57)

并且，在通过 JDBC 连接 MySQL 数据库时，option() 方法中涉及到的参数含义如下表所示：

参数名称	参数的值	含义
url	jdbc:mysql://localhost:3306/spark	数据库的连接地址
driver	com.mysql.jdbc.driver	数据库的 JDBC 驱动
dbtable	student	所要访问的表
user	root	用户名
password		用户密码

我编写了如下所示的向数据库中插入数据的代码：

```
package org.example
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.{Row, SQLContext, types}
import org.apache.spark.sql.types.{IntegerType, StringType,
StructField, StructType}
import org.apache.spark.{SparkConf, SparkContext}
import java.sql.Struct
import java.util.Properties

class InsertStudent{
}

object InsertStudent{
  def main(args:Array[String]):Unit={
    val sparkConf = new SparkConf().setAppName("insert-student").setMaster("local")
    val sc = new SparkContext(sparkConf)
    val studentRDD = sc.parallelize(Array("3 Zhang M 26", "4 Liu M 27")).map(_._split(" "))
    val schema = StructType(List(
      StructField("id", IntegerType, true),
      StructField("name", StringType, true),
      StructField("gender", StringType, true),
      StructField("age", IntegerType, true)))
    val rowRDD = studentRDD.map(p => Row(p(0).toInt, p(1).trim, p(2).trim, p(3).toInt))
    val studentDF = new SQLContext(sc).createDataFrame(rowRDD, schema)
    val prop = new Properties()
    prop.put("user", "root")
    prop.put("password", "frtFRT189260.")
    prop.put("driver", "com.mysql.jdbc.Driver")
  }
}
```

```
studentDF.write.mode("append").jdbc("jdbc:mysql://localhost:3306/spark", "spark.student", prop)
}
```

在上面的代码中，studentRDD 定义了将要插入的 student 信息（即包括 Zhang 和 Liu 两个 student 信息），而 schema 定义了插入信息的格式，rowRDD 定义了 student 信息的“转换方式”，然后通过 put 方法设置必要信息，最后通过 write 方法将数据写入到/spark 下的 student 表中。

将此代码按照和之前一样的思路打包为 jar 包，上传到 node1 并运行，可以看到如下图 19 所示的结果，可以看到插入成功：

```
root@frr-2020211616-0001:~
mysql> use spark;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from student;
```

id	name	gender	age
1	Li	F	23
2	Wang	M	24
3	Zhang	M	26
4	Liu	M	27

```
4 rows in set (0.00 sec)
```

图 19: student 信息插入成功（对应指导书图 60）

3. 实验问题与实验总结

本次实验中我也遇到了一些问题并进行了解决，现记录如下：

- (1) 使用 IDEA 创建项目时 Archetype 选择错误，如果按照实验指导书选择 org.scala 下的 scala-archetype-simple，则会导致模板与实验内容不符，应当选择 net.alchim31 下的 scala-archetype-simple；
- (2) 编写去重功能时不知如何实现，通过研读指导书内容，结合相应函数功能，完成了代码编写。

在本次实验中，我在实验一、二、三的基础上安装了 Spark 和 MySQL 数据库驱动程序两个组件，并进行了相应的配置。我使用 scala 编写了计数功能代码、去重功能代码和数据库插入代码，调试排除了相应的错误，并打包为 jar 包上传到主节点上进行运行，得到了预期的正确结果。总的来说，我在本次实验中复习了理论知识、进行了工程实践，圆满完成了实验任务，收获颇丰！