课程实验四：Spark 及 Hive 离线批处理实践

一、实验描述

本实验使用 Scala 语言编写 Spark 程序，完成单词计数任务。首先，在华为云购买 4 台服务器，然后搭建 Hadoop 集群和 Spark 集群(YARN 模式)，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。在此基础之上，进一步搭建 hive，从 hive 中读取数据并进行词频统计

二、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

三、实验环境

1. 服务器节点数量：4
2. 系统版本：Centos
3. Hadoop 版本：Apache Hadoop 2.7.7
4. Spark 版本：Apache Spark 2.1.1
5. JDK 版本：1.8
6. Scala 版本：scala2.11.8
7. IDEA 版本：IntelliJ IDEA Educational Edition 2021.2.3
8. Hive 版本：2.1.1
9. MySQL 版本：5.7.30

四、实验步骤

4.1 Hadoop 集群环境测试

步骤 1：在构建好 Hadoop 平台的主节点开启集群： start-all.sh，并在四个节点的终端执行 jps 命令: 并执行 ifconfig。

输出　调试控制台　终端

```
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]# jps ; ifconfig
7675 NameNode
8077 ResourceManager
7894 SecondaryNameNode
8416 Jps
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.127  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fec2:7ceb  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:c2:7c:eb  txqueuelen 1000  (Ethernet)
        RX packets 1001031  bytes 1213746745 (1.1 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 236155  bytes 201957261 (192.6 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 8397  bytes 2655273 (2.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8397  bytes 2655273 (2.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0001 ~]#
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0002 ~]#
[root@rcx-2019211279-0002 ~]#
[root@rcx-2019211279-0002 ~]#
[root@rcx-2019211279-0002 ~]# jps ; ifconfig
4788 DataNode
5099 Jps
4908 NodeManager
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.80  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe22:570c  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:22:57:0c  txqueuelen 1000  (Ethernet)
        RX packets 540294  bytes 394846785 (376.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 348459  bytes 1574387597 (1.4 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 21278  bytes 223024212 (212.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 21278  bytes 223024212 (212.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0002 ~]#
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0003 ~]#
[root@rcx-2019211279-0003 ~]#
[root@rcx-2019211279-0003 ~]#
[root@rcx-2019211279-0003 ~]# jps ; ifconfig
4281 Jps
4093 NodeManager
3973 DataNode
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.58  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe85:a5d  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:85:0a:5d  txqueuelen 1000  (Ethernet)
        RX packets 544034  bytes 670490665 (639.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 188246  bytes 110567861 (105.4 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 1434  bytes 469958 (458.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1434  bytes 469958 (458.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0003 ~]#
```


```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0003 ~]#
[root@rcx-2019211279-0003 ~]#
[root@rcx-2019211279-0003 ~]#
[root@rcx-2019211279-0003 ~]# jps ; ifconfig
4281 Jps
4093 NodeManager
3973 DataNode
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.58  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe85:a5d  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:85:0a:5d  txqueuelen 1000  (Ethernet)
        RX packets 544034  bytes 670490665 (639.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 188246  bytes 110567861 (105.4 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 1434  bytes 469958 (458.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1434  bytes 469958 (458.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0003 ~]#
```

步骤 2：在主节点的 root 测试 Hadoop 的集群可用性，并执行 ifconfig。

```
hadoop jar /home/modules/hadoop-2.7.7/share/hadoop/mapreduce/hadoop-mapreduce-
examples-2.7.7.jar pi 10 10
```

```
[root@rcx-2019211279-0001 ~]# hadoop jar ./hadoop-2.7.7/share/hadoop/mapreduc
e/hadoop-mapreduce-examples-2.7.7.jar pi 10 10
Number of Maps  = 10
Samples per Map = 10
22/04/30 04:17:24 WARN util.NativeCodeLoader: Unable to load native-hadoop li
brary for your platform... using builtin-java 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
22/04/30 04:17:26 INFO client.RMProxy: Connecting to ResourceManager at rcx-2
```

```
                Reduce shuffle bytes=280
                Reduce input records=20
                Reduce output records=0
                Spilled Records=40
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=1852
                CPU time spent (ms)=3480
                Physical memory (bytes) snapshot=2388721664
                Virtual memory (bytes) snapshot=14173601792
                Total committed heap usage (bytes)=1692008448
        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 20.667 seconds
Estimated value of Pi is 3.20000000000000000000
[root@rcx-2019211279-0001 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.127  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fec2:7ceb  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:c2:7c:eb  txqueuelen 1000  (Ethernet)
        RX packets 1003293  bytes 1214148092 (1.1 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 237587  bytes 202626440 (193.2 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 8677  bytes 2723122 (2.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8677  bytes 2723122 (2.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@rcx-2019211279-0001 ~]#
```

4.2 Spark 集群搭建（On Yarn 模式）

步骤 1：在 master 节点上解压 spark 压缩包。

1) 使用 xftp 上传 spark 压缩包 spark-2.1.1-bin-hadoop2.7.tgz

2) 解压 spark 压缩包

需要提前下载 zip 和 unzip，再执行

unzip -xzvf spark-2.1.1-bin-hadoop2.7.zip

步骤 2：配置环境变量。

1) 返回 root 目录；

cd /root/

2) 用 vim 编辑 .bash_profile 文件：

vim .bash_profile

添加如下内容：

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HDFS_CONF_DIR=$HADOOP_HOME/etc/hadoop

export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop

export PATH=$PATH:/root/spark-2.1.1-bin-hadoop2.7/bin

3) 运行 source 命令，重新编译.bash_profile，使添加变量生效:

source .bash_profile

步骤 3：配置 yarn-site.xml 文件。

1) 进入/root/hadoop-2.7.7/etc/hadoop 目录：cd ./hadoop-2.7.7/etc/hadoop

2) 使用 vim 编辑 yarn-site.xml 添加如下内容

```
<property>
    <name>yarn.nodemanager.pmem-check-enabled</name>
    <value>false</value>
</property>
<property>
```

```
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
</property>
```

3) 将 yarn-site.xml 文件发送到从节点：

scp yarn-site.xml rcx-2019211279-0002:/home/modules/hadoop-2.7.7/etc/hadoop/yarn-site.xml

scp yarn-site.xml rcx-2019211279-0003:/home/modules/hadoop-2.7.7/etc/hadoop/yarn-site.xml

scp yarn-site.xml rcx-2019211279-0004:/home/modules/hadoop-2.7.7/etc/hadoop/yarn-site.xml

步骤 4：重启 Hadoop 集群。

1) stop-all.sh (如果你已经启动了 hadoop 需要停止)

2) start-all.sh

3) 使用 jps 查看集群是否启动成功(结果应与图 1-图 4 一致)；

步骤 5：运行如下指令检验 spark 是否部署成功：

需要先给与相应权限

```
chmod 777 spark-2.1.1-bin-hadoop2.7 -R
```

再进行检验

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn --num-executors 4 --driver-memory 1g --executor-memory 1g --executor-cores 1 spark-2.1.1-bin-hadoop2.7/examples/jars/spark-examples_2.11-2.1.1.jar 10
```

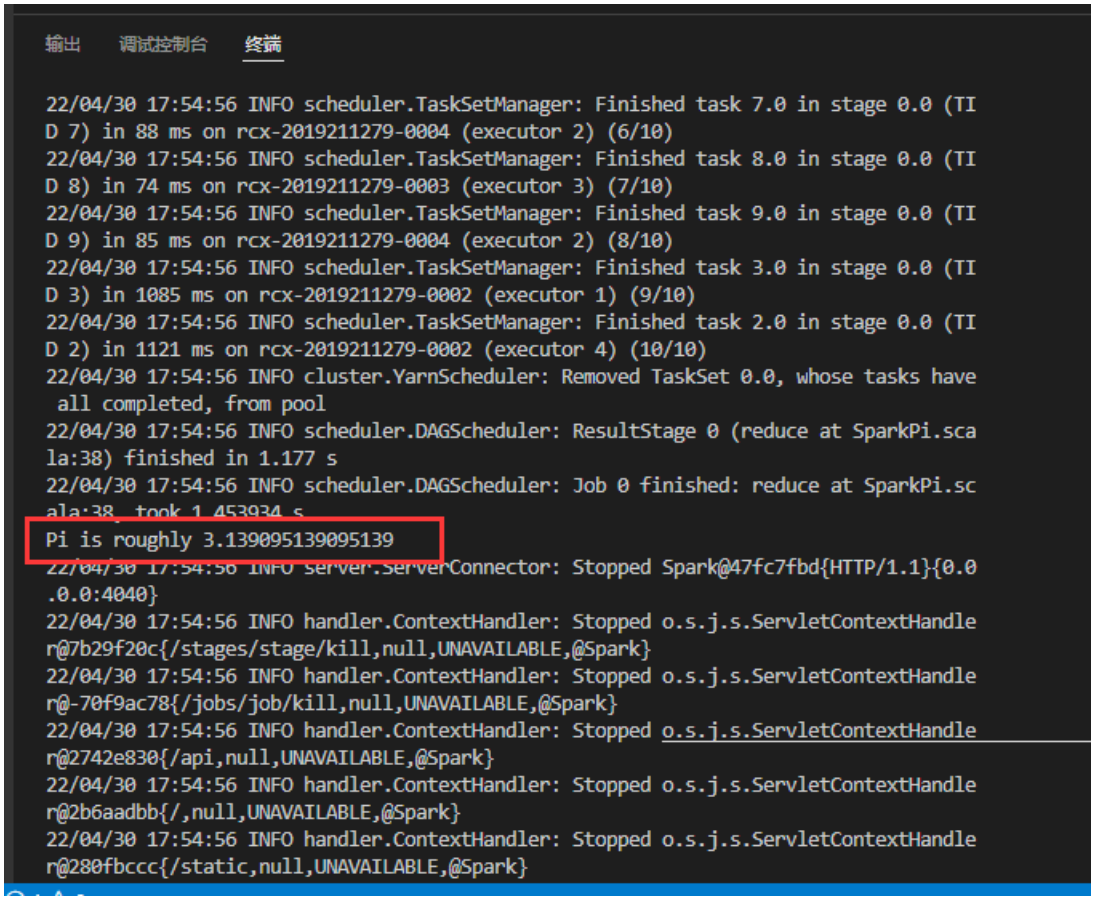


图 7

步骤 6：运行 spark-shell 命令，查看 spark 和 scala 版本信息

```
输出    调试控制台    终端

22/04/30 17:54:56 INFO util.ShutdownHookManager: Shutdown hook called
22/04/30 17:54:56 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-ccf98240-d094-4
4a5-9778-313764d4e59f
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]#
[root@rcx-2019211279-0001 ~]# spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/04/30 18:00:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pl
atform... using builtin-java classes where applicable
22/04/30 18:01:05 WARN metastore.ObjectStore: Failed to get database global_temp, returning No
SuchObjectException
Spark context Web UI available at http://192.168.0.127:4040
Spark context available as 'sc' (master = local[*], app id = local-1651312859268).
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.

scala>
```

图 8

可以看到 spark 版本为 2.1.1，scala 版本为 2.11.8。

4.3 Scala 程序编写

步骤 1：

创建项目，打开 IDEA，创建工程，依次输入 GroupId、ArtifactId 和 Version 的值，然后点击 next；

步骤 2：依赖设置：

1) 在 pom.xml 文件中找到 properties 配置项，修改 scala 版本号（此处对应 scala 安装版本），并添加 spark 版本号（此处对应 spark 安装版本）；

2) 找到 dependency 配置项，添加配置，分别是 scala 依赖和 spark 依赖。

3) 修改 pom.xml 文件后，点击 enable auto-import，根据 pom.xml 文件导入依赖包

步骤 3：设置语言环境：

1) 设置语言环境 language level，点击菜单栏中的 file，选择 Project Structure；选择 Modules，选择 Language level 为 8，然后点击 Apply， 点击 OK；

步骤 4：设置 java Compiler 环境：

1) 点击菜单栏中的 file，选择 Setting；

2) 依次选择 Build，Execution—->Compiler—->Java Compiler，设置 Project bytecode version 为 1.8，设置图中的 Target bytecode version 为 1.8，然后依次点击 Apply 和 OK；

步骤 5：文件配置：

1) 删除测试环境 test 中的测试类；

2) 删除 main 文件夹中，包名下的 App 文件；

步骤 6： 程序编写：

1) 依次打开 src—>main—>scala，在 org.zkpk.lab 上点击右键，创建 Scala Class；

2) 输入类名 ScalaWordCount

3) 在类 ScalaWordCount 中新建伴生对象 object ScalaWordCount；

4) 在伴生对象 object ScalaWordCount 中创建 main 方法;

5) 在 main 方法中创建列表 List 对象并赋值给常量 list

6) 创建 SparkConf 对象，对 Spark 运行属性进行配置，调用该对象的 setAppName 方法设置 Spark 程序的名称为“word-count”，调用 setMaster 方法设置 Spark 程序运行模式，一般分为两种：本地模式和 yarn 模式，这里我们采用 yarn 模式，参数为“yarn”，属性设置完成后赋值给常量 sparkConf；

7) 创建 SparkContext，参数为 sparkconf，赋值给常量 sc，该对象是 Spark 程序的入口；

8) 调用 SparkContext 对象 sc 的方法 parallelize，参数为列表对象 list，该方法使用现成的 scala 集合生成 RDD lines，类型为 String（RDD 为 Spark 计算中的最小数据单元），该 RDD 存储元素是字符串语句；

9) 调用 RDD 对象 lines 的 flatMap 方法按照空格切分 RDD 中的字符串元素，并存入新的 RDD 对象 words 中，参数类型为 String，该 RDD 存储的元素是每一个单词；

10) 调用 RDD 对象 words 的 map 方法，将 RDD 中的每一个单词转换为 kv 对，key 是 String 类型的单词，value 是 Int 类型的 1，并赋值给新的 RDD 对象 wordAndOne，参数为（String，Int）类型键值对；

11) 调用 RDD 对象 wordAndOne 的 reduceByKey 方法，传入的参数为两个 Int 类型变量，该方法将 RDD 中的元素按照 Key 进行分组，将同一组中的 value 值进行聚合操作，得到 valueRet，最终返回（key，valueRet）键值对，并赋值给新的 RDD 对象 wordAndNum，参数为（String，Int）类型键值对；

12) 调用 RDD 对象 wordAndNum 的 sortBy 方法，第一个参数为 kv 对中的 value，即单词出现次数，第二个参数为 boolean 类型，true 表示升序，false 表示降序；

13) 调用 ret 对象的 collect 方法，获取集合中的元素，再调用 mkString 方法，参数为“，”，将集合中的元素用逗号连接成字符串，调用 println 方法打印输出在控制台；

14) 调用 ret 对象的 saveAsTextFile，该方法的参数为运行时指定的参数，此方法的用处是将 Spark 程序运行的结果保存到指定路径，一般是把结果保存到 HDFS 中，所以这里的参数定义为：hdfs://rcx-2019211279-0001:8020 /spark_test，HDFS 根目录中不存在 spark_test 此目录， spark 程序会自动创建该目录；调用 SparkContext 对象 sc 的 stop 方法，释放 spark 程序所占用的资源；

```
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",
            "hi hello spark sparksql",
            "hello hello hi sparkstream",
            "hi hi hello 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(" ")})
        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://rcx-2019211279-0001:8020/spark_test")
        sc.stop()
    }
}
```

4.4 程序打包与运行

步骤 1：打开 File->Project Structure：

步骤 2：Project Settings 栏下的 Artifacts，点击“+”,选择 JAR->From modules with dependencies…:

步骤 3：填选主类名称：

步骤 4：选择 Build->Artifacts:

步骤 5：选择 Build：建立完成

步骤 6：使用压缩软件打开生成的 jar 包：

步骤 7：找到 META-INF 目录

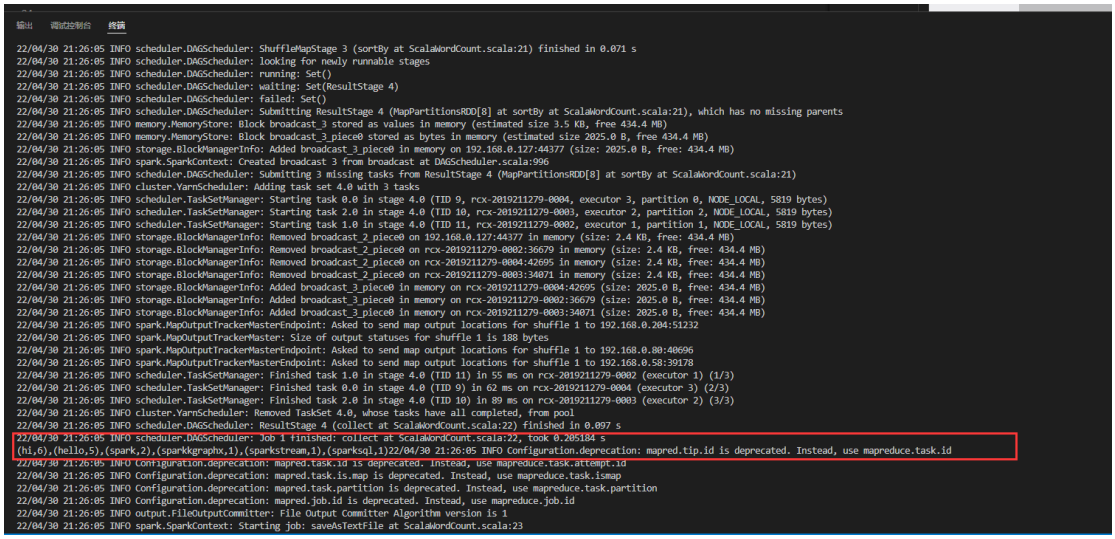
步骤 8：删除 MANIFEST.MF 文件

步骤 9：使用 WinScp 上传处理后的 jar 包到服务器：

步骤 10：使用 spark-submit 命令，在 hadoop 运行程序：

```
spark-submit --class org.example.ScalaWordCount --master yarn --num-executors 3 --driver-memory 1g -
-executor-memory 1g --executor-cores 1 spark-test.jar
```

得到如下结果：



```
输出    调试控制台    终端
22/04/30 21:26:05 INFO scheduler.DAGScheduler: ShuffleMapStage 3 (sortBy at ScalaWordCount.scala:21) finished in 0.071 s
22/04/30 21:26:05 INFO scheduler.DAGScheduler: looking for newly runnable stages
22/04/30 21:26:05 INFO scheduler.DAGScheduler: running: Set()
22/04/30 21:26:05 INFO scheduler.DAGScheduler: waiting: Set(ResultStage 4)
22/04/30 21:26:05 INFO scheduler.DAGScheduler: failed: Set()
22/04/30 21:26:05 INFO scheduler.DAGScheduler: Submitting ResultStage 4 (MapPartitionsRDD[8] at sortBy at ScalaWordCount.scala:21), which has no missing parents
22/04/30 21:26:05 INFO memory.MemoryStore: Block broadcast_3 stored as values in memory (estimated size 3.5 KB, free 434.4 MB)
22/04/30 21:26:05 INFO memory.MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 2025.0 B, free 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on 192.168.0.127:44377 (size: 2025.0 B, free: 434.4 MB)
22/04/30 21:26:05 INFO spark.SparkContext: Created broadcast 3 from broadcast at DAGScheduler.scala:996
22/04/30 21:26:05 INFO scheduler.DAGScheduler: Submitting 3 missing tasks from ResultStage 4 (MapPartitionsRDD[8] at sortBy at ScalaWordCount.scala:21)
22/04/30 21:26:05 INFO cluster.YarnScheduler: Adding task set 4.0 with 3 tasks
22/04/30 21:26:05 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 4.0 (TID 9, rcx-2019211279-0004, executor 3, partition 0, NODE_LOCAL, 5819 bytes)
22/04/30 21:26:05 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 4.0 (TID 10, rcx-2019211279-0003, executor 2, partition 2, NODE_LOCAL, 5819 bytes)
22/04/30 21:26:05 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 4.0 (TID 11, rcx-2019211279-0002, executor 1, partition 1, NODE_LOCAL, 5819 bytes)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on 192.168.0.127:44377 in memory (size: 2.4 KB, free: 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on rcx-2019211279-0002:36679 in memory (size: 2.4 KB, free: 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on rcx-2019211279-0004:42695 in memory (size: 2.4 KB, free: 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on rcx-2019211279-0003:34071 in memory (size: 2.4 KB, free: 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on rcx-2019211279-0004:42695 (size: 2025.0 B, free: 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on rcx-2019211279-0002:36679 (size: 2025.0 B, free: 434.4 MB)
22/04/30 21:26:05 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on rcx-2019211279-0003:34071 (size: 2025.0 B, free: 434.4 MB)
22/04/30 21:26:05 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 1 to 192.168.0.204:51232
22/04/30 21:26:05 INFO spark.MapOutputTrackerMaster: Size of output statuses for shuffle 1 is 188 bytes
22/04/30 21:26:05 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 1 to 192.168.0.80:40696
22/04/30 21:26:05 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 1 to 192.168.0.58:39178
22/04/30 21:26:05 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 4.0 (TID 11) in 55 ms on rcx-2019211279-0002 (executor 1) (1/3)
22/04/30 21:26:05 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 4.0 (TID 9) in 62 ms on rcx-2019211279-0004 (executor 3) (2/3)
22/04/30 21:26:05 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 4.0 (TID 10) in 89 ms on rcx-2019211279-0003 (executor 2) (3/3)
22/04/30 21:26:05 INFO cluster.YarnScheduler: Removed TaskSet 4.0, whose tasks have all completed, from pool
22/04/30 21:26:05 INFO scheduler.DAGScheduler: ResultStage 4 (collect at ScalaWordCount.scala:22) finished in 0.097 s
22/04/30 21:26:05 INFO scheduler.DAGScheduler: Job 1 finished: collect at ScalaWordCount.scala:22, took 0.205184 s
(hi,6),(hello,5),(spark,2),(sparkkgraphx,1),(sparkstream,1),(sparksql,1)22/04/30 21:26:05 INFO Configuration.deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
22/04/30 21:26:05 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
22/04/30 21:26:05 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
22/04/30 21:26:05 INFO Configuration.deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
22/04/30 21:26:05 INFO Configuration.deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
22/04/30 21:26:05 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
22/04/30 21:26:05 INFO spark.SparkContext: Starting job: saveAsTextFile at ScalaWordCount.scala:23
```

图 46

步骤 11：在 hdfs 上查看程序的输出：

```
输出    调试控制台    终端
22/04/30 21:26:07 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/04/30 21:26:07 INFO memory.MemoryStore: MemoryStore cleared
22/04/30 21:26:07 INFO storage.BlockManager: BlockManager stopped
22/04/30 21:26:07 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
22/04/30 21:26:07 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/04/30 21:26:07 INFO spark.SparkContext: Successfully stopped SparkContext
22/04/30 21:26:07 INFO util.ShutdownHookManager: Shutdown hook called
22/04/30 21:26:07 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-0a78f21e-fb1a-41cb-97a7-1de3cfdc9268
[root@rcx-2019211279-0001 ~]# hadoop fs -ls -R /
22/04/30 21:27:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
drwxr-xr-x   - root supergroup          0 2022-04-30 21:26 /spark_test
-rw-r--r--   3 root supergroup          0 2022-04-30 21:26 /spark_test/_SUCCESS
-rw-r--r--   3 root supergroup         17 2022-04-30 21:26 /spark_test/part-00000
-rw-r--r--   3 root supergroup         10 2022-04-30 21:26 /spark_test/part-00001
-rw-r--r--   3 root supergroup         46 2022-04-30 21:26 /spark_test/part-00002
drwx------   - root supergroup          0 2022-04-30 21:11 /tmp
drwx------   - root supergroup          0 2022-04-30 21:11 /tmp/hadoop-yarn
drwx------   - root supergroup          0 2022-04-30 21:11 /tmp/hadoop-yarn/staging
drwxr-xr-x   - root supergroup          0 2022-04-30 21:11 /tmp/hadoop-yarn/staging/history
drwxrwxrwt   - root supergroup          0 2022-04-30 21:11 /tmp/hadoop-yarn/staging/history/done_intermediate
drwxrwx---   - root supergroup          0 2022-04-30 21:11 /tmp/hadoop-yarn/staging/history/done_intermediate/root
```

图 47

```
drwxr-xr-x   - root supergroup          0 2022-04-30 20:54 /user
drwxr-xr-x   - root supergroup          0 2022-04-30 21:14 /user/root
drwxr-xr-x   - root supergroup          0 2022-04-30 21:26 /user/root/.sparkStaging
[root@rcx-2019211279-0001 ~]# hadoop fs -cat /spark_test/part-00000
22/04/30 21:28:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(hi,6)
(hello,5)
[root@rcx-2019211279-0001 ~]# hadoop fs -cat /spark_test/part-00001
22/04/30 21:28:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(spark,2)
[root@rcx-2019211279-0001 ~]# hadoop fs -cat /spark_test/part-00002
22/04/30 21:28:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(sparkkgraphx,1)
(sparkstream,1)
(sparksql,1)
```

图 48

五、使用 hive 数据源进行 wordcount

第一节：

本实验安装 MySQL 是为了给 Hive 提供元数据存储库，主要包括：yum 安装 MySQL、修改 MySQL root 密码、添加 zkpk 用户并赋予远程访问权限、修改数据库默认编码

1：查看并卸载系统自带的 mariadb-lib 数据库：

rpm -qa|grep mariadb mariadb-5.5.68-1.el7.aarch64

yum -y remove mariadb-*

2：添加 MySQLyum 源

1) 使用 WinSCP 上传 mysql 安装包

2) 安装 mysql 所需依赖

yum install -y perl openssl openssl-devel libaio perl-JSON autoconf

3)解压 mysql 安装包：

tar -xvf mysql-5.7.30.tar.gz

4)进入 aarch64 目录，对 rpm 包进行安装:

cd aarch64

yum install *.rpm

3：启动 MySQL 服务：

1)命令

systemctl start mysqld

2)查看启动状态

systemctl status mysqld

```
输出    调试控制台    终端

  mysql-community-embedded-compat.aarch64 0:5.7.30-1.el7.centos.a
  mysql-community-embedded-devel.aarch64 0:5.7.30-1.el7.centos.a
  mysql-community-libs.aarch64 0:5.7.30-1.el7.centos.a
  mysql-community-libs-compat.aarch64 0:5.7.30-1.el7.centos.a
  mysql-community-server.aarch64 0:5.7.30-1.el7.centos.a
  mysql-community-test.aarch64 0:5.7.30-1.el7.centos.a

Complete!
[root@rcx-2019211279-0001 aarch64]# systemctl start mysqld
[root@rcx-2019211279-0001 aarch64]# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2022-04-30 21:39:55 CST; 9s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
  Process: 11523 ExecStart=/usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid $MYSQLD_OPTS
(code=exited, status=0/SUCCESS)
  Process: 11473 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
 Main PID: 11526 (mysqld)
   CGroup: /system.slice/mysqld.service
           └─11526 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid

Apr 30 21:39:51 rcx-2019211279-0001 systemd[1]: Starting MySQL Server...
Apr 30 21:39:55 rcx-2019211279-0001 systemd[1]: Started MySQL Server.
```

图 49

4：修改 root 默认密码：

1) 查看 mysql 安装生成的随机默认密码

```
输出    调试控制台    终端

(code=exited, status=0/SUCCESS)
  Process: 11473 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
 Main PID: 11526 (mysqld)
   CGroup: /system.slice/mysqld.service
           └─11526 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid

Apr 30 21:39:51 rcx-2019211279-0001 systemd[1]: Starting MySQL Server...
Apr 30 21:39:55 rcx-2019211279-0001 systemd[1]: Started MySQL Server.
[root@rcx-2019211279-0001 aarch64]# grep 'temporary password' /var/log/mysqld.log
2022-04-30T13:39:52.819172Z 1 [Note] A temporary password is generated for root@localhost: o0YjK?RuWFsr
[root@rcx-2019211279-0001 aarch64]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.30

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

图 50

2) 登录 mysql

mysql -uroot -p

3) 修改 mysql 密码为:Rcx.1229

ALTER USER 'root'@'localhost' IDENTIFIED BY 'Rcx.1229';

5：修改 mysql 密码策略

1) 查看 msyql 密码策略的相关信息:

show variables like '%password%';

6：关闭密码策略；

1) 禁用密码策略，向 my.cnf 文件中[mysqld]下添加如下配置（/etc/my.cnf）：

vim /etc/my.cnf

[mysqld]

validate_password = off

2) 重新启动 mysql 服务使配置生效：

systemctl restart mysqld

7：配置默认编码为 utf8

1)修改/etc/my.cnf 配置文件，在[mysqld]下添加编码配置;

2)并且在 my.cnf 中添加 client 模块，输入 client 模块的相关编码格式;

vim /etc/my.cnf

```
validate_password = off

init_connect='SET NAMES utf8'

#

# Remove leading # and set to the amount of RAM for the most important data

# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%. # innodb_buffer_pool_size = 128M

# Remove leading # to turn on a very important data integrity option: logging # changes to the binary log between backups.

# log_bin

#

# Remove leading # to set options mainly useful for reporting servers.

# The server defaults are faster for transactions and fast SELECTs.

# Adjust sizes as needed, experiment to find the optimal values.
```

```
# join_buffer_size = 128M

# sort_buffer_size = 2M

# read_rnd_buffer_size = 2M

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted security risks symbolic-links=0

log-error=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

[client]

default-character-set=utf8
```

3)重新启动 mysql 服务

systemctl restart mysqld

4)登录 mysql:

mysql -uroot -p

5)查看编码

show variables like '%character%';

```
78

输出    调试控制台    终端

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

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)
```

图 52

第二节：

本节内容是 Hive 安装部署，主要内容包括：启动 hadoop 集群、解压并安装 Hive、创建 Hive 的元数据库、修改配置文件、添加并生效环境变量、初始化元数据

1：启动 Hadoop 集群

在 master 启动 Hadoop 集群:

start-all.sh

在 master、slave01、slave02 运行 JPS 指令，查看 Hadoop 是否启动成功；

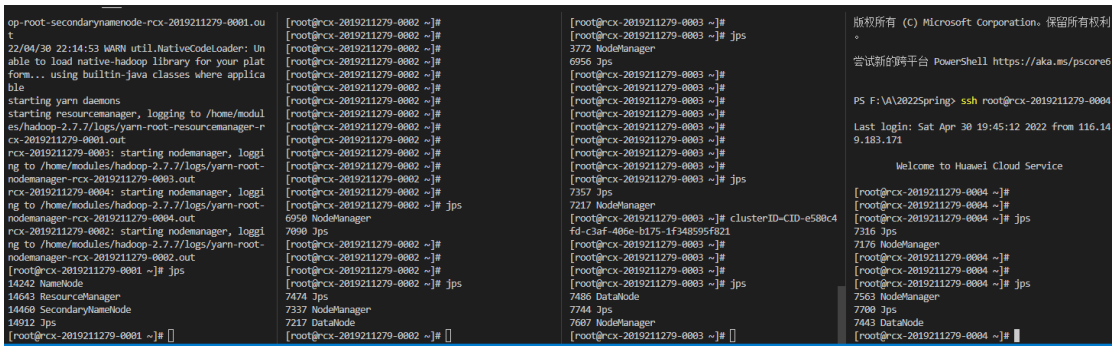


图 52-54

2:解压并安装 Hive

1) 使用 WinScp 上传 apache-hive-2.1.1-bin.tar.gz

2) 解压并安装 Hive

tar -zxvf /root/apache-hive-2.1.1-bin.tar.gz

3: 向 MySQL 中添加 hadoop 用户和创建名为（hive）的数据库；

1) 登录 mysql

mysql –uroot -p

2) 创建 hadoop 用户（密码：hadoop）:

grant all on *.* to hadoop@'%' identified by 'hadoop';

grant all on *.* to hadoop@'localhost' identified by 'hadoop';

grant all on *.* to hadoop@'master' identified by 'hadoop';

flush privileges;

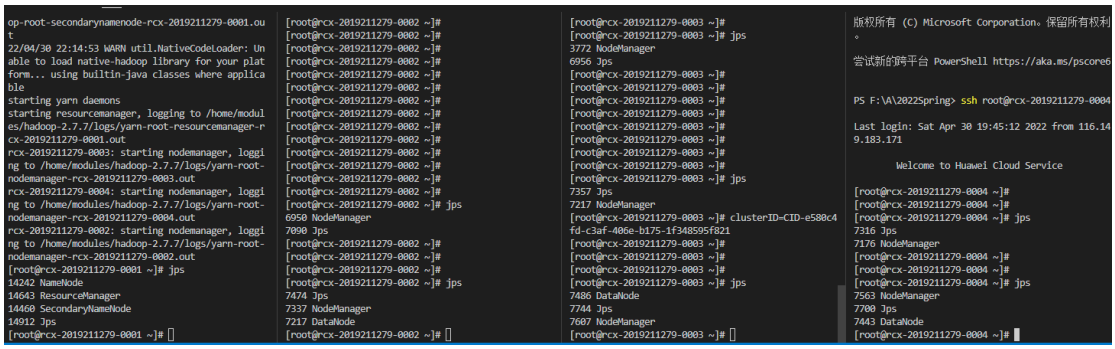


图 55

3) 创建数据库连接

create database hive;

4：配置 Hive

1) 进入 hive 安装目录下的配置目录:

cd /root/apache-hive-2.1.1-bin/conf/

2) 创建 hive 配置文件:

vim hive-site.xml

3) 添加如下内容：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>hive.metastore.local</name>
<value>true</value>
</property>
<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://rcx-2019211279-
0001:3306/hive?characterEncoding=UTF-8</value>
</property>
<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
</property>
<property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>hadoop</value>
</property>
```

```
<property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>hadoop</value>
</property>
<property>
    <name>hive.server2.authentication</name>
    <value>NOSASL</value>
</property>
<property>
    <name>hive.server2.enable.doAs</name>
    <value>false</value>
    </property>
</configuration>
```

5：复制 MySQL 连接驱动到 hive 根目录下的 lib 目录中：

cp /root/mysql-connector-java-5.1.28.jar /root/apache-hive-2.1.1-bin/lib/

cd apache-hive-2.1.1-bin/lib/

ll | grep mysql-connector-java-5.1.28.jar

6：配置系统 zkpk 用户环境变量

1) 打开配置文件：

cd ~

vim /root/.bash_profile

2) 将下面两行配置添加到环境变量中:

#HIVE

export HIVE_HOME=/root/apache-hive-

2.1.1-bin export

PATH=$PATH:$HIVE_HOME/bin

3) 使环境变量生效

source /root/.bash_profile

7：启动并验证 Hive 安装：

1) 初始化 Hive 元数据库

说明：该命令是把 hive 的元数据都同步到 mysql 中

schematool -dbType mysql -initSchema

8：修改 Hadoop 集群配置

1) 在/root/hadoop-2.7.3/etc/hadoop 路径下，修改文件 core-site.xml 下，添加如下内容：

```
<property>
<name>hadoop.proxyuser.root.hosts</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.root.groups</name>
  <value>*</value>
</property>
```

9 开启 Hive 远程模式：

hive --service metastore &

hive --service hiveserver2 &

10：hive 建库并导入数据

使用 winScp 或其他工具将 text.txt 文件传至服务器中

使用 hive 命令进入 hive 命令行进行建库和数据导入操作

```
SLF4J: Found binding in [jar:file:/root/apache-hive-2.1.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLog
gerBinder.class]
SLF4J: Found binding in [jar:file:/home/modules/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf
4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/root/apache-hive-2.1.1-bin/lib/hive-common-2.1.1.jar!/hive-log4j2
.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different executi
on engine (i.e. spark, tez) or using Hive 1.X releases.
hive> create database spark;
OK
Time taken: 1.306 seconds
hive> use spark;
OK
Time taken: 0.017 seconds
hive> create external table wordcount(content string) STORED AS TEXTFILE LOCATION '/spark/wordcount';
OK
Time taken: 0.379 seconds
hive> select * from wordcount limit 10;
OK
hello hi hi spark
hi hello spark sparksql

hello hello hi sparkstream

hi hi hello sparkkgraphx
Time taken: 0.996 seconds, Fetched: 6 row(s)
hive>
```

图 59

11：修改 wordcount 程序：

```
package org.example

import org.apache.spark.rdd.RDD

import org.apache.spark.sql.jdbc.{JdbcDialect, JdbcDialects}

import org.apache.spark.sql.{Row, SparkSession}

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




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://rcx-2019211279-0001:10000/spark;auth=noSasl")
      .option("user","root")
      .option("fetchsize","2000")
      .option("dbtable","spark.wordcount")
      .load()
    df.show(10)



    //val sparkConf = new SparkConf().setAppName("word-count").setMaster("yarn")
    //val sc = new SparkContext(sparkConf)
    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://rcx-2019211279-0001:8020/spark_test")
    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(".")

    }

    }

  }
```

12：运行程序并查看结果： jar 包打包方式和运行方式同前面相同

```
spark-submit --class org.example.ScalaWordCount --master yarn --num-executors 3 --
driver-memory 1g --executor-memory 1g --executor-cores 1 spark-test2.jar
```

```
22/05/01 00:09:37 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@-127a3460{/SQL/json,null,AVAILABLE,@Spark}
22/05/01 00:09:37 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@19f50fd0{/SQL/execution,null,AVAILABLE,@Spark}
22/05/01 00:09:37 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@-7d6dbc66{/SQL/execution/json,null,AVAILABLE,@Spark}
22/05/01 00:09:37 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@-55b7819e{/static/sql,null,AVAILABLE,@Spark}
22/05/01 00:09:37 INFO jdbc.Utils: Supplied authorities: rcx-2019211279-0001:10000
22/05/01 00:09:37 INFO jdbc.Utils: Resolved authority: rcx-2019211279-0001:10000
22/05/01 00:09:37 INFO jdbc.HiveConnection: Will try to open client transport with JDBC Uri: jdbc:hive2://rcx-2019211279-0001:10000/spark;auth=noSasl
OK
22/05/01 00:09:39 INFO codegen.CodeGenerator: Code generated in 306.014202 ms
22/05/01 00:09:39 INFO spark.SparkContext: Starting job: show at ScalaWordCount.scala:25
22/05/01 00:09:39 INFO scheduler.DAGScheduler: Got job 0 (show at ScalaWordCount.scala:25) with 1 output partitions
22/05/01 00:09:39 INFO scheduler.DAGScheduler: Final stage: ResultStage 0 (show at ScalaWordCount.scala:25)
22/05/01 00:09:39 INFO scheduler.DAGScheduler: Parents of final stage: List()
22/05/01 00:09:39 INFO scheduler.DAGScheduler: Missing parents: List()
22/05/01 00:09:39 INFO scheduler.DAGScheduler: Submitting ResultStage 0 (MapPartitionsRDD[2] at show at ScalaWordCount.scala:25), which has no missin
 parents
22/05/01 00:09:39 WARN util.SizeEstimator: Failed to check whether UseCompressedOops is set; assuming yes
22/05/01 00:09:39 INFO memory.MemoryStore: Block broadcast_0 stored as values in memory (estimated size 6.9 KB, free 434.4 MB)
22/05/01 00:09:40 INFO memory.MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 3.9 KB, free 434.4 MB)
22/05/01 00:09:40 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on 192.168.0.127:45151 (size: 3.9 KB, free: 434.4 MB)
22/05/01 00:09:40 INFO spark.SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:996
22/05/01 00:09:40 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 0 (MapPartitionsRDD[2] at show at ScalaWordCount.scala:25)
22/05/01 00:09:40 INFO cluster.YarnScheduler: Adding task set 0.0 with 1 tasks
22/05/01 00:09:40 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, rcx-2019211279-0004, executor 2, partition 0, PROCESS_LOCAL,
792 bytes)
22/05/01 00:09:40 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on rcx-2019211279-0004:46053 (size: 3.9 KB, free: 434.4 MB)
OK
22/05/01 00:09:41 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1826 ms on rcx-2019211279-0004 (executor 2) (1/1)
22/05/01 00:09:41 INFO cluster.YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
22/05/01 00:09:41 INFO scheduler.DAGScheduler: ResultStage 0 (show at ScalaWordCount.scala:25) finished in 1.842 s
22/05/01 00:09:41 INFO scheduler.DAGScheduler: Job 0 finished: show at ScalaWordCount.scala:25, took 2.135306 s
22/05/01 00:09:41 INFO codegen.CodeGenerator: Code generated in 40.961904 ms
+--------------------+
|   wordcount.content|
+--------------------+
|   hello hi hi spark|
|hi hello spark sp...|
|hello hello hi sp...|
|hi hi hello spark...|
+--------------------+
```

图 63-1

```
22/05/01 00:09:44 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 1 to 192.168.0.58:57450
22/05/01 00:09:44 INFO spark.MapOutputTrackerMaster: Size of output statuses for shuffle 1 is 150 bytes
22/05/01 00:09:44 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 2.0 (TID 2) in 146 ms on rcx-2019211279-0003 (executor 1) (1/1)
22/05/01 00:09:44 INFO cluster.YarnScheduler: Removed TaskSet 2.0, whose tasks have all completed, from pool
22/05/01 00:09:44 INFO scheduler.DAGScheduler: ShuffleMapStage 2 (sortBy at ScalaWordCount.scala:34) finished in 0.149 s
22/05/01 00:09:44 INFO scheduler.DAGScheduler: looking for newly runnable stages
22/05/01 00:09:44 INFO scheduler.DAGScheduler: running: Set()
22/05/01 00:09:44 INFO scheduler.DAGScheduler: waiting: Set(ResultStage 3)
22/05/01 00:09:44 INFO scheduler.DAGScheduler: failed: Set()
22/05/01 00:09:44 INFO scheduler.DAGScheduler: Submitting ResultStage 3 (MapPartitionsRDD[13] at sortBy at ScalaWordCount.scala:34), which has no mi
ing parents
22/05/01 00:09:44 INFO memory.MemoryStore: Block broadcast_3 stored as values in memory (estimated size 3.5 KB, free 434.4 MB)
22/05/01 00:09:44 INFO memory.MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 2017.0 B, free 434.4 MB)
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on 192.168.0.127:45151 (size: 2017.0 B, free: 434.4 MB)
22/05/01 00:09:44 INFO spark.SparkContext: Created broadcast 3 from broadcast at DAGScheduler.scala:996
22/05/01 00:09:44 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 3 (MapPartitionsRDD[13] at sortBy at ScalaWordCount.scala
4)
22/05/01 00:09:44 INFO cluster.YarnScheduler: Adding task set 3.0 with 1 tasks
22/05/01 00:09:44 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 3, rcx-2019211279-0003, executor 1, partition 0, NODE_LOCAL, 58
 bytes)
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on rcx-2019211279-0003:45585 (size: 2017.0 B, free: 434.4 MB)
22/05/01 00:09:44 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 0 to 192.168.0.58:57450
22/05/01 00:09:44 INFO spark.MapOutputTrackerMaster: Size of output statuses for shuffle 0 is 150 bytes
22/05/01 00:09:44 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 3.0 (TID 3) in 74 ms on rcx-2019211279-0003 (executor 1) (1/1)
22/05/01 00:09:44 INFO cluster.YarnScheduler: Removed TaskSet 3.0, whose tasks have all completed, from pool
22/05/01 00:09:44 INFO scheduler.DAGScheduler: ResultStage 3 (collect at ScalaWordCount.scala:35) finished in 0.075 s
22/05/01 00:09:44 INFO scheduler.DAGScheduler: Job 1 finished: collect at ScalaWordCount.scala:35, took 2.614075 s
(hi,6),(hello,5),(spark,2),(sparkstream,1),(sparksql,1),(sparkkgraphx,1)22/05/01 00:09:44 INFO storage.BlockManagerInfo: Removed broadcast_3_piece0 o
 192.168.0.127:45151 in memory (size: 2017.0 B, free: 434.4 MB)
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Removed broadcast_3_piece0 on rcx-2019211279-0003:45585 in memory (size: 2017.0 B, free: 434.4 MB)
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Removed broadcast_1_piece0 on 192.168.0.127:45151 in memory (size: 6.0 KB, free: 434.4 MB)
22/05/01 00:09:44 INFO Configuration.deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
22/05/01 00:09:44 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Removed broadcast_1_piece0 on rcx-2019211279-0003:45585 in memory (size: 6.0 KB, free: 434.4 MB)
22/05/01 00:09:44 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
22/05/01 00:09:44 INFO Configuration.deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
22/05/01 00:09:44 INFO Configuration.deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
22/05/01 00:09:44 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on 192.168.0.127:45151 in memory (size: 2.4 KB, free: 434.4 MB)
22/05/01 00:09:44 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on rcx-2019211279-0003:45585 in memory (size: 2.4 KB, free: 434.4 MB)
22/05/01 00:09:44 INFO spark.ContextCleaner: Cleaned accumulator 1
22/05/01 00:09:44 INFO spark.ContextCleaner: Cleaned accumulator 0
22/05/01 00:09:44 INFO spark.SparkContext: Starting job: saveAsTextFile at ScalaWordCount.scala:36
```

图 63-2

```
22/05/01 00:12:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   3 root supergroup          0 2022-05-01 00:09 /spark_test/_SUCCESS
-rw-r--r--   3 root supergroup         73 2022-05-01 00:09 /spark_test/part-00000
[root@rcx-2019211279-0001 ~]# hdfs dfs -cat /spark_test/part-00000
22/05/01 00:12:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(hi,6)
(hello,5)
(spark,2)
(sparkstream,1)
(sparksql,1)
(sparkkgraphx,1)
```

图 63-3

使用如下命令将结果从 hdfs 中导出：

hdfs dfs -get /spark_test/part-00000 /root/lab4/