

课程实验三：Spark Core Scala 单词计数

实验时间：2021 年 04 月 17 日

实验学生姓名：李志毅

学生班号、学号：2018211314 班 2018211582

一、实验结果截图

首先我本人华为云上三个服务器的内网 Ipv4 地址分别为：

名称/ID	监控	可用区	状态	规格/镜像	IP地址
ecs-f4a6-0003 2ce75551-b298-4f90-971d-f6af20e...		可用区2	关机	2vCPUs 4GB kc1.large.2 CentOS 7.6 64bit with ARM	119.3.212.133 (弹性公... 192.168.0.40 (私有))
ecs-f4a6-0002 ef3ff89a-8117-4cfa-b13c-dfc9a5ac...		可用区2	关机	2vCPUs 4GB kc1.large.2 CentOS 7.6 64bit with ARM	114.116.251.152 (弹性... 192.168.0.199 (私有))
ecs-f4a6-0001 39f27fc0-6dd0-403f-994f-1cf3f59d...		可用区2	关机	2vCPUs 4GB kc1.large.2 CentOS 7.6 64bit with ARM	124.70.111.172 (弹性... 192.168.0.9 (私有))

节点	公网 Ipv4 地址	内网 Ipv4 地址
master	119.3.212.133	192.168.0.40
slave01	114.116.251.152	192.168.0.199
slave02	124.70.111.172	192.168.0.9

按照实验指导书部署 Hadoop 集群并启动后，在三个节点上执行 Jps 分别查看 Java 进程状态，可以得到如图 29~31 的截图：

```

[root@master ~]# jps
2402 NameNode
2739 ResourceManager
1315 WrapperSimpleApp
2998 Jps
2588 SecondaryNameNode
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.40 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fea3:d3d0 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:a3:d3:d0 txqueuelen 1000 (Ethernet)
    RX packets 510426 bytes 545585109 (520.3 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 247303 bytes 695208402 (663.0 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 80 bytes 16732 (16.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 80 bytes 16732 (16.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

图 29 master 节点执行 jps 效果

```

[root@slave01 ~]# jps
1314 WrapperSimpleApp
2212 DataNode
2316 NodeManager
2430 Jps
[root@slave01 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.199 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:feb5:9686 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:b5:96:86 txqueuelen 1000 (Ethernet)
    RX packets 669647 bytes 889452011 (848.2 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 184845 bytes 18270653 (17.4 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 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

图 30 slave01 节点执行 jps 效果

```

[root@slave02 ~]# jps
1426 WrapperSimpleApp
2197 DataNode
2301 NodeManager
2415 Jps
[root@slave02 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.9 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe74:77f7 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:74:77:f7 txqueuelen 1000 (Ethernet)
    RX packets 524150 bytes 678857056 (647.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 159403 bytes 16389021 (15.6 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 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

图 31 slave02 节点执行 jps 效果

可以看到，Hadoop 集群部署完成并启动成功

本地编写 scala 程序并导出 jar 包，在主节点使用 spark-submit 命令运行后可以得到如下的截图 73：

```

[root@master ~]# hadoop fs -cat /spark_test/part-00000
21/04/17 02:38:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(hi,6)
(hello,5)
[root@master ~]# hadoop fs -cat /spark_test/part-00001
21/04/17 02:38:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(spark,2)
[root@master ~]# hadoop fs -cat /spark_test/part-00002
21/04/17 02:38:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(sparksql,1)
(sparkstreaming,1)
(sparkorahx,1)
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.40 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fea3:d3d0 prefixlen 64 scopeid 0x20<link>
    ether fa:16:3e:a3:d3:d0 txqueuelen 1000 (Ethernet)
    RX packets 1246456 bytes 1252830400 (1.1 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 639106 bytes 3264807212 (3.0 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 3382 bytes 4654398 (4.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3382 bytes 4654398 (4.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

```
[root@master ~]# hadoop fs -cat /spark_test/part-00000
21/04/17 02:40:27 WARN util.NativeCodeLoader: Unable to load native-hadoop
(hi,6)
(hello,5)
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.40 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fea3:d3d0 prefixlen 64 scopeid 0x20<link-local>
    ether fa:16:3e:a3:d3:d0 txqueuelen 1000 (Ethernet)
    RX packets 1246997 bytes 1252923693 (1.1 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 639400 bytes 3264839253 (3.0 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 3412 bytes 4657752 (4.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3412 bytes 4657752 (4.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

图 73 spark 统计词频结果

选做，开放华为云所有安全组，访问 <http://119.3.212.133:18088>
可以看到任务完成情况：

←

→

+

+

←

→

↺

↻

⚠

不安全

119.3.212.138

1808h@cluster

Del

⌵

🔍

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

🔖

图选做 查看任务完成情况

二、简要描述实验做了哪些工作？

本次实验分为云服务器端和本地端，

云服务器端首先在华为云上购买三台云主机，对三台服务器分别配置了 java 环境，配置了三节点互信，部署了 Hadoop 集群并启动成功。之后通过 On Yarn 模式搭建了 spark 环境。

本地端在本地 windows 安装 jdk1.8 和 scala 2.13.5 后，创建一个 maven 项目，并修改 pom.xml 文件，编写 scala 程序并导出完整 jar 包。

之后在云服务器上使用搭建好的环境执行命令 spark-submit, 该 spark 程序完成了单词计数任务，以空格分词计算每个单词的出现次数。

三.实验过程中遇到的问题和解决办法？

在本地实验中，我遇到了如下几个问题，并最终解决了这些问题

1.节点互信配置问题

问题描述：配置节点互信时，通过 ssh 检验始终不成功，如下图所示，在 master 节点使用 ssh slave01 命令登录 slave01 节点时，并未直接登录成功，而是需要输入了密码，同理在登录 slave02 节点时也出现了这个问题

```
[root@master ~]# ssh slave01
root@slave01's password:
Last login: Sat Apr 17 19:22:47 2021 from 192.168.0.40

Welcome to Huawei Cloud Service

[root@slave01 ~]# ssh slave02
Last login: Sat Apr 17 19:21:44 2021 from 59.64.129.252

Welcome to Huawei Cloud Service

[root@slave02 ~]# ssh master
Last login: Sat Apr 17 19:22:54 2021 from 192.168.0.199

Welcome to Huawei Cloud Service

[root@master ~]# ssh slave02
root@slave02's password:
Last login: Sat Apr 17 19:23:55 2021 from 192.168.0.199

Welcome to Huawei Cloud Service

[root@slave02 ~]# ssh slave01
Last login: Sat Apr 17 19:23:51 2021 from 192.168.0.40

Welcome to Huawei Cloud Service
```

图-1 互信检测问题

错误分析：我测试了所有节点之间互相执行 ssh 后的结果，发现只有在 master 节点上执行 ssh slave01 和 ssh slave02 命令登录需要输入密码，因此将问题定位到文件/root/.ssh/authorized_keys 上，同时由于是只有 master 节点登录其他节点需要输入密码，因此是 master 节点的秘钥出错，因此我使用 vim 命令查看了该文件，细致检查后发现在复制 master 节点上的 id_rsa.pub 文件内容时少复制了一个字符 's'，如下图：

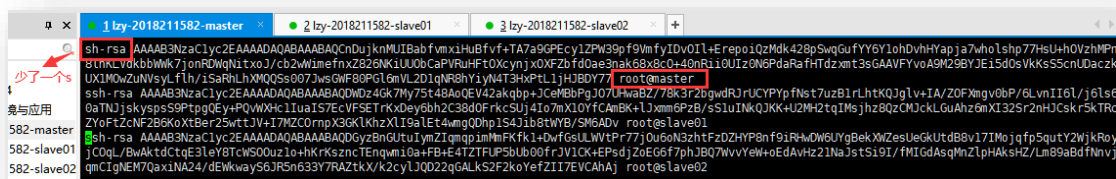


图-2 缺少字符's'

将 s 添加上并发给另外两个节点后，我解决了这个问题也成功启动了 Hadoop 集群

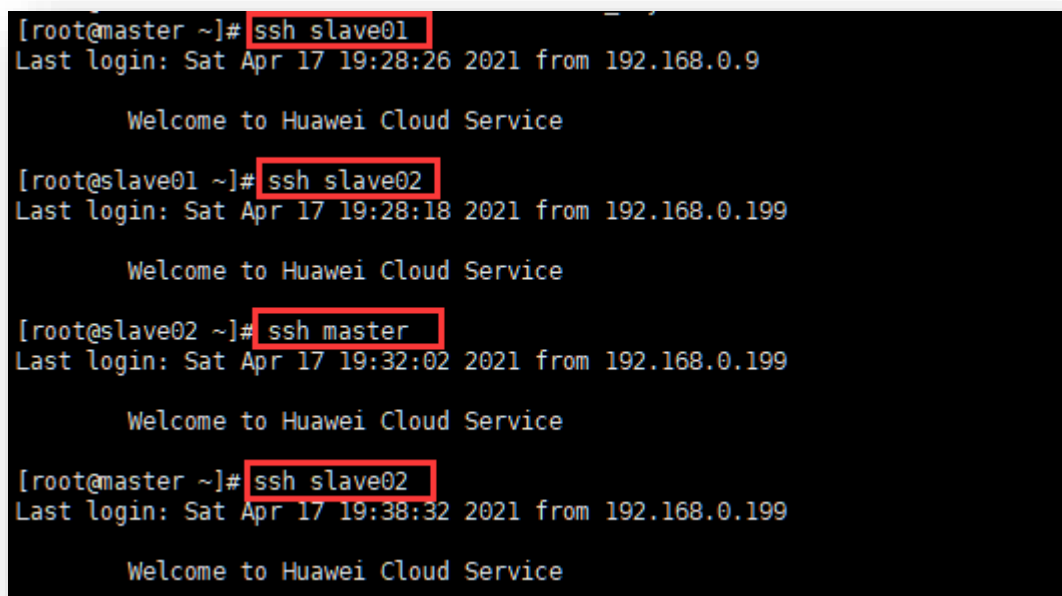


图-3 互信配置成功

问题思考：这个问题暴露出我在做实验时对于每一步操作的检查不够细致，问题产生的原因为我在 Linux 命令行中复制整个文件后并没有仔细检查粘贴时粘贴的内容是否完整，因此犯了这样的小错误，这也警示自己做实验时一定要细致检查每一步操作，因为这些步骤都是环环相扣的

2.scala 程序编写问题

问题描述: 本地编写完 scala 程序并打包成 jar 包在服务器上运行时, 发现运行结果不正确, 运行结果与实验指导相差很大, 程序并没有完成计算单词出现次数功能, 或者说程序分词是按照字母进行了分词, 并没有按照单词进行计算, 运行截图如下:

```
[root@master ~]# hadoop fs -cat /spark_test/part-00000
21/04/17 02:29:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(h,12)
(.,12)
(.,11)
(l,11)
[root@master ~]# hadoop fs -cat /spark_test/part-00001
(s,7)
(a,7)
(i,7)
(r,7)
(p,6)
(e,6)
(o,5)
(k,5)
[root@master ~]# hadoop fs -cat /spark_test/part-00002
(q,2)
(t,1)
(n,1)
(q,1)
(x,1)
(m,1)
```

图-4 程序执行结果不正确

错误分析: 出现这个问题, 我首先想到的是 scala 代码出现了问题, 检查后果然, 有一个低级错误, 我把所有的 `val` 都写出了 `var`,

```
object ScalaWordCount{
  def main(args: Array[String]): Unit={
    var list = List("hello hi hi spark",
      "hello spark hello hi sparksql",
      "hello hi hi sparkstreaming",
      "hello hi sparkgraphx")

    var sparkConf = new SparkConf().setAppName("word-count").setMaster("yarn")
    var sc = new SparkContext(sparkConf)
    var lines:RDD[String] = sc.parallelize(list)
    var words:RDD[String] = lines.flatMap((line:String)=>{line.split( regex = " ")})
    var wordAndOne:RDD[(String,Int)] = words.map((word:String)=>{(word,1)})
    var wordAndNum:RDD[(String,Int)] = wordAndOne.reduceByKey((count1:Int,count2:Int)=>{count1+count2})
    var ret = wordAndNum.sortBy(kv=>kv._2, ascending = false)
    print(ret.collect().mkString(","))
    ret.saveAsTextFile( path = "hdfs://master:9000/spark_test")
    sc.stop()
  }
}
```

图-5 原始 scala 代码

将所有的 var 修改成 val 后，我再次执行程序，发现依旧是同样的问题，问题并没有解决。因此我细致的思考了问题出现的原因，并仔细阅读了整个 scala 代码。由于问题出现是因为分词没有成功，使得统计单词出现次数时是按照字符进行统计，而并没有按照单词统计。因此我重点检查了代码中的

```
var words:RDD[String] = lines.flatMap((line:String)=>{line.split("")})
```

由此，我发现了问题所在，在 line 执行 split 分词函数时，他的分词依据是“ ” 而不是 “ ”，我将空字符修改成了空格字符后，spark 程序输出了我想要的结果，并解决了这个问题

```
object ScalaWordCount{
  修改一：将var修改成val
  def main(args: Array[String]): Unit={
    val list = List("hello hi hi spark",
      "hello spark hello hi sparksql",
      "hello hi hi sparkstreaming",
      "hello hi sparkgraphx")

    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(" ")})  修改二：将""修改成" "
    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://master:9000/spark_test")
    sc.stop()
  }
}
```

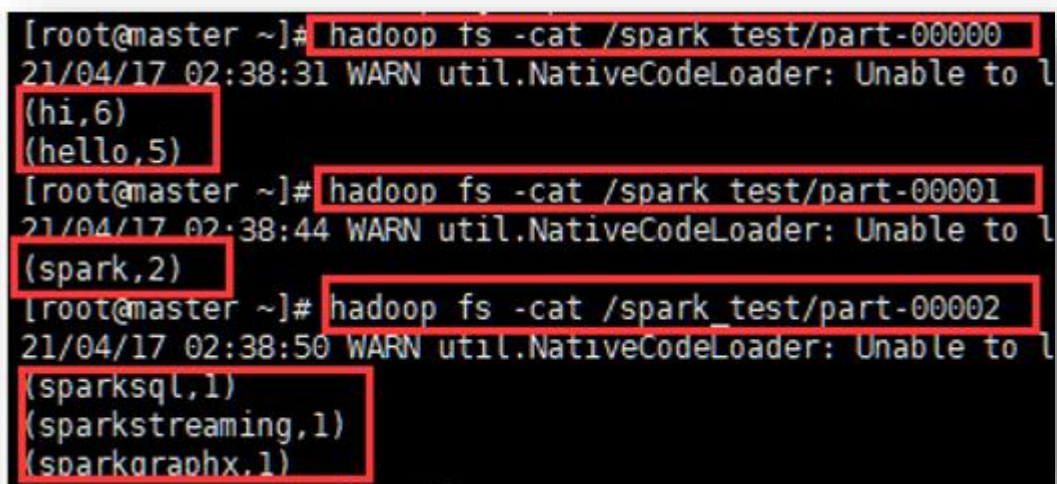
图-6 修改后 scala 代码

问题思考：产生这个问题的一个原因就是，首次编写 scala 程序时，对于语法不熟悉，由于最近一直在使用 JavaScript 编写相关程序，因此错误的将 scala 语法中的 val 写出了 JavaScript 中常用的

var, 虽然很快发现了这个问题, 但依旧没有找到本质错误。在细致的思考整个问题出现的原因和原理后, 我定位到了最终的问题并解决了它。这个问题让我意识到, 编写程序代码时不能想当然, 而要结合程序实际的要求, 编写完成后要 review 并调试一下代码, 看是否是按照解决问题的逻辑和正确性编写的代码, 要清楚整个代码在干什么! 而不是只是手动敲了敲没有语法错误就结束了, 这个问题着实给我敲响了警钟。

3. 执行结果问题

问题描述: 执行 spark-submit 后得到的结果中 part-00000 里没有 “(spark, 2)” 而是在 part-00001 中, 如图:



```
[root@master ~]# hadoop fs -cat /spark_test/part-00000
21/04/17 02:38:31 WARN util.NativeCodeLoader: Unable to load native code library: /lib64/libc.so.6
(hi,6)
(hello,5)
[root@master ~]# hadoop fs -cat /spark_test/part-00001
21/04/17 02:38:44 WARN util.NativeCodeLoader: Unable to load native code library: /lib64/libc.so.6
(spark,2)
[root@master ~]# hadoop fs -cat /spark_test/part-00002
21/04/17 02:38:50 WARN util.NativeCodeLoader: Unable to load native code library: /lib64/libc.so.6
(sparksql,1)
(sparkstreaming,1)
(sparkgraphx,1)
```

图-7 最终执行问题

问题解决: 在咨询了助教后, 助教学长热心的回答了问题, 原来是因为分布式处理造成的, 属于正常现象。至此我的整个实验完成, 感谢助教学长的热心解答和帮助! 谢谢!

附录

完整 scala 代码

```
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 sparkgraphx")

    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://master:9000/spark_test")

sc.stop()

}

}
```