分布式实时计算引擎 SPARK

一、实训说明

本次实训，主要是搭建分布式实时计算系统 spark。Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是——Job 中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。

二、实训环境

1）已经安装完成的 Hadoop 完全分布式环境
2）已经安装完成 zookeeper 分布式协调系统
3）使用软件：spark-2.3.3-bin-hadoop2.7.tgz
下载地址：
http://archive.apache.org/dist/spark/spark-2.3.3/spark-2.3.3-bin-hadoop2.7.tgz

三、实训内容

以下操作均在 hadoop 用户下进行

1. 安装 spark

1）解压安装包（主节点）

```
[hadoop@master ~]$ sudo tar -zxvf /home/hadoop/spark-2.3.3-bin-hadoop2.7.tgz -C
/usr
```

2）重命名安装路径（主节点）

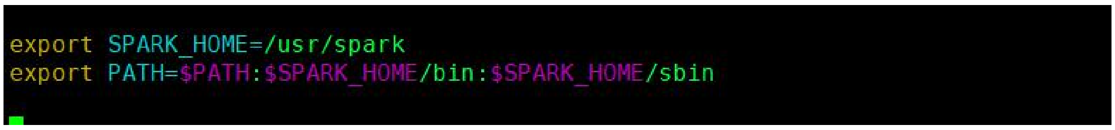
```
[hadoop@master ~]$ sudo mv /usr/spark-2.3.3-bin-hadoop2.7/ /usr/spark
```

3）配置 spark 的环境变量，并使环境变量生效（所有节点）

```
[hadoop@master ~]$ sudo vim /etc/profile
```

在环境变量中加入以下内容：

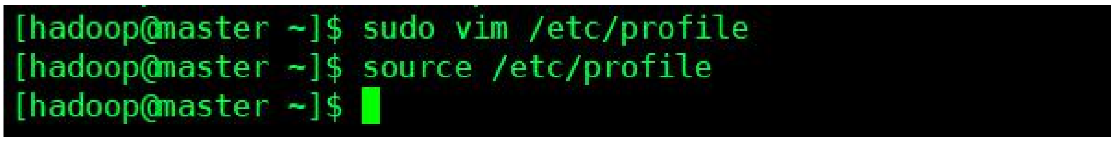
```
export SPARK_HOME=/usr/spark

export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```



```
export SPARK_HOME=/usr/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

4）使环境变量生效（所有节点）

```
[hadoop@master ~]$ source /etc/profile
```



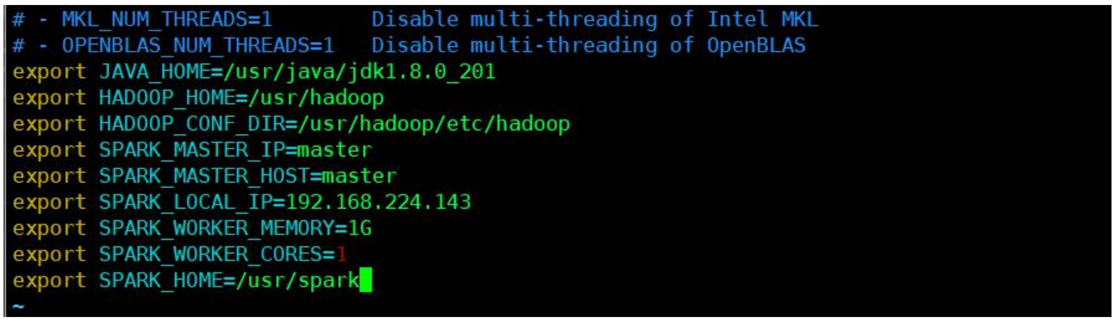
```
[hadoop@master ~]$ sudo vim /etc/profile
[hadoop@master ~]$ source /etc/profile
[hadoop@master ~]$
```

5）配置 spark-env.sh 配置文件（主节点）

```
[hadoop@master ~]$ sudo cp $SPARK_HOME/conf/spark-env.sh.template
$SPARK_HOME/conf/spark-env.sh
[hadoop@master ~]$ sudo vim $SPARK_HOME/conf/spark-env.sh
```

在配置文件中添加或修改以下内容，其中 SPARK_LOCAL_IP 的值为本机 IP

```
export JAVA_HOME=/usr/java/jdk1.8.0_201
export HADOOP_HOME=/usr/hadoop
export HADOOP_CONF_DIR=/usr/hadoop/etc/hadoop
export SPARK_MASTER_IP=master
export SPARK_MASTER_HOST=master
export SPARK_LOCAL_IP=192.168.224.143
export SPARK_WORKER_MEMORY=1G
export SPARK_WORKER_CORES=1
export SPARK_HOME=/usr/spark
```

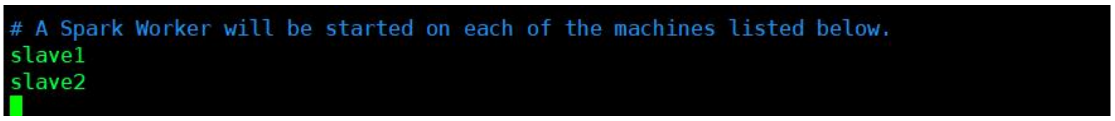


```
# - MKL_NUM_THREADS=1        Disable multi-threading of Intel MKL
# - OPENBLAS_NUM_THREADS=1   Disable multi-threading of OpenBLAS
export JAVA_HOME=/usr/java/jdk1.8.0_201
export HADOOP_HOME=/usr/hadoop
export HADOOP_CONF_DIR=/usr/hadoop/etc/hadoop
export SPARK_MASTER_IP=master
export SPARK_MASTER_HOST=master
export SPARK_LOCAL_IP=192.168.224.143
export SPARK_WORKER_MEMORY=1G
export SPARK_WORKER_CORES=1
export SPARK_HOME=/usr/spark
~
```

6）配置 slaves 配置文件（主节点）

```
[hadoop@master ~]$ sudo cp $SPARK_HOME/conf/slaves.template
$SPARK_HOME/conf/slaves
[hadoop@master ~]$ sudo vim $SPARK_HOME/conf/slaves
```

在配置文件中添加从节点的名字



```
# A Spark Worker will be started on each of the machines listed below.
slave1
slave2
```

7）更改 spark 启动文件名字，因为 spark 的启动文件和 hadoop 的启动文件同名

```
[hadoop@master ~k]$ sudo mv /usr/spark/sbin/start-all.sh
/usr/spark/sbin/start-spark-all.sh
```

8）将安装文件同步到 slave 节点（主节点）

```
[hadoop@master ~]$ sudo scp -r  /usr/spark/ slave1 :/usr

[hadoop@master ~]$ sudo scp -r  /usr/spark/ slave2:/usr
```

9）修改 slave 节点的 spark-env.sh 配置文件（从节点）
将 SPARK_LOCAL_IP 修改为本机 IP 地址（从节点）

```
[hadoop@slave2 root]$ sudo vim /usr/spark/conf/spark-env.sh
```

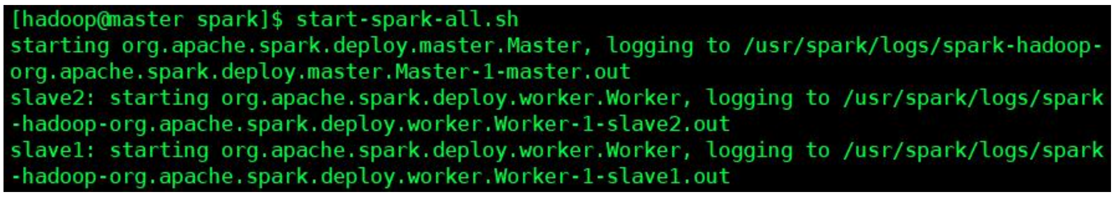
10）修改安装文件的属主权限（所有节点）

```
[hadoop@master ~]$ sudo chown -R hadoop:hadoop /usr/spark

[hadoop@slave1~]$ sudo chown -R hadoop:hadoop /usr/spark

[hadoop@slave2~]$ sudo chown -R hadoop:hadoop /usr/spark
```

2. 验证测试

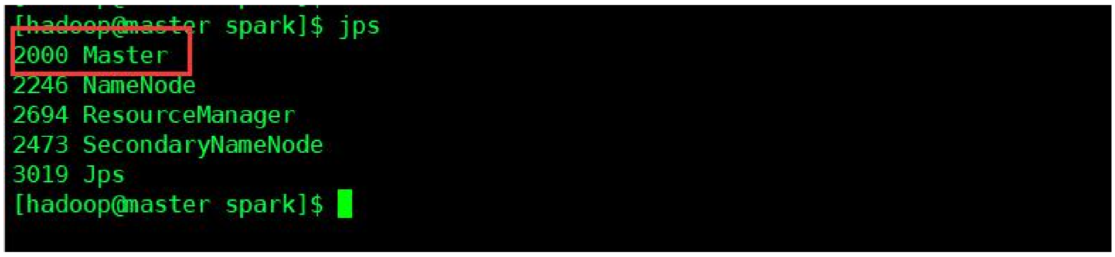
1）启动 spark

```
[hadoop@master spark]$ start-spark-all.sh
```



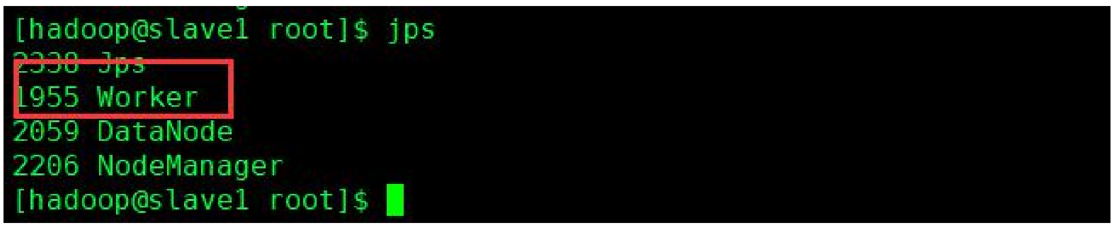
```
[hadoop@master spark]$ start-spark-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/spark/logs/spark-hadoop-
org.apache.spark.deploy.master.Master-1-master.out
slave2: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark/logs/spark
-hadoop-org.apache.spark.deploy.worker.Worker-1-slave2.out
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark/logs/spark
-hadoop-org.apache.spark.deploy.worker.Worker-1-slave1.out
```

2）查看 spark 的守护进程
master：



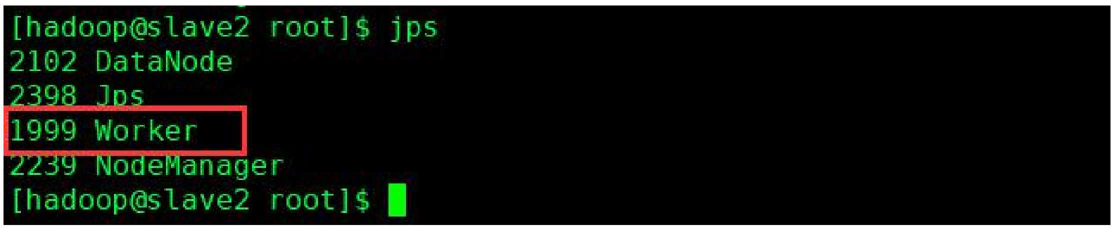
```
[hadoop@master spark]$ jps
2000 Master
2246 NameNode
2694 ResourceManager
2473 SecondaryNameNode
3019 Jps
[hadoop@master spark]$
```

slave1：



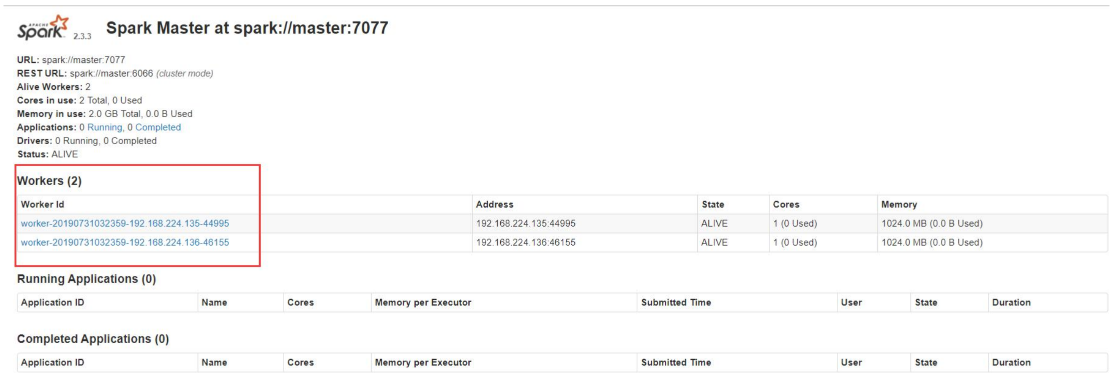
```
[hadoop@slave1 root]$ jps
2338 Jps
1955 Worker
2059 DataNode
2206 NodeManager
[hadoop@slave1 root]$
```

slave2：



```
[hadoop@slave2 root]$ jps
2102 DataNode
2398 Jps
1999 Worker
2239 NodeManager
[hadoop@slave2 root]$
```

3）在浏览器打开 spark 的 web 界面。192.168.224.134：8080



Spark Master at spark://master:7077

URL: spark://master:7077
REST URL: spark://master:6066 *(cluster mode)*
Alive Workers: 2
Cores in use: 2 Total, 0 Used
Memory in use: 2.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20190731032359-192.168.224.135-44995	192.168.224.135:44995	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20190731032359-192.168.224.136-46155	192.168.224.136:46155	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

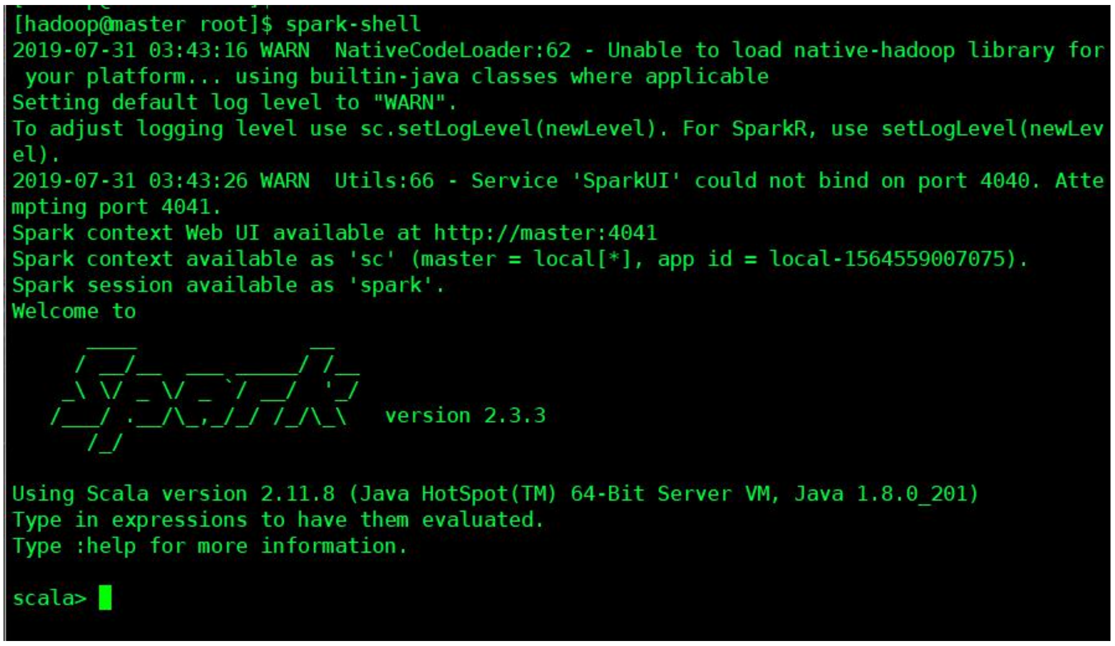
Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

3. 实例应用

1）打开 spark shell

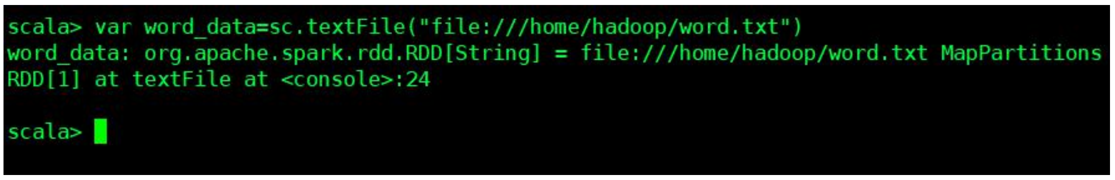
```
[hadoop@master spark]$ spark-shell
```



```
[hadoop@master root]$ spark-shell
2019-07-31 03:43:16 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for
 your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLev
el).
2019-07-31 03:43:26 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4040. Atte
mpting port 4041.
Spark context Web UI available at http://master:4041
Spark context available as 'sc' (master = local[*], app id = local-1564559007075).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.3
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_201)
Type in expressions to have them evaluated.
Type :help for more information.

scala> 
```

2）加载本地文件

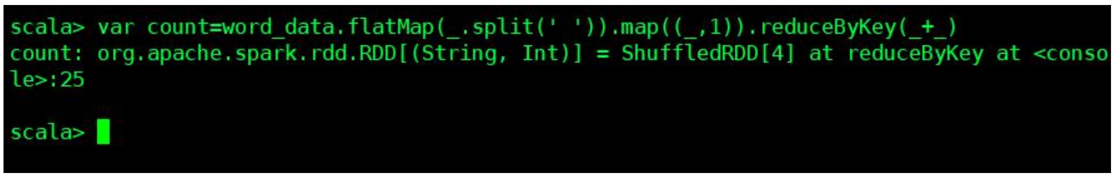
```
scala> var word_data=sc.textFile("file:///home/hadoop/word.txt")
```



```
scala> var word_data=sc.textFile("file:///home/hadoop/word.txt")
word_data: org.apache.spark.rdd.RDD[String] = file:///home/hadoop/word.txt MapPartitions
RDD[1] at textFile at <console>:24

scala> 
```

3）计算文本中的单词数量

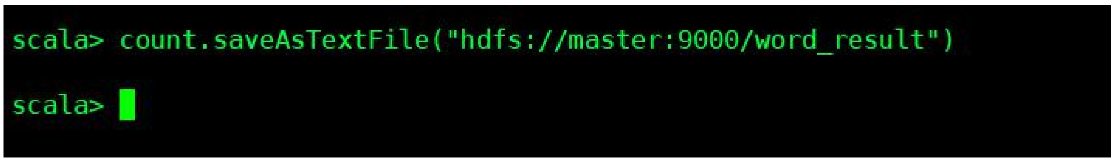
```
scala> var count=word_data.flatMap(_.split(' ')).map((_,1)).reduceByKey(_+_)
```



```
scala> var count=word_data.flatMap(_.split(' ')).map((_,1)).reduceByKey(_+_)
count: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <conso
le>:25

scala> 
```

4）将结果保存到 hdfs

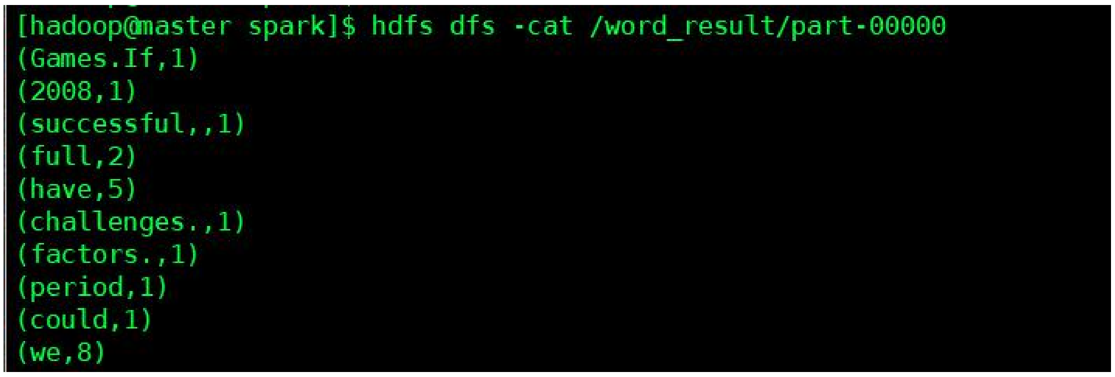
```
scala> count.saveAsTextFile("hdfs://master:9000/word_result")
```



```
scala> count.saveAsTextFile("hdfs://master:9000/word_result")

scala> 
```

5）查看结果

```
[hadoop@master spark]$ hdfs dfs -cat /word_result/part-00000
```



```
[hadoop@master spark]$ hdfs dfs -cat /word_result/part-00000
(Games.If,1)
(2008,1)
(successful,,1)
(full,2)
(have,5)
(challenges.,1)
(factors.,1)
(period,1)
(could,1)
(we,8)
```