



河南理工大学  
Henan Polytechnic University

## 教学上机实验报告

课程名称： 大数据分析技术

任课教师姓名： 吴正江

学生学号： 311809000608

学生姓名： 王荣胜

学生专业班级： 计实验 1801 班

2020~ 2021 学年 第 2 学期

# 河南理工大学

## 教学上机实验报告评价分值标准

| 序号  | 评价指标   | 分值 | 评价等级及参考分值 |    |    |    |    | 评价分 |
|-----|--|----|-----------|----|----|----|----|-----|
|     |  |    | 优         | 良  | 中  | 合格 | 差  |     |
| 1   | 实验报告内容完整充实   | 10 | 10        | 8  | 7  | 6  | 3  |     |
| 2   | 实验内容书写规范、字迹工整认真  | 10 | 10        | 8  | 7  | 6  | 3  |     |
| 3   | 实验过程叙述详细、概念正确，语言表达准确，结构严谨，条理清楚，逻辑性强，自己努力完成，没有抄袭。                     | 30 | 30        | 26 | 23 | 20 | 10 |     |
| 4   | 对实验过程中存在的问题分析详细透彻、深刻、全面、规范、，结合实验内容，有自己的个人见解和想法，并能结合该实验提出相关问题，给出解决方法。 | 30 | 30        | 26 | 23 | 20 | 10 |     |
| 5   | 实验结果、分析和结论正确无误   | 20 | 20        | 17 | 15 | 13 | 6  |     |
| 总得分 |  |    |           |    |    |    |    |     |

签名（签章）：

日期： 年 月 日

# 河南理工大学教学上机实验报告

上机时间 2021 年 3 月 18 日

## 实验题目：

Linux 安装与基本操作

## 实验目的和要求：

目的：

1. 安装 vmware。
2. 在 vmware 安装三台 CentOS7 的主机。
3. 练习使用 Linux 命令。

要求：

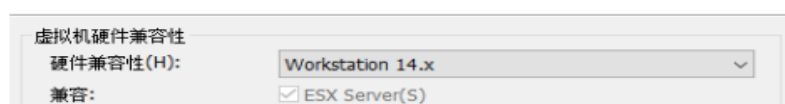
1. 使用“计算节点”模式。
2. 地址要求  
192.168.128.100 hadoop0  
192.168.128.101 hadoop1  
192.168.128.102 hadoop2
3. 设定 root 用户密码为 123。
4. 新建 hadoop 用户，密码 123。

## 实验过程：

### 一、虚拟机安装

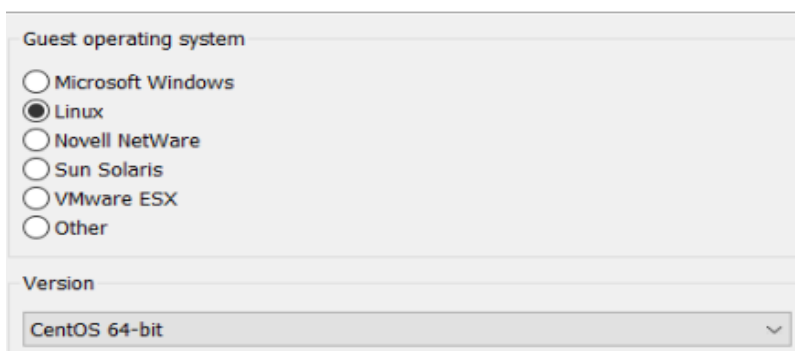
1. 下载 Vmware14.0 虚拟机软件，并按步骤安装；
2. 进入主页选择**创建虚拟机**；
3. 选择**自定义安装**，下一步即可；
4. 选择版本 14.x（也可以根据需求选择）：

**选择虚拟机硬件兼容性**  
该虚拟机需要何种硬件功能？



虚拟机硬件兼容性  
硬件兼容性(H): Workstation 14.x  
兼容: ☒ ESX Server(S)

5. 选择**稍后安装操作系统**；
6. 选择安装的系统（Linux、CentOS 64）：



Guest operating system  
☐ Microsoft Windows  
☒ Linux  
☐ Novell NetWare  
☐ Sun Solaris  
☐ VMware ESX  
☐ Other  
Version  
CentOS 64-bit

7. 选择核心大小和内存大小（主节点可以两个核心，两个从节点各一个核心，如果电脑配置好可以选择更好的搭配；内存根据电脑配置）：

### 处理器配置

为此虚拟机指定处理器数量。

#### 处理器

处理器数量(P): 2

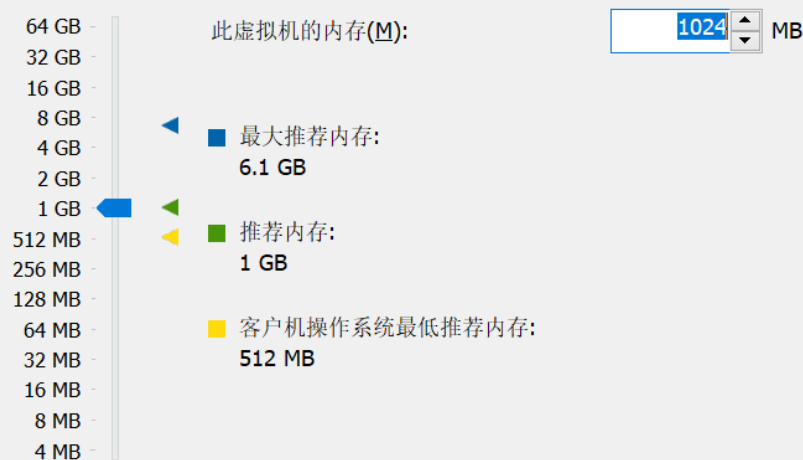
每个处理器的内核数量(C): 2

处理器内核总数: 4

### 此虚拟机的内存

您要为此虚拟机使用多少内存？

指定分配给此虚拟机的内存量。内存大小必须为 4 MB 的倍数。



建议的虚拟机配置搭配：

## Hardware

- (X2) 2CPU+4G RAM+20G HardDisk
- (X1) 4CPU+ 8G RAM+20G HardDisk
- **至少一台虚拟机选择2个核心。**

8. 一直点击下一步直到指定磁盘文件(此处最好选择指定在除 C 盘外的其它盘)；

新建虚拟机向导

×

### 指定磁盘文件

您要在何处存储磁盘文件？

#### 磁盘文件(E)

将使用多个磁盘文件创建一个 20 GB 虚拟磁盘。将根据此文件名自动命名这些磁盘文件。

node3.vmdk

浏览(B)...

9. 继续创建，在**自定义硬件**中删除不使用的（如显示器、USB 等），可以加速创建安装：

**已准备好创建虚拟机**

单击“完成”创建虚拟机，然后开始安装 **CentOS 7 64 位**。

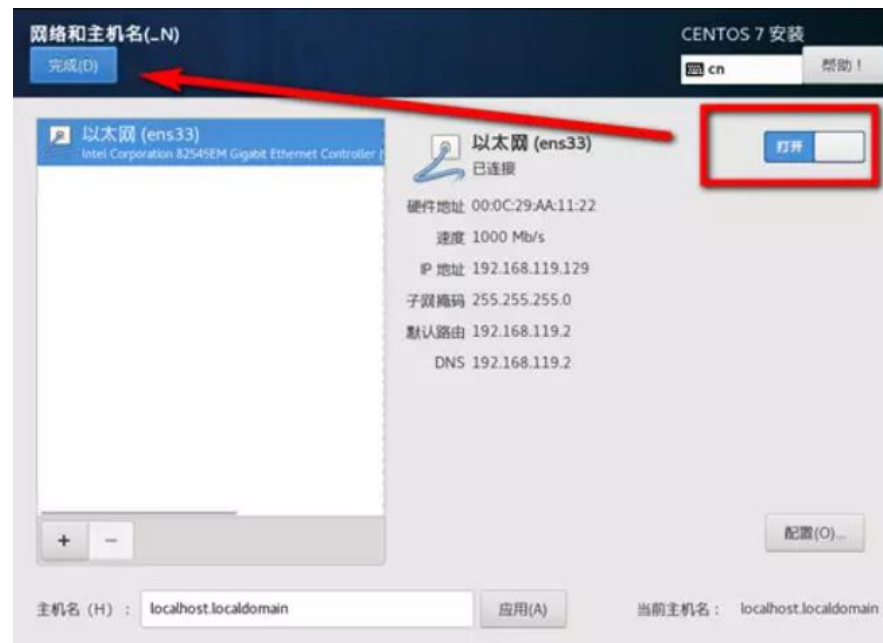
将使用下列设置创建虚拟机：

|        |   |
|--------|---|
| 名称：    | node3   |
| 位置：    | C:\Users\17749\Documents\Virtual Machines\node3 |
| 版本：    | Workstation 15.x                                |
| 操作系统：  | CentOS 7 64 位                                   |
| 硬盘：    | 20 GB, 拆分                                       |
| 内存：    | 1024 MB   |
| 网络适配器： | NAT   |
| 其他设备：  | 4 个 CPU 内核, CD/DVD, USB 控制器, 打印机, 声卡            |

自定义硬件(C)...

## 二、系统安装（以主节点安装为例）

1. 选择系统镜像文件开始系统安装；
2. 语言：中文简体；
3. 软件选择：计算节点、Linux 远程连接；
4. 点击安装位置，进去点完成即可；
5. 网络与主机名（主机名：hadoop0，网络 ip 设置为手动，并且设置静态 ip 为 192.168.128.100）：





6. 点击继续，设置密码为：123，用户名称为 hadoop0，等待完成安装；
7. 重启进入；
8. 从节点 1 和 2 重复以上步骤建立三台虚拟机（主机名分别为：hadoop0、hadoop1、hadoop2；ip 为：192.168.128.100、192.168.128.101、192.168.128.102）；

### 三、建立新用户并练习使用 Linux 命令

1. 以 root 用户登录三个虚拟机，在三个虚拟机上进行相同一下操作；
2. 新建用户：

```
1. # 新建 hadoop 用户
2. useradd hadoop
3. # 为 hadoop 用户设置密码
4. passwd hadoop
5. # 切换至 hadoop 用户
6. su - hadoop
```

3. 关闭防火墙：

```
1. firewall-cmd --state 查看一下防火墙状态(关闭后显示 notrunning, 开启后显示 running)
2. systemctl stop firewalld.service #停止 firewall
3. systemctl disable firewalld.service #禁止 firewall 开机启动
```

4. 设置 IP 与机器名称的对应表：

```
1. vi etc/hosts
2.
3. # 添加这三行
4. 192.168.159.100 hadoop0
5. 192.168.159.101 hadoop1
6. 192.168.159.102 hadoop2
```

5. 重启虚拟机;
6. 练习使用常用的 Linux 命令 (ls、cd、cat、mv...);

**实验结果:**

成功安装了三台虚拟机: hadoop0、hadoop1 和 hadoop2 并且进行了静态 ip 的设置, 三个节点之间可以用 ping 命令进行通信连通测试发现是正常的。学习使用了 linux 命令, linux 下命令的使用简洁快速, 很适合做开发使用。

**实验分析:**

本次实验进行了虚拟机的安装, 在虚拟机的安装中我们必须明白目的是什么, 比如我们要进行 hadoop 的开发, 那么计算节点就是一个不错的选择, 其次在集群开发中, 手动设置 ip 是非常必要的步骤, 除此之外, 关闭防火墙, 网络连通测试也是我们安装好虚拟机环境后非常有必要做的步骤。

**实验成绩:**

日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

# 河南理工大学教学上机实验报告

上机时间 2021 年 3 月 25 日

## 实验题目：

Hadoop 安装部署

## 实验目的和要求：

目的：

1. 在 Hadoop0 上部署 standalone 模式的 hadoop2.9。
2. 在 Hadoop0 上部署 pseudo-distributed mode 的 hadoop2.9。
3. 在 Hadoop0, hadoop1, hadoop2 上部署 fully-distributed mode 的 hadoop2.9。

要求：

1. hadoop0 为 master；hadoop0, hadoop1, hadoop2 为 slaves。
2. hdfs 系统部署在 \$HADOOP\_HOME/hdfs/name 、data 下。

## 实验过程：

一、安装 JDK 环境和 Hadoop 环境

1. 利用 SecureCRT 以 hadoop 用户身份上传 Java 和 hadoop 的安装包到主节点（hadoop0）；

2. 解压：

```
1. # 解压 java 安装包
2. tar -zxvf jdk-8u251-linux-x64.tar.gz
3. # 解压 hadoop 安装包
4. tar -zxvf hadoop-2.9.2.tar.gz
```

3. 配置环境变量：

```
1. vi /home/hadoop/.bash_profile
2.
3. # 添加以下内容
4. # hadoop 变量
5. HADOOP_HOME=/home/hadoop/hadoop-2.9.2
6. PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
7. # java 变量
8. JAVA_HOME=/home/hadoop/jdk1.8.0_251
9. PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin
10.
11. export JAVA_HOME
12. export HADOOP_HOME
13. export PATH
```

4. 刷新环境变量：



```
1. source /home/hadoop/.bash_profile
```

5. 测试:

```
1. # java 测试
2. java -version
3. # hadoop 测试
4. hadoop version
```

```
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.251-b08, mixed mode)
```

```
Hadoop 2.7.7
Subversion Unknown -r c1aad84bd27cd79c3d1a7dd58202a8c3ee1ed3ac
Compiled by stevel on 2018-07-18T22:47Z
Compiled with protoc 2.5.0
From source with checksum 792e15d20b12c74bd6f19a1fb886490
This command was run using /home/hadoop/hadoop-2.7.7/share/hadoop/common/hadoop-common-2.7.7.jar
```

输出版本信息即为配置成功!

## 二、Hadoop 配置

### 1. 进入配置目录

```
1. # 进入 hadoop 的安装目录
2. cd /home/hadoop/hadoop-2.9.2/etc/hadoop
```

### 2. 配置 hadoop-env.sh:

```
1. vi hadoop-env.sh
2. # 配置 java_home
3. export JAVA_HOME=/home/hadoop/jdk1.8.0_251
```

### 3. 配置 core-site.xml:

```
1. vi core-site.xml
2. # 添加以下内容
3. <configuration>
4.     <!-- Hadoop 临时文件存放路径, 默认在/tmp 目录下 -->
5.     <property>
6.         <name>hadoop.tmp.dir</name>
7.         <value>/home/hadoop/hadoop-2.9.2/data</value>
8.     </property>
9.     <!-- NameNode 结点的URI(包括协议、主机名称、端口号) -->
10.    <property>
11.        <name>fs.defaultFS</name>
12.        <value>hdfs://hadoop0:9000</value>
```

```
13.     </property>
14.     <!-- 设置文件删除后，被完全清空的时间，默认为0，单位
        为分钟 -->
15.     <property>
16.         <name>fs.trash.interval</name>
17.         <value>60</value>
18.     </property>
19. </configuration>
```

#### 4. 配置 hdfs-site.xml:

```
1. vi hdfs-site.xml
2. # 添加以下内容
3. <configuration>
4.     <!-- 块存储份数，默认为3 -->
5.     <property>
6.         <name>dfs.replication</name>
7.         <value>3</value>
8.     </property>
9.     <!-- 关闭权限校验，开发学习时可开启，默认为true -->
10.    <property>
11.        <name>dfs.permissions</name>
12.        <value>>false</value>
13.    </property>
14.    <!-- namenode 的http 访问地址和端口 -->
15.    <property>
16.        <name>dfs.namenode.http-address</name>
17.        <value>hadoop0:50070</value>
18.    </property>
19. </configuration>
```

#### 5. 配置 mapred-site.xml (重命名 mapred-site.xml.template):

```
1. vi mapred-site.xml
2. # 添加以下内容
3. <configuration>
4.     <!-- 设置运行MapReduce 任务方式为yarn，默认为local，可选
        项还有classic -->
5.     <property>
6.         <name>mapreduce.framework.name</name>
7.         <value>yarn</value>
8.     </property>
9. </configuration>
```

#### 6. 配置 yarn-site.xml:

```
1. vi yarn-site.xml
2. # 添加以下内容
3. <configuration>
4.     <!-- resourcemanager 的主机名 -->
5.     <property>
6.         <name>yarn.resourcemanager.hostname</name>
7.         <value>hadoop0</value>
8.     </property>
9.     <!-- 暴露给客户端的主机名及端口号 -->
10.    <property>
11.        <name>yarn.resourcemanager.address</name>
12.        <value>hadoop0:8032</value>
13.    </property>
14.    <!-- 分配给容器的物理内存量, 单位是MB, 设置为-1 则自动分配, 默认 8192MB -->
15.    <property>
16.        <name>yarn.nodemanager.resource.memory-mb</name>
17.        <value>1536</value>
18.    </property>
19.    <!-- NodeManager 上运行的服务列表, 可以配置成 mapreduce_shuffle, 多个服务使用逗号隔开 -->
20.    <property>
21.        <name>yarn.nodemanager.aux-services</name>
22.        <value>mapreduce_shuffle</value>
23.    </property>
24. </configuration>
```

#### 7. 配置 slaves:

```
1. vi slaves
2. # 添加以下内容
3. hadoop0
4. hadoop1
5. hadoop2
```

### 三、配置免密登录

#### 1. 生成密钥:

```
1. # 整个过程一直回车即可
2. ssh-keygen -t rsa
```

#### 2. 密钥分发:

1. # @前为用户名, @后为主机名, 在主节点上执行以下命令
2. `ssh-copy-id hadoop@hadoop0`
3. `ssh-copy-id hadoop@hadoop1`
4. `ssh-copy-id hadoop@hadoop2`
5. # 输入一次 hadoop 用户的密码即可通过验证

3. 分发 java 和 hadoop 环境到其它两个节点:

1. # 拷贝 JDK
2. `scp -r /home/hadoop/jdk1.8.0_251 hadoop@hadoop1:/home/hadoop`
3. `scp -r /home/hadoop/jdk1.8.0_251 hadoop@hadoop2:/home/hadoop`
4. # 拷贝 Hadoop
5. `scp -r /home/hadoop/hadoop-2.9.2 hadoop@hadoop1:/home/hadoop`
6. `scp -r /home/hadoop/hadoop-2.9.2 hadoop@hadoop2:/home/hadoop`
7. # 拷贝环境变量配置文件, 同名文件会自动覆盖
8. `scp -r /home/hadoop/.bash_profile hadoop@hadoop1:/home/hadoop`
9. `scp -r /home/hadoop/.bash_profile hadoop@hadoop2:/home/hadoop`

#### 四、集群启动与测试

1. 初始化:

1. `hdfs namenode -format`

2. 集群启动:

1. `start-all.sh`

3. jps 验证, 输出 6 个启动服务即为正确:

```
78691 DataNode
44402 Jps
78406 NameNode
80265 SecondaryNameNode
82461 ResourceManager
82620 NodeManager
```

实验结果：

搭建完成了 hadoop 环境和 java 环境，并且保证了全分布式搭建完成后可以正确的运行，在服务中，主节点 6 个服务开启，两个从节点都是 3 个服务开启。



Overview 'namenode1:9000' (active)

|                |  |
|----------------|--|
| Started:       | Tue Nov 19 16:33:48 CST 2019                     |
| Version:       | 2.6.5, re8c9fe0b4c252caf2ebf1464220599650f119997 |
| Compiled:      | 2016-10-02T23:43Z by sjlee from branch-2.6.5     |
| Cluster ID:    | CID-ef6a7b40-e878-4119-81b9-2e4ead16ce6f         |
| Block Pool ID: | BP-955946782-192.168.1.11-1574151877982          |

实验分析：

Hadoop 和 java 的安装其实就是一个环境变量的配置过程，同时必须要保证环境变量的配置一定是要生效的。在从节点的配置中，我们可以不必重复主节点的配置过程，采用 ssh 绵密登录和 scp 命令可以快速的传送配置文件和安装文件到从节点从而快速完成全分布式集群的搭建。这里要注意的就是全分布式的搭建中一定要保证各节点之间是防火墙关闭的，不然会出现不能访问 8088 和 50070 网页的情况。

实验成绩：

日期：\_\_\_\_\_年\_\_\_\_月\_\_\_\_日

# 河南理工大学教学上机实验报告

上机时间 2021 年 4 月 1 日

## 实验题目：

Spark 安装部署

## 实验目的和要求：

目的：

1. 在 Hadoop0 上部署 local 模式的 spark。
2. 在 Hadoop0\hadoop1\hadoop2 上部署 yarn 模式的 spark。
3. 在 Hadoop0\hadoop1\hadoop2 上部署 standalone 模式的 spark。
4. 安装 anaconda3

要求：

1. 在 Hadoop0 上开启 notebook 服务，使用宿主机完成对该服务的访问。
2. 在 Yarn 模式下，执行 sc.master，观察 hadoop0:8088 行为。
3. 在 standalone 模式下，执行 sc.master，观察 hadoop0:8080 行为。

## 实验过程：

一、安装配置 scale 和 spark

1. 利用 SecureCRT 以 hadoop 用户登录并上传以下三个文件；

spark-3.0.1-bin-hadoop2.7.tgz

Anaconda3-2020.11-Linux-x86\_64.sh

scala-SDK-4.7.0-vfinal-2.12-  
linux.gtk.x86\_64.tar.gz

2. 解压 scale 和 spark：

1. `tar -zxvf scala-SDK-4.7.0-vfinal-2.12-linux.gtk.x86_64.tar.gz`
2. `tar -zxvf spark-3.0.1-bin-hadoop2.7.tgz`

3. 配置环境变量：

1. `vi /home/hadoop/.bash_profile`
- 2.
3. `#SCALA`
4. `export SCALA_HOME=/home/hadoop/scala-2.12`
5. `export PATH=$SCALA_HOME/bin:$PATH`
- 6.
7. `#SPARK`
8. `export SPARK_HOME=/home/hadoop/spark-3.0`

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

#### 4. 刷新环境变量:

```
1. source /home/hadoop/.bash_profile
```

### 5. 分发安装包至从节点:

```
1. scp /home/hadoop/.bash_profile hadoop1:/home/hadoop
2. scp /home/hadoop/.bash_profile hadoop2:/home/hadoop
```

### 6.local 启动测试:

```
1. run-example SparkPi | grep "Pi is roughly"
2.
3. spark-shell --master local[*]
4. :q
5. pyspark --master local[2]
6. quit()
```

## 7. yarn 模式配置:

```
1. vi /home/hadoop/.bash_profile
2.
3. # 添加
4. export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

## 8. 刷新环境变量:

```
1. source /home/hadoop/.bash_profile
```

## 9. 继续配置 yarn 文件:

```
1. cd $HADOOP_CONF_DIR
2.
3. vi yarn-site.xml
4. >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
5. <property>
6.     <name>yarn.nodemanager.pmem-check-enabled</name>
7.     <value>>false</value>
8. </property>
9. <property>
10.    <name>yarn.nodemanager.vmem-check-enabled</name>
11.    <value>>false</value>
12.</property>
```

[illegible]

## 10. 从节点配置文件分发:

1. scp yarn-site.xml hadoop1:/home/hadoop-2.9.2/etc/hadoop/
2. scp yarn-site.xml hadoop2:/home/hadoop-2.9.2/etc/hadoop/

## 11. yarn 启动测试:

1. `start-dfs.sh`
2. `start-yarn.sh`
3. `pyspark --master yarn`
4. `spark-shell --master yarn`
- 5.
6. `stop-yarn.sh`
7. `stop-dfs.sh`

## 12. Standalone 模式配置:

分别在两个从节点上按照以上步骤上传解压 scale 和 spark 文件, 然后回到主节点进行操作。

- ```
1. cd $SPARK_HOME/conf
2. cp slaves.template slaves
3.
4. vi slaves
5. >>
6. #localhost
7. hadoop0
8. hadoop1
9. hadoop2
10.>>
11.cp spark-env.sh.template spark-env.sh
12.vi spark-env.sh
13.>>
14.export SPARK_MASTER=hadoop0
15.>>
16.
17.scp ./* hadoop1:/home/spark-3.0/conf
18.scp ./* hadoop2:/home/spark-3.0/conf
```

### 13. 启动测试:

- ```
1. start-master.sh
2. start-slaves.sh
3. spark-shell --master spark://hadoop0:7077
```



```
4. pyspark --master spark://hadoop0:7077
```

## 二、安装配置 Anaconda3

1. 上传 anaconda3 的安装包到主节点，并进行解压；
2. 启动安装：

```
1. sh Anaconda3-2020.11-Linux-x86_64.sh
2. ctrl+c
3. yes
4. /usr/local/program/anaconda3
5. yes
6. yes
```

3. 配置 anaconda3 的环境变量：

```
1. vi /home/hadoop/.bash_profile
2.
3. #anaconda3
4. export ANACONDA_HOME=/home/anaconda3
5. export PATH=$ANACONDA_HOME/bin:$PATH
6.
7. #pyspark
8. export PYSPARK_DRIVER_PYTHON=$ANACONDA_HOME/bin/ipython
9. export PYSPARK_PYTHON=$ANACONDA_HOME/bin/python
10. export PYSPARK_DRIVER_PYTHON_OPTS="notebook"
```

4. 刷新环境变量：

```
1. source /home/hadoop/.bash_profile
```

5. 进行 jupyter notebook 的相关配置：

```
1. conda config --set auto_activate_base False
2.
3. jupyter notebook --generate-config
4. /home/hadoop/.jupyter/jupyter_notebook_config.py
5. python
6. >>from IPython.lib import passwd
7. >>passwd()
8. enter
9. enter
10. (密钥)
11. >>quit()
12.
```

```

13.
14.c.NotebookApp.ip= '*'
15.c.NotebookApp.password='*****' #python 中的密码（填写上方密钥）
16.c.NotebookApp.open_browser=False
17.c.NotebookApp.port=8888
18.c.IPKernelApp.pylab='inline'

```

6. jupyter notebook 启动测试：

```

1. pyspark
2. pyspark --master spark://hadoop0:7077
3. pyspark --master yarn
4. 修改 C:\Windows\System32\drivers\etc\hosts
5. 192.168.159.100 hadoop0
6. http://hadoop0:8888
7. empty passwd

```

实验结果：

配置完成 spark 环境：

**Spark Master at spark://10.58.44.47:7077**

URL: spark://10.58.44.47:7077  
 Workers: 2  
 Cores: 16 Total, 0 Used  
 Memory: 4.0 GB Total, 0.0 B Used  
 Applications: 0 Running, 0 Completed  
 Drivers: 0 Running, 0 Completed  
 Status: ALIVE

| M                                   | Address       | Host, port, rack | State | Cores       | Memory              |
|-------------------------------------|---------------|------------------|-------|-------------|---------------------|
| worker-20150702172057-S1PA209-57802 | S1PA209-57802 |                  | ALIVE | 4 (0 Used)  | 2.0 GB (0.0 B Used) |
| worker-20150702172058-S1PA11-28095  | S1PA11-28095  |                  | ALIVE | 24 (0 Used) | 2.0 GB (0.0 B Used) |

**Running Applications**

| ID | Name | Cores | Memory per Node | Submitted Time | User | Status | Duration |
|----|------|-------|-----------------|----------------|------|--------|----------|
|----|------|-------|-----------------|----------------|------|--------|----------|

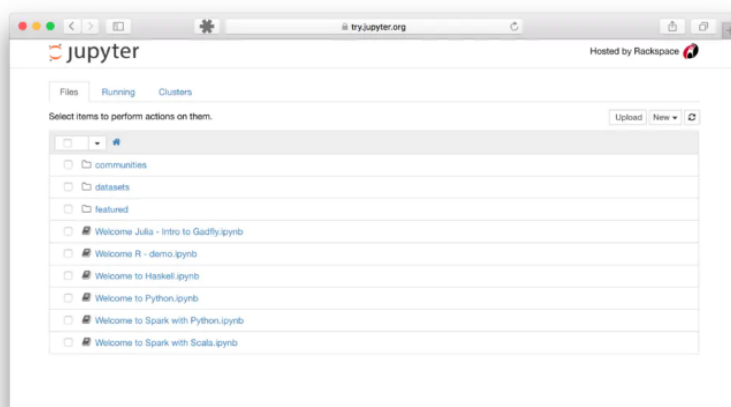
**Completed Applications**

| ID | Name | Cores | Memory per Node | Submitted Time | User | Status | Duration |
|----|------|-------|-----------------|----------------|------|--------|----------|
|----|------|-------|-----------------|----------------|------|--------|----------|

Spark 1.2.0

安装好了 anaconda3 和 scale，可以在 local、yarn 和 Standalone 模式下进行

jupyter notebook 的访问:



**实验分析:**

spark、scale、anaconda3 的安装其实都是环境变量的配置工作。

scala 是写分布式程序的一门非常方便的语言，因为 scala 几乎每个对象都有 map, reduce, filter 等方法，这跟 spark 的用法简直如出一辙。所以我们不仅安装了 anaconda3，而且安装了 scale。

三种模式：local 模式，本地运行。Standalone 模式，使用 Spark 原生的资源调度器。YARN 模式（生产模式中常用），使用 Hadoop 的 YARN 作为资源调度器。

**实验成绩:**

日期: \_\_\_\_年\_\_月\_\_日

# 河南理工大学教学上机实验报告

上机时间 2021 年 4 月 8 日

## 实验题目：

HDFS 文件系统的使用

## 实验目的和要求：

目的：

1. 启动全分布式集群，检查各个节点的进程是否完整。
2. 查看 Log 日志下是否存在错误记录。
3. 练习使用 hdfs 文件管理命令。

要求：

1. start-dfs.sh\start-yarn.sh 启动后，主节点 6 个进程，从节点 3 个进程。
2. \$HADOOP\_HOME/logs 文件夹下，查看 namenode, datanode, yarn 相关日志，使用 cat \*\*\*\* |grep "ERROR" 命令查看是否存在近期的 ERROR。
3. 在 HDFS 文件系统中完成以下操作：建立文件夹、上传文件、COPY 文件、文件的重命名、查看文件内容、删除文件、删除文件夹。
4. 关闭 HDFS, YARN。

## 实验过程：

一、启动集群并检查错误

1. 启动全分布式集群并检查：

```
1. start-all.sh
```

2. 检查主节点和从节点：

```
1. jps
```

主节点输出六个服务，两个从节点各三个服务即为正确

3. 查看 log 文件：

```
1. cd $HADOOP_HOME/logs
2.
3. # 采用 cat 命令进行 namenode, datanode, yarn 的 error 查看
4. cat **** |grep "ERROR"
```

通常错误输出会有空白处，查看错误的发生日期，如果发生于近期需要进行解决

4. 启动 yarn 和 hdfs：

```
1. start-hdfs.sh
2. start-yarn.sh
```

## 二、hdfs 命令学习

1. 在本地 Linux 文件系统的“/home/hadoop/”目录下创建一个文件 txt，里面可以随意输入一些单词：

```
1. cd /usr/local/hadoop
2.
3. touch flie.txt
4.
5. gedit file.txt
```

2. 在本地查看文件位置 (ls)：

```
1. ls -al
```

3. 在本地显示文件内容：

```
1. cat flie.txt
```

4. 使用命令把本地文件系统上的“txt”上传到 HDFS 中的当前用户目录的 input 目录下：

```
1. ./sbin/start-dfs.sh
2.
3. ./bin/hdfs dfs -mkdir -p /user/hadoop
4.
5. ./bin/hdfs dfs -mkdir input
6.
7. ./bin/hdfs dfs -put ./file.txt input
```

5. 查看 hdfs 中的文件 (-ls)：

```
1. ./bin/hdfs dfs -ls
```

6. 显示 hdfs 中该的文件内容：

```
1. ./bin/hdfs dfs -cat input/flie.txt
```

7. 删除本地的 txt 文件并查看目录：

```
1. ./bin/hdfs dfs -rm -ls input/flie.txt
```

8. 从 hdfs 中将 txt 下载到本地原来的位置：

```
1. ./bin/hdfs dfs -get input/flie.txt ~/hadoop
```

9. 从 hdfs 中删除 txt 并查看目录：

```
1. ./bin/hdfs dfs -rm -r /input/flie.txt
2.
3. ./bin/hdfs dfs -ls
```

10. 关闭 yarn 和 hdfs：

```
1. stop-hdfs.sh
2. stop-yarn.sh
```

实验结果：

在开启 hdfs 和 yarn 后，进入到 log 文件检查了自安装以来的错误和运行记录，通过 cat 命令我看到了自己初期安装的一些错误文件，但是后期没有再出现。

使用 hdfs 命令完成各种操作命令：

| Hadoop Overview   Hadoop   Hadoop   Hadoop   Hadoop   Hadoop   Hadoop   Hadoop |       |            |         |                     |             |            |        |
|--|-------|------------|---------|---------------------|-------------|------------|--------|
| Browse Directory   |       |            |         |                     |             |            |        |
| /  |       |            |         |                     |             |            | Out    |
| Permissions  | Owner | Group      | Size    | Last Modified       | Replication | Block Size | Name   |
| -rw-r--r--   | root  | supergroup | 1.36 KB | 2018/4/29 下午3:15:32 | 1           | 128 MB     | as     |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/4/29 下午3:16:41 | 0           | 0 B        | asdir  |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/4/29 下午3:18:25 | 0           | 0 B        | adire  |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/4/29 下午3:19:11 | 0           | 0 B        | bl.txt |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/5/30 上午9:17:36 | 0           | 0 B        | bbare  |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/4/29 下午3:19:34 | 0           | 0 B        | root   |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/5/29 上午9:18:00 | 0           | 0 B        | log    |
| -rw-r--r--   | root  | supergroup | 0 B     | 2018/5/29 下午2:31:12 | 0           | 0 B        | root   |

实验分析：

这次实验中，我对 HDFS 在 Hadoop 体系结构中的角色熟练使用有了一定的了解，也熟悉了一点 HDFS 操作常用的 Shell 命令，熟悉 HDFS 操作命令。达到本次实验目的。

实验成绩：

日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日