目录

1. 实验一： 基础环境配置

1.1. 实验目的

完成本实验，您应该能够：

- 掌握 linux 的网络部署
- 掌握 linux 的名称配置
- 掌握 linux 的主机名与 ip 映射

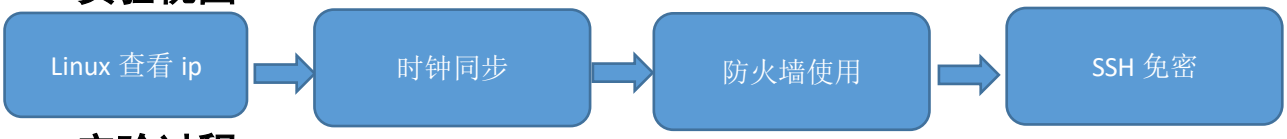
1.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 了解 ip 映射的含义

1.3. 实验环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	HDFS、YARN、MapReduce 等，其他服务根据实验需求安装

1.4. 实验视图



1.5. 实验过程

1.5.1. 实验任务一： Linux 基础环境配置

1.5.1.1. 步骤一：查看 ip

[root@VM-M-01594949483071 ~]# ip a

```
[root@VM-M-01594949483071 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP q
len 1000
    link/ether 02:00:29:17:00:06 brd ff:ff:ff:ff:ff:ff
    inet 192.168.90.205/24 brd 192.168.90.255 scope global dynamic ens35
       valid_lft 2696378sec preferred_lft 2696378sec
    inet6 fe80::48d9:128b:db06:18e1/64 scope link
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
 qlen 1000
    link/ether 52:54:00:d9:cc:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 sta
te DOWN qlen 1000
    link/ether 52:54:00:d9:cc:41 brd ff:ff:ff:ff:ff:ff
```

图 1-2(master)ip 地址

[root@VM-M-01594949481966 ~]# ip a

```
[root@VM-M-01594949481966 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP q
len 1000
    link/ether 02:00:4a:15:00:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.90.48/24 brd 192.168.90.255 scope global dynamic ens35
       valid_lft 2577524sec preferred_lft 2577524sec
    inet6 fe80::132e:5022:7372:6b5/64 scope link
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
 qlen 1000
    link/ether 52:54:00:d9:cc:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 sta
te DOWN qlen 1000
    link/ether 52:54:00:d9:cc:41 brd ff:ff:ff:ff:ff:ff
```

图 1-3(slave1)ip 地址

[root@VM-M-01594949480907 ~]# ip a

```
[root@VM-M-01594949480907 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP q
len 1000
    link/ether 02:00:2c:7d:00:04 brd ff:ff:ff:ff:ff:ff
    inet 192.168.90.66/24 brd 192.168.90.255 scope global dynamic ens35
       valid_lft 2584596sec preferred_lft 2584596sec
    inet6 fe80::d0dc:2e40:8f10:3bae/64 scope link
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
 qlen 1000
    link/ether 52:54:00:d9:cc:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 sta
te DOWN qlen 1000
    link/ether 52:54:00:d9:cc:41 brd ff:ff:ff:ff:ff:ff
```

图 1-4(slave2)ip 地址

1.5.1.2. 步骤四：修改主机名

[root@VM-M-01594949483071 ~]# hostnamectl set-hostname master

[root@VM-M-01594949483071 ~]# bash

[root@VM-M-01594949481966 ~]# hostnamectl set-hostname slave1

[root@VM-M-01594949481966 ~]# bash

[root@VM-M-01594949480907 ~]# hostnamectl set-hostname slave2

[root@VM-M-01594949480907 ~]# bash

保存并退出

1.5.1.3. 步骤五：主机映射

[root@master ~]# vi /etc/hosts

[root@slave1 ~]# vi /etc/hosts

[root@slave2 ~]# vi /etc/hosts

添加图 1.的配置

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.90.205  master
192.168.90.48   slave1
192.168.90.66   slave2
```

保存并退出

1.5.2. 实验任务二：时钟同步

1.5.2.1. 步骤一：修改配置文件

[root@master ~]# vi /etc/sysconfig/ntpd

```
# Command line options for ntpd
OPTIONS="-g"
YS_HWLOCK=yes
```

[root@slave1 ~]# vi /etc/sysconfig/ntpd

```
# Command line options for ntpd
OPTIONS="-g"
YS_HWLOCK=yes
```

[root@slave2 ~]# vi /etc/sysconfig/ntpd

```
# Command line options for ntpd
OPTIONS="-g"
YS_HWLOCK=yes
```

图 1-5 配置文件内容

保存并退出

1.5.2.2. 步骤二：同步时间

[root@master ~]# systemctl start ntpd

[root@master ~]# date

```
Thu Jul 30 17:39:28 CST 2020
```

[root@slave1 ~]# systemctl start ntpd

[root@slave1 ~]# date

```
Thu Jul 30 17:39:40 CST 2020
```

[root@slave2 ~]# systemctl start ntpd

[root@slave2 ~]# date

```
Thu Jul 30 17:39:52 CST 2020
```

图 1-6 时间同步查看

1.5.3. 实验任务三：防火墙

1.5.3.1. 步骤一：关闭防火墙（三台都要关闭）

[root@master ~]# systemctl stop firewalld.service

[root@slave1 ~]# systemctl stop firewalld.service

[root@slave2 ~]# systemctl stop firewalld.service

1.5.3.2. 步骤二：关闭防火墙自启

[root@master ~]# systemctl disable firewalld.service

[root@slave1 ~]# systemctl disable firewalld.service

[root@slave2 ~]# systemctl disable firewalld.service

1.5.3.3. 步骤三：查看防火墙状态

[root@master ~]# systemctl status firewalld.service

```
[root@master ~]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor p
reset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

[root@slave1 ~]# systemctl status firewalld.service

```
[root@slave1 ~]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor p
reset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

[root@slave2 ~]# systemctl status firewalld.service

```
[root@slave2 ~]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor p
reset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

1.5.4. 实验任务四：ssh 免密

1.5.4.1. 步骤一：创建免密（三个主机同时进行）

[root@master ~]# su - hadoop

[hadoop@master ~]$ ssh-keygen -t rsa -P ""

输入回车

[root@slave1 ~]# su - hadoop

[hadoop@slave1 ~]$ ssh-keygen -t rsa -P ""

[root@slave2 ~]# su - hadoop

[hadoop@slave2 ~]$ ssh-keygen -t rsa -P ""

```
[hadoop@master hadoop]$ ssh-keygen -t rsa -P ''
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:nWbU53wRj9XZX7sQBODQyJFx7+iI/te597ocyhYJ0hA hadoop@master
The key's randomart image is:
+---[RSA 2048]----+
|       .EBo..o. .=|
|        =+... . +*|
|         o.... +.=|
|        . =o. = .o|
|         S.*.. + o|
|       . oo o   o |
|      . . .. o.   |
|     .     ..+o..  |
|      .... .+o++.  |
+----[SHA256]-----+
```

```
[hadoop@slave1 root]$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:EDWKv5GhRdl9iG6RrWxHPoLCJzD4unDgDLGPcWcdt3g hadoop@slave1
The key's randomart image is:
+---[RSA 2048]----+
|  .     o+o= .   |
| o o   ooo*.= .  |
|  + +. .=B * .   |
| = o *+=+E +     |
| +B o.+++So .    |
| +oo     o       |
| .o     .        |
| .               |
|                 |
+----[SHA256]-----+
```

1.5.4.2. 步骤二：创建公钥

[hadoop@master ~]$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
[hadoop@slave1 ~]$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
[hadoop@slave2 ~]$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys

1.5.4.3. 步骤三：给公钥执行权限

[hadoop@master ~]$ chmod 700 ~/.ssh/authorized_keys
[hadoop@slave1 ~]$ chmod 700 ~/.ssh/authorized_keys
[hadoop@slave2 ~]$ chmod 700 ~/.ssh/authorized_keys

1.5.4.4. 步骤四：将公钥传输给 slave1 和 slave2

[hadoop@master ~]$ scp ~/.ssh/authorized_keys hadoop@slave1:~/.ssh/
[hadoop@master ~]$ scp ~/.ssh/authorized_keys hadoop@slave2:~/.ssh/
第一次传输是需要输入密码的，后面就不用了

1.5.4.5. 步骤五：登陆测试

[hadoop@master ~]$ ssh slave1
[hadoop@master ~]$ ssh slave2

```
[hadoop@master ~]$ ssh slave1
Last failed login: Sun Jul 19 12:11:01 CST 2020 from 192.168.90.205 on ssh:notty
There were 3 failed login attempts since the last successful login.
Last login: Sun Jul 19 12:02:53 2020
[hadoop@slave1 ~]$ exit
登出
Connection to slave1 closed.
[hadoop@master ~]$ ssh slave2
Last failed login: Sun Jul 19 12:28:26 CST 2020 from 192.168.90.205 on ssh:notty
There were 3 failed login attempts since the last successful login.
Last login: Sun Jul 19 12:05:37 2020
[hadoop@slave2 ~]$ exit
登出
Connection to slave2 closed.
```

好了，这是登陆成功了

2. 实验二： Hadoop 集群部署

2.1. 实验目的

完成本实验，您应该能够：

- 掌握 xftp 的使用
- 掌握 Linux 下解压安装文件
- 掌握 hadoop 和 jdk 的环境部署

2.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 了解 hadoop 配置文件的用法

2.3. 实验环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	HDFS、YARN、MapReduce、 jdk 等，其他服务根据实验需求安装

2.4. 实验视图

2.5. 实验过程

2.5.1. 实验任务一： Hadoop 软件安装

2.5.1.1. 步骤一：解压安装 Hadoop

[hadoop@master ~]$ su root
[hadoop@master hadoop]$ cd
[root@master ~]# tar -zxvf /opt/software/hadoop-2.7.1.tar.gz -C /usr/local/src/

2.5.1.2. 步骤二：更改 hadoop 文件名

[root@master ~]# mv /usr/local/src/hadoop-2.7.1 /usr/local/src/hadoop

2.5.1.3. 步骤三：配置 hadoop 环境变量

[root@master ~]# vi /etc/profile

进行如下配置

export HADOOP_HOME=/usr/local/src/hadoop

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

保存并退出

2.5.1.4. 步骤四：修改目录所有者和所有者组

上述安装完成的 Hadoop 软件只能让 root 用户使用，要让 hadoop 用户能够运行 hadoop 软件，需要将目录/usr/local/src 的所有者改为 hadoop 用户。

[root@master ~]# chown -R hadoop:hadoop /usr/local/src

[root@master ~]# ll /usr/local/src/

```
[root@master ~]# ll /usr/local/src/
总用量 0
drwxr-xr-x 9 hadoop hadoop 149 6月  29 2015 hadoop
[root@master ~]#
```

/usr/local/src 目录的所有者已经改为 hadoop 了。

2.5.2. 实验任务二：安装 JAVA 环境

2.5.2.1. 步骤一：解压安装 jdk

[root@master ~]# tar -zxvf /opt/software/jdk-8u152-linux-x64.tar.gz -C /usr/local/src

2.5.2.2. 步骤二：更改 jdk 的名称

[root@master ~]# mv /usr/local/src/jdk1.8.0_152/ /usr/local/src/java

[root@master ~]# chown -R hadoop:hadoop /usr/local/src/java

2.5.2.3. 步骤三：配置 java 的环境变量

[root@master ~]# vi /etc/profile

配置如下环境

export JAVA_HOME=/usr/local/src/java #JAVA_HOME 指向 JAVA 安装目录

export PATH=$PATH:$JAVA_HOME/bin #将 JAVA 安装目录加入 PATH 路径

保存并退出

2.5.2.4. 步骤四：生效环境变量

[root@master ~]# source /etc/profile

[root@master ~]# update-alternatives --install /usr/bin/java java /usr/local/src/java/bin/java 200

[root@master ~]# update-alternatives --set java /usr/local/src/java/bin/java

2.5.2.5. 步骤五：查看 java 和 hadoop

[root@master ~]# java -version

```
[root@master ~]# java -version
java version "1.8.0_152"
Java(TM) SE Runtime Environment (build 1.8.0_152-b16)
Java HotSpot(TM) 64-Bit Server VM (build 25.152-b16, mixed mode)
[root@master ~]#
```

[root@master ~]# hadoop version

```
[root@master ~]# hadoop version
Hadoop 2.7.1
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 15ecc87ccf4a022
8f35af08fc56de536e6ce657a
Compiled by jenkins on 2015-06-29T06:04Z
Compiled with protoc 2.5.0
From source with checksum fc0a1a23fc1868e4d5ee7fa2b28a58a
This command was run using /usr/local/src/hadoop/share/hadoop/common/hadoop-comm
on-2.7.1.jar
[root@master ~]#
```

这样就是安装成功了

2.5.3. 实验任务三：集群配置

2.5.3.1. 步骤一：进入到 hadoop 配置文件的目录下

[root@master ~]# cd /usr/local/src/hadoop/etc/hadoop

2.5.3.2. 步骤二：配置 core-site.xml

[root@master hadoop]# vi core-site.xml

在文件里添加如下配置

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://master:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>file:/usr/local/src/hadoop/tmp</value>
        </property>
</configuration>
```

保存并退出

2.5.3.3. 步骤三：配置 hadoop-env.sh

[root@master hadoop]# vi hadoop-env.sh

在文件的最下方添加如下环境配置

```
export JAVA_HOME=/usr/local/src/java
export HADOOP_PERFIX=/usr/local/src/hadoop
export HADOOP_OPTS="-Djava.library.path=$HADOOP_PERFIX/lib:$HADOOP_PERFIX/lib/natice"
```

保存并退出

2.5.3.4. 步骤四：配置 hdfs-site.xml

[root@master hadoop]# vi hdfs-site.xml

在文件里添加如下配置

```
<configuration>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:/usr/local/src/hadoop/dfs/name</value>
        </property>
```

```
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:/usr/local/src/hadoop/dfs/data</value>
        </property>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
</configuration>
```

保存并退出

2.5.3.5. 步骤五：配置 mapred-site.xml

将副本拷贝成 mapred-queues.xml

[root@master hadoop]# cp mapred-site.xml.template mapred-site.xml

[root@master hadoop]# vi mapred-site.xml

在文件里添加如下配置

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <property>
                <name>mapreduce.jobhistory.address</name>
                <value>master:10020</value>
        </property>
        <property>
                <name>mapreduce.jobhistory.webapp.address</name>
                <value>master:19888</value>
        </property>
</configuration>
```

保存并退出

2.5.3.6. 步骤六：配置 yarn-site.xml

[root@master hadoop]# vi yarn-site.xml

在文件里添加如下配置

```
<configuration>
        <property>
                <name>yarn.resourcemanager.address</name>
                <value>master:8032</value>
        </property>
        <property>
                <name>yarn.resourcemanager.scheduler.address</name>
                <value>master:8030</value>
        </property>
        <property>
```

```
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>master:8031</value>
    </property>
    <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>master:8033</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>master:8088</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
</configuration>
```

保存并退出

2.5.3.7. 步骤七：配置 masters 文件

执行以下命令修改 masters 配置文件。

[root@master hadoop]# vi masters　#加入以下配置信息

master

保存并退出

2.5.3.8. 步骤八：配置 slaves

[root@master hadoop]# vi slaves

在文件里改成如下配置

slave1

slave2

保存并退出

2.5.3.9. 步骤九：创建目录

[root@master hadoop]# mkdir -p /usr/local/src/hadoop/dfs/name

[root@master hadoop]# mkdir -p /usr/local/src/hadoop/dfs/data

[root@master hadoop]# mkdir -p /usr/local/src/hadoop/tmp

2.5.4. 实验任务四：主从节点文件的分发

2.5.4.1. 步骤一：分发 hadoop 目录

[root@master hadoop]# scp -r /usr/local/src/hadoop/ root@slave1:/usr/local/src/

[root@master hadoop]# scp -r /usr/local/src/hadoop/ root@slave2:/usr/local/src/

[root@master hadoop]# scp -r /usr/local/src/java/ root@slave1:/usr/local/src/
[root@master hadoop]# scp -r /usr/local/src/java/ root@slave2:/usr/local/src/

2.5.4.2. 步骤二：分发环境配置

[root@master hadoop]# scp -r /etc/profile root@slave1:/etc/
[root@master hadoop]# scp -r /etc/profile root@slave2:/etc/

2.5.4.3. 在每个节点上修改/usr/local/src/hadoop 目录的权限

[root@master ~]# chown -R hadoop:hadoop /usr/local/src/hadoop
[root@slave1~]# chown -R hadoop:hadoop /usr/local/src/hadoop
[root@slave2~]# chown -R hadoop:hadoop /usr/local/src/hadoop

2.5.4.4. 步骤三：生效环境配置

[root@slave1 ~]# source /etc/profile
[root@slave1 ~]# update-alternatives --install /usr/bin/java java /usr/local/src/java/bin/java 200
[root@slave1 ~]# update-alternatives --set java /usr/local/src/java/bin/java

[root@slave2 ~]# source /etc/profile
[root@slave2 ~]# update-alternatives --install /usr/bin/java java /usr/local/src/java/bin/java 200
[root@slave2 ~]# update-alternatives --set java /usr/local/src/java/bin/java

3. 实验三： Hadoop 集群启动测试

3.1. 实验目的

完成本实验，您应该能够：

- 掌握 hadoop 页面的端口
- 掌握节点的使用
- 掌握 mapreduce 的使用

3.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 了解 mapreduce 的含义

3.3. 实验环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘

运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	HDFS、YARN、MapReduce、jdk 等，其他服务根据实验需求安装

3.4. 实验视图

3.5. 试验过程

3.5.1. 实验任务一：hadoop 启动

3.5.1.1. 步骤一：格式化元数据

进入 hadoop 用户

[root@master hadoop]# su hadoop

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

[root@slave1 hadoop]# su hadoop

[hadoop@slave1 hadoop]$ source /etc/profile

[root@slave2 hadoop]# su hadoop

[hadoop@slave2 hadoop]$ source /etc/profile

[hadoop@master hadoop]$ hdfs namenode -format

状态为 0 显示的是成功

```
ges with txid >= 0
20/07/19 11:26:09 INFO util.ExitUtil: Exiting with status 0
20/07/19 11:26:09 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at master/192.168.90.205
************************************************************/
```

3.5.1.2. 步骤二：启动 hdfs

[hadoop@master ~]$ start-dfs.sh

```
[hadoop@master ~]$ start-dfs.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/local/src/hadoop/logs/hadoop-hadoop-n
amenode-master.out
```

3.5.1.3. 步骤三：启动 yarn

[hadoop@master ~]$ start-yarn.sh

```
[root@master hadoop]# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/src/hadoop/logs/yarn-hadoop-reso
urcemanager-master.out
```

3.5.2. 实验任务二：hadoop 的查看

3.5.2.1. 步骤一：进程的查看

[hadoop@master ~]$ jps

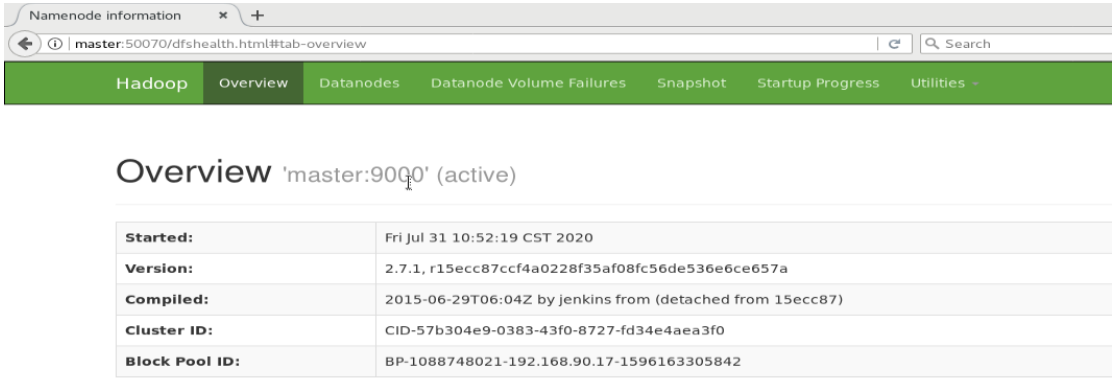
```
[hadoop@master hadoop]$ jps
44962 ResourceManager
44563 NameNode
45227 Jps
44767 SecondaryNameNode
[hadoop@master hadoop]$
```

```
[hadoop@slave1 hadoop]$ jps
35326 NodeManager
35454 Jps
35183 DataNode
```

```
[root@slave2 ~]# jps
34836 Jps
34565 DataNode
34702 NodeManager
[root@slave2 ~]#
```

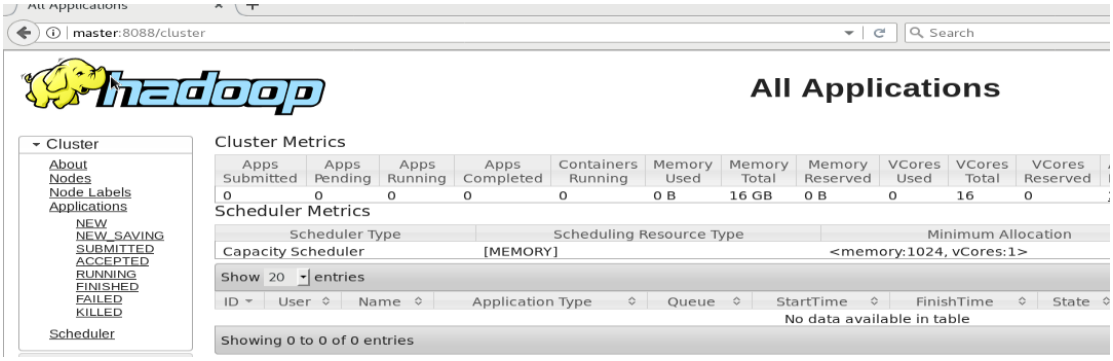
3.5.2.2. 步骤二：master：50070 查看

进入到浏览器查看



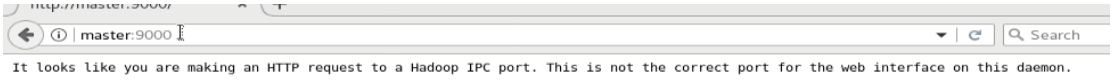
3.5.2.3. 步骤三：master：8088 查看

查看成功



3.5.2.4. 步骤四：master：9000 查看

显示有人访问



3.5.3. 实验任务三：mapreduce 测试

3.5.3.1. 步骤一：创建一个测试文件

[hadoop@master ~]$ vi a.txt

内容如下：

HELLO WORD

HELLO HADOOP

HELLO JAVA

3.5.3.2. 步骤二：在 hdfs 创建文件夹

[hadoop@master ~]$ hadoop fs -mkdir /input

3.5.3.3. 步骤三：将 a.txt 传输到 input 上

[hadoop@master ~]$ hadoop fs -put ~/a.txt /input

3.5.3.4. 步骤四：进入到 jar 包测试文件目录下

[hadoop@master hadoop]$ cd /usr/local/src/hadoop/share/hadoop/mapreduce/

3.5.3.5. 步骤五：测试 mapreduce

[hadoop@master mapreduce]$ hadoop jar hadoop-mapreduce-examples-2.7.1.jar wordcount /input/a.txt /output

成功如下：

```
20/07/19 14:06:34 INFO mapreduce.Job:  map 0% reduce 0%
20/07/19 14:06:38 INFO mapreduce.Job:  map 100% reduce 0%
20/07/19 14:06:44 INFO mapreduce.Job:  map 100% reduce 100%
20/07/19 14:06:45 INFO mapreduce.Job: Job job_1595138281785_0001 completed succe
ssfully
20/07/19 14:06:45 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=65
                FILE: Number of bytes written=231331
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=130
                HDFS: Number of bytes written=39
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
```

```
                Map input records=3
                Map output records=6
                Map output bytes=59
                Map output materialized bytes=65
                Input split bytes=95
                Combine input records=6
                Combine output records=5
                Reduce input groups=5
                Reduce shuffle bytes=65
                Reduce input records=5
                Reduce output records=5
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=117
                CPU time spent (ms)=1130
                Physical memory (bytes) snapshot=431865856
                Virtual memory (bytes) snapshot=4200841216
                Total committed heap usage (bytes)=314048512
    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=35
    File Output Format Counters
```

注：如果需要重复执行，需要删除输出目录，否则会报错

[hadoop@master mapreduce]$ hdfs dfs -rm -r -f /output

3.5.3.6. 步骤六：查看 hdfs 下的传输结果

[hadoop@master mapreduce]$ hadoop fs -lsr /output

```
[hadoop@master mapreduce]$ hadoop fs -lsr /output
lsr: DEPRECATED: Please use 'ls -R' instead.
-rw-r--r--   3 hadoop supergroup          0 2020-07-19 14:06 /output/_SUCCESS
-rw-r--r--   3 hadoop supergroup         39 2020-07-19 14:06 /output/part-r-00000
[hadoop@master mapreduce]$
```

3.5.3.7. 步骤七：查看文件测试的结果

[hadoop@master mapreduce]$ hadoop fs -cat /output/part-r-00000