Hadoop&HBase&Zookeeper
安装手册

基于 VMware、CentOS 的 Hadoop 集群安装

2014/11

河南省电力公司电力科学研究院智能电网所

ShangBingBing

目录

1. 概述

本次 Hadoop 集群安装实验，采用在 VMware 虚拟机下安装多个 CentOS 系统的方式进行；4 个 CentOS 系统中，其中，1 个为 Master 机，作为 NameNode；另外 3 个为 Slave 机，作为 DataNode，均采用独立静态 IP 地址配置。

2. 安装准备

2.1 软件系统清单

序号	名称	描述
1	JDK1.6-linux.bin	这里推荐使用 JDK1.6 版本,因为我们的 CentOS 为 32 位，所以这里的 JDK 也采用 32 位版本。可以从 Oracle 官方网站或者其他网站下载。
2	VMware-WorkStation-10	虚拟机安装程序，这里采用 VMware10 版本。
3	CentOS-6.5-i386	集群操作系统均采用 CentOS6.5 版本，32 位操作系统。
4	hadoop-2.2.0.tar	Hadoop 安装程序。可从 Apache 官网上下载。
5	hbase-0.96.2-hadoop2-bin.tar	HBase 安装程序。可从 Apache 官网上下载。
6	zookeeper-3.4.6.tar	
7		

2.2 IP 地址配置清单

序号	主机名	IP 地址
1	master.hadoop	192.168.58.130
2	slave1.hadoop	192.168.58.131
3	slave2.hadoop	192.168.58.132
4	slave3.hadoop	192.168.58.133

以上 4 个虚拟机操作系统中，**NETMASK=255.255.255.0，GATEWAY=192.168.58.2。**

3. 安装配置虚拟机

在开始安装 Hadoop 和 HBase 之前，需要先安装并配置好虚拟机环境。

3.1 安装虚拟机软件

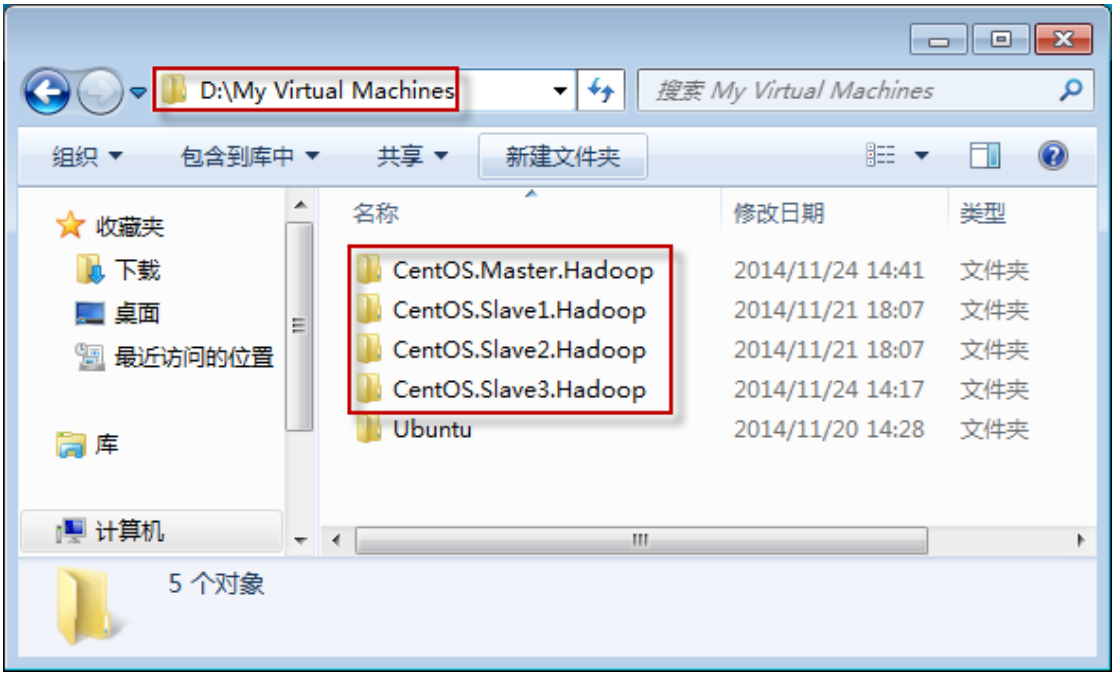
对于如何安装 VMware 软件，这里不做过多的说明，用户可以从网络上下载相关的安装教程细读，其实 VMware 的安装和其他的应用软件没有什么不同，并没有什么需要特殊声明的。

3.2 安装 CentOS 系统

3.2.1 安装说明

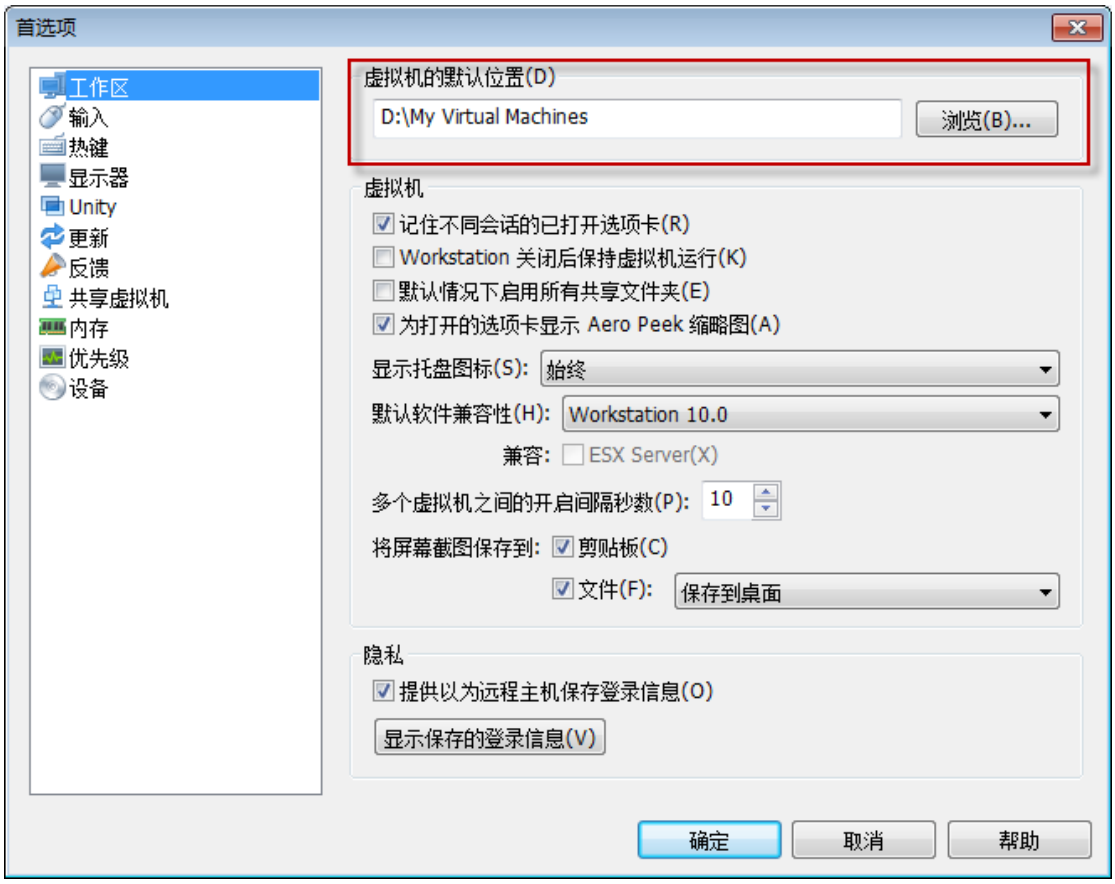
由于我们将要安装 4 个 CentOS 虚拟机，所以需要有足够大的磁盘空间。默认情况下，每个虚拟机最大的磁盘空间是 20G，如果磁盘空间允许的话，建议保持默认的设置。

4 个虚拟机可以集中安装在同一个磁盘中，也可以分开安装在不同的磁盘下，在本次实验中，将 4 个虚拟机集中安装在 D 盘下，具体位置如下如下图所示：



3.2.2 配置虚拟机默认路径

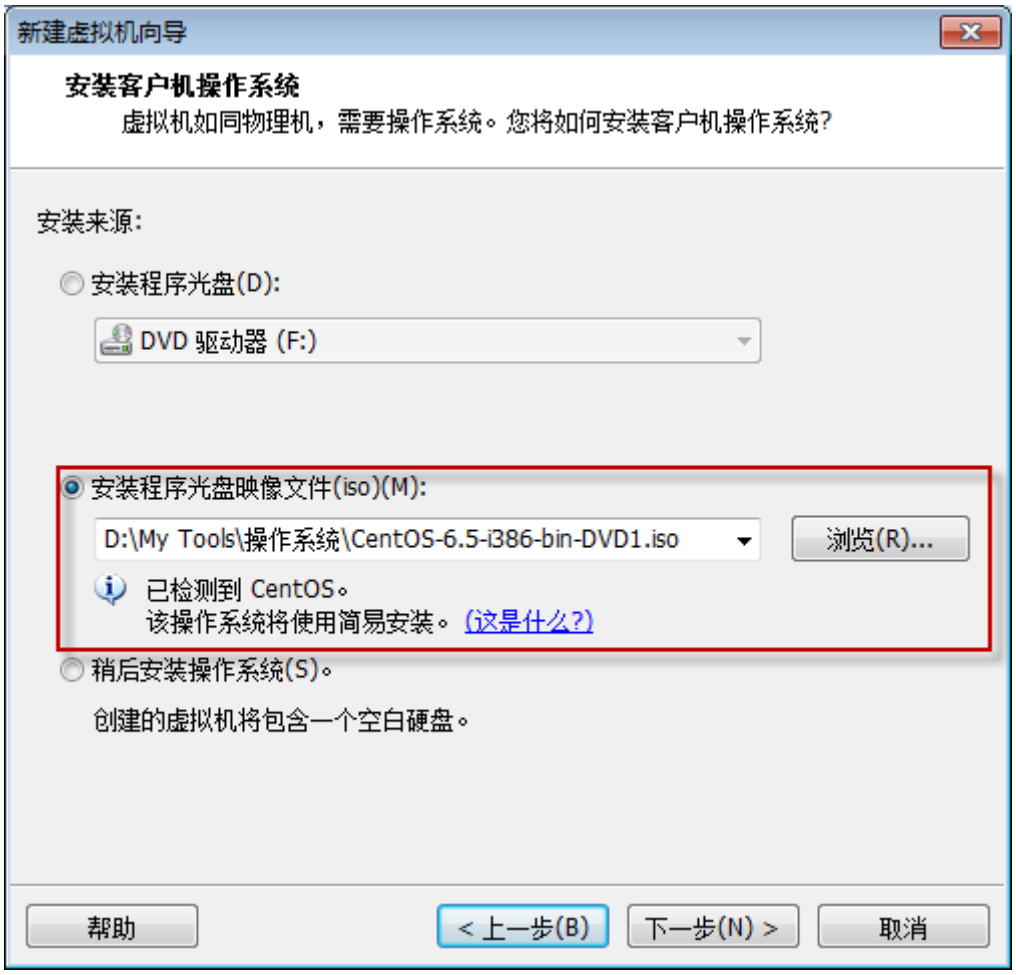
在安装 CentOS 系统之前，需要先在 VMware 中设置虚拟机的默认安装位置。打开 VMware，点击“编辑—首选项”，打开“首选项”窗口，如下图所示。



3.2.3 安装配置 CentOS 系统

- 在 VMware“主页”面板中点击“创建新的虚拟机”按钮，打开“新建虚拟机向导”窗口。

- 点击“下一步”，选择本地的 CentOS 镜像安装文件。



- 点击“下一步”，配置用户信息。在本次实验中，4 个操作系统的用户名和密码均为 hadoop。

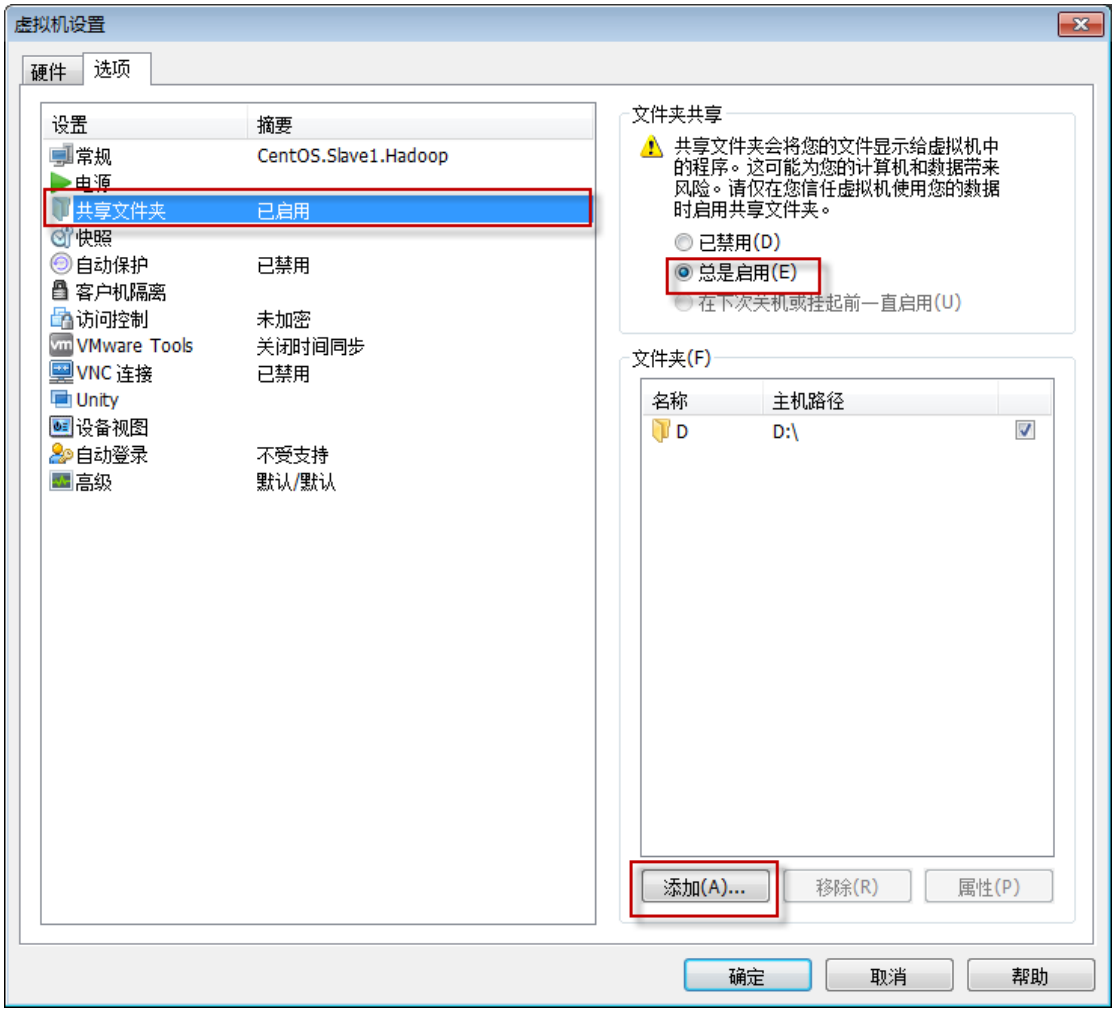
  这里需要特别说明一下，这里虽然创建了一个 hadoop 用户，但是在安装 hadoop 和 hbase 过程中并没有使用此用户，而是使用 root 用户，请谨记。如果你想在本次实验中使用 hadoop 或者其他用户安装 hadoop 和 hbase，请将下文中涉及 root 的地方替换成 hadoop 或者其他用户名称即可，同时要赋予给它对应的权限。

- 点击“下一步”，开始进行安装，不用害怕，没有什么大不了的，系统会自己自动安装。
- 安装完毕后，系统会加载登陆页面。

点击“其他”，输入用户名“root”，确定后输入密码“hadoop”，即可进入 CentOS 系统桌面。

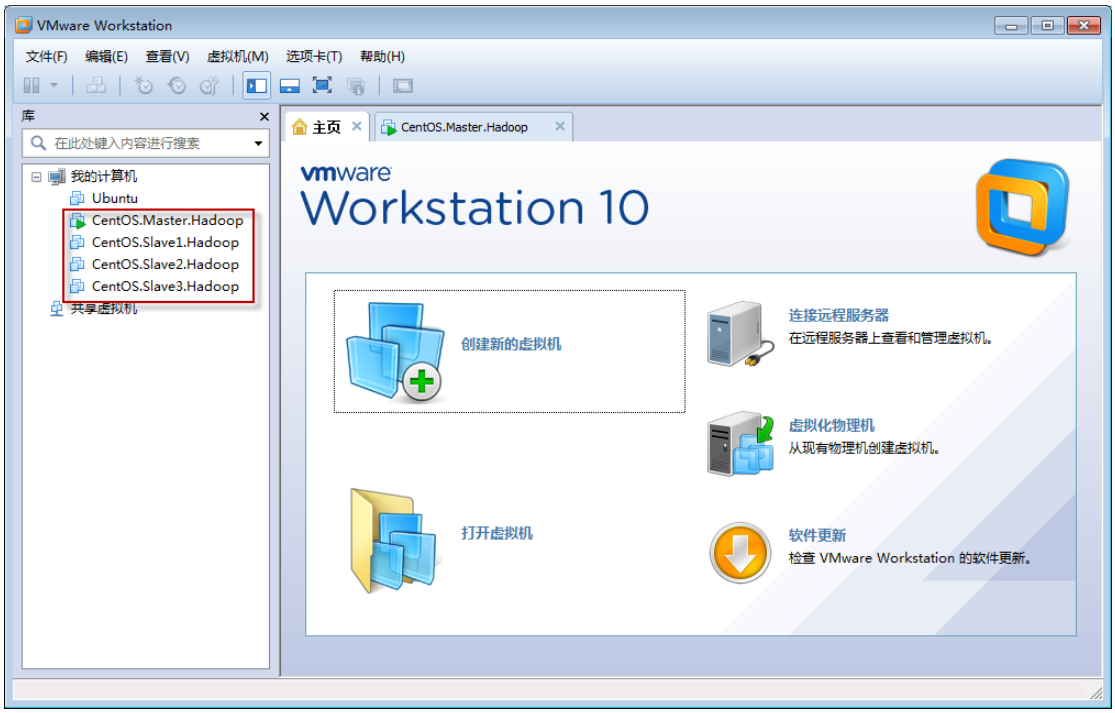
3.2.4 共享本机资源

打开 VMware，在左侧的虚拟机列表中选中一个虚拟机节点，单击右键，点击“设置”菜单，进入“虚拟机设置”窗口，进行如下图所示的配置。



这样虚拟机就可以跟本机进行资源共享操作了，便于从本机复制 JDK、Hadoop 和 HBase 等软件资源。

重复以上操作，安装 4 个 CentOS 虚拟机系统，安装结果如下图所示：

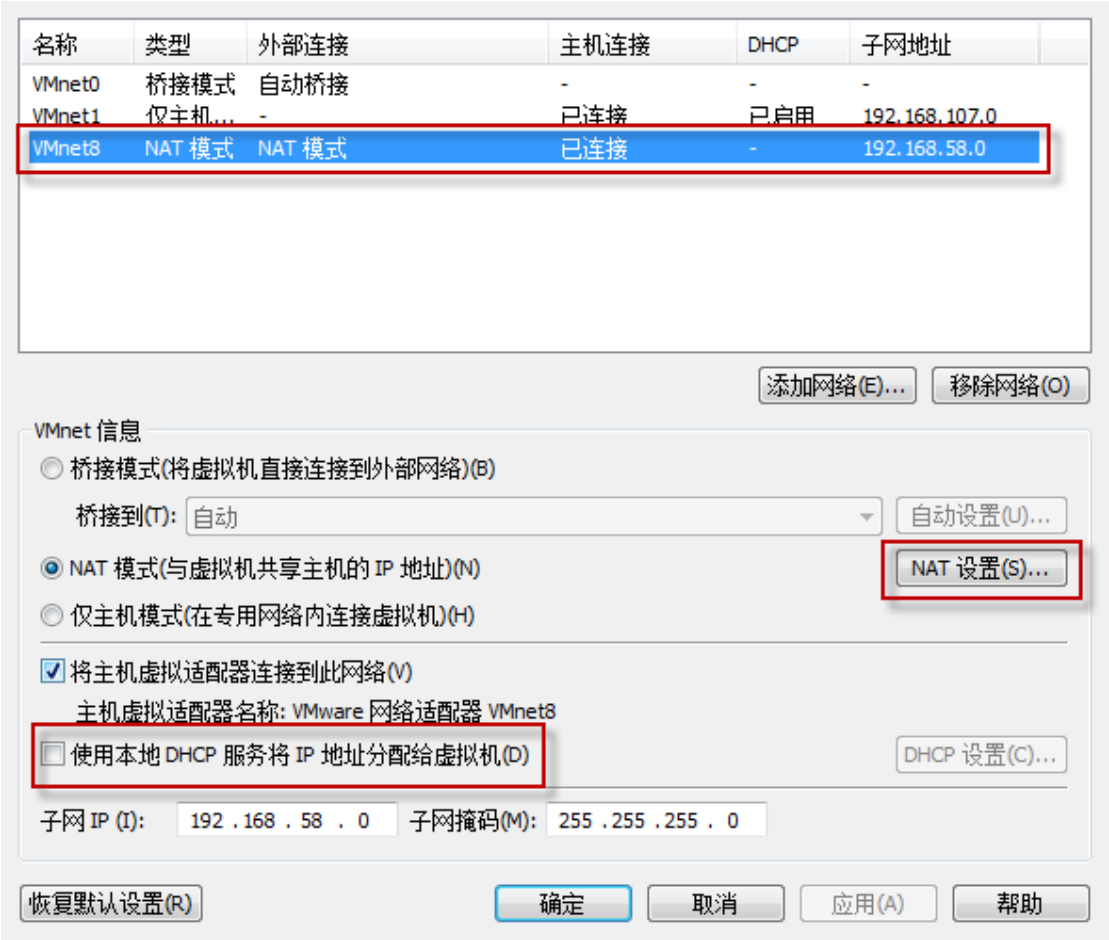


3.3 配置静态 IP 地址

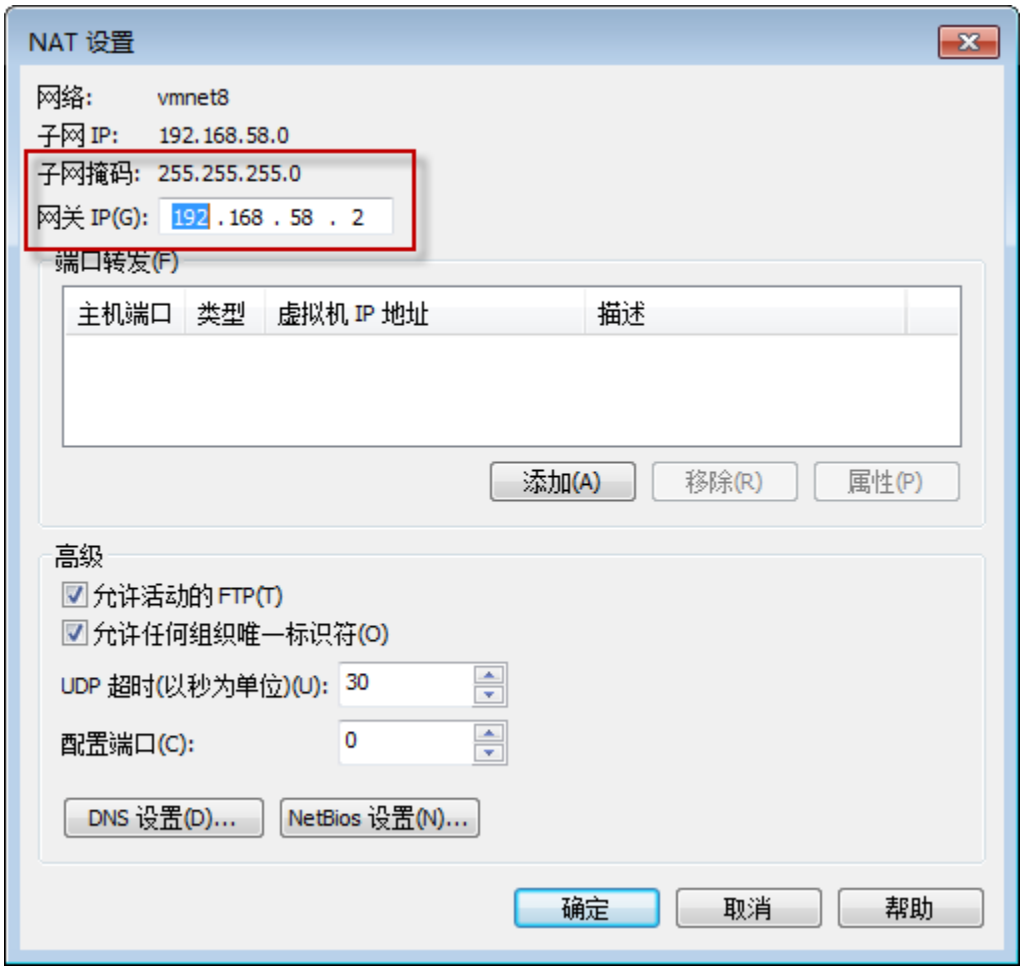
之所以要在每个虚拟机中配置静态 IP 地址，是因为 VMware 中虚拟机的 IP 地址默认配置策略是 DHCP，每次重启虚拟机后，IP 地址都会发生变化，对于 Hadoop 集群来说，这是绝对不允许的。

3.3.1 关闭 VMware 的 DHCP

在 VMware 中，点击“编辑—虚拟网络编辑器”，打开“虚拟网络编辑器”窗口。



选择 VMnet8，去掉 Use local DHCP service to distribute IP address to VMs 选项。点击 NAT Settings 查看一下 GATEWAY 地址：



点击 OK 就可以了。

3.3.2 设置 CentOS 静态 IP

涉及到三个配置文件，分别是：

/etc/sysconfig/network

/etc/sysconfig/network-scripts/ifcfg-eth0

/etc/resolv.conf

- 首先修改/etc/sysconfig/network

  NETWORKING=yes

  HOSTNAME=master.hadoop

  GATEWAY=192.168.58.2

- 然后修改/etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE="eth0"

#BOOTPROTO="dhcp"

BOOTPROTO="static"

IPADDR=192.168.58.130

NETMASK=255.255.255.0

HWADDR="00:0C:29:F4:76:5C"

IPV6INIT="no"

NM_CONTROLLED="yes"

ONBOOT="yes"

TYPE="Ethernet"

UUID="65718081-99bc-437f-81c3-24e34dc84d75"

DNS1=192.168.58.2

注意：这里 DNS1 是必须要设置的否则无法进行域名解析。

- 最后配置下/etc/resolv.conf：

  nameserver 192.168.129.2

  其实这一步可以省掉，上面设置了 DNS Server 的地址后系统会自动修改这个配置文件。

这样很简单几个步骤后虚拟机的 IP 就一直是 192.168.58.130 了。

重复以上步骤，将其他 3 个虚拟机的 IP 分别设置静态 IP 地址。

3.4 配置 hosts 文件

打开/etc/hosts 文件，添加集群中各个机器的 IP 地址与主机名映射信息，信息如下：

192.168.58.130 master.hadoop

192.168.58.131 slave1.hadoop

192.168.58.132 slave2.hadoop

192.168.58.133 slave3.hadoop

重复以上步骤，为其他 3 个虚拟机的 hosts 文件添加同样的配置；或者使用 scp 命令直接将 master 机器上的 hosts 文件复制到其他 3 个虚拟机中进行覆盖，命令如下：

 # scp –r /etc/hosts root@slave1.hadoop:/etc

 # scp –r /etc/hosts root@slave2.hadoop:/etc

 # scp –r /etc/hosts root@slave3.hadoop:/etc

3.5 安装配置 JDK 环境

在安装 JDK 之前，先确定系统中没有其他版本的 JDK，在终端窗口中输入 java –version 命令，如果查询出其他版本的 jdk，建议先删除。

3.5.1 安装 JDK

在 usr 目录中创建 java 目录，将 jdk1.6-linux.bin 拷贝到此目录下。

在终端执行以下命令进行安装。

 # chmod 755 jdk1.6-linux.bin

 # ./jdk1.6-linux.bin

安装完毕后，会在/usr/java 目录中生成 java 安装目录，将目录名称修改为 jdk1.6.0_13。

3.5.2 配置 JDK 环境

打开/etc/profile 文件，在下面追加以下信息。

 # java variable settings

export JAVA_HOME=/usr/java/jdk1.6.0_13

export JAVA_BIN=/usr/java/jdk1.6.0_13/bin

export PATH=$PATH:$JAVA_HOME/bin

export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

export JAVA_HOME JAVA_BIN PATH CLASSPATH

3.5.3 测试 JDK 环境

打开终端窗口，输入以下命令刷新环境变量信息

source /etc/profile

输入以下命令，测试 JDK 安装是否成功

java –version

重复以上步骤，在其他 3 个虚拟机中安装 JDK。对于在 profile 中添加 jdk 环境变量信息，可以使用 scp 命令直接将 master 机器上的 profile 文件复制到其他 3 个虚拟机中进行覆盖，命令如下：

 # scp –r /etc/profile root@slave1.hadoop:/etc

 # scp –r /etc/profile root@slave2.hadoop:/etc

 # scp –r /etc/profile root@slave3.hadoop:/etc

3.6 配置 ssh

使用 root 账户登录 master.hadoop 机器。

[root@master ~]# **ssh-keygen -t rsa -P ""**

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id_rsa):

Your identification has been saved in /root/.ssh/id_rsa.

Your public key has been saved in /root/.ssh/id_rsa.pub.

The key fingerprint is:

4f:25:be:15:5e:6c:b7:af:9c:97:29:dd:61:f8:d8:73 root@master.hadoop

The key's randomart image is:

```
+--[ RSA 2048]----+
|                 |
|                 .     |
|              . o + .|
|             . + + ..|
|           S o o .. |
|            o o . o.|
|             o     * *|
|                 + XE|
|                  =.o|
+-----------------+
```

[root@master ~]# **cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys**

[root@master ~]# **ssh-copy-id -i /root/.ssh/id_rsa.pub root@slave1.hadoop**

提示输入 slave1.hadoop 上的 root 用户密码，然后完成。

[root@master ~]# **ssh-copy-id -i /root/.ssh/id_rsa.pub root@slave2.hadoop**

提示输入 slave2.hadoop 上的 root 用户密码，然后完成。

[root@master ~]# **ssh-copy-id -i /root/.ssh/id_rsa.pub root@slave3.hadoop**

提示输入 slave3.hadoop 上的 root 用户密码，然后完成。

这样就能无密码登录，如下图所示：

【异常问题】

cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys 将公钥加到 authorized_keys 中后，使用命令

ssh slave1.hadoop 时可能会出现 Agent admitted failure to sign using the key 这样的错误信息。

解决办法：

使用 ssh-add 指令将私钥加进来（根据个人的密匙命名不同更改 id_rsa），

使用命令：

ssh-add /root/.ssh/id_rsa

一般来说这样就可以执行 ssh slave1.hadoop 命令了，不过我在经上面处理之后，得到的是如下的提示：

Could not open a connection to your authentication agent

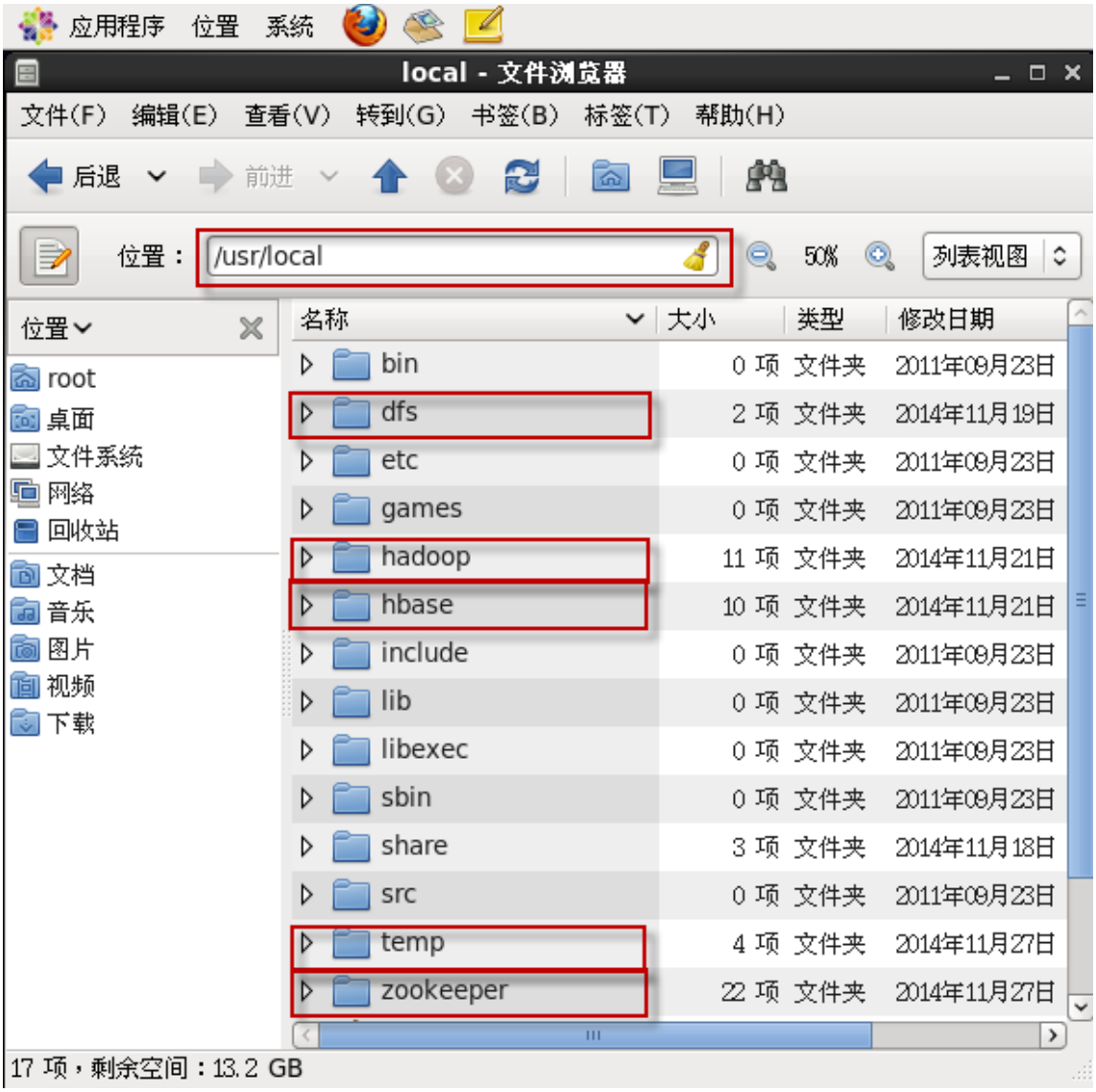
不急这个也是有解决办法的，即先在 shell 下执行 ssh-agent bash --login –i ，然后再执行 ssh-add 就好了！

通过以上设置，就可以从 master 无密码地访问 slave1、slave2 和 slave3 了，但是现在还无法从 slave1、slave2 和 slave3 无密码地访问 master，所以需要分别在 slave1、slave2 和 slave3 上执行上述步骤，达到 master←→slave1，master←→slave2，master←→slave3 双向无密码访问；slave1、slave2 和 slave3 三个 data 节点之间无需无密码访问，不需要进行上述步骤。

4. 安装 Hadoop

4.1 配置目录

在本次实验中，hadoop 和 hbase 以及 zookeeper 均安装在/usr/local 目录下，所以安装之前，需要将 hadoop-2.2.0.tar 和 hbase-0.96.2-hadoop2-bin.tar 以及 zookeeper-3.4.6.tar 分别拷贝到/usr/local 目录下，安装完成后，程序目录名称分别为 hadoop、hbase 以及 zookeeper。目录结构状况如下图所示：

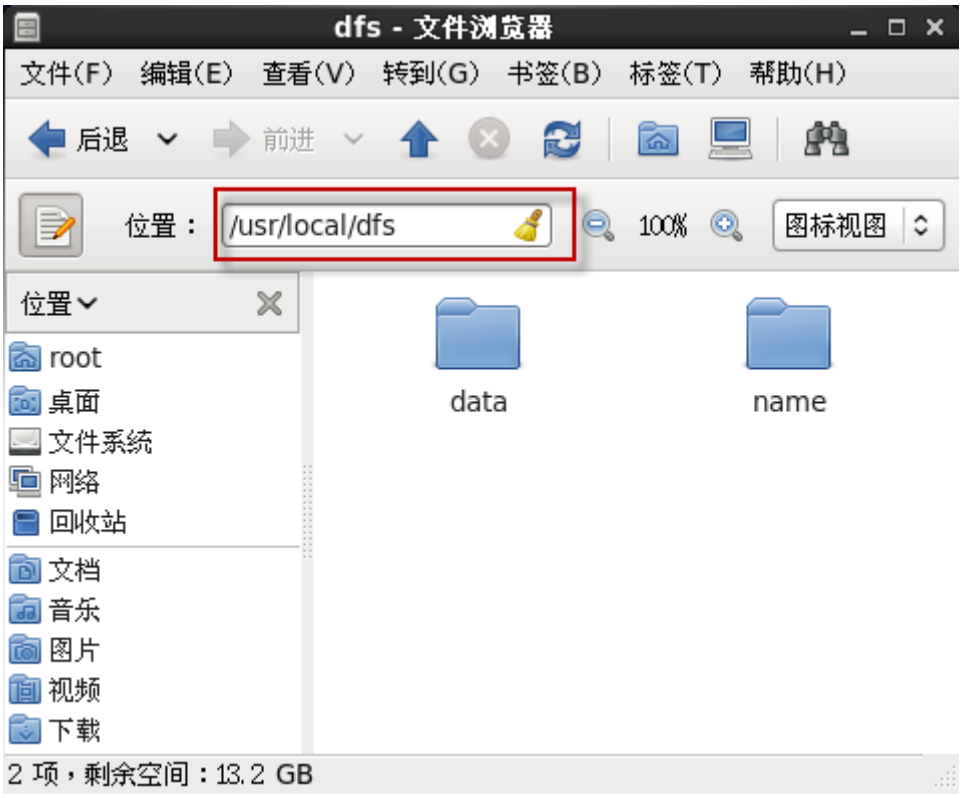


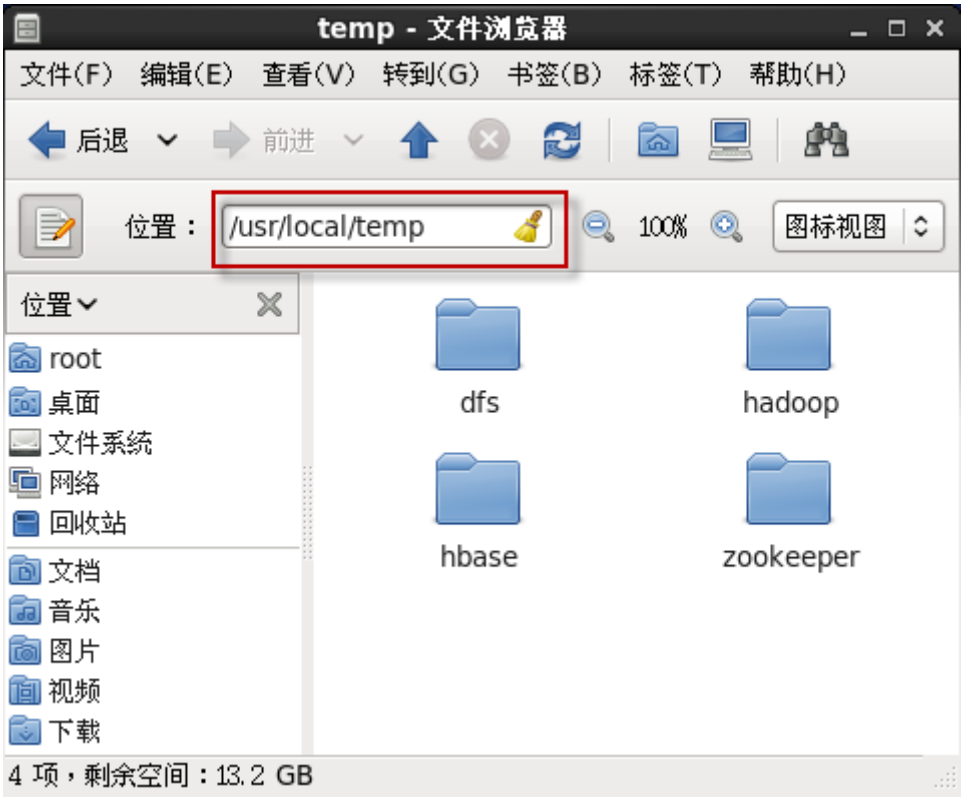
同时，创建以下目录结构。每个目录的用处，会在下面 hadoop、hbase 以及 zookeeper 安装配置过程中体现，这里不要着急。

目录名	描述
/usr/local/dfs	Hadoop 集群信息存储根目录。
/usr/local/dfs/data	存储数据节点信息。
/usr/local/dfs/name	存储 Name 节点信息。
/usr/local/temp	Hadoop 集群临时信息存储根目录。
/usr/local/temp/dfs	

/usr/local/temp/hbase	HBase 相关文件的临时目录
/usr/local/temp/hbase/pids	HBase 相关 pid 文件存放目录
/usr/local/temp/hadoop	Hadoop 相关文件的临时目录
/usr/local/temp/hadoop/pids	Hadoop 相关 pid 文件存放目录
/usr/local/temp/zookeeper	Zookeeper 相关文件的临时目录

目录结构状况如下图所示：





重复以上操作，在 master 和 3 个 slave 中分别配置目录。或者使用 scp 命令，将 master 节点上已经配置好的目录结构拷贝到其他三个 slave 节点上。命令如下。

【拷贝 dfs 目录】

```
[root@master ~]# scp –r /usr/local/dfs root@slave1.hadoop:/usr/local
[root@master ~]# scp –r /usr/local/dfs root@slave2.hadoop:/usr/local
[root@master ~]# scp –r /usr/local/dfs root@slave3.hadoop:/usr/local
```

【拷贝 temp 目录】

```
[root@master ~]# scp –r /usr/local/temp root@slave1.hadoop:/usr/local
[root@master ~]# scp –r /usr/local/temp root@slave2.hadoop:/usr/local
[root@master ~]# scp –r /usr/local/temp root@slave3.hadoop:/usr/local
```

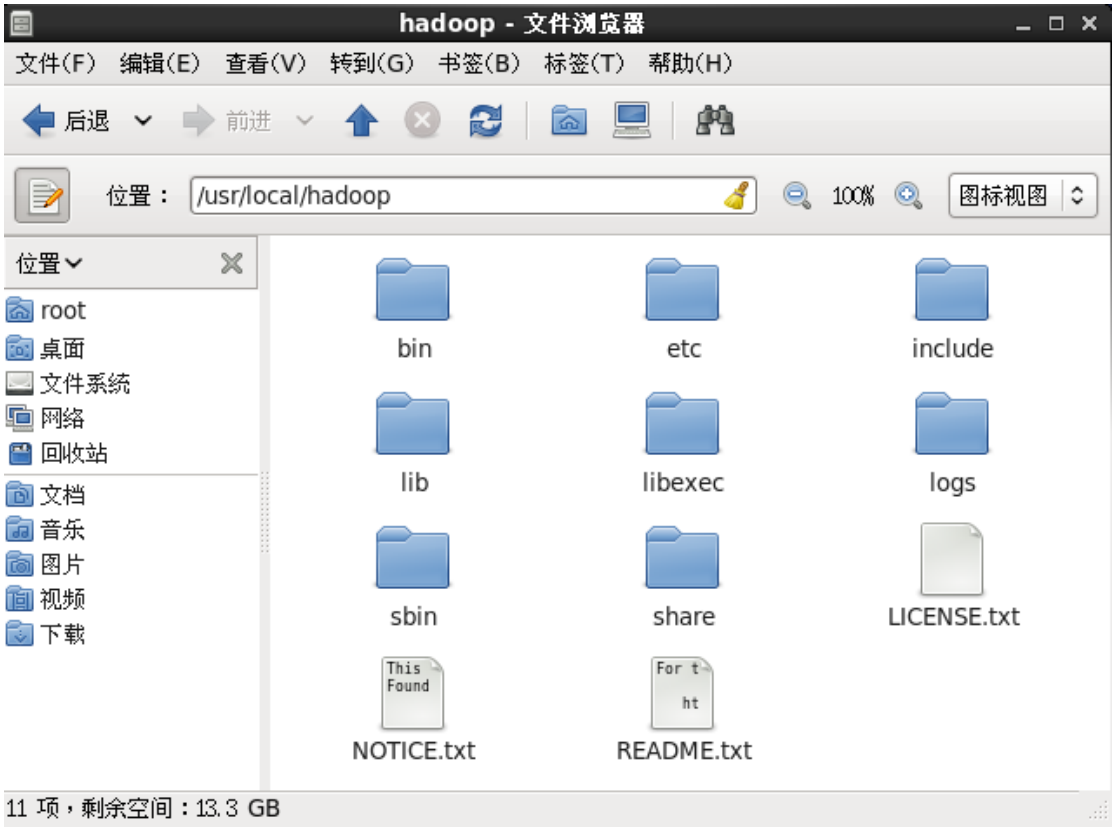
4.2 安装配置 Hadoop

4.2.1 安装 Hadoop

使用 root 账户登录 master.hadoop 机器。打开终端窗口，执行以下命令：

```
# cd /usr/local                    //进入local目录
# tar xzf hadoop-2.2.0.tar.gz      //解压缩hadoop安装程序
# mv hadoop-2.2.0 hadoop           //修改 hadoop 安装目录名称
```

安装后如下图所示：



4.2.2 配置 hadoop 环境变量

打开/etc/profile 文件，添加 hadoop 环境变量信息，信息如下：

```
# hadoop variable settings
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export
PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$HADOOP_HOME/lib
```

保存后，执行 source /etc/profile，刷新系统环境变量。

重复以上步骤，修改其他 3 个虚拟机的 profile 文件，或者使用 scp 命令直接将 master 节点上的 profile 文件复制到其他 3 个 slave 节点中进行覆盖，命令如下：

```
# scp –r /etc/profile   root@slave1.hadoop:/etc
# scp –r /etc/profile   root@slave2.hadoop:/etc
# scp –r /etc/profile   root@slave3.hadoop:/etc
```

4.2.3 配置 core-site.xml

打开/usr/local/hadoop/etc/hadoop/core-site.xml 文件。

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://master.hadoop:9000</value>
</property>
<property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/temp</value>
    <description>
        Abase for other temporary directories.
```

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

4.2.4 配置 hadoop-env.sh

打开/usr/local/hadoop/etc/hadoop/hadoop-env.sh 文件。

进行如下的修改：

```
export HADOOP_PID_DIR=/usr/local/temp/hadoop/pids
export JAVA_HOME=/usr/java/jdk1.6.0_13
```

4.2.5 配置 hdfs-site.xml

打开/usr/local/hadoop/etc/hadoop/hdfs-site.xml 文件。

```
<property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>master.hadoop:9001</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/dfs/name</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
```

```
    <value>/usr/local/dfs/data</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
<property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
```

4.2.6 配置 mapred-site.xml

打开/usr/local/hadoop/etc/hadoop/mapred-site.xml 文件。

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
        <name>mapreduce.job.tracker</name>
        <value>hdfs://master.hadoop:9101</value>
</property>
<property>
    <name>mapreduce.jobhistory.address</name>
    <value>master.hadoop:10020</value>
</property>
<property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>master.hadoop:19888</value>
</property>
```

4.2.7 配置 slaves

打开/usr/local/hadoop/etc/hadoop/slaves 文件。

添加 slave 信息。

```
slave1.hadoop
slave2.hadoop
slave3.hadoop
```

4.2.8 配置 yarn-env.sh

打开/usr/local/hadoop/etc/hadoop/yarn-env.sh 文件。

进行如下的修改：

```
export JAVA_HOME=/usr/java/jdk1.6.0_13
```

4.2.9 配置 yarn-site.xml

打开/usr/local/hadoop/etc/hadoop/yarn-site.xml 文件。

```
<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>
<property>
    <name>yarn.resourcemanager.address</name>
    <value>master.hadoop:8032</value>
</property>
```

```
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>master.hadoop:8030</value>
</property>
<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>master.hadoop:8031</value>
</property>
<property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>master.hadoop:8033</value>
</property>
<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>master.hadoop:8088</value>
</property>
```

4.3 部署其他 Slave 节点

Hadoop 在 master 机器上配置完毕后，通过 scp 命令将 hadoop 复制拷贝到其他几个 slave 中。命令如下：

```
# scp –r /usr/local/hadoop root@slave1.hadoop:/usr/local
# scp –r /usr/local/hadoop root@slave2.hadoop:/usr/local
# scp –r /usr/local/hadoop root@slave3.hadoop:/usr/local
```

4.4 启动关闭 Hadoop

在 hadoop 集群中，每个节点上的 hadoop 最终都必须启动起来，包括 master 节点和 3 个 slave 节点。启动顺序是先启动 master 上的 hadoop，然后按照顺序分别启动 slave1、slave2 和 slave3。

4.4.1 关闭防火墙

在每个节点启动 hadoop 之前，必须先关闭防火墙。命令如下：

#service iptables stop

4.4.2 格式化 HDFS 文件系统

在第一次启动hadoop之前，或者删除了hdfs文件配置信息，需要先格式化HDFS文件系统。打开终端窗口，进入/usr/local/hadoop/bin目录，执行下面的命令即可。

 # hadoop namenode –format

只需格式化一次，下次启动 hadoop 时不需要格式化。

4.4.3 启动 Hadoop

1. 启动 master 节点上的 hadoop

[root@master sbin]# start-all.sh

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

Starting namenodes on [master.hadoop]

master.hadoop: starting namenode, logging to

/usr/local/hadoop/logs/hadoop-root-namenode-master.hadoop.out

slave1.hadoop: starting datanode, logging to

/usr/local/hadoop/logs/hadoop-root-datanode-slave1.hadoop.out

slave3.hadoop: starting datanode, logging to

/usr/local/hadoop/logs/hadoop-root-datanode-slave3.hadoop.out

slave2.hadoop: starting datanode, logging to

/usr/local/hadoop/logs/hadoop-root-datanode-slave2.hadoop.out

Starting secondary namenodes [master.hadoop]

master.hadoop: starting secondarynamenode, logging to

/usr/local/hadoop/logs/hadoop-root-secondarynamenode-master.hadoop.out

starting yarn daemons

starting resourcemanager, logging to

/usr/local/hadoop/logs/yarn-root-resourcemanager-master.hadoop.out

slave2.hadoop: starting nodemanager, logging to

/usr/local/hadoop/logs/yarn-root-nodemanager-slave2.hadoop.out

slave3.hadoop: starting nodemanager, logging to

/usr/local/hadoop/logs/yarn-root-nodemanager-slave3.hadoop.out

slave1.hadoop: starting nodemanager, logging to

/usr/local/hadoop/logs/yarn-root-nodemanager-slave1.hadoop.out

【用 **jps** 命令检查验证】

[root@master sbin]# jps

5245 SecondaryNameNode

5070 NameNode

5632 Jps

5381 ResourceManager

出现上述结果，则表示启动正常。

2. 启动 slave 节点上的 hadoop

[root@slave1 sbin]# hadoop-daemon.sh start datanode

starting datanode, logging to

/usr/local/hadoop/logs/hadoop-root-datanode-slave1.hadoop.out

[root@slave1 sbin]# yarn-daemon.sh start nodemanager

starting nodemanager, logging to

/usr/local/hadoop/logs/yarn-root-nodemanager-slave1.hadoop.out

【用 jps 命令检查验证】

[root@slave1 sbin]# jps

5646 DataNode

5857 Jps

5744 NodeManager

出现上述结果，则表示启动正常。

4.4.4 关闭 Hadoop

1. **关闭 master 节点上的 hadoop**

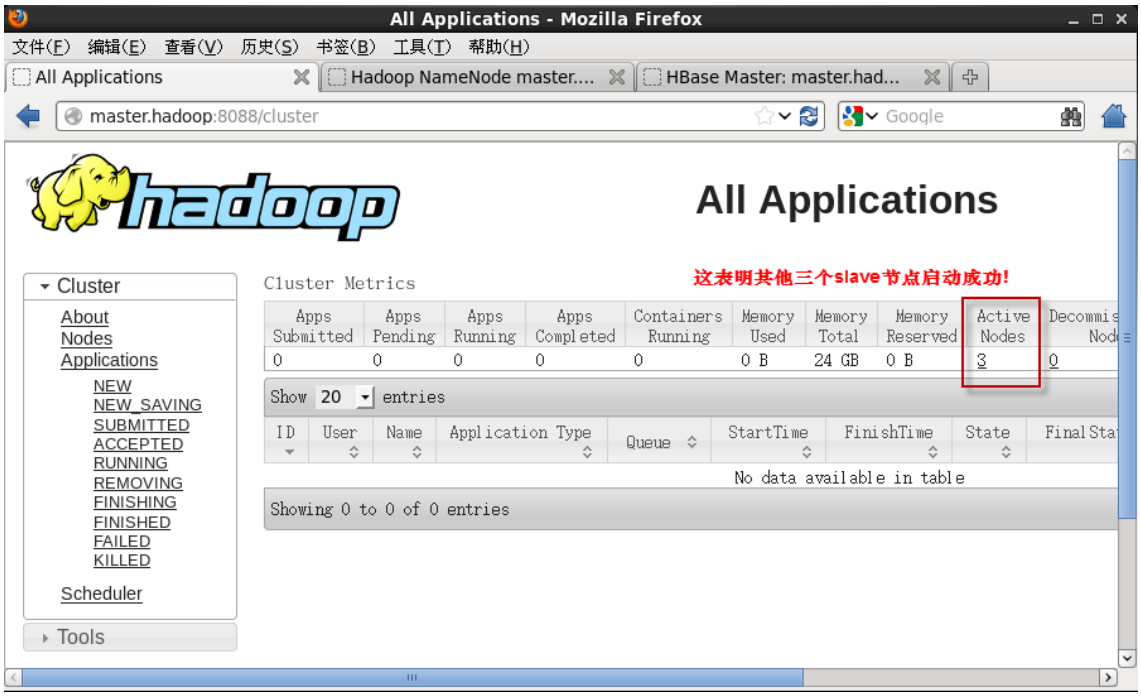
   [root@master sbin]# stop-all.sh

2. **关闭 slave 节点上的 hadoop**

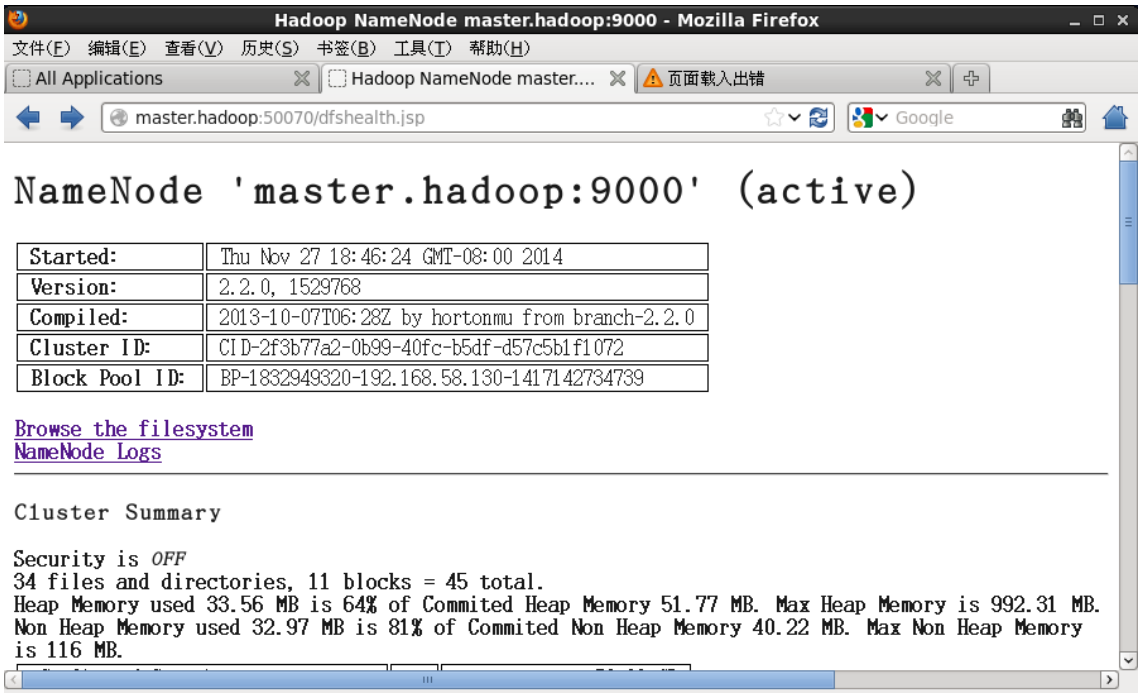
   [root@master sbin]# hadoop-daemon.sh stop datanode

4.4.5 测试 Hadoop

在浏览器中输入 http://master.hadoop:8088/cluster。



在浏览器中输入 http://master.hadoop:50070/dfshealth.jsp。



5. 安装 Zookeeper

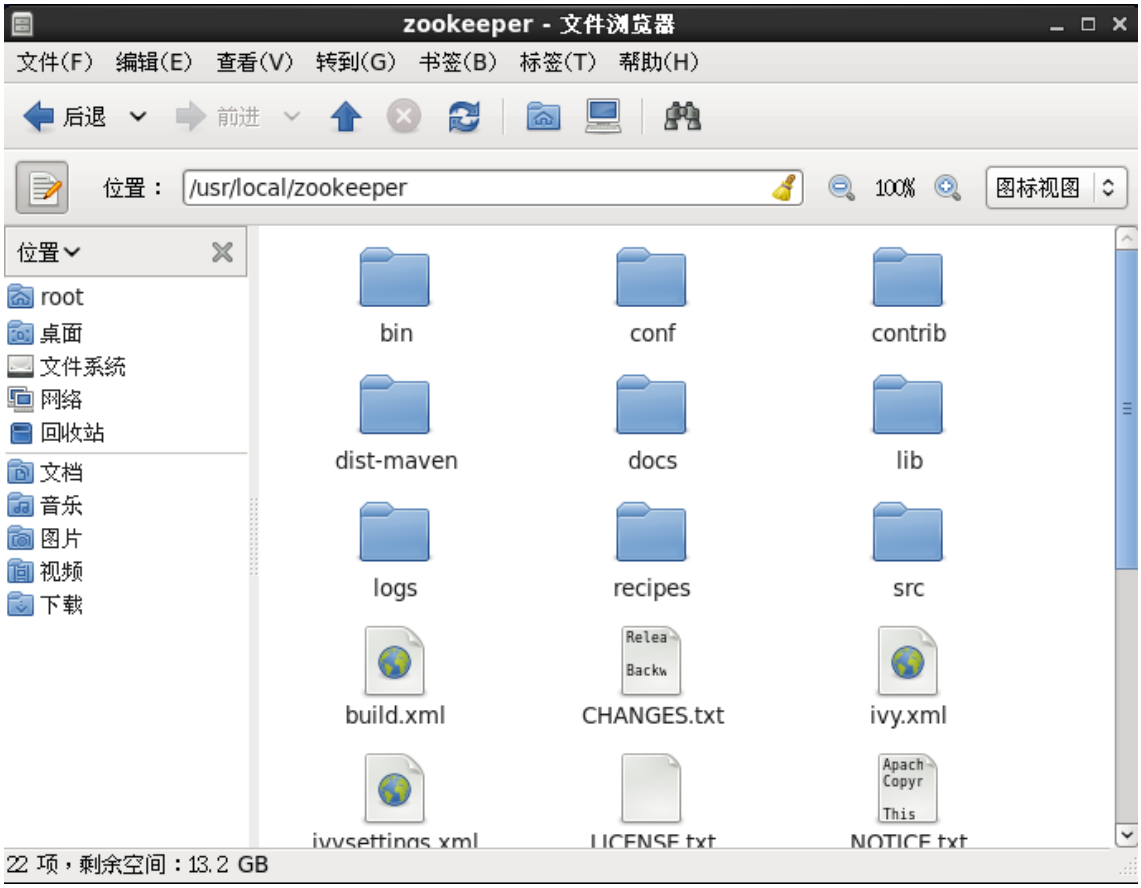
5.1 安装配置 Zookeeper

5.1.1 安装 Zookeeper

使用 root 账户登录 master.hadoop 机器。打开终端窗口，执行以下命令：

```
# cd /usr/local                          //进入local目录
# tar xzf zookeeper-3.4.6.tar.gz         //解压缩zookeeper安装程序
# mv zookeeper-3.4.6 zookeeper           //修改 zookeeper 安装目录名称
```

安装后如下图所示：



5.1.2 配置 Zookeeper 环境变量

打开/etc/profile 文件，添加修改 zookeeper 环境变量信息，信息如下：

```
# zookeeper variable settings
export ZOOKEEPER_HOME=/usr/local/zookeeper
export PATH=$PATH:$ZOOKEEPER_HOME/bin
```

保存后，执行 source /etc/profile，刷新系统环境变量。

重复以上步骤，修改其他 3 个虚拟机的 profile 文件，或者使用 scp 命令直接将 master 节点上的 profile 文件复制到其他 3 个 slave 节点中进行覆盖，命令如下：

```
# scp –r /etc/profile   root@slave1.hadoop:/etc
# scp –r /etc/profile   root@slave2.hadoop:/etc
```

 # scp –r /etc/profile root@slave3.hadoop:/etc

5.1.3 配置 zoo.cfg

进入 zookeeper 安装目录下的 conf 目录，复制 zoo_sample.cfg 文件，将其改名为 zoo.cfg，打开此文件，添加修改如下信息。

tickTime=2000

initLimit=5

syncLimit=2

dataDir=/usr/local/temp/zookeeper

dataLogDir=/usr/local/zookeeper/logs

clientPort=2181

server.1=master.hadoop:2888:3888

server.2=slave1.hadoop:2888:3888

server.3=slave2.hadoop:2888:3888

server.4=slave3.hadoop:2888:3888

5.1.4 配置 myid 文件

请看上面我们配置的 zoo.cfg 文件，我们将 zk 的 dataDir 配置为 /usr/local/temp/zookeeper 目录。再此目录中创建一个文件，命名为 myid，内容为 server 的序号。譬如在上面我们配置 master 为 server.1,则 myid 内容即为 1，slave1.hadoop 配置为 server.2,则 slave2.hadoop 中的 myid 内容即为 2，依此类推，配置其他几个 slave 节点的 myid。

5.2 部署其他 Slave 节点

Zookeeper 在 master 机器上配置完毕后，通过 scp 命令将 zookeeper 复制拷贝到其他几个 slave 中。命令如下：

 # scp –r /usr/local/zookeeper root@slave1.hadoop:/usr/local

 # scp –r /usr/local/zookeeper root@slave2.hadoop:/usr/local

 # scp –r /usr/local/zookeeper root@slave3.hadoop:/usr/local

5.3 启动关闭 Zookeeper

5.3.1 启动 zookeeper

启动 master 节点上的 zookeeper 和 slave 节点的 zookeeper 的命令式一样的。命令如下。

[root@master bin]# zkServer.sh start

JMX enabled by default

Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

[root@master bin]# jps

5245 SecondaryNameNode

5070 NameNode

6043 Jps

5381 ResourceManager

6007 QuorumPeerMain

5.3.2 关闭 zookeeper

关闭 master 节点上的 zookeeper 和 slave 节点的 zookeeper 的命令式一样的。命令如下。

[root@master bin]# zkServer.sh stop

6. 安装 HBase

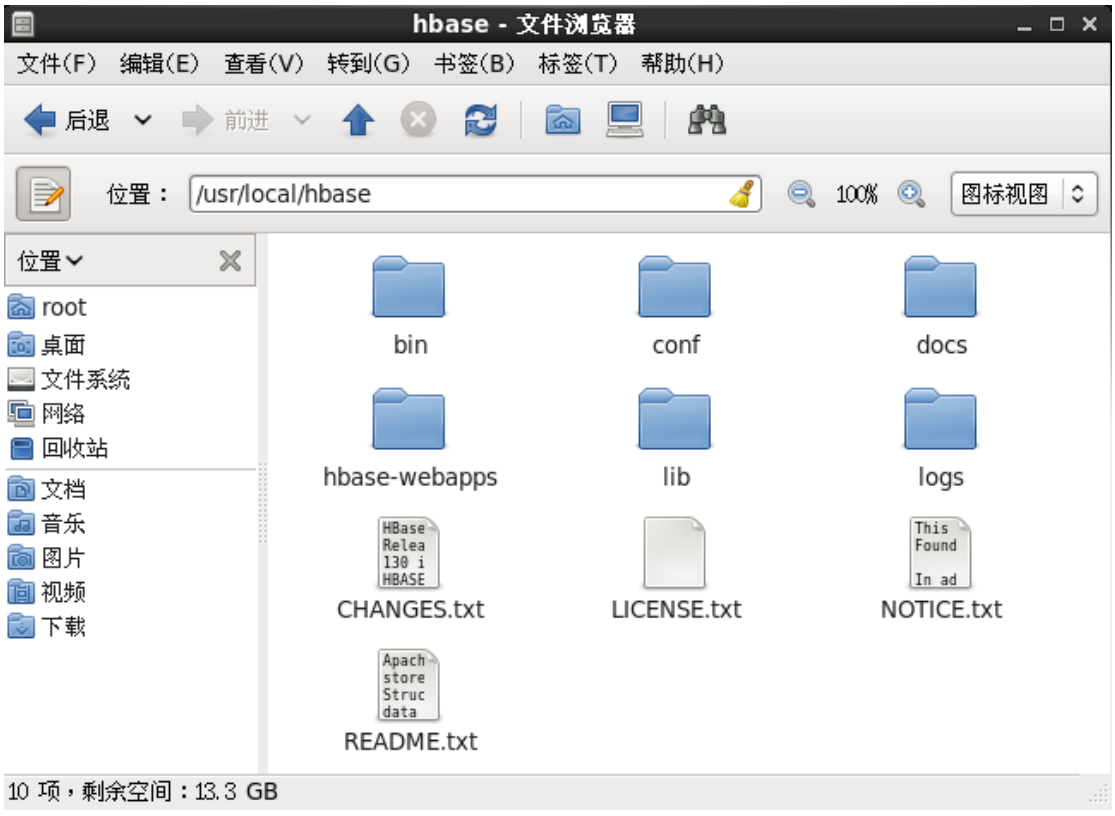
6.1 安装配置 HBase

6.1.1 安装 HBase

使用 root 账户登录 master.hadoop 机器。打开终端窗口，执行以下命令：

```
# cd /usr/local                          //进入local目录
# tar xzf hbase-0.96.2-hadoop2-bin.tar.gz    //解压缩hbase安装程序
# mv hbase-0.96.2-hadoop2 hbase          //修改 hbase 安装目录名称
```

安装后如下图所示：



6.1.2 配置 HBase 环境变量

打开/etc/profile 文件，添加 HBase 环境变量信息，信息如下：

```
# hbase variable setting
export HBASE_HOME=/usr/local/hbase
```

export PATH=$PATH:$HBASE_HOME/bin

保存后，执行# source /etc/profile，刷新系统环境变量。

重复上述操作，为其他几个 slave 机进行同样的配置；或者使用 scp 命令将 profile 文件直接复制到其他几个 slave 节点中进行覆盖操作，命令如下：

#scp -r /etc/profile root@slave1.hadoop:/etc

#scp -r /etc/profile root@slave2.hadoop:/etc

#scp -r /etc/profile root@slave3.hadoop:/etc

6.1.3 配置 hbase-env.sh

打开文件/usr/local/hbase/conf/hbase-env.sh，添加修改如下信息。

#Java 安装位置

export JAVA_HOME=/usr/java/jdk1.6.0_13

#采用独立的 ZooKeeper

export HBASE_MANAGES_ZK=false

#HBase 类路径

export HBASE_CLASSPATH=/usr/local/hadoop/etc/hadoop

#hbase pid config

export HBASE_PID_DIR=/usr/local/temp/hbase/pids

一个分布式运行的 Hbase 依赖一个 zookeeper 集群。所有的节点和客户端都必须能够访问 zookeeper。默认的情况下 Hbase 会管理 一个 zookeep 集群。这个集群会随着 Hbase 的启动而启动。当然，你也可以自己管理一个 zookeeper 集群，但需要配置 Hbase。你需要修改 conf/hbase-env.sh 里面的 HBASE_MANAGES_ZK 来切换。这个值默认是 true 的，作用是让 Hbase 启动的时候同时也启动 zookeeper。在本次实验中，我们采用独立的 zookeeper 来管理 hbase。

6.1.4 配置 hbase-site.xml

```
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://master.hadoop:9000/hbase</value>
</property>
<property>
    <name>hbase.master</name>
    <value>hdfs://master.hadoop:60000</value>
</property>
<property>
    <name>hbase.master.port</name>
    <value>60000</value>
</property>
<property>
    <name>hbase.master.maxclockskew</name>
    <value>180000</value>
</property>
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
<property>
    <name>hbase.zookeeper.quorum</name>
    <value>slave1.hadoop,slave2.hadoop,slave3.hadoop</value>
</property>
<property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2181</value>
    <description>Property from ZooKeeper's config zoo.cfg.The port at which the
```

```
clients will connect.
        </description>
    </property>
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>/usr/local/temp/zookeeper</value>
    </property>
    <property>
        <name>hbase.tmp.dir</name>
        <value>/usr/local/temp/hbase</value>
    </property>
    <property>
        <name>zookeeper.znode.parent</name>
        <value>/hbase</value>
    </property>
    <property>
        <name>zookeeper.session.timeout</name>
        <value>90000</value>
    </property>
    <property>
        <name>hbase.regionserver.restart.on.zk.expire</name>
        <value>true</value>
    </property>
    <property>
        <name>hbase.master.info.port</name>
        <value>60010</value>
    </property>
    <property>
        <name>hbase.regionserver.info.port</name>
```

```
        <value>60030</value>
    </property>
    <property>
        <name>hbase.client.scanner.caching</name>
        <value>200</value>
    </property>
    <property>
        <name>hbase.balancer.period</name>
        <value>300000</value>
    </property>
    <property>
        <name>hbase.client.write.buffer</name>
        <value>10485760</value>
    </property>
    <property>
        <name>hbase.hregion.majorcompaction</name>
        <value>7200000</value>
    </property>
    <property>
        <name>hbase.hregion.max.filesize</name>
        <value>67108864</value>
    </property>
    <property>
        <name>hbase.hregion.memstore.flush.size</name>
        <value>1048576</value>
    </property>
    <property>
        <name>hbase.server.thread.wakefrequency</name>
        <value>30000</value>
```

</property>

- hbase.rootdir

这个目录是 region server 的共享目录，用来持久化 Hbase。URL 需要是完全正确的，还要包含文件系统的 scheme。例如，要表示 hdfs 中的'/hbase'目录，namenode 运行在 node1 的 49002 端口。则需要设置为 hdfs://node1:49002/hbase。默认情况下 Hbase 是写到/tmp 的。不改这个配 置，数据会在重启的时候丢失。默认： file:///tmp/hbase-${user.name}/hbase

- hbase.cluster.distributed

Hbase 的运行模式。false 是单机模式，true 是分布式模式。若为 false,Hbase 和 Zookeeper 会运行在同一个 JVM 里面。默认： false

- 在 hbase-site.xml 配置 zookeeper：

当 Hbase 管理 zookeeper 的时候，你可以通过修改 zoo.cfg 来配置 zookeeper，一个更加简单的方法是在 conf/hbase-site.xml 里面修改 zookeeper 的配置。Zookeeer 的配置是作为 property 写在 hbase-site.xml 里面的。

对于 zookeepr 的配置，你至少要在 hbase-site.xml 中列出 zookeepr 的 ensemble servers，具体的字段是 hbase.zookeeper.quorum.该这个字段的默认值是 localhost，这个值对于分布式应用显然是不可以的.（远程连接无法使用）。

- hbase.zookeeper.property.clientPort

ZooKeeper 的 zoo.conf 中的配置。 客户端连接的端口。

- hbase.zookeeper.quorum

Zookeeper 集群的地址列表，用逗号分割。

例 如：

"host1.mydomain.com,host2.mydomain.com,host3.mydomain.com". 默 认 是 localhost,是给伪分布式用的。要修改才能在完全分布式的情况下使用。如果在 hbase-env.sh 设置了 HBASE_MANAGES_ZK， 这些 ZooKeeper 节点就会和 Hbase 一起启动。默认： localhost

运行一个 zookeeper 也是可以的，但是在生产环境中，你最好部署 3，5，7 个节点。部署的越多，可靠性就 越高，当然只能部署奇数个，偶数个是不可以的。你需要给每个 zookeeper 1G 左右的内存，如果可能的话，最好有独立的磁盘。（独

立磁盘可以确保 zookeeper 是高性能的。).如果你的集群负载很重，不要把 Zookeeper 和 RegionServer 运行在同一台机器上面。就像 DataNodes 和 TaskTrackers 一样

6.1.5 配置 regionservers

打开文件/usr/local/hbase/conf/regionservers，将作为 data 节点的 slave 主机名称分行输入进去并保存。

slave1.hadoop

slave2.hadoop

slave3.hadoop

6.2 部署其他 Slave 节点

HBase 配置完毕后，通过 scp 命令将 hbase 复制拷贝到其他几个 slave 中。

 # scp –r /usr/local/hbase root@slave1.hadoop:/usr/local

 # scp –r /usr/local/hbase root@slave2.hadoop:/usr/local

 # scp –r /usr/local/hbase root@slave3.hadoop:/usr/local

6.3 启动关闭 HBase

6.3.1 启动 HBase

1. 启动 master 节点上的 hbase

[root@master bin]# start-hbase.sh

starting master, logging to

/usr/local/hbase/logs/hbase-root-master-master.hadoop.out

slave3.hadoop: starting regionserver, logging to

/usr/local/hbase/bin/../logs/hbase-root-regionserver-slave3.hadoop.out

slave1.hadoop: starting regionserver, logging to

/usr/local/hbase/bin/../logs/hbase-root-regionserver-slave1.hadoop.out

slave2.hadoop: starting regionserver, logging to

/usr/local/hbase/bin/../logs/hbase-root-regionserver-slave2.hadoop.out

[root@master bin]# jps

6229 HMaster

5245 SecondaryNameNode

5070 NameNode

6442 Jps

5381 ResourceManager

6007 QuorumPeerMain

2. 启动 slave 节点上的 hbase

[root@slave1 bin]# hbase-daemon.sh start regionserver

regionserver running as process 6024. Stop it first.

[root@slave1 bin]# jps

5646 DataNode

5917 QuorumPeerMain

6186 Jps

5744 NodeManager

6024 HRegionServer

你现在已经启动 Hbase 了。Hbase 把 log 记在 logs 子目录里面. 当 Hbase 启动出问题的时候，可以看看 Log。

Hbase 也有一个界面，上面会列出重要的属性。默认是在 Master 的 60010 端口上 (HBase RegionServers 会默认绑定 60020 端口，在端口 60030 上有一个展示信息的界面)。如果 Master 运行在 node1，端口是默认的话，你可以用浏览器在 http://master.hadoop:60010 看到主界面。

一旦 Hbase 启动，可以看到如何建表，插入数据，scan 你的表，还有 disable 这个表，最后把它删掉。

6.3.2 关闭 HBase

1. 关闭 master 节点上的 hbase

[root@master bin]# stop-hbase.sh

2. 关闭 slave 节点上的 hbase

[root@slave1 bin]# hbase-daemon.sh stop regionserver

6.3.3 测试 HBase

1. 用 shell 连接你的 Hbase

 # ./bin/hbase shell

```
HBase Shell; enter'help<RETURN>' for list of supported commands.
Type"exit<RETURN>" to leave the HBase Shell
Version: 0.90.0, r1001068, FriSep 24 13:55:42 PDT 2010

hbase(main):001:0>
```

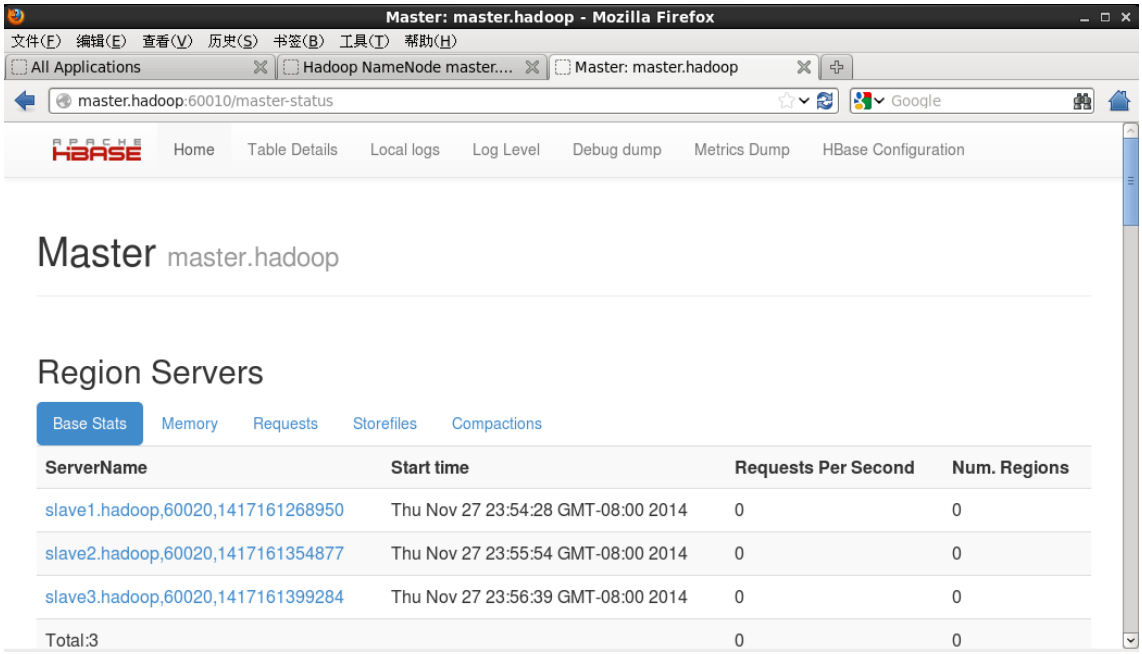
输入 help 然后 <RETURN> 可以看到一列 shell 命令。这里的帮助很详细，要注意的是表名，行和列需要加引号。

创建一个名为 test 的表，这个表只有一个 column family 为 cf。可以列出所有的表来检查创建情况，然后插入些值。

```
hbase(main):003:0> create 'myhbase_table','cf'
0 row(s) in 1.2200 seconds
hbase(main):003:0> list'table'
myhbase_table
1 row(s) in 0.0550 seconds
hbase(main):004:0> put 'myhbase_table','row1', 'cf:a', 'value1'
0 row(s) in 0.0560 seconds
hbase(main):005:0> put 'myhbase_table','row2', 'cf:b', 'value2'
0 row(s) in 0.0370 seconds
hbase(main):006:0> put 'myhbase_table','row3', 'cf:c', 'value3'
0 row(s) in 0.0450 seconds
.....
```

2. 打开 hbase web 界面

在浏览器中输入 http://master.hadoop:60010/master-status。



7. 疑难杂症

7.1 hbase.zookeeper.quorum 必须为奇数

为什么“hbase.zookeeper.quorum”必须配奇数个数的 DataNode?

zookeeper 有这样一个特性：集群中只要有过半的机器是正常工作的，那么整个集群对外就是可用的。也就是说如果有 2 个 zookeeper，那么只要有 1 个死了 zookeeper 就不能用了，因为 1 没有过半，所以 2 个 zookeeper 的死亡容忍度为 0；同理，要是有 3 个 zookeeper，一个死了，还剩下 2 个正常的，过半了，所以 3 个 zookeeper 的容忍度为 1；同理你多列举几个：2->0;3->1;4->1;5->2;6->2 会发现一个规律，2n 和 2n-1 的容忍度是一样的，都是 n-1。

7.2 NoRouteToHostException: No route to host

Hadoop 报错：NoRouteToHostException: No route to host

当用户上传数据到 HDFS 上时经常会出现这个错误

hdfs.DFSClient:Exception in createBlockOutputStream

java.net.NoRouteToHostException:No route to host

这种情况网上的解决办法 通常是告诉要关闭防火墙，至于 关于哪台主机的防火墙并没提。

查看日志文件，只说是 No route to host 没有提端口的事，解决方案是所有主机的关闭防火墙，namenode 和 slave 节点都要关闭。如果 只关闭 namenode 所在主机的防火墙问题依旧如此。

#service iptables stop

或

#chkconfig iptables on/off

7.3 集群时钟一致性问题

集群的时钟要保证基本的一致，看每台机器的时间是否一样。

如果你查询的时候或者是遇到奇怪的故障，可以检查一下系统时间是否正确!

设置集群各个节点时钟：date -s “2014-04-18 15:00:00”

7.4 pid 不存在的问题

hadoop 停止集群时，报错如下所示：

```
no namenode to stop
no datanode to stop
no secondery namenode to stop
no resourcemanager to stop
no nodemanager to stop
```

造成 hadoop 出现上述这个错误的原因很多，如果系统运行一切正常并且运行了很长时间，现在需要停止集群出现了上述错误，那么一个很可能的原因是 hadoop 的 pid 文件丢失，hadoop 的 pid 文件默认保存在/tmp 目录下，/tmp 目录下的文件很容易丢失，所以造成停止集群的时候出现上述错误。解决方式是在/etc/hadoop/hadoop-env.conf 文件中找到 pid 的配置项，修改其配置路径即可。

hbase 停止集群时,报错如下：

stopping hbasecat: /tmp/hbase-root-master.pid: No such file or directory

造成上述错误的原因是，默认情况下 hbase 的 pid 文件保存在/tmp 目录下，/tmp 目录下的文件很容易丢失，所以造成停止集群的时候出现上述错误。解决方式是在 hbase-env.sh 中修改 pid 文件的存放路径。

7.5 Hadoop 配置选项信息一览

hadoop 的集群配置中，大部分都用的默认设置，如果想要提高整个集群的性能，可以考虑通过修改配置的方法实现，配置项大部分都配置在这三个文件里：core_site.xml, hdfs_site.xml, mapred_site.xml, 下面将经常会用到的配置项总结如下：（待补充）

7.5.1 core_site.xml

fs.trash.interval

说明：hadoop 垃圾回收机制，每隔多长时间清理一次垃圾

value：1440

备注：默认值是 0，不打开垃圾收回机制。删除掉的文件将直接清除，不保存。如果设置了，则将保存到本地的.crash 文件夹下。

fs.checkpoint.dir

说明：本地文件系统 DFS secondaryname 节点存储临时图像目录

value：/disk2/cloudera/hadoop/dfs/namesecondary

备注：用于：hadoop namenode -importCheckpoint，从检查点目录装载镜像并保存到当前检查点目录，检查点目录由 fs.checkpoint.dir 指定。

hadoop.tmp.dir

说明：Hadoop 的默认临时文件存放路径

value：/home/hadoop/hadoop/tmp

备注：这个最好配置，如果在新增节点或者其他情况下莫名其妙的 DataNode 启动不了，就删除此文件中的 tmp 目录即可。

不过如果删除了 NameNode 机器的此目录，那么就需要重新执行 NameNode 格式化的命令

io.file.buffer.size

说明：读写序列文件缓冲区大小

value：16384

备注：值应该设置为硬件页面大小的倍数，默认设置为 4096，请设置为 4096 的倍数

io.bytes.per.checksum

说明：

value：

备注：

io.skip.checksum.errors

说明：

value：

备注：

io.compression.codecs

说明：

value：

备注：

io.serializations

说明：

value：

备注：

7.5.2 hdfs_site.xml

dfs.name.dir

说明：设定 DFS Name 节点中的命名空间表格文件，在本地系统中的保存位置。 可以设置多个，通过"，"分隔，fsimage 文件，会被复制到多个路径中，用于备份。

value：/home/hadoop/hadoop/name

备注：其中的一个路径，可以利用 nfs 映射到我们的共享文件夹下，形成异地备份

dfs.data.dir

说明：设定 DFS Data 节点中的数据块在本地系统中的保存位置。可以设置多个，通过 "，" 分隔。设置的文件夹如果不存在，将被忽略。

value：/home/hadoop/data1, /home/hadoop/data2

备注：

dfs.replication

说明：缺省的文件块被复制的次数。在文件被创建的时候可以指定复制的块数，如果在创建的时候没有指定，则使用该缺省值。

value：3

备注：

dfs.block.size

说明：新文件被分隔的缺省块容量。

value：268435456

备注：必须是 512 的倍数

dfs.datanode.max.xcievers

说明：datanode 所允许同时执行的发送和接受任务的数量

value：10240

备注：默认是 256.该值太小。可以修改为 10240

dfs.web.ugi

说明：hadoop 的 web 界面访问权限设置。

value：hadoop,hadoop

备注：value 语法：用户名，用户组

如何设置：

hadoop fs -chmod （修改文件所有者，文件所属组，其他用户的读、写、执行权限）

haddop fs -chown （修改文件所有者）

hadoop fs -chgrp （修改文件所属组）

dfs.permissions

说明：对 HDFS 是否启用认证。

value：true

备注：默认为 true

dfs.permissions.supergroup

说明：超级用户组名称

value：supergroup

备注：

dfs.safemode.threshold.pct

说明：启动的时候，NameNode 会等待所有的 datanode 报告 block 状态，查看所有的 block 的副本是否达到最低要求，当报告合格的数量达到设置的值，则退出 safemode。

value：0.95f

备注：默认为 0.999f

dfs.safemode.extension

说明：Namenode 在合格的 datanode 数目达到要求的时候，并不是马上离开 safemode 状态，会有一个扩展时间，让剩余的 datanode 来报告 block 信息，这个扩展时间默认是 30 秒，单位是毫秒。

value：30000

备注：时间默认是 30 秒，单位是毫秒

dfs.balance.bandwidthPerSec

说明：用于平衡数据。每秒平衡数据量最大带宽。可以放大。

value：10485760

备注：默认是 10485760

dfs.replication.min

说明：创建文件时的最小复制数。主要用于 dfs.safemode.threshold.pct

value：1

备注：默认为 1

dfs.datanode.handler.count

说明：datanode 上用于处理 RPC 的线程数。

value：3

备注：默认为 3，较大集群，可适当调大些，比如 8。需要注意的是，每添加一个线程，需要的内存增加。

dfs.datanode.du.reserved

说明：表示在 datanode 对磁盘写时候,保留多少非 dfs 的磁盘空间，从而避免 dfs 将所在的磁盘写满

value：10737418240

备注：默认为 0，单位是字节

dfs.df.interval

说明：磁盘使用统计刷新时间间隔，单位毫秒

value：60000

备注：默认是 60000，单位是毫秒

dfs.namenode.handler.count

说明：namenode 的 rpc 调用线程数。

value：10

备注：默认是 10

dfs.namenode.plugins

说明：

value：

备注：

dfs.datanode.plugins

说明：

value：

备注：

dfs.thrift.address

说明：

value：

备注：

dfs.hosts/dfs.hosts.exclude

说明：Data Node 白名单/黑名单文件

value：

备注：

7.5.3 mapred_site.xml

mapred.job.tracker

说明：JobTracker 的地址

value：

备注：格式为 hostname:port

mapred.local.dir

说明：运行 mapreduce 中间结果存储处，保存 MapReduce 临时文件的本地目录

value：

备注：可设置多个，用逗号分隔

mapred.system.dir

说明：HDFS 上 MapReduce 保存系统文件的目录

value：

备注：

mapred.job.tracker.handler.count

说明： jobtracker 同时与 tasktracker 通信的线程数

value：

备注：

mapreduce.jobtracker.staging.root.dir

说明：

value：

备注：

mapred.temp.dir

说明：

value：

备注：

mapred.child.java.opts

说明：设置 JVM 堆的最大可用内存，需从应用程序角度进行配置。

value：

备注：

mapred.tasktracker.map.tasks.maximum

说明：tasktracker 上同时运行的 map 的最大数量

value：2

备注：默认为 2

mapred.tasktracker.reduce.tasks.maximum

说明：tasktracker 上同时运行的 task 的最大数量

value：2

备注：默认为 2

mapred.hosts/mapred.host.exclude

说明：MapReduce 白名单/黑名单文件

value：

备注：

mapred.queue.names

说明： 队列名

value：

备注： hadoop MapReduce 系统默认有一个"default"的 Job 队列(pool).

mapred.map.tasks.speculative.execution

说明：

value：

备注：

mapred.reduce.tasks.speculative.execution

说明：

value：

备注：

io.sort.mb

说明：排序使用的最大内存

value：

备注：

io.sort.factor

说明： 排序因子。同时合并的数据流的数量

value：

备注：当一个 map task 执行完之后，本地磁盘上(mapred.local.dir)有若干个 spill 文件，merge sort 把这些文件合成一个。执行 merge sort 的时候，每次同时打开多少个 spill 文件由该参数决定。 打开的文件越多，不一定 merge sort 就越快，所以要根据数据情况适当的调整。

keep.failed.task.files

说明：

value：

备注：

mapred.job.reuse.jvm.num.tasks

说明：

value：

备注：

mapred.child.env

说明：

value：

备注：

mapred.child.ulimit

说明：

value：

备注：

mapred.output.compress/mapred.compress.map.output

说明：中间结果和最终结果是否要进行压缩，如果是，指定压缩方式（Mapred.compress.map.output.codec/ Mapred.output.compress.codec）。 推荐使用 LZO 压缩。Intel 内部测试表明，相比未压缩，使用 LZO 压缩的 TeraSort 作业，运行时间减少 60%，且明显快于 Zlib 压缩。

value

备注：

tasktracker.http.threads

说明:HTTP server 上的线程数。运行在每个 TaskTracker 上， 用于处理 map task 输出。

value：8

备注：tasktracker 开 http 服务的线程数。用于 reduce 拉取 map 输出数据，大集群可以将其设为 40~50。

fs.inmemory.size.mb

说明：reducer 在合并 map 输出数据使用的内存空间

value：

备注： 默认使用 200M

mapred.reduce.parallel.copies

说明：reducer 同时从 mapper 上拉取的文件数

value：

备注：

mapred.output.compression.codec

说明：

value：

备注：

mapred.map.output.compression.codec

说明：

value：

备注：

jobtracker.thrift.address

说明：

value：

备注：

mapred.jobtracker.plugins

说明：

value：

备注：