

Hadoop 集群（第 5 期）

——Hadoop 安装配置

1、集群部署介绍

1.1 Hadoop 简介

Hadoop 是 Apache 软件基金会旗下的一个开源分布式计算平台。以 Hadoop 分布式文件系统 (HDFS, Hadoop Distributed Filesystem) 和 MapReduce (Google MapReduce 的开源实现) 为**核心**的 Hadoop 为用户提供了系统底层细节透明的分布式基础架构。



对于 Hadoop 的集群来讲，可以分成两大类角色：Master 和 Slave。一个 **HDFS** 集群是由一个 NameNode 和若干个 DataNode 组成的。其中 NameNode 作为主服务器，管理文件系统的命名空间和客户端对文件系统的访问操作；集群中的 DataNode 管理存储的数据。**MapReduce** 框架是由一个单独运行在主节点上的 JobTracker 和运行在每个集群从节点的 TaskTracker 共同组成的。主节点负责调度构成一个作业的所有任务，这些任务分布在不同的从节点上。主节点监控它们的执行情况，并且重新执行之前的失败任务；从节点仅负责由主节点指派的任务。当一个 Job 被提交时，JobTracker 接收到提交作业和配置信息之后，就会将配置信息等分发给从节点，同时调度任务并监控 TaskTracker 的执行。

从上面的介绍可以看出，HDFS 和 MapReduce 共同组成了 Hadoop 分布式系统体系结构的核心。**HDFS** 在集群上实现**分布式文件系统**，**MapReduce** 在集群上实现了**分布式计算和任务处理**。HDFS 在 MapReduce 任务处理过程中提供了文件操作和存储等支持，MapReduce 在 HDFS 的基础上实现了任务的分发、跟踪、执行等工作，并收集结果，二者相互作用，完成了 Hadoop 分布式集群的主要任务。

1.2 环境说明

集群中包括 4 个节点：1 个 Master，3 个 Slave，节点之间局域网连接，可以相互 ping 通，具体集群信息可以查看“**Hadoop 集群（第 2 期）**”。节点 IP 地址分布如下：

机器名称	IP 地址
Master.Hadoop	192.168.1.2
Slave1.Hadoop	192.168.1.3
Slave2.Hadoop	192.168.1.4
Slave3.Hadoop	192.168.1.5

四个节点上均是 CentOS6.0 系统，并且有一个相同的用户 **hadoop**。Master 机器主要配置 NameNode 和 JobTracker 的角色，负责总管分布式数据和分解任务的执行；3 个 Slave 机器配置 DataNode 和 TaskTracker 的角色，负责分布式数据存储以及任务的执行。其实应该还有 1 个 Master 机器，用来作为**备用**，以防止 Master 服务器**宕机**，还有一个备用马上启

用。后续经验积累一定阶段后**补上**一台备用 Master 机器。

1.3 网络配置

Hadoop 集群要按照 1.2 小节表格所示进行配置，我们在“**Hadoop 集群（第 1 期）**”的 CentOS6.0 安装过程就按照提前规划好的主机名进行安装和配置。如果实验室后来人在安装系统时，没有配置好，不要紧，没有必要重新安装，在安装完系统之后仍然可以根据后来的规划对机器的主机名进行修改。

下面的例子我们将以 Master 机器为例，即主机名为“Master.Hadoop”，IP 为“192.168.1.2”进行一些主机名配置的相关操作。其他的 Slave 机器以此为依据进行修改。

1) 查看当前机器名称

用下面命令进行显示机器名称，如果跟规划的不一致，要按照下面进行修改。

```
hostname
```

```
[hadoop@Master ~]$ hostname  
Master.Hadoop  
[hadoop@Master ~]$
```

上图中，用“hostname”查“Master”机器的名字为“Master.Hadoop”，与我们预先规划的一致。

2) 修改当前机器名称

假定我们发现我们的机器的主机名不是我们想要的，通过对“**/etc/sysconfig/network**”文件修改其中“**HOSTNAME**”后面的值，改成我们规划的名称。

这个“**/etc/sysconfig/network**”文件是定义 hostname 和是否利用网络的不接触网络设备的对系统全体定义的文件。

设定形式：设定值=值

“/etc/sysconfig/network”的**设定项目**如下：

```
NETWORKING 是否利用网络  
GATEWAY 默认网关  
IPGATEWAYDEV 默认网关的接口名  
HOSTNAME 主机名  
DOMAIN 域名
```

用下面命令进行修改当前机器的主机名（**备注：**修改系统文件一般用 **root** 用户）

```
vim /etc/sysconfig/network
```

```
[root@Master ~]# vim /etc/sysconfig/network
```

通过上面的命令我们从“/etc/sysconfig/network”中找到“HOSTNAME”进行修改，查

看内容如下：

```
NETWORKING=yes
HOSTNAME=Master.Hadoop
GATEWAY=192.168.1.1
```

3) 修改当前机器 IP

假定我们的机器连 IP 在当时安装机器时都没有配置好，那此时我们需要对“**ifcfg-eth0**”文件进行配置，该文件位于“**/etc/sysconfig/network-scripts**”文件夹下。

在这个目录下面，存放的是网络接口（网卡）的制御脚本文件（控制文件），ifcfg-eth0 是默认的第一个网络接口，如果机器中有多个网络接口，那么名字就将依此类推 ifcfg-eth1，ifcfg-eth2，ifcfg-eth3，.....。

这里面的文件是相当重要的，涉及到网络能否正常工作。

设定形式：设定值=值

设定项目项目如下：

```
DEVICE 接口名（设备,网卡）
BOOTPROTO IP 的配置方法（static:固定 IP， dhcpHCP， none:手动）
HWADDR MAC 地址
ONBOOT 系统启动的时候网络接口是否有效（yes/no）
TYPE 网络类型（通常是 Ethernet）
NETMASK 网络掩码
IPADDR IP 地址
IPV6INIT IPV6 是否有效（yes/no）
GATEWAY 默认网关 IP 地址
```

查看“/etc/sysconfig/network-scripts/ifcfg-eth0”内容，如果 IP 不复核，就行修改。

```
[root@Master ~]# more /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
NM_CONTROLLED="yes"
ONBOOT=yes
HWADDR=00:25:64:C1:8D:FD
TYPE=Ethernet
BOOTPROTO=none
IPADDR=192.168.1.2
PREFIX=24
GATEWAY=192.168.1.1
DNS1=202.113.112.55
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System eth0"
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
[root@Master ~]#
```

如果上图中 IP 与规划不相符，用下面命令进行修改：

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

修改完之后可以用“ifconfig”进行查看。

```
[root@Master ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:25:64:C1:8D:FD
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::225:64ff:fe01:8dfd/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:381446 errors:0 dropped:0 overruns:0 frame:0
          TX packets:250908 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:179248257 (170.9 MiB)  TX bytes:274349033 (261.6 MiB)
          Memory:fdfe0000-fe000000
```

4) 配置 hosts 文件（必须）

“/etc/hosts”这个文件是用来配置主机将用的 DNS 服务器信息，是记载 LAN 内接续的各主机的对应[HostName 和 IP]用的。当用户在进行网络连接时，首先查找该文件，寻找对应主机名（或域名）对应的 IP 地址。

我们要测试两台机器之间知否连通，一般用“ping 机器的 IP”，如果想用“ping 机器的主机名”发现找不见该名称的机器，解决的办法就是修改“/etc/hosts”这个文件，通过把 LAN 内的各主机的 IP 地址和 HostName 的**一一对应**写入这个文件的时候，就可以解决问题。

例如：机器为“Master.Hadoop:192.168.1.2”对机器为“Slave1.Hadoop:192.168.1.3”用命令“ping”记性连接测试。测试结果如下：

```
[root@Master ~]# ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.580 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.227 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.248 ms
^C
--- 192.168.1.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2015ms
rtt min/avg/max/mdev = 0.227/0.351/0.580/0.163 ms
[root@Master ~]# ping Slave1.Hadoop
ping: unknown host Slave1.Hadoop
[root@Master ~]#
```

从上图中的值，直接对 IP 地址进行测试，能够 ping 通，但是对主机名进行测试，发现没有 ping 通，提示“unknown host——未知主机”，这时查看“Master.Hadoop”的“/etc/hosts”文件内容。

```
[root@Master ~]# more /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
[root@Master ~]#
```

发现里面没有“192.168.1.3 Slave1.Hadoop”内容，故而本机器是无法对机器的主机名为“Slave1.Hadoop”解析。

在进行 **Hadoop 集群** 配置中，需要在“/etc/hosts”文件中添加集群中所有机器的 IP 与主机名，这样 Master 与所有的 Slave 机器之间不仅可以通过 IP 进行通信，而且还可以通过主机名进行通信。所以在所有的机器上的“/etc/hosts”文件**末尾**中都要添加如下内容：

```
192.168.1.2 Master.Hadoop
192.168.1.3 Slave1.Hadoop
192.168.1.4 Slave2.Hadoop
192.168.1.5 Slave3.Hadoop
```

用以下命令进行添加：

```
vim /etc/hosts
```

```
[root@Master ~]# vim /etc/hosts
```

添加结果如下：

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.2  Master.Hadoop
192.168.1.3  Slave1.Hadoop
192.168.1.4  Slave2.Hadoop
192.168.1.5  Slave3.Hadoop
```

现在我们在进行对机器为“Slave1.Hadoop”的主机名进行 ping 通测试，看是否能测试成功。

```
[root@Master ~]# ping Slave1.Hadoop
PING Slave1.Hadoop (192.168.1.3) 56(84) bytes of data.
64 bytes from Slave1.Hadoop (192.168.1.3): icmp_seq=1 ttl=64 time=1.43 ms
64 bytes from Slave1.Hadoop (192.168.1.3): icmp_seq=2 ttl=64 time=0.249 ms
64 bytes from Slave1.Hadoop (192.168.1.3): icmp_seq=3 ttl=64 time=0.247 ms
^C
--- Slave1.Hadoop ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2630ms
rtt min/avg/max/mdev = 0.247/0.642/1.430/0.557 ms
[root@Master ~]#
```

从上图中我们已经能用主机名进行 ping 通了，说明我们刚才添加的内容，在局域网内能进行 DNS 解析了，那么现在剩下的事儿就是在其余的 Slave 机器上进行相同的配置。然后进行测试。（**备注：**当设置 SSH 无密码验证后，可以“scp”进行复制，然后把原来的“hosts”文件执行覆盖即可。）

1.4 所需软件

1) JDK 软件

下载地址：<http://www.oracle.com/technetwork/java/javase/index.html>

JDK 版本：[jdk-6u31-linux-i586.bin](#)

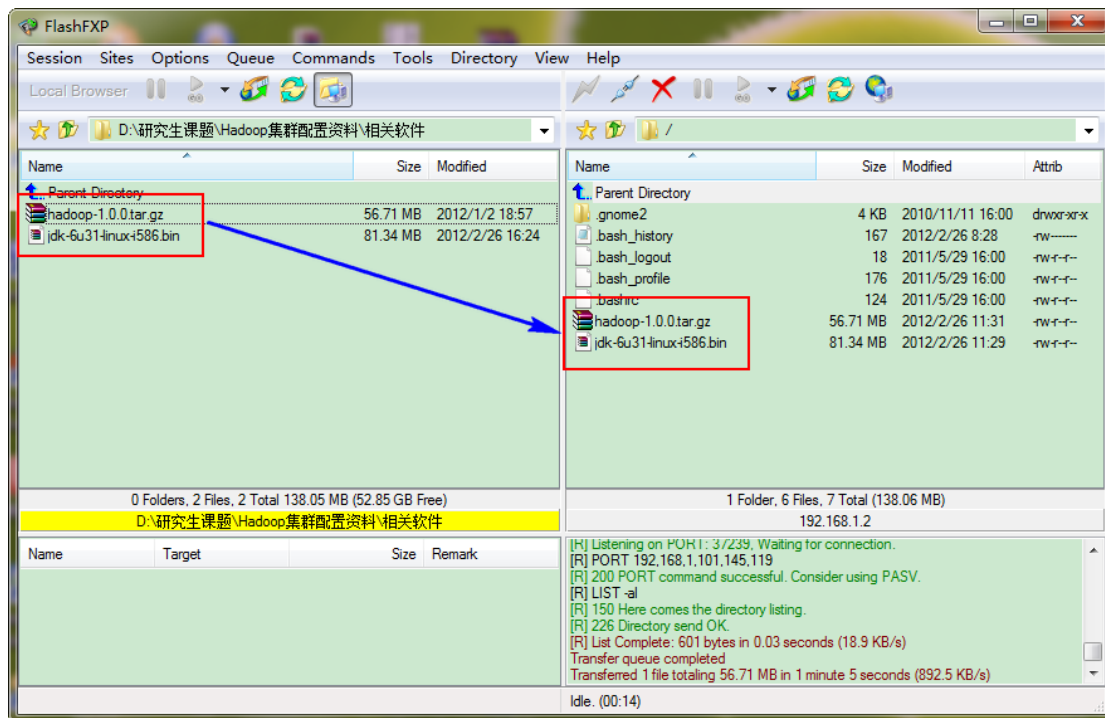
2) Hadoop 软件

下载地址：<http://hadoop.apache.org/common/releases.html>

Hadoop 版本：[hadoop-1.0.0.tar.gz](#)

1.5 VSFTP上传

在“**Hadoop 集群（第 3 期）**”讲了 VSFTP 的安装及配置，如果没有安装 VSFTP 可以按照该文档进行安装。如果安装好了，就可以通过 **FlashFXP.exe** 软件把我们下载的 JDK6.0 和 Hadoop1.0 软件上传到“**Master.Hadoop:192.168.1.2**”服务器上。



刚才我们用一般用户（**hadoop**）通过 FlashFXP 软件把所需的两个软件上传了跟目下，我们通过命令查看下是否已经上传了。

```
Last login: Sun Feb 26 18:43:12 2012 from 192.168.1.101
[hadoop@Master ~]$ ll
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[hadoop@Master ~]$
```

从图中，我们的所需软件已经准备好了。

2、SSH无密码验证配置

Hadoop 运行过程中需要管理远端 Hadoop 守护进程，在 Hadoop 启动以后，NameNode 是通过 SSH（Secure Shell）来启动和停止各个 DataNode 上的各种守护进程的。这就必须在节点之间执行指令的时候是不需要输入密码的形式，故我们需要配置 SSH 运用无密码公钥认证的形式，这样 NameNode 使用 SSH 无密码登录并启动 DataName 进程，同样原理，DataNode 上也能使用 SSH 无密码登录到 NameNode。

2.1 安装和启动SSH协议

在“Hadoop 集群（第 1 期）”安装 CentOS6.0 时，我们选择了一些基本安装包，所以我们需要两个服务：ssh 和 rsync 已经安装了。可以通过下面命令查看结果显示如下：

```
rpm -qa | grep openssh
rpm -qa | grep rsync
```

```
[hadoop@Master ~]$ rpm -qa | grep openssh
openssh-clients-5.3p1-20.el6.i686
openssh-server-5.3p1-20.el6.i686
openssh-5.3p1-20.el6.i686
[hadoop@Master ~]$ rpm -qa | grep rsync
rsync-3.0.6-5.el6.i686
[hadoop@Master ~]$
```

假设没有安装 ssh 和 rsync，可以通过下面命令进行安装。

```
yum install ssh          安装 SSH 协议
yum install rsync （rsync 是一个远程数据同步工具，可通过 LAN/WAN 快速同步多台主机间的文件）
service sshd restart    启动服务
```

确保所有的服务器都安装，上面命令执行完毕，各台机器之间可以通过密码验证相互登。

2.2 配置Master无密码登录所有Slave

1) SSH 无密码原理

Master（NameNode | JobTracker）作为客户端，要实现无密码公钥认证，连接到服务器 Slave（DataNode | Tasktracker）上时，需要在 Master 上生成一个密钥对，包括一个公钥和一个私钥，而后将公钥复制到所有的 Slave 上。当 Master 通过 SSH 连接 Slave 时，Slave 就会生成一个随机数并用 Master 的公钥对随机数进行加密，并发送给 Master。Master 收到加密数之后再用私钥解密，并将解密数回传给 Slave，Slave 确认解密数无误之后就允许 Master

进行连接了。这就是一个公钥认证过程，其间不需要用户手工输入密码。重要过程是将客户端 Master 复制到 Slave 上。

2) Master 机器上生成密码对

在 Master 节点上执行以下命令：

```
ssh-keygen -t rsa -P ''
```

这条命令是生成其**无密码密钥对**，询问其保存路径时**直接回车**采用默认路径。生成的密钥对：id_rsa 和 id_rsa.pub，默认存储在“**/home/hadoop/.ssh**”目录下。

```
[hadoop@Master ~]$ 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:
05:c2:0b:75:8c:4c:aa:72:a3:4d:5c:10:09:cc:fa:ba hadoop@Master.Hadoop
The key's randomart image is:
+--[ RSA 2048 ]-----+
|+.oo =+oo|
|o...+o..|
|.o. .|
|..o. .|
|..* S|
|*..|
|...|
|. |
|E.|
+-----+
```

查看“/home/hadoop/”下是否有“.ssh”文件夹，且“.ssh”文件下是否有两个刚生产的无密码密钥对。

```
[hadoop@Master ~]$ ll -a | grep .ssh
drwx-----. 2 hadoop hadoop 4096 2月 27 04:23 .ssh
[hadoop@Master ~]$ cd .ssh
[hadoop@Master .ssh]$ ll
总用量 8
-rw-----. 1 hadoop hadoop 1675 2月 27 04:23 id_rsa
-rw-r--r--. 1 hadoop hadoop 402 2月 27 04:23 id_rsa.pub
[hadoop@Master .ssh]$
```

接着在 Master 节点上做如下配置，把 id_rsa.pub 追加到授权的 key 里面去。

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
[hadoop@Master .ssh]$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
[hadoop@Master .ssh]$ ll
总用量 12
-rw-rw-r--. 1 hadoop hadoop 402 2月 27 04:35 authorized_keys
-rw-----. 1 hadoop hadoop 1675 2月 27 04:23 id_rsa
-rw-r--r--. 1 hadoop hadoop 402 2月 27 04:23 id_rsa.pub
```


在验证前，需要做两件事儿。第一件事儿是修改文件“**authorized_keys**”权限（**权限的设置非常重要，因为不安全的设置安全设置，会让你不能使用 RSA 功能**），另一件事儿是用 **root** 用户设置“**/etc/ssh/sshd_config**”的内容。使其无密码登录有效。

1) 修改文件“**authorized_keys**”

```
chmod 600 ~/.ssh/authorized_keys
```

```
[hadoop@Master ~]$ chmod 600 ~/.ssh/authorized_keys
[hadoop@Master ~]$ ll ~/.ssh/
总用量 16
-rw----- 1 hadoop hadoop 402 2月 27 05:09 authorized_keys
-rw----- 1 hadoop hadoop 1675 2月 27 05:08 id_rsa
-rw-r--r-- 1 hadoop hadoop 402 2月 27 05:08 id_rsa.pub
-rw-r--r-- 1 hadoop hadoop 784 2月 27 19:12 known_hosts
[hadoop@Master ~]$
```

备注：如果不进行设置，在验证时，仍提示你输入密码，在这里花费了将近半天时间来查找原因。在网上查到了几篇不错的文章，把作为“**Hadoop 集群_第 5 期副刊_JDK 和 SSH 无密码配置**”来帮助额外学习之用。

2) 设置 SSH 配置

用 **root** 用户登录服务器修改 SSH 配置文件“**/etc/ssh/sshd_config**”的下列内容。

```
[root@Master ~]# vim /etc/ssh/sshd_config
```

```
RSAAuthentication yes      # 启用 RSA 认证
PubkeyAuthentication yes   # 启用公钥私钥配对认证方式
AuthorizedKeysFile .ssh/authorized_keys # 公钥文件路径（和上面生成的文件同）
```

设置完之后记得**重启 SSH 服务**，才能使刚才设置有效。

```
service sshd restart
```

退出 root 登录，使用 **hadoop** 普通用户验证是否成功。

```
ssh localhost
```

```
[hadoop@Master ~]$ ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is 28:0b:51:4b:f5:87:53:9e:06:30:ba:86:da:23:f7:df.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Last login: Mon Feb 27 19:22:01 2012 from 192.168.1.3
[hadoop@Master ~]$
```

输入“yes”

从上图中得知无密码登录本级已经设置完毕，接下来的事儿是把**公钥**复制**所有**的 **Slave** 机器上。使用下面的命令格式进行复制公钥：

```
scp ~/.ssh/id_rsa.pub 远程用户名@远程服务器 IP:~/
```

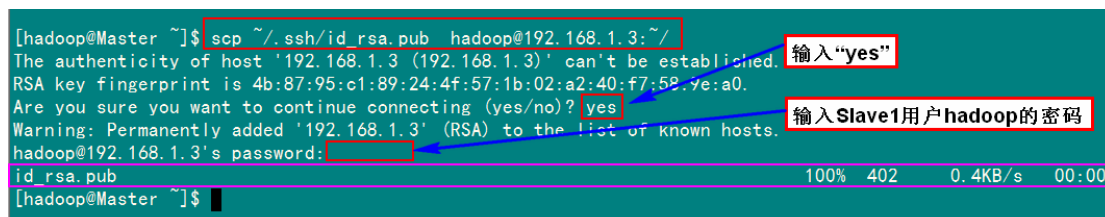
例如：

```
scp ~/.ssh/id_rsa.pub hadoop@192.168.1.3:~/
```

上面的命令是**复制**文件“**id_rsa.pub**”到服务器 IP 为“**192.168.1.3**”的用户为“**hadoop**”的“**/home/hadoop/**”下面。

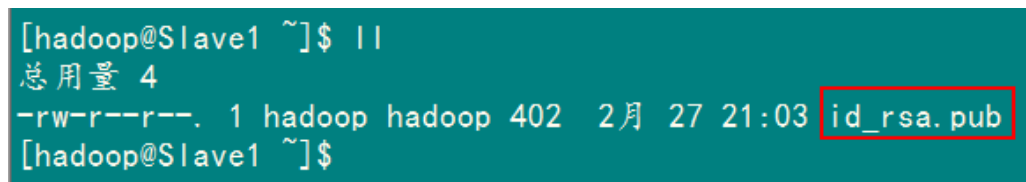
下面就针对 IP 为“192.168.1.3”的 Slave1.Hadoop 的节点进行配置。

1) 把 Master.Hadoop 上的公钥复制到 Slave1.Hadoop 上



```
[hadoop@Master ~]$ scp ~/.ssh/id_rsa.pub hadoop@192.168.1.3:~/
The authenticity of host '192.168.1.3 (192.168.1.3)' can't be established.
RSA key fingerprint is 4b:87:95:c1:89:24:4f:57:1b:02:a2:40:f7:52:7e:a0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.3' (RSA) to the list of known hosts.
hadoop@192.168.1.3's password:
id_rsa.pub                                100% 402    0.4KB/s  00:00
[hadoop@Master ~]$
```

从上图中我们得知，已经把文件“id_rsa.pub”传过去了，因为并没有建立起无密码连接，所以在连接时，仍然要提示输入输入 Slave1.Hadoop 服务器用户 hadoop 的密码。为了确保确实已经把文件传过去了，用 SecureCRT 登录 Slave1.Hadoop:192.168.1.3 服务器，查看“/home/hadoop/”下是否存在这个文件。



```
[hadoop@Slave1 ~]$ ll
总用量 4
-rw-r--r--. 1 hadoop hadoop 402  2月 27 21:03 id_rsa.pub
[hadoop@Slave1 ~]$
```

从上面得知我们已经成功把公钥复制过去了。

2) 在“/home/hadoop/”下创建“.ssh”文件夹

这一步**并不是必须的**，如果在 Slave1.Hadoop 的“/home/hadoop”**已经存在**就不需要创建了，因为我们之前并没有对 Slave 机器做过无密码登录配置，所以该文件是不存在的。用下面命令进行创建。（**备注：**用 hadoop 登录系统，如果不涉及系统文件修改，一般情况下都是用我们之前建立的普通用户 hadoop 进行执行命令。）

```
mkdir ~/.ssh
```

然后是修改文件夹“**.ssh**”的用户权限，把他的权限修改为“**700**”，用下面命令执行：

```
chmod 700 ~/.ssh
```

备注：如果不进行，即使你按照前面的操作设置了“authorized_keys”权限，并配置了

“/etc/ssh/sshd_config”，还重启了 sshd 服务，在 Master 能用 “ssh localhost” 进行无密码登录，但是对 Slave1.Hadoop 进行登录仍然需要输入密码，就是因为 “.ssh” 文件夹的权限设置不对。这个文件夹 “.ssh” 在配置 SSH 无密码登录时系统自动生成时，权限自动为 “700”，如果是自己手动创建，它的组权限和其他权限都有，这样就会导致 RSA 无密码远程登录失败。

```
[hadoop@Slave1 ~]$ ll -a
总用量 32
drwx-----. 3 hadoop hadoop 4096 2月 27 21:03 .
drwxr-xr-x. 4 root root 4096 2月 24 03:55 ..
-rw-----. 1 hadoop hadoop 418 2月 27 18:44 .bash_history
-rw-r--r--. 1 hadoop hadoop 18 5月 31 2011 .bash_logout
-rw-r--r--. 1 hadoop hadoop 176 5月 31 2011 .bash_profile
-rw-r--r--. 1 hadoop hadoop 124 5月 31 2011 .bashrc
drwxr-xr-x. 2 hadoop hadoop 4096 11月 12 2010 .gnome2
-rw-r--r--. 1 hadoop hadoop 402 2月 27 21:03 id_rsa.pub

[hadoop@Slave1 ~]$ mkdir ~/.ssh
[hadoop@Slave1 ~]$ ll -a
总用量 36
drwx-----. 4 hadoop hadoop 4096 2月 27 21:20 .
drwxr-xr-x. 4 root root 4096 2月 24 03:55 ..
-rw-----. 1 hadoop hadoop 418 2月 27 18:44 .bash_history
-rw-r--r--. 1 hadoop hadoop 18 5月 31 2011 .bash_logout
-rw-r--r--. 1 hadoop hadoop 176 5月 31 2011 .bash_profile
-rw-r--r--. 1 hadoop hadoop 124 5月 31 2011 .bashrc
drwxr-xr-x. 2 hadoop hadoop 4096 11月 12 2010 .gnome2
-rw-r--r--. 1 hadoop hadoop 402 2月 27 21:03 id_rsa.pub
drwxrwxr-x. 2 hadoop hadoop 4096 2月 27 21:20 .ssh

[hadoop@Slave1 ~]$ chmod 700 ~/.ssh
[hadoop@Slave1 ~]$ ll -a | grep .ssh
drwx-----. 2 hadoop hadoop 4096 2月 27 21:20 .ssh
[hadoop@Slave1 ~]$
```

对比上面两张图，发现文件夹 “.ssh” 权限已经变了。

3) 追加到授权文件 “authorized_keys”

到目前为止 Master.Hadoop 的公钥也有了，文件夹 “.ssh” 也有了，且权限也修改了。这一步就是把 Master.Hadoop 的公钥追加到 Slave1.Hadoop 的授权文件 “authorized_keys” 中去。使用下面命令进行追加并修改 “authorized_keys” 文件权限：

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

```
[hadoop@Slave1 ~]$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
[hadoop@Slave1 ~]$ chmod 600 ~/.ssh/authorized_keys
[hadoop@Slave1 ~]$ ll ~/.ssh | grep authorized_keys
-rw-----. 1 hadoop hadoop 402 2月 27 21:33 authorized_keys
[hadoop@Slave1 ~]$
```

4) 用 root 用户修改 “/etc/ssh/sshd_config”

具体步骤参考前面 Master.Hadoop 的“**设置 SSH 配置**”，具体分为两步：第 1 是修改配置文件；第 2 是重启 SSH 服务。

5) 用 Master.Hadoop 使用 SSH 无密码登录 Slave1.Hadoop

当前面的步骤设置完毕，就可以使用下面命令格式进行 SSH 无密码登录了。

```
ssh 远程服务器 IP
```

```
[hadoop@Master ~]$ ssh 192.168.1.3
Last login: Mon Feb 27 19:33:03 2012 from 192.168.1.2
[hadoop@Slave1 ~]$
```

从上图我们主要 3 个地方，第 1 个就是 SSH 无密码登录命令，第 2、3 个就是登录前后“@”后面的**机器名**变了，由“**Master**”变为了“**Slave1**”，这就说明我们已经成功实现了 SSH 无密码登录了。

最后记得把“/home/hadoop/”目录下的“id_rsa.pub”文件删除掉。

```
rm -r ~/id_rsa.pub
```

```
[hadoop@Slave1 ~]$ ll
总用量 4
-rw-r--r--. 1 hadoop hadoop 402 2月 27 21:03 id_rsa.pub
[hadoop@Slave1 ~]$ rm -r id_rsa.pub
[hadoop@Slave1 ~]$ ll
总用量 0
[hadoop@Slave1 ~]$
```

到此为止，我们经过前 5 步已经实现了从“**Master.Hadoop**”到“**Slave1.Hadoop**”SSH 无密码登录，下面就是重复上面的步骤把剩余的两台(**Slave2.Hadoop** 和 **Slave3.Hadoop**)Slave 服务器进行配置。**这样**，我们就完成了“配置 **Master** 无密码登录所有的 **Slave** 服务器”。

2.3 配置所有Slave无密码登录Master

和 Master 无密码登录所有 Slave 原理一样，就是把 Slave 的公钥**追加**到 Master 的“.ssh”文件夹下的“authorized_keys”中，记得是**追加 (>>)**。

为了说明情况，我们现在就以“**Slave1.Hadoop**”无密码登录“**Master.Hadoop**”为例，进行一遍操作，也算是**巩固**一下前面所学知识，剩余的“**Slave2.Hadoop**”和“**Slave3.Hadoop**”就按照这个示例进行就可以了。

首先创建“**Slave1.Hadoop**”自己的公钥和私钥，并把自己的公钥追加到“authorized_keys”文件中。用到的命令如下：

```
ssh-keygen -t rsa -P ''
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
[hadoop@Slave1 .ssh]$ 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:
88:a1:d4:59:3b:03:d1:5e:5b:60:e6:d4:95:0a:fc:dd hadoop@Slave1.Hadoop
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      oo..=o   .    |
|      .+. *+ o   .    |
|      .+.+. = o   .    |
|      . . o.+ o   . E    |
|      . . . S          |
+-----+
[hadoop@Slave1 .ssh]$ ll ~/.ssh
总用量 12
-rw-----. 1 hadoop hadoop 402  2月 27 21:33 authorized_keys
-rw-----. 1 hadoop hadoop 1675  2月 28 00:40 id_rsa
-rw-r--r--. 1 hadoop hadoop 402  2月 28 00:40 id_rsa.pub
[hadoop@Slave1 .ssh]$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
[hadoop@Slave1 .ssh]$
```

直接按回车键

查看是否已经生成公钥

追加公钥

接着是用命令“**scp**”复制“Slave1.Hadoop”的公钥“id_rsa.pub”到“Master.Hadoop”的“/home/hadoop/”目录下，并追加到“Master.Hadoop”的“authorized_keys”中。

1) 在“Slave1.Hadoop”服务器的操作

用到的命令如下:

```
scp ~/.ssh/id_rsa.pub hadoop@192.168.1.2:~/
```

```
[hadoop@Slave1 ~]$ scp ~/.ssh/id_rsa.pub hadoop@192.168.1.2:~/
The authenticity of host '192.168.1.2 (192.168.1.2)' can't be established.
RSA key fingerprint is 28:0b:51:4b:f5:87:53:9e:06:30:ba:86:da:23:f7:df.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.2' (RSA) to the list of known hosts.
hadoop@192.168.1.2's password:
id_rsa.pub 100% 402 0.4KB/s 00:00
[hadoop@Slave1 ~]$
```

输入“yes”

输入Master下hadoop的密码

2) 在“Master.Hadoop”服务器的操作

用到的命令如下:

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
[hadoop@Master ~]$ ll
总用量 141376
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 402 2月 28 00:56 id_rsa.pub
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[hadoop@Master ~]$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

然后删除掉刚才复制过来的“id_rsa.pub”文件。

```
[hadoop@Master ~]$ rm ~/id_rsa.pub
[hadoop@Master ~]$ ll
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[hadoop@Master ~]$
```

最后是测试从“Slave1.Hadoop”到“Master.Hadoop”无密码登录。

```
[hadoop@Slave1 ~]$ ssh 192.168.1.2
Last login: Mon Feb 27 21:02:56 2012 from localhost
[hadoop@Master ~]$
```

从上面结果中可以看到已经成功实现了，再试下从“Master.Hadoop”到“Slave1.Hadoop”无密码登录。

```
[hadoop@Master ~]$ ssh 192.168.1.3
Last login: Mon Feb 27 23:52:33 2012 from 192.168.1.104
[hadoop@Slave1 ~]$
```

至此“Master.Hadoop”与“Slave1.Hadoop”之间可以互相无密码登录了，剩下的就是按照上面的步骤把剩余的“Slave2.Hadoop”和“Slave3.Hadoop”与“Master.Hadoop”之间建立起无密码登录。这样，Master 能无密码验证登录每个 Slave，每个 Slave 也能无密码验证登录到 Master。

3、Java环境安装

所有的机器上都要安装 JDK，现在就先在 Master 服务器安装，然后其他服务器按照步骤重复进行即可。安装 JDK 以及配置环境变量，需要以“root”的身份进行。

3.1 安装JDK

首先用 root 身份登录“Master.Hadoop”后在“/usr”下创建“java”文件夹，再把用 FTP 上传到“/home/hadoop/”下的“jdk-6u31-linux-i586.bin”复制到“/usr/java”文件夹中。

```
mkdir /usr/java
cp /home/hadoop/jdk-6u31-linux-i586.bin /usr/java
```



```

[root@Master ~]# ll /usr
总用量 96
drwxr-xr-x. 2 root root 24576 2月 24 05:34 bin
drwxr-xr-x. 2 root root 4096 11月 11 2010 etc
drwxr-xr-x. 2 root root 4096 11月 11 2010 games
drwxr-xr-x. 33 root root 4096 2月 24 04:40 include
drwxr-xr-x. 53 root root 32768 2月 24 05:34 lib
drwxr-xr-x. 14 root root 4096 2月 24 05:34 libexec
drwxr-xr-x. 11 root root 4096 2月 24 04:37 local
drwxr-xr-x. 2 root root 4096 2月 24 05:34 sbin
drwxr-xr-x. 95 root root 4096 2月 24 04:41 share
drwxr-xr-x. 4 root root 4096 2月 24 04:37 src
lrwxrwxrwx. 1 root root 10 2月 24 04:37 tmp -> ../var/tmp
[root@Master ~]# mkdir /usr/java
[root@Master ~]# ll /home/hadoop
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[root@Master ~]# cp /home/hadoop/jdk-6u31-linux-i586.bin /usr/java
[root@Master ~]# ll /usr/java
总用量 83296
-rw-r--r--. 1 root root 85292206 2月 28 03:38 jdk-6u31-linux-i586.bin
[root@Master ~]#

```

先查看“usr”下是否已经存在“java”文件夹

发现没有

创建“java”文件夹

复制JDK到“java”文件夹下面

查看确认已经复制过来

接着进入“/usr/java”目录下通过下面命令使其JDK获得可执行权限，并安装JDK。

```

chmod +x jdk-6u31-linux-i586.bin
./jdk-6u31-linux-i586.bin

```

```

[root@Master ~]# cd /usr/java
[root@Master java]# ll
总用量 83296
-rw-r--r--. 1 root root 85292206 2月 28 03:38 jdk-6u31-linux-i586.bin
[root@Master java]# chmod +x jdk-6u31-linux-i586.bin
[root@Master java]# ll
总用量 83296
-rwxr-xr-x. 1 root root 85292206 2月 28 03:38 jdk-6u31-linux-i586.bin
[root@Master java]# ./jdk-6u31-linux-i586.bin

```

进入“/usr/java”目录

给JDK添加执行权限，对比发现权限和颜色变化

点击“回车”就开始安

按照上面几步进行操作，最后点击“Enter”键开始安装，安装完会提示你按“Enter”键退出，然后查看“/usr/java”下面会发现多了一个名为“jdk1.6.0_31”文件夹，说明我们的JDK安装结束，删除“jdk-6u31-linux-i586.bin”文件，进入下一个“配置环境变量”环节。

```

[root@Master java]# ll
总用量 83300
drwxr-xr-x. 10 root root 4096 2月 28 04:05 jdk1.6.0_31
-rwxr-xr-x. 1 root root 85292206 2月 28 03:38 jdk-6u31-linux-i586.bin
[root@Master java]# rm -rf jdk-6u31-linux-i586.bin
[root@Master java]# ll
总用量 4
drwxr-xr-x. 10 root root 4096 2月 28 04:05 jdk1.6.0_31

```

java安装目录

删除JDK安装文件

3.2 配置环境变量

编辑“/etc/profile”文件，在后面添加 Java 的“JAVA_HOME”、“CLASSPATH”以及“PATH”内容。

1) 编辑 “/etc/profile” 文件

```
vim /etc/profile
```

```
[root@Master ~]# vim /etc/profile
```

2) 添加 Java 环境变量

在“/etc/profile”文件的尾部添加以下内容：

```
# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31/
export JRE_HOME=/usr/java/jdk1.6.0_31/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

或者

```
# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin
```

以上两种意思一样，那么我们就选择第2种来进行设置。

```
unset i
unset pathmunge

# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin

— 插入 —
```

在“/etc/profile”文件末尾添加

3) 使配置生效

保存并退出，执行下面命令使其配置立即生效。

```
source /etc/profile
```

```
[root@Master ~]# source /etc/profile
[root@Master ~]#
```

3.3 验证安装成功

配置完毕并生效后，用下面命令判断是否成功。

```
java -version
```

```
[root@Master ~]# java -version
java version "1.6.0_31"
Java(TM) SE Runtime Environment (build 1.6.0_31-b04)
Java HotSpot(TM) Server VM (build 20.6-b01, mixed mode)
[root@Master ~]#
```

从上图中得知，我们以确定 JDK 已经安装成功。

3.4 安装剩余机器

这时用**普通用户 hadoop**通过下面命令格式把“Master.Hadoop”文件夹“/home/hadoop/”的 JDK 复制到其他 Slave 的“/home/hadoop/”下面，剩下的事儿就是在其余的 Slave 服务器上按照上图的步骤安装 JDK。

```
scp /home/hadoop/jdk-6u31-linux-i586.bin 远程用户名@远程服务器 IP:~/
```

或者

```
scp ~/jdk-6u31-linux-i586.bin 远程用户名@远程服务器 IP:~/
```

备注：“~”代表**当前**用户的主目录，**当前用户为 hadoop**，所以“~”代表“/home/hadoop”。

例如：把 JDK 从“Master.Hadoop”复制到“Slave1.Hadoop”的命令如下。

```
scp ~/jdk-6u31-linux-i586 hadoop@192.168.1.3:~/
```

```
[hadoop@Master ~]$ ll
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[hadoop@Master ~]$ scp ~/jdk-6u31-linux-i586.bin hadoop@192.168.1.3:~/
jdk-6u31-linux-i586.bin
[hadoop@Master ~]$
```

这说明已经复制完毕

100% 81MB 11.6MB/s 00:07

然后查看“Slave1.Hadoop”的“/home/hadoop”查看是否已经复制成功了。

```
[hadoop@Slave1 ~]$ ll
总用量 83296
-rw-r--r--. 1 hadoop hadoop 85292206 2月 28 05:02 jdk-6u31-linux-i586.bin
[hadoop@Slave1 ~]$
```

从上图中得知，我们已经成功复制了，现在我们就用**最高权限用户 root** 进行安装了。其他的与这个一样。

4、Hadoop集群安装

所有的机器上都要安装 hadoop，现在就先在 Master 服务器安装，然后其他服务器按照步骤重复进行即可。安装和配置 hadoop 需要以“**root**”的身份进行。

4.1 安装hadoop

首先用 **root** 用户登录“Master.Hadoop”机器，查看我们之前用 FTP 上传至“/home/Hadoop”上传的“**hadoop-1.0.0.tar.gz**”。

```
Last login: Tue Feb 28 21:53:30 2012 from 192.168.1.102
[hadoop@Master ~]$ su -
密码:
[root@Master ~]# ll /home/hadoop
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[root@Master ~]#
```

接着把“**hadoop-1.0.0.tar.gz**”复制到“/usr”目录下面。

```
cp /home/hadoop/hadoop-1.0.0.tar.gz /usr
```

```
[root@Master ~]# ll /home/hadoop
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[root@Master ~]# cp /home/hadoop/hadoop-1.0.0.tar.gz /usr
[root@Master ~]# ll /usr | grep hadoop
-rw-r--r--. 1 root root 59468784 2月 28 22:23 hadoop-1.0.0.tar.gz
[root@Master ~]#
```

下一步进入“/usr”目录下，用下面命令把“**hadoop-1.0.0.tar.gz**”进行解压，并将其命名为“hadoop”，把该文件夹的**读权限**分配给普通用户 **hadoop**，然后删除“**hadoop-1.0.0.tar.gz**”安装包。

cd /usr	#进入 “/usr” 目录
tar -zxvf hadoop-1.0.0.tar.gz	#解压 “ hadoop-1.0.0.tar.gz ” 安装包
mv hadoop-1.0.0 hadoop	#将 “hadoop-1.0.0” 文件夹 重命名 “hadoop”
chown -R hadoop:hadoop hadoop	#将文件夹 “hadoop” 读权限分配给 hadoop 用户
rm -rf hadoop-1.0.0.tar.gz	#删除 “ hadoop-1.0.0.tar.gz ” 安装包

```
[root@Master ~]# cd /usr
[root@Master usr]# ll
总用量 58176
dr-xr-xr-x. 2 root root    24576 2月 24 05:34 bin
drwxr-xr-x. 2 root root    4096 11月 11 2010 etc
drwxr-xr-x. 2 root root    4096 11月 11 2010 games
-rw-r--r--. 1 root root 59468784 2月 28 22:23 hadoop-1.0.0.tar.gz
drwxr-xr-x. 33 root root    4096 2月 24 04:40 include
drwxr-xr-x. 3 root root    4096 2月 28 04:06 java
dr-xr-xr-x. 53 root root   32768 2月 24 05:34 lib
drwxr-xr-x. 14 root root    4096 2月 24 05:34 libexec
drwxr-xr-x. 11 root root    4096 2月 24 04:37 local
dr-xr-xr-x. 2 root root    4096 2月 24 05:34 sbin
drwxr-xr-x. 95 root root    4096 2月 24 04:41 share
drwxr-xr-x. 4 root root    4096 2月 24 04:37 src
lrwxrwxrwx. 1 root root     10 2月 24 04:37 tmp -> ../var/tmp
[root@Master usr]# tar -zxvf hadoop-1.0.0.tar.gz
```

解压后，并重命名。

```
[root@Master usr]# ll
总用量 58180
dr-xr-xr-x. 2 root root    24576 2月 24 05:34 bin
drwxr-xr-x. 2 root root    4096 11月 11 2010 etc
drwxr-xr-x. 2 root root    4096 11月 11 2010 games
drwxr-xr-x. 14 root root    4096 12月 16 00:39 hadoop-1.0.0
-rw-r--r--. 1 root root 59468784 2月 28 22:23 hadoop-1.0.0.tar.gz
drwxr-xr-x. 33 root root    4096 2月 24 04:40 include
drwxr-xr-x. 3 root root    4096 2月 28 04:06 java
dr-xr-xr-x. 53 root root   32768 2月 24 05:34 lib
drwxr-xr-x. 14 root root    4096 2月 24 05:34 libexec
drwxr-xr-x. 11 root root    4096 2月 24 04:37 local
dr-xr-xr-x. 2 root root    4096 2月 24 05:34 sbin
drwxr-xr-x. 95 root root    4096 2月 24 04:41 share
drwxr-xr-x. 4 root root    4096 2月 24 04:37 src
lrwxrwxrwx. 1 root root     10 2月 24 04:37 tmp -> ../var/tmp
[root@Master usr]# mv hadoop-1.0.0 hadoop
[root@Master usr]# ll
总用量 58180
dr-xr-xr-x. 2 root root    24576 2月 24 05:34 bin
drwxr-xr-x. 2 root root    4096 11月 11 2010 etc
drwxr-xr-x. 2 root root    4096 11月 11 2010 games
drwxr-xr-x. 14 root root    4096 12月 16 00:39 hadoop
-rw-r--r--. 1 root root 59468784 2月 28 22:23 hadoop-1.0.0.tar.gz
drwxr-xr-x. 33 root root    4096 2月 24 04:40 include
drwxr-xr-x. 3 root root    4096 2月 28 04:06 java
dr-xr-xr-x. 53 root root   32768 2月 24 05:34 lib
drwxr-xr-x. 14 root root    4096 2月 24 05:34 libexec
drwxr-xr-x. 11 root root    4096 2月 24 04:37 local
dr-xr-xr-x. 2 root root    4096 2月 24 05:34 sbin
drwxr-xr-x. 95 root root    4096 2月 24 04:41 share
drwxr-xr-x. 4 root root    4096 2月 24 04:37 src
lrwxrwxrwx. 1 root root     10 2月 24 04:37 tmp -> ../var/tmp
```

把“/usr/hadoop” **读权限**分配给 **hadoop** 用户（**非常重要**）

```
[root@Master usr]# chown -R hadoop:hadoop hadoop
[root@Master usr]# ll
总用量 104
dr-xr-xr-x.  2 root    root    24576  2月 24 05:34 bin
drwxr-xr-x.  2 root    root    4096 11月 11 2010 etc
drwxr-xr-x.  2 root    root    4096 11月 11 2010 games
drwxr-xr-x. 15 hadoop  hadoop   4096  2月 28 22:51 hadoop
drwxr-xr-x. 33 root    root    4096  2月 24 04:40 include
drwxr-xr-x.  3 root    root    4096  2月 28 04:06 java
dr-xr-xr-x. 53 root    root   32768  2月 24 05:34 lib
drwxr-xr-x. 14 root    root    4096  2月 24 05:34 libexec
drwxr-xr-x. 11 root    root    4096  2月 24 04:37 local
dr-xr-xr-x.  2 root    root    4096  2月 24 05:34 sbin
drwxr-xr-x. 95 root    root    4096  2月 24 04:41 share
drwxr-xr-x.  4 root    root    4096  2月 24 04:37 src
lrwxrwxrwx.  1 root    root      10  2月 24 04:37 tmp -> ../var/tmp
[root@Master usr]#
```

删除“**hadoop-1.0.0.tar.gz**”安装包

```
[root@Master usr]# rm -rf hadoop-1.0.0.tar.gz
[root@Master usr]#
```

最后在“/usr/hadoop”下面创建 **tmp** 文件夹，把 Hadoop 的安装路径添加到“/etc/profile”中，修改“/etc/profile”文件（配置 java 环境变量的文件），将以下语句添加到**末尾**，并使其有效：

```
# set hadoop path
export HADOOP_HOME=/usr/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

1) 在“/usr/hadoop”创建“**tmp**”文件夹

```
mkdir /usr/hadoop/tmp
```

```
[root@Master hadoop]# pwd
/usr/hadoop
[root@Master hadoop]# mkdir /usr/hadoop/tmp
[root@Master hadoop]#
```

2) 配置“/etc/profile”


```
vim /etc/profile
```

```
[root@Master ~]# vim /etc/profile
```

配置后的文件如下：

```
unset i
unset pathmunge

# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin

# set hadoop path
export HADOOP_HOME=/usr/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
— 插入 —
```

3) 重启 “/etc/profile”

```
source /etc/profile
```

```
[root@Master ~]# source /etc/profile
[root@Master ~]#
```

4.2 配置hadoop

1) 配置 hadoop-env.sh

该 “**hadoop-env.sh**” 文件位于 “**/usr/hadoop/conf**” 目录下。

```
[root@Master conf]# pwd
/usr/hadoop/conf
[root@Master conf]# vim hadoop-env.sh
```

在文件的末尾添加下面内容。

```
# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31
```

```
# The scheduling priority for daemon processes. See 'man nice'.
# export HADOOP_NICENESS=10

# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31
— 插入 —
```

Hadoop 配置文件在 conf 目录下，之前的版本的配置文件主要是 Hadoop-default.xml 和 Hadoop-site.xml。由于 Hadoop 发展迅速，代码量急剧增加，代码开发分为了 core，hdfs 和 map/reduce 三部分，配置文件也被分成了三个 core-site.xml、hdfs-site.xml、mapred-site.xml。core-site.xml 和 hdfs-site.xml 是站在 HDFS 角度上配置文件；core-site.xml 和 mapred-site.xml 是站在 MapReduce 角度上配置文件。

2) 配置 core-site.xml 文件

修改 Hadoop 核心配置文件 core-site.xml，这里配置的是 HDFS 的地址和端口号。

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/hadoop/tmp</value>
    (备注：请先在 /usr/hadoop 目录下建立 tmp 文件夹)
    <description>A base for other temporary directories.</description>
  </property>
  <!-- file system properties -->
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.1.2:9000</value>
  </property>
</configuration>
```

备注：如没有配置 hadoop.tmp.dir 参数，此时系统默认的临时目录为：/tmp/hadoo-hadoop。而这个目录在每次重启后都会被干掉，必须重新执行 format 才行，否则会出错。

用下面命令进行编辑：

```
[root@Master conf]# vim core-site.xml
```

编辑结果显示如下：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/hadoop/tmp</value>
    <description>A base for other temporary directories.</description>
  </property>
  <!--file system properties-->
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.1.2:9000</value>
  </property>
</configuration>
```

3) 配置 **hdfs-site.xml** 文件

修改 Hadoop 中 HDFS 的配置，配置的备份方式默认为 3。

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
    (备注: replication 是数据副本数量，默认为 3，salve 少于 3 台就会报错)
  </property>
</configuration>
```

用下面命令进行编辑：

```
[root@Master conf]# vim hdfs-site.xml
```

编辑结果显示如下：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

4) 配置 **mapred-site.xml** 文件

修改 Hadoop 中 MapReduce 的配置文件，配置的是 JobTracker 的地址和端口。

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>http://192.168.1.2:9001</value>
  </property>
</configuration>
```

用下面命令进行编辑：

```
[root@Master conf]# vim mapred-site.xml
```

编辑结果显示如下：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>http://192.168.1.2:9001</value>
  </property>
</configuration>
```

5) 配置 masters 文件

有两种方案：

(1) 第一种

修改 localhost 为 Master.Hadoop

(2) 第二种

去掉“localhost”，加入 Master 机器的 IP：192.168.1.2

为保险起见，启用第二种，因为万一忘记配置“/etc/hosts”局域网的 DNS 失效，这样就会出现意想不到的错误，但是一旦 IP 配对，网络畅通，就能通过 IP 找到相应主机。

用下面命令进行修改：

```
[root@Master conf]# vim masters
```

编辑结果显示如下：

```
[root@Master ~]# more /usr/hadoop/conf/masters
192.168.1.2
```

6) 配置 slaves 文件 (Master 主机特有)

有两种方案：

(1) 第一种

去掉“localhost”，每行只添加一个主机名，把剩余的 Slave 主机名都填上。

例如：添加形式如下

```
Slave1.Hadoop
Slave2.Hadoop
Slave3.Hadoop
```

(2) 第二种

去掉“localhost”，加入集群中所有 Slave 机器的 IP，也是每行一个。

例如：添加形式如下

```
192.168.1.3
192.168.1.4
192.168.1.5
```

原因和添加 “masters” 文件一样，选择第二种方式。

用下面命令进行修改：

```
[root@Master conf]# vim slaves
```

编辑结果如下：

```
[root@Master ~]# more /usr/hadoop/conf/slaves
192.168.1.3
192.168.1.4
192.168.1.5
```

现在在 Master 机器上的 Hadoop 配置就结束了，剩下的就是配置 Slave 机器上的 Hadoop。

一种方式是按照上面的步骤，把 Hadoop 的安装包在用**普通用户 hadoop**通过“**scp**”复制到其他机器的“/home/hadoop”目录下，然后根据实际情况进行安装配置，**除了第6步，那是 Master 特有的**。用下面命令格式进行。（**备注：**此时切换到普通用户 hadoop）

```
scp ~/hadoop-1.0.0.tar.gz hadoop@服务器 IP:~/
```

例如：从“Master.Hadoop”到“Slave1.Hadoop”复制 Hadoop 的安装包。

```
[hadoop@Master ~]$ whoami
hadoop
[hadoop@Master ~]$ ll
总用量 141372
-rw-r--r--. 1 hadoop hadoop 59468784 2月 27 03:31 hadoop-1.0.0.tar.gz
-rw-r--r--. 1 hadoop hadoop 85292206 2月 27 03:29 jdk-6u31-linux-i586.bin
[hadoop@Master ~]$ scp ~/hadoop-1.0.0.tar.gz hadoop@192.168.1.3:~/
```

当前用户为“hadoop”

另一种方式是将 Master 上配置好的 hadoop 所在文件夹“/usr/hadoop”复制到所有的 Slave 的“/usr”目录下（实际上 Slave 机器上的 slavers 文件是不必要的，复制了也没问题）。用下面命令格式进行。（**备注：**此时用户可以为 hadoop 也可以为 root）

```
scp -r /usr/hadoop root@服务器 IP:/usr/
```

例如: 从 “Master.Hadoop” 到 “Slave1.Hadoop” 复制配置 Hadoop 的文件。

```
[hadoop@Master usr]$ su -
密码:
[root@Master ~]# scp -r /usr/hadoop root@192.168.1.3:/usr/
The authenticity of host '192.168.1.3 (192.168.1.3)' can't be established.
RSA key fingerprint is 4b:87:95:c1:89:24:4f:57:1b:02:a2:40:f7:58:9e:a0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.3' (RSA) to the list of known hosts.
root@192.168.1.3's password:
```

上图中以 root 用户进行复制, 当然不管是用户 root 还是 hadoop, 虽然 Master 机器上的 “/usr/hadoop” 文件夹用户 hadoop 有权限, 但是 Slave1 上的 hadoop 用户却没有 “/usr” 权限, 所以没有创建文件夹的权限。所以无论是哪个用户进行拷贝, 右面都是 “root@机器 IP” 格式。因为我们只是建立起了 hadoop 用户的 SSH 无密码连接, 所以用 root 进行 “scp” 时, 仍提示让你输入 “Slave1.Hadoop” 服务器用户 root 的密码。

查看 “Slave1.Hadoop” 服务器的 “/usr” 目录下是否已经存在 “hadoop” 文件夹, 确认已经复制成功。查看结果如下:

```
[root@Slave1 usr]# ll
总用量 104
dr-xr-xr-x.  2 root root 24576 2月 23 06:32 bin
drwxr-xr-x.  2 root root  4096 11月 11 2010 etc
drwxr-xr-x.  2 root root  4096 11月 11 2010 games
drwxr-xr-x. 15 root root  4096 2月 29 06:17 hadoop
drwxr-xr-x. 33 root root  4096 2月 23 04:51 include
drwxr-xr-x.  3 root root  4096 2月 28 05:19 java
dr-xr-xr-x. 53 root root 32768 2月 23 06:32 lib
drwxr-xr-x. 14 root root  4096 2月 23 06:32 libexec
drwxr-xr-x. 11 root root  4096 2月 23 04:45 local
dr-xr-xr-x.  2 root root  4096 2月 24 03:19 sbin
drwxr-xr-x. 95 root root  4096 2月 23 04:53 share
drwxr-xr-x.  4 root root  4096 2月 23 04:45 src
lrwxrwxrwx.  1 root root    10 2月 23 04:45 tmp -> ../var/tmp
[root@Slave1 usr]#
```

从上图中知道, hadoop 文件夹确实已经复制了, 但是我们发现 hadoop 权限是 root, 所以我们要给 “Slave1.Hadoop” 服务器上的用户 hadoop 添加对 “/usr/hadoop” 读权限。

以 root 用户登录 “Slave1.Hadoop”, 执行下面命令。

```
chown -R hadoop:hadoop (用户名: 用户组) hadoop (文件夹)
```

```
[root@Slave1 usr]# ll | grep hadoop
drwxr-xr-x. 15 root root  4096 2月 29 06:17 hadoop
[root@Slave1 usr]# chown -R hadoop:hadoop hadoop
[root@Slave1 usr]# ll | grep hadoop
drwxr-xr-x. 15 hadoop hadoop  4096 2月 29 06:17 hadoop
[root@Slave1 usr]#
```


接着在“Slave1.Hadoop”上修改“/etc/profile”文件（配置 java 环境变量的文件），将以下语句添加到末尾，并使其有效（source /etc/profile）：

```
# set hadoop environment
export HADOOP_HOME=/usr/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

如果不知道怎么设置，可以查看前面“Master.Hadoop”机器的“/etc/profile”文件的配置，到此为止在一台 Slave 机器上的 Hadoop 配置就结束了。剩下的事儿就是照葫芦画瓢把剩余的几台 Slave 机器按照《从“Master.Hadoop”到“Slave1.Hadoop”复制 Hadoop 的安装包。》这个例子进行部署 Hadoop。

4.3 启动及验证

1) 格式化 HDFS 文件系统

在“Master.Hadoop”上使用普通用户 **hadoop** 进行操作。（备注：只需一次，下次启动不再需要格式化，只需 start-all.sh）

```
hadoop namenode -format
```

某些书上和网上的某些资料中用下面命令执行。

```
[hadoop@Master ~]$ bin/hadoop namenode -format
```

我们在看好多文档包括有些书上，按照他们的 hadoop 环境变量进行配置后，并立即使其生效，但是执行发现没有找见“bin/hadoop”这个命令。

```
[hadoop@Master bin]$ bin/hadoop namenode -format
-bash: bin/hadoop: 没有那个文件或目录
[hadoop@Master bin]$
```

其实我们会发现我们的环境变量配置的是“\$HADOOP_HOME/bin”，我们已经把 bin 包含进入了，所以执行时，加上“bin”反而找不到该命令，除非我们的 hadoop 环境变量如下设置。

```
# set hadoop path
export HADOOP_HOME=/usr/hadoop
export PATH=$PATH:$HADOOP_HOME:$HADOOP_HOME/bin
```

这样就能直接使用“bin/hadoop”也可以直接使用“hadoop”，现在不管哪种情况，hadoop 命令都能找见了。我们也没有必要重新在设置 hadoop 环境变量了，只需要记住执行 Hadoop

命令时不需要在前面加“bin”就可以了。

```
[hadoop@Master bin]$ hadoop namenode -format
```

```
[hadoop@Master bin]$ hadoop namenode -format
Warning: $HADOOP_HOME is deprecated.

12/02/29 17:58:31 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = Master.Hadoop/192.168.1.2
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.0.0
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.0 -r 1214675; compiled
by 'hortoninfo' on Thu Dec 15 16:36:35 UTC 2011
*****/
12/02/29 17:58:31 INFO util.GSet: VM type = 32-bit
12/02/29 17:58:31 INFO util.GSet: 2% max memory = 17.77875 MB
12/02/29 17:58:31 INFO util.GSet: capacity = 2^22 = 4194304 entries
12/02/29 17:58:31 INFO util.GSet: recommended=4194304, actual=4194304
12/02/29 17:58:31 INFO namenode.FSNamesystem: fsOwner=hadoop
12/02/29 17:58:31 INFO namenode.FSNamesystem: supergroup=supergroup
12/02/29 17:58:31 INFO namenode.FSNamesystem: isPermissionEnabled=true
12/02/29 17:58:31 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
12/02/29 17:58:31 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s), acc
essTokenLifetime=0 min(s)
12/02/29 17:58:31 INFO namenode.NameNode: Caching file names occurring more than 10 times
12/02/29 17:58:31 INFO common.Storage: Image file of size 112 saved in 0 seconds.
12/02/29 17:58:31 INFO common.Storage: Storage directory /usr/hadoop/tmp/dfs/name has been successfully format
ted.
12/02/29 17:58:31 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at Master.Hadoop/192.168.1.2
*****/
[hadoop@Master bin]$
```

从上图中知道我们已经成功格式化了，但是美中不足就是出现了一个警告，从网上的得知这个警告并不影响 hadoop 执行，但是也有办法解决，详情看后面的“常见问题 FAQ”。

2) 启动 hadoop

在启动前关闭集群中所有机器的防火墙，不然会出现 datanode 开后又自动关闭。

```
service iptables stop
```

使用下面命令启动。

```
start-all.sh
```

```
[hadoop@Master bin]$ start-all.sh
```

执行结果如下：

```
[hadoop@Master ~]$ start-all.sh
starting namenode, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-namenode-Master.Hadoop.out
192.168.1.3: starting datanode, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-datanode-Slave1.Hadoop.out
192.168.1.4: starting datanode, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-datanode-Slave2.Hadoop.out
192.168.1.5: starting datanode, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-datanode-Slave3.Hadoop.out
192.168.1.2: starting secondarynamenode, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-secondarynamenode-Master.Hadoop.out
starting jobtracker, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-jobtracker-Master.Hadoop.out
192.168.1.4: starting tasktracker, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-tasktracker-Slave2.Hadoop.out
192.168.1.5: starting tasktracker, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-tasktracker-Slave3.Hadoop.out
192.168.1.3: starting tasktracker, logging to /usr/hadoop/libexec/./logs/hadoop-hadoop-tasktracker-Slave1.Hadoop.out
[hadoop@Master ~]$
```

可以通过以下启动日志看出，首先启动 namenode 接着启动 datanode1, datanode2, ..., 然后启动 secondarynamenode。再启动 jobtracker, 然后启动 tasktracker1, tasktracker2, ...。

启动 hadoop 成功后，在 Master 中的 tmp 文件夹中生成了 dfs 文件夹，在 Slave 中的 tmp 文件夹中均生成了 dfs 文件夹和 mapred 文件夹。

查看 Master 中 “/usr/hadoop/tmp” 文件夹内容

```
[hadoop@Master ~]$ ll /usr/hadoop/tmp
总用量 4
drwxrwxr-x. 4 hadoop hadoop 4096 2月 29 19:12 dfs
```

查看 Slave1 中 “/usr/hadoop/tmp” 文件夹内容。

```
[hadoop@Slave1 ~]$ ll /usr/hadoop/tmp
总用量 8
drwxrwxr-x. 3 hadoop hadoop 4096 2月 29 18:37 dfs
drwxrwxr-x. 3 hadoop hadoop 4096 2月 29 18:40 mapred
```

3) 验证 hadoop

(1) 验证方法一：用 “jps” 命令

在 Master 上用 java 自带的小工具 **jps** 查看进程。

```
[hadoop@Master ~]$ jps
11077 JobTracker
10797 NameNode
11268 Jps
10982 SecondaryNameNode
[hadoop@Master ~]$
```

在 Slave1 上用 jps 查看进程。

```
[hadoop@Slave1 logs]$ jps
2218 TaskTracker
2135 DataNode
2427 Jps
[hadoop@Slave1 logs]$
```

如果在查看 Slave 机器中发现 “DataNode” 和 “TaskTracker” 没有起来时，先查看一下日志的，如果是 “namespaceID” 不一致问题，采用 “常见问题 FAQ6.2” 进行解决，如果是 “No route to host” 问题，采用 “常见问题 FAQ6.3” 进行解决。

(2) 验证方式二：用 “hadoop dfsadmin -report”

用这个命令可以查看 Hadoop 集群的状态。

Master 服务器的状态:

```
[hadoop@Master ~]$ hadoop dfsadmin -report
Configured Capacity: 158534062080 (147.65 GB)
Present Capacity: 144843030528 (134.9 GB)
DFS Remaining: 144842932224 (134.9 GB)
DFS Used: 98304 (96 KB)
DFS Used%: 0%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
```

Slave 服务器的状态

```
Datanodes available: 3 (3 total, 0 dead)

Name: 192.168.1.4:50010
Decommission Status : Normal
Configured Capacity: 52844687360 (49.22 GB)
DFS Used: 28672 (28 KB)
Non DFS Used: 4570701824 (4.26 GB)
DFS Remaining: 48273956864 (44.96 GB)
DFS Used%: 0%
DFS Remaining%: 91.35%
Last contact: Wed Feb 29 23:49:47 CST 2012

Name: 192.168.1.5:50010
Decommission Status : Normal
Configured Capacity: 52844687360 (49.22 GB)
DFS Used: 28672 (28 KB)
Non DFS Used: 4546277376 (4.23 GB)
DFS Remaining: 48298381312 (44.98 GB)
DFS Used%: 0%
DFS Remaining%: 91.4%
Last contact: Wed Feb 29 23:49:47 CST 2012

Name: 192.168.1.3:50010
Decommission Status : Normal
Configured Capacity: 52844687360 (49.22 GB)
DFS Used: 40960 (40 KB)
Non DFS Used: 4574052352 (4.26 GB)
DFS Remaining: 48270594048 (44.96 GB)
DFS Used%: 0%
DFS Remaining%: 91.34%
Last contact: Wed Feb 29 23:49:49 CST 2012
```

4.4 网页查看集群

1) 访问 “<http://192.168.1.2:50030>”

Master Hadoop Map/Reduce Administration

State: RUNNING
Started: Wed Feb 29 22:13:25 CST 2012
Version: 1.0.0, r1214675
Compiled: Thu Dec 15 16:36:35 UTC 2011 by hortonfo
Identifier: 201202292213

Cluster Summary (Heap Size is 56.5 MB/888.94 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	0	3	0	0	0	0	6	6	4.00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (JobId, Priority, User, Name)

Examples: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Retired Jobs

none

2) 访问 “<http://192.168.1.2:50070>”

NameNode 'Master.Hadoop:9000'

Started: Wed Feb 29 22:13:22 CST 2012
Version: 1.0.0, r1214675
Compiled: Thu Dec 15 16:36:35 UTC 2011 by hortonfo
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary

7 files and directories, 1 blocks = 8 total. Heap Size is 57.5 MB / 888.94 MB (6%)
Configured Capacity : 147.65 GB
DFS Used : 96 KB
Non DFS Used : 12.75 GB
DFS Remaining : 134.89 GB
DFS Used% : 0 %
DFS Remaining% : 91.36 %
[Live Nodes](#) : 3
[Dead Nodes](#) : 0
[Decommissioning Nodes](#) : 0
Number of Under-Replicated Blocks : 0

NameNode Storage:

Storage Directory	Type	State
/usr/hadoop/tmp/dfs/name	IMAGE_AND_EDITS	Active

This is [Apache Hadoop](#) release 1.0.0

5、常见问题FAQ

5.1 关于 Warning: \$HADOOP_HOME is deprecated.

hadoop 1.0.0 版本，安装完之后敲入 hadoop 命令时，老是提示这个警告：

Warning: \$HADOOP_HOME is deprecated.

经查 `hadoop-1.0.0/bin/hadoop` 脚本和 “`hadoop-config.sh`” 脚本，发现脚本中对 `HADOOP_HOME` 的环境变量设置做了判断，笔者的环境根本不需要设置 `HADOOP_HOME` 环境变量。

解决方案一：编辑 “`/etc/profile`” 文件，去掉 `HADOOP_HOME` 的变量设定，重新输入 `hadoop fs` 命令，警告消失。

解决方案二：编辑 “`/etc/profile`” 文件，添加一个环境变量，之后警告消失：

```
export HADOOP_HOME_WARN_SUPPRESS=1
```

解决方案三：编辑 “`hadoop-config.sh`” 文件，把下面的 “`if - fi`” 功能注释掉。

```
if [ "$HADOOP_HOME_WARN_SUPPRESS" == "" ] && [ "$HADOOP_HOME" != "" ]; then
    echo "Warning: \$HADOOP_HOME is deprecated." 1>&2
    echo 1>&2
fi
```

我们这里本着不动 Hadoop 原配置文件的前提下，采用 “**方案二**”，在 “`/etc/profile`” 文件添加上面内容，并用命令 “`source /etc/profile`” 使之有效。

1) 切换至 root 用户

```
[hadoop@Master bin]$ su -
密码:
[root@Master ~]# vim /etc/profile
```

2) 添加内容

```
# set java environment
export JAVA_HOME=/usr/java/jdk1.6.0_31
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin

# set hadoop path
export HADOOP_HOME=/usr/hadoop
export HADOOP_HOME_WARN_SUPPRESS=1
export PATH=$PATH:$HADOOP_HOME/bin
```

3) 重新生效

```
[root@Master ~]# source /etc/profile
```

5.2 解决 “no datanode to stop” 问题

当我停止 Hadoop 时发现如下信息：


```
[hadoop@Master ~]$ stop-all.sh
stopping jobtracker
192.168.1.5: no tasktracker to stop
192.168.1.4: no tasktracker to stop
192.168.1.3: no tasktracker to stop
stopping namenode
192.168.1.3: no datanode to stop
192.168.1.4: no datanode to stop
192.168.1.5: no datanode to stop
192.168.1.2: stopping secondarynamenode
```

原因：每次 namenode format 会重新创建一个 namenodeId，而 tmp/dfs/data 下包含了上次 format 下的 id，namenode format 清空了 namenode 下的数据，但是没有清空 datanode 下的数据，导致启动时失败，所要做的就是每次 format 前，清空 tmp 下的所有目录。

第一种解决方案如下：

1) 先删除 “/usr/hadoop/tmp”

```
rm -rf /usr/hadoop/tmp
```

2) 创建 “/usr/hadoop/tmp” 文件夹

```
mkdir /usr/hadoop/tmp
```

3) 删除 “/tmp” 下以 “hadoop” 开头文件

```
rm -rf /tmp/hadoop*
```

4) 重新格式化 hadoop

```
hadoop namenode -format
```

5) 启动 hadoop

```
start-all.sh
```

使用第一种方案，有种不好处就是原来集群上的重要数据全没有了。假如说 Hadoop 集群已经运行了一段时间。建议采用第二种。

第二种方案如下：

1) 修改每个 Slave 的 namespaceID 使其与 Master 的 namespaceID 一致。

或者

2) 修改 Master 的 namespaceID 使其与 Slave 的 namespaceID 一致。

该 “namespaceID” 位于 “/usr/hadoop/tmp/dfs/data/current/VERSION” 文件中，前面蓝色的可能根据实际情况变化，但后面红色是不变的。

例如：查看“Master”下的“**VERSION**”文件

```
[hadoop@Master current]$ more VERSION
#Wed Feb 29 22:18:29 CST 2012
namespaceID=1432575221
cTime=0
storageType=NAME_NODE
layoutVersion=-32
```

本人建议采用**第二种**，这样方便快捷，而且还能防止误删。

5.3 Slave服务器中datanode启动后又自动关闭

查看日志发下如下错误。

ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: java.io.IOException: Call to ...
failed on local exception: java.net.NoRouteToHostException: **No route to host**

解决方案是：关闭防火墙

```
service iptables stop
```

5.4 从本地往hdfs文件系统上传文件

出现如下错误：

INFO hdfs.DFSCClient: Exception in createBlockOutputStream java.io.IOException: **Bad connect ack with firstBadLink**

INFO hdfs.DFSCClient: Abandoning block blk_-1300529705803292651_37023

WARN hdfs.DFSCClient: DataStreamer Exception: java.io.IOException: **Unable to create new block.**

解决方案是：

1) 关闭防火墙

```
service iptables stop
```

2) 禁用 selinux

编辑“/etc/selinux/config”文件，设置“**SELINUX=disabled**”

5.5 安全模式导致的错误

出现如下错误：

org.apache.hadoop dfs.SafeModeException: **Cannot delete ..., Name node is in safe mode**

在分布式文件系统启动的时候，开始的时候会有安全模式，当分布式文件系统处于安全模式的情况下，文件系统中的内容不允许修改也不允许删除，直到安全模式结束。安全模式主要是为了系统启动的时候检查各个 DataNode 上数据块的有效性，同时根据策略必要的复制或者删除部分数据块。运行期通过命令也可以进入安全模式。在实践过程中，系统启动的时候去修改和删除文件也会有安全模式不允许修改的出错提示，只需要等待一会儿即可。

解决方案是：关闭安全模式

```
hadoop dfsadmin -safemode leave
```

5.6 解决 Exceeded MAX_FAILED_UNIQUE_FETCHES

出现错误如下：

Shuffle Error: Exceeded MAX_FAILED_UNIQUE_FETCHES; bailing-out

程序里面需要打开多个文件，进行分析，系统一般默认数量是 1024，（用 `ulimit -a` 可以看到）对于正常使用是够了，但是对于程序来讲，就太少了。

解决方案是：修改 2 个文件。

1) “[/etc/security/limits.conf](#)”

```
vim /etc/security/limits.conf
```

加上：

```
soft nfile 102400
hard nfile 409600
```

2) “[/etc/pam.d/login](#)”

```
vim /etc/pam.d/login
```

添加：

```
session required /lib/security/pam_limits.so
```

针对第一个问题我纠正下答案：

这是 reduce 预处理阶段 shuffle 时获取已完成的 map 的输出失败次数超过上限造成的，上限默认为 5。引起此问题的方式可能会有很多种，比如网络连接不正常，连接超时，带宽较差以及端口阻塞等。通常框架内网络情况较好是不会出现此错误的。

5.7 解决 “Too many fetch-failures”

出现这个问题主要是结点间的连通不够全面。

解决方案是：

1) 检查 “/etc/hosts”

要求本机 ip 对应 服务器名

要求要包含所有的服务器 ip +服务器名

2) 检查 “.ssh/authorized_keys”

要求包含所有服务器（包括其自身）的 public key

5.8 处理速度特别的慢

出现 **map** 很快，但是 **reduce** 很慢，而且反复出现 “**reduce=0%**”。

解决方案如下：

结合解决方案 5.7，然后

修改 “conf/hadoop-env.sh” 中的 “export HADOOP_HEAPSIZE=4000”

5.9 解决hadoop OutOfMemoryError问题

出现这种异常，明显是 jvm 内存不够得原因。

解决方案如下：要修改所有的 datanode 的 jvm 内存大小。

```
Java -Xms 1024m -Xmx 4096m
```

一般 jvm 的最大内存使用应该为总内存大小的一半，我们使用的 8G 内存，所以设置为 4096m，这一值可能依旧不是最优的值。

5.10 Namenode in safe mode

解决方案如下：

```
bin/hadoop dfsadmin -safemode leave
```

5.11 IO写操作出现问题

```

0-1246359584298, infoPort=50075, ipcPort=50020):Got exception while serving
blk_-5911099437886836280_1292 to /172.16.100.165:
java.net.SocketTimeoutException: 480000 millis timeout while waiting for channel to be ready for write. ch :
java.nio.channels.SocketChannel[connected local=/
172.16.100.165:50010 remote=/172.16.100.165:50930]
    at org.apache.hadoop.net.SocketIOWithTimeout.waitForIO(SocketIOWithTimeout.java:185)
    at org.apache.hadoop.net.SocketOutputStream.waitForWritable(SocketOutputStream.java:159)
    .....

```

It seems there are many reasons that it can timeout, the example given in HADOOP-3831 is a slow reading client.

解决方案如下：

在 `hadoop-site.xml` 中设置 `dfs.datanode.socket.write.timeout=0`

5.12 status of 255 error

错误类型：

`java.io.IOException: Task process exit with nonzero status of 255.`

`at org.apache.hadoop.mapred.TaskRunner.run(TaskRunner.java:424)`

错误原因：

Set `mapred.jobtracker.retirejob.interval` and `mapred.userlog.retain.hours` to higher value. By default, their values are 24 hours. These might be the reason for failure, though I'm not sure restart.

解决方案如下：单个 `datanode`

如果一个 `datanode` 出现问题，解决之后需要重新加入 `cluster` 而不重启 `cluster`，方法如下：

```
bin/hadoop-daemon.sh start datanode
bin/hadoop-daemon.sh start jobtracker
```

6、用到的Linux命令

6.1 chmod命令详解

使用权限：所有使用者

使用方式：`chmod [-cfvR] [--help] [--version] mode file...`

说明：

Linux/Unix 的档案存取权限分为三级：档案拥有者、群组、其他。利用 `chmod` 可以藉以控制档案如何被他人所存取。

mode：权限设定字串，格式如下：`[ugoa...][[+|=][rwxX]...][...]`，其中 `u` 表示该档案的拥有者，`g` 表示与该档案的拥有者属于同一个群体(group)者，`o` 表示其他以外的人，`a` 表示这三者皆是。

`+` 表示增加权限、`-` 表示取消权限、`=` 表示唯一设定权限。

`r` 表示可读取，`w` 表示可写入，`x` 表示可执行，`X` 表示只有当该档案是个子目录或者该档案已经被设定过为可执行。

`-c`：若该档案权限确实已经更改，才显示其更改动作

`-f`：若该档案权限无法被更改也不要显示错误讯息

`-v`：显示权限变更的详细资料

`-R`：对目前目录下的所有档案与子目录进行相同的权限变更(即以递归的方式逐个变更)

```
--help : 显示辅助说明  
--version : 显示版本
```

范例：

将档案 file1.txt 设为所有人皆可读取

```
chmod ugo+r file1.txt
```

将档案 file1.txt 设为所有人皆可读取

```
chmod a+r file1.txt
```

将档案 file1.txt 与 file2.txt 设为该档案拥有者，与其所属同一个群体者可写入，但其他以外的人则不可写入

```
chmod ug+w,o-w file1.txt file2.txt
```

将 ex1.py 设定为只有该档案拥有者可以执行

```
chmod u+x ex1.py
```

将目前目录下的所有档案与子目录皆设为任何人可读取

```
chmod -R a+r *
```

此外 chmod 也可以用数字来表示权限如 `chmod 777 file`

语法为： `chmod abc file`

其中 a,b,c 各为一个数字，分别表示 User、Group、及 Other 的权限。

r=4, w=2, x=1

若要 rwx 属性则 $4+2+1=7$;

若要 rw-属性则 $4+2=6$;

若要 r-x 属性则 $4+1=5$ 。

范例：

`chmod a=rwx file` 和 `chmod 777 file` 效果相同

`chmod ug=rwx,o=x file` 和 `chmod 771 file` 效果相同

若用 `chmod 4755 filename` 可使此程式具有 **root** 的权限

6.2 chown命令详解

使用权限： root

使用方式： `chown [-cfhvR] [--help] [--version] user[:group] file...`

说明：

Linux/Unix 是多人多工作业系统，所有的档案皆有拥有者。利用 `chown` 可以将档案的拥有者加以改变。一般来说，这个指令只有是由系统管理者(root)所使用，一般使用者没有权限可以改变别人的档案拥有者，也没有权限可以自己的档案拥有者改设为别人。只有系统管理者(root)才有这样的权限。

```
user：新的档案拥有者的使用者
IDgroup：新的档案拥有者的使用者群体(group)
-c：若该档案拥有者确实已经更改，才显示其更改动作
-f：若该档案拥有者无法被更改也不要显示错误讯息
-h：只对于连结(link)进行变更，而非该 link 真正指向的档案
-v：显示拥有者变更的详细资料
-R：对目前目录下的所有档案与子目录进行相同的拥有者变更(即以递归的方式逐个变更)
--help：显示辅助说明
--version：显示版本
```

范例：

将档案 `file1.txt` 的拥有者设为 `users` 群体的使用者 `jessie`

```
chown jessie:users file1.txt
```

将目前目录下的所有档案与子目录的拥有者皆设为 `users` 群体的使用者 `lamport`

```
chown -R lamport:users *
```

```
-rw----- (600) -- 只有属主有读写权限。
-rw-r--r-- (644) -- 只有属主有读写权限；而属组用户和其他用户只有读权限。
-rwx----- (700) -- 只有属主有读、写、执行权限。
-rwxr-xr-x (755) -- 属主有读、写、执行权限；而属组用户和其他用户只有读、执行权限。
-rwx--x--x (711) -- 属主有读、写、执行权限；而属组用户和其他用户只有执行权限。
-rw-rw-rw- (666) -- 所有用户都有文件读、写权限。这种做法不可取。
-rwxrwxrwx (777) -- 所有用户都有读、写、执行权限。更不可取的做法。
以下是对目录的两个普通设定：
drwx----- (700) - 只有属主可在目录中读、写。
drwxr-xr-x (755) - 所有用户可读该目录，但只有属主才能改变目录中的内容
suid 的代表数字是 4，比如 4755 的结果是-rwsr-xr-x
sgid 的代表数字是 2，比如 6755 的结果是-rwsr-sr-x
sticky 位代表数字是 1，比如 7755 的结果是-rwsr-sr-t
```

6.3 scp命令详解

`scp` 是 `secure copy` 的缩写，`scp` 是 `linux` 系统下基于 `ssh` 登陆进行安全的远程文件拷贝命令。`linux` 的 `scp` 命令可以在 `linux` 服务器之间复制文件和目录。

scp 命令的用处：

scp 在网络上不同的主机之间复制文件，它使用 ssh 安全协议传输数据，具有和 ssh 一样的验证机制，从而安全的远程拷贝文件。

scp 命令基本格式：

```
scp [-1246BCpqr] [-c cipher] [-F ssh_config] [-i identity_file]
[-l limit] [-o ssh_option] [-P port] [-S program]
[[user@]host1:]file1 [...] [[user@]host2:]file2
```

scp 命令的参数说明：

- 1 强制 scp 命令使用协议 ssh1
- 2 强制 scp 命令使用协议 ssh2
- 4 强制 scp 命令只使用 IPv4 寻址
- 6 强制 scp 命令只使用 IPv6 寻址
- B 使用批处理模式（传输过程中不询问传输口令或短语）
- C 允许压缩。（将-C 标志传递给 ssh，从而打开压缩功能）
- p 保留原文件的修改时间，访问时间和访问权限。
- q 不显示传输进度条。
- r 递归复制整个目录。
- v 详细方式显示输出。scp 和 ssh(1)会显示出整个过程的调试信息。这些信息用于调试连接，验证和配置问题。
- c cipher 以 cipher 将数据传输进行加密，这个选项将直接传递给 ssh。
- F ssh_config 指定一个替代的 ssh 配置文件，此参数直接传递给 ssh。
- i identity_file 从指定文件中读取传输时使用的密钥文件，此参数直接传递给 ssh。
- l limit 限定用户所能使用的带宽，以 Kbit/s 为单位。
- o ssh_option 如果习惯于使用 ssh_config(5)中的参数传递方式，
- P port 注意是大写的 P, port 是指定数据传输用到的端口号
- S program 指定加密传输时所使用的程序。此程序必须能够理解 ssh(1)的选项。

scp 命令的实际应用

1) 从本地服务器复制到远程服务器

(1) 复制文件：

命令格式：

```
scp local_file remote_username@remote_ip:remote_folder
```

或者

```
scp local_file remote_username@remote_ip:remote_file
```

或者

```
scp local_file remote_ip:remote_folder
```

或者

```
scp local_file remote_ip:remote_file
```

第 1,2 个指定了用户名，命令执行后需要输入用户密码，第 1 个仅指定了远程的目录，文件名字不变，第 2 个指定了文件名

第 3,4 个没有指定用户名，命令执行后需要输入用户名和密码，第 3 个仅指定了远程的目录，文件名字不变，第 4 个指定了文件名

实例：

```
scp /home/linux/soft/scp.zip root@www.mydomain.com:/home/linux/others/soft
scp /home/linux/soft/scp.zip root@www.mydomain.com:/home/linux/others/soft/scp2.zip
scp /home/linux/soft/scp.zip www.mydomain.com:/home/linux/others/soft
scp /home/linux/soft/scp.zip www.mydomain.com:/home/linux/others/soft/scp2.zip
```

(2) 复制目录：

命令格式：

```
scp -r local_folder remote_username@remote_ip:remote_folder
```

或者

```
scp -r local_folder remote_ip:remote_folder
```

第 1 个指定了用户名，命令执行后需要输入用户密码；

第 2 个没有指定用户名，命令执行后需要输入用户名和密码；

例子：

```
scp -r /home/linux/soft/ root@www.mydomain.com:/home/linux/others/
scp -r /home/linux/soft/ www.mydomain.com:/home/linux/others/
```

上面 命令 将 本地 soft 目录 复制 到 远程 others 目录下，即复制后远程服务器上会有/home/linux/others/soft/ 目录。

2) 从远程服务器复制到本地服务器

从远程复制到本地的 scp 命令与上面的命令雷同，只要将从本地复制到远程的命令后面 2 个参数互换顺序就行了。

例如：

```
scp root@www.mydomain.com:/home/linux/soft/scp.zip /home/linux/others/scp.zip
scp www.mydomain.com:/home/linux/soft/ -r /home/linux/others/
```

linux 系统下 scp 命令中很多参数都和 ssh1 有关，还需要看到更原汁原味的参数信息，可以运行 man scp 看到更细致的英文说明。

参考文献

感谢以下文章的编作者，没有你们的铺路，我或许会走得很艰难，参考不分先后，贡献同等珍贵。

【1】Hadoop 实战——陆嘉恒——机械工业出版社

【2】实战 Hadoop——刘鹏——电子工业出版社

【3】Ubuntu11.10 下安装 Hadoop1.0.0（单机伪分布式）

地址：<http://my.oschina.net/se77en/blog/38804>

【4】hadoop1.0.0 安装记录

地址：<http://blog.csdn.net/ylqmf/article/details/7250235>

【5】Hadoop 集群安装详细步骤

地址：<http://www.yanjiuyanjiu.com/2012/01/03/hadoop-cluster-setup/>

【6】在 LINUX 下安装 JDK1.6

地址：<http://wenku.baidu.com/view/cf91f9d2240c844769eaeef3.html>

【7】Linux AS 安装 JDK 1.6

地址：<http://www.iteye.com/topic/421608>

【8】linux 安装 JDK

地址：<http://wenku.baidu.com/view/efe0c100cc17552707220821.html>

【9】CentOS 修改主机名

地址：<http://now-code.com/archives/233>

【10】CentOS(RedHat)命令行修改主机名(主机别名)

地址：<http://www.slyar.com/blog/centos-linux-hostname.html>

【11】CENTOS 修改主机名

地址：http://blog.csdn.net/forest_boy/article/details/5636696

【12】CentOS 之 SSH 安装与配置

地址：<http://apps.hi.baidu.com/share/detail/24759120>

【13】OpenSSH 无密码登陆

地址：<http://blog.csdn.net/jiedushi/article/details/6672894>

【14】Hadoop 使用常见问题以及解决方法

地址：<http://wenku.baidu.com/view/3b13d527a5e9856a56126029.html>

【15】No route to host 问题的解决

地址：<http://blog.csdn.net/shirdrn/article/details/7280040>

【16】hadoop 常见错误及处理方法

地址：<http://samwalt.iteye.com/blog/1099348>

【17】Hadoop 中常出现的错误以及解决方法

地址：http://blog.sina.com.cn/s/blog_759444350100t2r5.html

【18】安装出现的几种异常的处理方法

地址：<http://hi.baidu.com/xixitie/blog/item/a32f6913cacefb145aaf53dd.html>