**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Salve。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Salve，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 |
| Salve1.Hadoop | 192.168.1.3 |
| Salve2.Hadoop | 192.168.1.4 |
| Salve3.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Salve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。其实应该还应该有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**

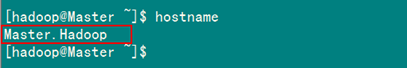
　　Hadoop集群要按照**1.2小节**表格所示进行配置，我们在"**Hadoop集群（第1期）**"的CentOS6.0安装过程就按照提前规划好的主机名进行安装和配置。如果实验室后来人在安装系统时，没有配置好，不要紧，没有必要重新安装，在安装完系统之后仍然可以根据后来的规划对机器的主机名进行修改。

　　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



　　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

201205161427002717.png

　　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：



**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Ethemet）

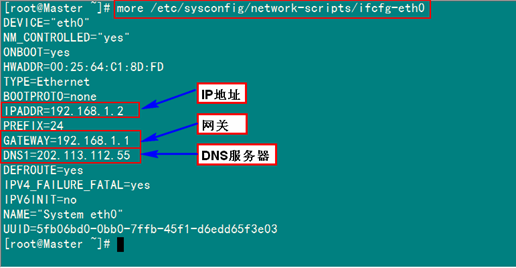
NETMASK 网络掩码

**IPADDR** IP地址

IPV6INIT IPV6是否有效（yes/no）

GATEWAY 默认网关IP地址

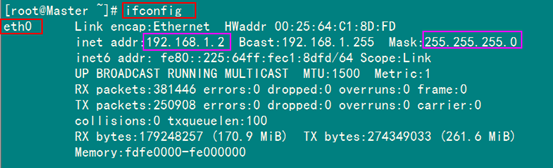
　　查看"/etc/sysconfig/network-scripts/ifcfg-eth0"内容，如果IP不复核，就行修改。



　　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/ifcgf-eth0

　　修改完之后可以用"ifconfig"进行查看。

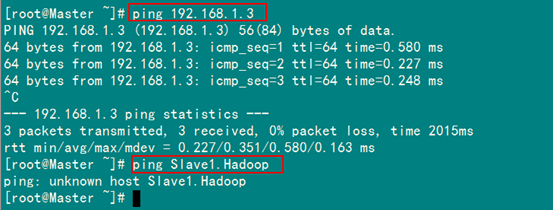


**4）配置hosts文件（必须）**

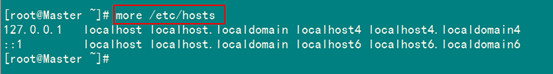
　　"**/etc/hosts**"这个文件是用来配置主机将用的**DNS**服务器信息，是记载LAN内接续的各主机的对应[HostName和IP]用的。当用户在进行网络连接时，首先查找该文件，寻找对应主机名（或域名）对应的IP地址。

　　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"Salve1.Hadoop:192.168.1.3"用命令"ping"记性连接测试。测试结果如下：



　　 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。



　　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

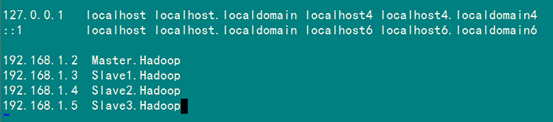
192.168.1.5 Slave3.Hadoop

用以下命令进行添加：

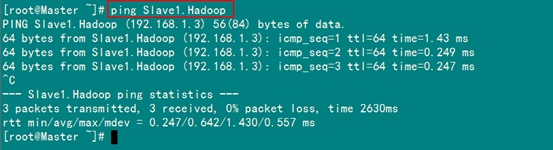
vim /etc/hosts

201205161427037403.png

　　添加结果如下：



　　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。



　　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

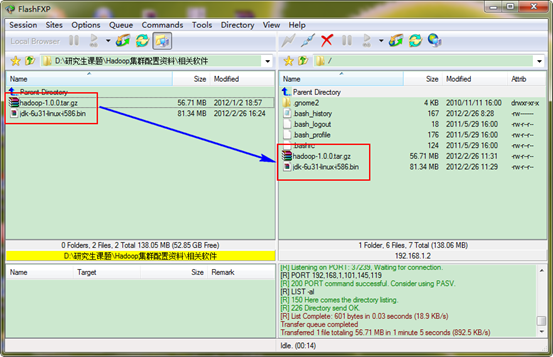
**2）Hadoop软件**

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

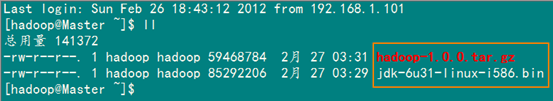
    Hadoop版本：hadoop-1.0.0.tar.gz

**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一下是否已经上传了。



    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**

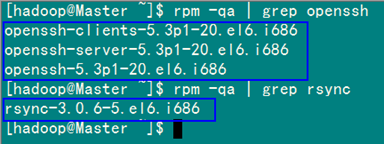
　　Hadoop运行过程中需要管理远端Hadoop守护进程，在Hadoop启动以后，NameNode是通过SSH（Secure Shell）来启动和停止各个DataNode上的各种守护进程的。这就必须在节点之间执行指令的时候是不需要输入密码的形式，故我们需要配置SSH运用无密码公钥认证的形式，这样NameNode使用SSH无密码登录并启动DataName进程，同样原理，DataNode上也能使用SSH无密码登录到NameNode。

**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



**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Salve**

**1）SSH无密码原理**

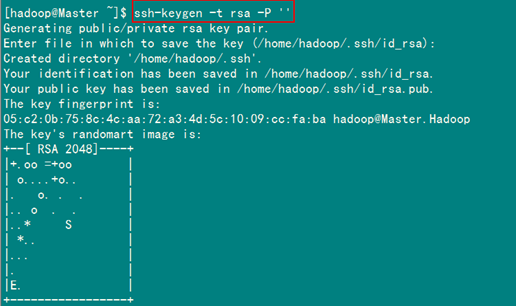
　　Master（NameNode | JobTracker）作为客户端，要实现无密码公钥认证，连接到服务器Salve（DataNode | Tasktracker）上时，需要在Master上生成一个密钥对，包括一个公钥和一个私钥，而后将公钥复制到所有的Slave上。当Master通过SSH连接Salve时，Salve就会生成一个随机数并用Master的公钥对随机数进行加密，并发送给Master。Master收到加密数之后再用私钥解密，并将解密数回传给Slave，Slave确认解密数无误之后就允许Master进行连接了。这就是一个公钥认证过程，其间不需要用户手工输入密码。重要过程是将客户端Master复制到Slave上。

**2）Master机器上生成密码对**

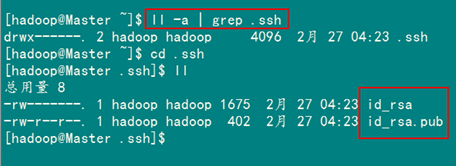
　　 在Master节点上执行以下命令：

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。



　　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。



　　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。

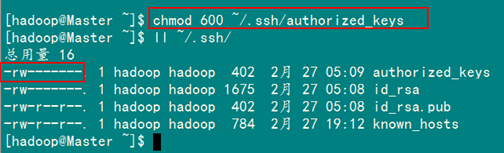
cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys



　　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



**备注：**如果不进行设置，在验证时，扔提示你输入密码，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。

201205161427114593.png

RSAAuthentication yes # 启用 RSA 认证

PubkeyAuthentication yes # 启用公钥私钥配对认证方式

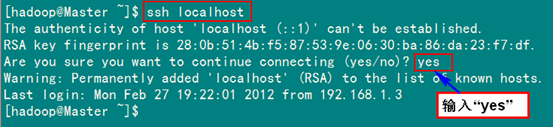
AuthorizedKeysFile .ssh/authorized\_keys # 公钥文件路径（和上面生成的文件同）

　　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



　　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:~/

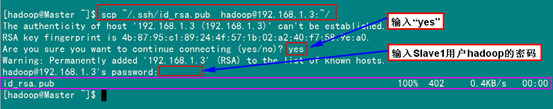
　　例如：

scp ~/.ssh/id\_rsa.pub hadoop@192.168.1.3:~/

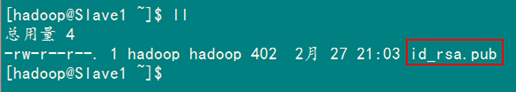
　　上面的命令是**复制**文件"**id\_rsa.pub**"到服务器IP为"**192.168.1.3**"的用户为"**hadoop**"的"**/home/hadoop/**"下面。

　　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**



　　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。



　　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**

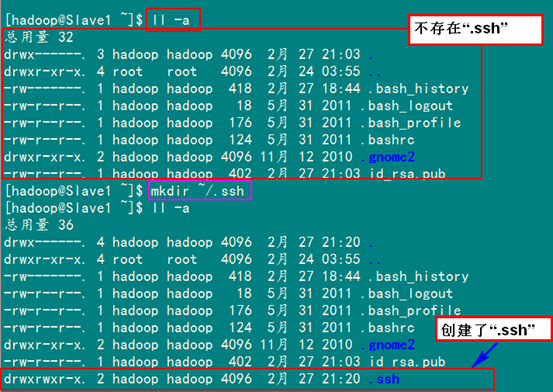
　　这一步**并不是必须**的，如果在Slave1.Hadoop的"/home/hadoop"**已经存在**就不需要创建了，因为我们之前并没有对Slave机器做过无密码登录配置，所以该文件是不存在的。用下面命令进行创建。（**备注：**用hadoop登录系统，如果不涉及系统文件修改，一般情况下都是用我们之前建立的普通用户hadoop进行执行命令。）

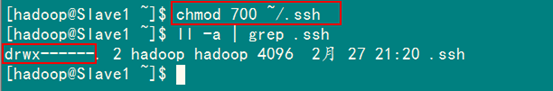
mkdir ~/.ssh

　　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。





　　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



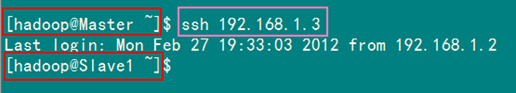
**4）用root用户修改"/etc/ssh/sshd\_config"**

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。

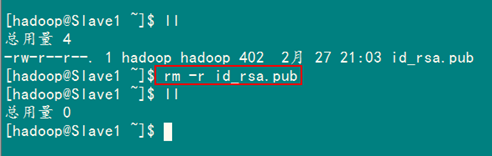
ssh 远程服务器IP



　　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



　　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**

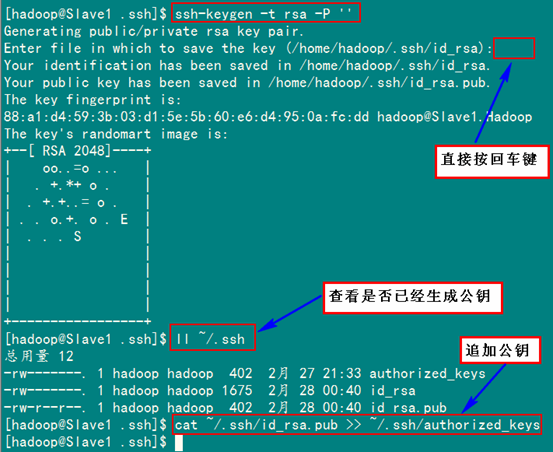
　　和Master无密码登录所有Slave原理一样，就是把Slave的公钥**追加**到Master的".ssh"文件夹下的"authorized\_keys"中，记得是**追加（>>）**。

　　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

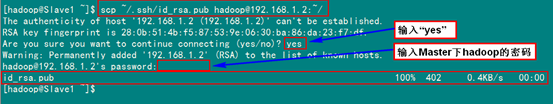


　　接着是用命令"**scp**"复制"Slave1.Hadoop"的公钥"id\_rsa.pub"到"Master.Hadoop"的"/home/hadoop/"目录下，并**追加**到"Master.Hadoop"的"authorized\_keys"中。

**1）在"Slave1.Hadoop"服务器的操作**

　　用到的命令如下：

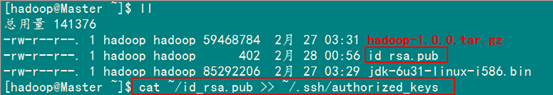
scp ~/.ssh/id\_rsa.pub     hadoop@192.168.1.2:~/



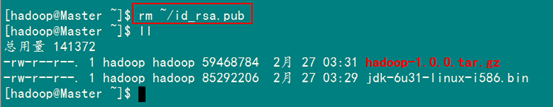
**2）在"Master.Hadoop"服务器的操作**

　　用到的命令如下：

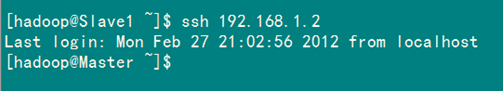
cat ~/id\_rsa.pub >> ~/.ssh/authorized\_keys



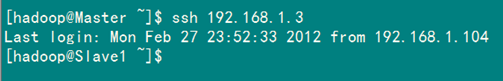
　　然后删除掉刚才复制过来的"id\_rsa.pub"文件。



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



　　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。



　　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**

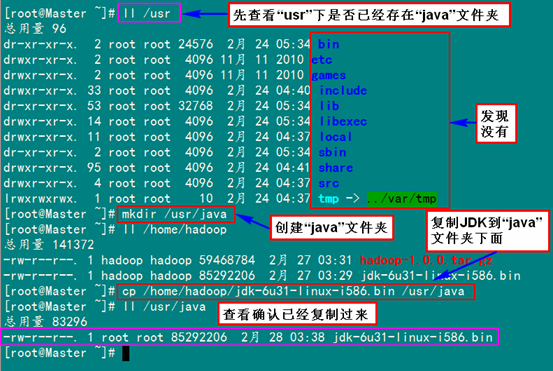
　　所有的机器上都要安装JDK，现在就先在Master服务器安装，然后其他服务器按照步骤重复进行即可。安装JDK以及配置环境变量，需要以"**root**"的身份进行。

**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

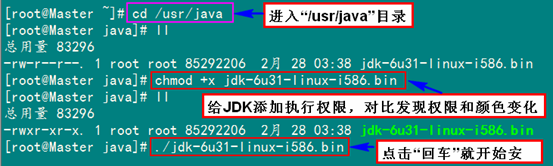
cp /home/hadoop/ jdk-6u31-linux-i586.bin /usr/java



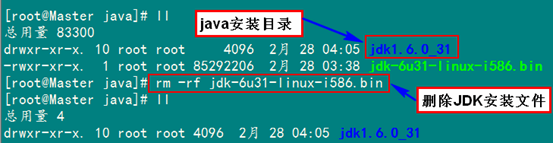
　　接着**进入**"**/usr/java**"目录**下**通过下面命令使其JDK获得可执行权限，并安装JDK。

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

./jdk-6u31-linux-i586.bin



　　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。



**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

201205161427277054.png

**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

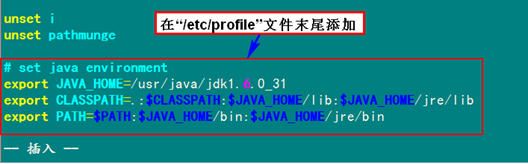
# set java environment

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。



**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

201205161427296822.png

**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



　　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:~/

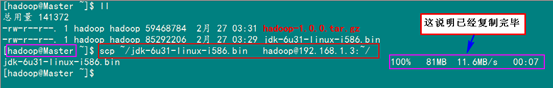
　　或者

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

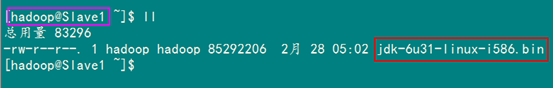
**备注：**"**~**"代表**当前**用户的主目录，当**前用户为hadoop**，所以"**~**"代表"**/home/hadoop**"。

**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:~/



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



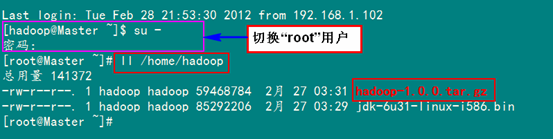
　　从上图中得知，我们已经成功复制了，现在我们就用**最高权限用户root**进行安装了。其他的与这个一样。

**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。

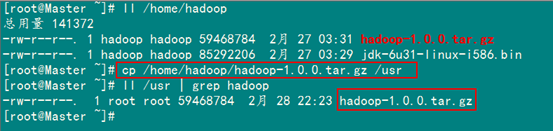
**4.1 安装hadoop**

　　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**"。



　　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



　　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。

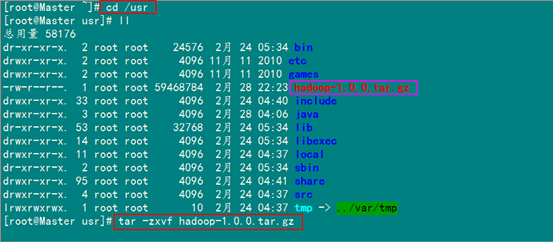
cd /usr #进入"/usr"目录

tar –zxvf hadoop-1.0.0.tar.gz #解压"**hadoop-1.0.0.tar.gz**"安装包

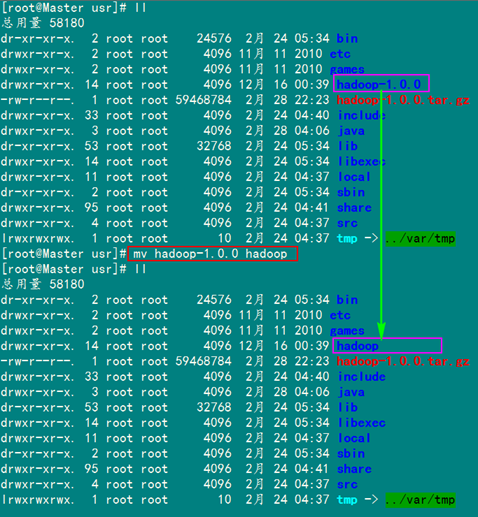
mv hadoop-1.0.0 hadoop #将"hadoop-1.0.0"文件夹**重命名**"hadoop"

chown **–R** hadoop:hadoop hadoop #**将文件夹"hadoop"读权限分配给hadoop用户**

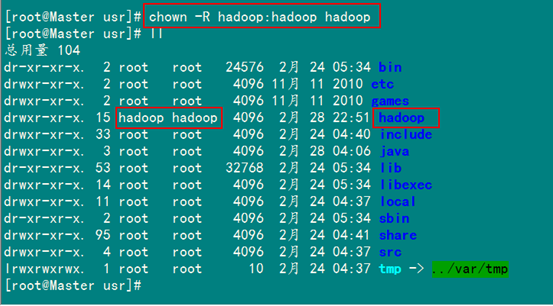
rm –rf hadoop-1.0.0.tar.gz                 #删除"**hadoop-1.0.0.tar.gz**"安装包



　　解压后，并重命名。



　　把"/usr/hadoop"**读权**限分配给**hadoop**用户（**非常重要**）



　　删除"**hadoop-1.0.0.tar.gz**"安装包

201205161427360.png

　　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

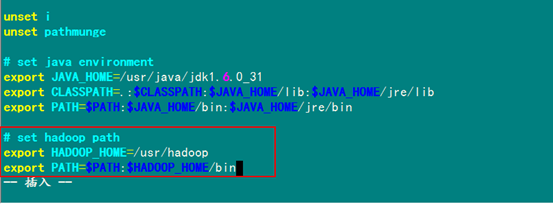


**2）配置"/etc/profile"**

vim /etc/profile

201205161427372126.png

　　配置后的文件如下：



**3）重启"/etc/profile"**

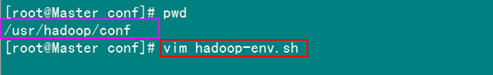
source /etc/profile

201205161427387260.png

**4.2 配置hadoop**

**1）配置hadoop-env.sh**

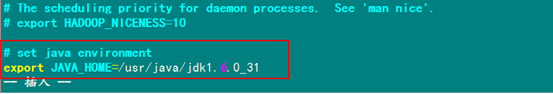
　　该"**hadoop-env.sh**"文件位于"**/usr/hadoop/conf**"目录下。



　　在文件的末尾添加下面内容。

# set java environment

export JAVA\_HOME=/usr/java/jdk1.6.0\_31



　　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>

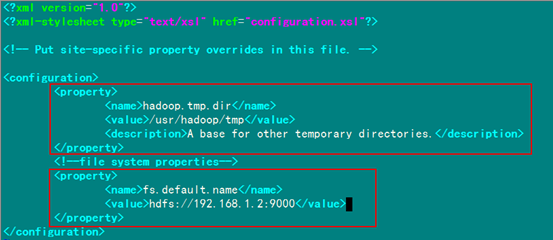
</configuration>

**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：

201205161427393258.png

    编辑结果显示如下：



**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)

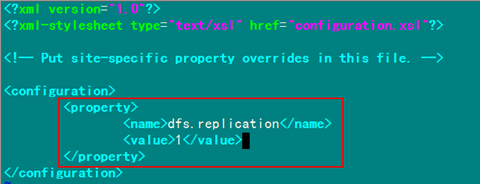
    </property>

<configuration>

　　用下面命令进行编辑：

201205161427403158.png

　　编辑结果显示如下：



**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：

201205161427414488.png

　　编辑结果显示如下：



**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：

201205161427412470.png

　　编辑结果显示如下：

201205161427423484.png

**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

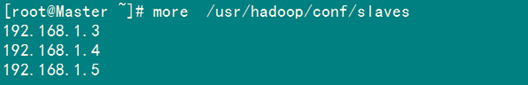
192.168.1.5

　　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：

201205161427424879.png

　　编辑结果如下：



　　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。



**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。



    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，扔提示让你输入"Slave1.Hadoop"服务器用户root的密码。

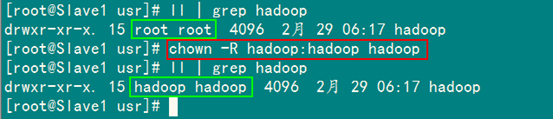
    查看"Slave1.Hadoop"服务器的"/usr"目录下是否已经存在"hadoop"文件夹，确认已经复制成功。查看结果如下：



　　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，所以我们现在要给"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**）



　　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，到此为此在一台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。

201205161427451517.png

　　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。



　　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坏境变量如下设置。

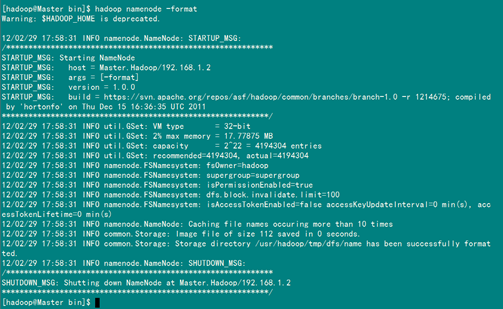
# set hadoop path

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。

201205161427467515.png



　　从上图中知道我们已经成功格式话了，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

201205161427479891.png

　　执行结果如下：



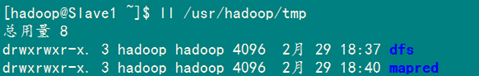
　　可以通过以下启动日志看出，首先启动namenode 接着启动datanode1，datanode2，…，然后启动secondarynamenode。再启动jobtracker，然后启动tasktracker1，tasktracker2，…。

　　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

201205161427495789.png

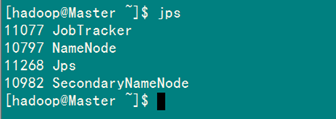
　　查看Slave1中"/usr/hadoop/tmp"文件夹内容。



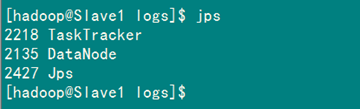
**3）验证hadoop**

　　（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。



　　在Slave1上用jps查看进程。

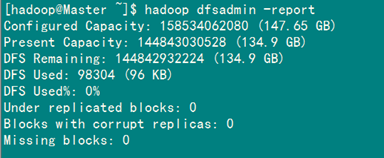


　　如果在查看Slave机器中发现"DataNode"和"TaskTracker"没有起来时，先查看一下日志的，如果是"namespaceID"不一致问题，采用"常见问题FAQ6.2"进行解决，如果是"No route to host"问题，采用"常见问题FAQ6.3"进行解决。

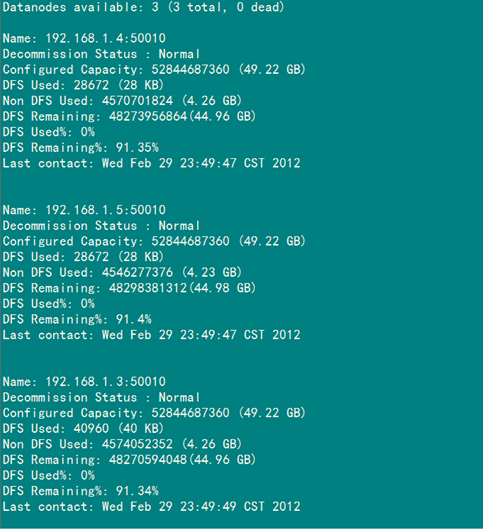
　　（2）验证方式二：用"hadoop dfsadmin -report"

　　用这个命令可以查看Hadoop集群的状态。

　　Master服务器的状态：

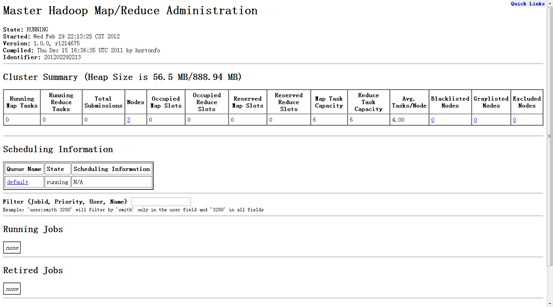


　　Slave服务器的状态

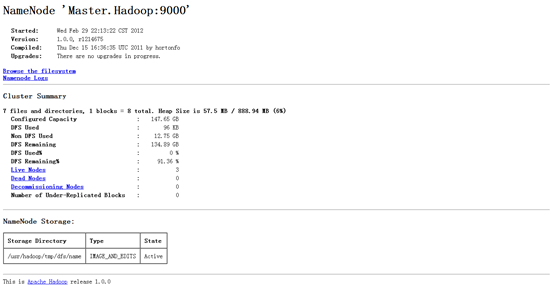


**4.4 网页查看集群**

**1）访问"http:192.168.1.2:50030"**



2）访问"**http:192.168.1.2:50070**"



**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.

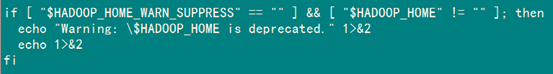
　　经查hadoop-1.0.0/bin/hadoop脚本和"hadoop-config.sh"脚本，发现脚本中对HADOOP\_HOME的环境变量设置做了判断，笔者的环境根本不需要设置HADOOP\_HOME环境变量。

　　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：

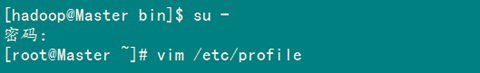
export HADOOP\_HOME\_WARN\_SUPPRESS=1

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

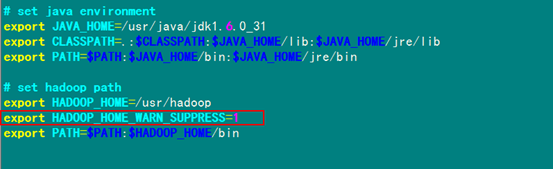


　　我们这里本着不动Hadoop原配置文件的前提下，采用"**方案二**"，在"/etc/profile"文件添加上面内容，并用命令"source /etc/profile"使之有效。

**1）切换至root用户**



**2）添加内容**

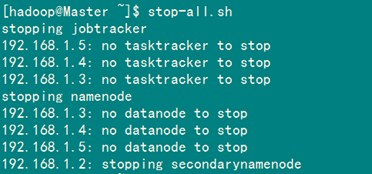


**3）重新生效**

201205161427559589.png

**5.2 解决"no datanode to stop"问题**

　　当我停止Hadoop时发现如下信息：



　　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fotmat前，清空tmp一下的所有目录。

**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：**

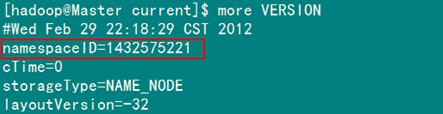
　　1）修改每个Slave的namespaceID使其与Master的namespaceID一致。

　　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



　　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.DFSClient: Exception in createBlockOutputStream java.io.IOException: **Bad connect ack with firstBadLink**

　　INFO hdfs.DFSClient: Abandoning block blk\_-1300529705803292651\_37023

　　WARN hdfs.DFSClient: 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 nofile 102400

hard nofile 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(即以递回的方式逐个变更)

--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=7。

**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(即以递回的方式逐个变更)

--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 [-1246BCpqrv] [-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。