高性能集群搭建指南（二） - 系统配置

2017-05-15 22:05:49

很多操作都需要root权限，建议在熟练之后切换到root用户下进行操作。(当然刚开始的时候一定要谨慎使用root用户，比较容易出错会出现各种问题，后果有时候比较严重）。所以不确定的东西建议先用虚拟机下试验一下咯。

## 一、hostname hosts 和 IP

### 1. hostname 设定

hostname 为本机主机名，操作系统会根据 /etc/hostname 中的内容来设置主机名  
由于hostname涉及到集群之间互联，因此每台机器的 hostname 和其他机器中 host 文件中对应的 hostname 应保持一致。

#### 1.1 暂时更改 hostname

$ hostname <your\_hostname>  
$ su -l #重新登录即生效

实际测试如下

[root@mic1 ~]# hostname #查看主机名  
mic1  
[root@mic1 ~]# hostname 2333 #修改  
[root@mic1 ~]# su -l #重新登录  
Last login: Fri May 19 01:02:10 CST 2017 on pts/0  
Intel(R) Parallel Studio XE 2017 Update 1 for Linux\*  
Copyright (C) 2009-2016 Intel Corporation. All rights reserved.  
[root@2333 ~]# hostname  
2333  
[root@2333 ~]#

#### 1.2 永久更改 hostname

将主机名写入 /etc/hostname 中，在系统下次重启后生效。

### 2. hosts

目前的以太网使用的是的 TCP/IP 协定，其中IP为第四版的 IPv4 。IPv4 为32 位，为了人脑易读已经转成四组十进制的数字了，每组为 0-255，例如 12.34.56.78 这样的格式。当我们利用 Internet 传送数据的时候，就需要这个IP来找到对应的主机。

然而IP这种数字不易记忆，为了应付这个问题，早期的开发者想到一个方法，那就是利用某些特定的文件将主机名称与IP作一个对应，如此一来，我们就可以同过主机名称来取得该主机的IP了！在linux中，这个文件为 /etc/hosts。（Winodws中为C:\Windows\System32\drivers\etc\hosts）

我们需要修改 /etc/hosts 中的内容，让集群之间通过主机名来相互识别。可以在服务端写好该文件之后把它分发到其余节点上。

以下是一个样例：

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
  
#addbylq 20170327  
10.111.0.101 mic1  
10.111.0.102 mic2  
10.111.0.103 mic3  
10.111.0.104 mic4  
10.111.0.105 mic5  
10.111.0.106 mic6  
10.111.0.107 mic7  
10.111.0.108 mic8  
10.111.0.109 mic9  
10.111.0.110 mic10  
10.111.0.111 mic11  
10.111.0.112 mic12

### 3. IP

在成功安装完成CentOS后，配置好网络配置就可以远程连接这些机器进行后面的配置了，这里先介绍以太网的配置。(有IB的话应该直接先配IB的网络，然后 hosts 文件中写入 IB 的IP地址）

通过 ip addr 命令查看各个网口的 IP详细信息。

[lq@mic1 ~]$ ip addr  
1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN   
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
 inet 127.0.0.1/8 scope host lo  
 valid\_lft forever preferred\_lft forever  
 inet6 ::1/128 scope host   
 valid\_lft forever preferred\_lft forever  
2: ens1f0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN qlen 1000  
 link/ether e8:61:1f:13:cd:c8 brd ff:ff:ff:ff:ff:ff  
3: ens1f1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN qlen 1000  
 link/ether e8:61:1f:13:cd:c9 brd ff:ff:ff:ff:ff:ff  
4: ens5f0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP qlen 1000  
 link/ether 30:5a:3a:77:53:bd brd ff:ff:ff:ff:ff:ff  
 inet 10.4.9.238/24 brd 10.4.9.255 scope global dynamic ens5f0  
 valid\_lft 72615sec preferred\_lft 72615sec  
 inet6 2001:da8:203:eb4:325a:3aff:fe77:53bd/64 scope global noprefixroute dynamic   
 valid\_lft 2591919sec preferred\_lft 604719sec  
 inet6 fe80::325a:3aff:fe77:53bd/64 scope link   
 valid\_lft forever preferred\_lft forever  
5: ens5f1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN qlen 1000  
 link/ether 30:5a:3a:77:53:be brd ff:ff:ff:ff:ff:ff  
6: ib0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 65520 qdisc pfifo\_fast state DOWN qlen 1024  
 link/infiniband 80:00:00:68:fe:80:00:00:00:00:00:00:e4:1d:2d:03:00:a5:f0:d6 brd 00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff  
7: ib1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 4092 qdisc pfifo\_fast state DOWN qlen 1024  
 link/infiniband 80:00:00:68:fe:80:00:00:00:00:00:00:e4:1d:2d:03:00:a5:f0:d7 brd 00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff  
 inet 10.4.9.101/24 brd 10.4.9.255 scope global ib1  
 valid\_lft forever preferred\_lft forever

#### 3.1 网络服务的几个守护进程

* network.service  
  网络服务
* NetworkManager.service #一般选择禁用 NetworkManager  
  NetworkManager守护程序通过管理主网络连接和其他网络接口（如以太网，WiFi和移动宽带设备），尽可能地使网络配置和操作尽自动化。 当该设备的连接变为可用时，NetworkManager将连接任何网络设备，除非该功能被禁用。所以感觉NetworkManager很烦人，直接把它禁用了吧。

$ systemctl disable NetworkManager  
$ systemctl stop NetworkManager

#### 3.2 配置文件

* 修改网口配置文件

各个网口的配置文件在 /etc/sysconfig/network-scripts 中

样例如下 ↓

/etc/sysconfig/network-scripts/ifcfg-ens5f0

TYPE="Ethernet"  
BOOTPROTO="static" #dhcp 修改为 static  
DEFROUTE="yes"  
PEERDNS="yes"  
PEERROUTES="yes"  
IPV4\_FAILURE\_FATAL="no"  
IPV6INIT="yes"  
IPV6\_AUTOCONF="yes"  
IPV6\_DEFROUTE="yes"  
IPV6\_PEERDNS="yes"  
IPV6\_PEERROUTES="yes"  
IPV6\_FAILURE\_FATAL="no"  
IPV6\_ADDR\_GEN\_MODE="stable-privacy"  
NAME="ens5f0"  
UUID="2de593e2-2c45-45de-bc60-141f814a49ab" #网卡的识别码，勿修改  
DEVICE="ens5f0" #网口名  
ONBOOT="yes" #在系统启动时是否启动该网口  
  
#添加以下内容  
IPADDR= "10.4.9.238" #静态IP  
NETMASK= "255.255.255.0" #子网掩码 设定为255.255.255.0  
GATEWAY= "10.4.9.1" #网关，最后一部分为1，其他和 IP 一致

* 修改 DNS 服务器配置

DNS 是域名和IP地址相互映射的一个分布式数据库；用户通过 DNS服务器将域名解析为IP地址。常用的 DNS服务器

$ vim /etc/resolv.conf

加入以下内容

# Generated by NetworkManager  
nameserver 202.106.0.20  
nameserver 208.67.222.222  
nameserver 202.112.128.51

修改完后重启network

$ systemctl restart network

#### 3.3 以太网带宽测试

* 安装 iperf

$ yum install iperf -y

* 服务端

$ iperf -s # 通过 -s 指定为Server端

* 客户端

$ iperf -c mic1 -P 30 -t 10 # 通过 -c 指定为Client端，mic1为服务端的hostname，也可以写IP -t 指定测试时间 /s  
  
#测试结果  
[ ID] Interval Transfer Bandwidth  
[ 15] 0.0-10.0 sec 13.9 MBytes 11.6 Mbits/sec  
[ 17] 0.0-10.0 sec 71.0 MBytes 59.5 Mbits/sec  
[ 29] 0.0-10.0 sec 17.8 MBytes 14.9 Mbits/sec  
[ 20] 0.0-10.0 sec 142 MBytes 119 Mbits/sec  
[ 24] 0.0-10.0 sec 21.4 MBytes 17.9 Mbits/sec  
[ 27] 0.0-10.0 sec 23.6 MBytes 19.8 Mbits/sec  
[ 30] 0.0-10.0 sec 30.8 MBytes 25.8 Mbits/sec  
[ 4] 0.0-10.0 sec 34.5 MBytes 28.9 Mbits/sec  
[ 9] 0.0-10.0 sec 33.9 MBytes 28.4 Mbits/sec  
[ 10] 0.0-10.0 sec 31.4 MBytes 26.3 Mbits/sec  
[ 12] 0.0-10.0 sec 76.9 MBytes 64.3 Mbits/sec  
[ 18] 0.0-10.0 sec 30.5 MBytes 25.5 Mbits/sec  
[ 22] 0.0-10.0 sec 15.9 MBytes 13.3 Mbits/sec  
[ 26] 0.0-10.0 sec 10.0 MBytes 8.37 Mbits/sec  
[ 31] 0.0-10.0 sec 70.5 MBytes 59.0 Mbits/sec  
[ 32] 0.0-10.0 sec 71.4 MBytes 59.7 Mbits/sec  
[ 8] 0.0-10.0 sec 71.1 MBytes 59.5 Mbits/sec  
[ 5] 0.0-10.0 sec 39.9 MBytes 33.3 Mbits/sec  
[ 16] 0.0-10.0 sec 38.5 MBytes 32.2 Mbits/sec  
[ 23] 0.0-10.0 sec 23.8 MBytes 19.9 Mbits/sec  
[ 28] 0.0-10.0 sec 20.5 MBytes 17.1 Mbits/sec  
[ 11] 0.0-10.0 sec 28.4 MBytes 23.7 Mbits/sec  
[ 19] 0.0-10.0 sec 16.0 MBytes 13.4 Mbits/sec  
[ 21] 0.0-10.0 sec 24.1 MBytes 20.2 Mbits/sec  
[ 7] 0.0-10.0 sec 65.2 MBytes 54.5 Mbits/sec  
[ 13] 0.0-10.1 sec 20.6 MBytes 17.2 Mbits/sec  
[ 14] 0.0-10.1 sec 33.1 MBytes 27.6 Mbits/sec  
[ 25] 0.0-10.1 sec 34.5 MBytes 28.8 Mbits/sec  
[ 6] 0.0-10.1 sec 16.6 MBytes 13.8 Mbits/sec  
[ 3] 0.0-10.1 sec 10.8 MBytes 8.92 Mbits/sec  
[SUM] 0.0-10.1 sec 1.11 GBytes 944 Mbits/sec   
  
# 结果为 944Mb/s 正常

#### 3.4 碰到过的一些问题

* 在配置好网络后重启网络服务失败，查看错误信息 报错 Failed to start LSB，这个问题很常见。有可能是网口的配置文件中写错了；当排除配置文件写错的可能性之后，基本就是连接的网线有问题，也就是说，问题可能出在连接服务器的网线、交换机、连接交换机的网线 中。这时可以拿笔记本去测试各个部分网络是否正常，直到找到并解决问题。

[root@mic1 ~]# systemctl restart network  
Job for network.service failed because the control process exited with error code. See "systemctl status network.service" and "journalctl -xe" for details.

[root@mic1 ~]# systemctl status network.service  
● network.service - LSB: Bring up/down networking  
 Loaded: loaded (/etc/rc.d/init.d/network; bad; vendor preset: disabled)  
 Active: failed (Result: exit-code) since Tue 2017-05-23 01:00:42 CST; 5s ago  
 Docs: man:systemd-sysv-generator(8)  
 Process: 22843 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=1/FAILURE)  
  
May 23 01:00:42 mic1 network[22843]: RTNETLINK answers: File exists  
May 23 01:00:42 mic1 network[22843]: RTNETLINK answers: File exists  
May 23 01:00:42 mic1 network[22843]: RTNETLINK answers: File exists  
May 23 01:00:42 mic1 network[22843]: RTNETLINK answers: File exists  
May 23 01:00:42 mic1 network[22843]: RTNETLINK answers: File exists  
May 23 01:00:42 mic1 network[22843]: RTNETLINK answers: File exists  
May 23 01:00:42 mic1 systemd[1]: network.service: control process exited, code=exited status=1  
May 23 01:00:42 mic1 systemd[1]: Failed to start LSB: Bring up/down networking.  
May 23 01:00:42 mic1 systemd[1]: Unit network.service entered failed state.  
May 23 01:00:42 mic1 systemd[1]: network.service failed.  
  
  
  
[root@mic1 ~]# journalctl -xe  
  
May 23 00:59:01 mic1 NetworkManager[1004]: <info> [1495472341.2099] ifcfg-rh: update /etc/sysconfig/network-scripts/ifcfg-ens1f0 (0864c1c2-37f4-46a0-a965-4d6fcfb31496,"ens1f0")  
May 23 00:59:01 mic1 network[22473]: Bringing up loopback interface: Could not load file '/etc/sysconfig/network-scripts/ifcfg-lo'  
May 23 00:59:01 mic1 network[22473]: Could not load file '/etc/sysconfig/network-scripts/ifcfg-lo'  
May 23 00:59:01 mic1 network[22473]: Could not load file '/etc/sysconfig/network-scripts/ifcfg-lo'  
May 23 00:59:02 mic1 network[22473]: Could not load file '/etc/sysconfig/network-scripts/ifcfg-lo'  
May 23 00:59:02 mic1 network[22473]: [ OK ]  
May 23 00:59:02 mic1 network[22473]: Bringing up interface ens1f0: /etc/sysconfig/network-scripts/ifcfg-ens1f0: line 17: qwer: command not found  
May 23 00:59:02 mic1 NetworkManager[1004]: <info> [1495472342.6493] audit: op="connection-activate" uuid="0864c1c2-37f4-46a0-a965-4d6fcfb31496" name="ens1f0" result="fail" reason="No suitable   
May 23 00:59:02 mic1 network[22473]: Error: Connection activation failed: No suitable device found for this connection.  
May 23 00:59:02 mic1 network[22473]: [FAILED]  
May 23 00:59:02 mic1 network[22473]: Bringing up interface ens5f0: [ OK ]  
May 23 00:59:03 mic1 NetworkManager[1004]: <info> [1495472343.4374] audit: op="connection-activate" uuid="2ab4abde-b8a5-6cbc-19b1-2bfb193e4e89" name="System ib0" result="fail" reason="No suita  
May 23 00:59:03 mic1 network[22473]: Bringing up interface ib0: Error: Connection activation failed: No suitable device found for this connection.  
May 23 00:59:03 mic1 network[22473]: [FAILED]  
May 23 00:59:03 mic1 network[22473]: Bringing up interface ib1: [ OK ]  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 network[22473]: RTNETLINK answers: File exists  
May 23 00:59:03 mic1 systemd[1]: network.service: control process exited, code=exited status=1  
May 23 00:59:03 mic1 systemd[1]: Failed to start LSB: Bring up/down networking.

* 能ping通自己但是无法ping通其他服务器，可能是 GATEWAY 写错了。
* 能ping通IP，但是无法ping通域名，一般是DNS的配置问题

## 二、SELinux和防火墙

把SELinux放到开来写，主要是因为它也会影响后面的配置工作，并且自己也被SELinux坑过太多次了，SElinux的详细说明见 man selinux 。

建议在装机完成后马上关闭SELinux。

### 1. SELinux模式

SELinux 更能遵从最小权限的理念。在缺省的 enforcing 情况下，一切均被拒绝，接着有一系列例外的政策来允许系统的每个元素（服务、程序、用户）运作时所需的访问权。当一项服务、程序或用户尝试访问或修改一个它不须用的文件或资源时，它的请求会遭拒绝，而这个行动会被记录下来。

由于SELinux未关闭，在搭本地源的时候出现过客户端无法访问的情况。

SELinux 拥有三个基本的操作模式，当中 enforcing 是缺省的模式。此外，它还有一个 targeted 或 mls 的修饰语。这管制 SELinux 规则的应用有多广泛，当中 targeted 是较宽松的级别。

* **enforcing**： 这个缺省模式会在系统上启用并实施 SELinux 的安全性政策，拒绝访问及记录行动。
* **permissive**： 在 permissive 模式下，SELinux 会被启用但不会实施安全性政策，而只会发出警告及记录行动。permissive 模式在排除 SELinux 的问题时很有用。
* **disabled**： SELinux 已被停用。

### 2. 关闭SELinux

SELinux 的配置文件为 /etc/selinux/config

$ vim /etc/selinux/config  
  
#将SELinux模式改为disabled  
SELINUX=disabled #默认为enforcing

### 3. 查看SELinux工作 状态

sestatus 或 getenforce

$ sestatus  
SELinux status: disabled  
  
$ getenforce  
Disabled

### 4. 关于 setenforce

setenforce 0 将SELinux临时改为 permissive 模式，注意并非是 disabled 模式。

DESCRIPTION

* Use Enforcing or 1 to put SELinux in **enforcing** mode.
* Use Permissive or 0 to put SELinux in **permissive** mode.

### 5. 关闭防火墙

防火墙关闭后的安全问题有待探讨

$ systemctl stop firewalld  
$ systemctl disable firewalld

### 6. 碰到的一些问题

之前弄错过 SELinux 的配置文件，修改 /etc/sysconfig/selinux 发现仍然无法禁用 SELinux。

## 三、yum 本地源 （可选）

这里先介绍一下CentOS的软件包管理工具 yum

Yum（Yellow dog Updater, Modified）由Duke University团队，修改Yellow Dog Linux的Yellow Dog Updater开发而成，是一个基于RPM包管理的字符前端软件包管理器。能够从指定的服务器自动下载RPM包并且安装，可以处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次次下载、安装。被Yellow Dog Linux本身，以及Fedora、Red Hat Enterprise Linux采用。

配置文件

* /etc/yum.conf
* /etc/yum/version-groups.conf
* /etc/yum.repos.d/
* /etc/yum/pluginconf.d/
* /var/cache/yum/

### 1. 服务端配置

#### 1.1 准备rpm包

* 将 Centos-Base 源同步至本地目录

在 CentOS-7-x86\_64-Everything-1611.iso 中有 yum 源中的全部rpm包，故将刻录好的启动盘中的软件包全部拷到本地即可。

$ mount /dev/sdb1 /mnt  
$ mkdir /yum/base  
$ cp -r /mnt/\* /yum/base/

这里也提供另一种用 reposync 指令来同步rpm包到本地目录的方法,执行以下脚本即可

#!/bin/bash  
cat << EOF > sync-centos.sh  
#!/bin/bash  
#  
# reposync  
#  
  
BASEDIR=/yum/  
mkdir -p $BASEDIR   
cd $BASEDIR  
  
reposync -n -r updates  
repomanage -o -c updates | xargs rm -fv  
createrepo updates  
  
reposync -n -r base --downloadcomps  
repomanage -o -c base | xargs rm -fv  
createrepo base -g comps.xml  
EOF  
  
chmod 755 sync-centos.sh  
sh sync-centos.sh

* 将epel源同步至本地目录

执行以下脚本即可

#!/bin/bash  
#epel源更新脚本  
  
#安装epel包  
rpm -ivh http://dl.fedoraproject.org/pub/epel/7/x86\_64/e/epel-release-7-9.noarch.rpm  
#yum install epel-release -y  
cd /yum/  
  
cat << EOF > sync-epel.sh  
#!/bin/bash  
  
BASEDIR=/yum/  
mkdir -p \$BASEDIR   
cd \$BASEDIR  
  
reposync -n -r epel  
repomanage -o -c epel | xargs rm -fv  
createrepo epel  
EOF  
  
chmod 755 sync-epel.sh  
sh sync-epel.sh

在无网络的情况下，可以提前准备好 epel本地源 ，并复制到 /yum/epel 目录下。

#### 1.2 配置repo

* 先做好备份

$ cp -r /etc/yum.repos.d /etc/yum.repos.d.bak  
$ cd /etc/yum.repos.d  
$ rm -f ./\*

* CentOS-Base源

$ vim CentOS-Server.repo

将以下内容写入 CentOS-Server.repo

[CentOS-Base]  
name=CentOS-$releasever - Base  
baseurl=file:///yum/base/  
enable=1  
gpgcheck=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

Base 写为其他也可以，gpgcheck 为软件包签名的验证，以防软件包损坏或被篡改，这里选择关闭。

* epel源

$ vim /etc/yum.repos.d/epel-Server

将以下内容写入 epel-Server

[epel-Server]  
name=epel-Server  
baseurl=file:///yum/epel  
enable=1  
gpgcheck=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

#### 1.3 更新软件包缓存

$ yum clean all  
$ yum makecache

这样服务端的 yum 源就配置好了，可以简单地验证是否正常。

$ yum install tree -y

#### 1.4 ftp服务器配置

我们通过在服务端的 ftp服务，实现局域网的 yum源的搭建。

* ftp 安装

$ yum install vsftpd -y

* ftp 配置  
  编辑 /etc/vsftpd/vsftpd.conf

$ echo "anon\_root=/yum/" >> /etc/vsftpd/vsftpd.conf

* ftp 服务启动

$ systemctl start vsftpd #启动服务  
$ systemctl enable vsftpd #启用开机启动  
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.service to /usr/lib/systemd/system/vsftpd.service.

### 2. 客户端

客户端的配置相对来说会快很多，只需要修改repo 文件即可

#### 2.1 配置 repo

$ cp -r /etc/yum.repos.d /etc/yum.repos.d.bak  
$ cd /etc/yum.repos.d  
$ rm -f ./\*

* CentOS-Base源

$ vim CentOS-Client.repo

将以下内容写入 CentOS-Client.repo

[CentOS-Media]  
name=CentOS-$releasever - Media  
baseurl=ftp://<hostname>/base #<hostname>也可以写IP  
enable=1  
gpgcheck=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

* epel源

$ vim /etc/yum.repos.d/epel-Client

将以下内容写入 epel-Client

[epel-Client]  
name=epel-Client  
baseurl=ftp://<hostname>/epel #<hostname>也可以写IP  
enable=1  
gpgcheck=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

#### 2.2 更新软件包缓存

$ yum clean all  
$ yum makecache

### 3.碰到过的一些问题

#### 3.1 服务端

* 未复制 repodata 目录导致 repomd.xml 缺失

failure: repodata/repomd.xml from Media: [Errno 256] No more mirrors to try.  
file:///var/www/html/base/Packages/repodata/repomd.xml: [Errno 14] curl#37 - "Couldn't open file /yum/base/repodata/repomd.xml"

* repo文件中 baseurl 写成了 mirrorlist 导致以下问题： 无法找到有效的 baseurl

One of the configured repositories failed (Unknown),  
 and yum doesn't have enough cached data to continue. At this point the only  
 safe thing yum can do is fail. There are a few ways to work "fix" this:  
  
 1. Contact the upstream for the repository and get them to fix the problem.  
  
 2. Reconfigure the baseurl/etc. for the repository, to point to a working  
 upstream. This is most often useful if you are using a newer  
 distribution release than is supported by the repository (and the  
 packages for the previous distribution release still work).  
  
 3. Disable the repository, so yum won't use it by default. Yum will then  
 just ignore the repository until you permanently enable it again or use  
 --enablerepo for temporary usage:  
  
 yum-config-manager --disable <repoid>  
  
 4. Configure the failing repository to be skipped, if it is unavailable.  
 Note that yum will try to contact the repo. when it runs most commands,  
 so will have to try and fail each time (and thus. yum will be be much  
 slower). If it is a very temporary problem though, this is often a nice  
 compromise:  
  
 yum-config-manager --save --setopt=<repoid>.skip\_if\_unavailable=true  
  
Cannot find a valid baseurl for repo: Media

#### 3.2 客户端

* 服务端未禁用 SELinux 导致无法找到 repomd.xml 文件

## 三、SSH无密码访问

SSH是一种网络协议，用于计算机之间的加密登录。因为受版权和加密算法的限制，现在大多使用OpenSSH。OpenSSH是SSH的替代软件，而且是开源的。

<http://www.ruanyifeng.com/blog/2011/12/ssh_remote_login.html>

ssh的工作机制

1. 远程主机收到用户的登录请求，把自己的公钥发给用户。
2. 用户使用这个公钥，将登录密码加密后，发送给远程主机。
3. 远程主机用自己的私钥，解密登录密码，如果密码正确，则允许用户登录。

ssh中的文件

1. id\_rsa  
   ssh私钥，用于解密登录密码
2. id\_rsa.pub  
   ssh公钥，用于加密
3. authorized\_keys  
   保存了已经认证的主机的公钥指纹，authorized\_keys 保证了被登录时的安全性。
4. known\_hosts 远程主机的公钥指纹，RSA公钥通过MD5计算产生当前公钥的指纹，用于提供给用户验证远程主机的密钥是否未被修改。known\_hosts 保证了登录其他机器时的安全性。

### 1. OpenSSH 安装

$ yum install openssh -y

### 2. 单机的无密码访问配置

实现无密码访问的条件是本机公钥存在于远程主机 authorized\_keys 中。

我们利用 ssh-keygen 生成密钥 id\_rsa 和 id\_rsa.pub ， 再通过 ssh-copy-id 将公钥拷贝至远程主机的 authorized\_keys ，即可实现对远程主机的无密码访问。

需要注意的是，无论是本地还是远程，无密码访问都是对用户而言的，而非是整个机器的所有用户。

$ ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa  
$ ssh-copy-id [user@]hostname

当然，拷贝公钥也可以通过 scp 来实现。

如果该用户先前已经生成了一组密钥，若重新生成，则公钥指纹也会被修改。因此想通过ssh连接该主机的用户需要把 known\_hosts 中对应的远程机器的公钥指纹删除。

### 3. 集群的无密码访问配置

整体思路先在所有机器中利用 ssh-keygen 产生密钥。然后在一台机器上，实现其对本机和其他机器各用户的无密码访问，接着把本机和其他机器的公钥复制至本机的 authorized\_keys 中。最后把该 autorized\_keys 文件复制到其他机器各用户的 ~/.ssh目录下。

脚本如下 ↓

#!/bin/bash  
   
#@HEADER  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#  
# CIS: Cluster Installation Script  
#  
# Contact:  
# Liu Quan (liuquan2049@buaa.edu.cn)  
#  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#@HEADER  
  
NODE\_NUMS=12  
ROOT\_PASSWD=root  
USER\_PASSWD=lq  
  
  
# ===================================================  
#ssh\_keygen 生成当前用户的密钥 -q quiet  
#--- overwrite? y  
ssh\_keygen()  
{  
 expect -c "set timeout -1;  
 spawn ssh-keygen -t rsa -P \"\" -f $HOME/.ssh/id\_rsa   
 expect {  
 \*Overwrite\*(y/n)?\* {send -- \"y\r\";exp\_continue;}  
 eof {exit 0;}   
 }";  
}  
  
# ssh\_keygen\_all 生成所有用户密钥，包括lq@mic1  
# --- $1 --- 用户名  
ssh\_keygen\_all()  
{  
for i in $(seq 1 12)  
do  
 if [ $i != 1 -o $1 != "root" ];then  
 ssh $1@mic$i "ssh-keygen -t rsa -P \"\" -f ~/.ssh/id\_rsa "  
 else   
 echo 2333333333333333333333333333333333333333333333333  
 fi  
done  
}  
  
  
  
# ssh\_copy\_id 配置当前用户无密码访问hosts中其他节点  
# --- $1 --- 当前用户名  
# --- $2 --- 当前用户密码  
ssh\_copy\_id()  
{  
for i in $(seq 1 $NODE\_NUMS)  
do  
node\_id=$i  
 expect -c "set timeout -1;   
 spawn ssh-copy-id $1@mic$node\_id;  
 expect {  
 \*(yes/no)\* {send -- yes\r;exp\_continue;}  
 \*password:\* {send -- $2\r;exp\_continue;}  
 eof {exit 0;}  
 }";  
  
done  
sleep 1  
}  
  
  
# scp\_pub 将其他用户公钥复制到mic1  
# --- $1 --- 用户名  
scp\_pub()  
{  
for i in $(seq 1 $NODE\_NUMS)  
do  
 scp $1@mic$i:~/.ssh/id\_rsa.pub tmp  
 cat tmp >> /root/.ssh/authorized\_keys  
done  
  
rm -f tmp  
}  
  
# scp\_aukeys 将已认证公钥复制到其他机器  
scp\_aukeys()  
{  
for i in $(seq 1 $NODE\_NUMS)  
do  
 scp /root/.ssh/authorized\_keys $1@mic$i:~/.ssh/authorized\_keys  
 scp /root/.ssh/known\_hosts $1@mic$i:~/.ssh/known\_hosts  
done  
}  
  
#=========================================================  
  
# main  
  
rm ~/.ssh/known\_hosts  
  
echo -e "\033[32m 开始生成用户密钥 ... \033[0m"  
sleep 1  
ssh\_keygen  
echo -e "\033[32m 用户密钥生成 完成！ \033[0m"  
sleep 1  
echo   
echo -e "\033[32m 开始配置无密码访问 ... \033[0m"  
sleep 1  
ssh\_copy\_id root $ROOT\_PASSWD  
ssh\_copy\_id lq $USER\_PASSWD  
echo -e "\033[32m 对其他主机无密码访问配置 完成！ \033[0m"  
sleep 1  
echo   
echo -e "\033[32m 开始生成其他主机用户密钥 ... \033[0m"  
sleep 1  
ssh\_keygen\_all root  
ssh\_keygen\_all lq  
echo -e "\033[32m 其他主机用户密钥生成 完成! \033[0m"  
sleep 1  
echo   
echo -e "\033[32m 开始加入其他主机公钥 ... \033[0m"  
sleep 1  
scp\_pub root  
scp\_pub lq  
echo -e "\033[32m 加入其他主机公钥 完成！ \033[0m"  
sleep 1  
echo   
echo -e "\033[32m 开始分发可用的认证公钥 ... \033[0m"  
sleep 1  
scp\_aukeys root  
scp\_aukeys lq  
echo -e "\033[32m 可用的认证密钥分发 完成！ ... \033[0m"  
sleep 1  
echo   
echo -e "\033[32m 集群无密码访问配置完成 \033[0m"  
sleep 1

## 四、InfiniBand 配置

如果我们有InfiniBand 交换机的话，InfiniBand的配置应放在第一步来做，因为它关系到了集群的网络互联。  
如果手头上暂时没有InfiniBand交换机的话，也可以在这里先把HCA卡的驱动程序装上，这样之后把IB线插上之后，就可以直接用了。 （当然也需要修改 hosts 中的IP，并重启守护进程）

### 1. 安装TCL TK

$ yum install tcl tk -y

### 2. 安装 OFED驱动

* 在官网中下载驱动程序
* Mellanox Ifiniband驱动: <http://www.mellanox.com/page/products_dyn?product_family=26&mtag=linux_sw_drivers>
* Redhat/Centos 7.3 版本: <http://www.mellanox.com/page/mlnx_ofed_eula?mtag=linux_sw_drivers&mrequest=downloads&mtype=ofed&mver=MLNX_OFED-4.0-2.0.0.1&mname=MLNX_OFED_LINUX-4.0-2.0.0.1-rhel7.3-x86_64.tgz>
* 解压并安装

$ wget http://www.mellanox.com/page/mlnx\_ofed\_eula?mtag=linux\_sw\_drivers&mrequest=downloads&mtype=ofed&mver=MLNX\_OFED-4.0-2.0.0.1&mname=MLNX\_OFED\_LINUX-4.0-2.0.0.1-rhel7.3-x86\_64.tgz   
  
$ tar xvf MLNX\_OFED\_LINUX-4.0-2.0.0.1-rhel7.3-x86\_64.tgz  
$ cd MLNX\_OFED\_LINUX-4.0-2.0.0.1-rhel7.3-x86\_64  
$ ./mlnxofedinstall

正常安装完成后输出如下，从以下输出可以了解到 HCA卡 的基本信息，如类型为EDR (100Gb/s)， 版本为 ConnectX-4。

Device #1:  
----------  
  
 Device Type: ConnectX4  
 Part Number: MCX456A-ECA\_Ax  
 Description: ConnectX-4 VPI adapter card; EDR IB (100Gb/s) and 100GbE; dual-port QSFP28; PCIe3.0 x16; ROHS R6  
 PSID: MT\_2190110032  
 PCI Device Name: 85:00.0  
 Base GUID: e41d2d0300a5f102  
 Versions: Current Available   
 FW 12.18.2000 12.18.2000   
 PXE 3.5.0110 3.5.0110   
  
 Status: Up to date  
  
  
Log File: /tmp/MLNX\_OFED\_LINUX-4.0-2.0.0.1.4161162.logs/fw\_update.log  
To load the new driver, run:  
/etc/init.d/openibd restart

* IP 配置

创建文件 /etc/sysconfig/network-scripts/ifcfg-ib0 ，写入以下内容并修改 IPADDR 和 NETWORK

DEVICE=ib0  
BOOTPROTO=static  
ONBOOT=yes  
NM\_CONTROLLED=yes  
NETMASK=255.255.255.0  
IPADDR=10.4.9.101  
BROADCAST=10.255.255.255  
NETWORK=10.4.9.0

这里配置完IP后，需要修改各个机器上的 hosts 文件。

* 开机启动

$ chkconfig opensmd on  
  
$ systemctl enable openibd  
Created symlink from /etc/systemd/system/sysinit.target.wants/openibd.service to /usr/lib/systemd/system/openibd.service.

* 重启网络接口和服务

$ ifdown ib0  
$ ifup ib1  
$ systemctl restart opensmd  
$ systemctl restart openibd

* 查看ib 状态

$ ibstatus  
CA 'mlx5\_0'  
 CA type: MT4115  
 Number of ports: 1  
 Firmware version: 12.17.2020  
 Hardware version: 0  
 Node GUID: 0xe41d2d0300a5f0d6  
 System image GUID: 0xe41d2d0300a5f0d6  
 Port 1:  
 State: Active #连接正常的状态是 Active  
 Physical state: LinkUp #连接正常的状态是 LinkUp  
 Rate: 10  
 Base lid: 65535  
 LMC: 0  
 SM lid: 0  
 Capability mask: 0x2651e848  
 Port GUID: 0xe41d2d0300a5f0d6  
 Link layer: InfiniBand  
CA 'mlx5\_1'  
 CA type: MT4115  
 Number of ports: 1  
 Firmware version: 12.17.2020  
 Hardware version: 0  
 Node GUID: 0xe41d2d0300a5f0d7  
 System image GUID: 0xe41d2d0300a5f0d6  
 Port 1:  
 State: Down  
 Physical state: Disabled  
 Rate: 10  
 Base lid: 3  
 LMC: 0  
 SM lid: 3  
 Capability mask: 0x2651e848  
 Port GUID: 0xe41d2d0300a5f0d7  
 Link layer: InfiniBand  
  
$ ibhosts #输出信息待补充

### 3. 重启

opensmd 服务会和其他服务冲突，目前觉得最好的解决办法是在第一次配置完后设置服务开机启动并直接重启机器，之后一切都正常。

### 4. IB读写带宽测试

* 服务端

$ ib\_write\_bw # 写带宽测试；延迟测试ib\_write\_lat方法类似

* 客户端

$ ib\_write\_bw mic1 # mic1 为服务端，暂时没有IB，无法贴出输出...

## 五、NFS文件系统 exports

NFS 是Network File System的缩写，即网络文件系统。一种使用于分散式文件系统的协定，由Sun公司开发，于1984年向外公布。功能是通过网络让不同的机器、不同的操作系统能够彼此分享个别的数据，让应用程序在客户端通过网络访问位于服务器磁盘中的数据，是在类Unix系统间实现磁盘文件共享的一种方法。

NFS 的基本原则是“容许不同的客户端及服务端通过一组RPC分享相同的文件系统”，它是独立于操作系统，容许不同硬件及操作系统的系统共同进行文件的分享。

NFS在文件传送或信息传送过程中依赖于RPC协议。RPC，远程过程调用 (Remote Procedure Call) 是能使客户端执行其他系统中程序的一种机制。NFS本身是没有提供信息传输的协议和功能的，但NFS却能让我们通过网络进行资料的分享，这是因为NFS使用了一些其它的传输协议。而这些传输协议用到这个RPC功能的。可以说NFS本身就是使用RPC的一个程序。或者说NFS也是一个RPC SERVER。所以只要用到NFS的地方都要启动RPC服务，不论是NFS SERVER或者NFS CLIENT。这样SERVER和CLIENT才能通过RPC来实现PROGRAM PORT的对应。可以这么理解RPC和NFS的关系：NFS是一个文件系统，而RPC是负责负责信息的传输。

### 1. 安装NFS服务

* nfs-utils :包括基本的NFS命令与监控程序
* rpcbind :支持安全NFS RPC服务的连接

$ yum install rpcbind nfs-utils -y

### 2. NFS系统守护进程

* nfsd：基本的NFS守护进程，主要功能是管理客户端是否能够登录服务器；
* mountd：RPC安装守护进程，主要功能是管理NFS的文件系统。当客户端顺利通过nfsd登录NFS服务器后，在使用NFS服务所提供的文件前，还必须通过文件使用权限的验证。它会读取NFS的配置文件 /etc/exports 来对比客户端权限。
* rpcbind：主要功能是进行端口映射工作。当客户端尝试连接并使用RPC服务器提供的服务（如NFS服务）时，rpcbind会将所管理的与服务对应的端口提供给客户端，从而使客户可以通过该端口向服务器请求服务。

设置NFS守护进程开机启动

$ systemctl enable rpcbind  
Created symlink from /etc/systemd/system/sockets.target.wants/rpcbind.socket to /usr/lib/systemd/system/rpcbind.socket.  
$ systemctl enable nfs  
Created symlink from /etc/systemd/system/multi-user.target.wants/nfs-server.service to /usr/lib/systemd/system/nfs-server.service.

### 3. NFS服务器配置

NFS 常用配置目录  
/etc/exports NFS服务的主要配置文件  
/usr/sbin/exportfs NFS服务的管理命令  
/usr/sbin/showmount 客户端的查看命令  
/var/lib/nfs/etab 记录NFS分享出来的目录的完整权限设定值 /var/lib/nfs/xtab 记录曾经登录过的客户端信息

NFS服务的配置文件为 /etc/exports，这个文件是NFS的主要配置文件，不过系统并没有默认值，所以这个文件不一定会存在，可能要使用vim手动建立，然后在文件里面写入配置内容。

/etc/exports 文件内容格式：

<输出目录> [客户端1 选项（访问权限,用户映射,其他）] [客户端2 选项（访问权限,用户映射,其他）]

目前我的配置选项为 ↓

/home/cluster \*(rw,no\_all\_squash,no\_root\_squash,fsid=0,insecure)

* 输出目录：
* 输出目录是指NFS系统中需要共享给客户机使用的目录，客户端和服务端目录必须一致，否则会出现程序无法在多机运行的情况。
* 客户端：
* 客户端是指网络中可以访问这个NFS输出目录的计算机， 客户端常用的指定方式为：
  + 指定ip地址的主机：10.4.9.101
  + 指定子网中的所有主机：10.4.9.0/24 10.4.9.0/255.255.255.0
  + 指定域名的主机：nfs.cnhzz.com
  + 指定域中的所有主机：\*.cnhzz.com
  + 所有主机：\*
* 选项：
* 选项用来设置输出目录的访问权限、用户映射等。我们把访问权限设为 rw，为使客户端的NFS目录权限正常，应选择 no\_all\_squash 和 no\_root\_squash 选项。
* NFS主要有3类选项：
  + 访问权限选项
    - ro：设置输出目录只读
    - rw：设置输出目录读写
  + 用户映射选项 (关于用户访问权限，后文将详述)
    - all\_squash：将远程访问的所有普通用户及所属组都映射为匿名用户或用户组（nfsnobody）
    - no\_all\_squash：与 all\_squash 取反（默认设置）
    - root\_squash：将root用户及所属组都映射为匿名用户或用户组（默认设置）
    - no\_root\_squash：与 rootsquash 取反
    - anonuid=xxx：将远程访问的所有用户都映射为匿名用户，并指定该用户为本地用户（UID=xxx）
    - anongid=xxx：将远程访问的所有用户组都映射为匿名用户组账户，并指定该匿名用户组账户为本地用户组账户（GID=xxx）
  + 其它选项
    - secure：限制客户端只能从小于1024的tcp/ip端口连接nfs服务器（默认设置）
    - insecure：允许客户端从大于1024的tcp/ip端口连接服务器
    - sync：将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性
    - async：将数据先保存在内存缓冲区中，必要时才写入磁盘
    - wdelay：检查是否有相关的写操作，如果有则将这些写操作一起执行，这样可以提高效率（默认设置）
    - no\_wdelay：若有写操作则立即执行，应与sync配合使用
    - subtree：若输出目录是一个子目录，则nfs服务器将检查其父目录的权限(默认设置)
    - no\_subtree：即使输出目录是一个子目录，nfs服务器也不检查其父目录的权限，这样可以提高效率

#### 4. 服务端及客户端启动NFS

配置好 /etc/exports 文件后，就可以启动NFS服务器了。

* 服务端

创建目录并修改文件夹权限

$ mkdir /home/cluster  
$ chmod 777 /home/cluster #在服务器端打开NFS目录 rwx 权限

/home/cluster 这里也可以选择其他目录，为了方便管理，推荐在 /home 目录下创建NFS目录。

重启NFS服务

$ systemctl restart rpcbind  
$ systemctl restart nfs

* 客户端

将服务端NFS目录挂载到本机相同目录下

$ mount [-t nfs] <hostname>:/home/cluster /home/cluster

#### 5. NFS 访问权限测试

* 测试文件的所有者是否正常
* [root@mic11 cluster]# touch root  
  [root@mic11 cluster]# su lq  
  [lq@mic11 cluster]$ touch lq  
  [lq@mic11 cluster]$ ll  
  总用量 0  
  -rw-rw-r-- 1 lq lq 0 5月 17 23:36 lq  
  -rw-r--r-- 1 root root 0 5月 17 23:36 root
* 关于权限的分析

1. 客户端连接时候，对普通用户的检查

* 如果明确设定了普通用户被压缩的身份，那么此时客户端用户的身份转换为指定用户；
* 如果 NFS 服务端 上面有同名用户，且 UID 和 GID 一致，那么此时客户端登录账户的身份转换为NFS 服务端 上面的同名用户；
* 如果没有明确指定，也没有同名用户，那么此时 用户身份被压缩成nfsnobody；

1. 客户端连接的时候，对root的检查

* 如果设置no\_root\_squash，那么此时root用户的身份被压缩为NFS server上面的>root；
* 如果设置了all\_squash、anonuid、anongid，此时root 身份被压缩为指定用户；
* 如果没有明确指定，此时root用户被压缩为nfsnobody；
* 如果同时指定no\_root\_squash与all\_squash 用户将被压缩为 nfsnobody，如果设置>了anonuid、anongid将被压缩到所指定的用户与组；

#### 6. 相关命令

$ nfsstat

查看NFS的运行状态，对于调整NFS的运行有很大帮助。

$ rpcinfo

查看rpc执行信息，可以用于检测rpc运行情况的工具，利用rpcinfo -p 可以查看出RPC开启的端口所提供的程序有哪些。

$ showmount

-a 显示已经于客户端连接上的目录信息  
-e IP 或者 hostname 显示此IP地址分享出来的目录

$ netstat

可以查看出nfs服务开启的端口，其中 nfs 开启的是2049，rpcbind 开启的是 111，其余则是rpc开启的。

#### 7. 特别注意！！！ 先卸载，再关机！！

卸载NFS目录

$ umount /home/cluster

NFS不太稳定，请注意规范操作。  
若仍未卸载NFS客户端，服务端直接关机会使NFS目录的文件句柄丢失的问题。NFS目录的读写量过大也可能出现崩溃的情况。（ASC16曾经出现过）

NFS 客户端关机的时候一点要确保NFS服务关闭，没有客户端处于连接状态！通过 showmount -a 或 fuser /home/cluster 可以查看，如果有的话通过 kill, killall, pkill, fuser -k /home/cluster 来结束。

#### 8. 通过挂载 tmpfs 提高NFS文件系统IO速度

在IO密集的程序中，磁盘的读写速度会限制程序的运行速度和优化效果。我们可以直接把一部分内存用作NFS目录，以提升IO速度，这种方式的速度要远快于普通机械硬盘，和SSD相比在读写速度上也更有优势。

* 挂载 tmpfs

$ mount -t tmpfs -o size=4G /home/cluster  
$ df -h  
Filesystem Size Used Avail Use% Mounted on  
/dev/mapper/cl-root 50G 19G 32G 37% /  
devtmpfs 12G 0 12G 0% /dev  
tmpfs 12G 2.4M 12G 1% /dev/shm  
tmpfs 12G 322M 12G 3% /run  
tmpfs 12G 0 12G 0% /sys/fs/cgroup  
/dev/sda1 1014M 174M 841M 18% /boot  
/dev/mapper/cl-home 74G 54M 74G 1% /home  
tmpfs 2.4G 0 2.4G 0% /run/user/0  
tmpfs 2.4G 16K 2.4G 1% /run/user/42  
tmpfs 2.4G 0 2.4G 0% /run/user/1000  
tmpfs 4.0G 0 4.0G 0% /home/cluster #成功挂载

也可以直接通过 mount 命令来查看是否成功挂载

$ mount | grep 'cluster'  
tmpfs on /home/cluster type tmpfs (rw,relatime,seclabel,size=4194304k)

注意正确的顺序是：  
1. 先把 tmpfs 挂载到 服务端的NFS目录上  
2. 再把NFS目录共享给客户端，  
3. 客户端挂载NFS目录。

这里先说明了NFS的配置和挂载，相信对NFS熟悉的话，tmpfs 也就不在话在了。

* 卸载 tmpfs

$ umount /home/cluster

注意若使用tmpfs，在卸载后，目录中的数据会丢失，故平时不推荐使用。在需要用到时，务必在卸载之前备份好重要数据。