集群架设

钟毅

厦门大学物理学系复杂系统研究室

2012 年 8 月

前言

集群就是一堆计算机通过某种方式联系在一起的统一的整体.它有一个主节点，负责管理账户，回答用户的请求，以及命令其它计算机即子节点(计算节点)完成某项任务. 为了能够给子节点下命令，我们必须搭建 SSH 服务，并且要达到无密码互访(因为子节点要反馈计算结果或者其它信息给主节点)；为了管理用户账号的方便，我们要建立 NIS 服务，而要建立 NIS 服务，又必须先搭建 NFS 服务. NFS 服务的一项重要的功能是，把主节点的某个或者某些目录共享给子节点，这样，只要在主节点的共享区域安装了某个软件，那么该软件就可以被所有子节点利用.我们复杂系统小组的成员都喜欢编写 Fortran 程序，所以我们有必要安装 Fortran 编译器------ifort 软件，这是 intel 公司提供的免费 Fortran 编译器.又考虑到我们经常要进行大规模模拟计算，这就必须安装 MPICH2 这个并行计算软件. 此外，我们还必须安装 Torque 这个作业管理器，对用户提交的作业进行管理.

有必要说明的是，这里展示的是每种软件的安装步骤，以及遇到问题时的处理方法；这里面没有对各种软件的功能进行详细的解说哟，也没有对 linux 的命令进行解说，要知道它们的功能，请参看最后面列出的参考文献吧. 还有一点必须说明，之所以整理这个安装笔记，是为了让师弟师妹们在自己架设集群的时候有一份比较完整的参考资料. 好了，废话少说，开始动手吧，O(∩_∩)O 哈哈~

一．安装与连接 internet

两台电脑 Hostname 分别为 centos1，centos2

1. 安装 CentOS6.3，分区的时候选择如下方式

   centos1 的分区：

/	:	90GB
/var	:	10GB
/tmp	:	10GB
Swap	:	3GB
/disk1	:	20GB

   centos2 的分区：

/	:	20GB
/var	:	10GB
/tmp	:	10GB
Swap	:	3GB
/disk2	:	90GB

   选择安装 basic server 的安装包。

2. 连接 internet

以 root 用户登录，然后分别做如下的事情

a. 设 IP

```
[root@centos1 ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="static"
BROADCAST="59.77.36.255"
HWADDR="00:26:2D:D0:69:18"
IPADDR="59.77.36.207"
NETMASK="255.0.0.0"
GATEWAY="59.77.36.1"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="001bde85-8379-4fcc-9b4e-8f98260f5893"
```

以上涂了颜色的为新加的

b. 启动网卡

```
[root@centos1 ~]# ifup eth0
[root@centos1 ~]# /etc/init.d/network restart
```

c. 看是否正常启动了

```
[root@centos1 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:26:2D:D0:69:18
          inet addr:59.77.36.207  Bcast:59.77.36.255  Mask:255.0.0.0
          inet6 addr: 2001:da8:e800:100:226:2dff:fed0:6918/64 Scope:Global
          inet6 addr: fe80::226:2dff:fed0:6918/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:747 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:81563 (79.6 KiB)  TX bytes:4544 (4.4 KiB)
          Interrupt:18
```

出现 IP 了就是正常启动了。

d. 设置 DNS 的 IP

```
[root@centos1 ~]# vi /etc/resolv.conf
nameserver 168.95.1.1
nameserver 139.175.10.20
```

e. 检查是否能够解析域名了

```
[root@centos1 ~]# nslookup www.baidu.com
Server:         168.95.1.1
Address:        168.95.1.1#53
Non-authoritative answer:
```

```
www.baidu.com      canonical name = www.a.shifen.com.
Name:      www.a.shifen.com
Address: 220.181.111.147
```

出现 IP 了就是设置成功了。

f. 调整时间,设置成现在的时间

```
[root@centos1 ~]# date -s 07/26/2012
Thu Jul 26 00:00:00 CST 2012
[root@centos1 ~]# date -s 17:10:15
Thu Jul 26 17:10:15 CST 2012
[root@centos1 ~]# clock –w
```

g. 对于 centos2 重复以上步骤就可以了。

现在就可以坐在某一台 WIN7 电脑前，用 putty 同时登录两台电脑了。

centos1 的 IP ： 59.77.36.207
centos2 的 IP ： 59.77.36.208

二．两台电脑间(即节点间)创建 SSH 信任连接

1.   在 centos1 与 centos2 中分别建立可以信任的主机的 Hostname 与它的 IP 的对应

   a. **[root@centos1 ~]# vi /etc/hosts**

```
   127.0.0.1      localhost
   59.77.36.207    centos1
   59.77.36.208    centos2

   [root@centos1 ~]# hostname
   centos1
   [root@centos1 ~]# hostname -i
   59.77.36.207
```

   b. **[root@centos2 ~]# vi /etc/hosts**

```
   127.0.0.1      localhost
   59.77.36.208 centos2
   59.77.36.207 centos1
   [root@centos2 ~]# hostname
   centos2
   [root@centos2 ~]# hostname -i
   59.77.36.208
```

2.   我们的目标是 a) 在节点 centos1 和 centos2 中有相同的用户，以及相同的用户账号密码
   b)可以在任意一节点上直接用当前的账号登陆到其它节点上去，而不用密码

（A）如果两台主机之间没有建立 NFS 与 NIS 服务，则执行下面的步骤：

   1)用 root 账户把 centos1(主结点)上的 /etc/passwd, /etc/shadow/, /etc/group/, /etc/gshadow/ 复制到其它节点上

去。（**[root@centos1 ~]#scp /etc/passwd root@59.77.36.208:/etc/passwd** ）.

2) 执行以下命令

在 centos1 中以 cv 登陆，

a.

```
[cv@centos1 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cv/.ssh/id_rsa):
Created directory '/home/cv/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cv/.ssh/id_rsa.
Your public key has been saved in /home/cv/.ssh/id_rsa.pub.
```

上面涂了颜色的后面直接用回车就可以。

b.

```
[cv@centos1 ~]$ ls -al
drwx------. 2 cv      cv      4096 Jul 27 16:30 .ssh
```

c.

```
[cv@centos1 .ssh]$ ls –al
-rw-------. 1 cv cv 1675 Jul 27 16:30 id_rsa
-rw-r--r--. 1 cv cv    392 Jul 27 16:30 id_rsa.pub
```

在 centos2 中以 cv 登陆，重复以上步骤，最后

```
[cv@centos2 .ssh]$ ls -al
-rw-------. 1 cv cv 1679 Jul 27 16:30 id_rsa
-rw-r--r--. 1 cv cv    392 Jul 27 16:30 id_rsa.pub
```

就是说在 centos1 与 centos2 中的 /home/cv/.ssh/目录下，都产生了私匙 id_rsa 与公匙 id_rsa.pub。我们应该把这两个公匙的内容集中起来，放入同一个文件 authorized_keys 中，然后把这个文件再分别放入 centos1 与 centos2 中的 /home/cv/.ssh/目录下，并且改变权限：

chmod 600 authorized_keys 然后就可以了。用到的命令为（要按顺序进行操作）

```
[cv@centos2 .ssh]$ cp    id_rsa.pub    authorized_keys
[cv@centos1 .ssh]$ scp id_rsa.pub cv@centos2:/home/cv/
[cv@centos2 ~]$ cat id_rsa.pub >> ./.ssh/authorized_keys
[cv@centos2 .ssh]$ scp authorized_keys cv@centos1:/home/cv/.ssh/
```

然后，在 centos2 上

```
[cv@centos2 .ssh]$ ssh centos1
Last login: Fri Jul 27 16:48:12 2012 from centos2
[cv@centos1 ~]$
```

在 centos1 上

```
[cv@centos1 .ssh]$ ssh centos2
Last login: Fri Jul 27 16:29:53 2012 from 59.77.36.239
[cv@centos2 ~]$
```

可见，已经不用密码就相互登录了。

(B) 如果两台主机之间已经建立 NFS 与 NIS 服务，则执行下面的步骤：

1)用 root 用户登录主节点，然后执行以下命令

```
[root ~]# useradd -g complex -m chong01（注：新增用户 chong01,强制建立/home/chong01 目录）
[root ~]# passwd chong01
New UNIX password:
Retype new UNIX password:
passwd: password updated successfully
[root ~]# /usr/lib/yp/ypinit –m    (注：这里是更新主节点上的 NIS 数据库，让子节点也认识新用户)
At this point, we have to construct a list of the hosts which will run NIS
servers.   BlueMoon3 is in the list of NIS server hosts.   Please continue to add
the names for the other hosts, one per line.   When you are done with the
list, type a <control D>.
            next host to add:   BlueMoon3
            next host to add:  （注：这边直接按 ctrl+d 组合键）
The current list of NIS servers looks like this:
BlueMoon3
Is this correct?   [y/n: y]   y
We need a few minutes to build the databases...
Building /var/yp/complexlab/ypservers...
………….
```

2)用新建立的账号 chong01 登录主节点，执行以下命令

```
[chong01 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/chong01/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
open /home/chong01/.ssh/id_rsa failed: Permission denied.（注：权限不够）
Saving the key failed: /home/chong01/.ssh/id_rsa.
[chong01 ~]$ ls -al
drw-------   2 chong01 complex 4096 2012-07-27 16:58 .ssh/
-rw-r--r--   1 chong01 complex   181 2012-07-27 16:58 .vimrc
[chong01 ~]$ chmod   +x   .ssh  （注：修改权限）
[chong01 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/chong01/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/chong01/.ssh/id_rsa.
Your public key has been saved in /home/chong01/.ssh/id_rsa.pub.
The key fingerprint is:
ee:ce:2e:27:07:ec:4d:28:09:32:9c:4e:5f:bf:a6:2e chong01@BlueMoon3
[chong01 ~]$ cd .ssh
[chong01 .ssh]$ ls -al
-rw------- 1 chong01 complex 1675 2012-07-27 17:01 id_rsa
```

```
-rw-r--r-- 1 chong01 complex    399 2012-07-27 17:01 id_rsa.pub
[chong01 .ssh]$ cp id_rsa.pub authorized_keys (注：这一步非常重要)
[chong01 .ssh]$ ssh anna01
The authenticity of host 'anna01 (192.168.215.1)' can't be established.
RSA key fingerprint is 06:21:14:d4:02:f2:3f:d3:40:67:13:d6:32:c2:7c:2e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'anna01,192.168.215.1' (RSA) to the list of known hosts.
Last login: Fri Jul 27 11:04:31 2012 from bluemoon3
[chong01@anna01~ ]$
```

可见，已经可以无密码登录子节点 anna01 了。

说明：1)主节点上架设了 NFS 与 NIS 服务，子节点把主节点的/home 挂载到自己的/home 目录上；所以当主节点上新增用户时，要重新建立数据库；同时，由于 SSH 以无密码方式登录别的节点时，是比较两个节点的$HOME/.ssh/目录下面文件 authorized_keys 里面的内容，所以这个文件一定要建立；又由于主节点与子节点共用/home，因此只要建立这个文件且把公匙放进去就能无密码访问了。特别注意，NIS 的作业在于，主节点与所有子节点上的账号密码都是一样的。

2) 权限问题最重要。比如用 chongyi 登录：

```
[chongyi@centos1 ~]$ ls –al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drwx------. 2 root       root       4096 Jul 28 10:07 test   （即/home/chongyi/下有属于 root 的目录 test）
[chongyi@centos1 ~]$ cd test
-bash: cd: test: Permission denied    (即 chongyi 无权限进入 test 目录)
[chongyi@centos1 ~]$ chmod 777 test
chmod: changing permissions of `test': Operation not permitted (即 chongyi 无权修改 test 目录的权限)
[chongyi@centos1 ~]$ su -  （用户切换成 root）
Password:
[root@centos1 ~]# cd /home/chongyi
[root@centos1 chongyi]# ls -al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drwx------. 2 root       root       4096 Jul 28 10:07 test
[root@centos1 chongyi]# chown -R chongyi:chongyi test (把 test 的账号与用户组都改为 chongyi)
[root@centos1 chongyi]# ls -al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drwx------. 2 chongyi chongyi 4096 Jul 28 10:07 test (目录 test 已经属于 chongyi)
[root@centos1 chongyi]# su - chongyi
[chongyi@centos1 ~]$ ls -al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drwx------. 2 chongyi chongyi 4096 Jul 28 10:07 test
[chongyi@centos1 ~]$ cd test
[chongyi@centos1 test]$             (chongyi 可以进入 test 目录)
[chongyi@centos1 test]$ cd ..
[chongyi@centos1 ~]$ ls -al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drwx------. 2 chongyi chongyi 4096 Jul 28 10:07 test
[chongyi@centos1 ~]$ chmod u-x test    (test 目录去掉 x 的权限)
```

```
[chongyi@centos1 ~]$ ls -al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drw-------. 2 chongyi chongyi 4096 Jul 28 10:07 test
[chongyi@centos1 ~]$ mkdir ./test/test2
mkdir: cannot create directory `./test/test2': Permission denied (无 x 权限后无法在 test 目录下新建东西)
[chongyi@centos1 ~]$ chmod u+x test (test 目录加上 x 的权限)
[chongyi@centos1 ~]$ ls -al
drwx------. 2 chongyi chongyi 4096 Jul 27 15:41 .ssh
drwx------. 2 chongyi chongyi 4096 Jul 28 10:07 test
[chongyi@centos1 ~]$ mkdir ./test/test2 (在 test 目录下新建目录 test2)
[chongyi@centos1 ~]$ cd test
[chongyi@centos1 test]$ ls -al
drwxrwxr-x. 2 chongyi chongyi 4096 Jul 28 10:25 test2 (在 test 目录下成功建立了目录 test2)
```

分析下面的权限：
drwxr-xr--. 2 cv complex 4096 Jul 28 10:25 test9
d:代表 test9 是目录
test9 属于用户 cv 的，test9 属于 complex 组的
cv 对 test9 的权限为 rwx
complex 组的成员对 test9 的权限为 r-x
其它别的用户，别的组的成员对 test9 的权限为 r--
3)安装套件或者执行过程中遇到问题时，首先检查文件或者目录的所有者以及所在的用户组，看它们的权限是什么；其次，要检查该文件或者目录的权限是不是套件所允许的，比如上面的 authorized_keys ，SSH 只允许它的权限是 600 时，才可以使用 SSH 无密码互访；再次，要检查该文件或者目录的所有者以及所在的用户组是不是套件所允许的，比如 PBS 作业管理系统就是不允许 root 提交作业；当然，还要检查防火墙中是否开了套件需要的端口，以及套件之间相互依赖的套件是否有启动等等。
4)在添加新用户时，要强制建立用户 home 目录，所以 useradd 命令要用-m 参数；另外在删除用户时，一定要连同用户的家目录一起删除，如 userdel –r username ，不然的话，删除一个用户后，再添加一个新的用户时，有的系统会把这个新用户安排在被删除用户的家目录中，这样会产生一堆的权限问题。

三．建立 NFS 服务

建立之前要明确几件事：
1)　NFS 服务器端与 NFS 客户端的账号关系
a. 每个账号都分配有 UID 与 GID，这个可以通过/etc/passwd 查看

```
[root@centos1 ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
chongyi:x:500:500::/home/chongyi:/bin/bash
cv:x:501:501::/home/cv:/bin/bash
```

账号：密码(x 代表已经转移到/etc/shadow)：UID：GID：用户信息说明：家目录：默认使用的 shell
文件系统的 inode 记录的是 UID 与 GID，不是账号与组名，即一个账号能不能读取文件，取决于它的 UID 与 GID.
b. 不同的主机，即使建立了相同的账号，但是主机分配给它们的 UID 与 GID 不一定是相同的.
c. 客户端进入服务器端读取文件时，客户端无论以什么账号登陆，最后都会映射成为服务器主机上面的一个账号，

如，客户端以 cv 账号，UID=501 登录服务器端，如果服务器端 UID=501 对应的账号为 cv01，那么客户端的身份变为 cv01，如果 UID=501 在服务器端不存在，则此时服务器端将把客户端压缩为匿名者，分配给它的账号为 nfsnobody ,而且 UID=65534.

d. 客户端进入服务器端读取文件的权限，取决于两个方面，一个方面是它被映射成为服务器端上账号所能读取的权限；另外一方面取决于服务器端/etc/exports 中规定的客户端的权限；一旦任何一方面给它限制说不能进行什么样的操作，则就不能进行那方面操作.

e. 为了让客户端的账号与服务器端的账号一一对应，就必须引进 NIS 服务器了.

2) 设置开机自启动某项服务

通常可以用 chkconfig , ntsysv 命令或者在/etc/rc.d/rc.local 中写入启动某项服务的命令即可：

a. 用 chkconfig 命令启动某项服务，它是有前提的，就是该服务的运行脚本必须放在/etc/init.d/中，可以用如下命令查看 chkconfig 管理的服务中是否有你需要的,

```
[root@centos1 ~]# chkconfig --list |more
abrt-ccpp        0:off   1:off   2:off   3:on    4:off   5:on    6:off
abrt-oops        0:off   1:off   2:off   3:on    4:off   5:on    6:off
……………………………………………………………………….
```

你也可以自己写一个脚本，即某个服务，先放入/etc/init.d/中，然后用相关的命令把它转化为 chkconfig 管理

开启[关闭]服务的命令为

```
[root@centos1 ~]# chkconfig  - -level  [0123456]  [service_name]  [on|off]
```

再用下面命令确认是否已经成功

```
[root@centos1 ~]# chkconfig  - -list | grep  [service_name]
```

其中，等级 0,1,2,3,4,5,6 是系统的运行等级，不同等级意味着系统开机初始化时开启的服务是不一样的，比如等级 3 对应的是“完整的含有网络功能的纯文本模式”，等级 5 是“完整的含有网络功能的 X Window 模式”.可以用下面的命令查看当前你的主机运行等级

```
[root@centos1 ~]# runlevel
N 3
```

实例：

```
[root@centos1 ~]#chkconfig --level 35 ypserv on
```

它代表无论是系统运行在等级 3 还是等级 5，在开机初始化过程中自动开启 ypserv 服务.

b. 由于/etc/rc.d/rc.local 是一个脚本，它是系统开机初始化完成之后，去读取其中的命令的，因此只需要把开启某项服务的命令写到里面去就可以了.例如，开机后自动关闭防火墙可以这样子：

```
[root@centos1 ~]# vi /etc/rc.d/rc.local
#!/bin/sh
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
touch /var/lock/subsys/local
service   iptables   stop (新加入的内容)
```

c. 用 chkconfig 命令启动的服务先于用/etc/rc.d/rc.local 开启的服务

d. 有好多个服务，如果有规定开启的顺序，则，如果用 chkconfig 命令，那么必须在/etc/init.d/中找到这些服务的脚本，然后打开它，找到下面那一行

chkconfig: xx xx xx ，其中 xx 是数字

修改第二组数字 xx 的值，就可以规定这些服务的顺序了(修改之后必须用命令重新把这些服务加入 chkconfig 管理).

如果是用/etc/rc.d/rc.local 管理，由于它是一个脚本，因此放在前面的就首先开启啦.
e. 当有多个服务，要求规定开启顺序时，推荐使用/etc/rc.d/rc.local，因为用 chkconfig 要修改各个服务的脚本，这会引起混乱(万一一个服务还和别的服务有先后依赖关系，则混乱了).

3) 开启|关闭防火墙的命令有：
[root@centos1 ~]# /etc/init.d/iptables status (查看状态)
[root@centos1 ~]# /etc/init.d/iptables stop （关闭）
[root@centos1 ~]# /etc/init.d/iptables start （开启）
或者
[root@centos1 ~]# service iptables status
[root@centos1 ~]# service iptables stop
[root@centos1 ~]# service iptables start (当设置了开机自动关闭后，可以用这命令重新开启防火墙)
开机自动关闭可以：
[root@centos1 ~]# chkconfig - -level 35 iptables off
或者，在/etc/rc.d/rc.local 中写入
/etc/init.d/iptables stop
或者写入
service iptables stop
4) 什么是命令？其实命令只是一个可执行的脚本而已；什么是命令的参数？其实它们只是脚本的参数而已.
例如，
[root@centos1 ~]# which man (对应 "脚本 脚本参数$0 $1 $2 ……")
/usr/bin/man (脚本输出值)
命令行的 which 其实就是一个可执行脚本，而 man 是第一个参数$0 . 之所以在任何目录下都可以用 which ,因为它把 which 这个脚本的绝对路径加入了 PATH 这个全局环境变量中；所以，当你自己写了一个可执行脚本，为了让它在所有目录下都可以执行，只需把该脚本所在的绝对路径加入 PATH 中.

5)安装 NFS
a. 在 centos1 也就是主节点，上安装 NFS Server
[root@centos1 ~]# yum install rpcbind (centos6.3 版本中，portmap 变改名为 rpcbind，是 RPC 主程序)
[root@centos1 ~]# yum install nfs-utils (是 NFS 的核心，主程序)
[root@centos1 ~]# mkdir -p /cluster/server (设置/disk1 的挂载点)

[root@centos1 ~]# vi /etc/exports (设置共享内容)
/home 59.77.36.208(rw,async,no_root_squash)
/disk1 59.77.36.208(rw,async,no_root_squash) 59.77.36.207(rw,async,no_root_squash)

[root@centos1 ~]# /etc/init.d/rpcbind start (启动 RPC)
[root@centos1 ~]# vi /etc/rc.d/rc.local (设置成开机自启动)
```
#!/bin/sh
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
touch /var/lock/subsys/local
/etc/init.d/rpcbind start
```

[root@centos1 ~]# exportfs –rv　　　　(挂载/etc/exports 的设置)

```
exporting 59.77.36.208:/disk1
exporting 59.77.36.207:/disk1
exporting 59.77.36.208:/home
```

[root@centos1 ~]# /etc/init.d/nfslock start　(开启 nfslock,并且设为开机自启动)

```
Starting NFS statd:                                         [  OK  ]
```

[root@centos1 ~]# /etc/init.d/nfs start　　(启动 NFS)

```
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
Stopping RPC idmapd:                                       [  OK  ]
Starting RPC idmapd:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
```

[root@centos1 ~]# vi /etc/rc.d/rc.local　　(设置成开机自启动)

```
…………………………………….
touch /var/lock/subsys/local
/etc/init.d/rpcbind start
/etc/init.d/nfslock stop
/etc/init.d/nfslock start
/etc/init.d/nfs stop   (先关掉)
/etc/init.d/nfs start   (再开启)
```

注：为什么先关掉再开启，因为怕不小心用 chkconfig 设了 nfs，这样 nfs 会先于 RPC 启动，可是 RPC 必须先于 NFS 启动，这样 NFS 才可以向 RPC 注册.

[root@centos1 cluster]# mount -t nfs -o rw,bg,soft 59.77.36.207:/disk1 /cluster/server

[root@centos1 disk1]# ls –al　(查看/disk1 的内容)

```
drwxr-xr-x.   2 root root   4096 Jul 30 11:10 haha
drwx------.   2 root root 16384 Jul 26 06:46 lost+found
```

[root@centos1 disk1]# cd /cluster/server

[root@centos1 server]# ls –al (/cluster/server 中的内容和/disk1 下的一样，挂载成功)

```
drwxr-xr-x. 2 root root   4096 Jul 30 11:10 haha
drwx------. 2 root root 16384 Jul 26 06:46 lost+found
```

[root@centos1 server]# vi /etc/rc.d/rc.local (加入开机自动挂载中)

```
…………………………..
/etc/init.d/nfs start
mount -t nfs -o rw,bg,soft 59.77.36.207:/disk1 /cluster/server
```

b. 在 centos2 也就是子节点，上安装 NFS Client

[root@centos2 ~]# yum install rpcbind　　(centos6.3 版本中，portmap 变改名为 rpcbind，是 RPC 主程序)

[root@centos2 ~]# yum install nfs-utils　　(是 NFS 的核心，主程序)

[root@centos2 ~]# mkdir -p /cluster/server　　(设置/disk1 的挂载点)

[root@centos2 ~]# /etc/init.d/rpcbind start

```
[root@centos2 ~]# /etc/init.d/nfslock start
[root@centos2 ~]# /etc/init.d/nfs start

[root@centos2 ~]# showmount -e 59.77.36.207  （检查是否连接成功）
clnt_create: RPC: Port mapper failure - Unable to receive: errno 113 (No route to host)
```
以上说明出错了，在于防火墙没有设置，因此，在 centos1 和 centos2 上都关闭防火墙
```
[root@centos1 ~]# service iptables stop
[root@centos2 ~]# service iptables stop
[root@centos2 ~]# showmount -e 59.77.36.207
Export list for 59.77.36.207:
/disk1 59.77.36.207,59.77.36.208
/home    59.77.36.208
[root@centos2 ~]# mount -t nfs -o rw,bg,soft 59.77.36.207:/home /home
[root@centos2 ~]# mount -t nfs -o rw,bg,soft 59.77.36.207:/disk1 /cluster/server
[root@centos2 ~]# cd /cluster/server (检查是否挂载成功)
[root@centos2 server]# ls –al (centos2 中/cluster/server 里面的内容和 centos1 中/disk1 下的内容一样，成功了)
drwxr-xr-x. 2 root root    4096 Jul 30 11:10 haha
drwx------. 2 root root 16384 Jul 26 06:46 lost+found
[root@centos2 server]# vi /etc/rc.d/rc.local (把以上内容放入开机自启动中)
#!/bin/sh
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
touch /var/lock/subsys/local
service iptables stop
/etc/init.d/rpcbind start
/etc/init.d/nfslock stop
/etc/init.d/nfslock start
/etc/init.d/nfs stop
/etc/init.d/nfs start
mount -t nfs -o rw,bg,soft 59.77.36.207:/home /home
mount -t nfs -o rw,bg,soft 59.77.36.207:/disk1 /cluster/server
```
并且，一定要把 service iptables stop 写入 centos1 中/etc/rc.d/rc.local 中

开机，重启两台电脑，看能否创建文件
```
[root@centos2 ~]# cd /cluster/server
[root@centos2 server]# ls -al
drwxr-xr-x. 2 root root    4096 Jul 30 11:58 haha
drwx------. 2 root root 16384 Jul 26 06:46 lost+found
[root@centos2 server]# cd haha
[root@centos2 haha]# ls -al
[root@centos2 haha]# mkdir haha2
[root@centos2 haha]# ls -al
drwxr-xr-x. 2 root root 4096 Jul 30    2012 haha2
```

```
[root@centos1 ~]# cd /disk1/haha
[root@centos1 haha]# ls -al
drwxr-xr-x. 2 root root 4096 Jul 30 11:59 haha2
```

由上面可以看出，开机重启后，能够在 centos2 中进入并且创建目录,在 centos1 的/disk1/haha/目录下多了 haha2 目录了.

特别注意：1)在 mount 的时候，mount 的参数不要设置成 noexec,nosuid,nodev 等，否则就没有权限创建目录文件，也没有权限挂载 U 盘等，会出现一堆权限问题.

2)当 centos2 把 centos1 的/home 挂载在自己的/home 目录下的时候，这个时候不要在 centos2 终端添加新用户，理由是：centos2 会把新用户的家目录建立在 centos1 的/home 下面，而不是自身的/home 目录里面，另外，会把密码等写人自己的/etc/passwd 等文件里面。这就造成了一个用户的信息被分散到不同的主机上面，会造成使用上权限上各种各样的麻烦. centos2 应该先卸载掉 centos1 的/home ,然后再建立新用户.

例如：

```
[root@centos2 ~]# df
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/sda6              20642428   1546720  18047132   8% /
tmpfs                    951776         0    951776   0% /dev/shm
/dev/sda5              52410948    184276  49564272   1% /disk2
/dev/sda8              10321208    154244   9642676   2% /tmp
/dev/sda9              10321208    223852   9573068   3% /var
59.77.36.207:/home    103212544   1562880  96406784   2% /home
59.77.36.207:/disk1    20158464    176128  18958336   1% /cluster/server
[root@centos2 ~]# umount /home   (卸载 centos1 的/home)
[root@centos2 ~]# df
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/sda6              20642428   1546720  18047132   8% /
tmpfs                    951776         0    951776   0% /dev/shm
/dev/sda5              52410948    184276  49564272   1% /disk2
/dev/sda8              10321208    154244   9642676   2% /tmp
/dev/sda9              10321208    223856   9573064   3% /var
59.77.36.207:/disk1    20158464    176128  18958336   1% /cluster/server
```

可见，卸载成功了.

```
[root@centos2 home]# ls -al
drwx------.  3 chongyi chongyi 4096 Jul 27 15:42 chongyi
drwx------.  3 cv      cv      4096 Jul 27 20:41 cv
[root@centos2 home]# useradd cv01
[root@centos2 home]# passwd cv01
Changing password for user cv01.
New password:
BAD PASSWORD: it does not contain enough DIFFERENT characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@centos2 home]# ls -al
drwx------.  3 chongyi chongyi 4096 Jul 27 15:42 chongyi
drwx------.  3 cv      cv      4096 Jul 27 20:41 cv
```

```
drwx------.   2 cv01       cv01       4096 Jul 30 14:47 cv01
```
可见，centos2 的/home 目录下多了 cv01 用户的家目录

```
[root@centos1 home]# ls -al
drwx------.   3 chongyi chongyi 4096 Jul 28 11:18 chongyi
drwx------.   3 cv          cv          4096 Jul 27 16:50 cv
```
此时，centos1 中没有 cv01 这个用户.

在 centos2 中卸载 centos1 的/home 后，用 cv01 登录 centos2，然后执行以下命令

```
[cv01@centos2 ~]$ cd /cluster/server/haha/haha2
[cv01@centos2 haha2]$ ls -al
drwxr-xr-x. 2 root root 4096 Jul 30 11:59 .
drwxr-xr-x. 3 root root 4096 Jul 30 11:59 ..
[cv01@centos2 haha2]$ touch haha3
touch: cannot touch `haha3': Permission denied (发现没有权限在共享目录中创建文件 haha3)
[root@centos2 ~]# cd /cluster/server/haha
[root@centos2 haha]# ls -al
drwxr-xr-x. 2 root root 4096 Jul 30 11:59 haha2 (原来 haha2 目录属于 root 用户，别的用户没有 w 权限)
[root@centos2 haha]# chmod o+w haha2    (给别的用户加上 w 的权限)
[root@centos2 haha]# ls -al
drwxr-xrwx. 2 root root 4096 Jul 30 11:59 haha2
[cv01@centos2 haha2]$ touch haha3 (再次创建 haha3 文件)
[cv01@centos2 haha2]$ ls -al
-rw-rw-r--. 1 nobody nobody       0 Jul 30 14:59 haha3 (创建成功)
```
值得注意的是，此时，haha3 这文件的所有者为 nobody,所在的用户组也为 nobody.这是因为 centos1 中没有与 centos2 中用户 cv01 的 UID 相同的用户.

3)一定要记住，在新创建一个新用户的时候，一定要把新用户的家目录，密码等所有原始信息放在同一台主机里面，否则会引起混乱.根据这个原则，当 centos2 挂载了 centos1 的/home 目录的时候，就不要在 centos2 上创建新用户了,应该在 centos1 上创建新的用户,如果要让这个新用户在 centos2 上也可以登录，则只能借助 NIS 服务器了.

4)每台主机，在运行的时候都只是读取自身的资源，除了挂载了别的主机的那一部分.比如，centos2 把 centos1 的/home 挂载在自己的/home 目录下,那么 centos2 读取/home 数据时，读到的是 centos1 中的/home 下的数据，其它没有挂载的部分，任然读取 centos2 自身的数据,比如读取自身的/etc 下面的数据.

5)在创建 NFS 服务的时候，记住各种软件的开启顺序，比如

```
[root@centos2 ~]# /etc/init.d/rpcbind start      (最先开启)
[root@centos2 ~]# /etc/init.d/nfslock start       (第二)
[root@centos2 ~]# /etc/init.d/nfs start              (第三)
[root@centos2 ~]# mount -t nfs -o rw,bg,soft 59.77.36.207:/disk1 /cluster/server    (第四)
```
还要注意，先关闭防火墙后才能 mount,否则不成功.这个在设置开机自启动的时候特别要注意.

当然咯，我们可以不必关掉防火墙，只要开启以上这些软件所需要的端口就好了，这里面为了设置方便，就直接关掉防火墙了.

四．建立 NIS 服务

NIS 的功能是，所有节点上的账号密码家目录等用户最原始的信息都建立在 NIS 服务器上面，而在子节点上不需要建立.而由 NIS 服务器管理的这些账号却可以在所有节点上登录.

建立 NIS 服务的时候，必须建立好了 NFS 服务,并且/home 目录已经共享出去了.所以，这个时候请不要在 NIS 的

客户端用 root 添加新用户，否则会很混乱的.

a. 在 centos1 也就是主节点，上安装 NIS Server

[root@centos1 ~]# yum install yp-tools

[root@centos1 ~]# yum install ypbind

[root@centos1 ~]# yum install ypserv

[root@centos1 ~]# nisdomainname cluster (设置 NIS 的域名为 cluster,并且把这行命令写入开机自启动中)

[root@centos1 ~]# vi /etc/sysconfig/network (只要启动 NIS，就设置好了 NIS 域名,则写入这个文件中)

```
NETWORKING=yes
HOSTNAME=centos1
NISDOMAIN=cluster (新增加的内容)
```

[root@centos1 ~]# vi /etc/ypserv.conf (设置客户端的查询权限)

```
127.0.0.0/255.255.255.0       : *          : *                    : none
59.77.36.207                  : *          : *                    : none
59.77.36.208                  : *          : *                    : none
*                             : *          : *                    : deny
```

[root@centos1 ~]# touch /etc/netgroup

[root@centos1 ~]# /etc/init.d/ypserv start (把这行命令写入开机自启动中)

```
Starting YP server services:                                [  OK  ]
```

[root@centos1 ~]# /etc/init.d/yppasswdd start (把这行命令写入开机自启动中)

```
Starting YP passwd service:                                 [  OK  ]
```

[root@centos1 ~]# /usr/lib/yp/ypinit –m   (制作数据库)

```
…………………………………………….
        next host to add:   centos1
        next host to add: (这边按下  Ctrl+d  组合键)
The current list of NIS servers looks like this:
centos1
Is this correct?   [y/n: y]   y
……………………………….
centos1 has been set up as a NIS master server.
Now you can run ypinit -s centos1 on all slave server.
```

[root@centos1 ~]# /etc/init.d/ypserv restart (重启 ypserv 服务)

```
Stopping YP server services:                                [  OK  ]
Starting YP server services:                                [  OK  ]
```

[root@centos1 ~]# /etc/init.d/yppasswdd restart (重启 yppasswdd 服务)

```
Stopping YP passwd service:                                 [  OK  ]
Starting YP passwd service:                                 [  OK  ]
```

这样，NIS 服务器端就设置好了.

注意:每次添加用户，或者用户密码更改等，都要重新制作数据库，并且要重启 ypserv 和 yppasswdd 服务.

b. 在 centos2 也就是子节点，上安装 NIS Client

[root@centos2 ~]# yum install yp-tools

[root@centos2 ~]# yum install ypbind

[root@centos2 ~]# nisdomainname cluster (把这一行命令放入/etc/rc.d/rc.local 中，变成开机自启动)

[root@centos2 ~]# vi /etc/sysconfig/network (加入下面内容，使得在启动 ypbind 时，就是设置好了 NIS 域名)

```
NETWORKING=yes
HOSTNAME=centos2
NISDOMAIN=cluster (新加的内容)
```

[root@centos2 ~]# vi /etc/yp.conf (建立 NIS 查询的主机的名称)

```
domain cluster
ypserver centos1
```

[root@centos2 ~]# vi /etc/passwd (修改密码验证方式，最后一行加入)

```
+::::::
```

[root@centos2 ~]# vi /etc/nsswitch.conf

```
passwd:       files nis nisplus
shadow:       files nis nisplus
group:        files nis nisplus
hosts:        files nis dns
```

[root@centos2 ~]# /etc/init.d/ypbind start (启动 NIS，并且加入开机自启动中)

```
Starting NIS service:                                    [  OK  ]
Binding NIS service:                                     [  OK  ]
```

可以用 yptest,ypwhich –x, ypcat 来检验是否设置成功，如果成功，则说明 NIS Server/Client 已经同步了,这些命令的用法请查看参考文献.

注意:1)客户端能够修改的是自己的密码，登录用的 shell ，这些用 yppasswd,ypchsh,ypchfn 来处理.如果要新添加或者删除用户，则必须在 NIS 的服务器端.

2)特别注意，在使用 NIS 的时候，要保证 NIS 的客户端除了 root 和系统自身的用户外，没有其它的普通用户，即使之前有普通用户，那最好能把它删除掉，有的删除不干净，则用 vi /etc/passwd 和 vi /etc/group 把它们删除干净，或者，你必须保证客户端和主机端已经有的用户的 UID,GID 还有各种原始信息都一样才行,否则引起混乱.比如,在创建中，centos1 中有普通用户 chongyi 和 cv，centos2 中有 chongyi 和 cv 和 cv01.当我在 centos1 中新加入 cv05 用户时,并且重新制作数据库后,用 cv05 在 centos2 上登录,ls –al 后发现 cv05 变成 cv01 了,好奇怪!后来在 centos2 端 umount /home 后，强制删除 chongyi，cv,cv01，并且把所有普通用户组都删除(进入 vi /etc/group 删除)后,再次重新制作数据库，然后在 centos2 上用 cv05 登录，就正常了.(可能是因为:1. centos1 和 centos2 仅仅共享/home 目录的内容，然而关于/etc/passed 等的内容是不共享的. 2. 由于是相同的版本，而且建立用户顺序相同，因此在 centos1 上新建的 cv05 与 centos2 中已经有的 cv01 的 UID,GID 都相同，所以，根据主机先阅读本主机，然后再阅读共享的内容.对于 NIS,则先阅读本主机的/etc/passwd,/etc/group,然后再阅读 NIS 主机上的内容.所以,当 cv05 在 centos2 上登录时，对比一下 UID,GID，发现这是 cv01 的，所以就把 cv01 的所有其它数据都给 cv05 啦，也认为是 cv01 在登录啦.)

五．安装 ifort 编译器

Intel 编译器功能强大，兼容性能好，它提供有支持 C , C++ 的编译器 icc ,以及支持 Fortran77 , Fortran90 的编译器 ifort. 此外，它还提供了功能齐全的数学库 MKL . 我们可以通过下面的网站下载到免费版的软件:
下载地址：http://software.intel.com/en-us/articles/non-commercial-software-download/

在安装之前，需要明确几件事：
1) 要区分“系统设置值”和“个人设置值”.
a. 系统设置值是所有用户都能够读取的设置文件里面的设置值，比如经常使用的是/etc/profile ------ 在这里可以

设置 PATH ，HOSTNAME ，HISTSIZE ，umask 等. 通常我们希望所有用户都能够使用某个命令或者脚本，我们会把该命令脚本的绝对路径放到 PATH 中，并且写入/etc/profile 里面.

b. 个人设置值，是指放在每个用户自己家目录下面的设置文件里的设置值.最经常使用的是$HOME/.bashrc ------这里也可以设置 PATH 等. 要注意的是，该设置值只对该用户有效.

c. 每个用户登录后，首先读取系统设置值，然后读取自己目录下的个人设置值，并且个人设置值是最终设置值. 使用个人设置值，使得同一台主机，不同的用户，却可以有不同的使用环境.

d. 特别强调，对于 root 用户，它的$HOME 是/root，即~ ,所以 root 的个人设置值放在~/.bashrc 中. 对于普通用户，比如 cv05 的$HOME 是/home/cv05 ，所以 cv05 的个人设置值在/home/cv05/.bashrc 中. 从这边也看出，安装软件的时候，如果希望大家都能用，那就不要安装在/home/下面，因为/home/是放置每个用户的家目录的地方,会产生很多权限问题.也不要安装在~/下面，因为那是 root 的家目录所在地，普通用户更是没有权限读取.

2) shell 变量以及变量的值.

a. 变量，它是一个符号，用来代表某种容易变化的真实的东西. 比如 PATH 就是一个变量，它代表这各种各样的脚本所在的绝对路径.而变量的取值是$PATH.

b. 如何让自定义变量转化为环境变量呢？使用命令： export 变量 XX ，就可以把 XX 变成环境变量啦.环境变量的功能是主程序，和一系列子程序，只要出现 XX ,就都用这个设置好的值$XX .

3) 如何给 PATH 变量添加一条新的路径，比如新添加一条/home/mpi/bin 路径？可以用以下两种方法

a. PATH=$PATH:/home/mpi/bin

b. PATH=``$PATH``:/home/mpi/bin

它们都代表在原来 PATH 值$PATH 的基础上再添加/home/mpi/bin .

注意，在一个设置文件中，给变量 XX 赋值完成后，只能出现一个“export 变量 XX ”.比如，在/etc/profile 中，我们加入了 PATH=$PATH:/home/mpi/bin , 又加了一行 PATH=$PATH:/home/mpi/sbin ,这样给 PATH 赋值完成之后，才加入一行 export PATH .

4) 如何让赋值立即生效？比如在/etc/profile 或者在$HOME/.brashrc 中修改了某个变量的值，那么如何让这些设置立即生效呢？利用以下命令即可：

```
source    /etc/profile
source    $HOME/.bashrc
```

注意，如果不用上面的命令，那么只有重启或者下次登录的时候才能使设置值生效.

5) 在/etc/profile 或者在$HOME/.bashrc 中的设置值是长期有效的，除非你重新修改里面的设置值. 而在命令行对某个变量赋值，那是暂时性的，它仅仅对这次登录有效，下次重启登录后就无效了.比如，在命令行写下以下内容

```
[root@centos2 ~]# export PATH=$PATH:/home/mpi/bin
```

那么，这个路径/home/mpi/bin 仅仅在这次登录时加入了 PATH，下次登录时，PATH 中已经没有它了 .

6) 为了更加方便管理系统，建议建立以下文件夹

a. /root/software (这是压缩包的放置位置)

b. /usr/local/src/XX (其中 XX 代表软件的名称，这是解压后的放置位置)

c. 再自定义一个 XX 软件的安装目录 .千万别把所有软件都安装在同一个目录下，否则删除的时候就麻烦了.也不要用默认的安装目录，否则也难以管理.

7) 安装完成后，必须把该软件的可执行脚本所在的路径告诉 PATH,放在/etc/profile 或者$HOME/.bashrc 中；同时要把它的帮助文件的位置告诉 MANPATH , 放在/etc/man.config 里面 ，比如在里面加一行

```
MANPATH    /cluster/server/program/mpich2/man
```

同时，必须让它们立即生效.

8) 对于多节点，就是多台主机安装软件的问题，要考虑以下的几个方面

a. 如果该软件是不分主节点与子节点的，换言之，在主节点与子节点上，该软件有相同的目录，而且设置值都一样，那么只需要把它安装在主节点的共享目录里，然后只需在各个节点上设置 PATH 与 MANPATH 就可以了.

b. 如果该软件在主节点和子节点上的设置是有差别的，那么要分别在主节点与子节点上安装，在子节点上安装的时候，要 umount /home 哟.

c. 总而言之，一定要确保软件的完整性，就是软件的所有内容，只安装在同一台主机上，不要放在不同主机上面.

9) 安装 ifort

a. 在 centos1 也就是主节点，上安装 ifort

[root@centos1 ~]# yum install gcc*   (安装 gcc , gfortran 等 这是安装 ifort 必须的编译环境)

```
……………………………………….
Installing : gcc-c++-4.4.6-4.el6.i686
Installing : gcc-objc-4.4.6-4.el6.i686
Installing : gcc-objc++-4.4.6-4.el6.i686
Installing : gcc-java-4.4.6-4.el6.i686
Installing : gcc-gnat-4.4.6-4.el6.i686
Installing : gcc-gfortran-4.4.6-4.el6.i686
………………………………………….
```

[root@centos1 ~]# cd /root/software (压缩包放置位置)

[root@centos1 software]# ls -al

```
-rw-r--r--. 1 root root 385362545 Jul 31 18:39 l_fcompxe_ia32_2011.11.339.tgz
```

[root@centos1 software]# cd /usr/local/src/ifort

[root@centos1 ifort]# tar -zxvf /root/software/l_fcompxe_ia32_2011.11.339.tgz

[root@centos1 ifort]# ls -al

```
drwxr-xr-x. 4 root root 4096 Jun 15 18:55 l_fcompxe_ia32_2011.11.339
```

[root@centos1 ifort]# cd l_fcompxe_ia32_2011.11.339 (解压后的位置)

[root@centos1 l_fcompxe_ia32_2011.11.339]# ./install.sh

```
Your system is protected with Security-enhanced Linux* (SELinux).
…………………………………………………………….
You may disable SELinux by either:
      - changing your /etc/sysconfig/selinux file.   Change the line containing
         "SELINUX=enforcing" to "SELINUX=disabled" or to "SELINUX=permissive".
      - OR changing your lilo.conf or grub.conf file. Add the "selinux=0"
         kernel argument.
```

[root@centos1 l_fcompxe_ia32_2011.11.339]# vi /etc/sysconfig/selinux

```
……..
SELINUX=disabled
…………….
```

然后重新启动系统，然后

[root@centos1 l_fcompxe_ia32_2011.11.339]# ./install.sh

然后选择一路回车就好了，当然最重要的地方，就是有提示选择自定义安装路径的，可以选择安装在共享的目录里面.我这里是选择了默认安装目录

安装目录：/opt/intel/composer_xe_2011_sp1.11.339

安装完成后，要设置环境变量，而安装过程中它有提示怎么设置：

```
…………………………………………………………………….
To get started using Intel(R) Composer XE 2011 Update 11 located in
/opt/intel/composer_xe_2011_sp1.11.339:
- Set the environment variables for a terminal window using one of the following
```

```
   (replace "intel64" with "ia32" if you are using a 32-bit platform).
      For csh/tcsh:
         $ source install-dir/bin/compilervars.csh intel64
      For bash:
         $ source install-dir/bin/compilervars.sh intel64
      To invoke the installed compilers:
         For C++: icpc
         For C: icc
         For Fortran: ifort
…………………………………………………………………
```
[root@centos1 /]# vi /etc/profile　(设置环境变量，这样在各个路径下都可以用了)

```
………………………….
unset i
unset pathmunge
source /opt/intel/composer_xe_2011_sp1.11.339/bin/ compilervars.sh ia32　(新加的内容)
```
[root@centos1 /]# source /etc/profile (使设置值生效)

[root@centos1 chongyi]# ls -al

```
-rw-r--r--    1 root       root       3587 Jul 31 19:31 fpu0d6-100.f90
```
[root@centos1 chongyi]# ifort -o fpu0d6-100 fpu0d6-100.f90

[root@centos1 chongyi]# ls -al

```
-rwxr-xr-x    1 root       root       533305 Jul 31 21:20 fpu0d6-100
-rw-r--r--    1 root       root         3587 Jul 31 19:31 fpu0d6-100.f90
```
[root@centos1 chongyi]# ./fpu0d6-100

[root@centos1 chongyi]# ls –al

```
……………………………….
-rw-r--r--    1 root       root         4900 Jul 31 21:21 600Ek.dat
-rw-r--r--    1 root       root         4851 Jul 31 21:21 650flux.dat
-rw-r--r--    1 root       root           49 Jul 31 21:21 670averflux.dat
-rwxr-xr-x    1 root       root       533305 Jul 31 21:20 fpu0d6-100
-rw-r--r--    1 root       root         3587 Jul 31 19:31 fpu0d6-100.f90
```
[root@centos1 chongyi]# chown chongyi:chongyi fpu0d6-100.f90

然后换成 chongyi 登录，然后仍然能够编译执行输出正确的结果,所以安装成功了.

[root@centos1 composer_xe_2011_sp1.11.339]# vi /etc/man.config (把 ifort 自带的 man 文件放入 MANPATH 中)

```
MANPATH /opt/intel/composer_xe_2011_sp1.11.339/man
```

b. 在 centos2 也就是子节点，上安装 ifort

由于在主节点上我把 ifort 安装在非共享文件里了，所以子节点也就必须装 ifort，只要重复以上步骤就可以啦. 如果 ifort 装在主节点的共享文件里，那么只要在主节点和子节点设置一下 PATH , MANPATH 就可以啦,子节点就没有必要安装了.

注意：1)要卸载 ifort 的话，要到安装目录/opt/intel/composer_xe_2011_sp1.11.339/bin 里面，找到 uninstall.sh ，然后执行它就可以卸载，卸载完成后把/opt/intel 里面所有内容删除就好了.

2)请用 man ifort 查看 ifort 的用法.

如果用 putty 查看有乱码，则双击 putty------->Category----→Window----→Translation 然后在选项框“Received data assumed to be in which character set ” 中选择 “UTF-8” ， 然后点击 Category 中的 Session 重新登录主机就好了.

六．安装 MPICH2 并行计算软件

MPICH2 是并行计算软件，可以在下面的地址中下载到源代码：
下载地址：http://www.mcs.anl.gov/research/projects/mpich2/index.php
a. 在 centos1 也就是主节点，上安装 MPICH2

```
[root@centos1 ~]# export FC=ifort (这里是声明环境变量，编译 MPICH2 的时候要和 ifort 绑定在一起)
[root@centos1 ~]# export F90=ifort
[root@centos1 ~]# echo $FC
ifort
[root@centos1 ~]# echo $F90
ifort
[root@centos1 mpich2-1.4.1p1]# ./configure --prefix=/cluster/server/program/mpich2 (出错了)
……………………………………………………………………..
checking for multiple weak symbol support... yes
configure: error: F90 and F90FLAGS are replaced by FC and FCFLAGS respectively in this configure, please unset F90/F90FLAGS and set FC/FCFLAGS instead and rerun configure again.
[root@centos1 mpich2-1.4.1p1]# unset F90
[root@centos1 mpich2-1.4.1p1]# unset F90FLAGS
[root@centos1 mpich2-1.4.1p1]# echo $F90

[root@centos1 mpich2-1.4.1p1]# export F77=ifort
[root@centos1 mpich2-1.4.1p1]# export FC=ifort
[root@centos1 mpich2-1.4.1p1]# ./configure --prefix=/cluster/server/program/mpich2
…………………………………………………………………….
config.status: executing default-2 commands
config.status: executing default-3 commands
Configuration completed.
[root@centos1 mpich2-1.4.1p1]# make
……………………………………..
make[2]: Leaving directory `/usr/local/src/mpich2-1.4.1p1/examples'
Make completed
make[1]: Leaving directory `/usr/local/src/mpich2-1.4.1p1'
[root@centos1 mpich2-1.4.1p1]# make install
………………………………………………………
Installing SLOG2SDK's share
Creating SLOG2SDK's bin
Installed SLOG2SDK in /cluster/server/program/mpich2
/cluster/server/program/mpich2/sbin/mpeuninstall may be used to remove the installation
Installed MPE2 in /cluster/server/program/mpich2
/cluster/server/program/mpich2/sbin/mpeuninstall may be used to remove the installation
make[1]: Leaving directory `/usr/local/src/mpich2-1.4.1p1/src/mpe2'
```

```
[root@centos1 mpich2-1.4.1p1]# cd /cluster/server/program/mpich2
[root@centos1 mpich2]# ls -al
drwxr-xr-x 2 root root 4096 Aug  1 15:01 bin
drwxr-xr-x 2 root root 4096 Aug  1 15:01 etc
drwxr-xr-x 3 root root 4096 Aug  1 15:01 include
drwxr-xr-x 3 root root 4096 Aug  1 15:01 lib
drwxr-xr-x 2 root root 4096 Aug  1 15:01 sbin
drwxr-xr-x 7 root root 4096 Aug  1 15:01 share
[root@centos1 mpich2]# which mpicc (未加入 PATH 时，找不到命令 mpicc)
/usr/bin/which: no mpicc in
(/opt/intel/composer_xe_2011_sp1.11.339/bin/ia32:/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32:/root/bin)
[root@centos1 mpich2]# vi /etc/profile
PATH=$PATH:/cluster/server/program/mpich2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/cluster/server/program/mpich2/lib
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
# By default, we want umask to get set. This sets it for login shell
[root@centos1 mpich2]# source /etc/profile
[root@centos1 mpich2]# which mpicc
/cluster/server/program/mpich2/bin/mpicc
[root@centos1 mpich2]# which mpiexec
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpiexec (这个地方指向出错了，后来才知道的哟)
[root@centos1 mpich2]# vi machinefile
centos1:2
centos2:2
[root@centos1 mpich2]# ls -al
drwxr-xr-x 2 root root 4096 Aug  1 15:01 bin
drwxr-xr-x 2 root root 4096 Aug  1 15:01 etc
drwxr-xr-x 3 root root 4096 Aug  1 15:01 include
drwxr-xr-x 3 root root 4096 Aug  1 15:01 lib
-rw-r--r-- 1 root root   20 Aug  1 15:18 machinefile
drwxr-xr-x 2 root root 4096 Aug  1 15:01 sbin
drwxr-xr-x 7 root root 4096 Aug  1 15:01 share
[root@centos1 mpich2]# vi /etc/profile
PATH=$PATH:/cluster/server/program/mpich2/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/cluster/server/program/mpich2/lib
export HYDRA_HOST_FILE=/cluster/server/program/mpich2/machinefile
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
# By default, we want umask to get set. This sets it for login shell
[root@centos1 mpich2]# source /etc/profile
[root@centos1 mpich2]# chmod g+x machinefile
[root@centos1 mpich2]# chmod o+x machinefile
[root@centos1 examples]# mpirun -np 6 ./cpi (发现出错了)
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpirun: line 86:
```

```
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpivars.sh: No such file or directory
WARNING: Unable to read mpd.hosts or list of hosts isn't provided. MPI job will be run on the current machine only.
Process 0 of 6 is on centos1
Process 3 of 6 is on centos1
Process 4 of 6 is on centos1
Process 1 of 6 is on centos1
Process 2 of 6 is on centos1
Process 5 of 6 is on centos1
pi is approximately 3.1415926544231239, Error is 0.0000000008333307
wall clock time = 0.001078
[root@centos1 examples]# cd /cluster/server/program/mpich2/
[root@centos1 mpich2]# ls -al
drwxr-xr-x 2 root root 4096 Aug  1 15:01 bin
drwxr-xr-x 2 root root 4096 Aug  1 15:01 etc
drwxr-xr-x 3 root root 4096 Aug  1 15:01 include
drwxr-xr-x 3 root root 4096 Aug  1 15:01 lib
-rw-r-xr-x 1 root root   20 Aug  1 15:18 machinefile
drwxr-xr-x 2 root root 4096 Aug  1 15:01 sbin
drwxr-xr-x 7 root root 4096 Aug  1 15:01 share
[root@centos1 mpich2]# chmod u+x machinefile
[root@centos1 examples]# source /etc/profile
[root@centos1 examples]# mpirun -f machinefile -np 6 ./cpi
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpirun:          line          86:
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpivars.sh: No such file or directory
WARNING: Unable to read machinefile or list of hosts isn't provided. MPI job will be run on the current machine only.
Process 0 of 6 is on centos1
Process 2 of 6 is on centos1
Process 5 of 6 is on centos1
Process 3 of 6 is on centos1
Process 4 of 6 is on centos1
Process 1 of 6 is on centos1
pi is approximately 3.1415926544231239, Error is 0.0000000008333307
wall clock time = 0.000947
[root@centos1 examples]# mpirun -np 6 ./cpi
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpirun:          line          86:
/opt/intel/composer_xe_2011_sp1.11.339/mpirt/bin/ia32/mpivars.sh: No such file or directory
WARNING: Unable to read mpd.hosts or list of hosts isn't provided. MPI job will be run on the current machine only.
Process 5 of 6 is on centos1
Process 4 of 6 is on centos1
Process 3 of 6 is on centos1
Process 2 of 6 is on centos1
Process 0 of 6 is on centos1
Process 1 of 6 is on centos1
pi is approximately 3.1415926544231239, Error is 0.0000000008333307
```

```
wall clock time = 0.000304
```
[root@centos1 examples]# vi /cluster/server/program/mpich2/machinefile

```
59.77.36.207:2
59.77.36.208:2
```

这么修改后，仍然出错了.

后来百度了一下，发现以下的内容：

http://zhidao.baidu.com/question/296691011.html

ivf 不含 mpi 实现

cluster 产品才包含 intel 的 mpi 实现，不等于 mpich2，它是基于 mpich2 和 mvapich 的符合 mpi-2 的实现

mpi 实现有 mpich2、mvapich、intel-mpi、lam、chimp、mpi-pro 等

就是说，intel 有自己的 mpi 实现，即 intel-mpi.

重新安装同个版本的 ifort，发现无法安装上去了:

```
Each component will be installed individually. If you cancel the installation,
components that have been completely installed will remain on your system. This
installation may take several minutes, depending on your system and the options
you selected.
--------------------------------------------------------------------------------
Installing Intel Fortran Compiler XE 12.1 Update 3 on IA-32 component... failed
--------------------------------------------------------------------------------
Installing Intel Debugger 12.1 Update 3 on IA-32 component... failed
--------------------------------------------------------------------------------
Installing Intel Math Kernel Library 10.3 Update 9 on IA-32 component... failed
________________________________________________________________________________

To create your support account, please visit the Subscription Services web site
https://registrationcenter.intel.com/RegCenter/registerexpress.aspx?clientsn=NR2
M-RKVWHF75

To get started using Intel(R) Composer XE 2011 Update 9 located in
/cluster/server/program/ifort/composer_xe_2011_sp1.9.293:
- Set the environment variables for a terminal window using one of the following
  (replace "intel64" with "ia32" if you are using a 32-bit platform).
     For csh/tcsh:
        $ source install-dir/bin/compilervars.csh intel64
     For bash:
        $ source install-dir/bin/compilervars.sh intel64
     To invoke the installed compilers:
        For C++: icpc
        For C: icc
        For Fortran: ifort
  To get help, append the -help option or precede with the man command.
- To view a table of getting started documents:
  install-dir/Documentation/en_US/documentation_f.htm.
```

Movies and additional training are available on our website at
www.intel.com/software/products.
以上是 2012.08.01 下午和晚上的做的，以上步骤没有成功
……
……
……
……
……
以下是 2012.08.02 上午做的，以下是正确安装的步骤，成功了
下载了另外一个版本的 ifort ,即 l_fcompxe_ia32_2011.9.293.tgz ,然后执行以下命令

[root@centos1 ~]# yum install libstdc++.so.5
然后安装 l_fcompxe_ia32_2011.9.293.tgz ：(发现只能选择它的默认安装目录)

```
……………………………………..
To get started using Intel(R) Composer XE 2011 Update 9 located in
/opt/intel/composer_xe_2011_sp1.9.293:
- Set the environment variables for a terminal window using one of the following
  (replace "intel64" with "ia32" if you are using a 32-bit platform).
      For csh/tcsh:
         $ source install-dir/bin/compilervars.csh intel64
      For bash:
         $ source install-dir/bin/compilervars.sh intel64
      To invoke the installed compilers:
         For C++: icpc
         For C: icc
         For Fortran: ifort
  To get help, append the -help option or precede with the man command.
- To view a table of getting started documents:
  install-dir/Documentation/en_US/documentation_f.htm.

Movies and additional training are available on our website at
www.intel.com/software/products.
--------------------------------------------------------------------------------
q. Quit [default]
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [q]:
[root@centos1 mpich2-1.4.1p1]# cd /opt/intel/composer_xe_2011_sp1.9.293/bin
[root@centos1 bin]# ls -al
-rwxr-xr-x   1 root root   3719 Aug   2 10:06 compilervars_arch.csh
-rwxr-xr-x   1 root root   4156 Aug   2 10:06 compilervars_arch.sh
-rwxr-xr-x   1 root root   1002 Aug   2 10:06 compilervars.csh
-rwxr-xr-x   1 root root    996 Aug   2 10:06 compilervars_global.csh
-rwxr-xr-x   1 root root    939 Aug   2 10:06 compilervars_global.sh
```

```
-rwxr-xr-x    1 root root     945 Aug    2 10:06 compilervars.sh
drwxr-xr-x    2 root root    4096 Aug    2 10:06 en_US
drwxr-xr-x    2 root root    4096 Aug    2 10:07 ia32
-rwxr-xr-x    1 root root     380 Aug    2 10:07 idbvars.csh
-rwxr-xr-x    1 root root     361 Aug    2 10:07 idbvars.sh
lrwxrwxrwx    1 root root      18 Aug    2 10:06 ifortvars.csh -> ./compilervars.csh
lrwxrwxrwx    1 root root      17 Aug    2 10:06 ifortvars.sh -> ./compilervars.sh
drwxr-xr-x    2 root root    4096 Aug    2 10:06 ja_JP
-rwxr-xr-x    1 root root 58976 Feb 13 10:38 loopprofileviewer.jar
drwxr-xr-x    6 root root    4096 Aug    2 10:06 sourcechecker
-rwxr-xr-x    1 root root 72169 Feb 14 17:44 uninstall.sh
```

[root@centos1 bin]# source ifortvars.sh (出错了，它说还得加上 ia32 或者 intel64)

ERROR: Unknown switch ''. Accepted values: ia32, intel64

[root@centos1 bin]# source ifortvars.sh ia32 (到安装目录里，让 ifortvars.sh 生效)

然后，在/etc/profile 中加入

source /opt/intel/composer_xe_2011_sp1.9.293/bin/compilervars.sh ia32

然后让/etc/profile 生效

[root@centos1 bin]# which ifort

/opt/intel/composer_xe_2011_sp1.9.293/bin/ia32/ifort

以上是在 centos1 即主节点上的，同理，在子节点 centos2 上，再次安装 ifort (仍然是用默认安装目录)，并且也要让它生效，也要把相关路径放入/etc/profile 中.

然后，回到主节点 centos1 上安装 mpich2,用的包是 mpich2-1.4.1p1.tar.gz ：

把它放在/usr/local/src 中解压，解压后

[root@centos1 src]# ls -al

```
drwxr-xr-x      4 root root            4096    Feb      14      19:05      l_fcompxe_ia32_2011.9.293
-rw-r--r--      1 root root   364628807      Aug      1       20:06      l_fcompxe_ia32_2011.9.293.tgz
drwxr-xr-x     15 cv      games            4096 Aug     2       10:55      mpich2-1.4.1p1
-rw-r--r--      1 root root      19502854      Aug      1       18:45      mpich2-1.4.1p1.tar.gz
```

[root@centos1 mpich2-1.4.1p1]# export F77=ifort

[root@centos1 mpich2-1.4.1p1]# export FC=ifort

[root@centos1 mpich2-1.4.1p1]# ./configure --prefix=/cluster/server/program/mpich2 (这是一个共享目录)

然后是，make , make install .

然后，在/etc/profile 中加入

```
……………………………….
PATH=$PATH:/cluster/server/program/mpich2/bin
export HYDRA_HOST_FILE=/cluster/server/program/mpich2/machinefile
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
……………………………………………………………………………………………..
```

在**/cluster/server/program/mpich2/**下面建立 **machinefile** 文件，内容为

[root@centos1 mpich2]# vi machinefile

```
centos1:2
centos2:2
```

并且设置 machinefile 的权限为:

[root@centos1 mpich2]# ls -al

```
drwxr-xr-x 2 root root     4096 Aug   2 10:57 bin
-rwxr-xr-x 1 root root 820955 Aug   2 11:06 cpi
drwxr-xr-x 2 root root     4096 Aug   2 10:57 etc
drwxr-xr-x 3 root root     4096 Aug   2 10:57 include
drwxr-xr-x 3 root root     4096 Aug   2 10:57 lib
-rwxr-xr-x 1 root root       21 Aug   2 10:37 machinefile
drwxr-xr-x 2 root root     4096 Aug   2 10:57 sbin
drwxr-xr-x 7 root root     4096 Aug   1 20:39 share
```

然后，在/etc/hosts.allow 中加入以下内容

[root@centos1 mpich2]# vi /etc/hosts.allow

```
# hosts.allow     This file contains access rules which are used to
#                      allow or deny connections to network services that
#                      either use the tcp_wrappers library or that have been
#                      started through a tcp_wrappers-enabled xinetd.
#                      See 'man 5 hosts_options' and 'man 5 hosts_access'
#                      for information on rule syntax.
#                      See 'man tcpd' for information on tcp_wrappers
centos1
centos2
```

以上做完成后，我们到子节点 centos2 上去做，在/etc/profile 中加入以下内容

…………………………………………….

PATH=$PATH:/cluster/server/program/mpich2/bin

export HYDRA_HOST_FILE=/cluster/server/program/mpich2/machinefile (这个很重要)

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

# By default, we want umask to get set. This sets it for login shell

……………………………………………………….

source /opt/intel/composer_xe_2011_sp1.9.293/bin/compilervars.sh ia32

在/etc/hosts.allow 中加入

[root@centos2 ~]# cat /etc/hosts.allow

```
# hosts.allow     This file contains access rules which are used to
#                      allow or deny connections to network services that
#                      either use the tcp_wrappers library or that have been
#                      started through a tcp_wrappers-enabled xinetd.
#
#                      See 'man 5 hosts_options' and 'man 5 hosts_access'
#                      for information on rule syntax.
#                      See 'man tcpd' for information on tcp_wrappers
centos1
centos2
```

然后，我们回到主节点 centos1 上面，测试

[root@centos1 ~]# which mpicc

/cluster/server/program/mpich2/bin/mpicc

[root@centos1 ~]# which mpiexec

/cluster/server/program/mpich2/bin/mpiexec (这里，我们的 mpiexec 指向了正确的位置)

注意，这里的指向和昨天那种出错了的安装不同，在错误安装中，它的 mpiexec 指向的是 ifort 安装目录下的 mpiexec .后来才了解到，intel 有自己的 intel-mpi ,所以，ifort 安装目录下面的 mpiexec 等，是为 intel-mpi 提供的连接，而不是给 mpich2 提供的连接.我们要用的 mpiexec 是 mpich2 安装目录里的.因此，昨天那个版本的 ifort 有问题，所以我们就换了一个比较老的版本，使得安装 mpich2 的时候，指向的 mpiexec 是 mpich2 的.

[root@centos1 ~]# cd /usr/local/src/mpich2-1.4.1p1/examples (这个目录是非共享的)

[root@centos1 examples]# mpirun -np 6 ./cpi (可执行程序 cpi 是 mpich2 安装包里自带的)

出现以下错误:

```
[proxy:0:1@centos2] launch_procs (./pm/pmiserv/pmip_cb.c:687): unable to change wdir to /usr/local/src/mpich2-1.4.1p1/examples (No such file or directory)
[proxy:0:1@centos2] HYD_pmcd_pmip_control_cmd_cb (./pm/pmiserv/pmip_cb.c:935): launch_procs returned error
[proxy:0:1@centos2] HYDT_dmxu_poll_wait_for_event (./tools/demux/demux_poll.c:77): callback returned error status
[proxy:0:1@centos2] main (./pm/pmiserv/pmip.c:226): demux engine error waiting for event
[mpiexec@centos1] control_cb (./pm/pmiserv/pmiserv_cb.c:215): assert (!closed) failed
[mpiexec@centos1] HYDT_dmxu_poll_wait_for_event (./tools/demux/demux_poll.c:77): callback returned error status
[mpiexec@centos1] HYD_pmci_wait_for_completion (./pm/pmiserv/pmiserv_pmci.c:181): error waiting for event
[mpiexec@centos1] main (./ui/mpich/mpiexec.c:405): process manager error waiting for completion
```

这说明，这里的可执行文件 cpi 不是处于主节点与子节点的共享区域，所以，出错了.应该把可执行文件放入共享区域才成.

把 cpi 拷贝进入/cluster/server/program/mpich2/下面后再执行

[root@centos1 examples]# cd /cluster/server/program/mpich2

[root@centos1 mpich2]# ls -al

```
drwxr-xr-x 2 root root     4096 Aug   2 10:57 bin
-rwxr-xr-x 1 root root 820955 Aug   2 11:06 cpi
drwxr-xr-x 2 root root     4096 Aug   2 10:57 etc
drwxr-xr-x 3 root root     4096 Aug   2 10:57 include
drwxr-xr-x 3 root root     4096 Aug   2 10:57 lib
-rwxr-xr-x 1 root root         21 Aug   2 10:37 machinefile
drwxr-xr-x 2 root root     4096 Aug   2 10:57 sbin
drwxr-xr-x 7 root root     4096 Aug   1 20:39 share
```

[root@centos1 mpich2]# mpirun -np 8 ./cpi

```
Process 1 of 8 is on centos1
Process 5 of 8 is on centos1
Process 0 of 8 is on centos1
Process 4 of 8 is on centos1
Process 6 of 8 is on centos2
Process 7 of 8 is on centos2
Process 3 of 8 is on centos2
Process 2 of 8 is on centos2 (在 centos1 和 centos2 上都有进程，这就说明成功了)
pi is approximately 3.1415926544231247, Error is 0.0000000008333316
wall clock time = 0.001053
```

[root@centos1 mpich2]# mpirun -f machinefile -np 9 ./cpi

```
Process 2 of 9 is on centos2
Process 6 of 9 is on centos2
Process 7 of 9 is on centos2
Process 3 of 9 is on centos2
Process 4 of 9 is on centos1
Process 5 of 9 is on centos1
Process 8 of 9 is on centos1
Process 0 of 9 is on centos1
Process 1 of 9 is on centos1
pi is approximately 3.1415926544231256, Error is 0.0000000008333325
wall clock time = 0.001074
```

我们把 cpi 拷贝到/home/chongyi 下面，然后以 chongyi 登录

```
[chongyi@centos1 ~]$ mpirun -np 8 ./cpi
Process 1 of 8 is on centos1
Process 5 of 8 is on centos1
Process 4 of 8 is on centos1
Process 0 of 8 is on centos1
Process 3 of 8 is on centos2
Process 6 of 8 is on centos2
Process 2 of 8 is on centos2
Process 7 of 8 is on centos2
pi is approximately 3.1415926544231247, Error is 0.0000000008333316
wall clock time = 0.000903
```

可见，普通用户也能够执行并行程序了.

```
[root@centos1 ~]# mpich2version
MPICH2 Version:         1.4.1p1
MPICH2 Release date:    Thu Sep   1 13:53:02 CDT 2011
MPICH2 Device:          ch3:nemesis
MPICH2 configure:       --prefix=/cluster/server/program/mpich2
MPICH2 CC:       gcc     -O2
MPICH2 CXX:      c++     -O2
MPICH2 F77:      ifort   -O2
MPICH2 FC:       ifort   -O2
```

看到了吧，我们要让 MPICH2 的 C , C++编译器和 gcc , c++绑定；让 fortran 编译器和 ifort 绑定,这里显示说已经绑定了，所以安装成功了.

```
[root@centos1 ~]# cd /cluster/server/program/mpich2/share
[root@centos1 share]# ls -al
drwxr-xr-x 7 root root 4096 Aug   2 10:57 doc
drwxr-xr-x 2 root root 4096 Aug   2 10:57 examples_collchk
drwxr-xr-x 2 root root 4096 Aug   2 10:57 examples_logging
drwxr-xr-x 2 root root 4096 Aug   2 10:57 logfiles
drwxr-xr-x 5 root root 4096 Aug   1 20:39 man
[root@centos1 share]# vi /etc/man.config (把帮助文件加进去)
………………………………………………………………
MANPATH /cluster/server/program/mpich2/share/man
```

…………………………………………………………………………………..
好了，上面终于完成了 mpich2 的安装.
注：请多阅读 mpich2 的安装指南，同时，注意到，由于 MPD 问题很多，因此，在最新版本中，mpich2 默认使用的是 hydra,也可以选择使用 smpd 和 gforker，但是都不再用 MPD 了.
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………
………………………………………………………………………………………………………

七．安装 Torque+Maui 作业管理

Torque下载地址: http://www.adaptivecomputing.com/resources/downloads/torque/
Maui下载地址: http://www.adaptivecomputing.com/products/open-source/maui/

```
[root@centos1 ~]# cd /usr/local/src
[root@centos1 src]# ls –al (在这个目录下面放置压缩包)
drwxr-xr-x.    4 root root              4096 Aug    2 19:27 .
drwxr-xr-x.   11 root root              4096 Jul   26 06:49 ..
drwxr-xr-x     4 root root              4096 Feb   14 19:05 l_fcompxe_ia32_2011.9.293
-rw-r--r--     1 root root         364628807 Aug    1 20:06 l_fcompxe_ia32_2011.9.293.tgz
-rw-r--r--     1 root root            901179 Aug    2 19:27 maui-3.3.1.tar.gz
-rw-r--r--     1 root root            898673 Aug    2 19:27 maui-3.3.tar.gz
drwxr-xr-x    15 cv games               4096 Aug    2 10:55 mpich2-1.4.1p1
-rw-r--r--     1 root root          19502854 Aug    1 18:45 mpich2-1.4.1p1.tar.gz
-rw-r--r--     1 root root           6141363 Aug    2 19:26 torque-4.0.2.tar.gz
-rw-r--r--     1 root root           6133826 Aug    2 19:27 torque-4.1.0.tar.gz
[root@centos1 src]# tar –zxvf ./ torque-4.0.2.tar.gz (解压)
[root@centos1 torque-4.0.2]# ./configure --prefix=/opt/torque --with-rcp=scp (配置)
…………………………………………………………………………………….
checking for struct stat64.st_mode... yes
checking if largefile compiles (looking at you, OSX)... no
checking for pthread_create in -lpthread... yes
checking for SSL_accept in -lssl... no
configure: error: TORQUE needs lib ssl in order to build (缺少函数库，那就先安装该函数库)
[root@centos1 torque-4.0.2]# yum install openssl-devel (安装函数库)
[root@centos1 torque-4.0.2]# ./configure --prefix=/opt/torque --with-rcp=scp (再次配置，安装目录放在/opt/torque 中)
……………………………………………..
Building components: server=yes mom=yes clients=yes
                     gui=no drmaa=no pam=no
```

```
PBS Machine type        : linux
Remote copy              : /usr/bin/scp -rpB
PBS home                   : /var/spool/torque
Default server          : centos1
Unix Domain sockets :
Linux cpusets             : no
Tcl                                : disabled
Tk                                  : disabled
Ready for 'make'.
```

[root@centos1 torque-4.0.2]# make

[root@centos1 torque-4.0.2]# make install (安装)

[root@centos1 torque-4.0.2]# cd /opt/torque

[root@centos1 torque]# ls –al (到目录/opt/torque 中看看)

```
drwxr-xr-x    2 root root 4096 Aug    2 19:46 bin
drwxr-xr-x    2 root root 4096 Aug    2 19:46 include
drwxr-xr-x    2 root root 4096 Aug    2 19:46 lib
drwxr-xr-x    2 root root 4096 Aug    2 19:46 sbin
drwxr-xr-x    3 root root 4096 Aug    2 19:46 share
```

[root@centos1 torque]# vi /etc/profile (把 torque 的可执行脚本的路径告诉 PATH)

```
PATH=$PATH:/cluster/server/program/mpich2/bin
PATH=$PATH:/opt/torque/bin:/opt/torque/sbin
export HYDRA_HOST_FILE=/cluster/server/program/mpich2/machinefile
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
```

[root@centos1 torque]# source /etc/profile

[root@centos1 torque-4.0.2]# ls –al (再次回到解压目录中)

```
…………………………………………
drwxrwxr-x      6 2000 11112      4096 May 17 06:15 contrib
…………………………………………………..
-rw-rw-r--      1 2000 11112      2150 May 17 03:49 README.trqauthd
-rwxrwxr-x      1 2000 11112      1729 May 17 03:49 torque.setup (发现有这个文件，很重要)
-rw-r--r--      1 root root      14763 Aug    2 19:42 torque.spec
```

[root@centos1 torque-4.0.2]# ./torque.setup root   (设置 root 为 torque 的管理员)

```
pbs_server port is: 15001
trqauthd daemonized - port 15005
trqauthd successfully started
initializing TORQUE (admin: root@centos1)
You have selected to start pbs_server in create mode.
If the server database exists it will be overwritten.
do you wish to continue y/(n)?y
Max open servers: 9
Max open servers: 9
```

[root@centos1 torque-4.0.2]# cd /var/spool/torque/server_priv (到安装目录下面去)

[root@centos1 server_priv]# ls -al

```
…………………………………………….
```

```
drwxr-x---    2 root root 4096 Aug    2 19:46 acl_groups
drwxr-x---    2 root root 4096 Aug    2 19:46 acl_hosts
drwxr-x---    2 root root 4096 Aug    2 19:46 acl_svr
drwxr-x---    2 root root 4096 Aug    2 19:46 acl_users
…………………………………………………..
drwxr-x---    2 root root 4096 Aug    2 19:46 hostlist
drwxr-x---    2 root root 4096 Aug    2 19:46 jobs
-rw-r--r--    1 root root    350 Aug    2 19:46 nodes
drwxr-x---    2 root root 4096 Aug    2 19:55 queues
-rw-------    1 root root    726 Aug    2 19:55 serverdb
-rw-------    1 root root        6 Aug    2 19:55 server.lock
-rw-------    1 root root        0 Aug    2 19:55 tracking
```
[root@centos1 server_priv]# vi nodes (编辑 nodes 文件，没有该文件的话要自己手动添加)
```
## This is the TORQUE server "nodes" file.
##
## To add a node, enter its hostname, optional processor count (np=),
## and optional feature names.
## Example:
##       host01 np=8 featureA featureB
##       host02 np=8 featureA featureB
## for more information, please visit:
## http://www.clusterresources.com/torquedocs/nodeconfig.shtml
centos1 np=8 normal
centos2 np=8 parallel (指定节点，节点的进程数目，节点属性)
```
[root@centos1 ~]# cd /var/spool/torque

[root@centos1 torque]# ls -al
```
……………………………………………………
-rw-r--r--      1 root root        8 Aug    2 19:46 server_name
drwxr-x---    13 root root 4096 Aug    2 19:59 server_priv
drwxrwxrwt      2 root root 4096 Aug    2 19:46 spool
drwxrwxrwt      2 root root 4096 Aug    2 19:46 undelivered    (这两个文件的权限是 1777 才行)
```
[root@centos1 torque]# pbs_server -t create (第一次可以这么开启 pbs_server)
```
You have selected to start pbs_server in create mode.
If the server database exists it will be overwritten.
do you wish to continue y/(n)?y
```
[root@centos1 torque]# qmgr (用该命令创建队列)
```
Max open servers: 9
Qmgr: creat queue normal queue_type=execution
Qmgr: set server default_queue=normal
Qmgr: set queue normal started=true
Qmgr: set queue normal enabled=true
Qmgr: set server scheduling=true
Qmgr: (Ctrl+d 退出)
```
[root@centos1 torque]# qterm -t quick (关闭 pbs_server)

[root@centos1 torque]# pbs_server (开启)

[root@centos1 server_priv]# pbsnodes -a

pbsnodes: Server has no node list MSG=node list is empty - check 'server_priv/nodes' file (有的时候是 nodes 文件没有保存好，则要重新创建 nodes 文件；有的时候是 torque 还没有反应过来，那么可以重启 pbs_server)

[root@centos1 server_priv]# qterm -t quick

[root@centos1 server_priv]# pbs_server

[root@centos1 server_priv]# qstat -q

```
server: centos1
Queue              Memory CPU Time Walltime Node   Run Que Lm    State
---------------- ------ -------- -------- ----   --- --- --    -----
normal               --        --         --        --      0     0 --      E R
                                                            ----- -----
                                                                0       0
```

[root@centos1 server_priv]# qmgr -c 'p s'

```
# Create queues and set their attributes.
# Create and define queue normal
create queue normal
set queue normal queue_type = Execution
set queue normal enabled = True
set queue normal started = True
# Set server attributes.
set server acl_hosts = centos1
set server log_events = 511
set server mail_from = adm
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 300
set server job_stat_rate = 45
set server poll_jobs = True
set server mom_job_sync = True
set server next_job_number = 0
set server moab_array_compatible = True
```

[root@centos1 server_priv]# pbsnodes -a

```
centos1
     state = down    (注:后来才知道，这说明 pbs_mom 没有开启)
     np = 8
     properties = normal
     ntype = cluster
     mom_service_port = 15002
     mom_manager_port = 15003
     gpus = 0
centos2
     state = down
     np = 8
```

```
        properties = parallel
        ntype = cluster
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0
```
[root@centos1 server_priv]# echo "sleep 30"|qsub

```
qsub can not be run as root
```
[root@centos1 server_priv]# su - chongyi

[chongyi@centos1 ~]$ echo "sleep 30"|qsub

```
qsub: submit error (No default queue specified MSG=requested queue not found)
```
[chongyi@centos1 ~]$ echo "sleep 30"|qsub -q normal

```
1.centos1
```
[chongyi@centos1 ~]$ qstat (发现该作业没有运行,以为是子节点没有安装的缘故，所以这个先放一边)

```
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1.centos1                 STDIN            chongyi                0 Q normal
```
[root@centos1 server_priv]# qmgr (创建队列 parallel)

```
Max open servers: 9
Qmgr: creat queue parallel queue_type=execution
Qmgr: set queue parallel started=true
Qmgr: set queue parallel enabled=true
Qmgr: set server scheduling=true
Qmgr: (Ctrl+d 退出)
```
[root@centos1 server_priv]# qterm -t quick

[root@centos1 server_priv]# pbs_server

[root@centos1 server_priv]# qstat -q

```
server: centos1
Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
parallel           --      --       --      --    0   0 --   E R
normal             --      --       --      --    0   1 --   E R
                                               ----- -----
                                                   0     1
```
[root@centos1 server_priv]# qmgr -c 'p s'

```
# Create queues and set their attributes.
# Create and define queue parallel
create queue parallel
set queue parallel queue_type = Execution
set queue parallel enabled = True
set queue parallel started = True
# Create and define queue normal
create queue normal
set queue normal queue_type = Execution
set queue normal enabled = True
set queue normal started = True
# Set server attributes.
set server scheduling = True
set server acl_hosts = centos1
set server log_events = 511
set server mail_from = adm
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 300
set server job_stat_rate = 45
set server poll_jobs = True
set server mom_job_sync = True
set server next_job_number = 2
set server moab_array_compatible = True
```
[root@centos1 ~]# cd /usr/local/src/torque-4.0.2 (到解压目录中)
[root@centos1 torque-4.0.2]# make packages (生成子节点需要的安装包)
```
Building packages from /usr/local/src/torque-4.0.2/tpackages
rm -rf /usr/local/src/torque-4.0.2/tpackages
mkdir /usr/local/src/torque-4.0.2/tpackages
Building ./torque-package-server-linux-i686.sh ...
libtool: install: warning: remember to run `libtool --finish /opt/torque/lib'
Building ./torque-package-mom-linux-i686.sh ...
libtool: install: warning: remember to run `libtool --finish /opt/torque/lib'
Building ./torque-package-clients-linux-i686.sh ...
libtool: install: warning: remember to run `libtool --finish /opt/torque/lib'
Building ./torque-package-devel-linux-i686.sh ...
libtool: install: warning: remember to run `libtool --finish /opt/torque/lib'
Building ./torque-package-doc-linux-i686.sh ...
Done.
The package files are self-extracting packages that can be copied
and executed on your production machines.    Use --help for options.
```
[root@centos1 torque-4.0.2]# scp torque-package-clients-linux-i686.sh centos2:/root (把包传给子节点 centos2)
```
torque-package-clients-linux-i686.sh                                    100%    590KB 589.9KB/s      00:00
```
[root@centos1 torque-4.0.2]# scp torque-package-mom-linux-i686.sh centos2:/root
```
torque-package-mom-linux-i686.sh                                            100%    809KB 808.8KB/s      00:00
```
[root@centos2 ~]# umount /home
[root@centos2 ~]# cd /root/
[root@centos2 ~]# ./torque-package-clients-linux-i686.sh –install (在子节点 centos2 中安装)
```
Installing TORQUE archive...
Done.
```
[root@centos2 ~]# ./torque-package-mom-linux-i686.sh --install
```
Installing TORQUE archive...
Done.
```
[root@centos2 ~]# cd /var/spool/torque

[root@centos2 torque]# ls -al

```
…………………………………….
drwxr-x--x    3 root root 4096 Aug   2 20:55 mom_priv
-rw-r--r--      1 root root     36 Aug   2 20:55 pbs_environment
-rw-r--r--      1 root root       8 Aug   2 20:55 server_name
-rw-r--r--      1 root root       8 Aug   2 20:55 server_name.new
drwxrwxrwt      2 root root 4096 Aug   2 20:55 spool
drwxrwxrwt      2 root root 4096 Aug   2 20:55 undelivered
```

[root@centos2 torque]# vi server_name (指出服务节点的主机名)

```
centos1
```

[root@centos2 torque]# cd /var/spool/torque/mom_priv

[root@centos2 mom_priv]# ls -al

```
……………………..
drwxr-x--x 2 root root 4096 Aug   2 20:55 jobs
```

[root@centos2 mom_priv]# vi config (在该目录下创建 config 文件，添加以下内容)

```
$pbsserver centos1 (指出服务节点的主机名)
$logevent 255
$usecp centos1:/home /home (指出服务节点共享的区域)
```

[root@centos2 mom_priv]# vi /etc/profile (在 centos2 上也要把 torque 的路径告诉 PATH)

```
………………………………
PATH=$PATH:/opt/torque/bin:/opt/torque/sbin
…………………………………………………………
```

[root@centos2 mom_priv]# source /etc/profile

[root@centos2 torque]# pbs_mom   (在子节点上开启 pbs_mom 服务)

[root@centos2 torque]# which pbs_mom

```
/opt/torque/sbin/pbs_mom
```

然后回到服务节点，即主节点 centos1 上，安装(其实也可以不用安装) maui 软件.

[root@centos1 src]# tar -zxvf maui-3.3.1.tar.gz

[root@centos1 src]# cd maui-3.3.1

[root@centos1 maui-3.3.1]# ./configure --prefix=/opt/maui --with-pbs=/opt/torque

```
…………………………………
configure successful.
```

[root@centos1 maui-3.3.1]# make   (出错了)

```
make -C src/moab all
make[1]: Entering directory `/usr/local/src/maui-3.3.1/src/moab'
gcc  -I../../include/  -I/usr/local/maui/include         -I/opt/torque/include  -D__LINUX     -D__MPBS              -g  -O2    -c
MPBSI.c
MPBSI.c:177: error: conflicting types for ‘get_svrport’
/opt/torque/include/pbs_ifl.h:681: note: previous declaration of ‘get_svrport’ was here
MPBSI.c:178: error: conflicting types for ‘openrm’
/opt/torque/include/pbs_ifl.h:682: note: previous declaration of ‘openrm’ was here
make[1]: *** [MPBSI.o] Error 1
make[1]: Leaving directory `/usr/local/src/maui-3.3.1/src/moab'
```

[root@centos1 maui-3.3.1]# cd /opt/maui

[root@centos1 maui]# ls -al

```
total 8
drwxr-xr-x    2 root root 4096 Aug    2 21:28 .
drwxr-xr-x. 5 root root 4096 Aug    2 21:28 ..
```

里面是空的，即 maui 没装上去!!!!!!没有安装上去就算了，那就启动 torque 自带的 pbs_sched 来管理作业了.

[root@centos1 ~]#pbs_server (开启各种服务)

[root@centos1 ~]#pbs_mom

[root@centos1 ~]#pbs_sched

[root@centos1 ~]# ps -e |grep pbs_

```
14273 ?            00:00:00 pbs_server
14302 ?            00:00:00 pbs_mom
14583 ?            00:00:00 pbs_sched
```

[root@centos1 ~]# which pbs_sched

/opt/torque/sbin/pbs_sched(这可以直接写在/etc/rc.d/rc.local 中，让开机自启动)

[root@centos1 ~]# which pbs_server

/opt/torque/sbin/pbs_server(这可以直接写在/etc/rc.d/rc.local 中，让开机自启动)

[root@centos1 ~]# which pbs_mom

/opt/torque/sbin/pbs_mom(这可以直接写在/etc/rc.d/rc.local 中，让开机自启动)

[chongyi@centos1 ~]$ qsub normal_job.pbs (再提交一个作业，出错了)

```
socket_connect error (VERIFY THAT trqauthd IS RUNNING)
Error in connection to trqauthd (-2)-[cannot connect to port 5 in socket_connect_addr - connection refused]
Communication failure.
qsub: cannot connect to server centos1 (errno=15096) Error getting connection to socket
```

上面的错误说明，trqauthd 这个服务没有开启 ，那就开启它吧.

[chongyi@centos1 ~]$ service trqauthd start

trqauthd: unrecognized service (没有把 trqauthd 加入 service 管理)

[chongyi@centos1 ~]$ which trqauthd

/opt/torque/sbin/trqauthd (这可以直接写在/etc/rc.d/rc.local 中，让开机自启动)

[chongyi@centos1 ~]$ trqauthd

```
Must be run as root
This program must be run as root!!!
```

[chongyi@centos1 ~]$ exit

```
logout
```

[root@centos1 ~]# trqauthd (启动 trqauthd)

```
hostname: centos1
pbs_server port is: 15001
trqauthd daemonized - port 15005
```

[chongyi@centos1 ~]$ qsub normal_job.pbs (可以提交作业了)

```
2.centos1
```

[chongyi@centos1 ~]$ qstat (发现还是没有运行，无论怎么怎么检查设置，都没有让它运行作业)

```
Job id                          Name                User                Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1.centos1                       STDIN               chongyi                       0 Q normal
```

```
2.centos1                  job_name          chongyi               0 Q normal
```

以上是 2012.08.03 上午做的，没有成功，后来才知道，其实就差一步而已.

……
……
……
……

以下是2012.08.03下午做的：部分参考了《资源管理软件TORQUE 与作业调度软件Maui 的安装、设置及使用》(作者：李会民(hmli@ustc.edu.cn) ，中国科学技术大学网络信息中心,2008年1月)的步骤.该文档的连接地址为：http://wenku.baidu.com/view/1b743c0203d8ce2f006623f8.html

由于上午安装的时候没有成功，所以以为是不是安装包有问题，下午又重新安装：

```
[root@centos1 src]# tar -zxvf torque-4.0.0.tar.gz
[root@centos1 torque-4.0.0]# ./configure --prefix=/opt/torque
……………………………………
Building components: server=yes mom=yes clients=yes
                     gui=no drmaa=no pam=no
PBS Machine type     : linux
Remote copy          : /usr/bin/scp -rpB
PBS home             : /var/spool/torque
Default server       : centos1
Unix Domain sockets :
Linux cpusets        : no
Tcl                  : disabled
Tk                   : disabled
Ready for 'make'.
[root@centos1 torque-4.0.0]# make
[root@centos1 torque-4.0.0]# make install
[root@centos1 torque-4.0.0]# cp contrib/init.d/trqauthd /etc/init.d/ (这些是为了开机自启动)
[root@centos1 torque-4.0.0]# cp contrib/init.d/pbs_server /etc/init.d/
[root@centos1 torque-4.0.0]# cp contrib/init.d/pbs_sched /etc/init.d/
[root@centos1 torque-4.0.0]# cp contrib/init.d/pbs_mom /etc/init.d/
[root@centos1 torque-4.0.0]# chkconfig --add trqauthd
[root@centos1 torque-4.0.0]# chkconfig --add pbs_server
[root@centos1 torque-4.0.0]# chkconfig --add pbs_sched
[root@centos1 torque-4.0.0]# chkconfig --add pbs_mom
[root@centos1 torque-4.0.0]# chkconfig --level 35 trqauthd on
[root@centos1 torque-4.0.0]# chkconfig --level 35 pbs_server on
[root@centos1 torque-4.0.0]# chkconfig --level 35 pbs_sched on
[root@centos1 torque-4.0.0]# chkconfig --level 35 pbs_mom on
[root@centos1 torque-4.0.0]# chkconfig –list
pbs_mom          0:off   1:off   2:off   3:on    4:on    5:on    6:off
pbs_sched        0:off   1:off   2:off   3:on    4:on    5:on    6:off
pbs_server       0:off   1:off   2:off   3:on    4:on    5:on    6:off
portreserve      0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

```
[root@centos1 torque-4.0.0]# scp contrib/init.d/pbs_mom centos2:/etc/init.d/
[root@centos2 init.d]# chkconfig --add pbs_mom
[root@centos2 init.d]# chkconfig --level 35 pbs_mom on
[root@centos1 torque-4.0.0]# scp torque-package-clients-linux-i686.sh centos2:/root
[root@centos1 torque-4.0.0]# scp torque-package-mom-linux-i686.sh centos2:/root
[root@centos2 ~]# ./torque-package-clients-linux-i686.sh --install
Installing TORQUE archive...
Done.
[root@centos2 ~]# ./torque-package-mom-linux-i686.sh --install
Installing TORQUE archive...
Done.
[root@centos2 ~]#pbs_mom (在子节点上开启 pbs_mom 服务就够了)
[root@centos1 ~]#trqauthd    (这个很重要,很多时候要最先开启这个服务)
[root@centos1 ~]#qterm –t quick
[root@centos1 ~]#pbs_server (注意 pbs_server , pbs_mom , pbs_sched 开启顺序)
[root@centos1 ~]#pbs_mom
[root@centos1 ~]#pbs_sched
[root@centos1 ~]# ps -e|grep pbs_
  5273 ?          00:00:00 pbs_server
12852 ?          00:00:00 pbs_mom
12854 ?          00:00:00 pbs_sched
[chongyi@centos1 ~]$ qsub normal_job.pbs
0.centos1
[chongyi@centos1 ~]$ qstat (发现还是没有运行作业)
Job id                   Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
0.centos1                 job_name         chongyi                0 Q normal
……………………………………………………………………………………………………………………..
```

下面这一步就是最重要的一步:

```
[root@centos1 ~]# qrun 0.centos1 (第一次提交作业的时候，作业的状态为 Q，此时用这个 qrun 命令强制执行)
[chongyi@centos1 ~]$ qstat (开始运行作业了)
Job id                   Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
0.centos1                 job_name         chongyi         00:03:37 R normal
…………………………………………………………………………………
[chongyi@centos1 ~]$ cat normal_job.pbs (提交的串行脚本内容)
#!/bin/bash
#PBS -N job_name
#PBS -o job.log
#PBS -e job.err
#PBS -q normal
cd $PWD
echo running on hosts `hostname`
echo time is `date`
echo directory is $PWD
echo this job runs on the following nodes:
cat $PBS_NODEFILE
echo this job has allocated 1 node
./a.out
```

能够运行作业，说明安装成功了.

………………………………………………………………………………………………………

………………………………………………………………………………………………………

接下来是对 torque 的设置进行改进，改为满足我们需要的那种：

a. 限制提交的作业的个数

命令：

```
qmgr -c "set queue test max_user_run = 7"
```

这样就限定这个队列的每个用户最多只能提交 7 个任务，还可以限制每次最多可以使用多少个 cpu 多少个核。比如：

```
set queue test resources_max.ncpus = 1
set queue test resources_max.nodes = 1
```

```
[root@centos1 jobs]# qmgr
Max open servers: 9
Qmgr: set queue normal max_user_run=2
[root@centos1 ~]# qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
28.centos1                 job_name         chongyi         00:22:20 R normal
29.centos1                 job_name         chongyi         00:01:34 R normal
30.centos1                 job_name         chongyi                0 Q normal
```

但是，别的用户还是可以提交可以运行的:

```
[root@centos1 ~]# qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
28.centos1                 job_name         chongyi         00:30:11 R normal
29.centos1                 job_name         chongyi         00:10:15 R normal
30.centos1                 job_name         chongyi                0 Q normal
31.centos1                 job_name         cv              00:00:41 R normal
```

建立队列,比如创建 normal

```
[root@centos1 ~]#qmgr
qmgr: create queue normal queue_type=execution
qmgr: set queue normal max_running = 4     (限制 normal 这个队列最多可以运行 4 个进程)
qmgr: set queue normal resources_default.neednodes = normal
qmgr: set queue normal resources_default.walltime = 7200:00:00 (限制一个程序能够运行的最长时间数,很重要)
```

```
qmgr: set queue normal enabled = True
qmgr: set queue normal started = True
qmgr: set server scheduling= True
```

看下面，这个队列只能运行 4 个程序了:

```
[root@centos1 ~]# qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
28.centos1                 job_name         chongyi         00:42:04 R normal
29.centos1                 job_name         chongyi         00:21:43 R normal
32.centos1                 job_name         chongyi         00:03:38 R normal
33.centos1                 job_name         chongyi         00:03:31 R normal
34.centos1                 job_name         chongyi                0 Q normal
35.centos1                 job_name         cv                     0 Q normal
```

说明：每个节点的进程数 np 必须要限制，这样使得提交到该节点上的所有程序在运行时 CPU 的占用率都是 100% .

b.队列名称和节点的属性的关系

```
[root@centos1 ~]# cd /var/spool/torque/server_priv
[root@centos1 server_priv]# cat nodes
centos1 np=8 normal
centos2 np=8 parallel (节点名称  进程个数  节点的属性)
[chongyi@centos1 ~]$ vi normal_job.pbs
#!/bin/bash
#PBS -N job_name
#PBS -o job.log
#PBS -e job.err
#PBS -q normal (把作业放在 normal 这个队列中)
cd $PBS_O_WORKDIR
echo running on hosts `hostname`
echo time is `date`
echo directory is $PWD
echo this job runs on the following nodes:
cat $PBS_NODEFILE
echo this job has allocated 1 node
./a.out
…………………………
[chongyi@centos1 ~]$ qsub -l nodes=1:parallel normal_job.pbs
```

上面那行命令的意思为:把 normal_job.pbs 提交给一个节点(nodes=1) ,必须是属性为 parallel 的节点.由于我们在/var/spool/torque/server_priv/下面的 nodes 文件中设置了：

centos2 np=8 parallel (节点名称 进程个数 节点的属性)

所以，这个时候，normal_job.pbs 将在 centos2 这个节点上运行.

注意：如果不是以“.sh”的脚本，那么提交该脚本前，要给它加上 x 的权限.

```
[chongyi@centos1 ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
43.centos1                 job_name         chongyi         00:01:56 R normall
[chongyi@centos1 ~]$ top
……………………………………
    1 root      20   0  2872 1420 1200 S  0.0  0.1   0:00.78 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
…………………………………….
```

可见，在 centos1 上找不到 a.out 程序的进程,因为作业不在这个节点运行.

```
[chongyi@centos1 ~]$ ssh centos2
[chongyi@centos2 ~]$ top
……………………………….
25394 chongyi   20   0  2984  744  540 R 99.2  0.0   0:43.57 a.out
……………………………….
```

可见，在 centos2 中找到了 a.out 进程，因为作业就指定在这个节点上运行的哟.

注意：在之前，创建了 normal 和 parallel 队列，而每个节点属性可以任何字符，所以即使设它们的属性为 normal 或者 parallel，它们也和队列没有关联.
那怎么让队列 normal 和节点属性相关联呢？比如，我有队列 normal,要提交给各种只要含有 normal 属性的节点，那该怎么做？可以这么做：

```
[chongyi@centos1 ~]$ vi parallel_job.pbs
#!/bin/bash
#PBS -N job_name
#PBS -o job.log
#PBS -e job.err
#PBS -q parallel     (指定提交到 parallel 这个队列中)
#PBS -l nodes=1:parallel (指定使用一个节点，这个节点的属性必须是 parallel)
cd $PBS_O_WORKDIR
echo running on hosts `hostname`
echo time is `date`
echo directory is $PWD
echo this job runs on the following nodes:
cat $PBS_NODEFILE
echo this job has allocated 1 node
./a.out
[chongyi@centos1 ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
46.centos1                 job_name         chongyi         00:00:56 R parallel
```

并且，用 top 命令，发现这个作业已经提交给 centos2 这个节点了.哈哈，有趣吧.
上面是只有 centos2 这一个节点的属性是 parallel 的情形，如果是两个呢？这个时候这么设置：

```
[root@centos1 ~]# cd /var/spool/torque/server_priv
```

[root@centos1 server_priv]# vi nodes

```
centos1 np=2 normal
centos2 np=2 normal
```
(这里设置两个节点的属性都是 normal,并且可以使用的进程数 np 都是 2 个)

然后，必须限制可以提交的个数：

[root@centos1 server_priv]#qmgr

```
qmgr ： set queue normal max_running = 4
```
(限制 normal 这个队列最多可以运行 4 个进程，这 4 个进程恰好是属性为 normal 的所有节点的进程数之和)

然后关闭 pbs_server , pbs_mom , pbs_sched 后重新打开它们

然后提交 5 个作业：

[chongyi@centos1 ~]$ qsub normal_job.pbs (连续提交 5 次)

[chongyi@centos1 ~]$ qstat

```
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
2.centos1                  job_name         chongyi         00:03:03 R normal
3.centos1                  job_name         chongyi         00:03:41 R normal
4.centos1                  job_name         chongyi         00:03:41 R normal
5.centos1                  job_name         chongyi         00:03:40 R normal
6.centos1                  job_name         chongyi                0 Q normal
```

[root@centos1 ~]# top

```
……………………………………
 2820 chongyi    20    0   2984   744    540 R 100.0   0.0    11:14.03 a.out
 2789 chongyi    20    0   2984   744    540 R 99.2    0.0    11:22.24 a.out
    1 root       20    0  2872 1412 1200 S    0.0   0.1    0:00.76 init
    2 root       20    0       0      0      0 S    0.0   0.0    0:00.00 kthreadd
……………………………………
```

[root@centos1 ~]# ssh centos2

[root@centos2 ~]# top

```
……………………………………
27876 chongyi    20    0   2984   744    540 R 99.2    0.0    12:01.13 a.out
27903 chongyi    20    0   2984   744    540 R 99.2    0.0    12:00.17 a.out
    1 root       20    0  3084 1568 1312 S    0.0   0.1    0:00.77 init
    2 root       20    0       0      0      0 S    0.0   0.0    0:00.00 kthreadd
……………………………………..
```

从以上的结果看出：由于限制所有用户提交在 normal 队列中的作业个数只能是 4 个，所以第 5 个就在等待啦；又由于限制每个节点只能有两个进程，所以啊，这 4 个作业被平均分配到两个节点上面去了啦.

要做到这样子，还要有脚本的配合才行：

[chongyi@centos1 ~]$ cat normal_job.pbs

```
#!/bin/bash
#PBS -N job_name
#PBS -o job.log
#PBS -e job.err
#PBS -q normal    (指定提交到 normal 这个队列中)
# PBS -l nodes=1:ppn=1:normal    (指定只能使用一个节点的一个进程，并且这个节点的属性必须是 normal)
```

```
cd $PBS_O_WORKDIR
echo $PBS_O_WORKDIR
echo running on hosts `hostname`
echo time is `date`
echo directory is $PWD
echo this job runs on the following nodes:
cat $PBS_NODEFILE
echo this job has allocated 1 node
./a.out
……………
```
通过上面的脚本，我们就把可执行程序 a.out 提交给 normal 队列，且属性为 normal 的节点了啦.
提交了 4 个作业后，节点的状态变为 job-exclusive 了：

```
[root@centos1 ~]# pbsnodes -a
centos1
        state = job-exclusive
        np = 2
        properties = normal
        ntype = cluster
        jobs = 0/2.centos1, 1/3.centos1
        …………………………………
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0
centos2
        state = job-exclusive
        np = 2
        properties = normal
        ntype = cluster
        jobs = 0/4.centos1, 1/5.centos1
        ……………………………….
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0
………………………………………………………………………………………………………………
……………………………………………………………………………………………………………….
```

说明：比较好的利用节点资源的一种方案是，就是队列和节点的属性能够对应起来，比如，当把作业提交到队列 normal 时，分配给该作业的节点恰好是那些属性为 normal 的节点. 这样子，方便管理.

c. 并行程序的提交脚本的格式

以上讨论的是串行程序的脚本和运行方式，下面讨论多节点并行的程序的提交脚本：

```
[root@centos1 ~]# cd /var/spool/torque/server_priv
[root@centos1 server_priv]# vi nodes
```

centos1 np=2 normal parallel
centos2 np=2 normal parallel(任何一个节点都可以有多个属性，这是很有用的)
然后重新启动 pbs_server , pbs_mom , pbs_sched ， 再执行以下命令:
[chongyi@centos1 ~]$ mpif90 a12800n1d0mv.f90 (产生 a.out 可执行文件)
[chongyi@centos1 ~]$ cat parallel.sh (编写提交的脚本)
#!/bin/bash
#PBS -N para_job
#PBS -o job.log
#PBS -e job.err
#PBS -q parallel (指定把作业提交到 parallel 队列中)
#PBS -l nodes=1:ppn=4:parallel(指定用一个节点，并且这个节点的属性是 parallel)
(注：原本 parallel 队列与节点的属性是没有关系的，但是我们通过上面的方法，把队列和属性联系在一起了.)
cd $PBS_O_WORKDIR
echo "start at `date`"
cat $PBS_NODEFILE
NPROCS=`wc -l<$PBS_NODEFILE`
mpirun -machinefile $PBS_NODEFILE -np $NPROCS ./a.out (执行可执行文件 a.out ,该文件是编译 a12800n1d0mv.f90 之后产生的.)
………………………………………………
[chongyi@centos1 ~]$ qsub parallel.sh
[chongyi@centos1 ~]$ qstat

```
Job id                    Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
28.centos1                 para_job         chongyi          00:05:10 R parallel
```

[chongyi@centos1 ~]$ ls -al

```
………………………………………………..
-rw-rw-r--   1 chongyi chongyi      3811 Aug   6 21:20 10001Ek.dat
-rw-rw-r--   1 chongyi chongyi      3811 Aug   6 21:20 10101Ek.dat
-rw-rw-r--   1 chongyi chongyi      3811 Aug   6 21:20 10201Ek.dat
-rw-rw-r--   1 chongyi chongyi      3811 Aug   6 21:20 10301Ek.dat
-rw-rw-r--   1 chongyi chongyi      2923 Aug   6 21:20 15001flux.dat
-rw-rw-r--   1 chongyi chongyi      2923 Aug   6 21:20 15101flux.dat
-rw-rw-r--   1 chongyi chongyi      2923 Aug   6 21:20 15201flux.dat
-rw-rw-r--   1 chongyi chongyi      2923 Aug   6 21:20 15301flux.dat
-rw-rw-r--   1 chongyi chongyi        49 Aug   6 21:20 17001averflux.dat
-rw-rw-r--   1 chongyi chongyi        49 Aug   6 21:20 17101averflux.dat
-rw-rw-r--   1 chongyi chongyi        49 Aug   6 21:20 17201averflux.dat
-rw-rw-r--   1 chongyi chongyi        49 Aug   6 21:20 17301averflux.dat
…………………………………………………
```

可见，计算结果出来了，并且有 4 个进程哟.
……
……

d. 如何让普通用户看到所有用户提交的作业

有一个问题，就是查看作业排队情况：

```
[root@centos1 ~]# qstat (能够看到所有人提交的作业)
Job id                   Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
2.centos1                 job_name         chongyi         03:53:27 R normal
3.centos1                 job_name         chongyi         03:53:07 R normal
4.centos1                 job_name         chongyi         03:52:07 R normal
5.centos1                 job_name         chongyi         03:52:07 R normal
6.centos1                 job_name         chongyi                0 Q normal
7.centos1                 job_name         cv                     0 Q normal
[root@centos1 ~]# su - chongyi
[chongyi@centos1 ~]$ qstat (只能看到自己提交的作业情况)
Job id                   Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
2.centos1                 job_name         chongyi         03:53:27 R normal
3.centos1                 job_name         chongyi         03:53:07 R normal
4.centos1                 job_name         chongyi         03:52:07 R normal
5.centos1                 job_name         chongyi         03:52:07 R normal
6.centos1                 job_name         chongyi                0 Q normal
[root@centos1 ~]# su - cv
[cv@centos1 ~]$ qstat (只能看到自己提交的作业情况)
Job id                   Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
7.centos1                 job_name         cv                     0 Q normal
```

那么，如何让普通用户能够看到所有人提交的作业的运行情况呢？可以用下面的命令：
(参考http://www.clusterresources.com/pipermail/torqueusers/2008-November/008263.html)

```
[root@centos1 ~]# qmgr
Max open servers: 9
Qmgr: set server query_other_jobs=true
```

然后我们看看

```
[root@centos1 ~]# su - cv
[cv@centos1 ~]$ qstat
Job id                   Name             User            Time Use S Queue
------------------------ ---------------- --------------- -------- - -----
8.centos1                 job_name         chongyi         01:33:21 R normal
9.centos1                 job_name         chongyi         01:33:15 R normal
10.centos1                job_name         chongyi         01:32:59 R normal
11.centos1                job_name         chongyi         01:32:58 R normal
12.centos1                job_name         chongyi                0 Q normal
13.centos1                job_name         cv                     0 Q normal
```

看到了吧，普通用户 cv 也能够看到所有人提交的作业了.用 qstat –f 命令可以看到更加详细的信息.

……

……

e. 一些常见的错误，以及处理方法

有的时候会遇到如下的一些错误：

e.1

```
[root@centos1 server_priv]# pbs_server
PBS_Server: LOG_ERROR::svr_recov_xml, No server tag found in the database file???
PBS_Server: LOG_ERROR::recov_svr_attr, Unable to read server database
pbs_server: failed to get server attributes
```

出错了，怎么办，没有关系，到解压目录中，再执行一次以下命令

```
[root@centos1 torque-4.0.0]# ./torque.setup root
initializing TORQUE (admin: root@centos1)
You have selected to start pbs_server in create mode.
If the server database exists it will be overwritten.
do you wish to continue y/(n)?y
Max open servers: 9
Max open servers: 9
[root@centos1 torque-4.0.0]# pbs_server
```

看到了吧，可以了.当然，这个时候主节点上的队列，节点属性等等都要重新设置了啦.

e.2

```
[root@centos1 ~]# qstat
socket_read_num error
Communication failure.
qstat: cannot connect to server centos1 (errno=15096) Error getting connection to socket
```

这个时候，关掉再重启这些服务就好了：

```
[root@centos1 ~]# ps -e |grep pbs_
  6391 ?           00:00:00 pbs_server
  6398 ?           00:00:00 pbs_mom
  6402 ?           00:00:00 pbs_sched
[root@centos1 ~]# kill 6391 6398 6402
[root@centos1 ~]# pbs_server
[root@centos1 ~]# pbs_mom
[root@centos1 ~]# pbs_sched
[root@centos1 ~]# qstat
```

e.3

在/etc/hosts 中必须加入 127.0.0.1 localhost，否则有的时候会造成无法通信.

```
[chongyi@centos1 ~]$ vi /etc/hosts
127.0.0.1 localhost (这个必须有，而且格式要正确)
59.77.36.207 centos1
59.77.36.208 centos2
```

```
[chongyi@centos2 ~]$ vi /etc/hosts
127.0.0.1 localhost
59.77.36.208 centos2
59.77.36.207 centos1
```

f. 用 id 这个命令可以查看用户的 UID,GID

```
[root@centos1 ~]# id root
uid=0(root) gid=0(root) groups=0(root)
[root@centos1 ~]# su - chongyi
[chongyi@centos1 ~]$ id chongyi
uid=500(chongyi) gid=500(chongyi) groups=500(chongyi)
```

g. 怎么使得编译与提交作业的脚本合并在一起呢？这个就得自己重新写一个脚本了啦.

首先，修改一下 nodes 文件，改为:

```
[root@centos1 scripts]# cat /var/spool/torque/server_priv/nodes
centos1 np=8 normal
centos2 np=8 parallel
```

其次，**黄平师兄**写了一个 jobp 脚本，用于提交并行程序,用法为:

jobp program.f90 节点数 每个节点要用的进程

这样，我们就把不需要自己编译就可以直接提交作业了. jobp 脚本的内容为(有些地方我作了修改，以适合我搭建的集群的情况)：

```
[root@centos1 scripts]# cat jobp
#!/bin/bash
# ---by P.Hwang
if [ $# -lt 3 -o $# -ge 5 ];then
      echo -e "for eg., \033[01;4mprogram.f90 or program.f\033[00m"
      echo -e "usage: \033[01;33mjobp program Nodes PPN [JOB_NAME_suffix]\033[00m"
      exit
fi
#cd `pwd`
if [ $# -eq 4 ];then
    sf="-"$4
fi
if [ -f $1.f90 ];then
mpif90 -O3    -msse3 -ipo -assume byterecl $1.f90 -o .$1$sf
retval=$?
elif [ -f $1.F90 ];then
mpif90 -O3    -msse3 -ipo -assume byterecl $1.F90 -o .$1$sf
retval=$?
elif [ -f $1.f ];then
mpif90 -O3    -msse3 -ipo -assume byterecl $1.f -o .$1$sf
retval=$?
elif [ -f $1.F ];then
```

```
mpif90 -O3   -msse3 -ipo -assume byterecl $1.F -o .$1$sf
retval=$?
elif [ ${1##*.} == "f90" -a -f $1 ];then
mpif90 -O3   -msse3 -ipo -assume byterecl $1 -o .$1$sf
retval=$?
elif [ ${1##*.} == "f" -a -f $1 ];then
mpif90 -O3   -msse3 -ipo -assume byterecl $1 -o .$1$sf
retval=$?
else
  echo -e "\033[01;31m[ERROR]\033[00m\nNo file \033[01;4m$1.f90\033[00m or \033[01;4m$1.f\033[00m found. Are
you kidding? :("
exit
fi
echo "=================================================================="
if [ $retval -ne 0 ];then
    echo -e "\033[01;31m\n[COMPILE ERROR]\nPlease check your ugly codes!:)\033[00m"
    exit
else
    echo -e "\033[01;32m\n[COMPILE SUCCESS]\nSubmission in process ...\033[00m"
fi

echo '#!/bin/bash' > ".$1$sf.sh"
echo "#PBS -N $1$sf" >> ".$1$sf.sh"
echo "#PBS -j oe" >> ".$1$sf.sh"
echo '#PBS -q parallel' >> ".$1$sf.sh"
echo "#PBS -l nodes=$2:ppn=$3:parallel" >> ".$1$sf.sh"
echo 'cd $PBS_O_WORKDIR' >> ".$1$sf.sh"
echo 'echo "Start at: "`date`' >> ".$1$sf.sh"
echo 'cat $PBS_NODEFILE' >>".$1$sf.sh"
echo 'NPROCS=`wc -l<$PBS_NODEFILE`' >>".$1$sf.sh"
echo 'mpirun -machinefile $PBS_NODEFILE -np $NPROCS' "./.$1$sf" >> ".$1$sf.sh"
echo 'echo "End at:     "`date`' >> ".$1$sf.sh"
echo -e "\033[01;4m"
qsub ".$1$sf.sh"
#echo -e "\033[00m"
if [ $? -ne 0 ];then
    echo -e "\033[00m\033[01;31m[SUBMISSION ERROR]\nPlease inform the administrator at once!\033[00m"
else
    echo -e "\033[00m\033[01;32m[SUBMISSION SUCCESS]\nCongratulations!\033[00m"
fi
```
……......
....

说明:1)它分为三部分，一部分是说你这个用法有没有符合规定;第二部分是判断你当前提交的目录下面的是 program.f90 , program.f 还是什么,即判断是以哪种形式存在的;第三部分是，把这个编译后的文件名，已经提交作

业所需要的脚本的内容全部导入到一个.$1$sf.sh 的脚本里(这个脚本的最前面有一个“.”号，说明它是隐藏文件，要用 ls –al 命令才能看到它)，其中$1$sf 是你提交的作业的名字,然后提交这个脚本.
2)我们只要把上面的脚本作适当的修改就可以提交串行程序了啦.
3)把脚本 jobp 放在某个目录下，比如/scripts 里面，那么把该目录的路径添加到 PATH 中，就可以在任何目录下使用该脚本了啦.

试一试吧，提交一个并行程序:

```
[chongyi@centos1 ~]$ jobp a12800n1d0mv.f90 1 4
ipo: remark #11001: performing single-file optimizations
ipo: remark #11006: generating object file /tmp/ipo_ifortyy4Wsk.o
==========================================================
[COMPILE SUCCESS]
Submission in process ...
58.centos1
[SUBMISSION SUCCESS]
Congratulations!
[chongyi@centos1 ~]$ ls -al
………………………………………………
-rw-rw-r--    1 chongyi chongyi      15935 Aug    7 19:34 20x-p.dat
-rw-rw-r--    1 chongyi chongyi      15935 Aug    7 19:34 21x-p.dat
-rw-r--r--    1 chongyi chongyi      15935 Aug    7 19:34 22x-p.dat
-rw-r--r--    1 chongyi chongyi      15935 Aug    7 19:34 23x-p.dat
-rwxrwxr-x    1 chongyi chongyi 1454839 Aug    7 20:29 .a12800n1d0mv.f90 (编译后产生的程序)
-rw-r--r--    1 chongyi chongyi        11050 Aug    7 19:34 a12800n1d0mv.f90 (源程序)
-rw-rw-r--    1 chongyi chongyi          275 Aug    7 20:29 .a12800n1d0mv.f90.sh (产生的新的提交作业的脚本)
[chongyi@centos1 ~]$ cat .a12800n1d0mv.f90.sh (这是用了 jobp a12800n1d0mv.f90 1 4  后产生的新脚本的内容)
#!/bin/bash
#PBS -N a12800n1d0mv.f90
#PBS -j oe
#PBS -q parallel
#PBS -l nodes=1:ppn=4:parallel
cd $PBS_O_WORKDIR
echo "Start at: "`date`
cat $PBS_NODEFILE
NPROCS=`wc -l<$PBS_NODEFILE`
mpirun -machinefile $PBS_NODEFILE -np $NPROCS ./.a12800n1d0mv.f90
echo "End at:      "`date`
…………………………………………………………
[root@centos1 scripts]# ssh centos2
Last login: Tue Aug    7 20:37:16 2012 from centos1
[root@centos2 ~]# top
……………………………………………………………………
15400 chongyi      20      0 21060 1736 1500 R 49.0    0.1      2:18.91 .a12800n1d0mv.f
15401 chongyi      20      0 21060 1724 1488 R 49.0    0.1      2:18.96 .a12800n1d0mv.f
```

```
15402 chongyi     20      0 21060 1744 1500 R 49.0    0.1      2:20.26 .a12800n1d0mv.f
15403 chongyi     20      0 21060 1728 1492 R 49.0    0.1      2:18.91 .a12800n1d0mv.f
  5444 root          20     0    2756    984    816 S    2.0    0.1      0:42.75 plymouthd
      1 root          20     0    3084 1628 1368 S    0.0    0.1      0:00.77 init
      2 root          20     0          0       0       0 S    0.0    0.0      0:00.00 kthreadd
…………………………………...
```

看到了吧，4 个进程都是在 centos2 这个子节点上,这正是我们期待的.

………

………

h. 通常用 pbsnodes –a 是能够看到节点空余情况的，那有没有更好的办法呢？有，编写脚本吧，把 pbsnodes –a 显示的内容以更加通俗的形式表现出来，这里介绍 showfree 这个脚本 .

[root@centos1 torque]# cat /scripts/showfree

```
#!/usr/bin/perl
## Usage: shownodes
#
$nodes=$0;
$nodes=~ s/.*\///g;
#print "Type \"$nodes -h\" for help!\n\n";
if ( $ARGV[0] eq "-h" )
{       print "Usage:\n";
        print "         shownodes\n\n";
        exit;
}

$black=30; $red=31; $green=32; $yellow=33; $blue=34; $purple=35; $magenta=36;$white=37; $default=0;
$unknowncolor=$blue;       $unknownbackground=$green;
$downcolor=$yellow;         $downbackground=$red;
$offlinecolor=$blue;       $offlinebackground=$yellow;
$busycolor=$green;          $busybackground=$purple;
$jobcolor=$red;                 $jobbackground=-1;
$freecolor=$green;          $freebackground=-1;

@nodetypes=("batch", "inter", "reserv");$typename{"inter"}    = "interactive";
$typename{"batch"}    = "batch";$typename{"reserv"} = "reserved";
@nodestats=("free", "job", "busy", "offline", "down", "unknown");
$statname{"free"}       = "free";
$statname{"job"}         = "job-exclusive";
$statname{"busy"}       = "busy";
$statname{"offline"} = "offline";
$statname{"down"}       = "down";
$statname{"unknown"} = "unknown";

#foreach $c (@nodestats){
```

```
#               setcolor(${$c . "color"});
#               setbackground(${$c . "background"});
#                 printf "%s", $statname{$c};
#                 setcolor($default);
#                 printf "    ";
#}

#get the path home of ljrs
$pbs_path = $ENV{'PBS_HOME'};
$pbsnodes="/opt/torque/bin/pbsnodes";   (这里要根据你自己搭建的集群的实际情况填写,用命令 which  pbsnodes 找到该路径  )
$nodefile="/var/spool/torque/server_priv/nodes";
#$nodefile="/scripts/nodes";

# get node status through the command ljrsnodes
open(PBS_INPUT, "$pbsnodes -a |");
SWITCH: while (!eof(PBS_INPUT)) {
                $line=<PBS_INPUT>;
                chomp($line);
                if ( substr($line,0,1) ne ' ' ) {
                                $n=$line;
                                next SWITCH;
                }
                $line =~ s/^ +//;
                @fields=split(' = ', $line);

                SWITCH: {
                                if ( $fields[0] eq "state" ){
                                                $stats{"$n"}=$fields[1];
                                                last SWITCH;
                                }
#add by luoyi 20041230    to get how many cpu per node
                                if ( $fields[0] eq "np" ){
                                                $job{"$n"}=$fields[1];
                                                $nodesnum{"$n"}=$fields[1];
                                                last SWITCH;
                                }
#end add
                                if ( $fields[0] eq "ntype" ){
                                                $types{"$n"}=$fields[1];
                                                last SWITCH;
                                }
#add by luoyi 20041230 to get how many cpu free
                                if ( $fields[0] eq "jobs" ){
```

```
                        $job{"$n"} -= split(',', $fields[1]) ;
                        last SWITCH;
                }
#end add
        }
}
printf "\n"; #---------------------------------------------------------------\n";
#add by luoyi 20041230   to get the mainserver's name   (注：luoyi 就是该脚本的作者吧？感谢他(她).)
#$ping_server = "$ljrs_path/bin/ping_server";

#open(SERVERFILE,"$ping_server |");
#while ($line=<SERVERFILE>) {
#          chomp($line);
#          $line =~ s/\t / /g;
#          $line =~ s/ \t/ /g;

#          if ($line =~ m/Main_server/)
#          {
#                  @fileds=split("    ",$line);

#                  $servername = $fileds[2];
#     }
#}

$qmgr_server="/opt/torque/bin/qmgr  -c  \"  list  server\"  ";  (这里也要根据你自己搭建的集群的实际情况填写,用命令
which qmgr 找到该路径  )


open (SERVERFILE,"$qmgr_server |");
while($line=<SERVERFILE>){
                chomp($line);
                if ($line =~ m/^Server/) {
                        @fileds=split(" ",$line);
                        $servername=$fileds[1];
                }
}
#print $servername."\n";
#end add


# add by luoyi 20041230 to get pool name property and nodes name in each pool


open(NODES,"$nodefile") ;
```

```
while($line=<NODES>){
                chomp($line);
                ($nodename,$npnum,$property)=split(" ",$line);

                $multi=0;
                for ($i=0;$i<$#Poolname+1;$i++) {
                        if ($Poolname[$i] eq $property) {
                                $multi=1;
                                last;
                        }
                }
                push @{$mypoolnode{$property}},$nodename;
        if ($multi == 0) {
                        push(@Poolname,$property);
                }
#               push(@poolnode{"$property"},$nodename);
}


# end add

# add by luoyi 20041230 to show pool node status
#printf "\n";
#
#         setcolor($white);
#               printf "------ Node ---Queue --- Free Procs ---\n";
#        setcolor($default);
#        printf "\n";

$noderow="";
for ($m=0;$m<$#Poolname+1;$m++) {

 #       printf "Poolname : $Poolname[$m] | Pool exec_mode : $modetype[$m]\n";

         setcolor($white);
                printf "                $Poolname[$m]\n";
         setcolor($default);
         setcolor($yellow);
                printf "--------------------------------------\n";
         setcolor($default);



         setcolor($yellow);
                printf "--------------------------------------\n";
                setcolor($default);
                        for ($k=0;$k<$#{$mypoolnode{$Poolname[$m]}}+1;$k++){
#                                   shownodestatus($mypoolnode{$Poolname[$m]}[$k]);
                                    $noderaw=$noderaw.$mypoolnode{$Poolname[$m]}[$k].",";
                        }
                        shownodestatus($noderaw);
                        $noderaw="";
               printf "\n";

}
#end add

sub shownodestatus()
{
setbackground($blue);

  #open the node file to get all of the nodes
                $i=0;

               ($line)=@_;

               @fields=split(",",$line);

               for ($j=0;$j<$#fields+1;$j++) {

               $nodename=$fields[$j];
               @tmp=split(',', $prop{"$nodename"});
               foreach $p ( @tmp ){
                        ++${$p}
                 }
               setbackground($blue);
               foreach $s ( split(',', $stats{"$nodename"}) ) {
                        $flag=0;
                        foreach $c (@nodestats)    {
                                    if ( $s eq $statname{$c} ){
                                                setcolor(${$c . "color"});
                                                setbackground(${$c . "background"});
                                                $flag=1;
                                        }
                                    if ( $s eq "offline" || $s eq "down" ) {
                                       $job{"$nodename"}='?';
                                       $nodesnum{"$nodenmae"}='?';
                                    }
                        }
                        if ( ! $flag ){
                                    setcolor($unknowncolor);
                                    setbackground($unknownbackground);
                                    $job{"$nodename"}='?';
                                    $nodesnum{"$nodenmae"}='?';

                             }
                             foreach $p ( @tmp ){
                                    ++${$p . $s};
                              }
                      }

                      $i++;
                       printf "     %-10s", $nodename."                    ".$job{"$nodename"}."\n";
                       #."/".$nodesnum{"$nodename"}."\n";
               #    printf "%-8s", $nodename."-".$job{"$nodename"};

                      if ($i % 16 ==0 ) {
                        printf "\n";
                       }
                       setcolor($default);
               }
setbackground($blue);
#while ( ++$i <= 10 ) {
while ( ++$i <= 8 ) {
               printf "       "
  }

#while ( ++$i <= 10 ) {
while ( ++$i <= 8 ) {
               printf "       "
  }

setcolor($default);
}

#printf "------------------------------------------------------------\n";
foreach $p ( @nodetypes ){
               if ( ${$p} > 0 ) {
                        prt("", ${$p}, "node   ", "nodes ",              "for " . $typename{"$p"} . " jobs:\n");
                        $flag=0;
                        foreach $c ( @nodestats ){
                                    $s=$statname{"$c"};
                                    prt($flag ? ", ":"", ${$p.$s}, "", "", $s);
                                    $flag=1;
                        }
                        printf ".\n";
               }
}
setcolor($default);
#=====================================================================
sub prt() {
               setcolor($magenta);
               printf $_[0];
               setcolor($purple);
               printf "%3d ", $_[1];
               setcolor($magenta);
               printf "%s%s", ( $_[1] <= 1 ) ? $_[2] : $_[3], $_[4];
}
sub setcolor () {
               ($_[0]!=0) ? printf "m", $_[0] : printf "";
}

sub setbackground () {
#       printf("m", $_[0]+10) if ( $_[0]>0 );
}
```

………

………....

试着用一用：

[root@centos1 ~]# showfree (可以看到有每个节点可以用的进程数)

```
               normal
---------------------------------------------------
     centos1                8

               parallel
--------------------------------------------------
     centos2                8
```

[root@centos1 ~]# su - chongyi

[chongyi@centos1 ~] $showfree (空的，什么都没有，怎么回事？)

[chongyi@centos1 ~] $

[chongyi@centos1 ~] $ cd /var/spool/torque/server_priv

```
-bash: cd: /var/spool/torque/server_priv: Permission denied
```

[root@centos1 ~]# cd /var/spool/torque

[root@centos1 torque]# ls -al

```
......................................................................
drwxr-x---    13 root root 4096 Aug    7 16:21 server_priv
```

```
drwxrwxrwt      2 root root 4096 Aug    7 19:30 spool
drwxrwxrwt      2 root root 4096 Aug    3 14:41 undelivered
```

原来 server_priv 这个文件不允许普通用户访问，为此，改变它的权限为：

```
[root@centos1 torque]# chmod o+r server_priv
[root@centos1 torque]# chmod o+x server_priv
[root@centos1 torque]# su - chongyi
[chongyi@centos1 ~] $ showfree

                    normal
-----------------------------------------------
       centos1                  8

                    parallel
-----------------------------------------------
       centos2                  8
```

看到了吧，现在普通用户也能够用 showfree 这个脚本了(另外，记得要把该脚本所在绝对路径放入 PATH 中). 这里再次说明，文件目录的权限是很重要的，在遇到问题的时候，应该要想到这一点.

参考文献

1.《鸟哥的 linux 私房菜------基础学习篇》
2.《鸟哥的 linux 私房菜------服务器架设篇》
3.   http://linux.vbird.org/linux_server/0600cluster.php 简易cluster架设-鸟哥网站上的文章
极力建议大家去看看以上的参考文献，我的 linux 基础知识，服务器架设知识基本上都是从它们那边来的.

通过百度，Google 搜索到的网站上的好资料：
关于 MPICH2 安装的
4. http://hxhg.gxu.edu.cn/lsq/?act=article&code=detail&cid=67&id=52
5. http://www.linuxidc.com/Linux/2011-12/50230.htm
6. http://linux.chinaitlab.com/set/879791_2.html
7. http://emuch.net/html/201205/4020149.html

关于 Torque+Maui 安装的
8. http://wenku.baidu.com/view/1b743c0203d8ce2f006623f8.html《资源管理软件TORQUE 与作业调度软件Maui的安装、设置及使用》(作者：李会民)
9. http://blog.csdn.net/educast/article/details/7168467
10. http://hi.baidu.com/flowaters/item/b5c1e926b8920bc9a4275a6b

关于并行计算编程的
11. 《高性能计算之并行编程技术------MPI 并行程序设计》都志辉 编著 ， 李三立 审阅 ， 陈渝 刘鹏 校对

我是通过上面那本书学习并行编程技术的，还不错哟.

致谢

非常感谢杨静华师兄提供了好多学习资料，感谢陈济舸提供了 PBS 的一些资料，感谢黄平师兄提供了并行作业脚本，感谢崔嵬提供了 linux 安装盘. 感谢陈顺达师兄的鼓励. 对了，要感谢熊大兴师兄在好多年前提供了并行编程资料.感谢师妹方萍，马颖的鼓励.

O(∩_∩)O 哈哈~，搞得跟写书一样，其实只是一个笔记而已，供大家参考哟. 如果你真的要搭建集群，那还得自己学习怎么搭建 NAT 服务，怎么设置防火墙，还有很多要学习，那和我没有关系了啦，O(∩_∩)O 哈哈~对了对了，整理完这个我就不想看第二遍了啦，很烦的说，所以有神马错误的，那就自己想办法纠正咯. 还有，每个人的架设环境不一样，所以按照我的步骤做，或许你会出错，这也不能怪我呀.

钟毅
厦门大学物理学系复杂系统研究组