

高性能集群搭建指南

——北京航空航天大学中德软件所



2017 v2

刘权 著

杨海龙 修订

目录

[高性能集群搭建指南（一） - CentOS安装 0](#_Toc487743395)

[一、 写在前面 0](#_Toc487743396)

[1. 一些建议 0](#_Toc487743397)

[2. 一些文档 0](#_Toc487743398)

[二、准备工作 1](#_Toc487743399)

[1. 制作U盘启动盘 1](#_Toc487743400)

[三、BIOS 介绍 1](#_Toc487743401)

[[划重点] 功耗控制 8](#_Toc487743402)

[四、CentOS 安装 9](#_Toc487743403)

[高性能集群搭建指南（二） - 系统配置 24](#_Toc487743404)

[一、hostname hosts 和 IP 24](#_Toc487743405)

[1. hostname 设定 24](#_Toc487743406)

[2. hosts 24](#_Toc487743407)

[3. IP 25](#_Toc487743408)

[二、SELinux和防火墙 30](#_Toc487743409)

[1. SELinux模式 30](#_Toc487743410)

[2. 关闭SELinux 30](#_Toc487743411)

[3. 查看SELinux工作 状态 31](#_Toc487743412)

[4. 关于 setenforce 31](#_Toc487743413)

[5. 关闭防火墙 31](#_Toc487743414)

[6. 碰到的一些问题 31](#_Toc487743415)

[三、yum 本地源 （可选） 31](#_Toc487743416)

[1. 服务端配置 32](#_Toc487743417)

[2. 客户端 34](#_Toc487743418)

[3.碰到过的一些问题 35](#_Toc487743419)

[三、SSH无密码访问 36](#_Toc487743420)

[1. OpenSSH 安装 37](#_Toc487743421)

[2. 单机的无密码访问配置 37](#_Toc487743422)

[3. 集群的无密码访问配置 37](#_Toc487743423)

[四、InfiniBand 配置 40](#_Toc487743424)

[1. 安装TCL TK 40](#_Toc487743425)

[2. 安装 OFED驱动 41](#_Toc487743426)

[3. 重启 43](#_Toc487743427)

[4. IB读写带宽测试 43](#_Toc487743428)

[五、NFS文件系统 exports 43](#_Toc487743429)

[1. 安装NFS服务 44](#_Toc487743430)

[2. NFS系统守护进程 44](#_Toc487743431)

[3. NFS服务器配置 44](#_Toc487743432)

[高性能集群搭建指南（三） - 软件环境 50](#_Toc487743433)

[一、软件环境 50](#_Toc487743434)

[1. Intel Parallel Stdio 50](#_Toc487743435)

[2. MPICH MVAPICH openMPI 66](#_Toc487743436)

[3. cuda 驱动 67](#_Toc487743437)

[4. RAPL 68](#_Toc487743438)

[5. 通过modules 管理环境变量 69](#_Toc487743439)

[高性能集群搭建指南（四） - 性能测试及功耗控制 71](#_Toc487743440)

[一. 计算性能测试 71](#_Toc487743441)

[1. 理论峰值的计算 71](#_Toc487743442)

[2. HPL(Linpack) 71](#_Toc487743443)

[二、 网络测试 76](#_Toc487743444)

[三、 内存测试 76](#_Toc487743445)

[四、CPU功耗控制 77](#_Toc487743446)

[1. 通过RAPL控制CPU功耗上限。 77](#_Toc487743447)

[2. 通过BIOS设置RAPL 78](#_Toc487743448)

[3. 通过cpu调频控制CPU功耗 78](#_Toc487743449)

[五、 GPU功耗控制 79](#_Toc487743450)

[1. 直接控制GPU功率上限（粗粒度） 79](#_Toc487743451)

[2. 控制GPU频率（细粒度） 79](#_Toc487743452)

# 高性能集群搭建指南（一） - CentOS安装

2017-05-15 22:05:49

## 写在前面

从参加超算竞赛起才接触到linux的世界，走了不少弯路，把一些小白学习时的心得记录下来，希望能对大家会有一些帮助。 因为出发点是详细地说明，所以篇幅会比较长。其实集群的搭建工作中涉及到的东西也比较有限，不过在学习完这些基本以后，大概就会有一种稍微入门的感觉了。

本指南的markdown版本在GitHub私有库上<https://github.com/liuquan2049/cis-buaa>，希望大家后续加以补充和完善。

### 1. 一些建议

* 多探索，勤google，不明白的地方随手看看 man。
* 学习时最好能明白其中的机制，不要觉得能用了就ok了。如果似懂非懂的话，在平时的使用过程中会重新出现很多问题，解决起来费时费力；在比赛中这个问题会更加严重，千里之堤，溃于蚁穴。（我觉得自己这方面没有做好 QAQ ）
* 数据做好备份，系统内重要文件在修改前务必先备份，修改的地方加上注释便于他人查看。

### 2. 一些文档

* man, man -k <command>
* [鸟哥的Linux私房菜:](http://cn.linux.vbird.org/)http://cn.linux.vbird.org/
* [RedHat的相关文档:](https://access.redhat.com/documentation/zh_cn/red-hat-enterprise-linux/?version=7/)https://access.redhat.com/documentation/zh\_cn/red-hat-enterprise-linux/?version=7/
* [Parallel Stido Documentation:](https://software.intel.com/en-us/intel-parallel-studio-xe-support/documentation)https://software.intel.com/en-us/intel-parallel-studio-xe-support/documentation
* [Intel Xeon 处理器:](http://www.intel.cn/content/www/cn/zh/products/processors/xeon.html)http://www.intel.cn/content/www/cn/zh/products/processors/xeon.html

## 二、准备工作

### 1. 制作U盘启动盘

#### 1.1 下载 ISO 文件

推荐[清华的镜像站:](https://mirrors.tuna.tsinghua.edu.cn/)速度较快http://mirrors.tuna.tsinghua.edu.cn/centos/7/isos/x86\_64/CentOS-7-x86\_64-Everything-1611.iso

#### 1.2 dd 命令

在Windows下也可以通过一些工具制作U盘启动盘。如 rufus-2.12 ， UltraISO 。

在linux下可以直接通过 dd 命令来实现，dd 的作用是用指定大小的块拷贝一个文件，和 cp 以字节方式读取的方式相比， dd 则是以扇区方式来读取，拷贝后的数据的排列方式保持不变。

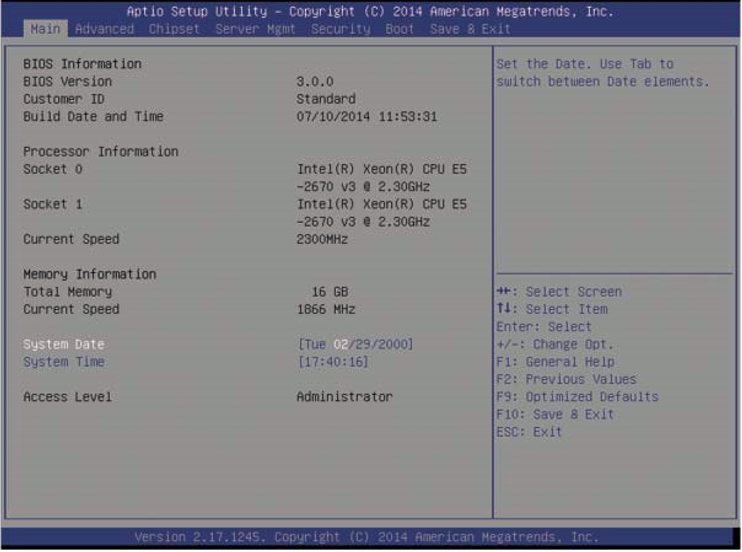
$ dd if=CentOS-7-x86\_64-Everything-1611.iso of=/dev/sdb1 bs=4M

## 三、BIOS 介绍

ASC比赛官方制定服务器一般是浪潮NF系列的机架式服务器，以下简单介绍一下 BIOS 中需要注意和修改的地方。

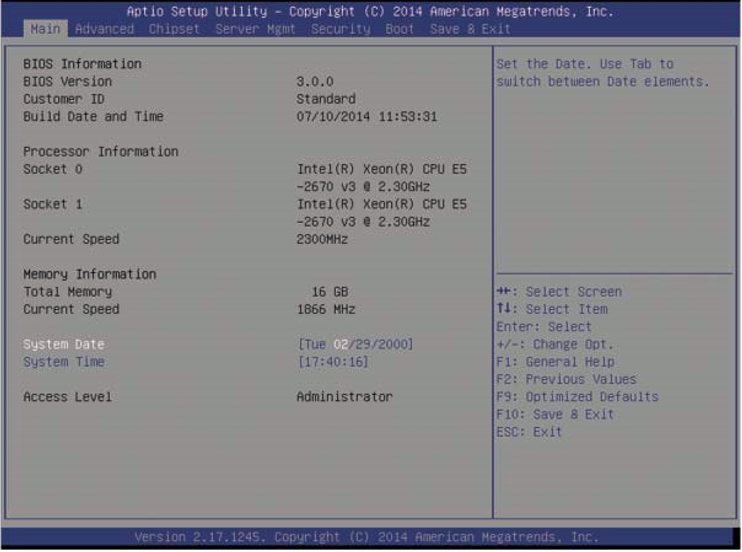
服务器一般会进入两次开机界面，在第二次开机界面时，按照提示，按 F11 可以选择启动方式； Del 可以进入 BIOS 设置菜单。

进入 BIOS 后界面如下：

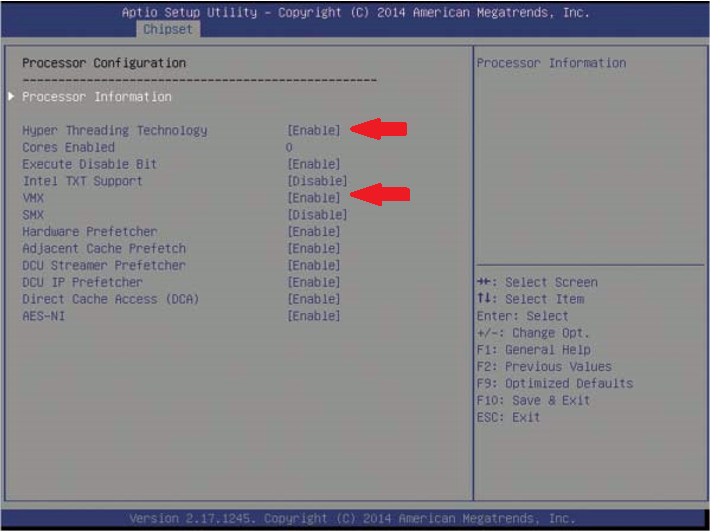


在BIOS中需要改动的地方不多，主要是涉及到服务器性能的一些选项。

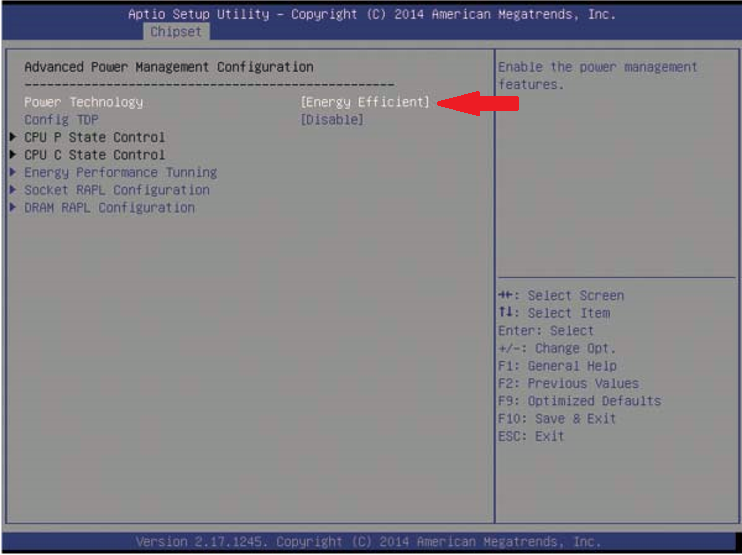
在 chipset 菜单中，选择 Processor Configuration。



将 Hyper Threading Technology 和 VMX 修改为Disable。



在 chipset 菜单中，选择 Processor Configuration 将 Power Technology 置于 Performance 模式下。

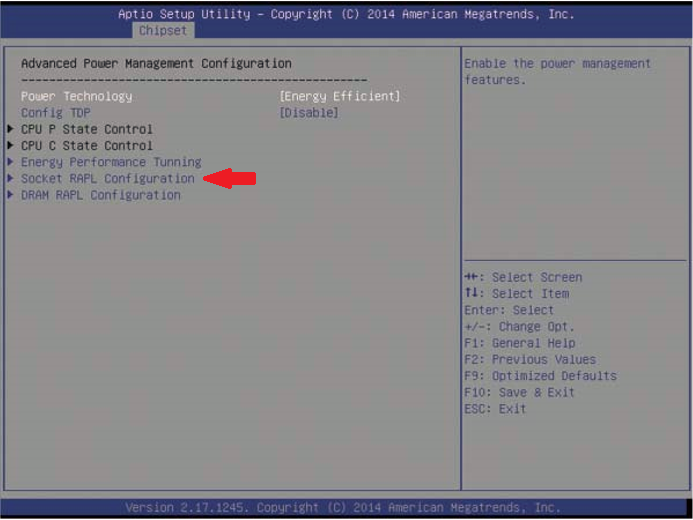


### [划重点] 功耗控制

在 chipset 菜单中，选择 Advanced Power Management Configuration,进入 Socket RAPL Configuration ,同时调整其中短时间和长时间的功率参数。

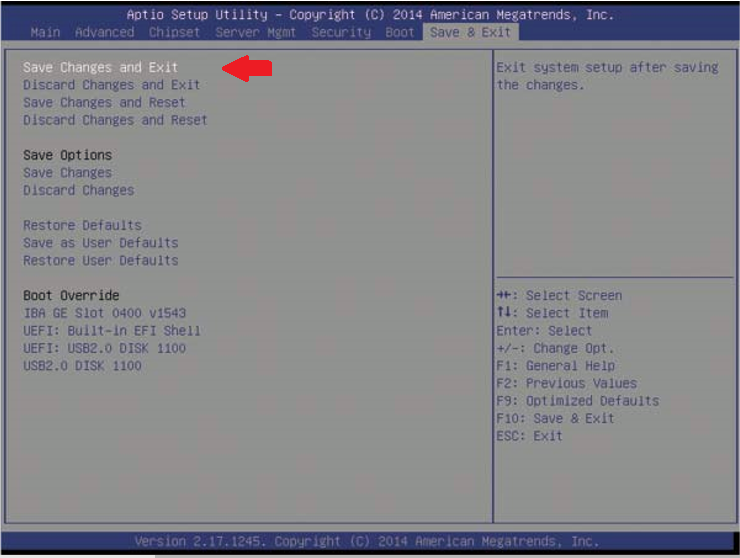
注：这个参数并不是CPU运行的实际功率，而是安装一定比例对应的，在ASC17中，我们调整这个参数到98，实际CPU功率约为180w。

后面会展开说明功耗控制的一些方法。当软件层面的RAPL无法调整功耗时，只能在BIOS中通过此步骤来控制CPU功耗。



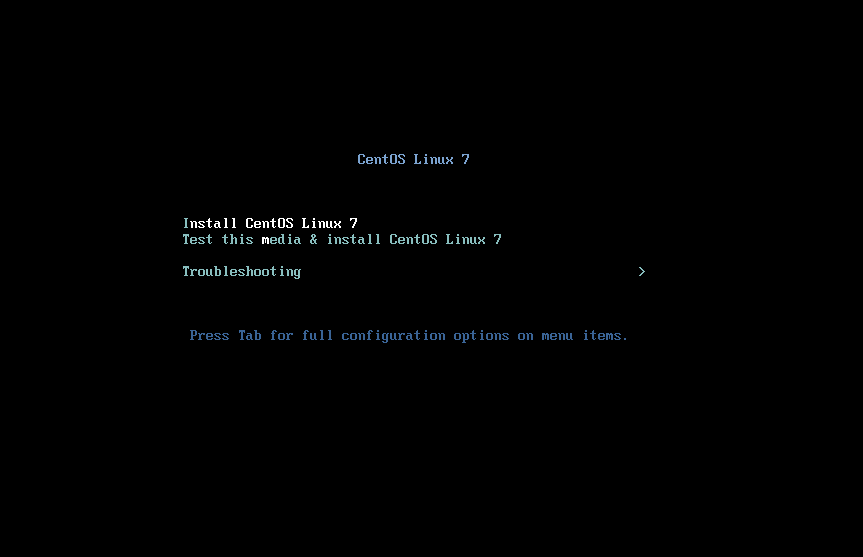
在 boot 菜单中，修改启动顺序，优先启动U盘；也可以在开机界面选择启动方式。

最后保存并退出 Save Changes and Exit 。



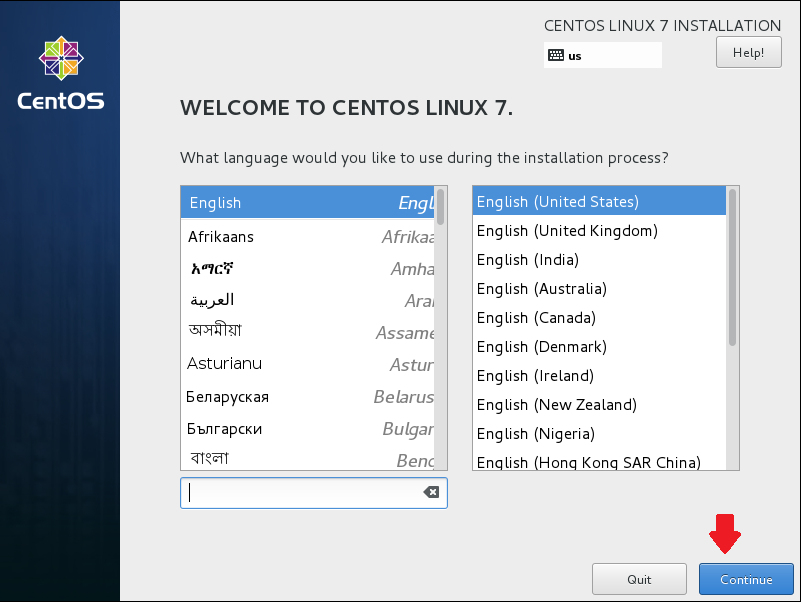
## 四、CentOS 安装

在系统进入U盘启动后，选择 Install CentOS Linux 7。



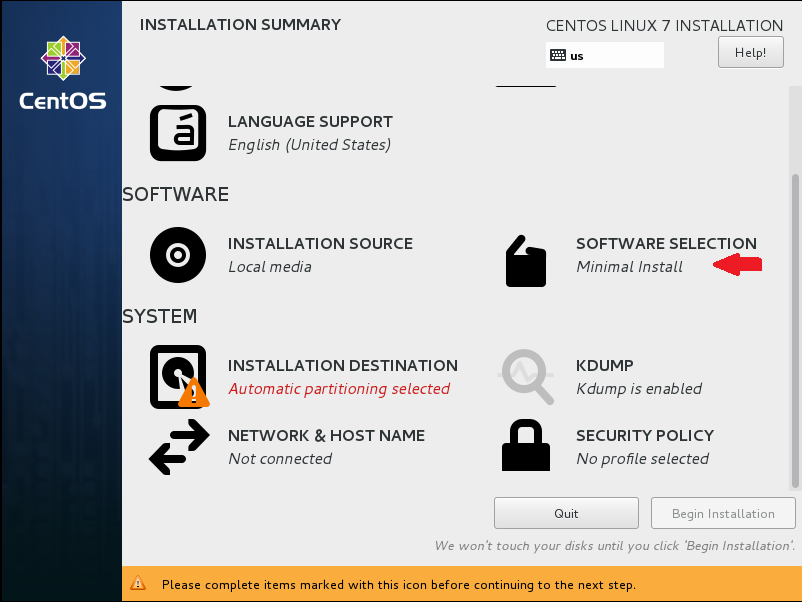
之后进入语言选择界面，选择默认的英文。

语言的话比较推荐英文。由于我英文很渣，初学的时候用装的是中文，后来在写expect脚本的时候，发现装中文确实不合适。



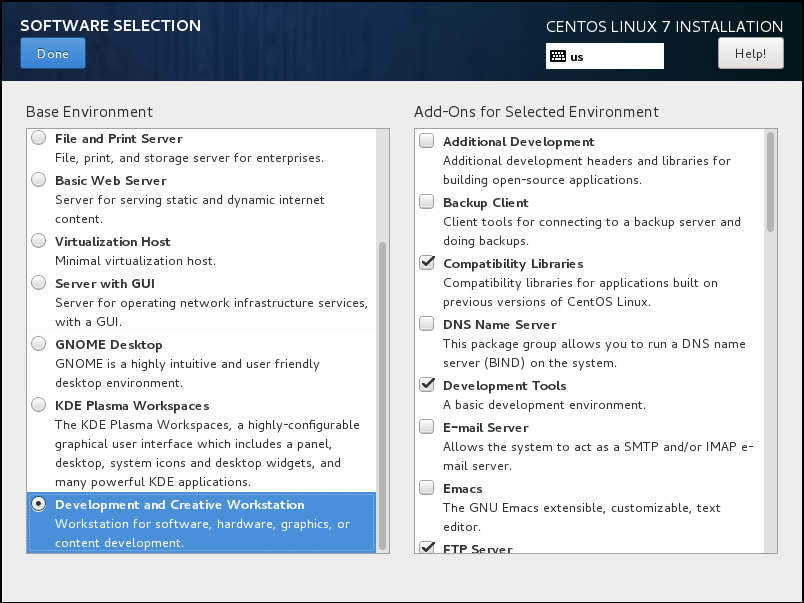
在安装选项中，我们需要设置 SOFTWARE SELECTION， INSTALLATION DESTINATION， KDUMP， NETWORK & HOST NAME；以下一一介绍说明。

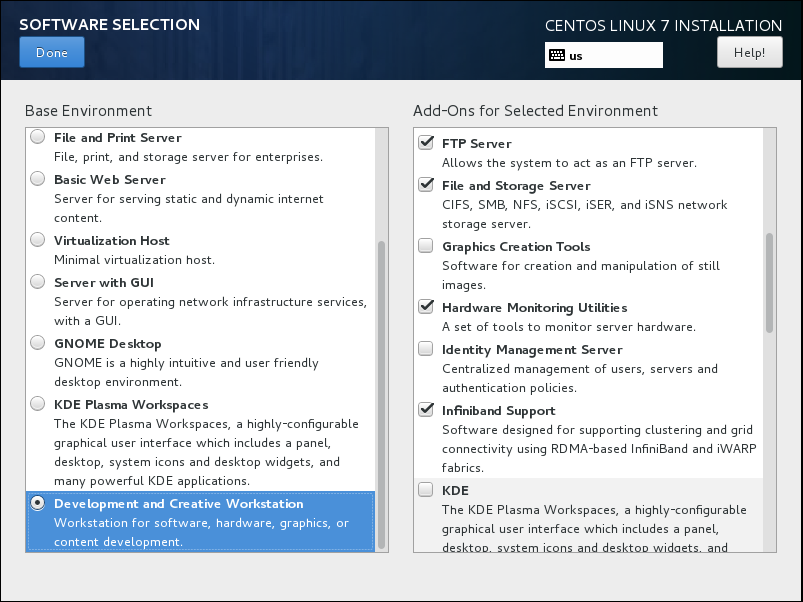
* SOFRWARE SELECTION

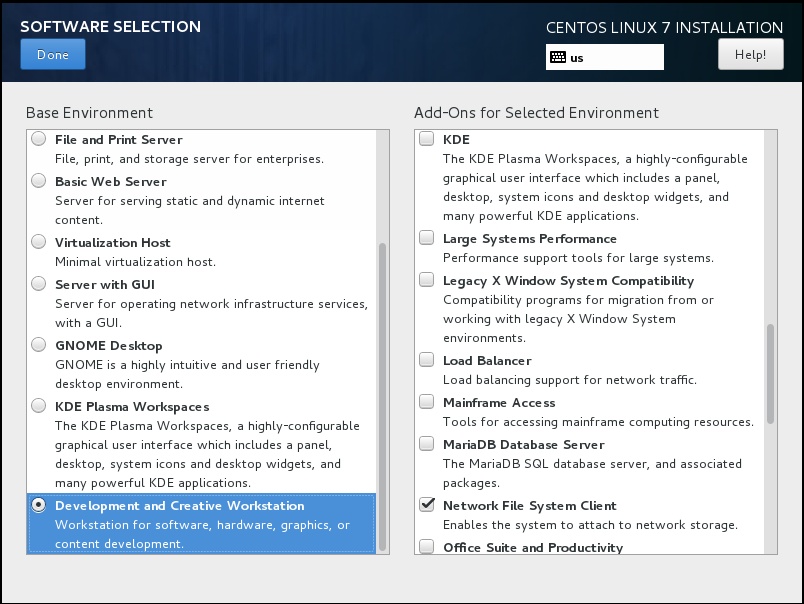


这一部分是安装系统的时候同时会安装的软件，以下是王鹿鸣学长推荐的额外需要的软件清单。

1. Compatibility Libraries
2. Development Tools
3. FTP Server
4. File an Storage Server
5. Hardware Monitoring Utilities
6. Infiniband Support
7. Network File System Client







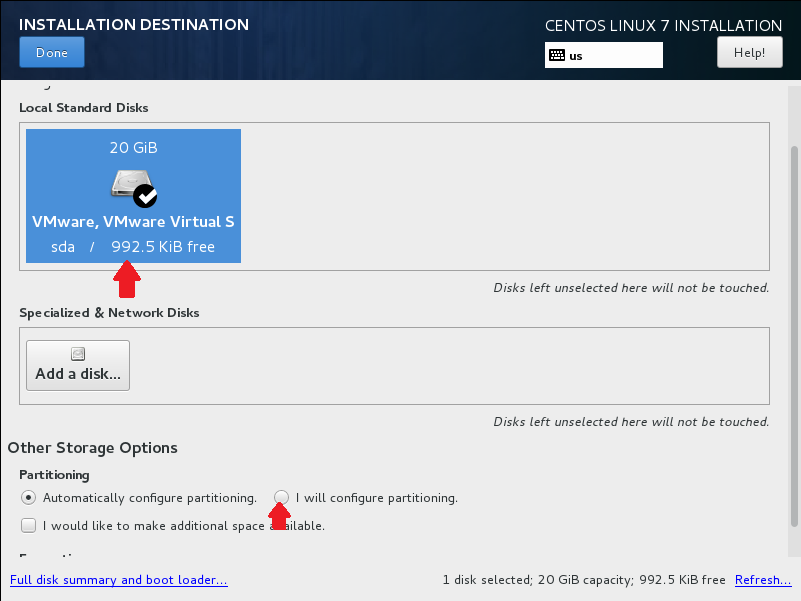
* INSTALLATION DESTINATION



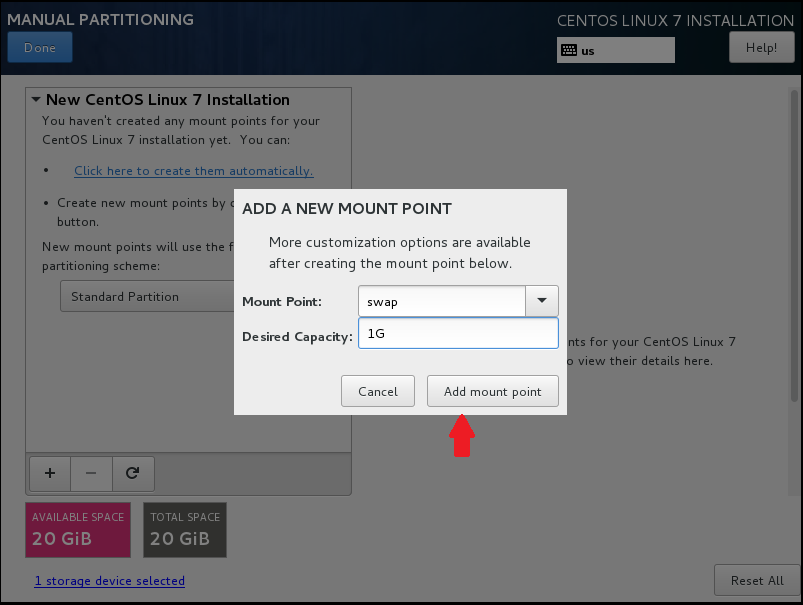
选择系统安装的位置，选择一块磁盘。（注意那个只有 16G的是U盘，不要误选了， 如果配了多块硬盘，那就都选上吧）

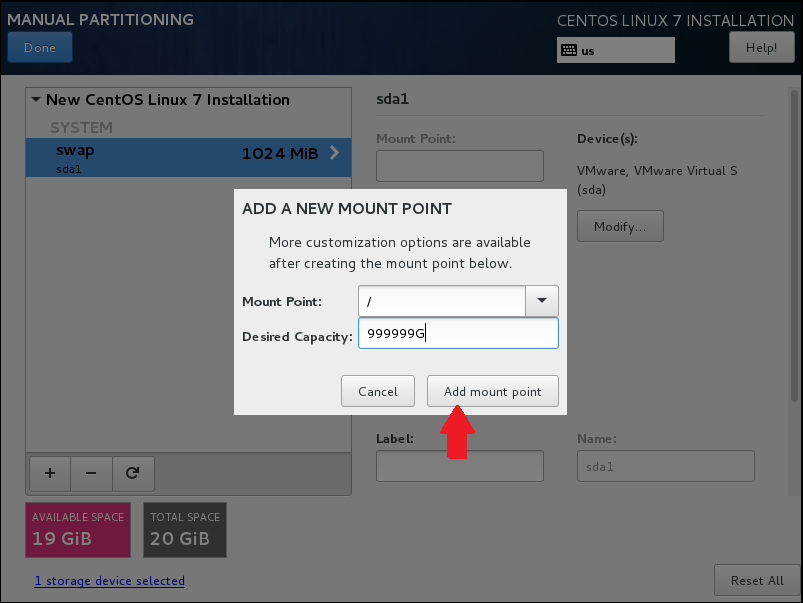
然后选择自定义分区。（如果是让系统自己分区的话，发现分的结果并不理想，它会把 /home 目录独立分区）

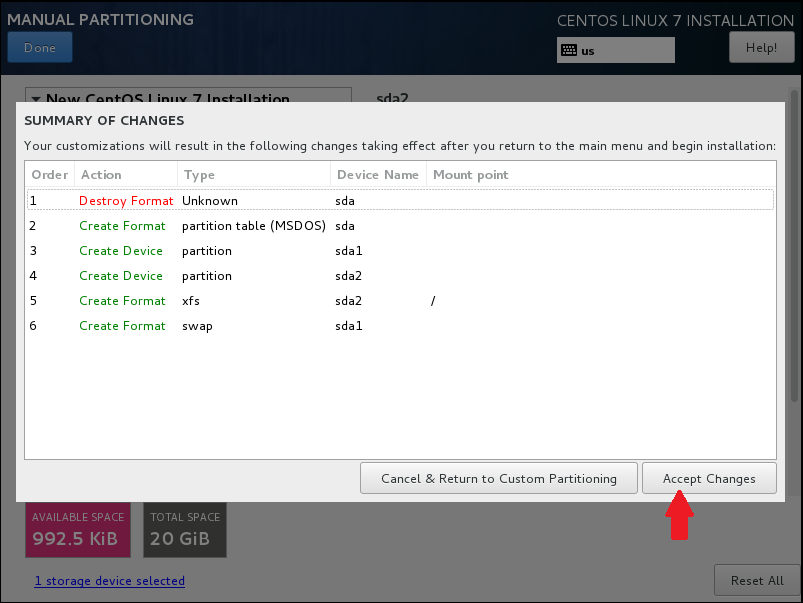
自定义分区时，选择标准分区。 swap 的大小一般和内存大小保持一致；剩下的空间全部分给 / 目录。 如果之前U盘启动选择了 UEFI方式，在这里需要分配200M空间给 /boot。



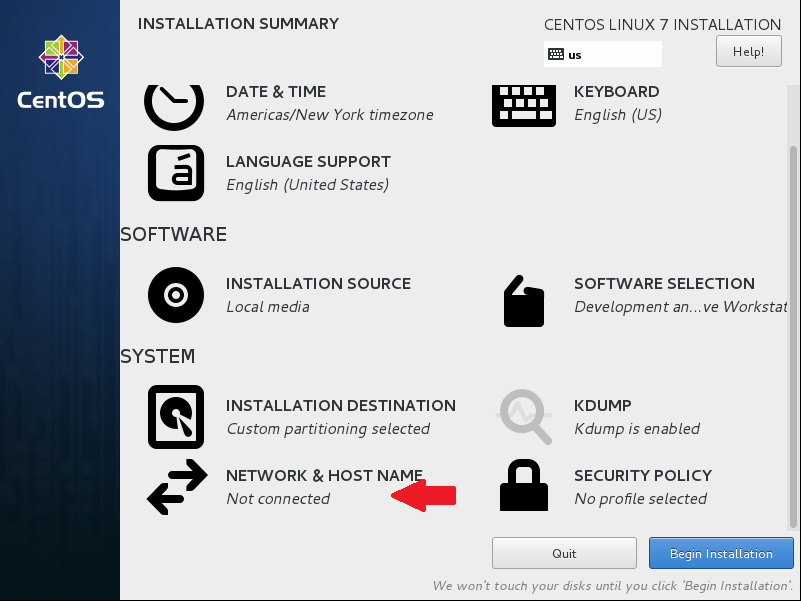






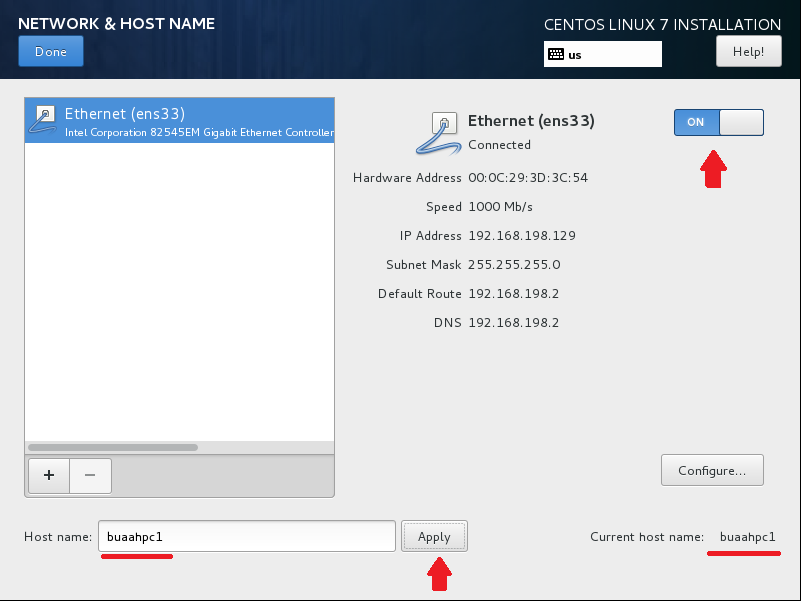


* NETWORK & HOST NAME



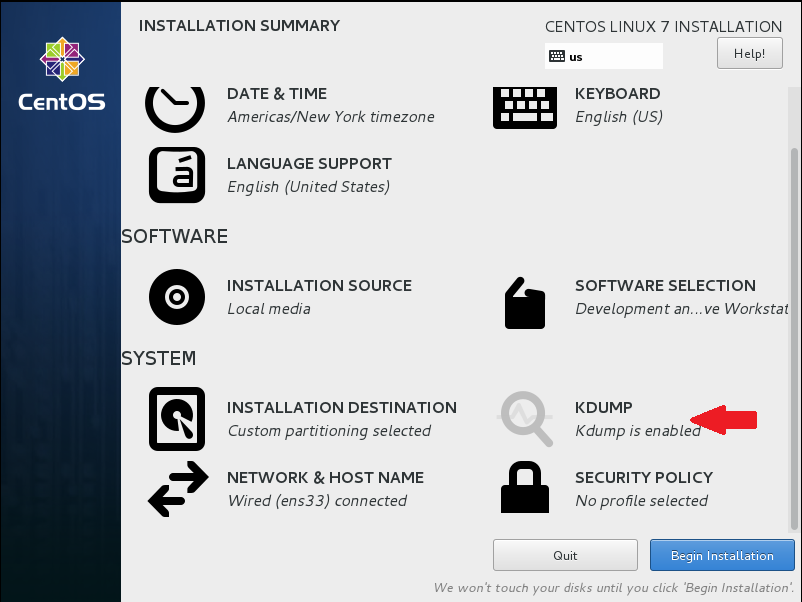
如果已经插上网线的话，我们可以启动该网口,这里它会以 DHCP 的方式自动配置网络。之后在系统配置时，我们还需要进一步配置静态IP。在这里启动该网口后，该网口的配置文件中的 onboot 选项会置于 yes，后面就不需要改动这个选项了。

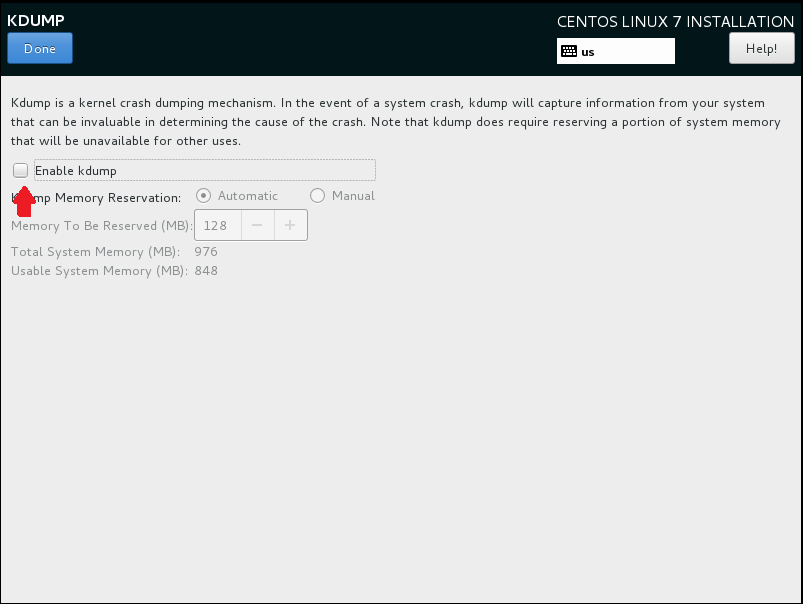
Host name 修改后需要 Apply 才会生效。



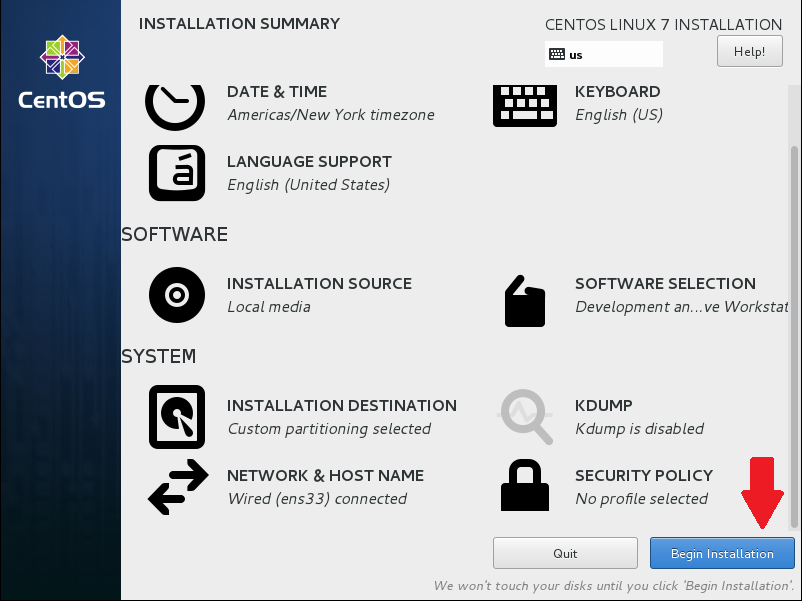
* KDUMP

kdump是在系统崩溃、死锁或者死机的时候用来转储内存运行参数的一个工具和服务，在这里选择关闭。

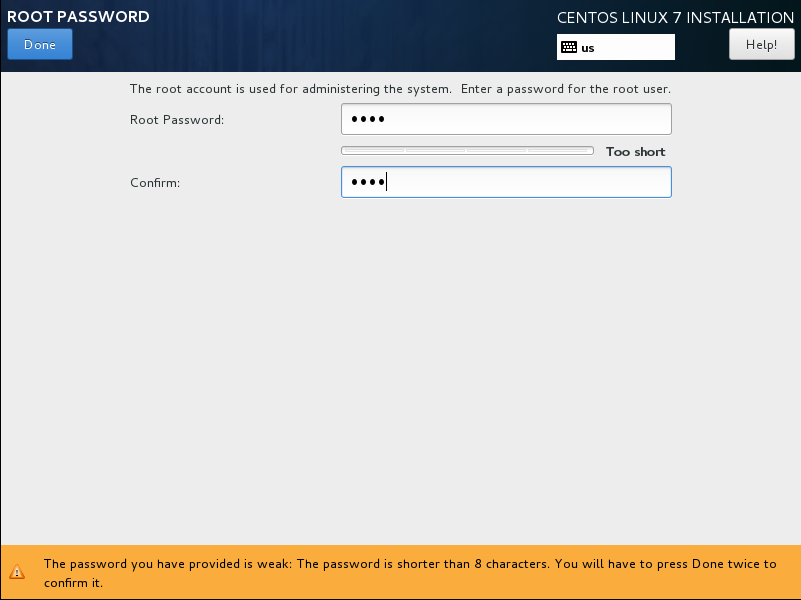


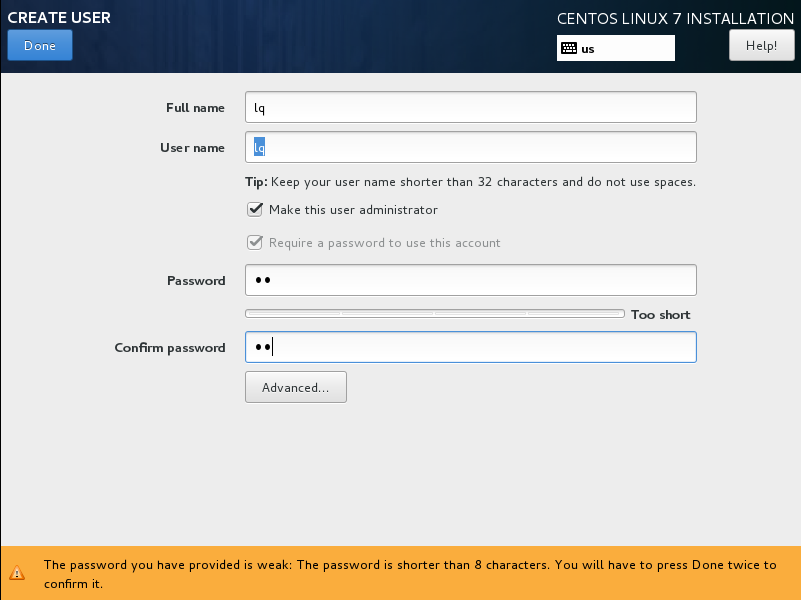


之后选择 Begin Installation。



设置 root 和 管理员用户密码

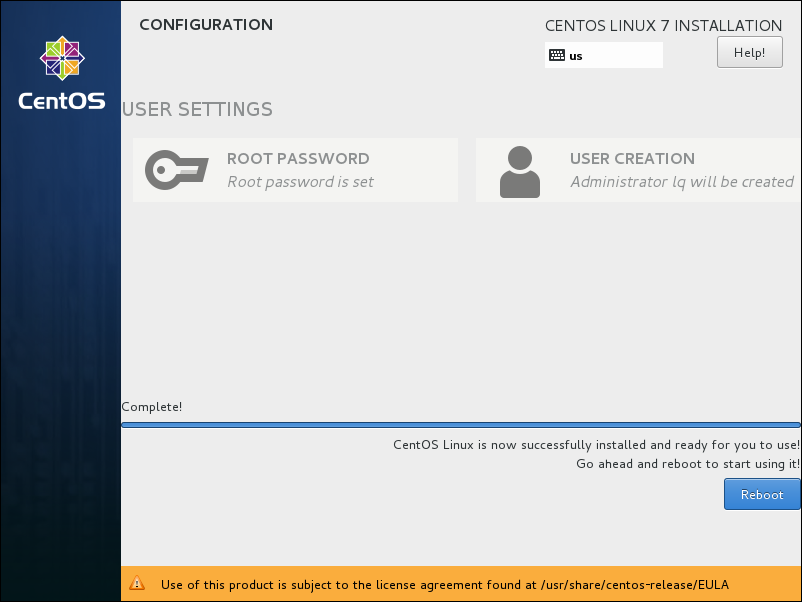




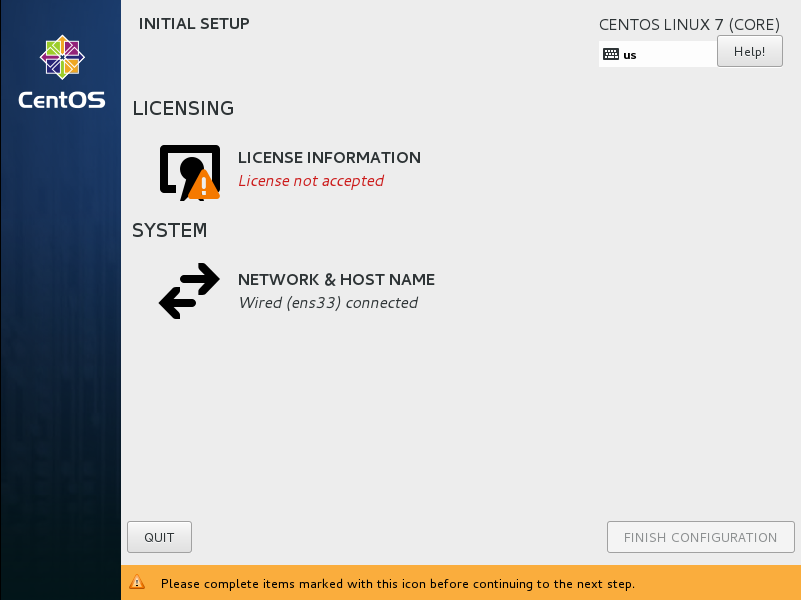
之后系统会进入安装过程，在其他的教程里面，大家会说这时候你可以去喝一杯咖啡了；你以为真的可以去喝咖啡啦？在进行这一步的时候我们需要去装下一台服务器了 o\_O

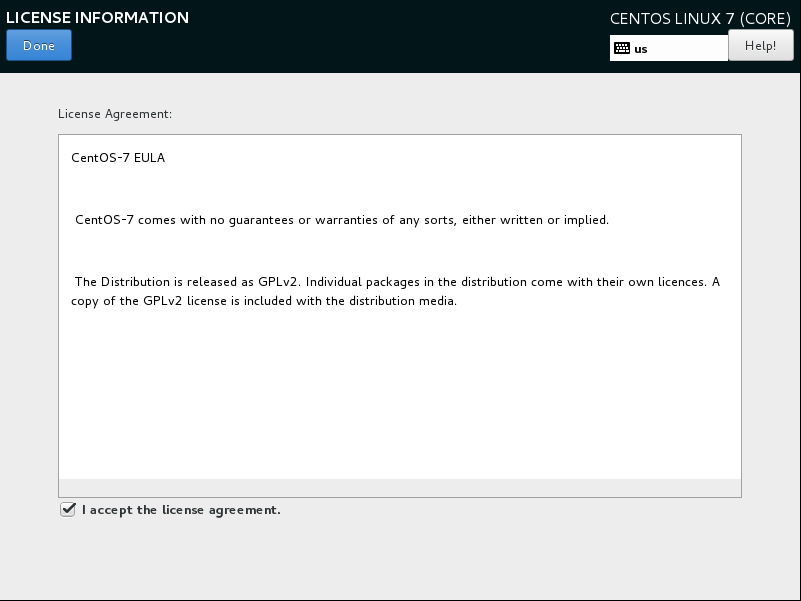


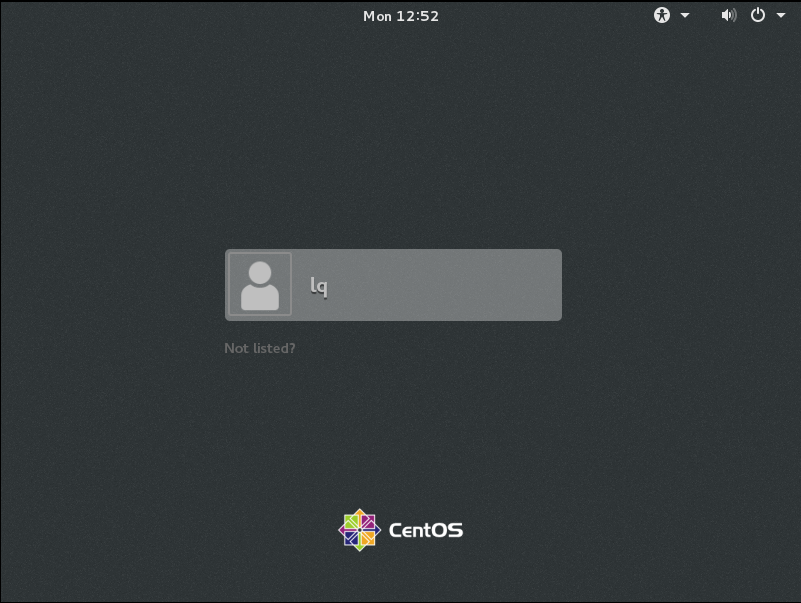
装完之后它是这个样子，需要重启一下。



重启过后接受许可证，系统安装部分就OK了。







在比赛时如果需要关闭 图形界面，可以用以下命令

$ systemctl disable gdm  
$ systemctl stop gdm

平时使用的时候由于 VTune 会用到GUI界面，这里我们就不禁用图形界面了。

按 Ctrl + Alt + F2 进入字符界面，然后就可以开心地进行下一步的系统配置工作啦 ˋ(′～｀")ˊ

在CentOS中，Ctrl + Alt + F1 为图形界面（如果有），Ctrl + Alt + F2-F6为字符界面，相当于可以开5个窗口，并可以随时切换。

# 高性能集群搭建指南（二） - 系统配置

2017-05-15 22:05:49

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

## 一、hostname hosts 和 IP

### 1. hostname 设定

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

#### 1.1 暂时更改 hostname

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

实际测试如下

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

#### 1.2 永久更改 hostname

将主机名写入 /etc/hostname 中，在系统下次重启后生效。

### 2. hosts

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

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

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

以下是一个样例：

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

### 3. IP

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

通过 ip addr 命令查看各个网口的 IP详细信息。

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

#### 3.1 网络服务的几个守护进程

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

$ systemctl disable NetworkManager  
$ systemctl stop NetworkManager

#### 3.2 配置文件

* 修改网口配置文件

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

样例如下 ↓

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

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

* 修改 DNS 服务器配置

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

$ vim /etc/resolv.conf

加入以下内容

# Generated by NetworkManager  
nameserver 202.106.0.20  
nameserver 208.67.222.222  
nameserver 202.112.128.51

修改完后重启network

$ systemctl restart network

#### 3.3 以太网带宽测试

* 安装 iperf

$ yum install iperf -y

* 服务端

$ iperf -s # 通过 -s 指定为Server端

* 客户端

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

#### 3.4 碰到过的一些问题

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

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

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

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

## 二、SELinux和防火墙

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

建议在装机完成后马上关闭SELinux。

### 1. SELinux模式

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

由于SELinux未关闭，在搭本地源的时候出现过客户端无法访问的情况。

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

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

### 2. 关闭SELinux

SELinux 的配置文件为 /etc/selinux/config

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

### 3. 查看SELinux工作 状态

sestatus 或 getenforce

$ sestatus  
SELinux status: disabled  
  
$ getenforce  
Disabled

### 4. 关于 setenforce

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

DESCRIPTION

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

### 5. 关闭防火墙

防火墙关闭后的安全问题有待探讨

$ systemctl stop firewalld  
$ systemctl disable firewalld

### 6. 碰到的一些问题

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

## 三、yum 本地源 （可选）

这里先介绍一下CentOS的软件包管理工具 yum

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

配置文件

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

### 1. 服务端配置

#### 1.1 准备rpm包

* 将 Centos-Base 源同步至本地目录

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

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

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

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

* 将epel源同步至本地目录

执行以下脚本即可

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

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

#### 1.2 配置repo

* 先做好备份

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

* CentOS-Base源

$ vim CentOS-Server.repo

将以下内容写入 CentOS-Server.repo

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

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

* epel源

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

将以下内容写入 epel-Server

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

#### 1.3 更新软件包缓存

$ yum clean all  
$ yum makecache

这样服务端的 yum 源就配置好了，可以简单地验证是否正常。

$ yum install tree -y

#### 1.4 ftp服务器配置

我们通过在服务端的 ftp服务，实现局域网的 yum源的搭建。

* ftp 安装

$ yum install vsftpd -y

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

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

* ftp 服务启动

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

### 2. 客户端

客户端的配置相对来说会快很多，只需要修改repo 文件即可

#### 2.1 配置 repo

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

* CentOS-Base源

$ vim CentOS-Client.repo

将以下内容写入 CentOS-Client.repo

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

* epel源

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

将以下内容写入 epel-Client

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

#### 2.2 更新软件包缓存

$ yum clean all  
$ yum makecache

### 3.碰到过的一些问题

#### 3.1 服务端

* 未复制 repodata 目录导致 repomd.xml 缺失

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

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

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

#### 3.2 客户端

* 服务端未禁用 SELinux 导致无法找到 repomd.xml 文件

## 三、SSH无密码访问

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

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

ssh的工作机制

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

ssh中的文件

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

### 1. OpenSSH 安装

$ yum install openssh -y

### 2. 单机的无密码访问配置

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

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

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

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

当然，拷贝公钥也可以通过 scp 来实现。

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

### 3. 集群的无密码访问配置

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

脚本如下 ↓

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

## 四、InfiniBand 配置

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

### 1. 安装TCL TK

$ yum install tcl tk -y

### 2. 安装 OFED驱动

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

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

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

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

* IP 配置

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

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

这里配置完IP后，需要修改各个机器上的 hosts 文件。

* 开机启动

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

* 重启网络接口和服务

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

* 查看ib 状态

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

### 3. 重启

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

### 4. IB读写带宽测试

* 服务端

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

* 客户端

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

## 五、NFS文件系统 exports

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

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

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

### 1. 安装NFS服务

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

$ yum install rpcbind nfs-utils -y

### 2. NFS系统守护进程

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

设置NFS守护进程开机启动

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

### 3. NFS服务器配置

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

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

/etc/exports 文件内容格式：

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

目前我的配置选项为 ↓

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

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

#### 4. 服务端及客户端启动NFS

配置好 /etc/exports 文件后，就可以启动NFS服务器了。

* 服务端

创建目录并修改文件夹权限

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

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

重启NFS服务

$ systemctl restart rpcbind  
$ systemctl restart nfs

* 客户端

将服务端NFS目录挂载到本机相同目录下

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

#### 5. NFS 访问权限测试

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

1. 客户端连接时候，对普通用户的检查

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

1. 客户端连接的时候，对root的检查

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

#### 6. 相关命令

$ nfsstat

查看NFS的运行状态，对于调整NFS的运行有很大帮助。

$ rpcinfo

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

$ showmount

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

$ netstat

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

#### 7. 特别注意！！！ 先卸载，再关机！！

卸载NFS目录

$ umount /home/cluster

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

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

#### 8. 通过挂载 tmpfs 提高NFS文件系统IO速度

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

* 挂载 tmpfs

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

也可以直接通过 mount 命令来查看是否成功挂载

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

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

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

* 卸载 tmpfs

$ umount /home/cluster

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

# 高性能集群搭建指南（三） - 软件环境

2017-05-15 22:05:49

## 一、软件环境

### 1. Intel Parallel Stdio

Intel Parallel Stido 内含 Compilers and Libraries, vtune amplifier, inspector, advisor 等。

Compilers and Libraries 包括 C, C++, Fortran编译器和 MKL，MPI库。  
VTune amplifier 是用来分析并行程序性能的工具。

#### 1.1 软件包和序列号

在 Intel 官网中可以申请30天注册版本： <https://software.intel.com/en-us/intel-parallel-studio-xe>

Intel 为学生提供了免费的序列号： <https://software.intel.com/en-us/qualify-for-free-software/student>

#### 1.2 安装 Parallel Stdio

基本全部按回车就行，以下是安装时的记录。

--------------------------------------------------------------------------------  
Initializing, please wait...  
Step 1 of 6 | Welcome  
--------------------------------------------------------------------------------  
Welcome to the Intel(R) Parallel Studio XE 2017 Update 4 for Linux\* setup program.  
--------------------------------------------------------------------------------  
--------------------------------------------------------------------------------  
  
  
You will complete the steps below during setup process:  
Step 1 : Welcome  
Step 2 : License agreement  
Step 3 : License activation  
Step 4 : Options  
Step 5 : Installation  
Step 6 : Complete  
  
--------------------------------------------------------------------------------  
Press "Enter" key to continue or "q" to quit: **#回车**  
  
  
  
  
Step 2 of 6 | License agreement  
--------------------------------------------------------------------------------  
To continue with the installation of this product you are required to accept   
the terms and conditions of the End User License Agreement (EULA). The EULA   
is displayed using the 'more' utility. Press the spacebar to advance to the   
next page or enter 'q' to skip to the end. After reading the EULA, you must   
enter 'accept' to continue the installation or 'decline' to return to the   
previous menu.  
--------------------------------------------------------------------------------  
  
....这里会有一大堆的用户协议，翻到最底下...  
  
  
Document Title and Version: End User License Agreement for the Intel(R) Software  
Development Products (Version March 2016)  
   
\* Other names and brands may be claimed as the property of others  
--------------------------------------------------------------------------------  
Type "accept" to continue or "decline" to go back to the previous menu: **#accept**  
  
  
  
--------------------------------------------------------------------------------  
Checking the prerequisites. It can take several minutes. Please wait...  
Step 2 of 6 | Prerequisites > Missing Optional Prerequisite(s)  
--------------------------------------------------------------------------------  
--------------------------------------------------------------------------------  
There are one or more optional unresolved issues. It is highly recommended to  
resolve them all before you continue. You can fix them without exiting the setup  
program and re-check. Or you can exit the setup program, fix them and run the  
setup program again.  
--------------------------------------------------------------------------------  
Missing optional prerequisites  
-- Driver is already loaded.  
--------------------------------------------------------------------------------  
1. Skip missing optional prerequisites [default]  
2. Show the detailed info about issue(s)  
3. Re-check the prerequisites  
  
h. Help  
b. Back to the previous menu  
q. Quit  
--------------------------------------------------------------------------------  
Please type a selection or press "Enter" to accept default choice [1]: **#1 跳过缺少可选的先决条件[默认]**  
  
  
  
  
  
Step 3 of 6 | License activation  
--------------------------------------------------------------------------------  
If you have purchased this product and have the serial number and a connection  
to the internet you can choose to activate the product at this time.  
Alternatively, you can choose to evaluate the product or defer activation by  
choosing the evaluate option. Evaluation software will time out in about one  
month. You can also use license file or Intel(R) Software License Manager.  
--------------------------------------------------------------------------------  
1. Use existing license [default]  
2. I want to activate my product using a serial number  
3. I want to evaluate Intel(R) Parallel Studio XE 2017 Update 4 Cluster Edition   
for Linux\* or activate later (EXPIRED)  
4. I want to activate by using a license file, or by using Intel(R) Software  
License Manager  
  
h. Help  
b. Back to the previous menu  
q. Quit  
--------------------------------------------------------------------------------  
Please type a selection or press "Enter" to accept default choice [1]: **#3 输入序列号激活，用之前提到的学生免费的序列号；没有序列号的话选使用也可以。**  
Please type your serial number (the format is XXXX-XXXXXXXX): **#2HWS-VWK**M5BRR 这是一个可用的序列号，不行的话再申请一个  
--------------------------------------------------------------------------------  
Checking serial number...  
--------------------------------------------------------------------------------  
Activation completed successfully.  
--------------------------------------------------------------------------------  
Press "Enter" key to continue: **#回车**  
  
  
  
  
  
Step 4 of 6 | Options > Configure Cluster Installation  
--------------------------------------------------------------------------------  
This product can be installed on cluster nodes.  
--------------------------------------------------------------------------------  
1. Finish configuring installation target [default]  
  
2. Installation target [ Current system only ]  
  
h. Help  
b. Back to the previous menu  
q. Quit  
--------------------------------------------------------------------------------  
Please type a selection or press "Enter" to accept default choice [1]: **#1 完成配置安装目标[默认]**  
  
  
  
Step 4 of 6 | Options > Pre-install Summary  
--------------------------------------------------------------------------------  
Install location:  
 /opt/intel  
  
Component(s) selected:  
 Intel(R) Trace Analyzer and Collector 2017 Update 3 585MB  
 Intel(R) Trace Analyzer for Intel(R) 64 Architecture   
 Intel(R) Trace Collector for Intel(R) 64 Architecture   
 Intel(R) Trace Collector for Intel(R) Many Integrated Core Architecture   
  
 Intel(R) Cluster Checker 2017 Update 2 179MB  
 Cluster Checker common files   
 Cluster Checker Analyzer   
 Cluster Checker Collector   
  
 Intel(R) VTune(TM) Amplifier XE 2017 Update 3 1.2GB  
 Command line interface   
 Sampling Driver kit   
 Graphical user interface   
  
 Intel(R) Inspector 2017 Update 3 361MB  
 Command line interface   
 Graphical user interface   
  
 Intel(R) Advisor 2017 Update 3 714MB  
 Command line interface   
 Graphical user interface   
  
 Intel(R) C++ Compiler 17.0 Update 4 603MB  
 Intel C++ Compiler   
  
 Intel(R) Fortran Compiler 17.0 Update 4 608MB  
 Intel Fortran Compiler   
  
 Intel(R) Math Kernel Library 2017 Update 3 for C/C++ 2.9GB  
 Intel MKL core libraries for C/C++   
 Intel(R) Xeon Phi(TM) coprocessor support for C/C++   
 Cluster support for C/C++   
 Intel TBB threading support   
 GNU\* C/C++ compiler support   
  
 Intel(R) Math Kernel Library 2017 Update 3 for Fortran 3.0GB  
 Intel MKL core libraries for Fortran   
 Intel(R) Xeon Phi(TM) coprocessor support for Fortran   
 Cluster support for Fortran   
 GNU\* Fortran compiler support   
 Fortran 95 interfaces for BLAS and LAPACK   
  
 Intel(R) Integrated Performance Primitives 2017 Update 3 2.7GB  
 Intel IPP single-threaded libraries: General package   
  
 Intel(R) Threading Building Blocks 2017 Update 6 100MB  
 Intel TBB   
  
 Intel(R) Data Analytics Acceleration Library 2017 Update 3 3.4GB  
 Intel(R) Data Analytics Acceleration Library 2017 Update 3   
  
 Intel(R) MPI Library 2017 Update 3 1.0GB  
 Intel MPI Benchmarks 2017 Update 2   
 Intel MPI Library for applications running on Intel(R) 64 Architecture   
 Intel MPI Library for applications running on Intel(R) Many Integrated   
Core Architecture  
  
 Intel(R) Debugger for Heterogeneous Compute 2017 Update 4 622MB  
 GNU\* GDB 7.6 and ELFDWARF library   
  
 GNU\* GDB 7.10 95MB  
 GNU\* GDB 7.10 on Intel(R) 64   
  
 Intel(R) Debugger for Intel(R) MIC Architecture 2017 Update 4 135MB  
 GNU\* GDB 7.8   
 GDB Eclipse\* Integration   
  
Install space required: 14.0GB  
  
Driver parameters:  
 Sampling driver install type: Driver will be built  
 Load drivers: yes  
 Reload automatically at reboot: yes  
 Per-user collection mode: no  
 Drivers will be accessible to everyone on this system. To restrict access,  
 select Customize Installation > Change advanced options > Drivers are accessible to  
 and set group access.  
  
Installation target:  
 Install on the current system only  
  
--------------------------------------------------------------------------------  
1. Start installation Now [default]  
2. Customize installation  
  
h. Help  
b. Back to the previous menu  
q. Quit  
--------------------------------------------------------------------------------  
Please type a selection or press "Enter" to accept default choice [1]:  **#1 开始安装[默认]**  
  
--------------------------------------------------------------------------------  
Checking the prerequisites. It can take several minutes. Please wait...  
  
  
  
  
  
Step 4 of 6 | Prerequisites > Missing Optional Prerequisite(s)  
--------------------------------------------------------------------------------  
--------------------------------------------------------------------------------  
There are one or more optional unresolved issues. It is highly recommended to  
resolve them all before you continue. You can fix them without exiting the setup  
program and re-check. Or you can exit the setup program, fix them and run the  
setup program again.  
--------------------------------------------------------------------------------  
Missing optional prerequisites  
-- 32-bit libraries not found  
--------------------------------------------------------------------------------  
1. Skip missing optional prerequisites [default]  
2. Show the detailed info about issue(s)  
3. Re-check the prerequisites  
  
h. Help  
b. Back to the previous menu  
q. Quit  
--------------------------------------------------------------------------------  
Please type a selection or press "Enter" to accept default choice [1]:  **#1 跳过缺少的可选条件[默认]**  
  
  
  
  
  
Step 5 of 6 | Installation  
--------------------------------------------------------------------------------  
Each component will be installed individually. If you cancel the installation,  
some components might remain on your system. This installation may take several   
minutes, depending on your system and the options you selected.  
--------------------------------------------------------------------------------  
Installing Intel(R) Trace Analyzer for Intel(R) 64 Architecture component... done  
--------------------------------------------------------------------------------  
Installing Intel(R) Trace Collector for Intel(R) 64 Architecture component... done  
--------------------------------------------------------------------------------  
Installing Intel(R) Trace Collector for Intel(R) Many Integrated Core  
Architecture component... done  
--------------------------------------------------------------------------------  
Installing Cluster Checker common files component... done  
--------------------------------------------------------------------------------  
Installing Cluster Checker Analyzer component... done  
--------------------------------------------------------------------------------  
Installing Cluster Checker Collector component... done  
--------------------------------------------------------------------------------  
Installing Command line interface component... done  
--------------------------------------------------------------------------------  
Installing Sampling Driver kit component... done  
--------------------------------------------------------------------------------  
Installing Graphical user interface component... done  
--------------------------------------------------------------------------------  
Installing Command line interface component... done  
--------------------------------------------------------------------------------  
Installing Graphical user interface component... done  
--------------------------------------------------------------------------------  
Installing Command line interface component... done  
--------------------------------------------------------------------------------  
Installing Graphical user interface component... done  
--------------------------------------------------------------------------------  
Installing Intel C++ Compiler for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Intel C++ Compiler for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing Intel Fortran Compiler for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Intel Fortran Compiler for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing Intel MKL core libraries for C/C++ for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Intel MKL core libraries for C/C++ for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing Intel(R) Xeon Phi(TM) coprocessor support for C/C++ component... done  
--------------------------------------------------------------------------------  
Installing Cluster support for C/C++ component... done  
--------------------------------------------------------------------------------  
Installing Intel TBB threading support for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Intel TBB threading support for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing GNU\* C/C++ compiler support for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing GNU\* C/C++ compiler support for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing Intel MKL core libraries for Fortran for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Intel MKL core libraries for Fortran for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing Intel(R) Xeon Phi(TM) coprocessor support for Fortran component... done  
--------------------------------------------------------------------------------  
Installing Cluster support for Fortran component... done  
--------------------------------------------------------------------------------  
Installing GNU\* Fortran compiler support for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing GNU\* Fortran compiler support for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Fortran 95 interfaces for BLAS and LAPACK for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Fortran 95 interfaces for BLAS and LAPACK for Intel(R) 64  
component... done  
--------------------------------------------------------------------------------  
Installing Intel IPP single-threaded libraries for IA-32: General package  
component... done  
--------------------------------------------------------------------------------  
Installing Intel IPP single-threaded libraries for Intel(R) 64: General package   
component... done  
--------------------------------------------------------------------------------  
Installing Intel TBB component... done  
--------------------------------------------------------------------------------  
Installing Intel DAAL for IA-32 component... done  
--------------------------------------------------------------------------------  
Installing Intel DAAL for Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing Intel MPI Benchmarks 2017 Update 2 component... done  
--------------------------------------------------------------------------------  
Installing Intel MPI Library for applications running on Intel(R) 64  
Architecture component... done  
--------------------------------------------------------------------------------  
Installing Intel MPI Library for applications running on Intel(R) Many  
Integrated Core Architecture component... done  
--------------------------------------------------------------------------------  
Installing GNU\* GDB 7.6 and ELFDWARF library component... done  
--------------------------------------------------------------------------------  
Installing GNU\* GDB 7.10 on Intel(R) 64 component... done  
--------------------------------------------------------------------------------  
Installing GNU\* GDB 7.8 component... done  
--------------------------------------------------------------------------------  
Installing GDB Eclipse\* Integration component... done  
--------------------------------------------------------------------------------  
Finalizing product configuration...  
Preparing driver configuration scripts... done  
--------------------------------------------------------------------------------  
Installing drivers. It may take several minutes... done  
--------------------------------------------------------------------------------  
Sampling driver built successfully  
Sampling driver loaded successfully  
Sampling driver boot script installed successfully  
--------------------------------------------------------------------------------  
  
  
  
  
  
Step 6 of 6 | Complete  
--------------------------------------------------------------------------------  
--------------------------------------------------------------------------------  
Thank you for installing Intel(R) Parallel Studio XE 2017 Update 4  
Cluster Edition for Linux\*.  
  
If you have not done so already, please register your product with Intel  
Registration Center to create your support account and take full advantage of  
your product purchase.  
  
Your support account gives you access to free product updates and upgrades  
as well as Priority Customer support at the Online Service Center  
https://supporttickets.intel.com.  
  
Click here https://software.intel.com/en-us/python-distribution   
to download Intel(R) Distribution for Python\*  
This download will initiate separately. You can proceed with the installation  
screen instructions.  
--------------------------------------------------------------------------------  
Press "Enter" key to quit: **#安装完成**

#### 1.3 环境变量

Parallel Stdio 有一大堆的动态链接库可执行文件和需要添加到环境变量中，Intel 已经写好了相应的脚本，运行它的脚本即可完成所有环境变量的添加。

我们把它写到 /etc/profile 中，在用户登录时会自动运行该脚本，完成环境变量的添加。

$ echo "source /opt/intel/parallel\_studio\_xe\_2017.4.056/psxevars.sh" >> /etc/profile  
  
$ source /etc/profle   
Intel(R) Parallel Studio XE 2017 Update 4 for Linux\*  
Copyright (C) 2009-2017 Intel Corporation. All rights reserved.

可以检验一下

$ echo $PATH  
/usr/local/bin:/opt/intel/vtune\_amplifier\_xe\_2017.3.0.510739/bin64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/bin/intel64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin:/opt/intel/debugger\_2017/gdb/intel64\_mic/bin:/opt/intel/vtune\_amplifier\_xe\_2017.2.0.499904/bin64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/bin/intel64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin:/opt/intel/debugger\_2017/gdb/intel64\_mic/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/ibutils/bin:/root/bin  
  
$ echo $LD\_LIBRARY\_PATH   
/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/compiler/lib/intel64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/compiler/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/lib:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/mic/lib:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/ipp/lib/intel64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/compiler/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mkl/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/tbb/lib/intel64/gcc4.7:/opt/intel/debugger\_2017/iga/lib:/opt/intel/debugger\_2017/libipt/intel64/lib:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/daal/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/compiler/lib/intel64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/compiler/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/lib:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/mic/lib:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/ipp/lib/intel64:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/compiler/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mkl/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/tbb/lib/intel64/gcc4.7:/opt/intel/debugger\_2017/iga/lib:/opt/intel/debugger\_2017/libipt/intel64/lib:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/daal/lib/intel64\_lin:/opt/intel/compilers\_and\_libraries\_2017.4.196/linux/daal/../tbb/lib/intel64\_lin/gcc4.4:/usr/local/lib:

source /opt/intel/bin/compilervars.sh intel64 也可以，缺点是未添加VTune的路径。

#### 1.4 VTune 的使用

VTune是Intel开发的一个比较强大的并行程序性能分析工具

* 使用 amplxe-cl 命令可进行性能数据收集和分析

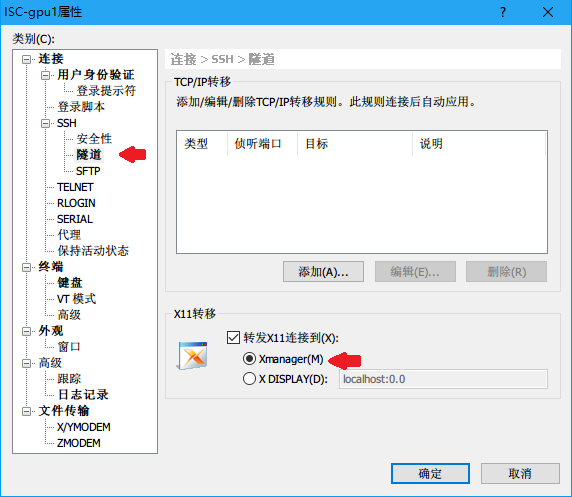
$ amplxe-cl -collect hotspots <需分析的可执行文件>

随后会在当前目录生成分析结果

* 通过 amplxe-gui 查看分析结果

由于需要使用 GUI 界面，故需要转发 X11

如果使用xshell连接服务器，可以转发 X11 到 xmanager中



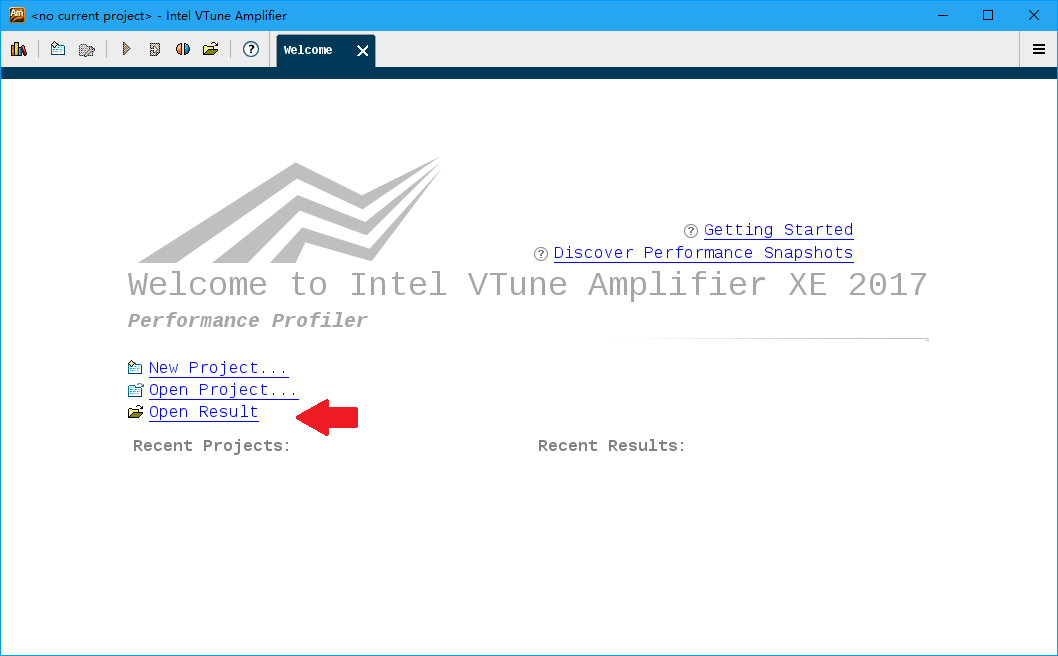
如果使用Linux的 Terminal 链接到服务器，可以通过 -Y 选项来转发X11

$ ssh lq@10.4.14.111 -Y

打开VTune分析界面

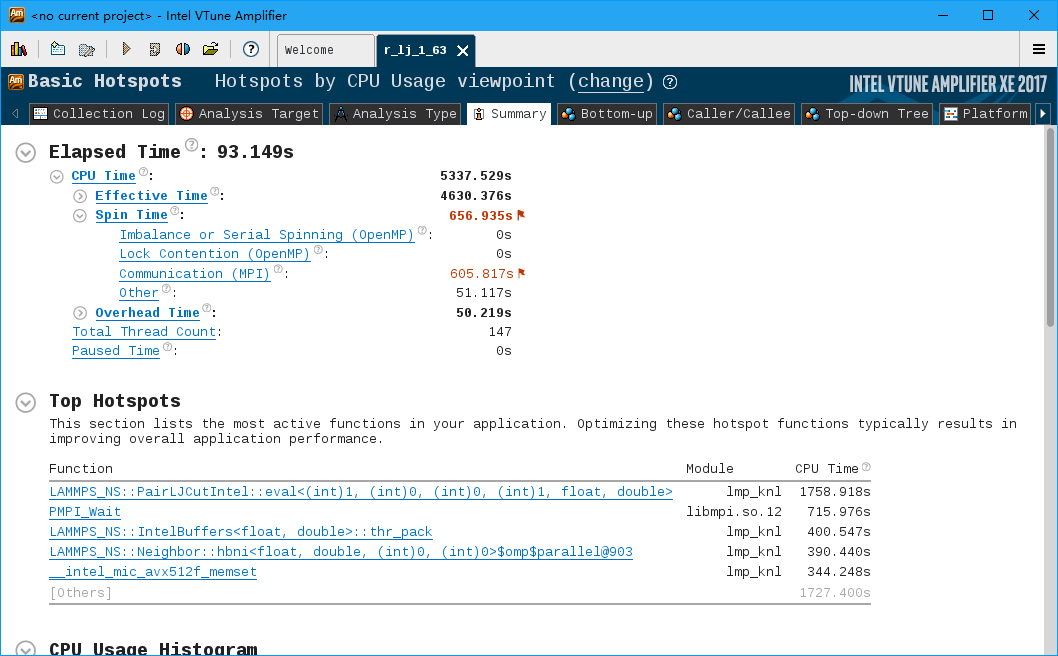
$ amplxe-gui

打开结果文件

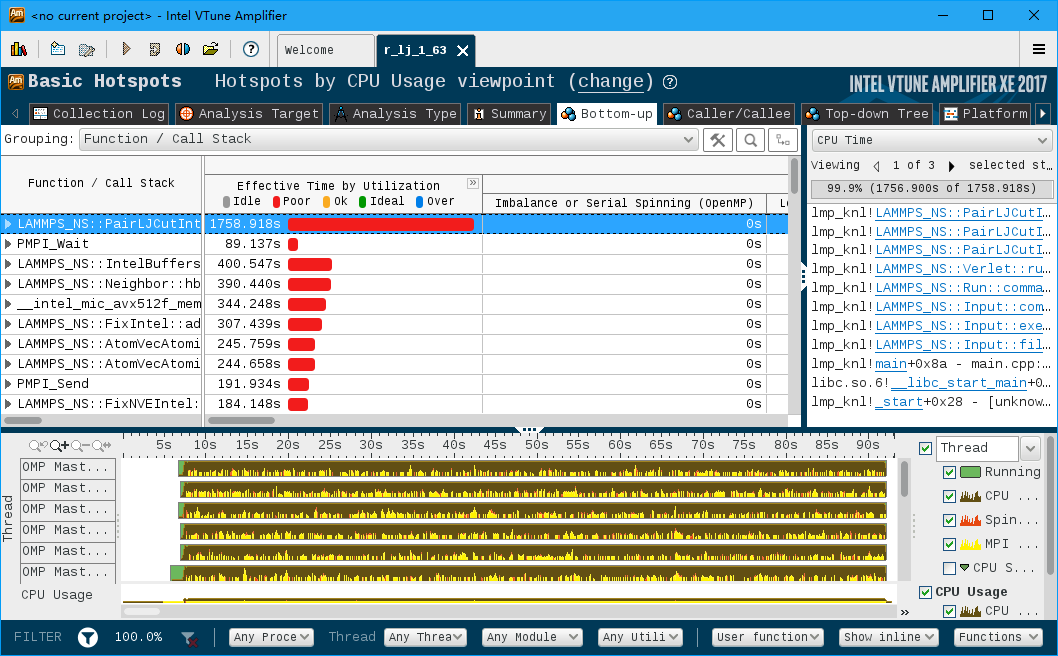




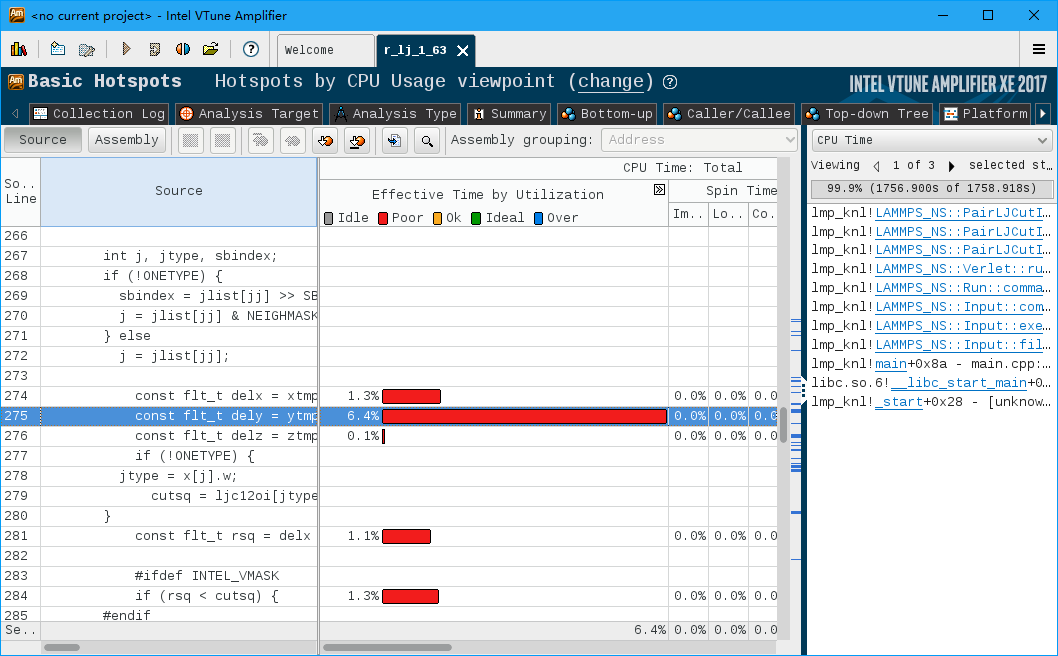
可以看到基本的分析结果



在Bottom-up中可以看到程序的瓶颈



进入热点函数中可以看到各行代码的时间占比



### 2. MPICH MVAPICH openMPI

MPICH 和 MVAPICH 以常规方式源码安装，下载官方源码解压即可。

下载链接如下 ↓  
[MPICH-3.2:http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz](http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz)  
[MVAICH-2.3a:](http://mvapich.cse.ohio-state.edu/download/mvapich/mv2/mvapich2-2.3a.tar.gz) <http://mvapich.cse.ohio-state.edu/download/mvapich/mv2/mvapich2-2.3a.tar.gz> [openMPI-2.1.1https://www.open-mpi.org/software/ompi/v2.1/downloads/openmpi-2.1.1.tar.gz](https://www.open-mpi.org/software/ompi/v2.1/downloads/openmpi-2.1.1.tar.gz)

#### 2.1 安装

$ ./configure --prefix=/opt/<dir\_name>  
$ make -j all #开启所有线程进行编译，也可以指定为特定线程数，如 make -j 20  
$ make install #需要权限

#### 2.2 环境变量

最好用过 module 来管理多个版本的mpi

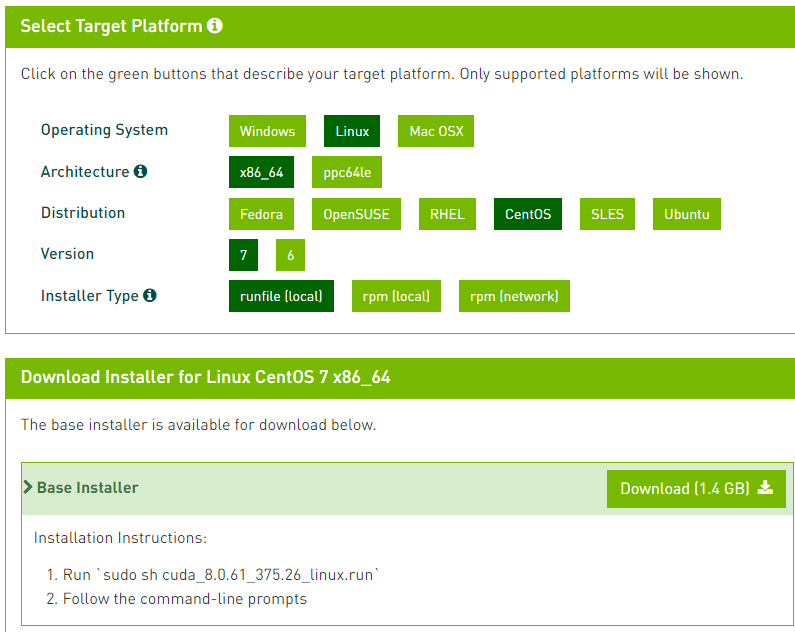
$ export PATH=<TOPdir>/bin:$PATH  
$ export LD\_LIBRARY\_PATH=<TOPdir>/lib:$LD\_LIBRARY\_PATH

### 3. cuda 驱动

#### 3.1 查看GPU

$ lspci | grep -i nvidia   
83:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)  
84:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)

#### 3.2 在nvidia开发者社区中下载软件包并安装



#### 3.3 环境变量

将环境变量添加至 /etc/profile 中

$ export LD\_LIBRARY\_PATH=/usr/local/cuda-8.0/lib64:$LD\_LIBRARY\_PATH  
$ export PATH=/usr/local/cuda-8.0/bin:$PATH

#### 3.4 查看GPU使用情况

$ nvidia-smi  
  
Tue May 23 15:02:42 2017   
+-----------------------------------------------------------------------------+  
| NVIDIA-SMI 375.26 Driver Version: 375.26 |  
|-------------------------------+----------------------+----------------------+  
| GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC |  
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |  
|===============================+======================+======================|  
| 0 Tesla K80 Off | 0000:83:00.0 Off | Off |  
| N/A 70C P0 113W / 149W | 3942MiB / 12205MiB | 81% Default |  
+-------------------------------+----------------------+----------------------+  
| 1 Tesla K80 Off | 0000:84:00.0 Off | Off |  
| N/A 58C P0 125W / 149W | 3942MiB / 12205MiB | 80% Default |  
+-------------------------------+----------------------+----------------------+  
   
+-----------------------------------------------------------------------------+  
| Processes: GPU Memory |  
| GPU PID Type Process name Usage |  
|=============================================================================|  
| 0 21271 C ./mini\_dft 3940MiB |  
| 1 21272 C ./mini\_dft 3940MiB |  
+-----------------------------------------------------------------------------+

### 4. RAPL

RAPL是CPU的功耗控制工具，其通过动态调整CPU频率来控制CPU的最高功率。

#### 4.1 查看CPU功耗信息

$ turbostat -i 1  
  
0.002701 sec  
 CPU Avg\_MHz %Busy Bzy\_MHz TSC\_MHz SMI CPU%c1 CPU%c3 CPU%c6 CPU%c7 CoreTmp PkgTmp Pkg%pc2 Pkg%pc3 Pkg%pc6 PkgWatt RAMWatt PKG\_% RAM\_%  
 - 89 7.18 1216 2539 0 30.26 3.14 59.42 0.00 43 48 25.74 0.00 0.00 43.50 9.68 0.00 0.00  
 0 98 7.96 1195 2572 0 27.03 0.00 65.01 0.00 38 45 25.28 0.00 0.00 18.40 5.58 0.00 0.00  
 1 119 9.47 1219 2584 0 28.75 0.00 61.78 0.00 41  
 2 145 11.72 1199 2574 0 29.63 12.31 46.35 0.00 40  
 3 105 8.52 1200 2562 0 25.53 10.98 54.96 0.00 43  
 4 95 7.52 1238 2550 0 35.52 0.00 56.97 0.00 41  
 5 65 5.35 1200 2548 0 31.44 4.72 58.49 0.00 37  
 6 66 5.39 1199 2547 0 33.05 3.07 58.49 0.00 38  
 7 74 6.02 1200 2544 0 33.09 1.37 59.52 0.00 40  
 8 74 6.01 1215 2542 0 33.10 3.51 57.39 0.00 38  
 9 100 7.90 1244 2537 0 30.90 1.86 59.34 0.00 38  
 10 72 5.91 1201 2536 0 29.84 0.00 64.25 0.00 40  
 11 69 5.39 1273 2532 0 27.91 0.00 66.70 0.00 39  
 12 91 7.45 1199 2547 0 28.14 12.70 51.71 0.00 38 48 25.78 0.00 0.00 25.11 4.09 0.00 0.00  
 13 66 5.30 1226 2546 0 27.75 10.88 56.06 0.00 41  
 14 71 5.57 1254 2533 0 30.28 0.00 64.15 0.00 41  
 15 70 5.76 1200 2530 0 33.03 0.00 61.21 0.00 43  
 16 92 7.03 1297 2526 0 33.50 2.85 56.62 0.00 39  
 17 67 5.48 1202 2530 0 34.73 5.15 54.64 0.00 42  
 18 77 6.35 1198 2530 0 34.08 3.58 55.99 0.00 41  
 19 64 5.17 1227 2529 0 30.65 1.97 62.22 0.00 43  
 20 64 5.28 1201 2525 0 29.36 0.00 65.37 0.00 41  
 21 74 6.15 1201 2518 0 27.66 0.00 66.19 0.00 40  
 22 63 5.25 1202 2507 0 26.06 0.00 68.68 0.00 40  
 23 248 20.45 1216 2495 0 25.20 0.00 54.35 0.00 39

* Busy\_MHZ: CPU忙碌时的平均时钟频率（处于“c0”状态）。
* CPU%c1: CPU处于c1状态的时间占比
* CPU%c3: CPU处于c3状态的时间占比
* CPU%c6: CPU处于c6状态的时间占比
* CPU%c7: CPU处于c7状态的时间占比
* PkgTmp: CPU温度
* PkgWatt: CPU功率
* RAMWatt: 内存功率

#### 4.2 重新编译

* 修改 RaplSetPowerSeprate.c

int main(int argc, char \*\*argv) {  
 int fd1,fd2;  
 int core1=0;  
  
 int core2=12; //@@lq core 2 is the first index of second CPU package.(number of one CPU cores) Example: for E5-2680v4 is 14

* 修改RaplPowerLimitDisable.c

fd=open\_msr(12); //@@lq modify the core num of one CPU here. Example: for E5-2680v4 is 14.

* 重新编译

$ make clean  
$ make

#### 4.3 设置功率

$ ./RaplSetPowerSeprate 120 120 #设置两个socket的最高功率

$ ./RaplPowerLimitDisable # reset

### 5. 通过modules 管理环境变量

#### 5.1 安装 modules

$ yum install environment-modules -y

#### 5.2 配置文件

示例如下

$ cat mvapich   
  
#%Module1.0  
prepend-path LD\_LIBRARY\_PATH /opt/mvapich2-2.3a/lib #动态链接库  
prepend-path PATH /opt/mvapich2-2.3a/bin #可执行文件

#### 5.3 加载

$ module use /home/cluster/modules #指定modules配置文件目录  
  
$ module load mpi/mvapich #加载环境变量模块  
$ module list #查看已加载模块  
Currently Loaded Modulefiles:  
1) mpi/mvapich  
  
$ module unload mpi/mvapich #卸载环境变量  
$ module list  
Currently Loaded Modulefiles:

# 高性能集群搭建指南（四） - 性能测试及功耗控制

2017-05-15 22:05:49

## 一. 计算性能测试

### 1. 理论峰值的计算

理论峰值仅代表着CPU的极限运算性能，只是一个参考。在实际中，由于CPU频率的调整、内存等其他硬件限制，和并行程序的并行效率等软件层面的限制；测试出来的计算性能会与理论峰值有些差距。

#### 1.1 支持 AVX2 指令集的 Intel64 CPU/MIC 的计算

主要有以下系列CPU：Haswell(v3); Broadwel(v4); Skylake (v5); Knights Corner(KNC)  
（ASC17用到的是 E5-2680v4）

此类CPU每个核心有16个向量化单元，即每秒可进行16次浮点运算。

DP FLOPS = cores/CPU \* CPU hz \* 16

如E5-2680v4为： 2（每个服务器有两块CPU） \* 14(每个CPU有14核心) \* 2.5GHz \* 16 = 1120 GFlops

#### 1.2 支持 AVX512 指令集的 CPU/MIC 的计算

目前支持 AVX512 指令集的只有 Knights Landing (KNL)

此类CPU每个核心有32个向量化单元，每秒可进行32次浮点运算。

DP FLOPS = cores/CPU \* CPU hz \* 16

如PHI-7210为：64（每块KNL有64核心） \* 1.3GHz \* 32 = 2662.4 GFlops

### 2. HPL(Linpack)

#### 2.1 HPL和Linpack介绍

Linpack现在在国际上已经成为最流行的用于测试高性能计算机系统浮点性能的benchmark。通过利用高性能计算机，用高斯消元法求解N元一次稠密线性代数方程组的测试，评价高性能计算机的浮点性能。

Linpack测试包括三类，Linpack100、Linpack1000和HPL。Linpack100求解规模为100阶的稠密线性代数方程组，它只允许采用编译优化选项进行优化，不得更改代码，甚至代码中的注释也不得修改。Linpack1000要求求解规模为1000阶的线性代数方程组，达到指定的精度要求，可以在不改变计算量的前提下做算法和代码上做优化。HPL即High Performance Linpack，也叫高度并行计算基准测试，它对数组大小N没有限制，求解问题的规模可以改变，除基本算法（计算量）不可改变外，可以采用其它任何优化方法。前两种测试运行规模较小，已不是很适合现代计算机的发展，因此现在使用较多的测试标准为HPL，而且阶次N也是linpack测试必须指明的参数。

HPL是针对现代并行计算机提出的测试方式。用户在不修改任意测试程序的基础上，可以调节问题规模大小N(矩阵大小)、使用到的CPU数目、使用各种优化方法等来执行该测试程序，以获取最佳的性能。HPL采用高斯消元法求解线性方程组。当求解问题规模为N时，浮点运算次数为(2/3\*N^3－2\*N^2)。因此，只要给出问题规模N，测得系统计算时间T，峰值=计算量(2/3\*N^3－2\*N^2)/计算时间T，测试结果以浮点运算每秒（Flops）给出

#### 2.2 HPL源码的编译

这里以官方版本的 HP-L2.2 为例。

选择合适的 Makefile，官方已经为不同架构、OS和编译器写好了一些，位于 TOPdir/setup 目录中。

$ cp setup/Make.Linux\_Intel64 ./

修改 Make.Linux\_Intel64，指定 TOPdir 位置

TOPdir = /where/you/install/hpl-2.2

编译

$ make arch=Linux\_Intel64 #也可以在 Makefile 中指定 arch;然后直接 make  
  
#编译正常后，生成了 bin 目录  
$ ls  
bin HISTORY INSTALL Make.Linux\_Intel64 man src TUNING  
BUGS hpl lib makes README testing www  
COPYRIGHT include Makefile Make.top setup TODO  
  
$ ls bin/Linux\_Intel64/  
HPL.dat xhpl

#### 2.3 HPL 在Intel64平台的测试

Intel MKL 库中带有优化过的 benchmark  
一般位于 /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mkl/benchmarks 中。

cd /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mkl/benchmarks/mp\_linpack  
vim HPL.dat

修改 HPL.dat

* Ns 在大规模集群上，N\*N\*8 一般为内存的80%，当集群规模较小时，这个比例会低一些。（如在ASC17中，选取的 N 为240000[应该不是最佳参数]，内存占比为67%，最终结果是 7.3TFlops ）  
  如果想快速测试，也可以选择一组小一些的参数。
* NBs 在 E5 及 KNL上测试发现是 192 为最佳参数。

$ vim HPL.dat  
  
HPLinpack benchmark input file  
Innovative Computing Laboratory, University of Tennessee  
HPL.out output file name (if any)  
6 device out (6=stdout,7=stderr,file)  
1 # of problems sizes (N)  
50000 Ns  
1 # of NBs  
192 NBs  
1 PMAP process mapping (0=Row-,1=Column-major)  
1 # of process grids (P x Q)  
1 2 Ps  
1 2 Qs  
16.0 threshold  
1 # of panel fact  
2 1 0 PFACTs (0=left, 1=Crout, 2=Right)  
1 # of recursive stopping criterium  
2 NBMINs (= 1)  
1 # of panels in recursion  
2 NDIVs  
1 # of recursive panel fact.  
1 0 2 RFACTs (0=left, 1=Crout, 2=Right)  
1 # of broadcast  
0 BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)  
1 # of lookahead depth  
0 DEPTHs (=0)  
0 SWAP (0=bin-exch,1=long,2=mix)  
1 swapping threshold  
1 L1 in (0=transposed,1=no-transposed) form  
1 U in (0=transposed,1=no-transposed) form  
0 Equilibration (0=no,1=yes)  
8 memory alignment in double ( 0)

运行HPL

$ export OMP\_NUM\_THREADS=28 #总核心数  
$ export MKL\_NUM\_THREADS=28  
$ .xhpl\_intel64\_dynamic  
  
  
--------------------------------------------------------------------------------  
Ax-b\_oo(eps(A\_oox\_oo+b\_oo)N)= 0.0045345 ...... PASSED  
mic3 Column=000384 Fraction=0.005 Kernel= 0.05 Mflops=1433902.37  
mic3 Column=000576 Fraction=0.010 Kernel=820465.80 Mflops=1149605.94  
mic3 Column=000768 Fraction=0.015 Kernel=824814.65 Mflops=1047665.59  
mic3 Column=001152 Fraction=0.020 Kernel=817380.27 Mflops=959001.49  
mic3 Column=001344 Fraction=0.025 Kernel=811729.33 Mflops=935323.50  
mic3 Column=001536 Fraction=0.030 Kernel=853574.35 Mflops=924556.05  
mic3 Column=001920 Fraction=0.035 Kernel=814404.21 Mflops=900944.94  
mic3 Column=002112 Fraction=0.040 Kernel=818573.97 Mflops=893093.53  
mic3 Column=002304 Fraction=0.045 Kernel=835688.62 Mflops=888227.39  
mic3 Column=002688 Fraction=0.050 Kernel=815766.35 Mflops=877614.73  
mic3 Column=002880 Fraction=0.055 Kernel=830918.27 Mflops=874518.04  
mic3 Column=003072 Fraction=0.060 Kernel=829651.65 Mflops=871745.04  
mic3 Column=003264 Fraction=0.065 Kernel=830779.03 Mflops=869381.82  
mic3 Column=003648 Fraction=0.070 Kernel=835668.02 Mflops=865950.94  
mic3 Column=003840 Fraction=0.075 Kernel=793611.73 Mflops=862315.32  
mic3 Column=004032 Fraction=0.080 Kernel=851690.33 Mflops=861843.77  
mic3 Column=004416 Fraction=0.085 Kernel=815254.39 Mflops=857935.27  
mic3 Column=004608 Fraction=0.090 Kernel=839449.17 Mflops=857220.41  
mic3 Column=004800 Fraction=0.095 Kernel=821844.63 Mflops=855887.10  
mic3 Column=005184 Fraction=0.100 Kernel=828691.61 Mflops=854017.78  
mic3 Column=005376 Fraction=0.105 Kernel=808901.12 Mflops=852502.22  
mic3 Column=005568 Fraction=0.110 Kernel=824008.89 Mflops=851600.16  
mic3 Column=005760 Fraction=0.115 Kernel=819855.29 Mflops=850629.38  
mic3 Column=006144 Fraction=0.120 Kernel=830269.63 Mflops=849483.98  
mic3 Column=006336 Fraction=0.125 Kernel=813456.77 Mflops=848491.23  
mic3 Column=006528 Fraction=0.130 Kernel=830828.00 Mflops=848031.17  
mic3 Column=006912 Fraction=0.135 Kernel=824199.60 Mflops=846857.05  
mic3 Column=007104 Fraction=0.140 Kernel=842360.80 Mflops=846752.60  
mic3 Column=007296 Fraction=0.145 Kernel=816742.67 Mflops=846056.54  
mic3 Column=007680 Fraction=0.150 Kernel=820905.53 Mflops=844961.16  
mic3 Column=007872 Fraction=0.155 Kernel=825957.16 Mflops=844563.90  
mic3 Column=008064 Fraction=0.160 Kernel=847857.36 Mflops=844629.05  
mic3 Column=008256 Fraction=0.165 Kernel=823622.84 Mflops=844213.59  
mic3 Column=008640 Fraction=0.170 Kernel=826474.17 Mflops=843549.73  
mic3 Column=008832 Fraction=0.175 Kernel=805635.33 Mflops=842845.33  
mic3 Column=009024 Fraction=0.180 Kernel=826354.40 Mflops=842554.62  
mic3 Column=009408 Fraction=0.185 Kernel=823713.08 Mflops=841919.66  
mic3 Column=009600 Fraction=0.190 Kernel=843652.87 Mflops=841947.32  
mic3 Column=009792 Fraction=0.195 Kernel=816310.42 Mflops=841535.15  
mic3 Column=010176 Fraction=0.200 Kernel=827674.46 Mflops=841114.98  
mic3 Column=010368 Fraction=0.205 Kernel=820677.75 Mflops=840811.63  
mic3 Column=010560 Fraction=0.210 Kernel=819496.91 Mflops=840502.40  
mic3 Column=010752 Fraction=0.215 Kernel=837126.18 Mflops=840455.57  
mic3 Column=011136 Fraction=0.220 Kernel=812985.65 Mflops=839703.56  
mic3 Column=011328 Fraction=0.225 Kernel=824705.22 Mflops=839506.78  
mic3 Column=011520 Fraction=0.230 Kernel=821189.71 Mflops=839270.88  
mic3 Column=011904 Fraction=0.235 Kernel=817064.62 Mflops=838718.66  
mic3 Column=012096 Fraction=0.240 Kernel=835432.24 Mflops=838679.76  
mic3 Column=012288 Fraction=0.245 Kernel=802802.08 Mflops=838247.56  
mic3 Column=012672 Fraction=0.250 Kernel=828727.27 Mflops=838033.65  
mic3 Column=012864 Fraction=0.255 Kernel=843865.81 Mflops=838096.33  
mic3 Column=013056 Fraction=0.260 Kernel=787007.22 Mflops=837520.21  
mic3 Column=013440 Fraction=0.265 Kernel=831731.15 Mflops=837401.09  
mic3 Column=013632 Fraction=0.270 Kernel=828051.40 Mflops=837306.93  
mic3 Column=013824 Fraction=0.275 Kernel=849740.35 Mflops=837426.51  
mic3 Column=014016 Fraction=0.280 Kernel=812580.92 Mflops=837181.71  
mic3 Column=014400 Fraction=0.285 Kernel=819944.15 Mflops=836856.68  
mic3 Column=014592 Fraction=0.290 Kernel=799944.32 Mflops=836509.00  
mic3 Column=014784 Fraction=0.295 Kernel=845354.79 Mflops=836586.31  
mic3 Column=015168 Fraction=0.300 Kernel=816536.46 Mflops=836235.60  
mic3 Column=015360 Fraction=0.305 Kernel=803021.79 Mflops=835947.59  
mic3 Column=015552 Fraction=0.310 Kernel=854329.19 Mflops=836094.58  
mic3 Column=015936 Fraction=0.315 Kernel=807391.66 Mflops=835624.74  
mic3 Column=016128 Fraction=0.320 Kernel=812608.68 Mflops=835442.17  
mic3 Column=016320 Fraction=0.325 Kernel=806923.12 Mflops=835218.68  
mic3 Column=016512 Fraction=0.330 Kernel=832295.28 Mflops=835196.89  
mic3 Column=016896 Fraction=0.335 Kernel=815558.38 Mflops=834907.49  
mic3 Column=017088 Fraction=0.340 Kernel=821441.93 Mflops=834811.37  
mic3 Column=017280 Fraction=0.345 Kernel=812260.62 Mflops=834651.61  
mic3 Column=017664 Fraction=0.350 Kernel=823159.13 Mflops=834495.87  
mic3 Column=017856 Fraction=0.355 Kernel=801996.42 Mflops=834275.36  
mic3 Column=018048 Fraction=0.360 Kernel=831701.28 Mflops=834258.82  
mic3 Column=018432 Fraction=0.365 Kernel=827402.18 Mflops=834172.95  
mic3 Column=018624 Fraction=0.370 Kernel=820058.34 Mflops=834085.92  
mic3 Column=018816 Fraction=0.375 Kernel=810101.01 Mflops=833938.94  
mic3 Column=019008 Fraction=0.380 Kernel=836988.59 Mflops=833956.70  
mic3 Column=019392 Fraction=0.385 Kernel=831515.76 Mflops=833928.93  
mic3 Column=019584 Fraction=0.390 Kernel=796233.11 Mflops=833710.41  
mic3 Column=019776 Fraction=0.395 Kernel=817330.62 Mflops=833619.58  
mic3 Column=020160 Fraction=0.400 Kernel=822896.69 Mflops=833504.93  
mic3 Column=020352 Fraction=0.405 Kernel=866136.11 Mflops=833666.70  
mic3 Column=020544 Fraction=0.410 Kernel=767393.58 Mflops=833302.66  
mic3 Column=020928 Fraction=0.415 Kernel=831902.60 Mflops=833288.88  
mic3 Column=021120 Fraction=0.420 Kernel=811938.47 Mflops=833183.88  
mic3 Column=021312 Fraction=0.425 Kernel=838562.29 Mflops=833209.04  
mic3 Column=021504 Fraction=0.430 Kernel=798961.18 Mflops=833043.96  
mic3 Column=021888 Fraction=0.435 Kernel=840818.91 Mflops=833113.13  
mic3 Column=022080 Fraction=0.440 Kernel=783361.28 Mflops=832881.48  
mic3 Column=022272 Fraction=0.445 Kernel=836998.56 Mflops=832899.10  
mic3 Column=022656 Fraction=0.450 Kernel=799428.90 Mflops=832607.87  
mic3 Column=022848 Fraction=0.455 Kernel=824515.17 Mflops=832574.58  
mic3 Column=023040 Fraction=0.460 Kernel=820378.76 Mflops=832525.07  
mic3 Column=023424 Fraction=0.465 Kernel=827059.37 Mflops=832482.32  
mic3 Column=023616 Fraction=0.470 Kernel=835901.92 Mflops=832495.22  
mic3 Column=023808 Fraction=0.475 Kernel=794697.95 Mflops=832348.00  
mic3 Column=024192 Fraction=0.480 Kernel=806653.30 Mflops=832156.52  
mic3 Column=024384 Fraction=0.485 Kernel=847053.68 Mflops=832208.04  
mic3 Column=024576 Fraction=0.490 Kernel=778749.11 Mflops=832010.68  
mic3 Column=024768 Fraction=0.495 Kernel=841170.11 Mflops=832041.42  
mic3 Column=025920 Fraction=0.515 Kernel=821099.19 Mflops=831831.63  
mic3 Column=026880 Fraction=0.535 Kernel=810236.56 Mflops=831515.99  
mic3 Column=027840 Fraction=0.555 Kernel=818209.96 Mflops=831341.08  
mic3 Column=028800 Fraction=0.575 Kernel=813029.56 Mflops=831121.60  
mic3 Column=029760 Fraction=0.595 Kernel=804420.63 Mflops=830829.39  
mic3 Column=030912 Fraction=0.615 Kernel=806813.10 Mflops=830549.43  
mic3 Column=031872 Fraction=0.635 Kernel=807366.11 Mflops=830349.65  
mic3 Column=032832 Fraction=0.655 Kernel=806102.31 Mflops=830162.86  
mic3 Column=033792 Fraction=0.675 Kernel=792432.87 Mflops=829900.31  
mic3 Column=034752 Fraction=0.695 Kernel=798952.39 Mflops=829711.73  
mic3 Column=039936 Fraction=0.795 Kernel=787867.30 Mflops=828814.92  
mic3 Column=044928 Fraction=0.895 Kernel=700457.74 Mflops=827735.23  
mic3 Column=049920 Fraction=0.995 Kernel=363372.03 Mflops=826632.56  
================================================================================  
TV N NB P Q Time Gflops  
--------------------------------------------------------------------------------  
WC00C2R2 50000 192 1 1 101.00 8.25147e+02  
HPL\_pdgesv() start time Tue May 16 224657 2017  
  
HPL\_pdgesv() end time Tue May 16 224838 2017  
  
--------------------------------------------------------------------------------  
Ax-b\_oo(eps(A\_oox\_oo+b\_oo)N)= 0.0038828 ...... PASSED  
================================================================================

如上，简单的结果为 825.147GFlops 计算性能正常。

#### 2.4 HPL在GPU上的测试

* NVIDIA 开发者社区提供了开源的Fermi版本的HPL:  
  <https://developer.nvidia.com/rdp/assets/cuda-accelerated-linpack-linux64>  
  由于显存占用问题，在Pascal架构下跑这个版本的HPL只能达到理论峰值的一半左右。
* 在Pascal架构下，NVIDIA提供了HPL的二进制版本，直接运行即可。（不出意外的话，之后的Volta架构也会有NVIDIA官方的HPL版本。） 在运行时可能遇到动态链接库没链上的问题，把它手动链上就OK。

## 二、 网络测试

见 高性能集群搭建指南（二） - 系统配置 中 以太网带宽测试 及 IB读写带宽测试 。

br

## 三、 内存测试

通过 Stream 测试内存带宽实际性能，Intel Parallel Stido 带有优化过的 Stream 可执行文件，直接执行即可

$ cd /opt/intel/clck2017.2.019/provider/share/stream/intel64  
$ .stream  
  
-------------------------------------------------------------  
STREAM version $Revision 5.10 $  
-------------------------------------------------------------  
This system uses 8 bytes per array element.  
-------------------------------------------------------------  
Array size = 64000000 (elements), Offset = 0 (elements)  
Memory per array = 488.3 MiB (= 0.5 GiB).  
Total memory required = 1464.8 MiB (= 1.4 GiB).  
Each kernel will be executed 10 times.  
 The best time for each kernel (excluding the first iteration)  
 will be used to compute the reported bandwidth.  
-------------------------------------------------------------  
Number of Threads requested = 24  
Number of Threads counted = 24  
-------------------------------------------------------------  
Your clock granularityprecision appears to be 1 microseconds.  
Each test below will take on the order of 23663 microseconds.  
 (= 23663 clock ticks)  
Increase the size of the arrays if this shows that  
you are not getting at least 20 clock ticks per test.  
-------------------------------------------------------------  
WARNING -- The above is only a rough guideline.  
For best results, please be sure you know the  
precision of your system timer.  
-------------------------------------------------------------  
Function Best Rate MBs Avg time Min time Max time  
Copy 46400.0 0.027587 0.022069 0.037249  
Scale 46989.9 0.026737 0.021792 0.031459  
Add 48192.7 0.039438 0.031872 0.058144  
Triad 59983.3 0.036007 0.025607 0.044139  
-------------------------------------------------------------  
Solution Validates avg error less than 1.000000e-13 on all three arrays  
-------------------------------------------------------------

## 四、CPU功耗控制

### 1. 通过RAPL控制CPU功耗上限。

源码可在 <https://github.com/PUPiL2015/PUPIL/tree/master/src/RAPL> 上下载。

#### 1.1 编译

在重新编译前，需要修改每个socket的核心数。

$ vim RaplSetPowerSeprate.c

int main(int argc, char \*\*argv) {  
 int fd1,fd2;  
 int core1=0;  
 int core2=14; //修改为每个socket的核心数。  
 ...  
}

$ vim RaplPowerLimitDisable.c

fd=open\_msr(14); //找到这一行，改为每个socket的核心数。

$ make #重新编译，会有一些警告信息，可以忽略。

#### 1.2 设置功耗上限

./RaplSetPowerSeprate 120 120 #将两个socket最高功率分别设为120w

#### 1.3 重置初始设置

./RaplPowerLimitDisable

### 2. 通过BIOS设置RAPL

在浪潮的机器上，存在以上文件无法修改功率上限的问题。只能通过BIOS中RAPL进行控制。

### 3. 通过cpu调频控制CPU功耗

linux内核带有控制cpu频率的功率cpupower <https://wiki.archlinux.org/index.php/CPU_frequency_scaling>

$ sudo cpupower frequency-set -g performance # 设置为高性能模式  
$ sudo cpupower frequency-set -u 2400mhz # 将所有cpu核心的时钟频率设置为2400mhz

$ cpupower frequncey-info # 查看cpu频率设置信息  
  
analyzing CPU 0:  
 driver: intel\_pstate  
 CPUs which run at the same hardware frequency: 0  
 CPUs which need to have their frequency coordinated by software: 0  
 maximum transition latency: Cannot determine or is not supported.  
 hardware limits: 1.20 GHz - 3.30 GHz  
 available cpufreq governors: performance powersave  
 current policy: frequency should be within 1.20 GHz and 2.40 GHz.  
 The governor "powersave" may decide which speed to use  
 within this range.  
 current CPU frequency: Unable to call hardware  
 current CPU frequency: Unable to call to kernel  
 boost state support:  
 Supported: yes  
 Active: yes

$ watch "cat /proc/cpuinfo | grep MHz" #查看实时cpu频率,  
  
cpu MHz : 2199.937  
cpu MHz : 2299.968  
cpu MHz : 2299.968  
cpu MHz : 1200.093  
cpu MHz : 1199.906  
cpu MHz : 1200.000  
cpu MHz : 1200.000  
cpu MHz : 1200.000  
cpu MHz : 1599.750  
cpu MHz : 2299.875  
cpu MHz : 2299.593  
cpu MHz : 2300.062  
cpu MHz : 1200.000  
cpu MHz : 1200.000  
cpu MHz : 1979.156  
cpu MHz : 2299.968  
cpu MHz : 1698.937  
cpu MHz : 1200.000  
cpu MHz : 1349.062  
cpu MHz : 1699.968  
cpu MHz : 1399.968  
cpu MHz : 2299.781  
cpu MHz : 1200.093  
cpu MHz : 1200.000  
cpu MHz : 2300.062  
cpu MHz : 2300.156  
cpu MHz : 1199.906  
cpu MHz : 1199.906

## 五、 GPU功耗控制

cuda8.0 中提供了 nvidia-smi 的接口来控制和查看GPU的相关信息。  
一些重要的选项有 -pm -e -p -r -ac -rac -pl

### 1. 直接控制GPU功率上限（粗粒度）

$ sudo nvidia-smi -pl 250 #设置GPU功率上限为250w  
  
Power limit for GPU 0000:06:00.0 was set to 250.00 W from 250.00 W.  
Power limit for GPU 0000:84:00.0 was set to 250.00 W from 250.00 W.  
All done.

### 2. 控制GPU频率（细粒度）

* 查看可选频率信息

$ nvidia-smi -q -i 1 -d SUPPORTED\_CLOCKS  
  
==============NVSMI LOG==============  
  
Timestamp : Thu Jul 13 15:02:43 2017  
Driver Version : 375.26  
  
Attached GPUs : 2  
GPU 0000:84:00.0  
 Supported Clocks  
 Memory : 715 MHz  
 Graphics : 1328 MHz  
 Graphics : 1316 MHz  
 Graphics : 1303 MHz  
 Graphics : 1290 MHz  
 Graphics : 1278 MHz  
 Graphics : 1265 MHz  
 Graphics : 1252 MHz  
 Graphics : 1240 MHz  
 Graphics : 1227 MHz  
 Graphics : 1215 MHz  
 Graphics : 1202 MHz  
 Graphics : 1189 MHz  
 Graphics : 1177 MHz  
 Graphics : 1164 MHz  
 Graphics : 1151 MHz  
 Graphics : 1139 MHz  
 Graphics : 1126 MHz  
 Graphics : 1113 MHz  
 Graphics : 1101 MHz  
 Graphics : 1088 MHz  
 Graphics : 1075 MHz  
 Graphics : 1063 MHz  
 Graphics : 1050 MHz  
 Graphics : 1037 MHz  
 Graphics : 1025 MHz  
 Graphics : 1012 MHz  
 Graphics : 999 MHz  
 Graphics : 987 MHz  
 Graphics : 974 MHz  
 Graphics : 961 MHz  
 Graphics : 949 MHz  
 Graphics : 936 MHz  
 Graphics : 923 MHz  
 Graphics : 911 MHz  
 Graphics : 898 MHz  
 Graphics : 885 MHz  
 Graphics : 873 MHz  
 Graphics : 860 MHz  
 Graphics : 847 MHz  
 Graphics : 835 MHz  
 Graphics : 822 MHz  
 Graphics : 810 MHz  
 Graphics : 797 MHz  
 Graphics : 784 MHz  
 Graphics : 772 MHz  
 Graphics : 759 MHz  
 Graphics : 746 MHz  
 Graphics : 734 MHz  
 Graphics : 721 MHz  
 Graphics : 708 MHz  
 Graphics : 696 MHz  
 Graphics : 683 MHz  
 Graphics : 670 MHz  
 Graphics : 658 MHz  
 Graphics : 645 MHz  
 Graphics : 632 MHz  
 Graphics : 620 MHz  
 Graphics : 607 MHz  
 Graphics : 594 MHz  
 Graphics : 582 MHz  
 Graphics : 569 MHz  
 Graphics : 556 MHz  
 Graphics : 544 MHz

* 查看当前频率

$ nvidia-smi -q -i 1 -d CLOCK  
  
==============NVSMI LOG==============  
  
Timestamp : Thu Jul 13 15:04:03 2017  
Driver Version : 375.26  
  
Attached GPUs : 2  
GPU 0000:84:00.0  
 Clocks  
 Graphics : 405 MHz  
 SM : 405 MHz  
 Memory : 715 MHz  
 Video : 835 MHz  
 Applications Clocks  
 Graphics : 1189 MHz  
 Memory : 715 MHz  
 Default Applications Clocks  
 Graphics : 1189 MHz  
 Memory : 715 MHz  
 Max Clocks  
 Graphics : 1328 MHz  
 SM : 1328 MHz  
 Memory : 715 MHz  
 Video : 1328 MHz  
 SM Clock Samples  
 Duration : 0.16 sec  
 Number of Samples : 3  
 Max : 1189 MHz  
 Min : 405 MHz  
 Avg : 431 MHz  
 Memory Clock Samples  
 Duration : 0.16 sec  
 Number of Samples : 3  
 Max : 715 MHz  
 Min : 715 MHz  
 Avg : 715 MHz  
 Clock Policy  
 Auto Boost : N/A  
 Auto Boost Default : N/A

* 设置GPU频率

$ sudo nvidia-smi -ac 715, 1088 #设置Graphics时钟频率为1088MHz  
  
Applications clocks set to "(MEM 715, SM 1088)" for GPU 0000:06:00.0  
Applications clocks set to "(MEM 715, SM 1088)" for GPU 0000:84:00.0  
All done.