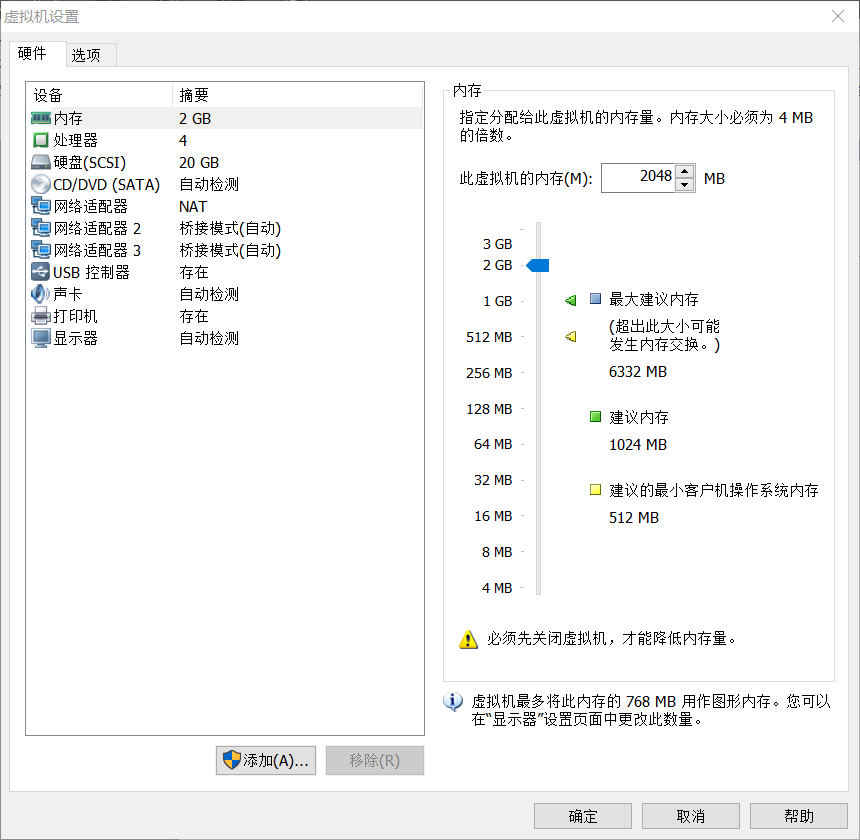
DPDK安装及示例程序使用指南（适用于虚拟机）

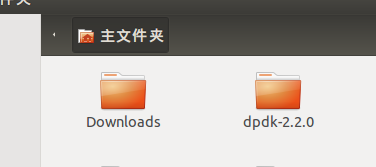
--torronto 2016.1.27

关于dpdk的介绍不用多说，主要就是它是intel开发的一个网络数据包查找转发的套件，用以分析网络数据的，所以只支持intel的网卡以及极少数除intel之外的网卡，具体支持的型号，官网有说明。因此，大多数时候，我们都是用虚拟机来仿真。

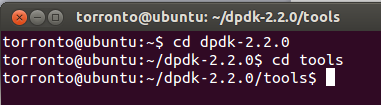
1. 在虚拟机中的ubuntu系统上手动设置2个网卡（一共3个），就使用默认的桥接模式，然后修改处理器个数为2个处理器，每个处理器2核心。内存分配，1GB以上，2GB更好。



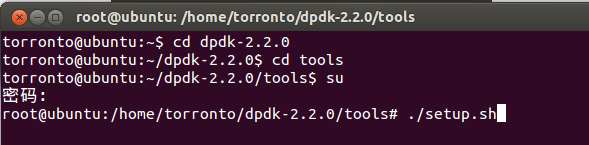
1. 去官网下载dpdk软件包，http://www.dpdk.eu/download
2. 将软件包解压在主目录下，根据个人喜好，因为后面编译和使用示例每次都要访问的。



1. 从终端进入



1. tools文件夹中有一个setup.sh方便新手完成dpdk的设置初始化操作：（当然，配置编译之前先进入特权模式）



1. 我们可以看到setup.sh里的一些选项如下：

------------------------------------------------------------------------------

RTE\_SDK exported as /home/torronto/dpdk-2.2.0

------------------------------------------------------------------------------

----------------------------------------------------------

Step 1: Select the DPDK environment to build

----------------------------------------------------------

[1] arm64-armv8a-linuxapp-gcc

[2] arm64-thunderx-linuxapp-gcc

[3] arm64-xgene1-linuxapp-gcc

[4] arm-armv7a-linuxapp-gcc

[5] i686-native-linuxapp-gcc

[6] i686-native-linuxapp-icc

[7] ppc\_64-power8-linuxapp-gcc

[8] tile-tilegx-linuxapp-gcc

[9] x86\_64-ivshmem-linuxapp-gcc

[10] x86\_64-ivshmem-linuxapp-icc

[11] x86\_64-native-bsdapp-clang

[12] x86\_64-native-bsdapp-gcc

[13] x86\_64-native-linuxapp-clang

[14] x86\_64-native-linuxapp-gcc

[15] x86\_64-native-linuxapp-icc

[16] x86\_x32-native-linuxapp-gcc

----------------------------------------------------------

Step 2: Setup linuxapp environment

----------------------------------------------------------

[17] Insert IGB UIO module

[18] Insert VFIO module

[19] Insert KNI module

[20] Setup hugepage mappings for non-NUMA systems

[21] Setup hugepage mappings for NUMA systems

[22] Display current Ethernet device settings

[23] Bind Ethernet device to IGB UIO module

[24] Bind Ethernet device to VFIO module

[25] Setup VFIO permissions

----------------------------------------------------------

Step 3: Run test application for linuxapp environment

----------------------------------------------------------

[26] Run test application ($RTE\_TARGET/app/test)

[27] Run testpmd application in interactive mode ($RTE\_TARGET/app/testpmd)

----------------------------------------------------------

Step 4: Other tools

----------------------------------------------------------

[28] List hugepage info from /proc/meminfo

----------------------------------------------------------

Step 5: Uninstall and system cleanup

----------------------------------------------------------

[29] Unbind NICs from IGB UIO or VFIO driver

[30] Remove IGB UIO module

[31] Remove VFIO module

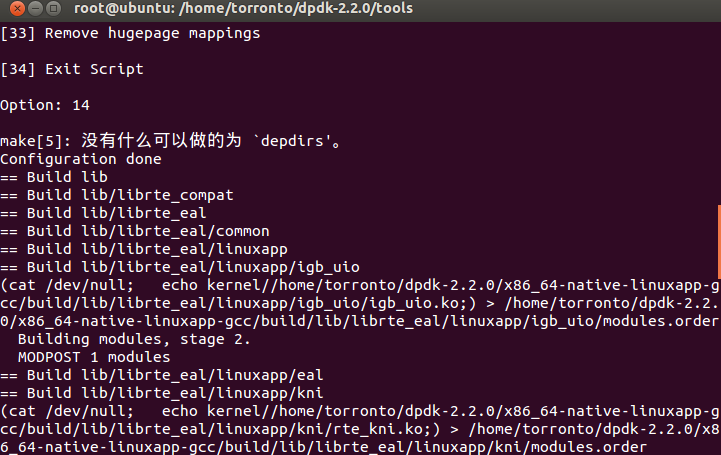
[32] Remove KNI module

[33] Remove hugepage mappings

[34] Exit Script

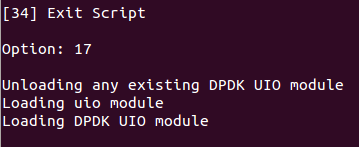
Option:

1. 先进行配置，我的系统是ubuntu12.04,64位，所以应该输入14，选择x86\_64-native-linuxapp-gcc进行编译，如果是32位系统则是x86\_x32-native-linuxapp-gcc，或者是i686-native-linuxapp-gcc，因为i386、i686都是32位的，平台选错了会导致后面编译不成功。编译界面如下，有点长。

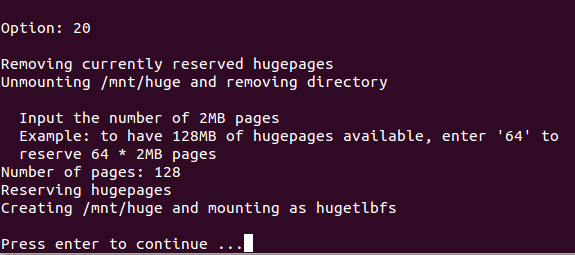


如果编译有错，往往是软件包依赖关系不满足，安装相应的软件包，对于ubuntu15.10出的错可能和内核有关，所以建议使用的linux3.x内核的。

1. 编译完成后，载入IGB组件

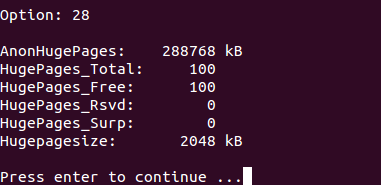


1. 设置页大小

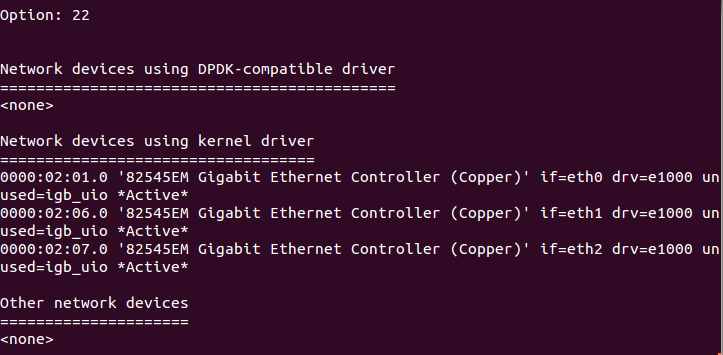


先设置位128，即128\*2MB的页大小

1. 选择[28] List hugepage info from /proc/meminfo可以查看页大小，以及多少页是空闲的

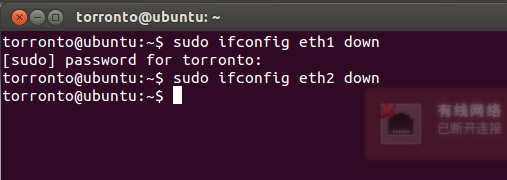


1. 查看当前网卡状态

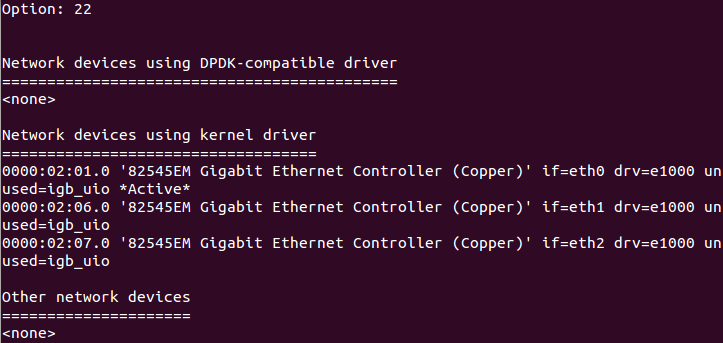


通过上图，我们看出有三块网卡，都是intel e1000型号的，下面两块网卡是我们后来手动添加的，然而都是active的状态，要将他们作为dpdk的网卡，就必须要先关掉

另开一个终端：



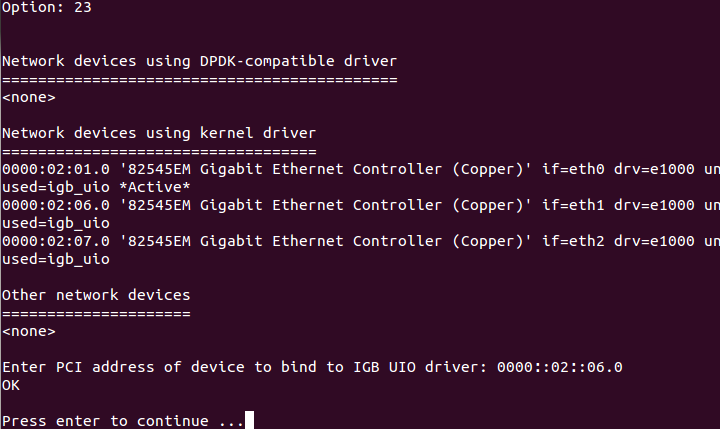
再次查看，网卡状态，可以进行网卡绑定了：



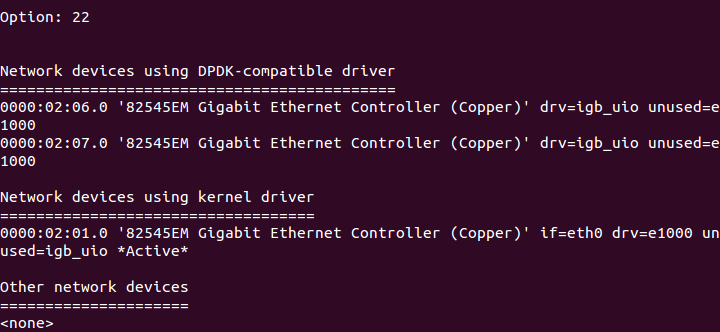
1. 绑定网卡

选择[23] Bind Ethernet device to IGB UIO module

输入网卡前面的数字序号，即可绑定该网卡，这里我们有两个需要绑定，eth1和eth2，它们的序号分别是0000:02:06.0和0000:02:07.0

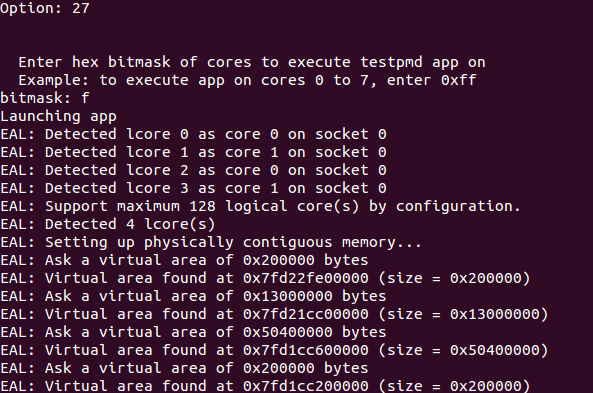


绑定完再次查看网卡状态，我们可以看见两个网卡都已经是用于dpdk了：

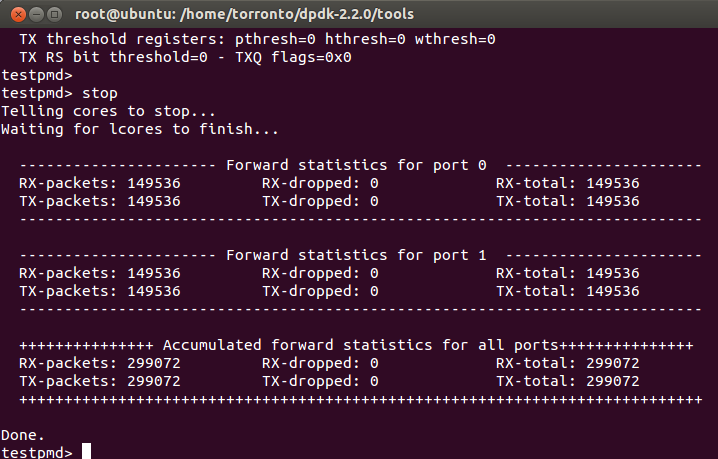


1. 到这一步dpdk的配置都已经结束了，下面可以运行一下dpdk自带的示例程序，比如说testpmd：

选择[27] Run testpmd application in interactive mode ($RTE\_TARGET/app/testpmd)



然后会出现testpmd>的显示



如下，即显示了start到stop之间发送接受的包数量

Port 0 Link Up - speed 1000 Mbps - full-duplex

Port 1 Link Up - speed 1000 Mbps - full-duplex

Done

testpmd> start

io packet forwarding - CRC stripping disabled - packets/burst=32

nb forwarding cores=1 - nb forwarding ports=2

RX queues=1 - RX desc=128 - RX free threshold=0

RX threshold registers: pthresh=0 hthresh=0 wthresh=0

TX queues=1 - TX desc=512 - TX free threshold=0

TX threshold registers: pthresh=0 hthresh=0 wthresh=0

TX RS bit threshold=0 - TXQ flags=0x0

testpmd>

testpmd> stop

Telling cores to stop...

Waiting for lcores to finish...

---------------------- Forward statistics for port 0 ----------------------

RX-packets: 149536 RX-dropped: 0 RX-total: 149536

TX-packets: 149536 TX-dropped: 0 TX-total: 149536

----------------------------------------------------------------------------

---------------------- Forward statistics for port 1 ----------------------

RX-packets: 149536 RX-dropped: 0 RX-total: 149536

TX-packets: 149536 TX-dropped: 0 TX-total: 149536

----------------------------------------------------------------------------

+++++++++++++++ Accumulated forward statistics for all ports+++++++++++++++

RX-packets: 299072 RX-dropped: 0 RX-total: 299072

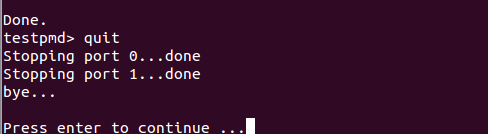
TX-packets: 299072 TX-dropped: 0 TX-total: 299072

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Done.

testpmd>

输入quit即可退出。

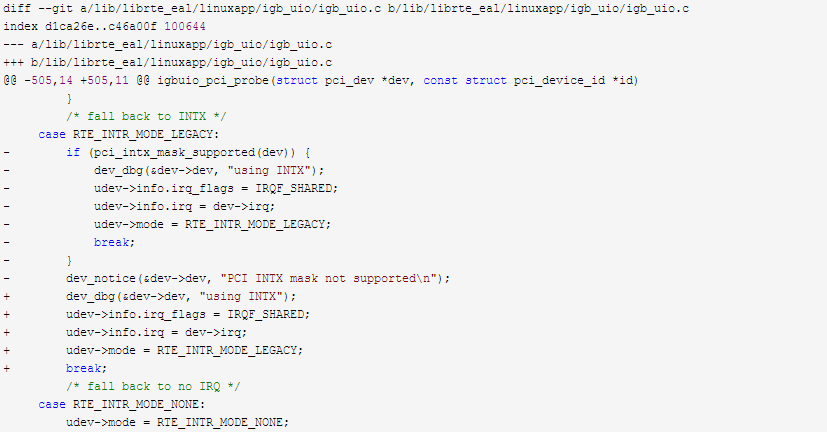


注意：每次使用dpdk的时候，都必须把加载IGB模块、设置页大小、绑定网卡做一遍，除非把这一些操作添加进启动项。

运行示例程序的过程中遇到的问题主要有两个，下面一一列举它们的解决办法：

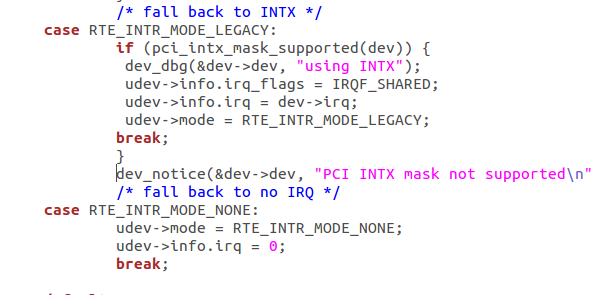
1. 在运行testpmd程序的过程中，显示EAL: Error reading from file descriptor 13: Input/output error刷屏。

解决办法：这是由于在虚拟机里运行造成的，dpdk的技术人员已经发布了补丁，只不过由于补丁：



然而由于补丁是1.7版本的，其他版本的只需要根据补丁内容对代码进行修改就可。

打开dpdk目录下lib/librte\_eal/linuxapp/igb\_uio里的igb\_uio.c文件，找到下图位置：



按照补丁说明，把

if (pci\_intx\_mask\_supported(dev)) {

dev\_dbg(&dev->dev, "using INTX");

udev->info.irq\_flags = IRQF\_SHARED;

udev->info.irq = dev->irq;

udev->mode = RTE\_INTR\_MODE\_LEGACY;

break;

}

dev\_notice(&dev->dev, "PCI INTX mask not supported\n");

代码替换为

dev\_dbg(&dev->dev, "using INTX");

udev->info.irq\_flags = IRQF\_SHARED;

udev->info.irq = dev->irq;

udev->mode = RTE\_INTR\_MODE\_LEGACY;

break;

保存退出，重新编译dpdk（从配置开始）即可成功运行。

1. 运行示例程序，显示EAL: Error - exiting with code: 1 Cause: Creation of mbuf pool for socket 0

意思是创建缓冲池失败，我们退回到分配页大小的地方，分配更大的hugepage，如果最大的hugepage仍然报错，就关闭虚拟机，给该虚拟机设置更大的内存，问题就得以解决。