VMware Player 搭建DPDK实验平台

原创 DPDK开源社区 2016-11-09

作者 马良

DPDK 今年越来越称为业内的热点， 很多初次上手的同学总是烦恼不知道如何入手，尤其是误以为DPDK只能运行在高端的服务器平台和千兆(含千兆以上)高端网卡设备上。殊不知，如果仅仅是以学习的话，无需任何特殊的硬件平台，只是需要一台稍微强劲些的PC(CPU i5, 内存8G以上) 就可以上手学习。

第一步: 安装所需软件

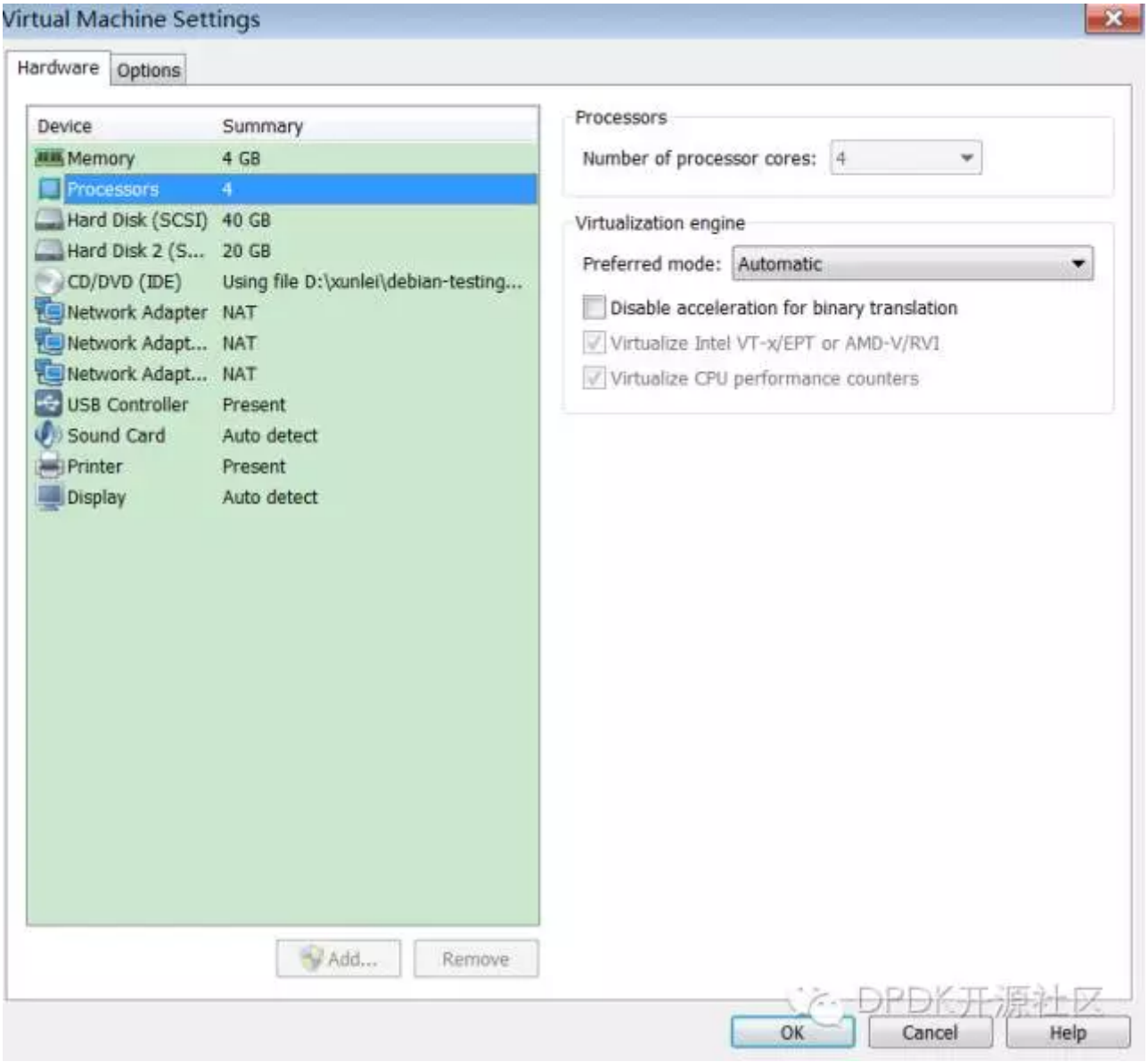
Linux/Windows 作为Host 环境

Vmware Player 12 作为hypervisor

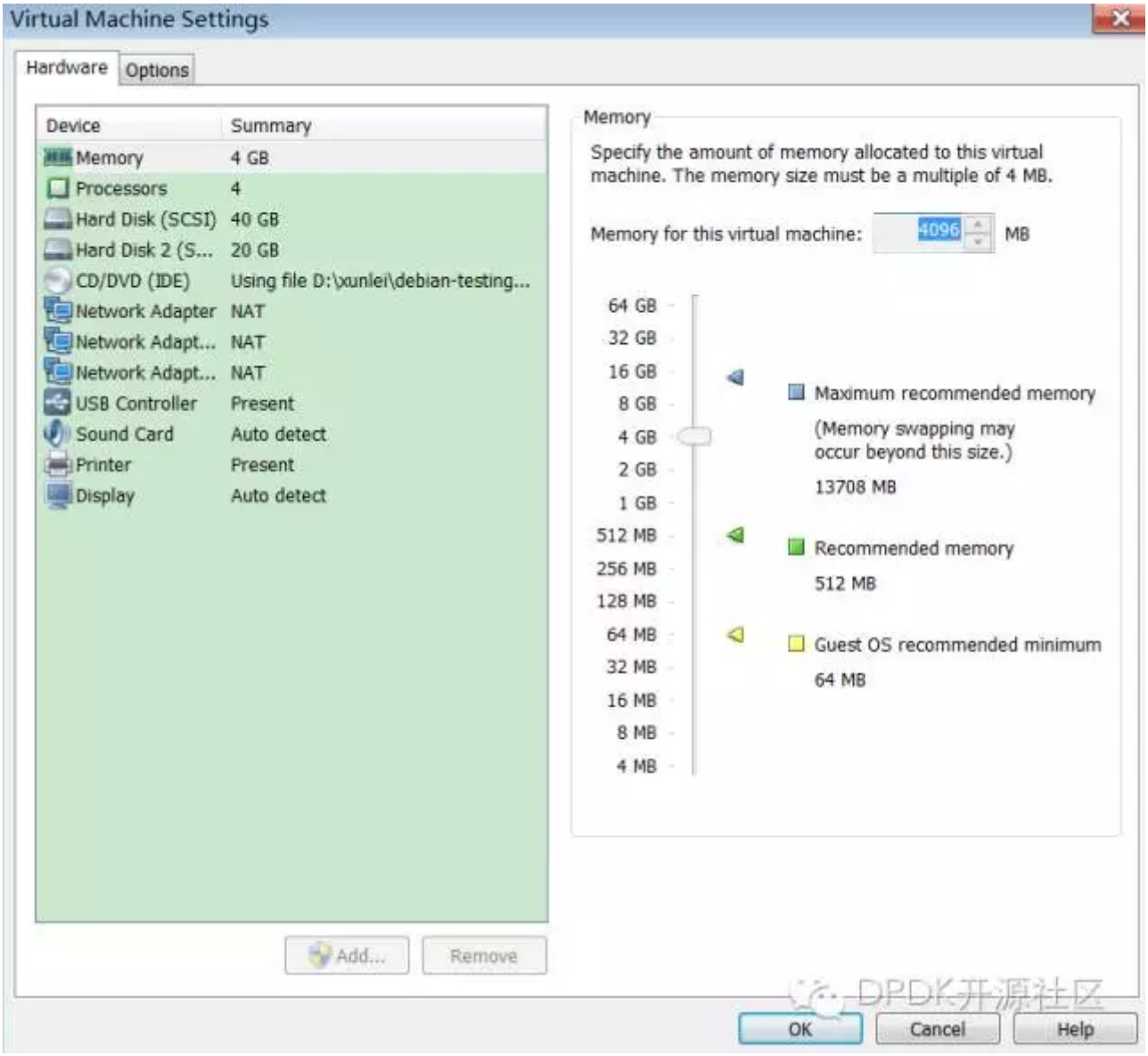
用Vmware Player 安装debian sid stable

第二步: Vmware Player的虚拟设备配置

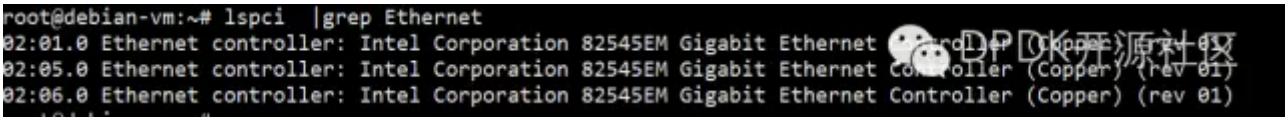
首先是设置为4core,并启动 vt-x. 这里cpu数量必须小于host的cpu实际数量。



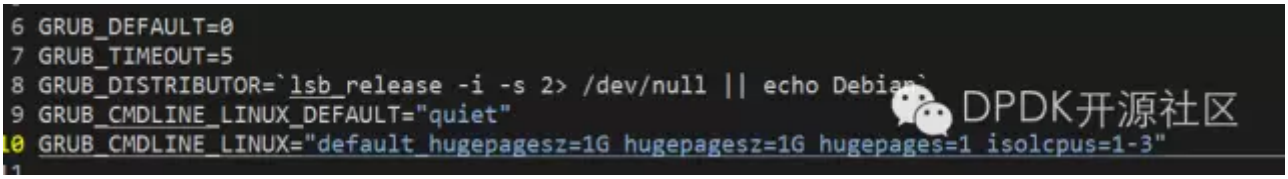
其次是设置内存， 4G 是基本的要求，因为要打开1G 巨页，内存不要太小。



然后添加3个网卡， vmware palyer模拟的是intel 的82554EM，恰好DPDK支持这款网卡，不过Vmware的模拟有一些问题,导致我们后面还需要打一个补丁，每个网卡都用NAT模式即可，添加后效果如下：



内核的启动参数写在 /etc/default/grub 的 GRUB_LINUX_CMD 后面。



后设置内核的启动参数，最终更新grub设置。

第三步：安装运行DPDK

首先前往dpdk.org 可以clone git repo 或者下载最新的发布，然后在debian 下面安装好编译器toolschain， 就不赘述首先要打一个补丁，补丁的地址

http://dpdk.org/dev/patchwork/patch/11622/

```
diff --git a/lib/librte_eal/linuxapp/igb_uio/igb_uio.c b/lib/librte_eal/linuxapp/igb_uio/igb_uio.c
index 72b2692..f5e9aeb 100644
--- a/lib/librte_eal/linuxapp/igb_uio/igb_uio.c
+++ b/lib/librte_eal/linuxapp/igb_uio/igb_uio.c
@@ -450,14 +450,11 @@  igbuio_pci_probe(struct pci_dev *dev, const struct pci_device_id *id)
                }
                /* fall back to INTX */
        case RTE_INTR_MODE_LEGACY:
-               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");
+                dev_dbg(&dev->dev, "using INTX");
+                udev->info.irq_flags = IRQF_SHARED;
+                udev->info.irq = dev->irq;
+                udev->mode = RTE_INTR_MODE_LEGACY;
+                break;
                /* fall back to no IRQ */
        case RTE_INTR_MODE_NONE:
                udev->mode = RTE_INTR_MODE_NONE;
```


这个补丁是必须的，否则会直接导致 应用crash。

接下来就是编译

```
make config T=x86_64-native-linuxapp-gcc
sed -ri 's,(PMD_PCAP=).*,\1y,' build/.config
make -j4
mkdir -p /mnt/huge
mount -t hugetlbfs nodev /mnt/huge
echo 1 > /sys/devices/system/node/node0/hugepages/hugepages-1048576/nr_hugepages
```

第四步：绑定网卡

然后就是进入 DPDK_ROOT/tools/

对网卡绑定用户态驱动框架 igb_uio

```
modprobe   uio
cd DPDK_ROOT/build/kmod
insmod    ./igb_uio.
```

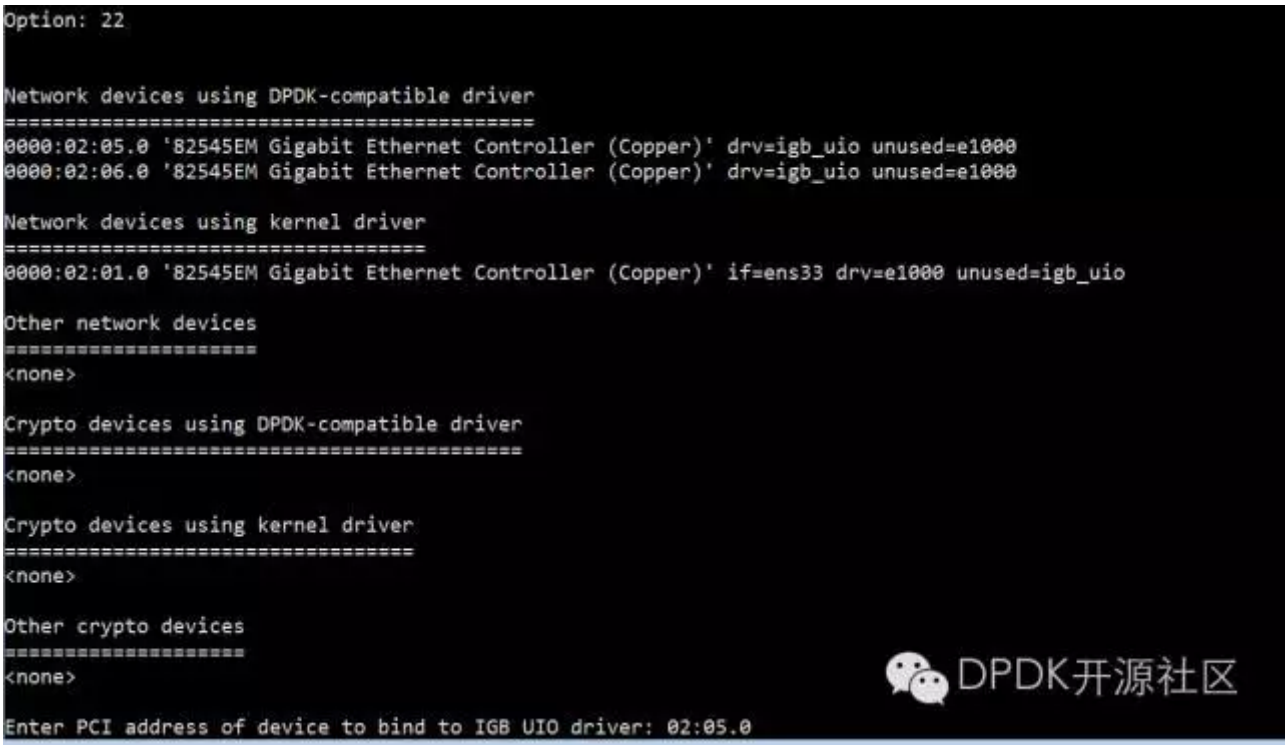
绑定2个网卡即可，还有一个需要留下继续使用内核驱动作为管理通道。

绑定方法有两种，一种是运行 dpdk-devbind.py 另一种是运行 dpdk-setup.py

对于初次使用的用户建议使用第二种， 交互式界面比较容易理解

运行以后首先选 22（bind ethernet device to igb_uio module)，这个数字可能根据版本不同不一致，注意功能描述。

绑定网卡是根据 网卡设备的 PCI-E Bus number: Device number: Function number ,简称BDF。如下图，我绑定igb_uio 到 02:05.0 这块网卡。输入BDF即可完成。



这里有一个关于debian网络配置的小细节： 在有三个网卡的情况下，缺省只有一块bus number最高的网卡会自动配置好ip地址。我们需要做一些小小的设置，这样三块网卡可以全部自动配置好。

在/etc/network/interface.d/ 目录下根据网卡设备名分别创建3个文件。例如， 在我的系统中三个网卡设备名分别是en33,en37,en38,则对应有三个同名文件。里面的内容为：

```
auto   ens33
iface   ens33 inet   dhcp
```

其余两个只要替换对应设备名即可。

第五步编译运行例子以及测试代码

首先回到 DPDK_ROOT

第三步编译结束之后，build 目录下面是生成的二进制文件(包括可执行文件以及ko，

然后copy 整个build 目录到 一个新的目录 下 DPDK_ROOT/x86_64-native-linuxapp-gcc

```
export   RTE_SDK=/opt/Code/dpdk   #请自行脑补其它shell，自己的代码路径
```

然后进入 DPDK_ROOT/build/app 可以先实验几个小的测试程序，例如，/test_pmd -- -i，进入交互模式，然后在控制台输入show port stats all。会有如下输出：

```
testpmd> show port stats all

  ######################## NIC statistics for port 0  ########################
  RX-packets: 127        RX-missed: 0          RX-bytes:  14252
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 0          TX-errors: 0          TX-bytes:  0

  Throughput (since last show)
  Rx-pps:            0
  Tx-pps:            0
  ############################################################################

  ######################## NIC statistics for port 1  ########################
  RX-packets: 127        RX-missed: 0          RX-bytes:  14252
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 0          TX-errors: 0          TX-bytes:  0

  Throughput (since last show)
  Rx-pps:            0
  Tx-pps:            0
  ############################################################################
```


我们可以看到我们绑定的两块网卡的一些基本状态信息。则基本大功告成， 我们现在可以编译正式的例子代码。

进入 DPDK_ROOT/example 目录， 挑选一个最为基本的例子l2fwd 就是二层包转发。

```
cd l2fwd ; make
```

生成的二进制可执行文件位于 l2fwd目录的build子目录下：/l2fwd -c f -n 2 -- -p 3

即可开始运行

```
Port statistics ====================================
Statistics for port 0 ------------------------------
Packets sent:                    5547886
Packets received:                1709697
Packets dropped:                       0
Statistics for port 1 ------------------------------
Packets sent:                    1709697
Packets received:                5550479
Packets dropped:                       0
Aggregate statistics ===============================
Total packets sent:              7260144
Total packets received:          7264794
Total packets dropped:                 0
====================================================
^C
```


大家可以观察包转发的包文数量的变化情况。至此，最基本的环境设置，编译运行已经完成。

作者简介：马良，现为Intel工程师。毕业于中国科学技术大学，主要从事高速包处理，虚拟化以及应用密码学等领域的研究。

DPDK开源社区 | 一个有用的公众号

投诉