Qemu/Kvm 搭建DPDK实验平台

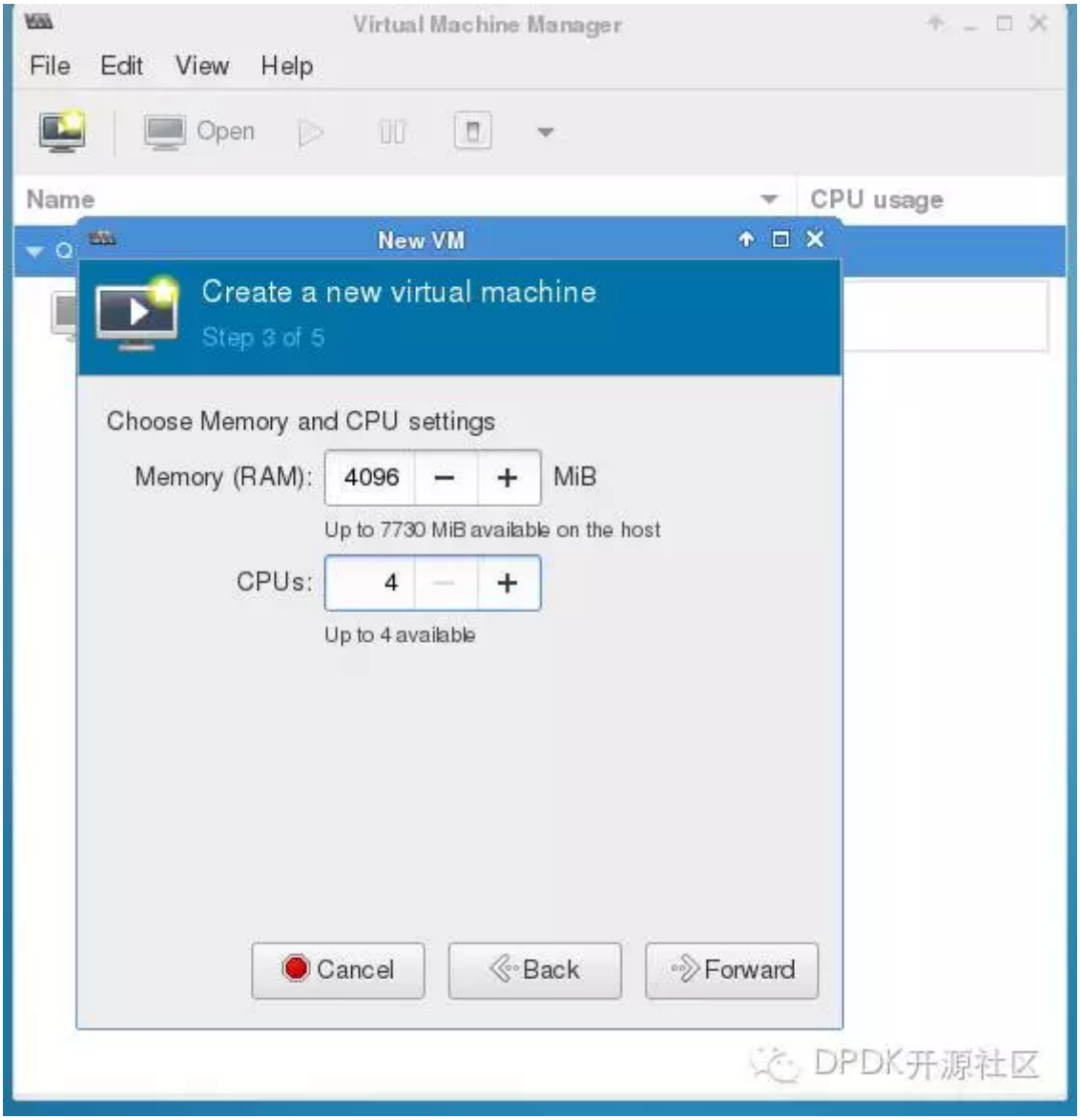
原创 DPDK开源社区 2016-11-09

作者 马良

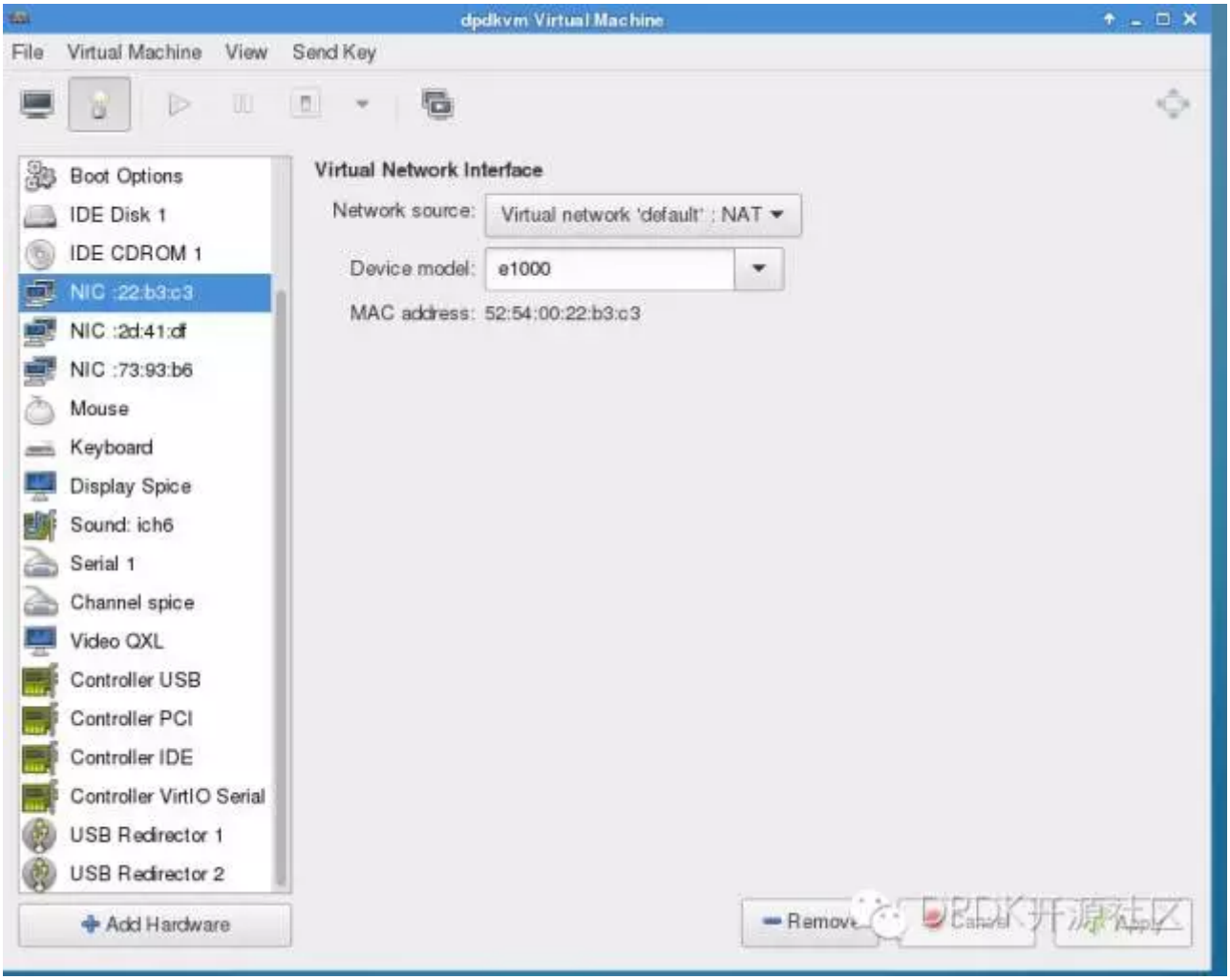
与Vmware Player 相比， Qemu/kvm的配置要相对复杂一些。 但是更加接近生产环境。首先需要一台安装有比较新的任何LInux 发布版本的机器。例如 centos 7, ubuntu 15.04/16.04 皆可。 CPU 至少是Intel Sandybridge 或者任何新于SandyBridge的型号。要求至少4逻辑核以上(HT也算) ， 内存4G或者以上。Guest 本例子采用的是ubutntu 16.04

第一步Host 软件安装：在Host主机上安装好 Qemu/Kvm，以及 virt-manager，这样生成虚拟机的XML配置文件会容易一些。安装好相关工具后， 即可开始安装Guest， 同时可以开启vncserver，便于远程登陆GUI 进行配置管理。

第二步安装Guest：利用virt-manager 可以从ubuntu16.04的iso文件安装整个OS(一定要安装openssh-server)， 在安装初始需要配置cpu数量为4， 内存为4G。其它按照默认配置安装即可，截图如下：



然后可以点击属性按钮，然后修改网卡的配置 网卡需要修改成e1000 类型，同时添加另外两个新的网卡(同样为e1000)类型。修改后的样子如下：



这里有个小技巧就是发现Guest OS的ip地址，虽然我们加入了3个网卡，但是ubuntu缺省只enable了一个，所以我们在host 执行arp -a 就会发现 192.168.122.80 的地址， ssh上去就是Guest。

```
# arp -a
? (192.168.122.80) at 52:54:00:22:b3:c3 [ether] on virbr0
? (192.168.1.9) at 00:1d:0f:33:08:9e [ether] on eno1
? (192.168.1.254) at 04:02:1f:0c:20:1b [ether] on eno1
```

第三步调整CPUID 标志位： virt-manager 不能做很细粒度的配置调整, 我们利用它生成了配置后可以用virsh 导出这个xml文件,

```
virsh  list
 Id    Name                           State
 3     dpdkvm                         running

virsh dumpxml 3 > dpdkvm.xml
```

Qemu/libvirt 会主动屏蔽一些CPUID 标志位, 这会给DPDK的配置带来一些困扰, 所以我们要针对这个导出的xml做一些细粒度的调整，分为以下两种情况：

1. Sandy bridge (不支持1G 巨页)

需要在xml文件中添加如下的部分

```
<vcpu placement='static'>4</vcpu>  #exist one
<cpu match='exact'>
<model>SandyBridge</model>
<feature policy='force' name='x2apic'/>
</cpu>
```

2. Haswell(支持巨页) pdpe1gb 是支持1G巨页的cpu id flag. 缺省是被屏蔽掉的

```
<cpu match='exact'>
<model>Haswell</model>
<feature policy='force' name='x2apic'/>
<feature policy='force' name='pdpe1gb'/>
<feature policy='disable' name='hle'/>
<feature policy='disable' name='smep'/>
<feature policy='disable' name='rtm'/>
</cpu>
```

现在shutdown 虚拟机 然后再用virsh 根据最新的 xml 创建新的虚拟机

```
virsh create  dpdkvm.xml
```

第四步 安装DPDK：完成以上三步以后， 就可以参照<<VMware Player 搭建DPDK 实验平台>>中的同样的步骤。

首先是在内核的启动参数中设置巨页, 需要注意的是 支持1G巨页的处理器可以设置为1G,否则只能设置为2M， 1G的巨页只需要1个,而2M的巨页最好设置为64个

```
# /etc/default/grub  in guest
GRUB_CMDLINE_LINUX="default_hugepagesz=2M hugepagesz=2M hugepages=64  isolcpus=1-3"
or
GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=1 isolcpus=1-3"
```

其余步骤都是相同的，就不赘述。

下面列出 使用64个2M巨页的testpmd执行情况：

```
cat /proc/meminfo

.........

HardwareCorrupted:       0 kB
AnonHugePages:        6144 kB
CmaTotal:                0 kB
CmaFree:                 0 kB
HugePages_Total:        64
HugePages_Free:         64
HugePages_Rsvd:          0
HugePages_Surp:          0
Hugepagesize:         2048 kB
DirectMap4k:         73720 kB
DirectMap2M:       4120576 kB
```

然后按照原有步骤编译执行testpmd

./testpmd -- -i --total-num-mbufs=2048 (需要限制一下mbuf的数量)

这个方案依然不是完整的生产环境， 不过对于学习和实验的目的来说以及足够。

谢谢大家的阅读，后续将深度介入I/O虚拟化技术细节，以及DPDK实现细节。

DPDK开源社区 | 一个有用的公众号

投诉