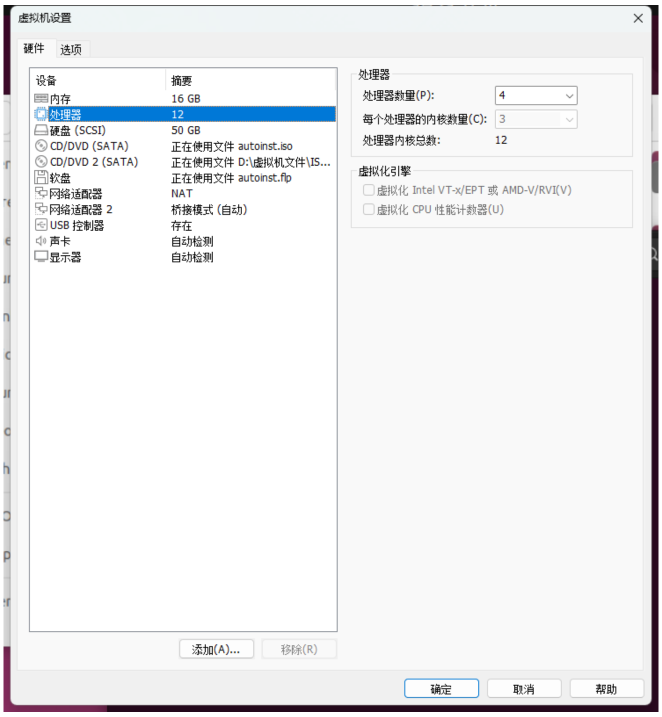
一、创建虚拟机



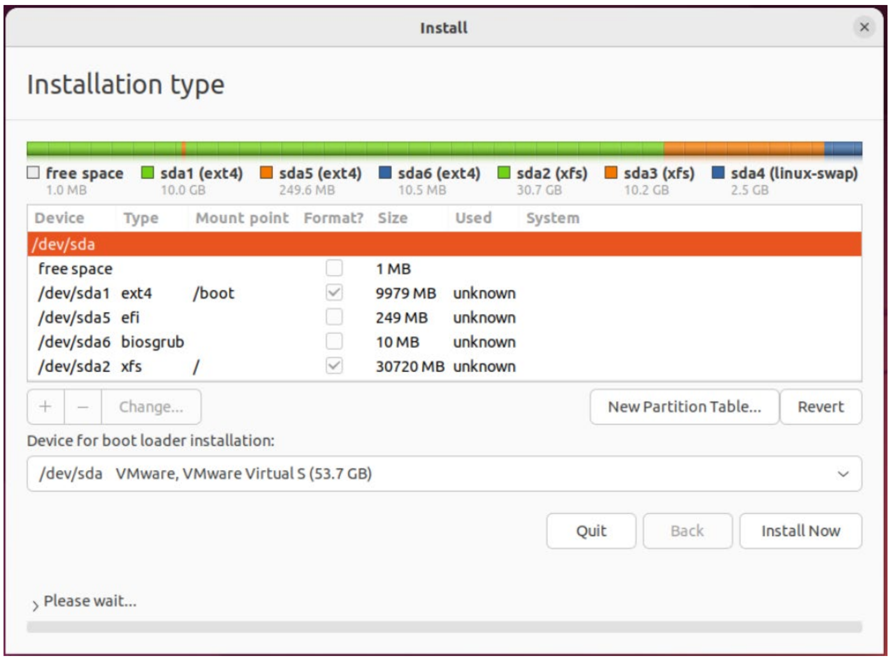
虚拟机设置

硬件 选项

设备	摘要
内存	16 GB
处理器	12
硬盘 (SCSI)	50 GB
CD/DVD (SATA)	正在使用文件 autoinst.iso
CD/DVD 2 (SATA)	正在使用文件 D:\虚拟机文件\IS...
软盘	正在使用文件 autoinst.flp
网络适配器	NAT
网络适配器 2	桥接模式 (自动)
USB 控制器	存在
声卡	自动检测
显示器	自动检测

处理器
处理器数量(P): 4
每个处理器的内核数量(C): 3
处理器内核总数: 12

虚拟化引擎
☐ 虚拟化 Intel VT-x/EPT 或 AMD-V/RVI(V)
☐ 虚拟化 CPU 性能计数器(U)



Install

Installation type

☐ free space 1.0 MB | 🟩 sda1 (ext4) 10.0 GB | 🟧 sda5 (ext4) 249.6 MB | 🟦 sda6 (ext4) 10.5 MB | 🟩 sda2 (xfs) 30.7 GB | 🟧 sda3 (xfs) 10.2 GB | 🟦 sda4 (linux-swap) 2.5 GB

Device	Type	Mount point	Format?	Size	Used	System
/dev/sda						
free space			☐	1 MB		
/dev/sda1	ext4	/boot	✓	9979 MB	unknown	
/dev/sda5	efi		☐	249 MB	unknown	
/dev/sda6	biosgrub		☐	10 MB	unknown	
/dev/sda2	xfs	/	✓	30720 MB	unknown	

Device for boot loader installation:
/dev/sda VMware, VMware Virtual S (53.7 GB)

Please wait...

sda1，挂载/boot 目录，ext4，9980MB
sda2，挂载/目录，xfs，30720MB
sda3，挂载/home 目录，xfs，10240MB
sda4，挂载 swap 分区，2048MB
sda5，挂载 efi 分区，250MB
sda6，挂载 bios 分区，10MB

二、安装相关依赖

```
sudo passwd root
su
cd /
sudo apt update -y
sudo apt upgrade -y
sudo apt install -y git curl build-essential

mkdir share/
cd share/
```

挂梯子：

（后续参考：https://github.com/ningmoon/v2ray　下方步骤给出）

```
curl -Ls https://mirrors.v2raya.org/go.sh | sudo bash
wget -qO - https://apt.v2raya.org/key/public-key.asc | sudo tee
/etc/apt/trusted.gpg.d/v2raya.asc
echo "deb https://apt.v2raya.org/ v2raya main" | sudo tee
/etc/apt/sources.list.d/v2raya.list
sudo apt update
sudo apt install v2raya
sudo systemctl enable --now　v2raya.service
```

浏览器输入：http://localhost:2017

设置名字：q，密码：123456

添加订阅地址：https://sub.。。。。。。

（连接时会遇到问题：failed to start v2ray-core: geoip.dat or geosite.dat file does not exists　解决参考：https://aisikao.ren/22633/　下方步骤给出）

```
wget https://github.com/v2fly/v2ray-core/releases/latest/download/v2ray-linux-64.zip
unzip v2ray-linux-64.zip -d ./v2ray
sudo mkdir -p /usr/local/share/v2ray
sudo cp ./v2ray/*dat /usr/local/share/v2ray
sudo install -Dm755 ./v2ray/v2ray /usr/local/bin/v2ray
```

连接完成

下载 NDN-DPDK：

```
git clone https://github.com/usnistgov/ndn-dpdk.git
cd ndn-dpdk/
cd docs/
sudo apt install --no-install-recommends ca-certificates curl gpg jq lsb-release -y
sudo SKIPROOTCHECK=1 ./ndndpdk-depends.sh
cd ..
sudo corepack pnpm install
sudo NDNDPDK_MK_RELEASE=1 make
sudo ./mk/install.sh
```

三、流量生成器

绑定网卡：

先创建一个桥接模式网卡（安装虚拟机时已添加则忽略），

然后进入虚拟机文件所在目录，用记事本打开 vmx 文件

找到 ethernet1.virtualDev ="e1000"，改为 ethernet1.virtualDev = "vmxnet3"

更改 IOMMU 设置：

sudo apt install net-tools -y

sudo apt install vim -y

vim /etc/default/grub

修改 GRUB_CMDLINE_LINUX 行的内容：

default hugepages=2048 hugepagesz=2M iommu=pt intel_iommu=on （这里我是 amd 处理器，或 intel）

这里打开了 iommu，巨页大小为 2M，数量为 2048，总共 4G

sudo update-grub

sudo reboot

加载 uio 驱动：

cd /share

git clone https://dpdk.org/git/dpdk-kmods

cd dpdk-kmods/linux/igb_uio

make clean all

sudo install -d -m0755 /lib/modules/$(uname -r)/kernel/drivers/uio

sudo install -m0644 igb_uio.ko /lib/modules/$(uname -r)/kernel/drivers/uio

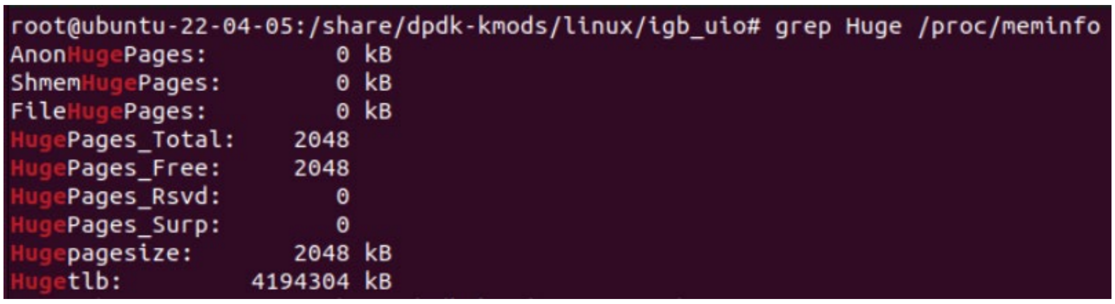
sudo depmod

sudo modprobe igb_uio（若后续重启计算机，还需重新绑定驱动，从此步开始）

检查是否加载成功：lsmod | grep igb_uio

顺便看下配置的巨页信息是否成功：grep Huge /proc/meminfo（total 要有显示数量）

（sudo /usr/local/bin/dpdk-hugepages.py --show 也可查看）

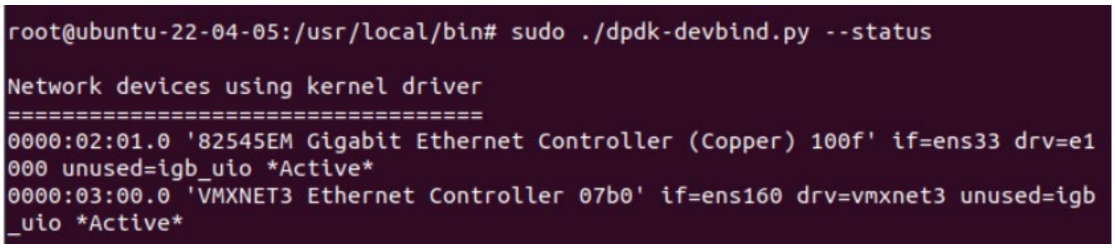


```
root@ubuntu-22-04-05:/share/dpdk-kmods/linux/igb_uio# grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:    2048
HugePages_Free:     2048
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:         4194304 kB
```

绑定

cd /usr/local/bin/

查看网卡 PCI 号：sudo ./dpdk-devbind.py --status



```
root@ubuntu-22-04-05:/usr/local/bin# sudo ./dpdk-devbind.py --status

Network devices using kernel driver
===================================
0000:02:01.0 '82545EM Gigabit Ethernet Controller (Copper) 100f' if=ens33 drv=e1
000 unused=igb_uio *Active*
0000:03:00.0 'VMXNET3 Ethernet Controller 07b0' if=ens160 drv=vmxnet3 unused=igb
_uio *Active*
```

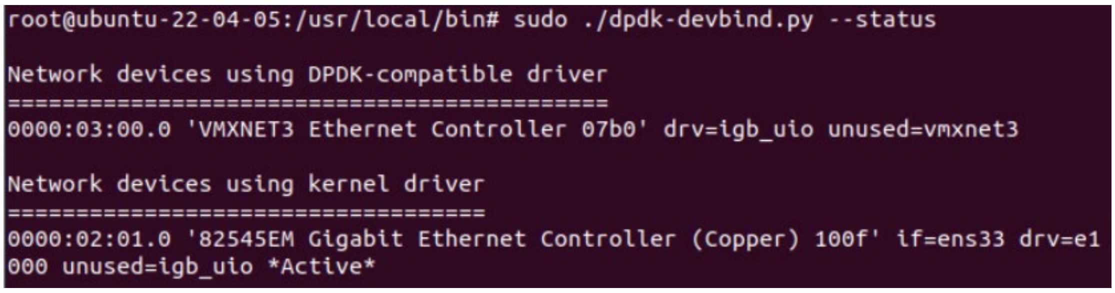
查看网卡：ifconfig

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.234.136  netmask 255.255.255.0  broadcast
192.168.234.255
        inet6 fe80::5e3:68fa:997d:1339  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:77:01:48  txqueuelen 1000  (Ethernet)
        RX packets 1747  bytes 551367 (551.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1059  bytes 115415 (115.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.116  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::7069:b279:59b2:8fb3  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:77:01:52  txqueuelen 1000  (Ethernet)
        RX packets 2052  bytes 223395 (223.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 320  bytes 33538 (33.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4967  bytes 681864 (681.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4967  bytes 681864 (681.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

这里先 down 掉 ens160 网卡：sudo ip link set enp4s0 down
ifconfig 发现 ens160 消失
解绑内核驱动 vmxnet3：sudo dpdk-devbind.py --unbind 0000:04:00.0
绑定 uio 驱动：sudo dpdk-devbind.py -b igb_uio 0000:04:00.0
再次查看绑定状态：sudo dpdk-devbind.py --status



```
root@ubuntu-22-04-05:/usr/local/bin# sudo ./dpdk-devbind.py --status

Network devices using DPDK-compatible driver
============================================
0000:03:00.0 'VMXNET3 Ethernet Controller 07b0' drv=igb_uio unused=vmxnet3

Network devices using kernel driver
===================================
0000:02:01.0 '82545EM Gigabit Ethernet Controller (Copper) 100f' if=ens33 drv=e1
000 unused=igb_uio *Active*
```

注：以上步骤在创建其它角色时都需要设置

新建终端输入：ndndpdk-svc　启动 ndn-dpdk 服务，后续出现问题会在此终端显示日志。（关闭服务：sudo killall ndndpdk-svc）

命令行方式激活流量生成器如下：
（还有一种 GraphQL 方式，登录 http://127.0.0.1:3030）
目录/user/local/share/ndn-dpdk/下，参照 trafficgen.schema.json 文件，在这个目录下新建一个 json 文件 trafficgen.json
cd /usr/local/share/ndn-dpdk
vim trafficgen.json

内容如下：

```
{
  "eal": {
    "cores": [ 0, 1, 2, 3, 4, 5, 6, 7 , 8, 9, 10, 11 ],
    "memChannels": 4,
    "disablePCI": false,
    "filePrefix": "ndn",
    "iovaMode": "PA"
  },
  "lcoreAlloc": {
    "HRLOG": [ 2 ],
    "PDUMP": [ 3 ]
  },
  "mempool": {
    "DATA": {
      "capacity": 8192,
      "dataroom": 2176
    },
    "INTEREST": {
      "capacity": 8192,
      "dataroom": 2176
    }
  },
  "socketFace": {
    "rxConns": {
            "ringCapacity": 4096
        },
        "txSyscall": {
            "disabled": false
        }
    }
}
```

解释：

1.eal

cores:指定可用的 CPU 核心，这里是核心 0-11。

memChannels：内存通道数，这里是 4，影响内存的访问效率。

disablePCI：如果为 true，则禁用 PCI 设备。

filePrefix：设置 DPDK 运行时创建的文件前缀。

iovaMode：设置了 IOVA 模式为物理地址（PA, Physical Address）。

2.lcoreAlloc

HRLOG：为角色 HRLOG 分配的逻辑核心，只有核心 2。

PDUMP：为角色 PDUMP 分配的逻辑核心，只有核心 3。

3.mempool

DATA 和 INTEREST：定义了数据包池的容量和数据区域大小，均为 8192 和 2176。

4.socketFace

rxConns：接收连接的环形缓冲区容量，设置为 4096。

txSyscall：指示系统调用的发送是否禁用。

执行 trafficgen.json 文件来激活流量生成器角色

```
cat trafficgen.json | ndndpdk-ctrl activate-trafficgen
```

显示 true 即为激活成功

对绑定 PCI 驱动的以太网适配器创建端口：

```
ndndpdk-ctrl create-eth-port --pci 03:00.0 --mtu 1500 --rx-flow 16
```

（--rx-flow 选项是因为具有 rte_flow API 功能，取决于 NIC 是否支持，指定的队列数是可以在以太网端口上创建的最大 face 数，如果不支持则不加此选项）

（ndndpdk-ctrl list-ethdev：列出创建的端口和其 mac 地址）

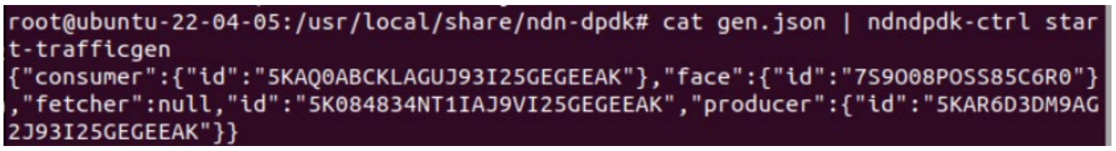
```
vim gen.json
```

内容如下：

```
{
    "face": {
        "scheme": "ether",
        "local": "00:0c:29:77:01:52",
        "remote": "00:0c:29:a1:cb:21",
        "mtu": 1500,
        "nRxQueues": 1,
        "outputQueueSize": 1024
    },
    "producer": {
        "patterns": [
            {
                "prefix": "/example/data",
                "replies": [
                    {
                        "payloadLen": 1024,
                        "weight": 9
                    },
                    {
                        "payloadLen": 512,
                        "weight": 1
                    }
                ]
            }
        ]
    },
    "consumer": {
        "interval": 1000000,
        "patterns": [
            {
                "prefix": "/example/data",
                "interestLifetime": 4000,
                "weight": 2
            },
            {
                "prefix": "/example/info",
                "interestLifetime": 2000,
                "weight": 3
            }
        ]
    }
}
```

执行 gen.json 文件来启动流量生成器角色

cat gen.json | ndndpdk-ctrl start-trafficgen



```
root@ubuntu-22-04-05:/usr/local/share/ndn-dpdk# cat gen.json | ndndpdk-ctrl star
t-trafficgen
{"consumer":{"id":"5KAQ0ABCKLAGUJ93I25GEGEEAK"},"face":{"id":"7S9O08POSS85C6R0"}
,"fetcher":null,"id":"5K084834NT1IAJ9VI25GEGEEAK","producer":{"id":"5KAR6D3DM9AG
2J93I25GEGEEAK"}}
```

查看 face 列表：ndndpdk-ctrl list-face，复制自己的流量生成器 ID

ndndpdk-ctrl get-face --id ECH9TQB1B3C8PA3E --cnt | jq .counters

四、转发器

```
ndndpdk-svc
cd /usr/local/share/ndn-dpdk
vim forwarder.json

{
   "eal": {
      "coresPerNuma": {
         "0": 2
      },
      "lcoresPerNuma": {
         "0": 6
      },
      "lcoreMain": 3
   },
   "lcoreAlloc": {
      "RX": { "0": 1 },
      "TX": { "0": 1 },
      "FWD": { "0": 2 },
      "CRYPTO": { "0": 0 }
   },
   "mempool": {
      "DIRECT": {
         "capacity": 24287,
         "dataroom": 9146
      },
      "INDIRECT": {
         "capacity": 24287
      }
   },
   "fib": {
      "capacity": 4095,
      "startDepth": 8
   },
   "pcct": {
      "pcctCapacity": 65535,
      "csMemoryCapacity": 20000,
      "csIndirectCapacity": 20000
   }
}

cat forwarder.json | ndndpdk-ctrl activate-forwarder
ndndpdk-ctrl create-eth-port --pci 03:00.0 --mtu 1500
ndndpdk-ctrl create-ether-face --local 00:0c:29:77:01:52 --remote 00:0c:29:a1:cb:21
```

```
ndndpdk-ctrl insert-fib --name /example/P --nh 286d21ff
```