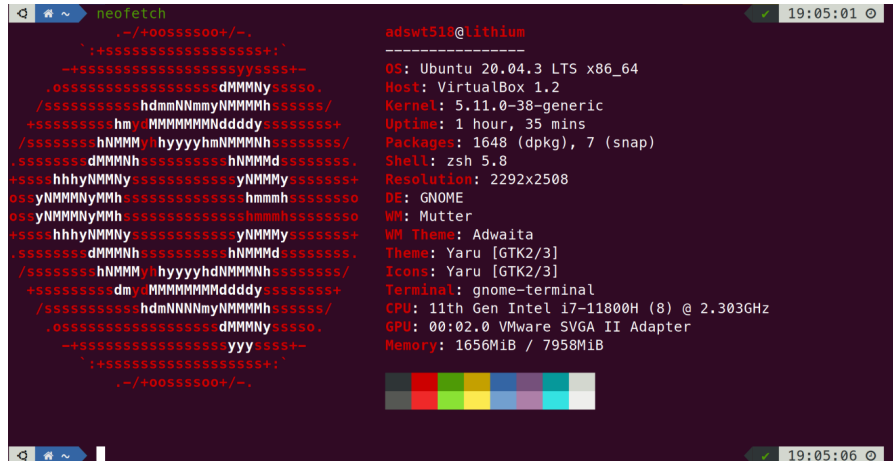


# EI313 Lab2

唐亚周 519021910804

## 1 Download QEMU and compile.

宿主为VirtualBox中的Ubuntu 20.04 LTS，具体设备信息如下：

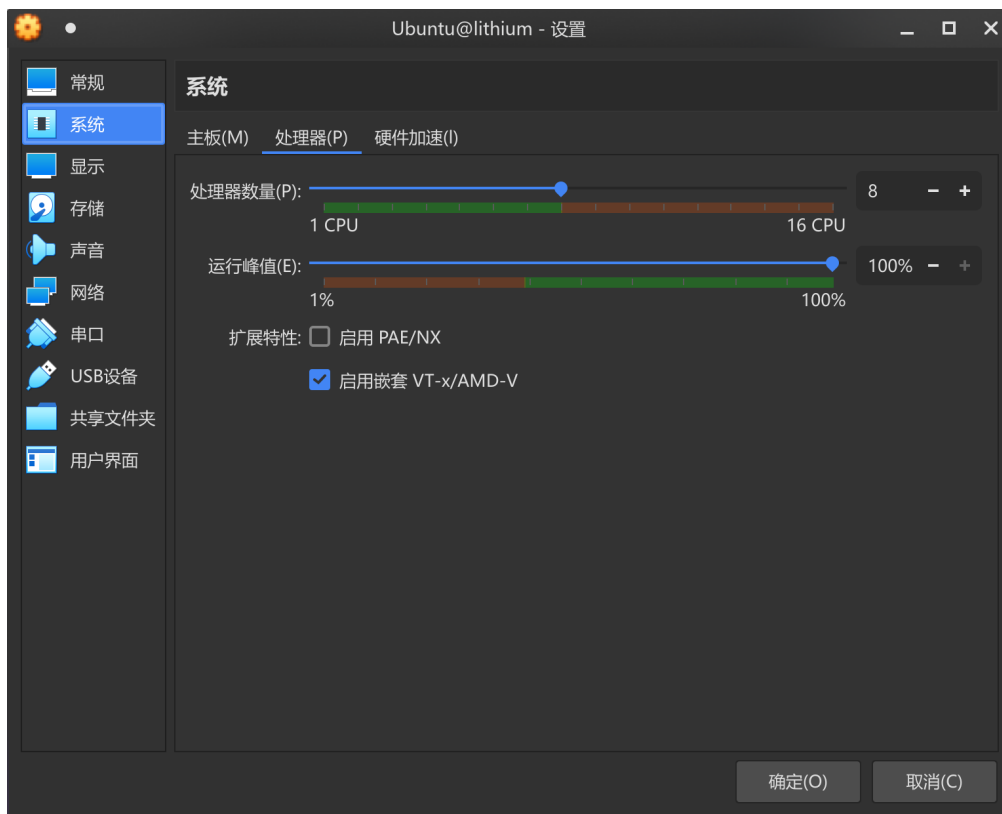


### 1.1 首先设置硬件支持虚拟化<sup>1</sup>

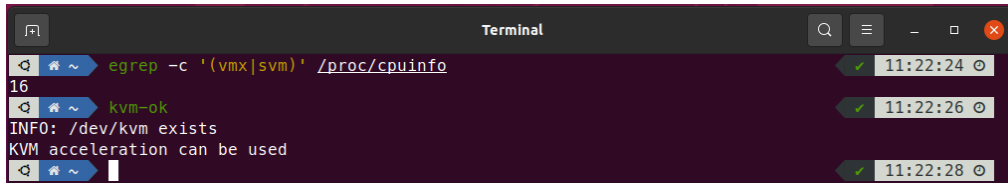
首先打开VirtualBox的嵌套虚拟化。在宿主机中输入

<sup>1</sup> `vboxmanage modifyvm "Ubuntu@lithium" --nested-hw-virt on`

可以看到嵌套VT-x/AMD-V已打开。



然后在VirtualBox虚拟机中测试

A terminal window titled "Terminal" with a dark background. It shows three commands being executed: 1. `egrep -c '(vmx|svm)' /proc/cpuinfo` which returns `16`. 2. `kvm-ok` which outputs `INFO: /dev/kvm exists` and `KVM acceleration can be used`. The right side of the terminal shows timestamps: 11:22:24, 11:22:26, and 11:22:28.

输出结果说明虚拟机的CPU支持虚拟化。

## 1.2 下载并编译安装QEMU 5.2.0

1. 首先安装各种依赖库。<sup>2</sup>

```
1 sudo apt-get install git libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev
2 sudo apt-get install git-email
3 sudo apt-get install libaio-dev libbluetooth-dev libbrlapi-dev libbz2-dev
4 sudo apt-get install libcap-dev libcap-ng-dev libcurl4-gnutls-dev libgtk-3-dev
5 sudo apt-get install libibverbs-dev libjpeg8-dev libncurses5-dev libnuma-dev
6 sudo apt-get install librbd-dev librdmacm-dev
7 sudo apt-get install libsasl2-dev libssl1.2-dev libseccomp-dev libsnappy-dev libssh2-
  1-dev
8 sudo apt-get install libvde-dev libvdeplug-dev libvte-2.90-dev libxen-dev liblzo2-dev
9 sudo apt-get install valgrind xfslibs-dev
10 sudo apt-get install libnfs-dev libiscsi-dev
```

2. 然后直接从官网下载源码并解压。

```
1 wget https://download.qemu.org/qemu-5.2.0.tar.xz
2 tar xvjf qemu-5.2.0.tar.xz
```

3. 配置并编译。注意参数 `--target-list=x86_64-softmmu`，代表编译目标为x86\_64架构CPU的QEMU，这样可以减少编译所需时间。<sup>3</sup>

```
1 cd qemu-5.2.0
2 ./configure --enable-kvm --target-list=x86_64-softmmu --enable-debug
3 make -j8
```

开始编译：

```
Terminal

parallels support: YES
sheepdog support: NO
capstone: internal
libpmem support: NO
libdaxctl support: NO
libudev: NO
default devices: YES
plugin support: NO
fuzzing support: NO
gdb: /usr/bin/gdb
thread sanitizer: NO
rng-none: NO
Linux keyring: YES

Found ninja-1.10.0 at /usr/bin/ninja
~/To/qemu-5.2.0 make -j8
changing dir to build for make ""...
make[1]: Entering directory '/home/adswt518/Tools/qemu-5.2.0/build'
/usr/bin/ninja build.ninja && touch build.ninja.stamp
ninja: no work to do.
/usr/bin/python3 -B /home/adswt518/Tools/qemu-5.2.0/meson/meson.py introspect --targets --tests --be
nchmarks | /usr/bin/python3 -B scripts/mtest2make.py > Makefile.mtest
AS      multiboot.o
AS      linuxboot.o
CC      linuxboot_dma.o
AS      kvmvapic.o
AS      pvh.o
CC      pvh_main.o
BUILD   multiboot.img
BUILD   linuxboot.img
```

编译完成:

```
Terminal

[2320/2342] Compiling C object tests/qtest/qos-test.p/pci-test.c.o
[2321/2342] Compiling C object tests/qtest/qos-test.p/ds1338-test.c.o
[2322/2342] Compiling C object tests/qtest/qos-test.p/sdhci-test.c.o
[2323/2342] Compiling C object tests/qtest/qos-test.p/virtio-test.c.o
[2324/2342] Compiling C object tests/qtest/qos-test.p/virtio-net-test.c.o
[2325/2342] Compiling C object tests/qtest/qos-test.p/eeepro100-test.c.o
[2326/2342] Compiling C object tests/qtest/qos-test.p/virtio-9p-test.c.o
[2327/2342] Compiling C object tests/qtest/qos-test.p/pcnet-test.c.o
[2328/2342] Compiling C object tests/qtest/qos-test.p/tulip-test.c.o
[2329/2342] Compiling C object tests/qtest/qos-test.p/es1370-test.c.o
[2330/2342] Compiling C object tests/qtest/qos-test.p/virtio-scsi-test.c.o
[2331/2342] Compiling C object tests/qtest/qos-test.p/virtio-rng-test.c.o
[2332/2342] Compiling C object tests/qtest/qos-test.p/vmxnet3-test.c.o
[2333/2342] Compiling C object tests/qtest/qos-test.p/usb-hcd-ohci-test.c.o
[2334/2342] Compiling C object tests/qtest/qos-test.p/e1000e-test.c.o
[2335/2342] Compiling C object tests/qtest/qos-test.p/virtio-blk-test.c.o
[2336/2342] Compiling C object tests/qtest/qos-test.p/e1000-test.c.o
[2337/2342] Compiling C object tests/qtest/qos-test.p/pca9552-test.c.o
[2338/2342] Compiling C object tests/qtest/qos-test.p/nvme-test.c.o
[2339/2342] Compiling C object tests/qtest/qos-test.p/megasas-test.c.o
[2340/2342] Compiling C object tests/qtest/qos-test.p/vhost-user-test.c.o
[2341/2342] Compiling C object tests/qtest/qos-test.p/tmp105-test.c.o
[2342/2342] Linking target tests/qtest/qos-test
make[1]: Leaving directory '/home/adswt518/Tools/qemu-5.2.0/build'
changing dir to build for make ""...
make[1]: Entering directory '/home/adswt518/Tools/qemu-5.2.0/build'
[1/49] Generating QAPI test (include) with a custom command
[2/18] Generating qemu-version.h with a meson_exe.py custom command
make[1]: Leaving directory '/home/adswt518/Tools/qemu-5.2.0/build'
~/To/qemu-5.2.0
```

然后安装:

```
~/Tools/qemu-5.2.0 sudo make install
changing dir to build for make "install"...
make[1]: Entering directory '/home/adswt518/Tools/qemu-5.2.0/build'
[1/18] Generating qemu-version.h with a meson_exe.py custom command
[1/2] Installing files.
Installing subdir /home/adswt518/Tools/qemu-5.2.0/qga/run to /usr/local/var/run
Installing trace/trace-events-all to /usr/local/share/qemu
Installing fsdev/virtfs-proxy-helper to /usr/local/libexec
Installing qemu-system-x86_64 to /usr/local/bin
Installing qga/qemu-ga to /usr/local/bin
Installing qemu-keymap to /usr/local/bin
Installing qemu-img to /usr/local/bin
Installing qemu-io to /usr/local/bin
Installing qemu-nbd to /usr/local/bin
Installing storage-daemon/qemu-storage-daemon to /usr/local/bin
Installing contrib/elf2dmp/elf2dmp to /usr/local/bin
Installing qemu-edid to /usr/local/bin
Installing qemu-bridge-helper to /usr/local/libexec
```

检测QEMU版本:

```
~/To/qemu-5.2.0/build ./qemu-img -V
qemu-img version 5.2.0
Copyright (c) 2003-2020 Fabrice Bellard and the QEMU Project developers
~/To/qemu-5.2.0/build
```

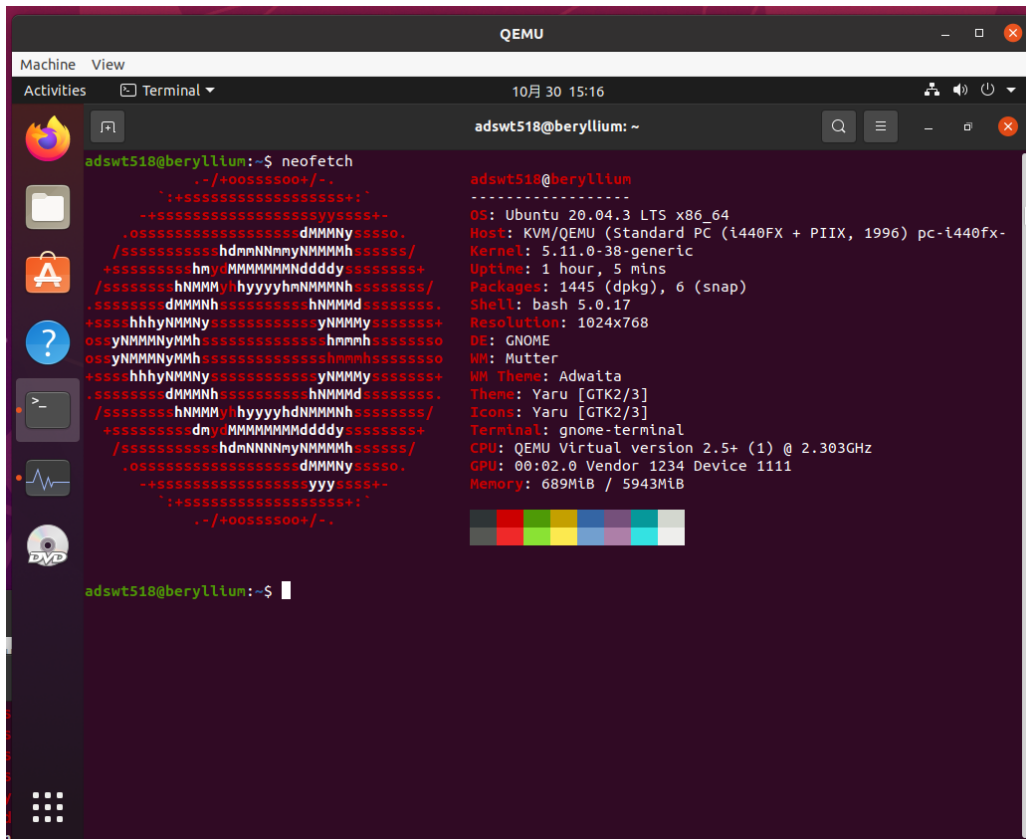
## 2 Create 2 VMs with TAP mode network (e1000 and virtio-net) by QEMU.

### 2.1 创建并安装QEMU虚拟机<sup>3</sup>

这里我同样选择Ubuntu 20.04 LTS作为QEMU虚拟机的系统。

```
~/VM $ qemu-img create -f qcow2 ubuntu.qcow2 15G
Formatting 'ubuntu.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=16106127360 lazy_refcounts=off refcount_bits=16
~/VM $ qemu-system-x86_64 -m 4G -drive format=qcow2,file=ubuntu.qcow2 -cd
rom ~/Downloads/ubuntu-20.04.3-desktop-amd64.iso
```

安装完毕，系统信息如图所示。



### 2.2 设置QEMU虚拟机的网络

#### 2.2.1 NAT配置<sup>4</sup>

首先安装必要的库

```
1 | sudo apt install bridge-utils iptables dnsmasq
```

然后在 `/usr/local/etc` 下创建文件 `qemu-ifup`，内容如下

```
1 | #!/bin/sh
2 | #
3 | # Copyright IBM, Corp. 2010
4 | #
5 | # Authors:
6 | # Anthony Liguori <aliguori@us.ibm.com>
7 | #
```

```
8 # This work is licensed under the terms of the GNU GPL, version 2. See
9 # the COPYING file in the top-level directory.
10
11 # Set to the name of your bridge
12 BRIDGE=br0
13
14 # Network information
15 NETWORK=192.168.53.0
16 NETMASK=255.255.255.0
17 GATEWAY=192.168.53.1
18 DHCP RANGE=192.168.53.2,192.168.53.254
19
20 # Optionally parameters to enable PXE support
21 TFTPROOT=
22 BOOTP=
23
24 do_brctl() {
25     brctl "$@"
26 }
27
28 do_ifconfig() {
29     ifconfig "$@"
30 }
31
32 do_dd() {
33     dd "$@"
34 }
35
36 do_iptables_restore() {
37     iptables-restore "$@"
38 }
39
40 do_dnsmasq() {
41     dnsmasq "$@"
42 }
43
44 check_bridge() {
45     if do_brctl show | grep "^$1" > /dev/null 2> /dev/null; then
46         return 1
47     else
48         return 0
49     fi
50 }
51
52 create_bridge() {
53     do_brctl addbr "$1"
54     do_brctl stp "$1" off
55     do_brctl setfd "$1" 0
```

```

56     do_ifconfig "$1" "$GATEWAY" netmask "$NETMASK" up
57 }
58
59 enable_ip_forward() {
60     echo 1 | do_dd of=/proc/sys/net/ipv4/ip_forward > /dev/null
61 }
62
63 add_filter_rules() {
64     do_iptables_restore <<EOF
65     # Generated by iptables-save v1.3.6 on Fri Aug 24 15:20:25 2007
66     *nat
67     :PREROUTING ACCEPT [61:9671]
68     :POSTROUTING ACCEPT [121:7499]
69     :OUTPUT ACCEPT [132:8691]
70     -A POSTROUTING -s $NETWORK/$NETMASK -j MASQUERADE
71     COMMIT
72     # Completed on Fri Aug 24 15:20:25 2007
73     # Generated by iptables-save v1.3.6 on Fri Aug 24 15:20:25 2007
74     *filter
75     :INPUT ACCEPT [1453:976046]
76     :FORWARD ACCEPT [0:0]
77     :OUTPUT ACCEPT [1605:194911]
78     -A INPUT -i $BRIDGE -p tcp -m tcp --dport 67 -j ACCEPT
79     -A INPUT -i $BRIDGE -p udp -m udp --dport 67 -j ACCEPT
80     -A INPUT -i $BRIDGE -p tcp -m tcp --dport 53 -j ACCEPT
81     -A INPUT -i $BRIDGE -p udp -m udp --dport 53 -j ACCEPT
82     -A FORWARD -i $1 -o $1 -j ACCEPT
83     -A FORWARD -s $NETWORK/$NETMASK -i $BRIDGE -j ACCEPT
84     -A FORWARD -d $NETWORK/$NETMASK -o $BRIDGE -m state --state RELATED,ESTABLISHED -j ACCEPT
85     -A FORWARD -o $BRIDGE -j REJECT --reject-with icmp-port-unreachable
86     -A FORWARD -i $BRIDGE -j REJECT --reject-with icmp-port-unreachable
87     COMMIT
88     # Completed on Fri Aug 24 15:20:25 2007
89     EOF
90 }
91
92 start_dnsmasq() {
93     do_dnsmasq \
94         --strict-order \
95         --except-interface=lo \
96         --interface=$BRIDGE \
97         --listen-address=$GATEWAY \
98         --bind-interfaces \
99         --dhcp-range=$DHCP RANGE \
100         --conf-file="" \
101         --pid-file=/var/run/qemu-dnsmasq-$BRIDGE.pid \
102         --dhcp-leasefile=/var/run/qemu-dnsmasq-$BRIDGE.leases \
103         --dhcp-no-override \

```

```

104     ${TFTPROOT:+"--enable-tftp"} \
105     ${TFTPROOT:+"--tftp-root=$TFTPROOT"} \
106     ${BOOTP:+"--dhcp-boot=$BOOTP"}
107 }
108
109 setup_bridge_nat() {
110     if check_bridge "$1" ; then
111         create_bridge "$1"
112         enable_ip_forward
113         add_filter_rules "$1"
114         start_dnsmasq "$1"
115     fi
116 }
117
118 setup_bridge_vlan() {
119     if check_bridge "$1" ; then
120         create_bridge "$1"
121         start_dnsmasq "$1"
122     fi
123 }
124
125 setup_bridge_nat "$BRIDGE"
126
127 if test "$1" ; then
128     do_ifconfig "$1" 0.0.0.0 up
129     do_brctl addif "$BRIDGE" "$1"
130 fi

```

并设置其权限

```

1 | sudo chmod 755 /usr/local/etc/qemu-ifup

```

## 2.2.2 配置QEMU虚拟机网络

### 1. VM with TAP mode network e1000

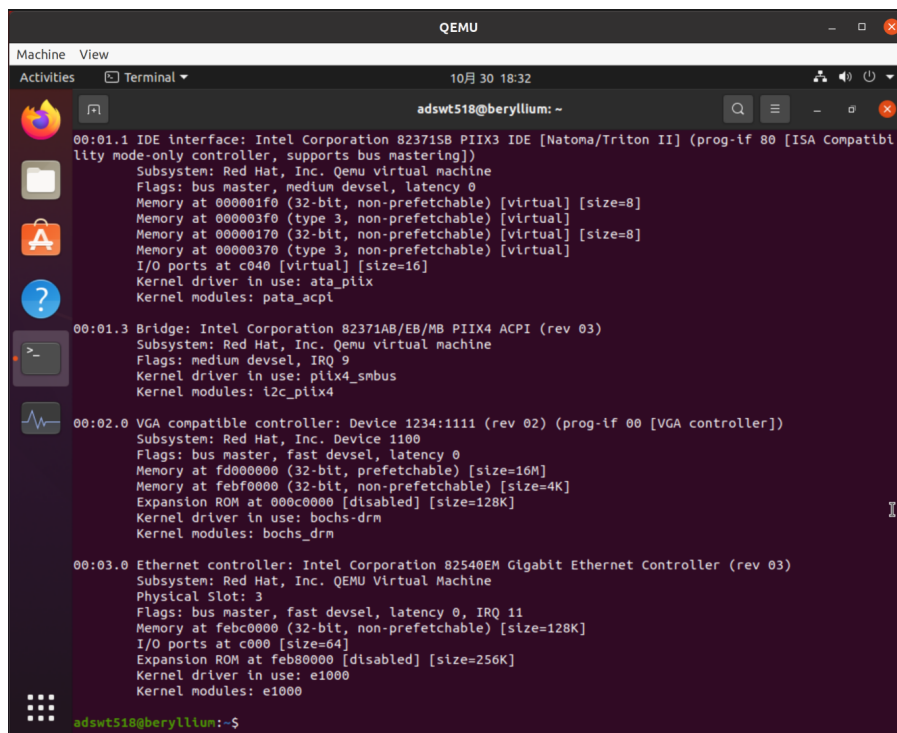
启动QEMU虚拟机：

```

1 | sudo qemu-system-x86_64 -m 4G -drive format=qcow2,file=ubuntu.qcow2 -enable-kvm -net
   nic,model=e1000 -net tap

```

在虚拟机中查看其网卡信息：



```
Machine View
Activities Terminal 10月30 18:32
adswt518@beryllium: ~
00:01.1 IDE interface: Intel Corporation 823715B PIIX3 IDE [Natoma/Triton II] (prog-if 80 [ISA compatible mode-only controller, supports bus mastering])
Subsystem: Red Hat, Inc. Qemu virtual machine
Flags: bus master, medium devsel, latency 0
Memory at 000001f0 (32-bit, non-prefetchable) [virtual] [size=8]
Memory at 000003f0 (type 3, non-prefetchable) [virtual]
Memory at 00000170 (32-bit, non-prefetchable) [virtual] [size=8]
Memory at 00000370 (type 3, non-prefetchable) [virtual]
I/O ports at c040 [virtual] [size=16]
Kernel driver in use: ata_piix
Kernel modules: pata_acpi

00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
Subsystem: Red Hat, Inc. Qemu virtual machine
Flags: medium devsel, IRQ 9
Kernel driver in use: piix4_smbus
Kernel modules: i2c_piix4

00:02.0 VGA compatible controller: Device 1234:1111 (rev 02) (prog-if 00 [VGA controller])
Subsystem: Red Hat, Inc. Device 1100
Flags: bus master, fast devsel, latency 0
Memory at fd000000 (32-bit, prefetchable) [size=16M]
Memory at febf0000 (32-bit, non-prefetchable) [size=4K]
Expansion ROM at 000c0000 [disabled] [size=128K]
Kernel driver in use: bochs-drm
Kernel modules: bochs_drm

00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)
Subsystem: Red Hat, Inc. QEMU Virtual Machine
Physical Slot: 3
Flags: bus master, fast devsel, latency 0, IRQ 11
Memory at febc0000 (32-bit, non-prefetchable) [size=128K]
I/O ports at c000 [size=64]
Expansion ROM at fe800000 [disabled] [size=256K]
Kernel driver in use: e1000
Kernel modules: e1000

adswt518@beryllium: ~$
```

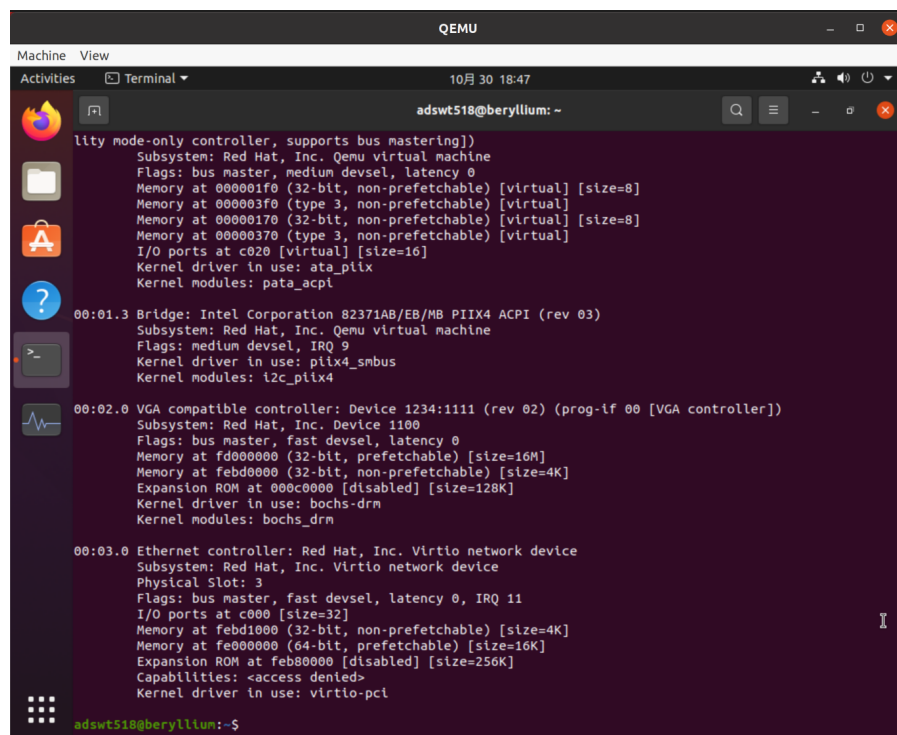
可以看到 `Kernel driver in use: e1000` , `Kernel modules: e1000` , 说明虚拟机使用e1000网卡。

## 2. VM with TAP mode network virtio-net

启动QEMU虚拟机:

- 1 `sudo qemu-system-x86_64 -m 4G -drive format=qcow2,file=ubuntu.qcow2 -enable-kvm -net nic,model=virtio-net-pci -net tap`

在虚拟机中查看其网卡信息:



```
Machine View
Activities Terminal 10月30 18:47
adswt518@beryllium: ~
lity mode-only controller, supports bus mastering])
Subsystem: Red Hat, Inc. Qemu virtual machine
Flags: bus master, medium devsel, latency 0
Memory at 000001f0 (32-bit, non-prefetchable) [virtual] [size=8]
Memory at 000003f0 (type 3, non-prefetchable) [virtual]
Memory at 00000170 (32-bit, non-prefetchable) [virtual] [size=8]
Memory at 00000370 (type 3, non-prefetchable) [virtual]
I/O ports at c020 [virtual] [size=16]
Kernel driver in use: ata_piix
Kernel modules: pata_acpi

00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
Subsystem: Red Hat, Inc. Qemu virtual machine
Flags: medium devsel, IRQ 9
Kernel driver in use: piix4_smbus
Kernel modules: i2c_piix4

00:02.0 VGA compatible controller: Device 1234:1111 (rev 02) (prog-if 00 [VGA controller])
Subsystem: Red Hat, Inc. Device 1100
Flags: bus master, fast devsel, latency 0
Memory at fd000000 (32-bit, prefetchable) [size=16M]
Memory at febd0000 (32-bit, non-prefetchable) [size=4K]
Expansion ROM at 000c0000 [disabled] [size=128K]
Kernel driver in use: bochs-drm
Kernel modules: bochs_drm

00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device
Subsystem: Red Hat, Inc. Virtio network device
Physical Slot: 3
Flags: bus master, fast devsel, latency 0, IRQ 11
I/O ports at c000 [size=32]
Memory at febd1000 (32-bit, non-prefetchable) [size=4K]
Memory at fe000000 (64-bit, prefetchable) [size=16K]
Expansion ROM at fe800000 [disabled] [size=256K]
Capabilities: <access denied>
Kernel driver in use: virtio-pci

adswt518@beryllium: ~$
```

可以看到 `Kernel driver in use: virtio-pci` , 说明虚拟机使用virtio驱动。



### 3 Connect to your VM through VNC viewer or SSH.

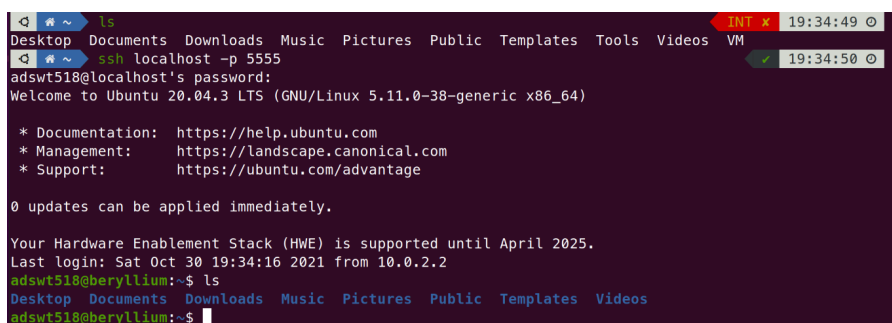
这里我使用SSH链接QEMU虚拟机。首先启动QEMU虚拟机，并设定端口。<sup>5</sup>

```
1 | sudo qemu-system-x86_64 -m 4G -drive format=qcow2,file=ubuntu.qcow2 -enable-kvm -net nic -net tap -device e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::5555-:22
```

然后在VirtualBox虚拟机（相对于QEMU虚拟机来说是宿主机）的终端中输入

```
1 | ssh localhost -p 5555
```

就可以建立宿主机与QEMU虚拟机之间的SSH连接，如图所示。



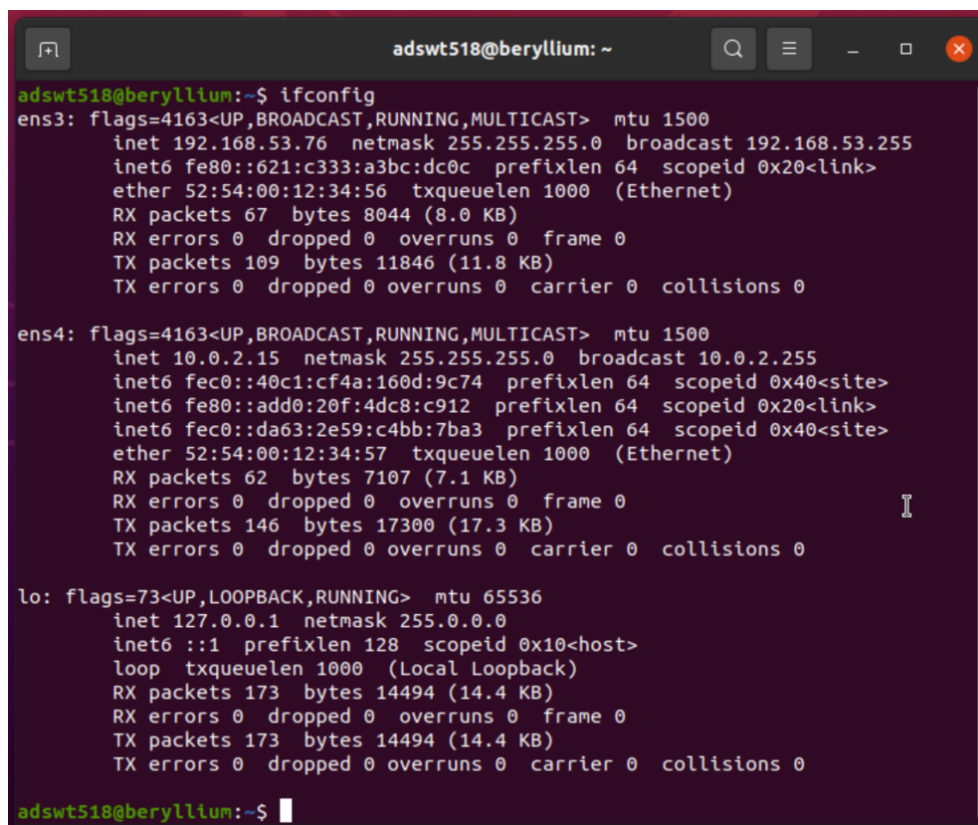
```
ls
Desktop Documents Downloads Music Pictures Public Templates Tools Videos VM
ssh localhost -p 5555
adswt518@localhost's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-38-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sat Oct 30 19:34:16 2021 from 10.0.2.2
adswt518@beryllium:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
adswt518@beryllium:~$
```

当然也可以获取QEMU虚拟机的IP地址来进行连接。使用 `ifconfig` 得到QEMU虚拟机网卡IP地址为 `192.168.53.76`。



```
adswt518@beryllium:~$ ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.53.76 netmask 255.255.255.0 broadcast 192.168.53.255
    inet6 fe80::621:c333:a3bc:dc0c prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:34:56 txqueuelen 1000 (Ethernet)
    RX packets 67 bytes 8044 (8.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 109 bytes 11846 (11.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fec0::40c1:cf4a:160d:9c74 prefixlen 64 scopeid 0x40<site>
    inet6 fe80::add0:20f:4dc8:c912 prefixlen 64 scopeid 0x20<link>
    inet6 fec0::da63:2e59:c4bb:7ba3 prefixlen 64 scopeid 0x40<site>
    ether 52:54:00:12:34:57 txqueuelen 1000 (Ethernet)
    RX packets 62 bytes 7107 (7.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 146 bytes 17300 (17.3 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 173 bytes 14494 (14.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 173 bytes 14494 (14.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

adswt518@beryllium:~$
```

然后在宿主机中

```
1 | ssh adswt518@192.168.53.76
```

就可以连接上了。

```
ssh adswt518@192.168.53.76
The authenticity of host '192.168.53.76 (192.168.53.76)' can't be established.
ECDSA key fingerprint is SHA256:zhNyqFzYPaetgk9FFTAAbN:INXfgOrBmipCh+8CGzKqCo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.53.76' (ECDSA) to the list of known hosts.
adswt518@192.168.53.76's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-38-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

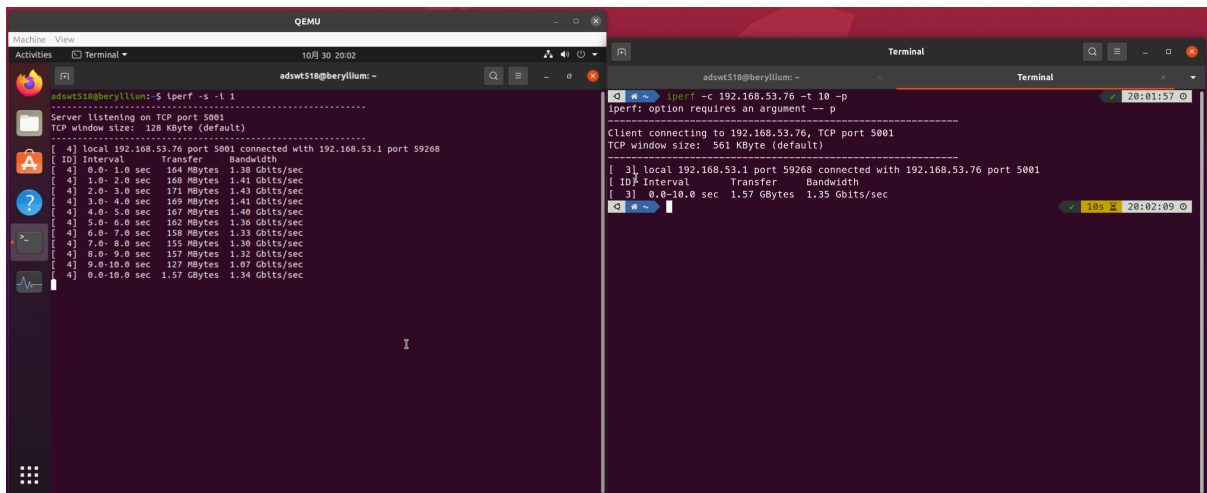
0 updates can be applied immediately.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sat Oct 30 19:35:07 2021 from 10.0.2.2
adswt518@beryllium:~$ logout
Connection to 192.168.53.76 closed.
```

## 4 Compare the network (e1000 and virtio-net) performance of your host machine and VMs.

这里我使用iperf进行测试宿主机和QEMU虚拟机之间的网络表现。在QEMU虚拟机的终端中输入 `iperf -s -i 1`，在宿主机的终端中输入 `iperf -c 192.168.53.76 -t 10 -d`。

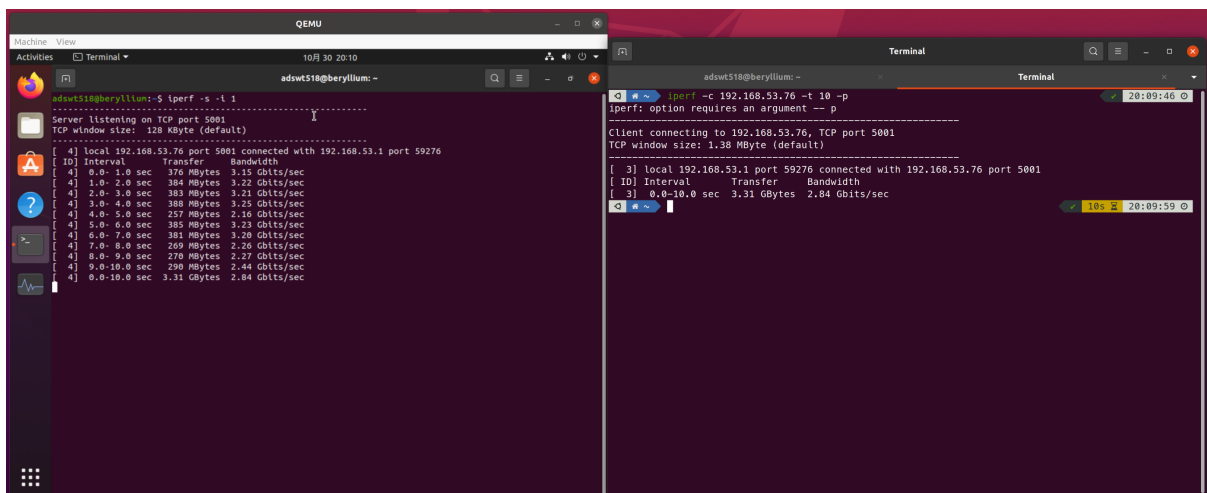
### 1. e1000下，平均带宽为1.35GB/s。



```
adswt518@beryllium:~$ iperf -s -i 1
Server listening on TCP port 5001
TCP window size: 128 MByte (default)
[ ID] Interval      Transfer     Bandwidth
[ 0] 0.0-1.0 sec   164 MBytes  1.38 Gbits/sec
[ 1] 1.0-2.0 sec   168 MBytes  1.41 Gbits/sec
[ 2] 2.0-3.0 sec   171 MBytes  1.43 Gbits/sec
[ 3] 3.0-4.0 sec   169 MBytes  1.41 Gbits/sec
[ 4] 4.0-5.0 sec   167 MBytes  1.40 Gbits/sec
[ 5] 5.0-6.0 sec   162 MBytes  1.36 Gbits/sec
[ 6] 6.0-7.0 sec   158 MBytes  1.33 Gbits/sec
[ 7] 7.0-8.0 sec   155 MBytes  1.30 Gbits/sec
[ 8] 8.0-9.0 sec   157 MBytes  1.32 Gbits/sec
[ 9] 9.0-10.0 sec 127 MBytes  1.07 Gbits/sec
[10] 0.0-10.0 sec  1.57 Gbytes  1.34 Gbits/sec

adswt518@beryllium:~$ iperf -c 192.168.53.76 -t 10 -p
iperf: option requires an argument -- p
Client connecting to 192.168.53.76, TCP port 5001
TCP window size: 561 KByte (default)
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0-10.0 sec  1.57 Gbytes  1.35 Gbits/sec
```

### 2. virtio-net下，平均带宽为2.84GB/s。



```
adswt518@beryllium:~$ iperf -s -i 1
Server listening on TCP port 5001
TCP window size: 128 MByte (default)
[ ID] Interval      Transfer     Bandwidth
[ 0] 0.0-1.0 sec   376 MBytes  3.15 Gbits/sec
[ 1] 1.0-2.0 sec   384 MBytes  3.22 Gbits/sec
[ 2] 2.0-3.0 sec   383 MBytes  3.21 Gbits/sec
[ 3] 3.0-4.0 sec   388 MBytes  3.25 Gbits/sec
[ 4] 4.0-5.0 sec   257 MBytes  2.10 Gbits/sec
[ 5] 5.0-6.0 sec   385 MBytes  3.23 Gbits/sec
[ 6] 6.0-7.0 sec   381 MBytes  3.20 Gbits/sec
[ 7] 7.0-8.0 sec   269 MBytes  2.26 Gbits/sec
[ 8] 8.0-9.0 sec   270 MBytes  2.27 Gbits/sec
[ 9] 9.0-10.0 sec  290 MBytes  2.44 Gbits/sec
[10] 0.0-10.0 sec  3.31 Gbytes  2.84 Gbits/sec

adswt518@beryllium:~$ iperf -c 192.168.53.76 -t 10 -p
iperf: option requires an argument -- p
Client connecting to 192.168.53.76, TCP port 5001
TCP window size: 1.38 MByte (default)
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0-10.0 sec  3.31 Gbytes  2.84 Gbits/sec
```

可以明显看出，与使用e1000时相比，使用virtio时，QEMU虚拟机的网络性能更好。

## 5 总结和反思

本次实验中，我首先尝试在物理机的Linux系统中直接安装QEMU并创建虚拟机，但由于我的笔记本电脑只有无线网卡，在设置无线网卡时遇到了一些困难，遂放弃，采用了VirtualBox+QEMU嵌套虚拟化的方法。

本次实验让我对QEMU有了初步的认识，也提高了解决问题的能力。

## 6 致谢

感谢我的同学陈浩南和刘梓睿对我的帮助，我在与他们的交流中，解决了很多问题。

- 
1. 在VirtualBox 6.1里面打开嵌套 VT-x/AMD-V 功能\_holderlinzhang的博客-CSDN博客 <https://blog.csdn.net/holderlinzhang/article/details/104260531> ↩
  2. Hosts\_Linux - QEMU <https://wiki.qemu.org/Hosts/Linux> ↩
  3. 从源码编译安装QEMU以及如何创建QEMU虚拟机\_Haifeng的博客-CSDN博客源码安装qemu [https://blog.csdn.net/haifeng\\_gu/article/details/108055083](https://blog.csdn.net/haifeng_gu/article/details/108055083) ↩ ↩
  4. Documentation\_Networking\_NAT - QEMU <https://wiki.qemu.org/Documentation/Networking/NAT> ↩
  5. Documentation\_Networking - QEMU <https://wiki.qemu.org/Documentation/Networking> ↩