

虚拟化和裸机网络性能测试

——工程实践与科技创新 作业 2

管仁阳-519021911058

实验目的

1. 学会使用 qemu 配置并搭建虚拟机
2. 学会使用 ssh 与 vnc viewer 连接服务器（虚拟机）
3. 比较 virtio-net 与 e1000 各自与主机之间的通信速度

实验环境

1. 主机系统：Ubuntu 20.04
2. 虚拟机系统：Ubuntu 20.04
3. qemu 版本：5.2.0

实验内容

本次实验经过了多次失败，过程中也经过多次更改底层配置文件、几次虚拟机崩溃，且最初的探索过程混乱无法截图。**因此，最后我按照我总结出来的正确步骤，完整地重做了一遍实验，并记录在本部分。**

任务一：下载 qemu 并编译

一、安装必要的依赖库

首先，使用以下命令安装实验中必要的依赖库

```
1 sudo apt-get install build-essential pkg-config zlib1g-dev
2 sudo apt-get install libglib2.0-0 libglib2.0-dev
3 sudo apt-get install libstdc++11-dev
4 sudo apt-get install libpixman-1-dev libfdt-dev
5 sudo apt-get install autoconf automake libtool
6 sudo apt-get install librbdev
7 sudo apt-get install libaio-dev
8 sudo apt-get install flex bison
```

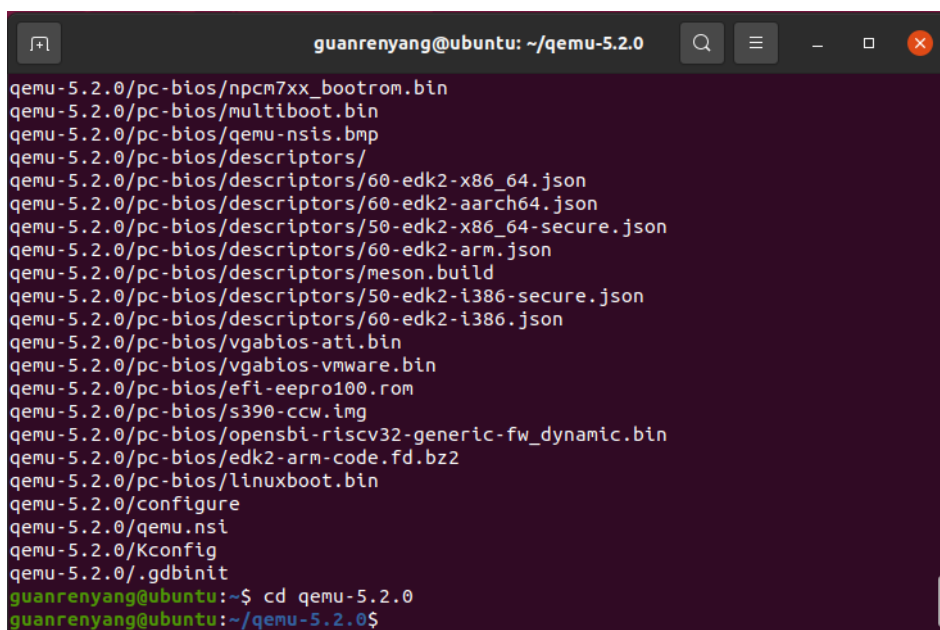
并且使用以下命令安装 SDL2。SDL 库提供对音频，键盘，鼠标，操纵杆和图形硬件的低级访问，这在 qemu 虚拟机的使用中是必要的。

```
1 sudo apt-get install libstdc++12-dev
2 sudo apt-get install libstdc++12-dev
3 sudo apt-get install libstdc++12-mixer-dev
4 sudo apt-get install libstdc++12-image-dev
5 sudo apt-get install libstdc++12-ttf-dev
6 sudo apt-get install libstdc++12-gfx-dev
```

二、下载并编译 qemu

登录 qemu 官方网站 <https://www.qemu.org/download/> 并根据其主页的指南，使用以下指令来下载 qemu、解压文件并进入解压后的目录中。

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



A terminal window titled 'guanrenyang@ubuntu: ~/qemu-5.2.0' showing the contents of the directory. The output is a long list of files and subdirectories, including 'pc-bios', 'configure', 'qemu.nsi', 'Kconfig', 'gdbinit', and various JSON descriptor files. The prompt shows the user has navigated to the directory.

```
guanrenyang@ubuntu: ~/qemu-5.2.0
qemu-5.2.0/pc-bios/npcm7xx_bootrom.bin
qemu-5.2.0/pc-bios/multiboot.bin
qemu-5.2.0/pc-bios/qemu-nvme.spc
qemu-5.2.0/pc-bios/descriptors/
qemu-5.2.0/pc-bios/descriptors/60-edk2-x86_64.json
qemu-5.2.0/pc-bios/descriptors/60-edk2-aarch64.json
qemu-5.2.0/pc-bios/descriptors/50-edk2-x86_64-secure.json
qemu-5.2.0/pc-bios/descriptors/60-edk2-arm.json
qemu-5.2.0/pc-bios/descriptors/meson.build
qemu-5.2.0/pc-bios/descriptors/50-edk2-i386-secure.json
qemu-5.2.0/pc-bios/descriptors/60-edk2-i386.json
qemu-5.2.0/pc-bios/vgabios-atl.bin
qemu-5.2.0/pc-bios/vgabios-vmware.bin
qemu-5.2.0/pc-bios/efi-efpro100.rom
qemu-5.2.0/pc-bios/s390-ccw.img
qemu-5.2.0/pc-bios/opensbi-riscv32-generic-fw_dynamic.bin
qemu-5.2.0/pc-bios/edk2-arm-code.fd.bz2
qemu-5.2.0/pc-bios/linuxboot.bin
qemu-5.2.0/configure
qemu-5.2.0/qemu.nsi
qemu-5.2.0/Kconfig
qemu-5.2.0/gdbinit
guanrenyang@ubuntu:~$ cd qemu-5.2.0
guanrenyang@ubuntu:~/qemu-5.2.0$
```

由于下载时的输出较长，只展示完成下载的截图。由图所示成功下载 qemu-5.2.0

使用以下指令进行配置 (configure)：

```
1 ./configure --target-list=x86_64-softmmu --enable-kvm --enable-debug --enable-vnc
```

其中：

--target-list 参数表示编译指定架构，如果不加这个参数默认编译所有架构。添加这个参数将加快 configure 的速度。

--enable-kvm 参数表示编译 kvm 模块，使得 qemu 可以通过 kvm 来访问虚拟哈服务。

--enable-debug 参数表示报告错误

--enable-vnc 参数表示启用 VNC

```
static build: NO
SDL support: YES
SDL image support: YES
GTK support: NO
GTK GL support: NO
Multipath support: NO
VNC support: YES
VNC SASL support: NO

gdb: /usr/bin/gdb
thread sanitizer: NO
rng-none: NO
Linux keyring: YES

Found ninja-1.10.0 at /usr/bin/ninja
guanrenyang@ubuntu:~/qemu-5.2.0$
```

输出中显示 qemu 支持 SDL 与 VNC，configure 成功。（由于输出内容较多，只展示关键输出与最后输出）

执行 make 与 make install 指令，即完成安装 qemu：

```
1 make
2 make install
```

```
guanrenyang@ubuntu: ~/qemu-5.2.0
AS      linuxboot.o
BUILD  linuxboot.img
BUILD  linuxboot.raw
SIGN   linuxboot.bin
CC      linuxboot_dma.o
BUILD  linuxboot_dma.img
BUILD  linuxboot_dma.raw
SIGN   linuxboot_dma.bin
AS      kvmvapic.o
BUILD  kvmvapic.img
BUILD  kvmvapic.raw
SIGN   kvmvapic.bin
AS      pvh.o
CC      pvh_main.o
BUILD  pvh.img
BUILD  pvh.raw
SIGN   pvh.bin
make[1]: Leaving directory '/home/guanrenyang/qemu-5.2.0/build'
changing dir to build for make ""...
make[1]: Entering directory '/home/guanrenyang/qemu-5.2.0/build'
[1/49] Generating qemu-version.h with a meson_exe.py custom command
[2/33] Generating QAPI test (include) with a custom command
make[1]: Leaving directory '/home/guanrenyang/qemu-5.2.0/build'
guanrenyang@ubuntu:~/qemu-5.2.0$
```

make 输出

```
guanrenyang@ubuntu: ~/qemu-5.2.0
Installing /home/guanrenyang/qemu-5.2.0/pc-bios/opensbi-riscv32-generic-fw_dynamic.elf to /usr/local/share/qemu
Installing /home/guanrenyang/qemu-5.2.0/pc-bios/opensbi-riscv64-generic-fw_dynamic.elf to /usr/local/share/qemu
Installing /home/guanrenyang/qemu-5.2.0/pc-bios/npc7xx_bootrom.bin to /usr/local/share/qemu
Installing /home/guanrenyang/qemu-5.2.0/build/pc-bios/descriptors/50-edk2-i386-secure.json to /usr/local/share/qemu/firmware
Installing /home/guanrenyang/qemu-5.2.0/build/pc-bios/descriptors/50-edk2-x86_64-secure.json to /usr/local/share/qemu/firmware
Installing /home/guanrenyang/qemu-5.2.0/build/pc-bios/descriptors/60-edk2-aarch64.json to /usr/local/share/qemu/firmware
Installing /home/guanrenyang/qemu-5.2.0/build/pc-bios/descriptors/60-edk2-arm.json to /usr/local/share/qemu/firmware
Installing /home/guanrenyang/qemu-5.2.0/build/pc-bios/descriptors/60-edk2-i386.json to /usr/local/share/qemu/firmware
Installing /home/guanrenyang/qemu-5.2.0/build/pc-bios/descriptors/60-edk2-x86_64.json to /usr/local/share/qemu/firmware
Installing /home/guanrenyang/qemu-5.2.0/pc-bios/keymaps/sl to /usr/local/share/qemu/keymaps
Installing /home/guanrenyang/qemu-5.2.0/pc-bios/keymaps/sv to /usr/local/share/qemu/keymaps
make[1]: Leaving directory '/home/guanrenyang/qemu-5.2.0/build'
guanrenyang@ubuntu:~/qemu-5.2.0$
```

make install 输出

由于输出较长，我们只展示最终结果。由图可见，make 和 make install 成功。

任务二：通过 qemu 创建 2 个虚拟机，带有 TAP 模式网络 (E1000 和 Viftio-net)

进入 build 文件夹下的 x86_64-softmmu 文件夹，通过 qemu-img 指令创建虚拟机镜像：

```
1 qemu-img create -f qcow2 ubuntu_1.img 20G
2 qemu-img create -f qcow2 ubuntu_2.img 20G
```

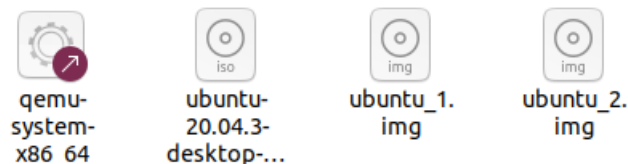
其中：

-f 用来指定镜像格式，qcow2 是最常用的镜像格式；ubuntu_1.img 和 ubuntu_2.img 是镜像名称；20G 是镜像文件的大小。

```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ qemu-img create -f qcow2 ubuntu_1.img 20G
Formatting 'ubuntu_1.img', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=21474836480 lazy_refcounts=off refcount_bits=16
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ qemu-img create -f qcow2 ubuntu_2.img 20G
Formatting 'ubuntu_2.img', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=21474836480 lazy_refcounts=off refcount_bits=16
```

创建虚拟机镜像

在 ubuntu 官网下载 ubuntu20.04 操作系统文件。（使用浏览器下载的方式）



操作系统镜像

在正式安装操作系统前我们需要检查 kvm 是否可用：

```
1 | grep -E 'vmx|svm' /proc/cpuinfo
```

```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ grep -E 'vmx|svm' /proc/cpuinfo
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat p
se36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perf
mon nopl xtopology tsc_reliable nonstop_tsc cpuid pni pclmulqdq vmx ssse3 fma cx16 pc
id sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hyp
ervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti ssbd ibrs ibpb stibp
tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid rdse
ed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves arat md_clear flush_l1d arch_ca
pabilities
vmx flags    : vnmi invvpid ept_x_only ept_ad tsc_offset vtptr mtf ept vpid unrestr
```

检查是否支持硬件虚拟化

指令有输出则说明硬件有虚拟化支持

再使用指令检查 kvm 是否可用:

```
1 | lsmod | grep kvm
```

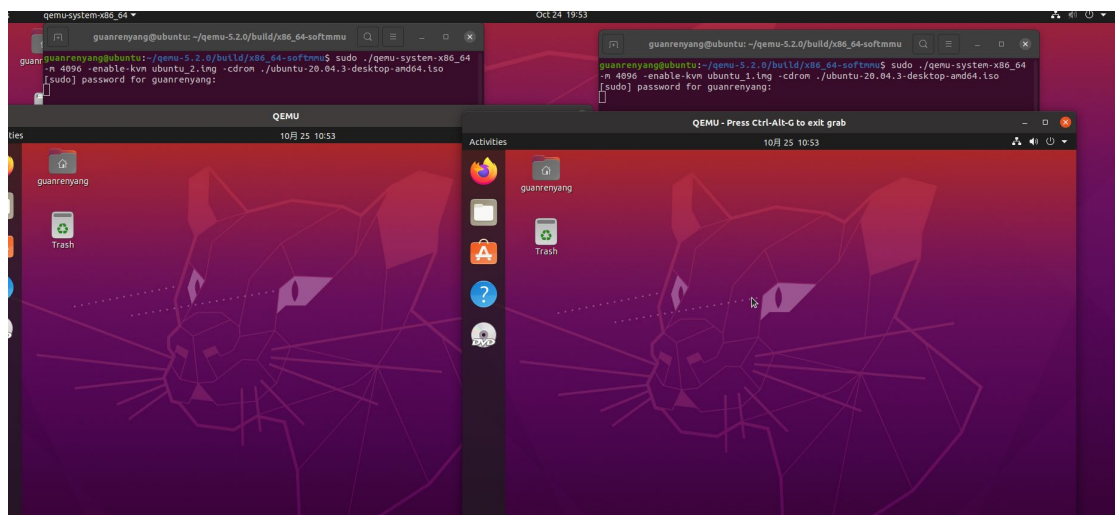
```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ lsmod | grep kvm
kvm_intel          294912  0
kvm                823296  1 kvm_intel
```

检查 kvm 是否可用

结果显示主机 kvm 可用

最后, 使用以下指令给虚拟机镜像安装操作系统:

```
1 | sudo ./qemu-system-x86_64 -m 4096 -enable-kvm ubuntu_1.img -cdrom ./ubuntu-20.04.1-desktop-amd64.iso
2 | sudo ./qemu-system-x86_64 -m 4096 -enable-kvm ubuntu_2.img -cdrom ./ubuntu-20.04.1-desktop-amd64.iso
```



安装虚拟机

之后使用可以使用以下指令开启虚拟机:

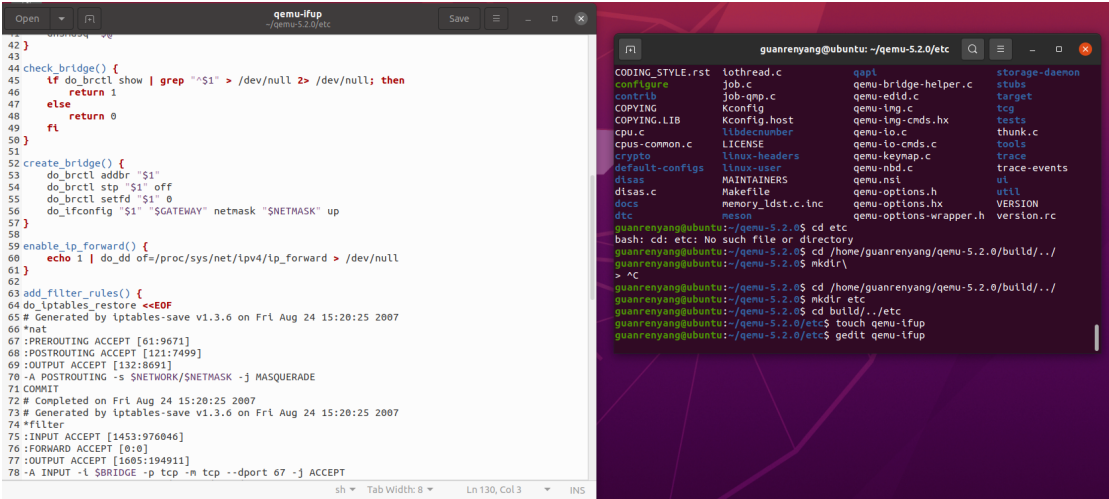
```
1 | sudo ./qemu-system-x86_64 -m 4096 -enable-kvm ubuntu_1.img -net tap -net nic,model=e1000
2 | sudo ./qemu-system-x86_64 -m 4096 -enable-kvm ubuntu_2.img -net tap -net nic,model=virtio-net-pci
```

其中镜像 ubuntu_1.img 使用 e1000, ubuntu_2 使用 virtio-net; 两个镜像都使用 tap (桥接) 网络模式。(实际上创建一个镜像启动两次也可以)

```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ sudo ./qemu-system-x86_64
-m 4096 -enable-kvm ubuntu_1.img -net tap -net nic,model=e1000
[sudo] password for guanrenyang:
qemu-system-x86_64: network script /home/guanrenyang/qemu-5.2.0/build/./etc/qemu-ifup
failed with status 256
```

直接以 tap 模式打开虚拟机出现错误

直接启动会报如上错误。信息显示需要配置 `qemu-ifup` 文件。需要根据链接 <https://wiki.qemu.org/Documentation/Networking/NAT> 进行配置

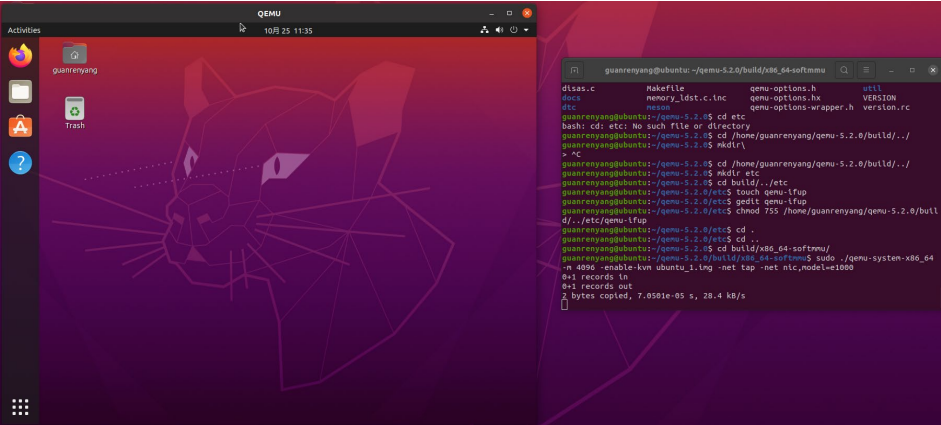


配置 qemu-ifup 文件-1

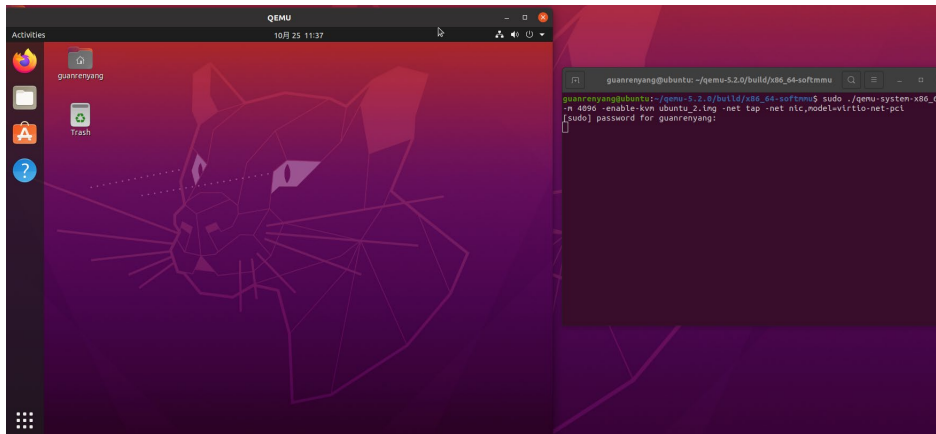
```
guanrenyang@ubuntu:~/qemu-5.2.0/etc$ chmod 755 /home/guanrenyang/qemu-5.2.0/build/./etc/qemu-ifup
```

配置 qemu-ifup 文件-2

启动后完成操作系统的安装即可。



e1000



virtio-net

结果显示两个虚拟机均安装成功：桥接（TAP）模式，分别为 e1000 与 virtio-net

任务三：使用 vnc 与 ssh 连接虚拟机

SSH

要使用虚拟机作为服务器，需要安装 ssh-server。在使用以上步骤打开虚拟机后，使用以下指令安装 openssh-server：

```
1 sudo apt install openssh-server
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /lib/s
systemd/system/ssh.service.
rescue-ssh.target is a disabled or a static unit, not starting it.
Setting up ssh-import-id (5.10-0ubuntu1) ...
Attempting to convert /etc/ssh/ssh_import_id
Setting up ncurses-term (6.2-0ubuntu2) ...
Processing triggers for systemd (245.4-4ubuntu3.11) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ufw (0.36-6) ...
```

安装 ssh-server 成功

在虚拟机内使用 ifconfig 指令（需要提前安装 net-tools）查看虚拟机的 ip 地址：

```
guanrenyang@ubuntu-Gry:~$ 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::bbc3:e50:4001:7a4c prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:12:34:56 txqueuelen 1000 (Ethernet)
    RX packets 814 bytes 965987 (965.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 485 bytes 40891 (40.8 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 191 bytes 16377 (16.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 191 bytes 16377 (16.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

虚拟机 ip 地址为 192.168.53.76

在主机内使用 ssh 用户名@主机名 来进行 ssh 连接：


```

guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ ssh guanrenyang@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:W0BQiE1A1TRMj/1outj/J6eH/z76erXPGVnZZB9V2dc.
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.
guanrenyang@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

95 updates can be applied immediately.
37 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

```

SSH 连接

结果显示 ssh 连接成功

VNC viewer

服务器（虚拟机）端配置

先使用 ssh 连接虚拟机，然后使用以下在服务器（虚拟机）安装必要依赖库

```

1 sudo apt update
2 sudo apt install xfce4 xfce4-goodies

```

使用以下指令安装服务器端 vnc-server（使用 TigerVNC）

```

1 sudo apt install tigervnc-standalone-server

```

使用 vncpasswd 指令配置 vnc 用户密码：

```

guanrenyang@ubuntu-Gry:~$ vncpasswd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n

```

接下来使用 Xfce 配置 TigerVNC，使用 nano 创建文件 ~/.vnc/xstartup 并将以下内容复制进去，保存退出。并使用 chmod 指令给此文件可执行权限。

```

guanrenyang@ubuntu-Gry:~$ nano ~/.vnc/xstartup
guanrenyang@ubuntu-Gry:~$ chmod u+x ~/.vnc/xstartup

```

```

GNU nano 4.8 /home/guanrenyang/.vnc/xstartup
#!/bin/sh
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS
exec startxfce4

```

使用 vncserver 指令启动 vnc 服务


```

guanrenyang@ubuntu-Gry:~$ vncserver
/usr/bin/xauth: file /home/guanrenyang/.Xauthority does not exist

New 'ubuntu-Gry:1 (guanrenyang)' desktop at :1 on machine ubuntu-Gry

Starting applications specified in /home/guanrenyang/.vnc/xstartup
Log file is /home/guanrenyang/.vnc/ubuntu-Gry:1.log

Use xtigervncviewer -SecurityTypes VncAuth -passwd /home/guanrenyang/.vnc/passwd
:1 to connect to the VNC server.

```

使用 `vncserver -list` 指令查看正在服务中的 vnc 会话
 并使用 `vncserver -kill :1`` 指令停止正在执行中的服务

```

guanrenyang@ubuntu-Gry:~$ vncserver -list

TigerVNC server sessions:

X DISPLAY #      RFB PORT #      PROCESS ID
:1           5901           6029
guanrenyang@ubuntu-Gry:~$ vncserver -kill :1
Killing Xtigervnc process ID 6029... success!

```

下面创建一个系统的单元文件，以便根据需要启动，停止和重新启动 VNC 服务，而不是手动启动 VNC 会话。

使用以下指令创建系统配置文件，并将对应内容复制进去。

```
1 sudo nano /etc/systemd/system/vncserver@.service
```

```

GNU nano 4.8 /etc/systemd/system/vncserver@.service
[Unit]
Description=Remote desktop service (VNC)
After=syslog.target network.target

[Service]
Type=simple
User=guanrenyang
PAMName=login
PIDFile=/home/%u/.vnc/%H%i.pid
ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill :%i > /dev/null 2>&1 || :'
ExecStart=/usr/bin/vncserver :%i -geometry 1440x900 -alwaysshared -fg
ExecStop=/usr/bin/vncserver -kill :%i

[Install]
WantedBy=multi-user.target

```

使用以下指令告知系统配置文件生成、使能 vnc 并启动 vnc 服务：

```

1 sudo systemctl daemon-reload
2 sudo systemctl enable vncserver@1.service
3 sudo systemctl start vncserver@1.service

```

```
guanrenyang@ubuntu-Gry:~$ sudo systemctl daemon-reload
guanrenyang@ubuntu-Gry:~$ sudo systemctl enable vncserver@1.service
Created symlink /etc/systemd/system/multi-user.target.wants/vncserver@1.service
→ /etc/systemd/system/vncserver@.service.
guanrenyang@ubuntu-Gry:~$ sudo systemctl start vncserver@1.service
```

使用以下指令查看 vnc 服务是否启动

```
1 | sudo systemctl status vncserver@1.service
```

vnc 开启成功

```
guanrenyang@ubuntu-Gry:~$ sudo systemctl status vncserver@1.service
● vncserver@1.service - Remote desktop service (VNC)
   Loaded: loaded (/etc/systemd/system/vncserver@.service; enabled; vendor pr
   Active: active (running) since Tue 2021-10-26 00:38:07 CST; 4s ago
     Process: 6430 ExecStartPre=/bin/sh -c /usr/bin/vncserver -kill :1 > /dev/nu
    Main PID: 6437 (vncserver)
       Tasks: 0 (limit: 4651)
      Memory: 724.0K
     CGroup: /system.slice/system-vncserver.slice/vncserver@1.service
            └─ 6437 /usr/bin/perl /usr/bin/vncserver :1 -geometry 1440x900 -alw
10月 26 00:38:06 ubuntu-Gry systemd[1]: Starting Remote desktop service (VNC)...
10月 26 00:38:06 ubuntu-Gry systemd[6430]: pam_unix(login:session): session ope
10月 26 00:38:07 ubuntu-Gry systemd[1]: Started Remote desktop service (VNC).
10月 26 00:38:07 ubuntu-Gry systemd[6437]: pam_unix(login:session): session ope
```

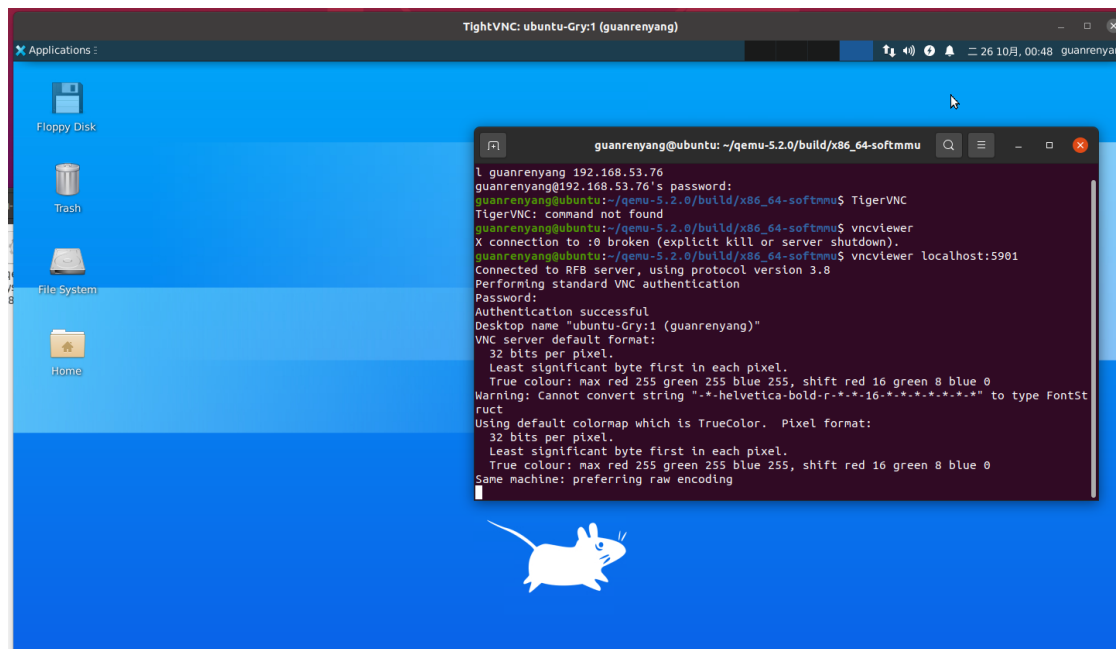
由于 VNC 不是加密协议，可能出现安全问题，所以创建一个 SSH 隧道，在主机 5901 端口和虚拟机的 5901 端口间安全地传递信息：

```
1 | ssh -L 5901:127.0.0.1:5901 -N -f -l `user_name` `serve_ip`
```

```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ ssh -L 5901:127.0.0.1:5901 -N -f -l
l guanrenyang 192.168.53.76
guanrenyang@192.168.53.76's password:
```

输入密码后隧道建立完成。

最后在主机上使用 `vncviewer localhost:5901` 指令连接服务器：



结果显示 VNC 连接完成

任务四：测试主机与虚拟机的网络速度

对于 virtio-net 我们使用 vnc 连接，对于 e1000 我们使用 ssh 连接。

在虚拟机中使用以下指令安装测速工具 iperf

```
1 | sudo apt-get install iperf
```

```
guanrenyang@ubuntu-Gry: /
File Edit View Search Terminal Help
guanrenyang@ubuntu-Gry:/$ sudo apt-get install iperf
[sudo] password for guanrenyang:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  iperf
0 upgraded, 1 newly installed, 0 to remove and 95 not upgraded.
Need to get 76.5 kB of archives.
After this operation, 213 kB of additional disk space will be used.
Get:1 http://cn.archive.ubuntu.com/ubuntu focal/universe amd64 iperf amd64 2.0.13+dfsg1-1build1 [76.5 kB]
Fetched 76.5 kB in 1s (55.6 kB/s)
Selecting previously unselected package iperf.
(Reading database ... 190629 files and directories currently installed.)
Preparing to unpack .../iperf_2.0.13+dfsg1-1build1_amd64.deb ...
Unpacking iperf (2.0.13+dfsg1-1build1) ...
Setting up iperf (2.0.13+dfsg1-1build1) ...
Processing triggers for man-db (2.9.1-1) ...
guanrenyang@ubuntu-Gry:/$
```

virtio-net

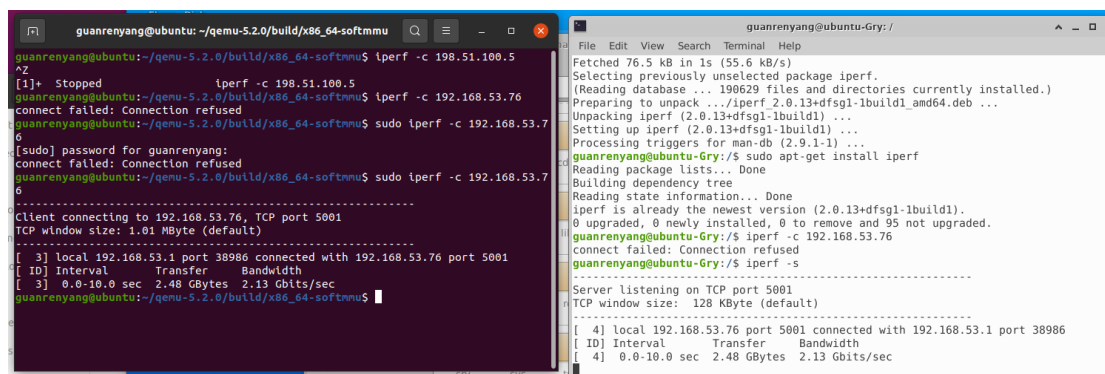
```
guanrenyang@ubuntu-Gry:~$ sudo apt-get install iperf
[sudo] password for guanrenyang:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  iperf
0 upgraded, 1 newly installed, 0 to remove and 95 not upgraded.
Need to get 76.5 kB of archives.
After this operation, 213 kB of additional disk space will be used.
Get:1 http://cn.archive.ubuntu.com/ubuntu focal/universe amd64 iperf amd64 2.0.13+dfsg1-1build1 [76.5 kB]
Fetched 76.5 kB in 1s (55.5 kB/s)
Selecting previously unselected package iperf.
(Reading database ... 190648 files and directories currently installed.)
Preparing to unpack .../iperf_2.0.13+dfsg1-1build1_amd64.deb ...
Unpacking iperf (2.0.13+dfsg1-1build1) ...
Setting up iperf (2.0.13+dfsg1-1build1) ...
Processing triggers for man-db (2.9.1-1) ...
```

e1000

```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ sudo apt-get install iperf
[sudo] password for guanrenyang:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-5.11.0-27-generic linux-hwe-5.11-headers-5.11.0-27
  linux-image-5.11.0-27-generic linux-modules-5.11.0-27-generic
  linux-modules-extra-5.11.0-27-generic
```

主机

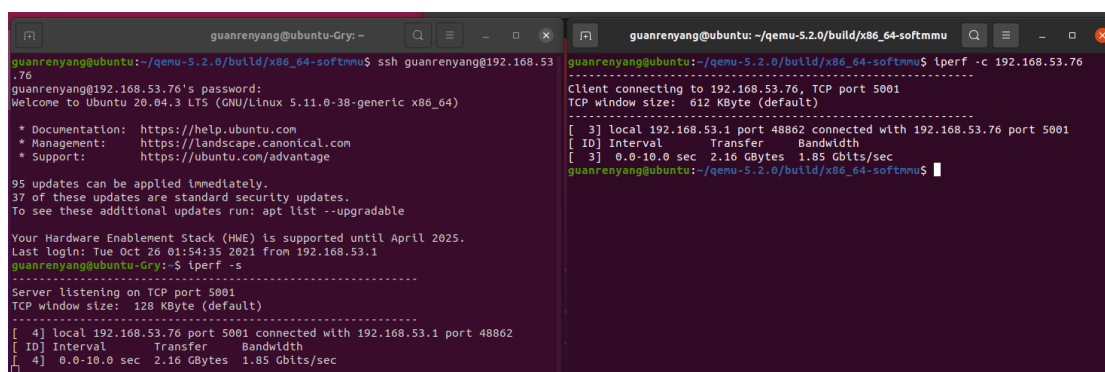
在虚拟机使用 `iperf -s` 指令设置 iperf 为服务器模式
在主机使用 `iperf -c server_ip` 指令来测试速度



```
guanrenyang@ubuntu: ~/qemu-5.2.0/build/x86_64-softmmu
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ iperf -c 192.168.53.76
^Z
[1]+  Stopped                  iperf -c 192.168.53.76
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ iperf -c 192.168.53.76
connect failed: Connection refused
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ sudo iperf -c 192.168.53.76
[sudo] password for guanrenyang:
connect failed: Connection refused
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ sudo iperf -c 192.168.53.76
Client connecting to 192.168.53.76, TCP port 5001
TCP window size: 1.01 MByte (default)
[ 3] local 192.168.53.1 port 38986 connected with 192.168.53.76 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  2.48 GBytes  2.13 Gbits/sec
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$
```

```
guanrenyang@ubuntu-Gry: /
File Edit View Search Terminal Help
Fetched 76.5 kB in 1s (55.6 kB/s)
Selecting previously unselected package iperf.
(Reading database ... 190629 files and directories currently installed.)
Preparing to unpack .../iperf_2.0.13+dfsg1-1build1_amd64.deb ...
Unpacking iperf (2.0.13+dfsg1-1build1) ...
Setting up iperf (2.0.13+dfsg1-1build1) ...
Processing triggers for man-db (2.9.1-1) ...
guanrenyang@ubuntu-Gry:/$ sudo apt-get install iperf
Reading package lists... Done
Building dependency tree
Reading state information... Done
iperf is already the newest version (2.0.13+dfsg1-1build1).
0 upgraded, 0 newly installed, 0 to remove and 95 not upgraded.
guanrenyang@ubuntu-Gry:/$ iperf -c 192.168.53.76
connect failed: Connection refused
guanrenyang@ubuntu-Gry:/$ iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 192.168.53.76 port 5001 connected with 192.168.53.1 port 38986
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-10.0 sec  2.48 GBytes  2.13 Gbits/sec
```

virtio-net



```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ ssh guanrenyang@192.168.53.76
guanrenyang@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

95 updates can be applied immediately.
37 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Tue Oct 26 01:54:35 2021 from 192.168.53.1
guanrenyang@ubuntu-Gry:~$ iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[ 4] local 192.168.53.76 port 5001 connected with 192.168.53.1 port 48862
[ ID] Interval      Transfer    Bandwidth
[ 4] 0.0-10.0 sec  2.16 GBytes  1.85 Gbits/sec
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$
```

```
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$ iperf -c 192.168.53.76
Client connecting to 192.168.53.76, TCP port 5001
TCP window size: 612 KByte (default)
[ 3] local 192.168.53.1 port 48862 connected with 192.168.53.76 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-10.0 sec  2.16 GBytes  1.85 Gbits/sec
guanrenyang@ubuntu:~/qemu-5.2.0/build/x86_64-softmmu$
```

e1000

结果显示：virtio-net 的速率快于 e1000 的速率，这与预期相符合——全虚拟化（e1000）的速率慢于半虚拟化（virtio-net）的速率。

总结与反思

本次作业主要涵盖了下载、编译、安装 qemu，通过 qemu 使用指定的网路模式创建虚拟机，使用 ssh 和 vnc-server 与服务器（虚拟机）通信的过程；并测试了全虚拟化（e1000）与半虚拟化（virtio-net）的速率。作业涉及的内容丰富，让我学到了基本的 linux 系统用于 qemu 虚拟机搭建这种工程性的知识，更让我对网络通信方式与全/半虚拟化技术有了更深入的理解。对于技术上的难点，我也不满足于在工程上解决问题、完成任务，而且查阅相关资料，力图深入理解各个任务的原理、各项操作的原因。

但是，本次作业也暴露出了我的许多问题，其中最主要的就是对全英文文档的畏难情绪。在任务起初我像无头苍蝇一样尝试各种不甚靠谱的方法来开启桥接（tap）模式，更改了大量底层文件，导致我不得不多次重新安装虚拟机从头开始。最终我通过查阅并仔细阅读 qemu 的官方文档解决了问题。

本次作业让我学会的不仅仅是搭建虚拟机、配置服务器、进行网络通信等工程能力，而且促使我对虚拟化技术、计算机网络有了更深刻的理解，更加让我克服了畏难心理、锻炼了我自主探索解决问题的能力，我因此受益匪浅。

有价值的参考文档

1. qemu 网络模式设置

<https://wiki.qemu.org/Documentation/Networking/NAT>

https://wiki.qemu.org/Documentation/Networking#The_-nic_option

2. VNC

<https://linuxize.com/post/how-to-install-and-configure-vnc-on-ubuntu-20-04/>

3. ssh

<https://linuxize.com/post/how-to-enable-ssh-on-ubuntu-20-04/>

4. iperf

<https://www.linode.com/docs/guides/install-iperf-to-diagnose-network-speed-in-linux/>