

DPDK

工程实践与科技创新III-D 虚拟化与云计算 EI313

李子龙 518070910095

2021 年 11 月 30 日

目录

1	要求	1
2	编译安装 DPDK	1
3	配置网卡	3
4	编译 l2fwd pktgen	4
5	运行测试	5
6	评估	6

1 要求

DPDK performance test:

- (1) Create two virtual machines on KVM.
- (2) Compile and install DPDK library on each virtual machine.
- (3) Compile and run DPDK sample application l2fwd on VM2, then compile and run pktgen-dpdk on VM1. pktgen-dpdk will record the size of the packages VM1 sends and the amount of packages received from VM2, while l2fwd just send back the packages it received from VM1.
- (4) Evaluate DPDK's performance of L2 forwarding.

2 编译安装 DPDK

按照官方文档^[1]编译 DPDK，如图 1、2、3 所示。

Listing 1: [INSTALL.sh](#)

```
1 #!/bin/bash
2
3 sudo -s
4
5 wget https://fast.dpdk.org/rel/dpdk-21.08.tar.xz
6 tar xf dpdk-21.08.tar.xz
```

```

7 cd dpdk-21.08
8
9 apt install meson
10 apt install python3-pip python3-pyelftools
11 pip3 install meson
12
13 # reboot if necessary
14 # do not build examples!
15
16 meson build
17 ninja -C build
18 ninja install
19 ldconfig
20
21 # ninja clean -C build
22 # if necessary
23
24 mkdir -p /dev/hugepagesz
25 mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
26 echo 64 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/
   nr_hugepages

```

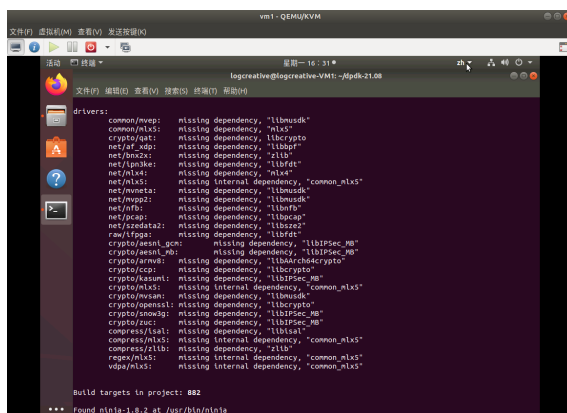


图 1: meson 配置

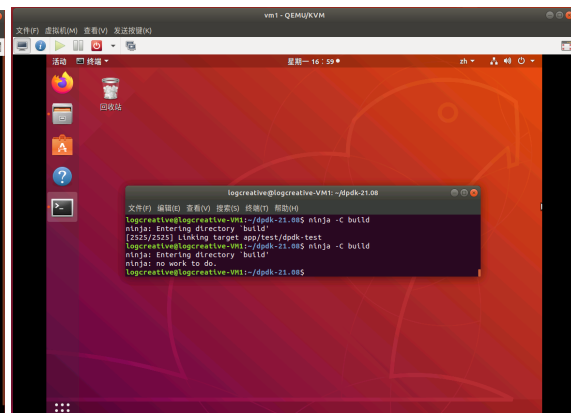


图 2: ninja 编译

之后对该虚拟机进行克隆，得到另一个已经安装 DPDK 的虚拟机，如图 4 所示。

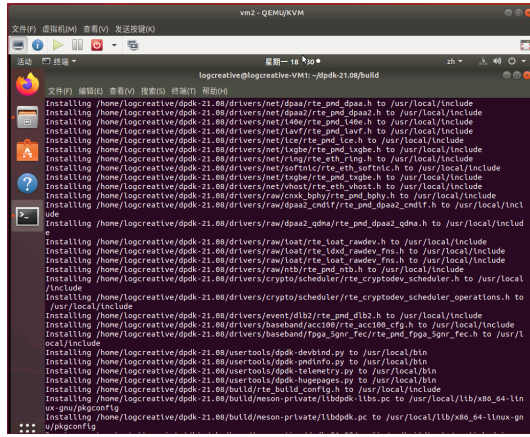


图 3: ninja 安装



图 4: 克隆虚拟机

3 配置网卡

通过 virt-manager 设置两个虚拟机为两个网卡的 NAT 网络连接（如图 5），然后需要在两个虚拟机上加载 uio_pci_generic 驱动^[2]，然后绑定在 DPDK 上^[3]（如图 6）。

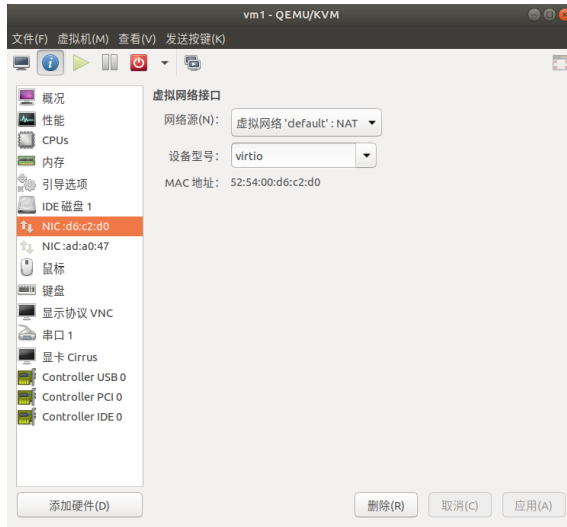


图 5: NAT 设置

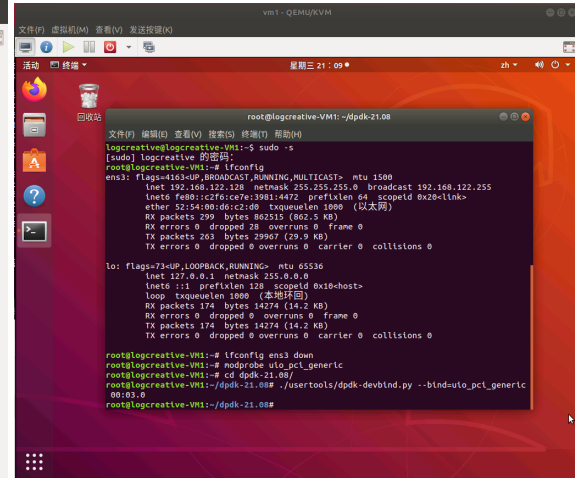


图 6: DPDK 绑卡

Listing 2: nic.sh

```
1 ifconfig ens3 down # disable the tap netcard
```

```

2 ifconfig ens6 down
3 sudo modprobe uio_pci_generic # start driver
4 ./usertools/dpdk-devbind.py --status
5 ./usertools/dpdk-devbind.py --bind=uio_pci_generic 00:03.0
6 ./usertools/dpdk-devbind.py --bind=uio_pci_generic 00:06.0

```

4 编译 l2fwd pktgen

这一步在上一步配置完成后进行，需要用到已经配置好的 DPDK 库。在 VM2 上编译 l2fwd，如图 7。编译前需要修改 l2fwd 的 main.c 其中一行，关闭混杂模式^[4]，也就是注释下面这一行（不同的版本可能行号不同）。

Listing 3: ../dpdk/examples/l2fwd/main.c

```
883 ret = rte_eth_promiscuous_enable(portid);
```

Listing 4: l2fwd.sh

```

1 sudo apt-get update --fix-missing
2 sudo apt install pkg-config
3 export RTE_SDK=~/dpdk-21.08
4 export RTE_TARGET=x86_64-native-linuxapp-gcc
5 cd ~/dpdk-21.08/examples/l2fwd
6 make

```

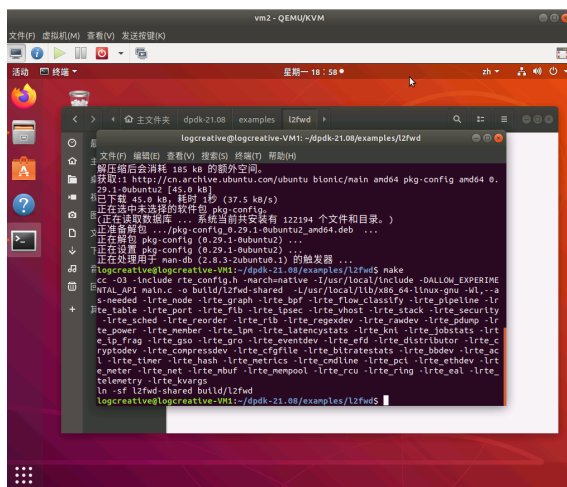


图 7: l2fwd 编译

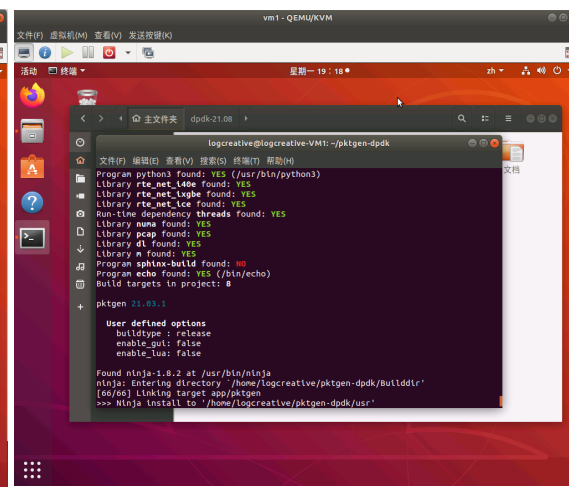


图 8: pktgen 编译

在 VM1 上编译 pktgen，如图 8。

Listing 5: `pktgen.sh`

```

1 sudo apt install git pkg-config numactl libnuma-dev libpcap-dev
2 git clone git://dpdk.org/apps/pktgen-dpdk
3 cd pktgen-dpdk
4 export RTE_SDK=~/dpdk-21.08
5 export RTE_TARGET=x86_64-native-linuxapp-gcc
6 make
7 # copy to the detectable directory
8 cp ./usr/local/bin/pktgen /usr/local/bin/pktgen

```

5 运行测试

同时打开两个虚拟机¹。启动测试的方法是在两个虚拟机上分别运行对应的部分^[5]，配置完成后如图 10 所示。

如果运行失败，需要重新分配 Hugepages（可能已经用光了），方法见第 2 节。注意在运行 `l2fwd` 时加了 `--no-mac-updating` 参数，因为不需要发送过多的 ARP 包污染计数^[4]。

在 `pktgen` 的 CLI 中更改 `count` 的数目，以检测瞬时的包接收性能。

Listing 6: `test.sh`

```

1 sudo -s
2
3 # vm 1
4 pktgen -l 0-1 -n 3 -- -P -m "[1].0"
5 set 0 dst ip 192.168.122.120
6 set 0 dst mac 52:54:00:de:b8:46
7 set 0 count 1000
8 start 0
9
10 # vm 2
11 ./build/l2fwd -c 0x3 -n 4 -- -p 3 -q 1 --no-mac-updating

```

这样得到的发包拓扑如图 9 所示。

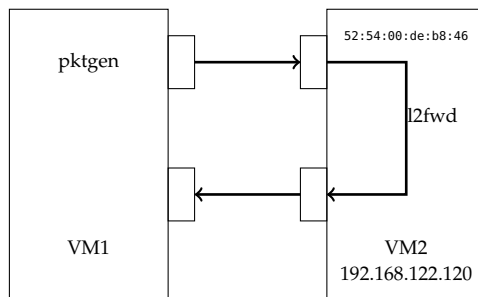


图 9: 发包拓扑

¹请不要分配超过限制的 CPU 数量，否则会闪退。这里为每一个虚拟机分配了 2 个核。

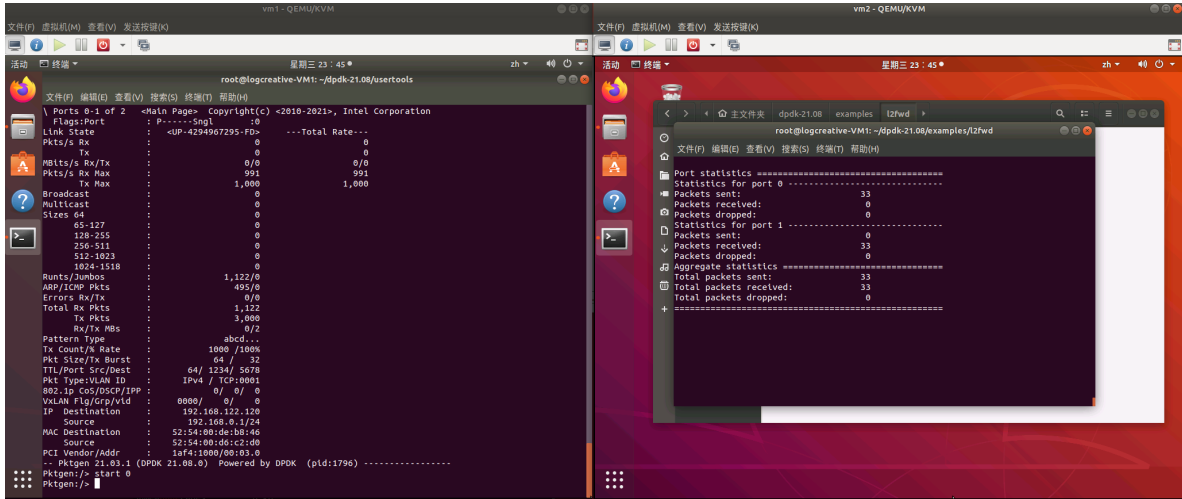


图 10: 两个虚拟机的配置

6 评估

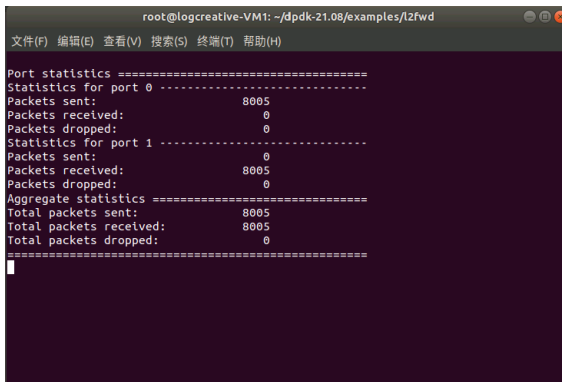


图 11: 发包数 8000

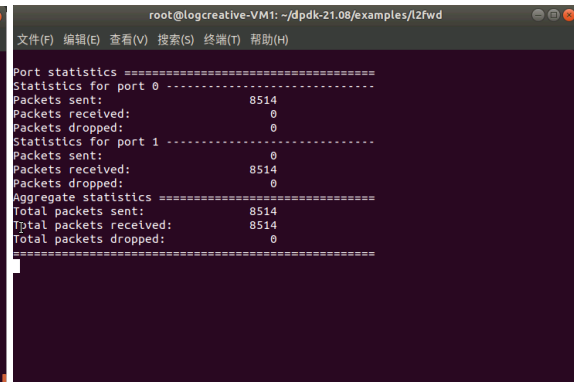


图 12: 发包数 10000

如图 11 所示，在瞬时发包数为 8000 时还是不失包的。而如图 12 所示，在瞬时发包数为 10000 时丢了一部分包（15%），也就是性能瓶颈大概在 8500 个包左右，性能还是很可观的，约等于几千个程序同时服务。

于此形成对照的是使用下面的脚本（使用原生的 `pktgen`^[6]），10000 发包数丢失 40% 左右的包，如图 13 所示。

Listing 7: [comparetest.sh](#)

```
1 sudo -s
2
3 # vm 1
4 modprobe pktgen
5 lsmod | grep -i pktgen
6 echo "add_device ens3" > /proc/net/pktgen/kpktgend_0
7 cat /proc/net/pktgen/ens3
8 cat /proc/net/pktgen/pgctrl
9 echo "dst_mac 52:54:00:DE:B8:46" > /proc/net/pktgen/ens3
10 echo "count 10000" > /proc/net/pktgen/ens3
11 echo "dst_min 192.168.122.120" > /proc/net/pktgen/ens3
12 echo "dst_max 192.168.122.120" > /proc/net/pktgen/ens3
13 echo "start" > /proc/net/pktgen/pgctrl
```

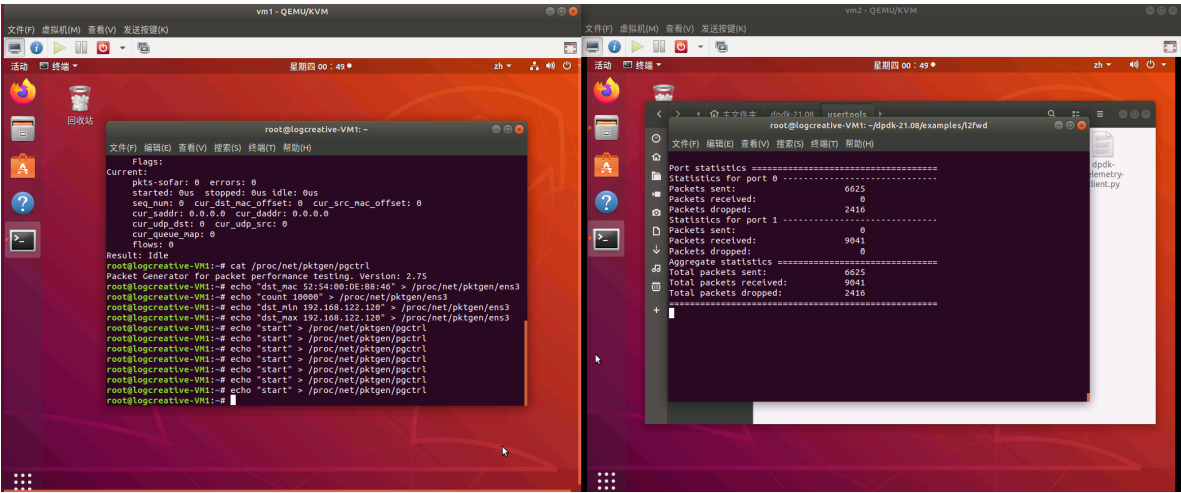


图 13: 对照实验

所以 DPDK 运行在操作系统的 User Space，利用自身提供的数据面库进行收发包处理，绕过了 Linux 内核态协议栈^[7]，可以大幅提高包的转发效率。

运行环境

附运行环境如表 1 所示。

表 1: 运行环境

系统	Ubuntu 18.04
核心数	2核4逻辑核
虚拟机分配逻辑核	2
虚拟机内存	1024MB
虚拟机网卡	virtio
DPDK 版本	21.08
pktgen 版本	20.02

参考文献

- [1] DPDK Project. Compiling the dpdk target from source[M/OL]. 2021. http://doc.dpdk.org/guides-21.08/linux_gsg/build_dpdk.html.
- [2] OSINSKI T. Configuring ovs-dpdk with vm[R/OL]. 2019. <https://osinstom.github.io/en/tutorial/configuring-ovs-dpdk-with-vm/#>.
- [3] 小蚂蚁CYJ. DPDK（三）：入门1—DPDK官方用例[EB/OL]. 2019. <https://www.cnblogs.com/xiaomayi-cyj/p/10543157.html>.
- [4] DPDK Project. L2 forwarding sample application (in real and virtualized environments) [EB/OL]. 2021. https://doc.dpdk.org/guides-18.08/sample_app_ug/l2_forward_real_virtual.html.
- [5] 简Ki. DPDK L2FWD使用[EB/OL]. 2018. <https://www.jianshu.com/p/cec11b3f4fb3>.
- [6] 简Ki. DPDK PKTGEN使用[EB/OL]. 2018. <https://www.jianshu.com/p/fa7d9f2c0f55>.
- [7] Hu先生的Linux. DPDK的基本原理、学习路线总结[EB/OL]. 2022. <https://zhuanlan.zhihu.com/p/347693559>.