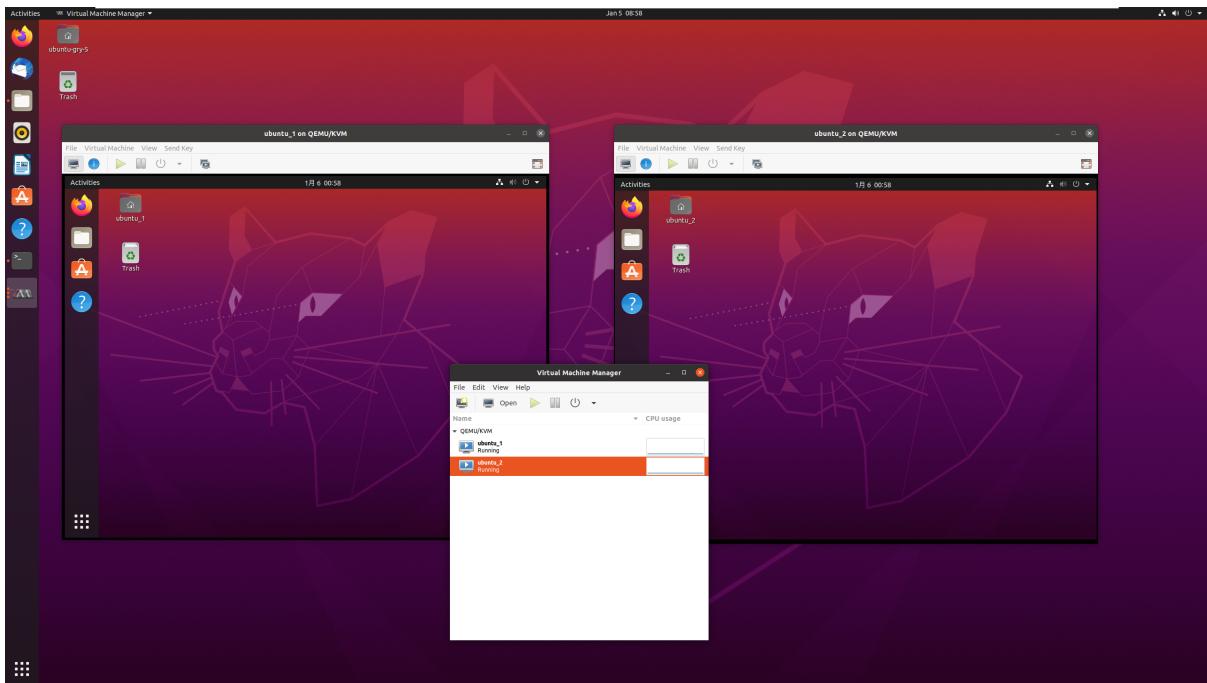




工程实践与科技创新 5

管仁阳-519021911058

一、创建两个虚拟机 (KVM)

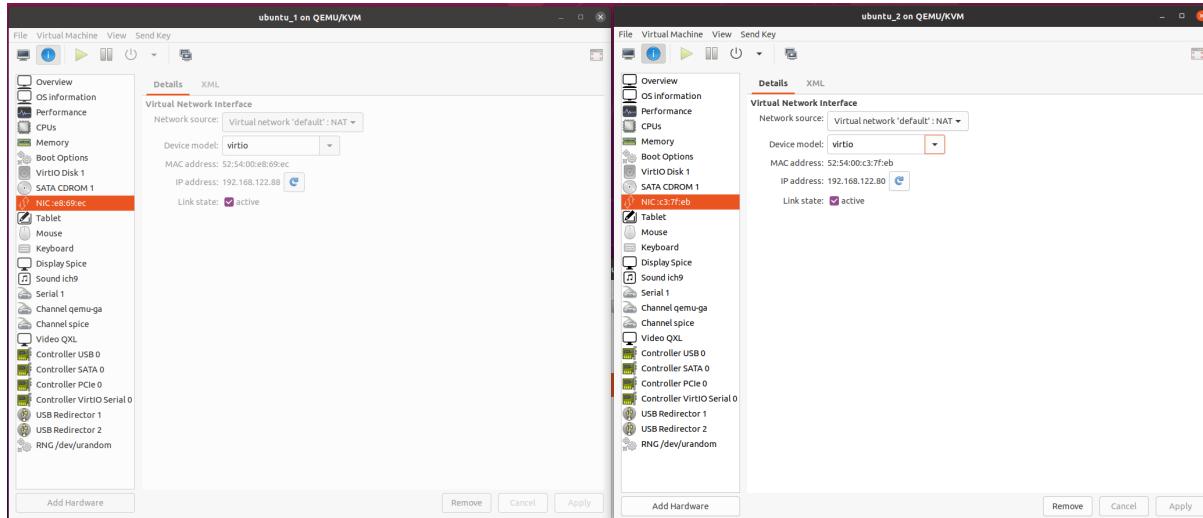


ubuntu_1 为 VM1。在性能评估中运行 **pktgen**

ubuntu_2 为 VM2。在性能评估中运行 **I2fwd**

Guest 网卡设置

VM1和VM2都使用NAT方式，virtio网卡。需要在 host 中建立网桥。



建立网桥

首先开启 guest。在 host 中执行 ifconfig 可以看到两个虚拟网卡 vnet0 和 vnet1。默认情况下这两个网卡与 virt-manager 的默认网桥virbr0相连。

```
vnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet6 fe80::fc54:ff:fee8:69ec prefixlen 64 scopeid 0x20<link>
        ether fe:54:00:e8:69:ec txqueuelen 1000 (Ethernet)
          RX packets 197 bytes 25946 (25.9 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 792 bytes 85699 (85.6 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vnet1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet6 fe80::fc54:ff:fec3:7feb prefixlen 64 scopeid 0x20<link>
        ether fe:54:00:c3:7f:eb txqueuelen 1000 (Ethernet)
          RX packets 197 bytes 26104 (26.1 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 786 bytes 85443 (85.4 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

我们需要使用brctl工具建立网桥，将vnet0、vnet1与virbr0解绑并与新建立的网桥绑定，最后启动网桥。在host执行命令

```
sudo brctl delif virbr0 vnet0 # 将vnet0与virbr0解绑
sudo brctl delif virbr0 vnet1 # 将vnet1与virbr0解绑
sudo brctl addbr br0          # 建立网桥 br0
sudo brctl addif br0 vnet0    # 将vnet0加入br0
sudo brctl addif br0 vnet1    # 将vnet1加入br0
sudo ifconfig br0 up          # 启动br0
```

使用如下指令查看网桥详情：

```
sudo brctl show
```

```
ubuntu-gry-5@ubuntu:~$ sudo brctl show
bridge name      bridge id      STP enabled      interfaces
br0              8000.fe5400c37feb    no            vnet0
                                         vnet1
```

二、编译安装DPDK

本部分介绍安装DPDK的全过程。新版DPDK使用meson+ninja全自动编译，因此只展示了成功运行 helloworld的截图

1. 安装依赖库

```
pip3 install meson ninja pwntools pyelftools
```

2. 获取源码

```
git clone git://dpdk.org/dpdk
```

3. 编译安装（使用-Dexamples=all参数编译所有样例）

```
cd dpdk
sudo meson -Dexamples=all build
cd build
sudo ninja install
```

安装成功以后，样例代码在在 /dpdk/build/examples/中

运行样例程序 helloworld

1. 分配 Hugepage（每次重启虚拟机都需要重新分配，且Hugepage数量要足够大）

```
mkdir -p /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
echo 512 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
```

2. 执行样例程序 helloworld

```
./build/examples/dpdk-helloworld
```

```
root@ubuntu-1:/home/ubuntu_1/dpdk# ./build/examples/dpdk-helloworld
EAL: Detected CPU lcores: 4
EAL: Detected NUMA nodes: 1
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available 1048576 kB hugepages reported
EAL: VFIO support initialized
EAL: Probe PCI driver: net_virtio (1af4:1041) device: 0000:01:00.0 (socket 0)
eth_virtio_pci_init(): Failed to init PCI device
EAL: Requested device 0000:01:00.0 cannot be used
TELEMETRY: No legacy callbacks, legacy socket not created
hello from core 1
hello from core 2
hello from core 3
hello from core 0
```

三、在VM2上编译运行样例程序：l2fwd

分配 Hugepage (如上所述)

将网卡与DPDK绑定

```
sudo ifconfig enp1s0 down # enp1s0 为guest的网卡名,可使用ifconfig查看
sudo modprobe uio_pci_generic
sudo ./dpdk/usertools/dpdk-devbind.py --bind=uio_pci_generic enp1s0
sudo ./dpdk/usertools/dpdk-devbind.py --status # 查看绑定状态
```

可见DPDK与网卡绑定成功

```
root@ubuntu-2:/home/ubuntu_2/dpdk# sudo ifconfig enp1s0 down # enp1s0 为guest的 网卡名,可使用
ifconfig查看
root@ubuntu-2:/home/ubuntu_2/dpdk# sudo modprobe uio_pci_generic
root@ubuntu-2:/home/ubuntu_2/dpdk# sudo ./usertools/dpdk-devbind.py --bind=uio_pci_generic en
p1s0
root@ubuntu-2:/home/ubuntu_2/dpdk# sudo ./usertools/dpdk-devbind.py --status # 查看绑定状态
Network devices using DPDK-compatible driver
=====
0000:01:00.0 'Virtio network device 1041' drv=uio_pci_generic unused=vfio-pci
```

运行 l2fwd

```
./dpdk/build/examples/dpdk-l2fwd -c 1 -n 2 -- -q 1 -p 1 -T 5
```

参数解释：

- -c：分配给DPDK的 **内核数**
- -n：每个CPU的 **内存通道数**
- --：之后为次参数
- -q：每个CPU管理的 **队列数**
- -p：端口
- -T：屏幕刷新间隔

l2fwd 启动成功截图：

```
root@ubuntu-2:/home/ubuntu_2/dpdk# ./build/examples/dpdk-l2fwd -c 1 -n 2 -- -q 1 -p 1 -T 5
EAL: Detected CPU lcores: 4
EAL: Detected NUMA nodes: 1
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available 1048576 kB hugepages reported
EAL: VFIO support initialized
EAL: Probe PCI driver: net_virtio (1af4:1041) device: 0000:01:00.0 (socket 0)
TELEMETRY: No legacy callbacks, legacy socket not created
MAC updating enabled
Notice: odd number of ports in portmask.
Lcore 0: RX port 0 TX port 0
Initializing port 0... done:
Port 0, MAC address: 52:54:00:C3:7F:EB

Checking link statusdone
Port 0 Link up at Unknown FDX Autoneg
L2FWD: entering main loop on lcore 0
L2FWD: -- lcoreid=0 portid=0

Port statistics =====
Statistics for port 0 -----
Packets sent: 0
Packets received: 0
Packets dropped: 0
Aggregate statistics =====
Total packets sent: 0
Total packets received: 0
Total packets dropped: 0
=====
```

四、在VM1上编译运行 pktgen-dpdk

编译安装

pktgen-dpdk也使用meson+ninja全自动安装

1. 获取pktgen-dpdk

```
git clone git://dpdk.org/apps/pktgen-dpdk
```

2. 安装依赖库

```
sudo apt-get install -y cmake libnuma-dev libpcap-dev
```

3. 编译安装

```
sudo meson build  
cd build  
sudo ninja build
```

安装完成，可执行文件为 /pktgen-dpdk/build/app/pktgen

运行pktgen-dpdk

运行pktgen-dpdk前需要完成启动DPDK的分配Hugepage和将网卡与DPDK绑定两个步骤。

然后执行

```
./pktgen-dpdk/build/app/pktgen -l 0-2 -n 2 -- -P -m "[1].0"
```

参数解释：

- -l : 给pktgen使用的核（不代表都要用上，但是只要指定两个。0-2代表三个核0,1,2）
- -n : 每个核的内存通道数
- -- : 之后每个参数为次参数
- -P : 启用混杂模式
- -m : 端口与核之间的逻辑映射，[1].0表示核1来负责处理端口0的收包与发包，具体如下图所示：

```

1.0, 2.1, 3.2          - core 1 handles port 0 rx/tx,
                        core 2 handles port 1 rx/tx
                        core 3 handles port 2 rx/tx
1.[0-2], 2.3, ...      - core 1 handle ports 0,1,2 rx/tx,
                        core 2 handle port 3 rx/tx
[0-1].0, [2/4-5].1, ... - cores 0-1 handle port 0 rx/tx,
                        cores 2,4,5 handle port 1 rx/tx
[1:2].0, [4:6].1, ...   - core 1 handles port 0 rx,
                        core 2 handles port 0 tx,
[1:2].[0-1], [4:6].[2/3], ... - core 1 handles port 0 & 1 rx,
                        core 2 handles port 0 & 1 tx
[1:2-3].0, [4:5-6].1, ... - core 1 handles port 0 rx, cores 2,3 handle port 0 tx
                        core 4 handles port 1 rx & core 5,6 handles port 1 tx
[1-2:3].0, [4-5:6].1, ... - core 1,2 handles port 0 rx, core 3 handles port 0 tx
                        core 4,5 handles port 1 rx & core 6 handles port 1 tx
[1-2:3-5].0, [4-5:6/8].1, ... - core 1,2 handles port 0 rx, core 3,4,5 handles port 0 tx
                        core 4,5 handles port 1 rx & core 6,8 handles port 1 tx
[1:2].[0:0-7], [3:4].[1:0-7], - core 1 handles port 0 rx, core 2 handles ports 0-7 tx
                                core 3 handles port 1 rx & core 4 handles port 0-7 tx
BTW: you can use "{}" instead of "[]" as it does not matter to the syntax.

```

pktgen-dpdk的启动与使用

启动pktgen-dpdk以后出现如下界面，显示了使用pktgen-dpdk的收发包的相关信息，具体指标如左列所示。

```

Link State      : <UP-4294967295-FD>    ---Total Rate---
Pkts/s Rx       : P-----Sngl      :0                      0
                  Tx       : <UP-4294967295-FD>          0
MBits/s Rx/Tx   :                               0                      0
Pkts/s Rx Max   :                               0                      0
                  Tx Max   : 0/0                    0/0
Broadcast       :                               3                      3
Multicast       :                               0                      0
Sizes 64        :                               0
                  65-127     : 0
                  128-255    : 0
                  256-511    : 2
                  512-1023   : 0
                  1024-1518  : 0
Runts/Jumbos    :                               0
ARP/ICMP Pkts   :                               0
Errors Rx/Tx    :                               10/0
Total Rx Pkts   :                               0/0
                  Tx Pkts   : 0/0
                  Rx/Tx MBs: 11
TCP Flags        :                               0
TCP Seq/Ack     :                               0/0
Pattern Type    :                               abcd...
TX Count/% Rate : Forever /100%
Pkt Size/Tx Burst: 64 / 128
TTL/Port Src/Dest: 64/ 1234/ 5678
Pkt Type:VLAN ID: IPv4 / TCP:0001
802.1p CoS/DSCP/IPP: 0/ 0/ 0
VxLAN Flg/Grp/vid: 0000/ 0/ 0
IP Destination  : 192.168.1.1
Source          : 192.168.0.1/24
MAC Destination : 00:00:00:00:00:00
Source          : 52:54:00:e8:69:ec
PCI Vendor/Addr : 1af4:1041:01:00.0
-- Pktgen 21.11.0 (DPDK 22.03.0-rc0) Powered by DPDK (pid:2072) -------

** Version: DPDK 22.03.0-rc0, Command Line Interface without timers
Pktgen:/> █

```

pktgen-dpdk提供了命令行界面，可以指定网卡的发送地址、开启发送、结束发送，如下所示：

1. 指定目的mac地址（0为VM1的发送端口, 52:54:00:C3:7F:EB 为VM2网卡的mac地址）

```
set 0 dst mac 52:54:00:C3:7F:EB
```

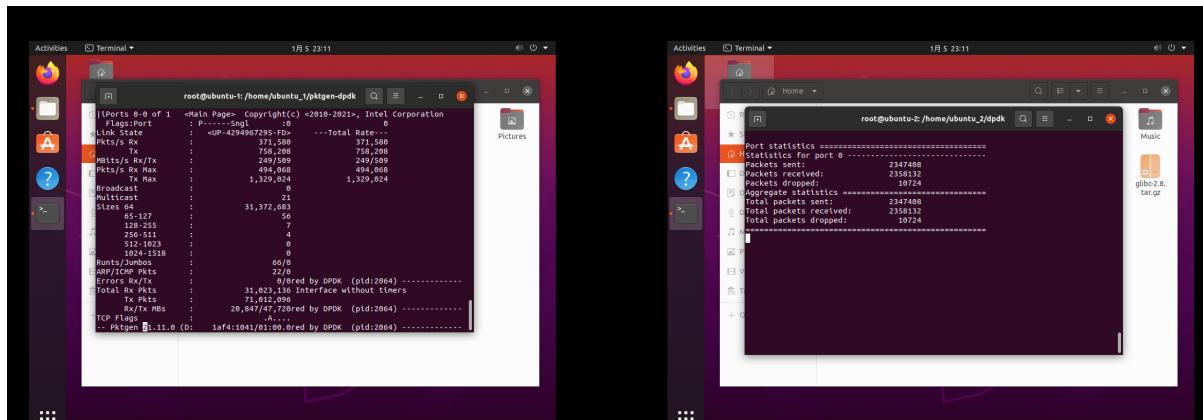
2. 端口 0 开始/停止 发包

```
start 0  
stop 0
```

五、评估DPDK的L2转发性能

VM1发包-VM2转发：确认可运行

使用如上所属方法分别在VM1上启动pktgen，并指定VM2的mac地址，开始发包；VM2启动 l2fwd 开始收包并转发。如下图所示，收发包运行成功。



测试条件

1. l2fwd

- 1 核
- 2 内存通道 / 核
- 1 队列 / 核

2. pktgen-dpdk

- 2 核
- 核 1 负责端口0的收发

性能测试过程

实际测试并未使用脚本记录中间数据，而使用同步测试方式，即先打开l2fwd收包，再开启pktgen发包，每隔10s打印信息，手动记录中间数据。

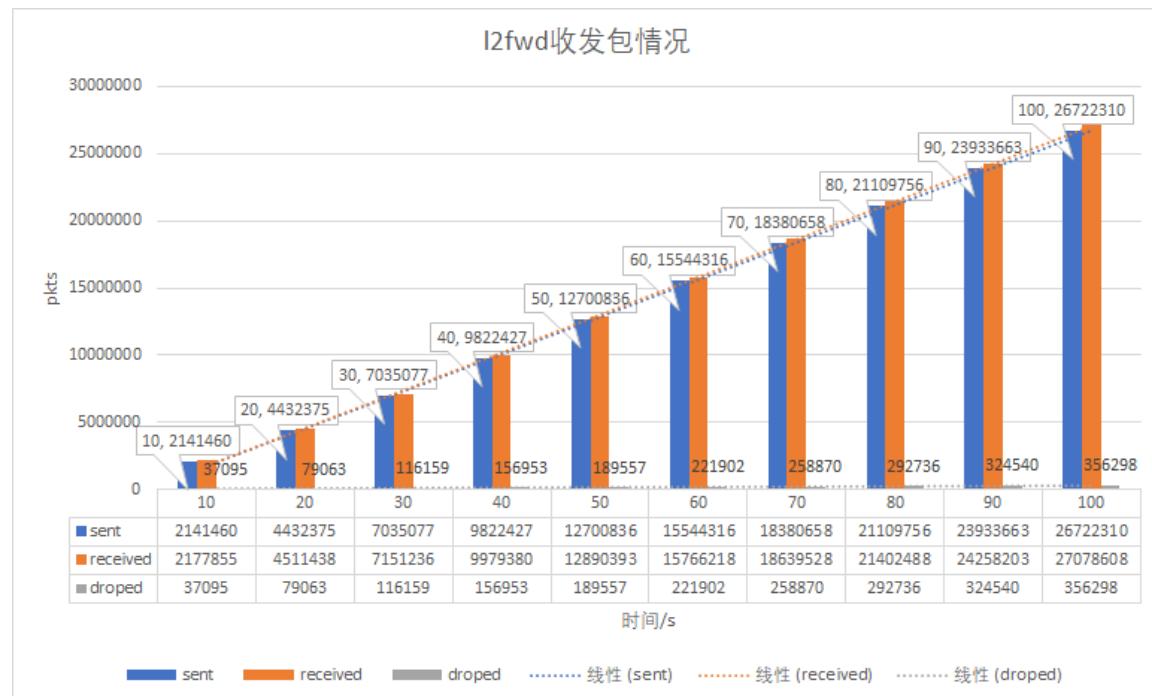
测试录屏：<https://jbox.sjtu.edu.cn/l/I11xSd>

原始数据：<https://jbox.sjtu.edu.cn/l/a10YBK>

测试结果

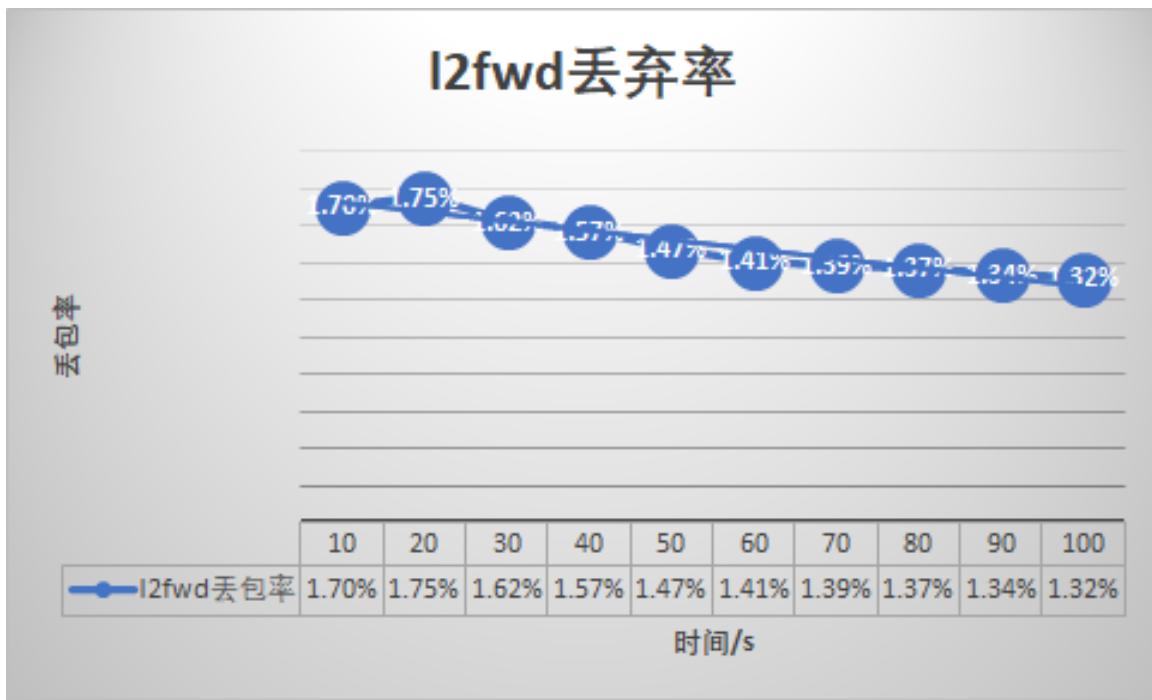
(一) l2fwd收发包情况

由下图可见，l2fwd平均每10s接受2766750个数据包，平均发送2672231。此处l2fwd存在丢包是因为virtio网卡存在自收发机制。



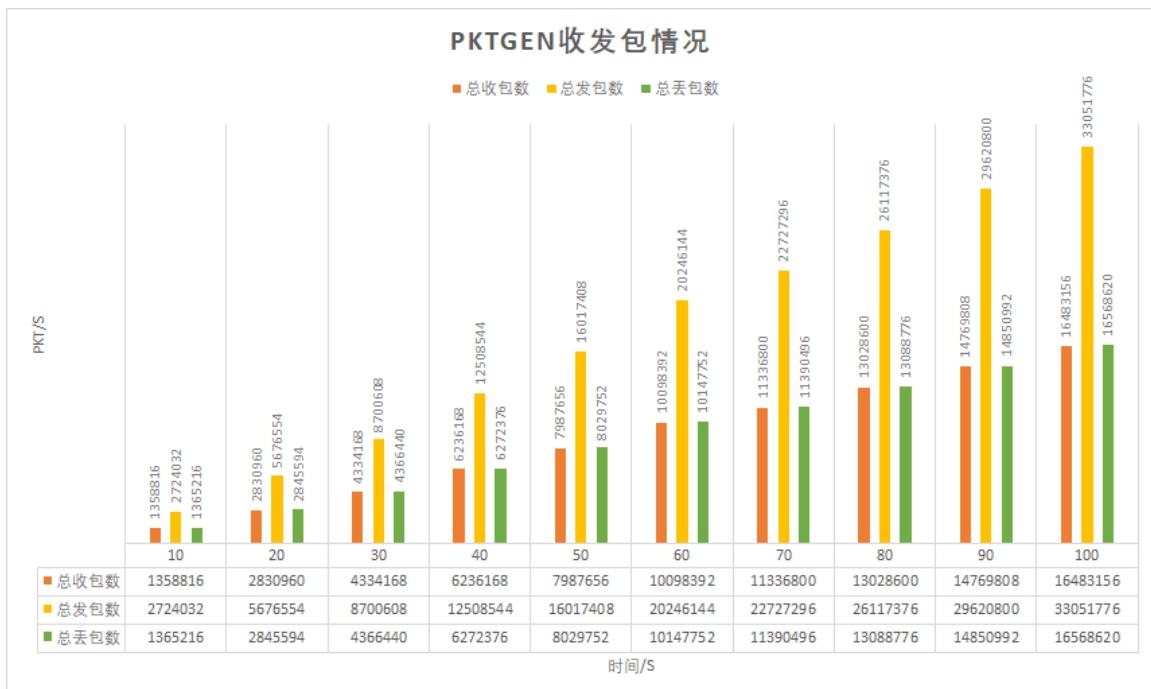
事实上，l2fwd的丢包率极低，平均1.49%。在后续衡量l2fwd的收发包速率时，将l2fwd的丢包率忽略不计，认为l2fwd收发包速率相等

l2fwd的丢弃率变化趋势如下图所示，随着接受、转发的包越来越多，丢弃率呈下降趋势。



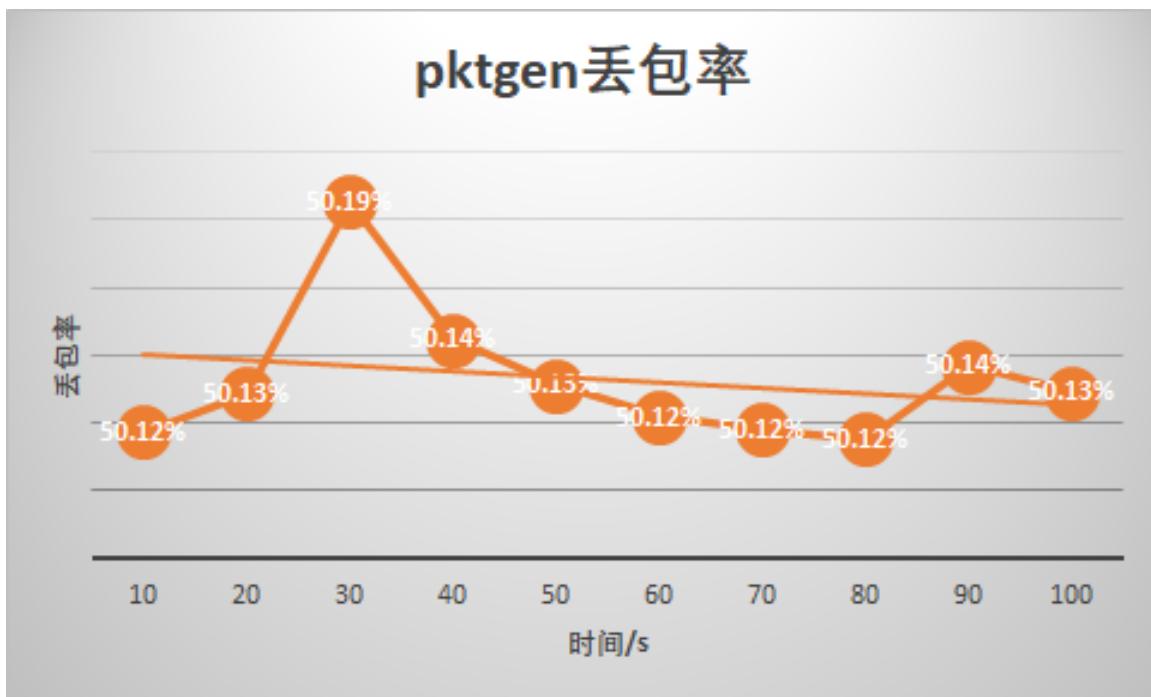
(二) pktgen收发包情况

pktgen-dpdk收发包情况如下图表所示。pktgen平均每10s发送数据包**3305178**个，接受数据包**1648316**个，丢弃数据包**1656862**个。平均丢包率**50.13%**，这一较高丢包率可以通过如下柱形图直观地看出。



pktgen的较高丢包率，是因为*l2fwd接受速率<pktgen发送速率与 pktgen接受速率<l2fwd转发速率*共同作用的结果（具体在下一部分分析）。

下图为pktgen丢包率（pktgen发送、接受，经过l2fwd转发）的变动曲线。丢包率恒定在 $50.13\% \pm 0.06\%$ ，此差异可忽略不计。



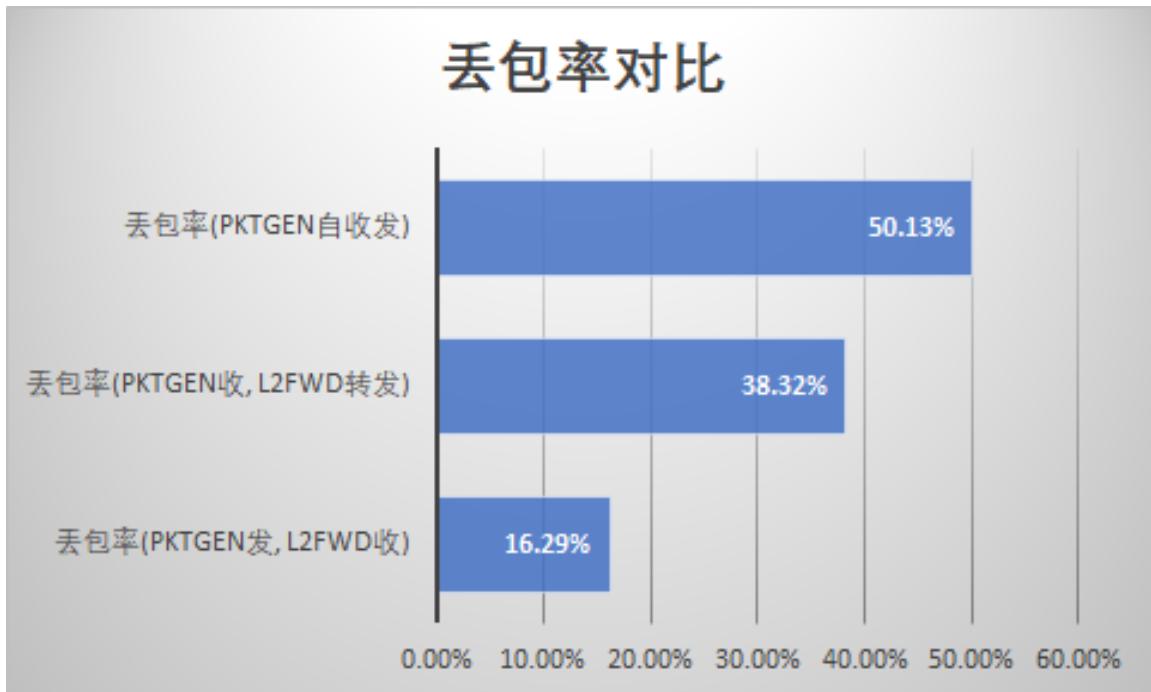
(三) pktgen与l2fwd之间收发性能对比

如下图所示，pktgen自收发平均丢包率为50.13%。这由两部分丢包组成：

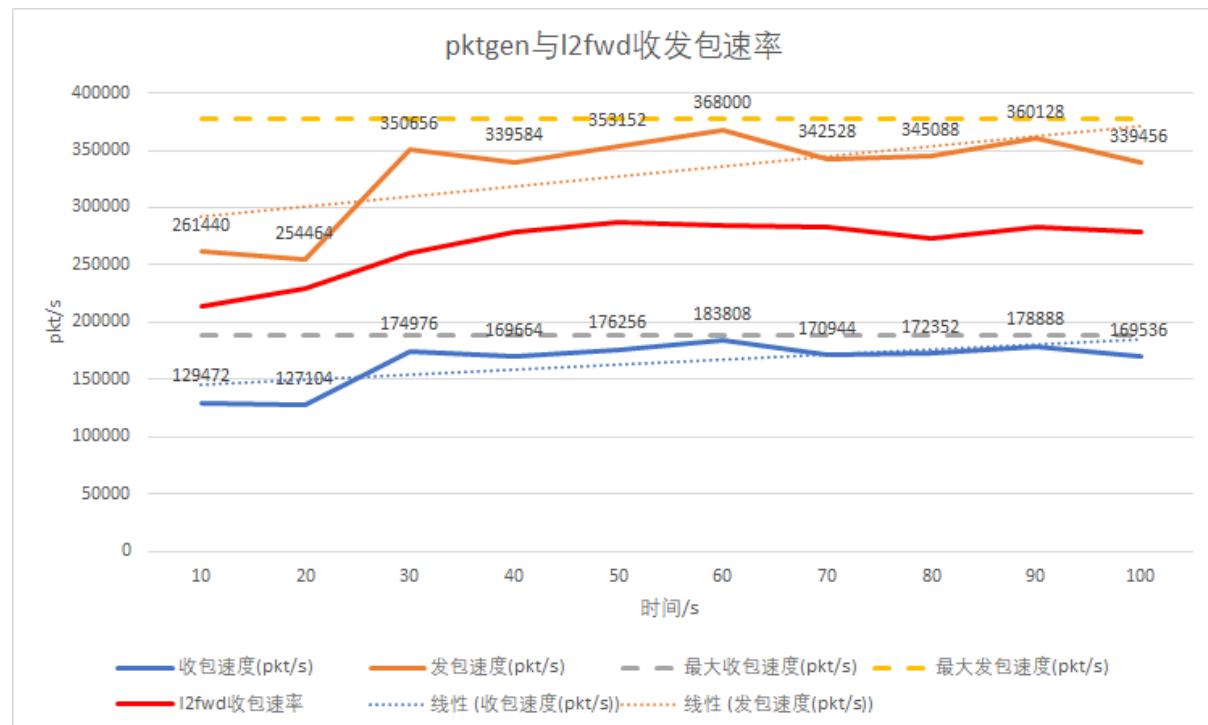
1. $\text{pktgen} \rightarrow \text{l2fwd}$, 2. $\text{l2fwd} \rightarrow \text{pktgen}$ ，丢包率分别为16.29%和38.32%。

存在丢包由三个原因：**收发速率不一致**，即pktgen使用较高的发送速率发包，而l2fwd以较低的速率收包。因此由大量数据包在队列中等待处理。此外缓冲队列有一个上限，超过队列容量的数据包会被丢弃。由于收发速率差异较大，因此缓冲区队列长时间处在满状态，大量数据包被丢弃。由于测试用仅只用一个缓冲队列，且pktgen和l2fwd均被分配了单核，因此丢包率较高。

丢包率对比



换言之，I2fwd的接受、转发速率是pktgen高速自收发的性能瓶颈，如下图所示。橙色线为pktgen发包速率，红色线为I2fwd转发速率（丢弃率忽略不计），蓝色线为pktgen收包速率。



六、感悟

通过本次实验，我进一步深化巩固了计算机网络与虚拟化网络的相关知识，通过测试体会了网络拥塞与收发速率对网络性能影响，并实操了网络主流网络性能测试工具DPDK与pktgen。