

LAB1 探究高效包处理平台 OpenNetVM 的原理与优化

实验目的：

在这个实验中，你将通过配置环境、发包测试吞吐、阅读与修改源码等，学习高效包处理平台 OpenNetVM 的原理与使用。通过亲身实践，你将会：

- ✓ 通过阅读 OpenNetVM 论文，了解 OpenNetVM 运作原理
- ✓ 熟悉 Pktgen、OpenNetVM 的使用
- ✓ 理解 OpenNetVM 的运作机制，探索其优点与不足
- ✓ 自行设计实验，对 OpenNetVM 做出性能调优

实验所需环境

推荐环境

- ✓ 2 个虚拟机。
- ✓ 其中 VM1 需要至少 2 个 core，而 VM2 需要至少 6 个 core（如果你仔细阅读了 [OpenNetVM 论文](#)，你将知道，OpenNetVM 本身需要 3 个 core 分别用于 statistics、RX thread 和 TX thread，而每个 NF 需要 1 个 core）。
- ✓ 虚拟机环境推荐使用 VMware + Ubuntu20.04

其他环境

使用 2 台服务器，每台服务器都至少需要 3 张网卡（2 张用于 DPDK 的网卡绑定），核数要求同上。

背景知识

云计算被视为计算机网络领域的一次革命，因为它的出现，社会的工作方式和商业模式都在发生巨大的改变。

云计算的本质，就是将计算资源从本地迁移到云端，实现“云化”。但是，如果只是简单地将服务器硬件搬到云端机房，只能叫做主机托管，缺乏足够的灵活性和效率。由此，云计算引入了虚拟化技术，即 NFV（network function virtualization，网络功能虚拟化技术）。

NFV 的核心思想是将网络硬件设备通过虚拟化技术，集成到通用的 x86 架构的服务器或者其他硬件平台上。然后在通用标准的硬件平台上，执行路由器、交换机、负载均衡、防火墙、入侵防御等功能。采用 NFV 技术，可以实现软件和硬件的彻底解耦。运营商不再需要购买厂商们制造的专用硬件设备，大幅降低了硬件资金投入。NFV 还具备自动部署、弹性伸缩、故障隔离和自愈等优点，可以大幅提升网络运维效率、降低风险和能耗。

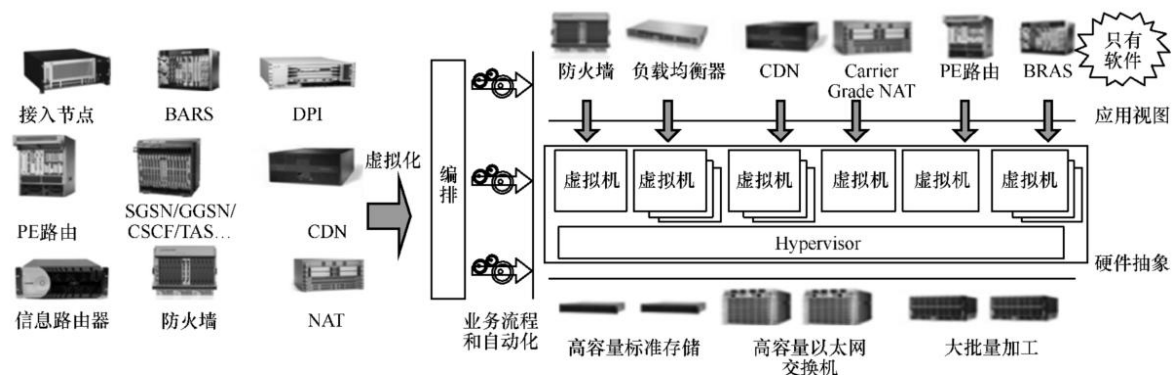


图 1 NFV 示意图

NFV 有 5 个最主要的组成部分，分别是 VNF（virtualized network function）、NFVI（NFV Infrastructure）、NFVO（NFV Orchestration）、VNFM（VNF manager）、VIM（Virtual Infrastructure）。不同的 VNF 可以部署在同一台服务器内，也可以跨多台服务器部署，并可以进一步连接以组成复杂的服务链。

我们此次实验会将多个 VNF 部署在一台虚拟机上，并使用高性能 NFV 平台 OpenNetVM 对 VNF 进行管理，使 VNF 之间、VNF 与网卡间实现高性能通信。

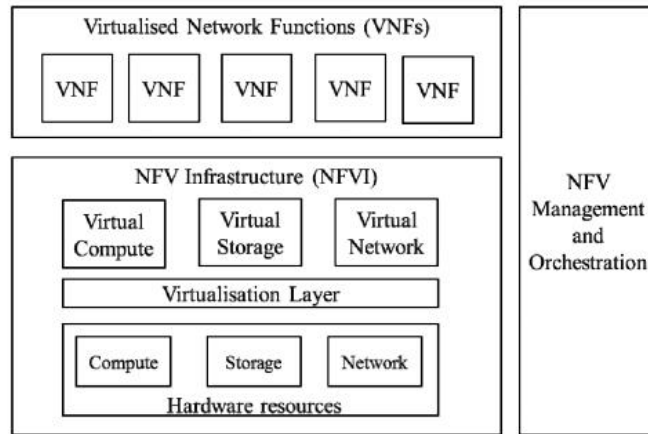


图 2 NFV 架构示意图

工具介绍

OpenNetVM - 基于 DPDK 和 Docker 容器的高性能 NFV 平台，支持通过服务链动态控制包。

Pktgen - 高速率的发包工具。

配置实验环境

本次实验中，你将配置如下图所示的实验环境：

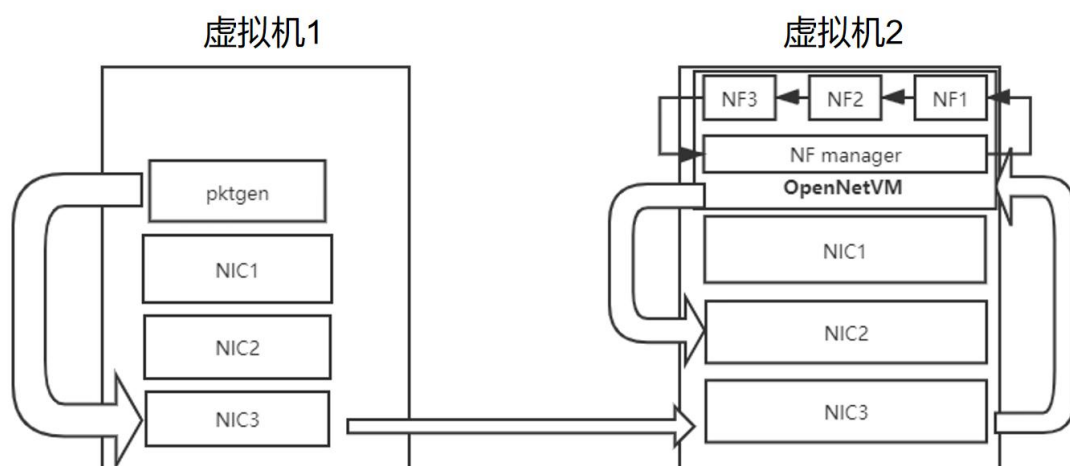


图 3 实验环境示意图（NIC1 用于联网）

在接下来的练习中，你将会使用 **pktgen** 按照事先设定好的参数生成数据包，

通过虚拟机 1 的一张网卡发往虚拟机 2，虚拟机 2 收到后，OpenNetVM 将其发送给目标 NF，目标 NF 会将其丢弃、发给网卡或传递给下一个 NF。你可以依据 OpenNetVM 报告的数据得知其吞吐量和丢包率，结合设置的包的大小，还可以计算出带宽。

在本次实验中，Service chain 的最大长度（即 NF 的数量）为 3。我们建议你先搭建单个 NF 的 service chain，测试成功后，再增加 NF 的数量。

注意，本实验安装环境的中文教程较少，请仔细阅读官方文档，认真理解各类命令与参数，尽早开始安装！！

OpenNetVM

从 [sdnfv/openNetVM: A high performance container-based NFV platform from GW and UCR. \(github.com\)](https://github.com/sdnfv/openNetVM) 下载 OpenNetVM 的 master 分支代码，参考 github 上的官方文档，学习其安装和使用。

注意：OpenNetVM 基于 DPDK，需要绑定两张网卡，在网卡被绑定前，它需要被设置为不活跃状态，这在安装文档中也有提到，具体步骤如下：

首先添加两张虚拟网卡，添加好后如图：



图 4 网络适配器 2/3 均为虚拟网卡 网络适配器用于联网

使用 `ifconfig` 查看网卡，`ifconfig xxx down` 将对应的两张网卡调至不活跃状态。

使用 `./dpdk/usertools/dpdk-devbind.py -s` 查看 NIC PCI device ID。

另外需要注意的是，如果你使用的是服务器，需要保留一张网卡用于你与服务服务器之间的连接。

NF

在 service chain 长度为 1 时，推荐选择 OpenNetVM 自带的 bridge 程序作为 NF。它只实现了简单的转发功能。具体请参考 [openNetVM/examples/bridge at master · sdnfv/openNetVM \(github.com\)](#)。

在 service chain 长度大于 1 时，请参考 [openNetVM/Examples.md at master · sdnfv/openNetVM \(github.com\)](#) 的 Linear NF Chain 进行配置。

我们推荐先使用 bridge 搭建单 NF 环境，在测试成功后，再对 service chain 进行扩展。注意，默认情况下 OpenNetVM manager 会将包首先发给 service ID 为 1 的 NF。

Pktgen

PKTGEN 有两种形式，一种是直接由 linux 系统自带的内核模块进行发包，另一种是依赖于 dpdk 的 pktgen，即 pktgen-dpdk。我们推荐你使用后者。另外，OpenNetVM 也是依赖于 DPDK 的，所以你可以直接使用 OpenNetVM 里的 pktgen-dpdk，参考 [Packet generation using Pktgen · sdnfv/openNetVM Wiki \(github.com\)](#) 以及 [Pktgen command line directory format — Pktgen 3.2.4 documentation \(pktgen-dpdk.readthedocs.io\)](#)

连接

通过正确设置 pktgen 的 dst_mac 和 dst_ip，两个虚拟机之间就能成功发包和接收，你将看到类似下图的数据显示，意味着我们的实验环境已基本就绪。如果你不太顺利，也可以参考 [DPDK PKTGEN 使用 - 简书 \(jianshu.com\)](#)

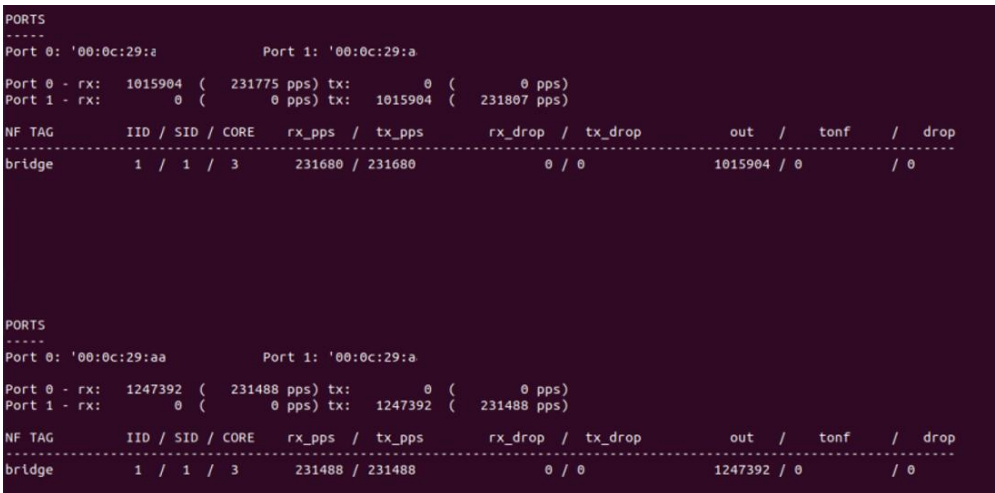


图 5 虚拟机 2 单个 NF 运行中的 OpenNetVM 数据统计

```
APP: Shutting down...
NF Activity summary
-----
NF tag: bridge
NF instance ID: 1
NF service ID: 1
NF assigned core: 3
-----
RX total: 1600327
RX total dropped: 0
TX total: 1591823
TX total dropped: 8504
NF sent out: 1600327
NF sent to NF: 0
NF dropped: 0
NF next: 0
NF tx buffered: 0
```

图 6 虚拟机 2 单个 NF 运行结束后的 NF 数据统计

思考与练习

Part 1 性能测试（5 分）

按照以上步骤安装好环境，将包的大小分别设置 64/128/256/512/1024 byte（你也可以添加其他你感兴趣的数值），在 NF 为 bridge, pktgen rate 分别为 10%、100%的情况下，测试出 OpenNetVM 稳定时的吞吐量，作出 2 张 packet size - throughput 折线图（3 分，折线图和截图要求请见“提交要求”，下同），并分析折线图趋势和造成该趋势的可能原因（2 分）。

通过这个练习中，你应当熟悉 pktgen、OpenNetVM 及其 NF 的使用。

Part 2 服务链性能（4 分）

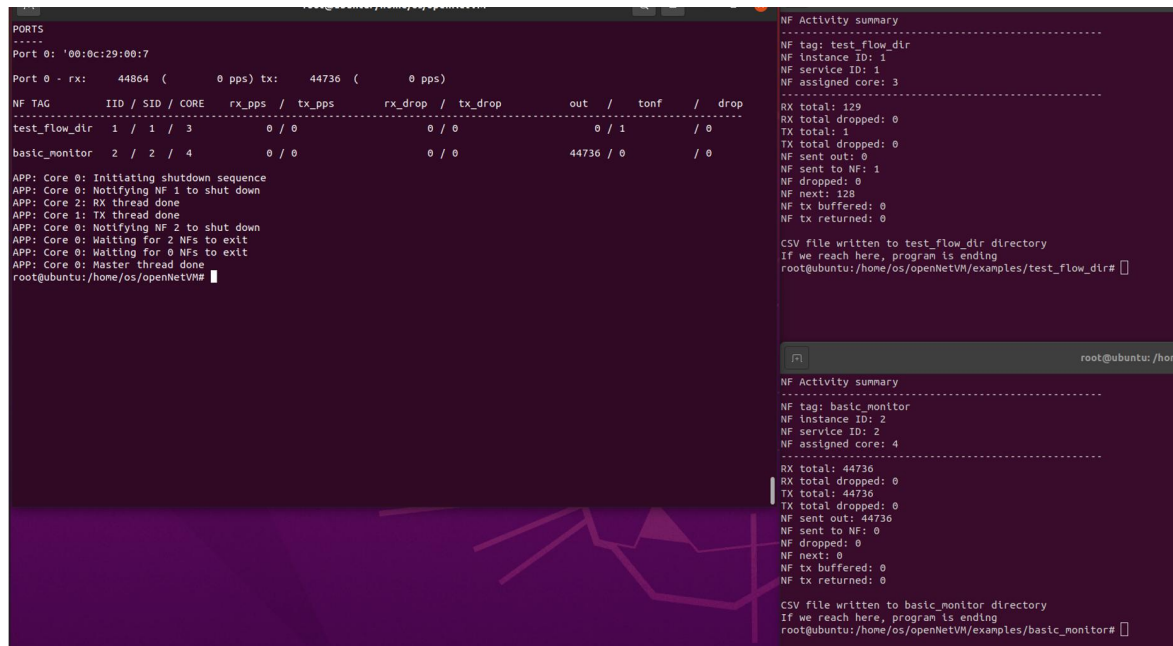
将包的大小设置为 64byte，在 NF 数量为 1/2/3 的情况下，统计 OpenNetVM 稳定时的吞吐量，作出 NF number-throughput 折线图（3 分），并分析折线图趋势和造成该趋势的可能原因（1 分）。

在这个练习中，你应当熟悉使用 OpenNetVM 搭建 Linear Chain。注意，教程某命令处多了个‘-d’，若报错请自行修改。

Part 3 流表维护（7 分）

在这个 part，我们需要建立起一个 test_flow_dir->basic_monitor 的服务链。当第一个包传递给 test_flow_dir 时，test_flow_dir 会将其五元组和目标 VNF（即 basic_monitor）的对应关系记录在 flow table 中，随后的时间里，OpenNetVM 每次收到包，都会在 flow table 中查询到该项（注意我们只有一个发包来源），并

将包转发给 basic_monitor，如图 7。



```
PORTS
-----
Port 0: '00:0c:29:00:7'

Port 0 - rx: 44864 ( 0 pps) tx: 44736 ( 0 pps)

NF TAG      IID / SID / CORE  rx_pps / tx_pps  rx_drop / tx_drop  out / tonf / drop
-----
test_flow_dir 1 / 1 / 3          0 / 0          0 / 0          0 / 1 / 0
basic_monitor 2 / 2 / 4          0 / 0          0 / 0          44736 / 0 / 0

APP: Core 0: Initiating shutdown sequence
APP: Core 0: Notifying NF 1 to shut down
APP: Core 2: RX thread done
APP: Core 1: TX thread done
APP: Core 0: Notifying NF 2 to shut down
APP: Core 0: Waiting for 2 NFs to exit
APP: Core 0: Waiting for 0 NFs to exit
APP: Core 0: Master thread done
root@ubuntu:/home/os/openNetVM#
```

```
NF Activity summary
-----
NF tag: test_flow_dir
NF instance ID: 1
NF service ID: 1
NF assigned core: 3
-----
RX total: 129
RX total dropped: 0
TX total: 1
TX total dropped: 0
NF sent out: 0
NF sent to NF: 1
NF dropped: 0
NF next: 128
NF tx buffered: 0
NF tx returned: 0

CSV file written to test_flow_dir directory
If we reach here, program is ending
root@ubuntu:/home/os/openNetVM/examples/test_flow_dir#
```

```
NF Activity summary
-----
NF tag: basic_monitor
NF instance ID: 2
NF service ID: 2
NF assigned core: 4
-----
RX total: 44736
RX total dropped: 0
TX total: 44736
TX total dropped: 0
NF sent out: 44736
NF sent to NF: 0
NF dropped: 0
NF next: 0
NF tx buffered: 0
NF tx returned: 0

CSV file written to basic_monitor directory
If we reach here, program is ending
root@ubuntu:/home/os/openNetVM/examples/basic_monitor#
```

图 7 test_flow_dir->basic_monitor 服务链

理论上来说，只有第一个包会需要被发往 test_flow_dir 用于流表的设置，剩下的包都应在查表后发往 basic_monitor，但事实并非这样。从图 7 可以看到，有部分的包被标记为了 NF next，并且这部分包没有被发往 basic_monitor。当你将 pktgen 的发包数量和速率设置合适时，甚至会出现没有包发给 basic_monitor 的情况。

请尝试进行以上设置，并给出 basic_monitor 一个包也没收到时的截图（1 分）。

调试程序，仔细观察输出的数据，并参考源代码，解释以上现象的出现原因（2 分）。

最后，修改代码，使得这部分被标记为 NF next 的包，至少可以正常地发给目标 NF。你可以试试在 test_flow_dir.c 下的 packet_handler 函数中直接设置好 packet 的 action 和 destination，也可以尝试其他思路。唯有一点：你的代码尽量不要使用常数（包括将 action 直接设置为 ONVM_NF_ACTION_TONF，destination 直接设置为一个常数等），你的 action 和 destination 数据都应从之前设置好的 flow_entry 中读出。

此题没有标准答案，言之有理即可，重在思路，代码修改应在 5 行左右，不应超过 50 行，不需要考虑过于复杂的情况。分析与思路：2 分（部分正确：1 分），代码：2 分（部分正确：1 分）。

拓展（8 分）

经过前述的练习，相信你已经对 OpenNetVM 的原理有了一定的理解与思考。在拓展部分，你应当根据自己的思考，从任何方面对 OpenNetVM 进行改进，如：优化 flow table，在其他条件相同的情况下提高吞吐或带宽等等。

评分标准如下（部分正确均得部分分数）：

- ✓ 清晰完整的优化思路：3 分
- ✓ 修改后的代码：3 分
- ✓ 优化效果：2 分

在批改时，我们将优先查看思路和代码，而非优化效果，与思路和代码不匹配的优化效果将不会得分。此题没有标准答案，有理有据，基本理解正确即可。如果你的优化效果甚微，请仔细分析原因，同样可以得到这部分的分数。禁止抄袭他人的优化方案，若发现，以 0 分计。

提交要求

- 提交格式：zip 或 7z 压缩包，命名为 学号_姓名_lab1
- 压缩包内容：
 - 实验报告.pdf，包括每题的思路和要求的绘图、截图
 - 修改过的代码文件，使用.c 或.h 格式，放在一个文件夹下，并使用一个 README 文件简要描述每个文件都进行了哪些修改，修改了哪几行。
- 折线图要求：在实验报告末尾附上要求作出的各折线图中各数据点的 OpenNetVM 统计截图（参考图 8，请标注截图都是属于哪张折线图）。
- 截图要求：每一张截图必须带有时间（包括折线图要求中的数据点截图），电脑右下角或虚拟机正上方的时间均可（如图 8）。

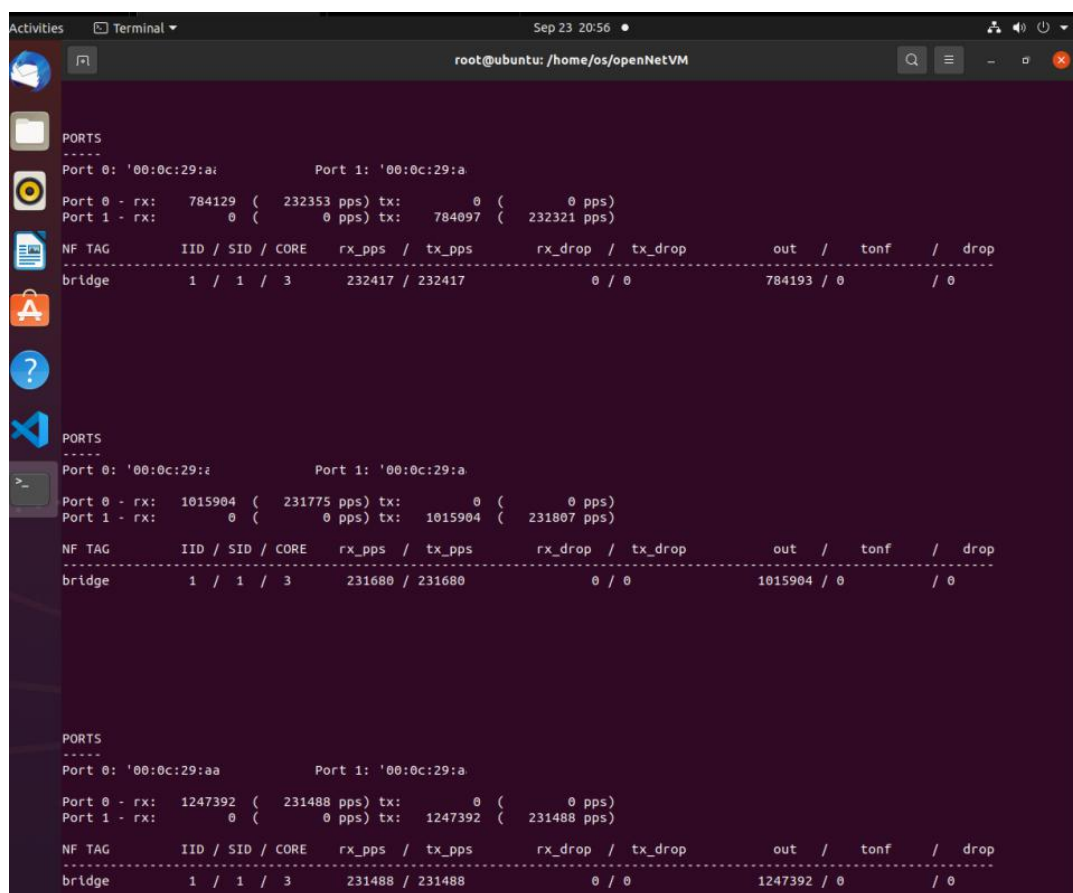


图 8 OpenNetVM 吞吐量稳定时的带时间统计报告截图

分数说明

总分 24 分，满分 20 分，超出 20 以 20 计，请合理分配时间。
另外，平时实验分数可能会按照分数分布最终成立比统计。

推荐阅读

[OpenNetVM: A Platform for High Performance Network Service Chains](#)

了解 OpenNetVM 基本原理