VFD大揭秘，一定有你想知道的！

原创 DPDK开源社区 2017-04-28

作者 张祺

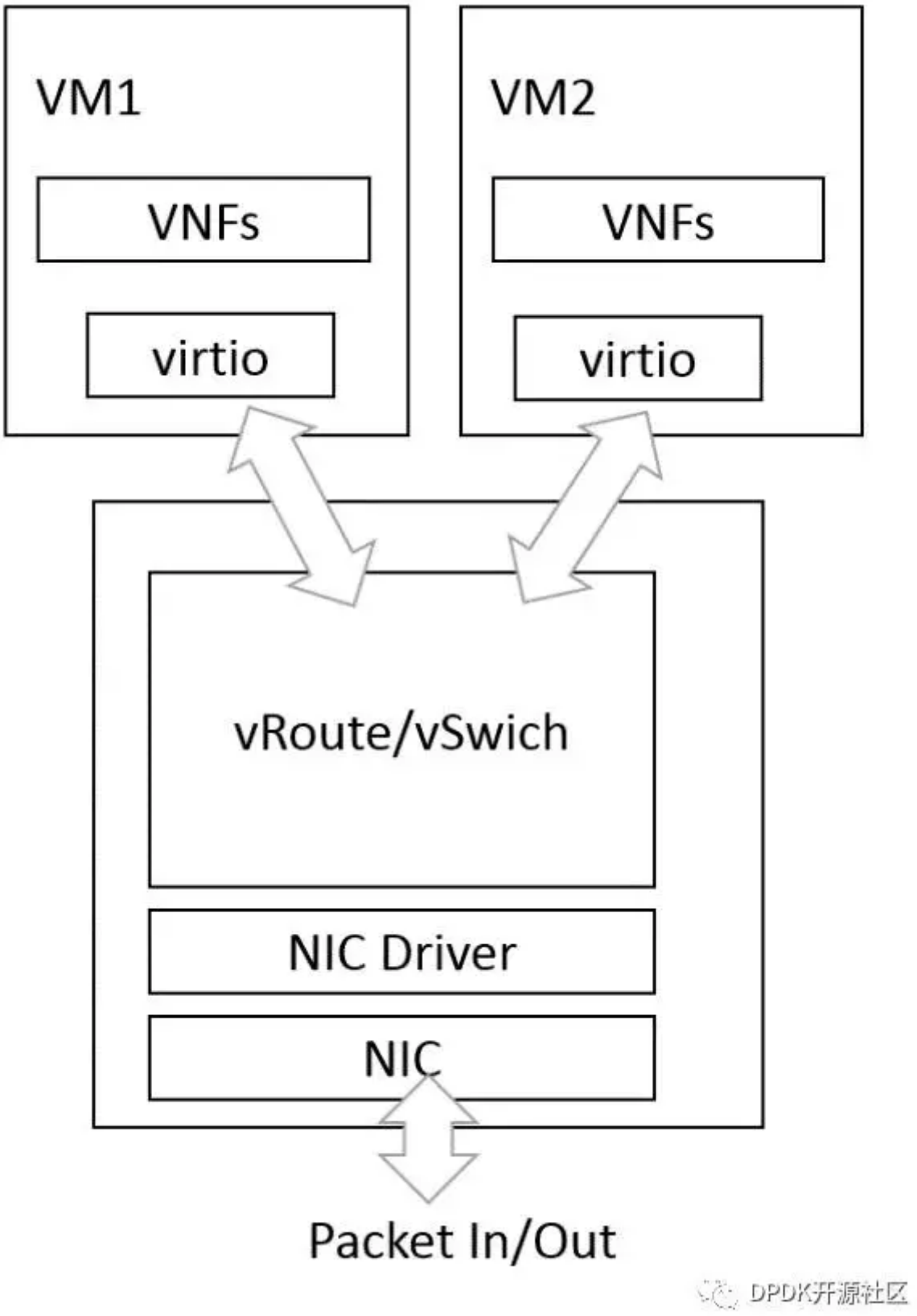
什么是VFD?

VFD 的全称是Virtual Function Daemon。在网络虚拟化（NFV）环境下，VFD被用来配置网卡上的虚拟端口。 VFD 由AT &T 公司开发，具有开源软件许可Apache License, Version 2.0， 2016年4月在GitHub上发布。（https://github.com/att/vfd）

VFD的相关背景

要说明VFD的由来，得先回顾一下网络功能虚拟化（NFV）的背景。

在经典的NFV架构中，虚拟交换机(vSwitch)是整个架构的核心，它同时和虚拟机上的虚拟网卡以及主机上的物理网卡相连，成为数据面（data plane）的中间枢纽，同时接受来自控制面（control plane）对包流表的配置。



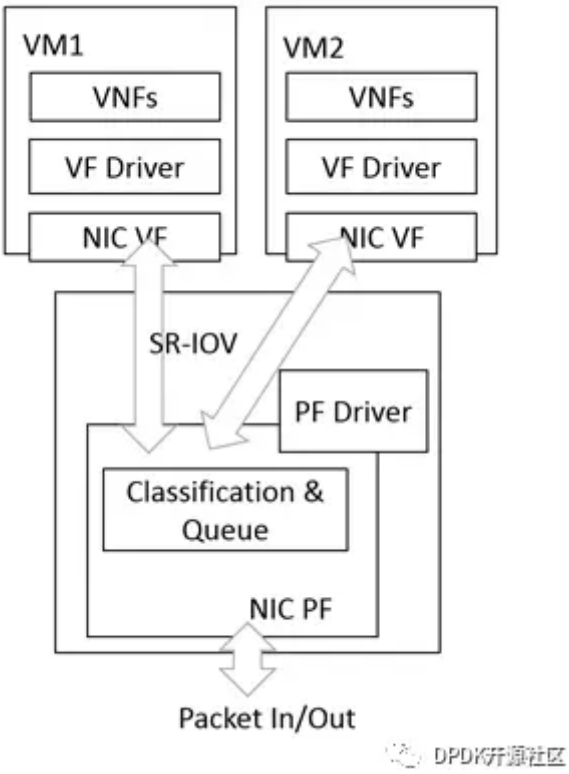
经典架构的优势是，它非常的灵活，vSwitch是纯软件实现，不依赖于具体硬件，可以随意创建虚拟端口，轻松实现类似OpenFlow的标准协议用来配置流表，同时基于virtio的VM可以随意的在不同平台上迁移。但是经典架构的最大问题则是性能，网络包从服务器外部进入虚拟机需要经过主机端的网卡驱动，vSwitch，然后通过共享内存进入虚拟机virtio驱动，最终到达应用程序，其间需要经过多次的内存copy，而性能问题已经是目前商用NFV不可忽视的一个问题。

针对NFV的性能提升，可以分为主机端和虚拟机端两部分。在虚拟机中，可以用DPDK的用户态的virtio PMD来做加速，内核协议栈被绕过，并减少了数据从内核空间到用户空间的复制。

而在主机端，目前比较流行的是两种方法:

第一种方法是通过DPDK对vSwitch 进行加速（如：OVS-DPDK是最为典型的开源实现），其原理与以上虚拟机中的方法一致。这种方法的意义是，即提升了整体性能又不失灵活性。但是这也无法避免在将数据包传入虚拟机中时，vSwitch需要将整个包的内容copy到Guest Memory这一过程，在需要大吞吐量的情况下，这仍然会是个问题。

第二种方法则是基于SR-IOV的硬件加速，虚拟机中的应用可以通过SR-IOV直接访问网卡上的数据，从而达到极致的传输性能。基于SR-IOV的NFV架构如下图：



由于vSwitch被绕过，包的传输路由完全依赖于对网卡本身所支持的分流和过滤功能的配置，因此路由功能简单是这种方法无法避免的缺点之一。另外，由于目前SR-IOV在虚拟机端不像virtio有通用的驱动，每种网卡都需要有各自的专有VF驱动，这也使得VM的迁移成为问题，不过相关标准正在制定之中，相信在不久的将来，这一问题可以得到解决。

为什么需要VFD?

VFD主要为第二种方法服务。

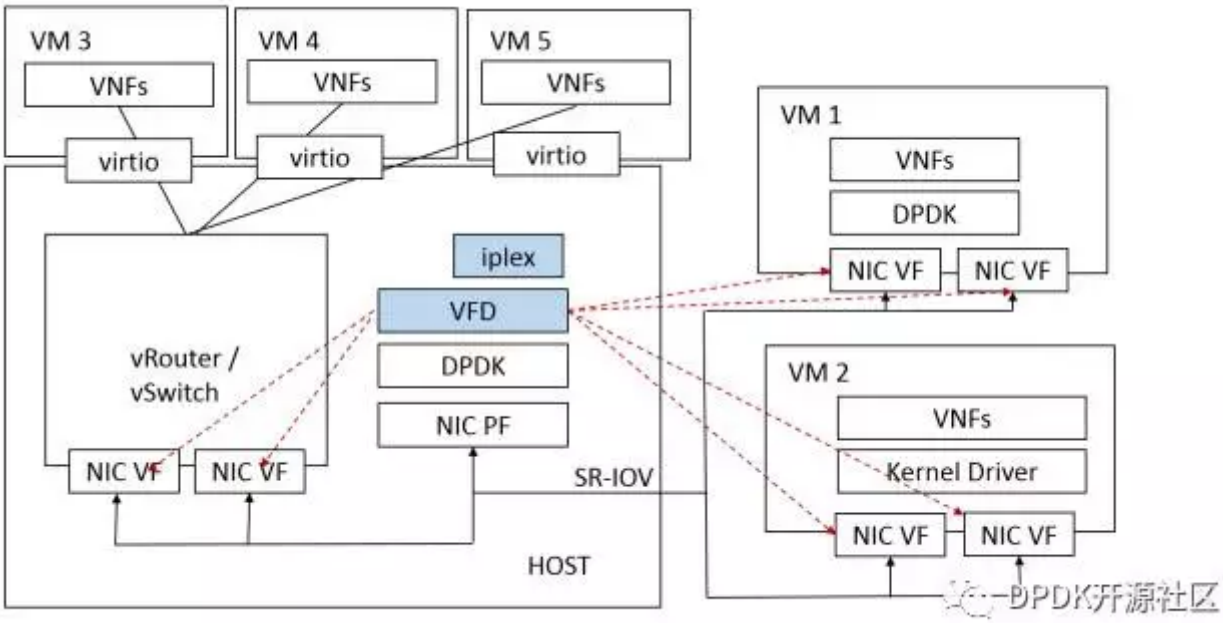
部署基于SR-IOV的NFV时会遇到的一个比较现实的问题：如何高效的配置这些虚拟端口。（例如，添加MAC地址, 设置VLAN过滤, 设置QoS带宽，MPLS插入/剥离等等）？自然而然我们会想到应该有这样的一个工具，它一方面能够提供标准的接口接受用户发送的命令，同时又能够引导不同网卡的驱动程序来完成这些任务。

与配置vSwitch上的纯软件实现的虚拟端口不同，配置基于SR-IOV的虚拟端口需要通过网卡驱动的帮助。支持SR-IOV的网卡通常会有PF驱动和VF驱动。而VF驱动运行在虚拟机里通常是被限制直接对硬件直接进行设置的，所以需要由PF驱动来完成，基本上所有支持SR-IOV的网卡都会有Linux内核PF驱动，同时Linux提供了一些网络接口管理工具 (如iproute2，net-tools等等)，这些工具定义了配置VF的标准接口，只要内核网卡驱动实现这些接口，就能够通过简单的命令行来完成配置任务。然而随着网卡虚拟端口所提供的功能日益增多，可配置的内容也越来越多，越来越复杂，这就对配置工具提出了更易于扩展，更易于更新的要求。

基于内核驱动来配置虚拟端口的方式受限于Linux内核的网络软件框架，更新成本相对较高，更新周期也相对较长。而随着DPDK功能不断完善，将DPDK作为PF端管理入口来配置VF端口已变得可行。由于DPDK是用户态以及发布周期短，基于DPDK的配置工具会变得更为灵活，更能够适应需求不断变化的环境, 而VFD便是其中的一次尝试。

VFD的应用场景

下图描述了VFD的应用场景。



在一台服务器上，某物理网卡PF由DPDK来管理。

由PF分出VF通过SR-IOV给VM1和VM2，在VM1和VM2上，可以根据所部署的应用特征，选择使用内核驱动（通常在需要用到内核协议栈的情况下）还是直接由DPDK来驱动VF。在这里，VFD是作为一个DPDK的应用部署在主机端，VFD可以同过网卡的DPDK PF驱动来对虚拟机上的VF进行配置。

当然，我们前面也提到了SR-IOV的局限性，有时候vSwtich并不能被完全绕开。好在使用SR-IOV并不限制vSwitch的部署，我们可以将两者结合在一起，将由SR-IOV分出的另一部分VF可以作为vSwitch的外部接口使用，我们看到一块物理网卡可以做到同时支持基于SR-IOV的NFV和基于vSwitch的NFV，而对于所有VF的管理, VFD在这里起到了关键的作用。

VFD的软件结构

VFD由一个后台程序(Daemon)和若干前台命令行工具组成，其中最重要的工具是iplex，对于VF的配置主要通过它来实现。

iplex是一个由python实现的小程序，它能够解析命令行然后转换成JSON格式的命令包,并通过Linux有名管道发送给后台程序，具体实现可参见源码：

https://github.com/att/vfd/blob/master/src/system/iplex

后台程序由C语言编写，它真正实现了VF的管理和配置，当然没有DPDK的支持，这一切也无法完成，所有对DPDK的封装在sriov.c这个文件中实现。

https://github.com/att/vfd/blob/master/src/vfd/sriov.c

在接收到来自有名管道的命令后，后台程序会根据要求去调用相应的DPDK API。

VFD还有一个重要的特性是对OpenStack的支持，VFD可以作为VF Agent来兼容nova, neutron, heat。具体可参照文档

https://github.com/att/vfd/tree/master/virt/openstack-kilo-mos

VFD的具体功能

在GitHub上已发布的VFD所提供的功能相对比较简单，而且只支持Intel 82599网卡，而在最新发布的DPDK17.02中，我们看到不仅加入了Intel i40e 网卡对VFD的支持，而且比原来82599还扩充了更多的功能，以下列出了一些功能（具体请查看DPDK 17.02的release notes）

· MAC 地址添加/修改/删除
· 单播/多播/广播的使能
· VLAN 过滤的设置
· VLAN RX 剥除/ TX插入
· Anti-spoofing
· TX 环回使能
· 收发包数据统计/清除
· Rate Limit 设置

另外在正在卡发的DPDK17.05中，我们还会看到更多新的功能（QoS，MPLS，QinQ等等），而这些功能都会在VFD的后续版本中被集成进去。

VFD的Roadmap

VFD是由AT &T 发起和维护的项目，目前并没有公开的Roadmap，但我们仍然可以在GitHub上看到一些关于VFD2.0的计划。

VFD2.0最大的亮点是引入了对QoS的支持，用户可以创建最多4个TC(Traffic Class)，对于每个TC，可以设置其优先级和带宽，并可将VF上RX/TX队列分配给它。

同时VFD2.0也会提供更为直观的系统管理工具和更为细化的系统配置方法，具体可参考以下文档：

https://github.com/att/vfd/blob/master/doc/qos/vfd_features_1.md

VFD对于DPDK的意义

VFD作一个配置工具，其本身还是相对比较简单的，但是他对DPDK的意义却非同一般，传统意义上，DPDK是用来为数据面加速的，而VFD则是将DPDK作为控制面的后台引擎部署在NFV中，这是前所未有的。应该说，在NFV这个大舞台中， VFD给DPDK带来了一个新的角色。

作者简介

张祺，英特尔软件工程师，主要从事DPDK PMD的开发工作

往期文章

- DPDK Release 17.02
- Generic Flow API简介
- 关于DPDK Cryptodev，你不得不明白的几点！
- 玩物志 | 什么！DPDK在盒子里？
- DPDK中的memcpy性能优化及思考
- 无锁队列详细分解 — 顶层设计
- VMware Player 搭建DPDK实验平台
- Qemu/Kvm 搭建DPDK实验平台
- 技术贴：利用DPDK加速容器网络
- DPDK IP分片与重组设计实现
- DPDK流分类优化，不得不知道的事~

长按扫描二维码关注我们

投诉