DPDK vhost-user详解

DPDK社区 DPDK与SPDK开源社区 11月27日

这篇文章是对vhost-user / virtio-pmd架构的深入技术研究，该架构针对基于DPDK的高性能用户空间网络，面向所有有兴趣了解这些基本细节的架构师和开发人员。

介绍

本文将在读者熟悉vhost-net架构的基础上展示使用vhost-net协议将网络处理从qemu移出并移入内核驱动程序的好处。在本文中，我们将更进一步，展示如何在客户机和主机上使用DPDK（数据平面开发包）将数据平面从内核中移出，并移入用户空间来提高网络性能使用。为实现此目标，我们还将详细研究vhost协议的新实现：vhost-user库。

读完本文，您应该会对基于vhost-user / virtio-pmd的架构中涉及的所有部分有深刻的了解，并且了解其使性能显著提升的原因。

DPDK及其优势

您可能已经听说过DPDK。该用户空间快速数据包处理库是许多网络功能虚拟化（NFV）应用程序的核心，它可以绕过内核的网络协议栈 ，完全在用户空间中实现这些应用程序。

DPDK是一组用户空间库，使用户可以创建优化的，高

性能的数据包处理应用程序，它具有许多优势，功能强大，在开发人员中非常受欢迎，下面列举了DPDK的一些优势：

· 多核亲和性——DPDK将每个不同的线程固定到特定的逻辑核心，以最大程度地提高并行性。

· 大页支持——DPDK具有多层的内存管理层（例如Mempool库或Mbuf库），但是在后台，所有内存均使用mmap分配在hugetlbfs中。使用2MB甚至1GB大页，使DPDK减少了缓存丢失和TLB查找。

· 无锁环形缓冲区——DPDK的数据包处理基于Ring库，Ring库提供了高效的无锁环形队列，该队列支持突发入队和出队操作。

· 轮询模式驱动——为避免中断开销，DPDK中提供了轮询模式驱动（PMD）抽象规范。

· VFIO支持——VFIO（虚拟功能I / O）提供了一个用户空间驱动程序开发框架，允许用户空间应用程序通过将I / O空间直接映射到应用程序的内存来直接与硬件设备进行交互。

除了这些功能之外，DPDK还支持其他两项技术，这些技术为我们提供了可大大提高云环境中网络应用程序性能的工具，它们是：

· Vhost-user库——实现vhost协议的用户空间库。

· Virtio-PMD——建立在DPDK的PMD抽象规范之上，virtio-pmd驱动程序实现了virtio规范，并允许以标准且有效的方式使用虚拟设备。

◦ ◦（ DPDK和OVS：完美结合 ）◦ ◦

DPDK的巨大优势之一是带来了已经广受欢迎的Open vSwitch(OVS-DPDK)的性能提升。Open vSwitch是一种功能丰富的多层分布式虚拟交换机，被广泛用作虚拟环境和其他SDN应用程序的主要网络层。

传统上将其分为基于快速内核的数据路径（fastpath），由流表和较慢的用户空间数据路径（slowpath）组成，后者处理与快速路径中的任何流都不匹配的数据包。通过集成OVS与DPDK，快速路径同样在用户领域，最大限度地减少了内核与用户领域的交互，并充分利用了DPDK提供的高性能。结果是，使用DPDK的OVS与原始OVS相比，性能提高了约10倍。

那么，如何将OVS-DPDK的功能和性能与基于virtio的架构相结合呢？在下一节中，我们将逐步为您介绍各个重要部分。

（DPDK中的Vhost-user库）

vhost协议是一组消息和机制，旨在将virtio数据路径处理从QEMU卸载到外部元素（配置virtio环并进行实际的数据包处理的处理程序），最重要的机制是：

· 一组消息，允许QEMU将virtqueue的内存布局和配置发送到处理程序。

· 一对eventfd类型的文件描述符，允许客户机绕过QEMU，并直接向处理程序发送通知或从处理程序接收通知：Available Buffer通知（从客户机发送至处理程序以指示有准备好处理的缓冲区）和Used Buffer通知（从处理程序发送给客户机以指示其已完成对缓冲区

的处理）

现在我们介绍vhost-user库。该库内置于DPDK中，是vhost协议的用户空间实现，可让qemu将virtio设备数据包处理工作卸载到任何DPDK应用程序（例如Open vSwitch）。

vhost-user库和vhost-net内核驱动程序之间的主要区别是通信通道。vhost-net内核驱动程序使用ioctl实现此通道，vhost-user库定义了通过unix套接字发送的消息结构。

可以将DPDK应用程序配置为提供unix套接字（服务器模式）并使QEMU连接到它（客户端模式），但是相反的情况也是可能的，这允许DPDK重新启动而无需重新启动VM。

在此套接字上，所有请求均由QEMU发起，其中一些请求需要响应，例如GET_FEATURES请求或任何设置了REPLY_ACK位的请求。

与使用vhost-net内核模块一样，vhost-user库允许主数据库通过执行以下重要操作来配置数据平面卸载：

1. 功能协商：virtio功能和特定vhost-user的功能都以类似的方式协商：首先，QEMU“获取”处理程序支持的功能的位掩码，然后“设置” 其与自身支持功能的子集。
2. 内存区域配置：QEMU设置内存映射区域，以便处理程序可以对它们进行mmap()。
3. Vring配置：QEMU设置=置要使用的虚拟队列数量及其在内存区域中的地址。请注意，vhost-user支持多队列，因此可以配置多个虚拟队列以提高性能。
4. 发送KICK和CALL文件描述符：通常，使用irqfd

和ioeventfd机制。

由于这种机制，DPDK应用程序现在可以通过与客户机共享内存区来处理数据包，并且可以直接与客户机之间发送和接收通知，而无需qemu干预。

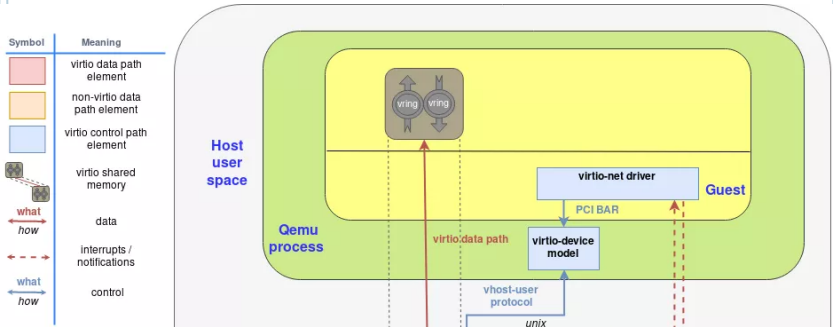
将一切结合在一起的最后一个元素是QEMU的virtio设备模型。它具有两个主要任务：

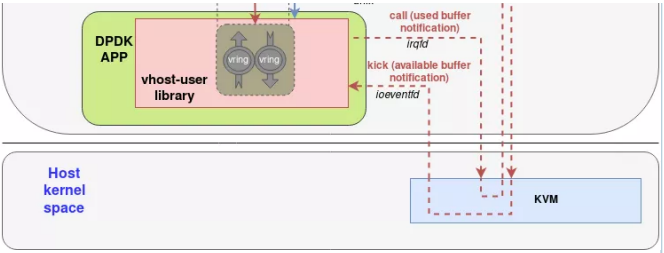
· 它模拟了一个virtio设备，该设备显示在客户机的特定PCI端口中，客户机可对该端口进行无缝探测和配置，此外，它还将ioeventfd映射到模拟设备的内存映射I/O空间，并将irqfd映射到它的全局系统中断（Global System Interrupt，GSI）。结果是，客户机没有意识到在没有QEMU干预的情况下，通知和中断都在vhost-user库之间来回转发。

· 它没有实现实际的virtio数据路径，而是在vhost-user协议中充当卸载数据处理的角色，将卸载该处理工作到DPDK流程中的vhost-user库中。

· 它处理来自控制虚拟队列的请求，在某些情况下，将其转换为虚拟主机用户请求并转发给vhost-user后端。

下图显示了作为DPDK-APP的一部分运行的虚拟主机用户库如何使用virtio-device-model和virtio-pci设备与qemu和客户机进行交互：





几个要点：

· virtio内存区域最初是由客户机分配的。

· 相应的virtio驱动程序通常通过virtio规范中定义的PCI BARs配置接口与virtio设备进行交互。

· virtio-device-model（位于QEMU内部）使用vhost-user协议配置vhost-user库，以及设置irqfd和ioeventfd文件描述符。

· 客户机分配的virtio内存区域由vhost用户库（即DPDK应用程序）映射（使用mmap 系统调用）。

· 结果是，DPDK应用程序可以直接在客户机内存中读取和写入数据包，并使用irqfd和ioeventfd机制直接对客户机发出通知。

（客户机中的用户空间网络）

我们已经介绍了DPDK的vhost-user实现如何使我们能够将数据路径处理从主机内核（vhost-net）卸载到专用的DPDK用户空间应用程序（例如Open vSwitch），从而显著提高网络的性能。现在，我们将看一看如何通过在客户机的用户空间中运行高性能网络应用程序（例如NFV服务）来取代virtio-net内核方法，以提高用户机网络性能。

为了能够直接在设备上运行用户空间网络应用程序，我们需要三个组件：

1. VFIO：VFIO是一个用户空间驱动程序开发框架，允许用户空间应用程序直接与设备进行交互（绕过内核）。
2. Virtio-pmd驱动程序：是基于Poll Mode Driver抽象规范构建的DPDK驱动程序，可实现virtio协议。
3. IOMMU驱动程序：IOMMU驱动程序用于管理虚拟IOMMU（I/O内存管理单元），该设备是对支持DMA的设备执行I/O地址映射的模拟设备。

让我们一一详细介绍这些组件。

◦ ◦（VFIO）◦ ◦

VFIO代表虚拟功能I/O。但是，vfio-pci内核驱动程序的维护者Alex Williamson建议将其称为“用于用户空间I/O的通用框架”，这种表述可能更准确。VFIO基本上是用于构建用户空间驱动程序的框架，功能有：

· 映射设备配置和I/O内存区域到用户内存

· 基于IOMMU组的DMA和中断重新映射和隔离。我们将在本文中更深入地介绍IOMMU的概念及其工作方式。现在，我们说它允许创建虚拟I / O存储空间，映射到物理内存（类似于普通MMU映射非IO虚拟内存的方式），因此，当设备要将DMA映射到虚拟I / O地址时，IOMMU将重新映射该地址并可能应用隔离和其他安全策略。

· 基于Eventfd和irqfd（还记得吗？）的信令机制可支持往返于用户空间应用程序的事件和中断。

引用内核文档：“如果要在VFIO之前编写驱动程序，则必须经历整个开发周期才能成为合适的内核驱动程序，必须以out-of-tree的方法维护，或者使用没有任何 IOMMU保护概念，中断支持功能有限的UIO框架，并且需要root特权才能访问诸如PCI配置空间之类的东西。”

VFIO公开了一个用户友好的API，用于创建字符设备（在/ dev / vfio /中），该设备支持用于描述设备的ioctl调用，设备描述符上的I / O区域及其读/写/mmap偏移以及描述和注册中断通知机制。

Virtio-pmd

DPDK提供了一个称为Poll Mode Driver（PMD）的驱动程序抽象规范，位于设备驱动程序和用户应用程序之间，为用户应用程序提供了很大的灵活性，同时保持了可扩展性，即为新设备实现驱动程序的能力。

下面列出了它的一些极其出色的功能：

· 一组API允许特定的驱动程序实现特定设备的接收和传输功能。

· 可以静态和动态配置每个端口和每个队列的硬件卸载

· 提供了用于统计信息的可扩展API，允许驱动程序定义自己的特定驱动程序的统计信息，并允许应用程序探查可用的统计信息并检索它们。

virtio轮询模式驱动（virtio-pmd）是使用PMD API的众多驱动程序之一，为使用DPDK编写的应用程序提供对virtio设备的快速无锁访问，从而提供了使用virtio的virtqueue进行数据包接收和传输的基本功能。。

除了任何PMD具有的所有功能之外，virtio-pmd驱动程序实现还支持：

· 接收时每个数据包具有灵活的可合并缓冲区，发送时每个数据包具有分散的缓冲区

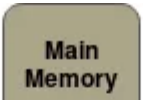
· 多播和混杂模式

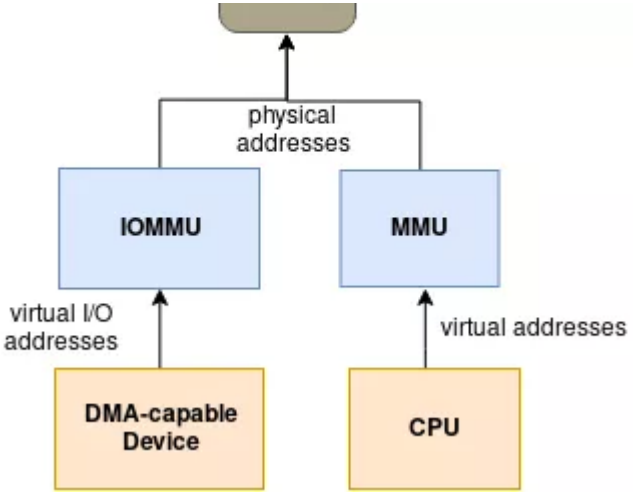
· MAC / VLAN过滤

结果是一个高性能的用户空间virtio驱动程序，该驱动程序允许任何DPDK应用程序无缝使用virtio的标准界面。

(IOMMU介绍)

IOMMU几乎等同于用于I / O空间的MMU（设备直接使用DMA访问内存）。它位于主内存和设备之间，为每个设备创建虚拟I / O空间并提供一种机制将虚拟I / O内存动态映射到物理内存。因此，当驱动程序配置设备的DMA（例如网卡）时，它会配置虚拟地址，并且当设备尝试访问它们时，它们将被 IOMMU重新映射。





它具有许多优点，例如：

· 可以分配较大的连续虚拟内存区域，而不必在物理内存中连续。

· 某些设备不支持足够长的地址来访问整个物理内存，IOMMU解决了该问题。

· 内存受到保护，不会受到恶意设备执行的DMA攻击，这些设备尝试访问未明确为其分配的内存。这些设备仅“看到”虚拟地址空间，并且运行中的操作系统专门控制IOMMU的映射。

· 在某些架构，支持中断重新映射，可以允许中断隔离和迁移。

像往常一样，一切都是有代价的，对于IOMMU来说，其不足是：

· 与页面转换服务有关的性能下降

· 添加的页面转换表会占用物理内存

最后，IOMMU提供了PCIe地址转换服务接口，设备可以使用该接口在内部设备表后备缓冲区（TLB）中查询和缓存地址转换。

vIOMMU——客户机的IOMMU

当然，如果存在物理IOMMU（例如Intel VT-d和AMD-VI），则qemu中将存在虚拟IOMMU。具体地说，QEMU的vIOMMU具有以下特征：

· 它将客户机I / O虚拟地址（IOVA）转换为物理地址（GPA），可以通过QEMU的内存管理系统将其转换为QEMU的主机虚拟地址（HVA）。

· 执行设备隔离。

· 实现I / O TLB API，以便可以从外部qemu中查询映射。

因此，为了使虚拟设备可以与虚拟IOMMU一起使用，我们必须：

1. 使用可用的API向vIOMMU中创建必要的IOVA映射。当前这些API是：

a. 用于内核驱动程序的内核DMA API

b. 用于用户空间驱动程序的VFIO

2. 使用虚拟I / O地址配置设备的DMA

(vIOMMU和DPDK集成)

虚拟IOMMU与任何用户空间网络应用程序之间的集成通常是通过VFIO驱动程序完成的。正如我们已经提到的那样，该驱动程序将执行设备隔离并将iova-gpa映射自动添加到IOMMU。

除了支持VFIO设置网络设备外，使用DPDK在IOMMU方面还有另一个非常重要的好处。由于DPDK使用的内存管理机制，该机制分配了一个静态内存池用于存储数据包缓冲区和虚拟队列，设备TLB同步消息的数量急剧下降，因此与之相关的性能损失也大大降低。大页面的使用进一步有助于优化TLB查找，因为更少的内存页面可以覆盖相同数量的内存。

vIOMMU和vhost-user集成

当在QEMU中被模拟的设备尝试DMA到客户机的virtio I/O空间时，它将使用vIOMMU TLB查找页面映射并执行安全的DMA访问。问题是如果实际的DMA被卸载到外部进程，例如使用vhost-user库的DPDK应用程序，会发生什么情况？

当vhost-user库尝试直接访问共享内存时，它必须将所有地址（I/O虚拟地址）转换为它自己的内存，这是通过向QEMU的vIOMMU请求设备TLB API的转换来实现的。Vhost-user库（以及vhost-kernel驱动程序）使用PCIe的地址转换服务标准消息集，以使用配置IOMMU支持时创建的辅助通信通道（另一个unix套接字）请求页面转换为QEMU。

总体而言，必须进行3种地址转换：

1. QEMU的vIOMMU将IOVA（I/O虚拟地址）转

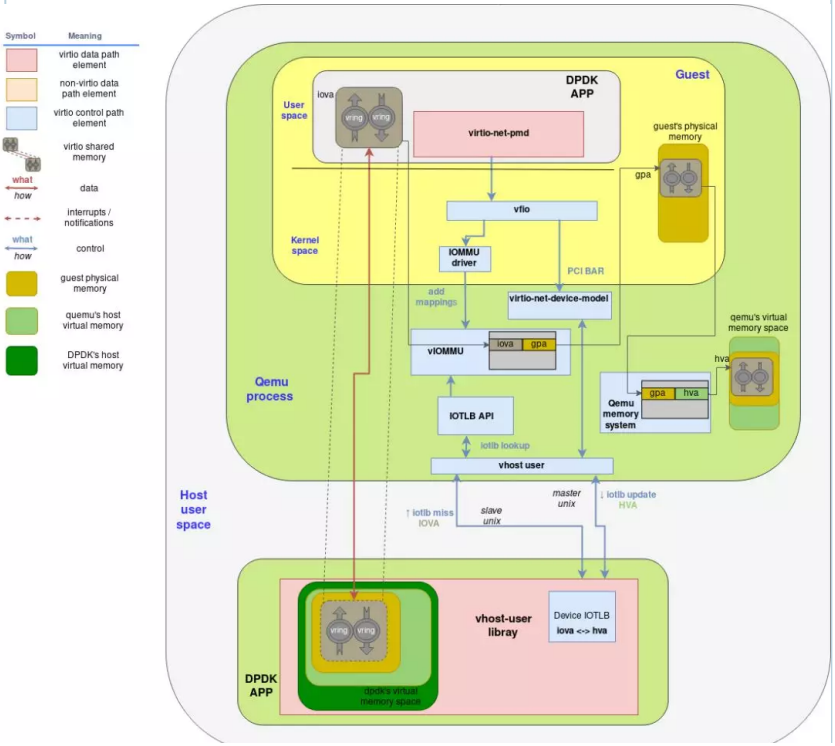
换为GPA（客户机物理地址）。

2. Qemu的内存管理将GPA（客户物理地址）转换为HVA（qemu进程地址空间内的主机虚拟地址）。

3. Vhost-user库将（QEMU的）HVA转换为自己的HVA。这通常很简单，只需在vhost-user库映射QEMU的内存时将QEMU的HVA添加到mmap返回的地址中即可。

显然，所有这些转换都可能对性能产生重要影响，尤其是在使用动态映射的情况下。但是，静态的大页分配（这正是DPDK所做的）可以最大程度地降低这种性能损失。

下图增强了以前的vhost-user架构，来包括IOMMU组件：



这幅复杂图示几个要点：

· 客户机的物理内存空间是客户机所感知的物理内存，但显然，它位于QEMU的进程（主机）虚拟地址内部。分配了虚拟队列内存区后，该内存最终会落在客户机的物理内存空间中的某个位置。

· 将I / O虚拟地址分配给包含虚拟队列的内存范围时，带有其关联的客户机物理地址（GPA）的条目将添加到vIOMMU的TLB表中。

· 另一方面，QEMU的内存管理系统知道客户机物理内存空间在其自己的内存空间中的位置，因此，它能够将客户机物理地址转换为主机（QEMU）的虚拟地址。

· 当vhost-user库尝试访问其没有转换的IOVA时，它将通过辅助unix套接字发送IOTLB未命中消息。

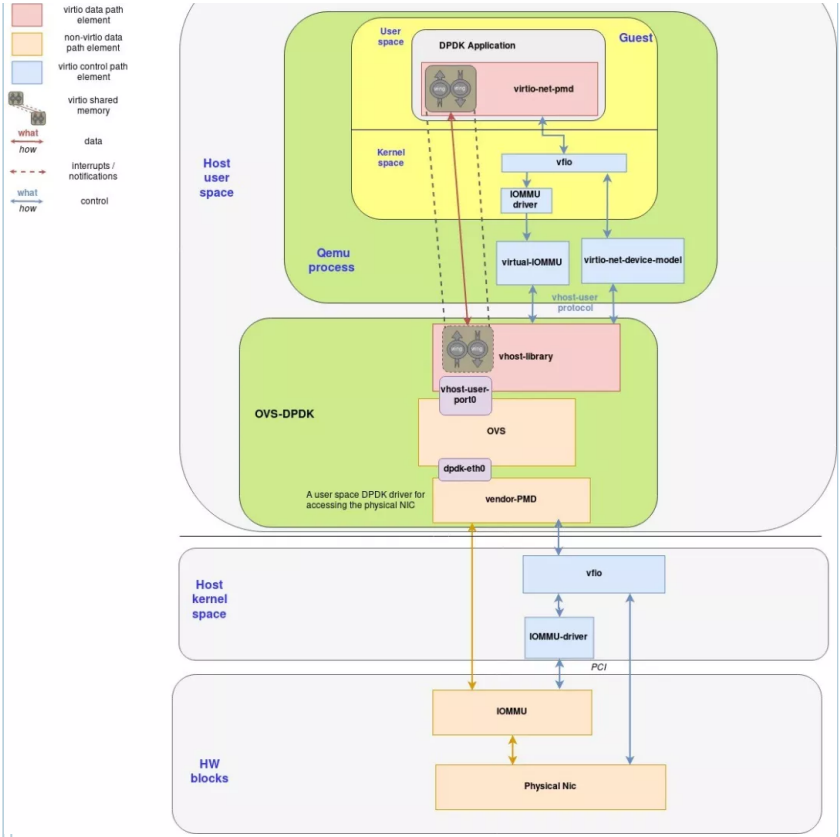
· IOTLB API接收请求并查找地址，首先将IOVA转换为GPA，最后将GPA转换为HVA，然后通过主机unix套接字将转换后的地址发送回vhost-user库。

· 最后，vhost-user库必须进行一次最终转换。由于已将qemu的内存映射到自己的内存，因此必须将qemu的HVA转换为自己的HVA并访问共享内存。

小结

在这篇文章中，我们介绍了许多组件，包括DPDK，virtio-pmd，VFIO，IOMMU等。

下图显示了这些组件块如何一起工作以实现vhost-user / virtio-pmd架构：

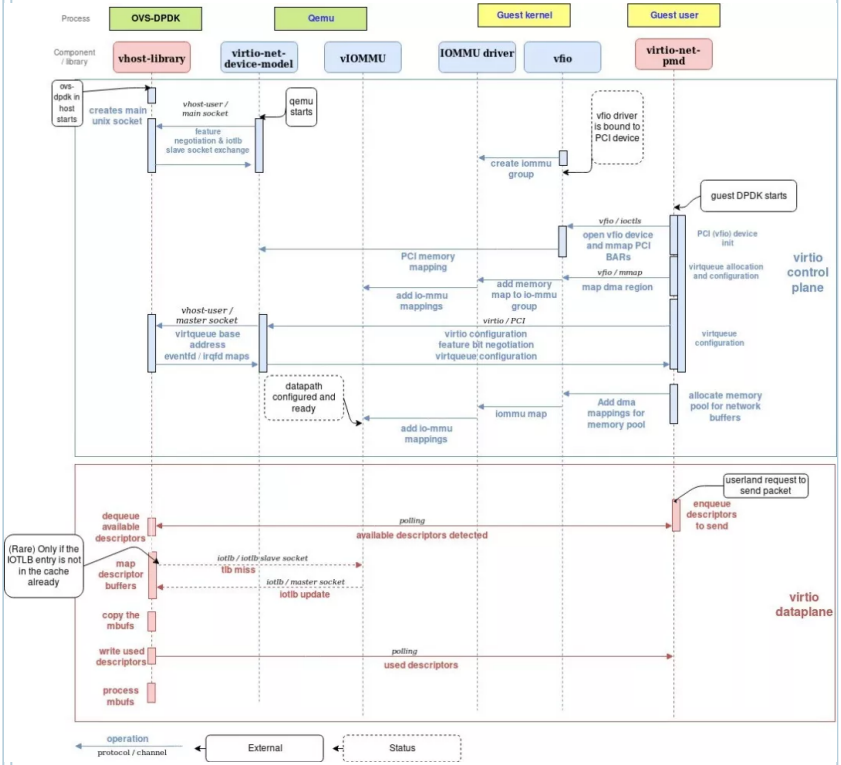


几个要点：

· 将该图与上一张图进行比较，我们添加了使用硬件IOMMU，VFIO和特定于供应商的PMD驱动程序，将主机的OVS-DPDK应用程序连接到物理网卡所需的组件。它等同于为客户机完成的操作，所以无需大惊小怪。

（流程示例）

以下流程图显示了设置virtio高性能数据平面所需的步骤：



(控制平面)

以下是设置控制平面所需的步骤：

1. 当主机(OvS)中的DPDK应用程序启动时，它将创建一个套接字(在服务器模式下)，用于与qemu进行与virtio相关的协商。

2. qemu启动时，它将连接到主套接字，如果vhost提供了 VHOST_USER_PROTOCOL_F_SLAVE_REQ 功能，则当主机（OvS）中的DPDK应用程序启动时，它

将创建一个套接字（以服务器模式使用），用于与qemu进行virtio相关的协商。

3. 当QEMU <-> vhost-library协商结束时，它们之间共享两个套接字。一个套接字用于virtio配置，另一个套接字用于iotlb消息交换。

4. 客户机启动，vfio驱动程序绑定到PCI设备，它将创建对iommu组的访问（取决于硬件拓扑）

5. 当dpdk应用程序在客户机中启动时，它执行以下初始化步骤：

a. 初始化PCI-vfio设备。vfio驱动程序还将PCI配置空间映射到用户内存。

b. 分配虚拟队列。

c. 使用vfio执行virtqueue内存空间DMA映射需求，通过IOMMU内核驱动程序将dma映射添加到vIOMMU设备。

d. 然后进行virtio功能协商。在这种情况下，用作virtqueue基址的地址是IOVA（在I / O虚拟地址空间中）。设置eventfd和irqfd的映射，以便可以在客户机和vhost-user库之间直接路由中断和通知，而无需QEMU的干预。

e. 最后，dpdk应用程序为网络缓冲区分配了大量的连续内存。该内存区域的映射也通过VFIO和IOMMU驱动程序添加到vIOMMU。

至此，配置完成，数据平面（虚拟队列和通知机制）可以使用了。

数据平面

为了传输数据包，需要执行以下步骤：

1. 客户机中的DPDK应用程序命令virtio-pmd发送数据包，它将缓冲区写入并将其对应的描述符添加到可用

的描述符环中。

2. 主机中的vhost-user PMD正在轮询虚拟队列，因此它立即检测到新的描述符可用并开始处理它们。

3. 对于每个描述符，vhost-user PMD映射其缓冲区（即：将其IOVA转换为HVA）。在极少数情况下，缓冲区内存是尚未在虚拟主机用户的IOTLB中映射的页面，将向QEMU发送请求。但是，客户机中的DPDK应用程序分配了静态大页面这一事实使对QEMU的IOTLB请求保持最小。

4. vhost-user PMD将缓冲区复制到mbufs（DPDK应用程序使用的消息缓冲区）中。

5. 描述符被添加到使用的描述符环中，客户机中的DPDK应用程序会立即检测到这些描述符，该客户机也正在轮询虚拟队列。

6. mbuf由主机中的DPDK应用程序处理。

（总结和结论）

DPDK是一项有前途的技术，因其为用户领域的高性能网络带来的好处而受到了广泛的关注，这项技术与Open vSwitch相结合，不仅可以提供现代虚拟环境所需的灵活性和性能，还将在NFV部署中发挥关键作用。为了充分利用此技术（作为数据中心交换数据路径和客户机中NFV应用程序的使能器），有必要在主机和客户机之间安全地创建有效的数据路径。这就是virtio-net技术发挥作用之处。

Vhost-user提供了一种可靠且安全的机制，可将网络处理任务卸载到基于DPDK的应用程序中。它和vIOMMU集成，提供隔离和内存保护，同时将QEMU从处理所有数据包的繁重工作中解放出来。

在客户机中，通过在DPDK（virtio-pmd）中实施virtio规范，可以在客户机系统中创建快速数据路径，从而利用DPDK的轮询模式驱动程序提供的高效内存管理和高速运行。

本文翻译自Red Hat blog，原文链接

https://www.redhat.com/en/blog/journey-vhost-users-realm

原文作者 Eugenio Perez Martin, Adrian Moreno

扫码关注我们

dpdkchina

进群交流更多技术知识

转载须知

DPDK与SPDK开源社区公众号文章转载声明

推荐阅读

Virtio网络的演化之路

Virtio-PMD的路径选择与用法

DPDK源代码网页浏览现在更方便了！

阅读原文