第3章

服务器虚拟化与网络技术

Half the work that is done in the world is to make things appear what they are not.

世界上一半的工作都是要让事物看起来不是它们原来的样子。

——E. R. Beadle

本章目标

学习完本章之后，应当能够：

(1) 理解并给出服务器虚拟化的基本概念、技术分类及优势。

(2) 了解硬件辅助虚拟化的基本概念。

(3) 列举服务器虚拟化中虚拟网络接入的常见技术。

(4) 理解目前广泛使用的容器网络技术。

在当前的云数据中心网络中，特别是多租户的数据中心网络里，在同一台物理服务器上同时运行多台虚拟机(Virtual Machine，VM)是十分常见的。这依赖于服务器的虚拟化及相关的网络技术来进行实现。服务器虚拟化技术对扩展数据中心可承载的实际用户数和提高硬件资源的利用率等方面，起着至关重要的作用。对于云计算服务提供商(如微软、亚马逊、谷歌和阿里云等公司)，通过虚拟化技术来降低实际使用的物理服务器数量，并提高这些物理服务器上的利用率，将能够有效地降低硬件及管理成本的投入。本章主要介绍服务器的虚拟化技术，以及与其相关的网络技术。

3.1 虚拟化简述

相对于普通 PC，云服务提供商拥有的物理服务器的硬件配置(包括 CPU、内存、网络带宽等)显然十分强大。然而，云服务的用户却并不需要每时每刻都让所租用的服务器处于满负荷运行的状态。因此，云服务提供商就可以通过虚拟化技术，将单一的物理服务器虚拟为更多的虚拟机，并把这些虚拟机租给不同的用户。这种做法既可以提高服务器资源的利用率，使得更多的用户享受到云计算带来的便利性，同时虚拟化的资源也非常方便进行管理和调度。

典型的例子：阿里云等公司为了应对“黑色星期五”或“双 11”等特殊时间

点的流量高峰，基本的办法就是通过准备足够多的服务器，来保证数据中心受到巨大流量冲击时仍能向用户提供服务。然而，在流量高峰恢复正常后，就不需要如此多的服务器了。因此，云服务提供商可以把这些暂时不需要的服务器经过虚拟化后租给其他用户来获取收入。这也是云计算服务能够兴起的重要商业动机之一。

3.1.1　虚拟化概念

虚拟化并不是一个新的概念，而是一种发展了几十年的技术。最早在 20 世纪 60 年代，IBM 公司就通过开发虚拟机监视器（Virtual Machine Monitor，VMM），采用时间共享的方式，将当时十分昂贵的大型机从逻辑上虚拟为多个小型机，使得多个用户能够共享大型机的计算资源，访问原先只能通过独占大型机才能获得的服务。这是一种典型的服务器虚拟化的思想。

当前对虚拟化较为全面的定义：

> **虚拟化**是一种将物理资源（包括计算、存储和网络等）进行抽象、转换和隔离，并最终向用户呈现出一个可动态配置的虚拟运行环境，使得用户在使用这些资源时就可以不受资源的物理配置与地理位置的限制。

云计算环境中的虚拟化包括服务器虚拟化、网络虚拟化、存储虚拟化，以及应用虚拟化等多种不同的虚拟化技术。本章主要关注服务器虚拟化技术。

在服务器虚拟化中，物理服务器通常称为宿主机（Host Machine），在其上运行的操作系统称为宿主操作系统（Host OS）。Hypervisor（即 VMM）是用来创建与运行虚拟机的软件、固件或硬件[①]。在 Hypervisor 上运行的虚拟机也称客户机（Guest Machine），在虚拟机中运行的操作系统被称为客户操作系统（Guest OS）。

如今云计算中的虚拟化技术，与 60 年前刚诞生的虚拟化技术并没有本质上的区别，都是一种资源管理技术，都是向用户提供在 Hypervisor 的统一管理下的资源。这里的用户，既可以是虚拟机（包括客户机操作系统和运行于其上的程序），也可以是单独的程序，如 Docker 运行环境中独立的（Self-contained）应用程序。因此，Hypervisor 与用户的关系类似于操作系统与应用程序的关系：Hypervisor 为用户统一管理他们所需要的各类资源。

虚拟化技术发展历史

从 20 世纪 60 年代开始，美国的计算机学术界就开始了虚拟技术的萌芽，1959 年 6 月，在国际信息处理大会上，克里斯托弗·斯特雷奇（Christopher Strachey）在纽约的信息技术国际会议上发表了题为《大型高速计算机分时技术》的论文，被认为是虚拟化技术最早的论述。

---

① Hypervisor 包括两类：Type 1 和 Type 2。其中，Type 1 类型的 Hypervisor 也称 Native 或 Bare-Metal Hypervisor；Type 2 类型的 Hypervisor 也称 Hosted Hypervisor。Type 2 类型的 Hypervisor 需要运行在传统操作系统之上。云计算环境下主要为 Type 1 类型的 Hypervisor。因此，如无特别说明，本书主要关注 Type 1 类型的 Hypervisor。

1964年,IBM公司的M44/44X项目实现了在同一台主机上模拟出多个7044系统,首次使用Virtual Machine和Virtual Machine Monitor等名词,被认为是世界上第一个支持虚拟化的系统。1972年,IBM剑桥实验室(CSC)发布了CP40,引入革命性的虚拟机、虚拟内存分时操作等概念,成为后来的S360主机的基础。

1972年,IBM公司发布了VM/370,这是一个用在S370、S390、zSeries上的虚拟机操作系统。

1998年,VMware公司成立,通过运行在Windows NT上的VMware启动Windows 95操作系统。

2001年,VMware公司推出第一个基于x86服务器的虚拟化产品。

2006年,Intel和AMD公司陆续宣布从处理器层面支持虚拟化,市场上出现更多的虚拟化解决方案。虚拟化逐渐成为IT行业的主流技术。

3.1.2 服务器虚拟化技术及分类

Hypervisor是一个完备的操作系统,它除具备传统操作系统的基本功能外,还具备虚拟化的功能,包括对物理资源的抽象、转换和隔离操作,而这些操作一般被认为是对物理资源的虚拟化操作。

现代计算机体系结构一般通过划分多个特权级来分隔系统软件和应用软件。按照运行级别的不同,CPU的指令可以简单分为两类。

(1) 特权指令(Privileged Instructions)。会修改操作系统本身的指令,或者使用外部资源操作(如磁盘和网络等)的指令等。这些指令一般需要通过调用操作系统提供的接口,进入系统模式后才能被实际执行。特权指令只能在最高级别上运行,在低级别状态下执行会产生Trap。

(2) 非特权指令(Non-Privileged Instruction)。不会影响操作系统对硬件的支配权的指令或不涉及I/O的指令。此类指令无须操作系统介入即可执行,可以在各个级别的状态下执行。

现代操作系统的工作模式通常包含用户态(User Mode)和内核态(Kernel Mode)。处于内核态的操作系统能够完全支配底层硬件,可以执行包括特权指令在内的任何指令。相对地,除了操作系统以外的其他程序(也称应用程序)则运行在用户态之上,仅能运行非特权指令。如果应用程序在用户态执行特权指令则会陷入内核态,并在内核态完成这些指令。

根据虚拟化机制的不同,可以将虚拟化技术分为4类。

1. 全虚拟化

在全虚拟化(Full Virtualization)模式下(见图3-1),Guest OS不修改任何代码就可以直接运行在Hypervisor上。这种模式下,来自Guest OS的非特权指令会直接被Hypervisor传递(Bypass)给物理服务器CPU执行;而特权指令(例如,修改虚拟机的运行模式或下面物理服务器的状态、读写时钟或者中断寄存器等)会触发异常,被

Hypervisor 捕获并做适当处理后再被物理服务器 CPU 运行(即 Trap-and-Emulate)。Hypervisor 一般是对每条指令进行解释和执行,通过二进制代码翻译(Binary Translation)等方式,模拟出该指令执行的效果。

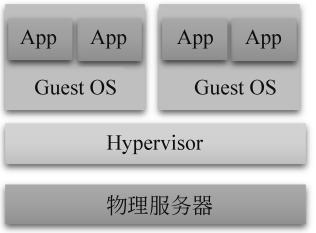

图 3-1　全虚拟化

全虚拟化方式的优点是无须修改 Guest OS 就可以运行虚拟机,即 Guest OS 根本感知不到自己运行在一个虚拟化环境中;缺点是性能开销较大。当前使用较多的全虚拟化软件,如 VMware Workstation、VirtualBox 和 Microsoft Hyper-V 等。

2. 半虚拟化

由于全虚拟化在频繁调用 Hypervisor 以处理特权指令的过程中,会带来较大的性能开销,因此,如何降低该部分开销成为提升 Guest OS 的性能亟须解决的问题。

图 3-2　半虚拟化

一种解决方法是采用半虚拟化(Para Virtualization)技术,如图 3-2 所示,也称操作系统辅助虚拟化(OS Assisted Virtualization),通过修改 Guest OS 内核,替换掉不能虚拟化的指令,通过超级调用(Hypercall)直接和 Hypervisor 进行通信。Hypervisor 提供了超级调用接口来满足关键内核操作,如内存管理、中断和时间保持等。这样就省去了全虚拟化中的异常捕获和模拟等操作,提高了虚拟化的效率,使得 Guest OS 的运行性能可以接近在物理服务器上的性能。

典型的半虚拟化技术包括 Xen 等。然而,由于半虚拟化要求对 Guest OS 进行一定的修改,在不开放源代码的操作系统(如 Windows 系统等)上较难使用。

3. 硬件辅助虚拟化

随着虚拟化技术的发展,硬件厂商开始研发新功能以简化虚拟化技术,包括 Intel VT(Intel Virtualization Technology)和 AMD-V(AMD Virtualization)等技术。该类 CPU 针对特权指令设计新的操作模式,包括 VMX root 模式和 VMX non-root 模式。Hypervisor 可以运行在 VMX root 模式下,而 Guest OS 运行在 VMX non-root 模式下。两种操作模式可以互相转换。通过这种在硬件上的区分,避免了在全虚拟化方式下,对特权指令的异常捕获、模拟等操作,提升了客户机性能。

随着 CPU 厂商支持虚拟化的力度越来越大,硬件辅助的全虚拟化技术的性能逐渐接近半虚拟化,特别是考虑到全虚拟化无须修改 Guest OS,因此硬件辅助虚拟化(Hardware Assisted Virtualization)成为一个重要的发展趋势。支持硬件辅助虚拟化的软件包括 VMware ESXi、Microsoft Hyper-V 和 Xen 3.0 等。3.2 节中将会对硬件辅助虚拟化相关技术进行详细介绍。

4. 操作系统层面的虚拟化

在云计算平台中，会经常遇到一些需要大规模部署统一软件的情况。以部署 Web 服务器集群来实现大流量情况下的负载均衡为例，把每个 Web 服务器实例都安装到一个单独的虚拟机实例中是完全可行的。然而，每个 Web 服务器实例都仅仅用到了 Guest OS 所提供的一小部分功能，使得这种方式会浪费大量的资源。

为了解决这个问题，把这些 Web 服务器实例部署在同一个操作系统中，使得它们可以共享 Host OS 提供的服务。然而，这些部署在同一操作系统中的 Web 服务器之间很可能会在资源使用上产生冲突，而要在这种环境下实现不同实例之间的资源隔离，就需要用到操作系统层面的虚拟化(OS Level Virtualization)。

这种虚拟化方式利用操作系统提供的资源管理技术，如 Linux 系统下的 namespace、cgroups 和 chroot 等，把应用程序所用到的资源进行抽象、转换和隔离，使这些程序在部署同一操作系统中也不会互相冲突。这种方式的优势在于占用服务器空间少，通常几秒内即可引导，同时可以弹性地增加或释放资源。此类虚拟化技术也称容器技术，其中的代表性产品是 Docker。

3.1.3 虚拟化技术优势和典型应用场景

目前，虚拟化技术已经在市场上得到了广泛的应用。特别是，虚拟化技术促进了云计算概念的产生和发展，已经成为云计算的主要支撑技术之一。两者相辅相成、互相促进。究其原因，是因为虚拟化技术为云计算的发展带来了以下两点优势。

(1) 提高资源利用率和资源隔离。通过使用虚拟化技术，云服务提供商能够把昂贵的物理服务器虚拟为数量众多的逻辑服务器。这样，物理服务器就可以让多用户同时共享使用，并且这些用户不会在资源调用上产生冲突，最终使物理服务器上的资源始终保持较高的利用率。

(2) 便于部署与迁移。Hypervisor 会对宿主机的物理资源进行抽象，并向 Guest OS 提供统一的运行环境，使后者并不会感知到自己运行在虚拟环境中。这样就可以使用户的程序能够被部署在任意物理服务器上，极大地方便了用户程序的部署与迁移。

由于虚拟化环境提供了比非虚拟化环境更高的资源利用率和便利性，使虚拟化技术的应用场景得到了较大的增加。以下用 3 个例子来阐述虚拟化技术的典型应用场景。

(1) 在数据中心环境中批量部署相同的程序运行环境。在没有虚拟化技术的环境中，要保证不同的运行环境不会在资源使用上产生冲突，管理员需要为每套运行环境准备一台物理服务器以及与之相应的操作系统。而在虚拟化场景中，管理员只需要准备足够的物理服务器集群，然后基于物理服务器集群批量建立虚拟机就可以完成任务。

(2) 在云计算平台中向多租户提供虚拟机以提高物理服务器的硬件资源利用率。在此场景中，租户通常不会让所租用的虚拟主机每时每刻都处于满负荷运行的状态，所以云服务提供商可以通过虚拟化技术让物理服务器承载更多的虚拟主机，以实现提高资源利用效率的目的。而对性能要求极高的用户，则可以向云服务提供商租用非虚拟化的主机，并独占该主机的所有资源。

(3) 在桌面环境通过虚拟化技术提供灵活的应用开发平台。在此场景中,用户通常需要使用虚拟化技术为运行于异构(Heterogeneous)操作系统下的程序提供运行环境,如在运行 Windows 系统的主机中运行 Linux 程序。用户就可以使用虚拟化软件(如 VMware 和 VirtualBox 等),在 Windows 系统中建立基于 Linux 系统的运行环境,并进行相关程序的开发与测试工作。

对于虚拟机技术,在云计算环境下,并不是所有的云数据中心都会采用虚拟机技术。在 IaaS 中,云服务商允许用户选择如何使用所租用的硬件资源,包括虚拟机的部署等;在 PaaS 中,云服务商一般会采用虚拟机技术来优化服务器资源;在 SaaS 中,云服务商除了可以采用虚拟机技术外,也有可能采用其他私有的方案来提供服务。

3.1.4　容器技术

在虚拟化技术中,另一个与虚拟机密切相关的概念是容器(Container)。容器指基于操作系统的资源隔离技术,为应用程序构建出一个轻量级、标准化的,并与其他应用程序互相隔离的运行环境。容器中包含应用程序本身以及必需的运行环境(一般统称镜像),使得该容器能够在任何具有容器引擎(如 Docker Engine)的环境中运行。图 3-3 为容器技术与虚拟机技术架构的对比。

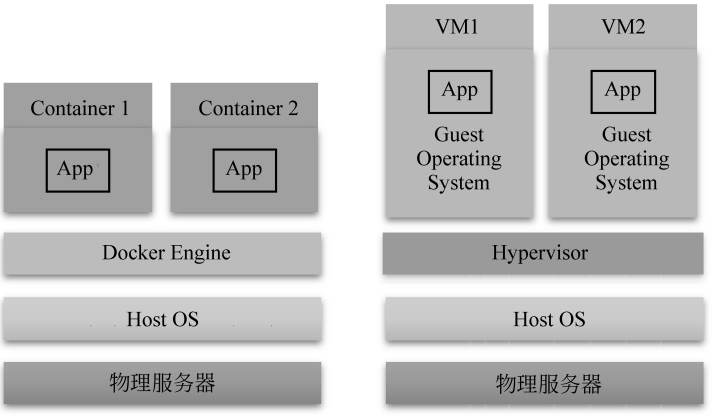

图 3-3　容器技术与虚拟机技术架构的对比

图 3-3(a)所示的容器技术封装了应用正常运行所需要的运行环境,而图 3-3(b)所示的虚拟机技术,可以通过独立的 Guest OS 和 Hypervisor 来提供完全独立于宿主机的运行环境。在虚拟机上进行的任何操作都不会改变宿主机本身。因此,虚拟机技术能够提供相比于容器技术更好的资源隔离性。然而,由于 Hypervisor 在对硬件资源进行虚拟化的过程中不可避免地会带来性能损失,而容器可以直接通过 Host OS 来使用物理服务器的硬件资源,使容器技术在硬件资源的利用率上明显优于虚拟机技术。

实际上,对于一个网络服务,如一个 Web 服务器实例,并不需要用到操作系统的所有功能。在大规模部署类似的服务时,为每个 Web 服务器实例都提供一个完整的操作系统将浪费许多资源。同时,使用虚拟机技术需要启动操作系统,而使用容器技术可以大大节

省开发、测试和部署的时间，做到“一次构建，到处运行”。另外，由于容器中通常会封装有正常运行服务所需要的依赖项，在新节点上部署服务时会较少遇到运行环境问题。由此可见，使用虚拟机技术的核心诉求是提高硬件资源的利用率；使用容器技术的核心诉求则是加速应用的开发、测试和部署流程。与重量级的虚拟机技术相比，容器技术只是一种轻量级的资源隔离技术。两者的简要对比见表 3-1。

表 3-1　容器技术与虚拟机技术的简要对比

项　　目	容　　器	虚　拟　机
启动时间	几百毫秒到几秒	几十秒到几分钟
占用存储空间	MB 级	GB 级
总体性能	接近裸机	较明显损失
单主机可部署数量	最多支持上千个容器	最多支持几十台虚拟机

现有研究指出，在云环境中，一台物理服务器甚至能容纳高达上千个容器。可以使用 Docker 官网提供的 RIO Calculator 来计算部署相同服务时分别使用 Docker 和使用虚拟机的成本差距。

容器技术的具体实现方式五花八门，目前最主流的实现是 Docker。Docker 架构如图 3-4 所示。Docker 的实现是基于 Linux 系统的资源隔离技术（namespace、cgroups 和 chroot 等），并在此基础上实现了良好的封装，使得用户不再需要考虑容器所需资源的实际管理操作。

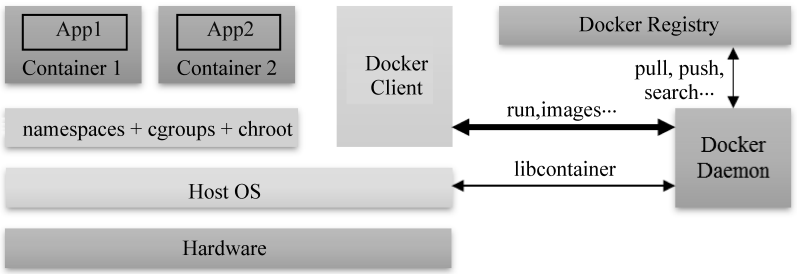

图 3-4　Docker 架构

虚拟机开启了云计算时代，而容器作为下一代虚拟化技术，正在逐渐改变业界开发、测试和部署应用的方式。

3.1.5　常见 Hypervisor

要动态管理云计算平台中的各项虚拟化资源，不可避免地要至少使用以下 3 种常用 Hypervisor 中的一种：VMware vSphere、Microsoft Hyper-V 和 OpenStack。其中，前两者是 VMware 和 Microsoft 公司推出的商用虚拟化平台，并提供了一些基本的管理功能；OpenStack 则是一种开源云操作系统，包含了一系列与虚拟化资源相关的管理功能。

从市场地位方面，VMware 公司在虚拟化领域中的地位相当于桌面操作系统领域中

的 Microsoft 公司、数据库领域中的 Oracle 公司或小型机领域中的 IBM 公司，长期以来占据了虚拟化领域的垄断地位。这不仅是因为 VMware 公司进入虚拟化的时间早、体量大，而是由于 VMware 公司的虚拟化产品具有高性能、高稳定性、高容错性和高可扩展性等一系列优秀特性。这些特性使 VMware vSphere 在私有云领域占比超过 50%[①]。

虽然用户通过付费就可以获取到 VMware vSphere 或 Microsoft Hyper-V 稳定高效的虚拟化服务。但是在庞大的数据中心环境下，完全依赖商用虚拟化技术需要付出昂贵的授权费用。因此，技术实力充足的用户（如阿里巴巴和谷歌等企业），已经开始趋向于使用开源工具（如 OpenStack 等）来实现云数据中心所需要的各项虚拟化功能，并结合通用的 x86 服务器和由 Linux 系统内核支持的 KVM 虚拟化等技术来降低成本。

图 3-5 为 OpenStack 架构。OpenStack 架构中主要包含以下 6 个核心项目：实现计算资源虚拟化的 Nova 项目、实现网络资源虚拟化（管理）的 Neutron 项目、为虚拟机提供 OS 镜像服务的 Glance 项目、提供分布式块存储服务的 Cinder 项目、提供身份认证服务的 Keystone 项目，以及提供对象存储服务的 Swift 项目。除此之外，还有其他的可选服务，如提供环境监控服务的 Ceilometer 项目和提供控制面板的 Horizon 项目等。受限于 OpenStack 的开源要求，其通常使用开源的 KVM 和 libvirt 以实现对资源的虚拟化。

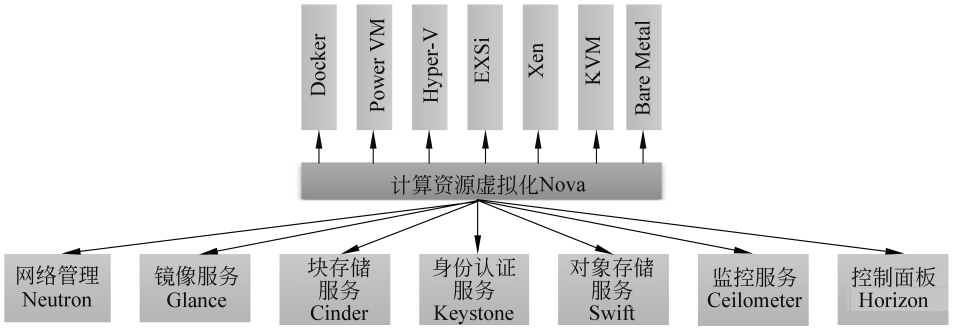

图 3-5　OpenStack 架构

3.1.6　主机网络技术

当前，数据中心环境下的主机网络技术（Host Networking）是一个十分热门的新兴研究领域，各界对于主机网络技术的领域范围尚未有明确的定义。然而，基于现有的一些研究成果，主机网络技术要解决的主要问题是数据包在数据中心的虚拟化网络中的“最后一千米”问题，即如何在主机范围内解决好数据包的调度问题，以达到尽可能地提高网络吞吐率和降低延迟的目的。因此，本节把主机网络技术的研究范围确定为数据中心服务器上的网络 I/O（即图 3-6 中的 Host Networking 部分）。

如图 3-6 所示，主机网络技术主要包括以下两方面的内容。

（1）服务器上虚拟机或容器之间的通信。本部分的研究重点在于如何实现同一物理

① 来源：RightScale 2018 State of the Cloud Report，2018。

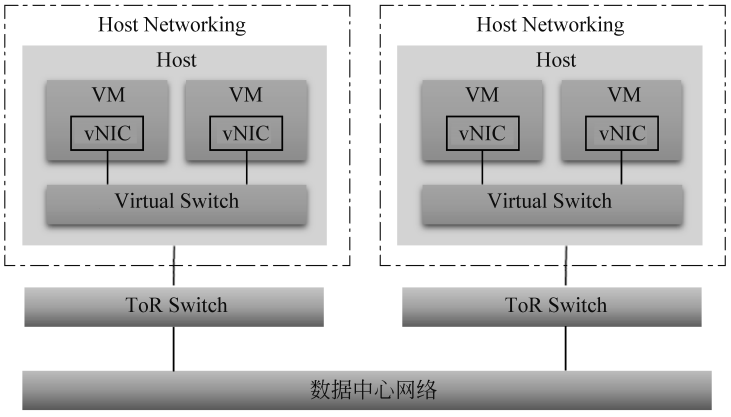

图 3-6　主机网络技术示意图

服务器上的不同虚拟机或容器之间的高效通信。此方向目前发展较为完善，主要包括基于虚拟交换机和传统的 TCP/IP 通信等。

(2) 虚拟机或容器的网卡与服务器外网之间的通信。本部分是主机网络技术的研究热点，其主要研究如何提高虚拟机或容器的网卡与服务器外网的通信效率，包括网卡上的数据包队列管理、网络管理策略的实现、高速 RPC 和 RDMA(Remote Direct Memory Access)等方式进行优化。

主机网络技术与服务器技术(特别是网络 I/O 技术)存在着密切的联系。例如，在论文 *Eiffel: Efficient and Flexible Software Packet Scheduling*(NSDI 2019)中，作者针对主机 NIC 上的数据包调度顺序问题，使用了基于 Find First Set 实现的软件排队器来代替昂贵的硬件排队器，更加方便管理员实现(Enforce)网络策略并取得更高的网络 I/O 吞吐率。在论文 *Loom: Flexible and Efficient NIC Packet Scheduling*(NSDI 2019)中，作者通过多队列实现了多租户环境下的策略实现和有序调度。

3.2　硬件辅助虚拟化

3.2.1　硬件辅助虚拟化概述

在服务器虚拟化中，由于客户机上的 Guest OS 运行在用户态之上，以及全虚拟化模式没有对 Guest OS 进行任何改动，如果需要让 Guest OS 认为自己正在运行于内核态，就需要让 Hypervisor 捕获 Guest OS 所执行的特权指令，并且通过软件的方式模拟这些指令。然而，完全通过软件模拟的方式来执行这些特权指令会给 CPU 带来较大的性能开销。因此，现代 CPU 一般都实现了特殊的技术来辅助优化 Hypervisor 处理特权指令，因此能有效降低 Trap-and-Emulate 过程给 CPU 带来的性能开销。例如，在常用的 x86 CPU 上，硬件辅助虚拟化技术主要有 Intel 公司的 VT-x(2005 年加入)技术以及 AMD 的 AMD-V 技术(2006 年加入)。

硬件辅助虚拟化技术并非十全十美。要启用硬件辅助虚拟化技术，除了要 CPU 在

硬件层面进行支持外，也需要操作系统和上层应用的支持。同时，启用硬件辅助虚拟化技术也会带来一些安全风险。

3.2.2　硬件辅助的网卡虚拟化技术

由于让 Hypervisor 拦截虚拟机的网络 I/O 操作并通过模拟网卡操作的方式来完成相关指令，会极大地消耗宿主机的 CPU 资源，使能被应用程序使用的 CPU 资源降低。同时，通过模拟的方式也无法充分利用网卡硬件资源；通过使用由网卡支持的硬件辅助虚拟化技术，可以有效降低进行大流量网络 I/O 操作时给宿主机 CPU 带来的负载。

本节主要介绍服务器虚拟化技术中的 3 种典型的硬件辅助网卡虚拟化技术：直接分配(Direct Assignment)、虚拟机设备队列(Virtual Machine Device Queue，VMDq)和单根 I/O 虚拟化(Single-Root Input/Output Virtualization，SR-IOV)。

1. 直接分配

直接分配技术指的是把一张物理网卡独占性地分配给一台虚拟机。典型的例子为在 Intel 平台上使用直接分配技术为虚拟机加速网络 I/O 处理，如图 3-7 所示。该技术要求宿主机支持并开启 VT-d(Virtualization Technology for Direct)I/O 功能，使宿主机中的每台虚拟机都能够独占性地获得宿主机中的一块网卡。虚拟机的网络 I/O 数据流在 VT-d 技术的支持下直接被送达独占使用的网卡。这样，虚拟机的网络 I/O 数据流不再需要被 Hypervisor 中的虚拟交换机处理，能极大地减轻 CPU 处理网络 I/O 的负担，并能够向虚拟机提供媲美使用物理网卡时的网络 I/O 性能。然而，这种方式要求为每台虚拟机都提供一块供其独占使用的物理网卡，当宿主机上需要部署几十甚至上百台虚拟机时，显然不可能提供如此多的物理网卡。

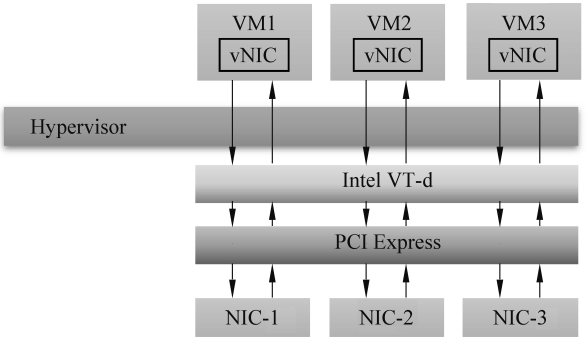

图 3-7　Intel 平台上使用直接分配技术

2. 虚拟机设备队列 VMDq

随着服务器中虚拟机数量的增加，对所有虚拟机通信流量的管理会消耗 CPU 资源，进而降低虚拟化的性能。例如，在网卡无虚拟化支持的情况下，网卡在收到数据包后，需要完成以下操作才能最终把数据包交付给指定的虚拟机。

(1) 网卡收到数据包,向 CPU(如 CPU0)发送中断。

(2) CPU0 收到中断信号并检查包头,确定这个数据包应当转发给哪台虚拟机。

(3) 中断目标虚拟机的 CPU,并由此 CPU 负责把数据包复制到虚拟机的内存中。

上述过程中会多次产生 CPU 中断,影响服务器和虚拟机上其他程序的正常运行。因此,Intel 公司提出了虚拟机设备队列技术来提高此类场景中对网络 I/O 数据流的处理速度,如图 3-8 所示。通过支持 VMDq 技术的网卡代替 CPU 处理这些数据流,能有效降低网络 I/O 给 CPU 带来的性能开销并提高网络吞吐率。

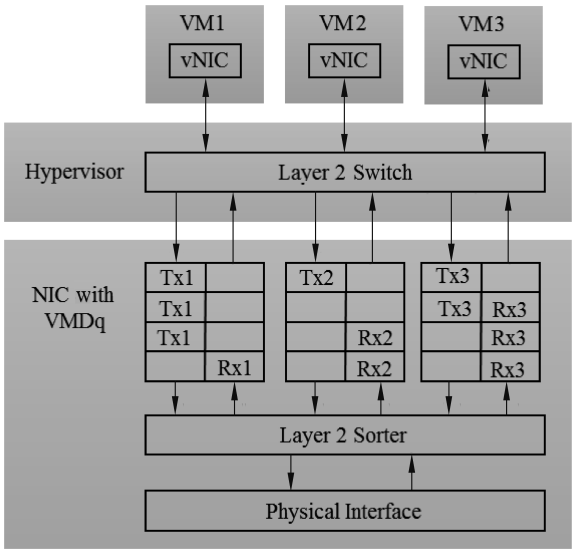

图 3-8 VMDq 模型

在具体实现上,VMDq 在服务器的物理网卡中为每台虚拟机创建了一个独立的队列。当网卡收到数据包时,网卡会先检测该数据包的帧头信息(如 MAC 地址、VLAN Tag 等),再决定应该将数据包转发至哪台虚拟机所对应的队列中,并最终由 Hypervisor 复制到目标虚拟机的内存空间中。发送数据时,虚拟机直接将数据包转发到相应队列中,网卡再根据各种调度机制(如轮询等)发送出去。这些过程都不会产生 CPU 中断,因此 VMDq 能够极大地降低 CPU 在处理网络 I/O 时的性能开销。然而,由于 Hypervisor 和虚拟交换机仍然需要将网络流量在 VMDq 和虚拟机之间进行复制,此类操作还是会带来一定的性能开销。

3. 单根 I/O 虚拟化

PCIe(Peripheral Component Interconnect Express)是一种通用串行总线,能够提供远高于 PCI 总线的带宽。在数据中心常用的 x86 系统结构中,CPU 一般使用 PCIe 总线接入高速设备(如显卡和高性能网卡),而接入的高速设备则形成以 CPU 为根的逻辑设备树。然而,即使采用了 PCIe 总线,CPU 在虚拟化环境下处理网卡产生的高速中断时,也需要消耗大量计算资源。

为了进一步降低 CPU 的负担，Intel 公司提出了 SR-IOV，如图 3-9 所示，允许不同的虚拟机共享当前物理服务器上的硬件资源（Root 指的是 CPU）。与 VMDq 不同，SR-IOV 使虚拟机能够以虚拟独占的方式直接连接到 PCIe 插槽上的高性能网卡，使虚拟机能够获得媲美独占物理网卡的 I/O 性能。

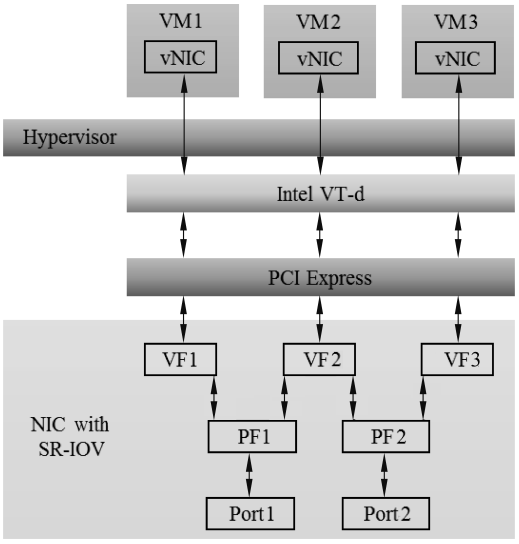

图 3-9　SR-IOV 实例

SR-IOV 包含两个重要概念：物理功能（Physical Function，PF）和虚拟功能（Virtual Function，VF）。PF 指的是物理网卡所提供的一项物理功能；VF 指的是由支持 SR-IOV 的物理网卡虚拟出来，并向虚拟机提供的一个能够实现虚拟独占的功能。VF 可以与其他 VF 共享同一个 PF 的物理资源，如缓存和端口等。

在网卡启用 SR-IOV 功能后，每个 PF 和 VF 都会被分配一个 PCIe Requester ID（RID），使 CPU 中的输入输出内存管理单元（I/O Memory Management Unit，IOMMU）能够区分不同 VF 的数据流。这样，网络数据流就能从 PCIe 网卡（实际上为 VF）直接被导向虚拟机，免去了 Hypervisor 中的虚拟交换机的参与，完全消除了这一部分操作所带来的性能开销，极大地提高了虚拟机的网络吞吐率。

PCIe 总线中除了 SR-IOV 技术外，还有 MR-IOV 技术。不同的是，MR-IOV 技术可以使多个服务器共享不同的 I/O 设备，如网卡、主机总线适配器（Host Bus Adapter，HBA）和主机通道适配器（Host Channel Adapter，HCA）等。由于 MR-IOV 技术实际应用较少，在此不再赘述。

3.3　虚拟网络接入技术

在虚拟化的数据中心网络环境中，如何合理高效地把虚拟机接入数据中心网络中，是规划、构建和管理虚拟化数据中心网络的关键问题。本节主要关注如何在数据中心网络的边缘部分，将虚拟机接入网络中。针对数据中心整体网络的虚拟化技术相关的内容，将

在第 4 章中详细阐述。

3.3.1 虚拟交换机

在虚拟化的数据中心环境中，每台物理服务器上所承载的虚拟机和容器的数量十分巨大(例如，根据工作负载和服务器性能，虚拟机的数量可以从几十个到上百个不等)。为每台虚拟机实例或者容器实例按 1∶1 的方式配备网卡和交换机等物理资源显然不可取。为了解决物理资源不能满足虚拟化场景需求的问题，业界很自然地提出了虚拟交换机技术。

虚拟交换机技术，指的是通过软件的形式来虚拟一台物理交换机，并实现与物理交换机同样的存储-转发功能。该技术一般让虚拟机实例或者容器实例首先连接到物理服务器上的虚拟交换机，并由后者统一连接到物理网卡，进而接入数据中心网络。当数据流的目的地为本物理服务器上的其他虚拟机实例或者容器实例时，虚拟交换机可以直接把数据流导向目的地而无须物理网卡参与。当数据流目的地不在本物理服务器上时，数据流会被虚拟交换机导向物理网卡。此种情况也可以使用 3.2.2 节中所述的硬件辅助网卡虚拟化技术加速处理过程。图 3-10 中阐述了虚拟交换机的一般工作方式，即在 Hypervisor 中集成一个二层虚拟交换机(vSwitch)，从虚拟机的虚拟网卡(vNIC)中发出的流量需要先经过 vSwitch 后才能被转发至下一跳。

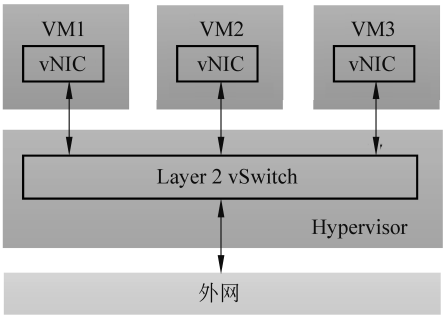

图 3-10 虚拟交换机

与传统物理交换机相比，虚拟交换机具有以下优势：可以很方便地进行按需部署；虚拟交换机上的网络策略能够更加方便地随着虚拟机的迁移而转移；可以通过在 Hypervisor 内进行部署的形式提高性能；能够很方便地通过软件升级的形式给虚拟交换机扩充新的功能等。

本节介绍 4 种广泛使用的虚拟交换机：VMware vSwitch、Microsoft Hyper-V Virtual Switch、Cisco Nexus 1000V 和 Open vSwitch。表 3-2 中对比了这 4 种虚拟交换机的特性。

表 3-2 虚拟交换机的特性对比

特　　性	VMware vSwitch	Microsoft Hyper-V Virtual Switch	Cisco Nexus 1000V	Open vSwitch
维护者	VMware	Microsoft	Cisco	VMware 和 Nicira

续表

特　　性	VMware vSwitch	Microsoft Hyper-V Virtual Switch	Cisco Nexus 1000V	Open vSwitch
目标平台	VMware ESXi	Microsoft Hyper-V	VMware vSphere	KVM、Xen 和 OpenStack
协议支持	VLANs、IPv6、Session Synchronization、Path Monitoring、VXLAN、GENEVE 和 NSX Gateway 等	PVLANS、Virtual Port ACL、Trunk Mode to Virtual Machines 和 WMI 等	VLAN、SPAN、QoS、Cisco vPath、Cisco VN-Link 等	NetFlow、sFlow、Port Mirroring、VLAN、LACP 等

1. VMware vSwitch

VMware vSwitch 是 VMware 公司用于在 vSphere 中连接 ESXi 环境虚拟机的虚拟交换机，包含集中式（如 vSwitch）和分布式（如 dvSwitch）两种。

在单台 ESXi 实例上部署虚拟机时，可以使用集中式的 vSwitch，可以支持 4096 台虚拟机接入网络。同时，vSwitch 也支持 Link Discovery（通过自动收集拓扑信息来方便排除网络故障）、NIC Teaming（多块物理网卡在进行链路聚合后再连接到 vSwitch，以提供高带宽和容错性）和 Traffic Shaping（可以限制流经 vSwitch 的数据流的带宽）等高级特性。

在数据中心等分布式环境中组件虚拟化网络时，可以使用 dvSwitch，以便把一个统一的虚拟交换机部署在多台 ESXi 实例上，并可最多支持 60 000 台虚拟机接入网络，以及在 vCenter 服务器上的统一配置接口，如图 3-11 所示。除了 vSwitch 所提供的特性外，

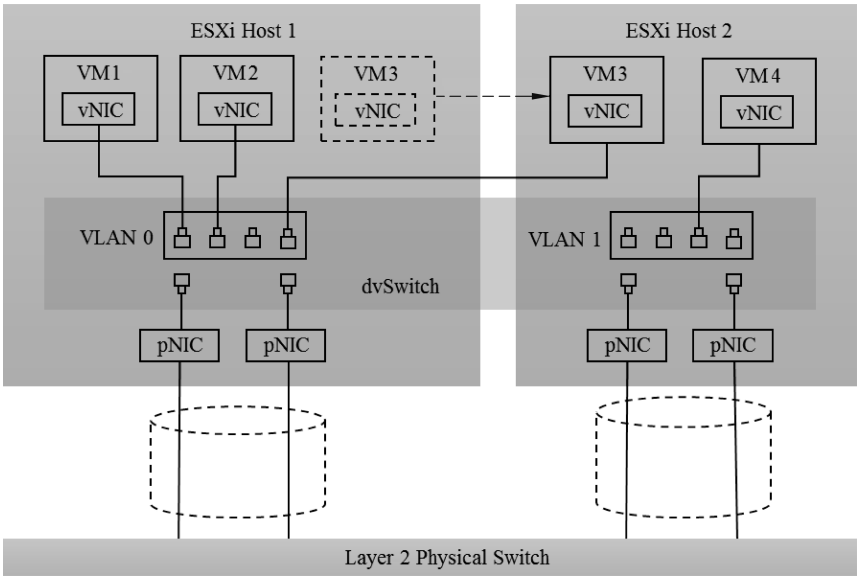

图 3-11　dvSwitch 模型

dvSwitch 还提供了许多新的高级特性，例如，Port Group Blocking（以组的形式来屏蔽经过指定端口组的数据流）、Per-port Policy（可以针对每个端口设置不同的网络策略）以及对 LLDP 的支持等。

2. Microsoft Hyper-V Virtual Switch

与 VMware 的虚拟交换机相似，Hyper-V 环境下也需要使用虚拟交换机，来实现让 Hyper-V 虚拟化环境中的虚拟机之间的互联互通。Hyper-V Virtual Switch 是一个二层虚拟交换机，提供了可编程性、动态策略和资源隔离等高级特性。然而，Microsoft Hyper-V Virtual Switch 只支持连接到以太网，而不支持连接到其他类型的网络（如光纤网络）。这使它的应用场景受到极大限制。此外，由于 Microsoft Hyper-V Virtual Switch 提供了对 NDIS（Network Device Interface Specification）过滤器和 WFP（Windows Filtering Platform）的支持，使它非常适合 Windows 虚拟化环境中虚拟交换机。

3. Cisco Nexus 1000V

作为传统的 ICT 设备厂商，Cisco 公司也与 VMware 公司合作，为 VMware ESXi 环境开发了软件交换机：Cisco Nexus 1000V。Cisco Nexus 1000V 能够较好地支持 VN-Tag 和VN-Link 等 Cisco 公司自研的接入层虚拟化技术。与 VMware vSwitch 和 Microsoft Hyper-V Virtual Switch 一样，Cisco Nexus 1000V 也是一款支持 VLAN、QoS 和 NetFlow 等云数据中心常用技术的分布式虚拟交换机。

Cisco Nexus 1000V 虚拟交换机包含两个主要部件：在 Hypervisor 内部作为虚拟交换机运行的虚拟以太网模块（Virtual Ethernet Module，VEM）和管理 VEM 的外部虚拟控制引擎模块（Virtual Supervisor Module，VSM），如图 3-12 所示。其中，VEM 会作为 VMware ESXi 环境中 Hypervisor 内部的虚拟交换机，处理虚拟机的网络数据流。由于 VEM 内置于 Hypervisor 中，故 VEM 能够很好地支持 VMware vMotion 和 Distributed Resource Scheduler（DRS）等虚拟机管理技术。VSM 作为管理组件，向管理员提供了数据方面的高可靠性、高可扩展性以及易于管理等高级特性。

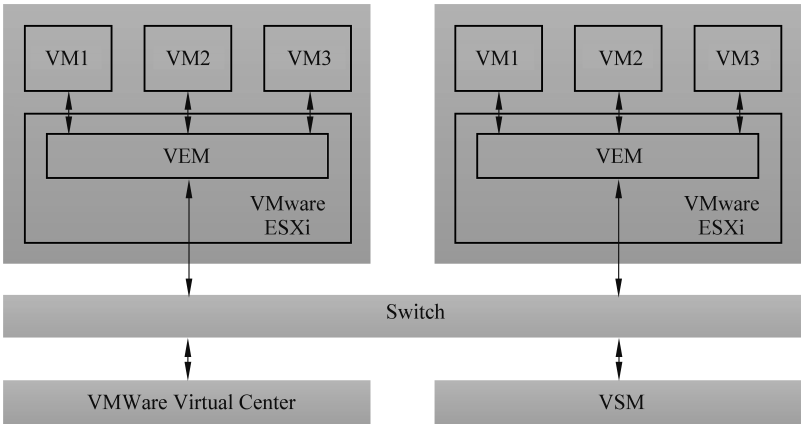

图 3-12　Cisco Nexus 1000V 虚拟交换机架构图

4. Open vSwitch

除了 VMware vSwitch 和 Microsoft Hyper-V Virtual Switch 这两种得到广泛使用的商用虚拟交换机外，开源社区也在 Nicira 公司的主导下基于 Apache 2.0 协议提出了开源的分布式多层虚拟交换机 Open vSwitch(OVS)，如图 3-13 所示。OVS 的目的是让大规模网络实现管理自动化，并且可以通过编程扩展(在 SDN 环境中使用)。为了维持对现有网络的兼容，OVS 也支持标准的管理接口和协议。由于 OVS 是一个开源项目，开发人员可以根据需要自行添加需要的功能而不必受限于商业软件的许可条款。而且，OVS 支持多种基于 Linux 系统的虚拟化技术(如 Xen 和 KVM)，使得 OVS 在 OpenStack、openQRM 和 OpenNebula 等开源分布式云计算平台中得到了广泛应用。OVS 在 Linux Kernel v3.3 以后已被加入了 Linux 系统内核中。

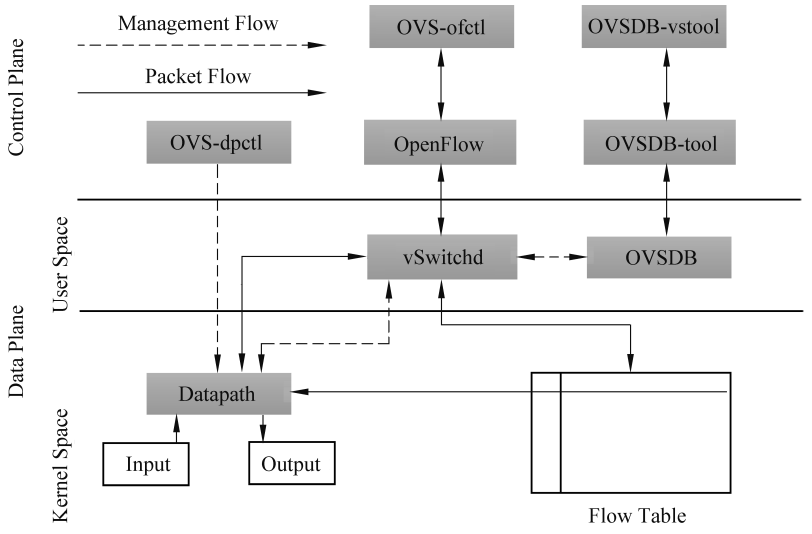

图 3-13　OVS 主要组件

在功能上，OVS 支持 VLAN、LACP(IEEE 802.1AX—2008)和 sFlow 等功能。而在控制和管理方面，OVS 也支持 OpenFlow 协议并内置了 Open vSwitch DataBase (OVSDB)，使 OVS 既能够被当作虚拟交换机使用，也可以当作物理交换机的控制面使用。同时，在 SDN 环境下，OVS 可以被任何支持 OpenFlow 协议的控制器控制，使得 OVS 在 SDN 相关的开发工作中具有举足轻重的地位。除 SDN 外，OVS 也可以单纯地作为一个独立的二层交换机使用，让其通过自动学习数据流的 MAC 地址而建立转发表。

3.3.2　边际虚拟网桥

对于普遍部署了虚拟机的云数据中心，把物理服务器上的数十台虚拟机通过适当的方式有效接入网络是一个核心要求。本节介绍一些数据中心网络中实现虚拟机接入网络的基本技术。这些技术同属于边际虚拟网桥(Edge Virtual Bridging，EVB)技术，其目标

就是让虚拟机以适当的方式共享宿主机上的物理网卡，并尽量达到与虚拟机独占物理网卡相近的性能。目前，实现 EVB 的技术主要有 IEEE 802.11Qbh（VN-Tag）和 IEEE 802.11Qbg（VEPA）等。

1. 边际虚拟网桥技术概述

边际虚拟网桥技术与 3.3.1 节中所介绍的虚拟交换机技术具有较大的联系。从网络的角度，虚拟机处于当前物理服务器上的所有虚拟机所组成的一个本地网络中。要把这些虚拟机接入外网，一个直观的想法就是在 Hypervisor 中内置一个虚拟交换机，通过桥接（Bridge）的方式把虚拟机和外网连接起来。这样，虚拟机的网络流量首先会经过这个虚拟交换机处理，然后才能进入外网（见图 3-10）。

通过在 Hypervisor 中集成的虚拟交换机处理本地虚拟网络的流量，需要占用宿主机的 CPU 资源。由于使用 CPU 进行处理操作（拆包、检查包头、封包和转发等）的效率远低于使用 ASIC 的硬件交换机。因此，在网络流量较大时，使用 CPU 进行网络处理将带来极大的性能开销。

另一种解决方案是只在虚拟交换机中保留最基本的储存-转发功能，其他的高级功能和复杂的策略（如过滤、安全、监控等）则通过把数据包交由外部硬件交换机处理的方式实现。这种解决方案除了部分地解决了虚拟交换机性能不足的问题外，也把与网络相关的管理操作从服务器上转移到了网络中的硬件交换机中，有利于明确网络与服务器的边界，简化了运维操作。

2. VN-Tag

为了实现对包括虚拟机通信在内的所有网络通信流量进行一致处理，并优化 VEB 的性能，Cisco 和 VMware 等厂商提出了私有的虚拟网络标签（VN-Tag）协议来实现虚拟机接入外网的功能。VN-Tag 技术主要分为两部分：VN-Link 技术和 FEX（Fabric Extender）技术。前者是位于服务器的交换组件（如支持 VN-Tag 的物理网卡），负责接入服务器上的各台虚拟机；后者主要部署在外网的物理交换机上，负责数据中心内虚拟机之间的互联互通。

图 3-14 中展示了标准以太网帧、使用了 VLAN 的以太网帧和使用了 VN-Tag 的以太网帧的格式。与 VLAN 相比，VN-Tag 保留了 VLAN Tag，并在 VLAN Tag 前添加了 VN-Tag。同时，把 VLAN 帧中的 EtherType 字段前移到 VN-Tag 的首部。与标准以太网帧相比，相当于在 EtherType/Len 字段后面添加了 VN-Tag。VN-Tag 中各字段的含义如表 3-3 所示。

VN-Tag 的具体通信过程如下：服务器上的交换组件收到来自源虚拟机的数据帧后，向数据帧中添加 VN-Tag，并设置标签中的相关字段，然后转发给外部的物理交换机。外部的物理交换机接收到此帧后，根据路由算法决定应该将此帧从哪个端口转发给目的服务器（根据 Destination VIF 进行转发）。目的服务器收到此帧后，先去除帧中的 VN-Tag，然后再把原始数据帧转发给目的虚拟机。

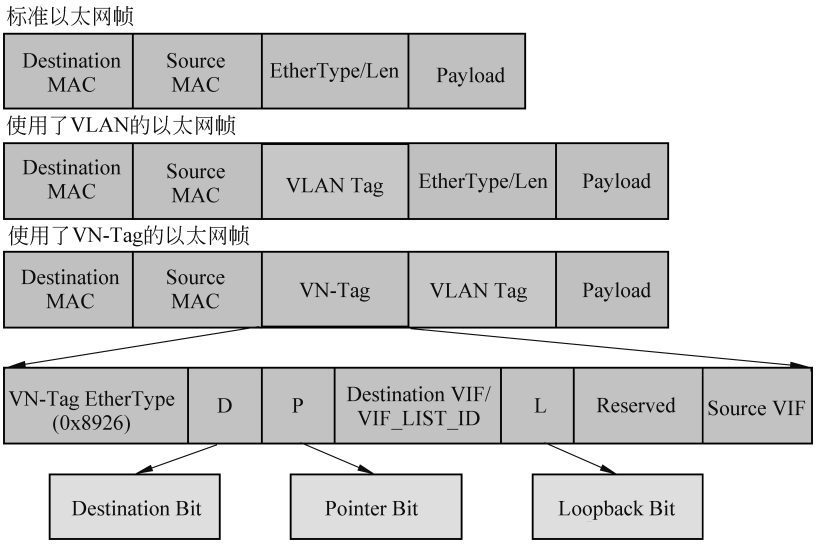

图 3-14　标准以太网帧、使用了 VLAN 的以太网帧和使用了 VN-Tag 的以太网帧的格式

表 3-3　VN-Tag 中各字段的含义

字 段 名 称	含　　义
Destination Bit	方向标志，1 标志此帧是从网桥被发送至接口虚拟器
Pointer Bit	指针标志，1 标志是否有 VIF_LIST_ID 包含在此帧内
Destination VIF	目标端口的 VIF_ID
VIF_LIST_ID	需要转发此帧的下行端口列表，用于广播或者多播操作
Reserved	保留字段
Loopback Bit	环回标志，1 标志此帧是一个从接收到此帧的端口返回的多播帧
Source VIF	源端口的 VIF_ID

在具体实现上，服务器上的交换组件（支持 VN-Tag 的物理网卡）只负责 VN-Tag 的添加与删除，使得虚拟机能够接入支持 VN-Link 的网络中，不负责任何与寻址或者策略相关的工作。FEX 技术使用 Cisco Nexus 2000 Series 交换机（N2K）作为 Cisco Nexus 5000 Series 交换机（N5K）在 ToR 上的分布式接入。由于 N2K 与 N5K 之间的连接具有高带宽、低延迟和低阻塞等优秀特性，保证了分布式接入的性能。VN-Tag 技术能够让数据中心的接入网络虚拟为一个“大二层网络”，并且网络策略能够随虚拟机的迁移而迁移。

3. VEPA

由于 VN-Tag 需要部署支持相关功能的网卡与交换机等设备，限制了实际网络中 VN-Tag 的部署。因此，研究人员也提出了另一种优化 EVB 性能的方式：虚拟以太网端

口聚合 VEPA(Virtual Ethernet Port Aggregator)通信模型,如图 3-15 所示。

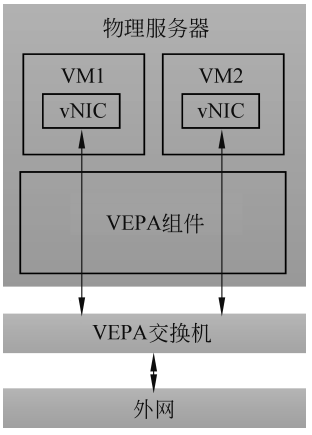

图 3-15 VEPA 通信模型

与 VN-Tag 相比,VEPA 没有修改数据帧,而是通过对转发规则的修改实现了把负载从宿主机 CPU 卸载到网卡上的功能。具体来说,就是 VEPA 要求在宿主机上的 VEPA 组件和虚拟机之间运行 VDP(Virtual Station Interface Discovery Protocol)。VDP 负责识别虚拟机的接入点,并提供对虚拟机迁移的支持。VEPA 协议要求虚拟机的网络出站和入站的网络流量全部需要经过 VEPA 组件,并且 VEPA 组件上的流量方向只能是流入或流出物理服务器,而不能在虚拟机之间流动。这样,VEPA 就能够通过把流量转发给外网的物理交换机,来实现高性能的数据包交换。

3.4 容器网络技术

由于容器技术能够以十分轻量级的方式来部署应用,使得以 Docker 为主的容器技术已经在数据中心等企业环境中得到广泛应用。不同于虚拟机网络,容器网络在设计和实现时面临更多的挑战。例如,容器采用 Network Namespace 提供网络在内核中的隔离,网络设计需更为慎重;容器在数量、动态性和分布上也都不同于虚拟机,面临着更多新的挑战。本节以 Docker 和 kubernetes 为例,介绍容器间的网络通信方式。

3.4.1 接入方式

以 Docker 为例,目前容器主要支持以下 6 种网络接入方式。

1. Bridge

桥接(Bridge)网络指的是利用 Linux Bridge 等虚拟网桥来连接到物理网卡,使容器拥有独立的 Network Namespace,如图 3-16 所示。默认情况下 Docker 会使用此网络接入方式。当 Docker 进程启动时,会在主机上创建一个名为 docker0 的虚拟网桥,主机上的其他容器会连接到该虚拟网桥上。在单一(Standalone)Docker 宿主机上部署 Docker 时,通常也使用这种方式来让 Docker 容器连接到网络。

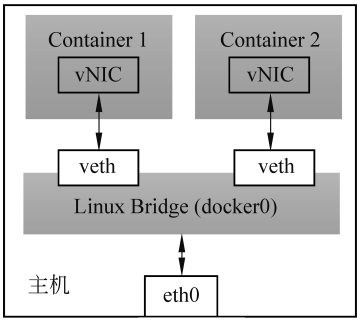

图 3-16 Bridge 接入方式

除了 Linux Bridge 外,Docker 还支持使用 OVS 等虚拟交换机进行桥接。使用 OVS 能够支持相比 Linux Bridge 更为强大的转发和管理功能,如 VLAN、Tunnel 和 Traffic Shaping 等高级网络功能。因此,生产环境中也多用 OVS 配合适当的 SDN 控制器实现容器的网络桥接需求。

2. Host

在 Host 方式下，容器直接使用主机的网络协议栈，并共享主机的 IP 地址，如图 3-17 所示。但是由于这种方式不会给容器分配独立的 Network Namespace，而是处于宿主机的网络环境中，共用宿主机的 L2～L4 网络资源。因此，该方式具有较低的额外开销，但是网络隔离性差，如 Docker 使用的端口容易与主机上其他程序的端口冲突（容器仍具有独立的进程空间和文件系统）。

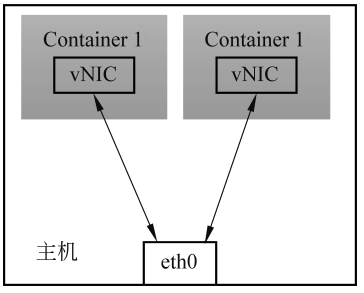

图 3-17　Host 接入方式

3. Container

Container 方式可以与其他容器共享网络协议栈，如图 3-18 所示。除了 Network Namespace 外，不同的容器仍然拥有独立的进程空间和文件系统。从网络角度，两个容器作为一个整体对外提供服务。

4. macvlan

macvlan 方式把 Docker 宿主机的物理网卡在逻辑上虚拟出多个虚拟网卡，并且给每个子接口分配一个虚拟的 MAC 地址，如图 3-19 所示。使用该方式的容器在发送数据时，如果目的 IP 地址位于本机，则直接转发到相应的容器中；否则交给物理网卡处理，以转发到处于外网上的目标主机。在接收数据时，采用类似的处理。

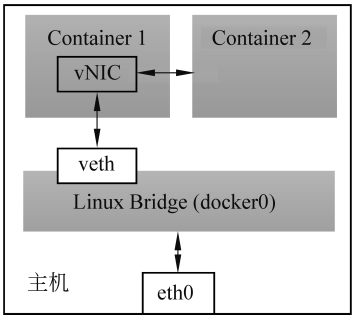

图 3-18　Container 接入方式

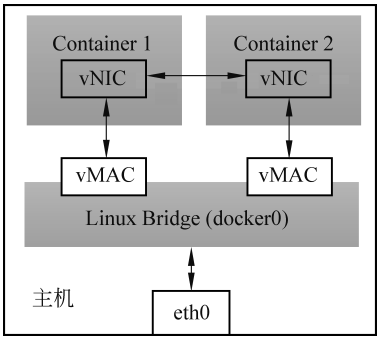

图 3-19　macvlan 接入方式

5. User-defined

该方式支持 Docker 1.9 及以上版本，由用户自行确定实际使用的网络类型，可以使用第三方插件，也可以使用一个新的 Bridge 网络，甚至可以使用由 OVS 等虚拟交换机软件来创建网络。