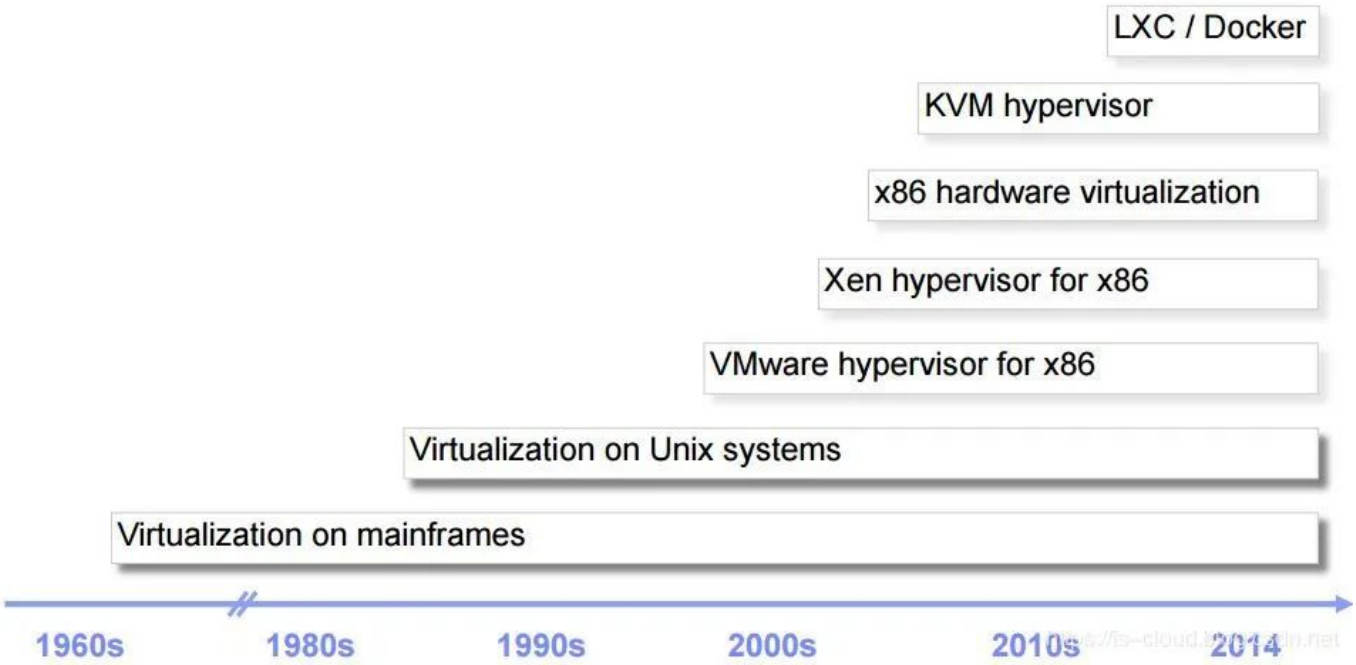
深入理解虚拟化

原创 极客重生 极客重生 2021-11-22 08:05

收录于话题
#深入理解Linux系统 29 #深入理解云计算 8

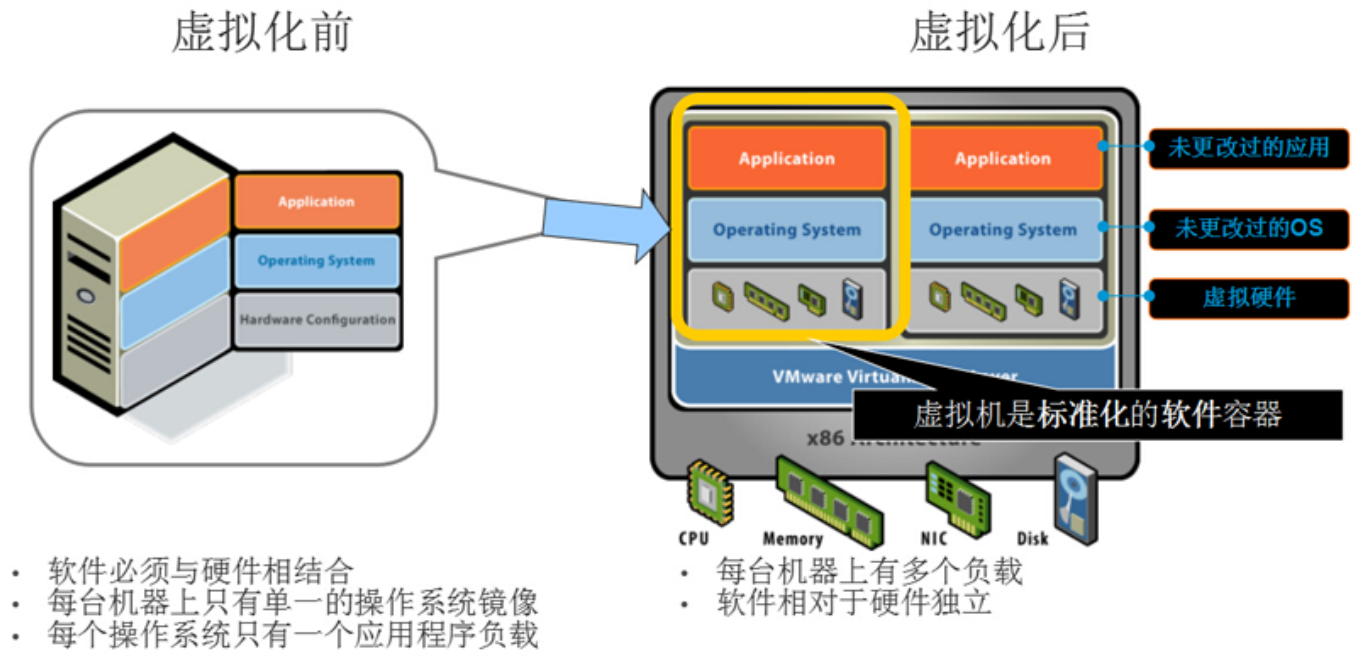


什么是虚拟化

虚拟化技术是云计算的根基，在计算机技术中，**虚拟化（技术）**或**虚拟技术**（英语：Virtualization）是一种资源管理技术，是将计算机的各种实体资源（**CPU、内存、磁盘空间、网络适配器**等），予以抽象、转换后呈现出来并可供分割、组合为一个或多个电脑配置环境。由此，打破实体结构间的不可切割的障碍，使用户可以比原本的配置更好的方式来应用这些电脑硬件资源。这些资源的新虚拟部分是不受现有资源的架设方式，地域或物理配置所限制。一般所指的虚拟化资源包括**计算(CPU+内存），网络，存储**。

虚拟化的本质

虚拟化本质是指资源的抽象化，要想资源充分利用，必须把资源最小单位化(池化)，这样上层才能按需使用资源，虚拟化不但解放了操作系统，也解放了物理硬件，大大提高了**资源的利用率**。



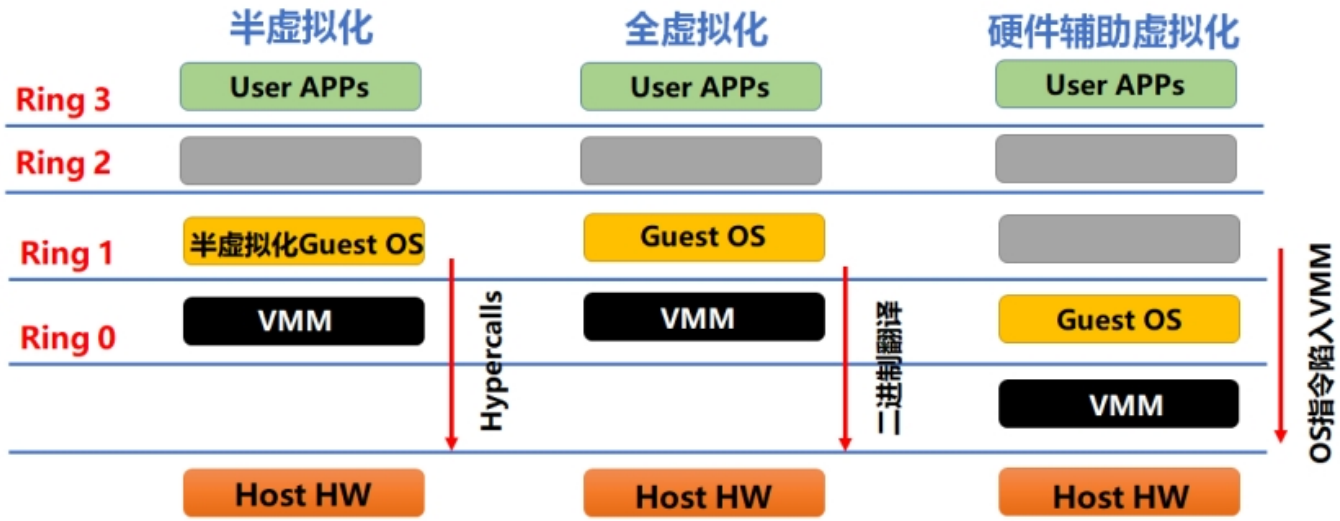
- 虚拟化管理程序**Hypervisor**（VMM），位于虚拟机与底层硬件设备之间的虚拟层，直接运行于硬件设备之上，负责对硬件资源进行抽象，为上层虚拟机提供运行环境所需资源，并使每个虚拟机都能够互不干扰、相互独立地运行于同一个系统中。

虚拟化主要分为几大类：

- **计算虚拟化**，针对CPU和内存资源虚拟化技术。
- **网络虚拟化**，针对网络链路资源虚拟化技术。
- **IO虚拟化**，针对IO资源虚拟化技术。
- **存储虚拟化**，针对磁盘存储资源虚拟化技术。

计算虚拟化

计算虚拟化通过虚拟化管理程序（Hypervisor或VMM）将物理服务器的硬件资源与上层应用进行解耦，形成统一的计算资源池，然后可弹性分配给逻辑上隔离的虚拟机共享使用。如图2、基于VMM所在位置与虚拟化范围可以分三种类型。



	全虚拟化	硬件辅助虚拟化	操作系统协助/半虚拟化
实现技术	二进制翻译,BT 和直接执行	遇到特权指令转到root模式执行	Hyper call
客户操作系统修改/兼容性	无需修改客户操作系统，最佳兼容性	无需修改客户操作系统，最佳兼容性	客户操作系统需要修改来支持hypercall，因此它不能运行在物理硬件本身或其他的hypervisor上，兼容性差，不支持Windows
性能	差	全虚拟化下，CPU需要在两种模式之间切换，带来性能开销；但是，其性能在逐渐逼近半虚拟化。	好。半虚拟化下CPU性能开销几乎为0，虚机的性能接近于物理机。
应用厂商	VMware Workstation/QEMU/Virtual PC	VMware ESXi/Microsoft Hyper-V/Xen 3.0/KVM	Xen

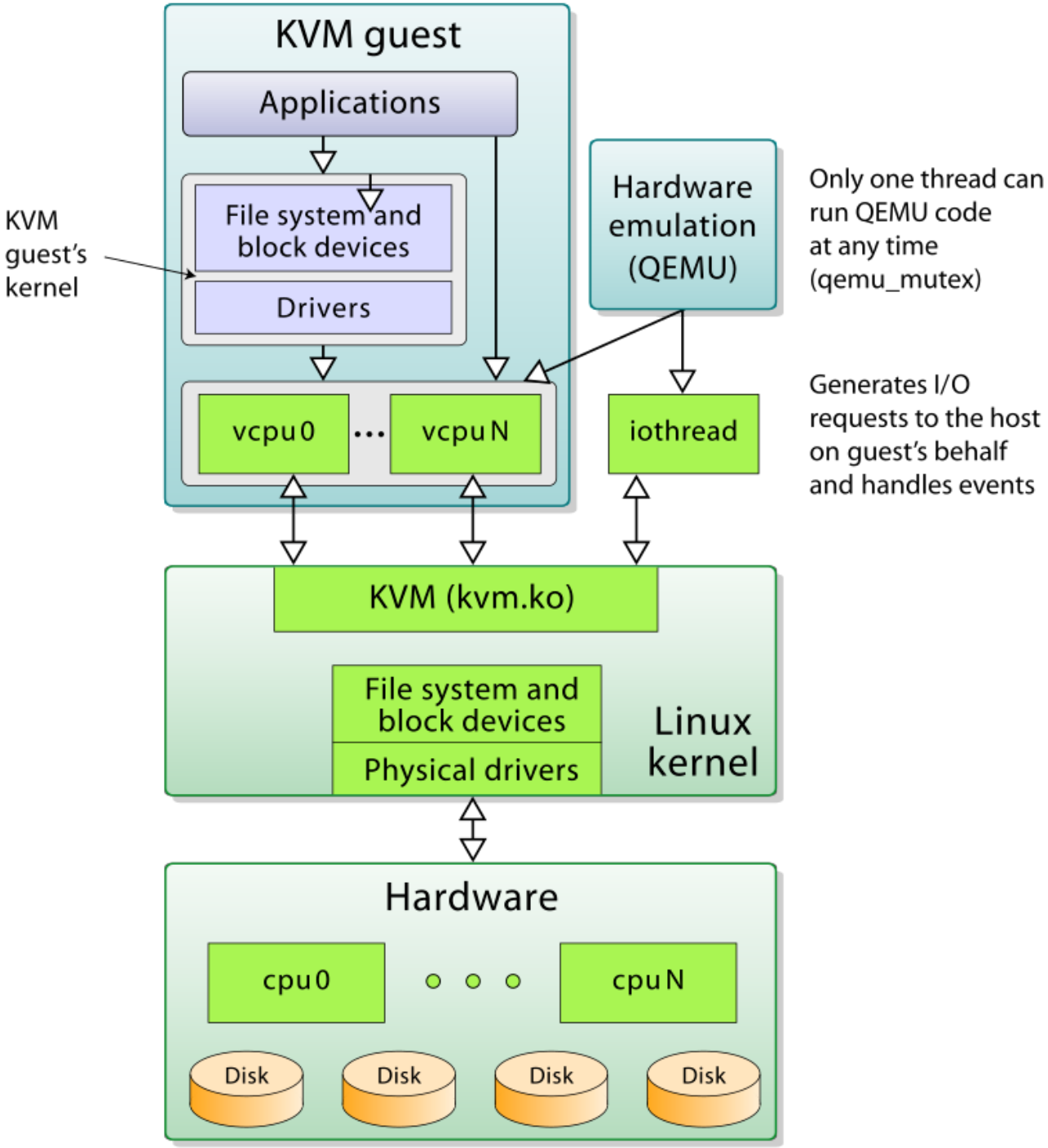
容器（应用级）：容器是一种更加轻量的应用级虚拟化技术，将应用的可执行文件及其所需的运行时环境与依赖库打包，实现一次构建，到处运行的目标。相比虚拟化，容器技术多了容器引擎层（如Docker），但上层应用无需与Guest OS绑定，可以实现秒级部署、跨平台迁移，灵活的资源分配，弹性调度管理等优势。容器、微服务与DevOps为云原生的三大要素，是推动企业技术中台建设与微服务化转型不可或缺的组件。

实现-KVM

KVM是基于虚拟化扩展（Intel VT 或者 AMD-V）的 X86 硬件的开源的 Linux 原生的全虚拟化解决方案。KVM 中，虚拟机被实现为常规的 Linux 进程，由标准 Linux 调度程序进行调度；虚机的每个虚拟 CPU 被实现为一个常规的 Linux 进程。这使得 KMV 能够使用 Linux 内核的已有功能。

但是，KVM 本身不执行任何硬件模拟，需要客户空间程序通过 /dev/kvm 接口设置一个客户机虚拟服务器的地址空间，向它提供模拟的 I/O，并将它的视频显示映射回宿主的显示屏。目前

这个应用程序是 QEMU。



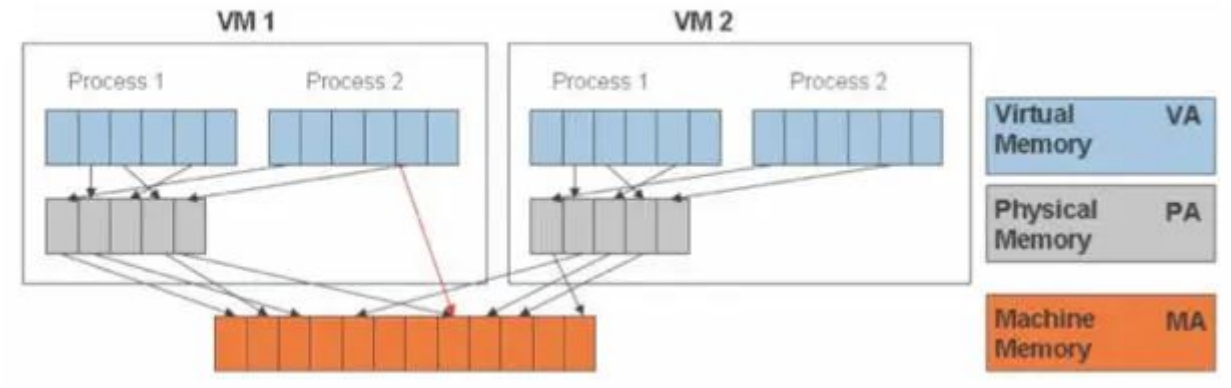
- **Guest**：客户机系统，包括CPU（vCPU）、内存、驱动（Console、网卡、I/O 设备驱动等），被 KVM 置于一种受限制的 CPU 模式下运行。
- **KVM**：运行在内核空间，提供CPU 和内存的虚级化，以及客户机的 I/O 拦截。Guest 的 I/O 被 KVM 拦截后，交给 QEMU 处理。
- **QEMU**：修改过的为 KVM 虚机使用的 QEMU 代码，运行在用户空间，提供硬件 I/O 虚拟化，通过 IOCTL /dev/kvm 设备和 KVM 交互。

KVM依赖的Intel/AMD 处理器的各种**虚拟化扩展**：

处理器	CPU 虚拟化	内存虚拟化	PCI Pass-through

Intel	VT-x	VPID，EPT	VT-d
AMD	AMD-V	ASID，NPT	IOMMU

内存虚拟化



除了CPU虚拟化，另一个关键是**内存虚拟化**，通过内存虚拟化共享物理系统内存，动态分配给虚拟机。虚拟机的内存虚拟化很象现在的操作系统支持的虚拟内存方式，应用程序看到邻近的内存地址空间，这个地址空间无需和下面的物理机器内存直接对应，操作系统保持着虚拟页到物理页的映射。

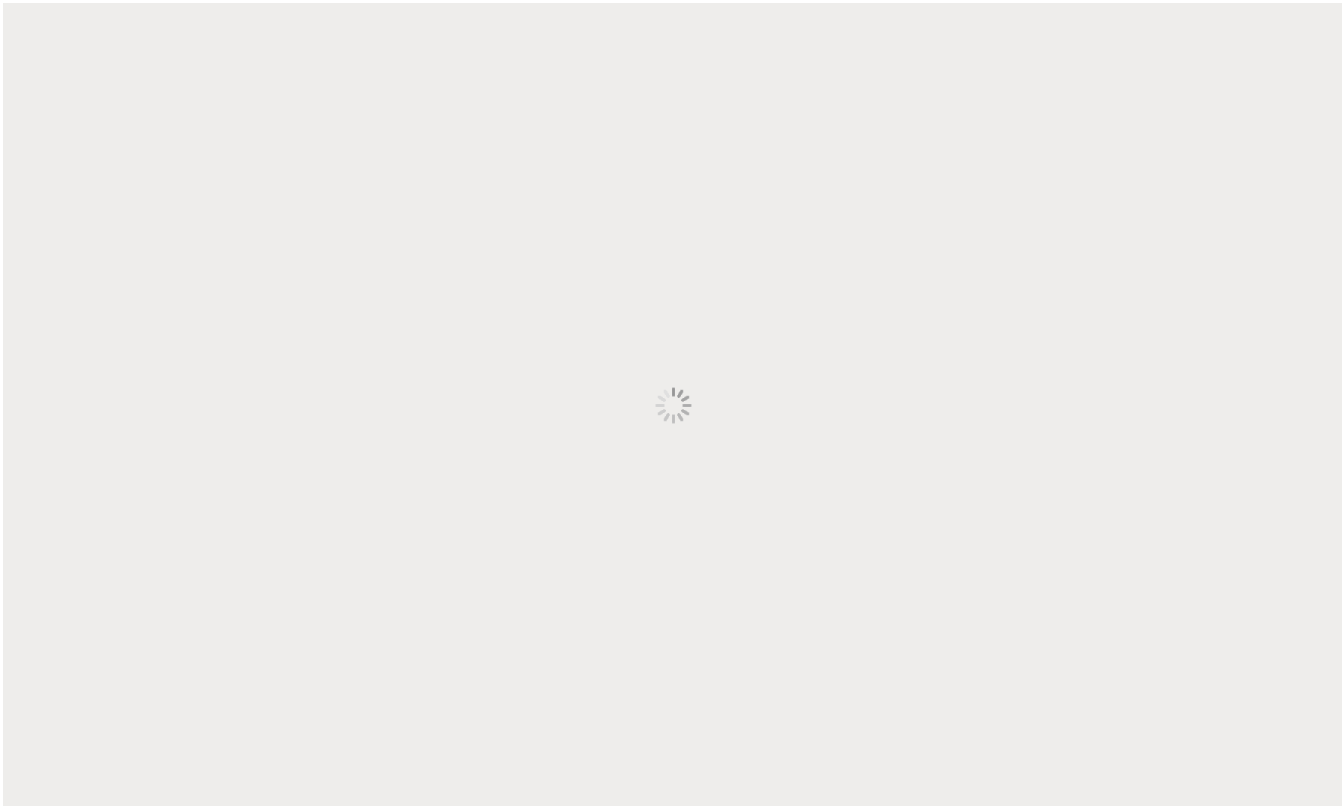
实现

现在所有的 x86 CPU 都包括了一个称为内存管理的模块**MMU**（Memory Management Unit）和 **TLB**(Translation Lookaside Buffer)，通过MMU和TLB来优化虚拟内存的性能。

KVM中，虚机的物理内存即为 qemu-kvm 进程所占用的内存空间。KVM 使用 CPU 辅助的内存虚拟化方式。在 Intel 和 AMD 平台，其内存虚拟化的实现方式分别为：

- AMD 平台上的 **NPT** （Nested Page Tables） 技术
- Intel 平台上的 **EPT** （Extended Page Tables）技术

EPT 和 NPT采用类似的原理，都是作为 CPU 中新的一层，用来将客户机的物理地址翻译为主机的物理地址。关于 EPT， Intel 官方文档中的技术如下:

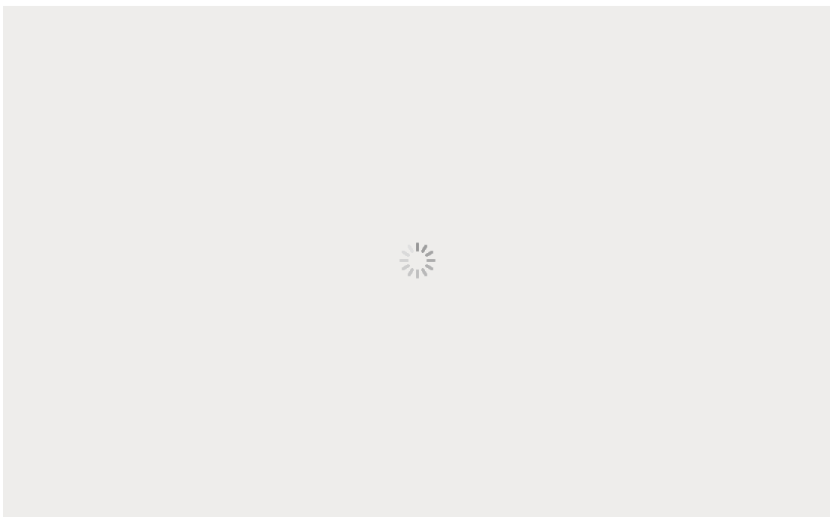


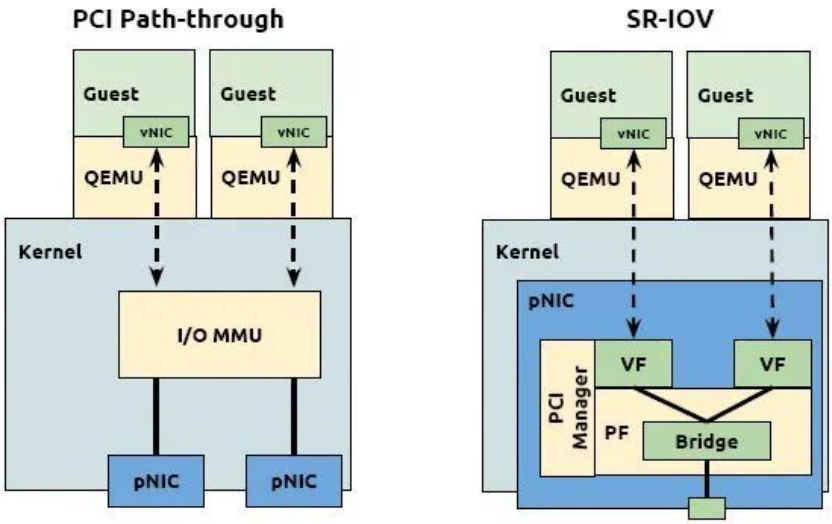
EPT的好处是，它的两阶段记忆体转换，特点就是将 Guest Physical Address → System Physical Address，VMM不用再保留一份 SPT (Shadow Page Table)，以及以往还得经过 SPT 这个转换过程。除了降低各部虚拟机器在切换时所造成的效能损耗外，硬体指令集也比虚拟化软体处理来得可靠与稳定。

I/O虚拟化

I/O虚拟化（Input/output virtualization，简称IOV）是虚拟化的一种新形式，是来自物理连接或物理运输上层协议的抽象，让物理服务器和虚拟机可以共享I/O资源。

I/O虚拟化实现

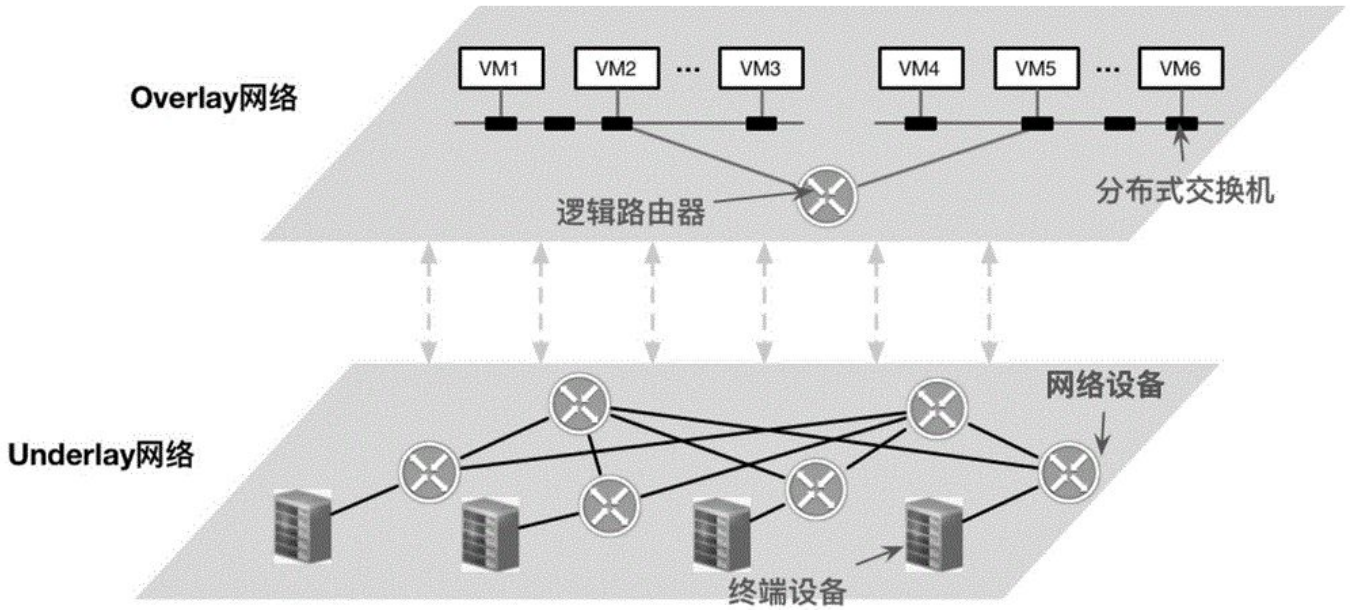




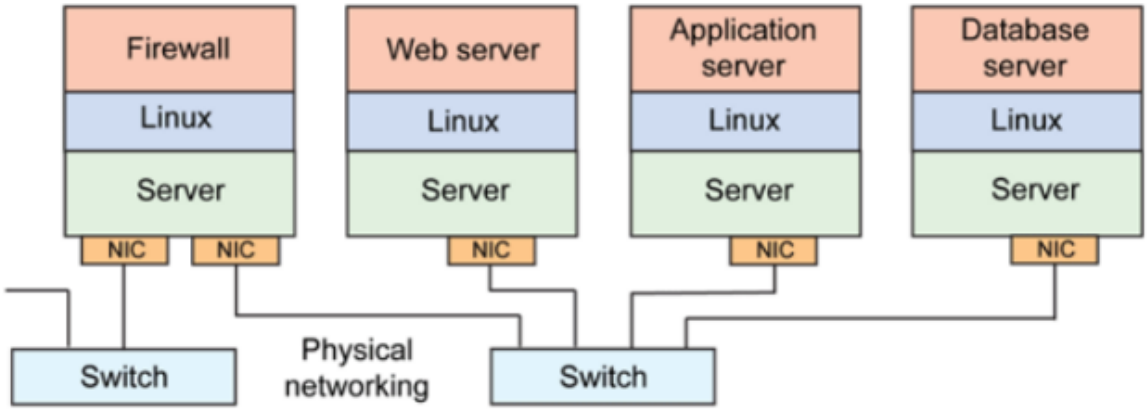
I/O 虚拟化方案的选择：

- I/O设备尽量使用准虚拟化（virtio 和 vhost_net）
- 如果需要实时迁移，不能使用 SR-IOV
- 对更高I/O要求又不需要实时迁移的，可以使用 SR-IOV
- 每种方案都有优势和不足，在特定环境下其性能有可能反而下降，因此在生产环境中使用各种虚拟化方式前需要经过完整测试

网络虚拟化



网络虚拟化 (NV) 是指将传统上在硬件中交付的网络资源抽象化到软件中。NV 可以将多个物理网络整合为一个基于软件的虚拟网络，或者可以将一个物理网络划分为多个隔离和独立的虚拟网络。



传统网络

- 在传统网络环境中，一台物理主机包含一个或多个网卡（NIC），要实现与其他物理主机之间的通信，需要通过自身的 NIC 连接到外部的网络设施，如交换机上。
- 这种架构下，为了对应用进行隔离，往往是将一个应用部署在一台物理设备上，这样会存在两个问题，

  1）是某些应用大部分情况可能处于空闲状态，

  2）是当应用增多的时候，只能通过增加物理设备来解决扩展性问题。不管怎么样，这种架构都会对物理资源造成极大的浪费。

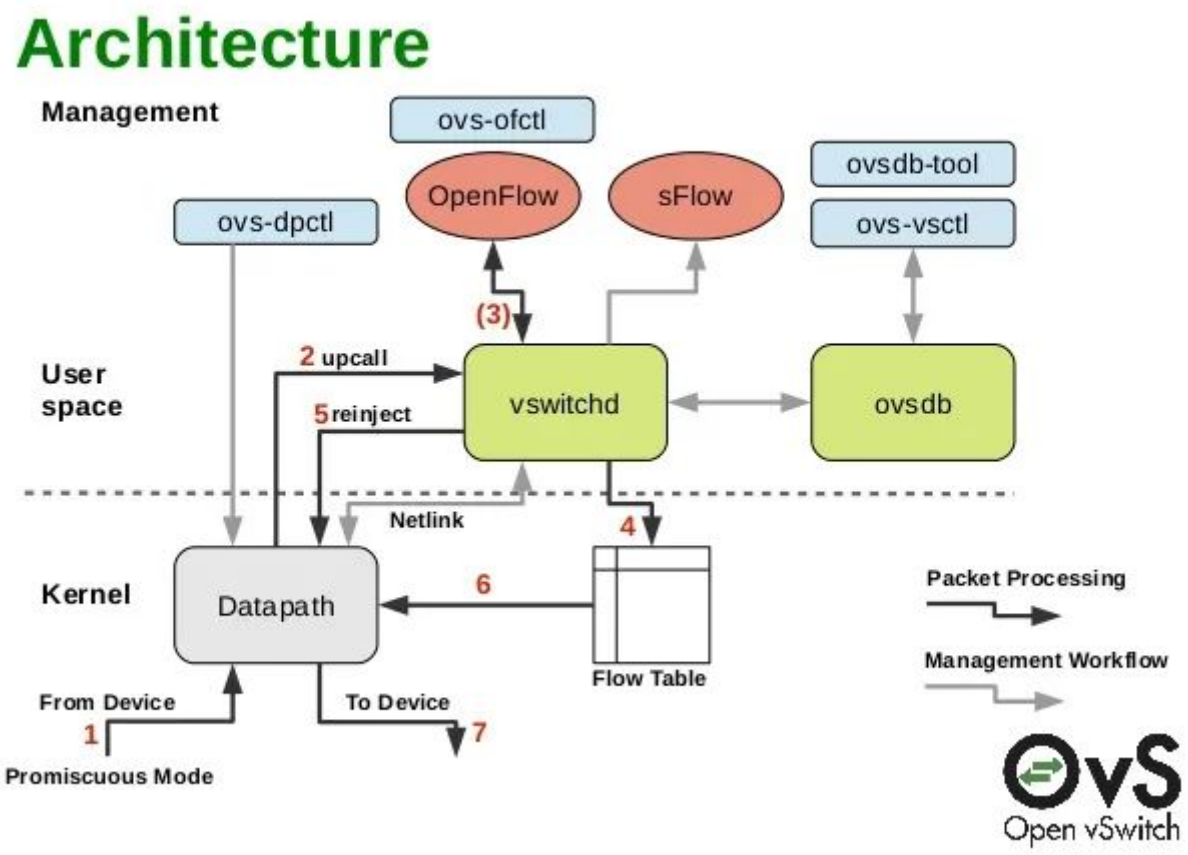
虚拟化网络

其中虚拟机与虚拟机之间的通信，由虚拟交换机完成，虚拟网卡和虚拟交换机之间的链路也是虚拟的链路，整个主机内部构成了一个虚拟的网络，如果虚拟机之间涉及到三层的网络包转发，则又由另外一个角色——虚拟路由器来完成。

虚拟化网络实现

虚拟交换机-OVS

Open vSwitch 是在开源 Apache 2 许可下获得许可的多层软件交换机。我们的目标是实现一个生产质量交换平台，该平台支持标准管理接口，并将转发功能开放给程序化扩展和控制。非常适合用作 VM 环境中的虚拟交换机。除了向虚拟网络层公开标准控制和可见性接口外，它还旨在支持跨多个物理服务器的分布。支持多种基于 Linux 的虚拟化技术，包括 Xen/XenServer、KVM 和 VirtualBox.

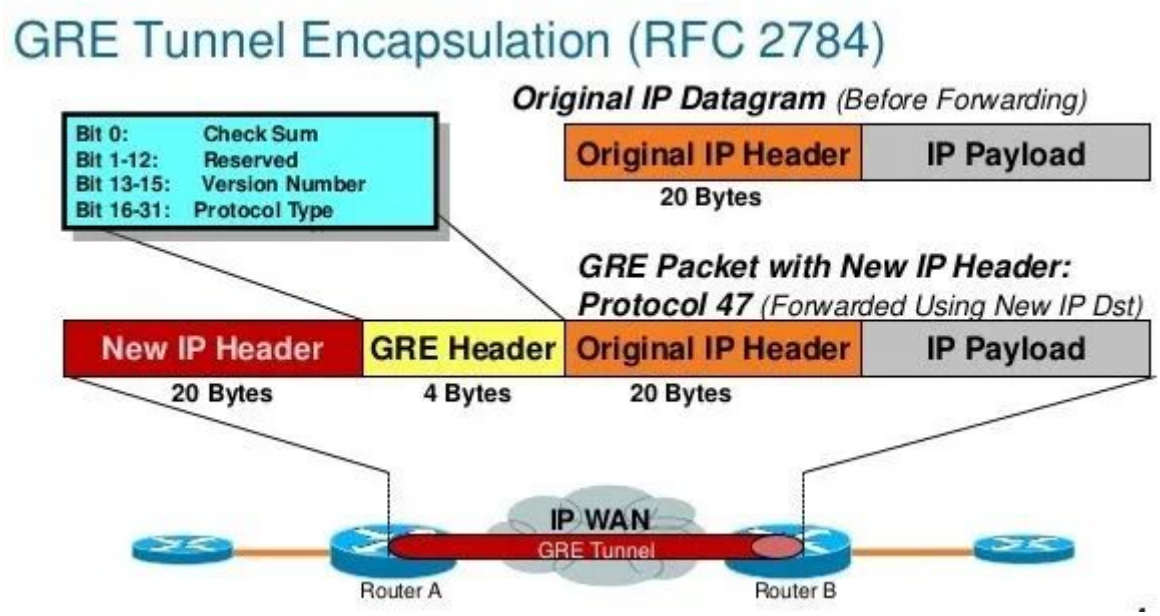


- datapath是负责数据交换的内核模块，其从网口读取数据，并快速匹配Flowtable中的流表项，成功的直接转发，失败的上交vswitchd处理。它在初始化和port binding的时候注册钩子函数，把端口的报文处理接管到内核模块。
- vswitchd是一个守护进程，是ovs的管理和控制服务，通过unix socket将配置信息保存到ovsdb，并通过netlink和内核模块交互。
- ovsdb则是ovs的数据库，保存了ovs配置信息。

虚拟网络实现技术-Overlay

当前主流overlay技术是 GRE 和 VXLAN技术. 通过增加扩展报文头来实现虚拟网络在物理网络之上传输报文。

GRE



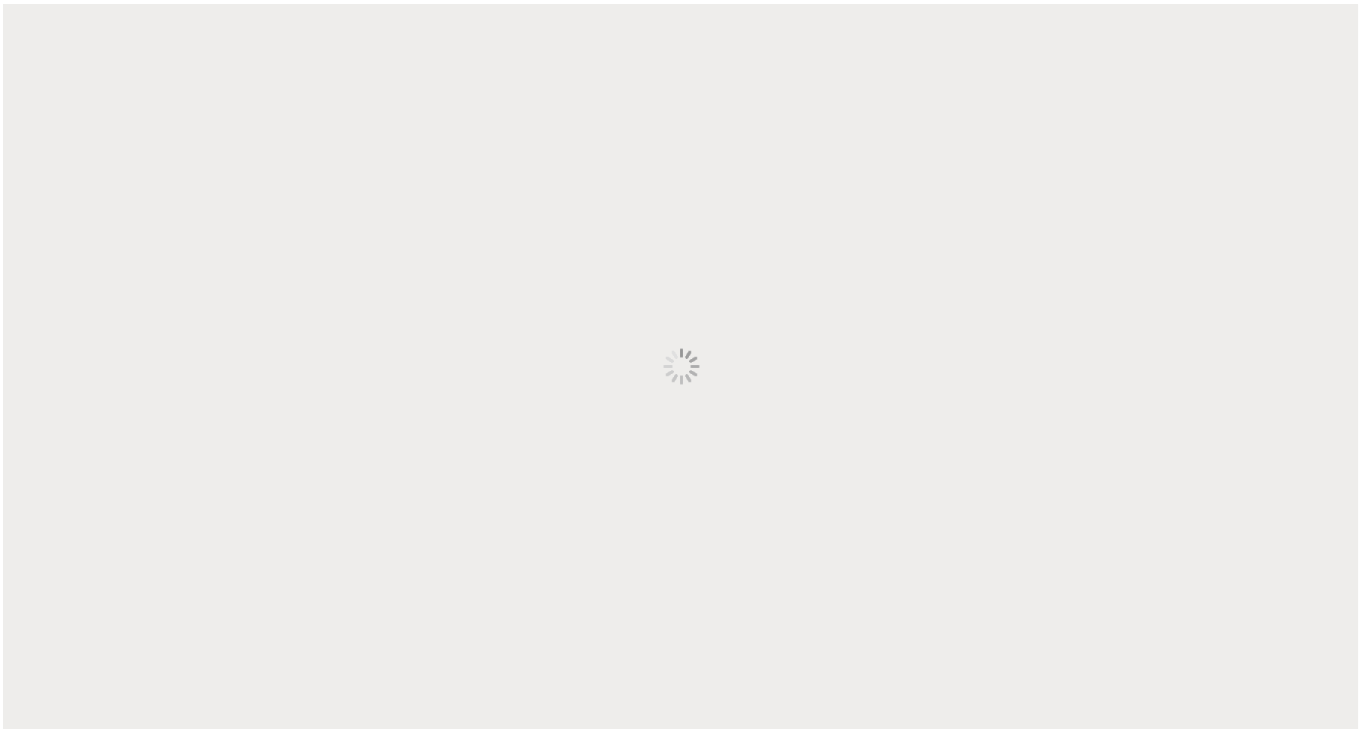
网络虚拟化使用通用路由封装 (NVGRE) 作为虚拟化 IP 地址的机制。在 NVGRE 中，虚拟机的数据包封装在另一个数据包中。此新 NVGRE 格式数据包的标头具有相应的源和目标提供程序区域 (PA) IP 地址。此外，它还具有 VSID (24 位虚拟子网 ID) ，该 ID 存储在新数据包的 GRE 标头中。

VXLAN

Virtual eXtensible Local Area Network (VXLAN) 是一种将2层报文封装到UDP包(Mac in UDP)中进行传输的一种封装协议。VXLAN主要是由Cisco推出的，VXLAN的包头有一个24bit的ID段，即意味着1600

万个独一无二的虚拟网段，这个ID通常是对UDP端口采取伪随机算法而生成的（UDP端口是由该帧中的原始MAC Hash生成的）。这样做的好处是可以保证基于5元组的负载均衡，保存VM之间数据包的顺序。

容器网络实现



CNI(Container Network Interface) 是 google 和 CoreOS 主导制定的容器网络标准，它 是在 RKT 网络提议 的基础上发展起来的，综合考虑了灵活性、扩展性、IP分配、多网卡等因素。CNI旨在为容器平台提供网络的标准化。不同的容器平台（比如目前的 Kubernetes、Mesos 和 RKT）能够通过相同的接口调用不同的网络组件。这个协议连接了两个组件：容器管理系统和网络插件，具体的事情都是插件来实现的，包括：创建容器网络空间（network namespace）、把网络接口（interface）放到对应的网络空间、给网络接口分配 IP 等。

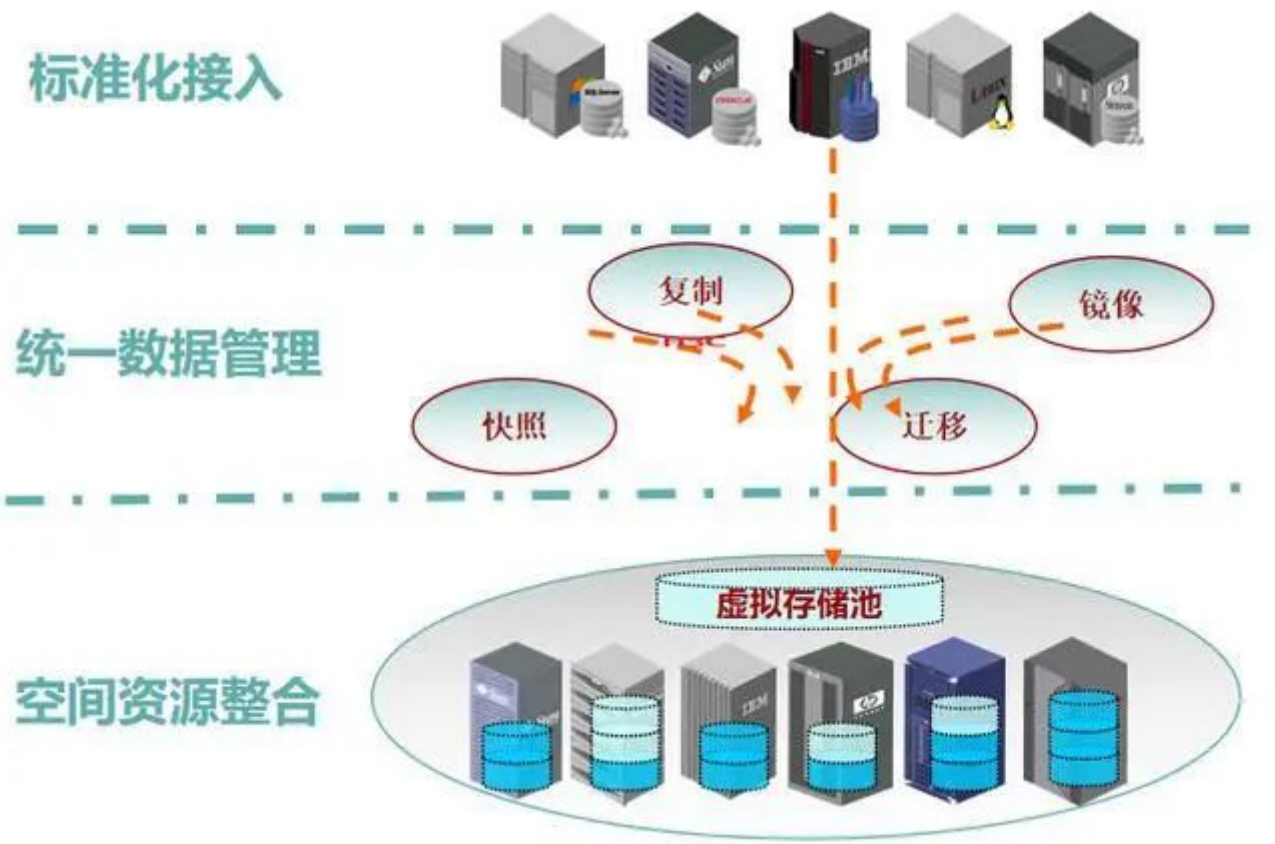
Kubernetes本身并不负责网络通信，Kubernetes提供了容器网络接口CNI（Container Network Interface），具体的网络通信交给CNI插件来负责，开源的CNI插件非常多，像Flannel、Calico等

	Calico	Flannel	Macvlan	Weave
网络模型	三层网络	VxLAN or UDP Channel	二层网络	VxLAN or UDP Channel
应用隔离	策略方式	Cidr	Cidr	Cidr
支持的协议	TCP, UDP, ICMP & ICMPv6	All	All	All
DNS支持	No	No	No	Yes
分布式存储的要求	Yes	Yes	No	No
加密通道	No	TLS	NaCl Library	No
容器子网限制	No	No	No	No

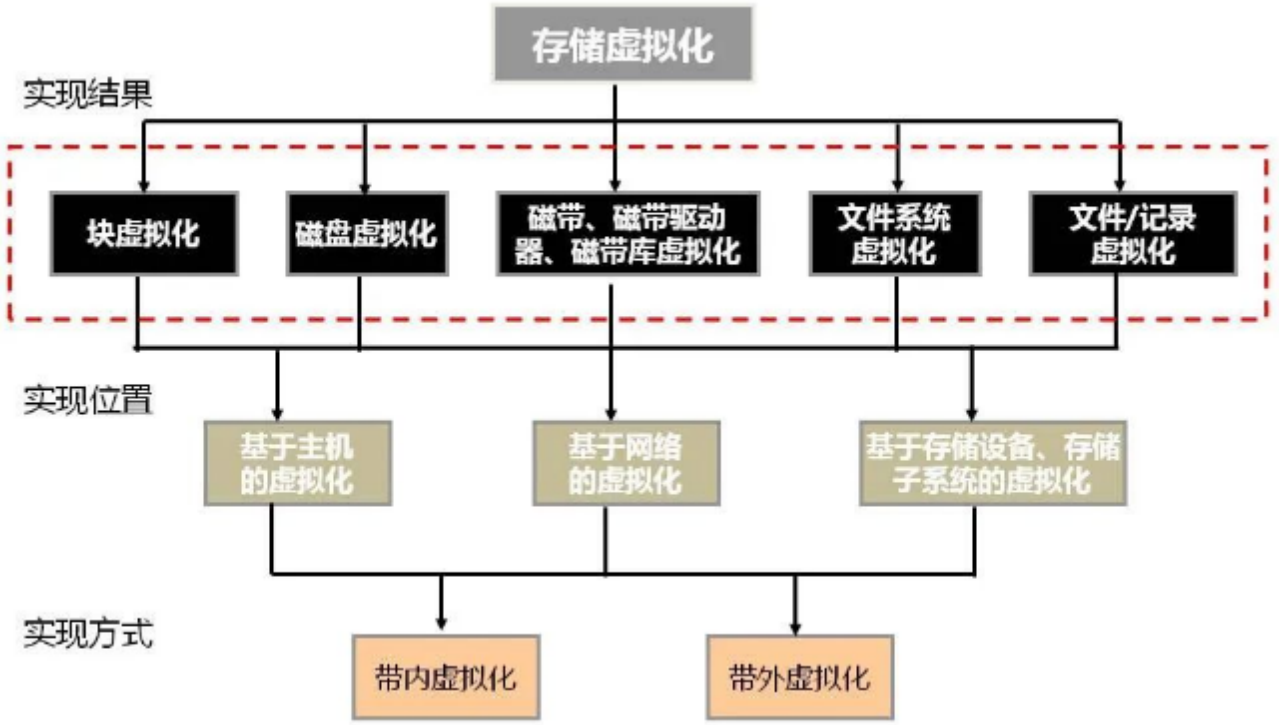
容器网络实现

存储虚拟化

存储虚拟化(Storage Virtualization)最通俗的理解就是对存储硬件资源进行抽象化表现。构建具有统一逻辑视图的存储资源池供用户按需使用。存储虚拟化将各个分散的存储系统 进行整合和统一管理，并提供了方便用户调用资源的接口。另外，存储虚拟化能够为后续的系统扩容提供便 利，使资源规模动态扩大时无需考虑新增的物理存储资源（如不同型号的存储设备）之间可能存在的差异。



实现



存储虚拟化的实现方式：

（1） 裸金属+逻辑卷

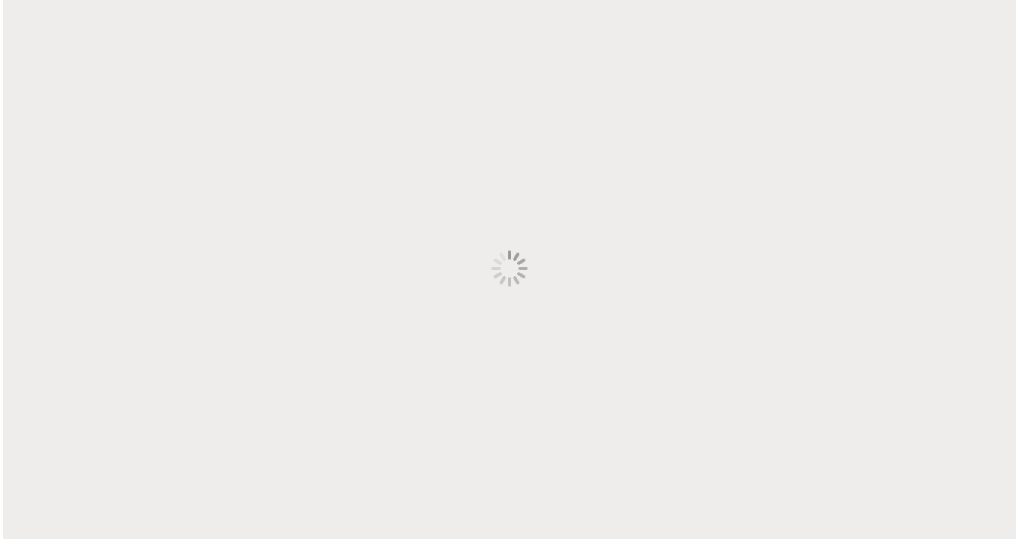
（2） 存储设备虚拟化

（3） 主机存储虚拟化+文件系统

存储虚拟化分类

文件、块和对象是三种以不同的方式来保存、整理和呈现数据的存储格式。这些格式各有各的功能和限制。

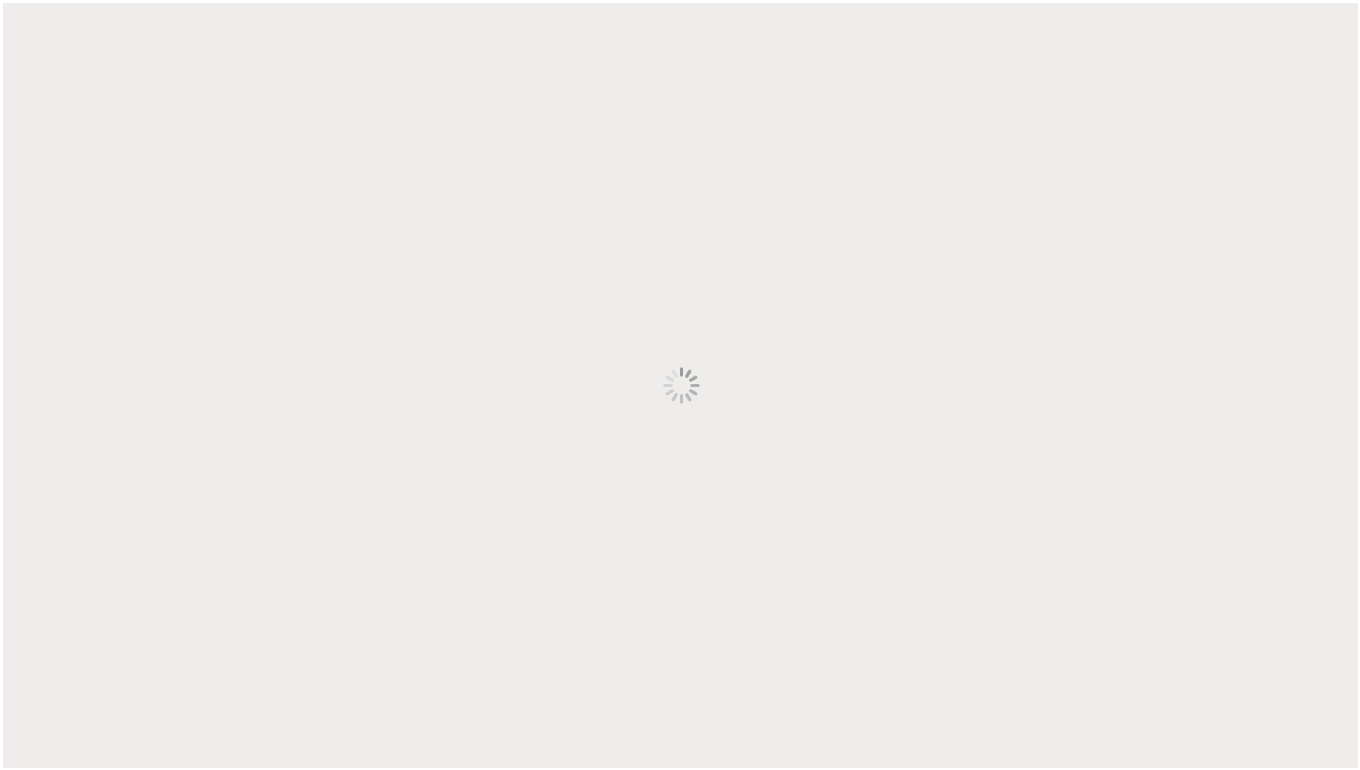
- 文件存储会以文件和文件夹的层次结构来整理和呈现数据；
- 块存储会将数据拆分到任意划分且大小相同的卷中；
- 对象存储会管理数据并将其链接至关联的元数据。



- **块存储**：即提供裸的块设备服务，裸设备什么都没有，需要用户自己创建分区、创建文件系统、挂载到操作系统才能用，挂一个块存储设备到操作系统，相当于插一个新U盘。只实现了read、write、ioctl等接口。SAN、LVM、Ceph RBD、OpenStack Cinder等都属于块存储服务。
- **文件存储**：可以简单理解为分布式文件系统，通常实现了POSIX接口，不需要安装文件系统，直接像NFS一样挂载到操作系统就能用。典型的文件存储如NAS、HDFS、CephFS、GlusterFS、OpenStack Manila等。
- **对象存储**：提供Web存储服务，通过HTTP协议访问，只需要Web浏览器即可使用，不需要挂载到本地操作系统，实现的接口如GET、POST、DELETE等，典型的对象存储如百度网盘、S3、OpenStack Swift、Ceph RGW等。

虚拟化管理工具

虚拟化管理工具是指与虚拟化环境及背后的实体硬件对接的软件，它的作用是简化资源管理、分析数据并简化运维。每个虚拟化管理系统都各不相同，但大多数系统都会提供简单的用户界面，还能简化虚拟机（VM）创建流程、监控虚拟环境、分配资源、编译报告，以及自动执行规则。



libvirt是一套用于管理硬件虚拟化的开源API、守护进程与管理工具。此套组可用于管理KVM、Xen、VMware ESXi、QEMU及其他虚拟化技术。libvirt内置的API广泛用于云解决方案开发中的虚拟机监视器编排层（Orchestration Layer）。

参考

https://kkutysllb.cn/2019/05/11

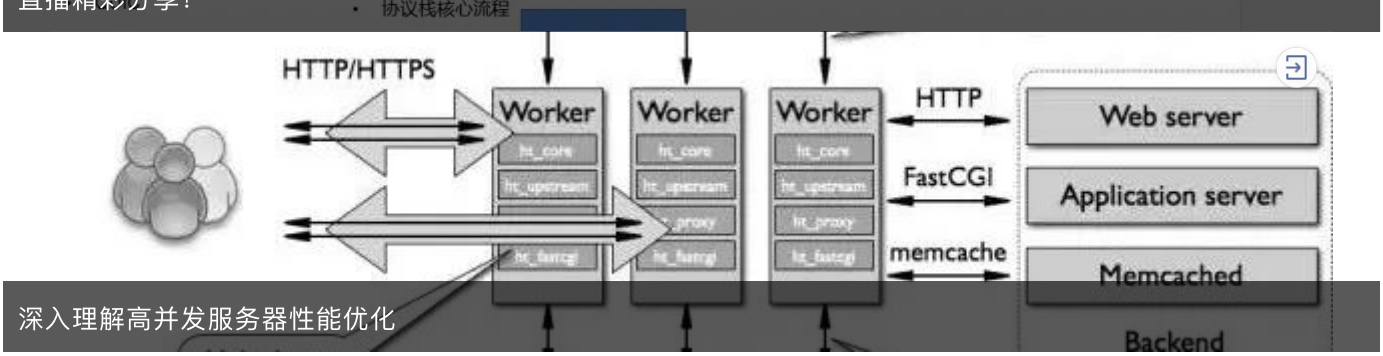
https://www.sdnlab.com/23191.html

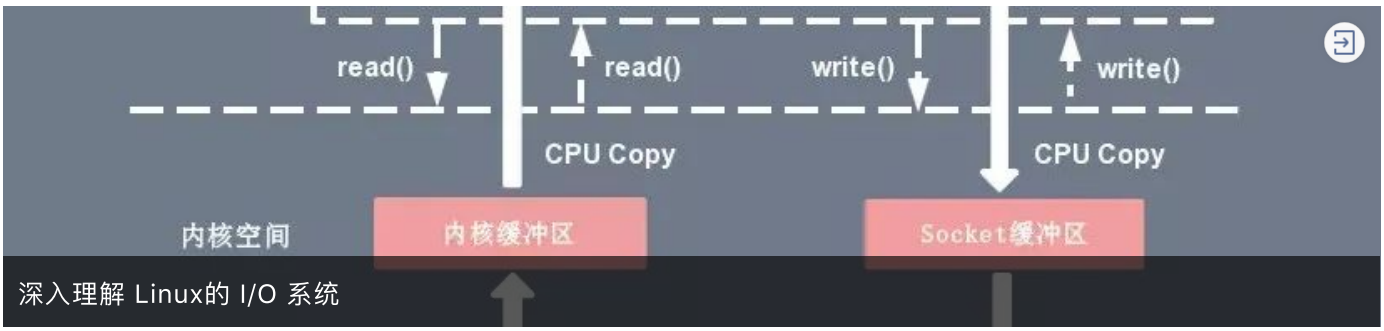
https://blog.csdn.net/sdulibh/article/details/52703687

https://cloud.51cto.com/art/202011/632813.htm

https://www.zhihu.com/question/2583484

- END -





收录于话题 #深入理解Linux系统 29

< 上一篇
深入理解Linux异步I/O框架 io_uring

下一篇 >
深入理解Linux内存子系统

文章已于2021/11/22修改

喜欢此内容的人还喜欢

Docker#(一)认识Docker

小怂读书

工业嵌入式系统的低延迟Linux内核（1）

毛毛虫AI进化之旅

Liunx服务器中使用dmidecode查询硬件信息

服务器测试