关注“架构师技术联盟”微信公众号专注技术架构和行业解决方案，构建专业交流平台，分享一线技术实践，洞察行业前沿趋势，内容覆盖云计算、大数据、超融合、软件定义网络、数据保护和解决方案。

《虚拟化技术最详细解析》

微店：http://weidian.com/s/343499232?wfr=c

目 录

（本文内容大部分自于各方渠道收集整理，用于交流学习）

1 虚拟化技术简介

1.1 虚拟化技术简介

1. 二进制翻译（Binary translation）

VMware 最早期的虚拟化是通过二进制翻译的方式，将虚拟机内的指令全部截获，并通过宿主机翻译之后执行。这样做的主要原因是 x86 的指令设计上有很大的缺陷，有很多特殊的敏感指令不能让客户机操作系统直接运行，否则会有安全问题。而二进制翻译的过程相当于做了指令级的监管，自然它的效率就很低。

2. 半虚拟化（Para-virtualization）

随着 Xen 的出现，半虚拟化成为了虚拟化技术的主流，半虚拟化通过修改客户机操作系统，让其感知到自己处于虚拟化环境下，并修改对敏感指令的使用来弥补上文提到的敏感指令的漏洞。这样虽然不是所有的指令都被翻译过，加快了速度，但是必须修改客户机操作系统也带来了诸多限制，比如无法启动 Windows 虚拟机。

3. 全虚拟化/硬件辅助的虚拟化（Hardware assisted virtualization）

硬件自身的问题，终极的解决方案还是需要硬件本身来完成，随着对虚拟化技术的需求提升，Intel 率先在自己的 CPU 上添加了 VT 技术，即由硬件来支持虚拟化环境。通过十年来的发展，目前硬件辅助的虚拟化已经成为主流。

1.1.1 虚拟化技术发展史

虚拟化技术最早出现在大型机时代。上世纪 60 年代，IBM 开始在其 CP-40 大型机系统中尝试虚拟化的实现，后来在 System/360-67 中采用，并衍生出 VM/CMS 到后来的 z/VM 等产品线。大型机上的虚拟化技术在之后 20 多年的发展中愈发成熟，但随着小型机以及 x86 的流行，大型机在新兴的服务器市场中已经失去了影响力。

由于处理器架构的不同，在大型机上已经成熟的虚拟化技术却并不能为小型机及 x86 所用。直到 2001 年，VMware 发布了第一个针对 x86 服务器的虚拟化产品。之后的几年间，英国剑桥大学的一位讲师发布了同样针对 x86 虚拟化的开源虚拟化项目 Xen，并成立 XenSource 公司（07 年 Citrix 将其收购）；惠普发布了针对 HP-UX 的 Integrity 虚拟机；Sun 跟 Solaris 10 一同发布了同时支持 x86/x64 和 SPARC 架构的 Solaris Zone；而微软也终于在 2008 年发布的 Windows Server 2008 R2 中加入了 Hyper-V。期间，VMware 被 EMC 收购，XenSource 则被思杰收购。

之后的几年间，VMware 逐渐在企业级市场中被广泛的接受，Xen 也逐渐在互联网领域展露头角。在成熟的服务器操作系统当中，Novell SUSE Linux Enterprise 10 是第一个采用 Xen 技术的。当时的 Xen 还很不成熟，乃至于红帽还为此取笑了 Novell 一番；不过几个月后，到了 RHEL 5.0 发布的时候，红帽决定也将 Xen 加入到自己的默认特性当中——那是 2006 年。一时之间，在 Linux 服务器领域，Xen 似乎成为了 VMware 之外的最佳虚拟化选择（事实上也没多少其他可选的）。

但是，作为一项 Linux 平台上的虚拟化技术，Xen 在很长一段时间内一直没有被接受到 Linux 内核的代码当中，这对于 Xen 的维护者而言，不仅意味着要多做很多工作，还意味着用户在废了半天劲装好 Xen 之后可能遇到意想不到的问题（注：2011 年 6 月发布的 Linux 内核 3.0 中已经加入了对 Xen 的支持——Xen 的工程师们表示这是清理了 7 年遗留代码、提交了 600 多个补丁的成果）。

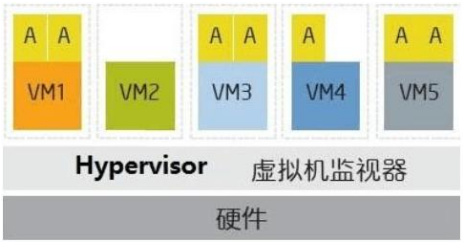
而红帽方面，也许是因为当时对这种脱离内核的维护方式很不爽，也许是因为采用 Xen 的 RHEL 在企业级虚拟化方面没有赢得太多的市场，也许是因为思杰跟微软走的太近了，种种原因，导致其萌生了放弃 Xen 的心思。2008 年 9 月，红帽收购了一家名叫 Qumranet 的以色列小公司，由此入手了一个叫做 KVM 的虚拟化技术（KVM，全称 Kernel-based Virtual Machine，意为基于内核的虚拟机）。事实上，当时整个 Xen 的市场表现的确一般，2008 年 Hyper-V 推出的时候，甚至有评论猜测思杰自己都会抛弃 Xen 而投奔 Hyper-V（当然，思杰后来在官方博客上否认了这个猜测，表示自己和微软只是合作的比较亲密而已）。

总之红帽决定选择了一个新兴的基于内核的虚拟化技术：KVM。而在正式采用 KVM 一年后，就宣布在新的产品线中彻底放弃 Xen，集中资源和精力进行 KVM 的工作。

至此各大虚拟化技术 VMware，Xen，KVM 等均出现，并找到自己位置。

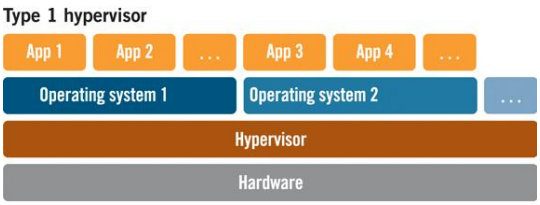
1.1.2 虚拟化技术核心 - Hypervisor 简介

在了解虚拟化技术之前，首先要了解什么是 Hypervisor，Hypervisor 是一种运行在物理服务器和操作系统之间的中间软件层,可允许多个操作系统和应用共享一套基础物理硬件，因此也可以看作是**虚拟环境中的“元”操作系统**，它可以协调访问服务器上的所有物理设备和虚拟机，也叫虚拟机监视器（Virtual Machine Monitor **VMM**）。**Hypervisor 是所有虚拟化技术的核心。**非中断地支持多工作负载迁移的能力是 Hypervisor 的基本功能。当服务器启动并执行 Hypervisor 时，它会给每一台虚拟机分配适量的内存、CPU、网络和磁盘，并加载所有虚拟机的客户操作系统。

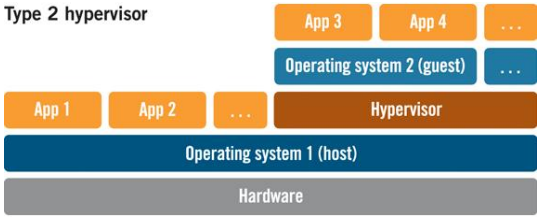


目前市场上各种 x86 管理程序(hypervisor)的架构存在差异，三个最主要架构类别包括：裸机型，托管型/主机型以及操作系统/容器型 Hypervisor，前两种最常见。

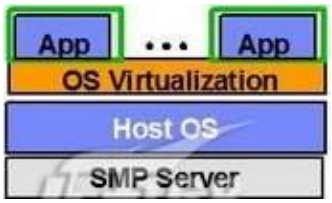
- 裸机型 Hypervisor Bare-metal(业界有时会统称其为 Type 1 Hypervisor)：最为常见，直接安装在硬件计算资源上，直接管理和调用硬件资源，不需要底层操作系统，可以理解为 Hypervisor 被做成了一个很薄的操作系统。操作系统安装并且运行在 Hypervisor 之上。主流的虚拟化产品都使用裸机型的 Hypervisor，其中包括 VMware ESX Server、Microsoft Hyper-V 和 Citrix XenServer。此种方案的性能处于主机虚拟化和操作系统虚拟化之间。



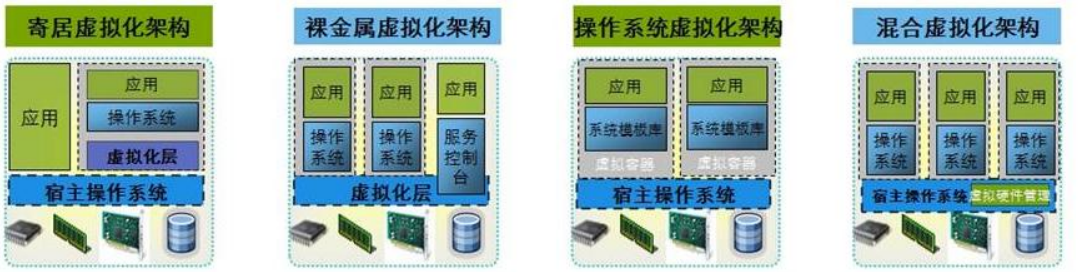
- 主机型的 Hypervisor Hosted(业界有时会统称其为 Type 2 Hypervisor)：也有一些这样的 Hypervisor 可以内嵌在硬件计算资源的固件套装中——和主板 BIOS 位于同一级别。也就是跑在操作系统上的应用软件。托管型/主机型 Hypervisor 运行在基础操作系统上，构建出一整套虚拟硬件平台（CPU/Memory/Storage/Adapter），使用者根据需要安装新的操作系统和应用软件，底层和上层的操作系统可以完全无关化，如 Windows 运行 Linux 操作系统。主机虚拟化中 VM 的应用程序调用硬件资源时需要经过:VM 内核->Hypervisor->主机内核，因此相对来说，性能是三种虚拟化技术中最差的。与使用这种方式的有 Hitachi Virtage、VMware ESXi 和 Linux KVM——基于内核的虚拟机。而宿主型的 Hypervisor 是运行在操作系统内部的应用程序，其它操作系统和应用程序实例可以运行在 VMware Server 和 Microsoft Virtual Server 之上，以及其它很多基于



终端的虚拟化平台，诸如 VMware Workstation、Microsoft Virtual PC 和 Parallels Workstation，这些都是宿主型的 Hypervisor。



- 容器/操作系统版 Hypervisor OS-Level/Container：最后一种不常用，虚拟机运行在传统操作系统上，创建一个独立的虚拟化实例（容器 Container），指向底层托管操作系统。叫操作系统版 Hypervisor，或者叫容器版 Hypervisor，OS-Level Hypervisor 有点事性能最好，耗费资源最少，缺点是操作系统唯一，如果底层操作系统跑的是 Windows，那么 VPS/VE 就都得跑 Windows，代表是 Parallels 公司（以前叫 SWsoft）的 Virtuozzo（商用产品）和 OpenVZ（开源项目）
- 混合虚拟化架构：有些 Hypervisor 架构不好定义，有时候就将其归类为混合虚拟化架构，比如 KVM，KVM 全称 Kernel-based Virtual Machine 即基于内核的虚拟机。KVM 是集成到 Linux 内核的 Hypervisor，是 X86 架构且硬件支持虚拟化技术（Intel VT 或 AMD-V）的 Linux 的全虚拟化解决方案。它是 Linux 的一个很小的模块，利用 Linux 做大量的事，如任务调度、内存管理与硬件设备交互等。如果你将 Linux 视为 OS，KVM 视为基于 OS 之上的 Hypervisor，那么就属于 Type 2 架构，如果将 Linux+KVM 视为 Hypervisor，那么 KVM 就属于 Type1 架构，业界关于 KVM 的归类一直在争论中，有些人就将 KVM 归为混合虚拟化架构了。
- 各类 Hypervisor 对比



	寄居虚拟化架构	裸金属虚拟化架构	操作系统虚拟化架构	混合虚拟化架构
优点	• 简单、易于实现	• 虚拟机不依赖于操作系统 • 支持多种操作系统，多种应用	• 简单、易于实现 • 管理开销非常低	• 相对于寄居虚拟化架构，没有冗余，性能高 • 可支持多种操作系统
缺点	• 安装和运行应用程序依赖于主机操作系统对设备的支持 • 管理开销较大，性能损耗大	• 虚拟层内核开发难度大	• 隔离性差，多容器共享同一操作系统	• 需底层硬件支持虚拟化扩展功能
厂家	VMware Workstation Vitrual PC	WMware ESXServer Citrix XenServer Microsoft Hyper-V 华为 UVP	Virtuozzo	Redhat KVM

裸金属虚拟化架构与混合虚拟化架构将是未来虚拟化架构发展的趋势

1.2 剖析系统虚拟化

简单而言，虚拟化（Virtulization）是表示计算机资源的抽象方法。通过虚拟化可以对包括基础设施，系统和软件等计算机资源的表示，访问和管理进行简化，并为这些资源提供标准的接口来接受输入和提供输出。

虚拟化技术有很多种，比如，网络虚拟化，内存的虚拟化，桌面虚拟化，应用虚拟化和虚拟内存等等。因为篇幅的原因，本系列将重点关注系统虚拟化，特别是 X86 平台。今后此系列当中提到的虚拟化皆指系统虚拟化。

系统虚拟化的目的通过使用虚拟化管理器（Virtual Machine Monitor，简称 VMM）是在一台物理机上虚拟和运行一台或多台虚拟机（Virtual Machine，简称 VM）。VMM 主要有两种形式：

1、Hypervisor VM，它直接运行在硬件（Bare Metal）上面，提供接近于物理机的性能，并在 I/O 上面做了特别多的优化，主要用于服务器类的应用，也被称为“Type 1”。

2、Hosted（托管）VM，它运行在物理机的操作系统上，虽然其本身性能不如 Hypervisor（因为它和硬件之间隔了一层 OS），但是其安装和使用非常方便，而且功能丰富，比如支持三维加速等特性，常用于桌面应用，也被称为“Type 2”。

1.2.1 系统虚拟机的分类

由于采用技术的不同，可以将系统虚拟化分为五大类：

1.2.1.1 硬件仿真（Emulation）

简介：属于 Hosted 模式，在物理机的操作系统上创建一个模拟硬件的程序（Hardware VM）来仿真所想要的硬件，并在此程序上跑虚拟机，而且虚拟机内部的客户操作系统（Guest OS）无需修改。知名的产品有 Bochs，QEMU 和微软的 Virtual PC（它还使用少量的全虚拟化技术）。

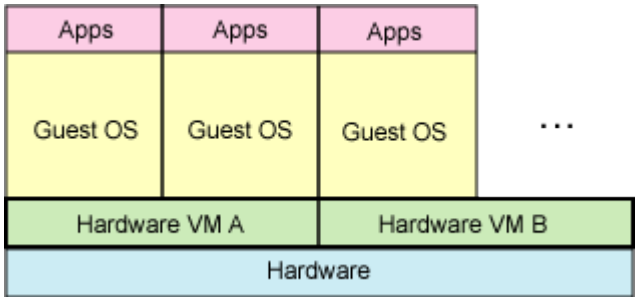
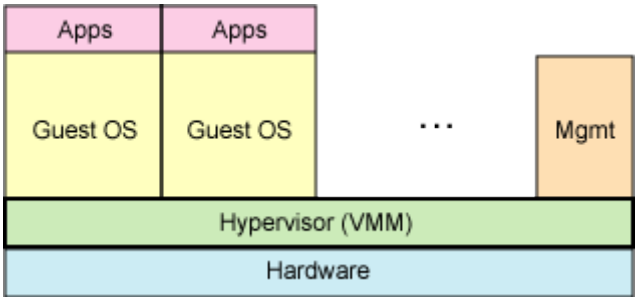


图 1. 硬件仿真架构图

优点：Guest OS 无需修改，而且非常适合用于操作系统开发，也利于进行固件和硬件的协作开发。固件开发人员可以使用目标硬件 VM 在仿真环境中对自己的实际代码进行验证，而不需要等到硬件实际可用的时候。



缺点：速度非常慢，有时速度比物理情况慢 100 倍以上。

未来：因为速度的问题，渐趋颓势，但是还应该有一席之地。

1.2.1.2 全虚拟化(Full Virtualization)

全虚拟化也成为原始虚拟化技术，该模型使用虚拟机协调 Guest 操作系统和原始硬件，VMM 在 Guest 操作系统和裸硬件之间用于工作协调，一些受保护指令必须由 Hypervisor（虚拟机管理程序）来捕获处理。

全虚拟化主要是在客户操作系统和硬件之间捕捉和处理那些对虚拟化敏感的特权指令，使客户操作系统无需修改就能运行，速度会根据不同的实现而不同，但大致能满足用户的需求。这种方式是业界现今最成熟和最常见的，而且属于 Hosted 模式和 Hypervisor 模式的都有，知名的产品有 IBM CP/CMS、VirtualBox、KVM、VMware Workstation 和 VMware ESX（它在其 4.0 版，被改名为 VMware vSphere）。

全虚拟化的运行速度要快于硬件模拟，但是性能方面不如裸机，因为 Hypervisor 需要占用一些资源。

- 优点：Guest OS 无需修改，速度和功能都非常不错，更重要的是使用非常简单，不论是 VMware 的产品，还是 Oracle 的 VirtualBox。
- 缺点：基于 Hosted 模式的全虚拟产品性能方面不是特别优异，特别是 I/O 方面。
- 未来：因为使用这种模式，不仅 Guest OS 免于修改，而且将通过引入硬件辅助虚拟化技术来提高其性能，在未来全虚拟化还是主流。

1.2.1.3 半虚拟化(Para Virtualization)

半虚拟化是另一种类似于全虚拟化技术，它使用 Hypervisor 分享存取底层硬件，但是它的 Guest 操作系统集成了虚拟化方面代码。该方法无需重新编译或引起陷阱，因为操作系统自身能够与虚拟进程进行很好的协作。

它与完全虚拟化有一些类似，它也利用 Hypervisor 来实现对底层硬件的共享访问，但是由于在 Hypervisor 上面运行的 Guest OS 已经集成与半虚拟化有关的代码，使得 Guest OS 能够非常好地配合 Hyperivosr 来实现虚拟化。通过这种方法将无需重新编译或捕获特权指令，使其性能非常接近物理机，其最经典的产品就是 Xen，而且因为微软的 Hyper-V 所采用技术和 Xen 类似，所以也可以把 Hyper-V 归属于半虚拟化。

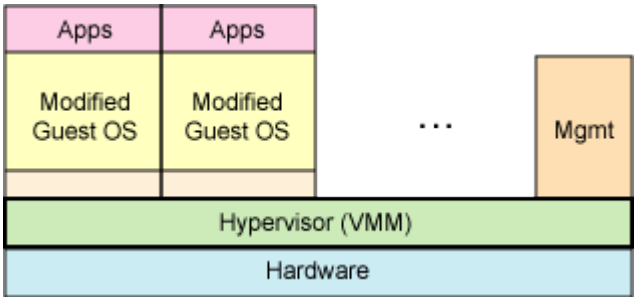
（半虚拟化由于需要修改 VM 操作系统的内核，所以没法支持 windows 这样闭源 OS，于是后来 Xen 也利用 CPU 虚拟化技术开始支持全虚拟化了，所以目前 Xen 支持半虚拟化和全虚拟化两种）

半虚拟化需要 Guest 操作系统做一些修改，使 Guest 操作系统意识到自己是处于虚拟化环境的，但是半虚拟化提供了与原操作系统相近的性能。

- 优点：这种模式和全虚拟化相比架构更精简，而且在整体速度上有一定的优势。
- 缺点：需要对 Guest OS 进行修改，所以在用户体验方面比较麻烦。

未来：情况比较类似，在公有云（比如 Amazon EC2）平台上应该继续占有一席之地，但是很难在其他方面和类似 VMware vSphere 这样的全虚拟化产品竞争，同时它也将会利用其它技术来提高速度，并简化架构。

1.2.1.4 硬件辅助虚拟化（Hardware Assisted Virtualization）



简介：Intel/AMD 等硬件厂商通过对部分全虚拟化和半虚拟化使用到的软件技术进行硬件化（具体将在下文详述）来提高性能。硬件辅助虚拟化技术常用于优化全虚拟化和半虚拟化产品，而不是独创一派，最出名的例子莫过于 VMware Workstation，它虽然属于全虚拟化，但是在它的 6.0 版本中引入了硬件辅助虚拟化技术，比如 Intel 的 VT-x 和 AMD 的 AMD-V。现在市面上的主流全虚拟化和半虚拟化产品都支持硬件辅助虚拟化，包括 VirtualBox，KVM，VMware ESX 和 Xen。

优点：通过引入硬件技术，将使虚拟化技术更接近物理机的速度。

缺点：现有的硬件实现不够优化，还有进一步提高的空间。

未来：因为通过使用硬件技术不仅能提高速度，而且能简化虚拟化技术的架构，所以预见硬件技术将会被大多数虚拟化产品所采用。

1.2.1.5 操作系统级虚拟化（Operating System Level Virtualization）

简介：这种技术通过对服务器操作系统进行简单地隔离来实现虚拟化，主要用于 VPS。主要的技术有 Parallels Virtuozzo Containers，Unix-like 系统上的 chroot 和 Solaris 上的 Zone 等。

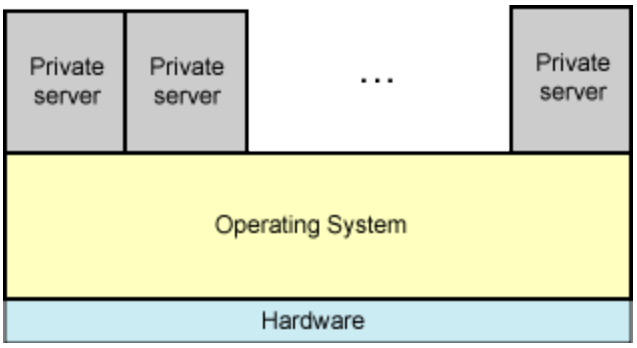


图 4. 操作系统级架构图

优点：因为它是对操作系统进行直接的修改，所以实现成本低而且性能不错。

缺点：在资源隔离方面表现不佳，而且对 Guest OS 的型号和版本有限定。

未来：不明朗，我觉得除非有革命性技术诞生，否则还应该属于小众，比如 VPS。

1.2.2 系统虚拟化的用处

1、软件测试，通过使用 VirtualBox 和 VMware Workstation 来配置测试环境，不仅比物理方式快捷很多，而且无需购买很多昂贵的硬件，更重要的是，通过它们自带的 SnapShot/Pause 功能可以非常方便地将错误发生的状态保存起来，这样将极有利于测试员和程序员之间的沟通。现在已经有很多软件都通过虚拟机的形式进行测试，最著名的例子，莫过于以 VirtualBox 虚拟机形式发布的 Chrome OS 测试版。我第一次接触到虚拟化技术强大威力就是在软件测试方面。

2、桌面应用，通过诸如 VirtualBox 和 VMware Workstation 等桌面虚拟化软件能让用户使用其他平台的专属软件，比如使用 Linux 的用户能够通过 VirtualBox 上虚拟的 Windows 环境来访问使用 ActiveX 技术的网上银行。

3、服务器整合，通过 VMware ESX 和 Xen 能够将多台物理机上的工作量整合到一台物理机上。现有普遍的整合率在 1：8 左右，也就是使用这些软件

能将原本需要八台物理机的工作量整合到一台物理机上。服务器整合不仅能减低硬件，能源和场地等开支，还能极大地简化 IT 架构的复杂度。

4、自动化管理，通过使用类似 DRS（Distributed Resource Scheduling，分布式资源调度），Live Migration（动态迁移），DPM（Distributed Power Management，分布式电源管理）和 HA（High Availability，高可用性）等高级虚拟化管理技术，能极大地提高整个数据中心的自动化管理程度。

5、加快应用部署，通过引入虚拟化应用发布格式 OVF（Open Virtualization Format），不仅能使第三方应用供应商更方便地发布应用，而且使系统管理员非常简单地部署这个应用（大多数情况下只要轻轻一个点击就可以完成整套部署工作）。

1.2.3 虚拟化的三大特征

在 1974 年 Popek 和 Goldberg 发表的虚拟化名篇《Formal Requirements for Virtualizable Third Generation Architectures》中定义了虚拟机（VM）可以被认为是物理机的一种高效隔离的复制，并指出虚拟机应具有的三大特征：

1、一致性， 一个运行于虚拟机上的程序，其行为应与直接运行于物理机上的同程序的行为基本一致，只允许有细微的差异，比如在系统时间方面。

2、可控性，VMM（虚拟化管理器）对系统资源有完全的控制能力和管理权限，包括资源的分配，监控和回收。

3、高效性，绝大多数的客户机指令应该由硬件直接执行而无需 VMM 的参与。

但是要满足这三点，并非易事，因为系统的指令集架构（ISA）需要相应地满足四个的条件：

1、CPU 能支持多个特权级，并且 VM 上面运行的指令能在底特权级（比如 Ring 3）下正确执行。

2、非特权指令（允许用户直接使用的指令）的执行效果不依赖于 CPU 的特权级。

3、敏感指令（对系统资源配置有影响的指令）都是特权指令（不允许用户直接使用的指令）。

4、必须支持一种内存保护机制来保证多个虚拟机之间在内存方面的隔离，例如段保护或页保护。

1.2.4 X86 架构如何被虚拟化

虽然 X86 架构在 PC 市场占据绝对的垄断地位，但是由于其在初始设计时，并没有考虑到虚拟化需求，所以它对虚拟化的支持不够，特别是它没有满足上面四个条件里面的第三个，其因为是 X86 的 ISA 有 17 条敏感指令（比如 LGDT 等）不属于特权指令。也就是说，当虚拟机执行到这些敏感指令的时

候，很有可能出现错误，将会影响到整个机器的稳定。更困难的是，上面所提出的问题只是 X86 虚拟化所需要面对的问题的一小部分而已，还有许许多多的问题还未涉及。

下面将分 CPU 虚拟化，内存虚拟化和 I/O 虚拟化这三部分来介绍全虚拟化，半虚拟化和硬件辅助虚拟化所采用的相关技术。

1.3 CPU 虚拟化

CPU 虚拟化的目标是使虚拟机上的指令能被正常地执行，而且效率接近物理机。

1.3.1 全虚拟化

主要采用优先级压缩（Ring Compression）和二进制代码翻译技术（Binary Translation）这两个技术。优先级压缩能让 VMM 和 Guest 运行在不同的特权级下，对 X86 架构而言，就是 VMM 运行在特权级最高 Ring 0 下，Guest 的内核代码运行在 Ring 1 下，Guest 的应用代码运行在 Ring 3 下。通过这种方式能让 VMM 截获一部分在 Guest 上执行的特权指令，并对其进行虚拟化。但是有一些对虚拟化不友好的指令则需要二进制代码翻译来处理，它通过扫描并修改 Guest 的二进制代码来将那些难以虚拟化的指令转化为支持虚拟化的指令。

1.3.2 半虚拟化

其通过修改 Guest OS 的代码，使其将那些和特权指令相关的操作都转换会发给 VMM 的 Hypercall（超级调用），而且 Hypercall 支持 Batch（批处理）和异步这两种优化方式，使得通过 Hypercall 能得到近似于物理机的速度，

1.3.3 硬件辅助虚拟化

主要有 Intel 的 VT-x 和 AMD 的 AMD-V 这两种技术，而且这两种技术在核心思想上非常相似，都是通过引入新的指令和运行模式，来让 VMM 和 Guest OS 能分别运行在其合适的模式下。在实现方面，VT-X 支持两种处理器工作方式：第一种称为 Root 模式（Operation），VMM 运行于此模式，用于处理特殊指令，另一种称为 Non-Root 模式（Operation），Guest OS 运行于此模式，当在 Non-Root 模式 Guest 执行到特殊指令的时候，系统会切换到运行于 Root 模式 VMM，让 VMM 来处理这个特殊指令。

1.4 内存虚拟化

内存虚拟化的目标是能做好虚拟机内存空间之间的隔离，使每个虚拟机都认为自己拥有了整个内存地址，并且效率也能接近物理机。

1.4.1 全虚拟化

影子页表 （Shadow Page Table），就是为每个 Guest 都维护一个“影子页表”，在这个表中写入虚拟化之后的内存地址映射关系，而 Guest OS 的页表则无需变动，最后，VMM 将影子页表交给 MMU 进行地址转换。

1.4.2 半虚拟化

页表写入法，当 Guest OS 创建一个新的页表时，其会向 VMM 注册该页表，之后在 Guest 运行的时候，VMM 将不断地管理和维护这个表，使 Guest 上面的程序能直接访问到合适的地址。

1.4.3 硬件辅助虚拟化

EPT（Extended Page Table，扩展页表），EPT 通过使用硬件技术，使其能在原有的页表的基础上，增加了一个 EPT 页表，通过这个页表能够将 Guest 的物理地址直接翻译为主机的物理地址，从而减低整个内存虚拟化所需的 Cost。

1.5 I/O 虚拟化

I/O 虚拟化的目标是不仅让虚拟机访问到它们所需要的 I/O 资源，而且要做好它们之间的隔离工作，更重要的是，减轻由于虚拟化所带来的开销。

1.5.1 全虚拟化

通过模拟 I/O 设备（磁盘和网卡等）来实现虚拟化。对 Guest OS 而言，它所能看到就是一组统一的 I/O 设备，同时 Guest OS 每次 I/O 操作都会陷入到 VMM，让 VMM 来执行。这种方式，对 Guest 而言，非常透明，无需顾忌底层硬件，比如 Guest 操作的是 SCSI 的设备，但实际物理机只有 SATA 的硬盘。

1.5.2 半虚拟化

通过前端（Front-End）/后端（Back-End）架构，将 Guest 的 I/O 请求通过一个环状队列传递到特权域（Privileged Domain，也被称为 Domain-0）。因为这种方式的相关细节较多，所以会在后文进行深入分析。

1.5.3 硬件辅助虚拟化

最具代表性莫过于 Intel 的 VT-d，AMD 的 IOMMU 和 PCI-SIG 的 IOV（I/O Virtulization）这三个技术。在这里介绍一下 VT-d，其核心思想就是让虚拟机能直接使用物理设备，但是这会牵涉到 I/O 地址访问和 DMA 的问题，而 VT-d 通过采用 DMA 重映射（Remapping）和 I/O 页表来解决这两个问题，从

而让虚拟机能直接访问物理设备。还有，IOMMU 和 VT-d 在技术上有很多相似之处。

1.6 总结

	全虚拟化	半虚拟化	硬件辅助虚拟化
CPU 虚拟化	二进制代码翻译	Hypercall	VT-x
内存虚拟化	影子页表	页表写入法	EPT
I/O 虚拟化	模拟 I/O 设备	前端/后端架构	VT-d

X86 虚拟化技术总结

随着硬件辅助虚拟化技术不断地发展和优化，将使其在速度和架构方面的优势更明显，但是由于全虚拟化和半虚拟化的一些技术在某些方面还是保持了一定的优势，比如半虚拟化的前端和后端架构和全虚拟化的二进制代码翻译技术。所以，我个人认为今后 X86 虚拟化技术的发展将会以硬件辅助虚拟化技术为主，同时以全虚拟化和半虚拟化技术为辅.

2 虚拟化厂商简介

市场主要厂商及产品：VMware vSphere、微软 Hyper-V、Citrix XenServer、IBM PowerVM、Red Hat Enterprise Virtualization、Huawei FusionSphere、开源的 KVM、Xen、VirtualBSD 等。

- Vmware:行业老大地位，市场占有率一直超过 50%，Openstack 阵营（VMware 是 EMC 独立运营子公司，03 年收购，EMC 内部讨论是否在未来将其剥离，15 年有结论）
- Xen: Citrix，之前属于 Openstack 阵营，后来退出，强力推动其 Cloudstack 阵营。公司桌面云使用 Xen 技术
- KVM: Ret Hat 目前掌握 KVM 技术，
- Hyper-V: Microsoft，Hyper-V 是基于 XEN 管理栈的修改
- VirtualBox: Oracle （开源，之前是 SUN 的，后被 Oracle 收购，用的很少）

下面的表格比较了一些主流的虚拟机软件的基本信息。还有很多这里没有列出：

名称	作者	主 CPU	客户 CPU	主系统	客户系统	许可
Hyper-V	Microsoft	x64+硬件辅助虚拟(AMD-V 或 Intel VT)	x64, x86	Windows Server 2008/ 2012, Windows 8	Windows 2000, Windows 2003, Windows 2008, Windows XP, Windows Vista, Linux	私有(免费)
KVM	Red Hat	Intel/AMD 处理器与 x86 虚拟化	x86/x86-64	Linux	Linux, Windows	GPL v2
Oracle VirtualBox	Oracle	x86, x86-64, VT-x, AMD-V	x86, (x86-64 仅在 VirtualBox 2 及	Windows, Linux, Mac OS X (Intel), Solaris,	DOS, Windows, Linux, OS/2, FreeBSD, Solaris	GPL v2

			x86-64 主机上）	FreeBSD		
Virtual PC 2007	Microsoft	x86, x86-64	x86	Windows Vista（商业版，企业版，旗舰版），XP Pro，XP Tablet PC 版	DOS，Windows，OS/2，Linux（Suse，Xubuntu），OpenSolaris（Belenix）	私有（从 2006 年 7 月起免费）
Virtual Server 2005 R2	Microsoft	Intel x86, x86-64	Intel x86	Windows 2003，XP	Windows NT，2000，2003，Linux（Red Hat and SUSE）	私有（免费）
VMware ESX Server	VMware	x86, x86-64	x86, x86-64	无：裸机安装	Windows，Red Hat，SuSE，Ubuntu，Netware，Solaris，FreeBSD 等	私有
VMware ESXi Server	VMware	x86, x86-64	x86, x86-64	无：裸机安装（嵌入式）	Windows，Red Hat，SuSE，Ubuntu，Netware，Solaris，FreeBSD 等	私有
VMware Fusion	VMware	x86, Intel VT-x	x86, x86-64	Mac OS X（Intel）	Windows，Linux，Netware，Solaris，其他	私有
VMware Server	VMware	x86, x86-64	x86, x86-64	Windows，Linux	DOS，Windows，Linux，FreeBSD，Netware，Solaris，虚拟设备	私有（免费）
VMware Workstation 6.0	VMware	x86, x86-64	x86, x86-64	Windows，Linux	DOS，Windows，Linux，FreeBSD，Netware，Solaris，Darwin，虚拟设备	私有

VMware Player 2.0	VMware	x86, x86-64	x86, x86-64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Netware, Solaris, Darwin, 虚拟设备	私有（免费）
Xen	Citrix，英国剑桥大学，Intel, AMD	x86, x86-64	x86, x86-64	NetBSD, Linux, Solaris	Linux, Solaris, Windows XP & 2003 Server（需要 3.0 版和 Vanderpool 或 Pacifica），九号项目，FreeBSD	GPL

2.1 VMware

2.1.1 VMware 简介

- VMware 的历史

早在 1998 年时，毕业于 MIT 的 Diane Greene 察觉到计算机资源的使用率过低，因此和 Dr. Mendel Rosenblum、Scott Devine、Dr. Edward Wang 以及 Edouard Bugnion 等人成立了 VMware 公司，专精于 OS in OS 的软件，期待能完全发挥硬件的性能，并为当时的专业 IT 人员提供一个测试、评估的低成本环境。

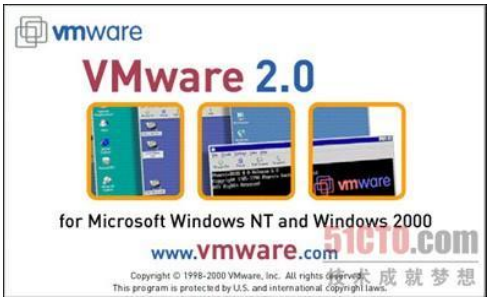
2.1.2 VMware 发展历程

1. X86 平台上的第一个虚拟机产品：VMware Workstation

当时 VMware Workstation 一上市就打响，使得 VMware 直接成为虚拟机的代名词。然而 Windows 系统的相对不稳定性，让 VMware 的工程师们把目光投向了相对较稳定的 Linux 系统，因此在 VMware Workstation 成功推出不久之后，也出现了 Linux 版本的产品，让 Linux 下的用户也可以同时运行 Windows 或其他的 Linux 套件。

2. Linux 下的虚拟机产品

虽然解决了 Windows 的问题，VMware Workstation 仍然无法摆脱寄居在固定操作系统上的恐惧。既然选择了较为稳定的 Linux，VMware 也顺势推出了运行在 RedHat、Mandrake 和 SUSE Linux 下的 VMware GSX Server，并且也拥有 Web 端的管理和客户端的管理程序。在



Linux 下，VMware 较不需要担心病毒或黑客的攻击，而操作系统本身宕机带来的危险性也相对较小，因此在 2000 年初期，Linux 成为 VMware 产品充分发挥的最好舞台。而 VMware 也在 2000 年的初期在 Linux 下推出了 VMware GSX Server 产品，这也是后来推出 VMware Server 版的基础。当然在 GSX Server 上最大的特色，就是使用了 Client/Server 架构的管理界面，更可以使用浏览器来连入 GSX Server 来管理。

3．真正的原生架构（Bare-metal）出现：VMware ESX Server

当然一劳永逸地解决寄居在其他操作系统上的问题是 VMware 工程师的梦想。VMware 也在稍后推出了自己的操作系统，称为 VMware ESX Server。ESX 以 Redhat 7.2 为基础，插入了自己的硬件核心（源自于 Dr. Mendel Rosenblum 开发的 SimOS），成为一个真正原生架构（Bare-metal）的虚拟机，而 ESX 的出现，正式宣告了 VMware 踏入企业界的领域。

Bare Metal 原来的意思就是"几乎只有纯硬件"。在 ESX 之前的 VMware 产品上，所有的虚拟机都必须安装在一个操作系统中的 Hypervisor 之上，如果将这个操作系统拿掉之后，就只剩下硬件了，那么如果有一个很小的 Hypervisor 架设在硬件之上，那么就"几乎只有纯硬件"了。

4．全新的企业平台：Virtual Infrastructure

原生架构的虚拟机能做的事竟然超乎原来的预期。而多台 ESX Server 在企业管理上也能满足更多的业务需求，因此 VMware 在 2006 年时，围绕着 ESX Server 也推出了类似 ESX 群集的 Virtual Infrastructure 架构，简称 VI3。当然在群集出现之后，客户端的管理、群集之间的管理等基本功能也加入到 VI3 中，使得 VI3 成为目前全球唯一最完整的虚拟机群集产品。后来 Virtual Infrastructure 版本从 VI4 改成 vSphere，是 VMware 当时最新的原生架构群集产品。

5．不得不提的 ESXi

就单纯的原生架构来说，底层的 Hypervisor 应该是越简单越好（参见图 2-27）。这么做的好处当然是避免和虚拟机争抢系统资源，而简单的 Hypervisor 出现兼容性或者和硬件交互时产生问题的机会也较小。

为了更简化 Hypervisor 这一层，VMware 大胆推出了最简单的 ESXi 版本的虚拟机。ESXi 最重要的特色就是将 Service Console 从 ESX Server 中拿掉，让整个 ESX Server 更加简单，在 ESXi 中，你必须使用 VI Client 来访问 ESXi，这种方式不但不会让 ESXi 更难用，反而使得整个 vSphere（VI4）的架构更精简。

ESXi 最棒的特色就是支持 USB 开机。在免去了容易出故障的硬盘之后，整个 ESXi 不仅更加有弹性，而且更加安全。你可以将 ESXi 安装在一个 1GB 的 U 盘中，并且在任何支持 ESX 硬件的服务器上开机，只要你将 IP 设定好之

后，这台机器就可以马上摇身一变成为 vSphere（VI4）中的一员。而这些 ESXi 在获得了正确的授权之后，不仅可以使用 VMotion、HA 等功能，而且能以更快的速度在 ESXi 服务器硬件宕机后马上部署一个新的 ESXi 服务器。更可喜的是，ESXi 目前已经有条件的免费开放了！

VMware 可以说是虚拟化技术的布道者，是它将虚拟化技术带到 x86 平台。该厂商目前也是虚拟化行业的龙头老大。VMware 作为全球最早涉及到虚拟化的一款软件产品，商用收费软件，元老级的身份，对虚拟化也有着自己独特的思考，易用性，可管理性比较好，对多种操作系统的支持比较好，磁盘 I/O 性能一直表现不好。另外，CPU 性能也比不上 Hyper-V

2.1.3 VMware ESX 和 VMware ESXi

VMware ESX 体系结构：在原始 ESX 体系结构中，虚拟化内核（称为 vmkernel）使用称为控制台操作系统（简称 COS 或服务控制台）的管理分区来扩充。 控制台操作系统的主要用途是提供主机的管理界面。 在控制台操作系统中部署了各种 VMware 管理代理，以及其他基础架构服务代理（例如名称服务、时间服务和日志记录等）。 在此体系结构中，许多客户都会部署来自第三方的其他代理以提供特定功能，例如硬件监控和系统管理。 而且，个别管理用户还会登录控制台操作系统运行配置和诊断命令及脚本。
VMware ESXi 体系结构：在 ESXi 体系结构中，移除了控制台操作系统，所有 VMware 代理均直接在 vmkernel 上运行。 基础架构服务通过 vmkernel 附带的模块直接提供。 其他获得授权的第三方模块（例如硬件驱动程序和硬件监控组件）也可在 vmkernel 中运行。 只有获得 VMware 数字签名的模块才能在系统上运行，因此形成了严格锁定的体系结构。 通过阻止任意代码在 ESXi 主机上运行，极大地改进了系统的安全性。

现在的 ESXi 有 3 个版本，ESXi Free 免费的版本，有很多限制和局限，对于小企业或许是个好的起步，但是对于想要构建高可用性的生产环境虚拟平台来说，就显得很不合适。ESXi 的另外 2 个版本是 Embedded 和 Installable。Embedded 版本是和硬件厂商合作，安装在 Flash 中的 ESXi 版本，通常 Flash 都是集成在主板上，或者是 USB Flash。ESXi Installable 则是安装版本的 ESXi，是最 common 的一个 ESXi 版本。

VMware vSphere 5.0 以后版本，所有底层虚拟化产品都改为 ESXi 产品

功能	ESX 4.1	ESXi 4.1	ESXi 5.0
服务控制台	目前	已移除	已移除
管理/配置 CLI	COS + vCLI	PowerCLI + vCLI	PowerCLI + vCLI（已增强）
高级故障排	COS	技术支持模式	ESXi Shell

脚本化安装	支持	支持	支持
从 SAN 启动	支持	支持	支持
SNMP	支持	支持（有限）	支持
Active Directory	集成	集成	集成
硬件监控	COS 中第三方代理	CIM 提供商	CIM 提供商
串行端口连接	支持	不支持	不支持
巨型帧	支持	支持	支持
通过自动部署实现主机的快速部署和集中管理	不支持	不支持	支持
自定义映像创建和管理	不支持	不支持	支持
安全 syslog	不支持	不支持	支持
管理界面防火墙	支持	不支持	支持

2.2 XEN

虚拟机自从成为媒体宠儿之后， VMware 俨然成为虚拟机的代名词。真正让 VMware 流行的产品应该还是 VMware Workstation，其在测试、评估等领域早已深入人心了。但说到企业级虚拟机这一领域，仍然还有许多厂商在努力，其中较有名的除了开源的 QEMU 之外，最重要的就是由剑桥大学的 lan Pratt 所开发的 Xen 虚拟机产品

Xen 在 2003 年第一次问世，当时大家对其半虚拟化印象深刻。由于使用了半虚拟化，Xen 占用的系统资源极低，最低可以只到 2%~8%，而 Xen 也宣称可以在一台实体机器上运行超过 128 个虚拟机。要达到这种高度虚拟化的水平，不但需要 Hypervisor 占用非常低的系统资源，更需要 Hypervisor 针对其上的虚拟机提供弹性化的管理协调能力，因此 Xen 的技术成分在当时的确让人耳目一新。

Xen 当时公司为 XenSource，07 年被 Citrix 收购。Citrix 早期的产品 MetaFrame 就是针对 Windows 服务器终端服务的解决方案，但 MetaFrame 并无法提供桌面 OS 的终端服务解决方案。为此，Citrix 一直在寻找更完整的解决方案。在经过多年苦思之后，2007 年 10 月 22 日，Citrix 在世人震惊之下收购了 XenSource，正式投身于虚拟机的产品。

XEN 有很大的用户群体。他的优点很多，对带宽的细节控制做的非常好，用户体验也很好，重做系统可以让用户直接通过网页进行，国外也有很多基于 XEN 开发的第三方 VPS 控制软件。但是他有一个致命的缺点，这个缺点是所有基于 Linux 为核心的，包括 Vmware、OpenVZ 都有的一个缺点，磁盘性能差。可能跟 Linux 下的驱动不完善有关吧。虚拟化技术，就是将日趋发展的 CPU 资源最大化的利用，但是磁盘性能不行，根本谈不上虚拟化。更别想获得多好的体验了。

2.3 KVM

KVM Kernel-based Virtual Machine 的简称，是一个开源的系统虚拟化模块，自 Linux 2.6.20 之后集成在 Linux 的各个主要发行版本中。它使用 Linux 自身的调度器进行管理，所以相对于 Xen，其核心源码很少。KVM 目前已成为学术界的主流 VMM 之一。

06 年之前，Linux 内核中还没有任何虚拟化实现，Xen 和 Vmware 依靠自己独有的技术分别在虚拟化不同领域如日中天，尤其值得一提的是，Xen 在开源领域几乎成为了虚拟化的事实标准。Avi Kivity 和他所在的以色列初创公司 Qumranet 提出并推动 KVM（基于 Linux 内核的全虚拟化方案），以其精简的架构，清晰的定位很快获得 Linux 社区多数开发人员的支持得以快速被合并进入主干，以 Avi Kivity 为主的工程师仅仅花了不到一年时间就让 Linux 社区接受 KVM 的设计方案并且通过了代码 Review，最终于 2006 年 10

月合并进入 2.6.20 主干，时至今日，人们依然对于 KVM 合并进入 Linux 主干的速度之快感到不可思议。

2008 年 9 月，红帽以 1.07 亿美元的价格收购 Qumranet，并且将红帽一直将 KVM 作为虚拟化战略的一部分，2009 年年底发布了红帽企业版 Linux 5.4，继续大力推行这种转型，鼓励用户使用 KVM 为其首选的虚拟化平台。2011 年，随着新版操作系统 Red Hat Enterprise Linux 6 的发布，红帽完全放弃了以开源 Xen 为虚拟化平台的思路，开始支持 KVM 作为 hypervisor。

为什么那么多的大公司对 KVM 感兴趣?其中最明显并且最重要的因素就是 KVM 是 Linux 内核的一部分。这个轻量级的虚拟化管理程序模块能直接与硬件交互，不需要修改虚拟化操作系统，因此性能更好，并且补丁包能够和 Linux 内核兼容，轻松控制虚拟化进程，同时减轻管理负担。

当然，KVM 也有成长的烦恼。KVM 究竟属于 Type 1 还是 Type 2hypervisor 呢?出现这种疑惑的原因在于 KVM 的基因——它属于操作系统的一部分，类似直接运行于硬件系统之上的裸机管理程序，不需要修改操作系统。这就符合 Type 1 hypervisor 的定义，也经常被称为硬件虚拟化引擎，更像是个安装在客户端上的操作系统，性能佳，运行稳定，减少了运行管理程序本身所需的花销。而 Type 2 hypervisor 更像是个应用，运行在基础操作系统上。如果将 KVM 看作 Type 2，那么所强调的高性能难达到，还可能存在安全风险。其实，对 Type 1 和 Type 2 的讨论不是是否采纳 KVM 的前提。无论 KVM 是“第一类”还是“第二类”，这都是语义上的概念。

此外，就是所有新生技术在发展中都会面临的难题：技术不成熟。KVM 的出现不过三四年时间，在可用资源、平台支持、管理工具、实施经验方面当然不能与出现八年之久的 Xen 相比。虽然目前 KVM 还缺少某些关键特性，例如存储的动态迁移，但是在后续的版本中都会开发出来。

性能方面，KVM 在逐渐显示其威力。不久前的 SPECvirt 虚拟化对照基准测试中，红帽企业 Linux 6.1、其内嵌的 KVM hypervisor 以及惠普 ProLiant DL980 G7 服务器三者结合，创造了新的成绩——最佳的虚拟化性能和最多的计算区块数量，并且六台虚拟机能同时运行一个应用程序。

虽然只是新生技术，但是由于其性能和实施的简易性，加上 Linux 企业市场中份额最大的红帽不遗余力的推广开发，KVM 将会持续成长壮大。

而开放虚拟化联盟(OVA)也在为 KVM 护航，这个由 IBM、红帽、英特尔等重量级厂商组成的联盟才成立不过半年，成员就迅速达到 200 以上。该联盟的宗旨致力于促进基于内核的虚拟机(KVM)等开放虚拟化技术的应用，鼓励互操作性，为企业在虚拟化方面提供更多的选择、更高的性能和更具吸引力的价格。

对于用户来说，可选择的主流虚拟化产品也越加清晰：红帽 KVM、VMware、Citrix 的 Xen、和微软的 Hyper-V。

2.4 Hyper-V

在 2006 年 VMware 将其 Server 产品免费化之后，整个虚拟机的市场立即产生了变化。原来大家心中的"OS in OS"观念立即提升到新的境界。微软是第一个发现这个庞大市场的竞争对手，也在当时立即宣称投入企业级的虚拟机市场，并在 2008 年成功推出了 Hyper-V。在 2010 年发布的 Windows Server 2008 R2 发布支持 Live/Quick Migration 的 Hyper-V R2 更新版，支持不需要重新启动的立即转移，

Hyper-V 并不是一个独立的产品，而是很适当地融入了 Windows Server 2008 系统中的"角色（Role）"。在 Windows Server 2008 中要安装 Hyper-V，必须以角色的方式将主机配置成虚拟机角色。这一点也让初次接触虚拟机的人直接怀疑 Hyper-V 原生是否真的是原生架构。既然需要一个 Windows Server 2008 当做宿主的操作系统，Hyper-V 原生怎么会是原生架构的呢？

Windows Server 2008 是第一个虚拟机

原生架构的最重要特色反而不是宿主的 OS，而是 Hypervisor 的位置。如果 Hypervisor 的位置是在 Ring 0 或是 Ring 1，只要能掌控其上的客户端 OS，并且为客户端 OS 提供监控和分配的服务，那么运行了 Hyper-V 的 Windows Server 2008 本身，是可以被视为一个客户端 OS 的。

在启动 Hyper-V 之前，Windows Server 2008 就是一个普通的 Windows Server 操作系统。整个操作系统掌握了所有的资源，所有的硬件都被操作系统直接访问。此时，计算机是没有虚拟机功能的，安装 Virtual Server/PC 或是 VMware Workstation/Server 均可以顺利运行。

2. 特殊的原生角色

然而启动了 Hyper-V 的角色之后，系统的本质即发生变化。首先在开机时，会载入 hvboot.sys 文件，这就是 Hyper-V 的 Hypervisor；载入 hvboot.sys 之后，以后加载的操作系统都被视为虚拟机，包括已经安装好的 Windows Server 2008。

此时 Windows Server 2008 会继续启动，但已经变成了一个虚拟机了。在 Hyper-V 中，将启动 Hyper-V 后的 Windows Server 2008 称为"父虚拟分区（Parent partition）"，算是一个最特殊的虚拟机。

"父虚拟分区"这个虚拟机和其他之后安装的虚拟机完全不同，其完全掌握着所有的资源，但 CPU 和网卡除外，因此这个 Windows Server 2008 也必须听从 Hypervisor 的指挥。

2.5 XEN ESXi Hyper-V 及 KVM 比较

- XEN 有简化虚拟模式，不需要设备驱动，能够保证每个虚拟用户系统相互独立，依赖于 service domains 来完成一些功能；

- Vmware ESXI 与 XEN 比较类似，包含设备驱动以及管理栈等基本要素，硬件支持依赖于 VMware 创建的驱动；
- Hyper-V 是基于 XEN 管理栈的修改；
- KVM 与 XEN 方式不同，KVM 是以 Linux 内核作为管理工具。

XEN 与 VMware ESXi，Hyper-V 以及 KVM 特点比较：

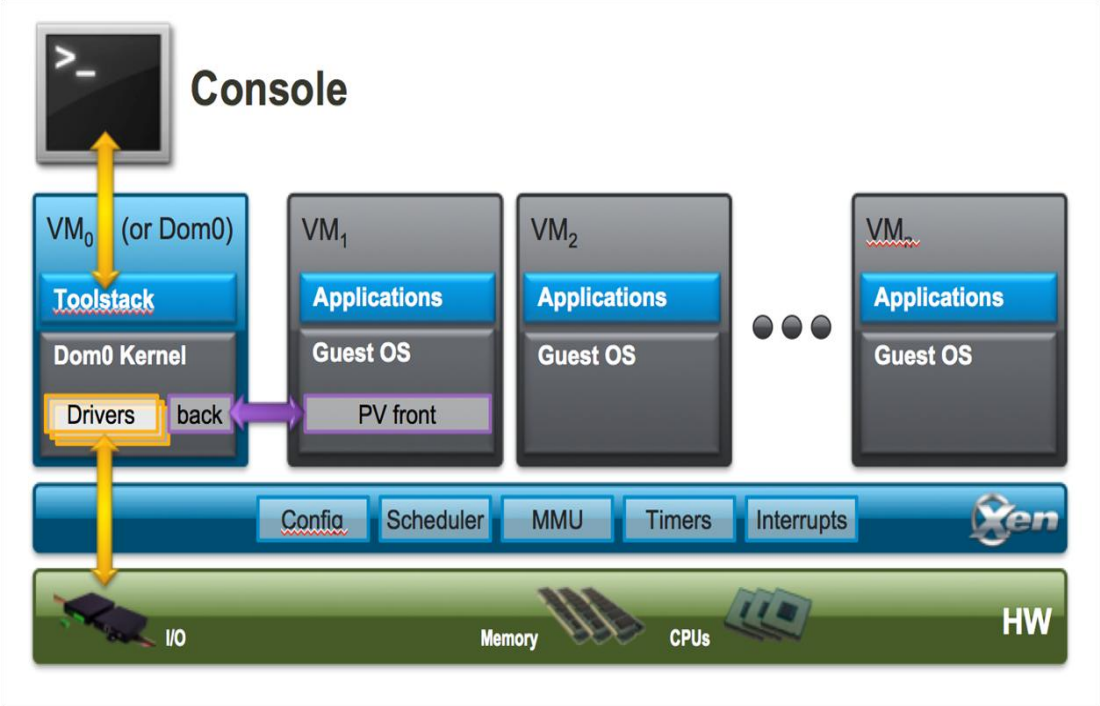
- XEN 有简化虚拟模式，不需要设备驱动，能够保证每个虚拟用户系统相互独立，依赖于 service domains 来完成一些功能；
- Vmware ESXI 与 XEN 比较类似，包含设备驱动以及管理栈等基本要素，硬件支持依赖于 VMware 创建的驱动；
- Hyper-V 是基于 XEN 管理栈的修改；

  KVM 与 XEN 方式不同，KVM 是以 Linux 内核作为管理工具得。

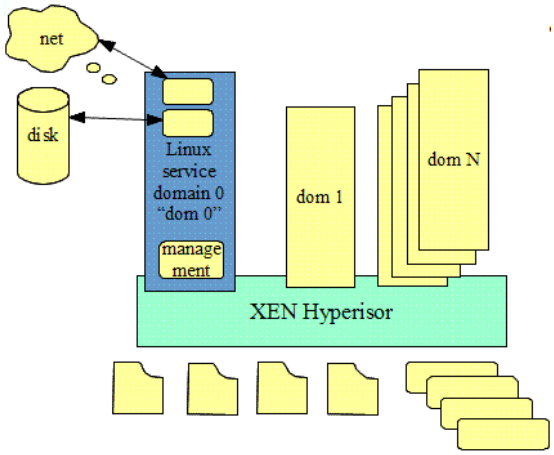
3 虚拟机的体系结构对比

3.1 XEN 体系结构

XEN 体系结构图



Xen 的 hypervisor（类似于一个微内核）直接运行在硬件之上，并先于 Domain0 启动，Domain0 作为 Xen 中的特权域，它可以用来创建新的虚拟机，并访问原生的设备驱动。上图是最早期的 Xen 的架构，可以看到每个虚拟域都有一个 frontend 的组件，这是 Xen 需要修改客户机操作系统的另一个方面，客户机要想访问设备必须通过 frontend Domain0 中的 backend 交互，由 Domain0 代为访问。

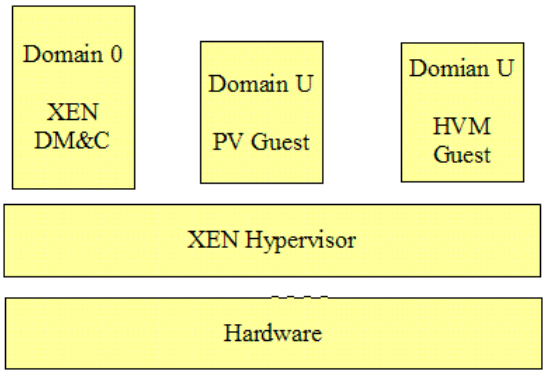


一个 XEN 虚拟机环境主要由以下几部分组成：

- XEN Hypervisor;
- Domain 0 —— Domain Management and Control(XEN DM&C);
- Domain U Guest(Dom U)
- PV Guest
- HVM Guest

下图显示除了各部分之间的关系：

Xen 三部分组成之间关系图



XEN Hypervisor ：

XEN Hypervisor 是介于操作系统和硬件之间的一个软件描述层。它负责在各个虚拟机之间进行 CPU 调度和内存分配。XEN Hypervisor 不仅抽象出虚拟机的硬件，同时还控制着各个虚拟机的执行。XEN Hypervisor 不会处理网络、存储设备、视频以及其他 I/O.

Domain 0：

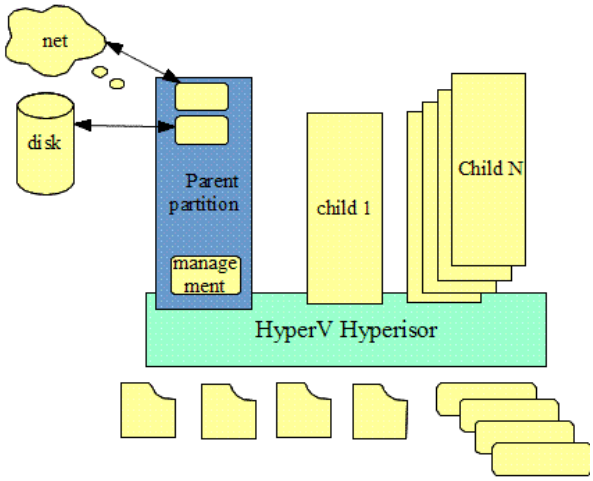
Domain 0 是一个修改过的 Linux kernel，是唯一运行在 Xen Hypervisor 之上的虚拟机，它拥有访问物理 I/O 资源的权限，同时和系统上运行的其他虚拟机进行交互。Domain 0 需要在其它 Domain 启动之前启动。

Domain U:

运行在 Xen Hypervisor 上的所有半虚拟化（paravirtualized）虚拟机被称为“Domain U PV Guests”，其上运行着被修改过内核的操作系统，如 Linux、Solaris、FreeBSD 等其它 UNIX 操作系统。所有的全虚拟化虚拟机被称为“Domain U HVM Guests”，其上运行着不用修改内核的操作系统，如 Windows 等。

3.2 Hyper-V 体系结构

Hyper-V 体系结构图



Hyper-V 是微软提出的一种系统管理程序虚拟化技术，采用微内核的架构，兼顾了安全性和性能的要求。Hyper-V 底层的 Hypervisor 运行在最高的特权级别下，微软将其称为 ring -1（而 Intel 则将其称为 root mode），而虚机的 OS 内核和驱动运行在 ring 0，应用程序运行在 ring 3 下，这种架构就不需要采用复杂的 BT（二进制特权指令翻译）技术，可以进一步提高安全性。从架构上讲 Hyper-V 只有“硬件—Hyper-V—虚拟机”三层，本身非常小巧，代码简单，且不包含任何第三方驱动，所以安全可靠、执行效率高，能充分利用硬件资源，使虚拟机系统性能更接近真实系统性能。

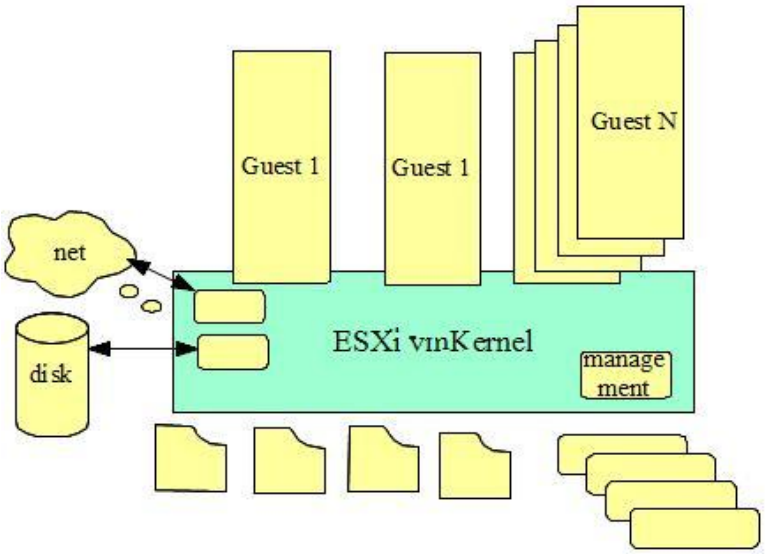
Hyper-V 支持分区层面的隔离。分区是逻辑隔离单位，受虚拟机监控程序支持，并且操作系统在其中执行。Microsoft 虚拟机监控程序必须至少有一个父 / 根分区，用于运行 64 位版本的 Windows Server 2008 操作系统。虚拟化堆栈在父分区中运行，并且可以直接访问硬件设备。随后，根分区会创建子分区用于承载来宾操作系统。根分区使用虚拟化调用应用程序编程接口 (API) 来创建子分区。

分区对物理处理器没有访问权限，也不能处理处理器中断。相反，它们具有处理器的虚拟视图，并运行于每个来宾分区专用的虚拟内存地址区域。虚拟机监控程序负责处理处理器中断，并将其重定向到相应的分区。Hyper-V 还可以通过输入输出内存管理单元 (IOMMU) 利用硬件加速来加快各个来宾虚

拟地址空间相互之间的地址转换。IOMMU 独立于 CPU 使用的内存管理硬件运行，并用于将物理内存地址重新映射到子分区使用的地址。从系统的结构图，我们可以看出来 Hyper-V 与 Xen 的架构很相似。

3.3 Vmware ESXi 体系结构

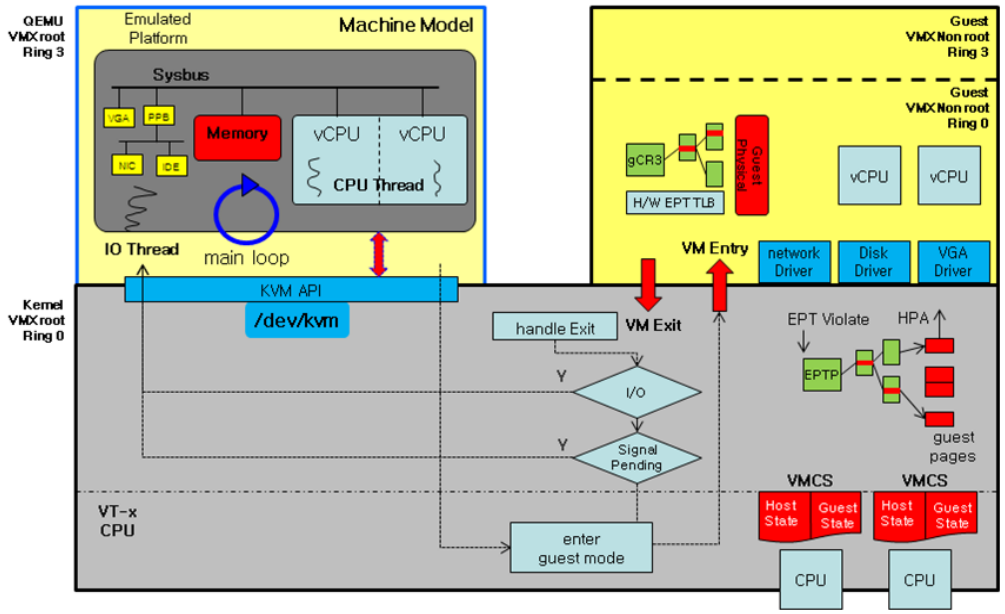
Vmware ESXI 体系结构图



由上图我们可以看出来管理工具也是直接嵌入到了 ESXi vmKernel 中，没有再分化出单独的管理工具，这一点与 Xen 是相区别的。

3.4 KVM 体系结构

KVM 体系结构图

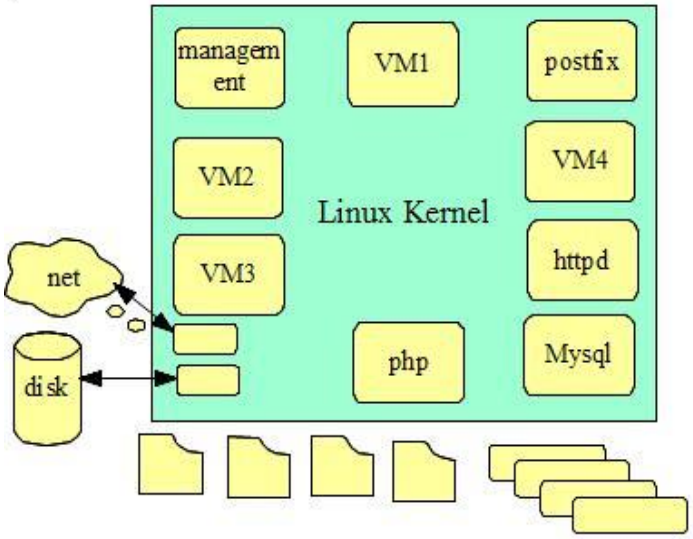


KVM 的兴起源于硬件虚拟化的出现，有了 Intel 的 VT 技术，才使得不修改操作系统同时高性能的虚拟机成为可能，而 KVM 借助了 QEMU 的设备模拟功能，并提供 CPU 和内存的虚拟化，成为了一个轻量级的全虚拟化解决方案。

VT 技术中，CPU 拥有两组 Ring，宿主机运行在 Root 模式，客户机运行在 Nonroot 模式，一旦客户机执行了敏感指令或者有中断等原因时，CPU 会通过 VM Exit 退出 Nonroot 并进入 KVM，由 KVM 检查退出原因，处理完毕后再由 VM Entry 重新进入 Nonroot 模式执行客户机的指令。

对于内存虚拟化，VT 技术后来发展了 EPT 功能。在没有 EPT 的时候，KVM 在客户机页表访问的同时也维护了一张结构一样的页表，这样速度慢并且在客户机内应用切换时开销太大。EPT 则很好的解决了这个问题，EPT 相当于是另一层的页表走查，其中记录的是客户机物理地址到机器地址的映射，每次客户机在访问内存时，MMU 除了通过 cr3 进行页表走查外，也会通过 eptp 来走查 EPT 页表最终算出正确的机器地址，这样大大加速了虚拟化情况下内存的访问效率。

设备的访问，KVM 全部交由 QEMU 来模拟。当然随着现在虚拟化技术的发展，不论是 Xen 还是 KVM 都提供了一系列的方案来提高设备访问的效率。



KVM 是一个独特的管理程序，通过将 KVM 作为一个内核模块实现，在虚拟环境下 Linux 内核集成管理程序将其作为一个可加载的模块可以简化管理和提升性能。在这种模式下，每个虚拟机都是一个常规的 Linux 进程，通过 Linux 调度程序进行调度。

通过以上四种虚拟机的体系结构图，我们可以看出他们在整个系统中的位置，以及相互之间的区别。

4 UVP 虚拟化技术

4.1 UVP 简介

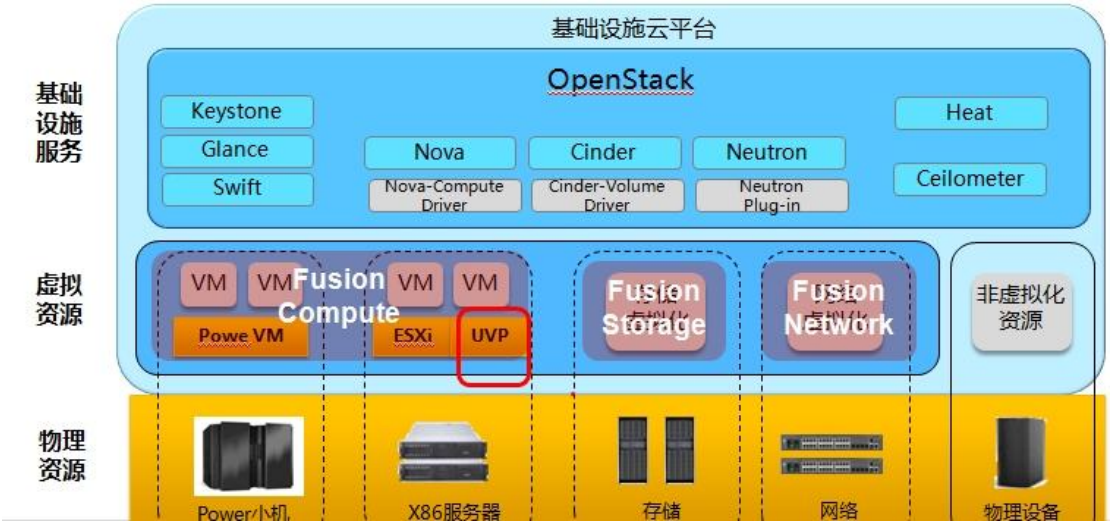
4.1.1 UVP 基本概念

UVP（Unified Virtualization Platform）是华为统一虚拟化平台，定位于构建针对电信业务环境的高竞争力的虚拟化平台。UVP 是华为基于云计算的数据中心解决方案的关键技术平台，它通过对服务器物理资源的抽象，将 CPU、内存、I/O 等服务器物理资源转化为一组统一管理、可灵活调度、动态分配的逻辑资源，并基于这些逻辑资源在单个物理服务器上构建多个同时运行、相互隔离的虚拟机执行环境。

UVP Hypervisor 采用优化裸金属架构，性能损耗 5%内

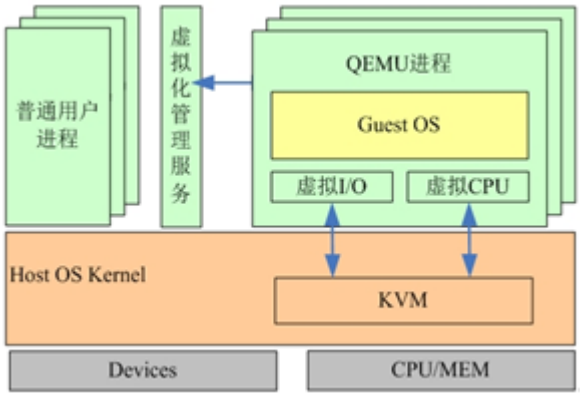
UVP Hypervisor 有两种，一种是 Xen，一种是 KVM；所以 UVP 不等同于 KVM，说法 UVP 就是 KVM 的说法是错的；另外，UVP/KVM/XEN 都是 hypervisor，与虚拟机镜像无关。
VAS Cloud 当前使用到的 UVP 或是 FusionSphere 的 hypervisor，都是 KVM 的。

UVP 在整个 FusionSphere 云平台的解决方案中的位置如下图所示：



4.1.2 区分 XEN 与 KVM

使用命令 uname -a 查看系统内核信息，系统内核版本号中带-xen 后缀的就是 xen，如果不带就是 kvm。UVP 与 KVM 关系。

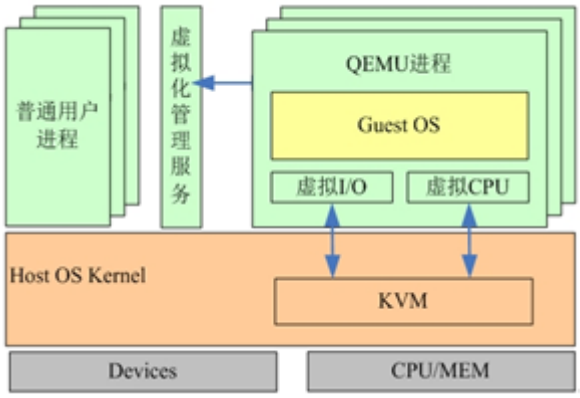


4.2 KVM 提供的功能

- 虚拟机生命周期管理（包括创建、启动、关闭、删除、重启虚拟机等）
- 虚拟机信息查询（包括虚拟机 IP、资源利用率、磁盘吞吐量等）
- 虚拟机资源调度（包括调整内存、VCPU 数目、绑定 VCPU 等）
- 虚拟机设备管理（包括磁盘挂卸载、网卡挂卸载）
- 在线迁移虚拟机（当虚拟机所在单板出现故障时可实现故障转移）
- 远程接入虚拟机

4.3 KVM 架构简图

KVM 是 Host OS 的内核中的一个模块，可以充分的利用内核资源来模拟硬件。KVM 模拟设备使用了 QEMU，QEMU 是一套由 Fabrice Bellard 所编写的以 GPL 许可证分发源码的模拟处理器。KVM 对外提供了 libvirt 接口对虚拟机进行各种操作，常见的命令行 libvirt shell(virsh)，另外也有图形界面 virt-viewer。



4.4 对 KVM 进行优化

下面是 UVP 对 KVM 进行的优化汇总

特性分类	特性名称	开源现状	UVP 特性增强描述
计算虚拟化	vNUMA(Host NUMA、GuestNUMA)	部分支持	提供虚拟机资源动态最优分配，以及运行过程中的资源动态均衡能力，实现虚拟机内存访问性能最大提升 10%
	透明大页	支持	增加主机大页配置能力
	pv-EOI	支持	增加 Euler linuxpv-EOI 支持，减少中断开销 5%以上
	APIC-v/Posted-interrupt	支持	增加硬件平台自适应判断，硬件不具备 apic-v 时自动开启 pv-EOI
	内存独占	不支持	提供虚拟机内存独占能力，实现虚拟机内存访问隔离及高性能
	虚拟机时钟支持 vdso	支持	增加 GuestOS 范围，时钟系统调用开销降为 0
	热迁移优化	不支持	优化热迁移处理，相比开源实现迁移时长降低 30%以上，业务中断时间降低 40%以上，资源开销降低 20%以上
	CPU、内存静态亲和性	不支持	提供 CPU、内存静态绑定能力，通过集群主机优选算法实现虚拟机资源静态分配，实时性提升 8 倍以上

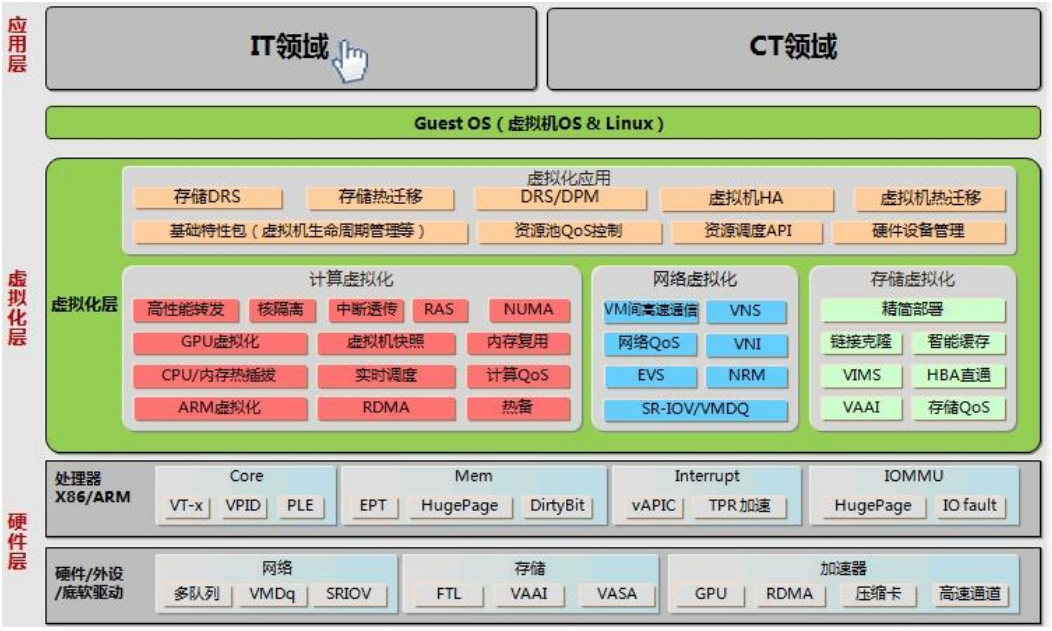
	CPU 空闲等待唤醒优化	不支持	提供单板 CPU 空闲等待优化能力，降低网络/存储 IO 时延 20%以上
	系统时钟优化	支持	增加虚拟化平台静态时钟中断配置能力，降低网络/存储 IO 时延 10%以上
	虚拟机中断亲和性	不支持	提供虚拟机中断亲和性设置能力，实现虚拟中断、设备模拟线程、虚拟 CPU 运行在统一物理 CPU，降低中断处理时延 5%
网络虚拟化	虚拟机软交换优化	不支持	VXLAN 优化 OVS 提升 20%以上
	Virtio-net	不支持	优化网络收发包处理、锁等，性能提升 10%以上。
	Netmap 软直通	不支持	屏蔽硬件驱动修改对业务影响，提供与 netmap 非虚拟化形态同等性能(40Gbps 线速，需要虚拟机业务适配 netmap API)。
存储虚拟化	Vhost-scsi	不支持	优化 IO 线程唤醒、调度，IO 时延降低 20%以上

5 FusionSphere 简介

华为 FusionSphere 主要包含了 FusionManager，FusionCompute，另外还包含了 FusionStorage 和 FusionNetwork，注意这两个模块和 FusionCompute 中的存储虚拟化和网络虚拟化不是一个概念，不要弄混了。

FusionManager 是一个云管理系统，通过统一的接口，对计算、网络和存储等虚拟资源进行集中调度和管理，提升运维效率，保证系统的安全性和可靠性，帮助运营商和企业构筑安全、绿色、节能的云数据中心。

FusionCompute 是云操作系统基础软件，主要由虚拟化基础平台和云基础服务平台组成，主要负责硬件资源虚拟化，以及对虚拟资源、业务资源、用户资源的集中管理。



可以看到，图中虚拟化应用对应了 VMware 的 vCenter 功能；计算虚拟化，存储虚拟化，网络虚拟化对应了 ESXi 的功能。

5.1 VMware 与 FusionSphere 对比分析

VMware 提供虚拟化的解决方案，主要包括桌面虚拟化、服务器虚拟化、云计算、数据中心以及应用虚拟化等，VMware 的虚拟化产品主要分为基础架构软件、管理软件、桌面软件以及应用软件等。VMware 在云计算方面，云基础架构主要由以下几部分组成：

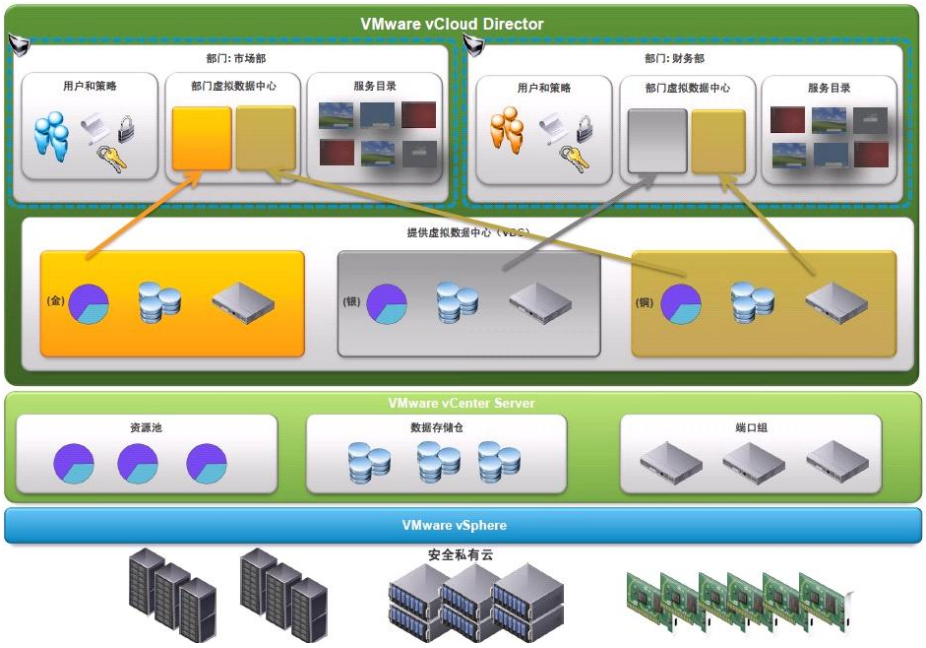
VMware vSphere，用于构建云计算基础架构的虚拟化平台，VMware ESXi 属于物理服务器的虚拟化层，是一个免费的 hypervisor，不需要操作系统的支持，它本身就是一个“操作系统”，可以用来管理硬件资源（如处理器、内存、存储器等），所有的系统都安装在它的上面，是服务器虚拟化的基础结构组件。

VMware vCenter Server，用于管理 VMware vSphere 环境，可以集中管理 ESXi 主机及其上的虚拟机，实现资源的自动调度和管理，是配置、置备和管理虚拟化环境的中央点。它提供基本的数据中心服务，如访问控制、性能监控和警报管理功能，为服务器部署增强可靠性和可管理性。

VMware vCloud Director，云管理平台，可将数据中心资源（包括计算、存储和网络）及其相关策略整合成虚拟数据中心资源池。vCloud Director 使客户能够按需交付基础架构，以便终端用户能以最大的敏捷性使用虚拟资

源。扩展模块、API 和开放式跨云标准使 vCloud Director 客户可以与现有管理系统集成，并提供在不同云环境之间迁移工作负载的灵活性。通过内置的安全性和基于角色的访问控制，可以在共享基础架构上整合数据中心和部署工作负载。通过有计划地对基础架构、用户和服务进行基于策略的池化，VMware vCloud Director 能够智能地实施策略并带来前所未有的灵活性和可移植性。通过使用 vSphere 和 VMware vCloud Director，可以构建安全、经济高效的混合云。

VMware 云基础架构如下图所示。



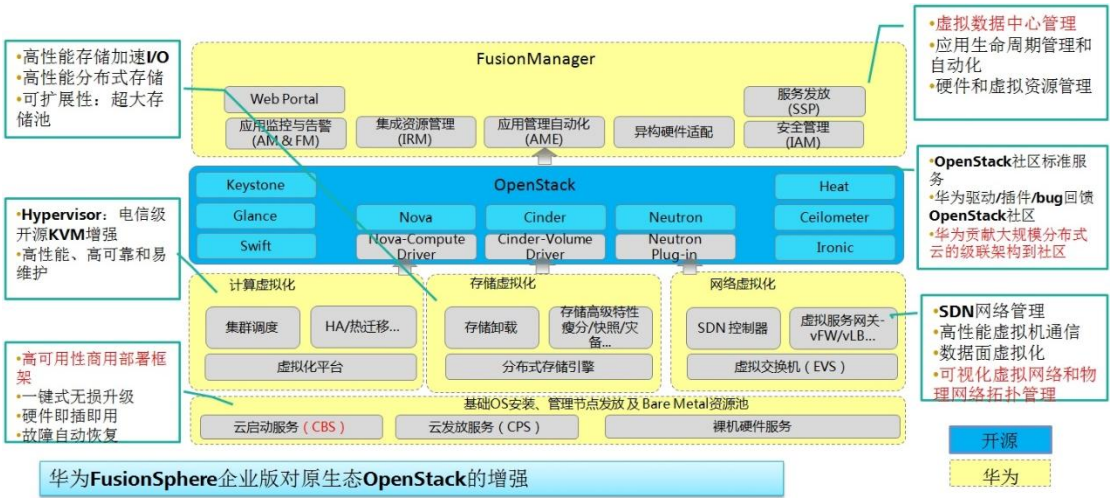
	VMware			FusionSphere		
	软件	概念	描述	软件	概念	描述
虚拟资源管理层	VMware vCenter Server	虚拟机	在物理服务器上划分出来的一台或者多台虚拟化的计算机系统。	FusionCompute VRM	虚拟机	概念一致
		模板	模板本质上是虚拟机，模板仅为虚拟机的属性。模板和虚拟机可相互转换。		模板	概念一致
		资源池	资源的集合。		资源池	概念一致
		群集	一组主机的集合；将主机添加到群集时，主机的资源将成为群集资源的一部分。		集群	概念基本一致，名字不同
		数据中心	主机和虚拟机等清单对象的主要容器；vCenter Server可包含多个数据中心。		站点	物理位置在同一个位置的一组物理服务器，可管理主机等资源。 概念基本一致，名字不同

5.2 FusionSphere 与 Openstack 关系

2012 年 10 月华为宣布正式加入 OpenStack 基金会。华为将和基金会的其他成员一起致力于推动 OpenStack 的发展与应用，共同建立一个更为开放的云计算生态系统。

OpenStack 基金会拥有的铂金会员包括 Ubuntu、惠普、IBM、SUSE 与红帽等，金牌会员有思科、戴尔、英特尔与 VMware 等。在 2013 年 OpenStack 峰会上，过 OpenStack 董事会成员投票表决，又有三家公司成为金牌会员，包括 aptira、HITACHI 与华为，其中，华为是中国首家获此殊荣的厂商。

FusionSphere 对 OpenStack 做了系列优化，主要优化内容如下图所示：



6 Xen 和 KVM 发展历史

在 Linus 明确表示 Linux Kernel 3.0 只是一个版本号的改变，而非里程碑式的飞跃后，许多人对此表达了失望，一个没有重量级功能的新版本似乎配不上这个新的版本号。不过对有些人来说，其中的一个新功能或许可以担

的上这个重任，那就是 Xen 的 block backend driver。这个功能加上之前在 2.6.37，2.6.38，2.6.39 添加的几个 Xen 相关的功能，使得即将发布的 Kernel 3.0 包含了所有成为 Xen 的 Domain0 所必须的功能，从此为 Xen 漫长的 Kernel 之路划上了一个句号，也标志着 Xen 的发展掀开了崭新的一页。

VMWare，Binary Translation 以及 Full Virtualization

提起虚拟化，一个不得不提的公司或者产品是 VMWare，如果说虚拟化最早的原型可以追溯到上世纪 70 年代的 IBM 的 VM 的话，那么当前的虚拟化热潮却是由 VMWare 引领的。相信有很多人跟我一样，对虚拟化的认知是通过 VMWare 得到的。当我初次接触 VMWare 时，我非常惊讶，居然有这样的产品实现了这么 nice 的功能。VMWare 的成功除了契合了时代的变迁所催生出的虚拟化需求外，也得益于自身产品的优秀。VMWare 产品的简单，便捷，易于理解当然是其中非常重要的一个优势，但更重要的原因来自于 VMWare 创造性的解决了 x86 平台内在不支持虚拟化的难题。

x86 平台难以虚拟化的本质主要来自于 CPU 的虚拟化。众所周知，x86 处理器的指令有 4 个特权等级，分别是 Ring 0 ~ 3。正常情况下，application 工作在最低特权级，即 Ring 3，而 kernel 工作在最高特权级，也就是 Ring 0，因为有许多硬件操作必须要在该级别下才能完成。在虚拟化的情况下，也就是有多个 OS 同时运行的情况下，显然这多个 OS 不能同时运行于 Ring 0，因为 OS 需要运行的某些 Ring 0 特权指令将互相干扰。因此一般的解决方案是将虚拟化软件（通常称作 Virtual Machine Monitor，或 hypervisor）放在 Ring 0，而将运行在虚拟机里的 guest OS 放到 Ring 1 或 Ring 3 中。一个正常的硬件设计是，当这些原本应该运行在 Ring 0 级别的指令在非 Ring 0 的级别里被执行时，处理器报错，这样运行在 Ring 0 的 hypervisor 就能捕获该错误，从而做相应的处理（比如模拟或替换该指令）实现虚拟化，然而 x86 处理器有一些特权指令在 Ring 0 及非 Ring 0 下都能执行， 并且有不同的含义，这就使得运行在 Ring 0 级别里的 hypervisor 无法捕获该指令，而该指令的运行也于原本的意义不同。

对此 VMWare 的解决方案是 Binary Translation，也就是 hypervisor 提前扫描 guest OS 待运行的指令，发现有这种无法捕获或无法虚拟化的特权指令时，将其替换成相应的一系列可捕获的指令，从而实现 guest OS 的虚拟化。当然除了 CPU 的虚拟化外，内存，I/O 设备的虚拟化也不那么容易，不过是 CPU 虚拟化方式的不同决定了各解决方案的不同。VMWare 的这种解决方式最大的优点是 guest OS 无需修改就可以运行在 VMWare 的虚拟机上，当然这个优点是相对于后来出现的 Xen 而言的，这种虚拟化方式也称作 Full Virtualization。

VMWare 的 Full Virtualization 解决方案一个致命的缺点是性能上的瓶颈，因为 hypervisor 要在运行时扫描指令，分析并替换，这个消耗是客观无法去掉的。当然 VMWare 采取了很多方法来弥补这些缺陷，使得虚拟机的运行

不至于慢的难以忍受，造成的结果就是实现相当复杂。因此在 VMWare 掀起了虚拟化的浪潮之后，许多想投入到这个领域中去的人开始想办法从其它角度来解决 x86 难以虚拟化的难题。

Xen 的出现以及 Paravirtualization

Xen 的开发人员就想出了一个巧妙的办法。Xen 最初始于剑桥大学的一个研究项目，他们的出发点是既然 x86 系统难以虚拟化，那么我就假定 guest OS 运行在一个类似 x86 的平台上，该平台由 Xen 来提供。具体点说就是 Xen 实现了一个类似微内核这样的系统，该系统运行在物理硬件平台上（Ring 0 级别），而 guest OS 运行在 Xen 上（Ring 1 级别），于 VMWare 不同的是，运行在 Xen 上的 guest OS 需要修改，因为 Xen 提供或虚拟的硬件环境已不再是 x86 平台，而是一个类 x86 平台，当然再上面的 application 不需要修改。这种虚拟化的方式称作 Paravirtualization。

这就意味着，guest 原本需要运行某些 Ring 0 级别的特权指令才能完成的任务，现在要修改为调用 Xen 提供的相应的指令（称作 hypercall，从实现方式来说类似于 system call）。显然开源的易于修改的 Linux 是最好的目标，这也是早期的 Xen 无法支持 Windows 系统的原因。然而 Xen 的架构还远不止与此，由于 Xen 运行在物理硬件与客户机操作系统之间，因此 Xen 的重要性不言而喻，稍有差错就可能导致所有的客户机崩溃，因此要求 Xen 的代码尽量简练，实际上是越少越好，而一个完整的 guest 运行所需要虚拟的硬件除了 CPU 外还有很多，例如内存，I/O 设备等等。显然如果 Xen 即支持底下的硬件设备，又要虚拟上面需要的硬件设备，那么 Xen 的实现无异于重写一个 Linux。Xen 的想法是 Xen 只支持最基本的硬件设备，也就是 CPU，内存，中断等，其它 I/O 设备由一个专门的 guest OS 来支持，而其余的 guest OS 要想访问该设备，需要通过 Xen，然后再传递到该特殊的 guest。Xen 为此将 guest 分为两个级别，一个是有访问 I/O 设备特权的特殊的 guest，称作 Domain0，其它的称作 DomainU。

因此对 Linux 来说，要想运行在 Xen 上，就必须要修改 Kernel 代码，并且根据 Linux 是运行在 Domain0 级别还是 DomainU 级别而修改的代码也不同。早期的 Xen 采用 Linux Kernel 2.6.18 作为基础，进行修改并实现了 Domain0 以及 DomainU 的功能。

由于 Xen 提供的虚拟化解决方式的性能优势，以及 Xen 的开源的特性，Xen 很快就被炒上了天。当初成立该研究项目的大学教授成立了公司，并获得了风投的青睐，继而被 Citrix 收购；各大发行版纷纷抢着支持 Xen 作为自己的虚拟化方案；很多创业公司也一夜之间冒了出来，研究 Xen 或者利用 Xen，以期望在未来的虚拟化与云计算大潮中能博得一位；Xen 俨然成了虚拟化的未来。

硬件虚拟化技术的进步以及 KVM

然而技术的发展往往非人所料，由于虚拟化浪潮的热火朝天，以及 x86 平台难以虚拟化的本质，使得 Intel 与 AMD 这两大芯片商开始思考如何在处

理器里添加功能克服原有的缺陷。于是在 2006 年，Intel VT-x 与 AMD-V 出现了。简单的说这两个 CPU 扩展就是在保持四个 Ring 特权级别的条件下，分出两个 forms，分别给客户机与 hypervisor 使用，由于每个 form 都有这四个 Ring 级别，因此客户机的特权指令的捕获就可以轻松实现了。硬件技术的出现很快就带动了软件技术的发展，很快有利用这种硬件虚拟化技术的软件出现了，没错，就是 KVM。

比起 Xen 来，KVM 的实现更加简洁而优雅，除了利用硬件的支持，KVM 还利用了现有的 Linux Kernel 的 CPU 调度等机制，因此 KVM 不需要像 Xen 那样重新实现一个复杂的 guest OS 的调度机制，这个任务交给 Linux Kernel 来完成，此时 guest OS 对 Linux Kernel 所在的 host 来说就是一个进程。此外 KVM 的实现方式也不需要修改 guest OS，因此 KVM 的出现很快引起了 Kernel 社区开发人员的兴趣，几乎是在最短的时间内，KVM 就进入了 Kernel，此时 Xen 仍然按照自己的开发模式在按部就班的进行着。

Xen 的问题

相对 KVM 来说，Xen 是一个庞大的项目，一个完整的 Xen 的搭建，需要四个组成部分，首先是最底层的 hypervisor，其次是需要修改的 Domain0 与 DomainU Kernel，最后是上层的应用管理进程。可以看出，hypervisor 与管理进程是独立的，可以轻易安装，然而 Domain0 与 DomainU 却需要修改后的 Kernel 支持。早期的 Xen 的源代码库里有一个修改后的 2.6.18，可以下载，编译然后安装。

这个办法并不是长久之计，Linux Kernel 的发展是很快的，一劳永逸的办法是尽快将 Xen 对 Kernel 的修改 merge 到上游 Kernel 中去。然而 Xen 的开发人员似乎并不热衷于 merge 对 Kernel 的修改。带来的问题是，对发行版来说，他们不得不花大量的精力来维护 Xen 相关的 Kernel patch，这个问题在 RHEL 5 上达到了登峰造极式的表现，RHEL 5 的 Kernel SRPM 里与 Xen 相关的 patch 有上百个。此外一些早期投入到 Xen 中去的小公司在快速开始后，发现他们不得不面对一些 tricky 的问题，而且很难判断是 Xen hypervisor 的问题还是 Dom0 或 DomU kernel 的问题，或者有时候在 backport 一些最新的 Kernel bug 修复到 Dom0/DomU 时困难重重。Xen 成了这些人的梦魇。

其实 Xen 的开发人员也不是没想过将 Kernel 的修改 merge 到上游去，然而如上面所说，Xen 对 Kernel 的修改是通过类似将 Linux 移植到一个新平台（x86 的一个子平台）的方式进行的，其中有大量的对 x86 平台代码的修改与复制，对此 Kernel 开发人员非常抵触。再加上 Xen 的开发人员并不热衷于此，Xen 进入 Kernel 之路面临着一个个障碍，而且一拖就是几年。

在 KVM 逐渐开始成熟起来，而 Xen 又迟迟无法进入 Kernel 之际，Red Hat 做出了一个艰难的决定，抛弃 Xen，转投 KVM，并收购了最初开发 KVM 的公司。Red Hat 的决定带来了巨大的影响，不少发行版都抛弃了 Xen，很多人开始预言 Xen 即将灭亡，关于 Xen 与 KVM 之争也随处可见，Red Hat

与 始终支持 Xen 的 Novell 就两者的优劣还吵了一架。同时随着 KVM 的成熟，Xen 进入 Kernel 的阻力更大，许多 Kernel 开发人员认为没有必要在 Kernel 里同时支持两个虚拟化框架，有 KVM 就足够了。而有关推动 Xen 修改进入 Kernel 的努力再次失败，Linus 明确表示，Xen 的代码从开发角度来说就是一个混乱，这样的代码进入 Kernel 只会给 Kernel 开发人员带来维护上的灾难，除非 Xen 的开发人员按照 Kernel 的开发规范重写 Kernel 相关的功能，否则 Xen 的修改不可能进入 Kernel。

漫漫人生路

在 Linus 明确对 Xen 的问题表态后，Xen 的开发人员开始了漫漫的 Kernel 人生路。此前在 pv_ops 出现后，Xen 的 DomU 部分已经进入了 Kernel，唯有剩下的 Dom0 部分。Xen 的开发人员开始一点一点的重写，提交，被打回，再重新修改提交，经过了两年时间的积累，Xen 对 Kernel 的修改终于在去年 2.6.37 时有了一个质的变化，2.6.37 包含了核心的 Dom0 支持，当然这还不够，Dom0 还需要一些 backend driver 来支持从 DomU 过来的设备访问请求，不过这些相对来说已不那么困难，轻舟已过万重山。

Xen 和 KVM

与此同时，在 Red Hat 加入 KVM 后，KVM 的发展也在日新月异。KVM 虽然利用了 CPU 的虚拟化功能，但对外围设备尤其是硬盘与网卡的虚拟还是通过 QEMU，这样的 Full Virtualization，效率并不高。为了提高效率，就要像 Xen 那样采用 Paravirtualization 的方式，即 guest 的 Kernel 知道自己访问的硬盘/网卡并不是真正的硬件设备，很快 Kernel 内部有了 VirtIO 的框架。

硬件的发展同样迅速，在第一代虚拟化技术催生了 KVM 这样的软件后，第二代的虚拟化技术致力于在性能上的提高，比如 Intel 的 EPT 以及 VT-d，AMD 的 RVI 以及 IO-MMU。在这些技术被 KVM 采用后（当然也被 Xen 采用），Xen 是否还具有天然的性能上的优势就真不好说了。相反，由于 Xen 的 Dom0 支持迟迟无法进入 Kernel，使得很多人在选择虚拟化技术时不得不三思。天平已然倾向了 KVM。

最后

到底 Xen 与 KVM 孰优孰劣，尤其是性能，不是一个轻易就可以下结论的问题。性能上的比试除了产品的架构外，更多依赖于任务本身是 CPU bound 还是 I/O bound，以及在测试过程中对搭建平台的一步步调整。不管怎么说，Xen 依然有存在的价值，Xen 也有大量的用户群。Xen 的 Kernel 部分正式进入 Linux Kernel 是一件值得高兴的事情，相信很多发行版将重新开始支持 Xen，至少在虚拟化技术前，我们又有了一个方便的选择。对于 Xen 来说，这也意味着它与 KVM 的竞争又站到了同一起跑线上。

纵观 Xen 这短短几年的发展，既有巅峰时的人人追捧，也有没落时的失意。除了虚拟化大环境技术上的变迁外，更主要的原因在于 Xen 的开发策略没有坚持通常所说的上游优先（upstream first），也就是在代码还没有进入

上游的 Kernel 之前，就发布出去，从而为日后的弯路打下了基础。这样的教训令人足戒。

Xen 的 Dom0 虽然进入了 Kernel，但 Xen 的故事并未结束，Xen 仍然有一些代码需要进入 Kernel，Xen 本身对硬件虚拟化技术的利用也有待提高，不管怎么说，Xen 又重新回到了人们的视野，至于 Xen 是否还能回到巅峰，只能拭目以待了。

7 Xen 和 KVM 的简单对比

性能方面在全虚拟化已经非常成熟的现在，Xen 和 KVM 几乎没有差别。

功能上 Xen 提供了大量的功能，可以满足各种各样的需求。

成熟度上，Xen 出现的更早，也跟成熟。

易用性上 KVM 作为一个 Linux 模块，十分容易部署使用。

社区的认同度，KVM 比 Xen 要高出很多，KVM 中大量复用了 Linux 的代码，而 Xen 都是自己的一套，KVM 更是基层到 Linux 的 mainline 中随着 Linux 一起更新。

趋势上，许多企业都逐渐放弃了 Xen 转投 KVM 的怀抱。

KVM and Xen 的区别

2、易用性上 KVM 作为一个 Linux 模块，十分容易部署使用

3、社区的认同度，KVM 比 Xen 要高出很多，KVM 中大量复用了 Linux 的代码

4、趋势上，许多企业都逐渐放弃了 Xen 转投 KVM 的怀抱；目前 NFV 项目，大家都选用 KVM

5、xen 的功能完备性要优于 kvm