华中科技大学出版社计算机丛书

Xen虚拟化技术

Xen Virtualization Technology

石 磊 邹德清 金 海 著

华中科技大学出版社
http://www.hustp.com

华中科技大学出版社计算机丛书

Xen虚拟化技术

Xen Virtualization Technology

石 磊 邹德清 金 海 著

华中科技大学出版社
（中国·武汉）

图书在版编目(CIP)数据

Xen虚拟化技术/石 磊 邹德清 金 海 著. —武汉:华中科技大学出版社,2009年5月
ISBN 978-7-5609-5203-1

Ⅰ.X… Ⅱ.石… Ⅲ.虚拟技术 Ⅳ.TP391.9

中国版本图书馆CIP数据核字(2009)第037988号

Xen虚拟化技术 石 磊 邹德清 金 海 著

策划编辑:沈旭日 余 奕
责任编辑:沈旭日 余 奕 封面设计:秦 茹
责任校对:周 娟 责任监印:周治超

出版发行:华中科技大学出版社(中国·武汉)
武昌喻家山 邮编:430074 电话:(027)87557437

录 排:武汉众心图文激光照排中心
印 刷:湖北新华印务有限公司

开本:787 mm×1 092 mm 1/16 印张:26.75 插页:2 字数:568 000
版次:2009年5月第1版 印次:2009年5月第1次印刷 定价:68.00元
ISBN 978-7-5609-5203-1/TP·679

前言 Preface

当前，计算系统的资源规模不断扩展，处理能力快速增强，资源种类日益丰富，应用需求也灵活多样。虚拟化技术能够动态组织多种计算资源，实现透明化的可伸缩计算系统架构，从而可以灵活构建满足多种应用需求的计算环境，提高计算资源的使用效率。

计算机世界里，“虚拟化”无处不在。从最早的内存虚拟化到存储虚拟化，以及近年来大行其道的资源虚拟化和网格计算。特别是多核/众核技术和系统级虚拟化技术的出现，使得具体的硬件体系结构和软件系统之间的紧密依赖关系得以有效隔离。在各种不同的虚拟化解决方案中，虚拟化软件 Xen 无疑是佼佼者。Xen 是剑桥大学教授 Ian Pratt 等开发的一个开源的虚拟机项目，其性能接近单机操作系统的性能。由于其具有优越的性能和开源性，所以被业界广泛看好，被认为是未来最有前途的虚拟化解决方案之一。

本书从 Xen 的体系结构出发，结合 Xen 3.1.0 源码，对 Xen 的半虚拟化技术进行了深入细致的分析和研究。根据 Xen 的技术特点，着重介绍 Xen 提供的用以控制和管理虚拟机的内核接口，以及相关的原理和操作应用，并详细讨论了 Xen 的基本机制和策略，核心子系统及安全应用模块。对 Xen 相关的核心技术和功能进行了较为细致和详尽的分析和说明，旨在使读者能够在尽量短的时间内对 Xen 的内部工作原理和机制获得较为全面的理解，为进一步研究 Xen 打下较为坚实的基础，希望能为国内虚拟化技术的研发做些铺路的工作。同时，作为第一本以“Xen 源码分析”为主旨的书，能够为未来同类型的著作起到抛砖引玉的作用。

本书和通常的源码分析的书有所不同，在清晰介绍原理的基础上，辅以源码分析。由于 Xen 本身是一个复杂系统，因此，本书仅摘取其中的关键代码进行分析，并给出注释，力图达到纲举目张的效果。同时，单独开辟了一章专门介绍全虚拟化技术的基本原理，并阐述了必要的数据结构和接口函数。在章节安排上，本书先总体上介绍虚拟化技术和 Xen 体系架构，接着介绍 Xen 的基本机制和策略，以及三个关键子系统，然后针对 Xen 的内部安全模块进行分析，最后针对 Xen 全虚拟化技术进行介绍。

阅读本书

第 1 部分　Xen 和虚拟化技术

该部分由两章组成，第 1 章对虚拟化技术的方方面面进行了综述，主要包括虚拟化技术的发展历史、各种虚拟化技术的分类，以及当前流行的虚拟化软件等；第 2 章针对 Xen 体系

结构进行了整体性介绍，也为后面章节对 Xen 各部分的详细剖析打下了基础。

第 2 部分　Xen 基本机制和策略

该部分主要讨论了与虚拟机控制和管理相关的基本机制和策略，包括：与虚拟机启动和管理相关的共享页机制；与虚拟机特权级控制和通信相关的超级调用 Hypercalls 和事件通道Event Channels；与虚拟机数据共享和传输相关的授权表 Grant Tables。这些内容分别对应本书的第 3、4、5 章。

第 3 部分　Xen 子系统

根据 Xen 体系结构的设计框架，主要讨论了 Xen 的三大核心子系统：CPU 虚拟化；内存虚拟化及 I/O 设备虚拟化；Xen 是如何为上层操作系统 Guest OS 提供硬件抽象的。这些内容分别对应了本书的第 6、7、8 章。

第 4 部分　Xen 安全机制

该部分主要讲述了 Xen 提供的与安全控制相关的模块，包括虚拟可信平台模块 VTPM 及访问控制模块 ACM。这部分内容分别对应本书的第 9、10 章。

第 5 部分　Xen 展望

该部分对 Xen 全虚拟化技术的原理进行了介绍，通过硬件虚拟化技术实现了对 Xen 全虚拟化的支持，并以 Intel VT 技术为重点讲述了硬件虚拟化的知识。这部分内容对应本书的第 11 章。

致谢

本书的写作、出版是在国家重点基础研究发展计划(973 计划)项目“计算系统虚拟化基础理论与方法研究”的大力支持下完成的。本书的撰写得到了华中科技大学服务计算技术与系统教育部重点实验室暨集群与网格计算湖北省重点实验室的系统安全研究组的多位博士生和硕士生的大力协助，该研究组同时参与了上述 973 计划的子课题“虚拟计算系统安全可信机制研究”。特别感谢项国富、陈刚、代炜琦、王圣兰、程戈、龙锦就、郑伟德、蒋雅利和李敏等为本书提供了很多有价值的材料；感谢赵峰博士为本书提供了不少中肯的修改意见；感谢计算机学院信息安全团队部分成员的支持，包括杜尚鑫、金闪、刘磊、黄楚等，他们参与了前期的源码分析工作。本书能得以完成离不开他们卓有成效的工作和辛勤的劳动，在此一并表示感谢！

我们在撰写本书的过程中力图做得更好，但百密难免一疏，欢迎大家批评指正，并请予以谅解。

作　者

2009 年 3 月

目录 Content

第 1 部分　Xen 和虚拟化技术

第 2 部分　Xen 基本机制和策略

第3部分 Xen子系统

第 5 部分 Xen 展望

第1部分

Xen和虚拟化技术

从虚拟化技术的发展历史出发，对虚拟化技术涉及的方方面面进行综述；并针对Xen体系结构进行整体性介绍，为深入剖析Xen虚拟化技术打下基础。

第1章 概述

在计算机世界里，虚拟化(Virtualization)无处不在。当我们通过互联网浏览网页时，就很有可能已经在和“虚拟化”打交道了，因为同我们进行交互的可能就是一个位于远端服务器上的虚拟机，如Java虚拟机(JVM)。虚拟化在操作系统的设计与实现中也有所体现：32位操作系统的虚拟内存技术，即采用内存页面与外存之间换入换出的办法，为每个应用进程虚拟出4GB内存空间。此外，操作系统还可以使应用程序共享同一个硬件。早在20世纪60年代，IBM对大型机使用的虚拟机(Virtual Machine，VM)可以允许多个用户和应用程序共享同一台机器，相互之间不会产生任何干扰。虚拟化的不同应用，产生了不同的虚拟化技术，如服务器虚拟化、网络虚拟化、微处理器虚拟化、文件虚拟化、存储虚拟化等技术。

1.1 虚拟化技术

虚拟化技术已有40多年的历史，它起源于对分时(Time Sharing)系统的需求。在1959年6月的国际信息处理大会(International Conference on Information Processing)上，科学家Christopher Strachey发表的一篇名为《大型高速计算机中的时间共享》(Time Sharing in Large Fast Computers)的学术报告，被认为是对虚拟化技术的最早论述。

1.1.1 虚拟化技术的发展历史

早期的操作系统只能处理单个任务，为了能够同时处理多个任务，分时系统的概念被提了出来，而分时系统的最早解决方案就是虚拟化技术。IBM最早发明了一种操作系统虚拟化技术，允许用户在一台主机上运行多个操作系统，让用户尽可能充分地利用昂贵的大型机资源。最早使用这项技术的是IBM 7044计算机，其基于麻省理工学院(MIT)开发的分时系统CTSS(Compatible Time Sharing System)和曼彻斯特大学的Atlas项目(世界最早的超级计算机之一)而开发，首次使用了请求调页和系统管理程序调用。

在随后的10年间，IBM以及其他几家公司陆续开发了一批含有虚拟化技术的产品，比如型号为Model 67的System/360主机，可以通过虚拟机监视器(VMM)虚拟所有硬件接口。“M44/44X”计算机项目定义了虚拟内存管理机制，对于用户来说，这些虚拟内存就好像

一个个“虚拟机”，为多个用户的程序提供了独立的计算环境。之后，IBM 更是认识到虚拟化技术的重要性，IBM360/40，IBM360/67，以及 VM/370 虚拟计算系统相继问世。这些机器在当时都具有虚拟机功能，它们通过 VMM 在物理硬件之上生成了很多可以运行独立操作系统软件的虚拟机。用户能够在任何一个配置这些虚拟系统的环境中运行当时流行的任何一种操作系统，进而使昂贵的大型机资源得到充分利用。

计算机技术的发展和市场竞争的需要，使得大型机上的虚拟化技术开始向小型机或 Unix 服务器上移植，使得 RISC(Reduced Instruction Set Computer)服务器与小型机成为虚拟化技术的下一个受益者。1999 年，IBM 公司在 AS/400 上提出了“逻辑分区(Logical Partition，LPAR)”技术和新的高可用性集群解决方案。在 POWER 管理程序上运行的 AS/400 LPAR使单台服务器工作起来如同 12 个独立的服务器一同工作一样。而在 2002 年，IBM 更进一步，其 AIX5L v5.2 还首次包括了动态逻辑分区(Dynamic Logical Partition，DLPAR)。DLPAR 允许在无需重启系统的情况下，将包括处理器、内存和其他组件在内的系统资源分配给独立的分区。这种在不中断运行的情况下进行资源分配的能力不仅令系统管理变得更加轻松，而且因为能够更好地使用资源而帮助降低总成本。

尽管 HP、SUN 也和 IBM 一样在自己的 RISC 服务器上提供了虚拟化技术，但真正使用大型机和小型机的用户还是少数，加上各家产品和技术之间并不兼容，使得虚拟化技术仍旧不太被公众所关注。

随着 x86 处理器性能的提升和应用的普及，虚拟化技术的发展已经惠及 x86 架构。在以前的虚拟化历程中，x86 架构并没有成为虚拟化技术发展的受益者，主要原因是 x86 架构在设计之初并没有考虑支持虚拟化技术，它本身的结构和复杂性使得在其之上进行虚拟化非常困难。然而，20 世纪 90 年代末期，VMware 和其他虚拟化软件厂商率先为虚拟化技术在 x86 服务器环境下的应用开辟了道路，使得虚拟化应用的前景更加广阔。他们实施的是一种软件解决方案，以 VMM 为中心，使 PC 服务器平台实现虚拟化。然而，在这种纯软件的“全虚拟化”模式中，每个客户操作系统(Guest OS)获得的关键平台资源都要由 VMM 控制和分配，以避免发生冲突。为此，须要利用二进制转换，而二进制转换的开销又使得“完全虚拟化”的性能大打折扣。为解决这个问题，引出了一种新的虚拟化模式，即在 Denali 项目和 Xen 项目中出现的半虚拟化技术。半虚拟化技术的实施不需要二进制转换，而是通过对 Guest OS 进行代码级修改，从而为操作系统提供了新的接口，以使新的、定制的 Guest OS 获得额外的性能和高扩展性。但是，修改 Guest OS 非常繁琐，带来了一些系统指令级别的冲突以及运行效率的问题，需要软件开发商、集成商等投入大量的时间和人力对系统进行优化。此时，虚拟化技术的发展已走到了硬件支持阶段，使半虚拟化的障碍得以排除。硬件虚拟化技术就是把纯软件虚拟化技术的各项功能用硬件电路来逐一实现。作为发挥多核处理器性能的一个有效手段，Intel 和 AMD 都在硬件级提供了对虚拟化的支持，比如 Intel 的 VT 技术和 AMD 的 SVM(Secure Virtual Machine)技术。Intel 的 VT(Virtualization Technology)技术提供了一个名为 VMX Root 的新特权级，专门用于运行 VMM，并对标准

x86架构进行了优化。在满足了对CPU半虚拟化和二进制转换技术的需求后，硬件已经能支持多种未经修改的Guest OS直接运行，使VMM的设计得到极大简化，进而使VMM能够按通用标准进行编写，减少了VMM运行的系统开销。

不过，与已经有多年历史的Unix服务器、大型主机上的虚拟化技术相比，x86服务器上的虚拟化仍旧处于早期阶段——根据Intel的蓝图，在处理器中集成硬件辅助虚拟化指令只是IA平台上的第一步，第二步则要实现I/O方面的虚拟化，直到最后实现整个IA平台的虚拟化。也就是说，目前的x86平台仅仅能够实现处理器级别的虚拟化，在I/O以及其他方面的虚拟化还需要进一步的研究。不仅如此，x86架构上的虚拟化技术还无法完美地实现虚拟分区之间的动态迁移，而这些在Unix平台、大型主机上早已不是问题。IBM公司的POWER6处理器甚至还提供了Live Partition Mobility功能，允许活动分区的在线迁移。目前，x86架构上的虚拟化技术的最高规划是支持8路SMP系统，实现对单个CPU资源的配置。

x86平台上虚拟化技术的逐步实现，首次向人们展示了虚拟化应用的广阔前景（因为x86平台可以提供便宜的、高性能和高可靠性的服务器）。更重要的是，一些用户已经开始配置虚拟化的生产环境，需要得到新的管理工具，以便随着虚拟化技术的发展而得到更大的效益。

1.1.2 虚拟化技术的实现层次及分类

为了满足不同的功能需求，目前已出现了许多不同种类的虚拟化解决方案。由于采用了不同的实现方式和抽象层次，因而这些虚拟化系统呈现出不同的特性。计算机系统的设计本身采用分层结构，如图1-1所示。首先，计算机硬件为直接运行在其上的软件（操作系统）提供的接口是一组指令集合（Instruction Set Architecture，ISA），不同处理器硬件提供的接口不尽相同。例如，Intel奔腾系列提供的接口是著名的x86指令集，它基于CISC（Complex Instruction Set Computer）；而IBM公司生产的Power系列处理器则基于RISC。理论上，如果操作系统希望在一个特定的硬件上运行，就必须遵守和使用该硬件提供的指令集。

为方便上层应用程序的开发，一个完善的操作系统往往会提供一组程序开发库，这些库为应用程序的编写提供了大量的应用程序编程接口（API），使得在编写应用程序时不用直接调用操作系统提供的更底层的功能，从而减轻了程序员的工作负担。

理论上，虚拟化技术采用的抽象层次可以在图1-1所示的几个层次中自由选取。而实际上，选择的多样性，就决定了虚拟化技术的多样性。然而，虚拟化技术的实质是一样的：将底层资源进行分区，并向上层提供特定的和多样化的执行环境。

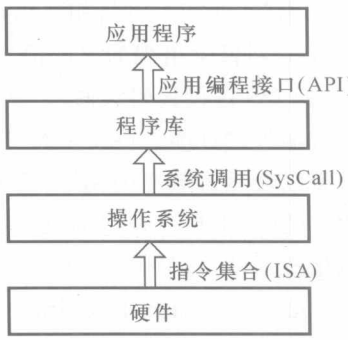

图1-1 计算机层次结构

下面结合当前主流的虚拟化解决方案和计算机系统结构，站在虚拟机实现所采用的抽象层次的角度对虚拟化系统进行分类。

1. 指令级虚拟化

指令级虚拟化又称为指令集架构级虚拟化(ISA 虚拟化)。它通过纯软件方法，模拟出与实际运行的应用程序(或操作系统)不同的指令集去执行，采用这种方法构造的虚拟机一般称为模拟器(Emulator)。一个典型的计算机系统由处理器、内存、总线、硬盘驱动器、磁盘控制器、定时器、多种 I/O 设备等组成。模拟器可将客户虚拟机发出的所有指令翻译成本地指令集，然后在真实的硬件上执行。这些指令包括典型的处理器指令(例如，x86 中的 add、sub 和 jmp 等)和特殊的 I/O 指令(例如，in/out 指令)。当然，一个成功的模拟器必须能够模拟真实机器所能做的一切事情，包括读 ROM 芯片、重启、系统开关等。指令级虚拟化系统的代表包括 Bochs 和 QEMU 等。

(1) Bochs

Bochs 是由 Kevil Lawon 领导的一群爱好者用 C++编写的开放源代码的 x86 PC 模拟器。它可以运行在大多数最流行的平台上，包括 x86、PowerPC、Alpha、Sun 和 MIPS，并可编译、模拟大多数版本的 x86 机器。Bochs 翻译每一条指令，从加电到重新启动模拟 Intel x86 CPU，定制 BIOS 并为所有标准 PC 外围设备装配设备模型，还支持无修改的软件执行(包括操作系统)。但是，Bochs 模拟 x86 软件需要额外的开销。

(2) QEMU

QEMU 是一个使用便携式动态翻译器的快速处理器模拟器。它支持两种操作模式：单纯用户空间模式和完整系统仿真模式。在前一个模式中，QEMU 可以将在 CPU 上编译的 Linux 进程装载到另一个 CPU 中，或者跨编译进行交叉调试。在后一个模式中，QEMU 可以模拟完整系统，包括处理器和外围设备。QEMU 采用基本块作为独立的翻译单元，在动态翻译过程中，将遇到的每一块代码都转换成主机指令集。与 Bochs 不同的是，它支持多种处理器架构的模拟，包括 x86、ARM、PowerPC 和 Sparc，而 Bochs 只支持 x86 的客户执行环境。另一方面，它像 Crusoe 一样，使用动态翻译以达到比较高的代码运行速度。

2. 硬件级虚拟化

硬件抽象层(Hardware Abstraction Layer，HAL)虚拟化实际上与指令集架构级虚拟化非常相似，其不同之处在于，这种类型的虚拟化所考虑的是一种特殊情况：客户执行环境和主机具有相同指令集的情况，并充分利用这一特点，让绝大多数客户指令在主机上直接执行，从而大大提高了执行的速度。如今大部分商业虚拟化软件均在流行的 x86 平台使用此虚拟化技术来提高效率，就说明了这类虚拟化技术的可行性与实用性。该虚拟化技术可以

将虚拟资源映射到物理资源并在虚拟机计算中使用本地硬件。当虚拟机需要访问关键物理资源时，模拟器接管其物理资源并妥善地进行多路复用。

这种虚拟化技术要能够正确工作，所构造的虚拟机就必须对其中的一些特权指令（例如，修改页表等操作）进行处理，执行时产生 Trap（陷入）并将它传递给下层 VMM 执行。这是因为在虚拟机中运行的未加修改的操作系统会利用特权指令得到 CPU 和内存资源。当某特权指令执行时产生一个 Trap，便马上将指令发送给 VMM，使得 VMM 可以完全控制虚拟机并保持每个虚拟机间的隔离。然后，该 VMM 在处理器中执行该指令，并将模拟结果及特权指令返回给虚拟机。大多数商业虚拟机软件，都使用像代码扫描和动态指令重写这样的技术来解决这些问题。

硬件级虚拟化是目前研究最广泛的虚拟化技术，相应的虚拟化系统也较多。其中，业界最具影响力的 VMware 和 Xen 都属于硬件级虚拟化的范畴。

（1）VMware

VMware 是 EMC 公司旗下独立的软件公司。1998 年 1 月，Stanford 大学的 Mendel Rosenblum 教授带领他的学生 Edouard Bugnion 和 Scott Devine，基于对虚拟机技术多年的研究成果创立了 VMware 公司，主要研究在工业领域应用的大型主机级的虚拟化技术，并于 1999 年发布了它的第一款产品：基于主机模型的桌面虚拟化软件 VMware Workstation。之后于 2001 年推出了面向企业服务器市场的 VMware GSX Server 和 VMware ESX Server。

其中，基于主机模型的桌面虚拟化软件 VMware Workstation 可以在 Windows、Linux 和 MAC OS 上运行。VMware Workstation 软件由 3 个部分组成：VMX 驱动、安装在最高特权级 0 环的 VMM 以及在 3 环的 VMware 应用程序（VM App）。VMX 驱动器安装在操作系统内部，以使 VMM 获得所需要的特权级别。当软件执行时，VM App 将 VMM 载入带有 VMX 驱动的内核内存，并授予 VMM 最高特权级。此时，主机操作系统仅知道 VM App 及 VMX 驱动的存在，并不知道 VMM，更不知道 VMM 已经被加载到内核中。因此，整个硬件环境中存在主机操作系统和 VMM 两个执行环境：主机世界（Host World）和 VMM 世界（VMM World）。VMM 可以直接访问物理处理器，亦可通过 VMX 驱动与主机操作系统通信。该系统的体系结构如图 1-2 所示。

启动 VMware Workstation 软件后，物理处理器或者运行在 VMM 世界、或者运行在主机世界，而 VMX 驱动则控制两个世界的转换。由于只有主机操作系统拥有设备驱动，因而只有在主机世界才能直接访问硬件。当 Guest OS 进行 I/O 操作时，VMM 将截取这个操作，并且通过 VMX 驱动切换到主机世界。在主机世界中，VM App 把 Guest OS 的 I/O 指令转换成更高级的系统调用，交由主机操作系统实现 I/O 操作，然后将结果返回给 VMM 世界。由于两个世界之间的切换需要保存和重建当前所有的硬件状态，因此对性能会造成很大影响。

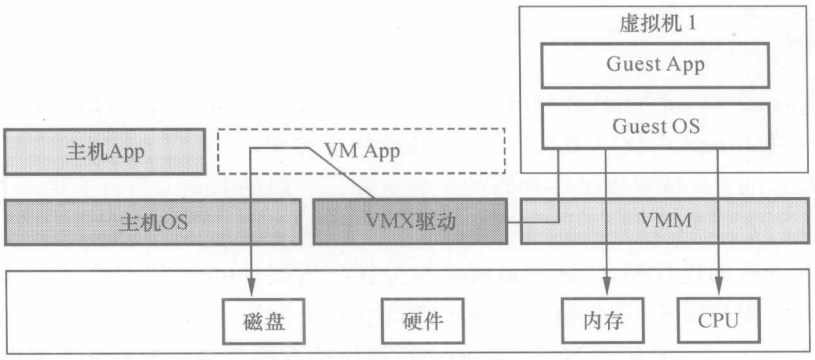

图 1-2　WMware Workstation 软件的系统结构

VMware ESX Server 采用基于独立监视器模型的虚拟化技术，即无需主机操作系统就可直接运行在物理机器上。因此，VMM 必须拥有硬件的设备驱动，满足处理 Guest OS 对系统资源的所有访问。VMware ESX Server 的性能要远高于 VMware Workstation 的。

尽管新的虚拟化解决方案不断涌现，但是 VMware 在目前的虚拟机市场上仍是最强者。VMware 首先提出并采用的气球驱动程序（Balloon Driver），影子页表（Shadow Page Table），虚拟设备驱动程序（Virtual Driver）等均已被后来的虚拟化软件（如 Xen）采用。

（2）Virtual PC

Virtual PC 采用全虚拟化技术实现，可以在 Windows 98 及以上的 Windows 操作系统和基于 PowerPC 的 Mac 操作系统上模拟 x86 机器，并在其中安装如 MS-DOS、Windows98/2000/NT/XP/Vista 等操作系统。Virtual PC 最初由 Connectix 公司开发，该公司被微软公司收购后更名为 Microsoft Virtual PC。

Virtual PC 与 VMware 类似，同样采用全虚拟化技术，为其虚拟机模拟包括主板、内存、硬盘、DVD/CD-ROM、软驱、声卡、串口、并口和 USB 口在内的硬件，如图 1-3 所示。不同的是，Virtual PC 还可模拟显卡，而 VMware 并没有模拟出显卡。VMware 为每一种 Guest OS 提供一个称为 vmware-tools 的软件包，来增强 Guest OS 的显示和鼠标功能，否则只使用 VGA，而 Virtual PC 模拟了一个比较通用的显卡：S3 Trio 32/64(4MB)。此外，Virtual PC 的网络共享方式与 VMware 的亦不同。VMware 通过虚拟的网卡实现网络共享，而 Virtual PC 通过在真实网卡上绑定 Virtual PC emulated switch 服务以实现网络共享。同时 Virtual PC 也不支持 SCSI 设备。一旦配置完毕，Virtual PC 将无法改变虚拟机所拥有的硬件设备。

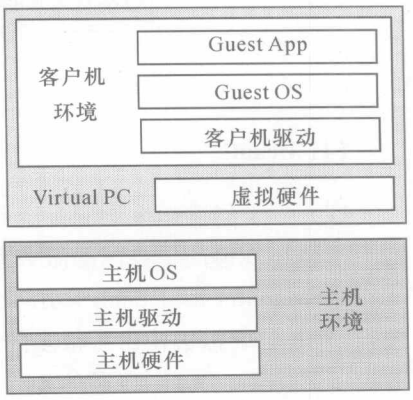

图 1-3　Virtual PC 体系结构