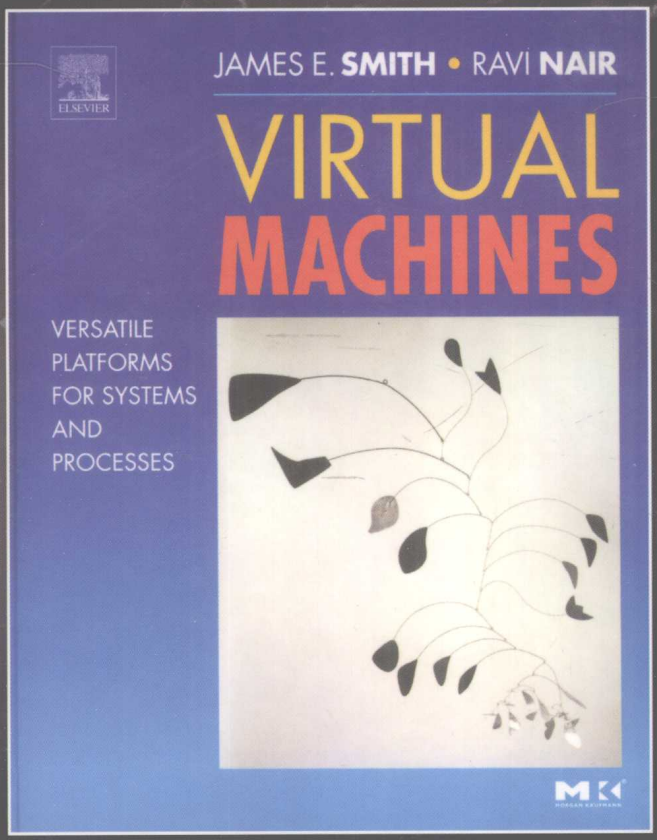
HZ BOOKS 华章教育
ELSEVIER

计 算 机 科 学 丛 书

虚拟机

系统与进程的通用平台

(美) James E. Smith 威斯康星大学麦迪逊分校
Ravi Nair IBM Thomas J.Watson研究中心 著
安虹 张昱 吴俊敏 中国科学技术大学 译



Virtual Machines

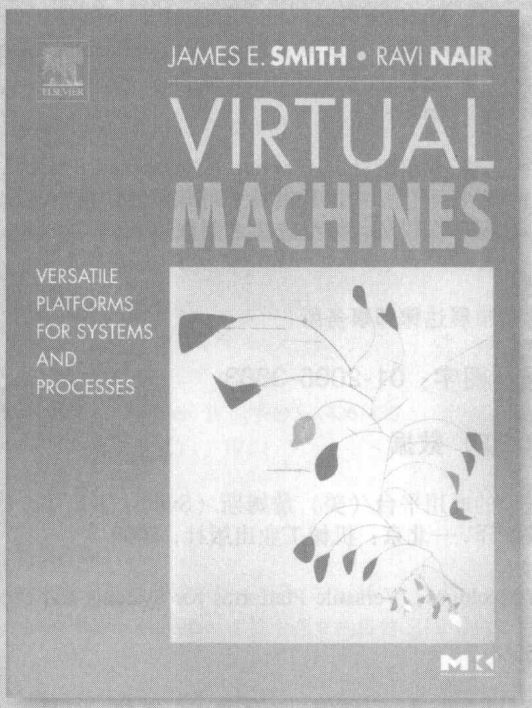
Versatile Platforms for Systems and Processes

机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

虚拟机
系统与进程的通用平台

(美) James E. Smith 威斯康星大学麦迪逊分校 Ravi Nair IBM Thomas J.Watson研究中心 著 安虹 张昱 吴俊敏 中国科学技术大学 译



Virtual Machines

Versatile Platforms for Systems and Processes

机械工业出版社
China Machine Press

本书从计算机体系结构研究者的角度，以计算机系统接口抽象层次中两个最重要的接口——应用二进制接口（Application Binary Interface，ABI）和应用程序接口（Application Program Interface，API）为边界，将计算机系统资源的各种虚拟化技术划分为进程虚拟机和系统虚拟机两大类展开讨论。内容包括体系结构、程序设计语言和编译、操作系统、系统安全等，并借助大量的案例（IBM 的 Daisy、HP 的 Dynamo、Intel/Microsoft 的 EI 等）阐明了虚拟机的基本概念和原理，清晰展现了虚拟化技术各种方法的各个层面及应用。

本书可以作为讲授计算机系统结构研究生课程《虚拟机技术》的教材或教学参考书。工作在虚拟机技术领域的专业人士可以用于自学这些领域的前沿技术。此外，本书还可以作为一本计算机系统软硬件参考资料，它收集了许多相关的实例资料。

Virtual Machines: Versatile Platforms for Systems and Processes
James E. Smith; Ravi Nair
ISBN: 978-1-55860-910-5

ISBN: 978-981-259-708-3

本书版权登记号：图字：01-2006-3883

图书在版编目（CIP）数据

虚拟机：系统与进程的通用平台/（美）詹姆斯（Smith，J. E.），（美）瑞维（Nair，R.）著；安虹，张昱，吴俊敏译. —北京：机械工业出版社，2009.3
（计算机科学丛书）
书名原文：Virtual Machines: Versatile Platforms for Systems and Processes

ISBN 978-7-111-25668-7

Ⅰ. 虚…　Ⅱ. ①詹…　②瑞…　③安…　④张…　⑤吴…　Ⅲ. 虚拟处理机　Ⅳ. TP338

中国版本图书馆 CIP 数据核字（2008）第 194684 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：杨庆燕
北京瑞德印刷有限公司印刷
2009 年 3 月第 1 版第 1 次印刷
184mm × 260mm · 24.75 印张
标准书号：ISBN 978-7-111-25668-7
定价：78.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

华章教育

译者序

进入21世纪，工艺技术的进步和计算机应用的变化推动了计算机体系结构的迅猛发展，赋予了计算机体系结构新的含义。现代计算机系统的硬件结构正在朝着片上多核、系统多级并行处理的方向发展，并且通过Internet网络互联起来，构成功能更强大、应用更广泛的系统。在系统的物理资源大大增加的同时，系统的物理实现也变得极为复杂，系统的可扩展性、可靠性、可用性、可管理性和安全性等方面都遇到了前所未有的、难以用单点体系结构技术来解决的一系列问题。1992年布特勒·兰普逊（Butler Lampson）在他获得图灵奖发表的演说中引用了大卫·韦勒（David Wheeler）的名言："计算机科学中的任何问题都可以通过增加一个中间层来解决"，阐明了用虚拟化技术来解决这一系列问题的大方向，揭示了虚拟机技术发展的历史必然。

本书的作者敏锐地观察到了现代计算机体系结构发展趋势的这一重大变化，从计算机体系结构研究者的角度，以计算机系统接口抽象层次中两个最重要的接口——应用二进制接口（Application Binary Interface，ABI）和应用程序接口（Application Program Interface，API）为边界，将计算机系统资源的各种虚拟化技术划分为进程虚拟机和系统虚拟机两大类展开讨论，清晰地展现了虚拟化技术各种方法的各个层面和各类应用。

第1章首先引入了计算机系统接口的抽象定义，讨论了虚拟化与各层接口的关系。然后从计算机体系结构的概念出发，对各种不同类型的虚拟机进行了分类总结，将虚拟机分为两个主要类型：进程虚拟机和系统虚拟机。

第2章至第6章侧重讨论进程虚拟机。第2章讨论在目标指令集体系结构（Instruction Set Architecture，ISA）上仿真源指令集体系结构的相关问题，并以一种CISC源指令集Intel IA-32、一种RISC目标指令集IBM PowerPC为例来说明；然后以Shade系统为例介绍了二进制翻译技术。第3章讨论进程虚拟机的实现问题，包括指令集的仿真和主机操作系统接口的仿真，最后介绍实例FX!32系统。第4章讨论通过代码优化获得更好的仿真性能的技术，包括各种程序剖析技术，此外还讨论了代码重排序技术，最后介绍了Dynamo动态二进制代码优化器。第5章介绍高级语言虚拟机的体系结构，特别是它们支持面向对象编程和安全的特征，这一章介绍了当今两个重要的面向对象虚拟机——Java虚拟机和微软的CLI。第6章进一步讨论高级语言虚拟机的实现问题，并以Jikes RVM作为案例研究说明本章的概念。

第7章至第9章侧重讨论系统虚拟机。第7章介绍协同设计虚拟机，并以Transmeta Crusoe处理器和IBMAS/400处理器的案例研究结束本章。第8章涉及经典的系统虚拟机及其实现方法，包括本地虚拟机和宿主虚拟机。此外还讨论对计算机系统三个主要资源：处理器、存储器、I/O的虚拟化技术，以及如何用硬件来提高虚拟机的性能。本章给出的研究实例包括：VMware和Intel VT-x（Vanderpool）。第9章讨论多处理器系统的虚拟化问题，包括对不同指令集的客户和主机平台多处理器系统的虚拟化。

第10章介绍了虚拟机技术新兴的应用领域，重点介绍了在安全领域（讨论入侵检测系统的原理以及虚拟机在系统攻防方面的应用潜力）、移动计算环境（讨论了商业上的应用：VMware的VMotion）以及计算网格（展示典型的系统虚拟机对新兴网格系统出现的重要作用）方面的应用。

附录为本书的主要章节提供了计算机系统结构的背景资料，讨论了处理器、存储器、I/O 在计算机系统中的作用。

虚拟机未来应用的广泛性意味着本书适合各种各样的读者，包括从事计算机系统结构、语言和编译、操作系统、应用软件等各个领域的教学和研发人员。本书从写作上具有如下特点：（1）结构清晰。本书从计算机体系结构研究者的角度，以计算机系统接口抽象层次中两个最重要的接口——应用的二进制接口和应用程序接口为边界，将计算机系统资源的各种虚拟化技术划分为进程虚拟机和系统虚拟机两大类展开讨论，清晰地展现了虚拟化技术各种方法的各个层面和各类应用。（2）全面系统。作者从学术和工业应用两个方面对虚拟机技术几十年的研究和发展历史进行了综述，从体系结构、程序设计语言和编译、操作系统及系统安全等多个专业领域深入探讨了虚拟机技术的应用。（3）理实交融。本书提供了大量实际虚拟机系统的原理说明及翔实的参考文献，包括 Shade 模拟系统、FX!32 系统、Dynamo/RIO、Java 和 CLI 等流行语言虚拟机、Jikes RVM、Transmeta Crusoe 处理器、IBM 的 AS/400 和 z/VM 系统、VMware 的主机虚拟机、Intel 的 VT-x 虚拟技术，以及多处理器虚拟系统——Cellular Disco。微软、惠普及其他工业研究团体的本领域研究人员对全书进行了审阅。（4）面向未来。本书除在各章节讨论了虚拟机技术的各种应用以外，还在最后一章专门讨论了一些新兴的虚拟机应用，包括安全领域、移动计算环境、以及计算网格方面的应用。

本书的翻译由中国科学技术大学计算机科学技术系的安虹、张昱和吴俊敏承担。安虹翻译了第 1，2，4，9，10 章和附录，张昱翻译了第 3，5，6 章，吴俊敏翻译了第 7，8 章。翻译完成后，三人进行了互校，最后由安虹对全书进行了统校。研究生隋秀峰、王莉、从明、任永青、王耀彬、李济川等在《虚拟机》课程的学习过程中对相关章节进行了讨论和总结，为理解本书做出了贡献。

本书涉及的知识面较宽，包括计算机体系的结构、编译和操作系统、应用开发环境等许多方面，观点较新，提出了许多全新的概念和方法。因此，在翻译的过程中我们深感难以全面准确地把握原文的译意，译文难免存在错误和不足，敬请读者批评指正。

译　者

2008 年 10 月于中国科学技术大学

前 言

为了获得新的能力，解决与计算机系统主要组件接口的多种问题，操作系统、编程语言和编译器、计算机体系结构三个领域都发展了虚拟机技术。支持操作系统的虚拟机技术过去曾相对活跃过，现在又重新引起人们的兴趣，因为它可以在保持高度安全性的同时获得高效的资源共享。虚拟技术在服务器以及其他网络应用，尤其是对安全要求很高的领域越来越受欢迎。在编程语言方面，虚拟机提供了平台无关性，而且支持动态和透明的翻译和优化。在处理器体系结构方面，虚拟机技术允许引入新的指令集以及动态优化，以降低功耗、提高性能。

工业界对一些标准接口的统一，使得虚拟机技术很可能在上面提到的各个领域的革新中起到重要的作用。新的指令集、操作系统、编程语言要想被广泛接受都需要与虚拟机技术结合。许多虚拟机技术发展的推动力及近来获得的大多数重要进步都来自工业界。

历史上，各种虚拟机技术均延伸到了计算机科学和工程学科中。不过，存在许多基础的交叉技术，如果能发挥这些技术的作用就能够以更好的组织方式来研究和设计虚拟机的实现。本书是按统一原则进行虚拟机技术研究的结晶。

本书介绍了对计算机体系结构的理解。按照传统的定义，体系结构就是一个接口。虚拟机把接口连在一起，同时扩大接口的灵活性和功能性。理解体系结构是理解虚拟技术的关键，本书从体系结构研究者的角度，首先保持与接口的相关问题的清晰，使得读者在阅读后对计算机系统接口的重要性，以及它们在计算机主要部件间相互作用时所扮演的角色有更深的认识。

虚拟机应用的广泛性意味着本书有各种各样需求的读者。虽然目前虚拟机还没有被大学作为一个学科来看待，但因为它与计算机科学和工程学的关键原理：体系结构、操作系统、编程语言紧密结合，使得虚拟机成为研究生课程一个很好的方向。本书的初版已经在四所不同大学的研究生课程中使用，并且取得了一定的成功。本书可以作为关于动态优化的编译器课程或包含典型系统虚拟机的操作系统课程的辅助教材。虚拟机技术正在被工业界广泛而快速地接受，从事相关工作的专业人士将会发现本书也可以用于自学前沿技术。本书还可以作为一本参考资料，因为它收集了许多领域的资料。

本书首先总结了各种不同的虚拟机，从一个视角建立起统一的讨论框架。后面的章节中将描述主要的几类虚拟机，突出它们之间的共同点和低层技术。下面粗略地介绍一下每章的纲要。

第 1 章我们引入了目前计算机系统中流行的抽象概念和接口定义。随后讨论了虚拟化以及其与接口的关系。接着介绍了计算机体系结构概念，总结了各种不同类型的虚拟机。虚拟机可以分为两个主要类型：进程虚拟机和系统虚拟机。本章的最后，我们深入讨论了虚拟机的分类，并提出了一种虚拟机的分类方法。

第 2 章我们讨论了用目标指令集体系结构（ISA）仿真源指令集体系结构的相关问题。说明了基本解释程序的工作，以及如何用线索化解释（threaded interpretation）来提高性能，并使用一种 CISC 源指令集 Intel IA-32、一种 RISC 目标指令集 IBM PowerPC 来说明所开发出的技术。接着介绍二进制转换的概念，并讨论代码发现和代码定位的问题。随后讨论控制转移的处理，许多指令集体系结构都有一些必须用特别方法处理的特殊特征（有些情况下，它们被称为“quirks”）。最后用一个在 Shade 模拟系统中仿真的例子来结束本章。

第3章讨论进程虚拟机的实现。进程虚拟机支持在由操作系统和低层硬件组成的主机平台上运行单独的客户应用程序。我们讨论虚拟机兼容性的含义，同时说明在一个进程虚拟机中的机器状态，包括寄存器状态和存储器状态如何被映射和维护。这一章我们还研究虚拟机运行时软件如何使用自修改代码和存储保护区。虚拟机的仿真包括两部分。首先讨论指令集的仿真，涉及到第2章中讨论的解释程序和二进制翻译。接着讨论主机操作系统接口的仿真。最后我们介绍了FX!32系统，FX!32系统包含很多本章所讨论的基本原理和思想。

第4章重点讨论用于进行代码优化以获得更好的仿真性能的技术。这一章讨论了执行这些优化所需的基本框架和为了帮助代码优化而在程序执行时必须收集的剖析信息。各种剖析技术都将被讨论。因为在更大的代码块上常常能开展更好的优化，我们介绍了动态基本块（dynamic basic block）、超块（superblock）、轨迹（traces）和树组（tree groups）的概念。这一章还延伸讨论了代码重排序及其局限性。各种代码优化技术，包括块间及块内的优化技术都会介绍。最后通过对Dynamo的案例研究来总结这一章，Dynamo是动态二进制的优化器，应用于使用相同源和目标指令集体系结构的系统中。

第5章介绍了高级语言虚拟机，追溯了从早期的Pascal P-code虚拟机到面向对象的虚拟机。重点介绍高级语言虚拟机的体系结构，特别是它们支持面向对象编程和安全的特征。现今两个重要的面向对象虚拟机分别是Java虚拟机和微软的CLI，这一章描述了它们的字节码、面向栈的指令集等特征。在介绍这两种虚拟机时，对指令集的介绍都附以对增强虚拟机的库和应用程序接口集合组成的整个平台的介绍。

第6章继续讨论高级语言虚拟机，重点是它们的实现。在前面的章节中，更多关注的是Java，因为它应用广泛而且有很多不同实现。要特别考虑的两个问题是安全和存储管理。垃圾收集的重要性和执行垃圾收集的技术放在一起讨论。随后讨论Java对象与在Java环境外编写的本地程序之间的交互。我们讨论使用第4章提到的代码优化技术以及针对面向对象模式的新技术来提高Java的性能。本章的概念通过案例研究Jikes RVM整合到一起。

第7章我们介绍协同设计（co-designed）虚拟机。在协同设计虚拟机中，传统指令集体系结构可以通过组合面向实现的指令集体系结构和运行在隐藏存储上的翻译来实现。我们讨论把源指令集体系结构的状态映射到实现的指令集体系结构上的技术，以及维护含有翻译后代码的代码缓存技术。本章还讨论诸如精确中断、缺页中断等许多难题。本章以Transmeta Crusoe处理器和IBMAS/400处理器这两个案例研究结束。

第8章涉及典型的系统虚拟机。系统虚拟机支持在主机平台上运行一个完整的客户操作系统及其所有的应用。我们说明了系统虚拟机的提出动机，并概要介绍系统虚拟机的实现方法，包括本地的（native）和宿主（hosted）的虚拟机。我们还讨论对计算机系统三个主要资源：处理器、存储器、I/O的虚拟化技术。处理器虚拟化的条件最早由Popek和Goldberg在70年代阐明，现在已经得到发展。本章还讨论了当指令集不符合这些条件时的虚拟化技术。在讨论存储器虚拟化时，重点讨论带有结构页表和结构TLB的系统。接着讨论各种I/O设备的虚拟化。然后我们把注意力转移到用硬件来提高虚拟机系统性能上，并以IBM z/VM作为例子来说明。本章最后给出了两个案例：VMware开发的宿主虚拟机系统和Intel为其IA-32架构开发的VT-x（Vanderpool）技术。

第9章，我们把注意力转移到了多处理器系统的虚拟化上。我们介绍了系统分区的概念，并为不同类型的分区开发出一种方法。接着我们讨论了物理分区和逻辑分区的原则。逻辑分区的一个例子是IBM LPAR，随后讨论使用系统管理程序进行逻辑分区。之后我们以Cellular Disco作为案例转向利用基于系统虚拟机的方法来对多处理器系统虚拟化。在本章的最后对采用不同指

令集体系结构的客户和主机平台的多处理器系统的虚拟化进行探讨，特别关注客户与主机之间不同存储器模型之间的桥接。

第10章讨论虚拟机技术的新兴应用。焦点聚焦于我们觉得在未来几年很重要的三个应用领域。第一个是安全领域，我们讨论现代计算机系统易受攻击的弱点，并简单介绍入侵检测系统的原理，讨论了虚拟机在攻击防护和从攻击中恢复方面的应用潜力，我们还讨论了二进制重写技术在关于RIO系统的安全上扮演的角色。第二个应用是将计算环境从一台机器迁移到另一台，这种技术被用在两个系统中：Internet挂起/恢复（Suspend/Resume）系统和Stanford Collective系统，我们讨论了这两个系统，并讨论了商业上的应用：VMware的VMotion。第三个新型应用是计算网格，我们概述了以网格作为计算的基础设施的动机，并与其他类型虚拟机的动机进行比较。最后展示了典型的系统虚拟机对新兴网格系统出现的重要作用。

附录其实是对计算机系统的浓缩，以提供本书主要章节的背景资料。在附录中，首先讨论了处理器、存储器、I/O在计算机系统中的作用；其次讨论了指令集体系结构，包括对用户应用程序的支持和对操作系统的支持；还讨论了页表和TLB。再次讨论了操作系统的主要组件和应用程序与操作系统之间的系统调用接口。最后，我们讨论了多处理器体系结构，包括集群结构和共享存储多处理器系统，还讨论了共享存储系统中的存储一致性问题。

本书可以不同的方法在课程中使用。大体上，本书是按照把虚拟机作为一门课程的论题（这是我们推荐的方式）来组织的。对于讲述虚拟机的操作系统课，教师可以在介绍第1章后直接进入第8到10章，第2到5章可以稍后作为了解实现细节来讨论。偏向硬件的课程可以从第1到4章开始，然后跳过5、6章，直接进入剩余章节。偏向编程语言的课程可以在完成第1章后直接进入第5章，然后再回到第2到第4章，最后通过第6章把所有内容结合到一起。使用本书中材料的任何课程都可以把第10章作为兴趣篇阅读。

特别感兴趣的技术人员可以自主决定他们阅读本书的顺序，在编写本书过程中，我们力图使读者可以从任何一章感兴趣的开始阅读，并且在阅读整章内容时，只需偶尔翻阅其他章节的内容。

能完成这本书，我们要感谢许多人。我们要特别感谢许多审阅人，他们是IBM研究中心的Michael；菲利浦研究中心的Jan Hoogerbrugge；微软研究中心的Jim Larus；Sun微系统的Tim Lindholm、Bernd Mathiske；以及哈佛大学的Mike Smith。他们耐心地通读了全文，向我们反馈了许多有价值甚至是批评的非常有用的意见。我们也要感谢许多通读了特定章节的审阅人，给予本书有价值的洞察和评价。这些审阅人包括IBM研究中心的Erik Altman、Peter Capek、Evelyn Duesterwald和Michael Gschwind；佛罗里达大学的Renato Figueiredo；加州大学Irvine分校的Michael Franz；明尼苏达大学的Wei Hsu；UPC-Barcelona的Toni Juan；Intel的Alain Kägi；Vmware的Beng-Hong；马萨诸塞大学的Eliot Moss；IBM Rochester研究中心的Frank Soltis；Intel的Richard Uhlig；IBM Endicott研究中心的Romney White；普林斯顿大学的Wayne Wolf和微软研究中心的Ben Zorn。我们很荣幸就虚拟机的各个方面与IBM的Vas Bala、Ek Ekanadham、Wolfram Sauer以及Charles Webb进行了讨论。

作者要感谢Sriram Vajapeyam在早期整理本书的资料时所做的贡献。威斯康辛大学麦迪逊分校和加泰罗尼亚理工大学Barcelona分校的学生们在学习虚拟机课程和开展虚拟机研究过程中提供了有价值的反馈意见。过去和现在对本书有过帮助的学生包括：Nidhi Aggarwal，Todd Bezenek，Jason Cantin，Wooseok Chang，Ashutosh Dhodapkar，Timothy Heil，Shiliang Hu，Tejas Karkhanis，Ho-Seop Kim，Kyle Nesbit，和Subramanya Sastry，还有一些学生在此未能一一列出。

本书的出版还包含了出版人Denise Penrose给予的指导，坚持不懈的努力和鼓励，以及她在

Morgan-Kaufmann 出版社的优秀员工们的支持，他们包括 Kimberlee Honjo，Angela Dooley，Alyson Day，和 Summer Block。

第一作者：我要感谢 IBM 研究中心的人，特别是 Dan Prener，他们在我写作本书的初稿期间（2000～2001）给予了支持。特别感激 Erik Altman，在我写作本书的过程中充当了宣传者。我还要感谢我的研究生们对本书的支持和所提出的有用的建议。最后，我要谢谢我的孩子们 Barbara，Carolyn 和 Jim，在我写作本书的过程中他们给予了鼓励和付出的耐心，容忍经常心烦意乱的父亲。

第二作者：我要感谢 Dan Prener，Eric Kronstadt，和 Jaime Moreno 所给予的鼓励和支持。我还要感谢与 Peter Capek，Dan Prener，Peter Oden，Dick Attanasio 和 Mark Mergen 在茶歇时间里的讨论。最后，我要感谢我的妻子 Indira，我的女儿 Rohini 和 Nandini，她们自始至终给予我爱和理解。她们给予我的总是超乎我的想像。

两位作者互致感谢，我们能有机会重温延续了 30 多年的友谊。我们有着极为相同的兴趣爱好，在写这本书的过程中学到了许多东西。读者如果能够体验我们有过的经历中的一小部分，写作这本书就是值得的。

James E. Smith
Ravi Nair

目 录

第 1 章　虚拟机导论

现代计算机是人类工程学中最先进的结构，人类对极度复杂性的控制能力成就了现代计算机今日的辉煌。计算机系统由许多硅芯片组成，而每个芯片则由上亿个晶体管构成。这些芯片相互连接，并与高速的输入/输出设备、网络设施一起组成可以运行软件的平台。操作系统、应用程序和库、图形和网络软件，所有这些协同工作，为数据管理、教育、通信、娱乐以及许多其他应用提供了强大的计算机环境。

管理计算机系统复杂性的关键是通过一些*定义明确的接口*把计算机系统划分成不同的*抽象层次*。抽象层次允许忽略或简化系统设计的底层实现细节，从而简化高层组件的设计。例如，硬盘被划分为不同的磁道和扇区，它的细节经过操作系统的抽象，使应用程序看到的硬盘是不同大小的文件集合（图 1-1）。在这之后应用程序可以创建、读、写文件，而并不需要了解硬盘是如何构造和组建的。

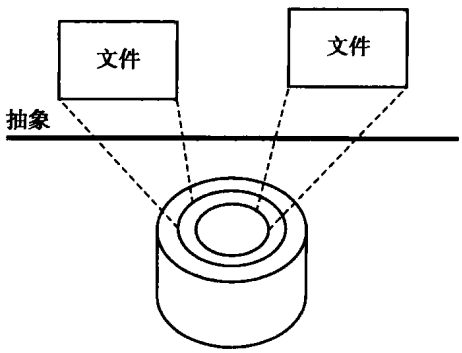


图 1-1　硬盘抽象成文件。一个层次的抽象提供了到底层资源的简单接口

计算机系统的抽象层次通过分层组织，底层由硬件实现，高层由软件实现。在硬件层，所有的组件都是物理的，有真实的特性，各部分通过定义的接口物理地连接在一起。在软件层，组件都是逻辑的，较少受物理特性的限制。本书主要涉及硬件和软件分界处及其附近的抽象层次，在这些层次上软件从运行它的*机器*中分离出来。

计算机软件由“*机器*”来执行（“机器（Machine）”这个术语从计算机出现时就有，现在比较流行的是“*平台*（Platform）”这个术语）。从操作系统的角度看，“机器”主要由一些硬件组成，包括一个或多个运行特定指令集的处理器、实存储器、I/O 设备。但是，“*机器*”这个术语的使用并不局限于计算机的硬部件。例如，从应用程序的角度看，“机器”是指操作系统与通过用户级二进制指令可访问的那部分硬件的组合。

现在我们再来讨论如何使用*定义明确的接口*来管理复杂的情况。明确的接口可以分解计算机的设计任务，使得硬件和软件设计组能够或多或少地独立开展工作。指令集就是这样一个接口。例如，Intel 和 AMD 的设计师开发实现 IA-32 指令集⊖的微处理器，同时微软的软件工程师

⊖ 有时候非形式地称为 x86。

⊜ 此页码为英文原书页码，与索引中的页码一致。——编辑注

开发把高级语言映射到该指令集的编译器。只要这两个设计小组都能够遵从该指令集的规范定义，编译好的软件就可以在由 IA-32 处理器组成的机器上正确执行。操作系统接口被定义为一组函数调用的集合，是计算机系统中另一个重要的标准化接口。像前面的 Intel 和微软的例子，定义明确的接口使不同公司或不同时间（甚至相差几年的时间）开发的计算机子系统之间可以交互。应用软件开发者不需要知道操作系统内部细节的变化，硬件和软件可以根据各自不同的开
2 发进度独立地升级。软件可以在实现相同指令集的不同平台上运行。

除了这些优点，定义明确的接口也有局限性。为某个接口规范设计的子系统或组件不能和那些为其他接口规范设计的子系统或组件一起工作。有不同指令集的处理器（如 Intel 的 IA-32 和 IBM 的 PowerPC），也有不同的操作系统（如 Windows 和 Linux）。应用程序以二进制程序分发后，就与特定的指令集和操作系统绑定了。一个操作系统则与实现特定的存储、I/O 系统接口的计算机绑定在一起。总的说来，指令集、操作系统、应用程序语言的多样性要求革新，反对停滞。但在实际应用中，这些多样性也削弱了它们之间协同工作的能力，尤其是限制了在网络计算机上像移动数据那样自由地移动软件。

在硬件/软件接口之下，硬件资源也会限制软件系统的灵活性。在高级语言和操作系统中，存储器和 I/O 的抽象已经消除了对硬件资源的很多依赖，但是有些依赖仍然存在。许多操作系统是为特定的系统结构开发的，例如是为单处理器或共享存储的多处理器开发的，而且被设计成能直接管理硬件资源。这就意味着，系统的硬件资源是由单一的操作系统管理的。在单一的管理体制下，所有的硬件资源被绑定到单个实体中。这样做，反过来又限制了系统的灵活性，不仅限制了可用的应用软件（如前面讨论的），特别是，当系统被多个用户或用户组共享时，在安全性和故障隔离方面也限制了系统的灵活性。

*虚拟化*提供了一种放宽前述的限制，增加灵活性的方法。当一个系统（或子系统），例如处理器、存储器或输入/输出设备被虚拟化，它的接口和所有通过接口可见的资源都被映射到实现它的真实系统的接口和资源上。从而，真实系统可以表现为一个不同的虚拟系统或多重的虚拟系统的集合。形式上，虚拟化是构建一个将虚拟的客户系统映射到真实的主机系统上（Popek and Goldberg 1974）的同态。如图 1-2 所示，这个同态是将客户机的状态映射到主机的状态（图 1-2 中的函数 V），对于修改客户机状态的操作序列 e（函数 e 将状态 S_i 修改为 S_j），在主机上也有相应的操作序列 e'，e'在主机上执行对主机状态的等价的修改（将 S'_i 变成 S'_j）。虽然这样的
3 同态可以同时用来描述虚拟化和抽象，但我们要加以区分：虚拟化不同于抽象，虚拟化不需要隐藏细节；虚拟系统的细节通常与底层真实系统的细节相同。

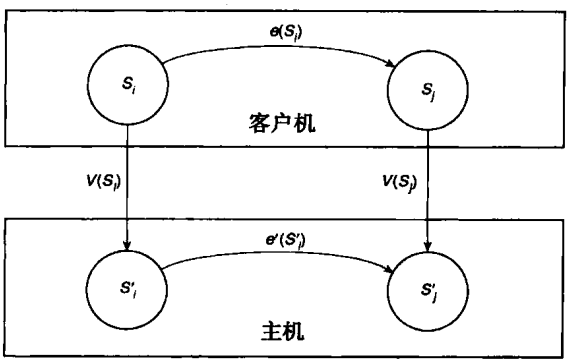


图 1-2 虚拟化。形式上，虚拟化是在客户系统和主机系统之间构建一个同态；e'o $V(S_i) = V$ o $e(S_i)$

再来看一下硬盘的例子。有些应用中需要将一个完整的大硬盘分成许多小的虚拟盘，这些虚拟盘中的每一个都被映射为真实盘上的一个大文件（如图 1-3）。虚拟软件提供虚拟盘内容和

真实盘内容之间的映射（同态中的 V 函数），把文件抽象作为中间步。每个虚拟盘表面上都包含一些逻辑磁道和扇区（虽然比大硬盘中的少）。对虚拟盘的写操作（同态中的 e 函数）被镜象为在主机系统中对文件的写操作和对相应的真实盘的写操作（同态中的 e' 函数）。在这个例子中，虚拟盘接口所提供的细节级别，如扇区和磁道寻址，都与真实盘相同，并没有发生任何抽象。

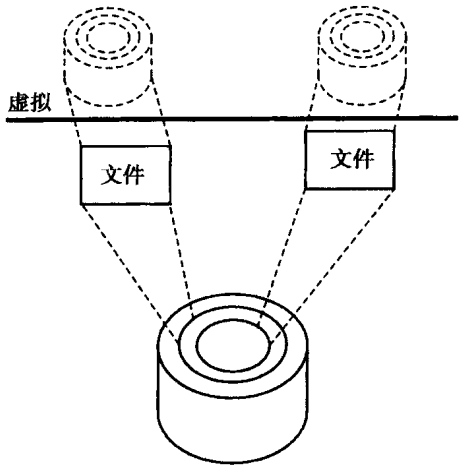


图 1-3 虚拟硬盘实现。虚拟化在同一抽象层提供不同的接口和/或资源

虚拟化的概念不仅可以被应用到硬盘等子系统中，也可以被用于整台机器。虚拟机（VM）通过在真实机器上增加一层软件来支持所需实现的虚拟机体系结构。例如，在 Apple Macintosh 上安装虚拟软件就可以提供一个 Windows/IA-32 虚拟机，在这个虚拟机上可以运行 PC 应用程 4
序。通常，虚拟机可以绕开真实机器的兼容性限制和对硬件资源的限制，从而获得更高的软件可移植性和灵活性。

有各种各样的虚拟机提供各种各样的用途。多拷贝虚拟机可以在一个硬件平台上为个人或用户组提供他们所需要的操作系统环境。这些不同的系统环境（可能有不同的操作系统）还提供隔离性和增强的安全性。一个大型多处理器服务器可以被划分成一些小的虚拟服务器，并同时保持整个系统硬件资源使用的平衡性。

虚拟机也可以利用仿真技术提供跨平台的软件兼容性。例如，一个实现了 PowerPC 指令集的平台可以被转换成一个可运行 IA-32 指令集的虚拟平台。从而，为一个平台所写的软件可以在另一个平台上运行。这种兼容性可以在系统级提供（例如，在 Macintosh 上运行一个 Windows 操作系统），也可以在程序或进程级提供（例如，在 Solaris/SPARC 平台上运行 Excel）。除了仿真，虚拟机还可提供动态、在线的二进制程序优化。最后，通过仿真，虚拟机还可以在支持现有的标准指令集的程序的同时实现新的私有指令集，如并入超长指令字（Very Long Instruction Words，VLIWs）。

上面提到的虚拟机都是为匹配现有的真实机器的体系结构构建的。然而，也有虚拟机没有 5
相应的真实机器。语言开发人员为一个新设计的高级语言而发明一个虚拟机的做法已经十分普遍。用这种高级语言编写的程序被事先编译成针对虚拟机的二进制代码，然后任何实现这个虚拟机的真实机器都可以运行已编译好的代码。Java 语言和 Java 虚拟机已经明确证实了这种方法的作用。这种方法达到了很高的平台无关性，可以获得相当灵活的网络计算环境。

操作系统开发人员、语言设计人员、编译器开发人员和硬件设计人员都从各自的角度研究和构建虚拟机。虽然每一种虚拟机应用都有它独特的性质，但是它们的基本概念和技术在虚拟机范围内有很多共同点。由于各种不同的虚拟机体系结构和支撑技术是由不同的工作组开发的，统一虚拟机的知识体系、理解各种不同形式虚拟机的共同的支撑技术显得特别重要。这本书的