国外计算机科学教材系列

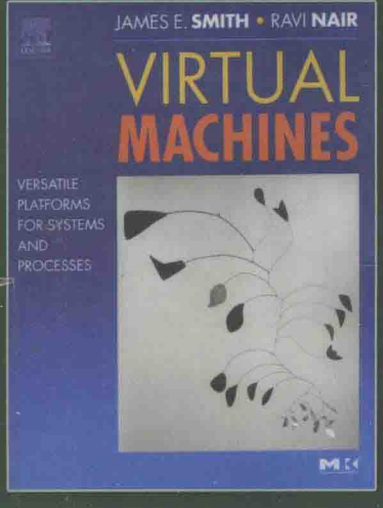
ELSEVIER

虚拟机

——系统与进程的通用平台

Virtual Machines

Versatile Platforms for Systems and Processes



英文版

[美] James E. Smith
Ravi Nair 著

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

国外计算机科学教材系列

虚 拟 机

—— 系统与进程的通用平台

（英文版）

Virtual Machines

Versatile Platforms for Systems and Processes

[美] James E. Smith
Ravi Nair 著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

本书介绍了虚拟机技术在操作系统、程序设计语言和计算机体系结构方面的应用。本书内容全面，处于学科的最前沿。书中以清晰而深入的方式，借助有效的应用实例研究，包括IBM的Daisy，HP的Dynamo以及Intel/Microsoft的EL等各种系统，阐明了虚拟机的基本概念和原理。内容涉及虚拟机的分类、虚拟机的抽象、用目标指令集系统仿真源指令集系统、进程虚拟机的实现、用于提高仿真性能的代码优化技术、高级语言虚拟机及其实现、协同设计虚拟机、多处理器系统的虚拟化，以及其他新兴的虚拟机应用。

本书适合于现代计算机体系结构设计、程序设计语言、操作系统和安全技术，以及系统实现方面的学生和从业者阅读。对未来计算系统感兴趣的读者，也会从本书中获益。

版权贸易合同登记号　图字：01-2006-3396

图书在版编目（CIP）数据

虚拟机——系统与进程的通用平台 = Virtual Machines: Versatile Platforms for Systems and Processes/（美）史密斯（Smith, J. E.）等著. -北京：电子工业出版社，2006.7
（国外计算机科学教材系列）
ISBN 7-121-02672-4

Ⅰ. 虚... Ⅱ. 史... Ⅲ. 虚拟处理机-教材-英文 Ⅳ. TP338

中国版本图书馆CIP数据核字（2006）第051678号

责任编辑：史　平
印　　刷：北京市天竺颖华印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
经　　销：各地新华书店
开　　本：787 × 980　1/16　　印张：41　字数：918千字
印　　次：2006年7月第1次印刷
定　　价：68.00元

出 版 说 明

21世纪初的5至10年是我国国民经济和社会发展的重要时期，也是信息产业快速发展的关键时期。在我国加入WTO后的今天，培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡，是我国面对国际竞争时成败的关键因素。

当前，正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期，为使我国教育体制与国际化接轨，有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材，以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验，翻译出版了“国外计算机科学教材系列”丛书，这套教材覆盖学科范围广、领域宽、层次多，既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时，我们也适当引进了一些优秀英文原版教材，本着翻译版本和英文原版并重的原则，对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上，我们大都选择国外著名出版公司出版的高校教材，如Pearson Education培生教育出版集团、麦格劳－希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者，如道格拉斯·科默（Douglas E. Comer）、威廉·斯托林斯（William Stallings）、哈维·戴特尔（Harvey M. Deitel）、尤利斯·布莱克（Uyless Black）等。

为确保教材的选题质量和翻译质量，我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士，也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中，为提高教材质量，我们做了大量细致的工作，包括对所选教材进行全面论证；选择编辑时力求达到专业对口；对排版、印制质量进行严格把关。对于英文教材中出现的错误，我们通过与作者联络和网上下载勘误表等方式，逐一进行了修订。

此外，我们还将与国外著名出版公司合作，提供一些教材的教学支持资料，希望能为授课老师提供帮助。今后，我们将继续加强与各高校教师的密切联系，为广大师生引进更多的国外优秀教材和参考书，为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

教材出版委员会

About the Authors

James E. Smith is a Professor in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison. He first joined the University of Wisconsin in 1976, after receiving his PhD in Computer Science from the University of Illinois. From 1979 to 1981, he took a leave of absence to work for the Control Data Corporation in Arden Hills, MN, participating in the design of the CYBER 180/990. From 1984 to 1989, he participated in the development of the ACA ZS-1, a scientific computer employing a dynamically scheduled, superscalar processor architecture. In 1989, he joined Cray Research, Inc. in Chippewa Falls, WI. While at Cray Research, he headed a small research team that participated in the development and analysis of future supercomputer architectures.

In 1994, he re-joined the ECE Department at the University of Wisconsin. His recent research concerns the development of the virtual machine abstraction as a technique for providing high performance through co-design and tight coupling of virtual machine hardware and software. Prof. Smith was the recipient of the 1999 ACM/IEEE Eckert-Mauchly Award for his contributions to the field of computer architecture.

Ravi Nair has been a Research Staff Member since 1978 at the IBM Thomas J. Watson Research Center, where he has helped in the architecture and design of a number of processors. He has worked in the areas of computer architecture, performance analysis, multiprocessor virtualization, design automation, and testing, and has several publications, patents, and IBM awards in these areas. Among the many design and analysis tools he has developed are binary rewriting tools for profiling, trace generation, and simulation. His current interests include processor microarchitecture, dynamic compilation, and virtual machine technology. Dr. Nair graduated with a B.Tech. degree in electronics and electrical communication from IIT, Kharagpur in 1974, and with a Ph.D. degree in Computer Science from the University of Illinois in 1978. He spent a sabbatical year at Princeton University and has also taught at Columbia University. Dr. Nair is a member of the IBM Academy of Technology and a Fellow of the IEEE.

Foreword

I've been a virtual machine addict for precisely as long as I've worked with computers. My first real job, which led to my first nontrival computer program, was to implement a virtual machine for a high-level programming language. There seemed to be something magical about the ability for a computer to imitate another computer, or my idea of what a computer *ought* to be.

Now almost 20 years later, less starry-eyed and more responsible, I am concerned that my work has utility, making it faster or easier to get something done than it would be otherwise. But lucky for me, virtual machines have proven ideally suited to the needs of the computing industry, where the appreciation of the virtual machine has expanded dramatically. It's no longer only an intellectual challenge to the researcher arguably on the "lunatic fringe" of programming language implementation, or the secret weapon of a small cadre of mainframe O/S engineers and pioneers of system virtualization.

Several major trends have contributed to an explosion of interest in virtual machine technologies. In particular the rise of the World Wide Web, by far the largest and most ubiquitous cross-platform computing environment to date, created enormous and visible opportunities for virtual machine-based computing. Initially targeting the WWW, VM technology hit the mainstream as a means of safely hosting cross-platform binary programs embedded in Web pages. From that beachhead it has expanded to become the prevalent approach to open platforms from Web and back office servers to cell phones and smart cards, where the equivalent benefits — cross-platform applications that are not tied to the underlying hardware or operating system — invariably apply. Virtual machines form the basis of Sun's Java and Microsoft's .NET platforms, currently the most popular software environments on the planet.

As new markets or applications are conceived, virtual machine technologies to support them are adapted from these stalwarts.

In other markets as well, virtual machine technology is seeing a renaissance. Companies such as VMware are finding virtual machine platforms to be an ideal way to provide greater resource control, scalability and performance across variations in host hardware or operating systems. Virtual machines are likely to be common at multiple levels of the data center or server farm.

When I was asked to review this book, it seemed an opportunity to read something I might not get to otherwise. I expected to focus on the chapter covering virtual machines for high-level languages. Little did I know that I would find myself excited by less familiar topics, for instance sending back surprised comments expressing admiration for the decades-old IBM AS/400 architecture, which I'd never understood before. It wasn't just the realization of how much those coming far before me had accomplished. Seeing virtual machine technology in this broader scope made design decisions in my familiar Java virtual machine architecture, and their implications, much clearer. Such perspective is valuable even to experienced practitioners of a particular VM art.

And I found myself once again thinking how cool it all is.

Tim Lindholm
Distinguished Engineer, Sun Microsystems, Inc.
Palo Alto
February 28, 2005

Preface

Virtual machine (VM) technologies have been developed in a number of contexts — operating systems, programming languages and compilers, and computer architecture — to enable new capabilities and to solve a variety of problems in interfacing major computer system components. Virtual machines for supporting operating systems are receiving renewed interest after years of relatively little activity, because they allow effective resource sharing while maintaining a high degree of security. Virtualization is becoming popular for servers and other network applications especially where security is of crucial importance. In the area of programming languages, virtual machines provide platform independence, and they support transparent dynamic translation and optimization. In processor architectures, virtual machine technologies allow the introduction of new instruction sets, as well as dynamic optimization for power reduction and/or performance improvement.

Because of industry consolidation around a small number of standard interfaces, virtual machine technology will likely be an important enabling feature for innovations in all of the above fields. Any new instruction set, operating system, or programming language will almost certainly require some accompanying virtual machine technology if it is to become widely accepted. Not coincidentally, much of the impetuses for virtual machine technologies, and most of the more significant recent developments, have come from industry.

Historically, the various VM techniques have been spread across computer science and engineering disciplines. However, there are a number of underlying, cross-cutting technologies, and there is much to be gained by pulling them together so that VM implementations can be studied and engineered

in a well-structured way. This book is an outgrowth of the idea that virtual machines should be studied as a unified discipline.

This book is also about computer architecture in its purist sense. As classically defined, an *architecture* is an *interface*. Virtual machines couple interfaces and extend the flexibility and functionality of the interfaces. Understanding architecture is key to understanding virtual machines, and this book is written from an architect's perspective, keeping interface issues clear and at the forefront. A goal is for the reader to come away with a much deeper understanding of the important computer system interfaces and the role these interfaces play when the major components interact.

The breadth of VM applications implies the audience for this book is fairly diverse. Although it is not currently recognized as a discipline with a targeted set of university courses, virtual machines makes an excellent topic for a graduate level course because it ties together the key disciplines of computer science and engineering: architecture, operating systems, and programming languages. Preliminary versions of this book have already been used, quite successfully, in graduate courses at four different universities. The book can also be used as a supplementary text for a compiler course on dynamic optimization or an operating system course covering classic system VMs. Virtual machine technologies are rapidly gaining broad acceptance in industry, and practicing professionals will find the book useful for self-education on this leading edge technology. The book can also serve as a useful reference as it collects material from a number of fields into one place.

The book begins by surveying the variety of VMs, putting them into perspective and building a framework for discussing VMs. The subsequent chapters describe the major types of VMs in an organized way, emphasizing the common, underlying technologies. Following is a rough outline summarizing each chapter.

In Chapter 1 we introduce the concept of abstraction and define the interfaces that are prevalent in today's computer systems. This is followed by a discussion of virtualization and its relationship to the interfaces. The notion of computer architecture is introduced next, followed by a survey of different types of virtual machines. VMs are shown to fall into two main categories, process virtual machines and system virtual machines. We end the chapter by refining this categorization further and suggesting a taxonomy for virtual machines.

In Chapter 2 we address issues related to the emulation of a source instruction set architecture (ISA) on a target ISA. We demonstrate the workings of a basic interpreter and show how threaded interpretation can help improve performance. The techniques developed are demonstrated using a CISC source ISA, the Intel IA-32, and a RISC target ISA, the IBM PowerPC. We then

introduce the notion of binary translation, and discuss the problems of code discovery and code location. This is followed by a discussion of the handling of control transfers. Many ISAs have special features (in some cases, they might be called "quirks") that must be handled in special ways. These are discussed next. The chapter is rounded out with a case study of emulation in the Shade simulation system.

Chapter 3 discusses the implementation of process virtual machines. A process virtual machine supports an individual guest application program on a host platform consisting of an operating system and underlying hardware. We discuss the implications of VM compatibility and show how the state of a machine, consisting of the register state and the memory state, is mapped and maintained in a process virtual machine. We address the issues of self-modifying code and of protecting the region of memory used by the VM runtime software. VM emulation consists of two parts. First, the emulation of the instruction set is discussed, with reference to interpretation and binary translation discussed in Chapter 2. This leads to a discussion of code caching techniques. Next, the emulation of the interface to the host operating system is addressed. We end the chapter by describing the FX!32 system, which embodies many of the fundamental ideas discussed in the chapter.

Chapter 4 focuses on techniques for the optimization of translated code for better emulation performance. It discusses the framework needed to perform such optimizations and the profile information that must be gathered at program execution time in order to facilitate optimizations. Various profiling techniques are discussed. Because optimizations often work better on larger code blocks, the concepts of dynamic basic blocks, superblocks, traces, and tree groups are introduced. The chapter includes an extensive discussion on code re-ordering and its limitations. Various types of code optimizations, both local and inter-block, are presented. The chapter concludes with a case-study of Dynamo, a dynamic binary optimizer, which applies optimization techniques in a system where the source and target ISAs are identical.

Chapter 5 introduces high-level language virtual machines and traces the transition from the early Pascal P-code VMs to object-oriented VMs. The emphasis in this chapter is on the architecture of high-level language VMs, especially those features supporting object-oriented programming and security. The two important object-oriented VMs of today, namely the Java Virtual Machine and the Microsoft CLI, are described. The features of their bytecode, stack-oriented instruction sets are described. In both cases, the description of the instruction set is supplemented by a discussion of the overall platform that augments the virtual machines with a set of libraries and APIs.

Chapter 6 continues the discussion of high-level language VMs by focusing on their implementation. As in the preceding chapter, more attention is given

to Java because of its widespread usage and the variety in its implementations. Two aspects given special consideration are security and memory management. The importance of garbage collection is discussed along with techniques for performing garbage collection. The interaction of Java objects with programs written natively outside the Java environment is discussed next. We then describe how the performance of Java can be enhanced through optimizations of code using techniques described in Chapter 4 as well as new techniques that are specific to the object-oriented paradigm. The concepts in the chapter are brought together using a case-study of the Jikes Research Virtual Machine.

In Chapter 7 we discuss co-designed virtual machines where a conventional ISA is implemented through a combination of an implementation-specific ISA and translation software that runs in concealed memory. We discuss techniques for mapping the state of the original ISA onto the implementation ISA and for maintaining the code cache containing translated code. Various sticky aspects, including the implementation of precise interrupts and page faults, are also discussed. We end the chapter with two case studies: the Transmeta Crusoe processor and the IBM AS/400 processor.

Chapter 8 deals with the classic system virtual machines. A system virtual machine supports a complete guest operating system and all its applications on a host platform. We provide a motivation for these virtual machines and outline the basic ways for implementing system virtual machines, including native and hosted VMs. We discuss techniques for the virtualization of the three main system resources: processors, memory, and I/O. The conditions for virtualizability of a processor, as first enunciated by Popek and Goldberg in the early '70s, are developed. Also discussed are techniques for virtualization when these conditions are not satisfied by an ISA. Memory virtualization is discussed with attention given both to systems with architected page tables and architected TLBs. Then virtualization is discussed for a variety of I/O devices. We next turn our attention to hardware assists to improve the performance of VM systems with the IBM z/VM as a running example. We end the chapter with two case studies, that of a hosted VM system developed by VMware, and that of the VT-x (Vanderpool) technology developed by Intel for their IA-32 architecture.

In Chapter 9 we shift our attention to the virtualization of multiprocessor systems. We introduce the notion of system partitioning and develop a taxonomy for different types of partitioning. We then discuss the principles behind physical partitioning and logical partitioning. The IBM LPAR feature is presented as a case study in logical partitioning. Following this is a discussion about logical partitioning using hypervisors. We then turn to a system VM-based approach to multiprocessor virtualization using a research system, Cellular Disco, as a case study. We end the chapter with a discussion of multiprocessor

virtualization where the guest and host platforms have different ISAs, with special attention on bridging the differences in memory models between a host and a guest.

Chapter 10 is a discussion of emerging applications for virtual machine technology. We focus on three application areas which we feel will be important in the coming years. The first is security. We discuss the vulnerability to attacks of modern computer systems and outline the principles behind intrusion detection systems. The potential of VM technology in protecting and recovering from attacks is discussed. The role of binary rewriting technology in security is also discussed with reference to the RIO system. The second application we focus on is that of migrating computing environments from one machine to another. The techniques used in two systems, the Internet Suspend/Resume system and the Stanford Collective system, are described. The commercial application of this technology in VMware's VMotion is also discussed. Our third emerging application is computational grids. We outline the motivation behind the emergence of the grid as a computing infrastructure and compare it to the motivations behind other types of virtual machines. We end the chapter by showing how classic system virtual machines are proving to be an important enabler for emerging grid systems.

The Appendix is essentially a condensed course in computer systems, providing background material for the main chapters. It discusses the roles of the processor, memory, and I/O in a computer system. This is followed by a discussion of ISAs, including support for user applications as well as for the operating system. Page tables and TLBs are discussed next. We follow this with a discussion of the major components of an operating system and the system call interface between an application and the operating system. Finally we discuss multiprocessor architectures, including cluster architectures and shared-memory multiprocessor systems. Memory coherence and memory consistency issues in shared-memory systems are also addressed.

The book may be used in a course in a variety of ways. Overall, the book is structured for a course focused on virtual machines as a topic in itself (an approach we advocate). For an operating system oriented treatment of virtual machines, an instructor can go straight to Chapters 8 through 10 after introducing the taxonomy of virtual machines in Chapter 1. Chapters 2 through 5 can then be discussed to get an idea of implementation details. A more hardware oriented course can, however, go through Chapters 1 through 4 and then skip Chapters 5 and 6, before covering the remaining chapters. A language-oriented course can go straight to Chapter 5 from Chapter 1, and then backtrack to do Chapters 2 through 4, ending with Chapter 6 to put everything together. Chapter 10 should be of interest to virtually any course using the material in the book.

The specific interests of practitioners will largely determine the order in which they cover the material in the book. We have written the book in such a way that an interested reader can start at the beginning of any chapter of interest and follow the material in the complete chapter with only occasional excursions to sections in other chapters referred to in the text.

There are many people we would like to thank for having made this book possible. We would particularly like to thank the many reviewers. Michael Hind of IBM Research, Jan Hoogerbrugge of Philips Research, Jim Larus of Microsoft Research, Tim Lindholm of Sun Microsystems, Bernd Mathiske of Sun Microsystems, and Mike Smith of Harvard University patiently went through the text of the entire book and provided us with valuable feedback, sometimes critical, but always useful. We also thank a number of reviewers who went through specific chapters or sets of chapters and provided us with their valuable insights and comments. These reviewers include Erik Altman, Peter Capek, Evelyn Duesterwald, and Michael Gschwind, all of IBM Research, Renato Figueiredo of the Univ. of Florida, Michael Franz of UC Irvine, Wei Hsu of the Univ. of Minnesota, Toni Juan of UPC-Barcelona, Alain Kägi of Intel, Beng-Hong Lim of VMware, Eliot Moss of Univ. of Massachusetts, Amherst, Frank Soltis of IBM Rochester, Richard Uhlig of Intel, Romney White of IBM Endicott, Wayne Wolf of Princeton University, and Ben Zorn of Microsoft Research. We also appreciate the discussions with Vas Bala, Ek Ekanadham, Wolfram Sauer, and Charles Webb of IBM, on various aspects of virtualization.

The authors would also like to acknowledge Sriram Vajapeyam for his contributions during the early development of this material, and the students at the University of Wisconsin-Madison and Universitat Politècnica de Catalunya in Barcelona for their valuable feedback while participating in VM courses and conducting VM research. At the risk of omitting someone, the past and current students who have been particularly helpful are Nidhi Aggarwal, Todd Bezenek, Jason Cantin, Wooseok Chang, Ashutosh Dhodapkar, Timothy Heil, Shiliang Hu, Tejas Karkhanis, Ho-Seop Kim, Kyle Nesbit, and Subramanya Sastry.

This book owes a lot to the guidance, persistence, and encouragement provided by our publisher, Denise Penrose, and the support provided by her excellent staff at Morgan-Kaufmann Publishers, including Kimberlee Honjo, Angela Dooley, Alyson Day, and Summer Block.

First author: I would like to thank the people at IBM Research, and Dan Prener in particular, for their support during my 2000-2001 sabbatical – the time this book had its genesis. I am especially grateful to Erik Altman for being a sounding board throughout the writing of the book. I also thank my graduate students for their support and helpful suggestions. Finally, I am grateful to my children Barbara, Carolyn, and Jim, for their encouragement and patience

during the writing of the book, and in general for putting up with a frequently distracted father.

Second author: I would like to thank Dan Prener, Eric Kronstadt, and Jaime Moreno for their encouragement and support in undertaking this project. Thanks also to Peter Capek, Dan Prener, Peter Oden, Dick Attanasio, and Mark Mergen for many interesting tea-time discussions. Finally, I would like to thank my wife, Indira, and my daughters, Rohini and Nandini, for their love and understanding at all times; they have given me more than I could have ever hoped for or imagined.

The authors are mutually grateful for the opportunity to renew a friendship that stretches back 30 years. We have had tremendous fun and have learnt a great deal in the process of writing this book. If you, the reader, experience just a fraction of what we have experienced, this book will have been worthwhile.

James E. Smith

Ravi Nair

Contents