TURING 图灵程序设计丛书

Advanced Design and Implementation of Virtual Machines

虚拟机设计与实现
以JVM为例

李晓峰（Xiao-Feng Li）◎著　单业◎译

- 虚拟机技术高手心得倾囊相授，深入、详尽剖析虚拟机工作原理
- 业内行家热忱推荐，不可多得的虚拟机研发的进阶秘笈

CRC Press
Taylor & Francis Group
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

“本书论述了关于虚拟机设计与实现的高级主题，已经成为我不可或缺的参考书。我向系统软件开发者，尤其是托管运行时系统的开发者，强烈推荐本书，因为本书能够清晰地解答他们在探索虚拟机相关话题时所产生的疑问。”

—— **周志德** Futurewei Technologies首席科学家

“当前语言虚拟机的应用范围越来越广，各类新的语言虚拟机也层出不穷，虚拟机技术的专业图书却不多见。本书是作者多年研究与实践的心得与沉淀，值得一读。”

—— **胡子昂** 华为硅谷基础软件实验室主任、华为Fellow

“关于虚拟机架构的设计，本书不但阐明了是什么，而且讨论了为什么。作者从架构师的角度高屋建瓴地分析了技术选择背后的来龙去脉，相信读者一定会受益匪浅。”

—— **慎熙鹏** 北卡罗来纳州立大学计算机系教授

“虚拟机技术在编程语言、编译器、计算机架构、Web系统等领域均有深远的影响。作者具备这些领域的工作经历，本书的写作也扎根于其在虚拟机研究及开发过程中的一手资料。本书通过循序渐进的方式，将虚拟机技术融入丰富的示例与代码中，使得读者能够同时在理论和实践的层面充分理解虚拟机技术的精髓。”

—— **朱子青** 英伟达高级软件架构师

“本书作者来自工业界，有着丰富的工程实践经验。书中同时包含了学术界在虚拟机相关领域的研究成果。其中重要的技术讨论，既有代码示例，又有理论分析，能帮助读者透彻地理解虚拟机技术的要点、难点。”

—— **林子超** 美国高通研发总监

CRC Press
Taylor & Francis Group

TURING
www.ituring.com.cn
来社区 邻偶子

图灵社区：iTuring.cn
热线：(010)51095183转600
分类建议 计算机 / 程序设计 / Java
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-52728-8
ISBN 978-7-115-52728-8
定价：129.00元

TURING 图灵程序设计丛书

Advanced Design and Implementation of Virtual Machines

虚拟机设计与实现
以JVM为例

李晓峰（Xiao-Feng Li）◎著
单业◎译

人民邮电出版社
北 京

图书在版编目（CIP）数据

虚拟机设计与实现 : 以JVM为例 / 李晓峰著 ; 单业译. -- 北京 : 人民邮电出版社, 2020.1
（图灵程序设计丛书）
ISBN 978-7-115-52728-8

Ⅰ. ①虚… Ⅱ. ①李… ②单… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2019)第270342号

内 容 提 要

本书从一位虚拟机（VM）架构师的角度，以易于理解、层层深入的方式介绍了各种主题和算法，尤其是不同 VM 通用的主要技术。这些算法用图示充分解释，用便于理解的代码片段实现，使得这些抽象概念对系统软件工程师而言具像化并可编程。书中还包括一些同类文献中较少涉及的主题，例如运行时辅助、栈展开和本地接口。本书集理论性与实践性于一身，不仅结合了高层设计功能与底层实现，而且还结合了高级主题与商业解决方案，是 VM 设计和工程实践方面的理想参考读物。

本书适合对虚拟机感兴趣的软件开发者及研究人员阅读。

◆ 著　　李晓峰
译　　单　业
责任编辑　温　雪
责任印制　周昇亮

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市祥达印刷包装有限公司印刷

◆ 开本：800×1000　1/16
印张：23.75
字数：560千字　　2020年1月第1版
印数：1－4 000册　　2020年1月河北第1次印刷
著作权合同登记号　图字：01-2019-7211号

定价：129.00元

读者服务热线：(010)51095183转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

版权声明

中文版序

亲爱的读者：

你现在看到的这本书讨论的是语言虚拟机的设计与实现技术。虚拟机这个概念的内涵在过去十几年里逐渐发生了变化，从特指实现某个语言的运行时技术，扩大到系统仿真的各类技术，甚至容器技术。其中一个原因是，语言虚拟机已经被广泛应用在各个领域，并与各种系统紧密结合，从而不再新奇和稀有。如果现在的程序员在平时的工作中接触不到某种语言的虚拟机，那倒是比较少见了。尽管如此，语言虚拟机的设计技术对大多数人来说，仍然深不可测。虽然市场上已经有了一些相关图书，网络上的各种文章也层出不穷，但现有的资料要么偏于理论和概念，要么仅限于讨论语言虚拟机的某个局部，很难让学习者对语言虚拟机技术有全面而系统的理解。学习者往往还要通过阅读某个虚拟机的源码来学习相关技术，但因为并不了解其设计决策的来龙去脉，所以在想改进一个虚拟机或者开发自己的虚拟机项目时，仍会感到力不从心。

笔者多年来一直从事虚拟机技术的研究与开发，同时也大量涉猎操作系统、编译器和语言设计的相关技术。由本人及所带领团队开发的各类虚拟机软件和技术已经被应用在数以亿计的服务器、个人计算机、手机和其他智能设备中，一些创新研究成果以学术论文的形式发表在著名的国际会议上。在这个过程中，笔者对虚拟机设计的特点有了较深的理解，经常受邀给一些研究和开发机构做报告或培训，并先后为北京大学和中国科技大学计算机系的研究生讲授过短期课程。笔者的一些博客文章，一度在 Google 搜索相关技术时排名居前，给业界同行带来了有益的启发。在和同行的交流中，笔者也往往会得到一些精彩的反馈，加深了自己对该领域的理解，并在项目中得以实践。这样，经过反复的“研发实践-提炼总结-交流反馈-吸收改进”，笔者对虚拟机的技术逐步形成了一套较为系统的设计方法论。这些心得体会，在现有的技术资料中很难找到较完整的表述，因此一些同事和朋友还是会经常向我咨询相关问题，并建议我能整理成文字，给相关开发人员提供帮助，并填补虚拟机技术文献的空白。这就是我写这本书的初衷。

把多年的知识积累系统地写出来，并做到深入浅出，这不是一件容易的事。由于笔者所从事的工作的特点，业余时间不是在加班，就是在学习充电，并且经常出差，这样的情况下要保障每天写书一小时，而且内容前后保持连贯、环环相扣，是需要极大的毅力的。从动笔开始，前前后后写了近四年才搁笔。当然，写作期间也有不少乐趣，特别是每当自己精心绘制出一幅图，将语

言难以表达的意思较好地表达出来时，总会端详片刻，体验一下表达的乐趣。但总体来说，遗憾比快乐更多。这么说的一个重要原因是，尽管用时颇长，但本来计划好的内容还是有很大一部分因种种原因未能写成，只能留待未来弥补了。好在成书的部分已经相对完整，基本概括了典型的语言虚拟机的所有核心部件的设计。对学习者来说，如果掌握了这些内容，那么对虚拟机技术的理解就已经相当深入了，足以支撑其进入任意一种语言虚拟机的设计开发。不过需要提醒的是，本书对读者的系统软件基础有一定的要求，即了解基本的编译器和操作系统技术。在遇到生疏的术语时，请查阅相关资料。

本书有两个特点：一是比较系统全面，很多内容在其他资料中难以见到，比如异常处理的实现等；二是内容尽量做到深入浅出，既有理论阐述，又有代码示例。笔者尽力将典型虚拟机设计的方方面面都有机地串联在一起，并解释它们的来龙去脉，而不是简单地进行技术堆砌。对每个主题的内容，本书也尽量按照平常的思维模式，循序渐进地讲解。不过请读者注意，本书的重点是虚拟机中特有的技术，如果某种技术在编译器或操作系统中已经有充分的讨论，本书则会略去。读者如果对那些技术感兴趣，应能找到比本书更好的资料。另外，本书内容虽然是虚拟机通用技术为主，但为了避免泛泛而谈，主要以 JVM 设计为例，并兼顾其他虚拟机。还有，近几年来，语言虚拟机设计技术又有了新的发展，比如异步编程，但本书没有涵盖，请读者海涵。读者若有任何对本书内容的批评和建议，请与我联系，不胜荣幸！

最后，感谢我的老朋友、著名编译专家周志德（Fred Chow）欣然为本书作序，同时感谢译者单业和图灵编辑团队对本书的出版所做的重要贡献。在我与图灵出版团队的交往中，我深切感受到他们对图书质量的严格要求和对读者的诚挚与负责。希望本书能为图灵的精品书单增添光彩。

李晓峰

2019 年 11 月 1 日于美国硅谷

序

传统上，一个计算系统建立在支持操作系统的硬件平台之上，应用程序在操作系统中以硬件执行的机器指令形式运行。随着编程语言的发展，程序员已经开始感恩动态或托管语言在提升编程效率方面的优势。通过提供更好的安全性和软件可移植性，虚拟机（VM）现已成为更适合软件程序的执行环境。目前最先进的 VM 设计代表了过去几十年的研究及开发活动的成果。这些工作大体上致力于改进 VM 在功能和性能方面的实现。随着时间积累，当前产品级质量的 VM 已经变得非常复杂，并且通常要耗费巨大的精力来实现。即使对经验丰富的软件工程师来说，理解 VM 如何工作也成为了一项挑战。

从李晓峰任职于 Intel 公司开始，我和他相识已经超过 15 年了。他带领开发了 Intel 平台上的多个编译器和托管运行时系统。李晓峰是 Apache Harmony 项目中 JVM 的主要贡献者。他还在 Perl、Ruby、JavaScript 以及 Android 相关的 VM 设计方面做了大量的研究工作。李晓峰在 VM 工程和产品方面拥有丰富的经验，这使他对 VM 设计的不同领域都有实质性的见解，而这又让他拥有独特的视角，能够在本书中探讨与 VM 相关的诸多话题。

作为研究者和工程师，李晓峰从系统架构师的独特视角撰写了本书。他强调工程实践中应考虑的因素，并关注各种组件之间的交互及合作方式，及其给接口层设计带来的影响。其他关于 VM 的书中通常很少讨论这些细节。本书还提供了详细的图示和代码片段，清晰地阐释了书中的观点。关于 VM 设计与实现的高级主题，本书已经成为了我不可或缺的参考书。我向系统软件开发者，尤其是托管运行时系统的开发者，强烈推荐本书，因为本书能够清晰地解答他们在探索 VM 相关话题时所产生的疑问。

李晓峰完成了这部关于 VM 的专著，在 VM 设计和工程实践方面做出了巨大的贡献。

周志德（Fred Chow）

Futurewei Technologies 首席科学家

前　言

本书的主题是为 Java 和 JavaScript 这样的编程语言设计和实现虚拟机。

虚拟机（VM），也称为托管运行时系统，或者托管运行环境。它还有一个更通俗的名字——沙盒技术。自几十年前被发明以来，VM 一直吸引着软件研究人员和开发者的兴趣和关注。这是因为 VM 为软件带来了重要属性，比如安全性、高生产率和可移植性。在当今的计算系统中，VM 已经变得无处不在——从物联网（Internet of Things，IoT）节点到移动电话、个人计算机，再到云平台。

我的许多从事软件相关工作的朋友都乐于学习 VM 的知识。他们经常向我咨询他们在日常工作中使用的 VM 的相关问题。我发现这些问题很多都和 VM 中的常用技术有关，但他们很难从现有的图书和其他文档中找到有用信息。究其原因，这些资料要么主要关注规范和原则，要么是研究论文，过于学术化。当我的朋友 Ruijun He——Taylor & Francis 出版集团的编辑——邀请我撰写 VM 主题的书时，我觉得专门为有兴趣探索 VM 工作机制的软件开发者写本书是一个好主意。

我曾应邀在高校和公司做关于 VM 的讲座。这些讲座笔记逐渐累积，似乎可以成书了。我曾以为把它们整理成书应该很容易，但实际情况是，当我试图用系统化和条理化的方式组织这些材料，并辅之以深刻的理论支持与实用的代码片段时，才发现这是一项艰巨的挑战。

我尽力让本书有别于类似主题的已有文献。因此，我从 VM 架构师的角度来组织内容，尝试用整体方法来设计 VM。本书大部分内容在 2014 年年底前完成。从那之后，我一直关注着业界 VM 的新发展。不过本书并不打算面面俱到地讨论各种 VM 实现，而是专注于不同 VM 通用的那些最重要的技术。我非常乐意根据读者的反馈来改进和调整本书的内容。若对本书有任何评论，可以反馈给本书出版社，或发邮件给本书作者：li@xiaofeng.info。[①]

李晓峰

① 本书中文版勘误可以到图灵社区页面（http://www.ituring.com.cn/book/2600）提交。——编者注

关于本书

随着运行时引擎的地位越发重要，并在日常计算系统中变得无处不在，软件社区对现代虚拟机设计和实现的详尽解释产生了强烈的需求，包括 Java 虚拟机（JVM）、JavaScript 引擎和 Android 执行引擎。社区希望看到的不止是形式化的算法描述，还有实用的代码片段。社区希望理解的不止是研究课题，还有工程上的解决方案。本书以独特的论述方式，结合了高层设计功能与底层实现，同时也结合了高级主题与商业解决方案，希望以此来满足上述需求。

本书采用整体方法来介绍 VM 体系结构的设计，将内容组织为一致的框架，以易于理解、层层深入的方式介绍了各种主题和算法。它关注 VM 设计的关键方面，而这些在其他书中通常是被忽略的，比如运行时辅助、栈展开和本地接口。这些算法用图示充分展示，用便于理解的代码片段实现，使得这些抽象概念对系统软件工程师而言具像化并可编程。

目　录

第一部分　虚拟机基础

第二部分　虚拟机设计

第四部分　垃圾回收优化

第五部分 线程交互优化

第一部分

虚拟机基础

第 1 章　虚拟机简介

本章介绍虚拟机的概念。虚拟机已经以各种形式发展了数十年。1995 年，Sun 公司发布了 Java 编程语言以及相应的 Java 虚拟机（JVM），由此虚拟机开始为普通开发者所知。

1.1　虚拟机类型

虚拟机是一个计算系统。计算系统的最终目标是执行预先编程的逻辑。这些逻辑可以以非常底层的形式表达，包含了实际计算机的所有细节；也可以通过脚本或标记语言在很高的层次上表达。从这个角度看，根据抽象层次和模拟范围，可以把虚拟机大致分为如下 4 种类型。

- 类型 1，**完整指令集架构（ISA）虚拟机**，提供完整的计算机系统 ISA 模拟或虚拟化。客户操作系统和应用程序在这个虚拟机上可以像在实际计算机上那样运行（例如 VirtualBox、QEMU 和 XEN）。
- 类型 2，**应用程序二进制接口（ABI）虚拟机**，提供客户进程 ABI 模拟。针对这套 ABI 的应用程序，可以在这个进程中与其他本地 ABI 应用程序进程并肩运行［例如安腾处理器上的 Intel IA-32 Execution Layer（IA-32 执行层）、Transmeta 提供 X86 模拟的 Code Morphing，以及 Apple 的用于模拟 PowerPC 的 Rosetta 转译层］。
- 类型 3，**虚拟 ISA 虚拟机**，提供一个运行时引擎，以便虚拟 ISA 编码的应用程序在其上执行。虚拟 ISA 通常定义了一套高层的、规模有限的 ISA 语义，所以不需要虚拟机模拟完整的计算机系统［例如 Sun Microsystems 的 JVM、Microsoft 的通用语言运行时（Common Language Runtime），以及 Parrot Foundation 的 ParrotVM］。
- 类型 4，**语言虚拟机**，提供一个运行时引擎来执行以客户语言编写的程序。程序通常以客户语言的源码形式提供给虚拟机，并没有预先完全编译为机器码。运行时引擎需要解释或翻译程序，还要实现一些像内存管理这样的由语言抽象出的功能（例如 Basic、Lisp、Tcl 和 Ruby 的运行时引擎）。

上述几种虚拟机类型的边界并不是完全清晰的。许多虚拟机的设计跨越了边界。例如，某个语言虚拟机可以把程序编译为某种虚拟 ISA，然后在这个虚拟 ISA 的虚拟机上执行，这样就利用了虚拟 ISA 虚拟机技术。即便如此，为虚拟机分类以便于社区交流仍是有意义的。

前两种虚拟机类型分别是 ISA 模拟和 ABI 模拟。它们的目标是在主机上运行为非本地 ISA 或 ABI 开发的现有客户操作系统或客户应用程序。它们有时也被称为模拟器。

另外两种虚拟机是语言运行时引擎，其目标是执行以虚拟 ISA 或客户语言形式编写的逻辑。在某些上下文中，虚拟 ISA 也被看作一种特殊的语言；除此之外，这两种语言运行时引擎并没有本质上的区别。

本书的主题是语言运行时引擎。除非特别指出，否则后面章节中的关键词“虚拟机”只表示语言运行时引擎，并且“运行时引擎”和“虚拟机”可以互换使用。使用“运行时引擎”这种表达是因为虚拟机提供的服务多数只在运行时可用。相比之下，在“编译器 + 操作系统”这套传统配置中，应用程序由编译器静态编译之后发布。出于同样的原因，有些人使用“运行时系统”来指代运行时可用的、让软件能够执行的服务。

1.2 为什么需要虚拟机

虚拟机是现代编程不可或缺的技术。虚拟机改善了（计算机）安全性、（编程）效率和（应用程序）可移植性。

对安全语言来说，虚拟机是必要的。这里**安全语言**（safe language）是一个非常宽泛的概念，主要指提供了内存安全、运算安全和控制安全特性的语言。通过安全语言，能尽早安全地捕获程序 bug 或运行错误。

(1) 内存安全确保内存中某种类型的数据总是遵循对这种类型的限制。例如，指针变量永远不会持有非法指针，数组元素永远不会越界。
(2) 运算安全确保对某种类型数据的运算总是遵循对这种类型的限制。例如，指针变量不允许进行任意算术运算。
(3) 控制安全确保代码执行流既不会卡住也不会跑飞（例如跳转到恶意代码段）。控制安全也可以被看作某种特殊的运算安全。

几乎所有的现代编程语言都是安全语言，例如 Java、C#、Java 字节码、Microsoft 中间语言（Microsoft Intermediate Language）以及 JavaScript，尽管它们各自的安全程度有所不同。

要支持安全语言，虚拟机往往是必要的，因为安全语言本身并不能满足所有的安全需求。例如，程序不应该直接分配一块没有类型的内存。它需要虚拟机来为它提供带类型的内存，例如某种类型的对象。如果没有虚拟机，那么安全语言必须引入非安全的操作支持，比如 Rust 语言。

虚拟机为安全语言的代码和数据提供“托管”。因此，这些代码和数据有时被称为“托管代码”和“托管数据”。相应地，虚拟机有时也被称为“托管运行时”“托管系统”或者“托管执行环境”。

因为用安全语言编写的程序更加难以被恶意代码攻击，所以在安全沙盒技术中有时会利用虚拟机。Google Chrome NaCl 技术就是一个例子。