面向二进制移植的虚拟化技术*

黄聪会, 陈 靖, 罗 樵, 张 黎, 郭一辰
(空军工程大学 信息与导航学院, 西安 710077)

摘 要: 从 ISA 和 ABI 两种不同层次出发,探讨了当前二进制移植存在的问题,分析了其对应解决方法的优劣,明确了虚拟化技术是实现二进制移植的重要手段。研究了支持 ISA 或 ABI 间二进制移植中的三种虚拟化方法,即解释和二进制翻译、资源虚拟化、内核虚拟化。提出了一种结合动态二进制优化技术的高效进程虚拟机 LongWin,其支持 Windows 应用程序在 Linux 操作系统上运行,实验结果表明,其性能与 Wine 相比提高了 6% ~10%。

关键词: 软件移植; 进程虚拟机; 虚拟化; 指令集体系结构; 应用程序二进制接口
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2012)11-4185-04
doi:10.3969/j.issn.1001-3695.2012.11.047

Virtualization technology for binary migration

HUANG Cong-hui, CHEN Jing, LUO Qiao, ZHANG Li, GUO Yi-chen
(*Information & Navigation College*, *Air Force Engineering University*, *Xi'an* 710077, *China*)

Abstract: Starting from two different levels of ISA and ABI, discussed the problems existed in binary migration, this paper analyzed the pros and cons of the corresponding solutions, and verfied the conclusion that virtualization technology was an important means to achieve binary migration. It studied the three virtualization methods, namely interpretation and binary translation, resource virtualization, kernel virtualization, which could support the realization of binary migration between ISA or ABI. It proposed the efficient process virtual machine named LongWin which combined the dynamic binary optimization technology. The proposed process virtual machine can help the Linux operating system to run Windows applications, and the experiment result indicates LongWin has a 6% to 12% performance improvement relative to Wine.

Key words: software migration; process virtual machine; virtualization; instruction set architecture; application binary interface

丰富的应用程序支持是桌面操作系统兴旺发达的基础。目前在桌面操作系统领域,Windows 系列操作系统大约占据 82.47% 的市场份额,而 Linux 操作系统仅占 2.41%[1]。因此,绝大部分软件供应商所提供软件的第一个版本通常针对 Windows 开发,随后才会开发 Linux 版本,甚至直接弃 Linux 于不顾。在此形势下,Windows 之外的操作系统要扩大其市场份额非常难。而二进制移植能使应用程序在异构操作系统上兼容运行,是实现 Linux 操作系统发展壮大的关键。

应用程序以二进制形式发布后,就与特定的指令集和操作系统绑定了,而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。这是应用程序无法在多个异构操作系统或硬件平台上运行的原因。因此消除应用程序与操作系统、操作系统与底层硬件之间的这种绑定关系是实现二进制移植的关键。

虚拟化是指创建某种事物虚拟版本的技术、方法和过程[2],是近年来计算机体系结构领域的研究热点。其主要目的是简化 IT 基础设施和资源管理的方式。例如,当计算机的处理器、存储器或输入/输出设备被虚拟化后,计算机可表现为一个或多个虚拟计算机的结合,通过虚拟计算机接口,可见的资源都被映射到真实计算机的接口和资源上。虚拟化技术具有创建事物虚拟版本,消除计算机体系结构间的耦合关系的能力,为二进制移植指明了方向。

1 二进制移植面临的挑战

采用分层思想的计算机体系结构存在两个重要的接口——ISA(instruction set archtecture,指令集体系结构)和 ABI(application binary interface,应用程序二进制接口),如图 1 所示。ISA 由用户 ISA 和系统 ISA 组成,ABI 主要由系统调用接口和用户 ISA 组成[3]。当应用程序以二进制形式发布时,它就已经跟特定的 ISA 和 ABI 进行了绑定。通常不同的 ISA 和 ABI 之间相互不兼容,由此导致应用程序无法在不同的 ISA 和 ABI 上运行。下面分别研究不同 ISA 和 ABI 间进行二进制移植面临的问题。

1.1 ISA 间二进制移植

指令集体系结构 ISA 是与程序设计有关的计算机架构的一部分,包括本地数据类型、指令、寄存器、地址模式、内存架构、中断和意外处理和外部 I/O。一个 ISA 包括一系列 opcodes(机器语言)的一个规格,本地命令由一个特定的 CPU 设计来实现[4]。

当前主要存在两种指令集体系结构,即精简指令集(reduced instruction set computer,RISC)和复杂指令集(complex in-

收稿日期: 2012-04-20; **修回日期**: 2012-05-26 **基金项目**: 国家自然科学基金资助项目(61172083)

作者简介:黄聪会(1985-),男,湖南衡阳人,博士研究生,主要研究方向为软件移植、虚拟化技术(huangdoubleten@126.com);陈靖(1964-),女,教授,博士,主要研究方向为网格计算、虚拟化技术、无线自组织网络;罗樵(1978-),男,博士,主要研究方向为分布式系统理论与技术;张黎(1987-),男,博士研究生,主要研究方向为无线自组织网络;郭一辰(1988-),男,硕士研究生,主要研究方向为分布式系统理论与技术.

struction set computer,CISC)。复杂指令集指令数目多而复杂,每条指令字长不相等,可执行若干低阶操作,诸如从内存读取、储存,计算操作可全部集于单一指令之中。常用的X86架构微处理器即采用复杂指令集。而精简指令集对指令数目和寻址方式都作了精简,实现容易,指令并行执进程度更好,编译器的效率更高。目前常见的精简指令集微处理器包括DEC Alpha、ARC、ARM、AVR、MIPS、PA-RISC、Power Architecture(包括PowerPC、PowerXCell)和SPARC等。

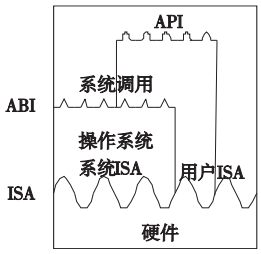

图1 计算机体系结构

尽管复杂指令集存在很多缺陷,但采用复杂指令集的X86微处理器在商业上获得了巨大的成功,而更先进的采用精简指令集的微处理器在市场上的表现不尽人意。究其原因,处理器的成功不仅取决于自身技术的先进程度,还取决于支持该处理器的应用程序的多寡。

目前存在三种方法实现ISA间二进制移植[5]:a)在新处理器上专门提供执行源ISA的硬件,如Intel的Itanium处理器存在专门执行X86指令的硬件;b)将源程序重新编译到目标指令集;c)使用虚拟化技术,解释或二进制翻译源ISA。方法a)增加了新处理器的硬件复杂度,影响了代码的执行效率,失去了开发新处理器的意义。方法b)中编译后的程序可达到很高的效率,但并不是所有应用程序都能找到源代码及其共享库的源代码。而方法c)则没有前两种方法的缺点,通过虚拟化技术,可实现ISA间二进制兼容,迅速扩展硬件/软件的适用范围,因此虚拟化技术是实现ISA间二进制移植的首选方法。

1.2 ABI间二进制移植

应用程序二进制接口ABI描述了应用程序与操作系统之间、应用程序与其程序库之间或者应用的组成部分之间的低层接口[6]。ABI掩盖了各种细节:如数据类型、大小和对齐;调用约定(控制着函数的参数如何传送以及如何接收返回值);系统调用的编码和应用程序如何向操作系统进行系统调用;目标文件的二进制格式、程序库等。

当前在桌面操作系统领域同样面临着处理器发展的所遇到的难题。Windows应用软件层出不穷,不断巩固着Windows操作系统的市场统治地位。而Linux操作系统却因应用软件缺乏以及强大的Windows用户使用惯性而发展缓慢。ABI间二进制移植使应用软件不再与操作系统绑定,将促进操作系统之间的公平竞争,为Linux等操作系统的发展创建有利环境。

目前存在两种方法实现ABI间二进制移植:a)制定二进制兼容标准,如Intel二进制兼容标准(iBCS),允许支持它的操作系统上的程序不经修改在其他支持此ABI的操作系统上运行;b)使用虚拟化技术,如资源虚拟化或内核虚拟化技术,虚拟出源ABI。方法a)虽能实现ABI间二进制兼容,但标准制定时间偏长,且仅限于同种或同类操作系统之间,很难在异构操作系统间制定ABI的二进制兼容标准。例如,很难说服微软参与制定Windows和Linux之间的ABI二进制兼容标准,因为这将损害微软的商业利益。而方法b)不存在方法a)的缺点,且不受异构操作系统的限制,是实现ABI间二进制移植的首选方法。

2 虚拟化技术在二进制移植中的应用

实现二进制移植的关键是解决二进制程序依赖的ISA或ABI与目标环境提供的ISA或ABI不一致的问题。虚拟化技术通过在目标环境中创建二进制程序所依赖的ISA或ABI的虚拟版本,可完美地实现二进制移植。下面探讨各种虚拟化技术在二进制移植中的应用。

2.1 解释和二进制翻译

通常ISA间二进制移植由指令集仿真实现。指令集仿真允许支持源ISA的二进制程序运行在执行目标指令集的主机上。可使用多种方法实现指令集仿真,不同的方法需要不同数量的计算资源,并提供不同的性能和可移植特性。其中最基本的方法是解释和二进制翻译[7]。

解释是指取一条源指令,对其进行分析,执行需要的操作,再取下一条源指令的这样一个循环过程,所有工作都是由软件完成。而二进制翻译将源指令块翻译为目标指令块,并且将翻译后的代码保存起来以便反复使用,分摊了取指和分析的代价[8]。与解释相比,二进制翻译有较大的初始翻译代价,但是执行代价较小。两者的选择取决于所移植软件期望执行源代码块的时间。也可改进解释或二进制翻译方法用于指令集仿真。例如,线索化解释消除了取指解释再循环,并且可通过将源指令预译码为更加高效的可解释的中间形式进一步提高效率。

2.2 资源虚拟化

资源虚拟化包括处理器、存储器和输入输出设备的虚拟化[9]。通过资源的虚拟化,可在一个硬件平台上虚拟出多个硬件平台,然后在多个虚拟的硬件平台上安装多个不同的操作系统,从而实现ABI间的二进制移植。例如,采用Linux版本的宿主虚拟机软件VMWare,可在Linux操作系统上虚拟一个硬件平台,然后再其上安装Windows操作系统,从而实现Windows应用程序在Linux系统上的运行,即解决了ABI间二进制移植的问题。

处理器虚拟化的关键在于客户机指令(包括系统级和用户级的指令)的执行,而执行有两种实现方式:a)通过仿真,当宿主机的ISA和客户机的ISA不同时,仿真是实现处理器虚拟化的唯一手段;b)在宿主计算机上使用直接本地执行技术[10]。这种方法只能在宿主机的ISA和客户机的ISA相同时使用,并且还要满足一些约束条件。存储器虚拟化[11]将应用程序编程人员可见的存储器逻辑视图和操作系统管理的真实存储器资源区别开。实现输入/输出子系统的虚拟化比较困难,因为每个I/O设备都有自己的特征和独特的控制方式[12],并且I/O设备的种类和数量也越来越多。对给定I/O设备类型的虚拟化策略包括建立设备的虚拟版本和虚拟该设备上的I/O活动。

2.3 内核虚拟化

内核虚拟化技术可实现异构操作系统的ABI间二进制移植。内核虚拟化的核心是在目标操作系统上创建一个虚拟内核,该虚拟内核可为源应用程序提供各种服务,以支持源应用程序向目标平台的二进制移植。根据虚拟内核在目标操作系

统中所处的位置,可分为用户空间内核虚拟化和内核空间内核虚拟化。

用户空间内核虚拟化即在目标操作系统的用户空间创建源操作系统的虚拟内核,以支持源应用程序在目标操作系统上的运行。此种虚拟化技术的优点是可移植性强、实现相对容易、适应性强,缺点是效率不高,其典型代表是 Wine[13]。Wine 支持 Windows 应用程序在 Linux 上运行,且具有很好的适应性。例如,Wine 可在各种 Linux 的发行版、各种 UNIX 以及 Mac OS 上运行。

内核空间内核虚拟化即在目标操作系统的内核空间创建源操作系统的虚拟内核,以支持源应用程序在目标操作系统上的运行。此类虚拟化技术的优点是效率高,但存在开发难度高、可移植性弱的缺点,其典型代表是 Linux 兼容内核 Longene[14]。Linux 兼容内核 Longene 是一个二进制兼容 Windows 和 Linux 应用软件和设备驱动程序的计算机操作系统内核。它在 Linux 内核的基础上利用 Linux 内核材料构建微软 Windows 内核功能模块,从而扩充 Linux 内核的支持能力,使之同时支持 Linux 和 Windows 的应用程序和设备驱动。Longene 严重依赖于 Linux 内核,因此很难将其移植到 UNIX 或 Mac OS 操作系统上。

3　一种新的高效进程虚拟机 LongWin

对异构操作系统的 ABI 间进行二进制移植,尤其是将 Windows 应用程序二进制移植到 Linux 操作系统,基于资源虚拟化的系统虚拟机解决方案虽然能实现二进制移植,但存在消耗大量硬件资源、须先启动虚拟机等缺点,且该解决方案对增强 Linux 的竞争力无大的帮助。内核虚拟化可使 Linux 操作系统直接运行 Windows 应用程序,但内核空间内核虚拟化与用户空间内核虚拟化相比,还不成熟,且开发难度大,因此用户空间内核虚拟化是当前 Windows 应用程序运行于 Linux 的理想解决方案。

当前最成熟的用户空间内核虚拟化解决方案即开源软件 Wine。然而 Wine 并不完善,其最大的缺点是效率不高。提高 Wine 的效率通常是通过改善 Wine 的设计,但该方法对提升 Wine 整体效率的作用不大,需要长期艰苦的工作才能取得良好的效果。因此本文提出采用动态二进制优化技术[15]间接地全面提升 Wine 的执行效率。结合开源软件 Wine 和开源的运行时代码操纵工具 DynamoRIO[16],设计出一种新的高效进程虚拟机 LongWin。

3.1　LongWin 体系结构

LongWin 主要由加载器、分发器、基本块构建器、代码缓存、OS 调用仿真管理器五部分组成,其体系结构如图 2 所示。

a)加载器主要作用是识别、加载 Windows 二进制映像,并为 Windows 二进制映像的运行做好初始化准备。

b)分发器的作用主要是对 Windows 二进制映像的指令进行解码,为基本块构建器、代码缓存服务。

c)基本块构建器的作用主要是在分发器解码指令的基础上构建基本块,并指导分发器将形成的基本块复制到代码缓存中。

d)代码缓存的作用主要是存储基本块、踪迹,执行基本块中指令;当基本块缺失时,通知分发器构建缺失的基本块。

e)OS 调用仿真管理器模拟仿真 Windows 系统调用,这是 Windows 应用程序在 Linux 上运行的基础。OS 调用仿真管理器还负责管理所创建的 Windows 进程。

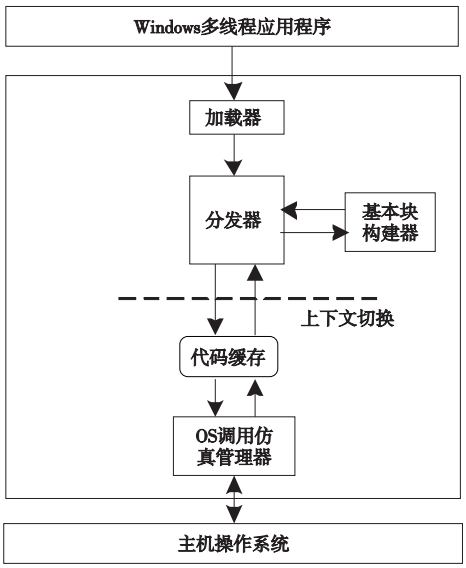

图2　LongWin的体系结构

3.2　LongWin 的实现

LongWin 的实现是在结合 Wine 的前提下改造 Linux 版的 DynamoRIO,使其能在 Linux 系统上优化运行 Windows 应用程序的过程。

LongWin 的五个组成部分中,分发器、基本块构建器、代码缓存是 DynamoRIO 的重要组成部分,OS 调用仿真管理器则相当于 Wine 的 WineServer 组件,因此它们的实现可借鉴 DynamoRIO 和 Wine 的源代码。而加载器涉及 Wine 模拟 Windows API 功能和 DynamoRIO 代码优化功能的结合,是 LongWin 的重要组件,需要做大量的修改工作。LongWin 加载器的执行流程如图 3 所示。

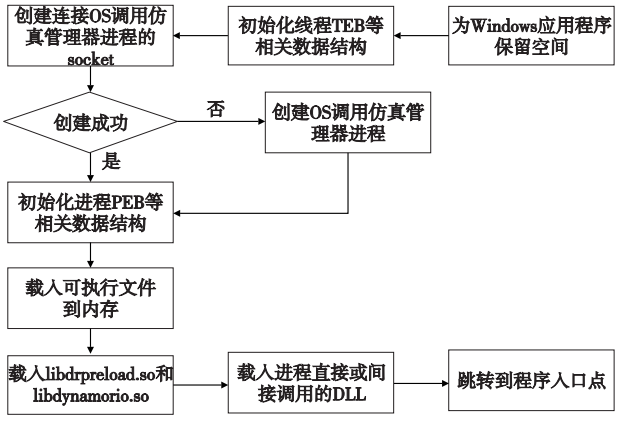

图3　LongWin加载器执行流程

LongWin 加载器执行流程中 libdrpreload. so 和 libdynamorio. so 的载入代表 LongWin 的分发器、基本块构建器和代码缓存组件装入内存并准备就绪,是 Wine 和 DynamoRIO 的功能结合的关键点。当 LongWin 加载器执行流程跳转到程序入口点时,LongWin 的分发器组件即开始工作。分发器读取 Windows 程序的内存映像,并调用基本块构建器创建基本块,将创建的基本块复制到代码缓存中。随后分发器通过上下文切换将控制权交给代码缓存,即开始执行代码缓存中的指令。当代码缓存遇到进行 Windows 系统调用的指令时,则与 OS 调用仿真管理器通信,并得到 Windows 系统调用的结果。

3.3 LongWin 性能评估

LongWin 提供了动态操纵和修改 Windows 应用程序指令的客户端编程接口。通过这些编程接口,可编写采用各种动态二进制优化技术的客户端,而 LongWin 相对于 Wine 的高性能则取决于这些客户端。因此,为比较 LongWin 和 Wine 之间的性能差别,分别编写采用冗余加载移除(redundant load removal)、间接分支分发(indirect branch dispatch)、定制踪迹(custom traces)三种动态二进制优化技术以及综合这三种技术的客户端[17]。

本次性能测试采用的计算机硬件配置为:Pentium® Dual-Core CPU E5200 2.5 GHz、2 GB 内存、160 GB 硬盘。测试方法为:在 Windows XP SP3 操作系统上,采用 VC6 编译器和 Intel 的 Fortran 编译器编译 SPEC CPU 2000 测试基准程序,记录它们在 Windows XP SP3 下的执行时间,作为本次性能测试的基准。然后在 Ubuntu 10.10 操作系统上分别记录在 Wine 1.3.17 支持下以及加载了不同客户端的 LongWin 支持下,测试基准程序的运行时间。实验结果如图 4 和 5 所示。

图 4 和 5 中的规范化执行时间是指 Ubuntu 10.10 下各项目的运行时间与 Windows XP SP3 下运行时间的比值。图 4 和 5 中第一条柱 Wine 代表在 Wine 的支持下基准程序的执行时间;第二条柱 base 代表在不采用任何客户端的 LongWin 的支持下基准程序的规范化执行时间,随后的四条柱分别为在采用冗余加载移除、间接分支分发、定制踪迹动态二进制优化技术以及综合这三种技术的客户端的 LongWin 的支持下基准程序的规范化执行时间。

从图 4 和 5 中可知,未加载优化客户端的 LongWin 与 Wine 相比性能会下降,而加载优化客户端的 LongWin 与 Wine 相比性能会提升。加载了综合三种优化技术的客户端的 LongWin 与 Wine 相比,运行 SPEC CPU 2000 整点基准程序的性能平均提高 6%,运行浮点基准程序的性能平均提高 10%。

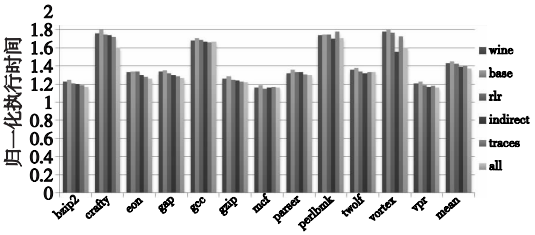

图4 整点基准程序规范化执行时间

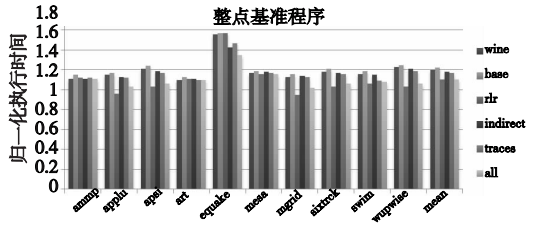

图5 浮点基准程序规范化执行时间

4 结束语

通过本文的总结分析和编程测试,得到如下结论:

a)不论是 ISA 还是 ABI 间二进制移植,虚拟化技术都是首选方法。

b)支持 ISA 或 ABI 间二进制移植的虚拟化方法有解释和二进制翻译、资源虚拟化、内核虚拟化。内核虚拟化中的用户空间内核虚拟化是当前 Windows 应用程序运行于 Linux 的理想解决方案。

c)提出了一种结合动态二进制优化技术的高效进程虚拟机 LongWin。该虚拟机支持 Windows 应用程序在 Linux 操作系统上运行,实验结果表明其性能与 Wine 相比提高了 6% ~ 10%。由于 LongWin 性能的提升取决于其加载的各种采用动态优化技术的客户端,因此其性能还可能进一步提高。

参考文献:

[1] WIKIPEDIA. Usage share of operating systems[EB/OL]. (2011-08-22)[2011-08-26]. http://en.wikipedia.org/wiki/Usage_share_of_operating_systems.

[2] 王兴波. 有关虚拟机及虚拟化技术的几点诠注[J]. 微型机与应用,2009,28(7):76-78.

[3] SMITH E, NAIR R. The architecture of virtual machines[J]. Computer,2005,38(5):32-38.

[4] VIKRAM A, CHRIS L, MICHAEL B, *et al*. LLVA: a low-level virtual instruction set architecture[C]//Proc of the 36th Annual ACM/IEEE International Symposium on Microarchitecture. Washington DC: IEEE Computer Society,2003:205-216.

[5] ERIK A, DAVID K, YARON S. Welcome to the opportunities of binary translation[J]. Computer,2000,33(3):40-45.

[6] WIKIPEDIA. Application binary interface[EB/OL]. (2011-06-24)[2011-08-29]. http://en.wikipedia.org/wiki/ Application_Binary_Interface.

[7] CIFUENTES C, MALHOTRA V M. Binary translation: static, dynamic, retargetable[C]//Proc of International Conference on Software Maintenance. Washington DC: IEEE Computer Society, 1996: 340-349.

[8] CHEN Wei, WANG Zhi-ying, CHEN Dan. An emulator for executing IA-32 applications on ARM-based systems[J]. Journal of Computers,2010,5(7):1133-1141.

[9] 陈小军,张璟. 虚拟化技术及其在制造业信息化中的应用综述[J]. 计算机工程与应用,2010,46(23):25-30.

[10] SUSANTA N, TZI-CKER C. A survey on virtualization technologies[R]. USA: ECSL,2005.

[11] 汤泉,李小勇. 文件支持的 Xen 存储虚拟化研究[J]. 计算机工程与应用,2009,45(16):77-79.

[12] HIMANSHU R, KARSTEN S. High performance and scalable I/O virtualization via self-virtualized devices[C]//Proc of the 16th International Symposium on High Performance Distributed Computing. New York: ACM,2007:179-188.

[13] 龚亚东, 张辉, 叶勇. Wine 内核及实现 Microsoft Windows 消息机理分析[J]. 计算机应用,2005,25(Z1):431-433.

[14] 王亚军,刘金刚. Windows 程序运行于 Linux 系统的技术[J]. 计算机应用,2009,29(8):2128-2131.

[15] DEREK B, EVELYN D. Design and implementation of a dynamic optimization framework for Windows[C]//Proc of the 4th ACM Workshop on Feedback-Directed and Dynamic Optimization. 2000.

[16] DEREK B. Efficient, transparent, and comprehensive runtime code manipulation[D]. Massachusetts: Massachusetts Institute of Technology, 2004.

[17] DEREK B, TIMOTHY G, SAMAN A. An infrastructure for adaptive dynamic optimization[C]//Proc of International Symposium on Code Generation and Optimization. Washington DC: IEEE Computer Society,2003:265-275.