第 42 卷 第 10 期 计 算 机 工 程 2016 年 10 月
Vol. 42 No. 10 Computer Engineering October 2016

·体系结构与软件技术· 文章编号: 1000-3428(2016)10-0107-06 文献标志码: A 中图分类号: TP311.54

基于龙芯的 Windows 软件二进制移植研究

苏 煜,张 涛,王金双,赵 敏

(解放军理工大学 指挥信息系统学院,南京 210007)

摘 要: 由于龙芯采用无内部互锁流水级的微处理器(MIPS)指令集且主要支持 Linux 操作系统,但在该平台上的应用软件相对匮乏,且很多传统 Windows 软件无法在该平台上直接运行。针对该问题,利用 Wine 仿真 Windows 系统调用,结合 QEMU 用户模式实现 x86 指令向 MIPS 指令动态翻译的方法,将运行在 Intel + Windows 上的应用软件向龙芯 + Linux 平台二进制移植。实验结果表明,相比于 Bochs + Wine 方案和 QEMU 全系统仿真方案,该方法虚拟 CPU 运算速率为 Bochs + Wine 方案的 8.04 倍和 QEMU 全系统仿真方案的 1.44 倍,I/O 读写速率为 Bochs + Wine 方案的 2.44 倍和 QEMU 全系统仿真方案的 7.38 倍,运行效率明显提高。

关键词: QEMU 虚拟机; Wine 仿真环境; 龙芯; 指令集翻译; 二进制移植

中文引用格式: 苏 煜,张 涛,王金双,等. 基于龙芯的 Windows 软件二进制移植研究[J]. 计算机工程,2016,42(10):107-112.

英文引用格式: Su Yu,Zhang Tao,Wang Jinshuang,et al. Research on Windows Software Binary Migration Based on Loongson[J]. Computer Engineering,2016,42(10):107-112.

Research on Windows Software Binary Migration Based on Loongson

SU Yu,ZHANG Tao,WANG Jinshuang,ZHAO Min

(College of Command Information Systems,PLA University of Science and Technology,Nanjing 210007,China)

【Abstract】 Since Loongson adopts Microprocessor without Interlocked Piped Stages(MIPS) instruction set and main supports the Linux operating system,application software on Loongson is still relatively scarce. Many traditional Windows software cannot run directly on the platform. To solve this problem,this paper proposes a method to simulate Windows system call by utilizing Wine emulator and uses QEMU user mode realize the dynamic translation from x86 instructions to MIPS instructions. It achieves software binary migration from Intel + Windows architecture to Loongson + Linux architecture. Experimental results show that this scheme has higher efficiency compared with Bochs + Wine and QEMU full system schemes. The rate of CPU calculation with this method is 8.04 times of Bochs + Wine scheme and 1.44 times of QEMU full system scheme. I/O calculation rate with this scheme is 2.44 times of Bochs + Wine scheme and 7.38 times of QEMU full system scheme.

【Key words】 QEMU virtual machine; Wine simulation environment; Loongson; instruction set translation; binary migration

DOI: 10.3969/j.issn.1000-3428.2016.10.019

1 概述

核心软硬件的自主化对于国家信息安全至关重要,而处理器正是计算机系统中最基础的硬件。目前,龙芯是国产处理器的典型代表,它采用无内部互锁流水级的微处理器(Microprocessor without Interlocked Piped Stages,MIPS)指令集,运行类 Linux 操作系统。由于该平台上应用软件还相对匮乏,且 Windows 程序无法直接在该平台上运行,这对传统应用的移植以及该平台的推广与应用造成一定的影响。

本文通过对主流开源虚拟机进行分析,结合龙芯 MIPS 架构特点,实现 QEMU + Wine 方案在龙芯平台运行 Windows 应用程序,以解决 QEMU + Wine架构分页机制与龙芯平台分页机制不一致的问题。

基金项目:国家部委基金资助项目。

作者简介:苏 煜(1991—),男,硕士,主研方向为系统软件安全;张 涛,教授、博士;王金双,副教授、博士;赵 敏,讲师、博士。

收稿日期:2015-10-12 修回日期:2015-11-28 E-mail:barton_sue@sohu.com

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

2 相关技术

当前,目标指令集支持 MIPS 架构的虚拟机主要有 Bochs[1] 和 QEMU[2]。Bochs 采用指令集解释技术,QEMU 采用指令集翻译技术。指令集解释通过取一条源指令,对其进行分析并执行需要的操作,再取下一条源指令的循环过程完成不同指令集的二进制转换,所有工作都是由软件完成。二进制翻译试图分摊取指和分析的代价,它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[3]。相对于解释技术,指令集翻译技术具有更高的执行效率[4]。QEMU 分为系统模式和用户模式,系统模式模拟整个系统,包括 CPU 和外部设备;用户模式仿真 CPU 和系统调用[5]。相比而言,采取 QEMU 用户模式提供应用虚拟化环境的轻量级方案能很好地解决全系统模拟带来的性能损失,增强了虚拟化系统的可用性。

Wine 是一个能够在多种依从可移植的操作系统接口上运行 Windows 应用的兼容层,它将 Windows API 调用翻译成为动态的 POSIX 调用,免除了系统性能和其他一些行为的内存占用,提高 Windows 应用的运行效率[6]。目前,上海交通大学的胡嘉杰等实现了在高级精简指令集机器(Advanced Reduced instruction set computing Machine,ARM)环境下 Windows 软件的零修改跨平台移植,它通过 Wine 仿真 Windows 系统调用,同时借助 QEMU 实现 x86 指令集到 ARM 指令集的翻译执行,并通过实验证明了 ARM 平台下基于 QEMU + Wine 模式实现的虚拟化方案拥有更高的效率[7]。文献[8]将 Wine 直接编译在龙芯平台上完成 Windows 系统调用模拟,同时利用开源虚拟机 Bochs 实现一些应用程序指令和 Windows 原生动态库的指令集转化。本文采用 Wine 和 QEMU 作为核心软件实现 Windows 应用在龙芯平台上的二进制移植。

3 系统设计与实现

3.1 系统架构

在龙芯 3B 处理器、Linux 操作系统上运行基于 x86 指令的 Windows 应用程序是一个既跨指令集又跨操作系统平台的实现,需要进行 2 次模拟执行来完成。整体设计方案如图 1 所示,系统主要分为2 个模块,即 Wine 提供的基于 x86 的 Windows 应用环境仿真模块和 QEMU 提供的基于 MIPS 的 x86 指令翻译模块。Windows 应用程序通过 Wine 加载器加载到内存地址空间,同时 Wine 提供 Wineserver 仿真 Windows 进程管理模块进行进程间通信、同步以及进程和线程的管理[9]。当应用程序调用 Windows 应用程序接口(Application Programming Interface,API)时,由 Wine 提供的动态链接库(Dynamic Link Library,DLL)仿真实现 Windows 库和系统调用的模拟。由此,所有的二进制程序都是基于 x86 环境运行。最后通过 QEMU 用户模式仿真 x86 指令集 CPU,运用指令集翻译的方式将 x86 的可执行代码翻译成 MIPS 指令,并在龙芯平台上运行。同时,实现 Linux 系统调用跨硬件平台转化,将 x86 架构下 Linux 系统调用直接转化为 MIPS 架构下 Linux 系统调用,减小翻译系统调用产生的性能损耗。

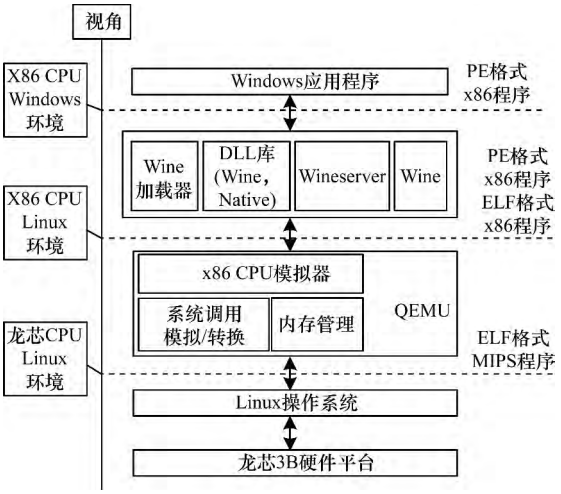


图 1 QEMU + Wine 虚拟化系统架构

3.2 Windows 应用环境仿真模块

Wine 实现在相同指令集平台下对 Windows 系统库的兼容,提供了 Windows 程序执行所需的二进制运行环境,以达到在类 Unix 平台上运行 Windows 程序的目的。Wine 的系统架构如图 2 所示。

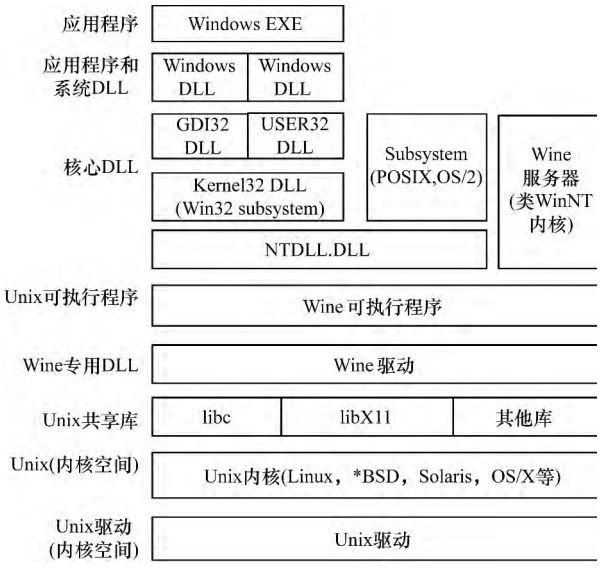


图 2 Wine 系统架构

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

为了能够运行上层的 Windows 应用程序和库,Wine 自实现了 3 个重要的 DLL 库——USER32,KERNEL32,GDI32 以及 NTDLL 的大部分功能[9]。这些动态链接库是 Windows 系统内核的主要实现,是上层库和应用的基础。Wineserver 维护一个守护进程实现 Windows 句柄管理和进程间通信、同步以及进程和线程的管理功能[10]。Wine 驱动实现了将 Windows 驱动 API 与 Linux 驱动接口对应连接,将其转化为通过 Linux 驱动接口对设备的调用运行[11]。

3.3　x86 指令翻译模块

QEMU 分为用户模式和系统模式,系统模式仿真一个完整的虚拟硬件平台,用户通过在虚拟硬件平台上安装操作系统运行应用程序。与之相比,由于 QEMU 用户模式不需要模拟外部设备,因此会有更高的效率。用户模式主要包括 CPU 模拟和系统调用模拟两部分。

3.3.1　CPU 模拟

QEMU 的 CPU 模拟通过指令集翻译方式实现。其模拟的系统架构叫目标架构,运行 QEMU 的系统架构叫主机架构。指令集翻译过程如图 3 所示,首先,将目标架构 x86 指令翻译为中间代码 TCG(Tiny Code Generator)[12],再根据主机架构龙芯 MIPS 指令平台将中间代码 TCG 翻译成平台可以执行的 MIPS 指令。为了提高指令翻译的效率,QEMU 设计翻译后指令缓存结构 TB(Translation Block)缓存翻译后指令,当指令被重复执行时可直接通过查找指令对应的 TB 结构运行,减少指令重复翻译消耗的时间[13]。

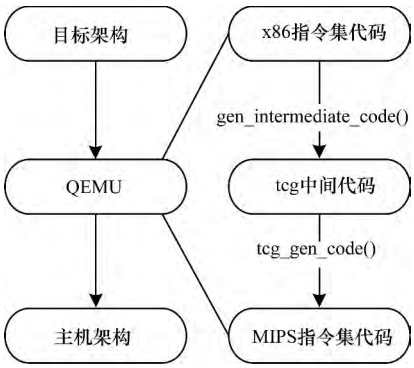


图 3　指令集翻译流程

3.3.2　QEMU 对系统调用的模拟

QEMU 用户模式在处理系统调用时,不像系统模式需要翻译系统调用处理程序,而是通过模拟 CPU 捕获中断,根据中断号解释中断。基于 x86 的 Linux 系统调用通过 QEMU 解释 80 h 中断,并根据获取的系统调用号确定系统调用类型,将所需的系统调用通过 MIPS 主机操作系统调用或直接通过 QEMU 编码进行实现。总体执行框图如图 4 所示,Windows 应用程序通过 Wine 模拟器重写 Windows 系统调用,将 Windows 系统调用转化为 Linux 系统调用,再通过 QEMU 用户模式将基于 x86 的 Linux 系统调用转化为基于 MIPS 的 Linux 系统调用。以进程创建为例,Windows 应用程序通过 Windows 句柄提出进程创建请求,该请求通过 socket 传送到 Wine 模拟器服务程序,由 Wine 根据传过来的句柄进行解释并转化为 Linux 系统调用 fork 实现,QEMU 模拟 CPU 捕获 Wine 程序的系统调用中断指令,根据系统调用号确定进程创建系统调用,将进程创建系统调用通过 MIPS 主机操作系统调用 fork 实现。通过该机制能将大部分的系统调用转化为通过主机系统调用实现,避免对系统调用进行指令集翻译而产生性能损耗。

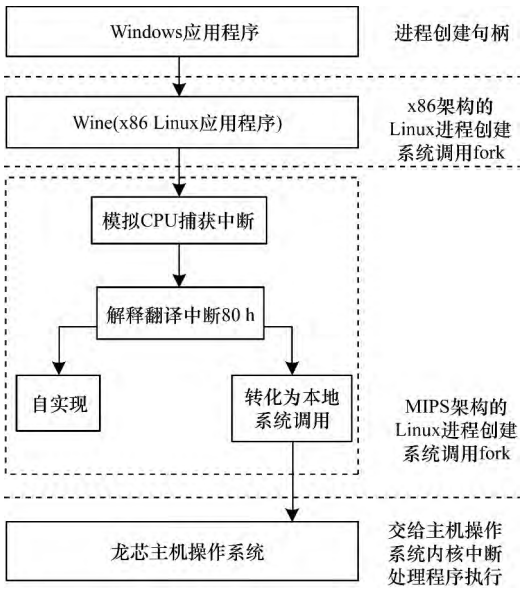


图 4　基于 QEMU 的用户模式仿真系统调用

4　关键技术

4.1　内存布局

龙芯平台采用内核空间 2 GB 加用户空间 2 GB 的内存布局,而 x86 采用内核空间 1 GB 加用户空间 3 GB 的内存布局。根据图 5 分析由于 Wine 默认将 Wine 加载器布局在 2 GB ~3 GB 之间,使得龙芯平台对 Wine 加载器的布局产生冲突。通过修改 Wine 代码重新布局 Wine 程序和 Windows 应用程序以及动态库的加载空间,使其用 0 GB ~2 GB 内存空间,使得内存布局符合龙芯支持布局。

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

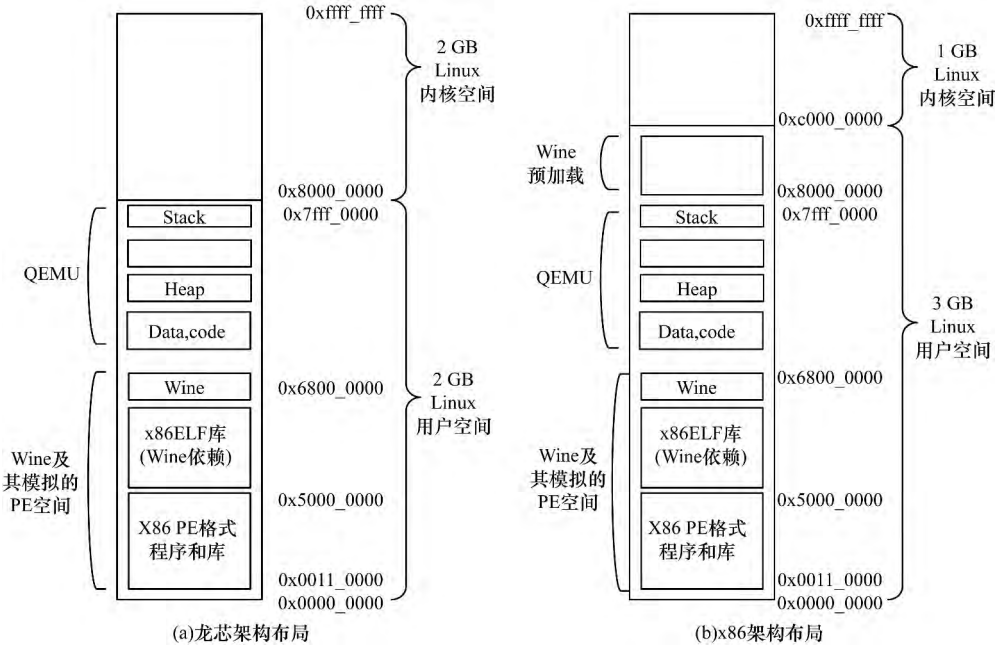


图 5 x86 架构和龙芯架构的地址空间布局对比

4.2 页处理单元

龙芯处理器支持分页大小，最小为 16 KB，QEMU 默认支持页大小是 4 KB。因此，在程序运行过程中会产生页属性错误、客户机申请内存失败以及申请的空间没有按要求初始化等问题。在内存占用率高的情况下，错误会频繁发生。从 QEMU 页访问权限管理入手解决上述问题，主要有以下 3 个方面:

(1) 对于共享映射，QEMU 根据是否是匿名映射做了不同的策略。如果是匿名映射，QEMU 内部对其开始地址和内存申请长度进行控制，使其返回 16 KB 对齐地址，同时将内存申请长度增大到 16 KB 地址对齐; 如果为文件映射，且有写属性，QEMU 直接返回内存，申请失败; 如果为文件映射，且无写属性，QEMU 拷贝此段地址内容到对应地址。

(2) 对于私有映射，如果是匿名映射，且不指定开始地址的方式，QEMU 利用自身内存管理表进行管理。此时页大小为 4 KB，支持 r/w/x 权限，不支持 share/private 权限; 如果是文件映射，例如 mmap 的情况，QEMU 将开始地址和申请内存长度处理后直接传入主机系统申请，对 2 个不同属性 4 KB 页在同一个 16 KB 页中这种情况，将其属性叠加。

(3) 忽略内存权限改小的操作。

4.3 多线程处理

Wine 中的程序通过 createThread 创建新的线程，由 Wineserver 进行线程调度，并发起线程的执行。新创建的线程从程序入口开始执行，线程执行遇到 x86 指令，需要 QEMU 来模拟执行。

由于每个线程都要有一个独立的模拟执行部件，而且在执行的时机上是物理并发的，因此需要 QEMU 创建多个模拟 CPU，每个模拟 CPU 用于执行一个线程中的 x86 指令。

根据上述原因，通过分配多个 QEMU 的 CPU 对象，产生多个虚拟 CPU。每个线程拥有一个 QEMU 的 CPU，所有 CPU 跟随 Wine 的调度并行执行。

5 系统评估

测试主要从功能和性能两方面进行评估。测试环境基于龙芯 3B 八核心 1 GHz 处理器，8 GB 内存，采用内核版本 2.6.32 的 Debian 7 操作系统。

5.1 功能测试

功能测试的目的是检测系统对不同类别软件支持是否完善。主要通过测试 Delphi，VC ++，MFC 等开发的简单单线程程序和多线程程序，以及复杂的管理信息系统(Management Information System，MIS) 程序来评估系统功能的完整性。实验基于 QEMU 1.7.2 版本和 Wine 1.6.2 版本进行系统设计。实验结果如表 1 所示，其中，√表示支持。通过实验验证，该架构支持多种语言开发的 Windows 应用软件，对数据库、多线程应用支持良好。

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

表 1　功能测试实验结果

应用程序名称	程序描述	原运行环境	现运行环境	安装	运行
Notepad. exe	Windows 系统自带文本编辑工具	Win7 / Vista / Win2003 / WinXP	QEMU + Wine	√	√
Spider. exe	用 VC + + 编写的蜘蛛纸牌游戏	Win7 / Vista / Win2003 / WinXP	QEMU + Wine	√	√
Firefox. exe	火狐浏览器	Win7 / Vista / Win2003 / WinXP	QEMU + Wine	√	√
Potplayer. exe	视频播放软件	Win7 / Vista / Win2003 / WinXP	QEMU + Wine	√	√
Foobar2000. exe	音频播放软件	Win7 / Vista / Win2003 / WinXP	QEMU + Wine	√	√
学籍信息管理系统. exe	依赖 Oracle 数据库开发的学员信息管理系统	Win7 / Vista / Win2003 / WinXP	QEMU + Wine	√	√

5.2　性能测试

将 QEMU + Wine 与 Bochs + Wine 和 QEMU 全系统模拟环境对比，本文采用了 superPI [14]，SQLIO [15] 2 个基准测试工具，从虚拟 CPU 效率和 I/O 读写效率 2 个方面进行系统性能评估，测试实例如表 2 所示。

表 2　测试实例

项目	实例 1	实例 2	实例 3
机器平台	龙芯 3B，主频 1 GHz	龙芯 3B，主频 1 GHz	龙芯 3B，主频 1 GHz
操作系统	内核版本 2.6.32 的 Debian 7 操作系统	内核版本 2.6.32 的 Debian 7 操作系统	内核版本 2.6.32 的 Debian 7 操作系统
系统架构	QEMU + Wine	Bochs + Wine	QEMU 全系统，客户操作系统为 Windows XP
测试方案 1	运行 superPI. exe 应用测试 CPU 性能	运行 superPI. exe 应用测试 CPU 性能	运行 superPI. exe 应用测试 CPU 性能
测试方案 2	运行 SQLIO 测试文件读写性能	运行 SQLIO 测试文件读写性能	运行 SQLIO 测试文件读写性能

superPI 是一个单线程基准测试工具，它通过利用高斯-勒让德算法计算 PI 的位数，由此比较 CPU 运算效率。实验数据如表 3 所示。

表 3　superPI 测试方案运算时间

计算小数点后位数	Bochs + Wine	QEMU 全系统	QEMU + Wine
16K	5 min23.365 s	1min0.593 s	40.382 s
32K	11 min37.255 s	1 min59.109 s	1min 26.404 s

SQLIO 用于测试文件读写性能，实验使用测试文件大小为 1 GB、测试持续时间为 300 s、I/O 单位请求数据量为 2 KB、I/O 线程数为 1 的测试方案，测试结果如表 4 所示。

表 4　SQLIO 测试方案的 I/O 读写速率　($MB \cdot s^{-1}$)

测试项目	Bochs + Wine	QEMU 全系统	QEMU + Wine
连续读取	1.43	0.49	3.69
连续写入	1.55	0.54	3.69
随机读取	1.45	0.49	3.68
随机写入	1.57	0.46	3.55

通过比较发现，QEMU + Wine 架构在 CPU 运算和 I/O 读写方面拥有更高的效率，虚拟 CPU 速率是 Bochs + Wine 架构的 8.04 倍，是 QEMU 全系统模式的 1.44 倍; I/O 读写速率是 Bochs + Wine 架构的 2.44 倍，是 QEMU 全系统模式的 7.38 倍。运用该方案解决自主化平台上的跨平台移植具有更高的实用价值。由于系统采用 QEMU 与 Wine 低耦合的实现方式提升了平台的兼容性，但 Wine 本身对 Windows 软件支持仍有欠缺，后期根据开源社区对 Wine 进行同步实时更新，完善对所支持应用程序的补充。

6　结束语

本文结合自主化平台需求，采用 QEMU + Wine轻量级方案，有效地解决了 Windows 应用软件在自主化平台的零修改移植问题。运用应用虚拟化技术，通过 QEMU 用户模式实现 CPU 模拟和系统调用仿真，并与 Windows API 仿真器 Wine 相结合实现应用二进制移植，提高了移植应用软件的运行速度。下一步将在该方案基础上结合动态信息流追踪和指令随机化技术，研究实现一个精确的主动防御系统，为自主化软硬件平台提供安全性分析与保障。

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

参考文献

[1] Lawton K P. Bochs: A Portable PC Emulator for Unix/x [J]. Linux Journal, 1996, 1996(29): 7.

[2] Bellard F. QEMU: A Fast and Portable Dynamic Translator [C]//Proceedings of Annual Conference on USENIX Annual Technical Conference. Berkeley, USA: USENIX Association, 2005: 41-46.

[3] Smith J, Nair R. Virtual Machines: Versatile Platforms for Systems and Processes [M]. Amsterdam, Holland: Elsevier, 2005.

[4] 黄聪会,陈 静,罗 樵,等. 面向二进制移植的虚拟化技术 [J]. 计算机应用研究, 2012, 29(11): 4185-4188.

[5] 陈 乔,蒋烈辉,董卫宇,等. 基于动态二进制翻译技术的仿真器研究 [J]. 计算机工程, 2011, 37(20): 277-279.

[6] Wine User's Guide [EB/OL]. (2015-05-25). https://www.winehq.org/docs/wineusr-guide/index.

[7] 胡嘉杰,蒋乐天. Android 系统中 Win32 环境的高性能虚拟 [J]. 计算机应用, 2015, 35(3): 886-890.

[8] 苏 煜,张 涛,靳国杰,等. 基于龙芯的 Windows 软件运行环境的设计与实现 [J]. 江苏科技大学学报: 自然科学版, 2014(4): 387-391.

[9] Chiueh T C. A Survey on Virtualization Technologies, Technology Report: 179 [R]. New York, USA: State University of New York at Stony Brook, 2005.

[10] Wine Developer's Guide [EB/OL]. (2015-05-25). https://www.winehq.org/docs/winedev-guide/index.

[11] 钟柏松. Wineserver 机制在 Linux 内核中实现的研究 [D]. 南昌: 南昌大学, 2012.

[12] Henderson A, Prakash A, Yan L K, et al. Make it Work, Make it Right, Make it Fast: Building a Platform-neutral Whole-system Dynamic Binary Analysis Platform [C]//Proceedings of International Symposium on Software Testing and Analysis. New York, USA: ACM Press, 2014: 248-258.

[13] Gligor M, Fournel N, Pétrot F. Using Binary Translation in Event Driven Simulation for Fast and Flexible MPSoC Simulation [C]//Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis. New York, USA: ACM Press, 2009: 71-80.

[14] Huang Conghui, Chen Jing, Zhang Li, et al. Performance Evaluation of Virtualization Technologies for Windows Programs Running on Linux Operating System [C]//Proceedings of Conference on Network Computing and Information Security. Berlin, Germany: Springer, 2012: 759-766.

[15] Corporation C. SQLIO Disk Subsystem Benchmark Tool [EB/OL]. (2015-07-03). http://www.microsoft.com/en-us/download/details.aspx?id=20163.

编辑 索书志

(上接第 106 页)

参考文献

[1] Blunden B. Memory Management-algorithms and Implementation in C/C++ [M]. [S. 1.]: Wordware Publishing, Inc., 2003.

[2] 李法龙,吴 刚,陈章龙. 位图在嵌入式系统内存管理中的应用 [J]. 计算机工程与设计, 2005, 26(4): 1020-1021.

[3] Daniel S G. A Class of Dynamic Memory Allocation Algorithm [J]. Communications of the ACM, 1973, 16(10): 615-618.

[4] Margolin B H, Parmelee R P, Schatzoff M. Analysis of Free Storage Algorithms [J]. IBM Systems Journal, 1971, 10(4): 283-304.

[5] 严蔚敏,吴伟民. 数据结构(C 语言版) [M]. 北京: 清华大学出版社, 1992.

[6] 李小群,孙玉芳. Linux 内存管理机制的分析与研究 [J]. 计算机科学, 2002, 29(4): 18-20.

[7] 曹国钧. 计算机内存管理及其优化技术 [M]. 成都: 电子科技大学出版社, 1998.

[8] Ellen S G. 内存管理深入剖析 [M]. 李谊瑞,周克昌,译. 北京: 电子工业出版社, 1995.

[9] 沈 勇,王志平,庞丽萍. 对伙伴算法内存管理的讨论 [J]. 计算机与数字工程, 2004, 32(3): 40-43.

[10] Puttkamer E V. A Simple Hardware Buddy System Memory Allocator [J]. IEEE Transactions on Computers, 1975, 24(10): 953-957.

[11] 何 伟,何建中. FreeRTOS 内存管理机制的分析和改进 [J]. 计算机工程, 2010, 36(10): 67-69.

[12] 姜力波. 基于 Linux 内存管理的分析和研究 [D]. 合肥: 中国电子科技大学, 2011.

[13] 余广桥,李玉涛,王江汉. 嵌入式终端平台内存管理算法比较 [J]. 单片机与嵌入式系统应用, 2012, 12(5): 64-65.

[14] 侯 捷. STL 源码剖析 [M]. 武汉: 华中科技大学出版社, 2002.

[15] 陈力军,魏永军,迮 超. 一种内存管理系统及其分配方法: CN01139150.2 [P]. 2001-07-21.

[16] Dong Wensheng, Shen Chunfeng. Memory Management System with Controlled Volume and High Performance [J]. Control Engineering of China, 2013, 20(S0): 69-71.

[17] 王 铮,李志军. 一个嵌入式系统的自适应动态内存分配方法 [J]. 计算机技术与发展, 2007, 17(3): 48-50.

编辑 索书志

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net