

Pcanel/V2——基于 Intel VT-x 的 VMM 架构

陈文智 姚 远 杨建华 何钦铭

(浙江大学计算机科学与技术学院系统结构研究所 杭州 310027)

摘 要 个人计算机硬件性能的迅速增强使得通过虚拟化技术建立多个相互隔离的计算域成为未来个人计算机的一种重要发展趋势. 为避免传统的 IA-32 架构在软件虚拟化领域所面临的 VMM 的设计和实现的困难, 作者设计并实现了基于 Intel VT-x 技术的 VMM 架构——Pcanel/V2. 该架构利用最新的硬件虚拟化技术, 通过配置出一个可控制的虚拟运行环境, 可以直接虚拟运行多个不修改源代码的客户操作系统. 在允许客户操作系统正常运行的同时控制它们对各种硬件资源的访问, 并能对客户操作系统运行过程中出现的各种情况进行相应的处理. Pcanel/V2 实现了 Linux 和 Vxworks 的同时运行, 相应的数据测试表明 Pcanel/V2 架构在简化了 VMM 设计复杂度的同时总体运行效率比软件虚拟化技术提高了约 10%.

关键词 虚拟化; VT-x 技术; 虚拟机监控软件; 客户操作系统; 系统隔离

中图法分类号 TP316 **DOI 号**: 10.3724/SP.J.1016.2009.01311

Pcanel/V2: A VMM Architecture Based on Intel VT-x

CHEN Wen-Zhi YAO Yuan YANG Jian-Hua HE Qin-Ming

(College of Computer Science and Engineering, Zhejiang University, Hangzhou 310027)

Abstract With the rapidly development of personal computer hardware, to construct multiple isolated computing domains through virtualization technology has already become a future direction of PC. To avoid the difficulties caused by implementing VMM on traditional IA-32 architecture through software virtualization technology, the authors designed and implemented a VMM based on Intel VT-x technology — Pcanel/V2. Pcanel/V2 utilizes the latest hardware virtualization technology to configure a controllable virtual execution environment and run multiple operating systems directly without any modification of source code. While the accesses to hardware resources by the guest operating system are monitored, various sensitive situations which occur in the guest operating system can also be handled correspondingly by Pcanel/V2. Concurrent execution of Linux and VxWorks on Pcanel/V2 has been realized and corresponding evaluation results show that Pcanel/V2 architecture simplifies the complexity of the design process while increasing the overall performance by approximately 10% compared to software virtual technology.

Keywords virtualization; VT-x technology; virtual machine monitor; guest operating system; system isolation

收稿日期: 2007-04-17; 最终修改稿收到日期: 2009-06-11. 本课题得到国家“九七三”重点基础研究发展规划项目(2007CB310906)、基础科研项目(A1420080190)、基金项目(9140A15040309JW0402, 9140A16070409JW0403, 9140A06050609JW0402, 2008ZH76007)资助.
陈文智, 男, 1969 年生, 博士, 副教授, 主要研究方向为嵌入式实时系统、分布式计算、虚拟化技术与可信计算等. E-mail: chenwz@zju.edu.cn. 姚 远, 男, 1982 年生, 硕士研究生, 主要研究方向为虚拟化技术、计算机系统结构和嵌入式系统. 杨建华, 男, 1973 年生, 博士, 讲师, 主要研究方向为嵌入式系统、计算机系统结构. 何钦铭, 男, 1965 年生, 博士, 教授, 主要研究领域为计算机软件工程、虚拟化技术等.

1 引言

虚拟化技术可以提高计算机物理资源的利用率和共享率^[1],并且为提高系统的可靠性、操作性、安全性和实时服务质量提供了新的途径.

当前主流的虚拟化技术是软件虚拟化技术,主要存在 3 种类型:全虚拟化、半虚拟化和预虚拟化.全虚拟化以 VMware^[2] 为代表,模拟出一整套硬件设备,其上运行的 Guest OS 不必修改源代码,但是运行效率较低.半虚拟化以 Xen^[3] 和 Denali^[4] 为代表,让 Guest OS 知道它们正工作在一个虚拟的环境中,通过修改它们以使之工作得更好,因此操作系统需针对这种方法进行修改和调整,但是效率较高.预虚拟化已在 L4^[5] 中运用,性能仅仅略低于半虚拟化技术,并且也不需要修改 Guest OS 源代码.

因为原有 IA-32 架构的设计原因,在其上实现软件虚拟化存在很多困难和问题,这也大大增加了 VMM(Virtual Machine Monitor,虚拟机监控软件)的设计难度.其中,最根本的原因就是特权级混淆.为了从根本上解决传统软件虚拟化技术的缺点,两大处理器厂商都推出了基于硬件的虚拟化技术——Intel 的 VT 技术^[6] 和 AMD 的 Pacifica 技术.其中,Intel VT 技术的一个重要的设计目标就是消除半虚拟化和二进制转化技术,简化 VMM 的实现,可以支持更大范围的、不需修改的操作系统,并且保持高性能^[7].Xen 进行了支持 VT 的尝试^[8],性能获得一定提升.KVM(Kernel-based Virtualization Machine,基于内核的虚拟机)^①也支持 VT 技术,它作为 Linux 内核的一个模块存在,充分利用了系统资源,但是它通过 QEMU 系统模拟的接口让 Guest OS 在 Linux 的用户态中运行,其性能比采用软件全虚拟化技术的 QEMU 要好,但是有些性能不如基于软件半虚拟化技术的 Xen^[9].构件化的多内核架构 Ppanel^[10-11],可以支持多个不同类型的操作系统在其上的并行运行.针对硬件虚拟化的优点和今后多核的发展趋势,我们在 Ppanel 基础上设计并开发了基于 VT-x 技术的 IA-32 VMM——Ppanel/V2.

本文第 2 节先介绍 Intel VT-x 对虚拟化技术的硬件支持;第 3 节和第 4 节介绍 Ppanel/V2 整体框架,包括总体架构和运行流程.第 5 节~第 9 节重点介绍 Ppanel/V2 设计的各个重要方面,它们是自主设计的,是不同于其它系统的关键所在;第 10 节展现 Ppanel/V2 原型系统并做了相关性能测试比

较;最后一节讨论今后工作要点.

2 Intel VT-x 技术简介

具有 VT 技术的处理器有两种运行模式:VMX 模式和非 VMX 模式.非 VMX 模式和不具备 VT 技术的处理器运行方式完全相同;VMX 模式就是 VT 技术实际发生功效的模式.在 VMX 模式中,又分为两种新的处理器操作模式:VMX root 操作模式和 VMX non-root 操作模式.VMX root 操作模式是提供给 VMM 使用的,它的功能与没有 VT 技术的 IA-32 非常相似(主要区别就是可以使用 VMX 指令).VMX non-root 操作模式提供了一个选择性的 IA-32 环境,该环境被 VMM 控制,被设计用来支持虚拟机.两种操作模式都支持所有的 4 个特权级,允许客户软件在其所期望的特权级上运行,也提供了 VMM 使用多个特权级的权利.

VT 技术定义了两种新的转换:从 VMX root 操作模式到 VMX non-root 操作模式的转换叫作 VM entry,从 VMX non-root 操作模式到 VMX root 操作模式的转换叫做 VM exit.这两种转换被叫做虚拟机控制结构(virtual-machine control structure VMCS)的一个新数据结构控制^[12].VMCS 包括一组客户机状态(guest-state area)和一组主机状态(host-state area),两种状态都对应着处理器不同组件的值.VM entry 转换把客户机状态装入到处理器的各个寄存器等相关状态中.VM exit 转换把处理器各个寄存器状态保存到客户机状态,然后将主机状态装入到处理器中.

VT 技术允许客户软件运行在其所期望的特权级上.客户软件的限制,不是来自特权级,而是来自它运行在 VMX non-root 操作模式上.这个本质使得基于 VT 技术的 VMM 可以解决上述提到的传统软件虚拟化中遇到的困难.

3 总体设计

Ppanel/V2 的设计思想有以下 3 条原则:

- (1) 在虚拟机中运行的 Guest OS 不需要改变源代码.
- (2) 充分利用 VT 技术提供的各项功能,减少设

① KVM white papers. http://www.qumranet.com/wp/kvm_wp.pdf

计复杂度.

(3) 尽可能利用 Guest OS 而不是 Pcanel /V2 来处理操作系统运行过程中遇到的各种情况.

Pcanel/V2 的总体架构如图 1 所示. Pcanel/V2 运行在 VMX root 操作模式的特权级 0 上, Guest OS(例如 Linux、Vxworks、Windows、Solaris 等操作系统)运行在 VMX non-root 操作模式的其所设计希望运行的特权级上(一般为特权级 0 和 3). Guest OS 所能直接使用的硬件就是 CPU, 所有其它的硬件设备都是 Pcanel/V2 的设备虚拟模块为其

虚拟出来的. 在 VMX non-root 操作模式中, 处理器操作有个很大的改变. 最重要的改变就是很多指令和事件会导致 VM exit. 一些指令(例如 INVD)会无条件地导致 VM exit, 因此永远不能在 VMX non-root 操作模式下执行. 其它指令(例如 INVLPG)和所有的事件可以通过配置 VMCS 中的 VM-execution control 域来实现有条件地导致 VM exit. 此外, 所有的硬件中断都是 Pcanel/V2 接收并处理. Pcanel/V2 实现了 Linux 和 Vxworks 的同时虚拟.

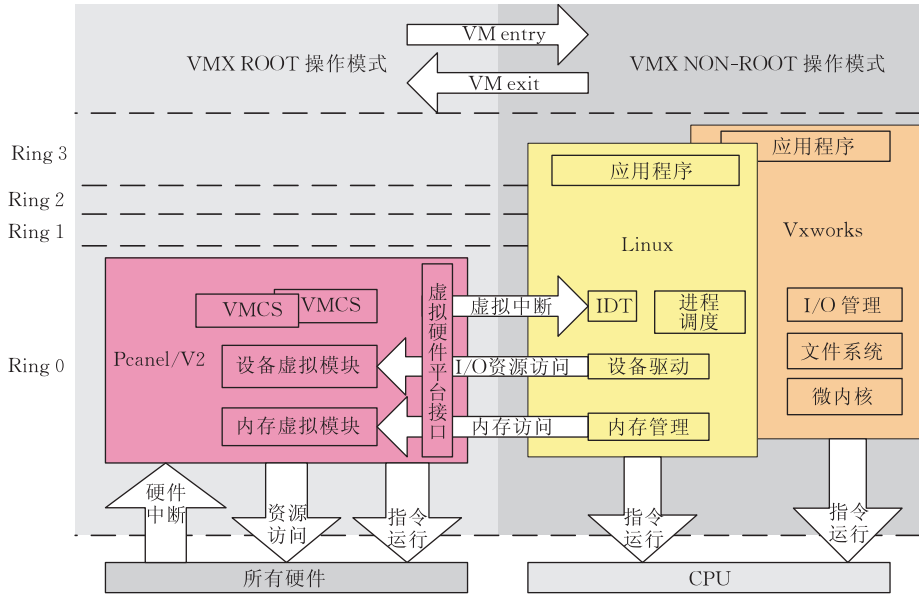


图 1 Pcanel/V2 总体架构图

需要强调的是, 整个架构设计中, 有些是 VT 技术所规定的(例如 Pcanel/V2 运行在 VMX root 操作模式下), 是所有支持 VT 技术的 VMM 的共性. 有些是 Pcanel/V2 自己所定义的(例如所有的硬件中断都通过 Pcanel/V2 处理), 是 Pcanel/V2 的特性.

4 运行时状态转换

图 2 显示了 Pcanel/V2 的运行流程.

- ① 开机初始化后 Pcanel/V2 判断处理器是否支持 VT 技术, 如果不支持就进入非 VMX 模式运行.
- ② 否则可以进入到 VMX 模式.
- ③ 在进行了 VMCS 的配置之后, 就可以通过 VM entry 进入到 Guest OS 运行.
- ④ Guest OS 运行过程中, 如果遇到异常, 或者资源访问等情况就会导致 VM exit.

⑤ Pcanel/V2 会对各种情况进行相应的处理后, 再返回到 Guest OS 运行.

系统稳定后在④和⑤之间不断切换.

5 VMCS 的配置

VMCS 的配置是整个 Pcanel/V2 设计的关键, 它控制了 VMX root 操作模式和 VMX non-root 操作模式之间的相互转换, 以及在 VMX non-root 操作模式中的处理器行为. 不同的配置会导致截然不同的 Guest OS 运行行为, 事实上, 基于 VT 技术开发的各类虚拟化系统具有不同特征的根本原因就在于各自对 VMCS 的配置不同.

Pcanel/V2 的设计基于 Intel IA-32 架构. 如果有多个 Guest OS 共同运行, 则每个 Guest OS 对应着一个 VMCS 并受其控制, 所以必须为每个 VMCS 分别初始化. 同时, Pcanel/V2 为多核扩展奠定了基础. 如果 CPU 为多核, 则每个核心运行一份 Pcanel/

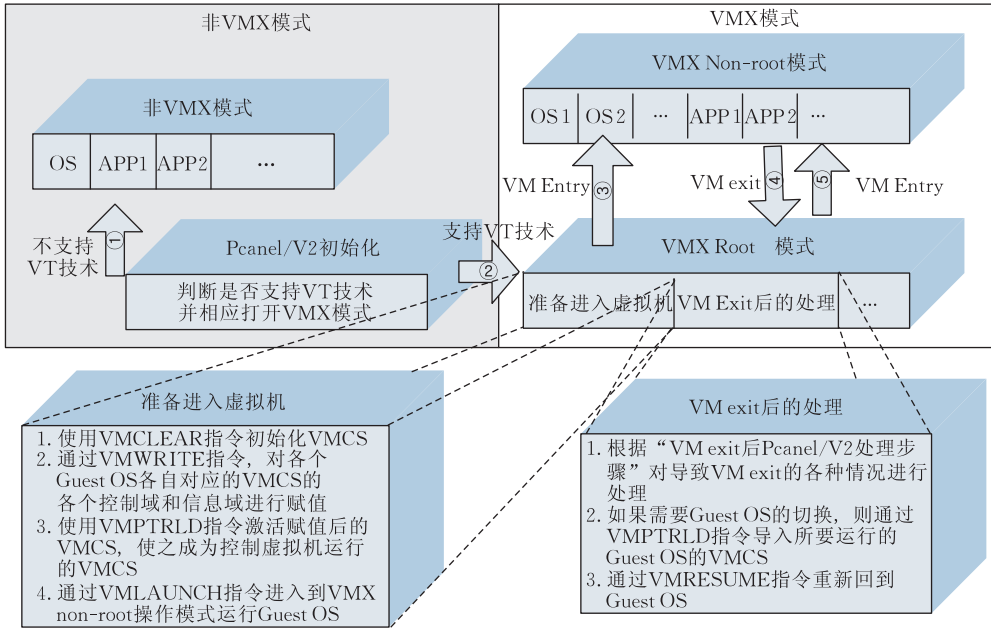


图 2 Pcnal/V2 运行流程图

V2 代码拷贝,再根据每个核心所虚拟的 Guest OS 数设置相同数目的 VMCS,但是必须修改 Pcnal/V2 以保证临界区数据在同一时刻只能被一个内核所修改。

VMCS 中的数据可以分为 6 组,所有域的值都是在 VMX root 操作模式下通过 VMWRITE 指令写入的,Pcnal/V2 对这 6 组数据域进行了如下配置:

(1) Guest-state 域. 当 VM entry 时,会将 guest-state 域中的信息导入至处理器的各个状态;当 VM exit 时,处理器的各个状态会保存到 guest-state 域中. 该域的值在第一次通过 VM entry 进入到 Guest OS 之前需要进行初始化,目的就是当 VM entry 完成后,创建出一个系统启动后的初时环境,以配合 Guest OS 的启动. 所有值都是根据 IA-32 在加电之后的各个初时状态配置的^[13].

(2) Host-state 域. 每次 VM exit 后,通过导入 host-state 域值进入到相同的处理环境中来对导致 VM exit 的各种原因进行处理。

(3) VM-execution control 域. 控制处理器在 VMX non-root 操作模式时的各种行为. 决定了何种情况下会导致 VM exit. 秉承之前所说的 Pcnal/V2 的设计原则的第 3 条,我们设计在 Guest OS 中运行以下敏感指令都不会导致 VM exit,它们是: NMI、RFLAGS、IF 设置、RDPMC、HLT、MWAIT、RDTSC、MOV DR、MONITOR、PAUSE、MOV from CR8、MOV to CR8. 而以下指令将产生 VM

exit,它们是外部中断、INVLPG、所有和内存有关的异常(例如缺页、段错误等)、I/O 端口的读写、MOV to CR3.

(4) VM-exit control 域. 控制 VM exit. 使得处理器在 64 位运行模式下运行,如果有外部中断产生,会将该中断信息保存在 VM-exit interruption-information 中,供 Pcnal/V2 处理。

(5) VM-entry control 域. 控制 VM entry. 使得处理器在 IA-32e 运行模式下运行。

(6) VM-exit information 域. 只读,描述了导致 VM exit 的各种原因。

6 VM entry 操作和 VM exit 操作

VM entry 实现了进入到 Guest OS 的运行. 每次 VM entry 时,首先进行基本的检查,以确保 VMCS 中各个域的值都配置正确. 然后通过 3 个途径导入处理器状态:从 VM-entry control 域中、从 guest-state 域中、从特定的控制寄存器中导入页目录表指针. 最后,如果 VM-entry interruption-information 域有效,则“注入”某个事件,就如同该事件真的在 Guest OS 上发生了一样。

导致 VM exit 的原因是当 Guest OS 在 VMX non-root 操作运行过程中,有些指令的执行或者事件的发生. 其中,有些指令,如 CPUID、VMCALL 等会无条件地导致 VM exit;有些指令,如 HLT、IN 等会根据 VMCS 中的配置来决定是否导致 VM

exit. 当 VM exit 发生时,会把导致 VM exit 的原因记录在 VM-exit information 域中,并且保存处理器的各个状态至 guest-state 域中. 然后通过 3 个途径导入处理器状态:从 VM-exit control 域中、从 host-state 域中、从特定的控制寄存器中导入页目录表指针. VM exit 结束时,处理器已经运行在 VMX root 操作模式下,Pcanel/V2 会进行后续的工作.

7 VM exit 后 Pcanel/V2 的处理

一般来说,每次 VM exit 之后,Pcanel/V2 都进入到一个相同的出口环境中执行后续代码,因此,要将这个出口环境的各个状态保存到 VMCS 中的 host state 域中. 在 VM entry 中,并没有保存 host state 域的步骤,就是因为这个域里的信息其实是相对稳定的,在第一次 VM entry 之前时就已经配置好了.

每次 VM exit 结束进入到的出口环境后,其代码的最主要功能是,根据 VM-exit information 域中的信息进行处理,可能是模拟某条指令,可能是进行内存操作,也可能会将某中断发回给 Guest OS,然后通过 VM entry 继续 Guest OS 的运行. 下面给出 Pcanel/V2 在 VM exit 后的处理方式. 先给出导致 VM exit 的原因,然后给出对应处理方式.

(1) VMX 指令的运行. 读取 vm-exit instruction length 域值. 然后修改 guest-state 域中的 RIP,加上 vm-exit instruction length 域值,这样就相当于使得 Guest OS 跳过该条指令的执行.

(2) CPUID 指令的运行. ①读取 guest state 中的 RAX. ②执行 CPUID 指令. ③将执行结果写入到 guest-state 域中的 RAX、RBX、RCX、RDX 中去.

(3) INVD 指令的运行. Pcanel/V2 不允许 Guest OS 具有清空高速缓存的能力(因为物理处理器高速缓存中的数据有可能包含其他 Guest OS 和 Pcanel/V2),所以 Pcanel/V2 的处理与对 VMX 指令的处理方式相同.

(4) MOV from CR3 指令的运行. 发送给内存虚拟模块处理.

(5) I/O 指令的运行. Pcanel/V2 读取 exit qualification 域获取端口号,通过对端口号的查询,确定 Guest OS 是对哪个虚拟设备进行操作,然后发送给相应的虚拟设备模块来完成操作,返回给 Guest OS. 具体步骤在设备虚拟章节中有描述.

(6) INVLPG 指令的运行. 发送给内存虚拟模

块处理.

(7) WRMSR 指令的运行. Pcanel/V2 使 WRMSR 指令无效,处理方式与对 VMX 指令的处理相同.

(8) 外部中断. Pcanel/V2 判断中断向量号并做出两种处理方式:直接发回给 Guest OS 处理,例如键盘中断,具体步骤是,将 VM-exit interruption-information 域中的信息拷贝到 VM-entry interruption-information 域中,然后通过“注入”发送虚拟中断给 Guest OS;第 2 种处理方式是,Pcanel/V2 使用自身的 IDT 来处理,例如时钟中断.

(9) 异常. 发送给内存虚拟模块处理. 其中,对于缺页异常,在 exit qualification 域中保存着导致缺页的线性地址.

(10) MOV to CR0 指令的运行. Guest OS 的上述操作 Pcanel/V2 无法完成,会导致异常.

(11) MOV to CR3 指令的运行. 这是 Guest OS 切换页表的操作,Pcanel/V2 发送给内存虚拟模块处理.

8 内存虚拟

Pcanel/V2 必须为各个虚拟机合理地分配和隔离各自的物理内存,同时实现 Guest OS 所使用的物理地址和实际硬件物理地址之间的映射关系. 因此内存虚拟模块的设计基于两个原则:(1) Pcanel/V2 掌握对物理内存的控制;(2) 支持 Guest OS 内存地址转换的功能.

为了使得 Pcanel/V2 对物理内存的完全控制,Pcanel/V2 就必须掌握处理器的地址转换机制. 所以,只有 Pcanel/V2 可以存取 CR3(保存着页目录表的基地址)和执行 INVLPG(唯一可以直接操作 TLB 的指令). 同时,Guest OS 自身也希望能够实现对内存地址转换的控制,它也会存取 CR3,也会执行 INVLPG. Pcanel/V2 必须允许并且支持这些操作. 基于以上原则,Pcanel/V2 的内存虚拟模块的设计如图 3 所示.

Guest OS 页表结构的功能就是“欺骗”Guest OS,使其认为掌握了对内存地址转换的控制. 事实上,Guest OS 页表结构并不直接或者间接地控制内存地址转换. 内存地址转换被处理器的 TLB 和 Pcanel/V2 页表结构所控制(TLB 中的映射关系是从 Pcanel/V2 页表结构中导入的).

我们允许 Guest OS 自由修改其自身的页表结

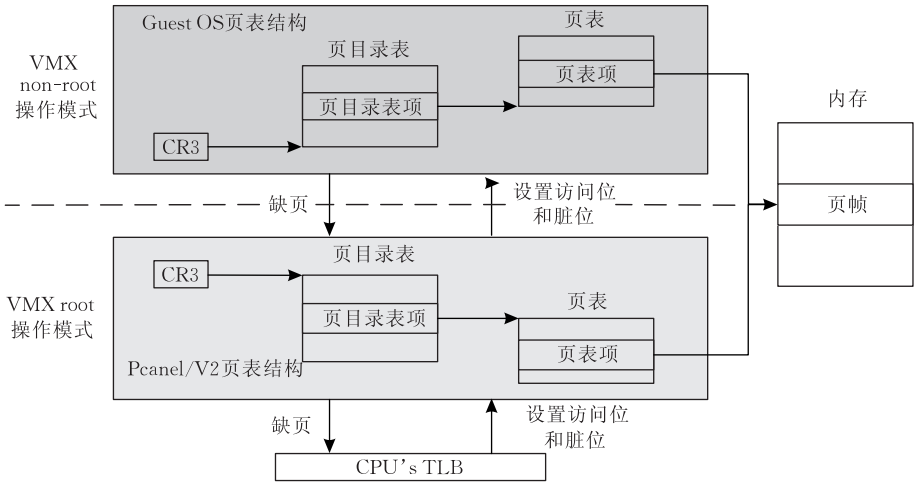


图 3 内存虚拟示意图

构而不用导致 VM exit,所以可能会导致 Guest OS 的页表层次与 Pcanel/V2 的页表层次不一致. 这种不一致分为两种情况,各自处理方式如下:

- (1) Guest OS 页表层次中比 Pcanel/V2 页表层次中有效信息多. 例如,有个线性地址到物理地址的内存转换关系在 Guest OS 页表结构中存在,但在 Pcanel/V2 页表结构中不存在. 这会导致缺页异常, Pcanel/V2 会更新其页表结构加入这个映射关系.
- (2) Guest OS 页表层次中比 Pcanel/V2 页表层次中有效信息少. 这种情况只会发生在 Guest OS 将其页表结构中某映射关系删除时(例如,标记某页不存在). Guest OS 在将某映射关系删除后,如果其程序设计正确,势必会执行 INVLPG 指令使得

在 TLB 中的该映射关系失效. 在 Pcanel/V2 设计中,指令 INVLPG 的执行会导致 VM exit, Pcanel/V2 就可以取得控制来完成其页表结构的相应更新.

此外,当处理器访问 TLB 时,也会更新 Pcanel/V2 页表结构中的 access bit 和 dirty bit. 为了保持 Pcanel/V2 页表结构和 Guest OS 页表结构的一致性, Pcanel/V2 必须也更新 Guest OS 页表结构相应页目录表项或者页表项的相应位.

内存虚拟模块还有一些事情要做,如前一节讨论,在 VM exit 后 Pcanel/V2 处理步骤中,有 4 种情况是要交给内存虚拟模块处理的,具体的处理方式如表 1 所述.

表 1 内存虚拟模块处理表

导致 VM exit 的原因	内存虚拟模块处理方式
MOV to CR3	对 CR3 进行写操作隐含着 TLB 的刷新和页表的转换,处理步骤如下: 1. 更新 guest-state 域中的 CR3. 2. Pcanel/V2 分配一个新的空页目录表. 3. 更新处理器 CR3 指向新目录表.
MOV from CR3	Pcanel/V2 在 VMX root 操作模式下模拟该指令的运行;将 guest state 域中的 CR3 值拷贝到目的寄存器或者内存地址中.
INVLPG 指令的执行	标记 Pcanel/V2 页表结构中页目录表项和页表项的相应项为不存在.
缺页异常(事实上,对缺页异常,首先通过设备虚拟模块处理,下文有详细说明)	如果缺页异常是因为 Guest OS 页表层次与 Pcanel/V2 页表层次不一致引起的,则通过更新 Pcanel/V2 页表层次(根据上文所述的两种情况处理)并重新执行导致异常的指令;否则交给 Guest OS 处理该异常.

9 设备虚拟

设备虚拟模块为每个 Guest OS 提供了一个抽象的 PC 硬件平台. 每个 Guest OS 看到的 PC 平台包括键盘、鼠标、实时时钟、8259 可编程中断控制器、8254 可编程计时器、CMOS、IDE 磁盘、软盘、光

驱和显示设备.

为了减少设计的复杂度, Pcanel/V2 重利用了 QEMU(一个开源的虚拟机项目)的设备模拟模块^[14]. 为每个 Guest OS 运行一个“设备模块系列”来模拟上述设备的实例. 设备虚拟模块的主要功能就是等待 Guest OS 的 I/O 事件,然后将其分派到相应的设备模拟模块去处理. 当完成了 I/O 请求之

后,向 Guest OS 返回结果。

通过之前对 VMCS 中 I/O bitmaps 的设置, Guest OS 对所有 I/O 端口的存取都会导致 VM exit. 在每次 VM exit 时,在 exit qualification 域中都会收集一些关于退出条件的信息,例如端口号、存取大小、方向、是否为串、有无 REP 前缀等, Pcanel/V2 将这些信息打包成一个 I/O 请求包并发给相应的设备模拟模块. 对 Guest OS 的一个 I/O 请求的处理过程如下:

- (1) I/O 存取导致 VM exit.
- (2) 通过读取 exit reason 域和 exit qualification 域来对指令解码.
- (3) 创建一个描述该事件的 I/O 请求包.
- (4) 将请求包发给相应的设备模拟模块处理.
- (5) 等待来自设备模拟模块相关 I/O 端口的回复或者 MMIO(Memory-Map I/O)操作.
- (6) 通过 VM entry 使得 Guest OS 继续运行.

大多数设备需要通过 Memory-Mapped I/O 来存取设备寄存器. 关键的中断控制器,例如 I/O APIC(Advanced Programmable Interrupt Controller),也需要 Memory-Mapped I/O 存取. 我们截取这些 Memory-Mapped I/O 存取作为缺页操作. 当每次因为缺页导致 VM exit 时,进行如下操作:

- (1) 检查 PTE 来确定所缺的页是否属于 Memory-Mapped I/O 的范围.
- (2) 如果是,对该指令解码,并发送 I/O 请求包给相应的设备模拟模块.
- (3) 如果不是,则将该缺页异常事件给内存虚拟模块处理.

10 实例研究

基于上述设计,我们开发了 Pcanel/V2 并在之上实现了对 Linux 和 Vxworks 的同时虚拟,这个虚拟方案兼顾了通用性、实时性和安全性. 通用的应用软件运行在 Linux 上,实时任务、有较好可靠性要求的任务运行在 Vxworks 上,同时 Pcanel/V2 保证了两个操作系统很好的隔离.

性能测试方面,我们选择以下系统作为对比:软件半虚拟技术的代表系统 Xen,在其上运行 Linux 系统;基于 VT 技术的 Xen-VT,在其上运行 Linux 系统直接运行于硬件机器平台上的 Linux. KVM 虽然属于基于 VT 硬件技术的虚拟系统,但是它让 Guest OS(Linux)在 Host OS(容纳 KVM 模块的

Linux)的用户态中运行,实际上相当于全虚拟化技术在 VT 技术上的扩展. 根据其官方白皮书^①及提供的测试数据^[9],其性能确实比采用软件全虚拟化技术的 QEMU 或 VMware 要好,但是却不如基于软件半虚拟化技术的 Xen,因此更比不上采用半虚拟化的基于 VT 的系统性能. 所以不作为比较对象.

硬件平台配置如下. 处理器:2.3GHz/800MHz FSB dual-core Intel Xeon processor,内存:4GB DDR2 533MHz memory,硬盘:160GB Seagate SATA disk,网卡: Intel E100 Ethernet controller). 在测试中把 Xen、Xen-Vt 和 Pcanel/V2 都分别配置成具有 512MB 内存,40GB 硬盘,其它不变.

使用的测试程序为 SPEC2000 和 SYSBENCH 测试程序组,它们是: (1) CPU. 测试 CPU 运算 2000 个素数加法时间. (2) THREADS. 测试并发 64 个线程平均每次响应时间,每次请求将调用 100 次 yields 方法,每个线程执行 2 次 lock 操作. (3) MUTEX. 测试 64 个线程互斥访问资源时平均请求响应时间, array 大小为 5000,每个线程互斥锁的数目为 5000, mutex-loops 互斥锁中的空循环数为 500. (4) MEMORY. 迁移 100MB 内存平均每块内存传输时间,内存块大小为 8K. (5) FILEIO. 随机读写文件时平均每次对写操作事件执行时间,采用随机模式读写,线程数为 16,总文件大小为 512MB,每次操作均创建和清理文件. (6) OLTP. 测试数据库并发访问的平均响应时间. (7) SPECINT. 测试计算整形数时间. (8) SPECFP. 测试计算浮点数时间.

我们把测试数据按在硬件平台上直接运行 Linux 的数据进行规一化,结果如图 4 所示. 图中每组的第 1 列代表 Xen,每组第 2 列代表 Xen-VT,第 3 列则为 Pcanel/V2. 可以发现,虽然某几项测试得分比 Xen 或 Xen-VT 低,但 Pcanel/V2 的总体性能比较理想,这也体现了硬件虚拟化固有的优势,符合我们当初的设计目标.

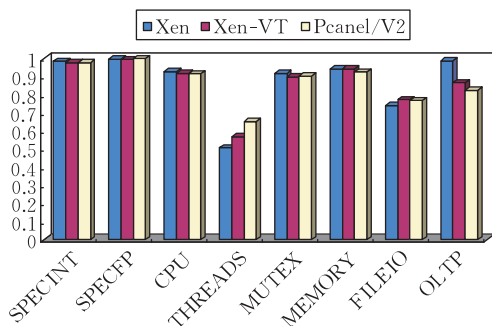


图 4 Pcanel/V2 系统性能测试

11 结束语

软件虚拟化技术已经发展了相当长的时间,但是硬件虚拟化技术是近一两年才开始发展起来的. Intel 在推出具有 VT 技术处理器的同时,也在进行相应 VMM 的开发, Xen 也在新版本中尝试加入对硬件虚拟化技术的支持. Pcancel/V2 是我们对硬件虚拟技术进行的初步实践,基本实现了硬件虚拟的目的,为今后的进一步优化打下了基础. 今后我们还将以下几个方面进行优化和完善: I/O 操作效率的进一步提高、对非真实硬件的虚拟化、对多核架构的支持、对 GUI 的支持等.

参 考 文 献

[1] Goldberg R P. Survey of virtual machine research. Computer, 1974, 7: 34-45

[2] Waldspurger C A. Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review, 2002, 36(SD): 181-194

[3] Barham P et al. Xen and the art of virtualization//Proceedings of the 19th ACM Symposium on Operating Systems Principles. Bolton Landing, NY, USA, 2003: 164-177

[4] Whitaker A, Shaw M, Gribble S. Scale and performance in the Denali isolation kernel//Proceedings of the 5th Symposium on Operating Systems Design and Implementation. Boston, Massachusetts, 2002: 195-210

[5] LeVasseur Joshua, Uhlig Volkmar, Chapman Matthew et al. Pre-virtualization: Slashing the cost of virtualization.



CHEN Wen-Zhi, born in 1969, Ph.D., associate professor. His research interests include embedded real-time system, distributed computing, virtualization technology and trusted computing.

YAO Yuan, born in 1982, M. S. candidate. His re-

Background

This work was partly supported by the Key Foundation Research and Development Program of China (973 Program) under Evaluating Theory and Means Research of Virtual Compute System (No.2007CB310906), the Foundation (Nos. A1420080190, 9140A15040309JW0402,

NICTA: Technical Report PA005520, 2005

[6] Neiger Gil, Santoni Amy, Leung Felix et al. Intel virtualization technology: Hardware support for efficient processor virtualization. Intel Technology Journal, 2006, 10(3): 167-177

[7] Uhlig R, Neiger G, Rodgers D et al. Intel virtualization technology. IEEE Computer, 2005, 38(5): 48-56

[8] Dong Yaozu, Li Shaofan et al. Extending Xen* with Intel® virtualization technology. Intel Technology Journal, 2006, 10(3): 193-203

[9] Michael Larabel. Linux KVM virtualization performance. Technical Report, Jan. 08, 2007

[10] Chen Wen-Zhi, Xie Cheng, Shi Jiao-Ying. A precise control core for component-based embedded operating system. Chinese Journal of Computers, 2006, 29(6): 867-874 (in Chinese)

(陈文智, 谢铨, 石教英. 一个构件化嵌入式操作系统的精确控制内核. 计算机学报, 2006, 29(6): 867-874)

[11] Chen Wen-Zhi, Xie Cheng, Shi Jiao-Ying. Component-based and model-driven embedded operating system. Journal of Zhejiang University: Engineering Science, 2005, 39(9): 1348-1352(in Chinese)

(陈文智, 谢铨, 石教英. 基于构件框架及模型驱动的操作系统内核. 浙江大学学报:工学版, 2005, 39(9): 1348-1352)

[12] Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3B, Nov. , 2006

[13] Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, Nov. , 2006

[14] Bellard Fabrice. QEMU, a fast and portable dynamic translator//Proceedings of the 2005 USENIX Annual Technical Conference. Anaheim, CA, 2005: 41-46

search interests include virtualization technology, computer system architecture and embedded system.

YANG Jian-Hua, born in 1973, Ph. D. , lecturer. His research interests include embedded system, computer architecture.

HE Qin-Ming, born in 1965, Ph. D. , professor. His research interests include computer software engineering, virtualization technology, etc.

9140A16070409JW0403,9140A06050609JW0402,2008ZH76007). The projects aim at developing the virtual platform supported by multiple architectures.

The research belongs to virtualization technology area which is developed rapidly by many companies such as

Microsoft, Intel and EMC and other open source projects such as Xen, KVM, QEMU and Denali. There are two kinds of virtualization technology: hardware virtualization technology and software virtualization technology. Hardware virtualization technology includes Intel VT technology and AMD Pacifica technology. Software virtualization technology includes full virtualization, para-virtualization and pre-virtualization. The virtualization technology is also an important portion of the 863 Program since 2007. The authors have engaged in several projects concerning virtualization technology and have developed a component-based embedded operating systems—Pcanel, which is based on component, middleware and real-time technologies and has the ability for configurability, reusability, adaptability, portability and robustibility. The previous research results also include three patents and four high quality papers.

The research of this paper is about the hardware virtual-

ization technology—A latest virtualization technology. This paper describes a VMM, called Pcanel/V2, which is based on the Pcanel and the Intel VT technology. The hardware virtualization technology is used to configure a controllable virtual execution environment, and to avoid changing the Guest OS source code while maintain a higher efficiency than traditional software virtualization technology. A control structure called VMCS of Pcanel/V2 is configured to construct a controllable environment for Guest OS. Pcanel/V2 can control the whole hardware rather than the Guest OS to deal with the resources management and can carry on corresponding processing for each kind of situation which appears in the guest operating system. Through Pcanel/V2, the authors have realized the basic purpose of hardware virtualization, and laid a foundation for further optimization for the future.