DOI: 10.3785/j.issn.1008-973X.2009.02.016

基于虚拟化平台的可信任计算基

陈文智,黄 炜,谢 铖,何钦铭
(浙江大学 计算机科学与技术学院,浙江 杭州 310027)

摘 要: 采用虚拟化技术来提高嵌入式设备的安全特性,避免了基于硬件方法带来的灵活性差、设备复杂度高和生产成本高的缺点.虚拟化平台 SmartVP 以软件方式实现了对系统环境的保护,SmartVP 在 ARM 处理器上划分出两组并行的计算资源,分别运行标准的实时操作系统 T-Kernel 和通用操作系统 Linux;通过保护 T-Kernel 上的软件代码和数据,实现了基础安全服务和接口,从而构造出一个可信任计算基.性能测试和实际应用结果均表明,SmartVP能够在提升嵌入式设备安全性的同时,有效地降低实施风险、时间和开发成本.

关键词: 虚拟化平台;可信任计算基;半虚拟化;资源分区;嵌入式设备;系统管理程序
中图分类号: TP316 **文献标识码**: A **文章编号**: 1008-973X(2009)02-0276-07

Trusted computing base using virtualization platform

CHEN Wen-zhi, HUANG Wei, XIE Cheng, HE Qin-ming
(*College of Computer Science and Technology, Zhejiang University, Hangzhou* 310027, *China*)

Abstract: Virtualization was introduced to increase the security of embedded devices while avoiding the shortcomings caused by the hardware-based approach such as poor flexibility, high complexity and high production cost of equipments. Virtualization platform SmartVP offers a secure system environment by the software-based approach. SmartVP partitions the computing resources on the ARM processor into two parallel sets, one for standard real-time operating system T-Kernel and the other for general-purpose operating system Linux. The trusted computing base of SmartVP is constructed by protecting the code and data of the software on T-Kernel, as well as by implementing the fundamental security services and interfaces. Both performance benchmark and applications show that SmartVP improves the security of embedded devices and reduces the implementing risk together with developing time and cost.

Key words: virtualization platform; trusted computing base; para-virtualization; resource partitioning; embedded device; hypervisor

随着越来越多的电子设备在连接网络的同时处理重要的任务和数据,安全特性对设备的性能和实用性具有相当显著的影响[1],也成为了电子设备领域关注的重要问题.一方面,移动终端具有开放的软件运行平台(如 WinCE[2] 和 MontaVista 的 Linux)意味着移动设备更容易受到攻击;另一方面,由于缺乏足够的安全技术,许多消费电子设备都无法对所提供的数字内容实现数字版权保护[3].随着蓝牙、无线局域网、3G 手机[4]等网络技术的迅速发展,嵌入式设备将面临更多的信息安全问题.迄今为止,大多数基于硬件来实现安全特性的电子设备都无法避免灵活性差以及设备复杂度、功耗和生产成本等提高的缺点.

虚拟化技术[5]为解决上述问题提供了思路并且已经被广泛运用.虚拟化技术通过把系统管理程序(hypervisor)作为系统中安全级别最高的部分并控

收稿日期: 2008-04-10. **浙江大学学报(工学版)网址**: www.journals.zju.edu.cn/eng
基金项目: 国家"973"重点基础研究发展规划资助项目(2007CB310906);浙江省科技计划重点资助项目(2006C21G2010099);国防基础科研项目(A1420080190).
作者简介: 陈文智(1969—),男,广西人,副教授,主要从事嵌入式实时系统、分布式计算和虚拟化技术研究. E-mail: chenwz@zju.edu.cn

制寄宿操作系统(guest OS)对资源具有不同级别的访问权限[6]，简单、灵活地实现安全策略. Xen 和 OSWare 软件[7]均采用半虚拟化方法(para-virtualization)在服务器领域实现了安全功能. 预虚拟化(pre-virtualization)技术[8]通过该方法来降低寄宿操作系统的性能损失.

在上述工作的基础上，本文将虚拟化技术移植到嵌入式环境中，提出了一种虚拟化平台 SmartVP. SmartVP 在 ARM 处理器上划分出两组并行的计算资源，分别运行标准的实时操作系统 T-Kernel 和通用操作系统 Linux，通过保护 T-Kernel 上的软件代码和数据，实现基础安全服务和接口，从而构造出一个可信任计算基(trusted computing base)，以软件保护的方式实现安全的软件运行环境.

1　SmartVP 的组成

信息安全领域的通用评估标准等级制度(evaluation assurance level, EAL)可以分为 7 级，其中 1 级最低，7 级最高. 许多小型的嵌入式操作系统均可采用 EAL 对系统安全性能的等级进行证实. 但是，随着系统规模和复杂程度的增加，确定 EAL 等级的困难随之增加. 采用虚拟化技术更容易发现操作系统的安全问题. 由于系统管理程序的规模比操作系统小得多，比较容易验证其安全性.

与 Xen 类似，虚拟化平台 SmartVP 采用半虚拟化方法，需要改动寄宿操作系统(如 T-Kernel 和 Linux 内核)的源代码. 但是在研究重点上与 Xen 存在不同之处. Xen 的目的是在 x86 平台上同时运行多个 Linux 操作系统，因此 Xen 研究许多与性能有关的问题. 此外，Xen 还研究如何在不同机器间搬迁正在运行的 Linux. 类似的研究思路曾经被 L4Linux 采用[9]，它与 Linux/x86 内核在二进制上兼容，可用于任何基于 PC 的 Linux 分发. 与 Xen 和 L4Linux 等虚拟化技术不同的是，SmartVP 向寄宿操作系统提供最小化的处理器和其他硬件资源，主要解决的问题是，在保证可信任计算基的前提下，创建高性能的用户态 ARM 处理器运行环境，既支持 Linux 这类通用操作系统，也支持 T-Kernel 这类实时操作系统.

SmartVP 的特点是，在基于 ARM 处理器的硬件平台上实现计算资源的分区，支持两个并行的执行领域：正常区域和安全区域. 在正常区域运行通用操作系统内核(如 Linux、WinCE、QNX 等)，在安全区域运行实时操作系统内核(如 T-Kernel、VxWorks等)或面向特定应用的定制的操作系统内核. 本文实现的 SmartVP 支持在正常区域运行通用操作系统 Linux，以及在安全区域运行实时操作系统 T-Kernel.

业界已经提出一些新的硬件安全技术用于实现计算资源的分区. ARM 在嵌入式领域内实现可信任计算的方法 TrustZone，它是基于可信任平台(trusted platform)的概念. TrustZone 将两个并行的执行区域分隔开：没有安全要求的正常区域和可信任的、可认证的安全区域. 安全区域由硬件增强的安全运行环境与安全软件组成，并向正常区域的操作系统和一般的应用程序提供安全服务和接口. 相对 TrustZone 而言，SmartVP 采用单纯的软件技术，可用于所有具备 MMU 功能的 ARM 处理器，能够在保证系统性能的同时实现相似的安全特性. SmartVP 重点支持 Linux 在 ARM 处理器核上的良好移植性以及对大量现存的驱动程序和应用程序的支持，包括许多厂商定制的硬件平台(如 OMAP 和 XScale 等).

SmartVP 的结构如图 1 所示，图中粗直线上、下分别代表处理器的用户态(user mode)和特权态(privileged mode)，图中包括一个极小的系统管理程序 V-Kernel以及 T-Kernel 和 Linux 内核，虚线所包括的区域形成安全的执行区域，即可信任计算基.

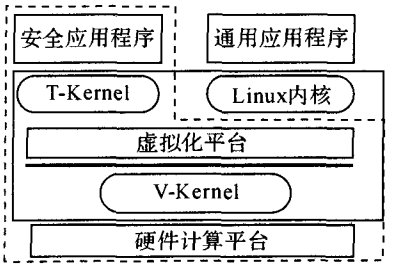

图 1　SmartVP 结构

Fig. 1　Structure of SmartVP

通过 SmartVP，寄宿操作系统运行在用户态，也就是说，将操作系统转变为一个应用程序的运行形态. 例如，能够提供一个 Linux 运行环境，保持对大量 Linux 应用程序的二进制兼容性. SmartVP 把整个嵌入式软件系统划分为两大部分：可信任计算基(包括 T-Kernel)和通用应用环境(包括 Linux 内核). 实时软件、安全软件等都运行在可信任计算中，而 Linux 内核则以用户态程序的形式，运行在可信任计算基之外的地址空间中. 因此，可信任计算基中的软件能够保证安全性.

1.1　系统管理程序 V-Kernel

系统管理程序 V-Kernel 为嵌入式系统提供了一种可信任的执行环境，可以和其他安全技术(如安全操作系统特性以及硬件加密模块)共同工作.

V-Kernel向用户态的操作系统提供了处理器特权指令、中断控制器、协处理器和 MMU 的访问接口.

V-Kernel 检查每一次由 T-Kernel 以及 Linux 内核发起的操作,确保可信任计算基的地址空间不受影响.官方发布的 T-Kernel 不支持 MMU,因此,SmartVP 为 T-Kernel 增加了 MMU 扩展功能.由于 V-Kernel 负责管理中断控制器,优先级更高的 T-Kernel 总是能够迅速抢占 Linux 内核和 Linux 应用程序的执行.

1.2 实时操作系统 T-Kernel

T-Kernel[10]及其前身 TRON 操作系统是源码开放的嵌入式实时操作系统内核,具有开放的标准,目前占据了全球嵌入式微处理器操作系统市场约60%的份额.T-Kernel 采用一个开放式标准平台结构 T-Engine,使在 T-Kernel 上开发的中间件很容易移植到不同的处理器架构.T-Kernel 具有良好的可移植性,它以硬件平台的规格作为移植工作的基础,包括处理器架构的移植和附属设备的移植两部分.这种移植性不同于 Linux 内核,后者以处理器架构作为移植工作的基础,然后为不同的硬件平台作进一步移植.为了与 Linux 内核共同运行,T-Kernel 只管理一部分内存地址空间,用于执行实时任务和安全软件模块.标准的 T-Engine 架构提供了一种称为 T-Bus 的机制,允许 T-Kernel 与其他操作系统进行通信.通过 SmartVP 支持 T-Bus 机制,T-Kernel 能够与 Linux 内核进行协同工作.

1.3 通用操作系统 Linux 内核

SmartVP 对官方发布的 Linux 内核源代码进行了少量修改.通过常规的移植方法,Linux 内核被移植到 SmartVP 上,不仅运行在处理器的用户态,并且具备完全二进制兼容的标准 Linux 运行环境,可以运行大量第三方的网络应用和图形用户界面等.对于 Linux 内核而言,虚拟化平台相当于一个指令处理器加上一个协处理器,它们向 Linux 内核提供对物理处理器的各种特权操作的接口,包括 MMU 管理、中断控制器和协处理器调用.Linux 内核负责虚拟地址页表的分配和管理,但是不能直接访问 MMU 的寄存器.Linux 内核和 Linux 应用程序的调度具有分级关系,即 V-Kernel 调度 Linux 内核,再由 Linux 内核调度 Linux 应用程序.

SmartVP 提供的安全服务和接口如图 2 所示.图中显示了 3 种具有安全要求的应用,其中深色矩形所示的软件模块是安全的、可信任的.普通应用程序直接运行于正常区域的 Linux 内核上,访问不安全的密钥等信息.电子钱包应用程序是安全的,它虽

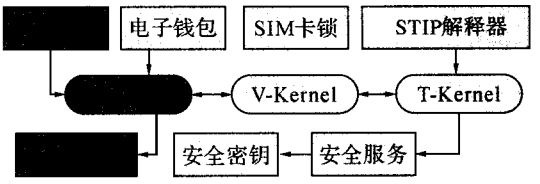

图 2 SmartVP 的安全服务和接口

Fig. 2 Security services and interfaces of SmartVP

然运行在 Linux 内核上,但是通过安全接口访问密钥等信息.该安全接口由 T-Kernel 上的安全服务实现,当电子钱包程序访问该接口时,V-Kernel 将切换到安全区域内执行 T-Kernel 以及相关的应用程序.SIM 卡锁应用完全运行在 T-Kernel 上,通过小型终端互操作性平台(small terminal interoperability platform, STIP)规范字节码的解释器,访问安全服务管理的密钥等信息.

2 SmartVP 的资源分区和共享

SmartVP 使得一个单核处理器上可以同时运行多个寄宿操作系统.这些寄宿操作系统互相保持独立的运行,但是可以通过 SmartVP 提供的通信机制进行协作.SmartVP 是一个极小的抽象层,管理重要的系统资源,将寄宿操作系统与底层硬件隔离,关键技术是资源的分区和共享.图 3 所示是一个典型的嵌入式设备的资源,图中(带有虚线框或实线框的)深色矩形代表安全区域,浅色矩形代表正常区域.正常区域包括:1)在寄宿操作系统之间的资源分区(partitioning),如内存、闪存、ROM 等,如图中用方格填充的矩形;2)对不可分区的资源的共享(sharing),典型的有处理器、指令/数据缓冲、协处理器、内存控制器、中断控制器等.共享又可分为 2 种,即简单的共享和通过虚拟设备驱动程序实现的共享,前者在图 3 中用斜线填充的矩形所示,后者在图 3中带虚线框的矩形所示.

图 3 资源分区和共享

Fig. 3 Partition and sharing of resources

2.1 资源的分区

典型的可分区的资源是物理内存.分区后的每一块资源只由一个寄宿操作系统进行排他式的使

用，因此，每个寄宿操作系统可以采用自定义的管理机制和策略，例如内存管理. 若一个设备只由一个寄宿操作系统使用，则同样属于可分区的资源. 对于这类资源，寄宿操作系统可以直接使用原有的设备驱动程序. 如图 3 中所示的加密(encryption)模块只由可信任计算基中的软件进行访问，通用 I/O 控制器只由 Linux 操作系统进行访问.

以 ARM Linux 为例，虚拟化平台 SmartVP 只更改了 Linux 内核中与 ARM 处理器相关的少量代码，包括中断控制器、协处理器和 MMU 管理，内核中的绝大部分代码(包括其他采用 ARM 处理器的特定硬件平台架构如 XScale、OMAP 等相关的代码)都可以运行在虚拟化平台 SmartVP 上. 因此，随着 Linux 内核版本升级，虚拟化平台 SmartVP 的维护成本远远小于 Xen 等系统管理程序或 L4Linux 等用户态 Linux 操作系统.

2.2　资源的共享

一些资源必须被多个寄宿操作系统使用，例如处理器和时钟信号等. 因此，这些资源被虚拟化，从而在寄宿操作系统之间得到共享. 为了保证效率，SmartVP 采用半虚拟化方法. 也就是说，寄宿操作系统的内核需要有一些改动，这些改动相当于将操作系统移植到一个与底层硬件非常相似的架构上.

SmartVP 总是将处理器核、指令/数据缓冲、内存控制器、显示屏等资源进行虚拟化. 通过抢占式的调度机制，处理器在多个寄宿操作系统间得到共享，因此 SmartVP 能够保证实时的寄宿操作系统具有更高的优先级. 当一个寄宿操作系统获得处理器资源后，它对自己的应用程序使用自定义的调度策略. MMU 被虚拟化后同样在多个寄宿操作系统间得到共享，因此每个寄宿操作系统仍然使用自定义的内存管理方法.

2.3　设备的共享

在相同的处理器上同时运行不同的操作系统，只是 SmartVP 所实现的基本功能之一. 正如一个操作系统支持多个进程并提供多种服务，如内存分配、调度策略、进程间通信以及对文件系统、网络协议等共享访问，SmartVP 同样向每一个寄宿操作系统提供多种服务，这些服务通过驱动程序的形式实现，包括对设备(如以太网、串行口等)的共享访问以及在寄宿操作系统之间的通信.

许多标准化的输入、输出设备如以太网、串行口等往往需要被多个寄宿操作系统访问，对于这类设备，SmartVP 实现了缺省的设备驱动程序管理物理硬件，并向寄宿操作系统提供相应设备的访问接口. 这一方法使得寄宿操作系统不直接访问设备同时可使用设备的标准功能. SmartVP 同样提供一类虚拟设备，实现不同的寄宿操作系统间的通信. 不同类型的虚拟设备适应不同的通信要求. 例如，虚拟的以太网设备用于实现一个私有的局域网络，因此寄宿操作系统之间可以通过标准的 TCP/IP 协议实现通信.

3　系统管理程序 V-Kernel

V-Kernel 是一个采取微内核结构的系统管理程序，支持虚拟化平台 SmartVP，使得 T-Kernel 和 Linux 内核在用户态运行. V-Kernel 只实现基本的资源分区和共享功能，与硬件有关的驱动程序则运行在安全的 T-Kernel 中或非安全的 Linux 内核中，与单一结构的复杂的系统管理程序 Xen 相比，具有更高的可靠性，更容易移植到不同的硬件平台上.

V-Kernel 是整个软件系统中唯一运行在处理器特权态的软件，负责处理硬件中断和 MMU 管理. V-Kernel 支持两个用户态线程，经过半虚拟化的 T-Kernel 和 Linux 内核分别在这两个线程中运行. 其中运行 T-Kernel 的线程具有较高的优先级，运行 Linux 内核的线程具有较低的优先级. 运行在处理器用户态的 Linux 内核，相当于一个 Linux 服务器，Linux 应用程序所调用的 SWI 指令被 V-Kernel转移至 Linux 内核的接口上. 通过标准的 MMU 保护机制，可信任计算基中的软件与 Linux 环境相分离. 即使整个 Linux 环境(包括 Linux 内核)出现致命错误，也不会影响可信任计算基. 同样的，Linux 环境可以随时关闭和启动. 下面详细介绍 V-Kernel 的设计.

3.1　地址空间

从地址空间和安全保护的角度看，整个系统中的软件按照如图 4 所示的位置进行分布. 特定应用软件，如专属固件(包括流媒体处理和安全模块等)和实时任务等，与 Linux 内核共享相同的地址空间，V-Kernel 通过硬件 MMU 保护前者的数据和代码

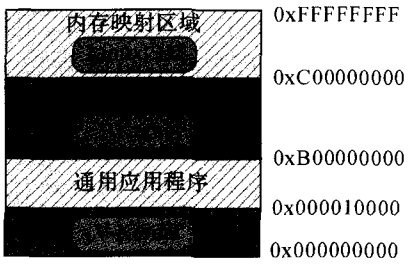

图 4　地址空间分布

Fig. 4　Address space distribution

不能被 Linux 内核访问. 专属固件中可能包括一些硬件驱动,它们需要和 Linux 内核有较为频繁的交互,相同地址空间使得驱动程序和 Linux 内核之间共享数据和代码更容易,效率更高. 每个 Linux 应用程序则运行在独立的地址空间. 图 4 中表示了整个地址空间,并标明了各部分软件的位置. V-Kernel 位于地址空间的最低端,用于接收硬件异常,并根据情况将处理器的控制权转交给 T-Kernel 或者 Linux 内核.

ARM 处理器架构支持 Domain 功能:一个 Domain指的是地址空间中的一部分,包括一组 Section、Large Pages 和 Small Pages,通过设置协处理器 CP15 可以快速地令一个 Domain 在地址空间中启用或关闭. Domain 功能在一定程度上实现了地址空间的分区功能. 尽管这并非真正的硬件虚拟化技术(如 Power970 处理器的 Logical Partitioning),但是 V-Kernel 使用 Domain 功能,能够迅速改变寄宿操作系统和应用程序等对地址空间的访问方式,并避免缓冲失配引起的开销. 第 5 章所做的性能测试表明,V-Kernel 对寄宿操作系统性能的影响是非常小的.

3.2 虚拟处理器架构

V-Kernel 向用户态的操作系统提供了虚拟的处理器硬件资源. 为简单起见,这个虚拟处理器的架构简称为虚拟化架构. 通过常规的硬件架构移植方法,Linux 内核被移植虚拟化架构上. 因此,Linux 内核和 Linux 应用程序实际上都在 V-Kernel 的一个线程中运行.

对于 Linux 内核而言,虚拟化架构相当于一个普通的处理器加上一个协处理器. 当物理处理器的架构是 ARM 时,虚拟化架构相当于一个基本的 ARM 指令处理器以及定制的一个具备 MMU 和中断处理的协处理器. 这个虚拟的协处理器提供一组寄存器进行操作,简称为虚拟寄存器组. 它们的访问地址位于 0x00000000～0x00001000 范围内(即地址空间的最开始的 4 K 字节范围). 这些虚拟的寄存器能够间接地向 Linux 内核提供对物理处理器的各种特权操作的接口,包括 MMU 管理、中断控制器和协处理器调用. Linux 内核负责虚拟地址页表(page table)的分配和管理,但是不能直接访问 MMU 的寄存器,而是由虚拟寄存器组中的 MMU 寄存器进行代理. V-Kernel 检查每一次对虚拟寄存器组的操作,确保可信任计算基的地址空间部分不受影响. 除了虚拟寄存器组的地址空间受到 V-Kernel 保护外,Linux 内核可以直接访问其余的地址空间,包括大部分硬件寄存器和内存,因此可以直接运行原有的驱动程序.

3.3 虚拟处理器调度

虚拟寄存器组不仅帮助 Linux 内核间接地进行处理器特权操作(例如进程上下文和地址空间的切换),而且能够实现一些定制的虚拟硬件,即在 T-Kernel上运行驱动程序,通过虚拟寄存器组向 Linux 内核提供虚拟的硬件访问接口.

Linux 内核和 Linux 应用程序共享一个虚拟处理器,因此 Linux 内核以及 Linux 应用程序的调度具有分级关系. 简单地说,就是由 V-Kernel 调度 Linux 内核,再由 Linux 内核调度 Linux 应用程序. V-Kernel 采用基于优先级的调度算法,调度所有用户态的线程,因此,优先级更高的线程(如 T-Kernel)总是能够迅速抢占 Linux 内核的执行.

4 可信计算基与标准化的接口

SmartVP 可以严格地区分(partitioning)安全的系统和不安全的系统,从而构造出一个可信任计算基. 图 4 中用右斜线填充的矩形表示的部分,是整个系统的可信任计算的基础,即可信任计算基(trusted computing base, TCB). 通常,计算机的所有硬件被包括在 TCB 中,本文主要关注 TCB 中的软件组成部分. 安全应用总是具有可信任的代码来源,因此 V-Kernel 向它们开放所有的调用接口. V-Kernel 和通用的硬件保护机制结合在一起,能够保护 TCB 的重要部分(包括 T-Kernel 和安全应用程序等)不受 Linux 内核和 Linux 应用程序的影响,如计算机病毒、软件故障、恶意攻击和安全漏洞等方面. V-Kernel对可信任计算基的保护是基于标准的内存保护机制,适用于所有具备 MMU 的处理器架构,因此具有广泛的通用性,配合硬件安全技术(如 TrustZone)则可以得到更好的执行效率.

同时,SmartVP 定义了一组接口,系统管理程序以函数调用的形式将这组接口提供给寄宿操作系统. 例如寄宿操作系统可以调用 vp_flushCache 接口刷新虚拟化平台的一部分或全部指令/数据缓冲,该接口在 C 语言中的定义如下:

```
typedef void (*_vp_flushCache)(void);
#define vp_flushCache ((_vp_flushCache)0x40)
```

预先定义这组接口的实现代码的函数地址从 VP_ENTRY 开始. 当寄宿操作系统调用 vp_flushCache() 时,将触发指令异常(instruction abort),此时 LR 寄存器的值为 0x44. 系统管理程序从(ENTRY+0x44)位置处获取相应的函数地址并执行之.

基于虚拟化平台实现可信任计算基，要求系统管理程序和寄宿操作系统之间具有一组标准化的虚拟化平台的接口，其作用有两方面：1)隔离了系统管理程序和寄宿操作系统的实现，例如 Linux、T-Kernel、WinCE、VxWorks 等多种寄宿操作系统都可以被移植到 SmartVP 上；2)同一个寄宿操作系统可以运行在由不同硬件平台厂商自行实现的系统管理程序上，达到一定程序的跨平台性.

表 1 所示的是虚拟化平台的一组基本接口，包括接口的名称和功能. 系统管理程序的实现应当支持它们. 对于不同的 ARM 处理器核以及硬件平台，系统管理程序还可以提供自定义的接口.

除了上述接口，寄宿操作系统的异常向量(exception vector)表分布在高端位置 0xffff0000 处. 当系统管理程序产生一个虚拟化平台的异常时，它令寄宿操作系统跳转到相应的异常向量处执行.

表 1　虚拟化平台的接口

Tab. 1　Interfaces of virtualization platform

名　称	功　能	名　称	功　能
vp_init	寄宿操作系统在内核启动时首先需要虚拟化平台进行初始化工作，在此区分出正常区域和安全区域	vp_switchPageTable	虚拟化平台从一个地址空间切换到另一个地址空间
vp_vprintf	寄宿操作系统向虚拟化平台打印一些信息	vp_switchMode	虚拟化平台从特权态返回到用户态
vp_newFirstLevelTable	当寄宿操作系统需要使用一个新的地址空间时，通过虚拟化平台的该接口生成一个新的第一级页表	vp_readRegister vp_writeRegister	系统管理程序提供了一组虚拟的寄存器，对应于虚拟化平台的中断控制器、定时器、协处理器等
vp_newCoarsePageTable	当寄宿操作系统需要使用一个新的地址空间时，通过虚拟化平台的该接口生成二级页表，每页大小为 4 K	vp_flushCache	刷新一部分或者全部的指令/数据缓冲
vp_setFirstLevelDescriptor	设置第一级页表中的项，即从虚拟地址的 1 M 字节空间到物理地址的映射关系	vp_flushDma	刷新 DMA 硬件机制的缓冲
vp_ setCoarsePageDescriptor	设置第二级页表中的项，即从虚拟地址的 4 K 字节空间到物理地址的映射关系	vp_flushTlb	刷新一部分或者全部的地址映射关系的缓冲
vp_copyMapping	寄宿操作系统提供一组虚拟地址到物理地址的映射关系，系统管理程序生成相应的一个第一级页表和一组第二级页表	vp_snapshot	对当前的虚拟化平台做一个备份点，使得寄宿操作系统可以在此状态暂停或者恢复运行

5　实验结果及分析

在测试和评估虚拟化平台 SmartVP 的过程中，采用 Intel 公司提供的 Lubbock 开发板，它具有 200 MHz 核心频率的 PXA255 处理器和丰富的周边器件. PXA255 处理器采用英特尔 Xscale 架构，兼容 ARMv5 处理器指令集，它除了应用于掌上电脑外，还可应用于智能手机、网络存储设备、骨干网路由器等嵌入式设备. Xscale 在 ARM 处理器核的基础上为多媒体处理作了许多扩展，包括乘/加法器 MAC 和特定的 DSP 型协处理器 CP0.

SmartVP 在实现可信任计算基的同时会对寄宿操作系统(包括 T-Kernel 和 Linux 内核)的性能造成一定影响. 由于 Linux 内核的复杂度比 T-Kernel高得多，对运行在 SmartVP 上的 Linux 内核(称为 VP-Linux)和官方发布的版本进行测试，评估 SmartVP 在系统调用、FIFO 延时和 PIPE 流量方面的三项性能指标，这组典型的指标表明了 V-Kernel对寄宿操作系统性能造成的影响.

Linux 内核(2.6.15 版本)经过半虚拟化方法的改造，运行在虚拟化平台 SmartVP 上，增加了系统调用的开销. 第一个基准测试是空的系统调用延迟，它表明应用程序与 Linux 内核之间通信的时间成本. 第二个基准测试是 FIFO 的通信延时，它表明在两个进程之间通信的时间成本. 第三个基准测试是进程间管道的通信效率，即数据流量.

测试结果如表 2 所示. 可以看出，VP-Linux 使系统调用的时间成本增加了 1 μs. 由于 Linux 内核中通常执行较多的工作(文件系统、网络协议等都位于 Linux 内核中)，许多系统调用的时间成本达到 ms 级以上，因此 VP-Linux 对应用程序性能的实际

表 2 VP-Linux 与 Linux 的性能

Tab. 2 Performance of VP-Linux and Linux

被测试平台	系统调用/μs	FIFO 延时/μs	PIPE 流量/(MB·s^{-1})
VP-Linux	13	164	12.10
Linux	12	160	12.58

影响较小. 由于两个 Linux 进程之间的通信需要经过 Linux 内核和 V-Kernel 的双重处理,在进程间通信方面产生额外的时间成本,增加了 2.5%的延时. 由于同样的原因,管道的数据流量减少了 3.8%.

对 SmartVP 的安全隔离性能进行了测试. 一方面,通过在运行于 SmartVP 上的 Linux 操作系统中创建用户进程和内核线程并尝试对非 Linux 资源进行直接访问,SmartVP 提供的接口访问机制都能够有效地屏蔽由 Linux 操作系统所发出的非法请求和操作;另一方面,通过在 Linux 内核中造成内存泄漏等漏洞,同时给 Linux 加载很高的进程负荷并导致 Linux 系统崩溃的情况下,由 SmartVP 提供的隔离机制使得 T-Kernel 保持良好的运行状态:由 T-Kernel控制的 CPU、内存和 I/O 等各种资源正常使用;运行在 T-Kernel 上的各个进程和相关资源也不会受到 Linux 内核崩溃的影响;Linux 经过重新启动后仍然能够正常运行.

这些测试说明,VP-Linux 的架构对单个 Linux 应用程序的性能影响很小. 通常在嵌入式系统的运行环境中,进程间的通信不像通用计算领域那样频繁,因此采用 SmartVP 不会对整体性能造成很大影响,与此同时,它很好地将系统的可信任计算基和 Linux 运行环境隔离开,保证了充分的安全性.

在多种软件平台中部署了 SmartVP,包括一个支持以太网交换机的嵌入式软件平台和一个支持视频点播和网络会议的分布式中间件,以及将一个构件化的嵌入式操作系统移植到 SmartVP 上. 这些软件系统的开发和运行结果表明,SmartVP 能够在提升嵌入式设备的安全性的同时,有效地降低实施风险、时间和开发成本.

6 结 语

本文提出和实现了一个采用半虚拟化方法的虚拟化平台 SmartVP,它支持可信任计算基,在处理器的用户态下运行标准的实时操作系统 T-Kernel 和通用操作系统 Linux,从而兼容大量第三方应用程序. 在 V-Kernel 的支持下,SmartVP 实现了一种虚拟的处理器架构,它是对操作系统内核中与 ARM 处理器相关代码的少量改动,主要包括中断控制器、协处理器和 MMU 管理. SmartVP 是单纯基于软件的安全解决方案,性能测试和实际应用结果都表明,SmartVP 能够在实现可信任计算基的同时,有效降低嵌入式设备上的实施风险、时间和开发成本. 但是该虚拟化平台仍然存在一些有待研究和改进的问题,如通过形式化方法验证其接口、支持复杂的多核处理器架构等.

参考文献(References):

[1] JONES S, WILIKENS M, MORRIES P, et al. Trust requirements in e-business [J]. **Communications of the ACM**, 2000, 43(12): 81-87.

[2] O'HARA R. Microsoft windows C E: a new handheld computing platform [C]// **Proceedings of the 1997 ACM Symposium on Applied Computing**. San Jose: ACM, 1997: 295-296.

[3] BOGDAN C P, CRISPO B, ANDREW S T. Support for multi-level security policies in DRM architectures [C]// **Proceedings of the 2004 Workshop on New Security Paradigms**. Nova Scotia: ACM, 2004: 3-9.

[4] THOMAS S M, EZZAT A D. Digital rights management in a 3G mobile phone and beyond [C]// **Proceedings of the 3rd ACM Workshop on Digital Rights Management**. Washington DC: ACM, 2003: 27-38.

[5] CRISWELL J, ANDREW L, DINAKAR D, et al. Secure virtual architecture: a safe execution environment for commodity operating systems [C]// **Proceedings of Twenty-first ACM Symposium on OS Principles (SOSP)**. Stevenson: ACM, 2007: 351-366.

[6] SESHADRI A, LUK M, NING Q, et al. SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes [C]// **Proceedings of Twenty-first ACM Symposium on OS Principles (SOSP)**. Stevenson: ACM, 2007: 335-350.

[7] DRAGOVIC B, FRASER K, HAND S, et al. Xen and the art of virtualization [C]// **Proceedings of the ACM Symposium on OS Principles (SOSP)**. New York: ACM, 2003: 164-177.

[8] LEVASSEUR J, UHLIG V, LESLIE B, et al. Pre-virtualization: uniting two worlds [C]// **Proceedings of the Twentieth ACM Symposium on OS Principles (SOSP)**. Brighton: ACM, 2005: 1-2.

[9] HARTIG H, HOHUMTH M, LIEDTKE J, et al. The performance of μ-kernel-based systems [C]// **Proceedings of the 16th ACM Symposium on OS Principles (SOSP)**. St. Malo: ACM, 1997: 66-77.

[10] SAKAMURA K, KOSHIZUKA N. T-Engine: the open, real-time embedded-systems platform [J]. **IEEE Micro**, 2002, 22(6): 48-57.