基于微内核的虚拟机间通信加速方法①

乔若轩[1,2], 吴 涛[1,2], 杨秋松[1]

[1](中国科学院软件研究所 基础软件国家工程研究中心, 北京 100190)
[2](中国科学院大学, 北京 100049)

摘 要: 基于微内核的虚拟化架构相较于传统的宏内核虚拟化架构, 具有可信计算基小, 易于完全形式化验证的特点. 然而, 在基于微内核的虚拟化架构中, 即使在同一物理机上运行的不同虚拟机, 虚拟机间通信仍需要通过调用网卡驱动传输数据, 通信效率低. 针对以上问题, 提出了一种同一物理机上不同虚拟机间的通信加速方法, 通过在网络服务中加入通信数据选择模块和转发模块, 使得虚拟机间数据的传输可以直接在内存中完成. 实验表明, 可以有效提高虚拟机间的通信效率.

关键词: 微内核; 虚拟化; 虚拟机间通信

Acceleration Method for Communication Between Microkernel Based Virtual Machines

QIAO Ruo-Xuan[1,2], WU Tao[1,2], YANG Qiu-Song[1]

[1](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Compared with traditional monolithic kernel based virtualization architecture, microkernel-based virtualization architecture has smaller trusted computing base and its security is easier to be verified by formal methods completely. However, in microkernel-based virtualization architecture, communication between virtual machines on the same computer still requires the involvement of network card driver, which leads to low efficiency. In this paper, we propose a method to accelerate communication between virtual machines on the same computer by adding data selection module and data transmission module in the network service of microkernel-base virtualization architecture. With the method, communication between virtual machines on same computer can be completed in the memory instead of network card driver. Experiment shows that our method can improve the efficiency of communication effectively.

Key words: microkernel; virtualization; inter-VM communication

1 引言

作为一种高效利用计算机硬件资源的方法, 虚拟化技术随着云计算产业的兴起得到了越来越多的关注. 虚拟化技术[1]可以在一台物理机上虚拟出多个虚拟机, 各个虚拟机中都可以运行独立的操作系统和应用而不相互影响. 典型的虚拟化产品有 Xen[2], Hyper-V[3], VMware Esxi[4]等.

在虚拟化技术中虚拟机运行于虚拟机监控器之上, 一旦虚拟机监控器被恶意程序攻破, 那么运行于虚拟机监控器之上的虚拟机的安全性就很难得到保证[5]. 目前主要的虚拟机产品都存在着可信计算基 TCB (Trusted Computing Base)过大, 安全隐患较多的问题, 其虚拟化架构的安全性难以得到验证.

减小系统的 TCB 可以有效增加系统安全性[6]. 采用微内核思路重新设计虚拟化架构, 非核心系统服务如设备驱动、网络支持、文件系统等实现在虚拟机监控器外, 虚拟机监控器仅提供基本的资源调度、进程间通信、地址空间管理等功能, 这使得基于微内核的虚拟化架构的 TCB 相较于 Xen, KVM, VMware ESXi 等降低了两至三个数量级, 对其代码进行彻底的安全

① 基金项目:中国科学院重大方向性项目(KGCX2-YW-125);国家自然科学基金(1318301,61432001)
收稿时间:2015-03-02;收到修改稿时间:2015-04-26

性验证成为可能[7]. 因此使用基于微内核的虚拟化架构可以更好地保证整个系统的安全性. 其中典型的代表就是 NOVA[8].

在很多应用场合, 如通信密集型的分布式应用、高性能网格应用中, 出于性能、扩展性等因素的考虑, 一项任务往往需要数量众多的虚拟机协同工作才能完成, 虚拟机间为协同工作需要通过通信的方式来传输数据和控制信息, 因此虚拟机间通信是不可缺少的. 为了保证系统的稳定性, 避免服务间冲突, 在一台服务器中常常只运行一个服务或应用, 这导致服务器的硬件资源利用率较低[9]. 采用虚拟化技术可以在一台服务器中虚拟出多台虚拟机, 在大大提高硬件资源利用率的同时, 也使得同一物理机间不同虚拟机间通信的比较频繁[10], 同一物理机上不同虚拟机间通信的效率十分对于整个系统的性能十分重要. 虚拟机间通信效率可以通过优化方法来提高[11,12], 然而在基于微内核的虚拟化架构 NOVA 中并没有对同一台物理机中虚拟机间通信与不同物理机间虚拟机通信进行区分, 虚拟机间通信都是调用网卡驱动通过网卡硬件来完成通信数据的传输. 同一台物理机上不同虚拟机间通信所传输的数据全部存在于本地物理机中, 不需要网卡读取其他数据, 它们间的通信可以在物理机的内存中完成, 若采用调用网卡驱动方式传输数据, 通信路径长, 性能开销大, 导致通信效率较低.

针同一物理机上不同虚拟机间通信效率较低的问题, 本文提出了一种高效的虚拟机间通信方法, 通过在物理机的内存中建立数据传输通道, 提高了同一物理机中不同虚拟机间通信的效率.

2 基于微内核的虚拟机网络通信

NOVA 是一个基于微内核的虚拟化架构, 通过将设备驱动、网络支持、文件系统等模块放于微内核之外, 大大减小了整个微内核架构的 TCB.

2.1 NOVA 总体架构

NOVA 架构如图 1 所示. NOVA 包含一个微虚拟机管理程序(microhypervisor)和用户级运行环境. 在 NOVA 中, microhypervisor 是运行于 CPU 的最高特权级 Ring 0 中, 负责进程间通信、资源调度, 中断控制等核心功能; 用户级环境负责为虚拟机操作系统的运行提供必要的功能, 这些功能包括根分区管理器(Root Partation Manager)、硬件设备驱动(Drivers)、虚拟机监控器(VMM)和为虚拟机提供必要功能的应用(Applicaions).

NOVA 中的虚拟机监控器(VMM)是一个运行于用户态的程序, 它负责为虚拟机执行敏感指令, 管理虚拟机内存地址和硬件物理内存地址间的映射, 以及为虚拟机提供虚拟硬件设备. 传统的虚拟化架构中, 所有的虚拟机都运行于同一虚拟机监控器中, 一旦虚拟机监控器受到攻击, 其上运行的所有虚拟机的安全都将受到威胁. 在 NOVA 中, 每个 VMM 上仅运行一个虚拟机, 不同 VMM 间具有良好的隔离性, 可以保证一个 VMM 受到攻击, 不会影响到其他 VMM 和虚拟机.

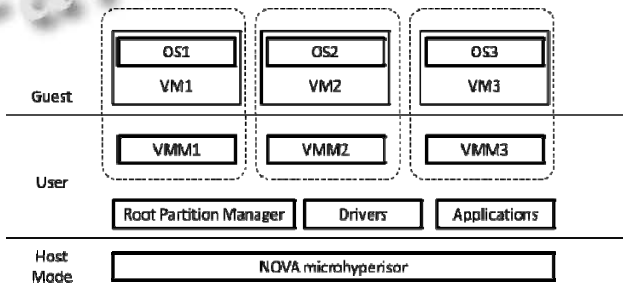

图 1 NOVA 总体架构

2.2 NOVA 虚拟机网络通信

NOVA 的用户级运行环境中为虚拟机提供必要功能的应用被称为服务(Service), 主要包括: 键盘服务、鼠标服务、硬盘读写服务、网络服务、时钟服务等. 各个服务运行于独立的地址空间, 彼此间相互隔离, 避免了一个服务崩溃导致整个虚拟化架构停止运行的情况产生.

NOVA 中服务间的数据传递是必须的, 如键盘服务需要和终端控制台通信才能通过键盘输入的方式来向终端控制台输入命令管理虚拟机, 硬盘读写服务也需要和驱动服务通信才可以将数据从硬盘读取到内存中.

在 NOVA 中服务间的通信是通过会话(Session)机制来实现的. 如图 2 所示, NOVA 中的会话包括客户端会话和服务端会话. NOVA 为客户端会话和服务端会话创建一块共享的数据空间作为服务间通信的数据缓冲区域. 会话对数据缓冲区域的读写操作采用典型的生产者-消费者模型. 当客户端会话发起数据传输请求时, 会通过服务端会话的 produce 方法向 Out Buffer 写入数据, 并设置相应的信号量, 服务端会话检测到信

号量的改变后, 通过自己的consumer_thread读取数据; 同理当服务端会话返回数据时, 会将数据写入 In buffer, 由客户端的 consumer_thread 读取.

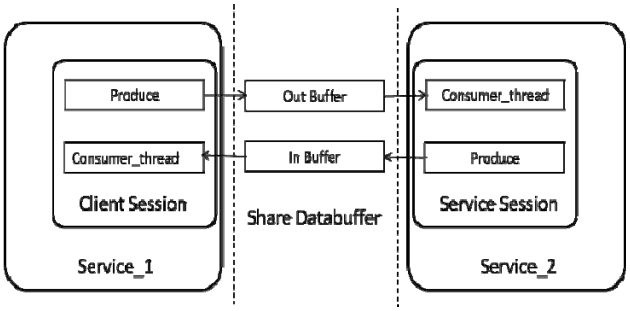

图 2　服务间通信

在 NOVA 虚拟机的通信由虚拟机操作系统、VMM、网络服务、网卡驱动的参与来完成. 如图 3 所示. 当虚拟机通过调用其操作系统 API 发出发送网络通信数据的请求时会引起 VM eixt, 虚拟机将 CPU 控制权交于 VMM. VMM 读取虚拟机网络通信请求, 并获得所需要发送数据包的内存地址和数据包大小, 并通过 Network Session 会话将所需发送数据写入数据缓冲中. 网卡驱动程序通过 Network Sessiondata 会话从数据缓冲中读取数据, 并通过计算机网卡将数据发送到网络中. 当计算机网卡接收到一段数据时会产生一个网卡 IO 中断. 网卡驱动程序从网卡中读取数据并将数据通过会话将数据传输到 VMM 中. VMM 接收到数据后会产生一个虚中断来通知虚拟机操作系统读取数据.

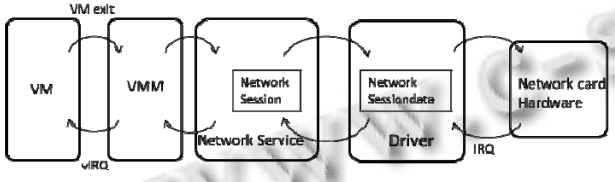

图 3　虚拟机网络通信流程

3　微内核虚拟机间通信加速方法设计

NOVA 中同一物理机上的不同虚拟机间的通信需要通过网卡驱动转发数据. 如图 4 所示, 虚拟机 VM1 发出通信请求, 需要发送的数据经过 VMM1 和 Network Service 到达网卡驱动来进行发送. 网卡发送的数据需要通过局域网路由器或 Internet 进行数据转发后回到网卡驱动, 引发网卡 IO 中断, 再经过 Network Service 和 VMM2 被 VM2 的读取. 可以看到采用这种方法数据传输路径较长, 性能开销较大, 因此导致虚拟机间通信效率较低.

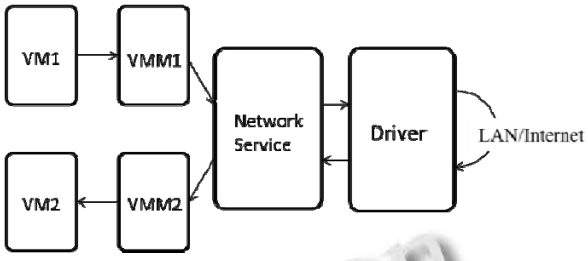

图 4　通过网卡转发数据的虚拟机间通信流程

针对同一物理机中不同虚拟机间的通信, 本文设计了一种虚拟机间通信加速方式, 缩短了同一物理机上不同虚拟机间通信的数据传输路径, 提高了通信效率.

3.1 通信加速方法架构设计

虚拟机通信数据按照通信双方是否在同一物理机中可以分为两类: 第一类是与其他物理机进行通信的数据, 本文中将其称为外部数据; 第二类是与同一物理机不同虚拟机通信的数据, 本文中将其称为内部数据. 本文所提出的通信加速方法针是对内部数据的传输进行加速, 因而需要包含两个关键步骤: 通信数据选择和通信数据转发. 通信数据选择阶段判断虚拟机通信数据的类别是否为内部数据; 通信数据转发阶段将内部通信数据传输到目标虚拟机中.

基于上述原因, 本文所设计的通信加速方法主要由通信数据选择模块和通信数据转发模块构成, 架构图如图 5 所示. 当虚拟机 VM1 发出通信请求后, 通信数据经由 VMM1 到达通信数据选择模块. 若通信数据选择模块的判别结果为此通信数据为外部数据, 则将通信数据交由网卡驱动处理; 若判别结果为内部数据, 且其通信的目标虚拟机为 VM2, 则通过通信数据转发模块将通信数据发送至与目标虚拟机 VM2 相对应的 Network Session 2. 数据到达 Network Session 2 后会经由 VMM2 被传输到目标虚拟机 VM2 中.

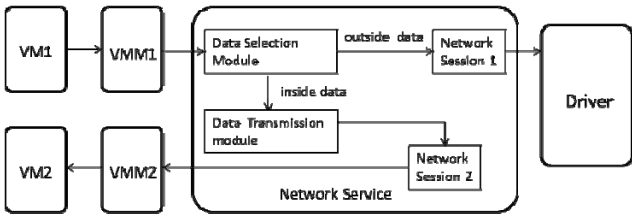

图 5　通信加速方法架构

此方案没有修改操作系统代码, 对操作系统及其上层应用来说是透明的, 用户无需对操作系统中的应用程序或动态库做更改就可以使用.

3.2 通信加速方法模块设计

虚拟机间的通信加速主要通过通信数据选择模块和通信数据加速模块来完成, 下面将详细阐述这两个模块的原理和处理流程.

3.2.1 通信数据选择模块

通信数据选择模块的工作原理是通过比较通信数据中所包含的目标的 MAC 是否属于本地虚拟机而实现的. 通信数据选择模块工作原理包括 MAC 地址获取和 MAC 地址匹配两部分.

在 NOVA 中 VMM 在用户创建虚拟机时被创建. 一个 VMM 被创建时, 它会通过创建一个对应的 Network Sesssion 和 Network SessionData 来建立与 Network Service 和网卡驱动的通信. 若通信成功建立, Network Service 会为 VMM 创建一个 MAC 地址. 通信数据选择模块会将 Network Service 为 VMM 生成的 MAC 地址储存在其 MAC List 中.

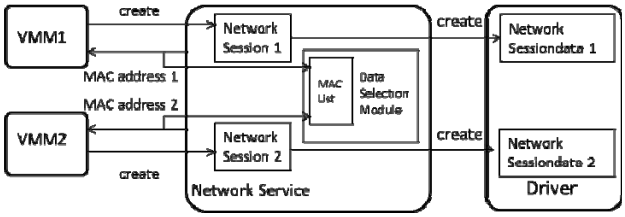

图 6 通信数据选择模块 MAC 地址获取

NOVA 中虚拟机间网络通信主要采用的是 TCP/IP 协议通过以太网进行通信, 数据传输以以太网帧为基本单位. 以太网帧的数据格式为 6Bytes 目的 MAC 地址、6Bytes 源 MAC 地址、2Bytes 类型字段, 其余为数据字段.

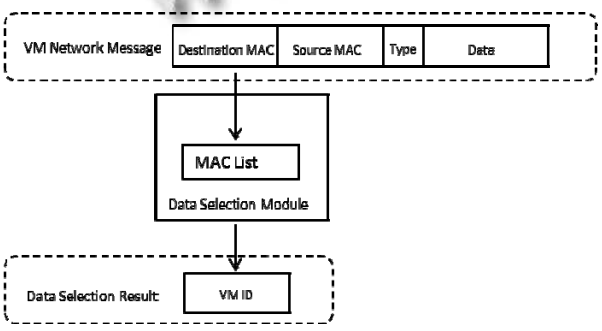

图 7 通信数据选择模块处理流程

通信数据选择模块运行流程如图 6 所示. 通信数据选择模块的输入为 VM 所发送的网络通信数据, 其数据格式为以太网帧; 通信数据选择模块的输出为场景判别结果, 数据格式为虚拟机 ID 号, 当 ID 为-1 时表示此通信数据为外部数据. 当虚拟机通信发起方所发送的数据经过虚拟机所在的 VMM 到达通信数据选择模块后, 通信数据选择模块读取以太网帧的目的 MAC 地址, 并与自身所储存的 MAC List 进行对比, 根据目的 MAC 是否存在于 MAC List 中及存在于 MAC List 中的位置返回对应的 VM ID.

3.2.2 通信数据转发模块

通信数据转发模块的主要工作原理是通过从数据选择模块得到的 VM ID 查到到对应的 Network Session, 并将通信数据写入对应的 Network Session 中.

在 NOVA 中每个 Service 都会维护一个记录所有与自身有通信的会话的列表 Session List, 通信数据转发模块获取 Session List 的网络通信会话子集 Network Session List. 通信数据转发模块处理流程如图 7 所示, 当输入通信场景判断结果后, 通信数据转发模块根据通信场景判断结果中的虚拟机 ID 查询对应的 Network Session 的地址, 通过调用对应的 Network Session 的 Produce 方法, 将虚拟机网络通信数据发送到目的虚拟机的 Network Session 中.

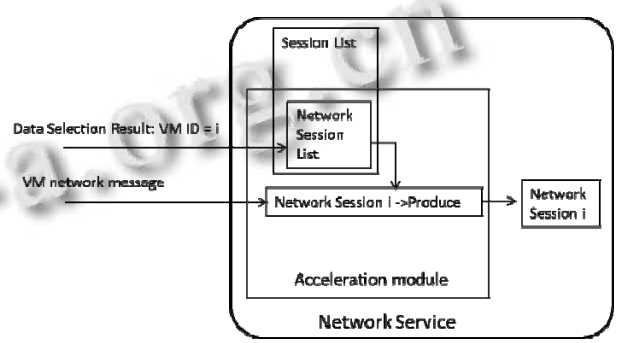

图 8 通信数据转发模块处理流程

4 实验及结果

4.1 实验配置

本实验运行的计算机处理器型号为 Inter(R) i5 处理器, 处理器核心数为 4, 主频为 2.5GHZ, 内存 4G, 运行 Ubuntu13.04 版本. NOVA 使用 qemu[13]提供硬件运行环境, qemu 模拟的网卡为 NE2K, qemu 与 Ubuntu 主机通信的网络模式为 TAP 模式. NOVA 中运行的虚拟机为 Linux 虚拟机, 内核版本为 3.16.6.

使用发包程序和接包程序完成数据的发送和接收. 发包程序输入参数为目标 IP、发送数据包大小、和数据发送持续时间, 通过调用 Linux socket API 向目标 IP 发送指定大小的数据包. 接包程序输入参数为接收数据包大小, 每接到一个数据包则程序的接包计数器加 1.

4.2 实验方案

虚拟机间通信加速方法在提升虚拟机传输内部通信数据速度的同时, 由于数据选择模块的性能开销, 也会对虚拟机传输外部通信数据时的速度造成损耗. 因此, 本实验分别设计实验方案测量虚拟机间通信速度的提升和虚拟机发送外部数据时的速度损耗.

为测量虚拟机间通信速度, 在 NOVA 中创建两台虚拟机, 分别为虚拟机 VM1 和虚拟机 VM2. VM1 中运行发包程序, VM2 中运行接包程序. 若发包程序的数据包大小为 a(Byte), 发送持续时为 t(s), 接收数据包数量为 n, 则计算虚拟机间通信速度 V(Mbps)计算方法如公式(1)所示

$$V = 8 \times a \times n / (t \times 1024^2) \qquad (1)$$

为测量虚拟机间通信加速方法对虚拟机发送外部数据时造成的速度损耗, 使用发包程序和接包程序分别测量 VM1 向本地 Ubuntu 系统中发送数据的通信速度和 VM1 接收本地 Ubuntu 系统所发来的数据的通信速度. 若不加入虚拟机间通信加速方法时 VM1 与 Ubuntu 的通信速度为 V_{before}, 加入虚拟机间通信加速方法后 VM1 与 Ubuntu 的通信速度为 V_{after}, 则通信加速方法损耗 Cost 计算方法如公式(2)所示.

$$Cost = (V_{before} - V_{after}) / V_{before} \qquad (2)$$

4.3 实验结果

在本文的实验中, 通信的数据包大小均采用从 64B 到 2KB 共六组实验数据, 数据包发送持续时间均为 1 分钟. 实验结果均采用十次实验结果的平均值. 实验结果如下所示.

表 1　虚拟机间通信速度(单位: Mbps)

数据包大小(Byte)	64	128	256	512	1024	2048
通信加速前	0.470	0.863	1.419	2.259	3.167	3.345

由图 9 可以看出, 当数据包较小时, 通信加速后与通信加速前差别不明显, 但当数据包大小大于 128Byte 时, 加入通信加速模块的虚拟机间通信速度相较于没有通信加速模块有了显著的提升.

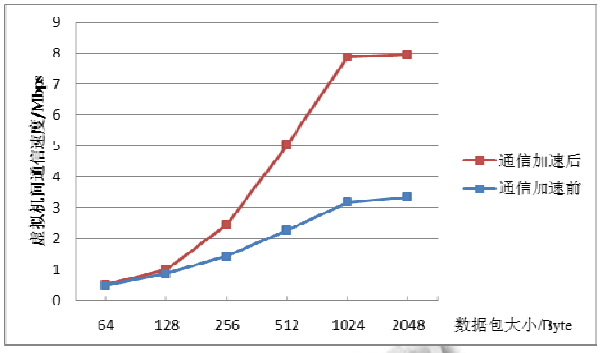

图 9　虚拟机间通信速度

表 2　虚拟机发送数据通信速度(单位: Mbps)

数据包大小(Byte)	64	128	256	512	1024	2048
通信加速前	0.751	1.550	2.872	5.919	11.627	13.214
通信加速后	0.740	1.480	2.790	5.901	11.621	12.901
速度损耗	0.015	0.045	0.029	0.003	0.001	0.024

表 3　虚拟机接收数据通信速度(单位: Mbps)

数据包大小(Byte)	64	128	256	512	1024	2048
通信加速前	3.445	6.782	12.852	21.579	30.630	29.057
通信加速后	3.439	6.689	12.766	21.115	29.175	28.375
速度损耗	0.002	0.014	0.007	0.022	0.047	0.023

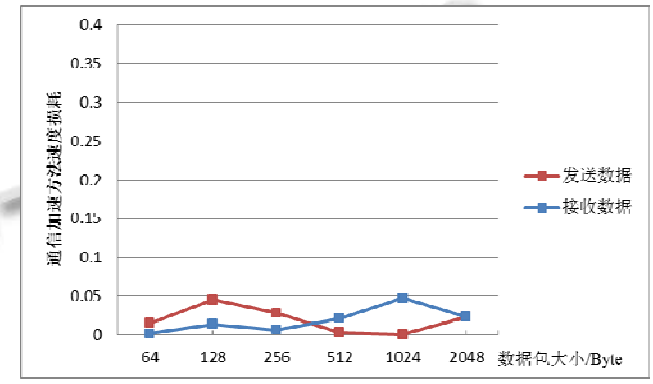

图 10　通信加速方法性能损耗

由图 10 可以看出, 加入通信加速方法后虚拟机与其他物理机通信的性能损失均小于 5%, 通信加速方法对虚拟机的通信性能影响较小.

5　结语

本文介绍了微内核虚拟机中虚拟机间通信目前所存在的问题, 分析了了其虚拟机间网络通信的原理和

流程. 针对目前微内核虚拟机中存在的同一物理机上不同虚拟机间通信路径较长, 性能开销较大的方法, 提出了一种高效的虚拟机间通信方案. 通过在微内核虚拟化架构的网络服务中加入通信加速模块, 使得虚拟机间的通信可以不经过网卡而直接在内存中完成. 实验表明, 当虚拟机间通信的数据包大小大于 128Byte 时, 加入通信加速模块后虚拟机通信效率有了较大提升, 且会对虚拟机与其他物理机的通信效率影响很小.

参考文献

1 Rosenblum M, Garfinkel T. Virtual machine monitors: Current technology and future trends. Computer, 2005, 38(5): 39–47.

2 Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 2003, 37(5): 164–177.

3 Velte A, Velte T. Microsoft Virtualization with Hyper-V. McGraw-Hill, Inc., 2009.

4 Ali Q, Kiriansky V, Simons J, et al. Performance evaluation of HPC benchmarks on VMware's ESXi server. Lecture Notes in Computer Science, 2012.

5 Sahoo J, Mohapatra S, Lath R. Virtualization: a survey on concepts, taxonomy and associated security issues. 2010 Second International Conference on Computer and Network Technology (ICCNT). IEEE. 2010. 222–226.

6 Mccune JM, Qu N, Li Y, et al. TrustVisor: efficient TCB reduction and attestation. 2010 IEEE Symposium on Security and Privacy (SP), 2010, 41(3):143–158.

7 Tews H. Formal methods in the Robin project: specification and verification of the Nova microhypervisor. Proc. of the C/C++ Verification Workshop., 2007.

8 Steinberg U, Kauer B. NOVA: a microhypervisor-based secure virtualization architecture. Proc. of the European Conference on Computer Systems. 2010. 209–222.

9 陈蔓莉.虚拟化技术在通信企业的应用.科技与企业, 2013, (19):126–126,129.

10 任怡,刘晓建,管剑波,等.一种支持在线迁移的虚拟机间快速通信方法.解放军理工大学学报(自然科学版),2012, (5):511–515.

11 Wang J, Wright KL, Gopalan K. XenLoop: a transparent high performance inter-vm network loopback. Proc. of the 17th International Symposium on High Performance Distributed Computing. ACM. 2008. 109–118.

12 Zhang X, McIntosh S, Rohatgi P, et al. XenSocket: a high-throughput interdomain transport for virtual machines. Middleware 2007. Springer Berlin Heidelberg. 2007. 184–203.

13 Bartholomew D. Qemu a multihostmultitargetemulator. Linux Journal, 2006, 2006(145): 3.