通用平台虚拟路由器转发性能测试与改进

葛敬国[1]，贺 鹏[2]，杨建华[2]，张建华[2]

(1. 中国科学院计算机网络信息中心 北京 海淀区 100190；2. 中国科学院计算技术研究所 北京 海淀区 100190)

【摘要】互联网的发展需要网络设备具备支持网络虚拟化和可扩展可编程两大特性。基于此出现了许多原型系统，虚拟路由器通过采用虚拟化技术和可扩展路由器软件来满足上述要求。但由于虚拟化本身所带来开销，虚拟路由器的数据包转发性能会受到影响。该文对不同的虚拟技术和可扩展路由软件进行介绍，搭建实验平台，在不同的虚拟环境下测试和评估了其数据包转发性能，并评估了几种常见的虚拟I/O加速技术。

关 键 词 数据包转发； 软件路由器； 虚拟I/O加速技术； 虚拟化； 虚拟路由器

中图分类号 TP393 文献标志码 A doi:10.3969/j.issn.1001-0548.2014.01.016

Evaluating and Optimizing the Forwarding Performance of Virtual Router Based on Commodity Hardware

GE Jing-guo[1], HE Peng[2], YANG Jian-hua[2], and ZHANG Jian-hua[2]

(1. Computer Network Information Center, Chinese Academy of Sciences Haidian Beijing 100190;

2. Institute of Computing Technology, Chinese Academy of Sciences Haidian Beijing 100190)

Abstract Virtualization and scalability & programmability are two key characteristics to be supported by network devices for the development of Internet. In recent years, researchers have proposed many prototypes based on these two requirements. The virtual router is one representative prototype, which realizes these two requirements by the virtualization technique and scalable routing software. However, due to the overhead of virtualization techniques, the forwarding performance of a virtual router is usually limited. This paper investigates different virtualization techniques and scalable routing software, establishes the test-bed and evaluates the forwarding performance under different virtualization techniques, and conducts the assessment on several typical virtual I/O acceleration techniques.

Key words packet forwarding; software router; virtual I/O acceleration technique; virtualization; virtual router

近几年来，网络虚拟化成为了一项被广泛关注的技术。虚拟路由器将一台物理路由设备虚拟成多台路由设备，多个虚拟设备并行处理不同的业务流量，并根据业务的特性提供相应的功能扩展，满足不同业务下的不同数据包处理策略的要求。许多国家和组织正在积极研究未来互联网的发展，提出了许多新的互联网架构，基于虚拟路由器构建的试验网可方便地进行未来网络体系结构的验证。

软件定义网络强调控制平面与数据平面的分离，使硬件标准化、设备软件化成为未来网络设备发展趋势。基于通用平台实现虚拟路由器广泛被关注，数据包转发性能是基于通用平台的虚拟路由器重要问题。文献[1]在充分挖掘单个物理设备的吞吐量潜力的基础上，采用集群方式，提出RouteBricks软件路由器。实验结果表明，使用4个通用服务器构建的RouteBricks集群，可以提供高达35 Gbps的吞吐率。文献[2]利用GPU的强大并行计算能力，研制出单台吞吐率可达39 Gbps的通用硬件路由器PacketShader。软件路由器的研究进展为虚拟路由器的设计提供了思路，即在通用平台上，使用可扩展的路由器软件，如Click[3]、XORP[4]等，以及虚拟化软件如Xen[5]、OpenVZ[6]等设计实现虚拟路由器。

网络带宽的激增，对路由器的吞吐率提出了非常高的要求。传统路由器对数据包处理简单，在数据平面和控制平面上做了充分的优化，提供了很高的吞吐率。虚拟路由器由于虚拟化技术的引入势必

收稿日期：2013－09－06；修回日期：2013－11－21

基金项目：国家973项目(2012CB315803)；国家科技支撑计划(2012BAH01B03)；国家863项目(2011AA01A101)；中科院先导专项项目(XDA01020304)

作者简介：葛敬国(1973－)，男，博士，副研究员，主要从事互联网体系结构与关键技术等方面的研究.

会对路由器I/O效率产生影响，对各种不同虚拟化技术所带来的性能开销进行评估是必要的。本文测试了几款代表性虚拟化软件，对其数据包转发性能进行了测量，并在此基础上评估了几种常见的I/O加速技术。

1 虚拟化软件路由器原理与实现

1.1 虚拟化软件路由器

虚拟路由器的典型框架如图1所示，虚拟路由器的架构由三大组件构成：数据平面、虚拟化软件、控制平面(路由器软件)。数据平面可以由专用硬件或者通用网卡构成；虚拟软件作为中间层(Hypervisor)，给不同虚拟机提供虚拟I/O通道。在虚拟机内部则可以运行路由器软件计算路由，或者用于承载实验网，进行新协议的实现，如OpenFlow[7]等。下面介绍虚拟化技术和路由器软件。

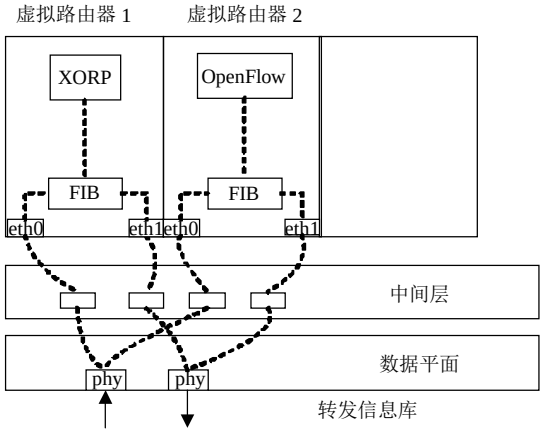

图1 虚拟路由器框架图

1.2 虚拟化实现

虚拟软件路由器所使用的虚拟化技术按照虚拟层次的不同，大致可以分为：

1) 全虚拟化(full virtualization)，又称本地虚拟化(native virtualization)。该技术在客户操作系统和底层硬件之间引入中间层Hypervisor，使客户操作系统无法察觉自身是运行在虚拟机上，多客户操作系统能够共享底层硬件。如当客户操作系统需要访问网卡时，Hypervisor将模拟实际网卡的行为，给客户操作系统产生一个收包中断，并将实际的数据包投递给客户操作系统。中间层技术的引入会带来I/O开销，如数据包的到达必须经过Hypervisor的转发才能投递给客户操作系统。近年来，随着虚拟化技术的普及，一些新的X86架构的CPU能够从硬件上支持全虚拟化，使全虚拟化环境下的客户操作系统能达到原生级别的运行速度，但是这些技术并没有解决虚拟化情况下I/O性能的问题。

2) 半虚拟化(para-virtualization)，又称准虚拟化技术。该技术同样使用Hypervisor来隔离底层硬件和操作系统，不同的是，它同时也修改客户操作系统的内核代码，使客户操作系统能够意识到自身运行在虚拟机上。该技术能够显著提升虚拟化技术的I/O性能，如当网卡收到一个数据包时，在原先的全虚拟化机制内，Hypervisor只能通过模拟硬件中断，将单个数据包传递到客户操作系统上；但在半虚拟化机制下，客户操作系统可以不使用网卡的原生驱动，而是采用I/O队列的方式，批量从Hypervisor获取数据包，极大地提升I/O性能。半虚拟化技术的局限在于不能应用于非开源的商业操作系统，如Windows等。但由于网络系统的主流操作系统都是Linux，这一局限对虚拟化路由器的设计没有影响。半虚拟化技术代表系统为Xen[5]、UML[9]等。

3) 操作系统级别虚拟化(operating system-level virtualization)。操作系统级别虚拟化与以上所有虚拟化都不相同。它并不虚拟硬件，而是利用操作系统所提供一些机制(如Linux中命名空间)，提供“容器”，将不同的进程放入“容器”内，并限制该组进程的CPU占用率和内存使用情况，达到虚拟化技术中资源的隔离与分配的目的。该虚拟化技术所有的虚拟机都是共享同一个内核的，也使得使用该技术的虚拟机都必须采用同一个系统。该虚拟化技术的代表软件为OpenVZ[6]、Linux-VServer[10]、LXC[11]等。由于并没有在硬件和客户操作系统之间引入中间层，该技术所带来的I/O开销非常小，接近原生操作系统的性能。其主要局限在于，使用该虚拟化技术不能运行多个异构的操作系统。

1.3 软件路由器介绍

1) Click[3]

Click是一个模块化的软件路由器，它将路由器设计的各个部分模块化(称为“元素”)，提供了一种灵活的可扩展的路由器软件套件。Click抽象出两种操作将不同元素连接起来，一种称为Push，即将一个数据包主动传递到下一个元素；一种称为Pull，即需要从下一个元素中获取一个数据包。Click提供了普通网卡的轮询驱动，能极大地提供普通网卡的转发性能。RouteBricks正是使用了Click的轮询驱动达到了很高的吞吐率。

2) XORP[4]

由于现有的商业路由器软件的封闭，导致了新协议的实现和研究没有测试平台，文献[4]设计并实现了XORP，旨在提供一个开放的、可扩展的路由器

软件平台，便于研究人员进行新路由协议的部署和实现。XORP已经被工业界所接纳，并被公司和教育机构广泛采纳。

XORP采用多进程机制，将数据包的交换和路由计算通过转发引擎抽象层(forward engine abstract)隔离，既保证了鲁棒性，也保证了性能。

2　不同虚拟平台的转发性能测试实验

当前，虚拟化软件种类繁多。本文根据虚拟化层次的不同，选取了KVM(全虚拟化)、Xen(半虚拟化)、OpenVZ(操作系统级别虚拟化)三款主流虚拟化软件，分别进行了数据包转发测试。

实验平台的各项参数如表1所示。

表1　实验平台参数

硬件名称	规格说明
CPU	2路四核，共八核
Intel Xeon L5420	每核 L1 Cache 32 K　16 K(指令)+ 16 K(数据) 四核两两共享L2Cache 6 M
内存	DDR2 667 MHz 16 GB
主板	SuperMicro X7DWU
网卡	Intel 82 571 EB 千兆，四口

2.1　实验设计

本文使用Spirent Testcenter作为测试仪表，采用RFC 2544标准测试方式测试了不同的虚拟软件的吞吐率，即以每秒帧转发率及该速率和理论最大转发率的比值作为性能指标。测试拓扑如图2所示，选取的包大小依次为64字节(最小包)、512字节(网络中的平均包大小)、1 518字节(最大包)。同时，根据上述实验结果，本文又设计了多虚拟机情况下的数据包吞吐率实验：将不同的虚拟机运行不同的网段中，测试其总体转发性能。

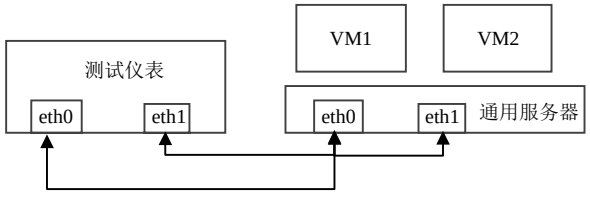

图2　转发流量拓扑

由于不同的虚拟软件的网络配置方式各有不同，为了测试的简便，只选取了桥接(bridge)模式的网络配置来进行以下对这些软件做一些简单的介绍，并分别说明不同虚拟软件的网络配置情况。

1) KVM，即内核虚拟机(Kernel virtual machine)，与主流虚拟化技术Xen的最大区别在于，它并没有重新实现一个Hypervisor，而是向内核添加一两个模块，将内核作为Hypervisor的一种虚拟化技术。KVM是全虚拟化技术，需要CPU对虚拟化有一定的支持。

2) Xen是剑桥大学开发了一套半虚拟化软件，它实现了一个精简的Hypervisor。Xen使用术语Domain来描述虚拟化环境。Xen具有两种Domain，Dom0和DomU。Dom0又称Driver Domain，运行在该Domain上的操作系统能够直接访问硬件。运行在DomU的操作系统则必须通过Dom0的代理。测试中Xen的网络配置采用桥接模式，分别测试了Dom0和DomU的转发性能。

3) OpenVZ是Swsoft公司专有软件Virtuozzo的开源版本，使用一种虚拟网络接口对的方式实现桥接。

2.2　实验结果与分析

2.2.1　Native Linux测试

首先对没有采用任何虚拟软件的系统平台进行了数据包转发测试，测试结果如表2所示。从测试结果可以看出，普通网卡对最小包的转发性能较差。这主要是由于普通网卡采取的中断方式所带来的系统开销过大导致的。一些研究表明[1-2]，采取轮询方式对小包转发能够带来很大的性能提升。

表2　Native Linux转发性能测试

帧长	吞吐率/(%)	吞吐率/f·s^{-1}	理论最大吞吐率/f·s^{-1}
64	26 171	778 898	2 976 190
512	100	469 924	469 924
1 518	100	162 548	162 548

图3～图6是分别对不同虚拟化软件的数据包转发性能测试后得出的结果。从结果中可以看出，全虚拟化带来的I/O性能开销大。这主要是由于数据包在内核中经过的路径太长，及上下文切换过多所导致的。而半虚拟化则由于其针对虚拟化环境做了一定的优化，性能上相对与全虚拟化有一定的改观，但是仍然与原生操作系统的性能有较大的差距，而操作系统级别的虚拟化则由于开销非常小，除小包转发外，在各项数据上都与原生操作系统非常接近。

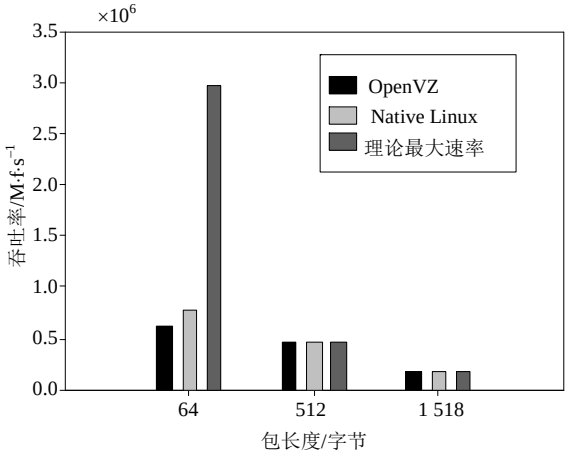

图3　OpenVZ性能测试

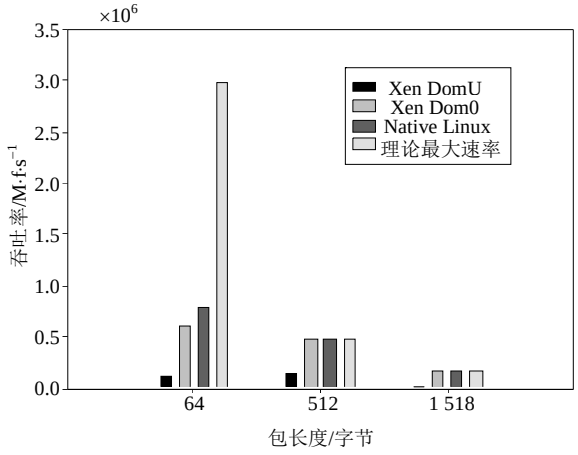

图4 Xen性能测试

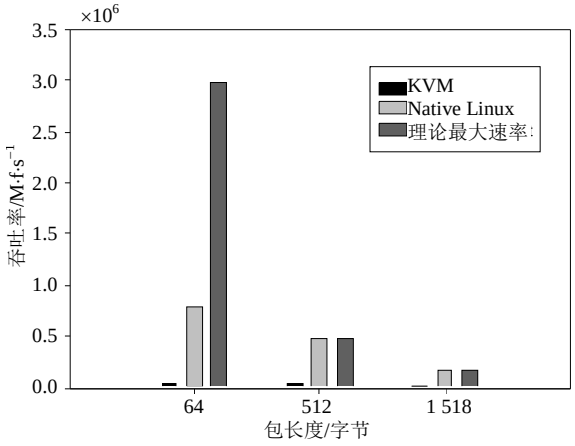

图5 KVM 性能测试

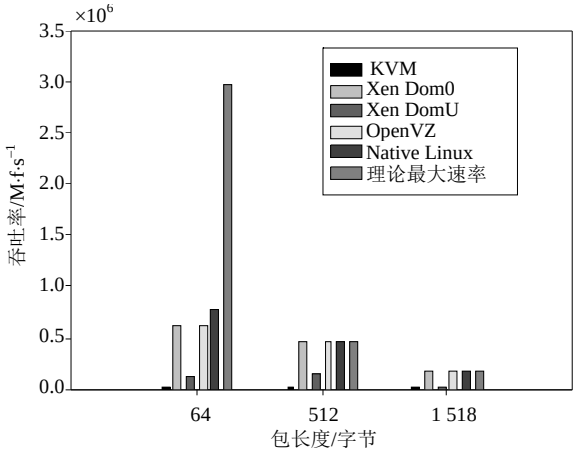

图6 几种不同技术性能对比

2.2.2 虚拟化技术性能测试结果与分析

根据实验结果表明，全虚拟化技术由于虚拟化开销太大，不适合作为虚拟路由器的虚拟化平台。半虚拟化和操作系统级别虚拟化各有优缺点。半虚拟化技术虽然在转发性能上与原生系统相比有很大的差距，但它提供了更好的隔离性和鲁棒性。根据实验结果，对Xen和OpenVZ进行了多虚拟机情况下的性能测试，结果分别如图7～图8所示。

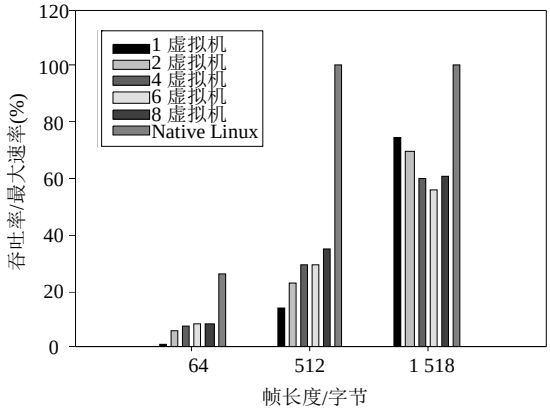

图7 Xen多虚拟机转发性能

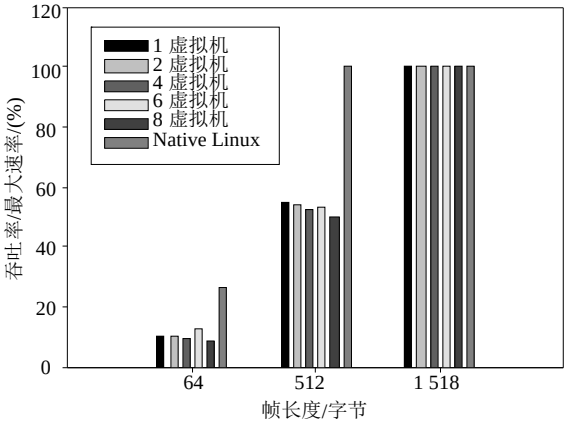

图8 OpenVZ多虚拟机转发性能

从实验结果得出，虚拟机数量的不同对于吞吐量的影响并不大。由此可见，提高吞吐率的途径在于缩短数据包的内核路径，给虚拟机直接访问设备的能力。

3 虚拟I/O加速技术

上述测试的实验结果可以得出，全虚拟化技术所带来的开销会对I/O操作的性能造成影响。研究人员对此做出了研究，设计出一些能够显著提升虚拟I/O性能的加速技术。

3.1 Virtio(半虚拟化I/O)[12]

Virtio技术旨在给Linux操作系统中越来越多的虚拟化软件提供一套标准化的半虚拟I/O接口。半虚拟化I/O能够让客户操作系统意识到自身运行在虚拟化环境中，并在这个基础上采用一定的优化措施，提升了客户操作系统的I/O性能。

3.2 Intel VT-d技术[13]与SR-IOV技术[14]

尽管半虚拟化I/O技术能够提升客户操作系统的I/O性能，但仍然存在上下文开销过高等问题。解决问题的关键在于取消Hypervisor的代理，让客户操

作系统能够直接访问到物理设备。具体说来，直接访问屋里设备可以划分为两个问题：1) 对客户操作系统的设备缓冲区直接DMA访问(DMA remapping)。2) 中断重映射。Intel VT-d技术(Intel virtualization technology for directed I/O)解决了以上两个问题。全虚拟化I/O需要由Hypervisor代理，客户操作系统才能访问底层硬件，由于DMA Remapping的存在，底层硬件能够直接将数据写到客户操作系统的缓冲区内，使I/O性能接近原生操作系统的性能。

VT-d技术提供了数据由底层硬件到客户操作系统的直接通路，SR-IOV(single-root I/O virtualization)则提供了将单个设备虚拟成多个设备的方法。SR-IOV为每个虚拟机提供独立内存空间、中断和DMA(direct memory access)流。SR-IOV引入了两个新的功能类型：物理功能(physical functions, PFS)和虚拟功能(virtual functions，VFs)。

1) 物理功能是一些支持 SR-IOV 扩展功能的PCIe 功能，被用于配置和管理 SR-IOV 功能特性。

2) 虚拟功能是一些精简的PCIe功能，包括数据迁移必需的资源，及经过谨慎精简的配置资源集。

SR-IOV架构的设计允许一个I/O设备支持多个虚拟功能，从而提供了一种不需要软件模拟就可以共享I/O设备和I/O端口的物理功能的方法。

3.3 PCI Passthrough技术[15]

PCI Passthrough技术是Xen开发人员开发的客户虚拟操作系统直接访问设备的技术。主要原理是利用软件模拟IOMMU，使客户操作系统能越过Hypervisor访问设备。PCI Passthrough优势在于，VT-d是一种从硬件上支持IOMMU的技术，尚未广泛普及。而PCI Passthrough则从软件层面支持了IOMMU，可为不支持VT-d的系统进行I/O性能优化。

3.4 软件I/O加速方法

1) RouteBricks[1]设计经验

RouteBricks主要使用了3种手段优化了单个路由器的转发性能：① 使用轮询方式代替传统硬件的中断方式，极大地提高了通用硬件对64字节小包的转发性能。② 采用批处理的方式。每次总线事务(bus transcation)都传递多个数据包的地址，充分利用了PCIE的总线带宽。③ 利用网卡的多队列特性，将不同队列与不同核进行绑定，极大地提升了数据处理的并行性。使用以上手段，可以使单台服务器的64字节小包转发新能达到9.77 Gb/s。

2) PacketShader[2]设计经验

PacketShader对Linux的内核协议栈进行了优化，将内核中sk_buff结构体由原来的208字节减少为8字节，采用批处理的方式极大地提升了数据包处理性能。实验表明，采用这种优化手段可以将数据包吞吐量提升到10.5 Gb/s，性能提升13.5倍。PacketShader同时也利用了多核多队列的特性，并且考虑到其采用的架构为NUMA(非一致性内存访问)，针对数据包的分发均衡性和提升数据局部性做了精心优化，使其整体转发性能达到40 Gbps。

3.5 虚拟I/O加速技术性能评估

本文对VT-d技术、半虚拟化I/O技术、PCI Passthrough技术所带来的I/O性能提升进行评估，使用了RFC2544标准吞吐率测试方法进行了测试。此外，对半虚拟化I/O还进行了IPerf[15]带宽测试。由于Xen不支持Virtio，本文所使用的实验平台上也无法为Xen打开VT-d的技术支持，最终在KVM上实验了VT-d技术和Virtio的技术，在Xen上测试了PCI Passthrough技术。

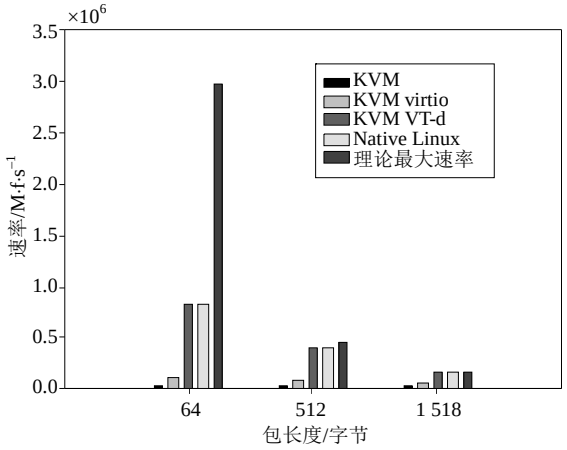

图9　KVM I/O加速

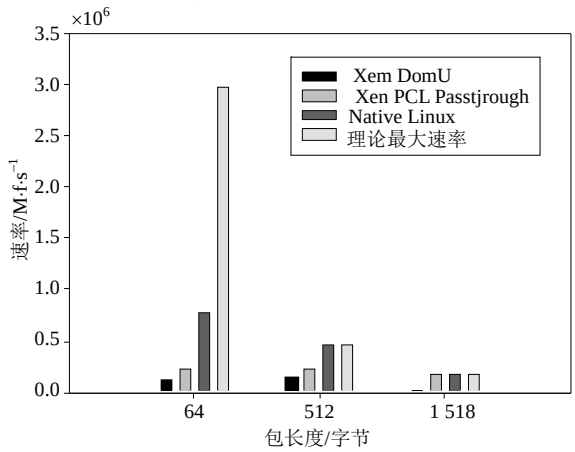

图10 PCI Passthrough软件模拟

表3 KVM半虚拟化I/O的IPerf带宽测试

	KVM		KVM virtio	
	单向/Mb·s^{-1}	双向/Mb·s^{-1}	单向/Mb·s^{-1}	双向/Mb·s^{-1}
带宽	210	117 332	749	555 517

实验结果表明，半虚拟化I/O技术对于客户操作系统确实能提高TCP传输的带宽，但是对于数据包转发，其他的性能提升则是有限的。VT-d技术则给客户操作系统直接访问设备硬件的能力，能够获得接近原生操作系统的转发性能，将虚拟化所带来的开销降至最低。PCI Passthrough技术由于软件模拟的开销，所提供的性能则在两者之间。

4 结 束 语

本文围绕虚拟路由器这一概念分别探讨了几种虚拟化技术、与之相关的I/O加速技术以及业界广泛认可几种可扩展路由器软件，并在此基础上，测量几款具有代表意义的虚拟化软件的转发性能，对虚拟路由器的研究和设计具有启发意义。实验结果表明，操作系统级别的虚拟化技术所引入的额外开销最小，然而该项技术由于其虚拟化程度较低，使用起来有一定的局限性。采用VT-d等硬件支持的虚拟化技术能够使运行在全虚拟化环境下的客户操作系统获得原生的I/O效率，同时又保持了虚拟化层次高的优势。虽然现有的虚拟路由器的设计均是采用OpenVZ等轻量级虚拟化技术，但可以预期，VT-d和SR-IOV技术将会对虚拟路由器的设计带来较大的影响。

在目前的研究中，并没有使用通用平台搭建的虚拟路由器，大部分已有研究均使用专用硬件来降低虚拟化开销，并采用OpenVZ等轻量级虚拟系统来进一步降低虚拟开销。然而，由于SR-IOV等技术的出现，虚拟化开销已经被降至最低，因此一些软件路由器提高I/O性能的经验可以完全用来借鉴和设计虚拟路由器。未来会出现使用SR-IOV技术的软件虚拟路由器的研究。

参 考 文 献

[1] DOBRESCU M, EGI N, ARGYRAKI K, et al. RouteBricks: Exploiting parallelism to scale software routers[C]// Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. [S.l.]: ACM, 2009: 15-28.

[2] HAN S, JANG K, PARK K S, et al. PacketShader: a GPU-accelerated software router[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 195-206.

[3] KOHLER E, MORRIS R, CHEN B, et al. The Click modular router[J]. ACM Transactions on Computer Systems, 2000, 18(3): 263-297.

[4] HANDLEY M, HODSON O, KOHLER E. XORP: An open platform for network research[J]. ACM SIGCOMM Computer Communication Review, 2003, 33(1): 53-57.

[5] BARHAM P, DRAGOVIC B, FRASER K, et al. Xen and the art of virtualization[J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 164-177.

[6] KOLYSHKIN K. Virtualization in linux[J/OL]. [2013-05-12]. http://wiki.openvz.org/Main_Page.

[7] MCKEOWN N, ANDERSON T, BAlAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networds[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[8] KIVITY A, KAMAY Y, LAOR D, et al. The Linux virtual machine monitor[C]//Proceedings of the Linux Symposium. Ottawa: [s.n.], 2007: 225-230.

[9] DIKE J. User mode Linux UML[J/OL]. [2013-05-12]. http://user-mode-linux. sourceforge.net/.

[10] SOLTESZ S, PÖTZL H, FIUCZYNSKI M E, et al. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors[J]. ACM SIGOPS Operating Systems Review, 2007, 41(3): 275-287.

[11] LAADAN O, HALLYN S E. Linux-CR: Transparent application checkpoint-restart in Linux[C]//Proc of the 12th Annual Linux Symposium. Ottawa: [s.n.], 2010.

[12] RUSSELL R. Virtio: towards a de-facto standard for virtual I/O devices[J]. ACM SIGOPS Operating Systems Review, 2008, 42(5): 95-103.

[13] BURGER T W. Intel VT-d technology[EB/OL]. [2013-05-12]. http://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices.

[14] DONG Y, YU Z, ROSE G. SR-IOV networking in Xen: Architecture, design and implementation[C]//Workshop on I/O Virtualization. Berkeley, USA: USENIX, 2008.

[15] BSc Y S. Xen VGA passthrough[J/OL]. [2013-05-12]. http://wiki.xen.org/wiki/ Xen_PCI_Passthrough.

[16] TIRUMALA A, QIN F, DUGAN J, et al. Iperf: the TCP/UDP bandwidth measurement tool[J/OL]. [2013-05-12]. http://dastnlanr. net/Projects.

编 辑 黄 莘