·云计算专题· 文章编号:1000-3428(2017)01-0001-07 文献标志码:A 中图分类号:TP316

I/O 密集型虚拟机的域间通信优化方法

郭松辉[1,2],李清宝[1,2],孙瑞辰[1],孙 磊[1]

(1.解放军信息工程大学 密码工程学院,郑州 450001;2.数学工程与先进计算国家重点实验室,郑州 450001)

摘 要: I/O 密集型虚拟机需要频繁地进行域间通信,为解决现有虚拟机域间通信效率低、延迟大的问题,提出一种基于双环形缓冲区的用户域与驱动域域间通信优化方法。在用户域中建立与驱动域共享的双环形缓冲区,由虚拟机监控器依据 I/O 任务表对驱动域的访问权限进行控制,减少处理器模式切换和内存映射开销。实验结果表明,与原虚拟机域间通信机制相比,使用该优化方法后的域间通信机制具有更高的吞吐率和更低的延迟,大幅提高了用户域与驱动域的域间通信性能。

关键词: I/O 密集型虚拟机;域间通信;双环形缓冲区;共享内存;授权表

中文引用格式:郭松辉,李清宝,孙瑞辰,等. I/O 密集型虚拟机的域间通信优化方法[J]. 计算机工程,2017,43(1):1-7.

英文引用格式: Guo Songhui,Li Qingbao,Sun Ruichen,et al. Inter-domain Communication Optimization Method of I/O Intensive Virtual Machines[J]. Computer Engineering,2017,43(1):1-7.

Inter-domain Communication Optimization Method of I/O Intensive Virtual Machines

GUO Songhui[1,2],LI Qingbao[1,2],SUN Ruichen[1],SUN Lei[1]

(1. Cryptography Engineering Institute,PLA Information Engineering University,Zhengzhou 450001,China;
2. State Key Laboratory of Mathematical Engineering and Advanced Computing,Zhengzhou 450001,China)

【Abstract】 The virtual machines which are I/O intensive need to communicate with each other frequently. In order to solve the problems of low efficiency and big latency in inter-domain communication of virtual machines,an inter-domain communiaction optimization method between driver domain and user domain based on Double Ring Buffer(DRB) is proposed. Two ring buffers in user domain are built to be shared with a driver domain,and the access rights of the driver domain is controlled by the virtual machines monitor according to I/O task tables,which reduces the overhead of processor mode switching and memory mapping. Experimental results show that the inter-domain communication mechanism after using the optimization method has higher throughput and lower delay compared with original inter-domain virtual machine communication mechanism,and it greatly improves the inter-domain communication performance of the user domain and drive domain.

【Key words】 I/O intensive virtual machine;inter-domain communication;Double Ring Buffer(DRB);shared memory;grant table

DOI:10.3969/j.issn.1000-3428.2017.01.001

0 概述

随着云计算技术的迅速发展,云计算中心成为云计算基础设施的一个重要组成部分。云计算中心为大量面向服务的应用提供专用、灵活的隔离执行环境,如:Web 服务器,邮件服务器,DNS 服务器等。云计算中心通过虚拟化技术,将大量分散的计算资源聚合到少量物理主机中,减少了部署成本,降低了能源消耗[1]。随着硬件处理能力的提升和虚拟化技术的发展,处理器虚拟化和内存虚拟化技术已经取得了较大的进步,虚拟化后的性能已接近非虚拟化环境下的性能[2]。但是,I/O 虚拟化技术的发展滞

基金项目:国家“973”计划项目(2011CB311801)。

作者简介:郭松辉(1979—),男,博士研究生,主研方向为云计算、虚拟化技术;李清宝,教授、博士、博士生导师;孙瑞辰,硕士研究生;孙 磊,副研究员、博士。

收稿日期:2015-12-22 **修回日期:**2016-02-04 **E-mail:**guo_song_hui@163.com

后于处理器和内存虚拟化技术的发展,导致虚拟系统的整体性能提升有限,尤其在 I/O 密集型(I/O Intensive)虚拟机中,由于应用程序需要频繁读写虚拟化 I/O 设备,低效率的虚拟化 I/O 设备成为系统的性能瓶颈。

虚拟机监控器(Hypervisor)是运行在物理服务器和操作系统之间的中间软件层,通过 Hypervisor 能够实现多个操作系统对物理硬件的虚拟化共享。当前有多种支持 I/O 虚拟化技术的 Hypervisor,Xen[3]是其中具有代表性的一种。基于软件技术的 I/O 虚拟化存在 2 个影响 I/O 性能的问题:1)上层虚拟域对 I/O 资源的竞争,引起吞吐率降低和处理延迟增加;2)处理器在特权模式和非特权模式之间的频繁切换,产生多次上下文保存操作。为提高I/O 设备虚拟化性能,学术界和工业界进行了大量的研究,通过为 I/O 设备和主机平台增加硬件支持,提升 I/O 设备虚拟化性能[4-5],但基于硬件辅助的 I/O 虚拟化技术牺牲了虚拟化技术中的安全隔离和资源共享功能,同时 Hypervisor 必须包含大量设备驱动,增加了 Hypervisor 复杂性,系统可维护性降低,并且由于需要更多硬件(IOMMU[6],SR-IOV[7]等)支持,应用成本增加,虚拟机迁移能力降低。因此,硬件辅助虚拟化技术在商业应用中受到限制。

在基于软件技术实现的 I/O 虚拟化时,由于存在用户域与驱动域之间的通信延迟,导致其性能比硬件辅助虚拟化技术的性能差。通过优化用户域与驱动域之间的通信,能够有效提高基于软件技术的 I/O 虚拟化性能。在虚拟机 Xen 中,基于内存共享的域间通信优化技术是目前学者广泛研究的提高域间通信性能的一类软件优化方法。虚拟机 Xen 通过授权表机制实现虚拟域间的数据共享,允许源域控制其内存页面被一个特定的目标域访问,其首要用法是将用户域内存的 I/O 缓冲区共享给驱动域,实现I/O数据传输。XenSocket[8]是一种基于 Unix 系统的 Unix Domain Socket 机制,实现虚拟机域间通信。XenLoop[9]以共享内存 IPC(Inter Processes Communication)的方式实现共享内存的域间通信,但 XenSocket,XenLoop 在通信过程中都存在多次超级调用,时间开销较大。文献[10]基于 KVM 共享内存建立虚拟域之间的直接网络通道,并对缓冲区打包处理,但用户域与驱动域之间仍然需要频繁的中断通知。文献[11]在 Hypervisor 中分配环形队列,然后映射给通信双方虚拟域,由双方域对队列进行自由访问,通信性能得到了优化,但由于不对双方的访问权限进行细粒度控制,因此在 I/O 密集型虚拟机中有多个用户域共享驱动域,存在非授权域间信息泄露的风险。文献[12]通过预先分配缓冲区,当需要读写操作时,从缓冲区分配相应的区域,减少了超级调用的次数,但由于发送与接收操作共用缓冲区,存在缓冲区分配时的竞争问题。

本文基于虚拟机监控器 Xen,提出一种新的基于双环形缓冲区(Double Ring Buffer,DRB)的域间通信优化方法,在用户域初始化时预先为发送和接收操作分别分配环形缓冲区,并对虚拟域的共享授权调用进行修改,支持对共享缓冲区的使用。在执行 I/O 传输时,虚拟域不再需要执行缓冲区分配和释放等超级调用,驱动域能够在 Hypervisor 的控制下,根据机器地址对用户域缓冲区直接进行读写访问。同时,通过采用伪缓冲区进行辅助访问的方法,实现域间通信时的连续读写,避免了因数据不连续而需要进行额外操作的问题,进一步提高通信效率。因此,基于 DRB 优化方法,能够有效提高 I/O 密集型虚拟机的域间通信效率。Hypervisor 能够对驱动域的访问地址空间及权限进行控制,从而实现多个用户域通过同一驱动域执行 I/O 传输时用户域数据的安全隔离。

1 相关知识

1.1 Xen I/O 虚拟化技术

Xen 是一种在科研领域和云计算应用领域广泛使用的虚拟机监控器,同时支持全虚拟化和半虚拟化 2 种工作模式[13],当采用半虚拟化模式时,虚拟域能够获得更高的 I/O 处理能力[14]。Xen 的 I/O 虚拟化结构如图 1 所示。

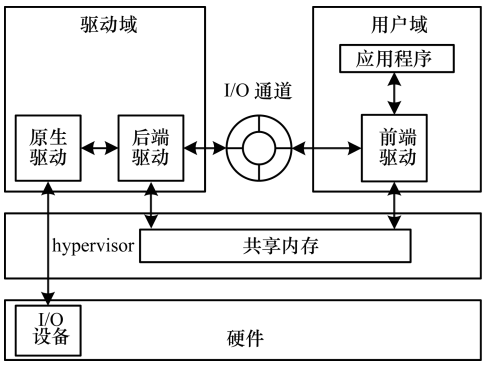

图 1 Xen I/O 虚拟化结构

Xen 在半虚拟化工作模式下采用前后端分离的虚拟设备驱动模型,虚拟设备驱动包含三部分:用户

域(User Domain,DomU)中的前端驱动,能够访问真实 I/O 设备的驱动域(Domain 0,Dom0)中的后端驱动和原生驱动。前端驱动为用户域提供虚拟设备,将用户域的 I/O 请求转发给后端驱动,后端驱动通过原生驱动控制硬件设备完成 I/O 操作。在整个 I/O设备访问的过程中,后端设备驱动是所有前端访问真实物理设备的代理。DomU 与 Dom0 通过 I/O 环传递 I/O 请求,利用共享内存传递 I/O 数据,使用事件通道实现域间异步事件通知机制。

当用户域中的上层应用需要执行 I/O 发送任务时,前端驱动将请求信息写入 I/O 环,通过事件通道通知后端驱动;后端驱动接收到 I/O 请求后,将请求提交给原生驱动。当硬件设备完成 I/O 请求后,后端驱动通知前端向用户域报告 I/O 操作完成。当执行 I/O 接收任务时,硬件设备在接收到数据包后,向 Hypervisor 产生中断,由 Hypervisor 通知驱动域进行接收,采用与发送相反的流程完成接收操作。

1.2　Xen 内存授权表机制

在基于 Xen 的 I/O 虚拟化架构中,驱动域需要具有访问用户域内存中 I/O 缓冲区的权限,以读取待发送 I/O 数据包。为满足这种访问需求,半虚拟化架构采用授权机制来共享用户域内存中的 I/O 缓冲区。授权机制允许驱动域以安全的方式访问用户 I/O 缓冲区,用户域对共享给驱动域的内存页面进行限制,只有这些受控制的缓冲区才能被用于I/O 操作。

Xen 为每个虚拟域建立授权表,基于授权表共享用户域内存,授权表中的条目由授权引用(Grant References,GR)标识。用户域通过 2 步共享内存页面给驱动域。

1)用户域为其待共享的内存页面分配一个授权引用。授权引用指向授权表中的一项,该授权引用在用户域与 Hypervisor 之间共享。授权表项包含共享内存页面地址、驱动域 ID 和访问许可。用户域使用简单的内存写操作填充授权表项,然后传递授权引用给驱动域。

2)驱动域使用授权引用来访问共享内存。驱动域通过超级调用把授权引用作为一个参数传递给 Hypervisor。Hypervisor 首先验证授权引用是否有效,然后读取用户域的授权表项,检验驱动域是否是用户域共享内存页面的目的域,并检验共享的内存页面是否是该用户域的内存页面。

如果这些检验通过,则 Hypervisor 锁定内存页面,映射页面到驱动域地址空间。锁定页面操作是为了防止在驱动域操作内存页面期间,用户域改变该页面的共享属性,实现了驱动域对共享内存页面的安全访问。

当 I/O 操作完成后,驱动域发出超级调用来解除对用户域页面的映射,然后通知用户域 I/O 操作完成并可撤销对相应页面的授权。用户域发出另一个超级调用,Hypervisor 首先从驱动域地址空间内解除页面映射,然后解除对共享内存页面的锁定。随后用户域通过对授权表项进行简单的写操作使其无效,撤销对内存页面的授权引用。共享内存页面的流程如图 2 所示。因此,对于每个授权的申请、撤销和相应的 I/O 操作,都会导致频繁的超级调用和上下文切换,对内存共享造成严重的性能影响。

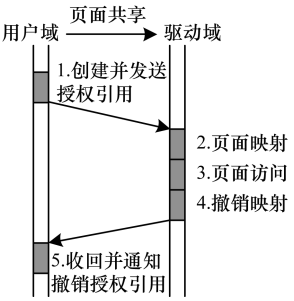

图 2　内存页面共享流程

2　基于 DRB 的 I/O 任务域间通信

频繁的超级调用是造成 I/O 虚拟化性能低的主要原因,减少每次页面映射中超级调用的使用次数能够使 I/O 虚拟化的性能得到提升,而虚拟域中通常会有多个线程或进程需要执行 I/O 收发任务,因此,本文提出一种基于双环形缓冲区的 I/O 任务域间通信优化机制。在该提出机制中设计并使用了一种新的域间通信模型,主要由接收环、发送环、接收任务表和发送任务表组成。本文提出机制在进行域间数据传递时,不需要调用申请内存分配、释放内存等超级调用,通过减少超级调用次数达到提升 I/O 虚拟化性能的目的,同时,该机制使用了环形缓冲区机制,能够实现对多个任务聚合后再批量处理,进一步减少域间通信开销,新模型的结构如图 3所示。

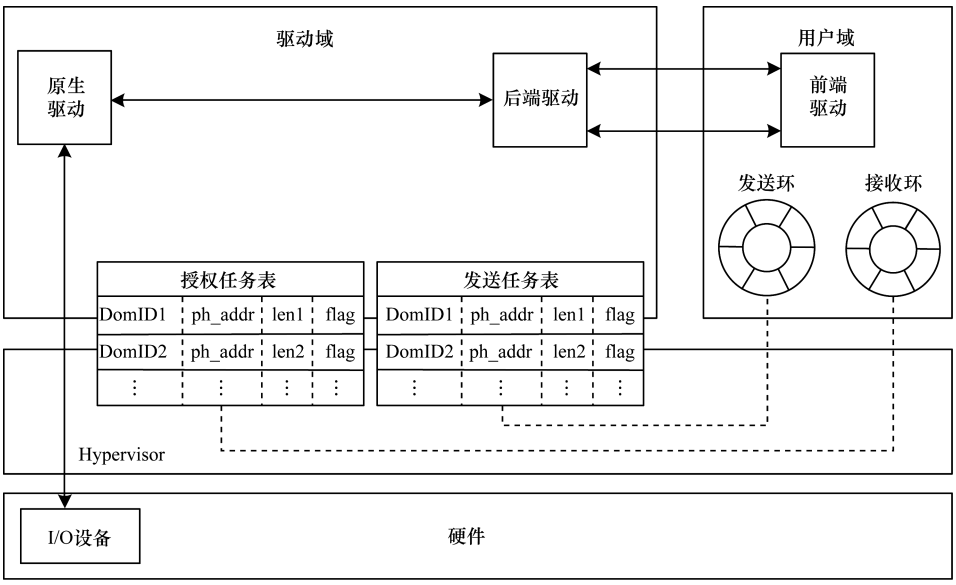

图 3 基于双环形缓冲区的 I/O 任务域间通信优化结构

当执行 I/O 发送任务时,用户域首先将数据写入发送环,然后通过超级调用通知 Hypervisor 根据任务在环形缓冲区中的存放信息更新 I/O 任务表;当用户域通过超级调用撤销授权时,利用超级调用通知 Hypervisor 从驱动域地址空间中解除对共享页面的锁定。因为存储环形缓冲区的共享页已经映射到驱动域地址空间,驱动域可以根据物理地址对发送环直接访问,能够从 I/O 任务表中读取驱动域访问的共享页面的机器地址空间,所以,驱动域执行发送任务的开销较低。执行接收任务的过程与发送任务相似,由驱动域将数据写入用户域接收环,完成接收操作。I/O 任务环和 I/O 任务表是本文机制的 2 个关键组成部分,下面对其工作原理进行分析。

2.1 I/O 任务环

文献[15]的虚拟域间基于单个缓冲区进行数据共享,需要使用互斥访问机制进行同步,防止共享数据发生不可预测的修改或读取,但是互斥操作会带来额外的系统开销。为解决该问题,本文设计了一种基于双环形缓冲区的域间数据传递方式。环形缓冲区在一个域进行写入操作,而在另一个域进行读取操作,通过对缓冲区的读写访问进行控制,使缓冲区对虚拟域呈现为环形空间结构。DRB 在用户域初始化时创建环形缓冲区,由用户域进行维护和管理,其结构如图 4 所示。双环形缓冲区采用“生产者-消费者”模型同步数据写入和读出操作,实现用户域与驱动域的域间数据共享,降低两者的耦合程度。在执行发送任务时,用户域作为请求生产者,在发送环中写入新任务;驱动域作为请求消费者,发送 I/O 任务并移除这些请求。执行接收任务时驱动域为请求生产者,用户域为请求消费者。

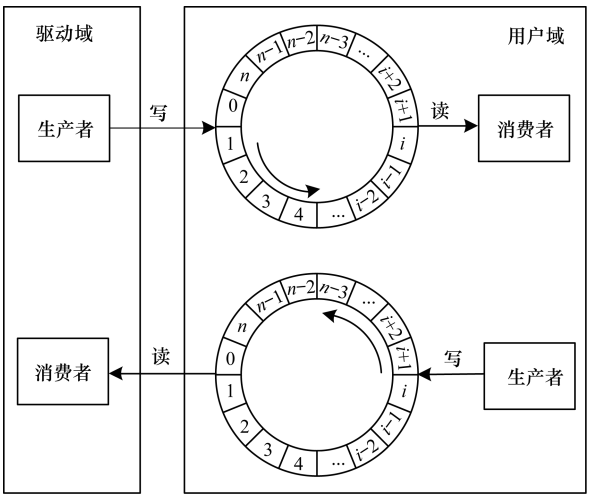

图 4 用户域与驱动域的数据传递方式

环形缓冲区在用户域生命周期内始终有效,用户域中执行 I/O 任务时,通过缓冲区与驱动域共享数据。本文机制改写了用户域超级调用函数,当用户发出授权用于发送 I/O 任务数据到驱动域时,并不真正地陷入到 Hypervisor 中申请相应的缓冲区,而是将缓冲区的当前写指针提交给应用,并根据任务数据包长度填充结构体中的任务长度,移动写指针到新的位置。当用户域在发送环中写入一个或多个任务数据后,授权驱动域访问,此时 Hypervisor 更新任务表中的相应项,并控制驱动域只能对授权的内存空间进行访问。在完成 I/O 发送任务后,驱动域通知用户域撤销对缓冲区的授权,从而将该缓冲区用作其他任务的收发。在执行 I/O 接收任务时,驱动域仍然使用由用户域提供的环形缓冲区,其调

用机制与发送任务时相同。执行 I/O 发送和接收任务的流程如图 5 所示。通过本文机制,用户域仅利用超级调用修改任务表中对应的标识位,撤销已完成 I/O 任务对环形缓冲区的占用,回收该部分缓冲区,而不执行原始的页面共享流程中复杂的解除映射操作,同时用户域不需要在每次 I/O 操作时都进行授权/撤销超级调用和映射/解除映射操作,减少了多个 I/O 任务的相同共享操作。

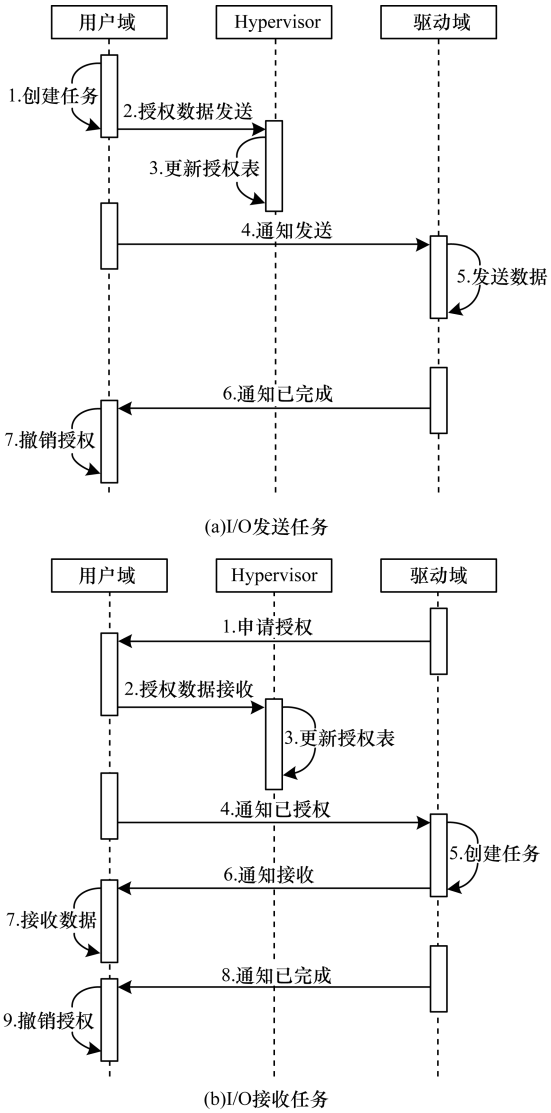

图 5　I/O 任务处理流程

另外,在虚拟机中,虚拟域的缓冲区地址是虚拟地址,虚拟地址通过“虚拟地址→物理地址→机器地址”[13]2 次映射到物理内存,在物理内存中并不一定连续。因此,在用户域初始化时,只有分配机器地址连续的存储空间作为共享缓冲区,才能在不同的虚拟域地址空间内对其进行连续的读写。指向环形缓冲区数据存储区域的指针在到达缓冲区所在机器地址末尾时,会返回到缓冲区起始地址。如图 6(a)环形缓冲区基本模型所示,当向缓冲区末尾写入数据时,如果所需空间的大小超过了当前写指针至缓冲区物理地址末尾的长度,需要将数据分为两部分:一部分存放在当前写指针至缓冲区末尾的空闲区域;另一部分则会存放到缓冲区起始地址开始的一段区域,导致数据在物理地址上不连续。

为此,本文引入伪缓冲区来处理这种情况。引入伪缓冲区的模型如图 6(b)所示,环形缓冲区占用指针 pStart 和 pEnd 指向的地址之间的一段区域,用于存放域间共享的数据。在使用环形缓冲区共享数据实现域间通信时,回收已经完成数据共享的缓冲区,为这部分回收的缓冲区维护一对指针 pSeudoStart 和 pSeudoEnd,用于指向缓冲区在使用过程中从起始地址处开始的可用区域,将其命名为伪缓冲区。当生产者往缓冲区写入数据时,首先判断 pWrite 至 pEnd 之间的区域是否能够完整存放待写入的区域,如果空间不够,则使用 pSeudoStart 至 pSeudoEnd 之间的区域存放,并修改各指针指向的地址。如果 pSeudoStart 至 pSeudoEnd 之间区域的空间不足,则等待该区域回收至足够大的空间时再写入。通过伪缓冲区的管理方法,为用户域与驱动域双方维护了连续的缓冲区,避免写入数据到达缓冲区末尾时,重新从缓冲区的首部开始写入的问题。

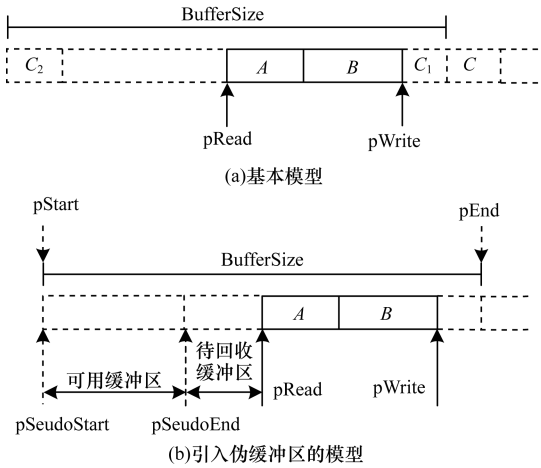

图 6　环形缓冲区模型

2.2　I/O 任务表

本文提出机制在首次对用户域进行初始化时,会通过超级调用在 Hypervisor 中为用户域建立 2 个任务表:接收任务表和发送任务表,分别对应用户域中的 I/O 任务接收环和发送环,存储 I/O 任务环的地址信息及 I/O 任务的相关控制位。该超级调用返回用户域与驱动域通信的任务表句柄,当驱动域从用户域接收到任务表句柄后,依据用户域提供的句

柄、任务表的物理地址和驱动域来映射任务表的地址范围发出超级调用,Hypervisor 在驱动域指定的地址空间映射 I/O 任务表,同时记录该虚拟地址范围。当再次创建新的用户域时,仅相应增加已有任务表的表项,不再另外新建任务表,任务表的表项数量与用户域数量相同。I/O 任务表对驱动域和 Hypervisor 均可见,用户域通过 Hypervisor 更新自身对应的任务表表项,Hypervisor 根据任务表表项对驱动域的读写地址范围进行控制。

如图 3 所示,任务表中每项包含 4 个域:

1)用户域 ID(DomID),由虚拟机监控器 Xen 在创建用户域时为每个用户域分配。

2)环形缓冲区中当前共享页面的机器地址(ph_addr),驱动域使用该页面来传递 I/O 任务数据包,对于发送任务表中的地址,用户域有读和写权限,驱动域只有读权限;对于接收任务表中的地址,驱动域有读和写权限,用户域只有读权限。

3)数据包长度(len),环形缓冲区中当前共享页面的长度。各虚拟域对数据包长度的读写权限与共享页面地址的权限相同。

4)任务状态标识(flag),在执行发送任务时,驱动域更改该标识为“忙”状态,指示 Hypervisor 页面正在被用作 I/O 操作,驱动域发送完成后,更改该标识为“闲”状态。当执行接收任务时,由用户域进行相应的状态更改。

3 实验结果与分析

3.1 实验设置

实验环境如下:服务器物理主机配置 8-core 3 GHz Intel Xeon E5-2690 CPU、128 GB 内存、10 Gb 网络接口卡,Xen4.4.1 虚拟机监视器,CentOS 6.5 操作系统。对于建立的每个虚拟用户域,分配 4 GB 内存、1 个虚拟 CPU、1 个虚拟网卡,DRB 缓冲区大小设置为 128 KB。

实验对比以下 3 种域间通信方法:

1)Front-back:用户域与驱动域通过标准前后端通信通道,采用页面映射模式共享数据缓冲区。

2)DRB:在用户域与驱动域之间通过共享双环形缓冲区,实现域间数据共享。

3)Loopback:非虚拟化主机上的操作系统的 2 个进程之间通过网络环路接口进行通信,测试结果作为其他测试场景下的参考标准。

实验采用 Netperf 2.7.0 作为测试工具,从吞吐率和响应延迟 2 个方面比较 Front-back,DRB 和 Loopback 这 3 种域间通信方法的性能。Netperf 是一款开源的网络性能测试工具,主要针对 TCP 和 UDP 传输进行测试。

3.2 性能分析

通过 Netperf 的 TCP_STREAM 流式通信功能,测试虚拟域单向吞吐率和端到端的响应延迟。首先测试不同消息大小对域间通信性能的影响,实验结果如图 7(a)所示。由于 DRB 在用户域初始化阶段完成了缓冲区分配、任务表初始化等工作,在通信过程中不需要对数据进行多次拷贝,也不需要每次通信均执行授权申请/撤销和缓冲区的申请/释放工作,因此性能明显高于 Front-back 模式,性能更接近 Loopback。当消息大小超过 128 KB 时,需要进行分包处理,吞吐率趋于稳定。Front-back 的页面映射共享机制由于受页面大小(4 KB)的限制,单次最大只能共享一个页面(4 KB)的数据,因此在消息不超过 4 KB时,吞吐率随着消息大小的增加而增加,当超过 4 KB 后,需要进行分包处理,吞吐率变化较为平缓。Loopback 由于没有缓冲区大小的限制,因此吞吐率随着消息大小的增加而增加。

如图 7(b)所示,DRB 的域间通信延迟低于 Front-back。对于 Front-back,当消息大小不超过其页面大小(4 KB)时,每次进行域间通信均需要执行相同数量的操作,延迟比较稳定。

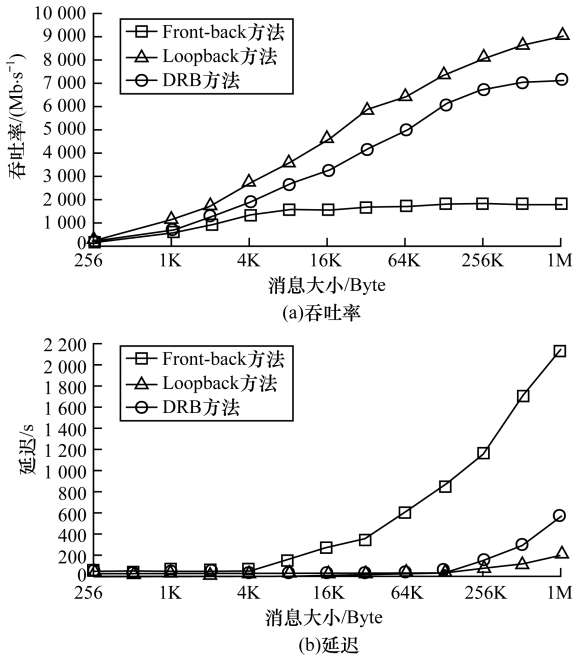

图 7 TCP 传输的吞吐率和延迟

当消息大小超过 4 KB 后,由于需要进行分包,因此导致延迟增加。消息越大,分包次数越多,执行的操作越多,延迟相应增加。DRB 由于可以在一侧处理数据时,另一侧同步写入数据,因此即使超过了

缓冲区容量 128 KB 时需要分包,但延迟增加仍较为平缓。

4　结束语

本文提出一种基于 DRB 的用户域与驱动域之间的通信优化方法,在用户域初始化时分别建立2 个环形缓冲区:接收环和发送环,通过 Hypervisor 将I/O环共享给驱动域,由 Hypervisor 依据 I/O 任务表对驱动域的访问地址空间及访问权限进行控制。在进行 I/O 任务的域间通信时,用户域与驱动域不需要执行完整的授权申请与撤销步骤,也不需要执行共享内存的分配与释放操作,只需对 I/O 任务表进行简单更新。环形缓冲区基于"生产者-消费者"机制工作,降低了用户域与驱动域在数据共享时的耦合程度。同时,利用伪缓冲区辅助管理写操作,避免了因剩余空间不足导致写入数据在物理内存中不连续的问题,实现了虚拟域对共享缓冲区的直接连续访问。实验结果表明,在 I/O 密集型虚拟机的高负载域间通信条件下,本文所提机制的吞吐率达到原通信机制的 3 倍以上,而通信延迟不足原机制的 1/3,其各项性能指标均优于原通信机制。下一步将对各用户域之间的通信优化方法进行研究,并结合域间 I/O 性能隔离,提出综合的域间通信优化方法。

参考文献

[1] 李雪竹,陈国龙. 云计算虚拟化平台的内存资源全局优化研究[J]. 计算机工程,2015,41(7):55-59.

[2] Lin Jenn-wei, Chen Chien-hung, Lin Chi-yi. Integrating QoS Awareness with Virtualization in Cloud Computing Systems for Delay-sensitive Applications[J]. Future Generation Computer Systems, 2014, 37(7): 478-487.

[3] Barham P, Dragovic B, Fraser K, et al. Xen and the Art of Virtualization[C]//Proceedings of the 19th ACM Symposium on Operating Systems Principles. New York, USA: ACM Press, 2003: 164-177.

[4] Peleg O, Morrison A, Serebrin B, et al. Utilizing the IOMMU Scalably[C]//Proceedings of 2015 USENIX Conference on Usenix Annual Technical Conference. Santa Clara, USA: USENIX Association, 2015: 549-562.

[5] Zeng Lingfang, Wang Yang, Feng Dan, et al. XColl-Opts: A Novel Improvement of Network Virtualizations in Xen for I/O-latency Sensitive Applications on Multi-cores[J]. IEEE Transactions on Network and Service Management, 2015, 12(2): 163-175.

[6] 侯建宁,董贵山,王　银,等. 基于虚拟化的系统安全增强及显卡透传研究[J]. 计算机工程,2012,38(8):224-227.

[7] 王　展,曹　政,刘小丽,等. 基于单根 I/O 虚拟化的多根 I/O 资源池化方法[J]. 计算机研究与发展,2015,52(1):83-93.

[8] Zhang Xiaolan, Mcintosh S, Rohatgi P, et al. XenSocket: A High-throughput Interdomain Transport for Virtual Machines[C]//Proceedings of ACM/IFIP/USENIX 2007 International Conference on Middleware. Berlin, Germany: Springer, 2007: 184-203.

[9] Wang Jian, Wright K L, Gopalan K. XenLoop: A Transparent High Performance Inter-VM Network Loopback[C]//Proceedings of the 17th International Symposium on High Performance Distributed Computing. New York, USA: ACM Press, 2008: 109-118.

[10] 丁圣阁,马汝辉,梁阿磊,等. 半虚拟化 I/O 模型的 KVM 虚拟机域间通信优化方法[J]. 计算机科学与探索,2011,5(12):1114-1120.

[11] Bai Yuebin, Ma Yao, Luo Cheng, et al. A high Performance Inter-domain Communication Approach for Virtual Machines[J]. Journal of Systems and Software, 2013, 86(2): 367-376.

[12] Frémal S, Bagein M, Manneback P. Optimizing Xen Inter-domain Communications[C]//Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing. New York, USA: ACM Press, 2015: 3-10.

[13] Chisnall D. The Definitive Guide to the Xen Hypervisor[M]. New Jersey, USA: Prentice Hall, 2007.

[14] Ning Fengfeng, Weng Chuliang, Luo Yuan. Virtualization I/O Optimization Based on Shared Memory[C]//Proceedings of 2013 IEEE International Conference on Big Data. New York, USA: IEEE Press, 2013: 70-77.

[15] Ram K K, Santos J R, Turner Y. Redesigning Xen's Memory Sharing Mechanism for Safe and Efficient I/O Virtualization[C]//Proceedings of the 2nd Conference on I/O Virtualization. Pittsburgh, USA: USENIX Association, 2010: 1-11.

编辑　陆燕菲