

XEN 虚拟 IO 的调度优化

王宇新, 惠新忠, 郭 禾, 刘 玮

(大连理工大学 计算机科学与技术学院, 辽宁 大连 116023)

摘 要: 提出一种基于驱动域授权表和优先级加权的虚拟 IO 调度策略, 根据不同客户域与请求资源的差异, 得出 IO 请求的调度优先级, 实现虚拟 IO 调度. 实验证明, 兼顾不同优先级域的调度策略提高了系统的吞吐率, 驱动域授权表机制减少了 IO 虚拟化中授权与页面映射操作的开销.

关键词: IO 虚拟化; I/O 环; 调度策略; 驱动域授权表

中图分类号: TP303

文献标识码: A

文章编号: 1000-7180(2010)08-0044-05

An Optimization for Virtual I/O Scheduling on XEN

WANG Yu-xin, HUI Xin-zhong, GUO He, LIU Wei

(School of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China)

Abstract: This paper presents a priority-scheduled strategy for I/O Virtualization with a Driver Domain Grant Table Set mechanism. The optimized mechanism calculates the priority for each new I/O request based on different weight of both domains and resources required, and reschedules the Virtual I/O request queue. The experiment results show that this strategy improves the throughput while giving consideration to domains' priorities and the Driver Domain Grant Table Set has greatly decreased the overheads on granting and mapping operation.

Key words: I/O virtualization; I/O ring; schedule strategy; driver domain grant table set

1 引言

虚拟机最早是为大型机系统提供虚拟化服务的, 随着单机系统性能的提高和计算资源的多样化, 虚拟机再次成为提高资源利用率, 整合计算环境的首选. 随着硬件厂商对虚拟化的硬件支持^[1], 虚拟机的性能已经接近物理计算机, 但是 IO 虚拟化始终是系统虚拟化中的一个瓶颈^[2]. Xen 是一种被广泛使用的开源虚拟机, 它使用驱动域模型来虚拟化 IO 设备. 驱动域简化了虚拟环境中设备支持的复杂程度, 为物理设备驱动提供一个安全的执行环境.

国内外对 IO 虚拟化展开了广泛的研究. VPE^[2]方法使用专用 CPU 核心来帮助 IO 虚拟化, 提高了虚拟 IO 的性能而不需要特殊硬件支持; NCQ^[3]机制用重排请求的方法提高 IO 吞吐率; 多端队列网卡的使用能减少包分发与数据拷贝开销, 提高网络 IO 的性能^[4]. 驱动域 VCPU 的 Credits 抢

占方法^[5], 也可以提高 IO 的性能^[6]. 在 Xen 的全虚拟化下加入准虚拟化驱动以提高网络 IO 的性能.

本文提出的是一种基于驱动域授权表(Driver domain Grant Table Set)的设备 IO 请求优先级调度策略. 驱动域根据客户域与 IO 请求, 加权得到虚拟 IO 请求的加权优先级, 重排 IO 环中的请求, 最后按序处理.

2 Xen 虚拟化架构与 IO 虚拟化

Xen 是一个轻量级虚拟机软件, Xen Hypervisor 向各个域 Domain 提供虚拟硬件资源, 同时管理这些资源. 如图 1 所示, Dom0 是特权域, 辅助 Xen 管理客户域 DomU, 并向 DomU 提供虚拟资源. 当多个 DomU 需要访问硬件时, Dom0 可能成为系统的瓶颈, 通过创建驱动域 IDD 来减轻 Dom0 的负担. 驱动域 IDD 为客户域提供设备驱动, 完成真实硬件的访问.

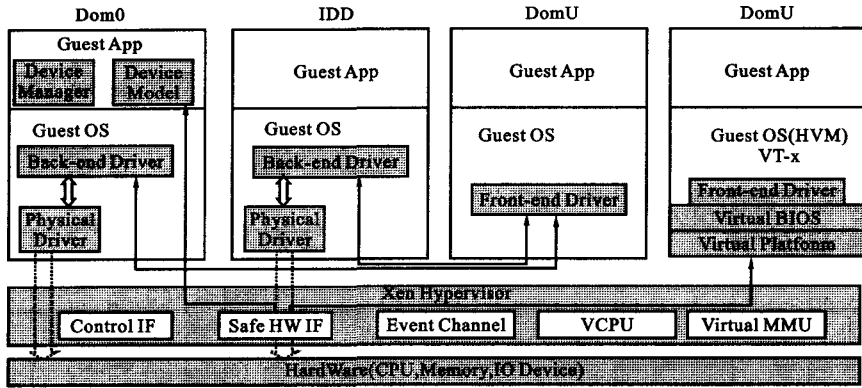


图 1 Xen 结构

Xen 的虚拟设备模型采用分离驱动模型,即分为前端、后端和本地设备驱动三部分.前端设备驱动位于 DomU,后端、本地设备驱动位于 Dom0 或 IDD 中.前端驱动为客户域提供了虚拟设备,将 IO 请求转发到后端,由后端控制本地驱动间接完成 IO 操作.前端驱动和后端驱动之间 IO 请求的传递是通过设备 IO 环与事件通道实现的.在 IO 环结构^[7]中,有两个生产者-消费者对:前端驱动是请求的生产者与响应的消费者,后端驱动为请求的消费者与响应的生产者,如图 2 所示.为了实现域间的高效通信,Xen 用授权表机制^[8]实现页面映射以避免不必要的内存拷贝.客户域进行 IO 访问时,前端驱动会生成一个授权索引 GR,并把 GR 和 IO 请求一同放入 IO 环中.驱动域取出请求处理时,将 GR 发送到 Xen,调用超级调用进行检查与设置.若通过检查,执行相应操作,则在驱动域接收到 Xen 的响应后可以进行 IO 操作,否则向客户域发送中断信号.

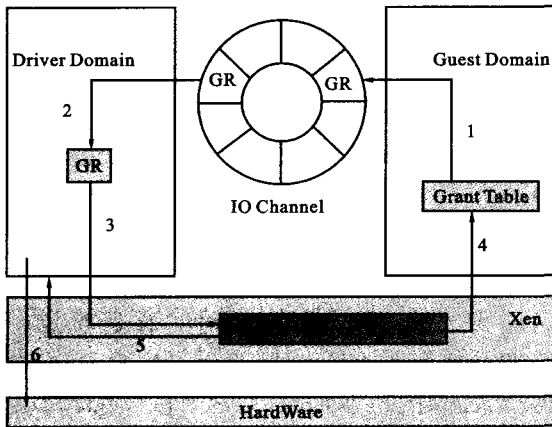


图 2 IO 虚拟化机制

虚拟 IO 的吞吐率.在一次 IO 访问中,有两次 Xen 和 Domain 之间的切换,影响 IO 效率,对此我们采用驱动域授权表机制来减少开销.

3 基于驱动域授权表的虚拟 IO 调度

3.1 设备 IO 请求调度

Xen 驱动域对设备 IO 请求的处理采用 FCFS 顺序,有利于 IO 密集型应用,但会影响软实时应用的 IO 性能,降低 IO 的吞吐率.对此,根据不同客户域与请求资源的差异计算每个 IO 请求的调度优先级 $request_priority$,如式(1),每次取出当前调度优先级最高的进行处理.

$$request_priority = weight * domain_priority + (1 - weight) * io_complexity_priority \quad (1)$$

式中, $domain_priority$ 表示 Domain 的优先级.驱动域中有与 Xen 共享的域优先级表 Domain Priority Table (DPT),记录每个域的优先级.Domain 启动时即设置相应的优先级,可通过超级调用修改,驱动域只有读权限. $io_complexity_priority$ 为 IO 请求复杂度决定的优先级,由驱动域的 IO Accounting 模块来计算. $weight$ 为权重,根据对客户域与 IO 请求本身优先级的关注程度的不同在 0~1 之间调整.

优化后 IO 请求调度过程如图 3 所示.当客户域提交 IO 请求时,将请求与授权描述 GR 放入队列尾部.驱动域调度器 IOReq Scheduler 取请求进行处理时,先由 IO Accounting 模块计算新插入请求的调度优先级,再由 IO Sort 模块根据各请求的调度优先级重新调整队列,最后由 IOReq Scheduler 取出优先级最高的 IO 请求进行处理.在 IO 环队列结构中设置一个指示器 $Order_Pos$ 标识有序队列的位置,当驱动域处理一个请求期间有多个请求达到队列时,则 $Order_Pos$ 位置后新插入的多个 IO 请求是无序

驱动域处理虚拟 IO 请求是先来先服务的 (FCFS),没有考虑域间差异和请求本身的复杂度,因此本文提出了一种优先级调度算法,以提高系统

序列,即需要排序部分。

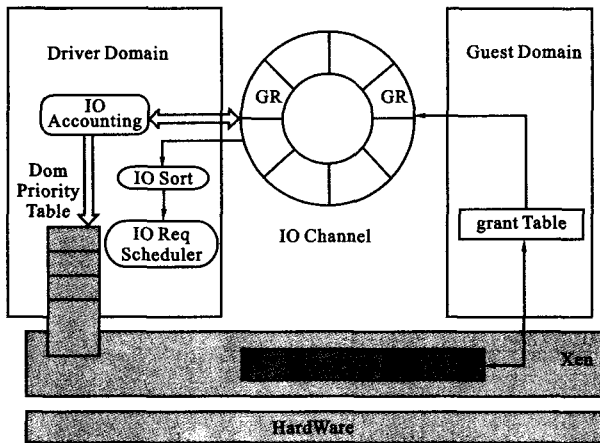


图3 IO请求调度

3.2 IO Accounting 与 IO Sorting

IO Accounting 模块的作用是计算 IO 请求优先级。此时需要考虑的是请求的资源数与域的优先级。IO Accounting 模块解析 IO 请求,根据请求结构体中所请求的资源数量来定义请求本身的优先级 $io_complexity_priority$,然后在 Domain 优先级表中取得对应域的优先级 $domain_priority$,再按照式(1)进行计算。 $io_complexity_priority$ 的计算区别于不同的 IO 设备。以块设备 IO 请求为例说明,其他设备请求调度优先级的计算类似。块设备 IO 请求结构体 `struct blkif_request` 中包含请求的段数 `nr_segments`、块设备请求段结构数组 `seg` 等,则该请求调度优先级计算如下:

$$io_complexity_priority = req_priority_account(blkif_request) * STANDERED_PRIORITY \quad (2)$$

式中,函数 `req_priority_account` 解析收到的请求,根据 `blkif_request` 中 `nr_segments` 来定义优先级系数,返回 0~1 的值,再映射到 0~STANDERED_PRIORITY(标准优先数)区间。然后由式(3),根据提交该请求的 DomID,使用函数 `dom_priority_account` 查询域优先级表 DPT,返回提交 IO 请求的 Domain 的优先级 $domain_priority$,该优先级的取值同样定义在是 0 STANDERED_PRIORITY 区间。

$$domain_priority = dom_priority_account(domid) \quad (3)$$

IO Sort 模块的作用是对设备 IO 环队列中的请求排序,排序的键值为请求的调度优先级。排序后要求调度优先级最高的请求排在队列头部位置,然后由请求调度器 IOReq Scheduler 取出调度优先级最

高的一个或一批请求同时处理,由后端驱动程序控制本地设备驱动完成 IO 操作。处理完一次 IO 请求后由于队列中剩余请求仍然是有序的,所以不需要再次排序,只有当 IO 环请求队列中有新的 IO 请求到来时才对该请求计算调度优先级,将其插入有序队列。

由于设备 IO 环队列负责所有域的 IO 设备请求,所以 IO 请求队列需要频繁的插入删除请求,删除请求后队列不需要做调整,因为删除后队列仍是有序的。当插入新请求时,需要计算其调度优先级插入队列使其有序。插入位置使用二分查找法得到,比较次数为 $O(\log N)$,将其插入正确位置平均移动次数 $N/2$,经验证此排序方法为最优。

虚拟设备 IO 请求调度所有的处理过程包括 IO Accounting、IO Sort 和 IOReq Schedule,都是在驱动域完成,这在一定程度上增加了驱动域的开销,但设备 IO 请求的调度只是对 IO 请求结构体的调度,计算复杂度并不高,因此不会成为设备 IO 性能的新瓶颈。为尽量减少驱动域的开销,本文提出一种驱动域授权表机制以进一步提高系统性能。

3.3 驱动域授权表机制

Xen 采用分离驱动结构,虚拟 IO 访问过程中需要进行授权项的创建销毁、活动授权项的检查设置、页面的映射回收等操作,这些操作都需要使用超级调用,带来的是 Xen 与 VM 的上下文切换。这是影响虚拟 IO 性能的一个重要因素。文献[9]提出了一种细度 IO 访问控制,防止恶意域 IO 请求致使性能降低。本文提出的是一种驱动域授权表机制,以减少虚拟 IO 访问带来的上下文切换。

驱动域授权表由 Xen 中的所有域的活动授权表构成,以 DomainID 为索引,其中的授权项信息包括页面的映射情况、页面的读写权限等。驱动域对驱动域授权表有只读权限,Xen 有读写权限。由于它位于驱动域空间,驱动域可以根据 GR 访问对应域的活动授权表,而不需使用超级调用,如图 4 所示。驱动域授权表机制带来的性能提升体现在两方面。

(1) 避免了授权项检查失败时不必要的上下文切换。驱动域取得 GR,利用授权表检查对应授权项,检查通过则进行页面映射,否则利用事件通道向对应域发送中断信号,减少了检查失败时超级调用的开销。

(2) 设置客户域缓冲池,减少授权与映射操作的开销。传统 Xen,IO 访问完毕后销毁授权项,撤销页面,对应页面会用作其他用途,这会带来很大的消

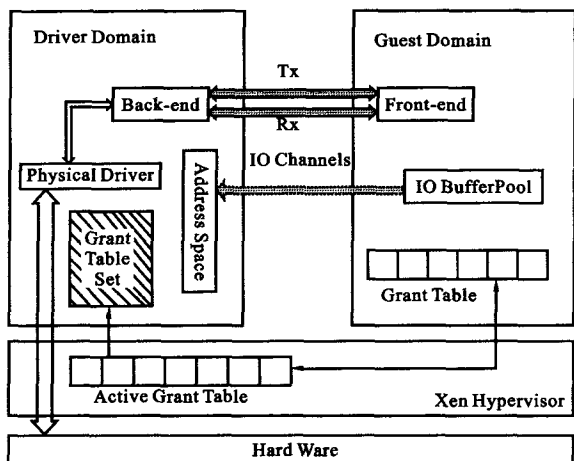


图4 驱动域授权表

耗。而在客户域设置一块页面缓冲池,所有页面已经映射到驱动域,IO操作完成后授权项不销毁,页面映射也不回收,可以对其进行重用。驱动域只需要检查对应的授权项,以获知对应页面是否正在被访问。这极大的减少了授权与页面映射操作的开销,增加了IO虚拟化效率。

同时随着硬件技术的发展,多端队列设备已经成为发展趋势,可将IO缓冲池中的页面映射用作多端队列设备的缓冲区,这会进一步提高IO的传输效率,减少驱动域开销。

4 评估

本文实验平台为四核处理器 Intel(R) Core (TM) i5 750 2.67GHz,4GB内存。实验在Xen3.4上实现,Dom0操作系统为CentOS 5.3,Driver Domain与两个GuestDomain操作系统均为Ubuntu8.10。使用iozone来测试块设备IO访问的性能,iperf来测试虚拟机网络IO性能,编写程序在各Domain同时提交IO请求,验证其优先级调度的正确性。

4.1 设备IO性能

使用IO性能测试基准程序iozone测试磁盘IO的性能,如图5所示。读写文件大小为4Gb,为防止系统缓存文件,对性能测试结果产生影响,文件不小于内存大小。本地Linux性能最高,写速度达到95.8Mb/s,读取速度可达到75.4Mb/s。Dom0有自己的驱动程序但Xen仍然需要一部分开销,性能次之,DomU由于所有的请求都是由驱动域驱动程序完成,因此效率最低,写速度9.2Mb/s,读速度达到41.5Mb/s。DomU的CPU使用率为9.19%,驱动域的CPU使用率为15.83%。驱动域添加了设备调度与驱动域授权表后DomU的读写效率有明显的提

升,写速率达到45Mb/s,读速率64.2Mb/s。此时驱动域的CPU占用率最高为19.67%,这是由于IO请求调度的开销与驱动域授权表查询的开销使CPU占用有所提高,相比读写性能的巨大提升,这些开销可忽略。

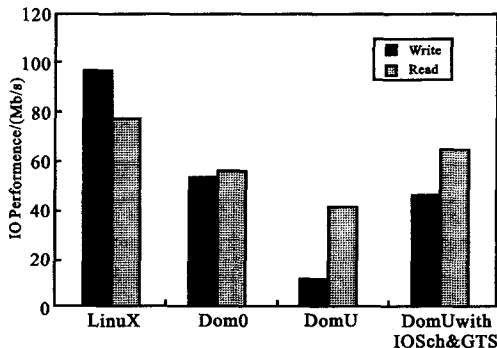


图5 块设备IO性能

使用iperf基准测试网络IO的性能,如图6所示。Iperf的测试是采用服务器客户端模型,在测试中另外一台测试机在同一局域网。CPU采用Intel Q8200 2.33GHz,内存2GB。本地Linux发送与接收速率分别为95.5Mb/s、96.2Mb/s,Dom0网络IO的性能接近本地Linux性能,DomU的发送与接收速率为87Mb/s、66.7Mb/s。当采用驱动域授权表机制与IO请求调度时,DomU的发送与接收速率达到本地Linux的94.3%,为90.5Mb/s、91.3Mb/s。此时驱动域CPU占用为23.4%,对系统性能没有大的影响。

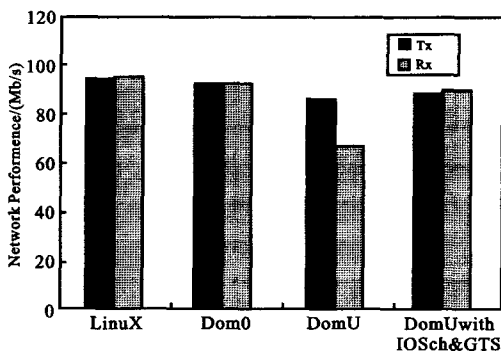


图6 网络IO性能

4.2 设备IO请求调度

编写一个程序在不同的域同时提交不同的IO请求,在IO请求完成时记录所需要的时间与完成的顺序,根据不同域的优先级与请求的复杂度估计各个请求的优先级,查看测试结果。以块设备IO为例,验证程序正确性。

测试中设置3个客户域,驱动域的优先级表记录其不同的优先级,各优先数在0到STANDERED

_PRIORITY(10)范围内.每个域提交 5 个 IO 请求,每个请求的所需资源不同且拥有不同优先级.计算后的调度优先级和最终完成顺序如表 1 所示,其中域优先级 Domx(y)表示 Domx 的优先数为 y,IO 请求资源 IORequest_x(y)表示 IO 请求 x 所需的资源数为 y,weight 取 0.6.

表 1 IO 请求调度优先级表

IO 请求资源 \ 域优先级	Dom1(10)	Dom2(8)	Dom3(5)
IORequest_1(3)	8.9 / 2	7.7 / 5	5.9 / 11
IORequest_2(7)	7.45 / 6	6.25 / 10	4.45 / 14
IORequest_3(5)	8.18 / 4	6.98 / 8	5.18 / 13
IORequest_4(2)	9.64 / 1	8.44 / 3	6.64 / 9
IORequest_5(8)	7.1 / 7	5.9 / 12	4.1 / 15

优先数越大说明优先级越高,驱动域应该首先处理,表中数据验证了 IO 请求完成顺序是按照优先级高低顺序排列,证明了设备 IO 请求调度算法的正确性.

5 结束语

本文提出一种基于驱动域授权表的设备 IO 请求调度策略.该策略利用虚拟设备 IO 模型中的设备 IO 请求队列,按照不同域与请求所需资源计算 IO 请求的优先级,然后根据其优先级的顺序调度请求进行处理.驱动域授权表机制减少了授权过程中 VM 切换,同时利用缓冲池的思想,进一步降低了设备访问中的授权与页面映射的开销.实验证明,驱动域授权表机制可以很好的提高 IO 设备的带宽,设备 IO 请求调度算法兼顾了域的应用的差异性,提高了系统吞吐率.

参考文献:

- [1] Gil Neiger, Amy Santoni, Felix Leung, et al. Intel virtualization technology: hardware support for efficient processor virtualization[J]. Intel Technology Journal, 2006, 10 (1):162-178.
- [2] Jiuxing Liu, Bulent Abali. Virtualization polling engine (VPE): using dedicated CPU cores to accelerate I/O virtualization[C]// ACM Proceedings of the 23rd international conference on Supercomputing. New York: Yorktown Heights, 2009.
- [3] Young Jin Yu, Dong in Shin, Hyeonsang Eom, et al. NCQ vs. I/O scheduler: preventing unexpected Misbehaviors[J]. ACM Transactions on Storage (TOS), 2010, 6 (1):15-27.
- [4] Kaushik Kumar Ram, Jose Renato Santos, Yoshio Turner, et al. Achieving 10 Gb/s using safe and transparent network interface virtualization[C]// Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. Washinton, 2009.
- [5] Guangdeng Liao, Danhua Guo, Laxmi Bhuyan, et al. Software techniques to improve virtualized I/O performance on multi-core systems [C]// Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems. USA:San Jose, 2008.
- [6] 胡冷非,李小勇. 基于 Xen 的 I/O 准虚拟化驱动研究[J]. 计算机工程, 2009, 35(23):258-262.
- [7] Paul Barham, Boris Dragovic. Xen and the art of virtualization[C]// ACM Proceedings of the 19th ACM symposium on Operating systems principles. USA: New York, 2003.
- [8] David Chisnall. The definitive guide to the xen hypervisor, First edition[M]. New York: Prentice Hall, 2007:59-71.
- [9] Sung-Min Lee, Sang-Bum Suh, Bokdeuk Jeong, et al. Fine-grained I/O access control of the mobile devices based on the xen architecture [C]// Proceedings of the 15th annual international conference on Mobile computing and networking. China:Beijing, 2009.

作者简介:

王宇新 男,(1973-),博士研究生,讲师.研究方向为计算机系统结构、高性能计算.

惠新忠 男,(1985-),硕士研究生.研究方向为计算机系统结构.

郭禾 男,(1955-),硕士,教授.研究方向为计算机系统结构、高性能计算.

刘玮 男,(1986-),硕士研究生.研究方向为计算机系统结构.