基于 Xen PVHVM 虚拟块设备的数据追踪及测试

刘旺，王洪波，程时端

（北京邮电大学网络与交换技术国家重点实验室，北京 100876）

摘要：虚拟块设备是虚拟化技术实际应用中关键的一环，本文分析了 Xen 平台中 PVHVM 模型通用性和性能两方面的优势，结合源代码、数据追踪和实验，对虚拟块设备的各模块间的数据交互流程进行了详细的分析，比较并阐述 PVHVM 模式下虚拟块设备的特点。

关键词：计算机应用技术；服务器虚拟化；虚拟块设备；Xen；PVHVM

中图分类号：TP315

Data Tracking and Test of Virtual Block Device on Xen PVHVM

Liu Wang, Wang Hongbo, Cheng Shiduan

(State Key Laboratory of Networking and Switching Technology,Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: Virtual block device is a key part of practical virtualization technology. This paper analyzes the advantages of both the versatility and performance of PVHVM model in the Xen platform. After that, the article carried out a detailed analysis of the data interaction processes between the virtual block device modules, then compared and elaborated different type of virtual block device backend--blkback's and blktap's characteristics based on the source code, data tracking and the experimental test.

Keywords: Computer Application Technology; Virtualization; VirtualBlock Device; Xen; PVHVM

0 引言

当前，计算机处理能力在快速增长，硬件资源的规模和种类在不断扩展，与此同时，在数据中心内，应用和需求越来越复杂、灵活，传统服务器的利用效率日趋降低，因此，虚拟化技术成为了一个重要的解决方案。虚拟化技术能够在单个计算机上运行多个相互隔离的虚拟机（Virtual Machine），动态提供透明化的可伸缩的计算机硬件资源，从而灵活构建满足需求的计算机软硬件环境。目前虚拟化技术已经成为数据中心的支撑性技术，出现了很多比较优秀的虚拟化平台和产品，如 Xen、Vmware、KVM 和 QEMU 等。

按照硬件资源的种类可以将虚拟化技术划分为 CPU 虚拟化、内存虚拟化和 I/O 虚拟化。I/O 设备具有异构性强、内部状态不易控制等特点，块设备是绝大部分服务器所必须的 I/O 设备，提供了大数据读/写服务，在使用虚拟化技术的数据中心内，虚拟块设备的读写速度和效率往往是整个服务器的瓶颈，成为了虚拟化技术中的难点和重点。据我们所知，本文是第一篇基于 Xen PVHVM 相关源代码及实验测试详细分析并比较虚拟块设备的文章。

本文结构如下，第一章简单介绍 Xen 平台，第二章结合源代码给出详细原理分析，第三章给出我们的测试和结果，最后在第四章进行总结和展望。

基金项目：高等学校博士学科点专项科研基金（200800131019）

作者简介：刘旺，（1988-），男，硕士研究生，主要研究方向：云计算与虚拟化。

通信联系人：王洪波，（1975-），男，副教授，主要研究方向：云计算与数据中心网络，互联网服务质量管理与监测等。E-mail: hbwang@bupt.edu.cn

1 Xen 虚拟化平台简介

Xen[1]是一个由剑桥大学主持开发的开放源代码的虚拟机监视器，相比于其他虚拟化技术，Xen 能获得接近硬件的高效能，典型情况下的效能损失只有 2%。因此，在 SOSP 会议上，Xen 一经发表就引起了广泛的关注。Xen3.0 及其之后的版本，基于硬件辅助虚拟化技术的支持，实现了全虚拟化的支持。Linux3.0 开始将 Xen 正式加入了 Linux 内核。

在 Xen 的术语中，直接运行在硬件层之上、管理硬件资源并对虚拟机进行调度的虚拟机监视器（Virtual Machine Monitor），叫做 Hypervisor；在 Xen 上运行的虚拟机叫做客户机（Guest Domain 或 Domain U），其内运行的操作系统称为 Guest OS；一个 Xen 系统中，必须有一个 Domain 0，拥有直接与物理设备交互的特权，负责处理 Domain U 的 I/O 请求。

2 XenPVHVM 虚拟块设备数据交互的分析

2.1 Xen PVHVM 模型的分析

虚拟块设备的设计目标是满足不同客户机对有限真实块设备的复用需求。所以，我们评价块设备虚拟化的指标是性能和通用性。性能是指块设备的读写速度越快性能越好；通用性是指虚拟块设备对客户机是否足够透明。目前，Xen 支持的模型包括 PV、HVM、PV HVM。

PV（Paravirtualization），由 Xen 首先提出并由于其高效的性能而逐步被其他虚拟化技术所借鉴，通过分离设备模型实现块设备的虚拟化。块设备后端驱动位于 Domain 0 中，负责与真实块设备的直接交互，完成虚拟设备地址到物理设备地址的转换，而块设备前端位于 Domain U 中，需要修改 Guest OS。HVM（Hardware-assisted Virtualizion Machine），HVM 类型客户机需要 Intel -VT[2]和 AMD-V[3]等硬件技术的支持。以 Intel 的技术为例，在 CPU 虚拟化方面采用 VT-x（Intel Virtualization technology for x86）技术，内存虚拟化方面使用 EPT（Extended Page Table）技术，I/O 设备虚拟化方面采用 VT-d 技术[4]。

PVHVM，又叫 PV on HVM。PV 模式虚拟块设备性能较好，但通用性透明性较差，而 HVM 模式下，如果不采用设备直接分配，HVM Guest 只是通过 Domain0 中一个特殊的 daemon——Qemu-dm 进行设备模拟，从而完成网络互联和磁盘访问请求，通用性好但性能无法满足实际应用中对块设备读写速度的要求。目前块设备虚拟化仅能够实现设备直接分配。包括 Intel、AMD 等公司在内的研究机构正在扩展 Linux 和 Xen，研究全面支持硬件辅助虚拟化的 I/O 虚拟化解决方案[5]。因此，Xen 也引入了 PVHVM 驱动。PVHVM 驱动是特别针对 HVM 模式进行优化的 PV 驱动，既能达到对上层虚拟机透明的需求，又能获得接近于 PV 模式的性能。在实际数据中心内，PV on HVM 取得了较好的效果[6]。新版本的 Xen4.3 在 PVHVM 的基础上权衡不同客户机类型的优缺点，进一步改进和简化了 Xen 的虚拟化架构，引入了 PVH（PV in an HVM Container）模型。PVHVM 是未来的重点研究方向。

2.2 PVHVM 虚拟块设备的数据追踪和分析

通过在 Linux 内核及 Xen 源代码中修改并添加追踪点，从而得到数据交互流程。关键追踪点主要包括前后端通信、读写请求传送、读写请求转换、用户态内核态交互等，利用 trace event 机制追踪函数调用和读写请求数据结构的转换。通过追踪点截获的数据，可以将数据交互流程分为两个步骤：1. 设备发现，从真实块设备抽象出虚拟块设备并让 Guest OS 加载相关驱动；2. 读写请求转换，即截获 Guest OS 中的读写请求，转换成对真实物理设备的读写请求。实现以上两个步骤的具体数据交互流程根据客户机类型的不同而不同。Xen4.1.2 常

用两套块设备后端：blkback 和 blktap2，数据交互的流程随着块设备后端的不同有所不同。

2.2.1 Blkback 数据交互流程的分析

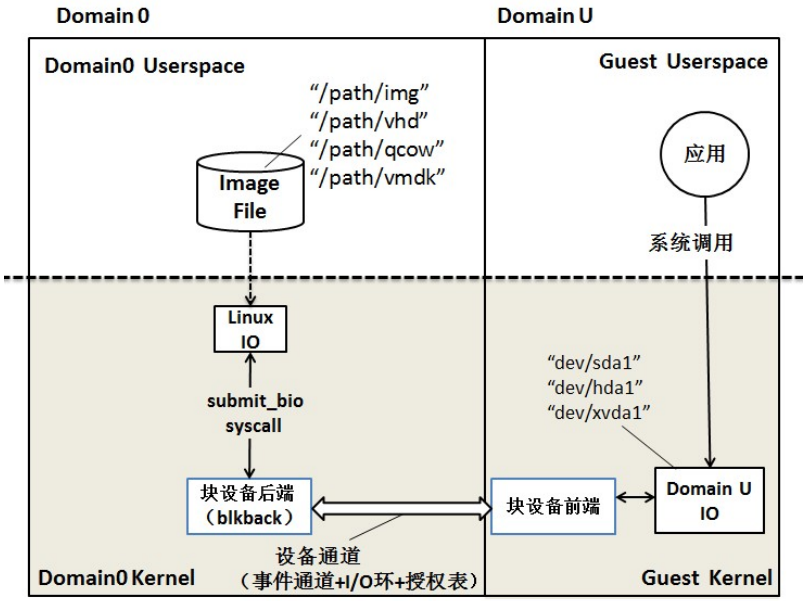

图 1 blkback 的数据交互流程

Blkback 的数据交互流程如图 1 所示，在 Domain U 中，用户态的应用程序发出读写系统调用，在 Domain U 的操作系统中的普通读写请求，被块设备前端封装成 Xen 格式，接着通过设备通道传送给块设备后端，块设备后端将读写请求经过一系列处理后,转化为一个或多个 bio 结构，submit_bio 提交。前后端通过事件通道（Event Chanel）、I/O 环（I/O Ring）以及授权表（Grant Table）完成通信和数据交互。从 Domain U 的角度看，块设备可以是 sda、had 或者 xvda 等真实设备，只是读写请求被前端截获并转交给了 Domain 0。从 Domain 0 中操作系统的角度看，bio 最后进入 I/O 调度层，Domain U 的磁盘对应着相应的磁盘镜像。

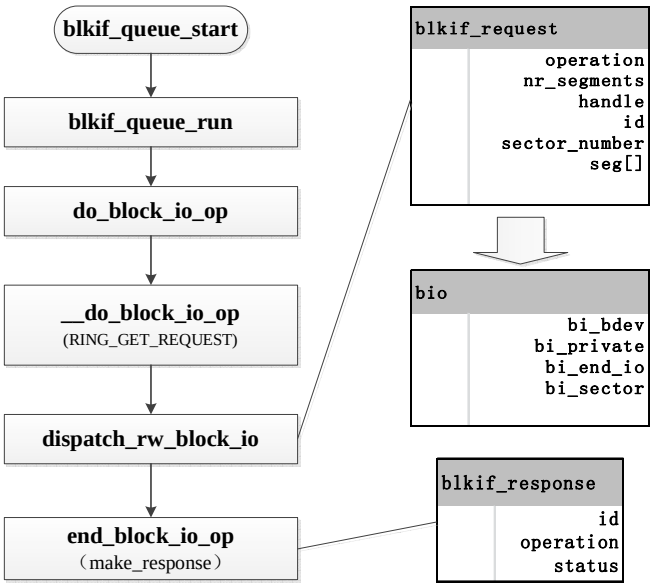

图 2 读写请求在块设备后端中的处理过程

图 2 是读写请求进入在块设备后端中的详细处理过程及函数。 RING_GET_REQUEST 从I/O环中取出块设备前端发送过来的读写请求（blkif_request格式），在dispatch_rw_block_io

中将 blkif_request 转化成 pending_req 和 phys_req，最后生成一个或多个通用的 bio 结构，之后的处理流程与 Linux 读写系统调用相同：bio 进入 I/O 调度层最后由真实驱动完成读写。同时，由 end_block_io_op 生成响应（blkif_response）。

2.2.2 Blktap 数据交互流程的分析

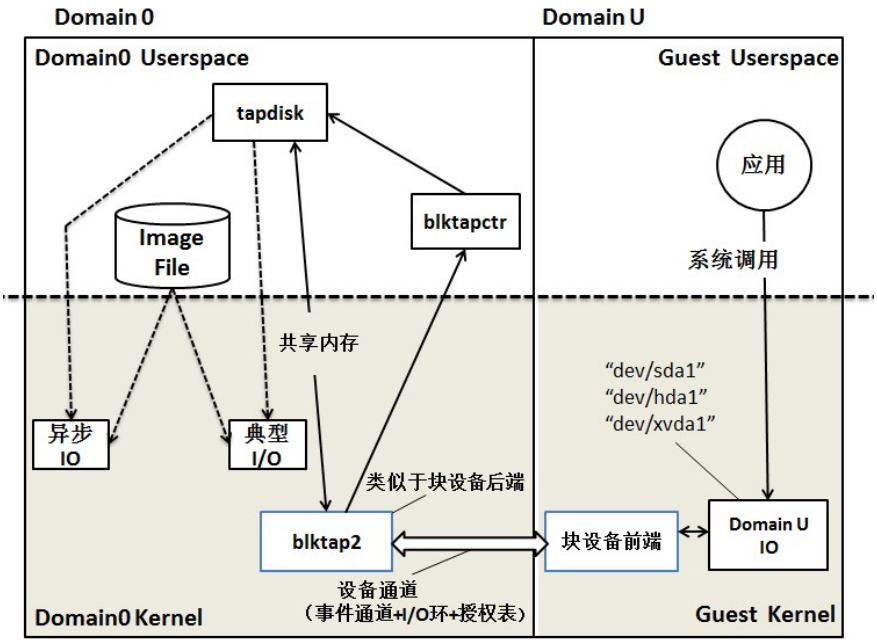

图 3 blktap 的数据交互流程

Blktap2 的数据交互流程如图 2 所示，DomainU 中流程与 blkback 类似，但 domain 0 中用户态的功能更加丰富了，并将原本处于内核态的部分功能移至用户态。与 blkback 不同是，blktap2 从设备通道中取出 Domain U 发送过来的读写请求（blkif_request 格式），交给处于用户态的 tapdisk，tapdisk 要通过对应的磁盘格式的驱动来区分磁盘镜像的类型，并且可以选择异步 IO 或者典型 IO 两种不同方式，对读写请求和数据进行检查，封装成 tio 结构，对 tio 进行合并优化后，通过系统调用提交 tio。Blktap2 与 tapdisk 通过共享内存的方式交换数据的权限，与设备/dev/xen/blktapX 对应。blktapctr 通过设备 blktap0 管理虚拟磁盘。blktap 仍需要 blkback 的帮助， blktap2 以及最新的 blktap3 已经可以独立承担块设备后端的功能。

3 实验测试

3.1 测试环境和方法

实验机器配置如下：CPU：AMD Athlon Ⅱ X4 645，四核 3.0GHz 主频，内存：8G，硬盘：1T，Domain 0 操作系统：装了上 Xen4.1.2 的 CentOS5.8（内核版本 2.6.32.12）。Domain U 分配单核 CPU 与 1G 内存，操作系统为 CentOS5.8（内核版本 2.6.18-308）。目前，Xen 的虚拟磁盘可以分为物理磁盘、逻辑卷和磁盘镜像[7]。我们统一使用 raw 格式磁盘镜像，分别使用 blktap（aio）、blktap2（aio）、blkback 以及 QEMU 四种块设备。

数据中心内服务器常用应用的读写场景包括小数据块读写（如数据库操作）和普通数据块读写。我们通过编写测试程序来读写虚拟磁盘来模拟。通过设置一次读或写的块的大小，模拟读写小数据块（块大小为 512B）和普通数据块（块大小为 4KB）两种情况。测试原理是用 dd 命令多次直接向硬盘读写 10G 大小的文件，结合/dev/zero 空设备，设置 iflag 及 oflag

为不使用缓存，多次测试取平均值，计算出只读速率、只写速率及读写速率。

3.2 测试结果

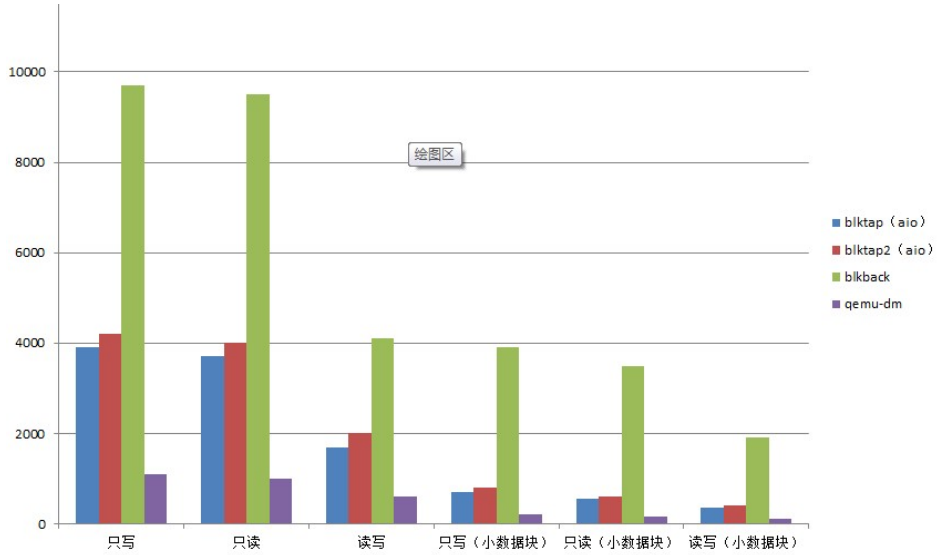

图 4 读写性能测试（单位 KB/S）

从图 4 中我们可以看到，只写普通数据块时，blkback 的性能大约是 blktap2 的 2.3 倍，blktap2 性能优于 blktap 大约 7%左右，采用 Qemu 硬件模拟的效率最差。只读数据块时，无论 blktap、blktap2 还是 blkback，性能都相较只写时效率有所下降，其中 blktap2 下降了大约 7%,说明虚拟块设备处理读请求时的数据交互效率比处理写请求时的效率相对要低。同时读写时，blkback 是 blktap2 的 2.05 倍。Blktap2 增加功能至用户态后却没有很好的解决性能下降的问题。

读写小数据块时，四种模式的性能都有所下降，因为读写次数增多导致读写请求的转换和处理的时间增多。Black 的读写速率下降了越 60%，但 blktap 及 blktap2 的降幅远远超过了 blkback，说明 blkback 处于内核态，完成读写请求转换后提交了 bio，但 blktap2 需要通过共享内存与用户态的 tapdisk 完成交互，实验结果符合我们的预测。

4 结论

本文在分析源代码的基础上，通过数据追踪及实验测试，分析了 Xen PVHVM 模式的原理和在实际应用中的优势，进而给出了虚拟块设备的模块交互和数据交互的原理，并结合源码详细分析了采用 blktap（blktap2）和 blkback 两种不同块设备后端时的数据交互的流程。blkback 数据流程相对简单且性能稳定，blktap 结合了虚拟存储层的发展趋势，支持两种 I/O 模式和多种磁盘镜像格式，功能模块划分更清晰且扩展功能更丰富。最后通过我们的实验和分析也可以看到 blktap、blktap2 和 blkback 的差异和特点。

[参考文献] (References)

[1] Xen Source Code, Xen4.1[OL]. [2012-9-1]. http://www.xen.org/
[2] Uhlig R.Neiger, G.Rodgers. Intel Virtualization Technology[C]. Computer Society on IEEE. 2005
[3] AMD,AMD-V[OL]. [2012-9-1]. http://china.amd.com.cn/BUSINESS/IT-SOLUTIONS/VIRTUALIZATION/
[4] Intel Corporation 2008,英特尔开源软件技术中心,复旦大学并行处理所. 系统虚拟化--原理与实现[M]. 北京：清华大学出版社，2009
[5] Younge Andrew J. Analysis of virtualization technologies for high performance computing environments[A].Cloud Computing (CLOUD), 2011 IEEE International Conference on[C]. 2011. 9-16.

[6] Whalen, Edward. Oracle VM Implementation and Adminstration guide[M]. Tata McGraw-Hill Education, 2011
[7] 杨亚军，高云伟. Xen 虚拟化环境中镜像文件的访问直接映射研究[J]. 高技术通信，2012，22(5):483-489

155