修改客户操作系统优化 KVM 虚拟机的 I/O 性能

张彬彬[1)] 汪小林[1)] 杨 亮[1)] 赖荣凤[1)] 王振林[2)] 罗英伟[1)] 李晓明[1)]

[1)](北京大学计算机科学技术系 北京 100871)

[2)](密西根理工大学计算机科学系 霍顿 美国)

摘 要 目前,运行在虚拟机上的客户操作系统(Guest OS)是面向物理机器开发的普通操作系统,其中存在不适应虚拟化环境的因素,影响虚拟机的 I/O 性能.作者通过测试发现了影响虚拟机 I/O 性能的一些问题,针对这些问题提出了优化方法:一方面,通过合并客户操作系统中连续的 I/O 指令,降低虚拟机的时钟中断频率,从而降低环境切换的开销;另一方面,消除客户操作系统中的冗余操作,包括在虚拟化环境下无效的函数、冗余的 I/O 调度以及虚拟网卡驱动对 NAPI 的支持,使虚拟机只执行必要的操作,从而提高系统的性能.

关键词 I/O 虚拟化;KVM;性能测试;优化

中图法分类号 TP319 **DOI 号**: 10.3724/SP.J.1016.2010.02312

Modifying Guest OS to Optimize I/O Virtualization in KVM

ZHANG Bin-Bin[1)] WANG Xiao-Lin[1)] YANG Liang[1)] LAI Rong-Feng[1)]

WANG Zhen-Lin[2)] LUO Ying-Wei[1)] LI Xiao-Ming[1)]

[1)](*Department of Computer Science and Technology, Peking University, Beijing* 100871)

[2)](*Department of Computer Science, Michigan Technological University, Houghton, USA*)

Abstract There are some operations in Guest OS causing the I/O performance degradation. For it is the general OS, which is oriented physical machines, directly running on the virtual machine as the Guest OS. This paper evaluates KVM I/O performance, find that the Guest OS should be adapted to the virtual environment for better performance, and proposes some optimizations. First, reduce VM Exits by merging successive I/O instructions and decreasing the frequency of timer interrupt; Second, removes some redundant operations from Guest OS, including the operations useless in virtual environment, the I/O scheduler whose results will be rescheduled in the Host OS, and the NAPI support in virtual NIC driver.

Keywords I/O virtualization; KVM; performance evaluation; optimization

1 引 言

虚拟机的 I/O 性能至今没有得到较好的解决.KVM 虚拟机(Kernel-based Virtual Machine)使用软件模拟的方式实现 I/O 设备的虚拟化,其实现方式是由内核中的 KVM 模块截获客户操作系统中的 I/O 请求,交给运行在宿主操作系统(Host OS)上的 QEMU,QEMU 将这些请求转换为对宿主操作系统的系统调用,通过宿主操作系统的设备驱动访

收稿日期:2010-08-22.本课题得到国家"九七三"重点基础研究发展规划项目基金(2007CB310900)、国家自然科学基金(90718028,60873052)、国家"八六三"高技术研究发展计划项目基金(2008AA01Z112)和教育部-英特尔信息技术专项科研基金(MOE-Intel-10-06)资助.**张彬彬**,女,1982 年生,博士研究生,主要研究方向为系统虚拟化.**汪小林**(通信作者),男,1972 年生,博士,副教授,主要研究方向为系统虚拟化、地理信息系统.E-mail: wxl@pku.edu.cn.**杨 亮**,男,1986 年生,硕士研究生,主要研究方向为系统虚拟化.**赖荣凤**,男,1984 年生,硕士研究生,主要研究方向为系统虚拟化.**王振林**,男,1970 年生,博士,副教授,主要研究方向为体系结构、编译、系统虚拟化.**罗英伟**,男,1971 年生,博士,教授,主要研究领域为系统虚拟化、地理信息系统.**李晓明**,男,1957 年生,博士,教授,主要研究领域为分布式系统、并行计算、互联网信息体系结构等.

问物理硬件,实现对 I/O 设备的虚拟化.该方法依赖 QEMU 对设备的模拟,因此实现简洁,但是由于 I/O 处理流程中涉及多个环境,切换较多,其 I/O 性能很不理想.虽然 KVM 较新的版本中已经将一些关键设备的虚拟化进行了优化,但是主要的设备,例如磁盘和网卡虚拟化的性能开销仍旧较大.

我们主要从客户操作系统的角度来发现问题.我们测试了 KVM 虚拟机磁盘和网卡的虚拟化性能,测试结果表明:由于目前运行在虚拟机上的客户操作系统是面向物理机器开发的普通操作系统,其中存在不适应虚拟化环境的因素,影响虚拟机的 I/O性能.根据测试结果,我们设计了一组优化方案,通过修改客户操作系统源码及其配置,达到优化 KVM 虚拟机的磁盘和网络 I/O 性能的目标.

2　KVM 虚拟化性能测试

比较 KVM 虚拟机系统和非虚拟化系统(Native)的 I/O 性能,了解 KVM 虚拟机 I/O 设备虚拟化开销,并分析其性能特征,其主要方法是分别在客户操作系统和宿主操作系统上运行磁盘和网络 I/O 的基准测试程序(benchmark).为了进一步了解性能开销可能的分布,我们修改 KVM,在基准测试程序运行时,获得指令级的客户操作系统的行为细节.

2.1　磁盘测试

测试环境:Intel Core2 Quad Q9550(2.83GHz),4GB 内存,SATA 硬盘(转速 7200),系统版本:KVM-76(默认配置),Linux 2.6.27.7.

测试内容:在虚拟机上运行 bonnie++[①],bonnie++ 使用文件系统调用,分别测试按块读/写的吞吐率和 CPU 利用率,并与 Native 环境下的测试结果进行对比.

按块写磁盘的性能测试结果如图 1 所示.

从图 1 中可看出,虚拟机按块顺序写磁盘的吞吐率和 CPU 利用率都只有 Native 的一半左右,存在较大的优化空间.

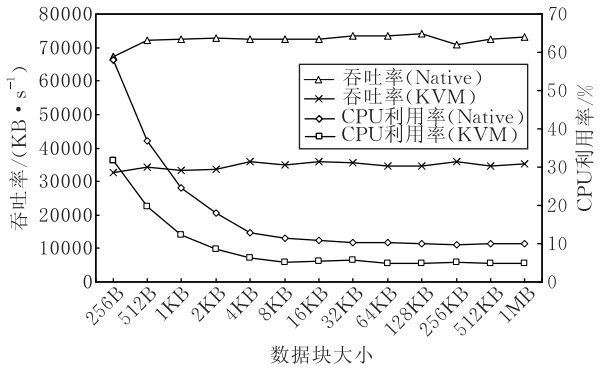

图 1　Native 和 KVM 虚拟机顺序写磁盘的吞吐率和 CPU 利用率比较

按块读磁盘的测试结果如图 2 所示.

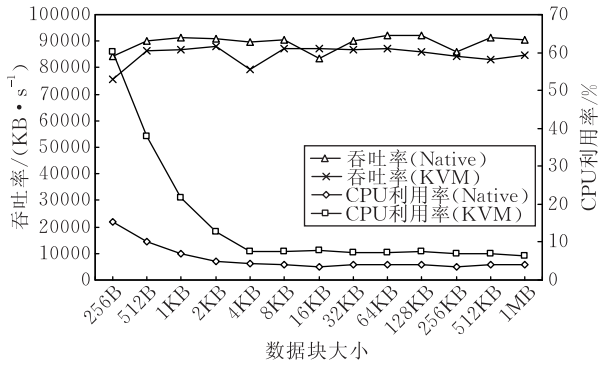

图 2　Native 和 KVM 虚拟机顺序读磁盘的吞吐率和 CPU 占用率比较

读磁盘时,如果缓存命中,就不用发生实际的读盘操作,则虚拟机可以不必陷入 VMM 进行处理,可以接近 Native 的性能,而其虚拟化开销主要表现于 CPU 的开销.因此,图 2 中,虚拟机按块顺序读磁盘的吞吐率与 Native 相当,而 CPU 占用率则比 Native高很多.

因此,在磁盘 I/O 方面,一是需要提高写磁盘的性能,二是需要降低读磁盘时 CPU 的利用率.而性能降低的具体原因,包括较高的虚拟机陷入频率、QEMU 模拟性能较低等.

2.2　网络测试

测试环境:物理机器 HostA 和 HostB,分别使用 Intel 的千兆网卡 82566DC 和 82567LM-2,以千兆网互联.在 HostB 上用 KVM 运行一个虚拟机 GuestB.

实验 1.　用 ping 测试虚拟化带来的网络数据包传输延迟.

测试内容:(1)物理主机 HostA ping 物理主机 HostB;(2)物理机 HostB ping 虚拟机 GuestB;(3)物理机 HostA ping 虚拟机 GuestB,分别测试之间的往返时延(RTT),测试结果如图 3 所示.

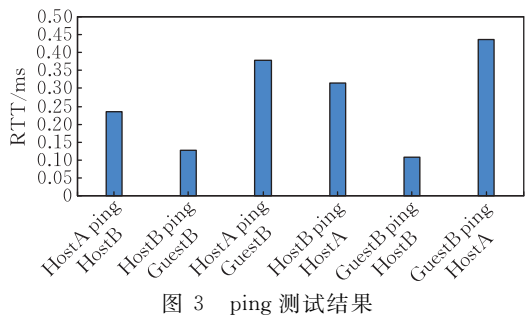

图 3　ping 测试结果

① bonnie++.[Online] http://www.coker.com.au/bonnie++/

从测试结果可以大致估计虚拟化带来的额外开销：数据包从 GuestB 传输到 HostA 的时间开销，相当于数据包从 GuestB 传输到 HostB 的时间加上数据包从 HostB 到 HostA 的传输时间. 也就是说，虚拟化的开销大致是 GuestB 到 HostB 的传输时间，从测试数据上看，这部分的时间开销约为 33%.

实验 2. Netperf 测试.

Netperf[①] 是用来测试网络吞吐率的开源基准测试程序. 主要是通过客户端和服务器端两个应用程序来进行网络数据的发送和接收，并计算出相关网络性能指标. 我们主要关心带宽(throughput)和延迟(latency).

测试过程：(1) 远端主机 HostA 上运行 netperf 的网络服务器端；(2) 本地主机 HostB 上运行 netperf 客户端进行性能测试；(3) 设置虚拟机 GuestB 使用不同的虚拟网卡(虚拟 rtl8139 网卡或虚拟 e1000 网卡)，分别运行 netperf 客户端进行测试. 测试结果如图 4 所示.

图 4 中，KVM 虚拟机的性能远低于 Native，但是，当虚拟机使用不同的网卡设置时，性能有一定的差异. 因此，模拟设备的处理能力对虚拟化环境下网络性能存在很大的影响. 如果为客户操作系统选择合适的模拟设备及适当的配置，可以得到相对较高的虚拟机网络 I/O 性能.

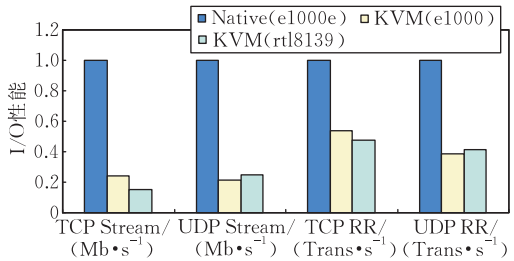

图 4 netperf 测试，虚拟机使用不同网卡配置

2.3 指令级测试

为了进一步分析性能下降的原因，我们抓取了虚拟机运行 SPECjbb[②] 和 netperf 时发生虚拟机陷入(VM Exit)的代码，其中最频繁导致虚拟机陷入的函数我们称之为“热点函数”，前 5 个热点函数显示在表 1 和表 2 中.

表 1 KVM 虚拟机运行 SPECjbb 时的热点函数

	陷入地址	陷入次数	陷入处理时间/s / 占总时间的百分比/%	发生陷入的函数
1	0xc041be3f	607251	4.29/17.66	*ack_ioapic_irq*
2	0xc042ffa6	602546	0.10/0.41	*__do_softirq*
3	0xc0407cf9	600625	0.38/1.55	*timer_interrupt*
4	0xc041add2	600223	8.57/35.28	*smp_apic_timer_interrupt*
5	0xc0407cf7	600092	3.08/12.68	*timer_interrupt*

表 2 KVM 虚拟机运行 netperf 时的热点函数

	陷入地址	陷入次数	陷入处理时间/s / 占总时间的百分比/%	发生陷入的函数
1	0xc04ee0fc	16545487	101.64/26.03	*ioread16*
2	0xc04ee0d1	14317411	85.33/21.85	*iowrite16*
3	0xc04ee153	7636364	62.26/15.94	*ioread32*
4	0xc04edfe4	6045896	44.53/11.40	*ioread8*
5	0xc0623d05	4401573	0.73/0.19	*_spin_lock_irqrestore*

结果显示，I/O 操作存在一定的热点效应，也就是说，少数指令导致了较多的虚拟机陷入. 我们可以针对热点代码进行优化.

此外需要注意的是 SPECjbb 的测试中，时钟中断(timer_interrupt)引起的虚拟机陷入排到了前五位，可以通过优化时钟中断优化 SPECjbb 的性能.

我们抓取了 bonnie++ 运行时的热点函数，前 4 个热点函数如表 3 所示. 由于这些热点函数在操作系统中会被多处频繁调用，因此我们还抓取了调用这些热点函数的上一级函数，表中的上层调用函数列出了调用热点函数最频繁的函数.

表 3 中，上层调用函数 *verify_pmtmr_rate* 用于调整主板时钟，该功能在虚拟机环境下冗余，可以消除该函数减少一部分虚拟机陷入. 热点函数的测试结果表明，目前的客户操作系统中存在一些操作影响了虚拟机的 I/O 性能，我们的优化工作主要针对这类问题进行.

表 3 KVM 运行 bonnie++ 时的热点代码

陷入地址	发生陷入的函数	上层调用函数地址	上层调用函数
0xc06442dd	*acpi_pm_read*	0xc06442e7	*verify_pmtmr_rate*
0xc0547047	*iowrite8*	0xc05e8fe3	*ata_sff_dev_select*
0xc0546f3c	*ioread8*	0xc05eaa44	*ata_bmdma_status*
0xc05470ff	*iowrite32*	0xc05ea9c2	*ata_bmdma_setup*

① netperf. [Online] http://www.netperf.org/netperf/

② SPEC JBB2005. [Online] http://www.spec.org/jbb2005/

3　优化方法及其效果

基于上一节的测试结果,我们发现,现有的客户操作系统是直接采用针对物理主机设计的操作系统,其中存在一些不适应虚拟化环境的因素,影响虚拟机的性能.

根据现有的分析,这些不适应的因素可以分为 3 类:(1)在虚拟化环境下一些引起短时间内频繁发生虚拟机陷入的操作.优化时,应该尽量降低这类操作发生的频率,或者可以通过批量处理进行优化.(2)冗余的操作.这些操作在虚拟化环境下不发生实际的作用;或者有些操作在客户操作系统和 VMM 中重复实现.冗余的操作通常可以直接去除.(3)可以由客户操作系统实现,也可以由 VMM 或者硬件实现的功能.通常情况下,越底层执行效率越高.因此,可以由 VMM 或硬件完成的功能通常都可以交由它们完成,尽量减少客户操作系统的工作,提高整体性能.

因此,我们试图针对 KVM 虚拟化环境的特征,修改客户操作系统的源码或其配置,达到降低环境切换代价、精简客户操作系统的目的,从而提高 KVM 虚拟机的 I/O 性能.针对第 3 类问题的优化已经有了较多的研究成果(详见 4.2 节),我们主要针对前两类客户操作系统的问题进行优化,本节讨论几种优化方法,并比较优化前后的性能.

3.1　降低环境切换代价

软件模拟的 I/O 虚拟化方法的每一次 I/O 处理过程存在多次环境切换,包括客户操作系统和 KVM 的切换、内核态的 KVM 和用户态的 QEMU 的切换以及 QEMU 进行系统调用以后用户态和宿主操作系统内核态的切换.其中除了客户操作系统和 KVM 的切换之外,其它切换都是 KVM 的设计构架所决定的,而客户操作系统和 KVM 的切换取决于客户操作系统的行为,且客户操作系统和 KVM 的切换通常会引起其它几种切换.因此,客户操作系统和 KVM 的切换频率是降低环境切换次数的关键,如果可以调整客户操作系统的行为,减少其切换到 KVM 的频率,则可以降低环境切换频率.通过上一节的指令级测试可以看出,KVM 的 I/O 操作存在一定程度的热点效应,有少量客户操作系统的代码引起了较多虚拟机陷入,可以针对这些代码进行优化.目前我们的工作主要包括以下两方面.

3.1.1　合并连续的 I/O 指令

客户操作系统的磁盘驱动中,存在一些代码段,包含了连续的 I/O 指令.每当执行到一条 I/O 指令,都会引起虚拟机陷入,使得这一段代码连续陷入.因此可以将这些指令合并成一个操作,一次陷入 KVM 进行处理.

我们采用了静态合并的方法,将一组 I/O 指令替换成一个 vmcall 调用(vmcall 类似于操作系统中的系统调用,客户操作系统中调用 vmcall 之后,将切换到 VMM 进行处理),一次主动陷入 KVM,替换方法是将每条 I/O 指令的信息,包括 in/out、端口、值,依次放入一个数组,该数组的地址和长度作为参数由 vmcall 传递给 KVM 处理.KVM 反过来从数组中将 I/O 操作的参数依次取出,逐条进行模拟.

例如,/driver/ide/ide-disk.c 中的*__ide_do_rw_disk* 函数,存在如下一段代码(代码段 1):

```
/driver/ide/ide-disk.c
hwif->OUTB(tasklets[1],IDE_FEATURE_REG);
hwif->OUTB(tasklets[3],IDE_NSECTOR_REG);
hwif->OUTB(tasklets[7],IDE_SECTOR_REG);
hwif->OUTB(tasklets[8],IDE_LCYL_REG);
hwif->OUTB(tasklets[9],IDE_HCYL_REG);
hwif->OUTB(tasklets[0],IDE_FEATURE_REG);
hwif->OUTB(tasklets[2],IDE_NSECTOR_REG);
hwif->OUTB(tasklets[4],IDE_SECTOR_REG);
hwif->OUTB(tasklets[5],IDE_LCYL_REG);
hwif->OUTB(tasklets[6],IDE_HCYL_REG);
hwif->OUTB(0x00|drive->select.all,IDE_SELECT_REG);
```

代码段 1　11 次连续的 I/O 操作

该段代码包括了 11 次连续的 outb,将会引起 11 次连续的虚拟机陷入.我们将这 11 条 outb 合并为一个 vmcall 调用,得到如代码段 2 所示的代码:

```
/driver/ide/ide-disk.c
struct io_insn io_out[11]; //保存 I/O 指令信息的数组
usigned long io_gpa, io_len;
#define IO_OUT(x, _type, _val, _port)   \
    io_out[x].type=_type,               \
    io_out[x].port=_port,               \
    io_out[x].val=_val

//把 I/O 指令的信息依次放入 io_insn 数组:
IO_OUT(0,OUTB,tasklets[1],IDE_FEATURE_REG);
IO_OUT(1,OUTB,tasklets[3],IDE_NSECTOR_REG);
IO_OUT(2,OUTB,tasklets[7],IDE_SECTOR_REG);
IO_OUT(3,OUTB,tasklets[8],IDE_LCYL_REG);
IO_OUT(4,OUTB,tasklets[9],IDE_HCYL_REG);
IO_OUT(5,OUTB,tasklets[0],IDE_FEATURE_REG);
IO_OUT(6,OUTB,tasklets[2],IDE_NSECTOR_REG);
IO_OUT(7,OUTB,tasklets[4],IDE_SECTOR_REG);
IO_OUT(8,OUTB,tasklets[5],IDE_LCYL_REG);
IO_OUT(9,OUTB,tasklets[6],IDE_HCYL_REG);
IO_OUT(10,OUTB,0x00|drive->select.all,IDE_SELECT_REG);

//由于 KVM 和客户操作系统所见地址空间不同,需要把 io_insn
//数组地址转换为物理地址,以便 KVM 访问:
io_gpa=virt_to_phys((unsigned long)io_out);
io_len=11; //I/O 指令数
vmcall(XKVM_IO_COALESCE,io_gpa,io_len,0);
//调用 vmcall 陷入 KVM,其参数包括 io_insn 的地址和长度
```

代码段 2　合并为一个 vmcall

原来的 11 次陷入变成 1 次陷入,可以极大地降低陷入 KVM 的频率.

我们修改了两个存在连续 I/O 操作的代码段,对修改前和修改后 inb、outb 调用点引起的虚拟机陷入数量进行了统计,结果如表 4 所示,修改后,陷入数量得到了极大的降低.

表 4 inb 和 outb 调用点引起的陷入数量
(连续的 I/O 指令合并前后的变化)

指令地址	修改前	修改后
0xc054fcc0	111050	17226
0xc054ecf0	15864	2343

该方式实际上是借鉴了半虚拟化[1]的思路,通过修改客户操作系统的代码,减少客户操作系统和 VMM 之间的切换.目前我们只实现了静态的修改,包括静态分析连续陷入的代码、静态合并指令.进一步的,我们可以实现动态的半虚拟化,也就是进行动态的代码段替换,包括在运行时进行动态监控,找到热点的代码段,动态合并指令,生成新的代码片段.然后通过加入跳转指令,或者通过替换包含热点代码的整个函数,实现代码段替换.或者,运行时,当一个 I/O 操作发生陷入,在 KVM 中预取客户操作系统的后续操作,当发现多个可以合并的 I/O 时,则一次模拟执行完毕,再返回客户操作系统.

我们比较了该优化前后的 I/O 性能,结果并不理想,除了 CPU 利用率有一定的降低,I/O 的吞吐率几乎没有变化.我们合并 I/O 操作的优化方式只减少了环境切换的开销,也就是减少了虚拟机陷入和返回所消耗的时间,并不对陷入之后在 KVM 和宿主操作系统中的操作进行优化,因此,优化结果说明这部分环境切换的开销还不是 I/O 性能开销的决定因素.

3.1.2 控制时钟频率

时钟中断是另一个频繁引起虚拟机陷入的原因,每个时钟中断到来时,都会引起虚拟机陷入 VMM.对于非实时应用,我们可以通过降低时钟频率,减少虚拟机陷入的频率.最简单的方法是直接修改客户操作系统的配置.

我们进一步实现了对客户操作系统透明的降低时钟频率的方法.KVM 虚拟机的时钟是由 KVM 模拟 PIT 实现的,PIT 可以产生 clock 信号,其频率近似是 1193181Hz,每当 clock 信号到来时,PIT 通道 0 的计数器自动减 1,当计数器减到 0 时,就通过 IRQ0 向系统产生一次时钟中断.KVM 虚拟的 PIT 也是这样的工作原理.因此,通过修改虚拟 PIT 通道 0 计数器的初始值,就可以设定时钟中断的频率,例如,1000Hz 的时钟频率,PIT 通道 0 计数器初始值为 1193181/1000=1193,100Hz 时,则为 1193181/100=11932.也就是说,如果修改系统初始化时写入 vPIT 通道 0 的计数器值,就可以对客户操作系统透明地修改其时钟频率.具体方法是修改 *pit_load_count* 函数中写入 vPIT 通道 0 的初始值.

我们实验了在客户操作系统时钟频率为 1000Hz(vHZ100)的情况下,由 KVM 将时钟频率实际设定为 100Hz(vHZ100),比较了修改前后的虚拟机 I/O 性能,如图 5 所示.

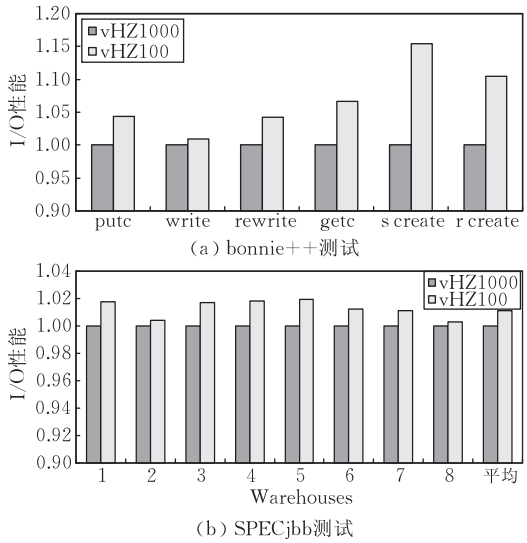

图 5 控制虚拟时钟频率的效果

结果表明,降低时钟频率能够一定程度地提高虚拟机地性能.

3.2 精简客户操作系统

冗余的操作是指在虚拟化环境下不发生实际的作用或者在客户操作系统和 VMM 或宿主操作系统中重复实现的操作.我们需要找到这些可能冗余的操作,将其从客户操作系统中去除,从而优化 KVM 虚拟机的 I/O 性能.

3.2.1 消除冗余函数

一个例子是在磁盘 I/O 的热点代码分布测试中,表 3 中的函数 *verify_pmtmr_rate*,该函数的主要工作是调整主板时钟,该操作在虚拟机环境下是多余的.对于这种操作,只需要修改客户操作系统,简单将其去除,就可以适当地提高客户操作系统的运行效率.实验结果如图 6 所示.

比较去除 *verify_pmtmr_rate* 函数前后虚拟机

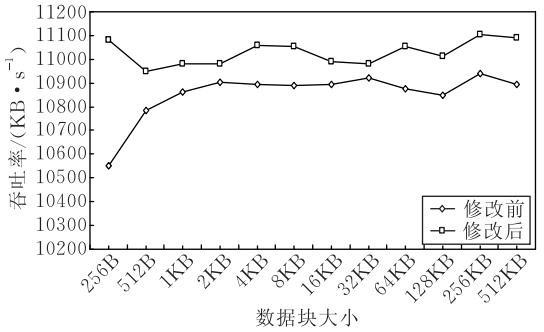

图 6　去除 *verify_pmtmr_rate* 函数前后虚拟机写磁盘性能

磁盘 I/O 的性能,可以看到,去掉该函数以后,虚拟机的写性能有了一定的提升.

3.2.2　消除冗余调度

在 I/O 请求提交驱动之前,操作系统通常要对 I/O 请求进行调度:通过对磁盘 I/O 请求的合并和排序,降低磁盘的寻道时间,提高磁盘旋转时读取数据的效率,从而降低 I/O 操作的总时间,增加单位时间内的 I/O 操作次数,提高系统整体的 I/O 效率.

KVM 基于宿主操作系统的结构,使得客户操作系统的 I/O 操作总是通过 QEMU 进行系统调用,在宿主操作系统内核中再逐层向下传递到物理硬件.该流程使得客户操作系统的 I/O 请求要经过客户操作系统和宿主操作系统的两次 I/O 调度,客户操作系统中的调度结果必然要由宿主操作系统再次调度,显然冗余.一方面,客户操作系统并不知道物理磁盘信息,这次调度不一定有效;另一方面,当多个客户操作系统同时运行时,通常情况下,所有 I/O 操作需要由宿主操作系统统一调度以获得整体性能的最大化.基于这两方面原因,我们去掉了客户操作系统中的磁盘 I/O 调度,使得所有 I/O 请求都直接提交给驱动,这样,既减少了冗余的操作过程,也使客户操作系统的 I/O 请求获得立即处理.

当宿主操作系统采用默认的 CFQ I/O 调度器①时,经过该优化,虚拟机按块写的性能提高了 6.01%;而虚拟机按块读的性能提高了 5.64%.但是,由于优化以后,客户操作系统中的所有 I/O 请求都按照先进先出的顺序进行处理,因此,难以对各进程的 I/O 带宽进行分配和控制.

3.2.3　简化客户操作系统网卡驱动

网卡的 NAPI 技术是指以主动查询结合中断的方式降低中断发生的频率.在虚拟化环境中,客户操作系统所看到的是虚拟网卡,虚拟网卡并不能直接产生中断,并可能采用其它特定的事件通知机制代替中断机制.因此,客户操作系统网卡驱动中的 NAPI 功能多余,反而会浪费 CPU 时间.因此,我们去掉了客户操作系统网卡驱动中的 NAPI 功能,并测试比较了优化前后的系统性能,结果如图 7 所示.

图 7 中可以看到该优化较大程度地改善了网络吞吐率,性能的提升超过了 60%.

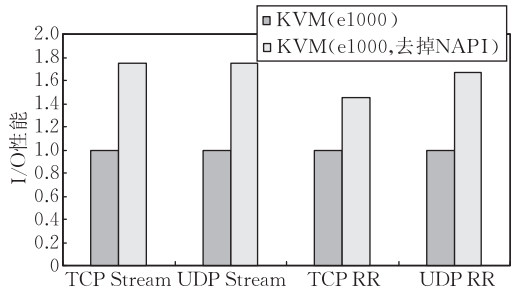

图 7　比较去掉 NAPI 前后,netperf 的性能

4　相关工作比较

KVM 虚拟机的 I/O 性能优化包括两方面:降低环境开销和优化客户操作系统.

降低环境切换开销主要可以通过两个方面实行:(1)降低环境切换的频率;(2)降低每次切换的开销.

降低环境切换频率主要是通过批量化的提交 I/O 操作,或者在可能的情况下减少设备中断.Sugerman 等[2]在 VMware Workstation 上实现了批量发送:当切换的频率超过一定阈值时,可以将客户操作系统每一次发送数据的请求缓存到一个队列中,当下一次中断发生时,一次切换到 Host 进行中断处理和数据发送.该优化同时能降低虚拟 IRQ 传送的开销,因为每一批数据传送完成后,只需要传递一个 IRQ.同时,Sugerman 等[2]考虑优化客户操作系统的驱动协议,设计一个利于虚拟化的调用接口,例如可以节省大量访问设备状态的 I/O 指令,能避免虚拟机陷入 VMM;也可以降低虚拟 IRQ 的发生率,从而降低 VMM 和 VM 的切换频率.借鉴半虚拟化的思路,Russell[3]等提出了 virtio,virtio 是一种通用的设备驱动,为块设备和网卡提供了相同的处理流程,可以用于多种不同的 VMM.其实现主要是在客户操作系统和 VMM 之间提供一个基于共享内存的环状缓冲区,客户操作系统和 VMM 分别作为消费者和生产者,通过环状缓冲区进行数据交换,此外,实现了一套事件通知机制,当

① Axboe J. Time Sliced CFQ I/O Scheduler. [Online] http://kerneltrap.org/node/4406

有数据被放到缓冲区以后,作为消费者的一方会接到通知,这套事件通知机制支持屏蔽,因此可以实现批量操作,避免频繁发生环境切换.

降低每一次切换的开销是要减少切换时所做的工作,Sugerman 等[2]修改了 VMware Workstation 上运行的客户操作系统的进程切换:当切换到空闲进程(idle process)时,不需要切换进程的页表结构,因为 idle 是一个内核线程,可以使用任何一个进程的内核页表.该优化把 MMU 引起的虚拟化开销降低了近一半.Sugerman 等[2]还对 VMware Workstation 提出了旁路宿主操作系统的优化:因为 VMware Workstation 的每一次虚拟机的 I/O 操作都要经过客户操作系统-VMM-宿主操作系统之间的切换过程,若由 VMM 直接访问硬件设备,则能降低至少一半切换开销.

切换时的开销还有一方面表现为切换后 TLB、Cache 等会发生失效. Menon 等[4]优化了 Xen 使客户操作系统支持超大页(superpage)和全局页面映射(global page mapping)等高级虚拟内存功能,极大降低了 TLB 的失效率.

此外,如果虚拟机能够直接访问硬件设备,则可极大降低环境切换的开销.近年来,出现了一些新技术和规范,如 Intel VT-d①、AMD IOMMU② 能够保证虚拟机 I/O 地址空间的隔离性,并在 PCI 总线桥上安装一个类似传统 MMU 的地址转换设备,提供硬件中断转发机制;PCI-SIG 提出了 SR-IOV 和 MR-IOV 规范③,旨在定义一个可分割共享的 I/O 设备的规范,按照该规范设计的 PCIe 设备,将具有统一的、可由多个虚拟机分割共享的界面.目前已经有 SR-IOV 设备面世,Dong[5-6] 等已经实现了虚拟机对 SR-IOV 网卡的直接访问,并达到了接近 Native 的性能.这些硬件支持的出现,使得虚拟机能够直接访问物理硬件,从而提高了性能,也简化了 I/O 设备虚拟化的实现.因此,针对软件的优化应该着重于如何优化客户操作系统,使得它更适应虚拟化环境,或者使客户操作系统、VMM 和宿主操作系统配合得更有效.

Ram 等[7]对客户操作系统的优化包括 3 个方面:(1)实现 LRO(Large Receive Offload),该技术通过将多个数据包中的数据合并成一个 TCP/IP 数据报,使一次协议栈的处理能处理大量数据.(2)软件预取,每次处理虚拟中断时,通常是循环处理一组数据包,可以在每轮循环中,访问下一轮要处理的数据报的报头,进一步地访问下两轮要处理的数据报的缓存结构,从而降低访存的延迟.(3)降低缓冲区大小,将缓冲区改成半个页面的大小,能够减小缓冲区的工作集,从而降低 TLB 的失效率.

Menon 等[4]改进了 Xen 的虚拟网卡接口,提供了现有的大部分网卡支持的分载(offload)功能(若底层的硬件并不支持分载,则由 Driver Domain 模拟,也能提高性能),包括散/聚(Scatter/Gather) I/O,TCP/IP 校验和分载(TCP/IP Checksum Offload)、TCP 分段分载(TCP Segmentation Offload,TSO).Scatter/Gather I/O 支持不连续的内存地址做连续的 DMA 操作;TSO 减少了整个系统需要处理的数据包数量,提高了系统的整体性能;TCP/IP Checksum Offload 降低了客户操作系统的负载.

我们主要从两个方面进行优化:一是降低环境切换开销,二是消除客户操作系统中的冗余操作.

通过修改客户操作系统,使邻近的 I/O 操作批量陷入 VMM,并且降低时钟频率,从而降低虚拟机陷入(VM Exit,指虚拟机在执行到敏感指令时从虚拟机运行状态陷入到虚拟机管理器(Virtual Machine Monitor,VMM)的运行状态)频率.通过修改客户操作系统,可以去掉那些只有在非虚拟环境下发挥作用的操作,减少了虚拟机的运算量.

我们的优化工作,一方面可以用于要求高 I/O 性能的环境;更重要的,虚拟机的早期目标仅在于提高物理机器的利用率,考虑的是如何将大量遗产服务器整合到部署了虚拟机的少量主机上,因此,虚拟机上运行的操作系统必须是遗产操作系统,这些操作系统是面向物理机器开发的,包含了一些不适应虚拟化环境从而影响虚拟机性能的因素.但是当虚拟机的应用越来越广泛,虚拟机的用途早已不限于整合遗产服务器,对虚拟机的性能和可用性提出了更高的要求.面向虚拟化环境,需要进一步研究如何裁剪并补充原有的操作系统,使其更适应虚拟化环境.我们已经看到,Linux 内核已经在逐渐纳入对虚拟化的支持.但是,什么样的操作系统更适合虚拟化

① Intel Corporation. Intel® Virtualization Technology for Directed I/O Architecture Specification. [Online] http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_ IO. pdf. 2007

② Advanced Micro Devices, Inc. AMD I/O Virtualization Technology (IOMMU) Specification. [Online] http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/344 34. pdf. 2007

③ PCI-SIG. I/O Virtualization. [Online] http://www.pcisig.com/specifications/iov/, 2007

的运行环境? 如何构造一个简洁高效的虚拟机操作系统? 还没有较为系统的结论. 我们的工作表明操作系统中的影响性能的因素分布较散,类型多样,目前还难以提出一种通用的解决方案,本文的工作可以为今后系统性研究提供借鉴.

5　结束语

对 KVM 虚拟机 I/O 性能的测试表明,现有的客户操作系统中存在一些不适应虚拟化环境的因素,影响了虚拟机的 I/O 性能. 因此针对 KVM 虚拟机环境,我们修改了客户操作系统,降低了虚拟机运行时发生环境切换的频率,简化了客户操作系统的操作,优化了 KVM 虚拟机的 I/O 性能. 现有的硬件虚拟化支持已经极大地提高了虚拟机的性能并降低了虚拟化的实现复杂度,因此,未来的方向应该着重改进操作系统,使得操作系统更加适应虚拟化环境. 下一步工作中,我们继续从客户操作系统入手,找到其不适应虚拟化 I/O 环境的其它因素进行优化. 此外还要进一步实现动态优化,从而无需修改客户操作系统的源码,而在其运行时,根据其运行特征,对影响虚拟机性能的操作进行动态的优化和二进制代码替换.

参　考　文　献

[1] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization//Proceedings of the 19th ACM Symposium on Operating Systems Principles. Bolton Landing, NY, USA, 2003: 164-177

[2] Sugerman J, Venkitachalam G, Lim B H. Virtualizing I/O device on VMware workstation's hosted virtual machine monitor//Proceedings of the General Track: 2001 USENIX Annual Technical Conference. Boston, MA, USA, 2001: 1-14

[3] Russell R. Virtio: Towards a De-Facto standard for virtual I/O devices. Operating System Review, 2008, 42(5): 95-103

[4] Menon A, Cox A L, Zwaenepoel W. Optimizing network virtualization in Xen//Proceedings of the Annual Conference on USENIX'06 Annual Technical Conference. Boston, MA, USA, 2006: 15-28

[5] Dong Y, Yu Z, Rose G. SR-IOV networking in Xen: architecture, design and implementation//Proceedings of the 1st Workshop on I/O Virtualization. San Diego, USA, 2008, article No. 10

[6] Dong Y, Dai J, Huang Z, Guan H, Tian K, Jiang Y. Towards high-quality I/O virtualization//Proceedings of the SYSTOR 2009: The Israeli Experimental Systems Conference. Haifa, Israel, 2009, article No. 12

[7] Ram K K, Santos J R, Turner Y, Cox A L, Rixner S. Achieving 10 Gb/s using safe and transparent network interface virtualization//Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. Washington, DC, USA, 2009: 61-70

ZHANG Bin-Bin, born in 1982, Ph. D. candidate. Her research interests focus on system virtualization.

WANG Xiao-Lin, born in 1972, Ph. D., associate professor. His research interests include system virtualization and geographic information system.

YANG Liang, born in 1986, M. S. candidate. His research interests focus on system virtualization.

LAI Rong-Feng, born in 1984, M. S. candidate. His research interests focus on system virtualization.

WANG Zhen-Lin, born in 1970, Ph. D., associate professor. His research interests include computer architecture, compiler and system virtualization.

LUO Ying-Wei, born in 1971, Ph. D., professor. His research interests include system virtualization and geographic information system.

LI Xiao-Ming, born in 1957, Ph. D. professor. His research interests include distributed system, parallel computing and Internet information architecture.

Background

Recently, virtualization technology is developing quickly. Some excellent virtual machine monitors are more and more widely used. Via optimizations in software and extension in hardware, virtualization of CPU and memory is now much improved. The performance is nearly the same to the Native system, and the complexity is also reduced. But performance in I/O virtualization is still a challenge, and there are no ideal solutions yet. Although the hardware extension helps to improve the performance and complexity, but there is still room for improvement. So there are still many resear-

ches on I/O virtualization. During the research, the authors found that the I/O performance of virtual machine is affected by the behavior of the Guest OS. Now, the general OSes, which are oriented physical machines, are directly used in virtual environment. So there are some behaviors causing the performance degradation.

Therefore, the work focuses on locating these inefficient operations in Guest OS and tries to modify them. That is tried to optimize the Guest OS to adapt the virtual environment for better performance. The experiments are based on KVM, but the optimizations can apply to other VMMs. The performance of disk and network in KVM is enhanced after optimizations, and sometimes the throughput is even increased by more than 60%. What are the requirements of an OS running in the virtual environment? How can make it efficient and simple? Guest OS modification requires further research. The authors' work is a beginning.

This work researches on I/O virtualization. It is a part of our projects about virtual machine research. These projects are supported by the National Grand Fundamental Research 973 Program of China under grant No. 2007CB310900, National Science Foundation of China under grant No. 90718028 and No. 60873052, National High Technology Research 863 Program of China under grant No. 2008AA01Z112, and MOE-Intel Information Technology Foundation under grant No. MOE-Intel-10-06.

Our past work includes whole system migration of virtual machine (migration including disk storage) and optimizations of memory virtualization. And we will continue our work on optimization of I/O virtualization.