doi:10.3772/j.issn.1002-0470.2020.09.003

龙芯 KVM 虚拟机 I/O 中断子系统的优化①

朱 琛②[*,**,***,****] 王 剑[*,**,***,****] 高 翔[****] 毛碧波[****] 李 星[****]

([*]计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

([****]龙芯中科技术有限公司 北京 100190)

摘 要 本文针对虚拟机 I/O 中断子系统的性能问题,以龙芯 KVM 虚拟机为实验平台,分析了 KVM 虚拟机中 I/O 中断子系统的性能瓶颈,并在 KVM 中实现虚拟 I/O 中断控制器的基础上,进一步采用只读页表代替陷入、类虚拟化等手段对其进行了优化。测试结果显示虚拟机处理 I/O 中断的吞吐量提升了 300% 以上。在部分 I/O 中断较频繁的网络和磁盘测试中,性能有 60.9% ~215.1% 的提升,文中的优化方法同样适用于其他架构的 KVM 虚拟机。

关键词 KVM 虚拟机; I/O 中断; I/O 虚拟化

0 引 言

随着 CPU 硬件辅助虚拟化技术的成熟,虚拟机技术得到了广泛的应用[1,2]。KVM(kernel-based virtual machine)是一种主流的基于 Linux 内核的虚拟机监控器(virtual machine monitor, VMM),有着优秀的可管理性和性能。配合 CPU 的硬件辅助虚拟化技术,在运算、访存密集型的应用中,能够达到宿主机相近的性能,是目前云计算使用的主流方案之一。在 KVM 虚拟机中,I/O(input/output)虚拟化目前应用最广泛的方式仍是不依赖硬件辅助的软件模拟。在 I/O 中断密集的场景中,软件模拟的 I/O 中断子系统成为影响虚拟机性能的瓶颈。本文以龙芯 KVM 虚拟机[3,4]为例介绍了虚拟 I/O 中断子系统的原理,对虚拟 I/O 中断子系统的性能瓶颈进行了分析,除了采用常规方式优化外,还在此基础上尝试使用多种手段对其性能进行进一步的优化,取得了较好的优化效果。

本文的内容如下:第 1 节以龙芯的 GS464E[5] 高性能 CPU 核为例介绍了 I/O 虚拟化和 I/O 中断虚拟化的基本原理。第 2 节分析了 I/O 中断子系统的性能瓶颈。第 3 节介绍了其他架构常用的优化方法,并对其进行了验证。第 4 节在常规优化方法的基础上进行了进一步的优化,并对效果进行了初步评估。第 5 节选择部分 I/O 中断密集的磁盘、网络测试项,验证了优化的效果。最后在第 6 节进行了总结。

1 背景

1.1 软件 I/O 虚拟化

I/O 虚拟化指的是外设相关的虚拟化,因为从 CPU 的角度来看,外设是通过一组 I/O 寄存器来访问的[6]。I/O 虚拟化的实质是虚拟机通过 VMM 构建的虚拟设备(以下简称 VDEV)复用有限的宿主机外设资源。虚拟设备为虚拟机模拟真实外设访问的

① 国家自然科学基金(61432016)资助项目。

② 男,1983 年生,博士生;研究方向:计算机系统结构;联系人,E-mail: zhuchen@ict.ac.cn

(收稿日期:2019-10-25)

效果,其本身又是宿主机外设驱动程序的一个客户端,可以通过宿主机操作系统提供的 API 访问真实物理外设,实现对真实外设的复用[7]。

I/O 虚拟化以 CPU 虚拟化技术为基础。以龙芯 GS464E 为例,该处理器有 4 种运行状态,即根模式用户态、根模式内核态、客模式用户态、客模式内核态。其中客模式的 2 个状态是专为虚拟化新增的。4 种状态在一定条件下会相互转化,虚拟机在客模式运行,在执行部分特权指令和触发特定异常的情况下会发生例外陷入根模式内核态,其状态机如图 1所示[8]。

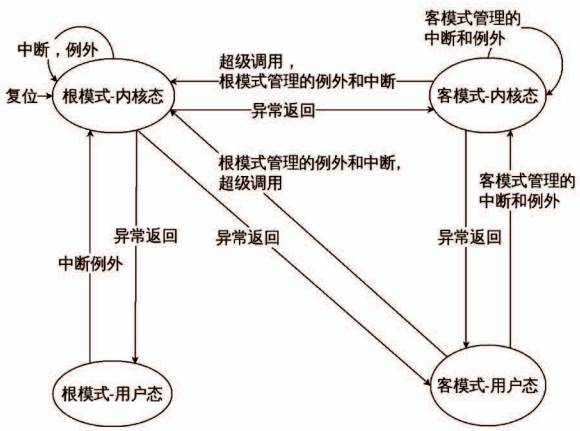

图 1　GS464E 模式状态机

虚拟机在宿主机操作系统看来是一个用户进程。虚拟机通过该进程获取宿主机的 CPU、内存、外设等物理硬件资源,并在不同线程之间共享。虚拟机的每个虚拟 CPU(以下简称 VCPU)有一个独立线程,虚拟机的内核和文件系统均在 VCPU 线程中运行。模拟 VDEV 的 QEMU 运行在根模式用户态。负责 CPU、内存虚拟化的 KVM 则运行在根模式内核态,是宿主机内核的一部分。会引起客模式陷入根模式内核态的例外如表 1 所示。其中 I/O 虚拟化用到的主要是转译后备缓冲器(translation lookaside buffer, TLB)例外。

当虚拟机在 VCPU 线程中访问 I/O 端口时,由于该地址不在 TLB 中,引起 TLB 例外陷入根模式内核态,随后由 KVM 处理,KVM 在判定地址不是内存且自己不能模拟后,会返回 QEMU 模拟。CPU 与外设的交互手段,除了寄存器访问和 I/O 中断之外还有直接存储器访问(direct memory access, DMA)。虚拟机中的 DMA 是 QEMU 通过 CPU 内存拷贝模拟的。

表 1　GS464E 中陷入根模式的例外

类型	原因
Reset/Soft Reset	硬件或软件复位
NMI	不可屏蔽的中断
Cache Error	Cache 校验错误
Hypercall	超级调用
TLB	TLB 例外
GPSI	敏感操作
FC	敏感状态
GRR	保留指令

1.2　I/O 中断虚拟化

I/O 中断虚拟机是中断虚拟化的一部分,支持硬件虚拟化辅助的 CPU 一般都会提供一定程度的中断虚拟化辅助。以 GS464E 处理器核为例,该处理器的协处理器 0(co-processor 0,简称 CP0,MIPS 中负责处理例外的协处理器)中有用于客模式中断辅助的相关寄存器,如表 2 中所示。

表 2　GS464E 中的虚拟机中断辅助寄存器

辅助寄存器	功能
GuestCtl0. PIP	硬件中断直通使能
GuestCtl2. VIP	软件注入中断
Guest. Cause. IP	中断原因寄存器

其中 Guest. Cause. IP(以下简称 GCIP)记录虚拟机的中断状态。当 GCIP 被置位时,虚拟机会陷入客模式管理的例外,跳转到对应中断向量入口执行处理程序。该寄存器不能由软件直接置位,而是由 GuestCtl0. PIP(以下简称 GPIP)、GuestCtl2. VIP(以下简称 GVIP)和硬件中断(以下简称 HWIP)共同决定的。其中断逻辑图如图 2 所示。其中的 n 代表是第几位,取值范围为 0 ~ 7。

通过设置 GPIP 和处理器的硬件中断路由器也可以将特定物理外设的 I/O 中断直接注入给虚拟机,但虚拟机使用的大部分外设是软件模拟的,且 GCIP 只有 8 位,一般只有 1 位用于 I/O 中断,因此

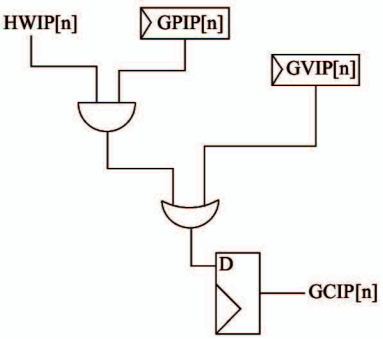

图 2　GS464E 客模式中断逻辑

I/O 中断采用的是软件注入方式,使用虚拟 I/O 中断控制器来管理多个虚拟外设中断。虚拟 I/O 中断控制器行为逻辑上模拟特定的物理 I/O 中断控制器,龙芯 KVM 虚拟机中使用的虚拟高级 I/O 中断控制器(以下简称 VIOAPIC)模拟自龙芯 7A 桥片内置的高级 I/O 中断控制器。VIOAPIC 自身也是一个虚拟外设。虚拟机 I/O 中断是 VCPU 和虚拟外设之间的一种同步手段。当虚拟外设发生了需要与 VCPU 线程同步的事件时,会将该请求通知 VIOAPIC。VIOAPIC 会根据中断触发逻辑置位 GVIP 寄存器。龙芯 KVM 虚拟机中断结构如图 3 所示。客模式中断被置位后,虚拟机会直接跳转到客模式内核对应的例外入口处理。

从硬件结构上看,I/O 中断控制器之后还有用于多核中断分发的中断路由器也需要模拟。在这里为了简化,将其看成是 VIOAPIC 的一部分,不单独分列。

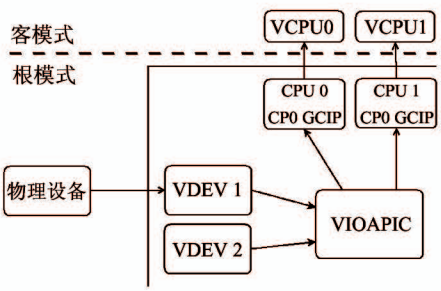

图 3　龙芯 KVM 虚拟机中断结构图

VIOAPIC 有中断线和中断消息(MSI)2 种触发逻辑,两者在虚拟机中都是软件模拟,没有本质区别。本文使用中断线模式,后续的优化方法对 MSI 和中断线 2 种模式都有效。虚拟 I/O 中断控制器不直接涉及真实外设,如果不考虑中断注入延迟导致的外设空闲,可以认为其只消耗 CPU 资源。

2　影响性能的主要问题

一个 I/O 中断从虚拟设备发出请求,到最终处理完毕可以分成 2 个阶段:中断注入和中断响应。在宿主机和虚拟机都以 1.2 G 龙芯 3A3000 单核运行并屏蔽其他中断的情况下,使一个虚拟设备连续触发 I/O 中断,分别测试虚拟机以不处理直接返回和正常 I/O 中断响应流程 2 种方式完成 100 万次循环所需的时间再除以循环次数,以此来估算中断注入和中断响应所需的时间。测试结果显示,100 万次中断注入耗时 2.68 s,以此估算一次完整中断注入流程实测需要约 3 217 个时钟周期。而一次中断注入加一次中断响应流程则耗时 76.48 s,需要约 91 769个时钟周期。减去中断注入流程消耗的时钟周期,一次中断响应需要 88 552 个时钟周期。因此影响 I/O 中断子系统性能的主要是中断响应。

虚拟 I/O 中断响应与宿主机 I/O 中断响应在客模式运行的代码是完全一致的,CPU 的主要行为区别是 I/O 寄存器的访问。一次 I/O 中断处理流程需要 8 次访问 VIOAPIC 的寄存器,如果增加 1 次冗余的寄存器读取,一次中断响应流程增加约 6 886 个时钟周期。以此估算,访问 VIOAPIC 的寄存器消耗了中断响应流程 62.2% 的时间,是影响虚拟 I/O 中断子系统性能的主要因素。

龙芯 7A 的中断响应流程如图 4 所示。流程图中的(A)、(B)、(C)、(D)步骤均有对 I/O 寄存器的访问。其中(A)、(B)、(D)中访问的是 VIOAPIC 的寄存器,(C)中访问的是 VDEV 的寄存器。

虚拟机中断处理的数据流图如图 5 所示。图中的(A)、(B)、(C)、(D)与流程图中的符号相对应。数据流图中的粗线表示会引发 CPU 例外或状态切换的动作,其中实线代表访问 VIOAPIC 的寄存器,虚线代表访问的 VDEV 的寄存器。本文中后续数据流图的含义也与此相同。由于(C)中对 VDEV 寄存器的访问不是本文的重点,且没有变化,后续的数据流图略去了这一部分。

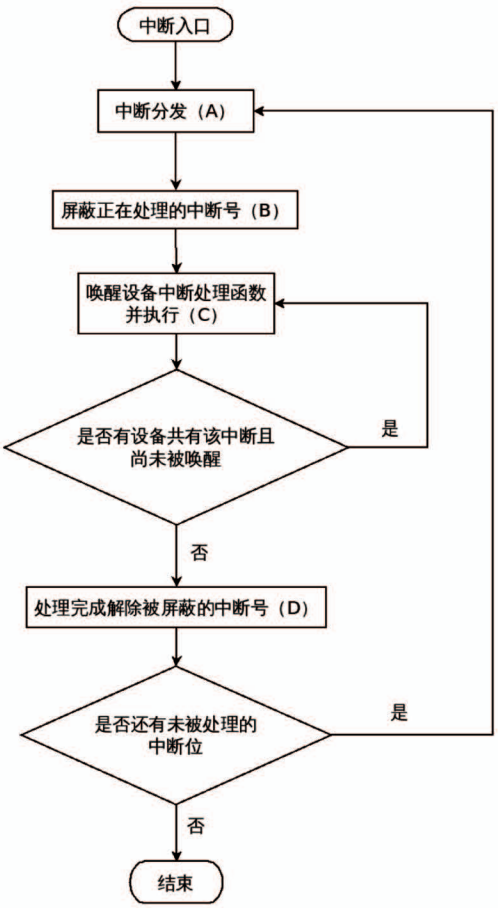

图 4　龙芯 7A I/O 中断响应流程图

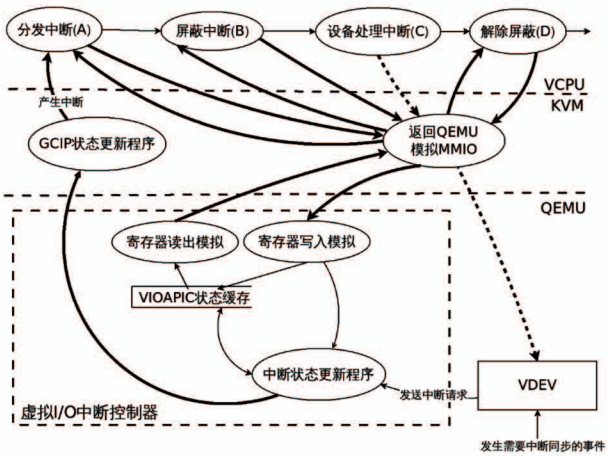

图 5　7A I/O 中断处理流程数据流图

从图 5 可知，VCPU 对 VIOAPIC 的一次寄存器访问需要经过 VCPU- > KVM- > QEMU- > KVM- > VCPU 的流程，其中每个步骤都需要保存恢复 CPU 的若干状态信息和线程信息，这是模拟开销大的主要原因。

3　常用优化方法

优化 I/O 模拟的常用优化方法是将 I/O 外设放到 KVM 中模拟，在 KVM 中模拟 VIOAPIC，中断注入区别不大，但在中断响应流程中，VCPU 访问 VIOAPIC 的寄存器，KVM 可以直接将数据返回给 VCPU，不需要再经过 QEMU，模拟流程简化为 VCPU- > KVM- > VCPU。经测试，访问一次 KVM 中模拟的 VIOAPIC 寄存器的开销约为 2 265 个时钟周期，一次完整的 I/O 中断响应消耗大约 37 985 个时钟周期，比在 QEMU 中模拟减少了 57% 的 CPU 开销，访问中断控制器的寄存器耗时占比从 62.2% 下降到 47.7%。优化后的数据流图如图 6 所示。

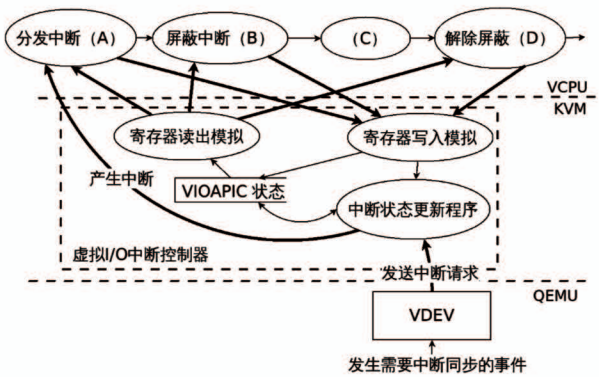

图 6　在 KVM 中模拟 VIOAPIC 的数据流图

4　进一步优化

虽然在 KVM 中模拟 VIOAPIC 明显提升了 I/O 中断响应的速度，但其中仍有近一半的耗时用于访问 VIOAPIC 的寄存器，还有继续优化的空间。

4.1　使用物理内存模拟寄存器读取

龙芯 GS464E 对寄存器访问的操作都是用普通访存指令完成的。在中断响应过程中，VCPU 对寄存器的读是为了获取中断控制器的状态，这些状态在中断产生前就已经到达了 KVM。可以提前将其保存到虚拟机可以访问的内存中，避免 VCPU 陷入根模式模拟。部分写寄存器由于涉及到 GCIP 的状态同步，仍需要被 KVM 捕获。VCPU 通过 0xe010000000 ~ 0xe010000fff 的地址访问 VIOAPIC

的寄存器。而在 0xe010000000 ~ 0xe01000ffff 的地址空间没有任何内存或其他设备。因此可以将寄存器的值填入一个 4 kB 的物理内存页中,将该页映射到虚拟机的 0xe010000000 ~ 0xe01000ffff 地址,并将该页在虚拟机页表中设置为只读属性,这样就能实现 VCPU 直接读取内存获得寄存器的值,而对相关地址的写会陷入 KVM。

虚拟机的页表填入过程需要 4 个地址:虚拟机虚拟地址(guest virtual address, GVA),虚拟机物理地址(guest physical address, GPA),宿主机虚拟地址(host virtual address, HVA),宿主机物理地址(host physical address, HPA)。GS464E 的虚拟机和宿主机共用页表项,虚拟机和宿主机的页表项用 2 个辅助标识 MID 和 VPID 区分,其中 MID 区分宿主机和虚拟机,VPID 区分是哪一个虚拟机。虚拟机的页表项只保存 GVA 到 HPA 的对应关系。

内核中利用 MIPS 的非缓存物理地址窗口访问寄存器[9],如 0xe010000000 对应的访问地址就是 0x90000e010000000,即内核访问的 GVA。修改后虚拟机中断响应的数据流图如图 7 所示。在页表项没有被换出的情况下,对 VIOAPIC 寄存器的读操作都不需要陷入 KVM,直接访问内存,只有寄存器写操作需要陷入 KVM 模拟。需要说明的是,GS464E 中虚拟机访问地址的缓存属性不受 MIPS 的非缓存和缓存窗口的约束,而是由页表属性决定的。

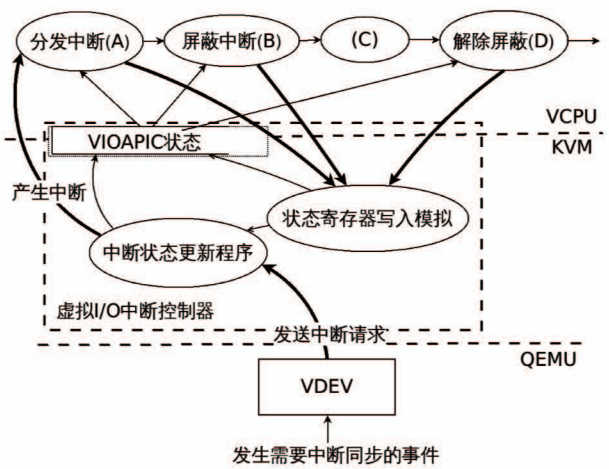

图 7　使用只读页表优化后的中断响应数据流图

优化后的一次完整 I/O 中断响应流程消耗大约 23 227 个时钟周期,较单纯的 KVM 模拟减少了 38.9%,I/O 中断控制器的寄存访问消耗占比进一步下降到了 29.2% 左右。这一优化不需要改动虚拟机,保持了虚拟机内核和宿主机内核的兼容性。

4.2　优化虚拟机的 I/O 中断控制逻辑

在 I/O 虚拟化中,在保持虚拟机用户态程序和宿主机完全兼容的情况下,通过修改少量内核态逻辑使其更适合虚拟化环境的方法,被称为类虚拟化。通过分析本文认为类虚拟化的方法也适用于 VIOAPIC 的优化。在中断线模式下 VIOAPIC 3 个最核心寄存器是中断请求寄存器(interrupt request register, IRR),中断状态寄存器(interrupt status register, ISR)和中断屏蔽寄存器(interrupt mask register, MASK),其三者的逻辑关系为:

ISR = IRR&(~ MASK)

其中中断响应过程中 VCPU 主要访问 ISR 和 MASK 寄存器,IRR 寄存器由 VDEV 更改。如果有任何新的中断请求 VIOAPIC 会先更新 IRR,再根据上述逻辑关系更新 ISR 最终更新目标 VCPU 所运行的物理 CPU 的 GVIP 寄存器。而对于 VCPU 来说,只能通过 MASK 来更新 ISR。

I/O 中断响应过程中,VCPU 更新 MASK 主要有 2 个作用:即屏蔽中断和解除中断的屏蔽。

由于 GCIP 是 VIOAPIC 软件注入的,只需确保下次 VIOAPIC 更新 GVIP 时,MASK 的值能够被 VIOAPIC 正确得知即可,因此屏蔽中断不需要同步操作。解除中断的屏蔽会实时更新 ISR,需要同步操作。

可以将 MASK 拆解为 2 个不同的寄存器:MASK 和 UNMASK。VCPU 访问 MASK 寄存器屏蔽 I/O 中断,使用 UNMASK 寄存器解除屏蔽。其中 MASK 虚拟机访问的是一个 VCPU 可读写的内存地址,TLB 命中时不需要陷入。而 UNMASK 会按照常规的寄存器模拟,MASK 写 1 代表该位的中断被屏蔽,UNMASK写 1 代表该位的中断被解除屏蔽。当 IRR 发生变化更新 ISR 时的逻辑关系仍为 ISR = IRR&(~ MASK),但在 VCPU 写 UNMASK 寄存器陷入 KVM 后,VIOAPIC 会按照如下步骤:

(1) MASK = MASK&(~ UNMASK)

(2) ISR = IRR&(~ MASK)

完成对 MASK、UNMASK 和 ISR 寄存器的同步操作。需要注意的是,同时只能有一个线程更新 ISR 或访问 MASK 和 UNMASK,因此需要在虚拟机和 KVM 中使用锁进行保护。优化后的数据流图如图 8 所示。

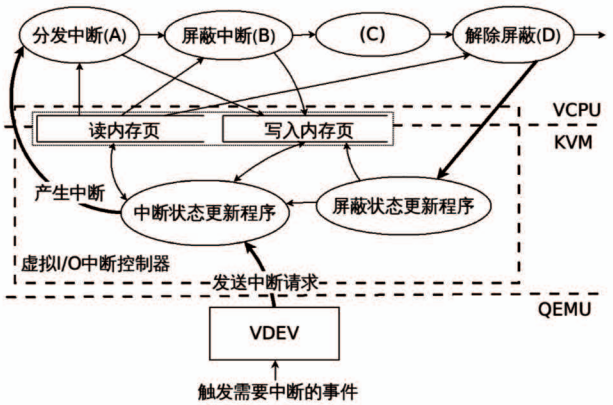

图 8　采用类虚拟化方法优化后的数据流图

和 4.1 节中的只读页表优化一样,VIOAPIC 相关寄存器的状态放在读页中,MASK 寄存器放在新增的写页中。MASK 更新时,先从内存写页中读出旧值,再与新的需要屏蔽的中断位取或后重新写回内存。修改后的响应过程 VCPU 只在解除屏蔽时写 UNMASK 寄存器需要陷入 KVM。经测试优化后一次完整中断所需的时钟周期约为 18 647 个时钟周期,其中访问 VIOAPIC 的寄存器消耗了约 12.1% 的时钟周期,连续处理 100 万个中断耗时从 76.5 s 缩短到 18.4 s,吞吐量提升了超过 3 倍,相较于单纯的 KVM 模拟,也提升了近 1 倍。

MSI 模式的 I/O 中断也可以用这种方法优化,只是优化的寄存器逻辑略有不同,在这里不作赘述。

5　性能测试

为了验证优化后的模拟 I/O 中断控制器,本研究选择了 I/O 中断密集的部分磁盘、网络测试项,测试优化前和优化后的效果,其中只读页表优化和中断处理逻辑优化都是以将中断控制器放置在 KVM 中模拟为基础的。测试环境如表 3 所示。

磁盘测试中比较了龙芯平台优化前后的性能。

表 3　龙芯 KVM 测试环境

	虚拟机	宿主机
CPU	3A3000 双核 1.2 GHz	3A3000 四核 1.2 GHz
内存	4 GB	16 GB
硬盘	Virtio-disk qcow	128 GB SSD(SATA3)
网卡	Virtio-net	7A 板载千兆网卡
操作系统	Loongnix 1.0	Loongnix 1.0
内核	Linux 3.10	Linux 3.10

网络测试不仅比较了龙芯优化前后的性能,还与商用计算机的 KVM 虚拟机进行了简单的效率对比。选择网络测试与商用计算机进行对比是因为网络测试在虚拟机和宿主机使用地址转换模式(NAT)连接时,并不需要通过真实外设交互,全过程只涉及软件,可以屏蔽由于外设性能差距带来的干扰。而测试所用的 netpef TCP _ RR 和 TCP _ CRR 测试项经性能分析工具分析在虚拟机和宿主机使用 NAT 连接时 95% 以上的时钟周期都用于中断逻辑模拟,对比的商用计算机使用 AMD 的 FX8300 处理器,该处理器的单核同频性能与 GS464E 相当[10]。为了测试方便,将核心数和核心频率设置为与 3A3000 相同,测试分值就可以直接反映 I/O 中断模拟的效率。具体测试环境如表 4 所示(以下将对比的商用计算机简称 FX8300)。

表 4　商用计算机 KVM 测试环境

	虚拟机	宿主机
CPU	FX8300 双核 1.2 GHz	FX8300 四核 1.2 GHz
内存	4 GB	16 GB
硬盘	Virtio-disk qcow	500 GB SSD(SATA3)
网卡	Virtio-net	主板板载千兆网卡
操作系统	Fedora 21	CentOS 7.6
内核	Linux 3.17	Linux 3.10

磁盘测试中使用 dd 命令循环将 0 写入文件,源数据使用/dev/zero,文件数据块大小从 8 kB 到 10 MB,测试使用 iflag = direct 参数,循环次数为 10 GB/块大小。测试结果如表 5 所示,结果的单位为MB/s。本节中测试栏中的 KVM 代表在 KVM 中实现 VIOAPIC。

表 5　磁盘性能测试结果

块大小	原始	KVM	只读内存页	类虚拟化
8 kB	5.3	14.4	15.8	16.7
16 kB	9.8	28.1	32.2	34.1
128 kB	45.7	113	125	131
1 MB	212	213	222	227
10 MB	241	245	245	245

在磁盘测试中,数据块较小时的优化效果明显。以 8 kB 数据块为例,相对于原始成绩,通用优化的提升幅度最大,提升了 171.6%,而只读页表优化和类虚拟化优化在通用优化的基础上又分别提升了 9.72% 和 15.97%,相对于原始成绩则提升了 198.1% 和 215.1%。在 16 kB、128 kB 的测试中的结果也与之相似,但在 1 MB、10 MB 的测试中,优化前后的性能没有明显变化。这是因为虚拟机与半虚拟化的磁盘设备主要通过环形缓冲区进行交互,在环形缓冲区未阻塞时不需要使用 I/O 中断同步[11]。数据块较大时,每次拷贝 VCPU 的准备时间长,磁盘写入的时间也较长,交互次数少,环形缓冲区很少出现阻塞。小块数据拷贝时,虚拟机和虚拟外设交互频繁,环形缓冲区经常阻塞。

网络测试中,虚拟机与宿主机使用 NAT 方式连接。使用 Netperf 中的 TCP _ RR、TCP _ CRR 方式测试,server 运行在宿主机上。测试结果如表 6 所示。

表 6　网络性能测试对比

	TCP _ CRR	TCP _ RR
优化前	1003.19	2620.77
KVM	1323.37	3978.53
只读内存页	1523.47	4471.23
类虚拟化	1614.32	4589.71
FX8300	1269.95	3558.72

通过网络测试的结果可知,优化前龙芯的 I/O 中断效率明显低于 FX8300,测试成绩仅为 FX8300 的 73.6% ~80%。在同样使用 KVM 模拟 I/O 中断控制逻辑的情况下,龙芯的模拟效率超过了 FX8300,测试成绩达到了 FX8300 的 104.2% ~111.7%,在使用只读页表和类虚拟化优化后,模拟效率又有了进一步的提高。在使用只读页表时,达到了 FX8300 的 120% ~125.6%。使用类虚拟化优化方法时,达到了 FX8300 的 127.1% ~128.9%。

与优化前相比,龙芯 KVM 在中断较为频繁的测试项目中,性能提升了 60.9% ~215.1%,其中最主要的是在 KVM 中模拟 VIOAPIC 带来的性能提升,占到了其中的 52% ~78%。另外 22% ~48% 的性能提升是只读页表和类虚拟化优化带来的。

6　结　论

I/O 中断模拟的效率较低是 I/O 虚拟化中普遍存在的问题。虽然半虚拟化 I/O 设备通过使用环形缓冲区能够有效地减少虚拟机 I/O 中断的次数,但在部分场景下虚拟机仍会触发较多的虚拟 I/O 中断,影响性能和用户体验。尤其在面对万兆网卡等高速 I/O 外设时,虚拟机将会面临更大的 I/O 中断压力。如果虚拟机使用传统的模拟 I/O 中断,部分场景下会损失 40% 左右的峰值性能[12]。为 I/O 中断虚拟化添加硬件辅助,可以有效地提升虚拟机的 I/O中断性能,相对于纯软件模拟有性能优势[13]。在没有硬件辅助的情况下,通过模拟流程和处理逻辑的优化,也可以较大幅度地提升 I/O 中断的性能。

本研究一方面参照其他架构的优化方法在 KVM 中实现 I/O 中断子系统的模拟,并验证了其有效性;另一方面探索让虚拟机直接使用内存读取 I/O寄存器数据,优化虚拟机中断处理逻辑,进一步提升了虚拟机处理 I/O 中断的性能。实测虚拟机的 I/O 中断吞吐率有了较大的提升,一次 I/O 中断的处理事件缩短了 80%。对于部分 I/O 中断较频繁的场景,龙芯 KVM 虚拟机的性能有了成倍的提高。优化后的I/O中断吞吐量能够满足龙芯虚拟机使用半虚拟化设备时的需求。其中将 I/O 寄存器映射到内存只读页、优化虚拟中断控制器逻辑的优化方法也可以用到其他架构的虚拟机上。

参考文献

[1] Campbell S, Jeronimo M. An introduction to virtualization [R]. Santa Clara: Intel, 2006

[2] Smith J E, Nair R. The architecture of virtual machines

[J]. *IEEE Computer*, 2005,38(5): 32-38

[3] 台运方. MIPS 架构混合虚拟化系统的设计实现与性能优化[D]. 北京:中国科学院计算技术研究所, 2014: 41-43

[4] 蔡万伟. 基于 MIPS 架构的系统虚拟机设计实现与性能优化[D]. 北京:中国科学院大学, 2012:38-39

[5] 吴瑞阳, 汪文祥, 王焕东, 等. 龙芯 GS464E 处理器核架构设计[J]. 中国科学:信息科学,2015,4:480-500

[6] Bugnion E, Nieh J, Tsafrir D. Hardware and software support for virtualization[J]. *Synthesis Lectures on Computer Architecture*, 2017, 12(1):1-206

[7] 英特尔开源软件技术中心,复旦大学并行处理研究所. 系统虚拟化:原理与实现[M]. 北京:清华大学出版社, 2009: 55

[8] 王俊儒. 基于龙芯 3A3000 的虚拟机访存优化技术研究[D]. 北京:中国科学院大学, 2019:15-16

[9] Sweetman D 著. MIPS 体系结构透视(第 2 版)[M]. 北京:机械工业出版社, 2007: 39-40

[10] 胡伟武. 自主 CPU 发展道路及在航天领域应用[J]. 上海航天, 2019, 36(1):5-13

[11] Russell R. Virtio: towards a De-Facto standard for virtual I/O devices[J]. *ACM SIGOPS Operating Systems Review——Research and Developments in the Linux Kernel Archive*, 2008,42(5):95-103

[12] Amit N, Gordon A, Har'El N , et al. Bare-metal performance for virtual machines with exitless interrupts[J]. *Communications of the ACM*, 2015, 59(1):108-116

[13] Dall C, Nieh J. KVM/ARM: the design and implementation of the linux ARM hypervisor[C]//International Conference on Architectural Support for Programming Languages & Operating Systems, Salt Lake City, USA, 2014: 333-347

Optimize the I/O interrupt subsystem of the Loongson KVM virtual machine

Zhu Chen[*,**,***,****], Wang Jian[*,**,***,****], Gao Xiang[****], Mao Bibo[****], Li Xing[****]

([*] State Key Laboratory of Computer Architecture, Institute of Computer Technology, Chinese Academy of Sciences, Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

([****] Loongson Technology Corporation Limited, Beijing 100190)

Abstract

This article focuses on the performance issues of the virtual machine I/O interrupt subsystem. Using the Loongson KVM virtual machine as the experimental platform, the performance bottleneck of the I/O interrupt subsystem in the KVM virtual machine is analyzed, and the conventional optimization method is verified. Based on this, two new optimization methods are tried and verified. Test results show that the virtual machine's processing I/O interrupt throughput has increased by more than 300% compared to the original system. In some network and disk tests that frequently require I/O interrupts, its performance has increased by 60.9% to 215.1%. The optimization methods in this article can also be used in other KVM virtual machines.

Key words: KVM virtual machine, I/O interrupt, I/O virtualization