高技术通讯 2024 年 第34 卷 第8 期:842-853

doi:10.3772/j.issn.1002-0470.2024.08.006

面向设备直通的高效低延时的中断直通方法①

吕 晨②[*,**,***] 张福新③[*,**,***] 朱 琛[****] 毛碧波[****] 邓平科[*****] 潘筱涵[*****]

([*]处理器芯片全国重点实验室(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)

([****]龙芯中科技术股份有限公司 北京 100190)

([*****]中国移动通信有限公司研究院 北京 100032)

摘 要 针对对称多处理器(SMP)虚拟机(VM)的虚拟中央处理器(vCPU)调度延迟会降低虚拟机输入/输出(I/O)响应性的问题,本文基于设备直通提出了一种高效低延迟的中断直通方法。该方法基于硬件辅助技术,搭建了中断直通架构,并设计了中断重定向机制,将直通设备中断从被抢占的 vCPU 重定向至正在运行的 vCPU。实验结果表明,网络往返时延平均减少了 34.1%,吞吐量最高提升 7.9%,Apache 测试每个服务器请求所需时间平均减少了 13.6%,磁盘 I/O 操作时延平均减少了 6.7% ~8.4%。实验结果证明,该方法能有效减少虚拟机虚拟 CPU 调度对 I/O 延迟的影响,提高虚拟机 I/O 响应性。

关键词 中断重映射;输入/输出(I/O)虚拟化;设备直通;基于内核的虚拟机(KVM);I/O 响应性

随着云计算规模的不断扩大和互联网建设的高速发展,云平台需要提供更高吞吐量和更低延迟的网络连接,对虚拟化技术的要求也越来越高。输入/输出(input/output,I/O)虚拟化已然成为虚拟化基础架构中最重要的一个组成部分,其效率对整个系统的性能有着至关重要的影响。近年来工业界和学术界都致力于减少 I/O 虚拟化的开销[1-2]。I/O 虚拟化的开销主要分为直接内存访问(direct memory access,DMA)相关和中断相关 2 部分[1]。设备直通技术允许虚拟机和外部 I/O 设备直接通信,消除了 DMA 相关的开销。因此,中断成为 I/O 虚拟化剩余的主要性能瓶颈[3-4]。

减少中断注入开销最有力的方式就是允许虚拟机中断直接注入虚拟机,而不需要虚拟机管理器(virtual machine monitor,VMM)的干预。当前主流的硬件辅助中断重映射机制提供直接注入虚拟机中断的解决方案。但是,目前云计算平台大多采用对称多处理器(symmetric multi-processing,SMP)虚拟机,虚拟机的中断注入可能受到虚拟中央处理器(virtual central processing unit,vCPU)调度的影响,导致额外的中断延迟,降低 I/O 响应性。在高负载情况下,该现象会更加明显。这种额外的中断延迟还会降低应用的性能和系统的实时性。

为了解决上述问题,本文在虚拟机设备直通时设计了一种高效低延迟的中断直通方法。该方法在注入中断的目的 vCPU 被调度出去时,有选择性地将直通设备中断重定向至正在运行的 vCPU,或将中断请求暂存在内存中并及时唤醒目的 vCPU。

① 中国科学院计算所-中国移动研究院联合创新平台(E351500000)资助项目。

② 女,1995 年生,博士生;研究方向:计算机系统结构,系统虚拟化;E-mail:lvchen20b@ict.ac.cn。

③ 通信作者,E-mail:fxzhang@ict.ac.cn。

(收稿日期:2023-08-30)

1　研究背景

在虚拟化环境中,VMM 频繁干预引入的大量虚拟机退出(virtual machine exit,VM-Exit)是导致虚拟化性能大打折扣的主要原因[5-6]。在 I/O 虚拟化中,导致 VM-Exit 的原因主要有 3 类[1]:一是虚拟机中的设备驱动程序发起 I/O 请求;二是 I/O 设备产生中断,需要注入虚拟机;三是中断处理程序执行完毕需要通知硬件中断控制器。硬件辅助的设备直通技术允许虚拟机直接访问其直通设备的寄存器,故不会触发第一类 VM-Exit,但是中断相关的 VM-Exit 仍然存在。

在虚拟化场景下,没有硬件辅助时直通设备的中断是无法直接注入虚拟机的,因为虚拟机的中断控制器是软件模拟的,虚拟机无法直接访问中断控制器的寄存器。因此,虚拟机对虚拟中断控制器寄存器的读写操作都将陷入 VMM 处理,采用软件注入[7]的方式。直通设备发起的物理中断由 VMM 转换为虚拟中断[8]再注入虚拟机。现代多处理器平台中高性能 I/O 设备(如万兆网卡)一般采用消息中断(message signaled interrupt,MSI)[9]。为了防止虚拟机通过故意发起恶意的 DMA 写,伪装成直通设备发起的 MSI 中断,从而对宿主机或者其他虚拟机进行攻击,输入输出内存管理单元(input/output memory management unit,IOMMU)引入了中断重映射机制[10]。IOMMU 通过中断重映射表对直通设备发起的中断进行合法性校验,消除了此类隐患,但中断注入仍需 VMM 的干预。

为了减少中断虚拟化过程中访问中断控制器寄存器导致的 VM-Exit,硬件厂商提供在硬件层面模拟中断控制器的方法,例如 Intel APICv[11]、AMD AVIC[12]和 ARM VGIC[13]。Intel APICv 提供硬件模拟的 APIC 寄存器和常规的 APIC 逻辑。模拟的寄存器(如 vISR、vIRR、vEOI 等)存放在虚拟高级可编程中断控制器(advanced programmable interrupt controller,APIC)页中,可供虚拟机直接访问。随后 Intel 在 APICv 的基础上引入 PI(Posted Interrupt),支持虚拟中断直接注入。

1.1　Posted Interrupt

Posted Interrupt 分为 CPU Posted Interrupt 和 VT-d Posted Interrupt(又称 IOMMU Posted Interrupt)[8]。前者对应于处理器间中断(inter-processor interrupt,IPI),后者针对直通设备中断,本文主要关注直通设备场景下的中断注入,故以 VT-d Posted Interrupt 为例简单介绍其工作原理。

VMM 为每个直通设备的中断源分配一个中断重映射表项(interrupt remapping table entry,IRTE),中断重映射表的基址存放在 IOMMU 寄存器中。每个 vCPU 对应一个 PI 描述符,用于记录中断请求、通知向量(notification vector,NV)和通知地址(notification destination,NDST),即目的 vCPU 所在的物理 CPU 核。其架构和处理流程如图 1 所示,包括:(1)设备产生一个中断;(2)IOMMU 根据 MSI 寄存器中的值索引中断重映射表项(interrupt remapping table entry,IRTE),得到中断向量和对应目的 vCPU 的 PI 描述符;(3)IOMMU 将中断信息记录在 PI 描述符中,并向 NDST 发送 NV;(4)目的物理 CPU 核(core1)收到通知后,将 PI 描述符中的中断信息同步到虚拟中断请求寄存器(virtual interrupt request register,vAPIC)页的 vIRR 中;(5)当虚拟机执行完毕中断处理程序时,写中断结束(end of interrupt,EOI)操作会直接更新 vAPIC 页中的虚拟寄存器,而不会引起 VM-Exit。

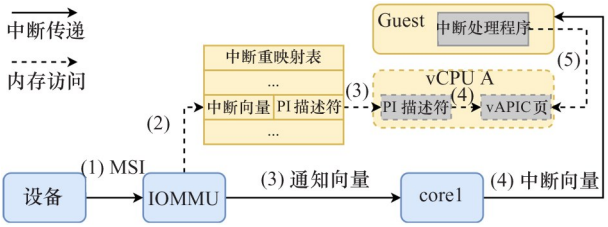


图 1　VT-d Posted Interrupt

1.2　研究问题

就中断而言,中断响应时间越短,系统的实时性越好,尤其是对延迟敏感的应用来说。Posted interrupt 支持中断直接注入,但也需要根据目的 vCPU 所处的状态分为 2 种情况:(1)目的 vCPU 正在运行,Posted Interrupt 允许中断直接注入;(2)目的 vCPU 未在运行,则该中断需要等下次 vCPU 运行时

才能注入虚拟机。由于虚拟化特性,一般对称多处理器(symetric multi-processing,SMP)虚拟机的 vCPU 复用物理 CPU 核。那么当一个中断在注入时,目标 vCPU 可能已经被调度出去,即当前未在运行。那么该中断只有在该 vCPU 被再次调度运行时才能处理。

图 2 展示了调度延迟对网络时延的影响。测试虚拟机的 4 个 vCPU 绑定在同一个物理核上以模拟 CPU 竞争激烈的压力场景,同时在测试虚拟机中运行一个 CPU 负载程序以维持虚拟机的 CPU 负载在一个可控的范围。将虚拟机的所有 vCPU 占虚拟机可用的所有物理核资源的百分比称为虚拟机负载。从图中可看出,网络时延随虚拟机负载的增加而增加,且上升速率也在增大。这是因为随着虚拟机负载的增大,vCPU 留给 I/O 处理的时间变少,vCPU 在调度上花费的时间增加,调度延迟会被引入中断注入延迟,降低 I/O 响应性[14-16]。

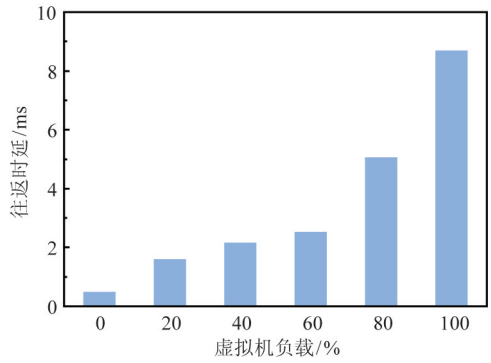


图 2　调度延迟对网络往返时延的影响

为了验证 vCPU 调度延迟和虚拟机负载的关系,本文进一步测试了 10 s 内不同超分比下 vCPU 调度延迟随虚拟机负载的变化情况,结果如图3所示。从图中可以看出,在相同虚拟机负载下,超分比越大,vCPU 调度延迟越高。同一超分比下,vCPU 调度延迟随虚拟机负载的增大而增加,且呈非线性关系。进一步说明虚拟机负载增加会导致 vCPU 调度延迟增加,从而引入额外的中断延迟,降低 I/O 响应性。

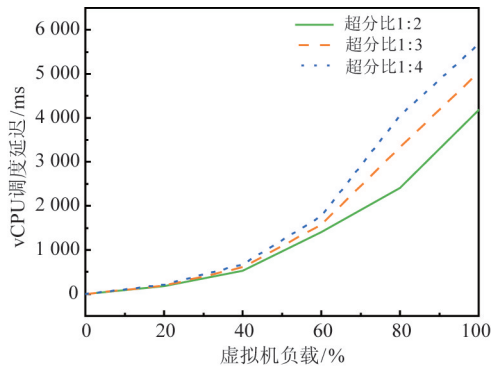


图 3　vCPU 调度延迟和虚拟机负载的关系

1.3　相关工作

许多研究工作为减少调度对中断注入延迟的影响提出了一系列的解决方案。这些解决方案主要分为软件方案和硬件辅助方案。软件方案又可以分为 guest OS 级方案和 VMM 级解决方案。

Guest OS 级方案:中断迁移。例如 vBalance[16]提出将 vCPU 的调度状态传给 guest OS,在 vCPU 被调度出去或中断不均衡时,将中断迁移到正在物理 CPU 核上运行的 vCPU 上,实现均衡的中断负载。但是该方案需要修改 guest OS,移植受限,而且每次重映射 guest OS 需要通过超级调用陷入 VMM 同步信息,引入额外的 CPU 开销。

VMM 级解决方案:(1)采用更小的 CPU 时间片。例如 vSlicer[14]将延迟敏感虚拟机的 Xen 信用调度器的时间片从 30 ms 减少至 5 ms。vTurbo[15]将虚拟机的 I/O 处理绑定到指定的一类 vturbo 核上,所有 vturbo 核在指定的物理核上以仅 0.1 ms 的时间片调度运行。虽然更小的时间片可以缩短调度延迟,但是频繁的调度会导致额外的上下文切换开销。(2)基于模拟中断控制器的中断重定向。例如 vINT[17]在虚拟机的虚拟 I/O APIC 接收到一个虚拟中断需要映射至目的 vCPU 时,选择中断负载最小的在线 vCPU 或预测最先运行的 vCPU 作为目的 vCPU。hBalance[18]优化 vBalance,将中断均衡功能从 guest OS 移动到 VMM。但该类方案涉及的虚拟中断控制器仍是 VMM 维护的软件实体。(3)将部分物理 APIC 寄存器暴露给虚拟机。例如 ELI[19-20]和 DID[1]清除了虚拟机控制结构(virtual machine control structure,VMCS)中的外部中断退出(external interrupt exiting,EIE)控制位,允许所有传递给虚拟机的中断不触发 VM-Exit。但虚拟机和宿主机共享相同的物理本地高级可编程中断控制器(local advanced programmable interrupt controller,LAPIC)寄存

器,容易造成安全隐患,也易损失部分虚拟化特性,如物理 CPU 核复用。

其他软件方案诸如侧核策略等。例如,ELVIS[21]使用专用 I/O 核轮询虚拟机的 I/O 请求,vTurbo[15]将虚拟机的 I/O 处理绑定到指定的 vturbo 核上。Directvisor[22]将每个 vCPU 绑定至一个物理 CPU,禁止 guest 对 MSI 读写导致的 VM-Exit。专用核的使用缺乏普遍适用性,容易造成专用核资源利用不足,对虚拟机迁移也有一定影响。

硬件辅助方案主要基于硬件厂商提供的 APIC 虚拟化支持。例如 Intel APICv、AMD AVIC、ARM VGIC 和 IBM XIVE。Intel 的 Posted Interrupt 描述符中紧急位,用于表示该中断是否需要目标 CPU 的立即响应。ES2[23-24]采用 Intel APICv 和 PI 技术(准确来说是 CPU Posted Interrupt)进行中断重定向优化,选择合适的中断目的 vCPU,同时提出了结合通知和轮询的混合 I/O 处理机制减少虚拟机 I/O 请求导致的 VM-Exit。不过 ES2 系统主要是针对半虚拟化 I/O模型设计的,对直通设备的适用性有待验证。

本文针对设备直通基于 MSI 中断模型,构建了硬件辅助中断直通架构,根据 vCPU 调度信息有选择地将中断重定向至正在运行的 vCPU 或仍将中断请求暂存在内存中并及时唤醒目的 vCPU,减少调度延迟对 I/O 处理的影响。相比 guest OS 级方案,本文方法不需要修改 guest OS,可移植性更高。相比 VMM 级解决方案,本文方法不需要修改调度器时间片,也避免了模拟中断控制器的开销和共享物理寄存器的安全隐患。本文方法基于硬件辅助方案的思想,但与 ES2[23-24]系统不同的是,本文主要针对设备直通场景,充分利用设备直通技术减少 VM-Exit 的优势,集中关注于中断直通的优化研究。

2　设　计

2.1　中断直通架构

本文基于硬件辅助技术搭建了中断直通框架,并设计了中断重定向模块,整体架构如图 4 所示。中断重映射模块负责映射直通设备中断、维护中断重映射表。硬件中断重映射单元(interrupt remapping engine,IRE)负责在收到直通设备中断时,将中

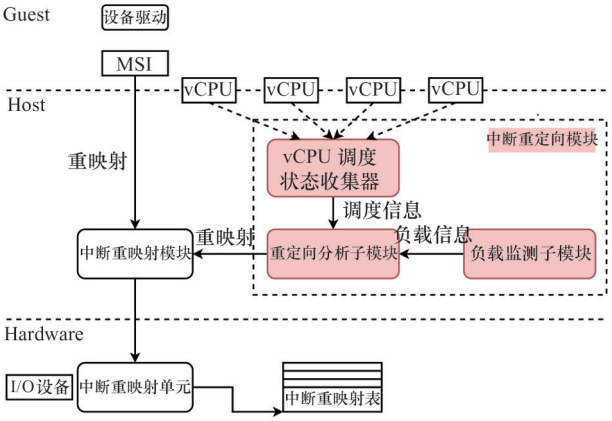


图 4　中断直通架构

断直接注入虚拟机或暂存于内存中。中断重定向模块负责在 vCPU 调度延迟高时将直通设备中断从被抢占的 vCPU 重定向到正在运行的 vCPU 上。因此,如果已知中断源指定的目的 vCPU 未在运行,则中断来临后会直通给重定向后的 vCPU。如果没有做重定向操作,则中断仍被记录在原来目的 vCPU 指定的内存中。

中断重映射模块在宿主机中。Guest OS 识别出直通设备之后,为直通设备分配 MSI 信息,包括目的 vCPU 和中断向量号。Guest OS 配置直通设备 MSI 相关信息时,会陷入 VMM 进行中断重映射。中断重映射模块将 MSI 信息中的源 vCPU 号映射为当前 vCPU 正在运行的物理 CPU 核号,将中断号映射为 guest 为 MSI 分配的向量号。最后映射项被写入中断重映射表,存放在内存中。

硬件中断重映射单元及相关结构示意图如图 5 所示。中断重映射的信息保存在中断重映射表(interrupt remapping table,IRT)中,该表存储在内存中,其基地址存储在 IRE 的 IRT Base 寄存器中。为了加速重映射过程,IRE 中设计有 IRT Cache 来缓存重映射表。IRE 处理中断注入的流程也如图 5 所示。IRE 收到直通设备中断后,以 MSI 信息作为索引找到中断重映射表项中存放的目的 vCPU 正在运行的物理 CPU 核号和 guest 的中断向量号。IRE 根据这些信息向目的 vCPU 发送特殊中断。如果目的 vCPU 正在运行,则中断直接写入硬件模拟的中断控制器寄存器,成功注入虚拟机。如果目的 vCPU 未在运行,则中断注入失败,IRE 将中断写入内存中该

vCPU 对应的待注入中断描述符中(pending interrupt descriptor,PID),等下次目的 vCPU 运行时,由 VMM 读取该中断再注入虚拟机中。

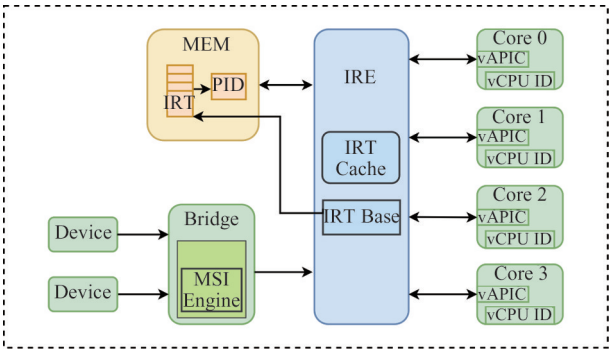


图 5　直通中断注入流程及相关结构示意图

2.2　中断重定向模块

中断重定向模块负责在 vCPU 调度延迟高时将直通设备中断从被抢占的 vCPU 重定向到正在运行的 vCPU 上,主要涉及以下几个组件。

(1)vCPU 调度状态收集器,负责跟踪 vCPU 调度状态。本文将正在物理 CPU 核上运行的 vCPU 记为“在线”vCPU,将在调度队列中处于等待状态的 vCPU 记为“离线”vCPU,即未在物理核上运行的 vCPU。在调度器中添加一个 vCPU 状态收集器,跟踪各个 vCPU 的调度状态,分别维护一个 vCPU 的在线列表和离线列表。但是对调度器本身没有任何修改。

(2)负载监测子模块,负责判断 vCPU 调度延时是否会影响中断延迟。指定阈值 T, 如果连续 t 时间,单位时间内负载均超过阈值 T, 则认为调度延迟高对中断延迟会有影响。该阈值需要根据经验值设置。

(3)重定向分析子模块,负责判断是否需要中断重定向以及选择新的目的 vCPU。根据负载监测模块传递的信息决策是否进行重定向。如果需要进行重定向,则根据 vCPU 调度信息选择合适的 vCPU 作为新的目的 vCPU,重定向的伪代码如算法 1 所示。

算法 1　中断重定向

```
1  //online_list, the list of all online vCPUs;
2  //offline_list, the list of all offline vCPUs;
3  //new_vcpu, the new dest vcpu for redirection
4  if a vcpu is being scheduled out then
5      /* get a new dest vcpu */
6      search online_list,
7      find the first online vcpu as new_vcpu;
8      if new_vcpu != NULL then
9          for each msi irq of assigned device with affinity
           to this vcpu do
10             send information of irq and new_vcpu to
               interrupt remapping module;
11             vcpu.redirectd = 1;
12         end
13     end
14 end
15
16 if a vcpu is being scheduled in then
17     if vcpu.redirectd == 1 then
18         for each msi irq of assigned device with affini-
           ty to this vcpudo
19             send information of irq and vcpu to inter-
               rupt remapping module;
20             vcpu.redirectd == -1;
21         end
22     end
23 end
```

3　实　现

本文基于内核的虚拟机[25](kernel-based virtual machine,KVM)实现该设计。KVM 是基于 Linux 内核的主流 VMM,作为一个开源项目,KVM 在学术界和工业界都有着广泛的研究和应用。KVM 通常和快速模拟器(Quick Emulator,QEMU)组合使用,因此 KVM 虚拟机也被称为 QEMU-KVM 虚拟机。

3.1　vCPU 调度状态收集器

在 KVM 中每个虚拟机对应宿主机上的一个 QEMU 进程,每个 vCPU 对应于 QEMU 进程中的一个线程。vCPU 调度器并不直接调度 vCPU,而是交由宿主机调度器统一调度运行。Linux 默认的调度器是完全公平调度器(completely fair scheduler,CFS)。由于宿主机调度器无法感知 vCPU 语义,所以它将 vCPU 线程同其他普通线程一样处理。调度器会为每个线程注册抢占通知回调函数,其中为每

个 vCPU 线程注册的回调函数是 kvm _ sched _ in 和 kvm _ sched _ out,分别在 vCPU 被调度运行和被抢占时执行。本文在这 2 个回调函数中跟踪 vCPU 调度状态,具体表现为:当一个 vCPU 被调度运行时,将其加入 vCPU 在线列表,并从离线列表中移除;当一个 vCPU 被调度出去时,将其从在线列表中移除,加入 vCPU 离线列表。

3.2　负载监测

本文设计时认为以上方案在 VM 高负载情况下有优化效果,VM 低负载时优化效果应该不明显,且可能会引入不必要的开销,所以增加了一个负载监测的考量,希望通过实验获得中断重定向对不同超分比和不同虚拟机负载情况下的优化效果,从而选择最佳的负载监测阈值。因此,在实现阶段并未设置具体的阈值。但是在本文实验测试中发现,中断重定向对低负载 VM 的 I/O 响应性也有优化效果,且对性能没有反向作用。因此,本文暂不设置阈值,如果后续研究有新的发现,可重新讨论阈值的设置或采用其他更优的方法。

3.3　中断重定向

本文仅对直通设备中断进行中断重定向。实际上目前需要考虑的中断源只有 2 个:网络中断和磁盘中断。设备在直通过程中,基于 irqfd 机制通过生产者消费者模型来建立中断重映射表项。所以只要遍历该生产者消费者结构就能找到所有的直通设备中断。直通设备采用 MSI 中断,在 guest OS 中分配时将 MSI 信息写入直通设备的配置寄存器中,对应的中断重映射表项需要由 VMM 在使能设备中断直通前更新完毕。所以在重定向时,并不直接修改 MSI 中断源的配置,仅在 vCPU 被抢占时找到该 vCPU 上分配的直通设备中断,将该中断重定向至新的目的 vCPU,最后更新对应的中断重映射表项。由于中断源的配置并没有改变,所以在源目的 vCPU 重新调度运行时,还可以恢复原先的中断重映射关系,以防中断负载不均衡。

4　实验结果和分析

本文基于 LoongArch[26] 架构实现上述方案。LoongArch 架构目前已获得 Linux、GDB、. NET、GCC、LLVM 等国际主流开源社区以及 UEFI(UEFI 规范、ACPI 规范)的广泛认可和支持,具有较好的自主性、先进性与兼容性[27]。

实验主要使用 2 台物理机,均采用 LoongArch 指令架构,其中 1 台作为宿主机,为虚拟机提供运行环境,包括中断直通硬件支持,配备 2.5 GHz 龙芯 3A6000 处理器芯片、8 核、32 GB 内存,1 块 Intel 82599 10 GB 网卡,1 块 Dahua C900 256 GB NVMe 固态硬盘;另外 1 台物理机作为对端服务器,参与网络的收发包工作,配备 2.3 GHz 龙芯 3A5000 处理器芯片、4 核、8 GB 内存,1 块 Intel 82599 10 GB 网卡。两台物理机通过 10 GB 网卡直连,未连入真实的网络环境中,以排除网络环境对实验的影响。宿主机的网卡通过设备直通技术直通给虚拟机使用,默认虚拟机运行 8 个 vCPU。宿主机和虚拟机均安装 Loongnix[28] 20.4 操作系统,内核采用 Linux 4.19。

实验采用了以下 4 种测试程序。(1) ping 命令,测试网络往返时延。ping 是一个轻量级的网络命令,往返时延是网络响应性测试的重要性能指标。测试时从连接的对端物理机向虚拟机连续发送间隔为 1 s 的 ping 指令,每次测试持续 60 s。(2) Netperf[29] 测试网络吞吐量,以此来评估网络性能。测试采用默认设置,每次测试持续 60 s。(3) Apachebench[30] 测试网络服务性能。测试请求数量为 1×10^4,并发用户数为 8。(4) Fio[31] 测试磁盘读写时延和每秒 I/O 操作次数(input /output operations per second,IOPS)。测试采用 4k 随机读模式,I/O 深度为 1,运行时间为 60 s。以上所有测试均执行 10 次,取平均值。

为了模拟较为真实的压力场景,将虚拟机的所有 vCPU 绑定在 2 个物理核上,并通过改变虚拟机 vCPU 的个数来改变 CPU 超分比,即平均一个物理核上运行几个 vCPU。为了在虚拟机管理器中频繁触发 vCPU 调度,在虚拟机中采用 lookbusy[32] 工具模拟系统负载,使所有 vCPU 的负载维持在一个稳定可调节的范围。将所有 vCPU 占虚拟机所有可用物理核资源的百分比称为虚拟机负载,即如果虚拟机共 8 个 vCPU,每个 vCPU 负载 5%,则虚拟机负载

为 8 ×5% ÷2 = 20%。

4.1　网络时延

图 6 显示了优化前后网络的往返时延随虚拟机负载变化的情况。从图中可以看出，首先，随着虚拟机负载的不断增大，优化前后的网络往返时延都呈增长趋势，证明 vCPU 调度延迟会被引入 I/O 延迟。其次，在不同超分比和不同虚拟机负载下，优化后的网络时延相比优化前都有一定程度的降低。在虚拟机有负载时，优化后的网络时延比优化前平均减少了 34.1%。虚拟机空载时，网络时延数量级很小且优化前后差距不大，这是因为由图 3 可知空载时 vCPU 调度延迟接近于 0，所以优化空间有限。另外，随着虚拟机负载的增大，优化效果大致逐渐明显。以图 6(a) 为例，优化后虚拟机负载 20.0% 时网络时延减少了 5.9%，虚拟机满载时网络时延减少了48.6%。相比高负载时，虚拟机低负载时优化效果较小是因为低负载时 vCPU 调度延迟小，对中断注入延迟的影响就小，通过中断重定向优化减少延迟的效果就没有负载高时明显。网络时延实验结果证明，中断重定向能有效减少虚拟机设备直通中断注入延迟，进而减少 I/O 处理延迟，提升 I/O 响应性。

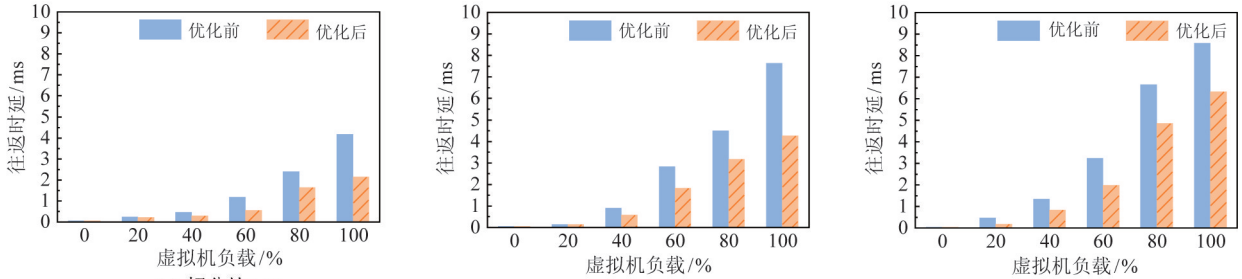


图 6　网络往返时延

图 7 显示了虚拟机在满载时网络往返时延的概率累计曲线。从图中可以看出，相比于优化前，优化后的曲线都更高，表示在相同时延内的占比更多。以图 6(b) 为例，优化后 0.1 ms 以内的往返时延从 39.0% 提升至 55.6%，1 ms 以内的往返时延从 40.5% 提升至 63.3%，10 ms 以内的往返时延从 67.0% 提升至 83.2%。图 6(c) 的优化效果相对前 2 种情况略有削弱，是因为此时超分比高，导致 CPU 时间片在调度上的消耗增加，留给虚拟机的算力减少。当前业界云平台推荐的最高 CPU 超分比为 1∶3[33-34]。超分比如果超过这个比例，虚拟机负载高时，宿主机 CPU 利用率会过高，虚拟机性能可能会下降，稳定性也会受影响。总的来说，实验结果证明了中断重定向作为优化，对 I/O 响应性有很好的提升作用。

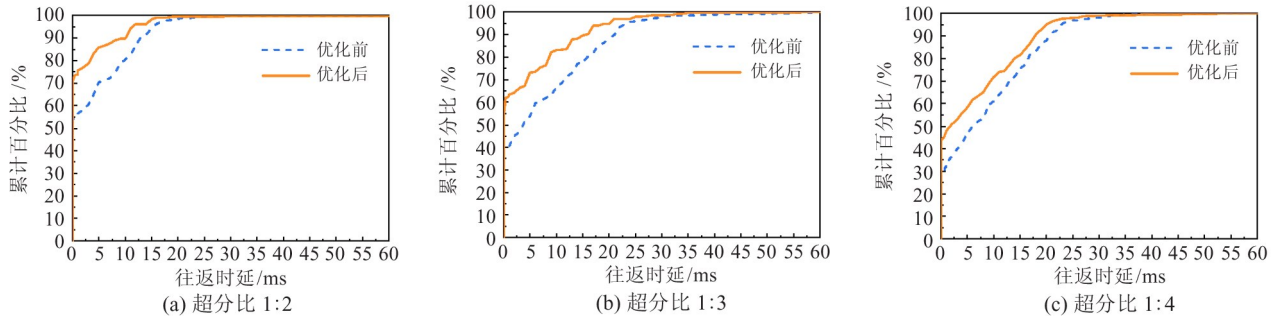


图 7　网络往返时延概率分布

4.2　网络吞吐量

图 8 展示了优化前后网络吞吐量随虚拟机负载变化的情况，对应同时期 10 s 内直通设备中断数如图 9 所示。从图上数据分析来看，中断重定向对性

能的优化作用没有对 I/O 响应性的优化作用来得明显,因为此时限制性能提升的主要因素是 CPU 资源。虚拟机的 vCPU 花费大多数的时间在计算型负载上面,I/O 所需的 CPU 资源紧张。但在虚拟机负载 100.0% 左右时,吞吐量提升了 7.9%,对应中断数也增加了 8.7%,可能是因为此时 CPU 资源和 I/O 延迟都是吞吐量的影响因素,无论优化哪个,对性能都有提升作用。总的来说,中断重定向不会导致吞吐量降低,吞吐量甚至略有提升,其对性能有正向作用。

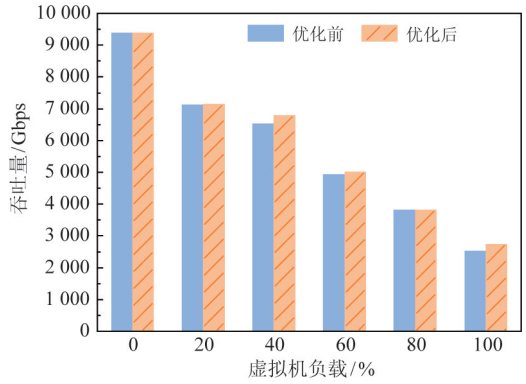


图 8　netperf 测试

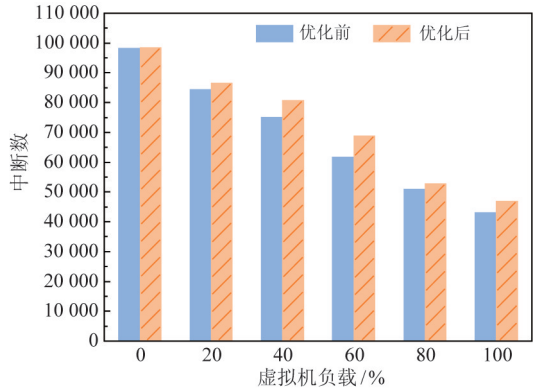


图 9　netperf 测试时中断数变化

4.3　Apache 测试

ApacheBench 是 Apache[35] 服务器自带的一个 Web 压力测试工具,它会给目标服务器带来巨大的负载,所以此时不需要模拟虚拟机负载。本文在虚拟机中部署 Apache 服务器,将对端物理机作为客户端。客户端运行 ApacheBench 工具,向 Apache 服务器发送网络请求。每个服务器请求平均所需时间如图 10 所示。优化后每个服务器请求所需的时间平均减少了 13.6%。

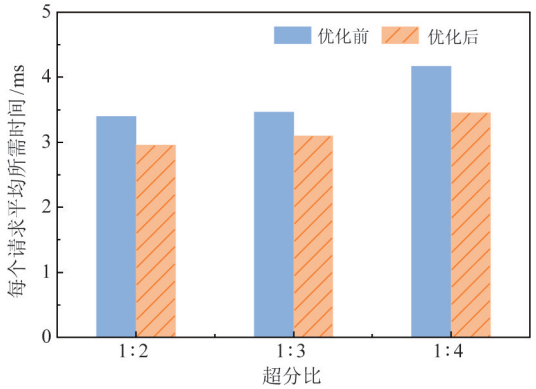


图 10　Apache 测试

4.4　磁盘测试

磁盘测试的结果如图 11 所示。从图中可知,磁盘 I/O 操作平均时延随虚拟机负载增加而增加,且在满载时骤增。优化后的平均时延比优化前均有所减少,超分比1∶2、1∶3和1∶4时分别平均减少了

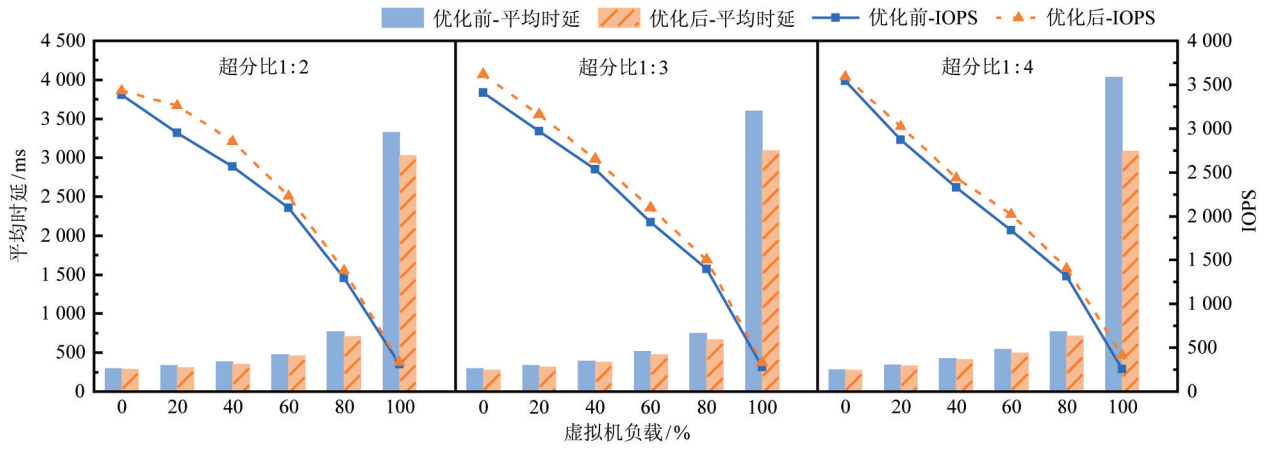
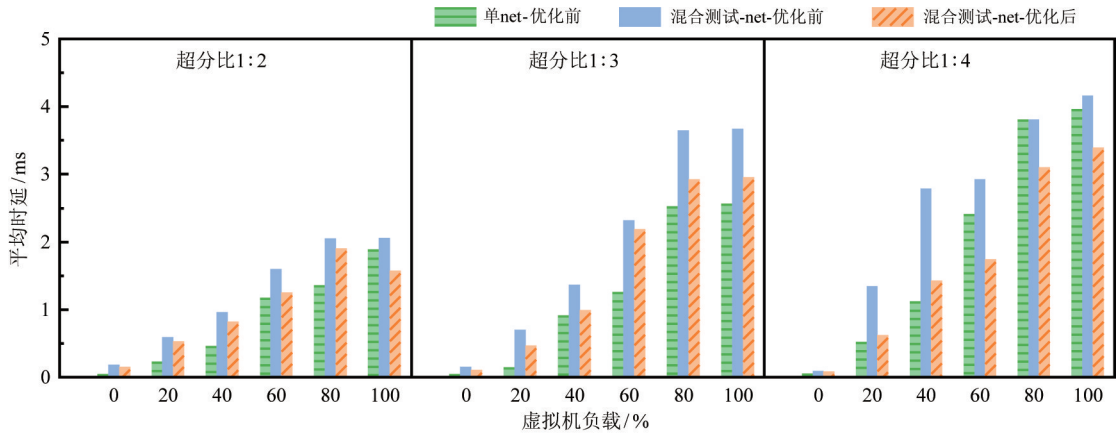


图 11　fio 测试

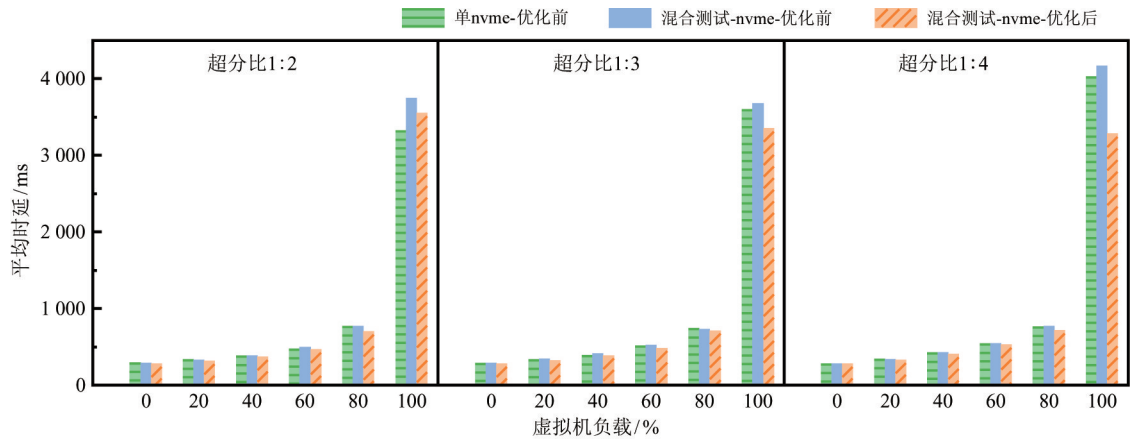
6.7%、8.4% 和 8.4%。由于此次 fio 测试参数设置的重点是针对读写时延，所得的 IOPS 并不能表示该磁盘最优的性能，但优化前后的 IOPS 对比也能体现出本文方法对性能的优化作用。超分比 1∶2、1∶3 和 1∶4 时优化后的 IOPS 比优化前分别平均提高了 7.3%、9.3% 和 15.2%。实验结果证明，中断重定向能有效减少磁盘 I/O 操作的时延，并提高吞吐量。

4.5　混合测试

本实验测试了网卡和磁盘同时直通且工作时的时延。测试时在虚拟机中同时执行 fio 命令和 ping 命令，参数和上文保持一致。将同条件下单设备工作时的结果也纳入对比范围，结果如图 12 所示。首先，混合测试和单独测试相比，同超分比和虚拟机负载下，网络平均时延比单独测试时的时延要高，磁盘时延仅在满载情况下增加较为明显。这是因为混合测试时在虚拟机中执行 fio 命令也占用了部分额外的虚拟机负载，而网络时延对虚拟机负载的敏感度比磁盘时延要高。其次，混合测试时优化前后对比，网卡和磁盘优化后的平均时延都比优化前的平均时延要低，超分比 1∶2、1∶3 和 1∶4 时网络时延分别平均减少了 16.0%、22.5% 和 32.5%，磁盘时延分别平均减少了 5.1%、5.7% 和 6.8%。实验结果证明中断重定向在多设备同时工作时也能发挥优化作用，减少直通中断注入延迟，进而减少 I/O 处理延迟，提升虚拟机整体 I/O 响应性。



（a）网卡测试结果



（b）磁盘测试结果

图 12　混合测试

5　讨　论

本文设计的开销主要来自于中断重映射表项的重新建立，如果硬件支持中断重映射缓存，还需要刷新对应缓存项。本文选择在 vCPU 切换时进行中断重定向，而不是中断来临时再执行重定向操作，一部

分也是出于开销问题考虑。因为在 vCPU 调度时,相比调度产生的上下文切换开销,重定向的开销很小,即使需要刷新中断重映射缓存,硬件执行产生的开销也明显小于 vCPU 调度软件开销。如果在中断来临的时候再做重定向的话,一方面重映射硬件单元在检查到中断无法直接注入后就已经将中断记录在对应目的 vCPU 的内存中了,再做重定向没有意义。另一方面由于需要 guest 退出至 VMM 以创建中断重映射表,又多了一次 VM-Exit,反而增加开销。因此,本实验中断重定向的开销很小,对性能几乎没有影响。

当然,对于一些对亲和性有强烈要求的应用来说,该方案可能不太适合。但是,本文主要针对直通设备中断,主要是网络中断和磁盘中断,目前绝大多网络或磁盘设备中断默认并不会强制绑定某个核。如果有强制绑核需要的应用,可以选择不使能中断重定向功能,即在目的 vCPU 未在运行时,仍将中断请求暂存于内存中。

本文方案并不仅局限于 LoongArch 架构,也适用于其他架构。以 X86 体系结构为例,首先,关于 vCPU 调度部分是和处理器架构无关的。其次,关于中断重定向部分,X86 支持 VT-d posted interrupt,利用 IRT 实现中断直通,同样可以根据 vCPU 调度状态通过修改 IRTE 并刷新对应缓存来实现本文方案。

6　结　论

本文基于设备直通提出了一种高效低延迟的中断直通方法,减少了虚拟机 vCPU 调度对 I/O 延迟的影响。本文基于硬件辅助技术,搭建了中断直通架构,并设计了中断重定向机制。实验结果表明网络往返时延平均减少了 34.1%,吞吐量最高提升 7.9%,Apache 测试每个服务器请求所需时间平均减少了 13.6%,磁盘 I/O 操作时延平均减少了 6.7% ~8.4%。这说明中断重定向优化有效提高了虚拟机 I/O 响应性,并取得良好的性能效益。本文设计基于 LoongArch 平台实现,但也适用于其他架构。后续工作将继续围绕构建高吞吐、低延迟的 I/O虚拟化架构展开,并加强系统的可扩展性和可迁移性。

参考文献

[1] TU C C, FERDMAN M, LEE C T, et al. A comprehensive implementation and evaluation of direct interrupt delivery[J]. ACM SIGPLAN Notices, 2015,50(7):1-15.

[2] ZHANG W, HU X, LI J, et al. CoINT: proactive coordinator for avoiding interrupt ability holder preemption problem in VSMP environment[C]// INFOCOM 2018-IEEE Conference on Computer Communications. Honolulu, USA:IEEE, 2018:477-485.

[3] LI J, XUE S, ZHANG W, et al. When I/O interrupt becomes system bottleneck: efficiency and scalability enhancement for SR-IOV network virtualization[J]. IEEE Transactions on Cloud Computing, 2019, 7(4): 1183-1196.

[4] DALL C, LI S W, LIM J T, et al. ARM virtualization: performance and architectural implications[C]// 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). Seoul, Korea: IEEE, 2016:304-316.

[5] SCHILDERMANS S, SHAN J, AERTS K, et al. Virtualization overhead of multithreading in X86 state-of-the-art and remaining challenges[J]. IEEE Transactions on Parallel and Distributed Systems, 2021,32(10):2557-2570.

[6] LIM J T, NIEH J. Optimizing nested virtualization performance using direct virtual hardware[C]// Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. Lausanne, Switzerland: ACM, 2020:557-574.

[7] 朱琛, 王剑, 高翔, 等. 龙芯 KVM 虚拟机 I/O 中断子系统的优化[J]. 高技术通讯, 2020,30(9):893-900.

[8] BUGNION E, NIEH J, TSAFRIR D, et al. Hardware and software support for virtualization[J]. Synthesis Lectures on Computer Architecture, 2017,12(1):206.

[9] PCISIG. PCI Express Base Specification Revision 5.0 [EB/OL]. [2023-07-16]. https://pcisig.com/specifications/.

[10] KEGEL A, BLINZER P, BASU A, et al. Virtualizing IO through the IO memory management unit[R]. Atlanta, USA: ASPLOS-XXI Tutorials, 2016.

[11] Intel. Intel © Virtualization technology for directed I/O architecture specification[EB/OL]. [2023-07-16]. http:

//www. intel. com/content/dam/www/public/us/en/documents/product-specifications/vt-directed-io-spec. pdf.

[12] AMD. AMD I/O virtualization technology (IOMMU) specification, 48882[EB/OL]. [2023-07-16]. https://www. amd. com/system/files/TechDocs/48882 _ IOMMU. pdf.

[13] ARM. ARM generic interrupt controller v3 and v4 - virtualization[EB/OL]. [2023-07-16]. https://developer. arm. com/documentation/107627/0101/? lang = en.

[14] XU C, GAMAGE S, RAO P N, et al. vSlicer: latency-aware virtual machine scheduling via differentiated-frequency CPU slicing[C]//Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing. Delft, Netherlands: ACM, 2012: 3-14.

[15] XU C, GAMAGE S, LU H, et al. vTurbo: accelerating virtual machine I/O processing using designated turbo-sliced core[C]//ATC 2013-2013 USENIX Annual Technical Conference. San Jose, USA: USENIX Association, 2013:243-254.

[16] CHENG L, WANG C L. vBalance: using interrupt load balance to improve I/O performance for SMP virtual machines[C]//Proceedings of the 3rd ACM Symposium on Cloud Computing. San Jose, USA: ACM, 2012:1-14.

[17] LI J, MA R, GUAN H, et al. vINT: hardware-assisted virtual interrupt remapping for SMP VM with scheduling awareness[C]//2015 IEEE 7th International Conference on Cloud Computing Technology and Science. Vancouver, Canada: IEEE, 2015:234-241.

[18] CHENG L, LAU F C M. Offloading interrupt load balancing from SMP virtual machines to the hypervisor[J]. IEEE Transactions on Parallel and Distributed Systems, 2016,27(11):3298-3310.

[19] GORDON A, AMIT N, HAR'EL N, et al. ELI: bare-metal performance for I/O virtualization[C]//Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems. New York, USA: ACM, 2012:411-422.

[20] 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.

[21] HAR'EL N, GORDON A, LANDAU A, et al. Efficient and scalable paravirtual I/O system[C]//Proceedings of the 2013 USENIX Conference on Annual Technical Conference. San Jose, USA: USENIX Association, 2013: 231-242.

[22] CHENG K, DODDAMANI S, CHIUEH T C, et al. Directvisor: virtualization for bare-metal cloud[C]//Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. Lausanne, Switzerland: ACM, 2020: 45-58.

[23] HU X, ZHANG W, LI J, et al. ES2: aiming at an optimal virtual I/O event path[C]//2017 46th International Conference on Parallel Processing. Bristol, UK: IEEE, 2017:141-150.

[24] HU X, LI J, MA R, et al. ES2: building an efficient and responsive event path for I/O virtualization[J]. IEEE Transactions on Cloud Computing, 2022, 10(2):1358-1372.

[25] KIVITY A, KAMAY Y, LAOR D. KVM: the Linux virtual machine monitor[C]//Proceedings of the Linux Symposium. Ottawa, Canada: Linux, 2007:225-330.

[26] LOONGSON TECHNOLOGY. LoongArch Documentation [EB/OL]. [2023-07-15]. https://loongson. github. io/LoongArch-Documentation/.

[27] 龙芯公司. LoongArch[EB/OL]. [2023-07-16]. https://www. loongson. cn/system/loongarch.

[28] 龙芯公司. Loongnix 操作系统 - 龙芯开源社区[EB/OL]. [2023-07-16]. http://www. loongnix. cn/zh/loongnix/.

[29] Netperf. Netperf[EB/OL]. [2023-07-16]. http://www. netperf. org.

[30] Apache. ab-Apache HTTP server benchmarking tool-Apache HTTP Server Version 2. 4[EB/OL]. [2023-07-16]. https://httpd. apache. org/docs/2. 4/programs/ab. html.

[31] Fio. fio[EB/OL]. [2023-07-16]. https://fio. readthedocs. io/en/latest/fio _ doc. html.

[32] Devin. lookbusy-a synthetic load generator[EB/OL]. [2023-07-16]. http://devin. com/lookbusy/.

[33] 华为. 虚拟化 CPU 超分复用比[EB/OL]. [2023-07-16]. https://forum. huawei. com/enterprise/zh/thread/580934109621010432.

[34] 电信天翼云. 云服务器虚拟化超分与虚机性能关系分析[EB/OL]. [2023-07-16]. https://www. ctyun. cn/developer/article/415236377866309.

[35] Apache. The Apache HTTP Server Project[EB/OL]. [2023-07-16]. https://httpd.apache.org/.

An efficient and low-latency interrupt passthrough method based on device passthrough

LV Chen[*,**,***], ZHANG Fuxin[*,**,***], ZHU Chen[****], MAO Bibo[****], DENG Pingke[*****], PAN Xiaohan[*****]

([*]State Key Laboratory of Processors, Institute of Computing 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)

([*****]China Mobile Communications Corporation Research Institute, Beijing 100032)

Abstract

Aiming at the problem that the virtual central processing unit (CPU) scheduling delay of symmetric multi-processing (SMP) virtual machine(VM) will reduce the virtual machine input/output (I/O) responsiveness, an efficient and low-latency interrupt passthrough method based on device passthrough is proposed in this paper. Based on hardware-assisted technology, this method builds an interrupt pass-through architecture, and designs an interrupt redirection mechanism to redirect interrupts from the preempted vCPUs to running ones. The experimental results show that the round-trip time of the network is reduced by an average of 34.1%, the throughput is increased by up to 7.9%, and the time required for each server request with Apache test is reduced by an average of 13.6%, and the average latency of disk I/O operations is reduced by 6.7% ~8.4%. The experimental results demonstrate that this method can effectively reduce the impact of virtual machine vCPU scheduling on I/O latency and improve virtual machine I/O responsiveness.

Key words: interrupt remapping, input/output (I/O) virtualization, device passthrough, kernel-based virtual machine(KVM), I/O responsiveness