基于 KVM 的远程声卡显卡虚拟化技术①

姜邦杰[1], 吴俊敏[1,2], 朱小东[1], 李科君[2], 罗 琳[2], 张朋飞[2], 胡 蝶[1]

[1](中国科学技术大学 计算机科学与技术学院, 合肥 230027)
[2](中国科学技术大学 苏州研究院, 苏州 215123)

摘 要: 针对家庭多媒体环境下的设备共享问题, 提出了一种基于 KVM 虚拟机的远程设备虚拟化技术方案. 该方案利用了 KVM 虚拟机的设备模拟技术, 在其设备模拟模块 QEMU 中实现远程设备虚拟化, 使得本地用户能够像使用本地的设备一样使用远程的真实物理设备. 接着描述了远程虚拟声卡和显卡的具体实现以及相应的优化措施, 并且利用远程虚拟显卡实现了屏幕扩展功能,最后通过实验进行性能分析. 本方案优点在于仅需要在用户态的虚拟设备层添加所需要的远程设备的虚拟化, 方便灵活且系统的安全性高同时不需要修改客户操作系统.

关键词: KVM 虚拟机; 设备虚拟化; 设备共享; 多媒体设备; QEMU 设备模拟; 屏幕扩展

Remote Sound Card and Graphics Card Virtualization Based on KVM

JIANG Bang-Jie[1], WU Jun-Min[1,2], ZHU Xiao-Dong[1], LI Ke-Jun[2], LUO Lin[2], ZHANG Peng-Fei[2], HU Die[1]

[1](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
[2](Suzhou Institute for Advanced study, University of Science and Technology of China, Suzhou 215123, China)

Abstract: A remote device virtualization design project based on Kernel-based Virtual Machine (KVM) is proposed according to the demand of sharing home multimedia device. The project first implements remote device virtualization using the technology of QEMU device emulation. And then we describe the detailed implementation of the sound card and graphics card and optimize them. At last we analyze the performance through experiments. Our design only needs to add a remote device virtualization in the virtual device layer in user mode which is flexible and reliable without modifying the operating system.

Key words: KVM; I/O virtualization; device sharing; multimedia device; QEMU; screen expansion

随着网络的高速发展以及人们生活水平的不断提高, 诸如手机, 电脑, 电视等家庭多媒体环境下的各种设备, 已经越来越智能化. 为了合理利用这些资源, 需要加强设备之间的相互协作, 如利用卧室的平板电脑播放电影可以在客厅的电视机中观看. 利用 DLNA(Digital Living Network Alliance)[1]可以将这些设备互联起来, 并能够使用各自功能和资源, 但使用 DLNA 有诸多缺点, 如易用性较差, 软件的维护比较困难等, 因此新的高效方便可靠的解决方案有待提出.

虚拟化技术能够实现资源的隔离, 安全性高, 且实现在系统层对上层用户透明. 基于这一优点, 我们可以运用设备虚拟化技术[2-4]来虚拟承载着各种数字多媒体信息的 I/O 设备, 这些设备可以是本地也可能是远程的设备. 利用这些虚拟设备间接实现了设备之上的数字媒体信息的共享.

目前国内外对远程设备的虚拟化存在一些研究. 如 SPICE[5](独立计算环境简单协议)是红帽企业虚拟化桌面版的三大主要技术组件之一, 它是一种具有自适应能力的远程提交协议, 它主要提供的是远程桌面虚拟化. DVMM[6](Distributed Virtual Machine Monitor)是一种分布式虚拟机监控系统, 通过它可以将多台计算机资源通过虚拟化整合为单一系统映像的虚拟机来集中管理, 它主要应用与集群系统.

本文根据家庭多媒体环境下, 各种设备共享的需

① 基金项目:中国科学院计算技术研究所国家重点实验室开放项目(CARCH201204);国家自然科学基金(61272132);
中央高校基本科研业务费专项资金(WK0110000020)
收稿时间:2013-03-16;收到修改稿时间:2013-05-06

求, 提出了一种基于 KVM[7-11]虚拟机的远程设备虚拟化技术. 该方案利用了 KVM 虚拟机的设备模拟技术, 在其设备模拟模块 QEMU 中实现远程设备虚拟化, 使得本地用户能够像使用本地的设备一样使用远程的真实物理设备, 且不修改客户操作系统和 KVM 内核模块, 灵活性好, 系统安全性高.

1 基于KVM的远程设备虚拟化技术

1.1 KVM 虚拟机简介

虚拟机(Virtual Machine)是指通过软件模拟具有完全硬件功能且运行在一个相对独立环境下的虚拟计算机系统. 运行在该虚拟系统上的操作系统叫做客户操作系统, 对于该客户操作系统所使用的设备可能是真实存在的硬件设备, 也可能是软件模拟的虚拟设备.

KVM (Kernel-based Virtual Machine)是一款基于 GPL 授权方式的开源虚拟机软件. 如上图 1 所示, KVM 作为内核的一个驱动模块加载到内核中. 其采用基于 Intel VT 技术的硬件虚拟化方法, 并结合了 QEMU 来提供设备虚拟化. 对于客户操作系统使用的几乎所有虚拟设备均由QEMU来模拟. KVM内核模块也即 KVM 驱动可以截获客户操作系统对设备的访问请求, 交由 QEMU 来处理, QEMU 通过解析对应的设备访问指令, 来模拟该指令访问真实设备的效果.

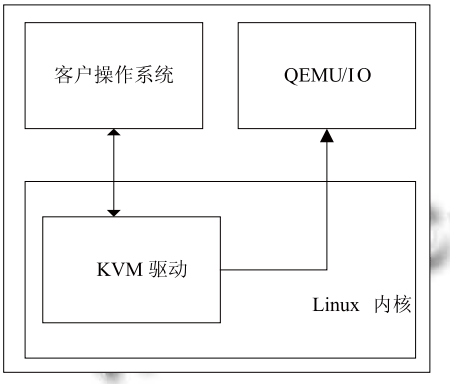

图 1　KVM 架构图

1.2 设计目标

我们的设计分为本地结点一端和远程结点一端, 本地结点利用 KVM 虚拟机运行客户操作系统, 而远程结点拥有真实的物理设备如声卡显卡, 并通过 QEMU 模拟来提供给本地结点使用. 设计目标以声卡显卡的共享为例: 在本地结点的客户操作系统中播放一首音乐可以在远程结点电脑上连接的音箱播放; 本地结点的客户操作系统桌面窗口可以在远程结点的电脑上显示并可以通过远程结点的鼠标操作; 将远程结点的桌面和本地桌面整合为一个整体实现屏幕的扩展功能.

1.3 设计思路

KVM 虚拟机中的 QEMU 已经实现了设备的虚拟化模块, 因此我们要想虚拟远程设备, 就需要在 QEMU 中添加对应的远程设备虚拟化模块. 对于每一个我们想要在本地使用的远程多媒体设备, 都可以在本地 KVM 虚拟机的 QEMU 中添加对应该设备的远程虚拟设备前端, 负责截获客户操作系统对远程虚拟设备的访问, 而对于远程拥有真实物理设备的一端则建立该虚拟设备的后端, 负责接收前端的设备访问指令并加以模拟, 最终调用真实的设备供本地使用. 我们的实现仅仅需要修改用户态的虚拟设备层, 与真实设备驱动和物理设备之间的耦合较小, 一方面简化了实现, 使得客户操作系统可以不经修改就能正常运行, 对硬件也没有特殊的要求, 另一方面, 因为改动的层次较为集中, 因此对系统其他模块的影响较小, 提高了整个系统的健壮性.

1.4 总体设计

如图 2 所示为系统的整体框架图, 主要分为如下模块: 本地设备模型, 远程设备模型前端, 远程设备模型后端以及通信模块. 下面对各个模块的功能作简要描述.

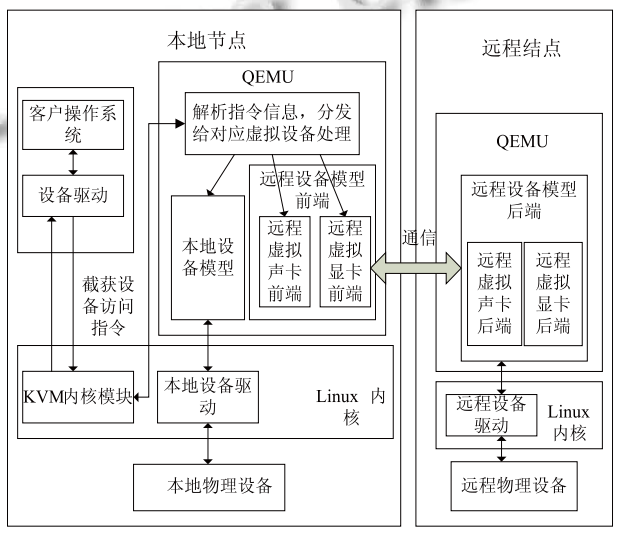

图 2　总体设计框架

(1) 本地设备模型模块主要负责模拟本地的设备, 该模块在 QEMU 中已经实现.

(2) 远程设备模型前端主要负责为本地结点的客户操作系统模拟远程的设备. 当客户操作系统发出对这些远程设备的访问时, 这些访问指令被 QEMU 截获交由远程设备模型前端来处理. 前端解析该指令, 并根据具体的指令将对应的信息通过通信模块发送给远程结点的后端来处理. 同理当从远程结点接收到后端的指令时, 解析该指令并对客户操作系统发出相应操作.

(3) 远程设备模型后端在远程结点, 该模块保存所有的远程虚拟设备状态信息, 它接收远程设备前端发送的设备访问指令, 并解析该指令然后通过远程设备驱动对真实的设备发出相应操作. 同理对于真实设备发出的指令, 它通过通信模块将其交由前端处理.

(4) 通信模块主要负责前后端设备模型的数据以及控制信息的传输, 为保证传输数据的可靠性, 我们采用稳定的 TCP/IP 协议流, 这套协议流保证了数据的正确传输.

远程设备前后端通过通信模块相互协作, 共同构成了远程设备的虚拟化模块.

2 远程虚拟声卡显卡的详细设计

2.1 远程虚拟声卡设计及优化

远程声卡的虚拟化我们选择 ES1370 声卡作为虚拟目标. 对于该声卡的虚拟化, 主要工作是处理三种设备访问指令: I/O 端口读写指令, 音频数据传输指令, 虚拟中断指令. 图 3 详细描述了远程设备模型前后端对这三种指令的模拟.

(1) I/O 端口读写指令: 客户操作系统发出 I/O 读写指令被远程虚拟声卡前端截获后通过通信模块发送到远程虚拟声卡后端, 对于读指令将对应的信息写入虚拟设备状态对应域, 而读指令则读取对应的状态信息并通过通信模块返回给前端并返回给客户操作系统.

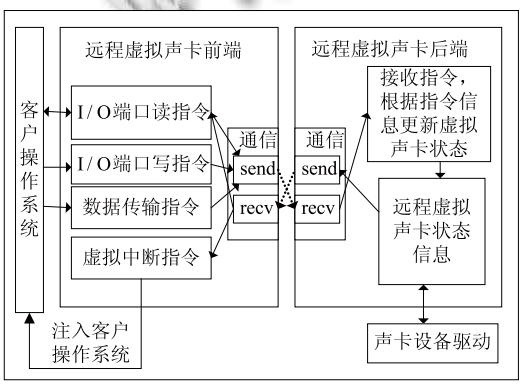

图 3 远程声卡虚拟化详细设计图

(2) 数据传输指令: 前端解析对应指令并将对应的数据传送到后端, 然后调用 ALSA 对应 lib 库函数实现播放.

(3) 虚拟中断指令: 虚拟中断指令由远程虚拟声卡后端发出, 然后通过通信模块交由前端来处理, 前端通过 KVM 内核模块在适当的时机注入客户操作系统中.

如图 3 所示, 远程虚拟声卡状态信息只保存在后端, 这么做是为了简化实现以及可靠性. 但是这么做也产生了一定的性能损失, 对于每条 I/O 端口读写指令, 均需要将其发送到远程结点, 一方面增加了网络的通信开销另一方面也增加了 I/O 端口读写指令的延迟. 本文给出的另一种实现方案就是在远程设备模型的前后端均保存相同的远程设备状态信息, 这样对于客户操作系统发出的 I/O 读指令只需要读取本地的远程设备模型前端中的状态信息即可, 既降低了端口访问指令的延迟又减少了网络的通信量, 当然额外的性能损失就是需要同步前后端的状态信息. 下文会利用实验详细分析这两种方案性能差别.

2.2 远程虚拟显卡设计及优化

远程显卡的虚拟化我们选择 VMWARE. 对该显卡的模拟主要包括一些必要的端口访问指令, 虚拟中断信息以及在屏幕显示所必需的数据信息. 对于端口以及中断信息的模拟, 在前文的远程声卡虚拟化已有详细描述, 在此我们主要描述如何模拟真实的显卡来实现本地客户操作系统的桌面窗口在远程结点屏幕上显示. 如图 4 所示远程虚拟显卡的实验结果, 客户操作系统运行在左面的计算机中, 而对应的桌面窗口却在右面的计算机.

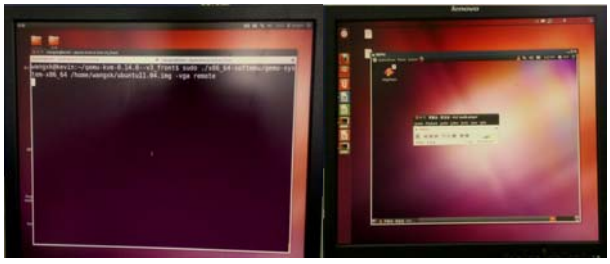

图 4 远程虚拟显卡实验图

对于 VMWARE 显卡, 与显示相关的数据主要包括三种. 一种是用来记录显示内存(VRAM)即整个屏幕对应的待刷新数据; 一种是用来记录屏幕中所有图形图像窗口(FIFO_VRAM)的左上角坐标以及长宽等,

这些窗口信息是要更新屏幕的依据，其中桌面屏幕本身也是一个窗口，左上角坐标为(0, 0)；还有一种是设备寄存器用来记录虚拟显卡的一些状态如屏幕的长，宽，深度和单位像素等.

系统初始化过程中完成对远程虚拟显卡的初始化，包括为显卡配置空间分配内存等，重点就是对上文提到的三种数据分配内存空间，然后建立前后端的通信连接. 当远程虚拟显卡工作时，远程虚拟显卡的前后端完成了远程显卡模拟的主要工作. 在 KVM 设备虚拟化的实现中主要包括两个线程，子线程主要负责截获客户操作系统对设备的访问，主线程则主要处理对设备访问指令的模拟，其中包括访问真实的设备.

如图 5 所示，对于远程虚拟显卡，子线程在通过 IOCTL 系统调用启动客户操作系统运行后，会检查是否有同步信号，也就是客户操作系统的桌面窗口是否

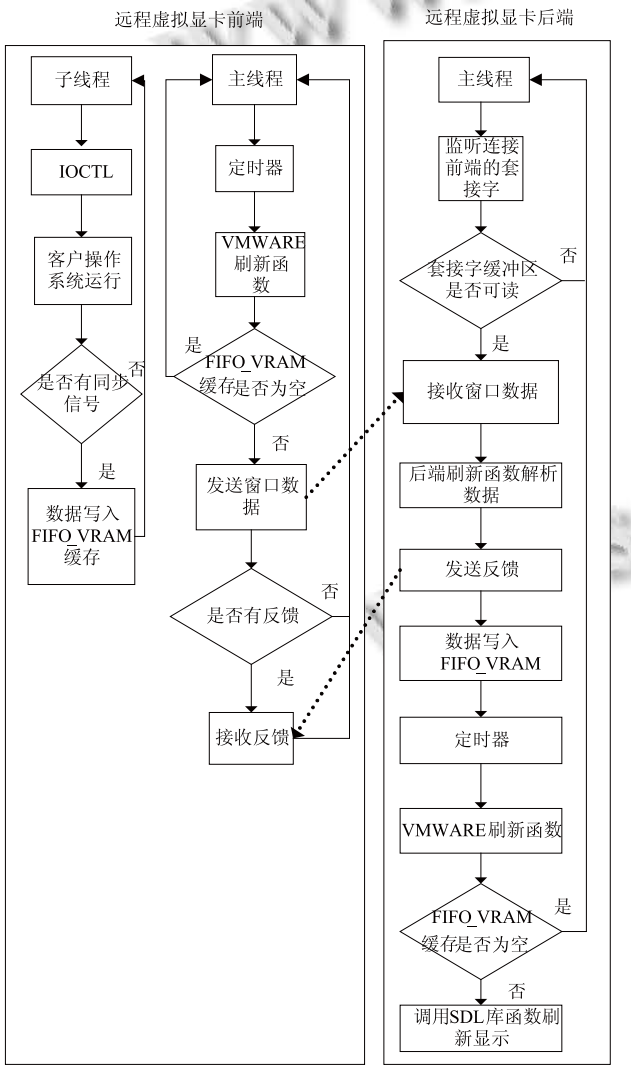

图 5　远程虚拟显卡刷新显示流程图

有新的数据需要刷新显示，如果无则返回，如果有那么就将对应需要显示的窗口信息写入 FIFO_VRAM 缓存中，而主线程利用定时器每隔一段时间调用 VMWARE 刷新函数，该函数的功能是检查对应 FIFO 缓存是否为空，如果为空则返回，如果有数据则获取对应的要刷新显示的窗口坐标以及长宽等然后从对应的显示内存(VRAM)中切割出对应需要刷新的数据，通过通信模块将这些数据发送给远程设备模型后端，后端的主线程在检测到对应数据到来后，将数据分别写入远程虚拟显卡后端的 VRAM 以及 FIFO_RAM 中，然后定时器调用显卡后端的刷新函数在屏幕中刷新显示，该刷新显示主要通过调用 SDL 库来实现.

容易发现在屏幕刷新时需要传输大量的数据到远程节点，在占用网络带宽的同时，也延长了屏幕刷新的时间，导致远程虚拟显卡性能的下降，典型的现象是在频繁的改变客户操作系统中所运行的应用窗口时，屏幕上的桌面显示会出现相对的延迟. 一个简单的想法是减少数据传输时间，具体途径有两种：一种是增加带宽，另一种是减少数据传输量. 对于带宽的增加比如可以从百兆局域网改为千兆局域网，而对于减少数据传输量，可以通过压缩算法，在远程虚拟显卡前端对数据进行压缩，而在后端解压缩. 由于 LZO 压缩算法已有开源库，所以我们选择该压缩算法，下文后会给出相应的性能比较.

2.3 利用远程虚拟显卡实现屏幕扩展

屏幕的扩展主要实现的功能是把远程结点的屏幕作为本地结点的扩展，当本地客户操作系统的桌面窗口拖动超出了本地屏幕显示的范围时，超出的部分将在远程结点的屏幕上显示，如图 6 所示，左面电脑桌面中间为客户操作系统桌面窗口，右面电脑屏幕显示了超出部分的客户操作系统桌面窗口，同样在实现了远程虚拟鼠标和键盘后在右端也可以对该桌面窗口操作，由于篇幅限制，在此不详述.

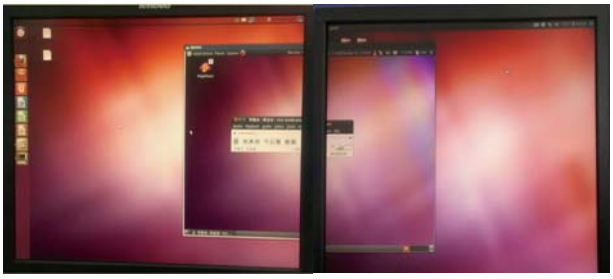

图 6　屏幕扩展实验图

要实现这种屏幕扩展功能一个自然的想法就是将本地超出屏幕部分的窗口数据实时地传送给远程结点, 在远程屏幕上刷新显示. 但是问题是我们的显示内存(VRAM)只记录了当前屏幕中的所有数据, 对于超出的部分数据则不会记录. 因此必须想办法记录这部分数据.

我们的实现方式是在本地结点的客户操作系统中同时启动一个本地虚拟显卡和上文提到的远程虚拟显卡, 远程虚拟显卡作为本地虚拟显卡的扩展, 本地虚拟显卡负责在本地屏幕上刷新显示, 而远程虚拟显卡则负责记录超出屏幕的部分并在远程结点屏幕上显示. 默认的客户操作系统只能使用一个虚拟显卡, 为了能够同时使用两块虚拟显卡, 必须在系统启动后进行设置, 使得远程虚拟显卡是本地虚拟显卡的扩展. 具体操作是:

当客户操作系统启动后, 首先配置以纯文本模式启动, 重启后以 root 用户身份运行: X -configure, 此时在 /root 下会生成一个 xorg.conf.new 文件, cp ~/xorg.conf.new /etc/X11/xorg.conf, 然后编辑 xorg.conf, 在其中加入表 1 所示的代码, 保存退出, 然后配置以图形模式启动, 最后保存重启, 在重启后会看到客户操作系统会同时使用两块显卡, 其中分布式虚拟显卡前端是原生 VMWARE 显卡的扩展, 如下图显示了客户操作系统中的两个虚拟显卡.

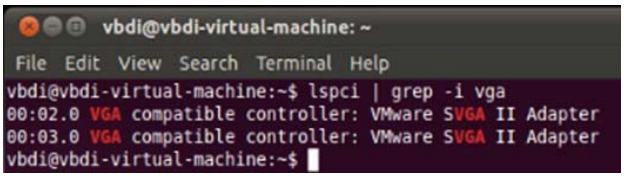

表 1 xorg.conf 配置添加代码

```
Section "ServerFlags"
        Option "Xinerama" "true"
EndSection
```

3 实验结果及分析

3.1 实验平台

本文实验基于 Intel i3-2120 双核处理器, 并以 r8169 网卡通过百兆网线连接. 客户与主机操作系统均为 Ubuntu 11.04, KVM 版本为 0.14.0. 我们的实验数据均采用重复十次实验并对结果取平均值获得.

3.2 远程虚拟声卡性能分析

如表 2 所示为本地虚拟声卡与远程虚拟声卡的 I/O 读写指令, 虚拟中断指令以及音频信息处理指令的耗时即延迟实验数据. 远程声卡 1 指的是本文最初实现的虚拟远程声卡, 而远程声卡 2 指的是前后端保存相同状态信息的远程虚拟声卡. 容易发现本地虚拟声卡的 I/O 读写指令和虚拟中断指令的延迟均在纳秒级, 而远程声卡的延迟则在毫秒级. 而对于音频数据传输的延迟, 本地虚拟声卡和远程声卡 1 均在毫秒级, 可以看出音频数据传输比其它的指令访问具有更大的延迟, 且远程声卡 1 的延迟比本地声卡的延迟高两个数量级.

表 2 虚拟声卡 I/O 访问指令耗时比较

	本地声卡	远程声卡 1	远程声卡 2
虚拟 I/O 读指令	60.4ns	0.209ms	0.001ms
虚拟 I/O 写指令	51.3ns	0.004ms	0.001ms
虚拟中断指令	0.038ns	0.017ms	0.016ms
音频数据传输	0.005ms	0.393ms	0.377ms

对于远程声卡 2, 因为我们的虚拟声卡的前后端均保存有一致的虚拟声卡状态信息, 因此每次客户操作系统发出的虚拟 I/O 读写指令不必发送到远程, 直接可以在远程声卡前端读或写对应信息, 所以节省了通信的延迟, 从实验数据来看, 相比于远程声卡 1, 虚拟 I/O 读指令延迟降低了 2 个数量级.

如图 7 所示, 我们在本地结点分别利用远程声卡 1 和远程声卡 2 播放一首 mp3 格式音乐并比较其网络通信量, 容易发现使用远程声卡 2 后, 其数据发送速率和接收速率均减少了 200KB/S 左右, 我们通过打印远程虚拟声卡的端口访问指令, 发现存在大量的端口读写指令, 远程声卡 2 通过前后端一致性优化后, 对于端口读指令直接可以在本地远程虚拟声卡前端获取, 所以降低了通信量.

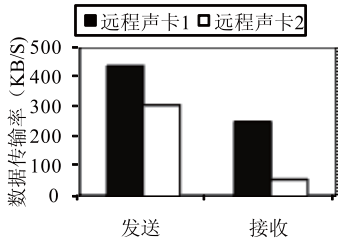

图 7 本地结点使用远程声卡 1 和 2 的通信速率比较

3.3 远程虚拟显卡性能分析

显卡的性能主要比较本地虚拟显卡, 远程虚拟显卡以及通过数据压缩优化后的远程虚拟显卡在不同的带宽下的屏幕上平均每个窗口单位面积刷新时间即每

个窗口的刷新时间除以窗口的面积, 这里我们采用10000 次连续刷新后取平均值. 如下表 3 所示.

表 3 虚拟显卡窗口刷新时间比较

虚拟显卡类型	网络环境	窗口单位刷新时间（ns）
本地虚拟显卡	百兆局域网	270
	千兆局域网	266
远程虚拟显卡	百兆局域网	1483
	千兆局域网	839
数据压缩后的远程虚拟显卡	百兆局域网	1123
	千兆局域网	767

本地虚拟显卡因为不涉及网络传输, 因此网络环境对其基本没有影响, 刷新时间均在270ns 左右. 在百兆局域网环境下, 远程虚拟显卡的刷新时间比本地显卡大四倍左右, 由此看出网络数据传输对刷新有很大的影响, 而压缩后的远程虚拟显卡比压缩前的刷新时间减少了 24.30%, 因此数据的压缩一定层度上减少了刷新的延迟. 同样与百兆网络相比, 千兆的网络对于远程虚拟显卡的窗口刷新时间也有很大幅度的减少.

4 结束语

随着虚拟化技术应用的日益深入和广泛的发展, 人们对虚拟化环境的应用需求也在不断提高. 本文针对家庭多媒体环境下设备之间共享需求, 提出了一种基于 KVM 虚拟机的解决方案, 并给出的详细设计, 同时针对一些性能问题给出了相应的优化措施, 并通过实验分析了优化前后的性能. 该方案从虚拟设备的角度, 完全实现在设备虚拟层, 因此实现简单修改模块较少且对上层用户透明. 因为主要针对家庭多媒体的应用, 下一步主要工作是对其它设备共享的实现以及性能的提高.

参考文献

1 Oh YJ, et al. The DLNA Proxy System Architecture for Sharing In-Home Media Contents via Internet. ICACT. The 8th International Conference,2006,3:1855–1858.

2 刘静波,郭玉东,王晓睿,刘勇.设备虚拟化方法研究与实现.计算机工程与设计,2011,32(8):2874–2877.

3 周平,马捷中.基于开源虚拟机的模拟设备的设计与实现.电子设计工程,2011,18(19):43–56.

4 Intel Corporation,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.北京:清华大学出版社,2009.

5 Redhat.Spice:An open remote computing solution.http://www.redhat.com/resourcelibrary/articles/agilboa-11-spice.

6 宋忠雷,肖利民.分布式 VMM 通信架构的研究与原型实现.计算机工程,2010,36(8):113–116.

7 Kivity A, Kamay Y, Laor D, Lublin U, Liguori A. KVM-the linux virtual machine monitor. Proc. of the Linux Symposium,2007,1:225–230.

8 Rosenblum M, Garfinkel T. Virtual Machine Monitors:Current Technology and Future Trends.IEEE Computer Society,2005.

9 刘锋.Kernel-based Virtual Machine 研究与其事件跟踪机制实现[硕士学位论文].成都:电子科技大学,2009.

10 李胜召,郝沁汾,肖利民.KVM 虚拟机分析.中国计算机学会体系结构专委会学术年会(ACA),2008.

11 Zhang BB, Wang XL, Lai RR, et al. A Survey on I/O Virtualization and Optimization. Fifth Annual of China Grid Conference(ChinaGrid),2010.