第36卷 第16期 计 算 机 工 程 2010年8月
Vol.36 No.16 Computer Engineering August 2010
·开发研究与设计技术· 文章编号：1000—3428(2010)16—0251—03 文献标识码：A 中图分类号：TP391.41

Kernel虚拟机的3D图形加速方法

何家俊，廖鸿裕，陈文智
(浙江大学计算机科学与技术学院，杭州 310027)

摘 要：针对当前Kernel虚拟机(KVM)中3D图形硬件加速不完善的现状，提出一种支持OpenGL加速的改进方法。把提供虚拟机3D应用程序加速的开源虚拟机图形库运行在KVM上，使其支持应用程序OpenGL图形加速，并通过对VMGL框架和功能的改进，使得宿主机渲染后的图像结果回显到KVM上，实现虚拟机上的图形加速。

关键词：虚拟化；Kernel虚拟机；VMGL技术；OpenGL图形加速

3D Graphics Acceleration Method on Kernel-based Virtual Machine

HE Jia-jun, LIAO Hong-yu, CHEN Wen-zhi
(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027)

【Abstract】Aiming at the fact that 3D graphics hardware acceleration is not unconsummated on Kernel-based Virtual Machine(KVM), a method which makes KVM support OpenGL is proposed. It uses an open source project which works for accelerating the 3D application in virtual machine on KVM, and supports accelerating the 3D application. By improving the framework and functionality of VMGL, KVM can receive and display the result, which is rendered and accelerated by the host.

【Key words】virtualization; Kernel-based Virtual Machine(KVM); VMGL technology; OpenGL graphics acceleration

虚拟化技术一直是IT业界的研究热点，不同的虚拟化技术不断涌现。近年来发展势头迅猛的虚拟化解决方案Kernel虚拟机(Kernel-based Virtual Machine, KVM)，由于其良好的构架设计，已成为主流的虚拟化解决方案，在虚拟化环境中运行3D应用程序也成为了一个应用方向[1]，但对于KVM来说，虚拟机中的3D加速还没有获得很好的支持。

1 VMGL项目

OpenGL发展至今已推出2.0版标准，成为跨平台最广泛的三维引擎[2]。而当前KVM并不支持OpenGL加速，因此，多伦多大学和卡内基梅隆大学的研究人员联合开发了一个开源的虚拟机图形库(Virtual Machine Graphic Library, VMGL)。VMGL是一个虚拟化环境中OpenGL渲染加速的解决方案。通过硬件渲染加速解决虚拟机中3D应用程序性能低下的问题。

VMGL能够很好地解决虚拟机中图像渲染性能低下的问题，但该项目仍处于开发初期，存在许多缺陷。本文的研究目的主要针对以下不足提出解决方法。

VMGL的设计希望它能够独立于VMM，但由于各VMM之间的差异，在项目发布时，VMGL只能运行于VMware虚拟机和Xen虚拟化解决方案中，而要在KVM中运行VMGL，需要另外进行配置。

在实现虚拟机OpenGL渲染加速方面，VMGL项目只完成了从虚拟机端到宿主机的2D输出和3D图形命令转移，但缺少将渲染结果再次传输到虚拟机的机制，宿主机渲染后的图像结果只能显示在宿主机中，而不是虚拟机。VMGL这一功能的缺失使虚拟机无法显示图形处理后的结果，并破坏了虚拟机与宿主机的相互隔离性。因此，完成回显功能对于虚拟化环境中的OpenGL加速解决方案具有重大意义。

2 VMGL的启动

2.1 KVM的配置与启动

为了启动KVM，需要确定宿主机的CPU是否支持虚拟化指令扩展，在Linux操作系统下通过命令检查CPU是否支持硬件虚拟化。确定了支持虚拟化后，在宿主机环境中还需要安装Qemu，因为KVM上层依靠Qemu为其模拟操作系统所需的外设。

对宿主机内核进行编译，Linux内核引入了Virtio(包含virtio_pci、virtio_rng、virtio_blk、virtio_net等模块)，它是一种通用的半虚拟化支持方案。由于半虚拟化扩展刚加入内核不久，它的许多特性在内核编译选项中并没有打开，为了支持半虚拟化，Linux在内核编译时需要打开Virtio相关的编译选项。

完成后需要安装虚拟机，用工具Qemu创建格式为qcow的硬盘镜像，在硬盘上安装虚拟机。虚拟机安装完成后，在宿主机中，利用管理员权限加载KVM模块和CPU扩展，完成后，宿主机中会出现/dev/kvm文件。接着加载tunnel/tap模块，因为需要通过桥接方式连接宿主机与虚拟机操作系统。最后，启动KVM，并设置网络连接。

2.2 VMGL的安装配置

VMGL需要分别在宿主机和虚拟机中编译安装，下载源代码包vmgl-0.1.tar.gz，解压后进行编译。编译除了需要gcc、make外，还需要imake、gmake、python和xmkmf。其中，

基金项目：国家“973”计划基金资助项目(2007CB310906)；国家基础科研基金资助项目(A1420080190, 2008ZH76007)

作者简介：何家俊(1984－)，男，硕士研究生，主研方向：操作系统，虚拟化；廖鸿裕，硕士研究生；陈文智，副教授

收稿日期：2010-03-15 **E-mail：**gzhejj@gmail.com

gmake 可以直接链接到系统的 make。编译过程中依赖的库函数有 OpenGL 库、libjpeg、libXaw、libXext、libXmu 等，编译成功后，在宿主机运行 make install-host 进行安装，在虚拟机运行 make install-guest 进行安装。

VMGL 安装完成后，需要按以下步骤配置运行：

(1)在虚拟机中安装 VMGL 时，该项目会对 Xorg 进行扩展，将 libvmglext.so 放在/usr/lib/xorg/modules/extensions/目录下。安装完成后还需要手动更新/etc/X11/xorg.conf 文件，在该文件中添加以下内容[3]：

```
Section "Module"
  Load "vmglext"
EndSection
```

(2)在虚拟机中运行 Xvnc，并将显示窗口设为 1。首次运行时会出现错误，这些错误都是由于 Linux 发行版之间目录结构的差异导致的，根据不同的 Linux 版本，通过链接到正确的目录来解决。

(3)在宿主机中启动编译好的 vncviewer，用 IP 地址连接到虚拟机设置好的图形窗口：

```
/usr/local/bin/vncviewer <ip-address-of-GuestOS>:1
```

(4)在虚拟机中设置环境变量 GLSTUB，连接到宿主机 IP 的 7001 端口：

```
$export GLSTUB=<ip-address-of-HostOS>:7001
```

根据以上方式把 VMGL 配置完成后，就可以在虚拟机中运行 3D 应用程序，运行结果会显示在宿主机上。

3 VMGL 项目的改进

3.1 框架设计

在原来的 VMGL 项目中，图像渲染完成后，图像结果直接显示在宿主机上，没有机制使结果返回虚拟机并显示出来。如图 1 所示，外面运行的是宿主机，里面是虚拟机，虚拟机上运行了 3D 应用程序，程序要求的图像在宿主机上处理完后，直接显示在宿主机上，虚拟机上的应用程序是黑屏。

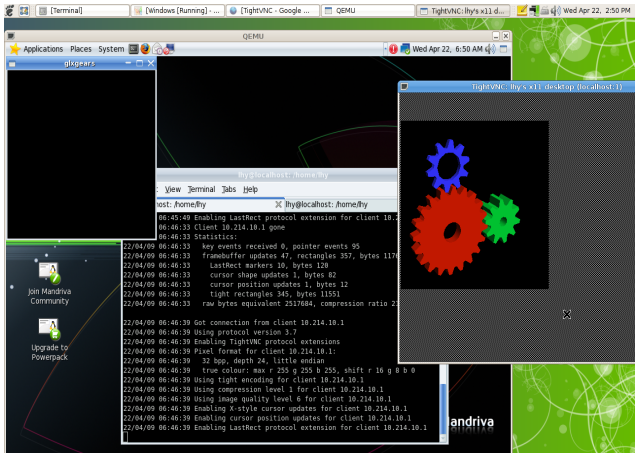

图 1 效果图

为了使得虚拟机能够显示图像结果，通过宿主机渲染后的图像数据应该再传回虚拟机。在原来的项目结构中，图像处理方面只有从 VMGL Guest 到 VMGL Host 的单向数据传输[4]。因此，需要设计另外一种机制，使得经过宿主机处理后的图像结果能够返回虚拟机，单向的图像数据传输变为双向数据传输。

在图 2 中，增加了一条从宿主机到虚拟机的数据传输路线，在主机操作系统完成了虚拟机要求的 3D 图像加速后，通过下文设计的机制，经过 VMGL Host 把结果传回到 KVM 上，虚拟机在接收到数据以后，配合已有的 2D 处理结果就可以完整地在虚拟机中显示应用程序的运行过程和结果[5]。

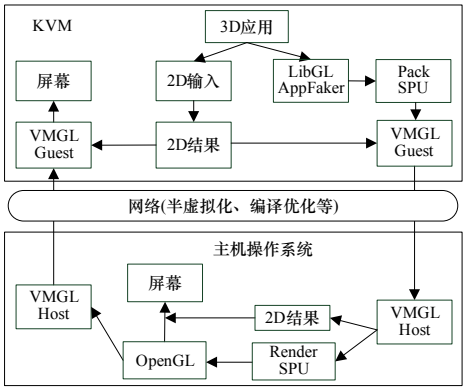

图 2 新的框架

3.2 图像回显

为实现经宿主机渲染加速后的图形命令结果能够回显在虚拟机上，需要对宿主机的运行结果进行一系列的处理和传输，并在虚拟机中实现显示。这一过程中主要涉及的设计实现可分为虚拟机给宿主机发送回显命令、宿主机图像渲染、虚拟机渲染结果的获取与显示等方面。

3.2.1 图形命令的发送

VMGL 项目实现了 KVM 支持 OpenGL 图形加速。但是，实际上图像的计算是在宿主机上完成的，需要应用程序的图形命令以及图像数据以某种形式进行封装，然后从虚拟机传输到宿主机，再进行解码翻译，如此获得原来的处理命令和图像数据。

在图 3 中，虚拟机的 3D 应用程序要求进行图形处理，调用 OpenGL 库函数。虚拟机处理线程接收到该请求，并转发到 VMGL 客户端，VMGL 客户端会把 OpenGL 命令和图像数据进行编码，然后以 VMGL 编码的形式传输给宿主机处理程序。宿主机处理线程需要对 VMGL 形式的命令进行解释，调用 VMGL 的服务端。VMGL 服务端把编码命令解码翻译后，得到原来 3D 应用程序要求的 OpenGL 处理命令及图像数据，再返回给宿主机处理线程。这样，宿主机处理线程就能按照这些命令和数据调用 OpenGL 库，进行 3D 图像渲染，得到 3D 图像加速的结果[5]。

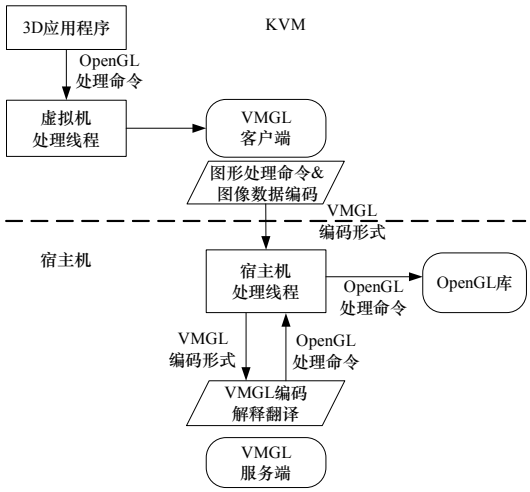

图 3 命令传输流程

利用这种从 VMGL Guest 到 VMGL Host 的命令传输方式，回显命令也可以按照这种方式传输到宿主机上。应用程

序 3D 图像渲染后要在虚拟机上显示，需要虚拟机给宿主机发出回显命令。回显命令经过 VMGL 客户端进行编码，以 VMGL 的编码形式传输到宿主机，再经过 VMGL 服务端的解释翻译，宿主机处理线程得到要求回显的命令，图像回显的开关就会打开，经过宿主机渲染后的图像结果返回给虚拟机。

按照这种虚拟机到宿主机的编码通信方式，虚拟机的命令可以完整地转发给宿主机，因此，可以对命令进行扩展。例如，应用程序得到 3D 图像处理结果后，需要以 2D 的形式简单表示出来，只要增加 2D 处理命令，以编码通信方式告诉宿主机。宿主机只要增加一个模块，实现 3D 图像结果的 2D 简化，然后把 2D 处理结果回显到虚拟机上即可。这样就不需要虚拟机获得处理结果后再进行 2D 转化，而是在宿主机上实现这个转化，缩减了回显需要从宿主机到虚拟机传输的内容。

3.2.2 渲染结果的获取与显示

在宿主机上进行 3D 图像计算后最终得到的结果形式都可以视作一个显示区域上所有像素点的三原色的值。因此，一旦宿主机渲染取得结果后，再加上图像所需显示的大小、质量等因素，发送给虚拟机，虚拟机就可以把结果显示出来[6]。为了使得图像渲染结果能够运行在虚拟机上，提出了以下机制，使得结果从宿主机发送回虚拟机，并在虚拟机中显示，如图 4 所示。

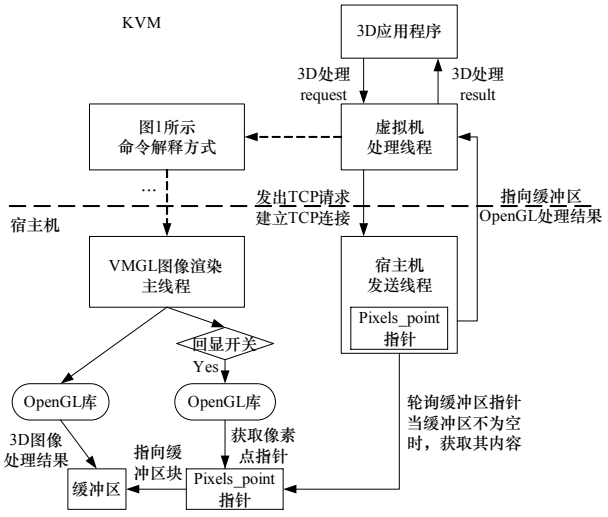

图 4 回显流程

在虚拟机中，有一个处理线程，处理应用程序的 OpenGL 图形处理请求。宿主机中有一个发送线程，该线程把经过 VMGL 主进程图像渲染后的结果发送给虚拟机的处理线程。初始时，处理线程给宿主机发送线程发出 TCP 请求，建立了 TCP 连接，发送线程与处理线程以 TCP 协议方式通信。

当虚拟机中的 3D 应用程序要求进行 OpenGL 图形处理时，处理线程会把图形处理命令以及图像数据以图 1 中编码解码的方式告诉宿主机的处理进程，处理进程再调用 OpenGL 库进行图像处理。经过 OpenGL 图像渲染后，各像素点的值放入缓冲区中，当双缓冲区交换时，表明处理结果已完成，并放入了缓冲区。此时，会判断回显开关是否打开，如果已打开，需要把图像结果返回给虚拟机显示，VMGL 主进程调用 OpenGL 库提供的 glReadPixels 函数，该函数中的一个参数是一个指针，该指针会指向存放渲染结果的内存缓冲区，当调用该函数时，指针就会指向缓冲块的内容[7]。

宿主机中的发送线程不断轮询该指针是否为空，当该指针指向缓冲区渲染结果时，发送线程探测到该指针不为空，取得该指针指向的内存块，并且发送线程读取内存块中图像结果的信息，形成一个图像文件头。这些信息包括图像总大小、文件头大小、宽度、高度等。发送线程把该文件头与内存块都发送给虚拟机，虚拟机中的处理线程接收到后，把处理结果返回给应用程序，在应用程序中显示出来。当这个过程完成后，再把该指针置为空指针，等待下一次处理。

另外，虚拟机只对有变化的帧缓冲区感兴趣，因此，宿主机只需要进行增量更新就可以了。例如，虚拟机中 3D 应用程序只是向右下角放大了窗口，那么在虚拟机中所显示的图像可能只是增加了右侧和下侧的显示区域，原来显示部分并没有改变，此时，宿主机在为虚拟返回图像结果时并不需要传输整个窗口区域，而只是变化了的部分。

3.2.3 结果实现

回显功能实现后，效果如图 5 所示，虚拟机的 3D 应用程序经过宿主机 OpenGL 图像渲染，结果返回给虚拟机，直接运行在应用程序的窗口中。

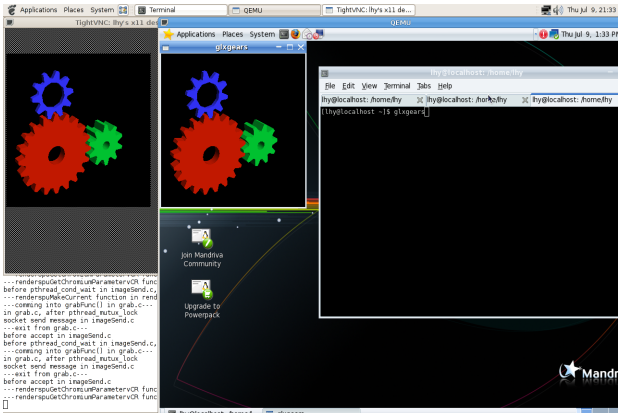

图 5 有回显功能的效果图

4 结束语

本文把 VMGL 图形库运行在 KVM 上，并对 VMGL 框架和功能进行了改进，使得 KVM 支持 3D 应用程序渲染加速，宿主机渲染后的图像结果可以回显到 KVM 上，完成了 KVM 虚拟化环境中的 OpenGL 加速。

参考文献

[1] Nanda S, Chiueh T. A Survey on Virtualization Technologies[R]. Department of Computer Science, Stony Brook University, Technical Report: ECSL-TR-179, 2005.

[2] 沈郑燕, 桑恩方, 李元首. 基于 OpenGL 的多波束剖面声纳数据 3D 可视化[J]. 计算机工程, 2009, 35(12): 198-200.

[3] 何晓龙, 宋吉广. KVM: 驶入虚拟化快车道: Linux 内核虚拟化 KVM 详解[J]. 软件世界, 2007, (11): 48-49.

[4] Sharma M. VMGL Brings 3-D Effects to VMs[EB/OL]. (2008-12-05). http://www.linux.com/feature/154368.

[5] Lagar-cavilla H A, Satyanarayanan M. VMM-independent Graphics Acceleration[C]//Proc. of International Conference on Virtual Execution Environments. [S. l.]: ACM Press, 2007.

[6] Humphreys G, Houston M, Ng R. Chromium: A Stream Processing Framework for Interactive Rendering on Clusters[EB/OL]. (2002-10-09). http://www-graphics.stanford.edu/papers/cr.

[7] 徐元进, 胡光道. 基于 OpenGL 的地学三维图的设计与实现[J]. 计算机工程, 2007, 33(3): 259-261.

编辑 顾逸斐