计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(12):25−31 [doi: 10.15888/j.cnki.csa.006096]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

GPU 虚拟化相关技术研究综述①

余时强, 张为华

(复旦大学 软件学院, 上海 201203)
(复旦大学 上海市数据科学重点实验室, 上海 201203)
(复旦大学 并行处理研究所, 上海 201203)

摘　要: 因为计算密集型应用的增多, 亚马逊和阿里巴巴等公司的云平台开始引入 GPU(Graphic processing unit) 加速计算. 云平台支持多用户共享 GPU 的使用, 可以提升 GPU 的利用效率, 降低成本; 也有利于 GPU 的有效管理. 通过虚拟机监视器以及各种软硬件的帮助, GPU 虚拟化技术为云平台共享 GPU 提供了一种可行方案. 本文综合分析了 GPU 虚拟化技术的最近进展, 先根据技术框架的共同点进行分类; 然后从拓展性、共享性、使用透明性、性能、扩展性等方面对比分析, 最后总结了 GPU 虚拟化的问题和发展方向.

关键词: GPU; 虚拟化; 云计算

引用格式: 余时强,张为华.GPU 虚拟化相关技术研究综述.计算机系统应用,2017,26(12):25–31. http://www.c-s-a.org.cn/1003-3254/6096.html

Survey of GPU Virtualization

YU Shi-Qiang, ZHANG Wei-Hua

(Software School, Fudan University, Shanghai 201203, China)
(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)
(Parallel Processing Institute, Fudan University, Shanghai 201203, China)

Abstract: The emergence of HPC cloud has inspired service provider to deploy GPU in the cloud ecosystem (e.g., Amazon EC2 GPU instance, Aliyun GPU Server). GPU as a computing accelerator is playing an indispensable role in clouding computing. Due to the intrinsic sharing feature of cloud, GPU sharing does not only boost the utilization, lower the cost, but also makes it easier to manage. GPU virtualization comes to solve this problem through Hypervisor and cooperation of software and hardware. This paper collects the methodologies of GPU virtualization and makes a classification and analysis. In addition, it concludes the existing problems and proposes the future works of GPU virtualization.

Key words: GPU; virtualization; cloud computing

1　引言

随着互联网应用的快速发展, 计算机每天需要处理大量的数据, 然而传统软硬件方法相对高昂的维护成本给相关应用的普及造成了极大的障碍. 为了解决该问题, 2006 年谷歌推出“Google 101 计划”, 并提出了云计算的概念. 云计算把 IT 能力转变为可管理的逻辑资源, 虚拟化是云计算的关键技术之一.

虚拟化技术有多种分类, 按实现方式可分为硬件虚拟化和软件虚拟化; 按运行模式可分成全虚拟化和半虚拟化. 软件虚拟化 (如 QEMU[7]) 用软件来仿真硬件平台, 硬件虚拟化 (如 IntelVT[8]) 扩展硬件单元提升 VMM 的处理效率. 全虚拟化 (如 VMware Workstation[4]) 为客户机提供完整的虚拟平台, 半虚拟化 (如 Xen[5]) 需操作系统和 VMM 配合实现.

① 收稿时间: 2017-03-06; 修改时间: 2017-03-23; 采用时间: 2017-04-05

虚拟机监视器又可以分成本地 VMM(类型 I), 和宿主 VMM(类型 II) 两种, 如图 1 所示. 本地 VMM 直接运行在硬件上, 而宿主 VMM 运行在操作系统上. Xen 为类型 I 的代表, 而 VMware Workstation 和 KVM 为类型 II 的代表.

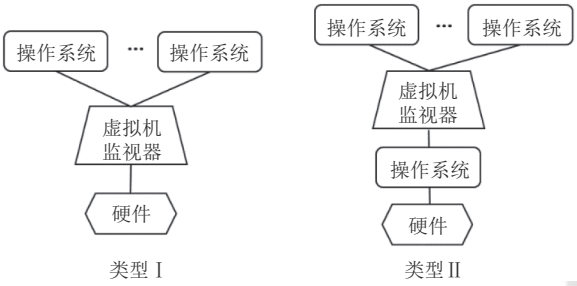

图 1 两种类型虚拟机

GPU(Graphics processing unit) 又称图形处理器, 最初用来加速计算机绘图工作, 已有非常成熟的编程库接口, 如 OpenGL 和 DirectX. 由于 GPU 包含很多计算核, 也可用于通用计算领域. 为了方便开发, NVIDIA 提供 CUDA 编程模型, 支持快速移植 CPU 端耗时但易于并行的部分至 GPU 端, 获取较好的加速效果.

GPU 是一种高性能计算硬件单元, 应用于很多领域, 例如视频编解码、天气预报和通用计算等[13]. GPU 的核数越来越多, 计算能力越来越强大. 亚马逊的 EC2[1]和阿里云[2]等云平台开始使用 GPU 辅助计算. 如果多个用户可以分享 GPU, 则可以提升 GPU 的利用率, 降低硬件成本. 因为多任务的需求, GPU 的虚拟化研究成为趋势.

GPU 的虚拟化技术, 可大致划分为设备模拟、API 重定向、设备直连和 GPU 全虚拟化四种方法, 各有优缺点. 设备模拟用软件模拟硬件, API 重定向封装驱动的 API 实现资源共享, 设备直连把 GPU 独立分配给虚拟机使用, GPU 全虚拟化修改 VMM 管理 GPU. GPU 虚拟化有多种方案, 功能日趋完善. 与 CPU 虚拟化相比, 虚拟 GPU 的个数仍然有限. 另外, 没有方案统一支持图形渲染和通用计算. 而且由于 GPU 自身体系结构的局限, GPU 在多用户的虚拟化场景下存在着严重的安全问题. 本文调研了近年来在 GPU 虚拟化方面的主要研究成果, 对相关技术进行了全面的分析和讨论. 在此基础上, 对现在 GPU 虚拟化存在的一些问题以及发展方向做出了总结和展望.

本文结构如下, 第 1 章介绍云计算与虚拟化相关的背景. 第 2 章介绍 GPU 虚拟化挑战. 第 3 章对当前 GPU 虚拟化技术进行归类, 包括设备模拟、API 重定向、设备直连、以及 GPU 全虚拟化四种, 并对各类别系统特征进行详尽的描述. 第 4 章比较各方法之间的优劣. 第 5 章描述 GPU 虚拟化现存的问题并预测接下来发展趋势, 最后一章总结全文.

2 GPU 虚拟化挑战

GPU 的用途非常广泛, 例如视频编解码、图像渲染、生物医疗和通用计算等方面. 各应用领域的计算需求并不一样. 支持多用户共享 GPU, 虚拟化技术需满足计算多样性的需求.

因为商业竞争, GPU 的设计细节并未公开, 每代产品之间存在很大的差异. 另外, GPU 的硬件接口也不开源, 更没有统一的标准. 为了高效使用 GPU, 系统只能基于厂家提供的驱动. 这些差异增加了 GPU 虚拟化的难度.

程序在虚拟化系统上运行较原生硬件系统带来的性能损耗也是很大的一个挑战, 性能损耗过高将降低 GPU 的利用率, 反而增加成本. GPU 虚拟平台对应用驱动的使用透明性也会很大程度决定其部署难度.

3 GPU 虚拟化方法

GPU 虚拟化主要完成硬件模拟和硬件共享两个目的. 硬件模拟是在没有真实 GPU 硬件可用的环境中, 采用软件模拟的方法仿真硬件设备, 从而提供虚拟 GPU 硬件. 硬件共享是管理 GPU 硬件资源从而使其可以被多用户共享.

为了实现 GPU 虚拟化的目标, 主流 GPU 虚拟化大体上可以分为设备模拟、API 重定向、设备直连和全虚拟化四种方法. 设备模拟在缺少物理硬件情形下使用纯软件方法模拟硬件单元, 供系统中需要 GPU 的应用使用, 用于显示模拟和 GPU 架构研究等领域. API 重定向方法将宿主虚拟机作为管理 GPU 的枢纽, 客户虚拟机在不改变调用接口的情形下使用前后端模块通信的方法共享 GPU. 设备直连解决了虚拟机无法使用 GPU 的问题. 全虚拟化通过修改虚拟机监视器, 使其对 GPU 进行资源管理, 实现 GPU 共享.

3.1 设备模拟

早期的虚拟机只消耗宿主机的计算和存储资源, 为了保证应用的运行, 只能通过模拟外设来提供相应

的功能. 设备模拟是一种用纯软件方式模拟 GPU 硬件逻辑单元功能的方法. 根据需求的不同, 软件模拟的详细程度会有所不同. 最简单的只提供 GPU 对应接口和相关功能. 最详细的可以提供 GPU 硬件每个时钟周期的工作过程. 由于软件方式进行 GPU 各种功能模拟, 该方法具有较好的灵活性和强大的功能, 然而, 由于通过软件的方式模拟硬件行为, 这种方法面临较大的性能挑战, 常用于简单的显示功能和 GPU 架构研究.

在图像处理方面, QEMU 可以模拟简单的 VGA 设备, 来满足虚拟机在使用 GPU 时的简易功能需求, 目前基本具备 2D 显示功能, 已经在一些需要用到 GPU 显示的虚拟系统中应用, 如 KVM 和 Xen 利用 QEMU 负责虚拟显示设备.

在通用计算方面, GPGPU-Sim[6]通过分析 CUDA 编程模型下发射线程的执行方式, 在 CPU 端进行软件模拟运行 PTX 指令集. 在并行编程模型中, GPU 发射一组 (grid) 线程并发执行, 这些线程又会被细分为块 (block). GPGPU-sim 用软件模拟的方式实现了包含六个阶段 (取指、译码、执行、内存 1、内存 2、写回) 的流水线执行单元. 32 个线程在软件模拟的计算单元上以 SIMD 的方式执行, 若线程之间发生条件分支, 则采用后支配的收敛机制进行同步[6]. GPGPU-sim 已广泛应用于各种体系结构或 GPU 新特性的相关研究中.

3.2 API 重定向

研究者发现 GPU 驱动提供了统一的调用接口. 图像渲染的 OpenGL 编程接口和通用并行计算的 CUDA 编程接口与 GPU 内部实现没有关系. API 重定向在接口层面上实现虚拟化, 采用对调用接口二次封装的方法. 这种方法相对简单, 在图像渲染和通用计算都有广泛的应用.

在 API 重定向框架下, 客户虚拟机和宿主虚拟机上设立前后端模块, 前端模块提供应用程序访问的通用接口, 后端模块链接 GPU 驱动直接访问 GPU(对于类型 II 的 VMM 而言, 此处宿主虚拟机是直接运行在硬件上的操作系统), 前后端模块采用 RPC 等协议实现通信, 具体如图 2 所示. 在图像渲染方面, VMGL[26]在客户虚拟机中添加的前端模块与 OpenGL 调用接口一致, 同时在宿主虚拟机中添加后端模块与原生 OpenGL 库链接, 利用 WireGL[28]网络传输协议将客户虚拟机的命令传送给宿主虚拟机, 从而间接访问 GPU. WireGL 通信协议提高了网络带宽的利用率. VMGL 能达到 86% 的原生 GPU 渲染效果.

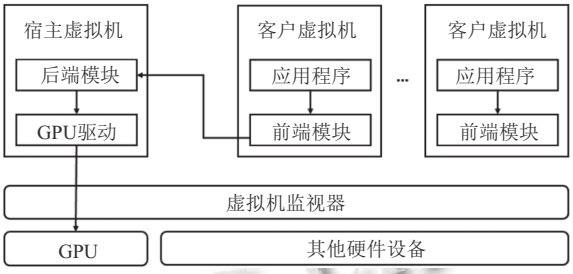

图 2 API 重定向示意图

API 重定向在 GPU 通用计算领域中也有大量应用, 架构基本如图 2 所示. 最主要的区别在于虚拟机监视器和前后端模块通信协议的选择. GVim[9]和 vCUDA[8]使用 Xen 作为实验平台, GVim 使用 XenStore[5]协议来实现前后端通信, 数据通过 pinned 内存映射和虚拟机间内核缓冲区共享传输, 性能主要受 XenStore 的行为以及内存拷贝的影响. vCUDA[22]对比了 VMRPC[34]和 XMLRPC[27]两种通信协议的性能, 发现 VMRPC 协议可以几乎达到原生性能, 而 XMLRPC 协议在进行大量数据传输的情况下性能损耗较为严重. gVirtuS[11]采用类型 II 的虚拟机监视器作为实验环境, 基于 TCP 协议开发 vmSocket 实现通信, 绕过了 CPU和 GPU 之间的数据传输, gVirtuS 在远程 GPU 利用方面有着前两者不可比拟的优势.

3.3 设备直连

早期虚拟机无法直接使用 GPU, 程序只能通过设备模拟来实现 GPU 独有的功能, 该方法只能提供简单功能, 性能较差. 通过将虚拟机中原生的 GPU 驱动与硬件设备对接, 能够极高的利用设备的计算能力. 设备直连是早期一些 IO 设备虚拟化的一种可选方式. 虚拟机能够直接访问硬件资源, 可达到原生系统性能. GPU 在该方法下只能给某一个虚拟机使用, 不能被多虚拟机共享.

Dong 等人[21]提出一种基于硬件 IOMMU(即 Intel 和 AMD 的 I/O 虚拟化技术) 实现 IO 直连的虚拟化方法, 实现了高质量的 I/O 共享. 该方法维护了设备的语义信息, 也保证了虚拟机在切换时 GPU 寄存器信息的一致性. 此外, 中断信息共享机制同步了缓存. 该系统在 PCIe 总线支持虚拟化拓展的情形下可直连 GPU.

一些云服务 (如亚马逊的 EC2) 提供商采用这种直连的方式在云平台上部署 GPU. 但是, 设备直连只能保证 GPU 被一个虚拟机使用, 不适合云计算场景下多用户共享使用的特点. 在计算量不足的任务执行过程中, GPU 的利用率低下, 浪费了计算资源. 设备直连的方法缺少必要的中间维护层和状态跟踪, 对虚拟机的迁移等高级特性支持不足.

3.4 GPU 全虚拟化

设备模拟方法只能模拟一些简单的硬件、而且性能低下, API 重定向虽然能够达到接近原生硬件的性能, 但是需要修改客户虚拟机中程序库, 设备直连方案虽然性能较好, 但是共享性非常差. 近几年的提出的 GPU 全虚拟化将上述方案混合实现客户虚拟机完全使用透明. 该方法对寄存器等硬件上下文信息使用软件模拟, 上下文切换后直连硬件设备能够充分利用 GPU.

GPU 全虚拟化指不需要对虚拟机中的驱动修改, 应用程序就能使用 GPU, 即使用透明性. GPU 全虚拟化技术比设备直连有更好的共享性, 同时在性能方面远超过设备模拟, 也不需要修改客户虚拟机中的驱动, 可以说发展至今最好的 GPU 虚拟化解决方案. Intel 的 Kung Tian 等人提出了 gVirt[18], 实现了图像渲染方面的 GPU 全虚拟方案, 并对该系统进一步优化, 提出 gHyvi[14]和 gScale[16]. 在通用计算领域, Yusuke Suzuki 等人提出 GPUvm[12]系统, 通过修改 VMM 实现了 GPU 全虚拟化.

gVirt 基于 Xen 开发, 整体架构如图 3 所示. 为了实现 GPU 全虚拟化, gVirt 对 Xen 的宿主虚拟机 (Dom 0) 的一些修改, 见图 3 中的灰色部分. gVirt 区分了 GPU 的资源, 性能关键资源, 包括显存以及指令缓存; 非性能关键资源, 包括一些 I/O 寄存器. 为了接近于原生 GPU 的性能, gVirt 用隔离机制直接访问性能关键资源, 用软件模拟的方法支持非关键资源的访问. gVirt 采用影子页表结构等内存划分技术, 保证每个虚拟机拥有自己独立的内存; 并提出了地址空间膨胀 (address space ballooning) 技术来减少地址翻译的开销. gVirt 实现了 GPU 调度器, 调度时间片比较长, 以降低 GPU 任务上下文切换的开销. GPU 的调度算法和 Xen 的 CPU 调度算法相互独立, 这使得 CPU 和 GPU 可以并行访问显存. gHyvi 基于 Xen 的严格写保护影子页表机制提出了一种宽松影子页表机制, 减少了影子页表的更新次数和陷入 VMM 同步的开销. gScale 动态分配显存, 避免虚拟机数目增加导致每个虚拟机平摊到的空间变少的问题. gScale 比 gVirt 的可扩展性提高了 4 到 5 倍.

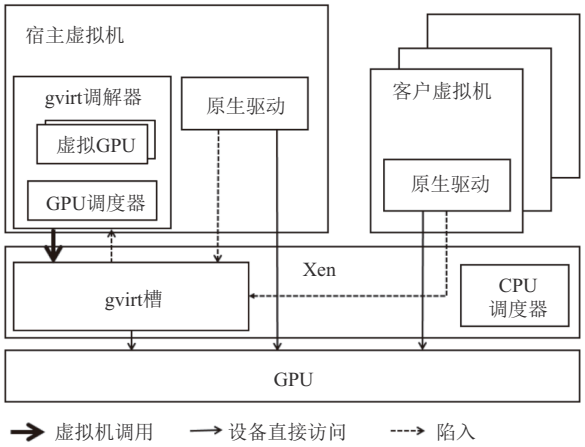

图 3 gVirt 框架示意图

在通用计算领域, GPUvm 的 GPU 全虚拟化实现的更加的彻底, 其修改只在 Xen 中完成. 与 gVirt 类似, GPUvm 也采用 GPU 影子页表的机制将显存进行隔离, 每个虚拟机访问属于自己那部分的内存. 除此之外, CPU 与 GPU 间的命令传输队列也被虚拟化, 即每个虚拟机都有各自的队列结构, 虚拟机进行切换时, 命令队列也会对应切换. GPUvm 采用一种宽带感知的非抢占算法进行调度, 能够保证 GPU 被在各个虚拟机之间均衡切换.

4 评价

设备模拟、API 重定向、设备直连和 GPU 全虚拟化从不同的角度提供了 GPU 虚拟化的解决方案. 设备模拟能够在没有真实硬件的环境下虚拟出 GPU, 该特性是后三者不可比拟的, 而且该方法能够模拟新型硬件单元, 帮助科研工作者探索新一代 GPU 硬件结构, 具有良好拓展性. 应用程序几乎不用修改就可以直接运行, 有着良好的使用透明性. API 重定向需要对接口进行二次封装, 因此使用透明性很差, 重定向过程带来了性能损耗. 宿主虚拟机后端模块可以与各客户虚拟机的前端模块进行对接, 实现多虚拟机共享. 设备直连在程序使用过程中能达到原生 GPU 性能, 由于只能给某个特定虚拟机使用, 所以共享性不好. GPU 全虚拟化方法使用真实物理硬件, 因此没有类似于软件模拟的拓展性; 该方法可直接运行现有的应用程序, 透明性良

好; 全虚拟化通过虚拟机监视器管理 GPU 资源, 实现 GPU 共享.

我们对评估特性进行了总结, 描述如下:

(1) 拓展性: 虚拟过程中 GPU 功能的可拓展性. 该特性评估对新功能的支持与延展性.

(2) 共享性: 在虚拟化场景下, 共享性是虚拟化的重要关注点. 该特性评估 GPU 利用率的影响以及 GPU 是否可以被多用户并发使用.

(3) 使用透明性: 客户虚拟机使用虚拟 GPU 时是否需要修改程序或库文件. 该指标评估在 GPU 虚拟化支持下对原应用程序的兼容性.

(4) 性能: 虚拟环境下 GPU 的使用性能 (与原生 GPU 使用做对比). 该指标评估 GPU 在虚拟化支持下的性能损耗.

(5) 扩展性: 系统虚拟出 GPU 的个数. 该指标评估 GPU 可被多少个虚拟机同时使用.

其中拓展性、共享性和使用透明性是不可度量的, 性能和扩展性是可度量的. 表 1 给出四种 GPU 虚拟化方法在各评估特性上的表现, 强表示满足某特性, 弱表示不满足该特性或支持有限.

表 1 各系统评估表

分类	系统名称	拓展性	共享性	性能	使用透明性	扩展性	使用领域
设备模拟	Qemu	强	弱	弱	强	弱	通用计算
	GPGPU-sim	强	弱	弱	强	弱	图像渲染
API重定向	VMGL	弱	强	强	弱	弱	图像渲染
	vCUDA	弱	强	强	弱	弱	通用计算
	Gvim	弱	强	强	弱	弱	通用计算
设备直连	Amazon EC2	弱	弱	强	强	弱	通用计算、图像渲染
	Intel VT-d	弱	强	强	强	弱	图像渲染
全虚拟化	gVirt	弱	强	强	强	弱	图像渲染
	GPUvm	弱	强	强	强	强	通用计算

5 展望

综前所述, GPU 虚拟化有多种技术方案, 但没有通用计算和图形渲染两个领域通用方案. 全虚拟化方案可在多虚拟机之间共享 GPU, 但 GPU 不可抢占的特性给资源调度带来了一定困难. 另外, GPU 的虚拟化很少考虑安全性问题.

根据本文对当前 GPU 虚拟化方案的分析, 我们认为 GPU 虚拟化今后可能会关注下列三个方面的研究:

(1) 可抢占性: 由于 GPU 核数较多, 抢占 GPU 需要保存大量的上下文信息, 开销较大, 所以目前市场上 GPU 都不支持抢占特性. 只用当前任务完成之后, GPU 才能被下个应用程序使用. 在 GPU 虚拟化的环境中, 多用户使用的场景会导致 GPU 进行频繁的任务切换, 可抢占的 GPU 能够防止恶意用户长期占用, 并且能够实现用户优先级权限管理.

(2) 通用性: 现在很多 GPU 虚拟化技术都是只针对某一种用途实现, 例如 VMGL 是能用来做图像渲染计算的虚拟化, 无法实现 GPU 的通用计算. GPUvm 则只针对于 NVIDIA 的 GPU 做通用计算的虚拟化, 无法用于图像处理. 这些系统都不是通用的 GPU 虚拟化解决方案, 而且只适用于某种虚拟机监视器架构. 通用的虚拟化架构有利于系统部署, 使得 GPU 在各种云计算平台发挥更大的作用, 所以 GPU 虚拟化的通用性也是未来需要解决的问题.

(3) 安全性: 由于云平台的多用户共享特性, GPU 虚拟化带来的安全问题也开始变得突出, 恶意用户可能会绕过当前虚拟化技术薄弱的保护机制进行攻击. Roberto 等人[3]通过分析 GPU 显存残余信息获取用户的私密数据, 所以如何保证虚拟机数据不被窃取也是 GPU 被用户共享的前提.

6 总结

随着计算密集任务的增加, 云平台部署 GPU 来提升系统性能. 为了在云平台上共享 GPU, 同时也有助于提升 GPU 的利用率且便于管理, GPU 虚拟化技术逐渐发展起来. 本文综合分析了 GPU 虚拟化技术近十年的发展, 分为设备模拟、API 重定向、设备直连和全虚拟化四种. 本文进一步剖析了各种系统之间的技术框架和细节, 同时也从拓展性、使用透明性、性能、共享性以及可扩展性等方面对各种技术进行了深入的分析和对比. 最后, 我们总结了 GPU 虚拟化存在的一些问题, 并展望了未来可能存在的研究.

参考文献

1 AMAZON. Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/. [2014].

2 高性能计算. https://hpc.aliyun.com/product/gpu_bare_metal/.

3 Virtualization. https://en.wikipedia.org/wiki/Virtualization. [2017-07-11].

4 Vmware. http://www.vmware.com/.

5 Barham P, Dragovic B, Fraser K, *et al*. Xen and the art of virtualization. ACM Sigops Operating Systems Review, 2003, 37(5): 164–177. [doi: 10.1145/1165389]

6 Bakhoda A, Yuan GL, Fung WWL, *et al*. Analyzing CUDA workloads using a detailed GPU simulator. Proc. of IEEE International Symposium on Performance Analysis of Systems and Software, 2009. ISPASS 2009. Boston, MA, USA. 2009. 163–174.

7 Bellard F. QEMU, a fast and portable dynamic translator. Proc. of the Annual Conference on USENIX Annual Technical Conference. Anaheim, CA, USA. 2005. 41.

8 Hiremane R. Intel virtualization technology for directed I/O (Intel VT-d). Technology@ Intel Magazine, 2007, 4(10).

9 Gupta V, Gavrilovska A, Schwan K, *et al*. GViM: GPU-accelerated virtual machines. Proc. of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing. Nuremburg, Germany. 2009. 17–24.

10 Duato J, Peña AJ, Silla F, *et al*. rCUDA: Reducing the number of GPU-based accelerators in high performance clusters. Proc. of International Conference on High Performance Computing and Simulation. Caen, France. 2010. 224–231.

11 Giunta G, Montella R, Agrillo G, *et al*. A GPGPU transparent virtualization component for high performance computing clouds. In: D'Ambra P, Guarracino M, Talia D, eds. Euro-Par 2010-Parallel Processing. Berlin Heidelberg, Germany. 2010. 379–391.

12 Suzuki Y, Kato S, Yamada H, *et al*. GPUvm: Why not virtualizing GPUs at the hypervisor? Proc. of the 2014 USENIX Conference on USENIX Annual Technical Conference. Philadelphia, PA, USA. 2014. 109–120.

13 Tian K, Dong Y, Cowperthwaite D. A Full GPU virtualization solution with mediated pass-through. Proc. of the 2014 USENIX Conference on USENIX Annual Technical Conference. Philadelphia, PA, USA. 2014. 121–132.

14 Dong Y Z, Xue M C, Zheng X, *et al*. Boosting GPU virtualization performance with hybrid shadow page tables. Proc. of the 2015 USENIX Conference on Usenix Annual Technical Conference. Santa Clara, CA, USA. 2015. 517–528.

15 Han SJ, Jang K, Park KS, *et al*. PacketShader: A GPU-accelerated software router. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 195–206. [doi: 10.1145/1851275]

16 Xue MC, Tian K, Dong YZ, *et al*. gScale: Scaling up GPU virtualization with dynamic sharing of graphics memory space. 2016 USENIX Annual Technical Conference (USENIX ATC 16). Denver, CO, USA. 2016.

17 Xia L, Lange J, Dinda P, *et al*. Investigating virtual passthrough I/O on commodity devices. ACM SIGOPS Operating Systems Review, 2009, 43(3): 83–94. [doi: 10.1145/1618525]

18 Gottschlag M, Hillenbrand M, Kehne J, *et al*. LoGV: Low-overhead GPGPU virtualization. Proc. of the 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC). Zhangjiajie, China. 2013. 1721–1726.

19 Gupta V, Schwan K, Tolia N, *et al*. Pegasus: Coordinated scheduling for virtualized accelerator-based systems. Proc. of the 2011 USENIX Conference on USENIX Annual Technical Conference. Portland, OR, USA. 2011. 3.

20 Dowty M, Sugerman J. GPU virtualization on VMware's hosted I/O architecture. ACM SIGOPS Operating Systems Review, 2009, 43(3): 73–82. [doi: 10.1145/1618525]

21 Dong YZ, Dai JQ, Huang ZT, *et al*. Towards high-quality I/O virtualization. Proc. of SYSTOR 2009: The Israeli Experimental Systems Conference. Haifa, Israel. 2009. Article No.12.

22 Shi L, Chen H, Sun JH, *et al*. vCUDA: GPU-accelerated high-performance computing in virtual machines. IEEE Trans. on Computers, 2012, 61(6): 804–816. [doi: 10.1109/TC.2011.112]

23 Qi ZW, Yao JG, Zhang C, *et al*. VGRIS: Virtualized GPU resource isolation and scheduling in cloud gaming. ACM Trans. on Architecture and Code Optimization (TACO), 2014, 11(2): Article No.17.

24 Rossbach CJ, Currey J, Silberstein M, *et al*. PTask: Operating system abstractions to manage GPUs as compute devices. Proc. of the Twenty-Third ACM Symposium on Operating Systems Principles. Cascais, Portugal. 2011. 233–248.

25 Kato S, McThrow M, Maltzahn C, *et al*. Gdev: First-class GPU resource management in the operating system. Proc.

of the 2012 USENIX Conference on Annual Technical Conference. Boston, MA, USA. 2012. 37.

26 Lagar-Cavilla HA, Tolia N, Satyanarayanan M, *et al*. VMM-independent graphics acceleration. Proc. of the 3rd International Conference on Virtual Execution Environments. San Diego, California, USA. 2007. 33–43.

27 Merrick P, Allen S, Lapp J. XML remote procedure call (XML-RPC): U.S., Patent 7028312. [2006-04-11].

28 Humphreys G, Eldridge M, Buck I, *et al*. WireGL: A scalable graphics system for clusters. Proc. of the 28th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. 2001. 129–140.

29 Becchi M, Sajjapongse K, Graves I, *et al*. A virtual memory based runtime to support multi-tenancy in clusters with GPUs. Proc. of the 21st International Symposium on High-Performance Parallel and Distributed Computing. Delft, The Netherlands. 2012. 97–108.

30 Dong YZ, Yang XW, Li JH, *et al*. High performance network virtualization with SR-IOV. Journal of Parallel and Distributed Computing, 2012, 72(11): 1471–1480. [doi: 10.1016/j.jpdc.2012.01.020]

31 Di Pietro RD, Lombardi F, Villani A. CUDA leaks: Information leakage in GPU architectures. arXiv:1305.7383, 2013.

32 Virtualization solution. http://www.amd.com/en-us/solutions/professional/virtualization.

33 Ravi VT, Becchi M, Agrawal G, *et al*. Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework. Proc. of the 20th International Symposium on High Performance Distributed Computing. San Jose, California, USA. 2011. 217–228.

34 Chen H, Shi L, Sun JH. VMRPC: A high efficiency and light weight RPC system for virtual machines. Proc. of the 18th International Workshop on Quality of Service (IWQoS). Beijing, China. 2010. 1–9.

35 Menychtas K, Shen K, Scott ML. Enabling OS research by inferring interactions in the black-box GPU stack. Proc. of the 2013 USENIX Conference on Annual Technical Conference. San Jose, CA, USA. 2013. 291–296.

36 NVIDIA grid virtual GPU technology. http://www.nvidia.com/object/grid-technology.html.

37 仝伯兵, 杨昕吉, 谢振平, 等. GPU 虚拟化技术及应用研究. 软件导刊, 2015, 14(6): 153–156.

38 赵冰. GPU 虚拟化中安全问题的研究[硕士学位论文]. 西安: 西安电子科技大学, 2014.