(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105786589 A
(43)申请公布日 2016.07.20

(21)申请号 201610107582.2

(22)申请日 2016.02.26

(71)申请人 成都赫尔墨斯科技有限公司
地址 610213 四川省成都市天府新区华阳街道天府大道南段846号

(72)发明人 张微 杨磊 罗涛 曾锦平 邱泳天 周益 陈乐吉 苏永生 杨学亮 雷智聪 唐迎力 付兵 谢琼 陈平

(74)专利代理机构 四川力久律师事务所 51221
代理人 韩洋 熊晓果

(51)Int.Cl.
G06F 9/455(2006.01)
G06F 9/50(2006.01)
G06T 1/20(2006.01)

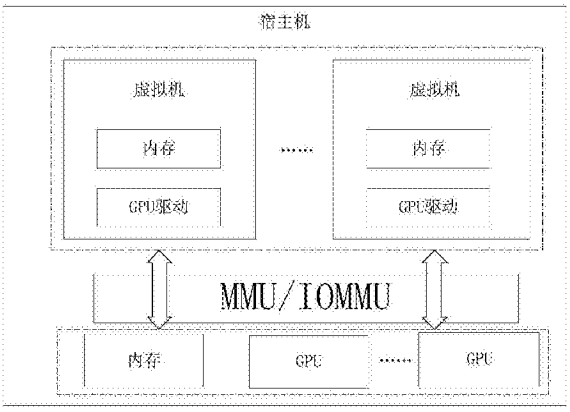
权利要求书2页 说明书6页 附图2页

(54)发明名称

一种云渲染系统、服务器及方法

(57)摘要

本发明公开了一种云渲染系统，包括宿主机及多个GPU，所述宿主机设置有多个虚拟机，每个所述虚拟机都配置有对应的一个GPU驱动；还包括：MMU，耦合至每个所述GPU驱动及每个所述GPU、耦合虚拟机内存与宿主机内存，其被配置为当任一个虚拟机请求访问GPU时，向该虚拟机的GPU驱动分配一个GPU地址，所述GPU地址用于访问该GPU；当任一个虚拟机请求访问虚拟机内存时，向该虚拟机分配对应的宿主机内存地址；IOMMU，耦合至每个所述GPU及虚拟机内存，其被配置为当任一个GPU请求访问虚拟机内存时，向该GPU分配对应的宿主机内存地址。本发明的系统，通过配置MMU与IOMMU使得多个虚拟机都能够独立直接访问GPU，相比现有技术中一般采用的Nvidia VGPU架构，本发明的系统价格低廉、渲染性能高。



CN 105786589 A

1.一种云渲染系统，其特征在于，包括宿主机及多个GPU，所述宿主机设置有多个虚拟机，每个所述虚拟机都配置有对应的一个GPU驱动；

所述云渲染系统还包括：MMU，耦合至每个所述GPU驱动及每个所述GPU、耦合虚拟机内存与宿主机内存，其被配置为当任一个虚拟机请求访问GPU时，向该虚拟机的GPU驱动分配一个GPU地址，所述GPU地址用于访问该GPU；当任一个虚拟机请求访问虚拟机内存时，向该虚拟机分配对应的宿主机内存地址；

IOMMU，耦合至每个所述GPU及虚拟机内存，其被配置为当任一个GPU请求访问虚拟机内存时，向该GPU分配对应的宿主机内存地址。

2.根据权利要求1所述的一种云渲染系统，其特征在于，设置内存地址空间，并将所述宿主机内存映射到内存地址空间，所述内存地址空间用于存储宿主机内存对应的地址，所述虚拟机通过访问内存地址空间中的地址来访问对应的宿主机内存。

3.根据权利要求2所述的一种云渲染系统，其特征在于，所述IOMMU还用于将所述内存地址空间存储的不连续的内存段映射为连续的内存段，以便GPU能够通过DMA技术进行数据读写。

4.根据权利要求1所述的一种云渲染系统，其特征在于，设置GPU地址空间，并将所述GPU映射到GPU地址空间，所述GPU地址空间用于存储GPU控制寄存器对应的地址，所述GPU驱动通过所述GPU地址空间来访问对应的GPU控制寄存器。

5.根据权利要求1-4任一项所述的一种云渲染系统，其特征在于，当任一个所述虚拟机启动时，该虚拟机与一个GPU通过MMU和/或IOMMU进行绑定，且在绑定期间，该已被绑定的GPU不能再与其他虚拟机进行绑定。

6.一种云渲染服务器，其特征在于，包括如权利要求1-5任一项所述的系统，还包括云服务管理平台，用于对所述系统的运行状态进行监控和管理，对使用所述系统的用户进行管理。

7.一种云渲染方法，其特征在于，包括以下步骤：

S1、虚拟机接收渲染请求，并将渲染请求发送到虚拟机的GPU驱动，接收渲染数据，并将渲染数据写入内存；

S2、虚拟机的GPU驱动访问GPU控制寄存器，并将渲染请求信息写入GPU控制寄存器；

S3、GPU根据所述渲染请求信息访问对应的虚拟机内存，并对所述对应的虚拟机内存中的渲染数据进行处理。

8.根据权利要求7所述的一种云渲染方法，其特征在于，所述虚拟机、所述GPU均有多个，且所述虚拟机与所述GPU一一对应，当有多个渲染请求时，每个所述虚拟机分别对应处理所述多个渲染请求中的任一个。

9.根据权利要求7或8所述的一种云渲染方法，其特征在于，所述将渲染数据写入内存包括，MMU耦合所述虚拟机内存与宿主机内存，当虚拟机请求访问虚拟机内存时，通过MMU向该虚拟机分配对应的宿主机内存地址，所述渲染数据根据所述宿主机内存地址进行存储。

10.根据权利要求9所述的一种云渲染方法，其特征在于，所述S2步骤包括：

S201、将宿主机的GPU物理地址空间段映射到宿主机的虚拟地址空间段；

S202、利用GPA-HVA转换表将所述虚拟地址空间段映射到虚拟机中的GPU地址空间；

S203、虚拟机的GPU驱动访问GPU地址空间段，并根据所述地址空间段上的地址信息访

问对应的GPU控制寄存器。

11.根据权利要求10所述的一种云渲染方法，其特征在于，所述S3步骤包括：

S301、IOMMU耦合所述虚拟机内存与GPU，并将虚拟机使用的不连续的内存地址空间映射为连续的地址空间段；

S302、GPU根据渲染请求信息对所述连续的地址空间段进行DMA读写，获取渲染所需的数据并完成渲染。

一种云渲染系统、服务器及方法

技术领域

[0001] 本发明涉及GPU虚拟化技术领域，特别涉及一种云渲染系统、服务器及方法。

背景技术

[0002] 云计算已经越来越普及，越来越多的厂商正在考虑将自己的业务转移到云服务提供商的云主机(如虚拟机，容器虚拟化)上。然而目前的云主机无法提供较强的3D渲染能力与GPGPU(General Purpose GPU通用计算图形处理器)计算能力，支持强3D实时渲染需求的应用和高性能计算应用，目前云服务提供商主要依赖于云服务管理平台、虚拟化软件以及硬件对虚拟化技术的支持，将物理资源进行切割、隔离、封装成为云主机，以此为基础对外提供服务。由于GPU(Graphics Processing Unit图形处理器)的复杂性和GPU硬件对虚拟化技术的支持滞后，使得很长时间内云主机没有直接的3D渲染能力。

[0003] 而目前各个公司一般使用Nvidia VGPU架构的方式提供3D云渲染服务，由于vGPU(Virtual GPU虚拟图形处理器)技术由Nvidia厂商独占，因此只有使用Nvidia提供的GRID GPU才能拥有3D渲染能力，然而作为垄断，这种GPU的价格比普通GPU价格高很多；其次，Nvidia VGPU架构的渲染性能较低，例如GRID K1在进行Unigine Heaven Benchmark 4.0测试时，其平均渲染帧率仅为8.5FPS，作为对比Nvidia GTX 970其平均渲染帧率为95.4，性能上相差一个数量级，该架构仅仅只能满足CAD等对3D渲染要求较低的业务需要，只有当虚拟设备能够直接访问到独立的GPU时，才能使GPU发挥出更好的渲染能力。

[0004] 专利CN201010612078.0公开了一种通用图形处理器虚拟化的实现方法、系统及装置，该专利文件公开的方法实现了不依赖Nvidia VGPU架构，使得多个虚拟设备能够访问GPU硬件的方法，通过将虚拟机V1访问的GPU地址配置真实的物理GPU地址，并使该虚拟机与其他多个虚拟机V2间共享同一内存的方式.其他虚拟机V2接收到请求后将信息存储到共享内存，V1读取共享内存信息，通过物理GPU进行数据处理，完成后将结果发送到共享内存中，并由发送该请求的虚拟机读取计算结果。该方法实现多虚拟机间接与GPU硬件进行通信。然而该方法中大多数虚拟机并未直接与GPU进行通信，而是将渲染请求发送给另一台预定的虚拟机，让其代为完成渲染任务，因此计算效率、渲染能力也不会很高。

[0005] 综上所述，现有的云渲染技术均未实现虚拟机直接访问硬件GPU进行渲染，且现有的云渲染设备价格昂贵、3D渲染性能较差。

发明内容

[0006] 为了解决这些潜在问题，本发明的目的在于克服现有技术中所存在的上述不足，提供一种能够使虚拟机直接访问硬件GPU，且价格便宜、渲染性能高的云渲染系统、服务器及方法。

[0007] 为了实现上述发明目的，本发明采用的技术方案是：

[0008] 一种云渲染系统，包括宿主机及多个GPU，所述宿主机设置有多个虚拟机，每个所述虚拟机都配置有对应的一个GPU驱动；

[0009] 所述云渲染系统还包括:MMU,耦合至每个所述GPU驱动及每个所述GPU、耦合虚拟机内存与宿主机内存,其被配置为当任一个虚拟机请求访问GPU时,向该虚拟机的GPU驱动分配一个GPU地址,所述GPU地址用于访问该GPU;当任一个虚拟机请求访问虚拟机内存时,向该虚拟机分配对应的宿主机内存地址;

[0010] IOMMU,耦合至每个所述GPU及虚拟机内存,其被配置为当任一个GPU请求访问虚拟机内存时,向该GPU分配对应的宿主机内存地址。

[0011] 进一步地,设置内存地址空间,并将所述宿主机内存映射到内存地址空间,所述内存地址空间用于存储宿主机内存对应的地址,所述虚拟机通过访问内存地址空间中的地址来访问对应的宿主机内存。

[0012] 进一步地,所述IOMMU还用于将所述内存地址空间存储的不连续的内存段映射为连续的内存段,以便GPU能够通过DMA技术进行数据读写。

[0013] 进一步地,设置GPU地址空间,并将所述GPU映射到GPU地址空间,所述GPU地址空间用于存储GPU控制寄存器对应的地址,所述GPU驱动通过所述GPU地址空间来访问对应的GPU控制寄存器。

[0014] 进一步地,当任一个所述虚拟机启动时,该虚拟机与一个GPU通过MMU和/或IOMMU进行绑定,且在绑定期间,该已被绑定的GPU不能再与其他虚拟机进行绑定。

[0015] 本发明同时提供一种云渲染服务器,包括本发明的云渲染系统,还包括云服务管理平台,用于对所述系统的运行状态进行监控和管理,对使用所述系统的用户进行管理。

[0016] 本发明还提供一种云渲染方法,包括以下步骤:

[0017] S1、虚拟机接收渲染请求,并将渲染请求发送到虚拟机的GPU驱动,接收渲染数据,并将渲染数据写入内存;

[0018] S2、虚拟机的GPU驱动访问GPU控制寄存器,并将渲染请求信息写入GPU控制寄存器;

[0019] S3、GPU根据所述渲染请求信息访问对应的虚拟机内存,并对所述对应的虚拟机内存中的渲染数据进行处理。

[0020] 进一步地,所述虚拟机、所述GPU均有多个,且所述虚拟机与所述GPU一一对应,当有多个渲染请求时,每个所述虚拟机分别对应处理所述多个渲染请求中的任一个。

[0021] 进一步地,所述将渲染数据写入内存包括,MMU耦合所述虚拟机内存与宿主机内存,当虚拟机请求访问虚拟机内存时,通过MMU向该虚拟机分配对应的宿主机内存地址,所述渲染数据根据所述宿主机内存地址进行存储。

[0022] 进一步地,所述S2步骤包括:

[0023] S201、将宿主机的GPU物理地址空间段映射到宿主机的虚拟地址空间段;

[0024] S202、利用GPA-HVA转换表将所述虚拟地址空间段映射到虚拟机中的GPU地址空间。

[0025] S203、虚拟机的GPU驱动访问GPU地址空间段,并根据所述地址空间段上的地址信息访问对应的GPU控制寄存器。

[0026] 进一步地,所述S3步骤包括:

[0027] S301、IOMMU耦合所述虚拟机内存与GPU,并将虚拟机使用的不连续的内存地址空间映射为连续的地址空间段;

[0028] S302、GPU根据渲染请求信息对所述连续的地址空间段进行DMA读写，获取渲染所需的数据并完成渲染。

[0029] 与现有技术相比，本发明的有益效果

[0030] 本发明的一种云渲染系统，通过配置MMU与IOMMU使得多个虚拟机都能够独立直接访问GPU，相比现有技术中一般采用的Nvidia VGPU架构，本发明的系统价格低廉、渲染性能高。

附图说明

[0031] 图1所示是本发明的一种云渲染系统模块框图。

[0032] 图2所示是本发明一个具体实施例的云渲染系统内部模块框图。

[0033] 图3所示是本发明的一种云渲染方法流程图。

[0034] 图4所示是实现本发明方法的内部原理图。

具体实施方式

[0035] 下面结合具体实施方式对本发明作进一步的详细描述。但不应将此理解为本发明上述主题的范围仅限于以下的实施例，凡基于本发明内容所实现的技术均属于本发明的范围。

[0036] 图1所示是本发明的一种云渲染系统模块框图，包括宿主机及多个GPU，所述宿主机设置有多个虚拟机，每个所述虚拟机都配置有对应的一个GPU驱动；

[0037] 所述云渲染系统还包括：MMU(Memory Management Unit内存管理单元)，耦合至每个所述GPU驱动及每个所述GPU、耦合虚拟机内存与宿主机内存，其被配置为当任一个虚拟机请求访问GPU时，向该虚拟机的GPU驱动分配一个GPU地址，所述GPU地址用于访问该GPU；当任一个虚拟机请求访问虚拟机内存时，向该虚拟机分配对应的宿主机内存地址；

[0038] IOMMU(Input/Output Memory Management Unit输入/输出内存管理单元)，耦合至每个所述GPU及虚拟机内存，其被配置为当任一个GPU请求访问虚拟机内存时，向该GPU分配对应的宿主机内存地址。

[0039] 本发明的宿主机表示为能够创建虚拟机的物理服务器实体，且运行在该宿主机上的虚拟机能够分享物理服务器的资源，就本发明而言，本发明的物理服务器实体上设置有多个虚拟机，即可称为宿主机。

[0040] 由于虚拟机在宿主机内运行时，被宿主机作为一个普通的进程，因此虚拟机对于物理内存与物理GPU的访问时需要进行地址转换，本发明利用MMU技术，使得虚拟机在访问内存时，通过MMU技术将其访问地址转换为对应的物理地址，由此，使得虚拟机访问内存或GPU时，都跟物理机的访问方式相同，在GPU处理渲染请求时需要访问内存中的渲染数据，通过IOMMU使得GPU能够直接访问内存中的数据，因此在进行数据处理时，数据处理方式也跟物理机单独处理方式相同，这相当于将虚拟机作为一个独立的主机进行渲染操作，其渲染效率、渲染性能也就与真正的独立主机无异。

[0041] 采用本发明的方案进行云渲染，一个宿主机上运行有多个虚拟机进程，当有多个不同的渲染任务时，每个虚拟机和与该虚拟机对应的GPU都能够独立、同时处理这些任务。例如当虚拟机1与GPU1协同处理一个渲染任务时，宿主机收到了另一项渲染请求，此时虚拟

机2与GPU2根据本发明的方案来处理第二个渲染任务，依次类推，在一个宿主机上运行的多个虚拟机与该多个虚拟机对应的多个GPU能够同时处理多个渲染请求，相比现有的技术方案，本发明的渲染能力更强、渲染效率更高。

[0042] 设置内存地址空间，并将所述宿主机内存映射到内存地址空间，所述内存地址空间用于存储宿主机内存对应的地址，所述虚拟机通过访问内存地址空间中的地址来访问对应的宿主机内存。

[0043] 本发明中，通过设置内存地址空间进行地址映射，方便MMU对内存进行管理，当有多个虚拟机同时读取内存时，根据内存地址空间对应的设置有序的进行内存访问，提高了渲染效率。

[0044] 所述IOMMU还用于将所述内存地址空间存储的不连续的内存段映射为连续的内存段，以便GPU能够通过DMA(Directional Memory Access直接内存访问)技术进行数据读写。

[0045] 由于虚拟机使用的内存区域一般是不连续的空间，对于这种非连续空间，传统的内存数据读取模式是通过CPU协助控制GPU读取对应地址段的内容，这种方式需要额外调度CPU资源，不仅产生资源浪费，也降低了GPU处理效率，而通过IOMMU技术，可以将这种非连续的内存区域映射为连续的地址空间段，这样，GPU就可以直接从内存中读取渲染数据进行渲染，进一步提高了渲染效率。

[0046] 设置GPU地址空间，并将所述GPU映射到GPU地址空间，所述GPU地址空间用于存储GPU控制寄存器对应的地址，所述GPU驱动通过所述GPU地址空间来访问对应的GPU控制寄存器。

[0047] 具体的，图2所示是本发明一个具体实施例的云渲染系统内部模块框图，虚拟机V1与虚拟机VN架构相同，虚拟机内存在访问内存时先访问位于地址空间的内存地址空间，内存地址空间提供与该虚拟机内存地址对应的宿主机内存，继而进行数据的读写操作；虚拟机的GPU驱动在访问GPU时先通过GPU地址空间得到映射设于GPU地址空间的GPU控制寄存器地址，通过GPU控制寄存器，继而控制GPU进行渲染。

[0048] 当任一个所述虚拟机启动时，该虚拟机与一个GPU通过MMU和/或IOMMU进行绑定，且在绑定期间，该已被绑定的GPU不能再与其他虚拟机进行绑定。

[0049] 如前所述，虚拟机在宿主机内运行时，被宿主机作为一个普通的进程，当虚拟机启动时，就会向该虚拟机分配一个未被其他虚拟机使用的GPU进行绑定，在绑定的同时建立起虚拟机与GPU间的地址映射关系，而当有多个虚拟机启动并在同一时刻运行时，每个虚拟机都对应独立的绑定了一个GPU，每个虚拟机的内存都被映射到地址空间的不同区域，每个GPU也都被映射到GPU地址空间，因此，在之后的信息交互时，每个虚拟机都能同时独立的处理渲染请求，直接根据他们对应的映射关系实现信息传递与信息存储。

[0050] 本发明在当虚拟机启动时进行绑定操作，在实际应用中，可以在虚拟机创建时进行绑定，也可以在虚拟机启动时进行绑定，可以主动进行绑定，也可以被动的在接收到某个命令时建立绑定，绑定时间可依照实际情况而定，或者在其他时间进行建立。

[0051] 本发明同时提供一种云渲染服务器，包括本发明的云渲染系统，还包括云服务管理平台，用于对所述系统的运行状态进行监控和管理，对使用所述系统的用户进行管理。

[0052] 作为本发明的一个具体的实施方式，在虚拟机创建时，云服务管理平台将虚拟机与GPU进行了绑定，并且对MMU、IOMMU也进行了映射关系的配置，使虚拟机能够直接访问到

物理内存与物理设备，当虚拟机与某个GPU绑定后，该GPU被与之绑定的虚拟机独占使用，直到该虚拟机被销毁或GPU资源被释放，才可重新绑定该GPU。

[0053] 所述云服务管理平台使用现有的平台系统，如openstack，Amazon Web Service，阿里云等，在此不再赘述。

[0054] 图3所示是本发明的一种云渲染方法流程图，包括以下步骤：

[0055] S1、虚拟机接收渲染请求，并将渲染请求发送到虚拟机的GPU驱动，接收渲染数据，并将渲染数据写入内存；

[0056] 虚拟机在进行信息处理时，是依照物理机的处理流程进行处理，每个虚拟机都设置有GPU驱动程序，用于驱动GPU工作，将渲染数据写入内存，实际上是通过MMU进行内存地址空间映射后，数据写入到了宿主机的物理内存中，虚拟机所使用的资源均是通过MMU、IOMMU映射方式访问的宿主机物理资源。

[0057] S2、虚拟机的GPU驱动访问GPU控制寄存器，并将渲染请求信息写入GPU控制寄存器；

[0058] S3、GPU根据所述渲染请求信息访问对应的虚拟机内存，并对所述对应的虚拟机内存中的渲染数据进行处理。

[0059] 在一个具体实施方式中，所述虚拟机、所述GPU均有多个，且所述虚拟机与所述GPU一一对应，当有多个渲染请求时，每个所述虚拟机分别对应处理所述多个渲染请求中的任一个。

[0060] 所述将渲染数据写入内存包括，MMU耦合所述虚拟机内存与宿主机内存，当虚拟机请求访问虚拟机内存时，通过MMU向该虚拟机分配对应的宿主机内存地址，所述渲染数据根据所述宿主机内存地址进行存储。

[0061] 在一个具体的实施方式中，MMU的这种耦合关系在虚拟机创建时就已经建立了，当渲染数据写入内存时，MMU直接将宿主机虚拟机地址(HVA，Host Virtual Address)转换为宿主机物理地址(HPA)，使渲染数据写入到宿主机内存。

[0062] 所述S2步骤包括：

[0063] S201、将宿主机的GPU物理地址空间段映射到宿主机的虚拟地址空间段；

[0064] S202、利用GPA-HVA转换表将所述虚拟地址空间段映射到虚拟机中的GPU地址空间。

[0065] S203、虚拟机的GPU驱动访问GPU地址空间段，并根据所述地址空间段上的地址信息访问对应的GPU控制寄存器。

[0066] 所述S3步骤包括：

[0067] S301、IOMMU耦合所述虚拟机内存与GPU，并将虚拟机使用的不连续的内存地址空间映射为连续的地址空间段；

[0068] S302、GPU根据渲染请求信息对所述连续的地址空间段进行DMA读写，获取渲染所需的数据并完成渲染。

[0069] GPU在进行DMA时，访问的是连续的Address Bus(地址总线)，然而在虚拟机中连续的物理地址(GPA，Guest Physical Address)对应的宿主机物理地址(HPA)实际上并非是连续的，因此需要通过IOMMU将GPU使用的Address Bus映射为连续HPA，继而进行DMA。

[0070] 实施例1：

[0071] 图4给出了实现本发明方法的内部原理图，其中，宿主机上运行的虚拟机软件选择qemu，操作系统为linux，多个独立GPU为Nvidia GTX970，该宿主机上设置有多个虚拟机，在一个虚拟机启动时，云服务管理平台分配一个未被使用的GPU与该虚拟机进行绑定，并且该GPU被该虚拟机独占使用，直到该虚拟机销毁，该GPU不能再被其他虚拟机共享使用，当另一个虚拟机也启动时，云服务管理平台也分配一个未被使用的GPU与该虚拟机进行绑定，来处理另一项渲染任务，实现了在一个宿主机上的多个虚拟机同时处理不同的渲染任务，这保证了虚拟机在协同渲染中数据的高效性与可靠性，相比现有的技术，其计算效率大大提升。在绑定过程中，vfio-pci驱动根据映射关系配置MMU与IOMMU，使得虚拟机的GPU驱动可以直接访问GPU硬件，GPU硬件可以直接访问虚拟机内存，具体的，宿主机将内存映射到HPA Space(Host Physical Address Space物理地址空间)的内存地址区域，对该区域的访问即可实现对内存的访问，将宿主机的GPU的物理资源映射到物理地址空间的GPU地址区域，虚拟机的GPU驱动可以访问该段地址空间，实现对GPU进行控制，并将不连续的虚拟机内存区域映射为连续的PCI地址空间段。按照上述配置建立的云渲染系统，其具体实施步骤如下：

[0072] 步骤一、虚拟机中的3D应用程序将渲染数据写入虚拟机内存中，将渲染请求发送到虚拟机的GTX970驱动程序；

[0073] 步骤二、GTX970驱动访问GPU地址空间段，将渲染请求写入GTX970控制寄存器中；

[0074] 步骤三、GTX970根据控制寄存器的信息，对连续的地址空间进行DMA，获取渲染所需的数据；

[0075] 步骤四、GTX970对渲染数据进行处理，将渲染结果进行存储，完成渲染。上面结合附图对本发明的具体实施方式进行了详细说明，但本发明并不限制于上述实施方式，在不脱离本申请的权利要求的精神和范围情况下，本领域的技术人员可以作出各种修改或改型。

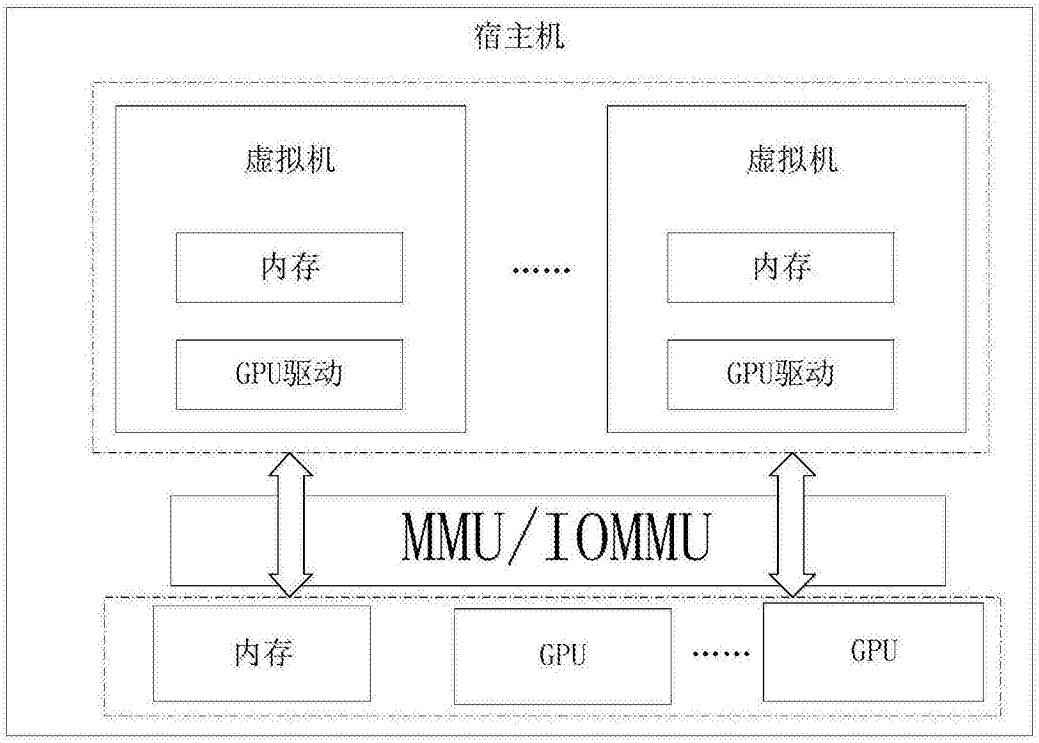

宿主机
虚拟机
内存
GPU驱动
……
虚拟机
内存
GPU驱动
MMU/IOMMU
内存
GPU
……
GPU

图1

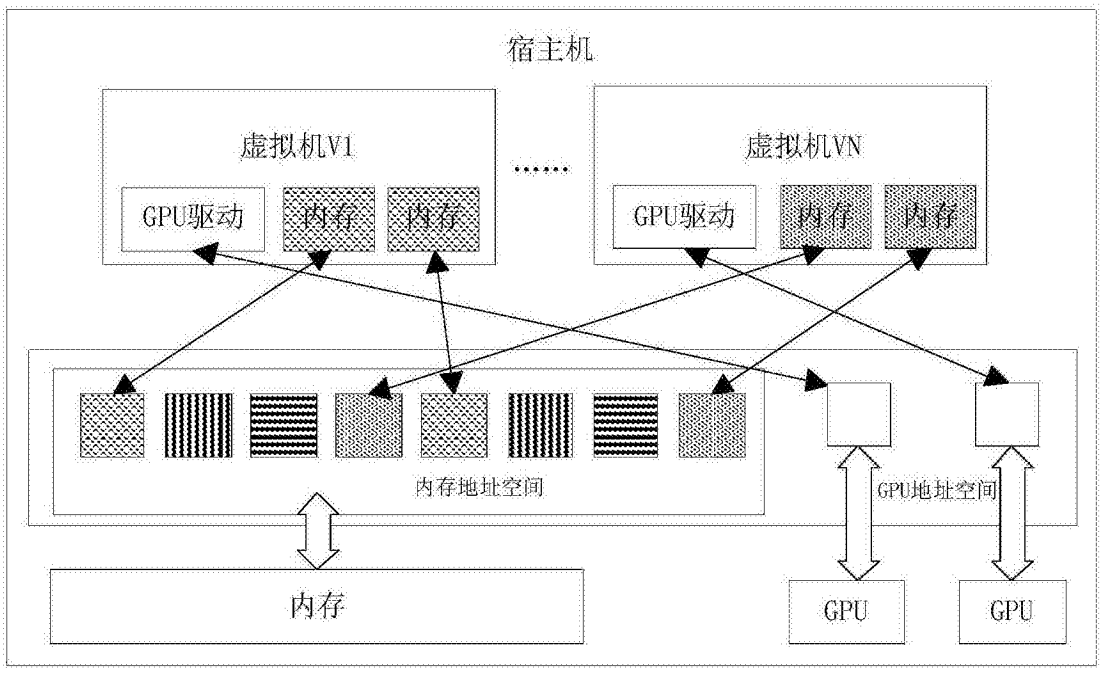

宿主机
虚拟机V1
GPU驱动
内存
内存
……
虚拟机VN
GPU驱动
内存
内存
内存地址空间
GPU地址空间
内存
GPU
GPU

图2

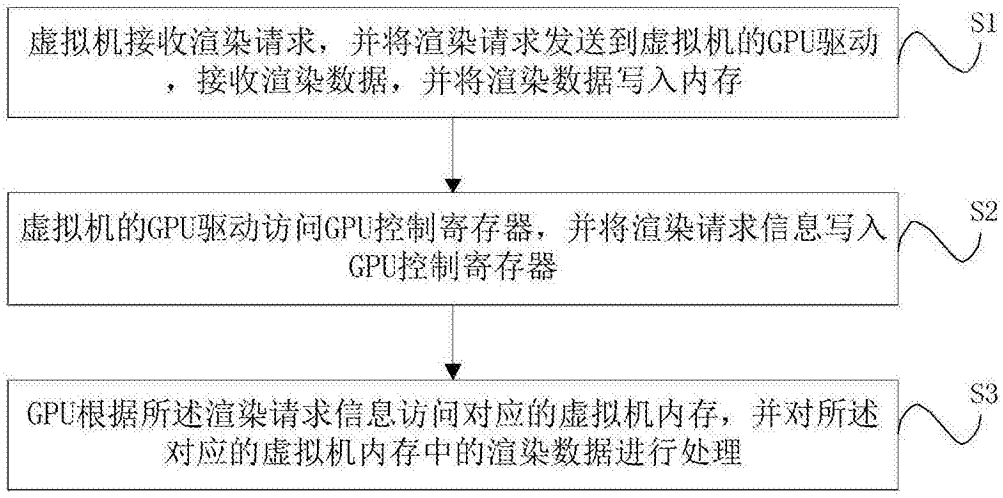

虚拟机接收渲染请求，并将渲染请求发送到虚拟机的GPU驱动，接收渲染数据，并将渲染数据写入内存
S1
虚拟机的GPU驱动访问GPU控制寄存器，并将渲染请求信息写入GPU控制寄存器
S2
GPU根据所述渲染请求信息访问对应的虚拟机内存，并对所述对应的虚拟机内存中的渲染数据进行处理
S3

图3

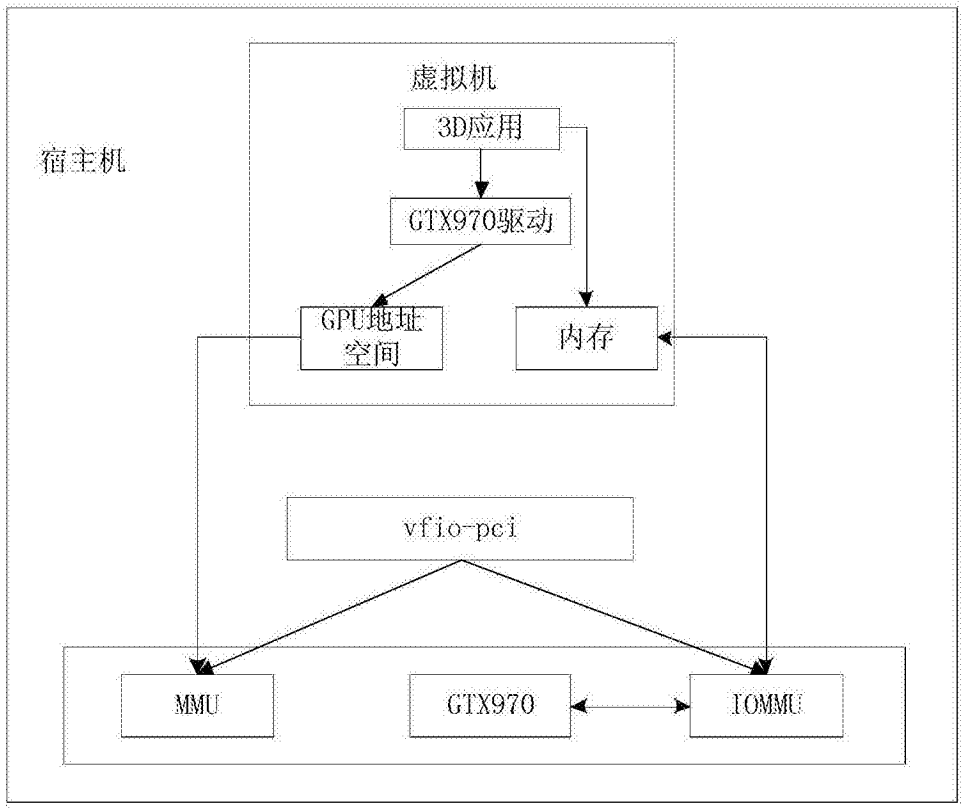

宿主机
虚拟机
3D应用
GTX970驱动
GPU地址空间
内存
vfio-pci
MMU
GTX970
IOMMU

图4