虚拟化组件升级与虚拟化实例状态保存方法、物理机、存储介质及程序产品

技术领域

本公开涉及云计算技术领域，尤其涉及一种虚拟化组件升级与虚拟化实例状态保存方法、物理机、存储介质及程序产品。

背景技术

虚拟化技术是云计算的关键技术，通过虚拟化技术，可以将物理机的各种资源虚拟化，实现多个 VM（Virtual Machine，虚拟机）在同一台物理机上运行，在互不影响的同时，实现资源共享。

为了实现虚拟化，物理机上部署有虚拟化管理器（Virtual Machine Monitor，VMM）。VMM 可以看作是一款为虚拟化而生的 OS（Operating System，操作系统），负责 CPU（Central Processing Uni，中央处理器）、内存等硬件资源的虚拟化，以及 VM 的创建和管理。

VMM 包括多种与虚拟化相关的组件，例如用于物理资源虚拟化的各种组件，被配置为 VM 创建和管理的组件，这些组件简称为虚拟化组件。实际应用中，出于问题修复或功能升级，需要对虚拟化组件进行升级。

但是，现有技术中的升级方案并不适合虚拟化组件，因此，亟需一种针对虚拟化组件进行升级的方案。

发明内容

本公开的多个方面提供一种虚拟化组件升级与虚拟化实例状态保存方法、物理机、存储介质及程序产品，用以降低虚拟化组件升级对虚拟化实例的影响。

本公开实施例提供一种虚拟化组件升级方法，包括：确定目标虚拟化组件，所述目标虚拟化组件是运行于物理机上的虚拟化管理器中待升级的虚拟化组件；对运行于所述物理机上的虚拟化实例进行状态保存，所述虚拟化实例是所述虚拟化管理器创建的；对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

本公开实施例还提供一种物理机，包括：硬件资源，所述硬件资源上运行有虚拟化管理器、虚拟化实例和升级程序；所述虚拟化管理器包括多个虚拟化组件，被配置为实现所述硬件资源的虚拟化以及所述虚拟化实例的创建和管理；其中，所述升级程序，被配置为确定所述多个虚拟化组件中待升级的目标虚拟化组件；对所述虚拟化实例进行状态保存；对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；以及基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

本公开实施例提供一种虚拟化实例状态保存方法，包括：响应虚拟化实例状态保存

操作，采集虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；所述虚拟化实例具有目标内存空间，所述目标内存空间包括系统内存空间，且所述虚拟化实例产生的第一内存数据以文件方式存储在所述系统内存空间中；在所述虚拟化实例恢复之前，以文件共享的方式，对所述第一内存数据进行原地驻留。

本公开实施例还提供一种物理机，包括硬件资源，所述硬件资源包括存储器和处理器；所述存储器，被配置为存储计算机程序，所述处理器与所述存储器耦合，被配置为执行所述计算机程序，以用于实现以上所述方法中的步骤。

本公开实施例还提供一种存储有计算机程序的计算机可读存储介质，当所述计算机程序被处理器执行时，致使处理器能够实现以上所述方法中的步骤。

本公开实施例还提供一种计算机程序产品，该计算机程序产品包括计算机程序/指令，当计算机程序/指令被处理器执行时，致使处理器能够实现上述方法实施例中的步骤。

在本公开实施例中，提供一种用于虚拟化组件的升级方法，在该方法中，先对虚拟化实例进行状态保存，然后，卸载需要升级的虚拟化组件，并重新安装和加载升级后的虚拟化组件；再对虚拟化实例进行状态恢复。其中，通过卸载虚拟化组件，然后重新安装和加载升级后的虚拟化组件，是对虚拟化组件级别的整体替换，并不会受限于升级复杂度，可以满足各种复杂度的升级需求，适用于对各种虚拟化组件进行升级。另外，在卸载虚拟化组件之前，提前对依赖虚拟化组件运行的虚拟化实例进行状态保存，在完成虚拟化组件升级之后，重新对虚拟化实例进行状态恢复，可以快速完成虚拟化实例的部署，有利于降低虚拟化组件升级过程对虚拟化实例的影响。

附图说明

此处所说明的附图用来提供对本公开的进一步理解，构成本公开的一部分，本公开的示意性实施例及其说明用于解释本公开，并不构成对本公开的不当限定。在附图中：

图 1a 为本公开一示例性实施例提供的一种物理机的结构示意图；

图 1b 本公开一示例性实施例提供的一种采用QEMU-KVM结合实现虚拟化的物理机的结构示意图；

图 2 为本公开一示例性实施例提供的一种虚拟化组件升级方法的流程示意图；

图 3a 为本公开又一示例性实施例提供的一种物理机的结构示意图；

图 3b 为本公开又一示例性实施例提供的另一种物理机的结构示意图；

图 4 为本公开一示例性实施例提供的一种虚拟化实例状态保存方法的流程示意图；

图 5 为本公开又一示例性实施例提供的一种虚拟化组件升级装置的结构示意图；

图 6 为本公开又一示例性实施例提供的一种虚拟化实例状态保存装置的结构示意图；

图 7 为本公开又一示例性实施例提供的另一种物理机的结构示意图。

具体实施方式

为使本公开的目的、技术方案和优点更加清楚，下面将结合本公开具体实施例及相应的

附图对本公开技术方案进行清楚、完整地描述。显然，所描述的实施例仅是本公开一部分实施例，而不是全部的实施例。基于本公开中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本公开保护的范围。

需要说明的是，本公开所涉及的用户信息（包括但不限于用户设备信息、用户个人信息等）和数据（包括但不限于用于分析的数据、存储的数据、展示的数据等），均为经用户授权或者经过各方充分授权的信息和数据，并且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准，并提供有相应的操作入口，供用户选择授权或者拒绝。另外，本公开涉及的各种模型（包括但不限于语言模型或大模型）是符合相关法律和标准规定的。

为了实现虚拟化，物理机上部署有虚拟化管理器，虚拟化管理器包括与虚拟化处理相关的多个组件，简称为虚拟化组件。在实际应用中，出于对虚拟化组件的问题修复或功能升级，都会面临对虚拟化管理器中虚拟化组件进行升级的问题。

在现有技术中，一些内核提供了热补丁（hotfix）机制。Kpatch 作为一种内核提供的热补丁机制，可以在用户无感知的情况下对内核模块进行热修复，在内核模块出现安全漏洞、发生故障（BUG）或者需要功能升级时，理论上都可以采用 Kpatch 这种方式进行升级。对于虚拟化管理器中的各种虚拟化组件，理论上也可以通过这种方式来做热升级。但是，Kpatch 存在一些局限性，它作为热补丁机制，更适合相对简单的修复情况，对于数据结构变化较大、数据对象的名称发生变化或者出现新的数据对象等较为复杂的情况，单独靠打补丁的方式是无法解决的，这也意味着 Kpatch 不适用于较为复杂的升级情况。但是，在实际应用中，虚拟化组件不仅多样化，通常功能也是相对复杂的，其面临的问题修复或功能升级也会比较多样化和复杂，这就需要通过其他的手段来解决虚拟化组件的升级问题。

在本公开实施例中，提供一种用于虚拟化组件的升级方法，在该方法中，先对虚拟化实例进行状态保存，然后，卸载需要升级的虚拟化组件，并重新安装和加载升级后的虚拟化组件；再对虚拟化实例进行状态恢复。其中，通过卸载虚拟化组件，然后重新安装和加载升级后的虚拟化组件，是对虚拟化组件级别的整体替换，因此不存在采用 Kpatch 做热修复类似的局限，也就是说，本方案不会受限于虚拟化组件升级复杂度，可以满足各种复杂程度的升级需求，适用于对各种虚拟化组件进行升级；另外，在卸载虚拟化组件之前，提前对依赖虚拟化组件运行的虚拟化实例进行状态保存，在完成虚拟化组件升级之后，重新对虚拟化实例进行状态恢复，可以快速完成虚拟化实例的部署，有利于降低虚拟化组件升级对虚拟化实例的影响。

进一步，在本公开实施例中，在对虚拟化实例进行状态保存和状态恢复过程中，采用内存数据原地驻留的方式，可以减少内存数据的拷贝操作，降低状态保存的耗时以及状态恢复的耗时，提高虚拟化实例的整体恢复效率，尽量将虚拟化实例从状态保存到恢复的过程做到用户无感知或在用户可容忍的时间范围内，达到热升级效果。

进一步，在本公开实施例中，在对虚拟化实例进行状态保存和状态恢复过程中，若存在多个虚拟化实例，以并发操作方式对多个虚拟化实例进行状态保存和恢复，有利于进一步提高虚拟化实例的恢复效率，尽量达到热升级效果。

进一步，在本公开实施例中，在需要对虚拟化实例使用的目标物理外设设备的驱动程序进行升级的情况下，还可以优化该驱动程序的卸载过程、安装过程、加载过程和/或初始化过程，以提高卸载、安装和/或加载过程的效率，进一步缩短整个升级过程的耗时，提高升级效率，降低虚拟化组件升级对虚拟化实例的影响。

以下结合附图，详细说明本公开各实施例提供的技术方案。

图 1a 为本公开一示例性实施例提供的一种物理机的结构示意图。如图 1 所示，在本公开实施例中，物理机包括硬件资源 11，硬件资源 11 之上运行有虚拟化管理器 12，虚拟化管理器 12 之上运行有一个或多个虚拟化实例，这些虚拟化实例由虚拟化管理器 12 负责创建和维护。

其中，物理机的硬件资源包括 CPU、内存、网卡、GPU（Graphics Processing Unit，图形处理器）等；除此之外，物理机的硬件资源还包括存储器、通信组件、显示器、电源组件、音频组件以及各种外部设备等，在 1a 中未作图示。

虚拟化管理器 12 一方面对物理机上的各种资源进行虚拟化，一方面负责创建和管理虚拟化实例。除此之外，虚拟化管理器 12 为虚拟化实例维护高效、隔离的运行环境；另外，虚拟化管理器 12 还可以作为虚拟化实例的核心任务监控系统，可用于调度任务、负载均衡、向管理员报告虚拟化实例的软硬件故障，并控制虚拟化实例的使用情况等。其中，一个虚拟化管理器 12 可以同时管理一个或多个虚拟化实例，对此不做限定。

如图 1a 所示，虚拟化管理器 12 包括虚拟化模块 121，虚拟化模块 121 一方面对物理机上的各种资源进行虚拟化，一方面负责创建和管理虚拟化实例，至少包括用于虚拟化的组件，例如用于 CPU 虚拟化的组件，被配置为内存虚拟化的组件，被配置为 IO（Input Output，输入输出）虚拟化的组件，被配置为 VM 创建和管理的组件等。

为了便于对与虚拟化相关的组件或模块进行统一描述，将物理机的传统 OS 部分称为宿主机（Host）OS122，并以宿主机 OS 属于虚拟化管理器 12 为例进行说明，也就是说，在本公开实施例中，虚拟化管理器包括虚拟化模块 121 和宿主机 OS122，如图 1a 所示。其中，宿主机 OS 主要负责物理机的设备管理、文件管理、内存管理和进程管理等，包含设备驱动程序、文件系统、内存管理模块以及进程管理模块等。宿主机 OS 中的一些功能也与虚拟化相关，例如在 IO 虚拟化时会涉及物理外部设备（也可称为 IO 设备）的驱动程序，物理外部设备的驱动程序属于宿主机 OS。

为了便于描述，在本公开实施例中，将虚拟化管理器 12 中所有与虚拟化相关的对象统称为虚拟化组件，这些虚拟化组件包括虚拟化模块 121 中的组件，也包括宿主机 OS122 中与虚拟化相关的驱动程序。

在本公开实施例中，并不限定虚拟化管理器 12 的实现方式，例如可以实现为 Xen、OpenVZ、KVM（Kernel-based Virtual Machine，基于内核的虚拟机）或者实现为 KVM 与 QEMU（Quick EMUlator，虚拟操作系统模拟器）结合等。其中，Xen 是一个开放源代码虚拟机化管理器，直接运行在物理机的硬件资源之上的用以替代操作系统的软件层，允许单个物理机上运行多个操作系统实例，被配置为提供高性能的虚拟化，且不依赖于特殊硬件支持；OpenVZ

是基于 Linux（一种开放源码的类 UNIX 操作系统）内核的操作系统级虚拟化技术（或称为容器化技术），允许物理机运行多个操作系统，提供了更轻量级的虚拟化方案；KVM 是一种基于 Linux 的开源虚拟化技术，具体地，KVM 可将 Linux 转变为虚拟化管理器，为虚拟机提供 CPU 虚拟化和内存管理等能力，KVM 可以利用硬件虚拟化技术来提高虚拟机性能；其中，KVM 和 QEMU 通常结合使用，QEMU 是一款基于用户态的虚拟化软件，KVM 依赖于 QEMU 来模拟 IO 设备，例如虚拟网卡、虚拟磁盘控制器等等，KVM 和 QEMU 结合使用（简称为 QEMU-KVM）可以提供完整的虚拟化解决方案。

其中，虚拟化管理器采用不同实现方式时，其包含的虚拟化组件会略有差异。在本实施例中，以虚拟化管理器采用 QEMU 与 KVM 结合实现为例，对虚拟化管理器包含的虚拟化组件进行举例说明，但不代表本公开实施例中虚拟化管理器只能采用 QEMU-KVM 结合实现。

如图 1b 所示，为本公开一示例性实施例提供的一种采用 QEMU-KVM 结合实现虚拟化的物理机，该物理机的内部软硬件架构自上而下包括：应用层、内核层和硬件层。其中，硬件层包括物理机的各种硬件资源 11。如图 1b 所示，物理机的硬件资源 11，包括但不限于 CPU111、GPU112、RAM（Random-access memory，内存）113 和磁盘 114 等等。除此之外，物理机的硬件资源还包括网卡模块、通信组件、显示器、电源组件、音频组件以及各种外部设备等，在 1b 中未作图示。在硬件资源 11 上运行有内核层，宿主机 OS122 位于内核层，在宿主机 OS122 中包含硬件层中各种硬件的驱动程序，例如包括 NIC（Network Interface Card，网络接口卡）、GPU（Graphics Processing Unit）等物理外部设备（简称为外设）的驱动程序。在本公开实施例中，物理外部设备是相对于宿主机的 CPU111 而言，所有通过各种类型的总线与 CPU111 互联的硬件模块都可作为本公开实施例中的物理外部设备，也可以称为物理 IO 设备；其中，与 CPU111 互联的总线可以是高速外设互联总线标准（Peripheral Component Interconnect Express，PCIe）总线或串行总线等，对此不做限定，具体视物理外部设备而定。

在本实施例中，内核层还包括 KVM 驱动、VFIO 驱动、VFIO_PCI 驱动、VFIO_MDEV 驱动等各种内核态虚拟化组件；应用层包括：QEMU。其中，QEMU 属于用户态虚拟化组件，可配合内核态的虚拟化组件完成对物理机的各种资源的虚拟化以及虚拟化实例的创建和管理。虚拟化模块 121 则横跨内核层和应用层，如图 1b 所示，虚拟化模块 121 包括位于内核层的 VFIO 驱动、VFIO_PCI 驱动、VFIO_MDEV 驱动和 KVM 驱动等内核态虚拟化组件，以及位于应用层的 QEMU。其中，应用层还包括运行在物理机上的各种虚拟化实例，虚拟化实例可以是云主机、弹性计算服务（Elastic Compute Service，ECS）、虚拟机或容器等，这些虚拟化实例可由虚拟化模块 121 负责创建。

其中，KVM 驱动是 Linux 内核的一部分，作为一个内核模快运行，主要作用是负责虚拟化实例的管理，实现 CPU 的虚拟化、内存的虚拟化等；QEMU 作为一个用户空间组件，主要负责模拟各种硬件设备，以及为虚拟化实例提供 I/O 设备模型和访问外设的途径。其中，VFIO 驱动的作用是将物理外设设备安全地映射到用户空间，使得用户空间进程可以直接使用 VFIO 驱动访问物理外设设备；VFIO_PCI 驱动主要负责将 PCI 设备直接透传到虚拟化实例中，使得虚拟机能够直接访问 PCI 设备的配置空间；VFIO_MDEV 驱动主要作用是允许硬件资源在

不同的虚拟机之间安全地共享，同时保证隔离性。在此说明，图 1b 中示出的各内核态虚拟化组件仅为示例，并不表示仅包含这些组件，也不表示必须包含这些组件。

进一步，在使用 VFIO 设备直通时，虚拟化实例能直接访问物理外设设备，但每个物理外设设备只能被一个虚拟化实例独占。为了解决物理外设设备的独占问题，可以采用 SR-IOV（Single Root I/O Virtualization，单根 I/O 虚拟化）直通技术支持将一个 PF（Physical Function，物理功能）虚拟出多个 VF（Virtual Function，虚拟功能），并将多个 VF 直通给不同虚拟化实例使用。也就是说，在虚拟化管理器中存在相关虚拟化组件会对物理外部设备进行虚拟化处理从而得到各个虚拟外部设备，虚拟外部设备也具有驱动程序，为了对虚拟外部设备的驱动程序与物理外部设备的驱动程序进行区分，将宿主机 OS122 中的驱动程序称为 PF 驱动，例如包括 GPU PF 驱动和 NIC PF 驱动。当然，这里示出的物理外设设备的 PF 驱动也仅为示例，并不代表仅包含这些 PF 驱动程序，也不代表必须包含这些 PF 驱动程序。

在图 1b 中示出的各种 PF 驱动程序、KVM 驱动、VFIO 驱动、VFIO_PCI 驱动、VFIO_MDEV 驱动以及 QEMU 等都都是本公开实施例中虚拟化组件的示例。对于上述物理机，在实际运行过程中，部分虚拟化组件可能发生故障，或者随着虚拟化技术的更新迭代需要对部分虚拟化组件进行升级，也就是说，出于问题修复或功能升级，面临对虚拟化管理器中虚拟化组件进行升级的问题。

为了实现对虚拟化组件的升级，本公开实施例提供的一种虚拟化组件升级方法，如图 2 所示，该方法包括：

S201：确定目标虚拟化组件，目标虚拟化组件是运行于物理机上的虚拟化管理器中待升级的虚拟化组件；

S202：对运行于物理机上的虚拟化实例进行状态保存，虚拟化实例是所述虚拟化管理器创建的；

S203：对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；

S204：基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复。

在本实施例中，目标虚拟化组件可以是虚拟化管理器中的一个或多个虚拟化组件，结合图 1b 所示架构，其中，GPU PF 驱动、NIC PF 驱动、QMEU、KVM 驱动或 VFIO 驱动等虚拟化组件可以作为本实施例中的目标虚拟化组件，本公开对此不作限制。需要说明的是，目标虚拟化组件可以包括虚拟化管理器中的一个或多个虚拟化组件；在对目标虚拟化组件进行升级时，若虚拟化管理器中存在不需要升级的其它虚拟化组件且其它虚拟化组件的运行不依赖于目标虚拟化组件，则不需要升级的其它虚拟化组件可以正常运行，可不受该升级过程的影响。

在一可选实施例中，为了确定目标虚拟化组件，物理机的硬件资源上还部署有：位于用户态的升级管理组件，该组件可以接收虚拟化组件升级的配置文件，配置文件中包括待升级的虚拟化组件的标识信息，根据配置文件确定目标虚拟化组件。或者，也可以展示配置虚拟化升级的交互界面，交互界面上展示各个虚拟化组件的标识信息，例如虚拟化组件的名称或

图标；响应用户在各个虚拟化组件的标识信息中的选择操作，确定被选择的标识信息对应的虚拟化组件作为目标虚拟化组件。或者，可以通过命令窗口向升级管理组件发送升级命令，升级命令中包含需要升级的虚拟化组件的标识信息，根据该升级命令可以确定目标虚拟化组件。

在本实施例中，对目标虚拟化组件进行升级会影响虚拟化实例的正常运行，为此，在对目标虚拟化组件进行升级之前，先保存虚拟化实例的状态，以供在升级完成后，根据保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复。其中，在保存虚拟化实例的状态的情况下，可以暂停虚拟化实例，以避免对目标虚拟化组件进行升级给虚拟化实例造成影响。其中，需要进行状态保存并暂停的虚拟化实例可以是物理机当时运行的全部虚拟化实例，当然对于部分脱离目标虚拟化组件能够正常运行的虚拟化实例也可以继续运行，而不用进行状态保存和暂停。考虑到虚拟化实例的运行通常是离不开各个虚拟化组件的，因为较为简单的实现方式是：在需要对目标虚拟化组件进行升级时，统一对当时物理机上运行的各个虚拟化实例均进行状态保存和暂停，并在完成目标虚拟化组件的升级之后，统一对各个虚拟化实例进行状态恢复，以便于各个虚拟化实例继续运行。为了便于虚拟化实例在状态恢复后，能够成功依赖升级后的目标虚拟化组件进行运行。在升级目标虚拟化组件时，可以做到目标虚拟化组件在接口方面在升级前后保持兼容，这些接口不仅包括目标虚拟化组件对虚拟化实例提供服务的接口（或称为面向虚拟化实例或虚拟化实例中运行的应用提供的访问接口），还包括该目标虚拟化组件调用其他虚拟化组件或非虚拟化内核组件所需的接口。在接口方面升级前后保持兼容主要涉及接口名称、接口的入参和出参等保持不变，既不影响虚拟化实例对升级后目标虚拟化组件的调用，也不影响升级后目标虚拟化组件对原本依赖的其他组件或模块的调用。

其中，对运行于物理机上的虚拟化实例进行状态保存主要包括：对虚拟化实例的状态数据和内存数据进行保存。其中，虚拟化实例的状态数据是指虚拟化实例运行所依赖的一些数据以及对虚拟化实例的运行状态具有影响的一些数据，例如当前分配到的 VCPU（Virtual CPU，虚拟 CPU）的状态数据、当前访问的虚拟设备状态、虚拟化实例的唯一性标识，例如 UUID（Universally Unique Identifier，通用唯一识别码）以及当前访问的文件句柄等等，具体会因为虚拟化实例的功能不同而有所不同。虚拟化实例的内存数据是指在虚拟化实例运行过程中产生并被保存至其分配到的内存空间中的数据，简称为内存数据，也会因为虚拟化实例的功能不同而有所不同。进一步，在虚拟化实例挂载持久化存储盘（例如云盘）的情况下，还涉及对持久化存储盘的保存操作，这个保存操作主要是指保存持久化存储盘与虚拟化实例的挂载关系。

其中，对升级前的目标虚拟化组件进行卸载操作，是指停止目标虚拟化组件的相关服务，并从物理机上删除目标虚拟化组件对应的各种资源文件，以及清理相关配置文件和运行时数据，其中，删除目标虚拟化组件对应的各种资源文件包括：若目标虚拟化组件是用户态虚拟化组件，可以采用类似传统应用软件的卸载方式卸载其安装包，若目标虚拟化组件是内核态虚拟化组件，可以使用一些卸载命令直接从操作系统中删除该内核态虚拟化组件。重新安装

升级后的目标虚拟化组件是指下载新版本的安装包（或内核模块），若是用户态虚拟化组件，可以采用类似传统应用软件的安装方式运行安装包进行安装，若是内核态虚拟化组件，可以使用一些安装命令或管理员权限等在操作系统中添加升级后的内核态虚拟化组件。进一步，对于用户态虚拟化组件来说，执行安装包即可完成安装，对于内核态虚拟化组件来说，则需进行编译并添加到内核中；加载升级后的目标虚拟化组件是指启动新安装的目标虚拟化组件，并进行初始化的过程；对用户态虚拟化组件来说，是指启动新安装软件服务，并完成相应初始化过程；对于内核态虚拟化组件来说则是指加载新的目标虚拟化组件到内核，并完成相应初始化过程。

相应地，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复，主要是指根据虚拟化实例对应的资源文件，重新创建虚拟化实例，并将虚拟化实例恢复到保存时刻的状态，以及将保存的虚拟化实例的内存数据重新提供给恢复后的虚拟化实例，以使虚拟化实例能够继续运行。

在本公开实施例中，并不限定虚拟化管理器的内部实现结构。凡是能够创建和管理虚拟化实例的结构均适用于本公开实施例。可选地，虚拟化管理器包括位于用户态的第一虚拟化组件，可以是但不限于：QEMU 或者是位于用户态的虚拟化控制组件等，第一虚拟化组件位于用户态，便于进行交互和调用，且不涉及更多的安全问题。基于此，对运行于物理机上的虚拟化实例进行状态保存，可以调用第一虚拟化组件，对虚拟化实例进行状态保存，第一虚拟化组件能够获取虚拟化实例的各种状态信息且具有权限访问和管理虚拟化实例的内存数据。相应地，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复，可以在目标虚拟化组件不包含第一虚拟化组件的情况下，调用第一虚拟化组件，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复；在目标虚拟化组件包含第一虚拟化组件的情况下，可以在第一虚拟化组件重新安装和加载后，调用升级后的第一虚拟化组件，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复。

在此说明，通过虚拟化管理器包括位于用户态的第一虚拟化组件对虚拟化实例进行状态保存和恢复，仅为一种示例，并不限于此。或者，也可以通过执行本公开实施例提供的升级方法的升级程序对虚拟化实例进行状态保存和恢复，只需该升级程序获取对虚拟化实例的各种状态信息以及内存数据的管理权限即可，该管理权限可以向虚拟化管理器中的 QEMU 或 KVM 驱动进行注册申请即可。无论是本公开实施例提供的升级程序还是第一虚拟化模块（如 QEMU）对虚拟化实例进行状态保存和恢复，进行状态保存和恢复的详细实施方式均可参见本公开下述实施例中的描述。

在本公开实施例中，对虚拟化实例进行状态保存，然后，卸载需要升级的目标虚拟化组件，并重新安装和加载升级后的目标虚拟化组件，是在驱动模块级别的整体替换，适用于各种复杂度的升级需求，不存在采用 Kpatch 做热修复类似的局限，能够对虚拟化组件出现的各种问题或者新开发的各种功能进行升级，不用担心新功能无法应用于存量的虚拟化实例的问题。

进一步，在本公开实施例中，在对目标虚拟化组件升级时，希望能够做到对目标虚拟化

组件的热升级，即希望在用户无感知到其承载于虚拟化实例中的应用被中断或者中断时间或延迟在可容忍范围内的情况下，完成目标虚拟化组件的升级，使得线上问题修复以及新功能新特性的上线可以在不通知用户的情况下进行，大幅降低了安全漏洞（BUG）的修复的周期，同时也降低了与用户沟通的成本。

为了实现上述目的，需要从保存虚拟化实例的状态数据和内存数据，到根据保存的状态数据和内存数据进行虚拟化实例恢复的整个过程，尽量快速完成，耗时尽量短，短到用户基本无感知或者在用户可忍受的响应延迟内。 如何做到将上述过程的耗时尽可能地缩短，在本公开实施例采用以下至少一种技术手段：

（1）对保存虚拟化实例的状态数据和内存数据的过程进行改进：通过分析保存虚拟化实例状态过程中比较耗时的操作，确定比较耗时的操作是内存数据的保存。在保存过程中，为了对内存数据进行保存，需要拷贝内存数据到另一内存或者持久化存储空间（磁盘文件）中，在内存数据量较大的情况下这个操作是比较耗时的。在本公开实施例中，采用内存数据驻留原地的技术，确保内存数据所在内存空间在目标虚拟化组件升级过程中不被注销或不被回收且内存空间不被修改。另外，在虚拟化实例恢复过程中，能够确保将该内存空间重新分配给恢复后的虚拟化实例，完成内存数据的恢复。这样，整个过程中，无需对内存数据进行任何拷贝操作，可以节约时间，提高目标虚拟化组件升级效率。

（2）对需要升级的目标虚拟化组件的卸载、安装和/或加载过程进行优化，以减少因目标虚拟化组件升级导致虚拟化实例停机时间，缩短整个过程的耗时，尽量做到热升级的效果。

需要说明的是，无论目标虚拟化组件是虚拟化模块中的组件还是宿主机 OS 中的 PF 驱动，只要存在潜在优化空间的，都可以进行优化。优化过程可以仅包含卸载过程、安装过程或加载过程，也可以包含任意两个过程或三个过程，对此不做限定，具体视目标虚拟化组件而定。

进一步可选地，在需要升级的目标虚拟化组件包括外设驱动（即宿主机 OS 中的 PF 驱动）的情况下，对外设驱动的卸载、安装、加载过程以及初始化过程中的至少一个过程进行优化，具体根据 PF 驱动对应的物理外设设备而定，不同物理外设设备对应驱动程序的卸载、安装、加载和初始化过程的优化空间可能不同，而且与硬件厂商也有关系，不同硬件厂商提供的硬件驱动的卸载、安装、加载和初始化过程也会有所不同。本实施例对具体的优化过程不做限定，凡是相对传统卸载、安装、加载或初始化过程能够缩短时间的优化方式均适用于本公开实施例。

（3）以并发方式对虚拟化实例进行状态保存和恢复：

在对运行于物理机上的虚拟化实例进行状态保存时，在虚拟化实例为多个的情况下，以并发操作方式对多个虚拟化实例进行状态保存，从而缩短状态保存的时间。相应地，在对运行于物理机上的虚拟化实例进行状态恢复时，在虚拟化实例为多个的情况下，以并发操作方式对多个虚拟化实例进行状态恢复。

在本公开实施例中，并不限定采用内存数据驻留原地的技术手段，可以结合内存数据所在内存空间的不同，采用相应的实施方式。

在一可选实施例中，在创建虚拟化实例时，可以为虚拟化实例分配系统内存空间。系统

内存空间是指宿主机上的物理内存，例如 RAM（Random Access Memory，随机存取存储器）。针对系统内存空间中的内存数据，为了实现内存数据的原地驻留，在虚拟化实例产生内存数据时，可以以文件方式将这些内存数据写入分配给虚拟化实例的系统内存空间中，而不是直接将产生的数据写入系统内存空间中；这样，在对虚拟化实例进行状态保存过程中，可以以文件共享的方式，实现对内存数据的原地驻留，为内存数据驻留系统内存空间提供了基础。

结合图 1b 所示的系统架构，对上述分配系统内存空间的详细过程、虚拟化实例状态保存过程以及虚拟化实例状态恢复过程做详细说明。

其中，对运行于物理机上的虚拟化实例进行状态保存，包括：采集虚拟化实例的状态数据，并将状态数据保存至指定文件中，可选地，指定文件可以是内存文件，也可以是持久化文件（例如磁盘文件）；其中，采集状态数据涉及捕获和存储足够的信息，以便后续可以从相同点对虚拟化实例进行状态恢复。可选地，在同一个物理机上，每个虚拟化实例具有唯一性标识，根据该唯一性标识可以采集与虚拟化实例对应的状态数据，包括但不限于 CPU、内存、网络设备以及存储设备等各种资源的配置信息。在一示例中，基于虚拟化管理器创建的 VM 实例，在同一个物理机上，每个 VM 实例被赋予标识信息，例如虚拟机名称（Name）、UUID 和 ID（Identifier，身份标识），这些标识信息可以用于管理、识别和操作虚拟机的状态，其中 UUID 作为唯一性标识，通过 UUID 可以获取对应的 VM 的状态数据。

其中，对运行于物理机上的虚拟化实例进行状态保存，还包括：对虚拟化实例的目标内存空间中的目标内存数据进行原地驻留。

在一可选实施例中，目标内存空间包括系统内存空间，目标内存数据包括以文件方式存储于系统内存空间中的第一内存数据，这里用于存储第一内存数据的文件可以是基于内存映射技术在系统内存空间中创建的内存映射文件，也可以是通过内存文件系统在系统内存空间中创建的内存文件，对此不做限定；相应地，对虚拟化实例的目标内存空间中的目标内存数据进行原地驻留，包括：以文件共享的方式，对第一内存数据进行原地驻留，这里的文件共享是指虚拟化实例在恢复前后对应的新旧进程共享同一内存映射文件或共享同一内存文件。其中，对虚拟化实例进行状态保存后，还会涉及虚拟化实例对应的旧进程的释放，以暂停虚拟化实例的运行；对虚拟化实例进行状态恢复，还会涉及虚拟化实例对应的新进程的创建，以恢复虚拟化实例。具体地，通过将内存映射文件或内存文件的访问路径分配或挂载到新进程上，即可实现新旧进程之间的文件共享。

在本实施例中，以文件共享的方式，对第一内存数据进行原地驻留，可以采用内存映射方式进行内存数据的存储和驻留，也可以通过内存文件方式进行内存数据的存储和驻留。下面分别进行说明：

内存映射方式：内存映射文件，是由一个持久化文件到一块内存空间的映射，借助持久化文件和内存空间之间的这种映射，应用（包括多个进程）可以直接对内存执行读取和写入操作，从而修改持久化文件。在本实施例中，虚拟化实例使用持久化文件（例如磁盘文件）进行内存数据的持久化保存，为了便于描述和区分，将该持久化文件称为目标持久化文件；与此同时，为了便于读写操作，可采用内存映射的方式将该目标持久化文件映射至虚拟化实

例分配到的系统内存空间中，即在系统内存空间中创建内存映射文件，该内存映射文件是对虚拟化实例使用的目标持久化文件的内存映射，内存映射文件通过将目标持久化文件映射到内存空间中，当虚拟化实例对映射的内存区域进行操作（如读取或写入）时，这些操作会反映到映射的目标持久化文件中。

目标持久化文件用于持久化存储内存映射文件中的数据。为了实现将这些操作反映到映射的目标持久化文件中，可以为虚拟化实例创建所述目标持久化文件，并在系统内存空间中，创建目标持久化文件对应的内存映射文件；在虚拟化实例产生第一内存数据过程中，将第一内存数据写入内存映射文件中；以及，按照设定的持久化策略，将内存映射文件中的数据存储到目标持久化文件中。其中，设定的持久化策略可以是定期持久化即按照一定的时间间隔定期将第一内存数据写入持久化文件中（也即执行持久化操作）；或者，还可以是空闲持久化即当检测到 CPU 空闲时执行持久化操作；或者，在每次虚拟化实例向内存映射文件写入第一内存数据后执行持久化操作；或者，在清除内存映射文件时执行持久化操作；或者，在虚拟化实例关闭或者暂停之前执行持久化操作。每种持久化策略都有其适应的场景，可以按需选择，本公开对此不作限定。

基于上述，以文件共享的方式，对第一内存数据进行原地驻留的一种实施方式包括：在存储第一内存数据时，将第一内存数据存储至系统内存空间中的内存映射文件中，在目标虚拟化组件的升级过程中，保持该内存映射文件不被释放，以实现第一内存数据的原地驻留；所述内存映射文件是对虚拟化实例使用的目标持久化文件的内存映射，目标持久化文件用于持久化存储该内存映射文件中的数据。

内存文件方式：在一些操作系统中提供有内存文件系统，内存文件系统可以以内存文件的方式在系统内存空间中进行数据存储，其中，内存文件不会随着虚拟化实例的释放而释放，只有在宿主机断电的情况下才会被释放。基于此，在本公开实施例中，以文件共享的方式，对第一内存数据进行原地驻留的另一种实施方式包括：在存储第一内存数据时，调用内存文件系统以内存文件的方式将第一内存数据存储至虚拟化实例分配到的系统内存空间中，这样，在目标虚拟化组件的升级过程中，借助于内存文件具有不被释放的特性，实现保持该内存文件不被释放的目的，进而实现第一内存数据的原地驻留。

进一步，Linux 内存管理采用“分页机制”，标准页的默认大小为 4 KB，而 HugeTLB 提供了标准页更大的页大小（如 2MB 或 1GB），映射的内存页变大，从而减少了用于内存映射的页表的大小，提高了内存访问效率。其中，采用 Hugetlb FS 方式在虚拟化实例恢复前后对应的新旧两个进程之间进行第一内存数据的原地驻留，即内存文件系统在系统内存空间中创建内存文件时采用 Hugetlb FS 方式创建更大粒度的内存页用来存储第一内存数据；在目标虚拟化组件的升级过程中，借助于该内存文件不会被释放的特性，实现第一内存数据的原地驻留，而且还可以减少用于内存映射的页表的大小，提高内存访问效率。

在本公开实施例中，并不限定对目标内存空间中的目标内存数据进行原地驻留的方式，凡是能够对目标内存数据进行原地驻留的均适用于本公开实施例。除此上述方式之外，还可以采用对内存空间进行锁定的方式实现对第一内存数据的原地驻留。具体地，在对虚拟化实

例进行状态保存过程中，可以锁定分配给虚拟化实例的系统内存空间，例如为该系统内存空间添加锁定标记，该锁定标记用于标识在虚拟化实例暂停过程中锁定该系统内存空间，不对其进行释放和修改；在虚拟化实例恢复后，根据该锁定标记将该系统内存空间重新分配给恢复后的虚拟化实例。进一步，为了便于确定锁定标记与虚拟化实例之间的对应关系，在对虚拟化实例进行状态保存过程中，可以先为该虚拟化实例的系统内存空间添加锁定标记，并将该锁定标记添加到虚拟化实例的状态数据中。这样，在虚拟化实例恢复后，可以从其状态数据中获取该锁定标记，进而将带有该锁定标记的系统内存空间重新分配给恢复后的虚拟化实例，从而在虚拟化实例状态保存和恢复过程中实现第一内存数据的原地驻留，无需进行内存拷贝，提高虚拟化实例的恢复效率。

进一步可选地，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复，包括：根据虚拟化实例对应的资源文件，重新创建虚拟化实例，虚拟化实例复用恢复之前的唯一性标识；进一步，根据虚拟化实例的唯一性标识（如 UUID），将目标内存空间重新分配给虚拟化实例，该目标内存空间中包含状态保存时原地驻留的目标内存数据；以及，根据指定文件中的状态数据，恢复虚拟化实例至进行状态保存时的运行状态。在一示例中，虚拟化实例可以具体实现为 VM，则基于所保存的虚拟化实例（旧 VM）的状态信息，对虚拟化实例（新 VM）进行状态恢复，包括：根据旧 VM 对应的资源文件重新创建的 VM（称为新 VM）复用旧 VM 的 UUID（唯一性标识），该 UUID 关联了旧 VM 的各种资源文件，这些资源文件会分配给恢复后的 VM；进一步，根据 UUID，将目标内存空间重新分配给对应的新 VM；以及，根据指定文件中的状态数据，恢复新 VM 至进行状态保存时的运行状态。

在另一可选实施例中，物理机包括各种带有内存空间且具有计算能力的外设设备，这里的外设设备是指广泛意义上的外设设备，包括通过各种互联总线（例如 PCIe）与物理机的 CPU 互联的设备。在实际应用中，根据应用需求，可以为虚拟化实例创建相应的虚拟外设，例如虚拟 GPU，并会分配相应的内存空间（例如 GPU 的显存），为了区分，将该内存空间称为外设内存空间。虚拟化实例运行过程中，在使用虚拟外设时产生的数据可存储到外设内存空间中。

基于此，在保存虚拟化实例的内存数据时，还涉及虚拟化实例在外设内存空间中的内存数据的保存。在一可选实施例中，目标内存空间还可以包括外设内存空间，外设内存空间是虚拟化实例使用的目标物理外设设备上的内存空间，目标物理外设设备可以是具有计算能力和内存空间的外设设备，例如可以是但不限于 GPU。进一步，目标内存数据包括存储于目标物理外设设备上的外设内存空间中的第二内存数据，则对虚拟化实例的目标内存空间中的目标内存数据进行原地驻留，还包括：对第二内存数据进行原地驻留。

其中，对于第二内存数据的原地驻留，首先包括：在对虚拟化实例进行状态保存和状态恢复过程中，确保不对外设内存空间中的第二内存数据进行改动。具体地，在对虚拟化实例进行状态保存过程中涉及旧进程的释放或销毁，在该过程中，不对外设内存空间中的第二内存数据进行改写；在对虚拟化实例进行状态恢复过程中涉及新进程的创建，在该过程中，确保不对外设内存空间进行初始化，以保证第二内存数据不被改动。进一步可选地，根据目标

物理外设设备的驱动程序是否是需要进行升级的目标虚拟化组件对第二内存数据进行原地驻留会有所不同，可分情况而定。具体地，若需要对目标物理外设设备的驱动程序进行升级，即在目标虚拟化组件包括目标物理外设设备的驱动程序的情况下，这会涉及该驱动程序的卸载、安装和加载过程，在这个过程中，不同厂商可能采用不同的策略来确保数据的安全性和设备的稳定性，例如，一些厂商可能会在驱动程序卸载时进行数据擦除；另一些厂商可能会在驱动程序加载时初始化内存。这些操作会影响到第二内存数据，因此对第二内存数据进行原地驻留还包括：保证在该驱动程序升级前后第二内存数据不发生变化。

基于上述，在目标虚拟化组件包括目标物理外设设备的驱动程序的情况下，通过对目标物理外设设备的驱动程序的卸载和加载过程进行改进，在卸载驱动程序时，禁止对外设内存空间进行改写操作；在重新加载升级后的驱动程序时，禁止对外设内存空间进行初始化操作，用以保证分配给虚拟化实例的外设内存空间中的第二内存数据在升级前后不发生变化。进一步可选地，在目标虚拟化组件不包括目标物理外设设备的驱动程序的情况下，不涉及卸载和重新加载，所以也就不涉及相关操作。

为了应对不同厂商的策略，确保外设内存空间中的第二内存数据可以原地驻留，可以对驱动程序进行改进，在驱动程序中增加内存驻留功能，该内存驻留功能包括在卸载过程中不对外设内存空间进行改写操作和在加载过程中不对外设内存空间进行初始化操作；其中，升级后的驱动程序默认开启内存驻留功能。在一可选实施例中，驱动程序中的内存驻留功能允许按需开启，即在卸载驱动程序之前，还可以通过驱动程序对外暴露的接口，开启驱动程序的内存驻留功能；在开启内存驻留功能的情况下，在卸载过程中不对外设内存空间进行改写操作，相应地，在加载过程中不对外设内存空间进行初始化操作；其中，升级后的驱动程序默认开启内存驻留功能。

可选地，可以通过传递内核模块参数来控制驱动程序的内存驻留功能是否启用。在驱动不升级时，可以不开启内存驻留功能；在驱动程序完成升级的时候，需要确保内存驻留功能是开启的，例如在加载的过程中，通过传递内核模块参数的方式，以启用驱动程序的内存驻留功能。当然控制内存驻留功能是否启用的方式不限于此，例如，在 Linux 系统中，除了传递内核模块参数的方式，还可以通过 Linux 命令控制内存驻留功能的启用；在 Windows 系统（微软公司以图形用户界面为基础研发的操作系统）中，提供了图形用户界面，用户可以通过图形用户界面配置内存驻留功能，包括启用或禁用该功能。

当然，驱动程序中的内存驻留功能也可以默认处于常开启状态，在这种情况下，可以直接对驱动程序进行卸载，且在卸载过程中不对外设内存空间进行改写操作。

进一步可选地，在目标虚拟化组件包括虚拟化实例使用的目标物理外设设备的驱动程序的情况下，在对升级前的目标虚拟化组件进行卸载操作之前，还可以优化驱动程序的卸载过程、安装过程、加载过程和/或初始化过程，这样可以在后续驱动程序涉及卸载、安装、加载和/或初始化时，缩短相应的时间，提高驱动程序的升级效率。

需要注意的是，上面虽然重点强调了对目标物理外设设备的驱动程序的卸载过程、安装过程、加载过程和/或初始化过程进行优化，主要是因为目标物理外设设备的驱动程序的卸载

过程、安装过程、加载过程和/或初始化过程通常比较复杂，耗时较多，具有更好的优化空间，但并不限于此。对于其它虚拟化组件，若卸载过程、安装过程、加载过程以及初始化过程中任意过程具有优化空间，也可以进行优化，进一步缩短升级时间，提高效率。另外，需要说明的是，关于目标虚拟化组件的卸载过程、安装过程、加载过程和/或初始化过程的优化均涉及具体的厂商，在本公开实施例中，并不关注具体的优化实现方案，凡是相比于传统方案能够缩短时间的优化方式均适用于本公开实施例。

进一步可选地，对目标内存空间中的目标内存数据（第一内存数据或二内存数据）进行原地驻留之前，可以获取配置该虚拟化实例的资源信息，根据该资源信息判断是否虚拟化实例是否使用了外设设备，在使用了外设设备的情况下，对第二内存数据进行原地驻留。

进一步可选地，在对目标内存空间中的目标内存数据（第一内存数据或第二内存数据）进行原地驻留之前，可以判断第一内存数据或第二内存数据是否大于或等于对应的阈值，若大于则需要进行原地驻留。可选地，第一内存数据和第二内存数据可以单独进行原地驻留，也可以同时进行原地驻留，本公开对此不作限制。

如图 3a 所示，本公开实施例还提供一种物理机，包括：硬件资源 11，硬件资源 11 上运行有虚拟化管理器 12、升级程序 13 和虚拟化实例；虚拟化管理器 12 包括多个虚拟化组件，被配置为实现硬件资源 11 的虚拟化以及虚拟化实例的创建和管理；

其中，升级程序 13，在具体实现上可以是一个升级脚本，也可以是非脚本式的升级程序；升级程序 13 用于确定多个虚拟化组件中待升级的目标虚拟化组件；对虚拟化实例进行状态保存；对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；以及基于所保存的所述虚拟化实例的状态信息，对虚拟化实例进行状态恢复。

如图 3b 所示，在一可选实施例中，多个虚拟化组件包括位于用户态的第一虚拟化组件 123 和位于内核态的第二虚拟化组件 124。进一步，目标虚拟化组件包括第一虚拟化组件 123 和/或第二虚拟化组件 124。可选地，位于用户态的第一虚拟化组件 123 可以是 QEMU，位于内核态的第二虚拟化组件 124 可以是 KVM。

在一可选实施例中，升级程序 13 具体用于：调用第一虚拟化组件 123，对虚拟化实例进行状态保存；以及，在目标虚拟化组件不包含第一虚拟化组件 123 的情况下，调用第一虚拟化组件 123，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复；在目标虚拟化组件包含第一虚拟化组件 123 的情况下，调用升级后的第一虚拟化组件 123，基于所保存的虚拟化实例的状态信息，对虚拟化实例进行状态恢复。

在一可选实施例中，升级程序 13 在对虚拟化实例进行状态保存时，具体用于：采集所述虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；以及，对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留。

关于本实施例方法中各步骤的详细实施方式以及有益效果已经在前述实施例中进行了详细描述，此处将不做详细阐述说明。

如图 4 所示，本公开实施例提供的一种虚拟化实例状态保存方法，如图 4 所示，该方法包括：

S401：响应虚拟化实例状态保存操作，采集虚拟化实例的状态数据，并将状态数据保存至指定文件中；

S402：在虚拟化实例恢复之前，以文件共享的方式，对虚拟化实例产生的第一内存数据进行原地驻留；其中，虚拟化实例具有目标内存空间，目标内存空间包括系统内存空间，第一内存数据以文件方式存储在系统内存空间中。

在一可选实施例中，以文件共享的方式，对第一内存数据进行原地驻留，包括：在存储第一内存数据时，将第一内存数据存储至系统内存空间中的内存映射文件中，在目标虚拟化组件的升级过程中，保持内存映射文件不被释放，以实现第一内存数据的原地驻留；内存映射文件是对虚拟化实例使用的目标持久化文件的内存映射，目标持久化文件用于持久化存储内存映射文件中的数据。

在一可选实施例中，上述方法还包括为虚拟化实例创建目标持久化文件，并在系统内存空间中，创建目标持久化文件对应的内存映射文件；在虚拟化实例产生第一内存数据过程中，将第一内存数据写入内存映射文件中；以及，按照设定的持久化策略， 将内存映射文件中的数据存储到目标持久化文件中。

在另一可选实施例中，以文件共享的方式，对第一内存数据进行原地驻留，包括：在存储第一内存数据时，调用内存文件系统以内存文件的方式将第一内存数据存储至系统内存空间中，在目标虚拟化组件的升级过程中，保持内存文件不被释放，以实现第一内存数据的原地驻留。

在一可选实施例中，目标内存空间还包括外设内存空间，外设内存空间是虚拟化实例使用的目标物理外设设备上的内存空间，外设内存空间中存储有虚拟化实例使用目标物理外设设备过程中产生的第二内存数据；上述方法还包括：在卸载目标物理外设设备的驱动程序时，禁止对外设内存空间进行改写操作；以及在重新加载驱动程序时，禁止对外设内存空间进行初始化操作，以实现对第二内存数据的原地驻留。

在一可选实施例中，上述方法还包括：根据虚拟化实例对应的资源文件，重新创建虚拟化实例，虚拟化实例复用恢复之前的唯一性标识；根据唯一性标识，将目标内存空间重新分配给虚拟化实例；以及，根据指定文件中的状态数据，恢复虚拟化实例至进行状态保存时的运行状态。

关于本实施例方法中各步骤的详细实施方式以及有益效果已经在前述实施例中进行了详细描述，此处将不做详细阐述说明。

需要说明的是，上述实施例所提供方法的各步骤的执行主体均可以是同一设备，或者，该方法也由不同设备作为执行主体。比如，步骤 401 至步骤 403 的执行主体可以为设备 A；又比如，步骤 401 和 402 的执行主体可以为设备 A，步骤 403 的执行主体可以为设备 B；等等。

另外，在上述实施例及附图中的描述的一些流程中，包含了按照特定顺序出现的多个操

作，但是应该清楚了解，这些操作可以不按照其在本文中出现的顺序来执行或并行执行，操作的序号如401、402等，仅仅是用于区分开各个不同的操作，序号本身不代表任何的执行顺序。另外，这些流程可以包括更多或更少的操作，并且这些操作可以按顺序执行或并行执行。需要说明的是，本文中的“第一”、“第二”等描述，是用于区分不同的消息、设备、模块等，不代表先后顺序，也不限定“第一”和“第二”是不同的类型。

图5为本公开又一示例性实施例提供的一种虚拟化组件升级装置的结构示意图。如图5所示，该虚拟化组件升级装置500包括：确定模块51、保存模块52、升级模块53和恢复模块54，其中：

确定模块51，被配置为确定目标虚拟化组件，所述目标虚拟化组件是运行于物理机上的虚拟化管理器中待升级的虚拟化组件；

保存模块52，被配置为对运行于所述物理机上的虚拟化实例进行状态保存，所述虚拟化实例是所述虚拟化管理器创建的；

升级模块53，被配置为对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；

恢复模块54，被配置为基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

在一可选实施例中，保存模块52在对运行于所述物理机上的虚拟化实例进行状态保存时，具体用于：采集所述虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留。

在一可选实施例中，所述目标内存空间包括系统内存空间，所述目标内存数据包括以文件方式存储于所述系统内存空间中的第一内存数据；保存模块52对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留时，具体用于：以文件共享的方式，对所述第一内存数据进行原地驻留。

进一步可选地，保存模块52在以文件共享的方式，对所述第一内存数据进行原地驻留时，具体用于：在存储所述第一内存数据时，将所述第一内存数据存储至所述系统内存空间中的内存映射文件中，在所述目标虚拟化组件的升级过程中，保持所述内存映射文件不被释放，以实现所述第一内存数据的原地驻留；所述内存映射文件是对所述虚拟化实例使用的目标持久化文件的内存映射，所述目标持久化文件用于持久化存储所述内存映射文件中的数据；或者，在存储所述第一内存数据时，调用内存文件系统以内存文件的方式将所述第一内存数据存储至所述系统内存空间中，在所述目标虚拟化组件的升级过程中，保持所述内存文件不被释放，以实现所述第一内存数据的原地驻留。

在一可选实施例中，保存模块52还用于：为所述虚拟化实例创建所述目标持久化文件，并在所述系统内存空间中，创建所述目标持久化文件对应的内存映射文件；在所述虚拟化实例产生所述第一内存数据过程中，将所述第一内存数据写入所述内存映射文件中；以及，按照设定的持久化策略，将所述内存映射文件中的数据存储到所述目标持久化文件中。

在一可选实施例中，所述目标内存空间包括外设内存空间，所述外设内存空间是所述虚拟化实例使用的目标物理外设设备上的内存空间；所述目标内存数据包括存储于所述外设内存空间中的第二内存数据；保存模块 52 在对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留时，具体用于：在所述目标虚拟化组件包括所述目标物理外设设备的驱动程序的情况下，在卸载所述驱动程序时，禁止对所述外设内存空间进行改写操作；在重新加载升级后的驱动程序时，禁止对所述外设内存空间进行初始化操作。

在一可选实施例中，在卸载所述驱动程序之前，保存模块 52 还用于：通过所述驱动程序对外暴露的接口，开启所述驱动程序的内存驻留功能；所述内存驻留功能包括在卸载过程中不对所述外设内存空间进行改写操作和在加载过程中不对所述外设内存空间进行初始化操作；

在一可选实施例中，恢复模块 54 在基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复时，具体用于：根据所述虚拟化实例对应的资源文件，重新创建所述虚拟化实例，所述虚拟化实例复用恢复之前的唯一性标识；根据所述唯一性标识，将所述目标内存空间重新分配给所述虚拟化实例；以及，根据所述指定文件中的状态数据，恢复所述虚拟化实例至进行状态保存时的运行状态。

在一可选实施例中，所述虚拟化管理器包括创建所述虚拟化实例的第一虚拟化组件；则保存模块 52 在对运行于所述物理机上的虚拟化实例进行状态保存时，具体用于：调用所述第一虚拟化组件，对所述虚拟化实例进行状态保存；相应地，恢复模块 54 在基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复时，具体用于：在所述目标虚拟化组件不包含所述第一虚拟化组件的情况下，调用所述第一虚拟化组件，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复；在所述目标虚拟化组件包含所述第一虚拟化组件的情况下，调用升级后的第一虚拟化组件，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

在一可选实施例中，在对升级前的目标虚拟化组件进行卸载操作之前，保存模块 52 还用于：在所述目标虚拟化组件包括所述虚拟化实例使用的目标物理外设设备的驱动程序的情况下，优化所述驱动程序的卸载过程、安装过程、加载过程和/或初始化过程。

在一可选实施例中，保存模块 52 在对运行于所述物理机上的虚拟化实例进行状态保存时，具体用于：在所述虚拟化实例为多个的情况下，以并发操作方式对多个虚拟化实例进行状态保存。

图 6 为本公开又一示例性实施例提供的一种虚拟化实例状态保存装置的结构示意图。如图 6 所示，虚拟化实例状态保存装置 600 包括：采集模块 61、保存模块 62 和驻留模块 63，其中：

采集模块 61，被配置为响应虚拟化实例状态保存操作，采集虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；

保存模块 62，被配置为将虚拟化实例产生的第一内存数据以文件方式存储在系统内存空间中，虚拟化实例具有目标内存空间，所述目标内存空间包括所述系统内存空间；

驻留模块 63，被配置为在所述虚拟化实例恢复之前，以文件共享的方式，对第一内存数据进行原地驻留。

在一可选实施例中，驻留模块 63 具体用于：在存储第一内存数据时，将第一内存数据存储至系统内存空间中的内存映射文件中，在目标虚拟化组件的升级过程中，保持内存映射文件不被释放，以实现第一内存数据的原地驻留；内存映射文件是对虚拟化实例使用的目标持久化文件的内存映射，目标持久化文件用于持久化存储内存映射文件中的数据；或者，在存储第一内存数据时，调用内存文件系统以内存文件的方式将第一内存数据存储至系统内存空间中，在目标虚拟化组件的升级过程中，保持内存文件不被释放，以实现第一内存数据的原地驻留。

在一可选实施例中，保存模块 62，还用于：为所述虚拟化实例创建所述目标持久化文件，并在所述系统内存空间中，创建所述目标持久化文件对应的内存映射文件；在所述虚拟化实例产生所述第一内存数据过程中，将所述第一内存数据写入所述内存映射文件中；以及，按照设定的持久化策略，将所述内存映射文件中的数据存储到所述目标持久化文件中。

在一可选实施例中，所述目标内存空间还包括外设内存空间，所述外设内存空间是所述虚拟化实例使用的目标物理外设设备上的内存空间，所述外设内存空间中存储有所述虚拟化实例使用所述目标物理外设设备过程中产生的第二内存数据；驻留模块 63，还用于：在卸载所述目标物理外设设备的驱动程序时，禁止对所述外设内存空间进行改写操作；以及在重新加载所述驱动程序时，禁止对所述外设内存空间进行初始化操作，以实现对所述第二内存数据的原地驻留。

在一可选实施例中，驻留模块 63，还用于：根据所述虚拟化实例对应的资源文件，重新创建所述虚拟化实例，所述虚拟化实例复用恢复之前的唯一性标识；根据所述唯一性标识，将所述目标内存空间重新分配给所述虚拟化实例；以及，根据所述指定文件中的状态数据，恢复所述虚拟化实例至进行状态保存时的运行状态。

图 7 为本公开一示例性实施例提供的一种物理机的结构示意图。物理机包括硬件资源，硬件资源包括：存储器 74 和处理器 75。

存储器 74，被配置为存储计算机程序，并可被配置为存储其它各种数据以支持在物理机的操作。这些数据的示例包括用于在物理机上操作的任何应用程序或方法的指令，联系人数据，电话簿数据，消息，图片，视频等。

存储器 74 可以由任何类型的易失性或非易失性存储设备或者它们的组合实现，如静态随机存取存储器（SRAM），电可擦除可编程只读存储器（EEPROM），可擦除可编程只读存储器（EPROM），可编程只读存储器（PROM），只读存储器（ROM），磁存储器，快闪存储器，磁盘或光盘。

处理器 75，与存储器 74 耦合，被配置为执行存储器 74 中的计算机程序，以用于：确定目标虚拟化组件，所述目标虚拟化组件是运行于物理机上的虚拟化管理器中待升级的虚拟化组件；对运行于所述物理机上的虚拟化实例进行状态保存，所述虚拟化实例是

所述虚拟化管理器创建的；对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

在一可选实施例中，处理器 75 在对运行于所述物理机上的虚拟化实例进行状态保存时，具体用于：采集所述虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留。

在一可选实施例中，所述目标内存空间包括系统内存空间，所述目标内存数据包括以文件方式存储于所述系统内存空间中的第一内存数据；处理器 75 对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留时，具体用于：以文件共享的方式，对所述第一内存数据进行原地驻留。

进一步可选地，处理器 75 在以文件共享的方式，对第一内存数据进行原地驻留时，具体用于：在存储第一内存数据时，将第一内存数据存储至系统内存空间中的内存映射文件中，在目标虚拟化组件的升级过程中，保持内存映射文件不被释放，以实现第一内存数据的原地驻留；内存映射文件是对虚拟化实例使用的目标持久化文件的内存映射，目标持久化文件用于持久化存储内存映射文件中的数据；或者，在存储第一内存数据时，调用内存文件系统以内存文件的方式将第一内存数据存储至系统内存空间中，在目标虚拟化组件的升级过程中，保持内存文件不被释放，以实现第一内存数据的原地驻留。

在一可选实施例中，处理器 75，还用于：为所述虚拟化实例创建所述目标持久化文件，并在所述系统内存空间中，创建所述目标持久化文件对应的内存映射文件；在所述虚拟化实例产生所述第一内存数据过程中，将所述第一内存数据写入所述内存映射文件中；以及，按照设定的持久化策略，将所述内存映射文件中的数据存储到所述目标持久化文件中。

在一可选实施例中，所述目标内存空间包括外设内存空间，所述外设内存空间是所述虚拟化实例使用的目标物理外设设备上的内存空间；所述目标内存数据包括存储于所述外设内存空间中的第二内存数据；处理器 75 在对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留时，具体用于：在所述目标虚拟化组件包括所述目标物理外设设备的驱动程序的情况下，在卸载所述驱动程序时，禁止对所述外设内存空间进行改写操作；在重新加载升级后的驱动程序时，禁止对所述外设内存空间进行初始化操作。

在一可选实施例中，在卸载所述驱动程序之前，处理器 75 还用于：通过所述驱动程序对外暴露的接口，开启所述驱动程序的内存驻留功能；所述内存驻留功能包括在卸载过程中不对所述外设内存空间进行改写操作和在加载过程中不对所述外设内存空间进行初始化操作；

在一可选实施例中，处理器 75 在基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复时，具体用于：根据所述虚拟化实例对应的资源文件，重新创建所述虚拟化实例，所述虚拟化实例复用恢复之前的唯一性标识；根据所述唯一性标识，将所述目标内存空间重新分配给所述虚拟化实例；以及，根据所述指定文件中的状态数

据，恢复所述虚拟化实例至进行状态保存时的运行状态。

在一可选实施例中，所述虚拟化管理器包括创建所述虚拟化实例的第一虚拟化组件；则处理器 75 在对运行于所述物理机上的虚拟化实例进行状态保存时，具体用于：调用所述第一虚拟化组件，对所述虚拟化实例进行状态保存；相应地，处理器 75 在基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复时，具体用于：在所述目标虚拟化组件不包含所述第一虚拟化组件的情况下，调用所述第一虚拟化组件，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复；在所述目标虚拟化组件包含所述第一虚拟化组件的情况下，调用升级后的第一虚拟化组件，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

在一可选实施例中，在对升级前的目标虚拟化组件进行卸载操作之前，处理器 75 还用于：在所述目标虚拟化组件包括所述虚拟化实例使用的目标物理外设设备的驱动程序的情况下，优化所述驱动程序的卸载过程、安装过程、加载过程和/或初始化过程。

在一可选实施例中，处理器 75 在对运行于所述物理机上的虚拟化实例进行状态保存时，具体用于：在所述虚拟化实例为多个的情况下，以并发操作方式对多个虚拟化实例进行状态保存。

进一步，如图 7 所示，该物理机还包括：通信组件 76、显示器 77、电源组件 78、音频组件 79 等其它组件。图 7 中仅示意性给出部分组件，并不意味着物理机只包括图 7 所示组件。另外，图 7 中虚线框内的组件为可选组件，而非必选组件，具体可视工作节点的产品形态而定。本实施例的工作节点可以实现为台式电脑、笔记本电脑、智能手机或 IOT 设备等终端设备，也可以是常规服务器、云服务器或服务器阵列等服务端设备。若本实施例的工作节点实现为台式电脑、笔记本电脑、智能手机等终端设备，可以包含图 7 中虚线框内的组件；若本实施例的工作节点实现为常规服务器、云服务器或服务器阵列等服务端设备，则可以不包含图 7 中虚线框内的组件。

本公开实施例还提供一种物理机，该物理机的实现结构与图 7 所示物理机的实现结构相同或类似，可参照图 7 所示物理机的结构实现。本实施例提供的物理机与图 7 所示实施例中物理机的区别主要在于：处理器执行存储器中存储的计算机程序所实现的功能不同。对本实施例提供的物理机来说，其处理器执行存储器中存储的计算机程序，可用于：响应虚拟化实例状态保存操作，采集虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；虚拟化实例具有目标内存空间，目标内存空间包括系统内存空间，且虚拟化实例产生的第一内存数据以文件方式存储在系统内存空间中；在虚拟化实例恢复之前，以文件共享的方式，对第一内存数据进行原地驻留。

在一可选实施例中，处理器在以文件共享的方式，对第一内存数据进行原地驻留时，具体用于：在存储第一内存数据时，将第一内存数据存储至系统内存空间中的内存映射文件中，在目标虚拟化组件的升级过程中，保持内存映射文件不被释放，以实现第一内存数据的原地驻留；内存映射文件是对虚拟化实例使用的目标持久化文件的内存映射，目标持久化文件用于持久化存储内存映射文件中的数据；或者，在存储第一内存数据时，

调用内存文件系统以内存文件的方式将第一内存数据存储至系统内存空间中，在目标虚拟化组件的升级过程中，保持内存文件不被释放，以实现第一内存数据的原地驻留。

在一可选实施例中，处理器还用于：为虚拟化实例创建目标持久化文件，并在系统内存空间中，创建目标持久化文件对应的内存映射文件；在虚拟化实例产生第一内存数据过程中，将第一内存数据写入所述内存映射文件中；以及，按照设定的持久化策略，将所述内存映射文件中的数据存储到所述目标持久化文件中。

在一可选实施例中，所述目标内存空间还包括外设内存空间，所述外设内存空间是所述虚拟化实例使用的目标物理外设设备上的内存空间，所述外设内存空间中存储有所述虚拟化实例使用所述目标物理外设设备过程中产生的第二内存数据；处理器还用于：在卸载所述目标物理外设设备的驱动程序时，禁止对所述外设内存空间进行改写操作；以及在重新加载所述驱动程序时，禁止对所述外设内存空间进行初始化操作，以实现对所述第二内存数据的原地驻留。

在一可选实施例中，处理器还用于：根据所述虚拟化实例对应的资源文件，重新创建所述虚拟化实例，所述虚拟化实例复用恢复之前的唯一性标识；根据所述唯一性标识，将所述目标内存空间重新分配给所述虚拟化实例；以及，根据所述指定文件中的状态数据，恢复所述虚拟化实例至进行状态保存时的运行状态。

关于本实施例方法中各步骤的详细实施方式以及有益效果已经在前述实施例中进行了详细描述，此处将不做详细阐述说明。

相应地，本公开实施例还提供一种存储有计算机程序的计算机可读存储介质，计算机程序被处理器执行时，致使处理器能够实现上述方法实施例中的各步骤。

相应地，本公开实施例还提供一种计算机程序产品，该计算机程序产品包括计算机程序/指令，当计算机程序/指令被处理器执行时，致使处理器能够实现上述方法实施例中的步骤。

上述存储器可以由任何类型的易失性或非易失性存储设备或者它们的组合实现，如静态随机存取存储器（Static Random-Access Memory，SRAM），电可擦除可编程只读存储器（Electrically Erasable Programmable Read Only Memory，EEPROM），可擦除可编程只读存储器（Erasable Programmable Read Only Memory，EPROM），可编程只读存储器（Programmable Read-Only Memory，PROM），只读存储器（Read-Only Memory，ROM），磁存储器，快闪存储器，磁盘或光盘。

上述通信组件被配置为便于通信组件所在设备和其他设备之间有线或无线方式的通信。通信组件所在设备可以接入基于通信标准的无线网络，如 WiFi，2G、3G、4G/LTE、5G 等移动通信网络，或它们的组合。在一个示例性实施例中，通信组件经由广播信道接收来自外部广播管理系统的广播信号或广播相关信息。在一个示例性实施例中，通信组件还包括近场通信（Near Field Communication，NFC）模块，以促进短程通信。例如，在 NFC 模块可基于射频识别（Radio Frequency Identification，RFID）技术，红外数据协会（Infrared Data Association，IrDA）技术，超宽带（Ultra Wide Band，UWB）技术，蓝牙（BlueTooth，BT）技术和其他

技术来实现。

上述显示器包括屏幕，其屏幕可以包括液晶显示器（Liquid Crystal Display，LCD）和触摸面板（TouchPanel，TP）。如果屏幕包括触摸面板，屏幕可以被实现为触摸屏，以接收来自用户的输入信号。触摸面板包括一个或多个触摸传感器以感测触摸、滑动和触摸面板上的手势。触摸传感器可以不仅感测触摸或滑动动作的边界，而且还检测与触摸或滑动操作相关的持续时间和压力。

上述电源组件，为电源组件所在设备的各种组件提供电力。电源组件可以包括电源管理系统，一个或多个电源，及其他与为电源组件所在设备生成、管理和分配电力相关联的组件。

上述音频组件，可被配置为输出和/或输入音频信号。例如，音频组件包括一个麦克风（Microphone，MIC），当音频组件所在设备处于操作模式，如呼叫模式、记录模式和语音识别模式时，麦克风被配置为接收外部音频信号。所接收的音频信号可以被进一步存储在存储器或经由通信组件发送。在一些实施例中，音频组件还包括一个扬声器，被配置为输出音频信号。

本领域内的技术人员应明白，本公开的实施例可提供为方法、系统、或计算机程序产品。因此，本公开可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本公开可采用在一个或多个其中包含有计算机可用程序代码的计算机可读存储介质（包括但不限于磁盘存储器、只读光盘（Compact Disc Read-Only Memory，CD-ROM）、光学存储器等）上实施的计算机程序产品的形式。

本公开是参照根据本公开实施例的方法、设备（系统）、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

在一个典型的配置中，计算设备包括一个或多个处理器（Central Processing Unit，CPU）、输入/输出接口、网络接口和内存。

内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器（Random Access Memory，RAM）和/或非易失性内存等形式，如只读存储器（ROM）或闪存（flash RAM）。

内存是计算机可读介质的示例。

计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存（Phase-change Random Access Memory，PRAM）、静态随机存取存储器（SRAM）、动态随机存取存储器（Dynamic Random Access Memory，DRAM）、其他类型的随机存取存储器（RAM）、只读存储器（ROM）、电可擦除可编程只读存储器（EEPROM）、快闪记忆体或其他内存技术、只读光盘只读存储器（CD-ROM）、数字多功能光盘（Digital Video Disc，DVD）或其他光学存储、磁盒式磁带，磁带磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括暂存电脑可读媒体（transitory media），如调制的数据信号和载波。

还需要说明的是，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括要素的过程、方法、商品或者设备中还存在另外的相同要素。

以上仅为本公开的实施例而已，并不用于限制本公开。对于本领域技术人员来说，本公开可以有各种更改和变化。凡在本公开的精神和原理之内所作的任何修改、等同替换、改进等，均应包含在本公开的权利要求范围之内。

权利要求书

1、一种虚拟化组件升级方法，包括：

确定目标虚拟化组件，所述目标虚拟化组件是运行于物理机上的虚拟化管理器中待升级的虚拟化组件；

对运行于所述物理机上的虚拟化实例进行状态保存，所述虚拟化实例是所述虚拟化管理器创建的；

对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；

基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

2、根据权利要求 1 所述的方法，其中，对运行于所述物理机上的虚拟化实例进行状态保存，包括：

采集所述虚拟化实例的状态数据，并将所述状态数据保存至指定文件中；

对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留。

3、根据权利要求 2 所述的方法，其中，所述目标内存空间包括系统内存空间，所述目标内存数据包括以文件方式存储于所述系统内存空间中的第一内存数据；

对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留，包括：以文件共享的方式，对所述第一内存数据进行原地驻留。

4、根据权利要求 3 所述的方法，其中，以文件共享的方式，对所述第一内存数据进行原地驻留，包括：

在存储所述第一内存数据时，将所述第一内存数据存储至所述系统内存空间中的内存映射文件中，在所述目标虚拟化组件的升级过程中，保持所述内存映射文件不被释放，以实现所述第一内存数据的原地驻留；所述内存映射文件是对所述虚拟化实例使用的目标持久化文件的内存映射，所述目标持久化文件用于持久化存储所述内存映射文件中的数据；

或者

在存储所述第一内存数据时，调用内存文件系统以内存文件的方式将所述第一内存数据存储至所述系统内存空间中，在所述目标虚拟化组件的升级过程中，保持所述内存文件不被释放，以实现所述第一内存数据的原地驻留。

5、根据权利要求 2 所述的方法，其中，所述目标内存空间包括外设内存空间，所述外设内存空间是所述虚拟化实例使用的目标物理外设设备上的内存空间；所述目标内存数据包括存储于所述外设内存空间中的第二内存数据；

对所述虚拟化实例的目标内存空间中的目标内存数据进行原地驻留，包括：

在所述目标虚拟化组件包括所述目标物理外设设备的驱动程序的情况下，在卸载所述驱动程序时，禁止对所述外设内存空间进行改写操作；

在重新加载升级后的驱动程序时，禁止对所述外设内存空间进行初始化操作。

6、根据权利要求 5 所述的方法，其中，在卸载所述驱动程序之前，还包括：

通过所述驱动程序对外暴露的接口，开启所述驱动程序的内存驻留功能；所述内存驻留功能包括在卸载过程中不对所述外设内存空间进行改写操作和在加载过程中不对所述外设内存空间进行初始化操作；

其中，所述升级后的驱动程序默认开启所述内存驻留功能。

7、根据权利要求 2 所述的方法，其中，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复，包括：

根据所述虚拟化实例对应的资源文件，重新创建所述虚拟化实例，所述虚拟化实例复用恢复之前的唯一性标识；

根据所述唯一性标识，将所述目标内存空间重新分配给所述虚拟化实例；以及

根据所述指定文件中的状态数据，恢复所述虚拟化实例至进行状态保存时的运行状态。

8、根据权利要求 1-7 任一项所述的方法，其中，在对升级前的目标虚拟化组件进行卸载操作之前，还包括：

在所述目标虚拟化组件包括所述虚拟化实例使用的目标物理外设设备的驱动程序的情况下，优化所述驱动程序的卸载过程、安装过程、加载过程和/或初始化过程。

9、一种物理机，包括：硬件资源，所述硬件资源上运行有虚拟化管理器、虚拟化实例和升级程序；所述虚拟化管理器包括多个虚拟化组件，被配置为实现所述硬件资源的虚拟化以及所述虚拟化实例的创建和管理；

其中，所述升级程序，被配置为确定所述多个虚拟化组件中待升级的目标虚拟化组件；对所述虚拟化实例进行状态保存；对升级前的目标虚拟化组件进行卸载操作，并重新安装和加载升级后的目标虚拟化组件；以及基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

10、根据权利要求 9 所述的物理机，其中，所述多个虚拟化组件包括位于用户态的第一虚拟化组件和位于内核态的第二虚拟化组件；所述目标虚拟化组件包括第一虚拟化组件和/或第二虚拟化组件；

所述升级程序具体用于：调用所述第一虚拟化组件，对所述虚拟化实例进行状态保存；以及

在所述目标虚拟化组件不包含所述第一虚拟化组件的情况下，调用所述第一虚拟化组件，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复；

在所述目标虚拟化组件包含所述第一虚拟化组件的情况下，调用升级后的第一虚拟化组件，基于所保存的所述虚拟化实例的状态信息，对所述虚拟化实例进行状态恢复。

11、一种虚拟化实例状态保存方法，包括：

响应虚拟化实例状态保存操作，采集虚拟化实例的状态数据，并将所述状态数据保

存至指定文件中；

所述虚拟化实例具有目标内存空间，所述目标内存空间包括系统内存空间，且所述虚拟化实例产生的第一内存数据以文件方式存储在所述系统内存空间中；

在所述虚拟化实例恢复之前，以文件共享的方式，对所述第一内存数据进行原地驻留。

12、根据权利要求 11 所述的方法，其中，以文件共享的方式，对所述第一内存数据进行原地驻留，包括：

在存储所述第一内存数据时，将所述第一内存数据存储至所述系统内存空间中的内存映射文件中，在所述目标虚拟化组件的升级过程中，保持所述内存映射文件不被释放，以实现所述第一内存数据的原地驻留；所述内存映射文件是对所述虚拟化实例使用的目标持久化文件的内存映射，所述目标持久化文件用于持久化存储所述内存映射文件中的数据；

或者

在存储所述第一内存数据时，调用内存文件系统以内存文件的方式将所述第一内存数据存储至所述系统内存空间中，在所述目标虚拟化组件的升级过程中，保持所述内存文件不被释放，以实现所述第一内存数据的原地驻留。

13、根据权利要求 11 或 12 所述的方法，其中，所述目标内存空间还包括外设内存空间，所述外设内存空间是所述虚拟化实例使用的目标物理外设设备上的内存空间，所述外设内存空间中存储有所述虚拟化实例使用所述目标物理外设设备过程中产生的第二内存数据；所述方法还包括：

在卸载所述目标物理外设设备的驱动程序时，禁止对所述外设内存空间进行改写操作；以及在重新加载所述驱动程序时，禁止对所述外设内存空间进行初始化操作，以实现对所述第二内存数据的原地驻留。

14、一种物理机，包括硬件资源，所述硬件资源包括存储器和处理器，所述存储器中存储有计算机程序，所述处理器与所述存储器耦合，被配置为执行所述计算机程序，以用于实现权利要求 1-8 以及权利要求 11-13 中任一项所述方法中的步骤。

15、一种存储有计算机程序的计算机可读存储介质，当所述计算机程序被处理器执行时，致使所述处理器能够实现权利要求 1-8 以及权利要求 11-13 中任一项所述方法中的步骤。

16、一种计算机程序产品，包括计算机程序/指令，当所述计算机程序/指令被处理器执行时，致使所述处理器能够实现权利要求 1-8 以及权利要求 11-13 中任一项所述方法中的步骤。