METHOD FOR UPGRADING VIRTUALIZATION MANAGER, DEVICE, AND STORAGE MEDIUM

[0001] The present disclosure claims priority to Application No. 202311206139.7, filed with the China National Intellectual Property Administration on September 18, 2023 and entitled "METHOD FOR UPGRADING VIRTUALIZATION MANAGER, DEVICE, AND STORAGE MEDIUM", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present disclosure relates to the field of computer technologies, and in particular, to a method for upgrading a virtualization manager, a device, and a storage medium.

BACKGROUND

[0003] In the cloud computing scenario, tenants, as users of the cloud, expect virtual machines to run their applications continuously and stably. A kernel-based virtualization management component (Kernel-based Virtual Machine, KVM), as a core virtualization component, is frequently released in new versions due to reasons such as fixing security vulnerabilities and functional issues. This requires cloud vendors to have the ability to perform a hot upgrade on the component, meaning that the operation of the virtual machine cannot be terminated during upgrading of the KVM, and the virtual machine cannot be suspended for a long time to avoid causing significant jitter to user services.

[0004] During a conventional KVM hot upgrade, a relatively long time of the suspension of the operation of the virtual machine results in relatively long virtual machine downtime.

SUMMARY

[0005] A plurality of aspects of the present disclosure provide a method for upgrading a virtualization manager, a device, and a storage medium, to reduce downtime of virtual instances during a hot upgrade of a virtualization manager.

[0006] Embodiments of the present disclosure provide a method for upgrading a virtualization manager, including:

suspending, by a first virtualization manager, running of a virtual instance on a host

in response to a virtualization manager upgrade event;

saving, by the first virtualization manager, virtual instance state information in the first virtualization manager to a target memory of the host, where the virtual instance state information is state information of the virtual instance;

providing, by the first virtualization manager, memory address information of the target memory to a second virtualization manager, where the second virtualization manager is configured to upgrade the first virtualization manager; and

obtaining, by the second virtualization manager, the virtual instance state information based on the memory address information; and restoring a state of the virtual instance in the second virtualization manager based on the virtual instance state information.

[0007] An embodiment of the present disclosure further provides a method for upgrading a virtualization manager. A first virtualization management component and a second virtualization management component which is a virtualization manager that upgrades the first virtualization management component run in a kernel mode of a host. The method includes:

creating, by a first virtualization management process running in a user mode of the host, a second virtualization management process in the user mode in response to a virtualization manager upgrade event, where the first virtualization management process collaborates with the first virtualization management component; and the second virtualization management process collaborates with the second virtualization management component;

saving, by the first virtualization management component in response to a virtual instance state saving request sent by the first virtualization management process, virtual instance state information in the first virtualization management component to a target memory in the kernel mode; and providing memory address information of the virtual instance state information to the first virtualization management process;

providing, by the first virtualization management process, the memory address information of the virtual instance state information to the second virtualization management process; and

obtaining, by the second virtualization management component in response to a virtual instance state restoration request sent by the second virtualization management process, the virtual instance state information based on the memory address information of the virtual instance state information included in the virtual instance state restoration request; and restoring a state of a virtual instance in the second virtualization management component based on the virtual instance state information.

[0008] An embodiment of the present disclosure further provides a method for upgrading

a virtualization manager. A first virtualization management component and a second virtualization management component which is a virtualization manager that upgrades the first virtualization management component run in a kernel mode of a host. A first virtualization management process that collaborates with the first virtualization management component runs in a user mode of the host. The method includes:

creating, by the first virtualization management process in response to a virtualization manager upgrade event, a second virtualization management process in the user mode, where the second virtualization management process collaborates with the second virtualization management component;

suspending, by the first virtualization management process, running of a virtual instance on the host;

saving, by the first virtualization management component in response to a virtual instance state saving request sent by the first virtualization management process, virtual instance state information in the first virtualization management component to a target memory in the kernel mode;

providing, by the first virtualization management component, memory address information of the virtual instance state information to the first virtualization management process, so that the first virtualization management process provides the memory address information to the second virtualization management process;

obtaining, by the second virtualization management component, the virtual instance state information based on the memory address information in a virtual instance state restoration request sent by the second virtualization management process; and

restoring, by the second virtualization management component, a state of the virtual instance in the second virtualization management component based on the virtual instance state information.

[0009] An embodiment of the present disclosure further provides a computing device. The computing device may be implemented as a host. The computing device includes a memory and a processor, where the memory is configured to store computer programs corresponding to the first virtualization manager and the second virtualization manager, and another computer program.

[0010] The processor is coupled to the memory and configured to execute the computer programs stored in the memory, to perform the steps in the foregoing methods for upgrading a virtualization manager.

[0011] An embodiment of the present disclosure further provides a computer-readable

storage medium storing a computer instruction, the computer instruction, when executed by one or more processors, causing the one or more processors to perform the steps of the foregoing methods for upgrading a virtualization manager.

[0012] In the embodiments of the present disclosure, during upgrading of a virtualization manager, when a virtual instance state is saved and restored, a memory address of virtual instance state information is communicated between an original virtualization manager and an upgraded new virtualization manager. Compared with a manner in a conventional solution in which the virtual instance state information needs to be replicated back and forth between a user-mode virtualization management process and a kernel-mode virtualization manager, an amount of data replicated between the original virtualization manager and the new virtualization manager, thereby reducing time overheads during virtual instance state restoration, and helping to reduce downtime of virtual instances caused by the upgrading of the virtualization manager.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The accompanying drawings described herein are used to provide a further understanding of the present disclosure, and constitute a part of the present disclosure. Exemplary embodiments of the present disclosure and descriptions thereof are used to explain the present disclosure, and do not constitute any inappropriate limitation on the present disclosure. In the accompanying drawings:

[0014] FIG. 1 is a schematic diagram of a process of performing a KVM hot upgrade using a conventional KVM hot upgrade scheme.

[0015] FIG. 2a is a schematic diagram of an internal structure of a host and a schematic diagram of an upgrade process of a virtualization manager according to an embodiment of the present disclosure.

[0016] FIG. 2b is a schematic diagram of another internal structure of a host and a schematic diagram of an upgrade process of a virtualization manager according to an embodiment of the present disclosure.

[0017] FIG. 3 is a schematic diagram of a detailed process of an upgrade process of a virtualization manager according to an embodiment of the present disclosure.

[0018] FIG. 4a and FIG. 4b are schematic flowcharts of a method for upgrading a virtualization manager according to an embodiment of the present disclosure.

[0019] FIG. 5 is a schematic structural diagram of a computing device according to an

embodiment of the present disclosure.

DESCRIPTION OF EMBODIMENTS

[0020] To make objectives, technical solutions, and advantages of the present disclosure clearer, the technical solutions of the present disclosure are clearly and completely described below with reference to specific embodiments of the present disclosure and corresponding accompanying drawings. Apparently, the described embodiments are merely some of the embodiments of the present disclosure, but not all of the embodiments. All other embodiments obtained by a person of ordinary skill in the art based on the embodiments of the present disclosure without paying creative efforts shall fall within the protection scope of the present disclosure.

[0021] A virtual machine manager (Virtual Machine Monitor, VMM) is a computer program that runs on a host system to implement virtualization management functions. The VMM generally includes a simulator QEMU (Quick Emulator) process and a kernel-based virtualization management component (Kernel-Based Virtual Machine, KVM).

[0022] The KVM is a virtualization component used in a kernel. A main function of the KVM is to implement central processing unit (CPU) virtualization, memory virtualization, interruption virtualization, and the like. After an operating system loads a KVM module, since a user cannot directly control operation of a kernel module, a virtual machine needs to be further created through another tool.

[0023] QEMU is a process running in a user mode. The virtual machine runs in a QEMU process, and one virtual CPU (VCPU) of the virtual machine corresponds to one thread of the QEMU process. A memory used by the virtual machine is allocated by the QEMU process on a host. In addition, the QEMU process is further responsible for simulation of a virtual machine device (for example, a network card, a keyboard, or a mouse).

[0024] However, the QEMU has poor performance when running independently, and high performance during runtime can be achieved by the KVM through interfaces provided by some hardware virtualization functions. The QEMU opens or closes a /dev/kvm device interface file through a command open() or close(), and calls, through a device input/output control (Input/Output Control, ioctl) interface ioctl(), an interface function provided by the KVM, thereby utilizing functions such as CPU virtualization, memory virtualization, and IO virtualization implemented by the KVM based on hardware-assisted virtualization technology.

[0025] Virtual machine manager hot upgrades include a QEMU hot upgrade and a KVM

hot upgrade. The present disclosure focuses on the KVM hot upgrade for description. The KVM hot upgrade means that during a KVM upgrade, the operation of a virtual machine is not terminated, nor is the virtual machine suspended for a long time, so as to avoid interrupting a user service deployed on the virtual machine and further avoid causing significant jitter to the user service deployed on the virtual machine.

[0026] A conventional KVM hot upgrade scheme shown in FIG. 1 usually adopts the idea of local live migration. The QEMU process runs in a user mode of the host, and the KVM runs in a kernel mode of the host. A new KVM is an upgraded version of an original KVM, which is generally loaded into the kernel mode by KVM operation and maintenance personnel. The kernel mode and the user mode are two different running states of an operating system. In the kernel mode, an operating system program may be run, and hardware may be operated, and in the user mode, only an application program can be run.

[0027] As shown in FIG. 1, the conventional KVM hot upgrade scheme mainly includes the following steps.

[0028] Step 1: An original QEMU process may create a new QEMU process in response to a KVM hot upgrade instruction, and load a binary file of a new QEMU into the new QEMU process, to implement the QEMU hot upgrade.

[0029] The KVM hot upgrade instruction is generally sent by a hot upgrade management component in the user mode. The hot upgrade management component may be triggered by operation and maintenance personnel to send the KVM hot upgrade instruction to the original QEMU process. Alternatively, the hot upgrade management component may periodically send the KVM hot upgrade instruction to the original QEMU process.

[0030] Step 2: A new QEMU process opens a new KVM, and creates a structure object related to a virtual machine (VM) in the new KVM, that is, creates a new VM in the new KVM. The structure object is a data structure for storing state information of the VM, such as kvm, kvm_vcpu, or kvm_device. The structure object in step 2 is in an initial state, and configured to store the state information of the VM.

[0031] After completing the creation of the new VM, the new QEMU replies a virtual machine creation completion message to the original QEMU.

[0032] Step 3: The original QEMU process suspends running of the virtual machine on a host in response to the virtual machine creation completion message.

[0033] Step 4: An original KVM synchronizes virtual machine state information in the original KVM to the original QEMU process through a large quantity of device input/output control (IOCTL) instruction calls. The virtual machine state information includes a CPU

register state, a state of a device on a mainboard on which the CPU is located, and the like. The device on the mainboard on which the CPU is located includes, but is not limited to, an interrupt controller, an interrupt-related device, another device, and the like.

[0034] Synchronization of state information recorded by one CPU register corresponds to one IOCTL call. Synchronization of a state of a device also corresponds to an IOCTL call. A large quantity of CPU registers and devices are provided, and a large quantity of IOCTL calls are needed to synchronize the virtual machine state information to the original QEMU.

[0035] Step 5: The original QEMU process performs serialization processing on the virtual machine state information and state information of a simulation device (such as a network card, a keyboard, or a mouse) of the virtual machine, and saves the serialized state information in a user-mode memory space. The new QEMU process may obtain the serialized state information from the user-mode memory space, and perform deserialization processing on the serialized state information, to obtain the virtual machine state information and the state information of the simulation device of the virtual machine.

[0036] Step 6: The new QEMU process transmits the virtual machine state information to the new KVM through a large quantity of IOCTL calls. The new KVM restores a state of the virtual machine based on the virtual machine state information, that is, restores a state of each structure object in the new KVM.

[0037] Step 7: The new QEMU process restores the running of the virtual machine. The original QEMU releases an occupied resource and exits.

[0038] It may be learned from the foregoing process of the conventional KVM hot upgrade that steps 4-6 of the conventional KVM hot upgrade are all completed when the running of the virtual machine is suspended. The foregoing steps 4-6 involve a large quantity of IOCTL calls and a plurality of data replication operations. These operations increase downtime of the virtual machine, resulting in relatively long downtime for the KVM hot upgrade, and causing relatively significant jitter in user applications deployed on the virtual machine. The downtime for the KVM hot upgrade refers to a duration for which the operation of the virtual machine is suspended during the KVM hot upgrade.

[0039] To reduce the downtime of the virtual machine during the KVM hot upgrade, in some embodiments of the present disclosure, during upgrading of a virtualization manager, when a virtual instance state is saved and restored, a memory address of virtual instance state information is transmitted between an original virtualization manager and an upgraded new virtualization manager. Compared with a manner in a conventional solution in which the virtual instance state information needs to be replicated back and forth between a user-mode

virtualization management process and a kernel-mode virtualization manager, an amount of data replicated between the original virtualization manager and the new virtualization manager, thereby reducing time overheads during virtual instance state restoration, and helping to reduce downtime of virtual instances caused by the upgrading of the virtualization manager.

[0040] The technical solutions provided in the embodiments of the present disclosure are described in detail below with reference to the accompanying drawings.

[0041] It should be noted that the same reference numerals represent the same object in the following accompanying drawings and embodiments. Therefore, once an object is defined in an accompanying drawing or embodiment, the object does not need to be further discussed in the subsequent accompanying drawings and embodiments.

[0042] FIG. 2a is a schematic diagram of an internal system structure of a host according to an embodiment of the present disclosure. The host may be implemented as any computer device having functions such as computing and storage, such as a server or a terminal device. The terminal device may be a computer, a workstation, a mobile phone, or the like.

[0043] A virtualization manager is deployed on the host. The virtualization manager is a computer program that manages a virtualization instance, which may be implemented as a software function module, a plug-in, or the like. The virtualization instance refers to a plurality of virtual servers deployed on a host using a virtual machine technology, which may provide different services. The virtualization instance may be a virtual machine (VM), a container, or a container group (such as a Pod). The virtualization instance may also be referred to as a client (Guest). An operating system of the host is responsible for allocating hardware resources among the plurality of virtualization instances, so that the plurality of virtualization instances are independent of each other. The operating system of the virtualization instance may communicate with hardware through the virtualization manager. Implementation forms of the virtualization instances are different, and the virtualization managers are different.

[0044] For example, in an embodiment in which the virtualization instance is a container or a container group, an operating system virtualization architecture may be used to divide a single operating system of the host into a plurality of containers, and a container manager is used for management. The expression "a plurality of" means 2 or more than 2. Correspondingly, the virtualization manager is implemented as a container manager.

[0045] For another example, in an embodiment in which the virtualization instance is a virtual machine, underlying hardware resources are no longer partitioned. Instead, a virtual machine manager (Virtual Machine Monitor, VMM) is installed on the operating system of the host. A virtual layer exists as application-level software, and does not relate to an operating

system kernel. The virtual layer simulates an independent set of hardware devices for each virtual machine, including hardware resources such as a processor, a memory, a mainboard, a graphics card, and a network card, and a client operating system is installed thereon. Correspondingly, the virtualization manager is implemented as a VMM.

[0046] New versions of the virtualization manager are often released for reasons such as fixing security vulnerabilities and functional problems. This requires a hot upgrade to be performed on the virtualization manager. The hot upgrade of the virtualization manager means that running of the virtual instance is not terminated during the upgrade of the virtualization manager, and a relatively long virtual machine downtime is not generated for the virtual instance during the upgrade of the virtualization manager, to avoid causing significant jitter to a user application deployed on the virtual instance. The downtime of the virtual instance refers to downtime of the virtual instance during the upgrade of the virtualization manager.

[0047] The upgrade process of the virtualization manager provided in the embodiments of the present disclosure is exemplarily described below.

[0048] As shown in FIG. 2a, when the virtualization manager needs to be upgraded, a computer program corresponding to a new virtualization manager that upgrades a currently running virtualization manager in a host may be loaded into the host, to obtain the new virtualization manager. In this embodiment of the present disclosure, for ease of description and distinction, a virtualization manager originally run on the host is defined as a first virtualization manager S10, and a new virtualization manager that upgrades the first virtualization manager is defined as a second virtualization manager S20.

[0049] During the upgrade of the virtualization manager, virtual instance state information in the first virtualization manager S10 needs to be synchronized to the second virtualization manager S20, and then the second virtualization manager S20 manages a virtual instance on the host. The virtual instance state information refers to state information of the virtual instance on a host, including but not limited to: state information of a virtual processor corresponding to the virtual instance (for example, state information recorded by registers corresponding to the virtual processor), state information of a device on a mainboard (for example, state information of an interrupt controller and another interrupt-related device), and the like.

[0050] During the upgrade of the virtualization manager, faster restoration of the state of the virtual instance in the second virtualization manager indicates a shorter suspension time of the virtual instance, and smaller jitter of the user application on the virtual instance. In this embodiment of the present disclosure, to reduce the suspension time of the virtual instance, the first virtualization manager S10 may suspend running of the virtual instance on the host in

response to a virtualization manager upgrade event (corresponding to step 1 in FIG. 2a).

[0051] The virtualization manager upgrade event may be triggered by an upgrade management component (not shown in FIG. 2a) running in a user mode. The upgrade management component may implement a software function module or a plug-in that manages virtualization management upgrade, or the like. The upgrade management component may detect whether the host has a new virtualization manager, and when detecting the new virtualization manager, triggers the virtualization manager upgrade event, and sends a process creation instruction to the first virtualization manager S10 (not shown in FIG. 2a). Alternatively, after loading the second virtualization manager into the host, the operation and maintenance personnel may call, through a call interface provided by the upgrade management component, the upgrade management component to trigger the virtualization manager upgrade event. The upgrade management component sends the process creation instruction to the first virtualization manager in response to the virtualization manager upgrade event.

[0052] After the running of the virtual instance is suspended, the state of the virtual instance may be migrated to the second virtualization manager S20, and the running of the virtual instance is restored in the second virtualization manager S20, to implement the hot upgrade of the virtualization manager. Based on this, in this embodiment, to migrate the state of the virtual instance to the second virtualization manager S20, the virtual instance state information in the first virtualization manager S10 is saved to a target memory of the host (corresponding to step 2 in FIG. 2a). Specifically, the first virtualization manager S10 may allocate the target memory in the host, and saves the virtual instance state information in the first virtualization manager S10 to the target memory. Further, the first virtualization manager S10 may provide memory address information of the virtual instance state information to the second virtualization manager S20 (corresponding to step 3 in FIG. 2a). The memory address information of the virtual instance state information may be a start address, a length, and the like of the virtual instance state information in a kernel-mode memory space.

[0053] Correspondingly, the second virtualization manager S20 may obtain memory address information of the target memory and obtain the virtual instance state information from the target memory based on the memory address information of the target memory. Further, the second virtualization manager S20 may restore the state of the virtual instance in the second virtualization management component S20 based on the virtual instance state information, to implement the hot upgrade of the virtualization manager (corresponding to step 4 in FIG. 2a).

[0054] In a conventional scheme, during the saving and restoration of the state of the virtual instance, the virtual instance state information needs to be replicated back and forth between

an original virtualization manager and a new virtualization manager, and the virtual instance state information further needs to be replicated back and forth between a user-mode virtualization management process and a kernel-mode virtualization manager, resulting in a relatively long time for the replication of a large amount of data. In this embodiment, during the saving and restoration of the state of the virtual instance, a memory address of the virtual instance state information is communicated between the original virtualization manager and the new virtualization manager. This reduces the amount of data replicated between the original virtualization manager and the new virtualization manager, reduces the virtual instance downtime caused by the upgrade of the virtualization manager, and then reduces the jitter of user applications on the virtual instance, thereby ensuring service performance.

[0055] In the embodiments of the present disclosure, as shown in FIG. 2b, the host is partitioned into a user mode and a kernel mode. The kernel mode and the user mode are two different running states of an operating system. In the kernel mode, an operating system program may be run, and hardware may be operated, and in the user mode, only an application program can be run. Correspondingly, the virtualization manager includes: a virtualization management process 10 deployed in the user mode, and a virtualization management component 20 deployed in the kernel mode.

[0056] The virtualization management component 20 deployed in the kernel mode is a virtualization component used in a kernel, whose main function is to implement processor virtualization, memory virtualization, interrupt virtualization, and the like. After an operating system loads the virtualization management component 20, since a user cannot directly control operation of a kernel module, a virtual machine needs to be further created through another tool.

[0057] The virtualization management process 10 is a process running in the user mode. The virtual machine runs in the virtualization management process 10, and one virtual processor of the virtual machine corresponds to one thread of the virtualization management process. A memory used by the virtual machine is allocated by the virtualization management process on a host. In addition, the virtualization management process is further responsible for simulation of a virtual machine device (for example, a network card, a keyboard, or a mouse).

[0058] However, the virtualization management process has poor performance when running independently, and high performance during runtime can be achieved by the virtualization management component 20 through interfaces provided by some hardware virtualization functions.

[0059] In some embodiments, the virtualization management component 20 may be a

KVM. Correspondingly, the virtualization management process 10 may be a QEMU process. The QEMU opens or closes a /dev/kvm device interface file through a command open() or close(), and calls, through a device input/output control (ioctl) interface ioctl(), an interface function provided by the KVM, thereby utilizing functions such as CPU virtualization, memory virtualization, and IO virtualization implemented by the KVM based on hardware-assisted virtualization technology. To be specific, the user-mode virtualization management process 10 collaborates with the kernel-mode virtualization management component 20, to implement management of a virtual instance on a host.

[0060] New versions of the virtualization manager are often released for reasons such as fixing security vulnerabilities and functional problems. This requires a hot upgrade to be performed on the virtualization manager. The hot upgrade of the virtualization manager includes a hot upgrade of a virtualization management process and a hot upgrade of a virtualization management component deployed in a kernel mode. The present disclosure focuses on the hot upgrade of the virtualization management component deployed in the kernel mode for description.

[0061] When the virtualization management component 20 running in the kernel mode needs to be upgraded, a computer program corresponding to a new virtualization management component that upgrades the virtualization management component 20 currently running in the kernel mode may be loaded into the kernel mode, to obtain a new virtualization management component deployed in the kernel mode. In the embodiments of the present disclosure, for ease of description and distinction, a virtualization management component originally running in the kernel mode of the host is defined as a first virtualization management component 201, and a new virtualization management component that upgrades the first virtualization management component is defined as a second virtualization management component 202.

[0062] During the upgrade of the kernel-mode virtualization management component, virtual instance state information in the first virtualization management component 201 needs to be synchronized to the second virtualization management component 202, and then the second virtualization management component 202 manages a virtual instance on the host. The virtual instance state information refers to state information of the virtual instance on a host, including but not limited to: state information of a virtual processor corresponding to the virtual instance (for example, state information recorded by registers corresponding to the virtual processor), state information of a device on a mainboard (for example, state information of an interrupt controller and another interrupt-related device), and the like.

[0063] During the upgrade of the kernel-mode virtualization management component 20,

faster restoration of the state of the virtual instance in the second virtualization management component 202 indicates a shorter suspension time of the virtual instance, and smaller jitter of the user application on the virtual instance. In the embodiments of the present disclosure, to reduce the downtime of the virtual instance, with reference to FIG. 2b and FIG. 3, an original virtualization management process in the user mode may create a new virtualization management process in the user mode in response to a virtualization manager upgrade event (corresponding to step 1 in FIG. 2b and steps 1-3 in FIG. 3). In the embodiments of the present disclosure, for ease of description and distinction, the original virtualization management process in the user mode is defined as a first virtualization management process 101, and a new virtualization management process created by the first virtualization management process 101 is defined as a second virtualization management process 102.

[0064] As shown in FIG. 3, the virtualization manager upgrade event may be triggered by an upgrade management component 30 running in the user mode. The upgrade management component 30 may implement a software function module or a plug-in that manages virtualization management upgrade, or the like. The upgrade management component 30 may detect whether a new virtualization management component exists in the kernel mode of the host, and when detecting a new virtualization management component, triggers a virtualization manager upgrade event (corresponding to step 1 in FIG. 3), and sends a process creation instruction to the first virtualization management process 101 (corresponding to step 2 in FIG. 3). Alternatively, after loading the second virtualization management component 202 to the kernel mode, the operation and maintenance personnel may call, through a call interface provided by the upgrade management component 30, the upgrade management component 30 to trigger the virtualization manager upgrade event. The upgrade management component 30 sends the process creation instruction to the first virtualization management process 101 in response to the virtualization manager upgrade event.

[0065] For the first virtualization management process 101, the virtualization manager upgrade event is implemented by receiving the process creation instruction sent by the upgrade management component 30. Further, the first virtualization management process 101 creates the second virtualization management process 102 in the user mode in response to the process creation instruction (corresponding to step 3 in FIG. 3). The second virtualization management process 102 collaborates with the second virtualization management component 202 to manage the virtual instance on the host.

[0066] In some embodiments, the first virtualization management process 101 may create a child process of the first virtualization management process 101 in the user mode in response

to the virtualization manager upgrade event; and load a binary file of the virtualization management process into the child process, to obtain the second virtualization management process 102 (corresponding to step 3 in FIG. 3).

[0067] Specifically, as shown in step 3 in FIG. 3, the first virtualization management process 101 may call, in response to the virtualization manager upgrade event, a fork function (such as a fork function) to create a child process of the first virtualization management process 101. The first virtualization management process 101 is a parent process, and the created process is referred to as a child process. Since the fork function can duplicate a memory image of the parent process, a new process receives a duplicate of the data from an address space of the parent process. After the fork function, the two processes respectively execute the subsequent instructions in their own memory space.

[0068] In this embodiment, the fork function creates a duplicated process (that is, a child process) of a call process (that is, the first virtualization management process 101), but many applications require the child process to execute code different from that of the parent process of the child process. An exec function provides a function of covering an image of the call process (that is, the parent process) with a new image. The fork-exec functions are used in combination, so that the child process (through the exec function) may execute a new program, and a parent process continues to execute the original program. Correspondingly, as shown in step 3 in FIG. 3, the first virtualization management process 101 may call the exec function to load a binary file of a virtualization management process into the child process, to obtain the second virtualization management process 102.

[0069] If a new version of a binary file exists in the user mode of the host, the binary file of the virtualization management process loaded into the child process is that version of the binary file. In this case, the second virtualization management process is a virtualization management process upgraded from the first virtualization management process, and the virtualization management process is upgraded. If the new version of the binary file does not exist in the user mode of the host, the binary file of the virtualization management process loaded into the child process is a binary file corresponding to the original virtualization management process (that is, the first virtualization management process), and the virtualization management process is not upgraded.

[0070] In this embodiment, the second virtualization management process 102 collaborates with the kernel-mode second virtualization management component 202 to manage the virtual instance. Specifically, the upgrade management component 30 may add an identifier of the second virtualization management component 202 to the process creation instruction, and

during creation of the second virtualization management process 102, the first virtualization management process 101 may provide the identifier of the second virtualization management component 202 to the second virtualization management process 102.

[0071] Based on the identifier of the second virtualization management component 202, the second virtualization management process 102 may open the second virtualization management component 202. Further, the second virtualization management process 102 may create, in the second virtualization management component 202, a structure object associated with the virtual instance (corresponding to step 4 in FIG. 3). The structure object associated with the virtual instance refers to a data structure for recording state information of the virtual instance in the kernel-mode virtualization management component 20, and includes, but is not limited to, a data structure for recording state information of a register associated with a virtual processor corresponding to the virtual instance, a data structure for recording state information of a virtual device corresponding to the virtual instance, and the like.

[0072] Specifically, the second virtualization management process 102 may obtain, based on the identifier of the second virtualization management component 202, an interface (such as a /dev/kvm interface) provided by the second virtualization management component 202 to the user mode, and the second virtualization management process 102 may open, through an open command (such as an open() command), the interface (such as the /dev/kvm interface) provided by the second virtualization management component 202 to the user mode, to open the second virtualization management component 202. The second virtualization management component 202 may return a file handle to a second virtualization management process 102. The second virtualization management process 102 may create, in the second virtualization management component 202 based on the file handle, a structure object associated with the virtual instance.

Specifically, the second virtualization management process 102 may create, in the second virtualization management component 202 through an IOCTL instruction, a structure object associated with the virtual instance. The IOCTL instruction includes: a command that is issued to the second virtualization management component 202 for execution and a file handle. The IOCTL instruction may be IOCTL (fd1, cmd). The fd1 is a file handle, and the cmd is a command issued to the second virtualization management component 202 for execution. In this case, the command is a create command cmd. Correspondingly, the second virtualization management component 202 may create the structure object associated with the virtual instance in the second virtualization management component 202 in response to the IOCTL instruction. The structure object associated with the virtual instance created in this case is an initial state of a virtual processor, a device, and the like corresponding to the virtual instance, which may be

configured to run a new virtual instance.

[0073] After the creation of the structure object associated with the virtual instance is completed, the second virtualization management process 102 may return a structure object creation completion message to the first virtualization management process 101 (corresponding to step 5 in FIG. 3). The first virtualization management process 101 may suspend the running of the virtual instance on the host in response to the virtualization manager upgrade event (corresponding to step 2 in FIG. 2b and step 6 in FIG. 3). Specifically, the first virtualization management process 101 may suspend the running of the virtual instance on the host (corresponding to step 2 in FIG. 2b and step 6 in FIG. 3) after the creation of the structure object associated with the virtual instance is completed. Specifically, the first virtualization management process 101 may suspend the running of the virtual instance on the host after receiving the structure object creation completion message.

[0074] After the running of the virtual instance is suspended, a state of the virtual instance may be migrated to the second virtualization management component 202, and the running of the virtual instance is restored in the second virtualization management component 202, to implement the hot upgrade of the kernel-mode virtualization management component. Based on this, in this embodiment, to migrate the state of the virtual instance to the second virtualization management component 202, the first virtualization management process 101 may request the first virtualization management component 201 to save state information of the virtual instance after the running of the virtual instance is suspended. Specifically, after the running of the virtual instance is suspended, the first virtualization management process 101 may send a virtual instance state saving request to the first virtualization management component 201 (corresponding to step 3 in FIG. 2b and step 7 in FIG. 3).

[0075] Correspondingly, the first virtualization management component 201 saves the virtual instance state information in the first virtualization management component 201 to a target memory in the kernel mode in response to the virtual instance state saving request (corresponding to step 4 in FIG. 2b and step 8 in FIG. 3). Specifically, the first virtualization management component 201 allocates the target memory in the kernel mode in response to the virtual instance state saving request, and saves the virtual instance state information in the first virtualization management component 201 to the target memory in the kernel mode. Further, the first virtualization management component 201 may provide memory address information of the virtual instance state information to the first virtualization management process 101 (corresponding to step 5 in FIG. 2b and step 9 in FIG. 3). The memory address information of the virtual instance state information may be a start address, a length, and the like of the virtual

instance state information in a kernel-mode memory space.

[0076] In some embodiments, after the running of the virtual instance is suspended, the first virtualization management process 101 may send the virtual instance state saving request to the first virtualization management component 201 through an IOCTL instruction. Correspondingly, the virtual instance state saving request may be implemented as an IOCTL instruction. The IOCTL instruction is used to request the first virtualization management component 201 to save the virtual instance state information. The IOCTL instruction may include a save command, such as a save command, instructing the first virtualization management component 201 to save the virtual instance state information, may further include a file handle fd0 provided by the first virtualization management component 201, and the like. Correspondingly, the first virtualization management component 201 may save the virtual instance state information in the first virtualization management component 201 to the target memory in the kernel mode in response to the IOCTL instruction, and return the memory address information of the virtual instance state information as a call result of the IOCTL instruction to the first virtualization management process 101.

[0077] Optionally, the first virtualization management component 201 may perform serialization on the virtual instance state information in the first virtualization management component 201 in response to the IOCTL instruction, to obtain serialized virtual instance state information (corresponding to step 8 in FIG. 3). In this embodiment, the performing serialization on the virtual instance state information in the first virtualization management component 201 mainly means converting the virtual instance state information into a correspondence between a structure object and state information of each structure object. In this way, when the state of the virtual instance is migrated to the second virtualization management component 202, the second virtualization management component 202 may determine, based on the serialized virtual instance state information, which state information corresponds to which field of the structure object of the virtual instance. Optionally, the correspondence between the structure object and the state information of each structure object may be implemented as a key-value pair (Key-Value, KV) structure. A field identifier of the structure object is a key, and state information corresponding to the field of the structure object is a value.

[0078] Correspondingly, the first virtualization management component 201 may return the memory address information of the serialized virtual instance state information as the call result of the IOCTL instruction to the first virtualization management process 101 (corresponding to step 9 in FIG. 3).

[0079] Further, the first virtualization management process 101 may provide the memory address information of the virtual instance state information to the second virtualization management process 102 (corresponding to step 6 in FIG. 2b and step 10 in FIG. 3). The second virtualization management process 102 may request, based on the memory address information of the virtual instance state information, the second virtualization management component 202 to restore a state of the virtual instance. Specifically, the second virtualization management process 102 may send a virtual instance state restoration request to the second virtualization management component 202. The virtual instance state restoration request includes the memory address information of the virtual instance state information (corresponding to step 7 in FIG. 2b and step 11 in FIG. 3).

[0080] Correspondingly, the second virtualization management component 202 may obtain, in response to the virtual instance state restoration request, the virtual instance state information from the target memory in the kernel mode based on the memory address information of the virtual instance state information included in the request (corresponding to step 7 in FIG. 2b and step 12 in FIG. 3).

[0081] In some embodiments, the second virtualization management process 102 may send the virtual instance state restoration request to the second virtualization management component 202 through an IOCTL instruction. Correspondingly, the virtual instance state restoration request may be implemented as an IOCTL instruction. The IOCTL instruction is used to request the second virtualization management component 202 to restore the virtual instance state information. The IOCTL instruction may include a restore command, such as a restore command, instructing the second virtualization management component 202 to restore the virtual instance state information, may further include a file handle fd1 and the memory address information of the virtual instance state information that are provided by the second virtualization management component 202, and the like.

[0082] Correspondingly, the second virtualization management component 202 may obtain the memory address information of the virtual instance state information in response to the IOCTL instruction, and obtain the virtual instance state information from the target memory in the kernel mode based on the memory address information of the virtual instance state information included in the request.

[0083] For the foregoing embodiment in which the virtual instance state information is stored in the target memory in the kernel mode in a serialized form, the second virtualization management component 202 may obtain the serialized virtual instance state information from the target memory in the kernel mode based on the memory address information of the

serialized virtual instance state information, and perform deserialization processing on the serialized virtual instance state information to obtain the virtual instance state information (corresponding to step 12 in FIG. 3).

[0084] In this embodiment, the kernel-mode virtualization management component may directly perform serialization and deserialization processing on the virtual instance state information in the kernel mode, and a process in which the user-mode virtualization management process directly performs serialization and deserialization processing on the virtual instance state information in the user mode is skipped, thereby reducing a processing duration of the virtual instance state information, reducing downtime of the virtual instance during a hot upgrade of the kernel-mode virtualization management component, reducing jitter of a user application on the virtual instance, and ensuring service performance.

[0085] Further, the second virtualization management component 202 may restore the state of the virtual instance in the second virtualization management component 202 based on the virtual instance state information, to implement the hot upgrade of the kernel-mode virtualization management component (corresponding to step 8 in FIG. 2b and step 13 in FIG. 3).

[0086] Specifically, the second virtualization management component 202 may assign a value to a structure object in the second virtualization management component 202 based on the virtual instance state information, thereby restoring the state of the virtual instance in the second virtualization management component 202.

[0087] In this embodiment, during upgrading of the kernel-mode virtualization management component, a user-mode original virtualization management process sends a virtual instance state saving request to an original virtualization management component, and the original virtualization management component saves the virtual instance state information to a kernel-mode memory space in response to the request, and returns a memory address of the virtual instance state information, so that the original virtualization management process saves the state of the virtual instance through a system call. Compared with a manner of saving the state of the virtual instance through a large quantity of IOCTL-type system calls in the conventional scheme, this embodiment can reduce the number of system calls during the saving of the state of the virtual instance, thereby reducing time overheads of system calls during the saving of the state of the virtual instance, helping to reduce the downtime of the virtual instance caused by the upgrade of the kernel-mode virtualization management component, reducing the jitter of the user application on the virtual instance, and ensuring the service performance.

[0088] On the other hand, in this embodiment, the new virtualization management process

in the user mode sends, to the new virtualization management component, the virtual instance state restoration request including the memory address of the virtual instance state information, and the new virtualization management component may obtain the virtual instance state information based on the memory address of the virtual instance state information, and restore the state of the virtual instance in the new virtualization management component, so that the new virtualization management process restores the state of the virtual instance through a system call. Compared with a manner of implementing the restoration through a large quantity of IOCTL-type system calls in the conventional scheme, this embodiment can reduce the number of system calls during the restoration of the state of the virtual instance, thereby reducing time overheads of system calls during the restoration of the state of the virtual instance, helping to reduce the downtime of the virtual instance caused by the upgrade of the virtualization management component, further reducing the jitter of the user application on the virtual instance, and ensuring the service performance.

[0089] Furthermore, in a conventional scheme, during the saving and restoration of the state of the virtual instance, the virtual instance state information needs to be replicated back and forth between the user-mode virtualization management process and the kernel-mode virtualization management component, and replication of a large amount of data takes a long time. In this embodiment, during the saving and restoration of the state of the virtual instance, a memory address of the virtual instance state information is communicated between the user-mode virtualization management process and the kernel-mode virtualization management component. This reduces the amount of data replicated between the user-mode virtualization management process and the kernel-mode virtualization management component, may further reduce the downtime of the virtual instance caused by the upgrade of the virtualization management component, and may reduce the jitter of user applications on the virtual instance, thereby ensuring service performance.

[0090] In addition to the foregoing hot upgrade of the kernel-mode virtualization management component, the embodiments of the present disclosure further provide decoupling the upgrade of the user-mode virtualization management process from the upgrade of the kernel-mode virtualization management component, enabling flexible selection of whether to upgrade the user-mode virtualization management process during the hot upgrade of the kernel-mode virtualization management component. For example, a selection interface for whether to upgrade the user-mode virtualization management process may be provided to a user, allowing the user to independently choose whether to upgrade the virtualization management process.

[0091] In this embodiment, the second virtualization management component 202 may

return a restoration completion message to the second virtualization management process 102 after restoring the state of the virtual instance. Optionally, the second virtualization management component 202 may return, to the second virtualization management process 102, the restoration completion message as a call result of the foregoing IOCTL instruction for requesting the second virtualization management component 202 to restore the virtual instance state information. For an embodiment in which a selection is made to upgrade the user-mode virtualization management process, the second virtualization management process 102 may restore running of the virtual instance on the host in response to the restoration completion message.

[0092] For an embodiment in which a selection is made not to upgrade the user-mode virtualization management process, since a file handle of the upgraded second virtualization management component 202 is opened by the second virtualization management process 102, the first virtualization management process 101 needs to obtain an operation handle of the second virtualization management component 202, so that the upgraded second virtualization management component 202 can be operated. Based on this, the second virtualization management process 102 may provide, in response to the restoration completion message, a target file handle opened by the second virtualization management process 102 to the first virtualization management process 101 (corresponding to step 14 in FIG. 3), and release a resource occupied by the second virtualization management process 102. The target file handle opened by the second virtualization management process 102 is a file handle provided for the opened second virtualization management component 202, including the file handle fd1 during the opening of the second virtualization management component 202, and also including another file handle for operating the second virtualization management component 202.

[0093] After obtaining the target file handle, the first virtualization management process 102 may release the file handle of the original first virtualization management component 201, and continue the running based on the target file handle of the second virtualization management component 202. Specifically, the first virtualization management process 102 may restore, based on the target file handle in the second virtualization management component 202, the running of the virtual instance on the host (corresponding to step 9 in FIG. 2b and step 15 in FIG. 3).

[0094] Specifically, the first virtualization management process 102 sends, to the second virtualization management component 202 based on the target file handle, an IOCTL instruction requesting to run a virtual instance. The IOCTL instruction may include a command requesting to run the virtual instance, such as a run command. The second virtualization

management component 202 restores the running of the virtual instance on the host in the second virtualization management component 202 in response to the IOCTL instruction.

[0095] In this embodiment, the file handle opened by the new virtualization management process is returned to the original virtualization management process, so that only the kernel-mode virtualization management component may be upgraded, and the user-mode original virtualization management process may be kept, thereby improving operation and maintenance flexibility of the virtualization manager.

[0096] It should be noted that the foregoing upgrading process of the virtualization management component may be applicable to upgrading of a single virtualization management component, or may be applicable to upgrading of a plurality of virtualization management components simultaneously. The expression "a plurality of" means 2 or more than 2. The process of upgrading the plurality of virtualization management components simultaneously is the same as upgrading each virtualization management component individually, and the upgrade process is the same as the upgrading process of a single virtualization management component. When the plurality of virtualization management components are upgraded simultaneously, a new virtualization management component (that is, the second virtualization management component) corresponding to each original virtualization management component (that is, the first virtualization management component) may be the same or different, which is flexibly set specifically by operation and maintenance personnel according to actual requirements.

[0097] In addition to the virtualization manager upgrade system provided in the foregoing embodiments, an embodiment of the present disclosure further provides a method for upgrading a virtualization manager. The method for upgrading a virtualization manager provided in the embodiments of the present disclosure is exemplarily described below.

[0098] FIG. 4a is a schematic flowchart of a method for upgrading a virtualization manager according to an embodiment of the present disclosure. As shown in FIG. 4a, the method includes the following steps.

[0099] 41. A first virtualization manager suspends running of a virtual instance on a host in response to a virtualization manager upgrade event.

[0100] 42. The first virtualization manager saves virtual instance state information in the first virtualization manager to a target memory of the host, where the virtual instance state information is state information of the virtual instance.

[0101] 43. The first virtualization manager provides memory address information of the target memory to a second virtualization manager, where the second virtualization manager is

configured to upgrade the first virtualization manager.

[0102] 44. The second virtualization manager obtains the virtual instance state information based on the memory address information.

[0103] 45. The second virtualization manager restores a state of the virtual instance in the second virtualization manager based on the virtual instance state information.

[0104] New versions of the virtualization manager are often released for reasons such as fixing security vulnerabilities and functional problems. This requires a hot upgrade to be performed on the virtualization manager. When the virtualization manager needs to be upgraded, a computer program corresponding to a new virtualization manager that upgrades a currently running virtualization manager in a host may be loaded into the host, to obtain the new virtualization manager. In this embodiment of the present disclosure, for ease of description and distinction, a virtualization manager originally run on the host is defined as a first virtualization manager, and a new virtualization manager that upgrades the first virtualization manager is defined as a second virtualization manager.

[0105] During the upgrade of the virtualization manager, virtual instance state information in the first virtualization manager needs to be synchronized to the second virtualization manager, and then the second virtualization manager manages a virtual instance on the host. For description of the virtual instance state information, reference may be made to related content of the foregoing embodiments, and details are not described herein again.

[0106] During the upgrade of the virtualization manager, faster restoration of the state of the virtual instance in the second virtualization manager indicates a shorter suspension time of the virtual instance, and smaller jitter of the user application on the virtual instance. In this embodiment of the present disclosure, to reduce the suspension time of the virtual instance, in step 41, the first virtualization manager may suspend the running of the virtual instance on the host in response to a virtualization manager upgrade event. For description of the virtualization manager upgrade event, reference may be made to related content of the foregoing embodiments, and details are not described herein again.

[0107] After the running of the virtual instance is suspended, the state of the virtual instance may be migrated to the second virtualization manager, and the running of the virtual instance is restored in the second virtualization manager, to implement the hot upgrade of the virtualization manager. Based on this, in this embodiment, to migrate the state of the virtual instance to the second virtualization manager, in step 42, the first virtualization manager may save the virtual instance state information in the first virtualization manager to the target memory of the host. Specifically, the first virtualization manager may allocate the target

memory in the host, and saves the virtual instance state information in the first virtualization manager to the target memory. Further, in step 43, the first virtualization manager may provide memory address information of the virtual instance state information to the second virtualization manager. The memory address information of the virtual instance state information may be a start address, a length, and the like of the virtual instance state information in a kernel-mode memory space.

[0108] Correspondingly, in step 44, the second virtualization manager may obtain memory address information of the target memory and obtain the virtual instance state information from the target memory based on the memory address information of the target memory. Further, in step 45, the second virtualization manager may restore the state of the virtual instance in the second virtualization management component based on the virtual instance state information, to implement the hot upgrade of the virtualization manager.

[0109] In a conventional scheme, during the saving and restoration of the state of the virtual instance, the virtual instance state information needs to be replicated back and forth between an original virtualization manager and a new virtualization manager, and the virtual instance state information further needs to be replicated back and forth between a user-mode virtualization management process and a kernel-mode virtualization manager, resulting in a relatively long time for the replication of a large amount of data. In this embodiment, during the saving and restoration of the state of the virtual instance, a memory address of the virtual instance state information is communicated between the original virtualization manager and the new virtualization manager. This reduces the amount of data replicated between the original virtualization manager and the new virtualization manager, reduces the virtual instance downtime caused by the upgrade of the virtualization manager, and then reduces the jitter of user applications on the virtual instance, thereby ensuring service performance.

[0110] In the embodiments of the present disclosure, the host is partitioned into a user mode and a kernel mode. Correspondingly, the virtualization manager includes: a virtualization management process deployed in the user mode, and a virtualization management component deployed in the kernel mode. The hot upgrade of the virtualization manager includes a hot upgrade of the virtualization management process and a hot upgrade of the virtualization management component deployed in a kernel mode. The following focuses on the hot upgrade of the virtualization management component deployed in the kernel mode for description.

[0111] FIG. 4b is a schematic flowchart of another method for upgrading a virtualization manager according to an embodiment of the present disclosure. As shown in FIG. 4b, the method main includes the following steps.

[0112] 401. A first virtualization management process creates a second virtualization management process in a user mode in response to a virtualization manager upgrade event, where the second virtualization management process collaborates with a second virtualization management component.

[0113] 402. The first virtualization management process suspends running of a virtual instance on a host.

[0114] 403. The first virtualization management component saves virtual instance state information in the first virtualization management component to a target memory in a kernel mode in response to a virtual instance state saving request sent by the first virtualization management process.

[0115] 404. The first virtualization management component provides memory address information of the target memory to the first virtualization management process, so that the first virtualization management process provides the memory address information to the second virtualization management process.

[0116] 405. The second virtualization management component obtains the virtual instance state information based on the memory address information in a virtual instance state restoration request sent by the second virtualization management process.

[0117] 406. The second virtualization management component restores a state of the virtual instance in the second virtualization management component based on the virtual instance state information.

[0118] When the virtualization manager running in the kernel mode needs to be upgraded, a computer program corresponding to a new virtualization management component that upgrades the virtualization management component currently running in the kernel mode may be loaded into the kernel mode, to obtain a new virtualization management component deployed in the kernel mode. In the embodiments of the present disclosure, for ease of description and distinction, a virtualization management component originally running in the kernel mode of the host is defined as a first virtualization management component, and a new virtualization management component that upgrades the first virtualization management component is defined as a second virtualization management component.

[0119] During the upgrade of the kernel-mode virtualization management component, virtual instance state information in the first virtualization management component needs to be synchronized to the second virtualization management component, and then the second virtualization management component manages a virtual instance on the host.

[0120] In the embodiments of the present disclosure, to reduce the downtime of the virtual

instance, in step 401, an original virtualization management process in the user mode creates a new virtualization management process in the user mode in response to a virtualization management component upgrade event. In the embodiments of the present disclosure, for ease of description and distinction, the original virtualization management process in the user mode is defined as a first virtualization management process, and a new virtualization management process created by the first virtualization management process is defined as a second virtualization management process. The second virtualization management process collaborates with the second virtualization management component to manage the virtual instance on the host.

[0121] In some embodiments, the first virtualization management process creates a child process of the first virtualization management process in the user mode in response to the virtualization manager upgrade event; and loads a binary file of a virtualization management process into the child process, to obtain the second virtualization management process.

[0122] Specifically, the first virtualization management process calls a fork function (such as a fork function) to create a child process of the first virtualization management process in response to the virtualization manager upgrade event; and controls the first virtualization management process to call an exec function to load a binary file of a virtualization management process into the child process, to obtain the second virtualization management process.

[0123] If a new version of a binary file exists in the user mode of the host, the binary file of the virtualization management process loaded into the child process is that version of the binary file. In this case, the second virtualization management process is a virtualization management process upgraded from the first virtualization management process, and the virtualization management process is upgraded. If the new version of the binary file does not exist in the user mode of the host, the binary file of the virtualization management process loaded into the child process is a binary file corresponding to the original virtualization management process (that is, the first virtualization management process), and the virtualization management process is not upgraded.

[0124] In this embodiment, the second virtualization management process collaborates with the kernel-mode second virtualization management component to manage the virtual instance. Specifically, the second virtualization management process opens the second virtualization management component. Further, the second virtualization management process creates, in the second virtualization management component, a structure object associated with the virtual instance.

[0125] Specifically, the second virtualization management process may obtain, based on the identifier of the second virtualization management component, an interface (such as a /dev/kvm interface) provided by the second virtualization management component to the user mode. Further, the second virtualization management process opens, through an open command (such as an open() command), the interface (such as the /dev/kvm interface) provided by the second virtualization management component to the user mode, to open the second virtualization management component, and obtain a file handle returned by the second virtualization management process. Further, the second virtualization management process may create, in the second virtualization management component based on the file handle, a structure object associated with the virtual instance.

[0126] Specifically, the second virtualization management process creates, in the second virtualization management component through an IOCTL instruction, the structure object associated with the virtual instance. The IOCTL instruction includes a command that is issued to the second virtualization management component for execution and used for creating a data structure object, and a file handle. Correspondingly, the second virtualization management component creates, in the second virtualization management component in response to the IOCTL instruction, the structure object associated with the virtual instance. The structure object associated with the virtual instance created in this case is an initial state of a virtual processor, a device, and the like corresponding to the virtual instance, which may be configured to run a new virtual instance.

[0127] After the creation of the structure object associated with the virtual instance is completed, the second virtualization management process may return a structure object creation completion message to the first virtualization management process. After the creation of the structure object associated with the virtual instance is completed, in step 402, the first virtualization management process may suspend the running of the virtual instance on the host in response to the virtualization manager upgrade event. The foregoing step 41 may be implemented as this step 402.

[0128] After the running of the virtual instance is suspended, a state of the virtual instance may be migrated to the second virtualization management component, and the running of the virtual instance is restored in the second virtualization management component, to implement the hot upgrade of the kernel-mode virtualization management component. Based on this, in this embodiment, to migrate the state of the virtual instance to the second virtualization management component, the foregoing step 42 may be implemented such that the first virtualization management process may request, after the running of the virtual instance is

suspended, the first virtualization management component to save state information of the virtual instance. Specifically, after the first virtualization management process suspends the running of the virtual instance, the first virtualization management process may send a virtual instance state saving request to the first virtualization management component.

[0129] Correspondingly, in step 403, the first virtualization management component saves the virtual instance state information in the first virtualization management component to the target memory in the kernel mode in response to the virtual instance state saving request. Specifically, the first virtualization management component allocates the target memory in the kernel mode in response to the virtual instance state saving request, and saves the virtual instance state information in the first virtualization management component to the target memory in the kernel mode. Further, in step 404, the first virtualization management component may provide the memory address information of the target memory to the first virtualization management process.

[0130] In some embodiments, after the running of the virtual instance is suspended, the first virtualization management process may send the virtual instance state saving request to the first virtualization management component through an IOCTL instruction. Correspondingly, the virtual instance state saving request may be implemented as an IOCTL instruction. The IOCTL instruction is used to request the first virtualization management component to save the virtual instance state information. The IOCTL instruction may include a save command, such as a save command, instructing the first virtualization management component to save the virtual instance state information, may further include a file handle fd0 provided by the first virtualization management component, and the like. Correspondingly, the first virtualization management component saves the virtual instance state information in the first virtualization management component to the target memory in the kernel mode in response to the IOCTL instruction, and returns the memory address information of the virtual instance state information as a call result of the IOCTL instruction to the first virtualization management process.

[0131] Optionally, the first virtualization management component performs serialization on the virtual instance state information in the first virtualization management component in response to the IOCTL instruction, to obtain serialized virtual instance state information. Further, the first virtualization management component returns the memory address information of the serialized virtual instance state information as the call result of the IOCTL instruction to the first virtualization management process.

[0132] Further, the first virtualization management process provides the memory address

information of the target memory to the second virtualization management process. The second virtualization management process may request, based on the memory address information of the target memory, the second virtualization management component to restore a state of the virtual instance. Specifically, the second virtualization management process may send a virtual instance state restoration request to the second virtualization management component. The virtual instance state restoration request includes the memory address information of the virtual instance state information.

[0133] Correspondingly, in step 405, the second virtualization management component obtains, in response to the virtual instance state restoration request, the virtual instance state information from the target memory in the kernel mode based on the memory address information of the target memory included in the request.

[0134] In some embodiments, the second virtualization management process sends the virtual instance state restoration request to the second virtualization management component through an IOCTL instruction. Correspondingly, the virtual instance state restoration request may be implemented as an IOCTL instruction. The IOCTL instruction is used to request the second virtualization management component to restore the virtual instance state information. The IOCTL instruction may include a restore command, such as a restore command, instructing the second virtualization management component to restore the virtual instance state information, may further include a file handle fd1 and the memory address information of the virtual instance state information that are provided by the second virtualization management component, and the like.

[0135] Correspondingly, the second virtualization management component obtains the memory address information of the target memory in response to the IOCTL instruction, and obtains the virtual instance state information from the target memory in the kernel mode based on the memory address information of the target memory included in the request.

[0136] For the foregoing embodiment in which the virtual instance state information is stored in the target memory in the kernel mode in a serialized form, the second virtualization management component may be controlled to obtain the serialized virtual instance state information from the target memory in the kernel mode based on the memory address information of the serialized virtual instance state information, and perform deserialization processing on the serialized virtual instance state information to obtain the virtual instance state information.

[0137] In this embodiment, the kernel-mode virtualization management component may directly perform serialization and deserialization processing on the virtual instance state

information in the kernel mode, and a process in which the user-mode virtualization management process directly performs serialization and deserialization processing on the virtual instance state information in the user mode is skipped, thereby reducing a processing duration of the virtual instance state information, reducing downtime of the virtual instance during a hot upgrade of the kernel-mode virtualization management component, reducing jitter of a user application on the virtual instance, and ensuring service performance.

[0138] Further, in step 406, the second virtualization management component may restore the state of the virtual instance in the second virtualization management component based on the virtual instance state information, to implement the hot upgrade of the kernel-mode virtualization management component.

[0139] Specifically, the second virtualization management component assigns a value to a structure object in the second virtualization management component based on the virtual instance state information, thereby restoring the state of the virtual instance in the second virtualization management component.

[0140] In this embodiment, during upgrading of the kernel-mode virtualization management component, a user-mode original virtualization management process sends a virtual instance state saving request to an original virtualization management component, and the original virtualization management component saves the virtual instance state information to a kernel-mode memory space in response to the request, and returns a memory address of the virtual instance state information, so that the original virtualization management process saves the state of the virtual instance through a system call. Compared with a manner of saving the state of the virtual instance through a large quantity of IOCTL-type system calls in the conventional scheme, this embodiment can reduce the number of system calls during the saving of the state of the virtual instance, thereby reducing time overheads of system calls during the saving of the state of the virtual instance, helping to reduce the downtime of the virtual instance caused by the upgrade of the virtualization management component, reducing the jitter of the user application on the virtual instance, and ensuring the service performance.

[0141] On the other hand, in this embodiment, the new virtualization management process in the user mode sends, to the new virtualization management component, the virtual instance state restoration request including the memory address of the virtual instance state information, and the new virtualization management component may obtain the virtual instance state information based on the memory address of the virtual instance state information, and restore the state of the virtual instance in the new virtualization management component, so that the new virtualization management process restores the state of the virtual instance through a

system call. Compared with a manner of implementing the restoration through a large quantity of IOCTL-type system calls in the conventional scheme, this embodiment can reduce the number of system calls during the restoration of the state of the virtual instance, thereby reducing time overheads of system calls during the restoration of the state of the virtual instance, helping to reduce the downtime of the virtual instance caused by the upgrade of the virtualization management component, further reducing the jitter of the user application on the virtual instance, and ensuring the service performance.

[0142] Furthermore, in a conventional scheme, during the saving and restoration of the state of the virtual instance, the virtual instance state information needs to be replicated back and forth between the user-mode virtualization management process and the kernel-mode virtualization management component, and replication of a large amount of data takes a long time. In this embodiment, during the saving and restoration of the state of the virtual instance, a memory address of the virtual instance state information is communicated between the user-mode virtualization management process and the kernel-mode virtualization management component. This reduces the amount of data replicated between the user-mode virtualization management process and the kernel-mode virtualization manager, may further reduce the downtime of the virtual instance caused by the upgrade of the virtualization management component, and may reduce the jitter of user applications on the virtual instance, thereby ensuring service performance.

[0143] In addition to the foregoing hot upgrade of the kernel-mode virtualization management component, the embodiments of the present disclosure further provide decoupling the upgrade of the user-mode virtualization management process from the upgrade of the kernel-mode virtualization management component, enabling flexible selection of whether to upgrade the user-mode virtualization management process during the hot upgrade of the kernel-mode virtualization management component. For example, a selection interface for whether to upgrade the user-mode virtualization management process may be provided to a user, allowing the user to independently choose whether to upgrade the virtualization management process.

[0144] In this embodiment, the second virtualization management component may return a restoration completion message to the second virtualization management process after restoring the state of the virtual instance. Optionally, the second virtualization management component returns, to the second virtualization management process, the restoration completion message as a call result of the foregoing IOCTL instruction for requesting the second virtualization management component to restore the virtual instance state information. For an embodiment in which a selection is made to upgrade the user-mode virtualization

management process, the second virtualization management process may restore running of the virtual instance on the host in response to the restoration completion message.

[0145] For an embodiment in which a selection is made not to upgrade the user-mode virtualization management process, since a file handle of the upgraded second virtualization management component is opened by the second virtualization management process, the original first virtualization management process needs to obtain an operation handle of the second virtualization management component, so that the upgraded second virtualization management component can be operated. Based on this, the second virtualization management process provides, in response to the restoration completion message, a target file handle opened by the second virtualization management process to the first virtualization management process, and releases a resource occupied by the second virtualization management process.

[0146] After the first virtualization management process obtains the target file handle, the first virtualization management process releases the file handle of the first virtualization management component, and continues the running based on the target file handle of the second virtualization management component. Specifically, the first virtualization management process restores the running of the virtual instance on the host in the second virtualization management component based on the target file handle.

[0147] Specifically, the first virtualization management process sends, to the second virtualization management component based on the target file handle, an IOCTL instruction requesting to run a virtual instance. The IOCTL instruction may include a command requesting to run the virtual instance, such as a run command. Further, the second virtualization management component restores the running of the virtual instance on the host in the second virtualization management component in response to the IOCTL instruction.

[0148] In this embodiment, the file handle opened by the new virtualization management process is returned to the original virtualization management process, so that only the kernel-mode virtualization management component may be upgraded, and the user-mode original virtualization management process may be kept, thereby improving operation and maintenance flexibility of the virtualization manager.

[0149] It should be noted that the steps of the method provided in the foregoing embodiments may be performed by the same device, or the method is performed by different devices. For example, steps 401 and 402 may be performed by a device A. For another example, step 401 may be performed by the device A, and step 402 may be performed by a device B.

[0150] In addition, some processes described in the foregoing embodiments and the drawings include a plurality of operations that appear in a specific sequence. However, it should

be clearly understood that the operations may not be performed in the sequence in which the operations appear in this specification or may be performed in parallel. Sequence numbers of the operations, such as 401 and 402, are merely used for distinguishing between different operations, and do not indicate any execution sequence. In addition, the processes may include more or fewer operations, and the operations may be performed in sequence or in parallel.

[0151] Correspondingly, an embodiment of the present disclosure further provides a computer-readable storage medium storing a computer instruction, the computer instruction, when executed by one or more processors, causing the one or more processors to perform the steps of the foregoing methods for upgrading a virtualization manager.

[0152] FIG. 5 is a schematic structural diagram of a computing device according to an embodiment of the present disclosure. The computing device may be implemented as a host. As shown in FIG. 5, the computing device may include a memory 50a and a processor 50b. The memory 50a is configured to store computer programs corresponding to the first virtualization manager and the second virtualization manager described above, and another computer program. In this embodiment, the processor 50b is coupled to the memory 50a and configured to execute the computer program stored in the memory 50a, to perform the steps in the foregoing method for upgrading a virtualization manager. For detailed descriptions of the method steps, reference may be made to the foregoing embodiments, and details are not described herein again.

[0153] In some embodiments, a first virtualization management component and a second virtualization management component which is a virtualization manager that upgrades the first virtualization management component run in a kernel mode of a host. A first virtualization management process that collaborates with the first virtualization management component runs in a user mode of the host. The memory 50a is configured to store computer programs corresponding to the first virtualization management component, the second virtualization management component, and the first virtualization management process described above, and another computer program.

[0154] In this embodiment, the processor 50b is coupled to the memory 50a and configured to execute the foregoing computer program, to perform the steps in the method for upgrading a virtualization manager shown in the foregoing FIG. 4b and a specific implementation thereof. For detailed descriptions of the method steps, reference may be made to the foregoing embodiments, and details are not described herein again.

[0155] In some optional implementations, as shown in FIG. 5, the computing device may further include optional components such as a communication component 50c, a power supply

component 50d, a display component 50e, and an audio component 50f. Only some components are schematically provided in FIG. 5, which does not mean that the computing device necessarily includes all of the components shown in FIG. 5, nor does it mean that the computing device can only include the components shown in FIG. 5.

[0156] In addition, the components in the dashed boxes in FIG. 5 are optional rather than mandatory, which depends on the product form of the electronic device. The computing device in this embodiment may be implemented as a terminal device such as a desktop computer, a notebook computer, a mobile phone, or an Internet of Things device, or may be various server devices such as a conventional server, a cloud server, or a server cluster.

[0157] In the embodiments of the present disclosure, the memory is configured to store a computer program, and may be configured to store various other data to support operations on a device on which the memory is located. The processor may execute the computer program stored in the memory to implement corresponding control logic. The memory may be implemented by any type of volatile or non-volatile storage device or a combination thereof, such as a static random-access memory (SRAM), an electrically erasable programmable read only memory (EEPROM), an erasable programmable read only memory (EPROM), a programmable read only memory (PROM), a read only memory (ROM), a magnetic memory, a flash memory, a magnetic disk, or an optical disk.

[0158] In the embodiments of the present disclosure, the processor may be any hardware processing device that can execute the foregoing method logic. Optionally, the processor may be a central processing unit (CPU), a graphics processing unit (GPU), or a microcontroller unit (MCU), or may be a programmable device such as a field-programmable gate array (FPGA), a programmable array logic (PAL) device, a general array logic (GAL) device, or a complex programmable logic device (CPLD), or an advanced reduced instruction set computer (RISC) processor (Advanced RISC Machines, ARM), a system on chip (SoC), which is not limited thereto.

[0159] In the embodiments of the present disclosure, the communication component is configured to facilitate wired or wireless communication between a device in which the communication component is located and another device. The device in which the communication component is located may access a wireless network based on a communication standard, such as wireless fidelity (Wi-Fi), 2G or 3G, 4G, 5G, or a combination thereof. In an exemplary embodiment, the communication component receives a broadcast signal or broadcast-related information from an external broadcast management system through a broadcast channel. In an exemplary embodiment, the communication component

may further be implemented based on a near field communication (NFC) technology, a radio frequency identification (RFID) technology, an infrared data association (IrDA) technology, an ultra wide band (UWB) technology, a Bluetooth (BT) technology, or other technologies.

[0160] In the embodiments of the present disclosure, the display component may include a liquid crystal display (LCD) and a touch panel (TP). If the display component includes the touch panel, the display component may be implemented as a touch screen, to receive an input signal from a user. The touch panel includes one or more touch sensors to sense touch, sliding, and gestures on the touch panel. The touch sensor may not only sense a boundary of a touching or sliding operation, but also detect a duration and a pressure related to the touching or sliding operation.

[0161] In the embodiments of the present disclosure, the power supply component is configured to provide power to various components of a device in which the power supply component is located. The power supply component may include a power supply management system, one or more power supplies, and other components associated with generation, management, and power distribution for the device in which the power supply component is located.

[0162] In the embodiments of the present disclosure, the audio component may be configured to output and/or input an audio signal. For example, the audio component includes a microphone (MIC). When the device in which the audio component is located is in an operation mode, such as a call mode, a record mode, and a speech recognition mode, the microphone is configured to receive an external audio signal. The received audio signal may be further stored in a memory or sent through a communication component. In some embodiments, the audio component further includes a loudspeaker configured to output an audio signal. For example, for a device having a language interaction function, voice interaction with a user may be implemented through an audio component.

[0163] It should be noted that user information (including but not limited to user equipment information, user personal information, and the like) and data (including but not limited to data for analysis, stored data, displayed data, and the like) involved in the present disclosure are all authorized by the user or information and data fully authorized by all parties. The collection, use, and processing of relevant data need to conform to relevant laws, regulations, and standards of relevant countries and regions, and provide corresponding operation portals for the user to choose to authorize or refuse.

[0164] It should be further noted that the descriptions such as "first" and "second" in this specification are used to distinguish different messages, devices, modules, and the like, and do

not represent the order of precedence, nor do they limit "first" and "second" to different types.

[0165] A person skilled in the art should understand that the embodiments of the present disclosure may be provided as a method, a system, or a computer program product. Therefore, the present disclosure may take the form of a complete hardware embodiment, a complete software embodiment, or an embodiment combining software and hardware. In addition, the present disclosure may be in the form of a computer program product implemented on one or more computer-usable storage media (including but not limited to a magnetic disk memory, a compact disc read-only memory (CD-ROM), an optical memory, or the like) that include computer-usable program code.

[0166] The present disclosure is described with reference to the flowcharts and/or block diagrams of the method, the device (system), and the computer program product in the embodiments of the present disclosure. It should be understood that computer program instructions can implement each process and/or block in the flowcharts and/or block diagrams and a combination of processes and/or blocks in the flowcharts and/or block diagrams. These computer program instructions may be provided for a general-purpose computer, a special-purpose computer, an embedded processor, or a processor of another programmable data processing device to generate a machine, so that the instructions executed by the computer or the processor of the another programmable data processing device generate an apparatus for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0167] The computer program instructions may also be stored in a computer-readable memory that can direct the computer or the another programmable data processing device to operate in a specific manner, so that the instructions stored in the computer-readable memory generate a product that includes an instruction apparatus. The instruction apparatus implements functions specified in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0168] These computer program instructions may also be loaded onto a computer or other programmable data processing devices, so that a series of operating steps are performed on the computer or other programmable devices to generate a computer-implemented process, and the instructions executed on the computer or other programmable devices provide steps for implementing the functions specified in one or more processes of the flowchart and/or one or more blocks of the block diagram.

[0169] In a typical configuration, a computing device includes one or more processors (a CPU, and the like), an input/output interface, a network interface, and a memory.

[0170] The memory may include forms such as a non-permanent memory in a computer-readable medium, a random access memory (RAM), and/or a non-volatile memory, for example, a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of the computer-readable medium.

[0171] The storage medium of the computer is a readable storage medium, which may also be referred to as a readable medium. The readable storage medium includes both permanent and non-permanent, removable and non-removable media, which can implement information storage through any method or technology. The information may be a computer-readable instruction, a data structure, a program module, or other data. Examples of a storage medium of the computer include but are not limited to a phase-change memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), another type of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (Digital Video Disc, DVD) or another optical storage or magnetic cassette tape, a disk storage or another magnetic storage device, or any other non-transmission medium, which may be configured to store information accessible by a computing device. As defined herein, the computer-readable medium does not include a transitory computer-readable medium (transitory media), such as a modulated data signal and a carrier.

[0172] It should be further noted that a term "comprise", "include", or any other variant thereof is intended to encompass non-exclusive inclusion, such that a process, a method, a commodity, or a device including a series of elements not only includes those elements, but also includes other elements not listed explicitly, or further includes elements inherent to the process, the method, the commodity, or the device. Without more limitations, an element defined by a statement "include a/an ..." does not exclude presence of other identical elements in the process, the method, the commodity, or the device that includes the foregoing element.

[0173] The foregoing content is merely the embodiments of the present disclosure and is not intended to limit the present disclosure. For a person skilled in the art, various modifications and variations may be made to the present disclosure. Any modification, equivalent replacement, and improvement made within the spirit and principle of the present disclosure shall fall within the scope of the claims of the present disclosure.

CLAIMS

1. A method for upgrading a virtualization manager, comprising:

suspending, by a first virtualization manager, running of a virtual instance on a host in response to a virtualization manager upgrade event;

saving, by the first virtualization manager, virtual instance state information in the first virtualization manager to a target memory of the host, wherein the virtual instance state information is state information of the virtual instance;

providing, by the first virtualization manager, memory address information of the target memory to a second virtualization manager, wherein the second virtualization manager is configured to upgrade the first virtualization manager; and

obtaining, by the second virtualization manager, the virtual instance state information based on the memory address information; and restoring a state of the virtual instance in the second virtualization manager based on the virtual instance state information.

2. The method according to claim 1, wherein the first virtualization manager comprises a first virtualization management process running in a user mode;

wherein the suspending, by a first virtualization manager, running of a virtual instance on a host in response to a virtualization manager upgrade event comprises:

suspending, by the first virtualization management process, the running of the virtual instance on the host in response to the virtualization manager upgrade event.

3. The method according to claim 2, wherein the first virtualization manager further comprises a first virtualization management component running in a kernel mode;

wherein the saving, by the first virtualization manager, virtual instance state information in the first virtualization manager to a target memory of the host comprises:

sending, by the first virtualization management process, a virtual instance state saving request to the first virtualization management component; and

saving, by the first virtualization management component in response to the virtual instance state saving request, the virtual instance state information in the first virtualization management component to the target memory in the kernel mode.

4. The method according to claim 3, wherein a second virtualization management component further runs in the kernel mode; the second virtualization component is a component for upgrading the first virtualization management component, and the second virtualization manager comprises the second virtualization management component;

the method further comprises:

creating, by the first virtualization management process, a second virtualization management process in the user mode in response to the virtualization manager upgrade event, wherein the second virtualization manager further comprises the second virtualization management process; and the second virtualization management process collaborates with the second virtualization management component;

providing, by the first virtualization management component, the memory address information of the target memory to the first virtualization management process; and

providing, by the first virtualization management process, the memory address information of the target memory to the second virtualization management process.

5. The method according to claim 4, wherein the obtaining, by the second virtualization manager, the virtual instance state information based on the memory address information comprises:

sending, by the second virtualization management process, a virtual instance state restoration request to the second virtualization management component, wherein the virtual instance state restoration request comprises the memory address information of the target memory; and

obtaining, by the second virtualization management component in response to the virtual instance state restoration request, the virtual instance state information based on the memory address information of the target memory;

wherein the restoring a state of the virtual instance in the second virtualization manager based on the virtual instance state information comprises:

restoring, by the second virtualization management component, the state of the virtual instance in the second virtualization management component based on the virtual instance state information.

6. The method according to claim 4, wherein the creating, by the first virtualization management process, a second virtualization management process in the user mode in response to the virtualization manager upgrade event comprises:

creating, by the first virtualization management process, a child process of the first virtualization management process in the user mode in response to the virtualization manager upgrade event; and loading a binary file of a virtualization management process into the child process, to obtain the second virtualization management process.

7. The method according to claim 5, further comprising:

opening, by the second virtualization management process, the second virtualization management component, and creating, in the second virtualization management component, a

structure object associated with the virtual instance;

wherein the restoring, by the second virtualization management component, the state of the virtual instance in the second virtualization management component based on the virtual instance state information comprises:

assigning, by the second virtualization management component, a value to the structure object based on the virtual instance state information, to restore the state of the virtual instance in the second virtualization management component.

8. The method according to claim 7, wherein the opening, by the second virtualization management process, the second virtualization management component, and creating, in the second virtualization management component, a structure object associated with the virtual instance comprises:

opening, by the second virtualization management process, the second virtualization management component, to obtain a file handle provided by the second virtualization management component; and

creating, by the second virtualization management process based on the file handle, the structure object associated with the virtual instance in the second virtualization management component.

9. The method according to claim 8, wherein the creating, by the second virtualization management process based on the file handle, the structure object associated with the virtual instance in the second virtualization management component comprises:

sending, to the second virtualization management component by the second virtualization management process based on the file handle, a first device input/output control (IOCTL) instruction for requesting the second virtualization management component to create a structure object; and

creating, by the second virtualization management component in response to the first IOCTL instruction, the structure object associated with the virtual instance.

10. The method according to claim 5, wherein the sending, by the first virtualization management process, a virtual instance state saving request comprises:

sending, to the first virtualization management component by the first virtualization management process, a second IOCTL instruction for requesting the first virtualization management component to save the virtual instance state information, as the virtual instance state saving request; and

wherein the sending, by the second virtualization management process, a virtual instance state restoration request to the second virtualization management component comprises:

sending, to the second virtualization management component by the second virtualization management process, a third IOCTL instruction for requesting the second virtualization management component to restore a state of the virtual instance in the first virtualization management component, as the virtual instance state restoration request, wherein the third IOCTL instruction comprises memory address information of the virtual instance state information.

11. The method according to claim 5, wherein the saving, by the first virtualization management component in response to the virtual instance state saving request, the virtual instance state information in the first virtualization management component to the target memory in the kernel mode comprises:

obtaining, by the first virtualization management component, the virtual instance state information in the first virtualization management component in response to the virtual instance state saving request sent by the first virtualization management process;

performing, by the first virtualization management component, serialization processing on the virtual instance state information in the first virtualization management component, to obtain serialized virtual instance state information; and

saving, by the first virtualization management component, the serialized virtual instance state information to the target memory in the kernel mode.

12. The method according to claim 11, wherein the obtaining, by the second virtualization management component in response to the virtual instance state restoration request, the virtual instance state information based on the memory address information of the target memory comprises:

obtaining, by the second virtualization management component in response to the virtual instance state restoration request, the serialized virtual instance state information based on the memory address information of the target memory; and

performing, by the second virtualization management component, deserialization processing on the serialized virtual instance state information, to obtain the virtual instance state information.

13. The method according to claim 5 or any one of claims 7 to 12, wherein after the restoring, by the second virtualization management component, the state of the virtual instance in the second virtualization management component based on the virtual instance state information, the method further comprises:

providing, by the second virtualization management process, a target file handle opened by the second virtualization management process to the first virtualization management process,

and releasing a resource occupied by the second virtualization management process; and

restoring, by the first virtualization management process, running of the virtual instance in the second virtualization management component based on the target file handle.

14. The method according to claim 13, wherein the restoring, by the first virtualization management process, running of the virtual instance in the second virtualization management component based on the target file handle comprises:

sending, to the second virtualization management component by the first virtualization management process based on the target file handle, a fourth IOCTL instruction requesting to run the virtual instance; and

restoring, by the second virtualization management component, the running of the virtual instance in the second virtualization management component in response to the fourth IOCTL instruction.

15. A computing device, configured to be implemented as a host;

wherein the computing device comprises a memory and a processor, wherein the memory is configured to store computer programs corresponding to a first virtualization manager and a second virtualization manager, and another computer program; and

the processor is coupled to the memory and configured to execute the computer program stored in the memory to perform steps of the method according to any one of claims 1 to 14.

16. A computer-readable storage medium storing a computer instruction, the computer instruction, when executed by one or more processors, causing the one or more processors to perform steps of the method according to any one of claims 1 to 14.

17. A computer program, comprising a computer instruction, the computer instruction, when the computer instruction is executed by a processor, steps of the method according to any one of claims 1 to 14 are implemented.