

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2016 年 11 月 10 日 (10.11.2016)



(10) 国际公布号
WO 2016/177260 A1

- (51) 国际专利分类号:
G06F 9/445 (2006.01)
- (21) 国际申请号: PCT/CN2016/079159
- (22) 国际申请日: 2016 年 4 月 13 日 (13.04.2016)
- (25) 申请语言: 中文
- (26) 公布语言: 中文
- (30) 优先权:
201510231006.4 2015 年 5 月 7 日 (07.05.2015) CN
- (71) 申请人: 阿里巴巴集团控股有限公司 (ALIBABA GROUP HOLDING LIMITED) [—/CN]; 英属开曼群岛大开曼乔治城资本大厦一座四层 847 号邮箱, Grand Cayman (KY)。

- (72) 发明人: 及
- (71) 申请人 (仅对美国): 傅俊康 (FU, Junkang) [CN/CN]; 中国浙江省杭州市余杭区文一西路 969 号 3 号楼 5 楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。 张献涛 (ZHANG, Xiantao) [CN/CN]; 中国浙江省杭州市余杭区文一西路 969 号 3 号楼 5 楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。
- (74) 代理人: 北京国昊天诚知识产权代理有限公司 (CO-HORIZON INTELLECTUAL PROPERTY INC.); 中国北京市朝阳区小关北里甲 2 号渔阳置业大厦 B 座 605, Beijing 100029 (CN)。
- (81) 指定国 (除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

[见续页]

(54) Title: LIBVIRT SOFTWARE HOT UPGRADING METHOD AND DEVICE

(54) 发明名称: 利博伟特软件热升级方法及设备

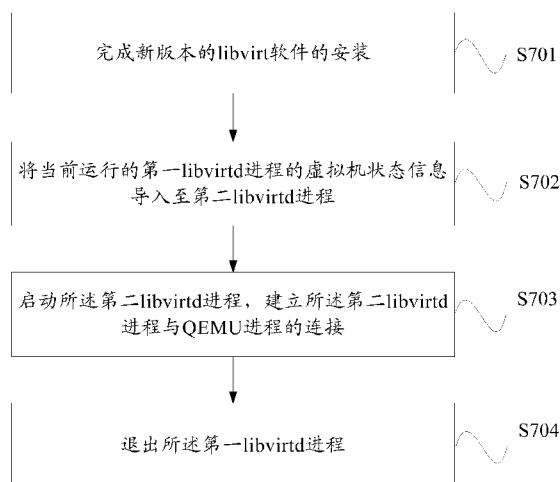


图 7

- S701 Installation of a new version of libvirt software is completed
- S702 Virtual machine state information of a currently running first libvirt process is imported to a second libvirt process
- S703 The second libvirt process is initialized to establish a connection between the second libvirt process and a QEMU process
- S704 Exit the first libvirt process

(57) Abstract: Disclosed are a libvirt software hot upgrading method and device. Specifically, installation of a new version of libvirt software is completed (S701); virtual machine state information of a currently running first libvirt process is imported to a second libvirt process (S702); the second libvirt process is initialized to establish a connection between the second libvirt process and a machine virtualizer (QEMU) process (S703); and exit the first libvirt process (S704). Compared with the prior art, since installation of the new version of libvirt software has been completed after the new second libvirt process is initialized, the second libvirt process already runs based on the new version of software codes. Meanwhile, the virtual machine state information of the second libvirt process has been updated to conform to the first libvirt process before initiated, so that libvirt software-related functions can be realized through the correct second libvirt process and thus interruption of these libvirt software-related functions is avoided during an upgrading process.

(57) 摘要: 一种利博伟特 (libvirt) 软件热升级的方法及设备。具体地, 完成新版本的 libvirt 软件的安装 (S701); 将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程 (S702); 启动所述第二 libvirtd 进程, 建立所述第二 libvirtd 进程与机器虚拟机 (QEMU) 进程的连接 (S703); 退出所述第一 libvirtd 进程 (S704)。与现有技术相比,

在新的第二 libvirtd 进程启动后, 由于此时已经完成新版本的 libvirt 软件的安装, 第二 libvirtd 进程已经是基于新版本的软件代码运行的, 同时该第二 libvirtd 进程的虚拟机状态信息已经在启动之前更新至与第一 libvirtd 进程一致, 使 libvirt 软件的相关功能能够通过正确的第二 libvirtd 进程实现, 从而在升级过程中避免 libvirt 软件的相关功能中断。



WO 2016/177260 A1



(84) **指定国** (除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ,

CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第 21 条(3))。

利博伟特软件热升级方法及设备

技术领域

本申请涉及通信及计算机领域，尤其涉及一种利博伟特（libvirt）软件热升级方法及设备。

5

背景技术

随着互联网技术的发展，虚拟化技术已经得到了越来越广泛的应用。libvirt 软件是一款应用较为广泛的虚拟化软件，用于管理虚拟机和虚拟化功能，如虚拟机创建/关闭，虚拟机热迁移，VCPU（Virtual Central Processing Unit，虚拟处理器）热插拔，磁盘热插拔等。libvirt 包含一个守护进程 libvirtd，在 KVM（Kernel-based Virtual Machine，基于内核的虚拟机）虚拟化环境下，libvirt 的相关功能都需要通过 libvirtd 进程来实现，因此在 libvirt 软件的使用过程中，需要保证 libvirtd 进程一直保持运行状态。

10

15

20

在对 libvirt 软件进行版本升级时，由于原来运行的利博伟德（libvirtd）进程是基于旧版本的软件代码，在完成新版本的软件安装后，libvirtd 进程无法在运行状态下直接调用新版本的软件代码。目前采用的方式是，在安装完新版本的软件后，重启 libvirtd 进程，使得 libvirtd 进程能够基于新版本的软件代码运行，从而完成 libvirt 软件的升级。但是，上述方式必然导致 libvirtd 进程停止运行一段时间，从而造成 libvirt 软件的相关功能在该段时间内不可用，例如无法创建虚拟机等。

综上所述，现有的 libvirt 软件升级方法会造成 libvirt 软件的相关功能在一定时间内中断。

发明内容

25

本申请的目的是提供一种 libvirt 软件热升级方法及设备，解决目前存在的 libvirt 软件的相关功能会在升级过程中中断的问题。

有鉴于此，本申请提供一种 libvirt 软件热升级方法，所述方法包括：完成新版本的 libvirt 软件的安装；

将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程，其中，所述第一 libvirtd 进程基于旧版本的 libvirt 软件运行，所述第二 libvirtd 进程基于所述新版本的 libvirt 软件运行；

5 启动所述第二 libvirtd 进程，建立所述第二 libvirtd 进程与机器虚拟机（QEMU）进程的连接；

退出所述第一 libvirtd 进程。

进一步地，在将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之前，还包括：

10 阻塞 QEMU 进程发往 libvirtd 进程的第一事件消息，其中，所述第一事件消息包括在虚拟机状态改变时由 QEMU 进程发送给 libvirtd 进程的事件消息；以及

在退出所述第一 libvirtd 进程之后，还包括：

取消对 QEMU 进程发往 libvirtd 进程的第一事件消息的阻塞。

15 进一步地，所述阻塞 QEMU 进程发往当前运行的第一 libvirtd 进程的第一事件消息，包括：

由当前运行的所述第一 libvirtd 进程向所述 QEMU 进程发送第一命令，所述 QEMU 进程根据所述第一命令中止向 libvirtd 进程发送所述第一事件消息；

取消对 QEMU 进程发往 libvirtd 进程的第一事件消息的阻塞，包括：

20 由当前运行的所述第二 libvirtd 进程向所述 QEMU 进程发送第二命令，所述 QEMU 进程根据所述第二命令恢复向 libvirtd 进程发送所述第一事件消息。

进一步地，启动所述第二 libvirtd 进程，包括：

25 启动所述第二 libvirtd 进程时，所述第二 libvirtd 进程创建一个对应于所述第二 libvirtd 进程的连接路径，其中，所述对应于所述第二 libvirtd 进程连接路径用于建立客户端与所述第二 libvirtd 进程的连接；

退出所述第一 libvirtd 进程之前，还包括：

客户端在所述第二 libvirtd 进程启动后，根据 libvirtd 进程的连接路径的创建时间，选择对应于所述第二 libvirtd 进程的连接路径与所述第二

libvirtd 进程建立连接。

进一步地，在启动所述第二 libvirtd 进程之前，还包括：

获取所述第一 libvirtd 进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirtd 进程的虚拟机状态信息内，其中，所述第一时间段为将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程至启动所述第二 libvirtd 进程之间的时间区间。

进一步地，获取所述第一 libvirtd 进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirtd 进程的虚拟机状态信息内，包括：

在将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之前，启动监听程序，该监听程序与所述第一 libvirtd 进程建立连接；

所述监听程序获取由所述第一 libvirtd 进程在所述第一时间段内发出的第二事件消息，其中，所述第二事件消息指向所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息；

所述监听程序建立与所述第二 libvirtd 进程的连接，并根据所述第二事件消息向所述第二 libvirtd 进程添加所述新产生的虚拟机状态信息。

进一步地，所述监听程序建立与所述第二 libvirtd 进程的连接，并向所述根据所述第二事件消息向所述第二 libvirtd 进程添加所述新产生的虚拟机状态信息，包括：

所述监听程序建立与所述第二 libvirtd 进程的连接；

所述监听程序将保存有所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息的状态文件拷贝至所述第二 libvirtd 进程对应的虚拟机状态文件夹中；

所述监听程序通过所述连接调用所述第二 libvirtd 进程的导入接口，所述导入接口根据所述第二事件消息，将所述状态文件中将所述新产生的虚拟机状态信息导入至所述第二 libvirtd 进程。

进一步地，启动所述第二 libvirtd 进程之后，所述监听程序退出。

进一步地，所述监听程序与所述第一 libvirtd 进程的连接为只读连接；
所述监听程序退出，包括：

所述第一 libvirtd 进程检查自身作为服务器端的连接，当确定当前仅
存在一个客户端与所述第一 libvirtd 进程建立连接并且该连接是只读连接
5 时，则指示所述监听程序退出。

进一步地，退出所述第一 libvirtd 进程，包括：

所述第一 libvirtd 进程检查自身作为服务器端的连接，当确定当前不
存在客户端与所述第一 libvirtd 进程建立有连接时，则退出所述第一
libvirtd 进程。

10 进一步地，所述第一 libvirtd 进程检查自身作为服务器端的连接之后，
还包括：

当确定当前存在客户端与所述第一 libvirtd 进程建立有连接时，则第
一 libvirtd 进程在满足预设条件后重新检查自身作为服务器端的连接，直
至确定当前不存在客户端与所述第一 libvirtd 进程建立有连接时，退出所
15 述第一 libvirtd 进程。

进一步地，所述预设条件包括：

有新的客户端与所述第一 libvirtd 进程建立或者断开连接；或
经过设定的时间间隔。

根据本申请的另一方面，还提供了一种 libvirt 软件热升级设备，所述
20 设备包括：

第一装置，用于完成新版本的 libvirt 软件的安装；

第二装置，用于将当前运行的第一 libvirtd 进程的虚拟机状态信息导
入至第二 libvirtd 进程，其中，所述第一 libvirtd 进程基于旧版本的 libvirt
软件运行，所述第二 libvirtd 进程基于所述新版本的 libvirt 软件运行；

25 第三装置，用于启动所述第二 libvirtd 进程，建立所述第二 libvirtd 进
程与 QEMU 进程的连接；

第四装置，用于退出所述第一 libvirtd 进程。

进一步地，所述设备还包括：

第五装置，用于在将当前运行的第一 libvirtd 进程的虚拟机状态信息

导入至第二 libvirt 进程之前，阻塞 QEMU 进程发往 libvirt 进程的第一事件消息，其中，所述第一事件消息包括在虚拟机状态信息改变时由 QEMU 进程发送给 libvirt 进程的事件消息；以及

5 用于在退出所述第一 libvirt 进程之后，取消对 QEMU 进程发往 libvirt 进程的第一事件消息的阻塞。

进一步地，所述第五装置用于：

在将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程之前，指示当前运行的所述第一 libvirt 进程向所述 QEMU 进程发送第一命令，所述 QEMU 进程根据所述第一命令中止向 libvirt 进程
10 发送所述第一事件消息；以及

在退出所述第一 libvirt 进程之后，指示当前运行的所述第二 libvirt 进程向所述 QEMU 进程发送第二命令，所述 QEMU 进程根据所述第二命令恢复向第二 libvirt 进程发送所述第一事件消息。

进一步地，所述第三装置，用于：

15 启动所述第二 libvirt 进程时，所述第二 libvirt 进程建立一个对应于所述第二 libvirt 进程的连接路径，其中，所述对应于所述第二 libvirt 进程连接路径用于建立客户端与所述第二 libvirt 进程的连接；

所述设备还包括：

20 第六装置，用于在退出所述第一 libvirt 进程之前，指示客户端在所述第二 libvirt 进程启动后，根据 libvirt 进程的连接路径的创建时间，选择对应于所述第二 libvirt 进程的连接路径与所述第二 libvirt 进程建立连接。

进一步地，所述设备还包括：

25 第七装置，用于在启动所述第二 libvirt 进程之前，获取所述第一 libvirt 进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirt 进程的虚拟机状态信息内，其中，所述第一时间段为将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程至启动所述第二 libvirt 进程之间的时间区间。

进一步地，所述第七装置包括：

第七一模块，用于在将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程之前启动一监听程序，该监听程序与所述第一 libvirt 进程建立连接；

5 第七二模块，用于控制所述监听程序获取由所述第一 libvirt 进程在所述第一时间段内发出的第二事件消息，其中，所述第二事件消息指向所述第一 libvirt 进程创建新虚拟机所新产生的虚拟机状态信息；

第七三模块，用于控制所述监听程序建立与所述第二 libvirt 进程的连接，并根据所述第二事件消息向所述第二 libvirt 进程添加所述新产生的虚拟机状态信息。
10

进一步地，所述第七三模块，包括：

第七三一单元，用于控制所述监听程序建立与所述第二 libvirt 进程的连接；

第七三二单元，用于控制所述监听程序将保存有所述第一 libvirt 进程创建新虚拟机所新产生的虚拟机状态信息的状态文件拷贝至所述第二 libvirt 进程对应的虚拟机状态文件夹中；
15

第七三三单元，用于控制所述监听程序通过所述连接调用所述第二 libvirt 进程的导入接口，及控制所述导入接口根据所述第二事件消息，将所述状态文件中将所述新产生的虚拟机状态信息导入至所述第二 libvirt 进程。
20

进一步地，所述设备还包括第八装置，用于在启动所述第二 libvirt 进程之后，指示所述监听程序退出。

进一步地，所述监听程序与所述第一 libvirt 进程的连接为只读连接；

所述第八装置，用于在启动所述第二 libvirt 进程之后，指示所述第一 libvirt 进程检查自身作为服务器端的连接，当确定当前仅存在一个客户端与所述第一 libvirt 进程建立连接并且该连接是只读连接时，则指示所述监听程序退出。
25

进一步地，所述第四装置，用于：

指示所述第一 libvirt 进程检查自身作为服务器端的连接，当确定当

前不存在客户端与所述第一 libvirtd 进程建立有连接时，则退出所述第一 libvirtd 进程。

进一步地，所述第四装置，还用于：

在所述第一 libvirtd 进程检查自身作为服务器端的连接之后，当确定
5 当前存在客户端与所述第一 libvirtd 进程建立有连接时，则指示第一 libvirtd 进程在满足预设条件后重新检查自身作为服务器端的连接，直至确定当前不存在客户端与所述第一 libvirtd 进程建立有连接时，退出所述第一 libvirtd 进程。

进一步地，所述预设条件包括：

10 有新的客户端与所述第一 libvirtd 进程建立或者断开连接；或
经过设定的时间间隔。

与现有技术相比，本申请在新的第二 libvirtd 进程启动后，由于此时已经完成新版本的 libvirt 软件的安装，第二 libvirtd 进程已经是基于新版本的软件代码运行的，同时该第二 libvirtd 进程的虚拟机状态信息已经在
15 启动之前更新至与第一 libvirtd 进程一致，使 libvirt 软件的相关功能能够通过正确的第二 libvirtd 进程实现，从而在升级过程中避免 libvirt 软件的相关功能中断。

此外，本申请中通过在特定时间对 QEMU 进程发往 libvirtd 进程的第一事件消息进行阻塞或者取消阻塞，以及通过监听程序对第一时间段内通
20 过第一 libvirtd 进程创建新虚拟机的操作进行监听，避免第二 libvirtd 进程的虚拟机状态信息发生错误。

附图说明

通过阅读参照以下附图所作的对非限制性实施例所作的详细描述，本
25 申请的其它特征、目的和优点将会变得更明显：

图 1 示出根据本申请一个方面的一种 libvirt 软件热升级设备的结构示意图；

图 2 示出了本申请涉及的 libvirt 软件在 KVM/QEMU 虚拟化环境下的功能示意图；

图 3 示出了本申请实施例提供的一种优选的 libvirt 软件热升级设备的结构示意图；

图 4 示出了本申请实施例提供的另一种优选的 libvirt 软件热升级设备的结构示意图；

5 图 5 示出了本申请实施例提供的 libvirt 软件热升级设备中第七装置的结构示意图；

图 6 示出了本申请实施例提供的 libvirt 软件热升级设备中第七装置中第七三模块的结构示意图；

10 图 7 示出了根据本申请另一个方面的一种 libvirt 软件热升级方法的流程图；

图 8 示出了本申请实施例提供的一种优选的 libvirt 软件热升级方法的流程图；

图 9 示出了本申请实施例提供的进一步优选的 libvirt 软件热升级方法的流程图；

15 图 10 (a) 至图 10 (f) 示出了应用本申请实施例提供的方法对 KVM/QEMU 虚拟化环境下的 libvirtd 软件进行热升级过程中交互原理图。

附图中相同或相似的附图标记代表相同或相似的部件。

具体实施方式

20 在本申请一个典型的配置中，终端、服务网络的设备和可信方均包括一个或多个处理器 (CPU)、输入/输出接口、网络接口和内存。

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

25 计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存 (PRAM)、静态随机存取存储器 (SRAM)、动态随机存取存储器 (DRAM)、其他类型的随机存取存储器 (RAM)、只读存储器 (ROM)、电

可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器 (CD-ROM)、数字多功能光盘 (DVD) 或其他光学存储、磁盒式磁带, 磁带磁盘存储或其他磁性存储设备或任何其他非传输介质, 可用于存储可以被计算设备访问的信息。按照本文中的界定, 计算机可读
5 介质不包括非暂存电脑可读媒体 (transitory media), 如调制的数据信号和载波。

图 1 示出根据本发明一个方面的一种 libvirt 软件热升级设备 1 的结构示意图, 其中, 该设备 1 包括第一装置 110、第二装置 120、第三装置 130 和第四装置 140。具体地, 第一装置 110 完成新版本的 libvirt 软件的安装;
10 第二装置 120 将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程; 第三装置 130 启动所述第二 libvirtd 进程, 建立所述第二 libvirtd 进程与 QEMU 进程的连接; 第四装置 140 退出所述第一 libvirtd 进程。在新的第二 libvirtd 进程启动后, 由于此时已经完成新版本的 libvirt 软件的安装, 第二 libvirtd 进程已经是基于新版本的软件代码运行的, 同
15 时该第二 libvirtd 进程的虚拟机状态信息已经在启动之前更新至与第一 libvirtd 进程一致, 使 libvirt 软件的相关功能能够通过正确的第二 libvirtd 进程实现, 从而在升级过程中避免 libvirt 软件的相关功能中断。

在此, 设备 1 包括但不限于网络设备、触摸终端或网络设备与触摸终端通过网络相集成所构成的设备。在此, 所述网络设备包括但不限于如网
20 络主机、单个网络服务器、多个网络服务器集或基于云计算的计算机集合等实现; 或者由用户设备实现。在此, 云由基于云计算 (Cloud Computing) 的大量主机或网络服务器构成, 其中, 云计算是分布式计算的一种, 由一群松散耦合的计算机集组成的一个超级虚拟计算机。优选地, 设备 1 还可以是运行于所述网络设备、触摸终端或网络设备与触摸终端通过网络相集
25 成所构成的设备上的脚本程序。

本申请涉及的 libvirt 软件包括一套 API (Application Programming Interface, 应用程序接口) 库、一个守护进程(libvirtd 进程)和一个命令行工具 (virsh 工具), 用以对虚拟机监视器 (Hypervisor) 进行虚拟化管理。图 2 示出了 libvirt 软件在 KVM/QEMU 虚拟化环境下的功能示意图, 在

libvirt 软件的运行过程中，virsh 工具 210 与 libvirtd 进程 220 之间的连接为短连接，在传输一次数据至后两者之间的连接随即断开，在该连接中 virsh 工具 210 作为客户端（client），而 libvirtd 进程 220 则作为服务器端（server）。而 libvirtd 进程 220 与 QEMU 进程 230 之间的连接为长连接，
5 在传输完数据后一直保持连接，直至收到断开该连接请求，在该连接中 libvirtd 进程 220 作为客户端，而 QEMU 进程 230 则作为服务器端。

在设备 1 中，第一 libvirtd 进程和第二 libvirtd 进程可以通过修改原有的 libvirtd 进程获得，例如在第一 libvirtd 进程和第二 libvirtd 进程中增加一标记参数，在启动可以传入不同的标记参数，以使两个带有不同标记参
10 数的 libvirtd 进程可以共存，并且保有各自不同的虚拟机状态信息。例如将 0 和 1 作为不同的标记参数，则第一 libvirtd 进程和第二 libvirtd 进程则分别可以 libvirtd-0 和 libvirtd-1 来表示。libvirtd-0 和 libvirtd-1 各自创建有一个用于保存虚拟机状态信息的虚拟机状态文件（这些文件存放在各自的虚拟机状态文件夹中，每个虚拟机对应一个状态文件），在 libvirtd-1 启动
15 时，可以将 libvirtd-0 的虚拟机状态文件复制给 libvirtd-1，以完成虚拟机状态信息的导入。

以采用 KVM/QEMU 的虚拟化方案为例，每个虚拟机对应一个 QEMU 进程，当 QEMU 进程在启动时，相当于创建了一台新的虚拟机。在如图 2 所示的虚拟化环境下，libvirtd 进程作为服务器端，可以接收来自 virsh 工
20 具发来的命令，并向其返回结果。同时，libvirtd 进程可以根据该来自 virsh 工具的命令进一步构造发往 QEMU 进程的命令，并作为客户端向 QEMU 进程发送该构造的命令，例如暂停/恢复虚拟机等。而 QEMU 进程在收到相关命令后，则会执行对应的操作，并将结果返回 libvirtd 进程。此外，QEMU 进程还会通过 monitor（QEMU 监控器）主动向连接到 monitor 的
25 libvirtd 进程发送事件消息（event），例如用户关闭虚拟机后，QEMU 进程会发送 stop event 这一事件消息给 libvirtd 进程，以更新 libvirtd 进程的虚拟机状态信息。

在 QEMU 进程启动时，会通过 libvirtd 进程传入配置参数使其提供一个 monitor，该 monitor 是 QEMU 进程与 libvirtd 进程进行交互的接口，

libvirtd 进程与 QEMU 进程之间的数据收发均会通过 monitor 进行。本申请为了解决现有技术的缺陷，需要配置两个相对独立的 libvirtd 进程，即第一 libvirtd 进程和第二 libvirtd 进程，那么同样需要提供两个相对独立的 monitor，分别用于与第一 libvirtd 进程、第二 libvirtd 进程保持独立的通信。

- 5 本实施例中，同样可以通过传入 0 和 1 这两个不同的标记参数，来配置两个 monitor，如 monitor-0 和 monitor-1，使得 libvirtd-0 与 monitor-0 建立连接，libvirtd-1 则与 monitor-1 建立连接，进行独立的通信。

在此，所述虚拟机状态信息可以通过文件、表格或者内存数据等形式保存。例如，本实施例中所述的虚拟机状态信息采用.xml 文件的形式保存，
10 该文件以虚拟机的名称命名，如名称为 VM1 的虚拟机，其对应的虚拟机状态文件为 VM1.xml。对于每个 libvirtd 进程，都有用于保存该虚拟机状态文件的文件夹，假设在升级前已经有 3 个虚拟机正在运行，其对应虚拟机状态文件为 VM1.xml、VM2.xml、VM3.xml，这些文件内记录有上述三个虚拟机的虚拟机状态信息。第二 libvirtd 进程启动之前，会将上述 3 个
15 虚拟机状态文件拷贝至第二 libvirtd 进程的虚拟机状态文件夹中。在此，本领域技术人员应能理解上述虚拟机状态信息的保存形式仅为举例，其他现有的或今后可能出现的虚拟机状态信息的保存形式如可适用于本申请，也应包含在本申请保护范围以内，并在此以引用方式包含于此。

在实际的处理过程中，第二装置 120 将当前运行的第一 libvirtd 进程
20 的虚拟机状态信息导入至第二 libvirtd 进程之后，至第三装置 130 启动所述第二 libvirtd 进程，建立所述第二 libvirtd 进程与 QEMU 进程的连接时，会存在一定的处理时间。在该处理时间内，若发生关闭虚拟机的情形，第二 libvirtd 进程无法通过导入第一 libvirtd 进程的虚拟机状态信息获知该状态改变的信息，并且由于此时第二 libvirtd 进程还未与 QEMU 进程建立连
25 接，QEMU 进程对应的所述关闭虚拟机的消息事件（stop event）无法送达第二 libvirtd 进程，使得第二 libvirtd 进程无法获得这一关闭虚拟机所产生的虚拟机状态信息。因此，在第二 libvirtd 进程与 QEMU 进程建立连接后，第二 libvirtd 进程的虚拟机状态信息与虚拟机的实际运行状态，造成虚拟状态错误。

为了进一步解决上述问题，本申请的提供了一种优选的实施方式，如图 3 所示。除图 1 所示的装置外，所述设备 1 还可以包括第五装置 150。具体地，第五装置在第一装置将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程之前，阻塞 QEMU 进程发往 libvirt 进程的第一事件消息；以及在第四装置退出所述第一 libvirt 进程之后，取消对 QEMU 进程发往 libvirt 进程的第一事件消息的阻塞。在此，本领域技术人员应当理解，第一装置 110、第二装置 120、第三装置 130 和第四装置 140 分别与图 1 实施例中对应装置的内容相同或基本相同，为简明起见，故在此不再赘述，并以引用的方式包含于此。

在此，所述第一事件消息包括在虚拟机状态信息改变时由 QEMU 进程发送给 libvirt 进程的事件消息，用以告知 libvirt 进程虚拟机状态发生了变化，使得 libvirt 进程能够根据所述第一事件消息更新其虚拟状态信息。仍以 KVM/QEMU 的虚拟化方案为例，QEMU 进程在虚拟机状态改变时会生成相应的第一事件消息并发送给 libvirt 进程，所述的第一事件消息可以包括：stop event、pause event 和 resume event 等，这些第一事件消息分别由关闭虚拟机、暂停虚拟机和恢复虚拟机等操作生成。由于 QEMU 进程中配置了两个 monitor，因此第一事件消息生成后会分别通过两个 monitor 向与该 monitor 对应的 libvirt 进程发送，当 libvirt 进程与 monitor 之间的连接正常时，第一事件消息即会送达 libvirt 进程。在本例中，若第一事件消息产生时，第二 libvirt 进程还未与 QEMU 进程建立连接，该第一事件消息将无法送达。

当阻塞 QEMU 进程发往 libvirt 进程的第一事件消息之后，上述第一事件消息在产生后，不会向任何一个 libvirt 进程发送，而是由 QEMU 进程保存在内部。当退出所述第一 libvirt 进程后，再取消对 QEMU 进程发往 libvirt 进程的第一事件消息的阻塞。取消阻塞后，所述 QEMU 进程发往 libvirt 进程的第一事件消息可能包括：1、被阻塞而保存在 QEMU 进程内部的第一事件消息，如果取消阻塞之前，有第一事件消息产生并被阻塞，那么取消阻塞后会继续发送这些第一事件消息；2、取消阻塞后新产生的第一事件消息。由于此时第二 libvirt 进程与 QEMU 进程已经建立连

接，第二 libvirt 进程仍然可以收到第二装置将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程之后，至第三装置启动所述第二 libvirt 进程，建立所述第二 libvirt 进程与 QEMU 进程的连接之前所产生的第一事件消息，由此更新自身的虚拟机状态信息，以避免虚拟机状态错误。其中，上述的 libvirt 进程表示第一 libvirt 进程与第二 libvirt 进程的总称，即 QEMU 进程发往 libvirt 进程的第一事件消息是同时发往第一 libvirt 进程与第二 libvirt 进程，只要第一 libvirt 进程与第二 libvirt 进程处于启动状态，并且与 QEMU 进程的 monitor 建立了连接，就可以收到所述第一事件消息；同理，进行阻塞后，第一事件消息不会发往第一 libvirt 进程和第二 libvirt 进程中的任何一个。

具体地，所述第五装置在将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程之前，指示当前运行的所述第一 libvirt 进程向所述 QEMU 进程发送第一命令，所述 QEMU 进程根据所述第一命令中止向 libvirt 进程发送所述第一事件消息；以及所述第五装置在退出所述第一 libvirt 进程之后，指示当前运行的所述第二 libvirt 进程向所述 QEMU 进程发送第二命令，所述 QEMU 进程根据所述第二命令恢复向第二 libvirt 进程发送所述第一事件消息。接上例，所述第一命令和第二命令可以采用 KVM/QEMU 的虚拟化方案中常用的 QEMU Monitor Command（QEMU 监控命令）来实现，添加分别用于阻塞和取消阻塞的 QEMU Monitor Command，并且通过 QEMU 进程，使其能够在接到相应的 QEMU Monitor Command 后进行阻塞或取消阻塞。

本领域技术人员应能理解上述第一命令和第二命令仅为举例，其他现有的或今后可能出现的能够控制 QEMU 进程中止或者恢复向 libvirt 进程发送所述第一事件消息的命令如可适用于本申请，也应包含在本申请保护范围以内，并在此以引用方式包含于此。

优选地，第三装置在启动所述第二 libvirt 进程时，所述第二 libvirt 进程创建一个对应于所述第二 libvirt 进程的连接路径，其中，所述对应于所述第二 libvirt 进程的连接路径用于建立客户端与所述第二 libvirt 进程的连接。对应地，所述设备还包括第六装置（未示出），该第六装置在

退出所述第一 libvirtd 进程之前，指示客户端在所述第二 libvirtd 进程启动后，根据 libvirtd 进程的连接路径的创建时间，选择对应于所述第二 libvirtd 进程的连接路径与所述第二 libvirtd 进程建立连接。

在此，对于两个先后创建的连接路径，优先选择最近创建的连接路径。

- 5 接上例，所述连接路径可以是第二 libvirtd 进程作为服务器端的 domain socket（域套接字）路径，该 domain socket 路径会在 libvirtd 进程启动时自动创建。仍以前述两个 libvirtd 进程 libvirtd-0 和 libvirtd-1 为例，两个 libvirtd 进程能够拥有各自独立的 domain socket 路径，在 libvirtd-0 和 libvirtd-1 启动时，所创建的 domain socket 路径分别为
- 10 /var/run/libvirt/libvirt-sock-1 和 /var/run/libvirt/libvirt-sock-2。假设 libvirtd-0 为在版本升级前开始运行的 libvirtd 进程，而 libvirtd-1 为新启动的 libvirtd 进程，libvirtd-0 中作为服务器端的 domain socket 路径的创建时间为 t1，而 libvirtd-1 中作为服务器端的 domain socket 路径的创建时间为 t2，t2 晚于 t1。若两个 libvirtd 进程同时运行，客户端在尝试连接 libvirtd 进程时（例
- 15 如配置有 virsh 工具的设备向 libvirtd 进程发送命令时）会发现同时存在两个服务器端的 domain socket 路径，此时客户端会根据 libvirtd 进程的 domain socket 路径的创建时间，优先选择最近创建的 domain socket 路径来建立连接。如实施例 1 中，由于 t2 晚于 t1，客户端会优先选择与 libvirtd-1 建立连接，那么此时客户端向 libvirtd 进程发送的命令只会发送到
- 20 libvirtd-1。

由此，在两个 libvirtd 进程同时运行时，若配置有 virsh 工具（或者其它采用 C 或者 python 语言编写的管理工具程序）的设备需要对虚拟机进行配置，其配置命令会发送至新启动的第二 libvirtd 进程。

- 25 在第二 libvirtd 进程还未启动时，尝试连接 libvirtd 进程的客户端仍然会与原来的第一 libvirtd 进程连接。在本实施例涉及的 KVM/QEMU 的虚拟化方案中，创建新虚拟机的操作需要由配置有 virsh 工具的设备向 libvirtd 进程发送命令，再由 libvirtd 进程根据该命令对 QEMU 进程配置，因此，在第二装置 120 将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之后，至第三装置 130 启动所述第二 libvirtd 进程

之前的这一段时间内，第二 libvirt 进程还未完成启动，若在此时创建新虚拟机，是通过原来的第一 libvirt 进程完成，则第二 libvirt 进程无法更新该新创建的虚拟机状态信息，可能会造成新创建的虚拟机状态信息被遗漏。

5 针对上述问题，本申请进一步提供了一种优选的实施方式，其结构如图 4 所示，除图 1 所示的装置外，所述设备 1 还包括第七装置 170。具体地，第七装置 170 在启动所述第二 libvirt 进程之前，获取所述第一 libvirt 进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirt 进程的虚拟机状态信息内；
10 其中，所述第一时间段为将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程至启动所述第二 libvirt 进程之间的时间区间。在此，本领域技术人员应当理解，第一装置 110、第二装置 120、第三装置 130 和第四装置 140 分别与图 1 实施例中对应装置的内容相同或基本相同，为简明起见，故在此不再赘述，并以引用的方式包含于此。

15 由于在第一时间段之前创建新虚拟机所生成的虚拟机状态信息可以通过导入的方式写入第二 libvirt 进程的虚拟机状态文件；而第一时间段之后则是直接通过第二 libvirt 进程进行新虚拟机的创建，因此第二 libvirt 进程直接可以获知此时的创建新虚拟机所生成的虚拟机状态信息。本申请通过获取所述第一 libvirt 进程在第一时间段内创建新虚拟机所新
20 产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirt 进程的虚拟机状态信息内，以此来避免新创建的虚拟机状态信息被遗漏。

进一步地，图 5 示出了图 4 中所述第七装置的结构图，该第七装置包括：

25 第七一模块 171，用于在将当前运行的第一 libvirt 进程的虚拟机状态信息导入至第二 libvirt 进程之前启动一监听程序，该监听程序将所述监听程序与所述第一 libvirt 进程建立连接；

第七二模块 172，用于控制所述监听程序获取由所述第一 libvirt 进程在所述第一时间段内发出的第二事件消息，其中，所述第二事件消息指向

所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息；

第七三模块 173, 用于控制所述监听程序建立与所述第二 libvirtd 进程的连接, 并根据所述第二事件消息向所述第二 libvirtd 进程添加所述新产生的虚拟机状态信息。

5 在此, 所述监听程序与所述第一 libvirtd 进程和第二 libvirtd 进程均会建立连接, 为了便于在该场景下发送相关的数据, 两个连接均为长连接, 并且相对于 libvirtd 进程, 该监听程序是作为客户端。例如, 虚拟机状态信息是通过文件的形式保存, 新创建的虚拟机的名称为 VM4 和 VM5, 其虚拟机状态信息保存于 VM4.xml 和 VM5.xml 文件中。对应地, 为了能够
10 使第二事件消息指向所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息, 第二事件消息需要包含的信息至少包括创建了新的虚拟机这一事件, 以及新创建的虚拟机的标识 (例如虚拟机的名称)。例如, 通过 VM5 这个虚拟机名称, 即可找到 VM5.xml 这个保存有虚拟机状态信息的文件。

15 在实际应用中, 除创建虚拟机之外的操作, 例如关闭虚拟机, 也可能由 virsh 工具 (或者其它采用 C 或者 python 语言编写的管理工具程序) 发起并通过 libvirtd 进程关闭该虚拟机对应的 QEMU 进程。此时, 监听程序也会监听到这些操作 (例如关闭虚拟机) 所对应的事件消息, 但是针对这些无关的事件消息, 监听程序会忽略掉, 不作处理。

20 进一步地, 本申请还提供了一种第七三模块的结构, 该第七三模块的具体结构如图 6 所示, 包括第七三一单元 1731、第七三二单元 1732 和第七三三单元 1733。

 其中, 第七三一单元 1731 控制所述监听程序建立与所述第二 libvirtd 进程之间的连接; 在此, 监听程序会在所述第二 libvirtd 进程启动之前等
25 待该第二 libvirtd 进程启动, 在其启动成功后即会自动与第二 libvirtd 进程建立连接。

 第七三二单元 1732 控制所述监听程序将保存有所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息的状态文件拷贝至所述第二 libvirtd 进程对应的虚拟机状态文件夹中; 接上例, 对于新创建的虚拟机

VM4 和 VM5，其虚拟机状态信息保存在 VM4.xml 和 VM5.xml 这两个状态文件中，监听程序会将这两个文件拷贝至第二 libvirtd 进程对应的虚拟机状态文件夹中。

第七三三单元 1733 控制所述监听程序通过所述连接调用所述第二 libvirtd 进程的导入接口，及控制所述导入接口根据所述第二事件消息，将所述状态文件中将所述新产生的虚拟机状态信息导入至所述第二 libvirtd 进程并持久化的存放在对应的虚拟机状态文件夹的虚拟机状态文件中。在此，第二事件消息中包含了事件由于第二事件消息中包含了虚拟机的标识，导入接口可以根据虚拟机的标识，例如通过虚拟机名称 VM5 找到 VM5.xml 这个保存有虚拟机状态信息的状态文件，由这个文件中将虚拟机状态信息导入第二 libvirtd 进程。

作为另一种优选的实施方式，结合图 4 和图 5 所示的设备 1，该设备还可以包括第八装置（未示出）。该第八装置在启动所述第二 libvirtd 进程之后，指示所述监听程序退出。例如，所述第二 libvirtd 进程启动之后，由于监听程序所提供的功能已经没有存在的必要，会通过某一特定信号来对监听程序进行指示，使得监听程序退出。

具体地，所述监听程序与所述第一 libvirtd 进程之间的连接为只读连接；所述第八装置，具体用于在启动所述第二 libvirtd 进程之后，指示所述第一 libvirtd 进程检查自身作为服务器端的连接，当确定当前仅存在一个客户端与所述第一 libvirtd 进程建立连接并且该连接是只读连接时，则指示所述监听程序退出，实现监听程序在完成工作后有效退出。

优选地，上述设备 1 中的第四装置具体用于：

指示所述第一 libvirtd 进程检查自身作为服务器端的连接，当确定当前不存在客户端与所述第一 libvirtd 进程建立有连接时，则退出所述第一 libvirtd 进程。第一 libvirtd 进程退出后，会自动断开与所述 QEMU 进程的连接。

在实际应用中，如果在第一 libvirtd 进程退出时，有客户端与第一 libvirtd 进程存在连接，那么那这些连接当前的功能调用就会失败。比如有连接到第一 libvirtd 进程的客户端在进行一些操作还未完成，此时直接退

出第一 libvirtd 进程后，那些操作就会失败了。因此，在退出之前首先确定没有客户端与第一 libvirtd 进程存在连接，能够避免特定时间点上的某些操作失效。

进一步地，所述第四装置在所述第一 libvirtd 进程检查自身作为服务器端的连接之后，若确定当前存在客户端与所述第一 libvirtd 进程建立有连接，则指示第一 libvirtd 进程在满足预设条件后重新进程检查自身作为服务器端的连接，直至确定当前不存在客户端与所述第一 libvirtd 进程建立有连接时，退出所述第一 libvirtd 进程。

例如，所述预设条件为：有新的客户端与所述第一 libvirtd 进程建立或者断开连接。假设所述第一 libvirtd 进程检查自身作为服务器端的连接，确定当前还存在两个客户端与其建立有连接，分别为 client1 和 client2，当其中一个客户端（例如 client2）与第一 libvirtd 进程断开连接时，则会触发第一 libvirtd 进程重新检查自身作为服务器端的连接。此外，当另一个客户端 client3 与第一 libvirtd 进程建立了连接后，同样会触发第一 libvirtd 进程重新检查自身作为服务器端的连接。

再如，所述预设条件为：经过设定的时间间隔。假设所述第一 libvirtd 进程检查自身作为服务器端的连接，确定当前还存在两个客户端与其建立有连接，在检查完成的同时启动计时，当计时达到设定的时间后，触发第一 libvirtd 进程重新检查自身作为服务器端的连接。

本领域技术人员应能理解上述预定条件仅为举例，其他现有的或今后可能出现的预设条件如可适用于本发明，也应包含在本发明保护范围以内，并在此以引用方式包含于此。

图 7 示出了根据本申请另一个方面的一种 libvirt 软件热升级方法的流程图。其中，所述方法包括以下步骤：

步骤 S701，完成新版本的 libvirt 软件的安装；

步骤 S702，将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程；

步骤 S703，启动所述第二 libvirtd 进程，建立所述第二 libvirtd 进程与 QEMU 进程的连接；

步骤 S704, 退出所述第一 libvirtd 进程。

在新的第二 libvirtd 进程启动后, 由于此时已经完成新版本的 libvirt 软件的安装, 第二 libvirtd 进程已经是基于新版本的软件代码运行的, 同时该第二 libvirtd 进程的虚拟机状态信息已经在启动之前更新至与第一
5 libvirtd 进程一致, 使 libvirt 软件的相关功能能够通过正确的第二 libvirtd 进程实现, 从而在升级过程中避免 libvirt 软件的相关功能中断。

本申请涉及的 libvirt 软件包括一套 API (Application Programming Interface, 应用程序接口) 库、一个守护进程(libvirtd 进程)和一个命令行工具 (virsh 工具), 用以对虚拟机监视器 (Hypervisor) 进行虚拟化管理。

10 图 2 示出了 libvirt 软件在 KVM/QEMU 虚拟化环境下的功能示意图, 在 libvirt 软件的运行过程中, virsh 工具 210 与 libvirtd 进程 220 之间的连接为短连接, 在传输一次数据至后两者之间的连接随即断开, 在该连接中 virsh 工具 210 作为客户端 (client), 而 libvirtd 进程 220 则作为服务器端 (server)。而 libvirtd 进程 220 与 QEMU 进程 230 之间的连接为长连接,
15 在传输完数据后一直保持连接, 直至收到断开该连接的请求, 在该连接中 libvirtd 进程 220 作为客户端, 而 QEMU 进程 230 则作为服务器端。

第一 libvirtd 进程和第二 libvirtd 进程可以通过修改原有的 libvirtd 进程获得, 例如在第一 libvirtd 进程和第二 libvirtd 进程中增加一标记参数, 在启动可以传入不同的标记参数, 以使两个带有不同标记参数的 libvirtd
20 进程可以共存, 并且保有各自不同的虚拟机状态信息。例如将 0 和 1 作为不同的标记参数, 则第一 libvirtd 进程和第二 libvirtd 进程则分别可以 libvirtd-0 和 libvirtd-1 来表示。libvirtd-0 和 libvirtd-1 各自创建有一个用于保存虚拟机状态信息的虚拟机状态文件 (这些文件存放在各自的虚拟机状态文件夹中, 每个虚拟机对应一个状态文件), 在 libvirtd-1 启动时, 可以
25 将 libvirtd-0 的虚拟机状态文件复制给 libvirtd-1, 以完成虚拟机状态信息的导入。

以采用 KVM/QEMU 的虚拟化方案为例, 每个虚拟机对应一个 QEMU 进程, 当 QEMU 进程在启动时, 相当于创建了一台新的虚拟机。在如图 2 所示的虚拟化环境下, libvirtd 进程作为服务器端, 可以接收来自 virsh 工

具发来的命令，并向其返回结果。同时，libvirtd 进程可以根据该来自 virsh 工具的命令进一步构造发往 QEMU 进程的命令，并作为客户端向 QEMU 进程发送该构造的命令，例如暂停/恢复虚拟机等。而 QEMU 进程在收到相关命令后，则会执行对应的操作，并将结果返回 libvirtd 进程。此外，

5 QEMU 进程还会通过 monitor 主动向连接到 monitor 的 libvirtd 进程发送事件消息 (event)，例如用户关闭虚拟机后，QEMU 进程会发送 stop event 这一事件消息给 libvirtd 进程，以更新 libvirtd 进程的虚拟机状态信息。

在 QEMU 进程启动时，会通过 libvirtd 进程传入配置参数使其提供一个 monitor，该 monitor 是 QEMU 进程与 libvirtd 进程进行交互的接口，

10 libvirtd 进程与 QEMU 进程之间的数据收发均会通过 monitor 进行。本申请为了解决现有技术的缺陷，需要配置两个相对独立的 libvirtd 进程，即第一 libvirtd 进程和第二 libvirtd 进程，那么同样需要提供两个相对独立的 monitor，分别用于与第一 libvirtd 进程、第二 libvirtd 进程保持独立的通信。本实施例中，同样可以通过传入 0 和 1 这两个不同的标记参数，来配置两

15 个 monitor，如 monitor-0 和 monitor-1，使得 libvirtd-0 与 monitor-0 建立连接，libvirtd-1 则与 monitor-1 建立连接，进行独立的通信。

在此，所述虚拟机状态信息可以通过文件、表格或者内存数据等形式保存。例如，本实施例中所述的虚拟机状态信息采用.xml 文件的形式保存，该文件以虚拟机的名称命名，如名称为 VM1 的虚拟机，其对应的虚拟机

20 状态文件为 VM1.xml。对于每个 libvirtd 进程，都有用于保存该虚拟机状态文件的文件夹，假设在升级前已经有 3 个虚拟机正在运行，其对应虚拟机状态文件为 VM1.xml、VM2.xml、VM3.xml，这些文件内记录有上述三个虚拟机的虚拟机状态信息。第二 libvirtd 进程启动之前，会将上述 3 个虚拟机状态文件拷贝至第二 libvirtd 进程的虚拟机状态文件夹中。在此，

25 本领域技术人员应能理解上述虚拟机状态信息的保存形式仅为举例，其他现有的或今后可能出现的虚拟机状态信息的保存形式如可适用于本申请，也应包含在本申请保护范围以内，并在此以引用方式包含于此。

在实际的处理过程中，将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之后，至启动所述第二 libvirtd 进程，建立所

述第二 libvirtd 进程与 QEMU 进程的连接时，会存在一定的处理时间。在该处理时间内，若发生关闭虚拟机的情形，第二 libvirtd 进程无法通过导入第一 libvirtd 进程的虚拟机状态信息获知该状态改变的信息，并且由于此时第二 libvirtd 进程还未与 QEMU 进程建立连接，QEMU 进程对应的所述关闭虚拟机的事件消息（stop event）无法送达第二 libvirtd 进程，使得第二 libvirtd 进程无法获得这一关闭虚拟机所产生的虚拟机状态信息。因此，在第二 libvirtd 进程与 QEMU 进程建立连接后，第二 libvirtd 进程的虚拟机状态信息与虚拟机的实际运行状态，造成虚拟状态错误。

为了进一步解决上述问题，本申请的提供了一种优选的实施方式，在将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之前，阻塞 QEMU 进程发往 libvirtd 进程的第一事件消息；以及在退出所述第一 libvirtd 进程之后，取消对 QEMU 进程发往 libvirtd 进程的第一事件消息的阻塞。

在此，所述第一事件消息包括在虚拟机状态改变时由 QEMU 进程发送给 libvirtd 进程的事件消息，用以告知 libvirtd 进程虚拟机状态发生了变化，使得 libvirtd 进程能够根据所述第一事件消息更新其虚拟状态信息。仍以 KVM/QEMU 的虚拟化方案为例，QEMU 进程在虚拟机状态改变时会生成相应的第一事件消息并发送给 libvirtd 进程，所述的第一事件消息可以包括：stop event、pause event 和 resume event 等，这些第一事件消息分别由关闭虚拟机、暂停虚拟机和恢复虚拟机等操作生成。由于 QEMU 进程中配置了两个 monitor，因此第一事件消息生成后会分别通过两个 monitor 向与该 monitor 对应的 libvirtd 进程发送，当 libvirtd 进程与 monitor 之间的连接正常时，第一事件消息即会送达 libvirtd 进程。在本例中，若第一事件消息产生时，第二 libvirtd 进程还未与 QEMU 进程建立连接，该第一事件消息的无法送达。

当阻塞 QEMU 进程发往 libvirtd 进程的第一事件消息之后，上述第一事件消息在产生后，不会向任何一个 libvirtd 进程发送，而是由 QEMU 进程保存在内部。当退出所述第一 libvirtd 进程后，再取消对 QEMU 进程发往 libvirtd 进程的第一事件消息的阻塞。取消阻塞后，所述 QEMU 进程发

往所述 libvirtd 进程的第一事件消息可能包括：1、被阻塞而保存在 QEMU 进程内部的第一事件消息，如果取消阻塞之前，有第一事件消息产生并被阻塞，那么取消阻塞后会继续发送这些第一事件消息；2、取消阻塞后新产生的第一事件消息。由于此时第二 libvirtd 进程与 QEMU 进程已经建立连接，第二 libvirtd 进程仍然可以收到将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之后，至启动所述第二 libvirtd 进程，建立所述第二 libvirtd 进程与 QEMU 进程的连接之前所产生的第一事件消息，由此更新自身的虚拟机状态信息，以避免虚拟机状态错误。其中，上述的 libvirtd 进程表示第一 libvirtd 进程与第二 libvirtd 进程的总称，即 QEMU 进程发往 libvirtd 进程的第一事件消息是同时发往第一 libvirtd 进程与第二 libvirtd 进程，只要第一 libvirtd 进程与第二 libvirtd 进程处于启动状态，并且与 QEMU 进程的 monitor 建立了连接，就可以收到所述第一事件消息；同理，进行阻塞后，第一事件消息不会发往第一 libvirtd 进程和第二 libvirtd 进程中的任何一个。

具体地，所述阻塞 QEMU 进程发往当前运行的第一 libvirtd 进程的第一事件消息，包括：由当前运行的所述第一 libvirtd 进程向所述 QEMU 进程发送第一命令，所述 QEMU 进程根据所述第一命令中止向 libvirtd 进程发送所述第一事件消息。取消对 QEMU 进程发往 libvirtd 进程的第一事件消息的阻塞，包括：由当前运行的所述第二 libvirtd 进程向所述 QEMU 进程发送第二命令，所述 QEMU 进程根据所述第二命令恢复向 libvirtd 进程发送所述第一事件消息。接上例，所述第一命令和第二命令可以采用 KVM/QEMU 的虚拟化方案中常用的 QEMU Monitor Command（QEMU 监控命令）来实现，添加分别用于阻塞和取消阻塞的 QEMU Monitor Command，并且通过 QEMU 进程，使其能够在接到相应的 QEMU Monitor Command 后进行阻塞或取消阻塞。

本领域技术人员应能理解上述第一命令和第二命令仅为举例，其他现有的或今后可能出现的能够控制 QEMU 进程中止或者恢复向 libvirtd 进程发送所述第一事件消息的命令如可适用于本申请，也应包含在本申请保护范围以内，并在此以引用方式包含于此。

优选地，所述启动所述第二 libvirtd 进程，具体包括：启动所述第二 libvirtd 进程时，所述第二 libvirtd 进程创建一个对应于所述第二 libvirtd 进程的连接路径，其中，所述对应于所述第二 libvirtd 进程的连接路径用于建立客户端与所述第二 libvirtd 进程的连接。客户端在所述第二 libvirtd 进程启动后，根据 libvirtd 进程的连接路径的创建时间，选择对应于所述第二 libvirtd 进程的连接路径与所述第二 libvirtd 进程建立连接。其中，所述客户端除了配置有 virsh 工具的设备之外，还可以是其它尝试与 libvirtd 进程进行连接的设备或者程序。

在此，对于两个先后创建的连接路径，优先选择最近创建的连接路径。

接上例，所述连接路径可以是第二 libvirtd 进程作为服务器端的 domain socket（域套接字）路径，该 domain socket 路径会在 libvirtd 进程启动时自动创建。仍以前述两个 libvirtd 进程 libvirtd-0 和 libvirtd-1 为例，两个 libvirtd 进程能够拥有各自独立的 domain socket 路径，在 libvirtd-0 和 libvirtd-1 启动时，所创建的 domain socket 路径分别为 /var/run/libvirt/libvirt-sock-1 和 /var/run/libvirt/libvirt-sock-2。假设 libvirtd-0 为在版本升级前开始运行的 libvirtd 进程，而 libvirtd-1 为新启动的 libvirtd 进程，libvirtd-0 中作为服务器端的 domain socket 路径的创建时间为 t1，而 libvirtd-1 中作为服务器端的 domain socket 路径的创建时间为 t2，t2 晚于 t1。若两个 libvirtd 进程同时运行，客户端在尝试连接 libvirtd 进程时（例如配置有 virsh 工具的设备向 libvirtd 进程发送命令时）会发现同时存在两个服务器端的 domain socket 路径，此时客户端会根据 libvirtd 进程的 domain socket 路径的创建时间，优先选择最近创建的 domain socket 路径来建立连接。如实施例 2 中，由于 t2 晚于 t1，客户端会优先选择与 libvirtd-1 建立连接，那么此时客户端向 libvirtd 进程发送的命令只会发送到 libvirtd-1。

由此，在两个 libvirtd 进程同时运行时，若配置有 virsh 工具（或者其它采用 C 或者 python 语言编写的管理工具程序）的设备需要对虚拟机进行配置，其配置命令会发送至新启动的第二 libvirtd 进程。

在第二 libvirtd 进程还未启动时，尝试连接 libvirtd 进程的客户端仍然

会与原来的第一 libvirtd 进程连接。在本实施例涉及的创建新虚拟机的操作需要由配置有 virsh 工具的设备向 libvirtd 进程发送命令，再由 libvirtd 进程根据该命令对 QEMU 进程配置，因此，在将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之后，至启动所述第二 libvirtd 进程之前的这一段时间内，第二 libvirtd 进程还未完成启动，若在此时创建新虚拟机，是通过原来的第一 libvirtd 进程完成，则第二 libvirtd 进程无法更新该新创建的虚拟机状态信息，可能会造成新创建的虚拟机状态信息被遗漏。

针对上述问题，本申请进一步提供了一种优选的实施方式，通过增加一获取并添加新产生的虚拟机状态信息的步骤来实现。该步骤为：获取所述第一 libvirtd 进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirtd 进程的虚拟机状态信息内，其中，所述第一时间段为将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程至启动所述第二 libvirtd 进程之间的时间区间。

由于在第一时间段之前创建新虚拟机所生成的虚拟机状态信息可以通过导入的方式写入第二 libvirtd 进程的虚拟机状态文件；而第一时间段之后则是直接通过第二 libvirtd 进程进行新虚拟机的创建，因此第二 libvirtd 进程直接可以获知此时的创建新虚拟机所生成的虚拟机状态信息。本申请通过获取所述第一 libvirtd 进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二 libvirtd 进程的虚拟机状态信息内，以此来避免新创建的虚拟机状态信息被遗漏。

进一步地，结合图 7，图 8 示出了本申请实施例提供的一种优选的 libvirt 软件热升级方法的流程图，其中前述获取并添加新产生的虚拟机状态信息的步骤具体包括以下步骤：

步骤 S801，在将当前运行的第一 libvirtd 进程的虚拟机状态信息导入至第二 libvirtd 进程之前，启动监听程序，该监听程序与所述第一 libvirtd 进程建立连接；

步骤 S802, 所述监听程序获取由所述第一 libvirtd 进程在所述第一时间段内发出的第二事件消息, 其中, 所述第二事件消息指向所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息;

5 步骤 S803, 所述监听程序建立与所述第二 libvirtd 进程的连接, 并根据所述第二事件消息向所述第二 libvirtd 进程添加所述新产生的虚拟机状态信息。

在此, 所述监听程序与所述第一 libvirtd 进程和第二 libvirtd 进程均会建立连接, 为了便于在该场景下发送相关的数据, 两个连接均为长连接, 并且相对于 libvirtd 进程, 该监听程序是作为客户端。例如, 虚拟机状态
10 信息是通过文件的形式保存, 新创建的虚拟机的名称为 VM4 和 VM5, 其虚拟机状态信息保存于 VM4.xml 和 VM5.xml 文件中。对应地, 为了能够使第二事件消息指向所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息, 第二事件消息需要包含的信息至少包括创建了新的虚拟机这一事件, 以及新创建的虚拟机的标识 (例如虚拟机的名称)。例如, 通过
15 VM5 这个虚拟机名称, 即可找到 VM5.xml 这个保存有虚拟机状态信息的文件。

在实际应用中, 除创建虚拟机之外的操作, 例如关闭虚拟机, 也可能由 virsh 工具发起并通过 libvirtd 进程关闭该虚拟机对应的 QEMU 进程。此时, 监听程序也会监听到这些操作 (例如关闭虚拟机) 所对应的事件消
20 息, 但是针对这些无关的事件消息, 监听程序会忽略掉, 不作处理。

进一步地, 结合图 7、图 8, 图 9 示出了一种进一步优选的 libvirt 软件热升级方法的流程图, 上述步骤 S803 的具体处理步骤, 包括:

步骤 S901, 所述监听程序建立与所述第二 libvirtd 进程之间的连接。在此, 监听程序会在所述第二 libvirtd 进程启动之前等待该第二 libvirtd 进
25 程启动, 在其启动成功后即会自动与第二 libvirtd 进程建立连接。

步骤 S902, 所述监听程序将保存有所述第一 libvirtd 进程创建新虚拟机所新产生的虚拟机状态信息的状态文件拷贝至所述第二 libvirtd 进程对应的虚拟机状态文件夹中。接上例, 对于新创建的虚拟机 VM4 和 VM5, 其虚拟机状态信息保存在 VM4.xml 和 VM5.xml 这两个状态文件中, 监听

程序会将这两个文件拷贝至第二 libvirtd 进程对应的虚拟机状态文件夹中。

步骤 S903, 所述监听程序通过所述连接调用所述第二 libvirtd 进程的导入接口, 所述导入接口根据所述第二事件消息, 将所述状态文件中将所述新产生的虚拟机状态信息导入至所述第二 libvirtd 进程, 并持久化的存放在对应的虚拟机状态文件夹的虚拟机状态文件。在此, 第二事件消息中包含了事件由于第二事件消息中包含了虚拟机的标识, 导入接口可以根据虚拟机的标识, 例如通过虚拟机名称 VM5 找到 VM5.xml 这个保存有虚拟机状态信息的状态文件, 将这个文件中将虚拟机状态信息导入第二 libvirtd 进程。

作为另一种优选的实施方式, 上述方法中在启动所述第二 libvirtd 进程之后, 所述监听程序退出。例如, 所述第二 libvirtd 进程启动之后, 由于监听程序所提供的功能已经没有存在的必要, 会通过某一特定信号来对监听程序进行指示, 使得监听程序退出。

具体地, 所述监听程序与所述第一 libvirtd 进程之间的连接为只读连接; 所述监听程序退出, 具体包括: 所述第一 libvirtd 进程检查自身作为服务器端的连接, 当确定当前仅存在一个客户端与所述第一 libvirtd 进程建立连接并且该连接是只读连接时, 则指示所述监听程序退出, 实现监听程序在完成工作后有效退出。

优选地, 所述退出所述第一 libvirtd 进程, 具体包括: 所述第一 libvirtd 进程检查自身作为服务器端的连接, 当确定当前不存在客户端与所述第一 libvirtd 进程建立有连接时, 则退出所述第一 libvirtd 进程。第一 libvirtd 进程退出后, 会自动断开与所述 QEMU 进程的连接。

在实际应用中, 如果在第一 libvirtd 进程退出上, 有客户端与第一 libvirtd 进程存在连接, 那么那这些连接当前的功能调用就会失败。比如有连接到第一 libvirtd 进程的客户端在进行一些操作还未完成, 此时直接退出第一 libvirtd 进程后, 那些操作就会失败了。因此, 在退出之前首先确定没有客户端与第一 libvirtd 进程存在连接, 能够避免特定时间点上的某些操作失效。

进一步地, 所述第一 libvirtd 进程检查自身作为服务器端的连接之后,

还包括：当确定当前存在客户端与所述第一 libvirtd 进程建立有连接时，则第一 libvirtd 进程在满足预设条件后重新检查自身作为服务器端的连接，直至确定当前不存在客户端与所述第一 libvirtd 进程建立有连接时，退出所述第一 libvirtd 进程。

- 5 例如，所述预设条件为：有新的客户端与所述第一 libvirtd 进程建立或者断开连接。假设所述第一 libvirtd 进程检查自身作为服务器端的连接，确定当前还存在两个客户端与其建立有连接，分别为 client1 和 client2，当其中一个客户端（例如 client2）与第一 libvirtd 进程断开连接时，则会触发第一 libvirtd 进程重新检查自身作为服务器端的连接。此外，当另一个
- 10 客户端 client3 与第一 libvirtd 进程建立了连接后，同样会触发第一 libvirtd 进程重新检查自身作为服务器端的连接。

- 再如，所述预设条件为：经过设定的时间间隔。假设所述第一 libvirtd 进程检查自身作为服务器端的连接，确定当前还存在两个客户端与其建立有连接，在检查完成的同时启动计时，当计时达到设定的时间后，触发第
- 15 一 libvirtd 进程重新检查自身作为服务器端的连接。

本领域技术人员应能理解上述预定条件仅为举例，其他现有的或今后可能出现的预设条件如可适用于本发明，也应包含在本发明保护范围以内，并在此以引用方式包含于此。

- 图 10（a）至图 10（f）示出了应用本申请提供的方法对 KVM/QEMU
- 20 虚拟化环境下的 libvirtd 软件进行热升级过程中交互原理图。在实际应用中，为实现本申请的方案，至少需要在现有 libvirt 软件的基础上对其软件代码进行如下改动：

- 1、为每个 libvirtd 进程增加一个标记参数，启动时可以传入标记参数 0 或者 1，使带 0 和 1 标记的两个 libvirtd 可以共存，分别记为 libvirtd-0
- 25 和 libvirtd-1。两个 libvirtd 进程有各自的保存虚拟机状态信息的文件，各自的作为服务器端的 domain socket 路径。

2、libvirtd 进程在创建虚拟机配置 QEMU 进程时配置两个 monitor，分别传入不同的标记参数，即 monitor-0，monitor-1，libvirtd-0 会连接 monitor-0，libvirtd-1 会连接 monitor-1。

3、客户端（如 virsh 工具）连接 libvirtd 进程时，如果存在两个 domain socket 路径，则优先连接最近创建那个 domain socket 路径。

4、为 libvirtd 进程增加一个 SIG_USR1 信号处理函数，在收到 SIG_USR1 信号时，检查其作为服务器端的连接，在当前 client 时连接只有一个只读连接时往指定程序发送 SIG_TERM 信号，在当前没有连接时进程自动退出。

5、修改 QEMU 进程，增加可以 pending（阻塞）或者取消 pending 发往 libvirtd 进程的 event 的功能，添加相应的 QEMU Monitor Command 来控制 QEMU 进程进行 pending 或者取消 pending。

6、为 libvirtd 进程增加一导入接口，通过调用该接口可以从另一 libvirtd 进程的虚拟机状态文件中读取并导入虚拟机状态信息。

假设在当前情况下，libvirtd-0 处于运行状态，libvirtd-1 未启动，如图 10（a）所示。Virsh 会在需要时，作为客户端与 libvirtd-0 建立短连接，发送相关命令。在此，图中所示的 Virsh 应当理解为一个或者同一类用以实现类似功能的设备或程序，并不以图中所示的数量为限，其功能参考图 2 中的 virsh 工具 210。QEMU 进程能够通过配置的 monitor 向 libvirtd 进程发 event。任一 event 均会通过 monitor-0 和 monitor-1 分别向 libvirtd-0 和 libvirtd-1 发送 event。此时，由于 libvirtd-1 未启动，则 event 仅会被发送至 libvirtd-0。

进行热升级时可以采用如下的处理流程：

1、安装新版本的 libvirt 软件。

2、pending QEMU 进程发往 libvirtd 进程的 event。图 10（b）示出了此时的数据交互图。由于 QEMU 进程 pending 其发往 libvirtd 进程的 event，原本需要发送的 event 会被保存在 QEMU 进程内部，暂停发送。

3、启动一监听程序 C1，并且以只读方式连接（长连接）当前运行的 libvirtd-0，对 libvirtd-0 进行监听。图 10（c）示出了此时的数据交互图，当通过 libvirtd-0 创建新的虚拟机时，libvirtd-0 会发送相应的 event 给监听程序 C1。

4、将 libvirtd-0 的虚拟机状态信息导入到 libvirtd-1，然后启动

libvirtd-1。图 10 (d) 示出了此时的数据交互图，由于 libvirtd-1 启动时，会创建 domain socket 路径，当 libvirtd-0 和 libvirtd-1 同时运行时，客户端在尝试建立连接时，会优先连接至最近创建的 domain socket 路径。因此，libvirtd-1 启动后尝试建立短连接的 Virsh 均会自动连接至 libvirtd-1。

5 同时，如果在将 libvirtd-0 的虚拟机状态信息导入到 libvirtd-1 后，还有通过 libvirtd-0 的连接创建了新的虚拟机，则上一步启动的监听程序会监听到对应的 event，监听程序会通过 libvirtd-1 的导入接口将新产生的虚拟机状态信息导入到 libvirtd-1 中。

10 5、发送 SIG_USR1 信号到 libvirtd-0，libvirtd-0 在收到后对其作为服务器端的连接进行检查，当只有一个客户端与其建立有只读连接时，发送一个 SIG_TERM 信号给监听程序，监听程序退出；当没有客户端与其连接时，Libvirtd-0 退出。Libvirtd-0 退出后，会自动断开其作为客户端的连接（与 monitor-0 的连接）。图 10 (e) 示出了此时的数据交互图。

15 6、取消 pending QEMU 进程发往 libvirtd 进程的 event。图 10 (f) 示出了此时的数据交互图，对于 pending 后保存在内部、暂停发送的那些 event，恢复发送，后续产生的 event 也可以正常进行发送。

此时由于 libvirtd-1 已经是基于更新后的 libvirtd 软件代码运行，即已经完成了整个升级过程。此后，仅需要重复上述过程，启动 libvirtd-0 并退出 libvirtd-1，即可实现再一次热升级。

20 综上所述，在新的第二 libvirtd 进程启动后，由于此时已经完成新版本的 libvirt 软件的安装，第二 libvirtd 进程已经是基于新版本的软件代码运行的，同时该第二 libvirtd 进程的虚拟机状态信息已经在启动之前更新至与第一 libvirtd 进程一致，使 libvirt 软件的相关功能能够通过正确的第二 libvirtd 进程实现，从而在升级过程中避免 libvirt 软件的相关功能中断。

25 此外，本申请中通过在特定时间对 QEMU 进程发往 libvirtd 进程的第一事件消息进行阻塞或者取消阻塞，以及通过监听程序对第一时间段内通过第一 libvirtd 进程创建新虚拟机的操作进行监听，避免第二 libvirtd 进程的虚拟机状态信息发生错误。

显然，本领域的技术人员可以对本申请进行各种改动和变型而不脱离

本申请的精神和范围。这样，倘若本申请的这些修改和变型属于本申请权利要求及其等同技术的范围之内，则本申请也意图包含这些改动和变型在内。

需要注意的是，本申请可在软件和/或软件与硬件的组合体中被实施，
5 例如，可采用专用集成电路（ASIC）、通用目的计算机或任何其他类似硬件设备来实现。在一个实施例中，本申请的软件程序可以通过处理器执行以实现上文所述步骤或功能。同样地，本申请的软件程序（包括相关的数据结构）可以被存储到计算机可读记录介质中，例如，RAM 存储器，磁或光驱动器或软磁盘及类似设备。另外，本申请的一些步骤或功能可采用
10 硬件来实现，例如，作为与处理器配合从而执行各个步骤或功能的电路。

另外，本申请的一部分可被应用为计算机程序产品，例如计算机程序指令，当其被计算机执行时，通过该计算机的操作，可以调用或提供根据本申请的方法和/或技术方案。而调用本申请的方法的程序指令，可能被存储在固定的或可移动的记录介质中，和/或通过广播或其他信号承载媒体中的
15 的数据流而被传输，和/或被存储在根据所述程序指令运行的计算机设备的工作存储器中。在此，根据本申请的一个实施例包括一个装置，该装置包括用于存储计算机程序指令的存储器和用于执行程序指令的处理器，其中，当该计算机程序指令被该处理器执行时，触发该装置运行基于前述根据本申请的多个实施例的方法和/或技术方案。

20 对于本领域技术人员而言，显然本申请不限于上述示范性实施例的细节，而且在不背离本申请的精神或基本特征的情况下，能够以其他的具体形式实现本申请。因此，无论从哪一点来看，均应将实施例看作是示范性的，而且是非限制性的，本申请的范围由所附权利要求而不是上述说明限定，因此旨在将落在权利要求的等同要件的含义和范围内的所有变化涵括
25 在本申请内。不应将权利要求中的任何附图标记视为限制所涉及的权利要求。此外，显然“包括”一词不排除其他单元或步骤，单数不排除复数。装置权利要求中陈述的多个单元或装置也可以由一个单元或装置通过软件或者硬件来实现。第一，第二等词语用来表示名称，而并不表示任何特定的顺序。

权利要求书

1. 一种利博伟特软件热升级方法，其中，所述方法包括：
完成新版本的利博伟特软件的安装；

5 将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程，其中，所述第一利博伟德进程基于旧版本的利博伟特软件运行，所述第二利博伟德进程基于所述新版本的利博伟特软件运行；

启动所述第二利博伟德进程，建立所述第二利博伟德进程与机器虚拟机进程的连接；

退出所述第一利博伟德进程。

10 2. 根据权利要求 1 所述的方法，其中，在将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程之前，还包括：

阻塞机器虚拟机进程发往利博伟德进程的第一事件消息，其中，所述第一事件消息包括在虚拟机状态改变时由机器虚拟机进程发送给利博伟德进程的事件消息；以及

15 在退出所述第一利博伟德进程之后，还包括：

取消对机器虚拟机进程发往利博伟德进程的第一事件消息的阻塞。

3. 根据权利要求 2 所述的方法，其中，所述阻塞机器虚拟机进程发往当前运行的第一利博伟德进程的第一事件消息，包括：

20 由当前运行的所述第一利博伟德进程向所述机器虚拟机进程发送第一命令，所述机器虚拟机进程根据所述第一命令中止向利博伟德进程发送所述第一事件消息；

取消对机器虚拟机进程发往利博伟德进程的第一事件消息的阻塞，包括：

25 由当前运行的所述第二利博伟德进程向所述机器虚拟机进程发送第二命令，所述机器虚拟机进程根据所述第二命令恢复向利博伟德进程发送所述第一事件消息。

4. 根据权利要求 1 至 3 任一项所述的方法，其中，启动所述第二利博伟德进程，包括：

启动所述第二利博伟德进程时，所述第二利博伟德进程创建一个对应于所述第二利博伟德进程的连接路径，其中，所述对应于所述第二利博伟德进程连接路径用于建立客户端与所述第二利博伟德进程的连接；

退出所述第一利博伟德进程之前，还包括：

- 5 客户端在所述第二利博伟德进程启动后，根据利博伟德进程的连接路径的创建时间，选择对应于所述第二利博伟德进程的连接路径与所述第二利博伟德进程建立连接。

5.根据权利要求 1 至 4 任一项所述的方法，其中，在启动所述第二利博伟德进程之前，还包括：

- 10 获取所述第一利博伟德进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二利博伟德进程的虚拟机状态信息内，其中，所述第一时间段为将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程至启动所述第二利博伟德进程之间的时间区间。

- 15 6. 根据权利要求 5 所述的方法，其中，获取所述第一利博伟德进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二利博伟德进程的虚拟机状态信息内，包括：

- 20 在将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程之前，启动监听程序，该监听程序与所述第一利博伟德进程建立连接；

所述监听程序获取由所述第一利博伟德进程在所述第一时间段内发出的第二事件消息，其中，所述第二事件消息指向所述第一利博伟德进程创建新虚拟机所新产生的虚拟机状态信息；

- 25 所述监听程序建立与所述第二利博伟德进程的连接，并根据所述第二事件消息向所述第二利博伟德进程添加所述新产生的虚拟机状态信息。

7. 根据权利要求 6 所述的方法，其中，所述监听程序建立与所述第二利博伟德进程的连接，并向所述根据所述第二事件消息向所述第二利博伟德进程添加所述新产生的虚拟机状态信息，包括：

所述监听程序建立与所述第二利博伟德进程的连接；

所述监听程序将保存有所述第一利博伟德进程创建新虚拟机所新产生的虚拟机状态信息的状态文件拷贝至所述第二利博伟德进程对应的虚拟机状态文件夹中；

5 所述监听程序通过所述连接调用所述第二利博伟德进程的导入接口，所述导入接口根据所述第二事件消息，将所述状态文件中将所述新产生的虚拟机状态信息导入至所述第二利博伟德进程。

8. 根据权利要求 6 或 7 所述的方法，其中，启动所述第二利博伟德进程之后，所述监听程序退出。

10 9. 根据权利要求 8 所述的方法，其中，所述监听程序与所述第一利博伟德进程的连接为只读连接；

所述监听程序退出，包括：

所述第一利博伟德进程检查自身作为服务器端的连接，当确定当前仅存在一个客户端与所述第一利博伟德进程建立连接并且该连接是只读连
15 接时，则指示所述监听程序退出。

10. 根据权利要求 1 至 9 任一项所述的方法，其中，退出所述第一利博伟德进程，包括：

所述第一利博伟德进程检查自身作为服务器端的连接，当确定当前不存在客户端与所述第一利博伟德进程建立有连接时，则退出所述第一利博
20 伟德进程。

11. 根据权利要求 10 所述的方法，其中，所述第一利博伟德进程检查自身作为服务器端的连接之后，还包括：

当确定当前存在客户端与所述第一利博伟德进程建立有连接时，则第一利博伟德进程在满足预设条件后重新检查自身作为服务器端的连接，直至
25 确定当前不存在客户端与所述第一利博伟德进程建立有连接时，退出所述
所述第一利博伟德进程。

12. 根据权利要求 11 所述的方法，其中，所述预设条件包括：

有新的客户端与所述第一利博伟德进程建立或者断开连接；或

经过设定的时间间隔。

13. 一种利博伟特软件热升级设备，其中，所述设备包括：

第一装置，用于完成新版本的利博伟特软件的安装；

5 第二装置，用于将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程，其中，所述第一利博伟德进程基于旧版本的利博伟特软件运行，所述第二利博伟德进程基于所述新版本的 libvirt 软件运行；

第三装置，用于启动所述第二利博伟德进程，建立所述第二利博伟德进程与机器虚拟机进程的连接；

第四装置，用于退出所述第一利博伟德进程。

10 14. 根据权利要求 13 所述的设备，其中，所述设备还包括：

第五装置，用于在将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程之前，阻塞机器虚拟机进程发往利博伟德进程的第一事件消息，其中，所述第一事件消息包括在虚拟机状态信息改变时由机器虚拟机进程发送给利博伟德进程的事件消息；以及

15 用于在退出所述第一利博伟德进程之后，取消对机器虚拟机进程发往利博伟德进程的第一事件消息的阻塞。

15. 根据权利要求 14 所述的设备，其中，所述第五装置用于：

20 在将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程之前，指示当前运行的所述第一利博伟德进程向所述机器虚拟机进程发送第一命令，所述机器虚拟机进程根据所述第一命令中止向利博伟德进程发送所述第一事件消息；以及

在退出所述第一利博伟德进程之后，指示当前运行的所述第二利博伟德进程向所述机器虚拟机进程发送第二命令，所述机器虚拟机进程根据所述第二命令恢复向第二利博伟德进程发送所述第一事件消息。

25 16. 根据权利要求 13 至 15 任一项所述的设备，其中，所述第三装置，用于：

启动所述第二利博伟德进程时创，所述第二利博伟德进程建一个对应于所述第二利博伟德进程的连接路径，其中，所述对应于所述第二利博伟

德进程连接路径用于建立客户端与所述第二利博伟德进程的连接；

所述设备还包括：

第六装置，用于在退出所述第一利博伟德进程之前，指示客户端在所述第二利博伟德进程启动后，根据利博伟德进程的连接路径的创建时间，
5 选择对应于所述第二利博伟德进程的连接路径与所述第二利博伟德进程建立连接。

17. 根据权利要求 13 至 16 任一项所述的设备，其中，所述设备还包括：

10 第七装置，用于在启动所述第二利博伟德进程之前，获取所述第一利博伟德进程在第一时间段内创建新虚拟机所新产生的虚拟机状态信息，并将新产生的虚拟机状态信息添加至所述第二利博伟德进程的虚拟机状态信息内，其中，所述第一时间段为将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程至启动所述第二利博伟德进程之间的时间区间。

15 18. 根据权利要求 17 所述的设备，其中，所述第七装置包括：

第七一模块，用于在将当前运行的第一利博伟德进程的虚拟机状态信息导入至第二利博伟德进程之前启动一监听程序，该监听程序与所述第一利博伟德进程建立连接；

20 第七二模块，用于控制所述监听程序获取由所述第一利博伟德进程在所述第一时间段内发出的第二事件消息，其中，所述第二事件消息指向所述第一利博伟德进程创建新虚拟机所新产生的虚拟机状态信息；

第七三模块，用于控制所述监听程序建立与所述第二利博伟德进程的连接，并根据所述第二事件消息向所述第二利博伟德进程添加所述新产生的虚拟机状态信息。

25 19. 根据权利要求 18 所述的设备，其中，所述第七三模块，包括：

第七三一单元，用于控制所述监听程序建立与所述第二利博伟德进程的连接；

第七三二单元，用于控制所述监听程序将保存有所述第一利博伟德进

程创建新虚拟机所新产生的虚拟机状态信息的状态文件拷贝至所述第二利博伟德进程对应的虚拟机状态文件夹中；

第七三三单元，用于控制所述监听程序通过所述连接调用所述第二利博伟德进程的导入接口，及控制所述导入接口根据所述第二事件消息，将
5 所述状态文件中将所述新产生的虚拟机状态信息导入至所述第二利博伟德进程。

20. 根据权利要求 18 或 19 所述的设备，其中，所述设备还包括第八装置，用于在启动所述第二利博伟德进程之后，指示所述监听程序退出。

21. 根据权利要求 20 所述的设备，其中，所述监听程序与所述第一利
10 博伟德进程的连接为只读连接；

所述第八装置，用于在启动所述第二利博伟德进程之后，指示所述第一利博伟德进程检查自身作为服务器端的连接，当确定当前仅存在一个客户端与所述第一利博伟德进程建立连接并且该连接是只读连接时，则指示所述监听程序退出。

15 22. 根据权利要求 13 至 21 任一项所述的设备，其中，所述第四装置，用于：

指示所述第一利博伟德进程检查自身作为服务器端的连接，当确定当前不存在客户端与所述第一利博伟德进程建立有连接时，则退出所述第一利博伟德进程。

20 23. 根据权利要求 22 所述的设备，其中，所述第四装置，还用于：

在所述第一利博伟德进程检查自身作为服务器端的连接之后，当确定当前存在客户端与所述第一利博伟德进程建立有连接时，则指示第一利博伟德进程在满足预设条件后重新检查自身作为服务器端的连接，直至确定当前不存在客户端与所述第一利博伟德进程建立有连接时，退出所述第一
25 利博伟德进程。

24. 根据权利要求 23 所述的设备，其中，所述预设条件包括：

有新的客户端与所述第一利博伟德进程建立或者断开连接；或
经过设定的时间间隔。

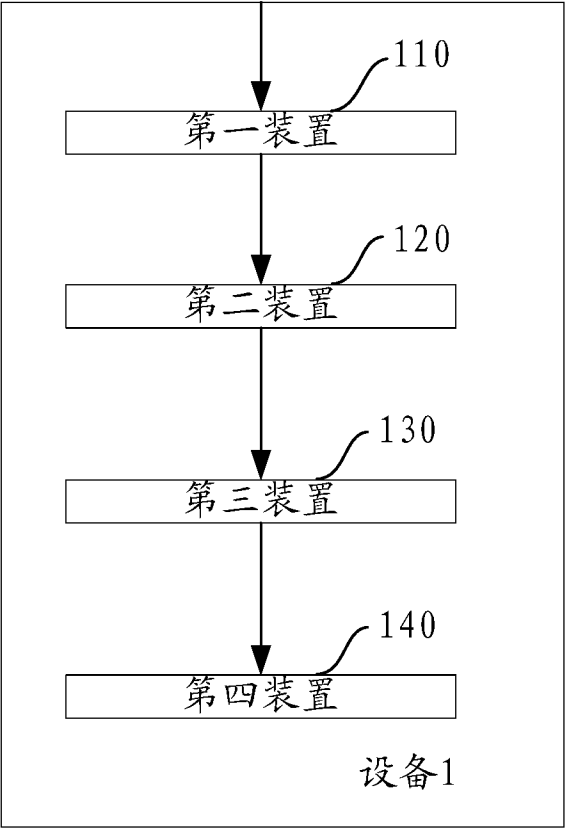


图 1

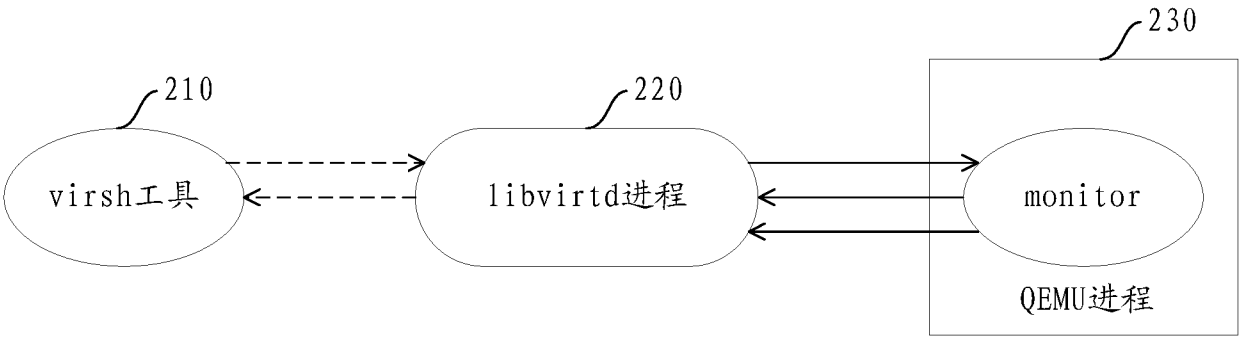


图 2

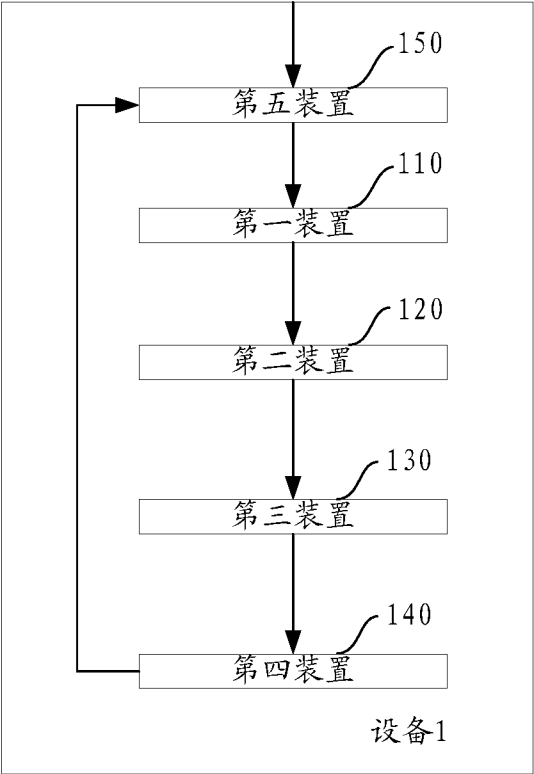


图 3

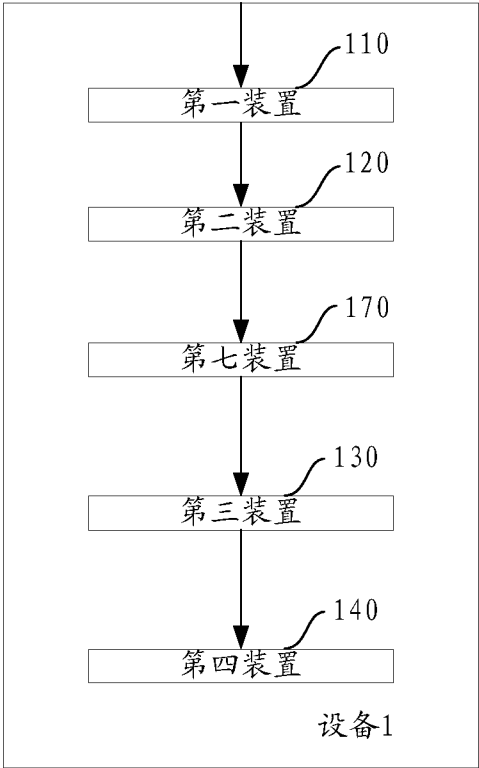


图 4

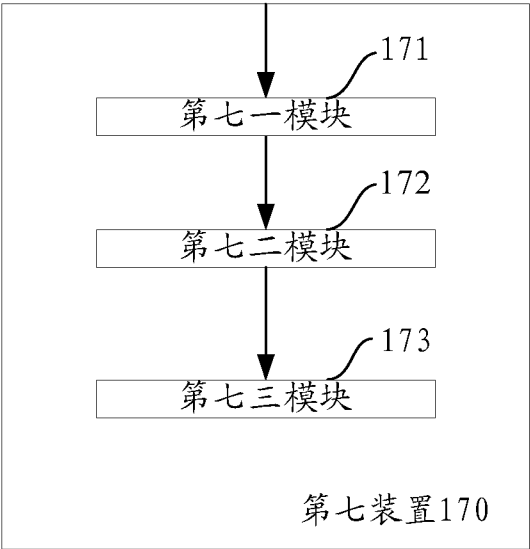


图 5

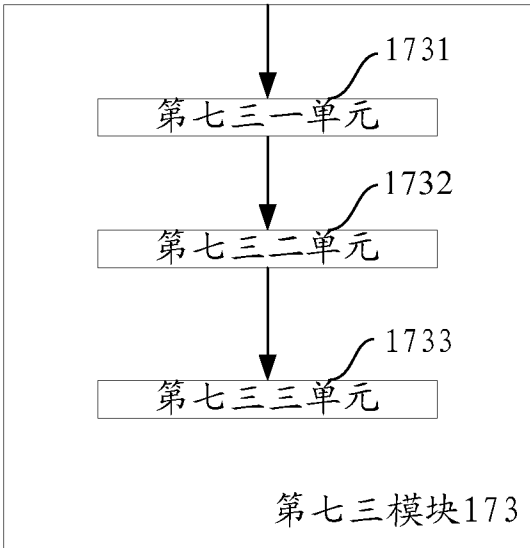


图 6

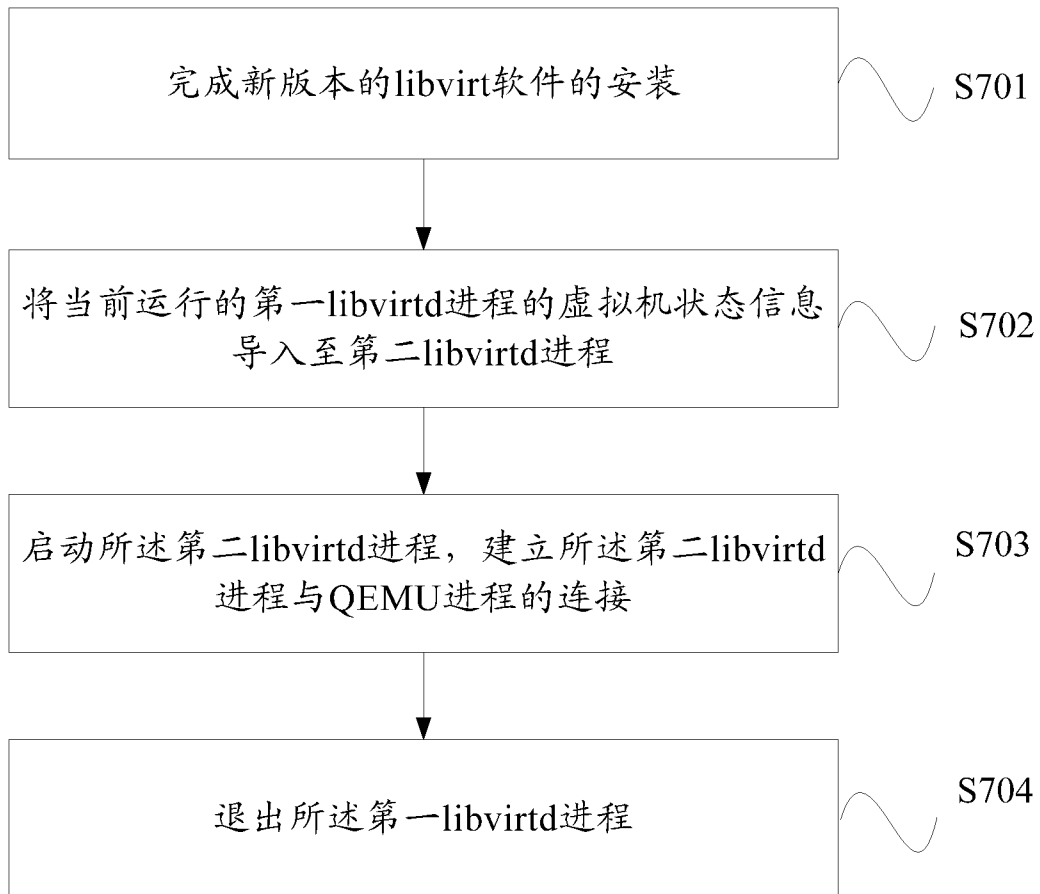


图 7

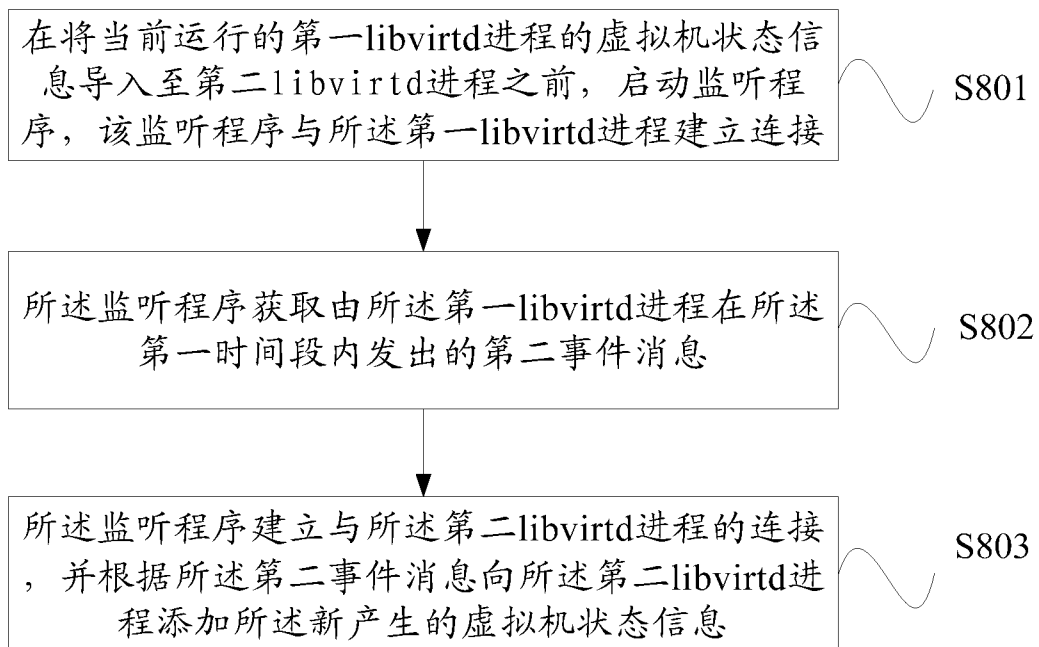


图 8

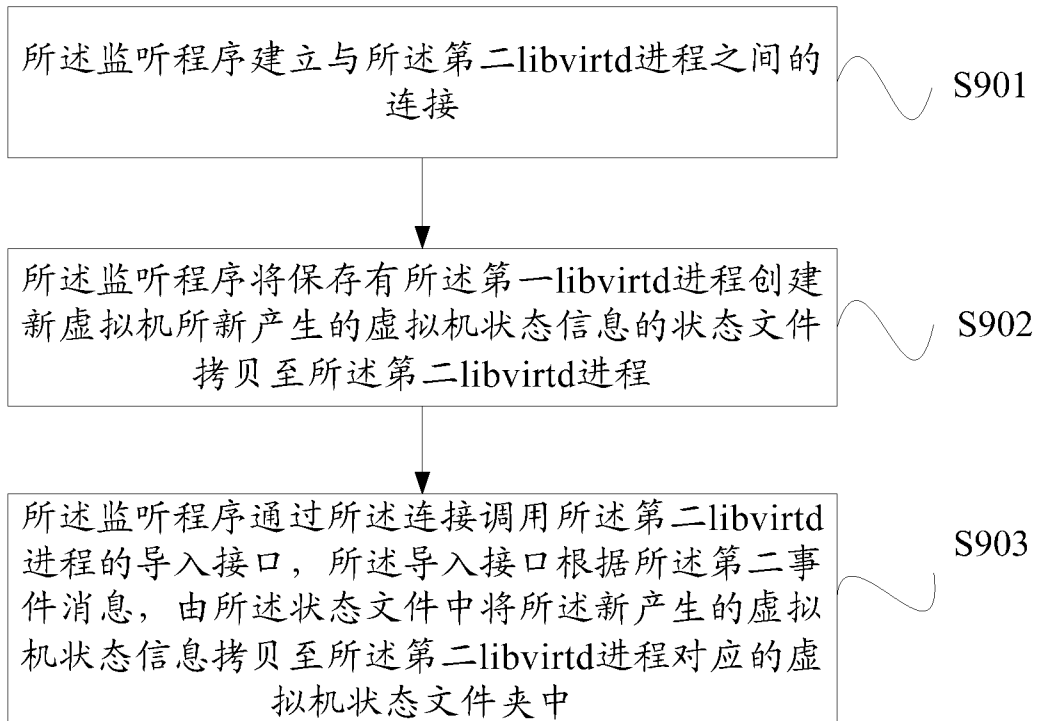


图 9

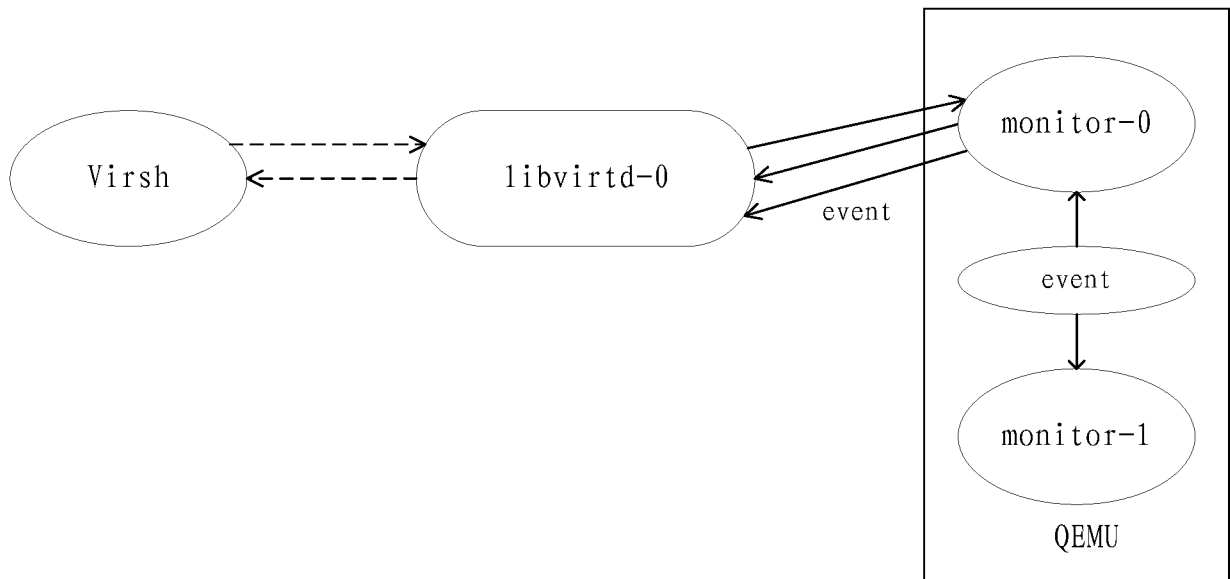


图 10 (a)

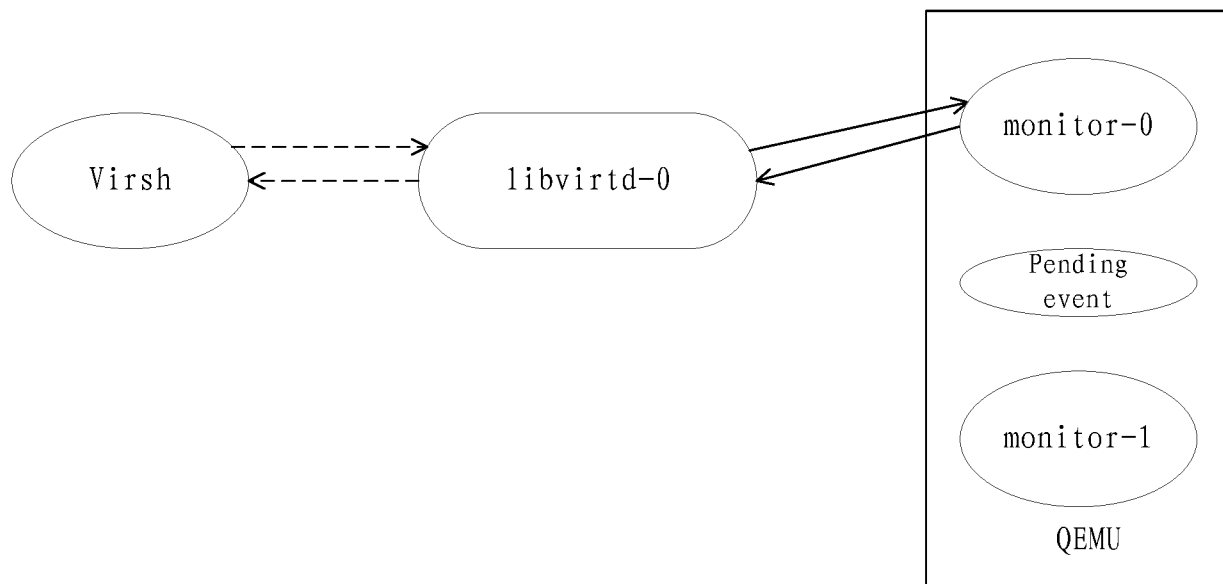


图 10 (b)

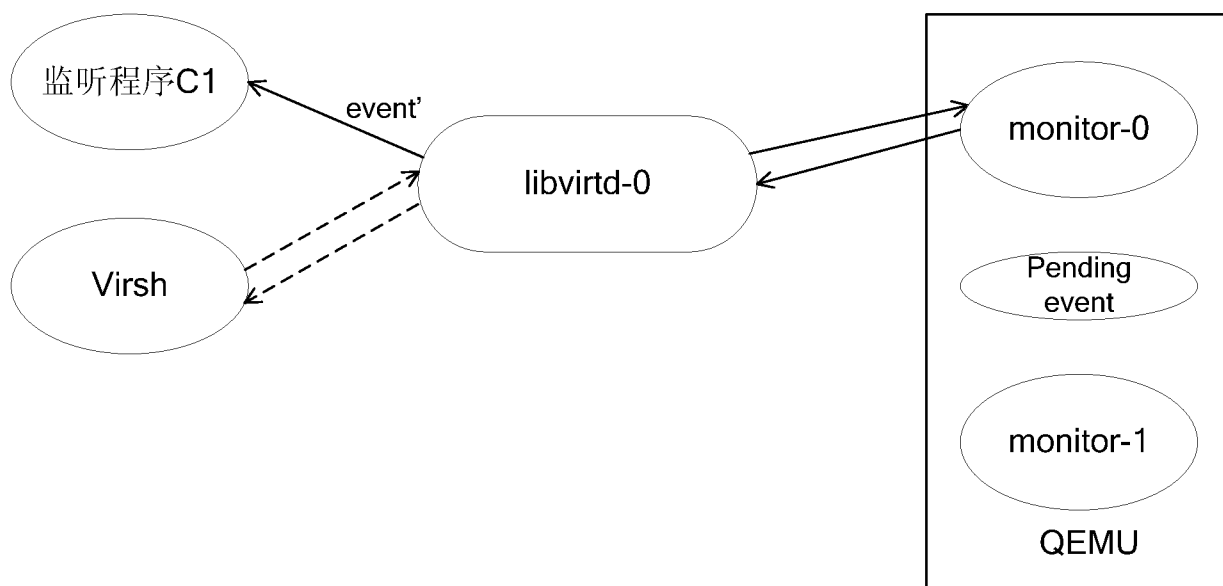


图 10 (c)

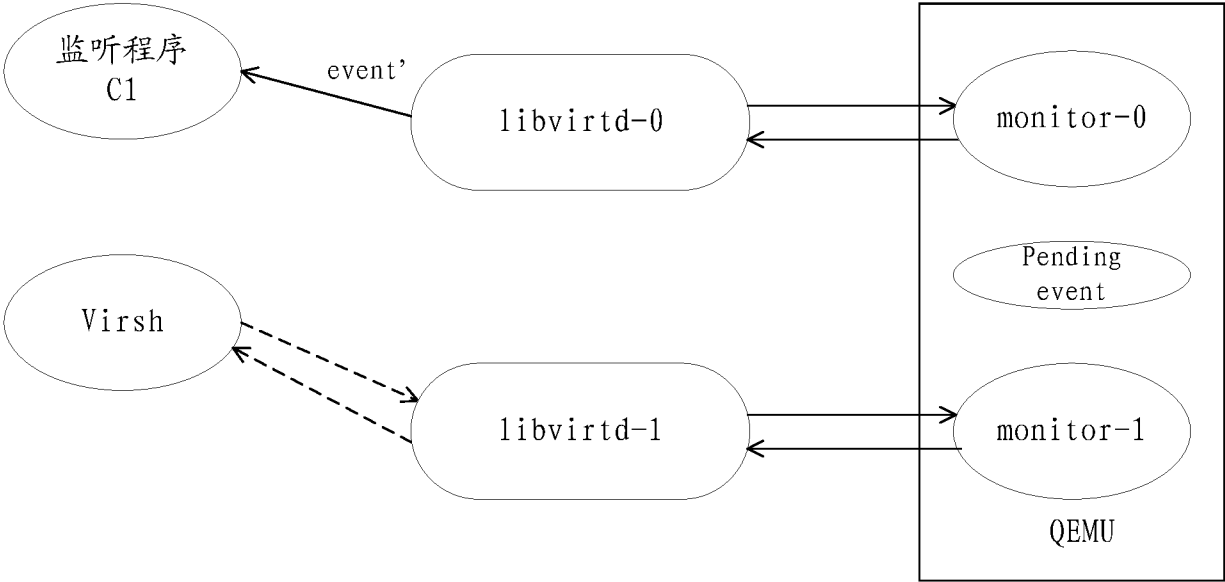


图 10 (d)

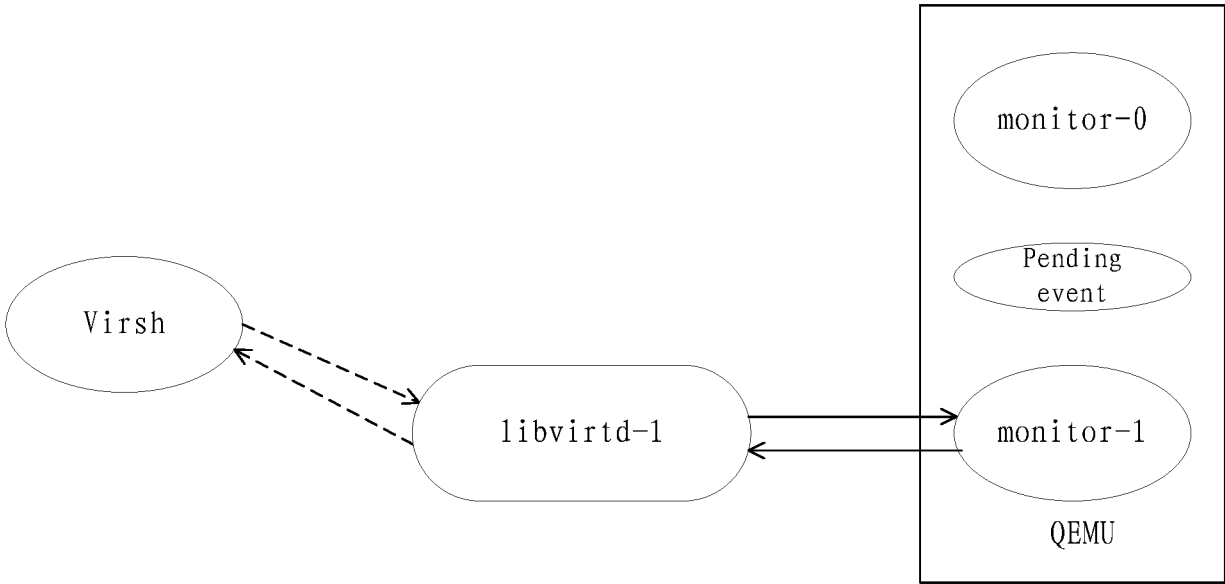


图 10 (e)

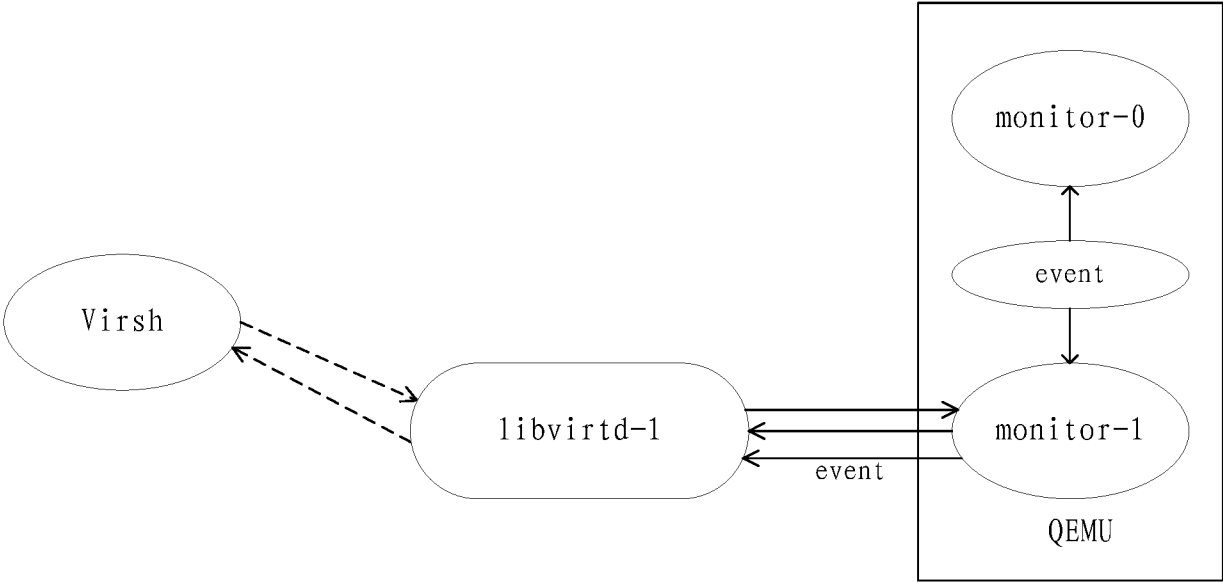


图 10 (f)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2016/079159

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/445 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, WPI, EPODOC, CNKI, GOOGLE: libvirt, update, edition, virtual, status, process, exit, install

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 103473101 A (SI-TECH INFORMATION TECHNOLOGY LTD.) 25 December 2013 (25.12.2013) description, paragraphs [0035]-[0050]	1-24
A	CN 101329633 A (FOUNDER GROUP et al.) 24 December 2008 (24.12.2008) the whole document	1-24
A	CN 103136030 A (HONGFUJIN PRECISION INDUSTRY (SHENZHEN) CO., LTD. et al.) 05 June 2013 (05.06.2013) the whole document	1-24
A	US 2011113136 A1 (INMON CORP.) 12 May 2011 (12.05.2011) the whole document	1-24

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:	“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
“A” document defining the general state of the art which is not considered to be of particular relevance	“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
“E” earlier application or patent but published on or after the international filing date	“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	“&” document member of the same patent family
“O” document referring to an oral disclosure, use, exhibition or other means	
“P” document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 26 June 2016	Date of mailing of the international search report 19 July 2016
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10) 62019451	Authorized officer ZHAO, Tianqi Telephone No. (86-10) 61648112

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2016/079159

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 103473101 A	25 December 2013	None	
CN 101329633 A	24 December 2008	None	
CN 103136030 A	05 June 2013	EP 2784671 A1	01 October 2014
		WO 2013075440 A1	30 May 2013
		US 2013232487 A1	05 September 2013
		TW 201322134 A	01 June 2013
		JP 2013114684 A	10 June 2013
US 2011113136 A1	12 May 2011	None	

国际检索报告

国际申请号

PCT/CN2016/079159

A. 主题的分类

G06F 9/445 (2006.01) i

按照国际专利分类 (IPC) 或者同时按照国家分类和 IPC 两种分类

B. 检索领域

检索的最低限度文献 (标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库 (数据库的名称, 和使用的检索词 (如使用))

CNPAT, WPI, EPODOC, CNKI, GOOGLE: 利博伟特, 热升级, 新版本, 虚拟机, 状态信息, 进程, 退出, 安装, libvirt, update, edition, virtual, status, process, exit, install

C. 相关文件

类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求
A	CN 103473101 A (北京思特奇信息技术股份有限公司) 2013年 12月 25日 (2013 - 12 - 25) 说明书第35-50段	1-24
A	CN 101329633 A (北大方正集团有限公司 等) 2008年 12月 24日 (2008 - 12 - 24) 全文	1-24
A	CN 103136030 A (鸿富锦精密工业深圳有限公司 等) 2013年 6月 5日 (2013 - 06 - 05) 全文	1-24
A	US 2011113136 A1 (INMON CORP.) 2011年 5月 12日 (2011 - 05 - 12) 全文	1-24

☐ 其余文件在C栏的续页中列出。☒ 见同族专利附件。

* 引用文件的具体类型:

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件 (如具体说明的)

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期

2016年 6月 26日

国际检索报告邮寄日期

2016年 7月 19日

ISA/CN的名称和邮寄地址

中华人民共和国国家知识产权局 (ISA/CN)
中国北京市海淀区蓟门桥西土城路6号 100088

传真号 (86-10) 62019451

授权官员

赵天奇

电话号码 (86-10) 61648112

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2016/079159

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
CN	103473101	A	2013年 12月 25日	无			
CN	101329633	A	2008年 12月 24日	无			
CN	103136030	A	2013年 6月 5日	EP	2784671	A1	2014年 10月 1日
				WO	2013075440	A1	2013年 5月 30日
				US	2013232487	A1	2013年 9月 5日
				TW	201322134	A	2013年 6月 1日
				JP	2013114684	A	2013年 6月 10日
US	2011113136	A1	2011年 5月 12日	无			

表 PCT/ISA/210 (同族专利附件) (2009年7月)