(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103329094 A
(43)申请公布日 2013.09.25

(21)申请号 201180058005.4

(22)申请日 2011.09.29

(30)优先权数据
61/389,117 2010.10.01 US
61/507,199 2011.07.13 US
61/507,201 2011.07.13 US
61/507,203 2011.07.13 US
61/507,206 2011.07.13 US
61/507,209 2011.07.13 US
13/247,166 2011.09.28 US

(85)PCT申请进入国家阶段日
2013.05.31

(86)PCT申请的申请数据
PCT/US2011/053909 2011.09.29

(87)PCT申请的公布数据
WO2012/044784 EN 2012.04.05

(71)申请人 Z124
地址 开曼大开曼岛

(72)发明人 B. 里弗斯 P. 里弗斯 R. 特尔茨 D. 里弗斯 S. 瑟帕尔 C. 泰格

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 黄小临

(51)Int.Cl.
G06F 9/22(2006.01)
G06F 9/44(2006.01)

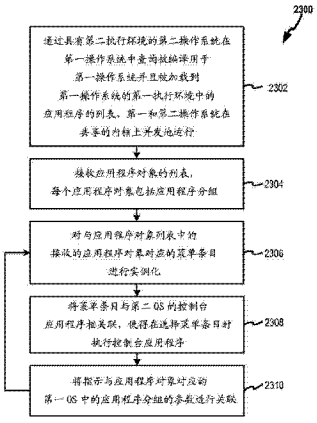
权利要求书2页 说明书30页 附图24页

(54)发明名称

交叉环境重定向

(57)摘要

一种具有在共享的内核上并发地并且独立地运行的移动操作系统和桌面操作系统而不对其进行可视化的移动计算设备。该移动操作系统为移动计算设备提供了适合于移动环境的用户体验。当该移动计算设备与第二用户环境对接时，该桌面操作系统提供了完全的桌面用户体验。交叉环境呈现和用户交互支持在多操作系统计算环境中提供了无缝的计算体验。可以使用移动操作系统的呈现引擎将交叉环境呈现到非扩展的或者扩展的渲染上下文中。为了便于透明的交叉环境使用模型，提供了从与桌面操作系统相关联的用户环境对移动操作系统的应用程序和/或镜像的上下文的访问。通过桌面操作系统的用户环境的菜单内的菜单图标或者菜单列表条目来提供对移动操作系统的应用程序和/或镜像的上下文的访问。移动计算设备可以是在修改的安卓内核上运行安卓移动操作系统和全桌面Linux分布的智能电话。



CN 103329094 A

1. 一种方法，包括：

通过具有第二执行环境的第二操作系统在第一操作系统中查询被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境中的应用程序的列表，第一和第二操作系统在共享的内核上并发地运行；

在第二操作系统中接收应用程序对象的列表，每个应用程序对象包括应用程序分组；

在第二操作系统的菜单中对与应用程序对象的列表中的应用程序对象对应的第一菜单条目进行实例化；

将第一菜单条目与第二操作系统的控制台应用程序相关联，使得在选择第一菜单条目时执行第二操作系统的控制台应用程序；以及

将指示与应用程序对象对应的第一操作系统中的应用程序分组的参数与第一菜单条目相关联。

2. 根据权利要求 1 的方法，其中第一操作系统的第一图形服务器与第二操作系统的第二图形服务器不兼容。

3. 根据权利要求 1 的方法，其中第一图形服务器包括 Android 图形服务器，第二图形服务器包括 X-windows 类型的图形服务器。

4. 根据权利要求 1 的方法，其中第二操作系统的图形服务器是 X-windows 类型的图形服务器并且其中对应于应用程序对象的第一操作系统的应用程序使用与第二操作系统的 X-windows 类型的图形服务器不兼容的图形库。

5. 根据权利要求 1 的方法，其中应用程序对象包括应用程序名称、应用程序分组名称，以及应用程序图标。

6. 根据权利要求 1 的方法，其中第一菜单条目在被选择时将应用程序分组名称作为执行参数传送给控制台应用程序。

7. 根据权利要求 1 的方法，其中第二操作系统的菜单包括对应于第二应用程序的第二菜单条目，第二应用程序被编译用于第二执行环境并且被加载到第二执行环境中。

8. 根据权利要求 1 的方法，其中第一操作系统包括移动操作系统，第二操作系统包括桌面操作系统。

9. 一种方法，包括：

通过第一操作系统从第二操作系统接收请求以在第一操作系统中启动被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境内的第一应用程序，第一和第二操作系统在共享的内核上并发地运行；

通过第一操作系统分配虚拟显示器；

在第一操作系统内的进程中启动第一应用程序；

将第一应用程序的刷新通知与虚拟显示器相关联；以及

通过迭代地执行以下步骤来保持第一应用程序的应用程序图形：

通过第一操作系统监视第一应用程序的应用程序图形信息；以及

在应用程序图形信息被更新时向第二操作系统的控制台应用程序进行通知。

10. 根据权利要求 9 的方法，其中向第二操作系统的控制台应用程序进行通知包括：向控制台应用程序传送文件描述符，所述文件描述符指示在共享的存储位置处应用程序图形信息是可用的，所述共享的存储位置可由第一操作系统和第二操作系统访问。

11. 根据权利要求 10 的方法，其中通过进程间通信信道传送文件描述符。

12. 根据权利要求 11 的方法，其中进程间通信信道包括本地套接字。

13. 根据权利要求 11 的方法，其中进程间通信信道包括 Unix 域套接字。

14. 根据权利要求 11 的方法，其中进程间通信信道包括 Binder 接口。

15. 根据权利要求 14 的方法，其中应用程序图形信息包括第一应用程序的表面信息。

16. 根据权利要求 14 的方法，其中应用程序图形信息包括第一应用程序的呈现的图形信息。

17. 根据权利要求 14 的方法，其中，响应于用户对第二操作系统的菜单的菜单条目的选择，第二操作系统发起请求以便启动第一应用程序。

18. 一种方法，包括 ：

通过第一操作系统从第二操作系统的控制台应用程序接收请求以启动被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境内的第一应用程序，第一和第二操作系统在共享的内核上并发地运行 ；

在第一操作系统上的第一进程内运行第一应用程序，第一应用程序包括由第一操作系统的表面管理器保持的图形信息 ；

通过第一操作系统将控制台应用程序与第一操作系统的虚拟显示器标识符相关联 ；以及

注册控制台应用程序以便接收第一应用程序的图形信息的刷新通知。

19. 根据权利要求 18 的方法，还包括 ：

通过表面管理器呈现第一操作系统的帧缓冲存储器中的图形帧 ；以及

如果第一应用程序的图形信息被更新，则向控制台应用程序进行通知。

交叉环境重定向

[0001] 相关申请的交叉引用

[0002] 本申请是非临时的，并且要求2010年10月1日提交的标题为“Multi-Operating System Portable Docking Device”、序列号为No.61/389,117的美国临时申请，2011年7月13日提交的标题为“Dockable Mobile Software Architecture”、序列号为No.61/507,199的美国临时申请，2011年7月13日提交的标题为“Cross-Environment Communication Framework”、序列号为No.61/507,201的美国临时申请，2011年7月13日提交的标题为“Multi-Operating System”、序列号为No.61/507,203的美国临时申请，2011年7月13日提交的标题为“Auto-Configuration of a Docked System in Multi-OS Environment”、序列号为No.61/507,206的美国临时申请，以及2011年7月13日提交的标题为“Auto-Waking of a Suspended Secondary OS in a Dockable System”、序列号为No.61/507,209的美国临时申请的申请日的权益，其中针对所有目的通过引用在此而并入前述优先权申请的全部内容。

技术领域

[0003] 本申请总地涉及移动计算环境领域，并且更具体地涉及通过在单个移动计算设备中使用多个操作系统来支持多个用户环境。

背景技术

[0004] 移动计算设备在当今社会变得无处不在。例如，在2008年年底，百分之九十的美国人拥有移动无线设备。同时，移动设备的能力正在快速进步，包括将高级计算能力与移动电话能力进行集成的智能电话。移动提供商已经基于若干不同平台在最近三年内发布了数以百计的新的智能电话(例如，苹果iPhone,Android,BlackBerry,Palm, 以及Windows Mobile)。在美国，智能电话在2010年中期的占有率达到了大约23%，并且在一些年龄段超过了35%。在欧洲，智能电话市场从2009年到2010年增长了41%，仅仅在五个最大的欧洲国家，在2010年7月就有超过6000万智能电话用户。

[0005] 尽管智能电话在普及度和计算能力方面取得进步，但它们提供了有限的用户体验。具体地，它们典型地具有被修改用于移动设备硬件的操作系统和一组受限制的可用于修改后的操作系统的应用程序。例如，许多智能电话运行Google的Android（安卓）操作系统。Android仅仅运行被专门开发用于在基于Java的虚拟机运行时环境中运行的应用程序。另外，尽管Android是基于修改的Linux内核，但它使用与Linux不同的标准C库、系统管理器和服务。相应地，为Linux编写的应用程序在不进行修改或者移植的情况下不能在Android上运行。类似地，苹果的iPhone使用iOS移动操作系统。同样，尽管iOS起源于Mac OS X，但为OS X开发的应用程序不能在iOS上运行。因此，尽管许多应用程序可用于诸如Android和iOA之类的移动操作系统，但许多其它的用于诸如Linux和Mac OS X之类的桌面操作系统的普通应用程序不可用于移动平台。

[0006] 相应地，智能电话典型地适合于一组有限的用户体验并且提供主要为移动环境设

计的应用程序。具体地，智能电话不提供适合的桌面用户体验，它们也不运行多数普通的桌面应用程序。结果，许多用户携带并且使用多个计算设备，包括智能电话、膝上型计算机，和/或平板计算机。在这种情况下，每个设备具有其自身的 CPU、存储器、文件存储设备，和操作系统。

[0007] 智能电话与其它计算设备之间的连接和文件共享涉及通过无线或者有线连接将一个设备（例如，运行移动 OS 的智能电话）链接到完全不同的第二个设备（例如，运行桌面 OS 的笔记本、台式机或者平板计算机）。通过在每个设备上单独运行的应用程序之间进行数据同步而跨越各设备共享信息。被典型地称为“同步”的该过程是麻烦的并且通常需要用户的主动管理。

发明内容

[0008] 本发明的实施例针对在单个移动计算设备中提供智能电话的移动计算体验和辅助终端环境的合适用户体验。辅助终端环境可以是通过有线（例如 USB、Firewire、Thunderbolt 等等）或者无线（例如蓝牙、WiFi 等等）连接而连接到计算设备的视觉呈现设备（例如，监视器或者显示器）、输入设备（例如，鼠标、触控板、触摸屏、键盘等等），以及其它计算外设（例如，HDD、光盘驱动器、存储棒、相机、打印机等等）的某些组合。在实施例中，与移动环境的用户体验相关联的移动操作系统和与辅助终端环境的用户体验相关联的桌面操作系统并发地（concurrently）并且独立地在共享的内核上运行。

[0009] 根据与各个实施例一致的一个方面，由一种方法提供交叉环境的重定向，该方法包括：通过具有第二执行环境的第二操作系统在第一操作系统中查询被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境中的应用程序的列表，第一和第二操作系统在共享的内核上并发地运行；在第二操作系统中接收应用程序对象的列表，每个应用程序对象包括应用程序分组；在第二操作系统的菜单中对与应用程序对象的列表中的应用程序对象对应的第一菜单条目进行实例化；将第一菜单条目与第二操作系统的控制台应用程序相关联，使得在选择第一菜单条目时执行第二操作系统的控制台应用程序，以及将指示与应用程序对象对应的第一 OS 中的应用程序分组的参数与第一菜单条目相关联。

[0010] 根据与各个实施例一致的其它方面，第一操作系统的第一图形服务器可以与第二操作系统的第二图形服务器不兼容。第一图形服务器可以是 Android 图形服务器，第二图形服务器可以是 X-windows 类型的图形服务器。第二操作系统的图形服务器可以是 X-windows 类型的图形服务器并且对应于应用程序对象的第一操作系统的应用程序可以使用与第二操作系统的 X-windows 类型的图形服务器不兼容的图形库。

[0011] 根据与各个实施例一致的其它方面，应用程序对象可以包括应用程序名称、应用程序分组名称，以及应用程序图标。第一菜单条目在被选择时可以将应用程序分组名称作为执行参数传送给控制台应用程序。第二操作系统的菜单可以包括对应于第二应用程序的第二菜单条目，第二应用程序被编译用于第二执行环境并且被加载到第二执行环境中。第一操作系统可以包括移动操作系统，第二操作系统可以包括桌面操作系统。

[0012] 根据与各个实施例一致的其它方面，一种用于应用程序或者图形上下文的交叉环境重定向的方法包括：通过第一操作系统从第二操作系统接收请求以在第一操作系统中启动被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境内的第一应用程

序，第一和第二操作系统在共享的内核上并发地运行；通过第一操作系统分配虚拟显示器，在第一操作系统内的进程中启动第一应用程序，将第一应用程序的刷新通知与虚拟显示器相关联；以及通过迭代地执行以下步骤来保持第一应用程序的应用程序图形：通过第一操作系统监视第一应用程序的应用程序图形信息，并且在应用程序图形信息被更新时向第二操作系统的控制台应用程序进行通知。

[0013] 根据与各个实施例一致的其它方面，向第二操作系统的控制台应用程序进行通知可以包括：向控制台应用程序传送文件描述符，所述文件描述符指示在共享的存储位置处应用程序图形信息是可用的，共享的存储位置可由第一操作系统和第二操作系统访问。可以通过进程间通信信道传送文件描述符，所述进程间通信信道可以是 Unix 域套接字或者 Binder（绑定器）接口。应用程序图形信息可以包括第一应用程序的表面信息。可替换地，应用程序图形信息可以包括呈现的第一应用程序的图形信息。响应于用户对第二操作系统的菜单的菜单条目的选择，第二操作系统可以发起请求以便启动第一应用程序。

[0014] 根据与各个实施例一致的其它方面，一种用于应用程序或者图形上下文的交叉环境重定向的方法包括：通过第一操作系统从第二操作系统的控制台应用程序接收请求以启动被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境内的第一应用程序，第一和第二操作系统在共享的内核上并发地运行；在第一操作系统上的第一进程内运行第一应用程序，第一应用程序包括由第一操作系统的表面管理器保持的图形信息，通过第一操作系统将控制台应用程序与第一操作系统的虚拟显示器标识符相关联，以及注册控制台应用程序以便接收第一应用程序的图形信息的刷新通知。该方法还可以包括：通过表面管理器呈现第一操作系统的帧缓冲存储器中的图形帧，并且如果第一应用程序的图形信息被更新，则向控制台应用程序进行通知。

附图说明

[0015] 在参考附图中图示了本发明的实施例，其中类似的标号在整个对图的描述中指代类似的元素。

[0016] 图 1 图示了根据各个实施例的提供多个用户计算体验的计算环境。

[0017] 图 2 图示了根据各个实施例的用于移动计算设备的示例性的系统架构。

[0018] 图 3 图示了根据各个实施例的用于计算环境的操作系统架构。

[0019] 图 4 图示了采用实施例的各个方面的示例性的计算环境。

[0020] 图 5 图示了根据各个实施例的用于计算环境的操作系统架构的方面。

[0021] 图 6 更详细地图示了根据各个实施例的、可以用于配置移动计算设备的操作系统架构的示例性的引导例程。

[0022] 图 7 图示了根据各个实施例的、用于提供应用程序和 / 或用户交互空间的交叉环境呈现的操作系统架构配置。

[0023] 图 8 图示了根据各个实施例的具有多个用户环境的计算环境。

[0024] 图 9 图示了根据各个实施例的交叉环境远程呈现的方面。

[0025] 图 10 示出了根据各个实施例的、一种用于在非扩展的渲染上下文（rendering context）中进行交叉环境远程呈现的例示性方法的流程图。

[0026] 图 11 图示了根据各个实施例的、一种用于交叉环境远程呈现的注册和绘制过程

流程。

[0027] 图 12 示出了根据各个实施例的、一种用于在非扩展的渲染上下文中进行交叉环境呈现的另一例示性方法的流程图。

[0028] 图 13 图示了根据各个实施例的、用于对交叉环境应用程序提供用户交互支持的操作系统架构配置 300b。

[0029] 图 14 图示了根据各个实施例的、使用非扩展的图形上下文呈现的交叉环境应用程序的用户交互支持的方面。

[0030] 图 15 图示了根据各个实施例的、使用扩展的渲染上下文跨越多个 OS 进行并发的用户界面支持的方面。

[0031] 图 16 示出了根据各个实施例的、一种用于在扩展的渲染上下文中进行交叉环境远程呈现的例示性方法的流程图。

[0032] 图 17 示出了根据各个实施例的、另一种用于在扩展的渲染上下文中进行交叉环境呈现的例示性方法的流程图。

[0033] 图 18a 图示了根据各个实施例的、在扩展的渲染上下文中在交叉环境呈现时可以采用的用户环境。

[0034] 图 18b 图示了根据各个实施例的、在扩展的渲染上下文中在交叉环境呈现时可以采用的扩展的输入队列。

[0035] 图 19 图示了根据各个实施例的、在扩展的渲染上下文中在交叉环境呈现时可以采用的一种接收输入事件的方法。

[0036] 图 20 示出了根据各个实施例的、一种提供镜像的上下文的交叉环境呈现的例示性方法的流程图。

[0037] 图 21 示出了根据各个实施例的、另一种提供镜像的上下文的交叉环境呈现的例示性方法的流程图 2100。

[0038] 图 22 图示了根据各个实施例的、交叉环境重定向的方面。

[0039] 图 23 图示了根据各个实施例的、一种为执行交叉环境重定向的方面可以采用的例示性方法的流程图。

[0040] 图 24 图示了根据各个实施例的、另一种为执行交叉环境重定向的方面可以采用的例示性方法的流程图。

具体实施方式

[0041] 包括计算能力的移动电话设备(即,智能电话、手机、移动站、便携式通信设备等等)日益普及。这些智能电话中的许多包括在移动处理器上运行的移动操作系统(“OS”)。尽管移动处理器和移动 OS 已经增加了这些设备的能力,但智能电话还没有取代诸如桌面或者笔记本计算机之类的个人计算机(“PC”)环境(即,Windows,Mac OS X,Linux)的趋势,这至少是因为其提供的用户体验有限。具体地,在智能电话上看到的(多个)用户接口设备典型地被裁制为适应移动环境。例如,智能电话典型地使用小的拇指风格的 QWERTY 键盘、触摸屏显示器、棘轮,和 / 或滚轮作为用户接口设备。移动 OS、以及为移动 OS 开发的应用程序(即,“Apps”)典型地被设计用于包括移动处理器和在移动设备上存在的(多个)用户接口设备的移动环境的约束条件。因此,已经为 PC 操作环境开发的许多应用程序不可用于移动

OS（即，不能被编译并且不能在移动 OS 上运行）。另外，对于诸如打字或者编辑文档之类的某些任务，全尺寸键盘和大显示器比典型地在智能电话上看到的用户接口组件更易使用。

[0042] 相应地，用户典型地使用单独的计算设备用于每种计算体验，包括智能电话、平板计算机、膝上型计算机，和 / 或桌面计算机。在这种情况下，每个设备具有其自身的 CPU、存储器、文件存储设备，和 OS。智能电话与其它设备之间的连接和文件共享涉及通过无线或者有线连接将一个设备（例如，运行移动 OS 的智能电话）链接到完全不同的第二设备（例如，运行桌面 OS 的笔记本、台式机或者平板计算机）。通过在每个设备上单独运行的应用程序之间进行数据同步而跨越各设备共享信息。典型地被称为“同步”的该过程是麻烦的并且通常需要用户的主动管理。

[0043] 图 1 图示了根据各个实施例的计算环境 100，其利用包括与单独的用户交互空间（即，用户环境）相关联的多个操作系统的移动设备提供了多种用户计算体验。计算环境 100 的第一用户交互空间 115 包括移动计算设备 110 的（多个）显示器 116 和 I/O 设备 118。当移动计算设备 110 作为单机移动设备操作时，移动 OS130 通过用户交互空间 115 呈现典型的移动计算用户体验。由移动 OS130 提供的移动计算体验典型地包括移动电话能力以及适合于包括（多个）显示器 116 和（多个）I/O 设备 118 的用户交互空间 115 的图形用户界面（“GUI”）。例如，（多个）显示器 116 可以是（多个）触摸屏显示器，并且可以使用（多个）触摸屏显示器 116 主要通过移动 OS130 的基于手势的 GUI 来控制在移动 OS130 上运行的应用程序（即，“Apps”）。

[0044] 在计算环境 100 中，移动计算设备 110 可以与包括 I/O 设备 144、146 和 / 或 148 的辅助终端环境 140 对接（dock）。在实施例中，通过将移动计算设备 110 的端口 120 连接到辅助终端环境 140 的端口 142 而将移动计算设备 110 与辅助终端环境 140 对接。在这种情况下，辅助终端环境 140 呈现计算环境 100 的第二用户交互空间。在某些情况下，第二用户交互空间可以更适合于桌面计算体验。在这些情况下，桌面 OS160 可以与辅助终端环境 140 相关联以便通过第二用户交互空间提供笔记本计算机、平板计算机，或者桌面计算机环境的完整能力。

[0045] 在实施例中，移动 OS130 和桌面 OS160 在移动计算设备 110 的处理器上的共享内核上并发地运行。在 2011 年 8 月 24 日提交的题为“MULTI-OPERATING SYSTEM”的美国专利申请 NO. 13/217,108 中更详细地描述了在共享的内核上并发执行移动 OS 和桌面 OS，在此通过引用将其并入。以此方式，单个移动计算设备可以通过第一用户交互空间提供移动计算体验并且通过第二用户交互空间提供桌面计算体验。尽管携带一个可以通过单独的用户交互空间并发地执行多个操作系统的移动设备的能力解决了用户的多个问题，但每个用户交互空间（通过并发地运行移动 OS 和桌面 OS）通常提供了单独的一组可用的应用程序和用户功能。

[0046] 本发明的实施例针对便于在第二 OS（例如，桌面 OS160）内以看得见的方式执行第一 OS（例如移动 OS130）中运行的应用程序，其中第一和第二 OS 在共享的内核上并发地运行。值得注意的，对于被编译用于第一（例如，非兼容的）OS 并且在第一 OS 中运行的应用程序而言，在第二 OS 中向用户提供输入（例如，输入设备）和输出（例如，显示器、音频等）支持涉及要解决多个问题。当处理并发运行的多个应用程序的显示和交互性时，可能出现附加的问题。

[0047] 例如，考虑第一和第二应用程序二者被编译用于第一 OS 并且在第一 OS 上并发地运行。然而，用户期望通过与第一 OS 相关联的输入 / 输出设备（例如使用移动计算环境的触摸屏显示器）观看第一应用程序的图形输出并且与第一应用程序交互，以及通过与第二 OS 相关联的输入 / 输出设备（例如使用桌面计算环境的显示器、键盘，以及鼠标）观看第二应用程序的图形输出并且与第二应用程序交互。处理这种情形涉及并发地处理多个显示环境中的图形并且并发地处理全部在单独的（例如非兼容的）操作系统上的单独的应用程序的多个输入 / 输出流。

[0048] 相应地，实施例提供了各种新颖的技术用于：在第二 OS 的用户交互空间内访问第一 OS 的应用程序，在第二 OS 的用户交互空间内显示在第一 OS 中运行的应用程序，以及通过第二 OS 的用户交互空间处理与那些应用程序的用户交互。实施例包括支持交叉环境应用程序的各种显示和用户交互特征的第二 OS 的控制台应用程序（console application）。

[0049] 一组实施例使用所谓的“非扩展的”渲染上下文提供对跨越多个 OS 计算环境的并发的用户界面支持的技术。另一组实施例使用所谓的“扩展的”渲染上下文提供对跨越多个 OS 计算环境的并发的用户界面支持的技术。又一组实施例使用所谓的“镜像的”上下文提供对跨越多个 OS 的并发的用户界面支持的技术。再一组实施例提供从第二 OS 的用户交互空间到第一 OS 上可用的应用程序的访问。以下将更充分地描述这些组的实施例中的每一个。

[0050] 如上所述，计算环境 100 通过与并发地运行多个操作系统的移动设备相关联的多个用户交互空间来提供多种用户计算体验。具体地，因为移动计算设备 110 包括多个 OS，其中每个 OS 适合于特定的计算环境，移动计算设备 110 可用适配于外部 I/O 设备以便利用单个移动计算设备提供宽范围的用户体验。例如，用户可能具有移动计算设备 110 和在膝上型外壳内包括键盘、显示器、和 / 或（多个）定点设备的辅助终端环境 140。当移动计算设备 110 与膝上型类型的辅助终端环境对接时，通过辅助终端环境 140 可得到桌面 OS160 的完整能力。

[0051] 图 2 图示了根据各个实施例的移动计算环境 110 的示例性的硬件系统架构。移动计算设备硬件 112 包括移动处理器 114，移动处理器 114 包括一个或多个 CPU 核 204 以及外部显示接口 220。通常，移动计算设备硬件 112 还包括 I/O 设备 118、存储器 206、存储设备 208、连接到触摸屏显示器 116 的触摸屏显示器控制器 210、连接到电池 216 的电源管理 IC214、蜂窝调制解调器 218、通信设备 222，和 / 或通过各种通信信号和接口连接到处理器 114 的其它设备 224。I/O 设备 118 通常包括按钮以及在移动计算设备 110 中可以采用的其它用户接口组件。例如，I/O 设备 118 可以包括一组按钮（例如，回退、菜单、主屏、搜索等等）、屏外（off-screen）手势区、棘轮、滚轮、QWERTY 键盘等等。其它设备 224 可以包括例如，GPS 设备、LAN 连接、麦克风、扬声器、相机、加速计，和 / 或 MS/MMC/SD/SDIO 卡接口。外部显示接口 220 可以是任何适合的显示接口（例如，VGA、DVI、HDMI 等等）。

[0052] 处理器 114 可以是基于 ARM 的移动处理器。在实施例中，移动处理器 114 是基于 ARM 的移动处理器，诸如德州仪器 OMAP3430、Marvell PXA320、Freescale iMX51，或者 Qualcomm QSD8650/8250。然而，移动处理器 114 可以是另一种基于 ARM 的适合的移动处理器或者基于其它处理器架构（诸如例如基于 x86 处理器架构或者基于其它 RISC 处理器架构）的处理器。

[0053] 尽管图 2 图示了用于移动计算设备 110 的一种示例性的硬件实现方式 112，但是其它架构也是在本发明的范围之内可以想到的。例如，图 2 中图示的移动处理器 114 外部的各种组件可以被集成到移动处理器 114 中。可选地，图 2 中所示的被集成到移动处理器 114 内部的外部显示接口 220 可以位于移动处理器 114 的外部。另外，采用系统总线、离散的图形处理器，和 / 或其它架构性变型的其它计算机架构也适合于采用本发明的方面。

[0054] 图 3 图示了根据各个实施例的可以采用来在移动计算设备 110 上并发地运行移动 OS130 和桌面 OS160 的 OS 架构 300。如图 3 图示的，移动 OS130 和桌面 OS160 是独立的操作系统。具体地，移动 OS130 和桌面 OS160 可以具有独立并且不兼容的用户库、图形系统，和 / 或框架层。可以将 OS 架构 300 的功能和指令作为计算机程序代码存储在移动计算设备 110 的有形的计算机可读介质上。例如，可以将 OS 架构 300 的指令存储在移动计算设备硬件 112 的（多个）存储设备 208 中。

[0055] 在 OS 架构 300 中，移动 OS130 和桌面 OS160 在共享的内核 320 上并发地运行。这意味着移动 OS130 和桌面 OS160 同时在共享的内核 320 上运行。具体地，移动 OS130 和桌面 OS160 二者例如通过对共享的内核 320 进行系统调用而通过同一内核接口 322 连接到共享的内核 320。共享的内核 320 管理用于移动 OS130 和桌面 OS160 二者的处理的任务调度。在这一点上，移动 OS130 和桌面 OS160 在共享的内核 320 上独立地并且并发地运行。另外，如硬件接口 312 所图示的，共享的内核 320 直接在移动计算设备硬件 112 的移动处理器 114 上运行。具体地，共享的内核 320 直接管理移动计算设备硬件 112 的计算资源，诸如 CPU 调度、存储器访问，和 I/O。在这一点上，硬件资源没有被虚拟化，意味着移动 OS130 和桌面 OS160 通过内核接口 322 而不是通过虚拟化的存储器或 I/O 访问来进行系统调用。

[0056] 如图 3 图示，移动 OS130 具有库层 330、应用框架层 340，以及应用层 350。在移动 OS130 中，应用 352 和 354 在移动 OS130 的应用框架层 340 支持的应用层 350 中运行。应用框架层 340 包括在移动 OS130 上运行的应用程序所使用的（多个）管理器 342 和（多个）服务 344。例如，应用框架层 340 可以包括窗口管理器、活动管理器、分组管理器、资源管理器、电话管理器、手势控制器，和 / 或用于移动环境的其它管理器和服务。应用框架层 340 可以包括执行为移动 OS130 开发的应用程序的移动应用运行时环境。可以针对移动计算资源，诸如低处理功率和 / 或有限的存储空间，来优化移动应用运行时环境。移动应用运行时环境可能依赖于用于进程隔离、存储器管理以及线程支持的内核。库层 330 包括实施诸如 I/O 和字符串操作、图形功能、数据库能力、通信能力，和 / 或其它功能和能力之类的普通功能的用户库 332。

[0057] 如图 3 图示的，桌面 OS160 具有库层 360、框架层 370，以及应用层 380。在桌面 OS160 中，应用 382 和 384 在桌面 OS160 的应用框架层 370 支持的应用层 380 中运行。应用框架层 370 包括在桌面 OS160 上运行的应用程序所使用的（多个）管理器 372 和（多个）服务 374。例如，应用框架层 370 可以包括窗口管理器、活动管理器、分组管理器、资源管理器，和 / 或桌面环境所共有的其它管理器和服务。库层 360 可以包括实施诸如 I/O 和字符串操作、图形功能、数据库能力、通信能力，和 / 或其它功能和能力之类的普通功能的用户库 362。

[0058] 在本公开的各种实施例中，桌面 OS160 与移动 OS130 运行在单独的执行环境中。例如，移动 OS130 可以运行在根执行环境中，桌面 OS160 可以运行在根执行环境下建立的次级执行环境中。在移动 OS130 上运行的处理和应用程序访问根执行环境中的用户库 332、（多

个)管理器 342 和(多个)服务 344。在桌面 OS160 上运行的处理和应用程序访问次级执行环境中的用户库 362、(多个)管理器 372 和(多个)服务 374。

[0059] 在实施例中,移动 OS130 和桌面 160 是具有不兼容的用户库、图形系统,和 / 或应用框架的独立的操作系统。因此,为移动 OS130 开发的应用可能不能直接在桌面 OS160 上运行,为桌面 OS160 开发的应用可能不能直接在移动 OS130 上运行。例如,在移动 OS130 的应用层 350 中运行的应用程序 352 可能与桌面 OS160 不兼容,这意味着应用程序 352 不能在桌面 OS160 上运行。具体地,应用 352 可能依赖对于桌面 OS160 的(多个)管理器 372、(多个)服务 374、和 / 或库 362 不可用或者不兼容的移动 OS130 的(多个)管理器 342、(多个)服务 344、和 / 或库 332。

[0060] 结果,移动 OS130 和桌面 OS160 可能具有不同组的可用应用程序。在这一点上,OS 架构 300 的移动 OS130 和桌面 OS160 通过经由单独的用户交互空间可访问的单独的多组应用程序提供了单独的用户体验。用户可以通过与移动 OS130 相关联的第一用户交互空间访问移动 OS130 上可用的(即,被编译用于移动 OS130 并且被加载到移动 OS130 的执行环境中的)应用程序,并且通过与桌面 OS160 相关联的第二用户交互空间访问在桌面 OS160 上可用的应用程序。

[0061] 如上所述,移动操作系统典型地不使用与桌面操作系统相同的图形环境。桌面 OS 的图形环境被设计为针对灵活性和高性能。例如,由某些桌面 OS 使用的 X-window 系统以更高的处理和系统资源为代价提供了平台和网络的独立性。相反,移动 OS 的图形环境被设计为更多地针对效率和移动计算环境的特定用户输入设备而较少地针对灵活性。因为移动 OS 和桌面 OS 的图形环境经常不同,所以不能通过将来自移动 OS 的图形服务器的图形信息重定向到桌面 OS 的图形服务器而将在移动 OS 上运行的应用程序重定向为在桌面 OS 的用户空间中进行显示。

[0062] 最广泛采用的移动 OS 是 Google (谷歌) 的 Android (安卓)。尽管 Android 是基于 Linux,但它包括针对移动环境和移动处理器而对内核和其它 OS 层进行的修改。具体地,尽管 Linux 内核是针对 PC(即, x86)CPU 架构设计的,但 Android 内核是针对基于 ARM 的移动处理器而修改的。针对在移动硬件架构中典型存在的设备也特别调整了 Android 设备驱动器,所述移动硬件架构包括触摸屏、移动连接(GSM/EDGE, CDMA, Wi-Fi 等等)、电池管理、GPS、加速计,以及相机模块,连同其它设备。另外,Android 既不具有原生的 XWindow System 也不支持完整组的标准 GNU 库,这使得难以将现有的 GNU/Linux 应用程序或者库移植到 Android 上。

[0063] 苹果的(在 iPhone 上运行的)iOS 操作系统和微软的 Windows Phone7 是针对移动环境和移动硬件架构被类似修改的。例如,尽管从 Mac OS X 桌面 OS 中导出 iOS,但普通的 Mac OS X 应用程序不能在 iOS 上原生地运行。具体地,通过标准的开发者套件(“SDK”)开发 iOS 应用程序以便在 iOS 的“Cocoa Touch”运行时环境内运行,其对于诸如基于触摸的输入、推送通知和系统服务之类的关键 iOS 特征提供了基本的应用程序基础结构和支持。因此,为 Mac OS X 编写的应用程序不能在没有移植的情况下在 iOS 上运行。另外,将 Mac OS X 应用程序移植到 iOS 上可能是困难的,这是因为两个 OS 的用户库和 / 或应用框架层之间的差异、和 / 或移动硬件和桌面硬件的系统资源之间的差异。

[0064] 在与 OS 架构 300 一致的一个实施例中,Android 移动 OS 和全 Linux OS 独立地并

且并发地在修改的Android内核上运行。在该实施例中，Android OS可以是修改的Android分布，而Linux OS(“Hydroid”)可以是修改的Debian Linux桌面OS。图4-6更详细地图示了根据各个实施例的可以在OS架构300中采用的Android移动OS430、Android内核520，以及Hydroid OS660。

[0065] 如图4图示的，Android OS430包括通过应用框架层440访问的库层432中的一组C/C++库。库层432包括特别针对Android开发的、要比“glibc”LinuxC库更小并更快的“bionic”系统C库439。库层432还包括进程间通信(“IPC”)库436，其包括Android OS的“Binder”IPC机制的基本类。Binder是特别针对Android开发的以便允许进程和服务之间进行通信。图4中的库层432中示出的其它库包括：支持媒体格式的记录和回放的媒体库435、管理对显示子系统的访问以及合成来自多个应用程序的图形层的表面(surface)管理器434、2D和3D图形引擎438，以及轻量关系数据库引擎437。可以被包括在库层432中但是未在图4中画出的其它库包括位图和矢量字体呈现库、公用库、浏览器工具(即，WebKit等等)，和/或安全通信库(即，SSL等等)。

[0066] Android OS430的应用框架层440提供了开发平台，该开发平台允许开发者使用设备硬件的组件、访问位置信息、运行后台服务、设置警报，向状态栏添加通知，等等。框架层440还允许应用程序公开它们的能力并且使用其它应用程序的所公开的能力。Android移动OS430的应用框架层440的组件包括活动管理器441、资源管理器442、窗口管理器443、对接管理器444、硬件和系统服务445、桌面监视器服务446、多显示器管理器447，以及远程通信服务448。Android移动OS430的框架层440中可以包括的其它组件包括：视图系统、电话管理器、分组管理器、位置管理器，和/或通知管理器，连同其它管理器和服务。

[0067] 在Android OS430上运行的应用程序在Android面向对象应用框架之上的Android运行时环境433中的Dalvik虚拟机431内运行。Dalvik虚拟机431是基于注册的虚拟机，并且运行被设计用于减少存储器使用和处理需求的紧凑的可执行的格式。在Android OS430上运行的应用程序包括主屏幕451、电子邮件应用程序452、电话应用程序453、浏览器应用程序454，和/或(多个)其它应用程序(“APP(s)”)455。

[0068] Android OS图形系统使用客户机/服务器模型。表面管理器(“SurfaceFlinger”)是图形服务器，应用程序是客户机。SurfaceFlinger保持显示器ID的列表并且跟踪向显示器ID分配应用程序。在一个实施例中，移动计算设备110具有多个触摸屏显示器116。在该实施例中，显示器ID0与一个触摸屏显示器116相关联，显示器ID1与另一个触摸屏显示器116相关联。显示器ID2与这两个触摸屏显示器116相关联(即，同时在这两个显示器上显示应用程序)。ID大于2的显示器是虚拟显示器，意味着它们不与在移动计算设备硬件112上物理存在的显示器相关联。

[0069] Android应用程序的图形信息包括窗口、视图和画布(canvas)。利用底层表面对象来实现每个窗口、视图，和/或画布。表面对象是双缓冲的(前和后缓冲)并且在绘制的过程中被同步。SurfaceFlinger在共享的存储器池中保持所有的表面，共享的存储器池允许Android内的所有处理访问并且绘制到它们中而不用花费高的拷贝操作并且也不用使用服务器侧的绘制协议，诸如X-Windows。应用程序总是绘制到后缓冲区中，而SurfaceFlinger从前缓冲区进行读取。SurfaceFlinger创建每个表面对象、保持所有的表面对象，并且还为每个应用程序保持表面对象的列表。当应用程序完成在后缓冲区中的绘制时，其向

SurfaceFlinger 通知事件，这使得后缓冲区互换到前缓冲区，并且将呈现表面信息的任务排队给帧缓冲区。

[0070] SurfaceFlinger 监视所有的窗口改变事件。当一个或多个窗口改变事件出现时，SurfaceFlinger 将表面信息呈现给帧缓冲区用于一个或多个显示。呈现包括合成表面，即，基于表面的维度、透明度、z 顺序以及可见性合成最终的图像帧。呈现还可以包括硬件加速(例如，用于图形处理硬件的 OpenGL2D 和 / 或 3D 接口)。SurfaceFlinger 在所有的表面对象上循环并且将它们的前缓冲区以其 Z 顺序呈现给帧缓冲区。

[0071] 图 5 更详细地图示了根据各个实施例的修改后的 Android 内核 520。修改后的 Android 内核 520 包括触摸屏显示器驱动器 521、(多个) 相机驱动器 522、(多个) 蓝牙驱动器 523、共享的存储分配器 524、(多个) IPC 驱动器 525、(多个)USB 驱动器 526、(多个)WiFi 驱动器 527、(多个) I/O 设备驱动器 528，和 / 或功率管理模块 530。(多个) I/O 驱动器 528 包括用于外部 I/O 设备的设备驱动器，所述外部 I/O 设备包括可以通过端口 120 连接到移动计算设备 110 的设备。修改后的 Android 内核 520 可以包括其它驱动器和功能块(包括低存储器抑制器(low memory killer)、内核调试器、日志记录能力，和 / 或其它硬件设备驱动器)。

[0072] 图 6 更详细地图示了根据各个实施例的 Hydroid OS660。Hydroid 是能够运行为标准 Linux 分布开发的几乎任何应用程序的全 Linux OS。具体地，Hydroid OS660 的库层 662 包括支持联网、图形处理、数据库管理、和其它普通程序功能的 Linux 库。例如，任何库 662 可以包括“glibc”Linux C 库 664、Linux 图形库 662 (例如，GTK、OpenGL 等等)、Linux 公用库 661、Linux 数据库、和 / 或其它 Linux 用户库。应用程序运行在使用 X-Server674、窗口管理器 673、和 / 或桌面环境 672 的 X-Windows Linux 图形环境内的 Hydroid 上。图示的应用程序包括字处理器 681、电子邮件应用程序 682、电子数据表应用程序 683、浏览器 684 和其它(多个) 应用程序 685。

[0073] Linux OS 图形系统基于 X-windows(或者“X11”) 图形系统。X-windows 是独立于平台、联网的图形框架。X-windows 使用客户机 / 服务器模型，其中 X-server 是图形服务器而应用程序是客户机。X- 服务器控制与 Linux OS 相关联的输入 / 输出硬件，诸如显示器、触摸屏显示器、键盘、(多个)定点设备等等。在这一点上，X-windows 提供服务器侧绘制图形架构，即，X-server 保持包括窗口和像素映射的可绘制的内容。X-client 通过交换用于描述通信信道上的绘制操作的数据分组而与 X-server 通信。X-client 通过标准例程库(“Xlib”) 访问 X 通信协议。例如，X-client 可以向 X-server 发送在客户机窗口上绘制矩形的请求。X-server 向 X-client 发送输入事件，例如键盘或者定点设备输入，和 / 或窗口移动或者调整大小。输入事件与客户机窗口有关。例如，如果指针在窗口内时用户进行点击，则 X-server 向与该窗口相关联的 X-client 发送包括输入事件的分组，所述输入事件包括该动作和与该窗口有关的事件的定位。

[0074] 因为操作系统框架、图形系统、和 / 或库中的差异，为 Android 编写的应用程序通常不能在 Hydroid OS660 上运行，为标准 Linux 分布编写的应用程序通常不能在 Android OS430 上运行。在这一点上，Android OS430 和 HydroidOS660 不是字节码兼容的，这意味着被编译用于其中一个并且对于所述其中一个可执行的程序在另一个上不能运行。

[0075] 在一个实施例中，Hydroid OS660 包括通过共享的内核 520 便于与 Android OS430

通信的交叉环境通信框架的组件。这些组件包括 IPC 库 663，其包括 Android OS 的 Binder IPC 机制的基础类和远程通信服务 671。

[0076] 在一个实施例中，Hydroid OS660 在 Android 根环境内创建的改变根目录的（chrooted）（利用“chroot”命令创建的）辅助执行环境中运行。在辅助执行环境内运行 Hydroid OS660 内的处理和应用程序，使得通过这些处理和应用程序看见的外观根目录是辅助执行环境的根目录。以此方式，Hydroid OS660 可以在不进行修改的情况下运行为标准 Linux 发布编写的程序，这是因为 Linux 用户库 662 对于在改变根目录的（chrooted）辅助执行环境中的 HydroidOS660 上运行的处理是可用的。

[0077] 参照回到图 3，移动 OS130 和桌面 160 是在移动设备上共享的内核 320 上活动并发执行的。移动 OS130 和桌面 OS160 可能关于用户库、图形系统，和 / 或应用框架是不兼容的。因此，移动 OS130 和桌面 OS160 具有不同组的可用的应用程序，这意味着至少一些在移动 OS130 上可用的应用程序在桌面 OS160 上不可用，反之亦然。相应地，OS 架构 300 的移动 OS130 和桌面 OS160 通过经由单独的用户交互空间可访问的不同组的应用程序来提供单独的用户体验。用户可以通过与移动 OS130 相关联的用户交互空间来访问移动 OS130 上可用的（即，被编译用于移动 OS130 并被加载到其执行环境中的）应用程序，并且通过与桌面 OS160 相关联的用户交互空间访问在桌面 OS160 上可用的应用程序。

[0078] 本发明的实施例扩展了 OS 架构 300 的功能以便在多 OS 计算环境中提供更多无缝的计算体验。实施例包括第一操作系统的应用程序和 / 或用户空间在第二操作系统的用户交互空间中的交叉环境呈现，即便第一操作系统和第二操作系统的图形环境不兼容。实施例还包括交叉环境应用程序的用户交互支持，以及从第二操作系统的用户交互空间访问第一操作系统的应用程序。该功能使得例如在移动 OS130 上运行的移动 OS 应用程序能够被与桌面 OS160 相关联的用户交互空间显示并且与之交互。例如，尽管用户正在通过与桌面 OS160 相关联的用户交互空间与桌面 OS160 交互，但用户可能希望访问不可用于桌面 OS160（即，不兼容并且不能在其上运行）的移动 OS130 的特定应用程序。使用以下公开的各个实施例，用户可以通过与桌面 OS160 相关联的用户交互空间对为移动 OS130 编译并在其上运行的应用程序进行访问、显示并且与之交互。特别地，实施例向交叉环境交互支持提供了移动 OS 的任何应用程序，意味着为使用以下的实施例不需要修改移动 OS 应用程序来包括特定的交换环境支持。

[0079] 为提供从第二 OS（例如，桌面 OS160）的用户交互空间对第一 OS（例如，移动 OS130）的应用程序和 / 或用户交互空间的无缝交叉环境用户交互支持，可能期望实时地呈现应用程序和 / 或用户交互空间的图形数据（即，图形上下文或者活动的显示）以供在第二 OS 的用户交互空间中显示。在该上下文中，实时（或者立即）呈现意味着将应用程序的图形数据足够快地呈现给第二 OS 的用户交互空间，使得用户可以与应用程序进行交互而不存在由于与图形信息的传递相关联的延迟引起的应用程序性能的明显的或者实质性的降低。在这一点上，如在此所描述的，用于实时（或者立即）交叉环境呈现的技术提供了图形信息的快速传递，例如具有有限的帧数延迟或者与从第一 OS 到第二 OS 拷贝或者传递图形信息相关联的其它延迟。然而，这不意味着图形传递不花费任何时间，并且在此公开的交叉环境呈现技术可以被视为是立即的或者实时的，即便在图形信息被显示在第二 OS 的用户交互空间上之前经过了有限的时间段。

[0080] 为了实现应用程序的交叉环境呈现，可以从在第一操作系统中运行的应用程序向第二操作系统的图形系统传递潜在的大量图形数据。现有的机制不能够在没有潜在影响显示更新或者帧率的情况下传递所需的图形数据。例如，在移动计算环境的约束条件下直接传递图形数据可能不现实。可以使用压缩技术来减少要传递的总的数据量，但是以增加的压缩和解压缩信息的处理需求为代价。可以使用类似远程桌面的系统来传送矢量图形或者图形更新信息，然而，当传递大量的图形数据或者用于快速改变图形内容时，这些典型地也是缓慢的。

[0081] 图 7 图示了根据各个实施例的 OS 架构配置 300a，可以采用该 OS 架构配置 300a 来在第二 OS 的用户交互空间内提供第一 OS 的应用程序和 / 或用户交互空间的交叉环境呈现，其中第一 OS 和第二 OS 并发地在共享的内核上运行并且与包括单独的显示设备的单独的用户交互空间相关联。OS 架构配置 300a 的组件使得在桌面 OS160 的用户交互空间内呈现在移动 OS130 上运行的应用程序和 / 或移动 OS130 的图形上下文，其中 OS130 和桌面 OS160 在共享的内核 320 上并发地运行。在各个实施例中，通过桌面 OS160 的控制台应用程序 782 在桌面 OS160 的用户交互空间中的控制台窗口中显示应用程序。在一种实现方式中，控制台应用程序 782 是通过桌面 OS160 的 X-windows 类型图形系统在桌面 OS160 的用户交互空间内显示的 X-windows 类型的应用程序。

[0082] 在 OS 架构配置 300a 中，移动 OS130 包括为移动 OS130 的应用程序（例如，应用程序 752 和 754）分配和管理表面信息（例如，表面 726、727 和 728）的图形服务器 734。在一个实施例中，图形服务器 734 使用匿名的共享存储器为图形表面分配存储器（即，在进程之间共享被命名的存储器块，这允许内核空闲）。图形服务器 734 还分配和跟踪移动 OS130 的显示器，包括移动计算设备硬件 112 内集成的显示器（即本地显示器）和通过其可以远程显示（即，在另一 OS 的用户交互空间内的）移动 OS130 的应用程序的所谓的虚拟显示器。移动 OS130 还可以包括在与移动 OS130 相关联的显示屏幕上同时显示多个应用程序的窗口系统。在一个实施例中，移动 OS130 包括提供用于从桌面 OS160 访问移动 OS130 的组件的远程通信信道的服务。

[0083] OS 架构配置 300a 包括在移动 OS130 上运行并且在与移动 OS130 相关联的第一用户交互空间内显示的第一应用程序 752。根据以下描述的实施例，OS 架构配置 300a 包括也在移动 OS130 上运行但是通过交叉环境呈现而在与桌面 OS160 相关联的第二用户交互空间内显示的第二应用程序 754。在一个实施例中，通过在桌面 OS160 上运行的控制台应用程序 782 在第二用户交互空间的控制台窗口内显示应用程序 754。

[0084] 通常，移动 OS130 的应用程序将用户通过其与应用程序交互的视图实例化（instantiate）。例如，应用程序可以具有组成应用程序窗口的单个视图。在各视图内，应用程序将应用程序界面的特定区域的绘制对象实例化。绘制对象可以被称为画布或者绘制层，并且是应用程序通过其绘制图形信息的对象。当应用程序将画布或者层实例化时，图形服务器 734 为与画布或者层相关联的表面信息分配存储器并且将绘制对象返回给应用程序，应用程序然后使用其来绘制画布或者层的表面的图形信息。图形服务器 734 然后监视表面信息并且当表面信息被更新时将表面信息呈现给帧缓冲区（例如，帧缓冲区 716）。典型地，用户还通过视图对象与应用程序进行交互。视图对象包括用户在视图对象上执行动作时由移动 OS 输入队列 736 调用的事件监听程序（listener）。

[0085] 如在 OS 架构 300a 中图示的，通过图形服务器 734 在共享的存储器 724 中分配表面 726、727 和 / 或 728。共享的存储器 724 由共享的内核 320 管理并且通过在移动 OS130 和桌面 OS160 上运行的所有进程可访问。如上所述，共享的存储器 724 可以是被命名的共享存储器。尽管被命名的共享存储器可由在共享的内核 320 上运行的所有进程访问，但是其它进程不能通过名称来访问被命名的共享存储器的区域。相应地，必须通过进程间通信机制向被命名的共享存储器的区域传送文件描述符以便跨越进程边界向被命名的共享存储器传送参考。共享的内核 320 还包括 IPC 驱动器 725，其允许移动 OS130 和桌面 OS160 中的进程跨越进程边界与另一个进行通信。IPC 驱动器 725 可以是例如 Unix 域套接字驱动器、AndroidBinder 驱动器，和 / 或网络套接字驱动器。

[0086] OS 架构配置 300a 的桌面 OS160 包括控制台应用程序 782 和窗口系统 774。编译控制台应用程序 782 并且在桌面 OS160 上运行，以及在与桌面 OS160 相关联的用户交互空间中的控制台窗口内显示。窗口系统 774 可以包括窗口管理器、图形服务器、和 / 或图形设备接口，所述图形设备接口提供用于表示图形对象并且通过桌面 OS 帧缓冲区 718 将图形对象传送给显示设备的基础。例如，窗口系统 774 可以通过桌面 OS 帧缓冲区 718 在桌面 OS160 的用户交互空间内显示控制台应用程序 782。窗口系统 774 还向控制台应用程序 782 提供来自与控制台应用程序 782 相关联的桌面 OS160 的用户环境的输入事件。例如，窗口系统 774 可以将来自与桌面 OS160 相关联的用户交互空间的定点设备位置或者手势信息提供给控制台应用程序 782。

[0087] 在图 7 中，出于易于图示的原因，将包括共享的存储器 724、移动 OS 帧缓冲区 716 和桌面 OS 帧缓冲区 718 的存储块示为位于共享的内核 320 内。然而，这些存储块在物理上位于移动计算设备 110 的有形的存储器存储设备元素内并且由共享的内核 320 管理。例如，这些存储块可以位于图 2 中图示的移动计算设备硬件 112 的处理器 114 上的 RAM 中、和 / 或存储设备 206 的 RAM 中。

[0088] OS 架构配置 300a 为在与第二操作系统相关联的用户交互空间中交叉环境呈现在共享的内核上运行的第一操作系统的应用程序提供了支持，其中第一和第二操作系统在共享的内核上并发地运行。可以采用 OS 架构配置 300a 来提供计算环境中的交叉环境呈现支持，所述计算环境通过多个用户交互空间提供了多种用户计算体验。例如，可以在如图 1 所图示的计算环境 100 中使用 OS 架构配置 300a。

[0089] 图 8 图示了根据各个实施例的计算环境 800。计算环境 800 具有第一用户环境，该第一用户环境包括移动计算设备硬件 112 的 (多个) 触摸屏显示器 116 和其它 I/O 设备 118。该用户环境表示用户通过其与移动 OS130 进行交互的第一用户交互空间。计算环境 800 的第二用户环境包括显示监视器 844、键盘 846，和 / 或定点设备 848。用户环境 840 可以通过对接连接器 841 和对接电缆 843 连接到移动计算设备 110。对接连接器 841 和对接电缆 843 可以包括通过对接接口 122 与移动计算设备 110 的端口 120 连接的端口，如图 1 图示的。在这一点上，用户环境 840 向移动计算设备 110 提供了对接的辅助终端环境。在计算环境 800 中，桌面 OS160 可以与对接的辅助终端环境 840 相关联，使得用户可以通过由辅助终端环境 840 提供的用户交互空间与桌面 OS160 进行交互。尽管辅助终端环境 840 被图示为典型的桌面类型的计算环境，但桌面 OS160 可以通过其它类型的计算环境 (包括膝上型、平板，和 / 或其它类型的计算环境) 呈现第二用户交互空间。

[0090] 可以在计算环境 800 内使用 OS 架构配置 300a 来提供在第一 OS(即,移动 OS) 上运行并在第二 OS 的用户环境(即,与桌面 OS160 相关联的用户环境 840) 中显示的应用程序的交叉环境呈现。例如,可以在桌面 OS160 的用户交互空间 880 上的控制台窗口 882 内显示在移动 OS130 上运行的应用程序 754。用户交互空间 880 内的其它窗口(包括窗口 884 和 886)可以是在桌面 OS160 上运行的其它应用程序的窗口。对于用户,计算环境 800 提供了对应用程序 754 的无缝的计算体验,这是因为可以在桌面 OS160 的用户交互空间内使用应用程序 754,仿佛它正在桌面 OS160 上运行,尽管事实上应用程序 754 正在移动 OS130 上运行。

[0091] 图 8 图示了类似于桌面的辅助终端环境 840。在这种情况下,用户可以使用辅助终端环境 840 的键盘 846 和鼠标 848(即,主要的基于定点设备的 GUI)通过控制台窗口 882 与移动 OS 的应用程序和 / 或用户交互空间进行交互。然而,移动 OS 用户交互空间的应用程序和 / 或镜像的交叉环境呈现可以用于其它辅助终端环境。例如,桌面 OS160 可以与包括触摸屏显示器的类似于平板计算机的辅助终端环境相关联。在这种情况下,用户可以以用户典型地与移动 OS130 的用户交互空间进行交互的非常相同的方式(即,主要基于手势的 GUI)与移动 OS130 的交叉环境应用程序或者镜像的用户交互空间进行交互。

[0092] 如上所讨论的,本发明的实施例针对在多个 OS 计算环境中提供对跨越交叉环境应用程序和 / 或镜像的用户交互空间的并发的用户界面支持。在一个示例中,为交叉环境应用程序提供用户界面支持以便使得通过第二 OS 的用户交互空间对在第一 OS 上运行的应用程序进行显示并进行交互,其仿佛实质上原生地在第二操作系统上运行。

[0093] 非扩展的渲染上下文实施例

[0094] 一些实施例处理跨越多个 OS 的并发的用户界面支持,而不扩展第一操作系统的图形渲染上下文。典型地将第一 OS(例如,移动 OS,Android) 配置为定义单个的、活动的用户交互空间。用户交互空间包括活动的显示器(例如,具有相关联的特征,诸如分辨率)以及允许用户与在活动的显示器上显示的元素进行交互的一个或多个活动的输入设备。相应地,第一 OS 建立渲染上下文,通过该渲染上下文可以呈现正在运行的应用程序的表面信息以供活动显示器进行显示。

[0095] 然而,如上所描述的,在此描述了用于有效地欺骗第一 OS 来并发地处理多个用户交互空间的新颖技术。此外,这些技术使得多个用户交互空间与多个计算环境上的不同的(例如不兼容的)操作系统相关联。一些实施例涉及通过交叉环境远程呈现来处理显示器输出的技术。其它实施例涉及用于处理那些上下文中的用户交互的技术。

[0096] 在交叉环境远程呈现时,在第二 OS 内呈现在第一 OS 上运行的并且在与第二 OS 相关联的计算环境内显示的应用程序的应用程序图形。在一个实施例中,在第二 OS 上运行的控制台应用程序从共享的存储器中访问应用程序的表面信息并且在与第二 OS 相关联的计算环境的控制台窗口内呈现该应用程序。

[0097] 假设日历应用程序和字处理应用程序二者被编译用于移动设备上的第一 OS(例如,移动 OS130) 并且并发地运行在移动设备上的第一 OS 上。第二 OS(例如非移动 OS,如桌面 OS160) 使用共享的内核正在移动设备上并发地运行。用户已经将移动设备与第二计算环境(桌面计算环境)进行对接,并且想要通过桌面计算环境与字处理应用程序进行交互。期望以对于用户而言透明的方式使用移动计算环境(例如,移动设备)的第二 OS 来处理桌面

计算环境的用户交互空间。

[0098] 图 9 图示了根据各个实施例的交叉环境远程呈现的方面。在图 9 图示了应用程序呈现图示 900 中，第一应用程序 910（例如，日历应用程序）计算第一 OS 内的第一表面 912 的更新。第一操作系统将第一表面 912 存储在共享的存储空间中的第一存储位置。例如，第一存储位置可以是被命名的共享存储器的一区域。第一应用程序 910 更新第一表面 912 的后缓冲区 914。类似地，第二应用程序 930（例如，字处理应用程序）使用第一 OS 计算第二表面 932 的更新。通过第一 OS 将第二表面 932 存储在共享存储空间的第二存储位置中（例如，被命名的共享存储器的第二区域）。

[0099] 第一 OS 确定何时要发起呈现序列。例如，第一 OS 可以在表面 912 和 / 或 932 的表面信息已经改变时发起呈现序列。第一 OS 可以对包括第一表面 912 和第二表面 932 的所有表面执行单循环，确定与特定应用程序相关联的表面信息是否已经改变。在呈现序列中，如果表面 912 的表面信息已改变，第一 OS 互换前缓冲区 916 和后缓冲区 914，使得在后缓冲区 914 中的表面信息现在在前缓冲区 916 中。第一操作系统将第一表面 912 呈现到第三存储位置 920，以便创建要在与第一 OS 相关联的第一用户交互空间内显示的最终图像 918。

[0100] 如果第二表面 932 的表面信息已经改变，则第一 OS 向第二 OS 通知第二表面 932 的表面信息已经改变。具体地，第一 OS 通过进程间通信信道向第二 OS 的控制台应用程序发送绘制通知以指示表面 932 已经被更新。该绘制通知可以包括前图像 936 的文件描述符，和 / 或前图像 936 的共享的存储器空间的其它特征（包括缓冲区大小、层顺序等等）。控制台应用程序将文件描述符映射到其处理空间以获得对第二存储位置的参考。通过对第二存储位置的参考，第二 OS 的控制台应用程序从前图像缓冲区 936 中直接读取表面信息并且在与第二 OS 相关联的第二用户交互空间内的控制台窗口 940 中呈现第二表面 932 的前图像 936 的表面信息。以此方式，控制台应用程序可以实时地在控制台窗口 940 中呈现第二应用程序 930 而无须跨越进程拷贝图形帧或者表面信息。取而代之，使用通过进程间通信传送的文件描述符，控制台应用程序通过将第二表面 932 的共享存储器映射到其自身的处理空间而直接读取表面信息。

[0101] 图 10 示出了根据各个实施例的、在非扩展渲染上下文中进行交叉环境远程呈现的例示性方法的流程图 1000。实施例保持第一应用程序（例如，日历应用程序）和第二应用程序（例如，字处理应用程序）的应用程序图形的显示，二者被编译用于第一操作系统并处于第一操作系统的活动的并发执行中。

[0102] 方法 1000 在块 1004 开始，使用第一操作系统计算对第一应用程序的表面的更新。计算对表面的更新可能涉及使用应用程序数据来确定哪些表面已经改变并且以何种方式改变。例如，用户交互可能已经使得一些表面改变位置、顺序（例如，一个层可能部分地在另一个层的前面，被另一个层完全隐藏，等等）、大小、颜色、纹理，等等。

[0103] 在块 1008，使用第一操作系统呈现第一应用程序的这些更新的表面以在第一存储位置生成第一图形帧。例如，第一 OS 的呈现引擎预先建立与显示器相关联的渲染上下文。然后可以通过更新的表面信息进行迭代来呈现图形帧以便根据与渲染上下文相关联的显示器的特征（例如，分辨率）来有效地绘制来自第一应用程序的表面的可见部分的全部或者部分图像。将该图形帧呈现给第一存储位置。存储位置可以是帧缓冲存储器、共享存储器，或者任何其它有用的存储器位置。

[0104] 在一些实施例中，在块 1012，向与第一操作系统相关联的第一计算环境的显示器显示来自第一存储位置的呈现的第一图形帧。例如，将第一图形帧呈现给移动设备的帧缓冲区的后缓冲区部分。结果，帧缓冲区翻转(即，后缓冲区部分变成前缓冲区部分)并且将帧缓冲区的现在的前缓冲区部分显示给移动设备的显示器。

[0105] 在块 1016，使用第一操作系统计算对第二应用程序的表面的更新。这可能与用于第一应用程序的表面的块 1004 的计算基本上相同地进行。然而，与第一应用程序的更新的表面数据不同，不通过第一 OS 来呈现第二应用程序的更新的表面信息。相反，在块 1020，在第二存储位置存储第二应用程序的更新的表面。该第二存储位置是可由在移动设备的共享内核上并发地运行的第一和第二 OS 二者访问的共享的存储位置。

[0106] 在块 1024，使用第二操作系统的控制台应用程序来呈现第二应用程序的更新的表面以便在第三存储位置生成第二图形帧。例如，图 7 的控制台应用程序 782 可以将应用程序 754 的更新的表面呈现到(例如与第二计算环境的显示器相关联的)第二 OS 的帧缓冲存储器中。在一些实施例中，在块 1028，向与第二操作系统相关联的第二计算环境的显示器显示来自第三存储位置的第二图形帧。例如，桌面计算环境显示器的显示驱动器访问帧缓冲存储器以便访问和显示第二图形帧。在某些实现方式中，控制台应用程序还保持控制台交互空间，并且将第二图形帧呈现到控制台交互空间。例如，控制台应用程序在桌面显示器上呈现一个或多个窗口，并且在这些窗口之一上显示第二应用程序的图形。

[0107] 在一些实施例中，方法 1000 在各块上迭代以便为两个应用程序并发地保持图形环境。例如，移动 OS 计算对第一应用程序的图形的更新并且向移动帧缓冲存储器呈现这些更新；然后移动 OS 计算对第二应用程序的图形的更新并且将这些更新存储到共享的存储器中，桌面 OS 的控制台应用程序从共享的存储器中将这些更新呈现到桌面帧缓冲存储器；然后方法 1000 对于下一组更新进行重复。特别地，一些实现方式不按顺序和 / 或并行地执行某些步骤。例如，可能基本上同时计算对第一和第二应用程序的表面的更新(即，基本上并行地执行块 1004 和 1016)，尽管仅仅将本地呈现这些更新的表面的一部分(例如在块 1008)并且将为远程呈现存储这些更新的表面的其它部分(例如在块 1020 和 1024)。

[0108] 在一个实施例中，Android 移动 OS 和全 Linux OS (例如，Hydroid)正在移动设备的共享内核上并发地运行。当从 Hydroid OS660 启动 Android 应用程序时，在 Hydroid OS660 上启动控制台应用程序并且请求 Android OS430 启动 Android 应用程序并向控制台应用程序发送 Android 应用程序的绘制通知。Android OS430 启动 Android 应用程序，将应用程序与虚拟显示器 ID 相关联并且注册控制台应用程序以便接收对该应用程序的绘制通知。例如，Android 图形服务器(即，SurfaceFlinger)可以分配未使用的虚拟显示器 ID 并且通过应用程序对象列表将应用程序与虚拟显示器 ID 相关联。SurfaceFlinger 在共享存储器中分配用于 Android 应用程序的表面信息的存储器，并且注册控制台应用程序以便接收对该 Android 应用程序的绘制通知。

[0109] 当 SurfaceFlinger 确定表面信息被更新时，SurfaceFlinger 通过进程间通信信道向控制台应用程序发送绘制通知。例如，SurfaceFlinger 可以通过 unix 域套接字、Binder 接口、和 / 或网络套接字向控制台应用程序发送绘制通知。绘制通知包括表面信息的文件描述符。该文件描述符可以通过 SurfaceFlinger 基于表面的被命名的共享存储器区域的名称空间(namespace)生成。绘制通知还可以包括数据格式、缓冲区大小、和表面位

置信息。控制台应用程序将文件描述符映射到其处理空间并且通过被映射的文件描述符从共享的存储位置进行读取。控制台应用程序然后直接根据表面信息呈现应用程序的图形帧。然后通过桌面 OS 图形系统(即,通过 Hydroid OS600 的 X-windows 图形系统)显示所呈现的图形帧。

[0110] 图 11 更详细地图示了根据各个实施例的交叉环境远程呈现的注册和绘制处理流程 1100。初始地,在 Hydroid OS600 上运行的控制台应用程序 1102 在步骤 1106 通过共享内核 520 的 IPC 信道 525 请求在 Hydroid OS600 的用户环境中开始并且显示 Android 应用程序,或者从 Android OS430 的用户环境转移到 Hydroid OS600 的用户环境。Android 在步骤 1108、1110 和 1112 从 SurfaceFlinger434 请求应用程序的虚拟显示器 ID,开始应用程序,并且为虚拟显示器设置参数。在步骤 1114,Android 通过 IPC 信道 525 向控制台应用程序返回显示器 ID。在步骤 1116,应用程序请求新表面,SurfaceFlinger 在步骤 1118 通过创建表面类 1104 的实例而创建该新表面。

[0111] 控制台应用程序 1102 在步骤 1120 对渲染器(renderer)对象进行实例化,这通过 Hydroid OS600 的 X-window 系统呈现 Android 表面信息。在步骤 1122,控制台应用程序 1102 向 SurfaceFlinger434 注册渲染器对象 1124 的可远程访问的(remotable)接口以便接收 Android 应用程序的表面信息的绘制通知。在一个实施例中,渲染器对象的可远程访问的接口包括可以通过 IPC 信道 525 调用的 draw() 和 clear() 方法。在步骤 1126 和 1128,SurfaceFlinger 将 IPC 对象附接到表面,使得在表面信息已经被更新时将通过可远程访问的接口向控制台应用程序 1102 进行通知。

[0112] 步骤 1130 和 1132 是处理流程 1100 的呈现循环的一部分。在呈现循环中,SurfaceFlinger 向控制台应用程序 1102 通知表面信息被更新并且通过 IPC 信道 525 将文件描述符传送给表面信息。例如,可以调用控制台应用程序渲染器的绘制方法并且将文件描述符传送给表面。控制台应用程序 1102 将文件描述符映射到其处理空间,并且访问所参考的共享存储位置以便读取表面信息并呈现要在与 Hydroid OS600 相关联的第二计算环境的显示器上显示的图形帧。

[0113] 图 12 示出了根据各个实施例的、在非扩展渲染上下文中进行交叉环境呈现的另一例示性方法的流程图 1200。如同图 10 的方法 1000,实施例保持第一应用程序和第二应用程序的应用程序图形的显示,第一应用程序和第二应用程序二者被编译用于第一操作系统并处于第一操作系统的活动的并发执行中。

[0114] 方法 1200 在块 1204 开始,建立第一操作系统的第一渲染上下文。渲染上下文可以对于与之相关联的显示器是唯一的。例如,移动 OS 的实现方式可以与具有某个分辨率的移动设备显示器相关联。可以建立渲染上下文来与相关联的显示器的分辨率匹配,使得将针对该分辨率而合适地呈现图形。在块 1208,方法 1200 使用第一操作系统计算对第一应用程序的表面的更新。如上讨论的,计算对表面的更新可能涉及使用应用程序数据来确定哪些表面已经改变并且以何种方式改变。

[0115] 然后,在块 1212 使用第一操作系统呈现第一应用程序的更新后的表面,以在第一存储位置生成第一图形帧。呈现典型地涉及将图形原语(primitive)转换为形成用于显示的图形帧的比特(例如,位图)。例如,表面信息在数学上定义每个表面的属性,包括形状、大小、颜色、分层顺序(其在哪些其它表面的前面或后面)、透明度等等。OS 的呈现引擎可以

解释表面数据以便按照呈现所有表面信息的函数而确定在渲染上下文中的每个位置(例如“X,Y”位置)处显示什么比特(例如使用迭代合成、射线追踪,和/或其它技术)。使用第一渲染上下文可以将第一应用程序的更新的表面呈现为位图以存储到第一存储器位置(例如,帧缓冲存储器、共享存储器,或者其它任何有用的存储位置)中。

[0116] 在一些实施例中,在块1216,向与第一操作系统相关联的第一计算环境的显示器显示来自第一存储位置的第一图形帧。例如,移动设备显示器的显示驱动器访问相关联的帧缓冲存储器以便显示在第一渲染上下文中生成的位图。在块1220,拆解(disestablish)(例如,“拆卸”)第一渲染上下文。

[0117] 在块1224,建立第一操作系统的第二渲染上下文。在一些实现方式中,第二渲染上下文与第一渲染上下文相同。然而,还可以根据第二计算环境的显示器(例如桌面显示器)的特征建立第二渲染上下文。在块1228使用第一操作系统计算对第二应用程序的表面的更新。然后,在块1232,在第一操作系统的第二渲染上下文中呈现这些更新以便在第二存储器位置生成第二图形帧。特别地,该第二存储位置是可由正在移动设备的共享内核上并发地运行的第一和第二操作系统二者访问的共享的存储位置。

[0118] 在一些实施例中,在块1236,向与第二操作系统相关联的第二计算环境的显示器显示来自第二存储位置的第二图形帧。值得注意的,与图10中不同,使用第一OS的呈现引擎来呈现两个应用程序的更新后的图形帧。例如,第二OS的控制台应用程序可以访问第二共享存储位置以便直接取得所呈现的位图以供第二计算环境的显示器显示。在块1240,拆解第二渲染上下文。

[0119] 在一些实施例中,方法1200在块上迭代以并发地保持两个应用程序的图形环境。例如,移动OS迭代性地建立、使用并且拆卸第一应用程序的渲染上下文;然后,建立、使用并且拆卸第二应用程序的渲染上下文。使用这种技术,可以由一个OS(即,通过第一OS的呈现引擎)执行所有呈现,并且不需要提供或者使用另一OS的呈现功能。然而,该技术涉及与重复地建立并且拆卸渲染上下文相关联的附加开销。

[0120] 交叉环境呈现的以上实施例描述了使用非扩展的渲染上下文可以在多OS计算环境中的第二OS的用户交互空间的控制台窗口内怎样显示在第一OS上运行的应用程序的图形帧。为了支持与这种交叉环境应用程序的用户交互,实施例以交叉环境应用程序接收输入事件的方式将来自第二OS的用户交互空间的输入事件重定向到第一OS,使得仿佛其来自第一OS的用户交互空间(即,应用程序通过相同的事件句柄(handler)和/或视图接收输入事件,通过事件句柄(handler)和/或视图其将接收如被显示在第一OS的用户交互空间内的用户输入)。

[0121] 参照返回图7和图8,配置桌面OS160以便通过适合于桌面计算体验的辅助终端环境840来提供第二用户交互空间。如上所述,使用上述的交叉环境呈现的实施例,可以在移动OS130的用户交互空间内显示应用程序752,同时通过在桌面OS160上运行的控制台应用程序782在第二用户交互空间的控制台窗口882中显示在移动OS130上运行的应用程序754。特别地,应用程序752和754可以是为移动OS130编译的、在移动OS运行时环境内运行并通过移动OS框架接收输入事件的任何应用程序,而没有针对显示或者远程交互(不在移动OS130的用户交互空间上)的修改。

[0122] 图13图示了根据各个实施例的向交叉环境应用程序提供用户交互支持的OS架构

配置 300b。在 OS 架构配置 300b 中，共享内核 320 中的设备驱动器实现组成辅助终端环境 840 的 I/O 设备 844、846 和 / 或 848 的硬件接口。通过设备驱动器，这些设备的输入事件出现在共享内核 320 的 I/O 设备 1360 的设备 1364、1366、和 / 或 1368 中。因为共享内核 320 的系统资源对移动 OS130 和桌面 OS160 二者可用，移动 OS 确定输入设备 1364、1366、和 / 或 1368 上的输入事件是否旨在用于移动 OS130。当桌面 OS160 被配置为提供第二用户交互空间(即，移动计算设备 110 被对接到辅助终端环境 840)时，移动 OS130 忽略来自与第二用户交互空间相关联的输入设备的输入事件。在 OS 架构配置 300b 中，移动 OS130 忽略来自设备 1364、1366、和 / 或 1368 的输入事件。在这种情况下，桌面 OS160 接受来自设备 1364、1366、和 / 或 1368 的输入事件。

[0123] 桌面 OS160 处理来自设备 1364、1366、和 / 或 1368 的输入事件并且确定在桌面 OS160 内怎样将事件分发到各种窗口或者 GUI 元素。例如，桌面 OS160 的窗口系统 774 可以接受来自设备 1364、1366、和 / 或 1368 的输入事件。在一个实施例中，桌面 OS160 的图形系统是 X-windows 图形系统。

[0124] 在一个实例中，用户在控制台应用程序 782 的控制台窗口 882 内的屏幕位置处点击定点设备的按钮。对应的输入事件出现在设备 1368 中并且桌面 OS160 接收并且处理输入事件。窗口系统 774 将输入事件指向到控制台应用程序 782，通过控制台应用程序 782，使用交叉环境呈现的实施例来显示在移动 OS130 上运行的应用程序 754。如上所述，应用程序 754 是移动应用程序并且使用移动 OS130 的库来将视图和其它事件句柄实例化，所述移动 OS130 的库接收来自移动 OS130 的输入队列 736 的输入以便接受用户输入。为了以应用程序 754 将合适地解释输入事件的方式而将输入事件呈现给应用程序 754，控制台应用程序 782 将输入事件映射到移动 OS130 的图形上下文并且通过虚拟输入设备 1370 将事件传送给移动 OS130 的输入队列 736。虚拟输入设备 1370 对于移动 OS130 表现为具有用于移动 OS130 的合适的输入事件协议的输入设备。另外，控制台应用程序 782 将输入事件相对于移动 OS130 的图形上下文进行格式化。移动 OS130 将虚拟输入设备 1370 与应用程序 754 相关联使得应用程序 754 从移动 OS 事件队列中接收输入事件。例如，移动 OS130 可以将虚拟输入设备 1370 与应用程序 754 的虚拟显示器 ID 相关联。以此方式，应用程序 754 接收并且处理输入事件，就仿佛通过移动 OS130 的用户交互空间来显示应用程序 754 并且与之交互。

[0125] 图 14 图示了根据各个实施例的、使用非扩展的图形上下文呈现的交叉环境应用程序的用户交互支持的方面。当从连接到移动计算设备 110 的输入设备(例如，键盘 846、定点设备 848)接收输入事件时，方法 1400 在块 1402 开始。如上所述，输入事件可以出现在共享内核的输入设备中。在块 1404，移动 OS 确定移动计算设备 110 是否对接并且桌面 OS 是否与输入设备相关联。例如，如果移动计算设备 110 没有与辅助终端环境对接并且桌面 OS 被挂起(suspend)，则移动 OS 可以确定桌面 OS 没有与输入设备相关联。如果桌面 OS 被挂起或者输入设备不是与桌面 OS 相关联的辅助终端环境的一部分，则在块 1406 移动 OS 接受来自输入设备的输入命令。

[0126] 如果桌面 OS 没有被挂起并且输入设备是与桌面 OS 相关联的辅助终端环境的一部分，则移动 OS 忽略输入设备上的输入事件并且桌面 OS 在块 1408 接受输入事件。在块 1410，桌面 OS 将输入事件分发到桌面 OS 内的合适的窗口或者 GUI 元素。如果输入事件没有被指

向到控制台窗口，则在块 1412 该输入事件被指向到另一窗口或者 GUI 元素。如果输入事件被指向到控制台窗口(例如，用户点击控制台窗口区域内的定点设备)，则在块 1414 该输入事件被传送给控制台应用程序作为一输入事件。

[0127] 在块 1416，为来自控制台应用程序的输入事件生成虚拟的输入设备。可以在共享内核中生成虚拟的输入设备或者可以将输入事件直接写入到移动 OS 输入设备。在块 1418，控制台应用程序将输入事件映射到移动 OS 的图形上下文。例如，控制台应用程序可以将输入事件在控制台应用程序的控制台窗口中的位置映射到移动 OS 的图形上下文内的位置。控制台应用程序可以针对移动 OS 的输入格式或者输入模式状态对输入事件进行转译。例如，输入格式可以在桌面 OS 和移动 OS 之间不同。即使具有公共的输入格式，桌面 OS 和移动 OS 可以具有不同的输入模式状态。例如，桌面 OS 可以使用非触摸模式输入状态处理输入事件，而移动 OS 是触摸模式输入状态。在这种情况下，可以对桌面 OS 用户交互空间中由定点设备生成的输入事件进行转译以便表现为虚拟输入设备中的基于手势的事件。

[0128] 在块 1420，在移动 OS 中将虚拟输入设备与用于在控制台窗口内显示的应用程序的虚拟显示设备相关联。例如，多个应用程序可能正在移动 OS 上运行并且在桌面 OS 的用户交互空间上的不同控制台窗口内显示。在桌面 OS 的用户交互空间中的单独的控制台窗口内显示的每个应用程序在移动 OS 内被分配了虚拟显示器 ID。因此，当移动 OS 通过虚拟设备接收到来自桌面 OS 的控制台应用程序的输入事件时，移动 OS 可以通过虚拟显示器 ID 将虚拟设备映射到正确的应用程序。在块 1422，输入事件被传送给与虚拟显示器相关联的应用程序并且应用程序可以处理输入事件，仿佛其通过移动 OS 的用户交互空间出现。特别地，上述方法对移动 OS130 的任何应用程序起作用，应用程序不需要被具体指定为通过桌面 OS 的控制台应用程序接受输入事件。

[0129] 扩展的渲染上下文实施例

[0130] 一些实施例通过在第一操作系统内建立扩展的渲染上下文来处理跨越多个 OS 的并发的用户界面支持。如上所讨论的，第一 OS(例如，移动 OS，Android) 典型地被配置为利用单个活动的渲染上下文来定义单个的、活动的用户交互空间。在此描述了新颖的技术，用于通过将多个所谓的“上下文空间”平铺(tile)到单个的、扩展的呈现空间并且将每个上下文空间与不同的显示器相关联来有效地欺骗第一 OS 并发地处理多个用户交互空间。实施例包括用于处理对多个用户交互空间的显示输出的技术和用于处理这些上下文中的用户交互的技术。

[0131] 返回到以上参照非扩展的渲染上下文实施例讨论的示例，再次假设日历应用程序和字处理应用程序二者被编译用于第一 OS(例如，移动 OS，Android OS) 并且二者并发地在移动设备上的第一 OS 内运行。第二 OS(例如，非移动 OS，如 Hydroid) 使用共享内核正在移动设备上并发地运行。用户已经将移动设备与第二计算环境(桌面计算环境)对接，并且桌面计算环境与第二 OS 的用户交互空间相关联并且正在显示第二 OS 的用户交互空间。用户期望通过第二 OS 的桌面计算环境与在第一 OS 上运行的字处理应用程序进行交互。期望以对用户透明的方式使用移动计算环境(即，移动设备)的第二 OS 来处理桌面计算环境的用户交互空间。

[0132] 图 15 图示了根据各个实施例的、使用扩展的渲染上下文跨越多个 OS 的并发用户界面支持的方面。在图 15 图示的应用程序呈现图示 1500 中，第一应用程序 1510(例如，日

历应用程序）计算对第一 OS 内第一表面 1512 的更新。通过第一 OS 将第一表面 1512 存储在共享的存储空间中的第一存储位置。具体地，第一应用程序 1510 更新第一表面 1512 的后缓冲区 1514。类似地，第二应用程序 1530（例如，字处理应用程序）使用第一操作系统计算对第二表面 1532 的更新。再次，使得第二应用程序 1530 对第二表面 1532 的更新到后缓冲区 1534。通过第一 OS 将第二表面 1532 存储在共享的存储空间中的第二存储位置。

[0133] 第一 OS 确定表面信息何时被改变并且发起呈现序列。第一 OS 可以在包括第一表面 1512 和第二表面 1532 的所有表面上执行单循环，确定与特定应用程序相关联的表面信息何时已经改变。在呈现序列中，第一 OS 确定表面信息何时被改变并且互换第一表面 1512 的前缓冲区 1516 和后缓冲区 1514，以及第二表面 1532 的前缓冲区 1536 和后缓冲区 1534。第一 OS 在共享的存储空间中的第三存储位置建立扩展的渲染上下文 1520 并且将第一表面 1512 呈现到扩展的渲染上下文 1520 的第一上下文空间 1522。第一 OS 将第二表面 1532 呈现到扩展的渲染上下文 1520 的第二上下文空间 1524。

[0134] 在实施例中，扩展的渲染上下文 1520 可以在存储空间中与第一 OS 的帧缓冲区重叠。例如，扩展的渲染上下文 1520 的第一上下文空间 1522 的存储位置可以与第一 OS 的帧缓冲区共同扩张（coextensive）。第一 OS 向第二 OS 传送通知在第三存储位置呈现最终图像。例如，第一 OS 可以通过进程间通信信道向第二 OS 的控制台应用程序传送对扩展的上下文 1520 的第二上下文空间 1524 的共享存储位置的文件描述符。第二 OS 访问第三存储位置处的扩展的图形上下文的第二部分以便取得呈现的图形帧 1538 以供在第二 OS 的控制台窗口 1540 中显示。例如，第二 OS 的控制台应用程序可以将文件描述符映射到其处理空间并且从第三存储位置读取呈现的图形帧以供在第二 OS 的用户交互空间内显示。以此方式，第二 OS 实时地在其用户交互空间内显示第二应用程序的呈现的图形帧。

[0135] 图 16 示出了根据各个实施例的使用扩展的渲染上下文进行交叉环境呈现的例示性的方法的流程图 1600。实施例保持了第一应用程序（例如，日历应用程序）和第二应用程序（例如，字处理应用程序）的应用程序图形的显示。假定这两个应用程序被编译用于第一操作系统（例如，Android OS）并且处于第一操作系统内的活动的并发执行中，但是用户期望通过与第二 OS 相关联的第二计算环境与第二应用程序进行交互。特别地，这些技术可以应用于其中两个 OS 不兼容（例如，为第一 OS 编译的应用程序可能不能直接在第二 OS 上执行）的环境。在一些实现方式中，如上所述，两个 OS 在移动设备的共享内核上独立地并且并发地运行。

[0136] 方法 1600 在块 1604 开始，建立第一 OS 的扩展的渲染上下文。如上讨论的，根据单个的活动显示器的特征典型地建立渲染上下文。然而，建立扩展的渲染上下文以便使得第一上下文空间与第一应用程序相关联并且第二上下文空间与第二应用程序相关联。第一和第二上下文空间是非重叠的。

[0137] 在一些实施例中，通过平铺本地设备的活动显示器（例如，具有共享内核的移动设备的显示器）和任何虚拟显示器（即，与在第二 OS 的用户交互空间内显示的控制台窗口相关联的第一 OS 的显示器）来生成扩展的渲染上下文，以形成对于第一 OS 看上去为一个大的显示器。扩展的渲染上下文的区域被指定为非重叠的上下文空间以便保持它们与其各自的物理或者虚拟显示器的关联性。特别地，在一些实现方式中，不同的上下文空间可能具有不同的分辨率或者其它特征。此外，在某些实施例中，上下文空间不是连续的。例如，以如下方式

来建立扩展的上下文，即：在每个未被指定给任何上下文空间的上下文空间之间留出空间。

[0138] 在块 1608，使用第一操作系统来计算对第一应用程序和第二应用程序的表面的更新。然后在块 1612 使用第一操作系统来呈现更新的表面以在可由（例如可以在共享的内核上并发地运行的）第一操作系统和第二操作系统访问的共享存储位置中生成扩展的图形帧。扩展的图形帧的第一部分与（与第一应用程序相关联的）第一上下文空间相关联以及扩展的图形帧的第二部分与（与第二应用程序相关联的）第二上下文空间相关联。当在块 1608 进行呈现时，将第一应用程序的更新的表面呈现给扩展的图形帧的第一部分，并且将第二应用程序的更新的表面呈现给扩展的图形帧的第二部分。值得注意的，以此方式，扩展的图形帧有效地包括被平铺到其合适的上下文空间中的两个应用程序的呈现的表面。

[0139] 在一些实施例中，在块 1616，向与第一 OS 相关联的第一计算环境的（多个）显示器显示来自共享存储位置的与第一上下文空间相关联的扩展的图形帧的第一部分。例如，如上讨论的，共享的存储位置是移动设备的帧缓冲存储器（或者被拷贝到帧缓冲存储器），移动设备的显示设备驱动器访问帧缓冲存储器以便向其（多个）显示器显示扩展的图形帧的第一部分。此外，在一些实施例中，在块 1620，向与第二操作系统相关联的第二计算环境的显示器显示来自共享存储位置的与第二运动空间相关联的扩展的图形帧的第二部分。例如，如上讨论的，将共享存储位置拷贝到与第二（例如桌面）计算环境相关联的第二 OS 的帧缓冲存储器，并且显示设备驱动器将扩展的图形帧的第二部分显示给第二计算环境的显示器。

[0140] 在参照图 12 讨论的实施例中，通过第二 OS 远程地执行对第二应用程序的更新的图形的呈现。在参照图 15 讨论的实施例中，通过移动设备的呈现引擎本地地执行对两个应用程序的呈现，但是连续地建立并且拆解渲染上下文。关于图 16 讨论的实施例允许通过移动设备的呈现引擎本地地执行两个应用程序的呈现，同时保留单个的、尽管被扩展的、呈现上下文（即，没有拆解渲染上下文）。

[0141] 图 17 示出了根据各个实施例的、使用扩展的渲染上下文进行交叉环境呈现的另一例示性方法的流程图 1700。如在图 16 的方法 1600 中，实施例保持第一应用程序和第二应用程序的应用程序图形的显示，第一应用程序和第二应用程序二者被编译用于第一操作系统并处于第一操作系统的活动的并发执行中。方法 1700 通过在块 1704 建立第一操作系统的扩展的渲染上下文并且在块 1708 使用第一操作系统计算对第一应用程序和第二应用程序的表面的更新而开始。如上讨论的，建立扩展的渲染上下文以使得第一上下文空间与第一应用程序相关联并且第二上下文空间与第二应用程序相关联。第一和第二上下文空间是非重叠的。在一些实现方式中，以分别与块 1604 和 1608 基本上相同的方式来执行块 1704 和 1708。

[0142] 在块 1712，使用第一操作系统根据第一上下文空间呈现第一应用程序的更新的表面，以便在第一操作系统的帧缓冲区中生成第一图形帧。例如，第一上下文空间可以与特定的分辨率、特定的平铺偏移（例如，开始“X”位置）等等相关联。在一些实现方式中，以与图 16 的方法 1600 中扩展的图形帧的各个部分的生成基本上相同的方式来生成第一图形帧。在一些实施例中，在块 1716，从帧缓冲区向与第一操作系统相关联的第一计算环境的显示器显示与第一上下文空间相关联的第一图形帧。

[0143] 在块 1720，使用第一操作系统呈现第二应用程序的更新的表面，以便在共享的存

储位置生成第二图形帧。如上讨论的，共享的存储位置可由(例如，可以在共享的内核上并发地运行的)第一操作系统和第二操作系统访问。在一些实施例中，在块 1724，从共享的存储位置向与第二操作系统相关联的第二计算环境的显示器显示与第二运动空间相关联的第二图形帧。

[0144] 特别地，图 16 和图 17 二者的实施例为每个应用程序建立具有上下文空间的扩展的渲染上下文。然而，图 16 的方法 1600 将所有的图形更新呈现到单个的扩展位图中，图 17 的方法 1700 将图形更新呈现到单独的若干位图中。例如，依赖于如何管理和 / 或访问存储器，可能期望一种或者其它技术。

[0145] 如同图 12 的实施例，在图 16 和 17 的实施例中，使用第一 OS 的呈现引擎呈现两个应用程序的更新的图形帧。使用第一 OS 的呈现引擎允许两个应用程序使用在第一 OS 中可用的移动设备的硬件加速能力。例如，在图 12、16 和 / 或 17 的实施例中，可以通过使用 2D 或者 3D 硬件辅助呈现来呈现第一和第二应用程序中任一个或者二者。

[0146] 使用扩展的渲染上下文对第一 OS (即，移动 OS，Android) 上运行的应用程序的图形帧进行远程显示对于第一 OS 提供了一种用于对使用单个渲染上下文对在多个用户交互空间内显示的多个应用程序提供呈现的方式。然而，扩展的渲染上下文产生了要处理通过扩展的渲染上下文显示的应用程序的输入事件的问题。具体地，必须配置第一 OS 的输入队列来处理来自通过扩展的渲染上下文的单独的虚拟显示器显示的多个应用程序的多个输入事件。

[0147] 交叉环境用户交互支持的实施例针对处理在第一 OS 上运行的并且通过第一 OS 的扩展的渲染上下文而在多个单独的用户交互空间(即，移动设备用户交互空间和桌面 OS 用户交互空间)上显示的多个应用程序的用户输入事件。实施例包括扩展的输入队列，其中将来自用于远程显示的应用程序的虚拟输入设备的输入事件映射到输入队列内的单独的运动空间。例如，第一应用程序(例如，日历应用程序) 正在第一 OS 上运行并且正在被显示给与第一设备相关联的第一显示器(例如，第一 OS 正在其上运行的移动设备的显示器)。第二应用程序(例如，字处理应用程序) 也正在与第一应用程序并发地运行，但是被呈现在扩展的渲染上下文的上下文空间 (即，虚拟显示器) 中并且被显示在第二 OS 的第二用户交互空间上，第二 OS 与第一 OS 在移动设备的共享内核上并发地运行。第一 OS 通过包括第一上下文空间(即，移动设备显示器) 中的第一应用程序和第二上下文空间中的第二应用程序二者的应用程序图形的扩展的渲染上下文来呈现图形帧。通过在第二 OS 上运行的控制台应用程序在第二 OS 的用户交互空间上显示第二上下文空间。

[0148] 通过第二 OS(即，桌面 OS，Hydroid) 接收通过扩展的渲染上下文远程显示的应用程序的输入事件，并且以与以上关于非扩展图形上下文描述的相同的方式通过第二 OS 的控制台应用程序将其传送给虚拟输入设备。然而，如上所述，在移动 OS 中从虚拟输入设备接收的输入事件与在第二 OS 的用户交互空间内显示的控制台窗口有关。虚拟输入设备被映射到与对应于远程显示的应用程序的虚拟显示器相关联的扩展的输入队列内的运动空间。扩展的输入队列允许第一 OS 使用单个输入队列正确地处理来自旨在用于多个并发执行的应用程序的多个本地和虚拟输入设备的输入。

[0149] 图 18a 和图 18b 图示了根据各个实施例的使用扩展的渲染上下文的交叉环境应用程序的用户交互支持的方面。图 18a 图示了正在远程地显示在第一 OS 上运行的应用程序

的用户交互空间 1810（例如，显示在第一 OS 上运行的应用程序的第二 OS 的 GUI）。例如，第一、第二和第三应用程序可以正在第一 OS 上运行（即，在第一 OS 的活动的并发执行中）。根据上述的实施例，第一 OS 可以在第一 OS 的扩展的渲染上下文的第一、第二和第三上下文空间中显示第一、第二和第三应用程序。用户交互空间 1810 中的控制台窗口 1812 和 1814 可以分别显示在第一 OS 上运行的第二应用程序和第三应用程序。

[0150] 图 18b 图示了用于提供对在第一 OS 上运行的每个应用程序的用户交互支持的第一 OS 的扩展的输入队列 1840。扩展的输入队列 1840 包括第一运动空间 1842、第二运动空间 1844，和第三运动空间 1846。第一运动空间与第一 OS 的扩展的渲染上下文的第一上下文空间相关联，其典型地用于呈现第一 OS 的非虚拟显示器 1852（即与移动计算设备 110 的（多个）显示器 116 相关联的上下文空间）。第二和第三运动空间分别与虚拟显示器 1854 和 1856 相关联，虚拟显示器 1854 和 1856 通过第二和第三上下文空间呈现。

[0151] 当出现要被指向到远程显示的应用程序的控制台窗口的输入事件时，该输入事件被指向到与虚拟显示器相关联的运动空间，通过所述虚拟显示器显示应用程序。例如，如果用户在用户交互空间 1810 的控制台窗口 1812 内利用定点设备点击，如输入事件 1820 所指示的，第二 OS 的窗口系统将输入事件指向到与控制台窗口 1812 相关联的控制台应用程序。如上所述，控制台应用程序将输入事件映射到虚拟输入设备。然而，该输入事件与控制台窗口 1812 有关。如果将输入事件直接馈入到第一 OS 的输入队列，则该输入事件将不被指向到正确的应用程序事件句柄。因此，来自虚拟输入设备的输入事件 1820 被重新映射到第二运动空间 1844。以此方式，扩展的输入队列将输入事件指向到接收并处理输入事件 1820 的第二应用程序的事件句柄。

[0152] 在实施例中，虚拟显示器在移动 OS 输入队列中被偏移。例如，在图 18b 中，虚拟显示器 1854 和 1856 在移动 OS 输入队列 1840 中被偏移了虚拟显示器偏移 1858。虚拟显示器偏移 1858 防止虚拟显示器在输入队列中相邻地出现，虚拟显示器在输入队列中相邻地出现可能导致旨在用于一个虚拟显示器的输入事件被在与不同应用程序相关联的运动空间内解释。虚拟显示器偏移应该足以大到根本不能用作实际的虚拟显示器分辨率参数。在一个实施例中，虚拟显示器偏移 1858 被选择为 10000 个像素。

[0153] 图 19 图示了根据各个实施例的用于接收通过扩展的渲染上下文显示的交叉环境应用程序的输入事件的方法 1900。例如，方法 1900 可以用于处理在第一 OS 内运行的第一应用程序和第二应用程序的输入事件，第一应用程序被本地显示在第一 OS 的用户交互空间上，第二应用程序通过第一 OS 的扩展的渲染上下文被远程显示在第二 OS 的用户交互空间中。

[0154] 方法 1900 开始于块 1902，当在第一 OS 中接收到第一用户输入时，第一应用程序和第二应用程序处于第一 OS 内的活动的并发执行中，在与第一 OS 相关联的第一用户环境内显示第一应用程序，在与第二 OS 相关联的第二用户环境内显示第二应用程序，第一和第二操作系统在共享的内核上并发地运行，第一 OS 通过扩展的渲染上下文的第一虚拟显示器呈现第二应用程序的图形帧来保持第二应用程序的应用程序图形。在块 1904，第一 OS 建立包括第一运动空间和第二运动空间的扩展的输入队列，第二运动空间对应于第一虚拟显示器。例如，第一操作系统为第二应用程序分配第一虚拟显示器，建立具有第一上下文空间和第二上下文空间的扩展的渲染上下文，将第一虚拟显示器与第二上下文空间相关联，并且

使用上述技术通过扩展的渲染上下文的第二上下文空间来呈现第二应用程序的图形帧。

[0155] 在块 1906，第一 OS 在第一虚拟输入设备处接收来自第二 OS 上运行的第一控制台应用程序的第一用户输入事件，通过第二 OS 正在显示第二应用程序的呈现的图形帧。在块 1908，将第一虚拟输入设备映射到第一操作系统的扩展的输入队列的第二运动空间。将第一虚拟输入设备映射到第二运动空间使得第一 OS 的扩展的输入队列正确地将来自第一虚拟输入设备的输入事件关联到第二应用程序的视图内的事件句柄上。具体地，当输入事件被映射到第二运动空间时，第一 OS 将输入事件视为出现在扩展的输入队列中与第二应用程序相关联的位置。在块 1910，第一 OS 将第一用户输入事件传送给来自被映射的第一虚拟输入设备的第二应用程序。扩展的输入队列使用被扩展的渲染上下文的平铺的属性以使得输入队列处理来自多个用户交互空间的多个输入事件并且将输入事件指向到想要的应用程序的合适的事件句柄。

[0156] 镜像的上下文实施例

[0157] 以上在保持对跨越多个操作系统上的多个应用程序的并发用户交互空间支持的背景下描述了扩展的和非扩展的渲染上下文的实施例。在许多情况下，期望镜像单个用户交互空间的上下文。期望在与第二 OS 相关联的第二计算环境(例如，与 Hydroid OS 相关联的桌面环境)中对第一 OS 进行并发地观看和交互(即，“镜像”交互空间)。通过镜像的用户交互空间，用户可以与第一 OS 交互，仿佛通过本地设备进行交互(即，用户可以浏览第一 OS 可用的应用程序、开始和停止应用程序，使用搜索能力，等等)。

[0158] 第一 OS(例如，移动 OS，Android) 典型地被配置为定义单个的、活动的用户交互空间。用户交互空间包括一活动的显示器(例如，具有相关联的特征，如分辨率)以及一个或多个允许与在活动显示器上显示的元素进行用户交互的活动的输入设备。提出了新颖的技术来使用交叉环境呈现以提供跨越多个 OS 的一个或多个镜像的用户交互空间。如上讨论的，即使在多个 OS 不兼容和 / 或在共享的内核上独立并且并发地运行的情况下，实施例也可操作。

[0159] 可以使用以上关于保持对交叉环境应用程序的并发用户交互支持所参照的许多相同的系统元素来完成保持具有镜像上下文的并发用户交互支持。例如，参照图 7，移动 OS130 的图形上下文可以是主动地显示应用程序(例如，应用程序 752 和 / 或 754)和 / 或移动 OS130 的主屏幕(例如，Android OS130 的主屏幕应用程序 451)。用于主动显示的应用程序和 / 或移动 OS 的主屏幕的表面信息可以被存储在共享的存储器 724 内。可以通过控制台应用程序 782 在桌面 OS160 的用户交互空间内显示移动 OS130 的镜像的上下文。

[0160] 图 20 示出了根据各个实施例的、用于提供镜像的用户交互空间的图形上下文的交叉环境呈现的例示性方法的流程图 2000。方法 2000 开始于块 2004，使用第一操作系统计算对被编译用于第一操作系统并且处于第一操作系统内的活动的执行中的第一应用程序的一组表面的更新。例如，进行计算以确定表面形状、大小、纹理、分层等等的改变。然后使用第一操作系统在块 2008 呈现表面更新，以便生成图形帧。图形帧可以是反映应用的更新的图形信息的位图。

[0161] 在块 2012，将图形帧存储在第一操作系统和第二操作系统二者可访问的共享的存储位置。在一些实施例中，第一和第二 OS 正在共享的内核上并发地运行。在块 2016，可以使用第一操作系统将图形帧显示给第一计算环境的第一显示器上的第一应用程序的第一

应用程序显示。例如，共享的存储位置可以是帧缓冲存储器，或者可以被拷贝到第一操作系统的帧缓冲区。本地设备（例如，正在运行共享内核）的显示设备驱动器访问帧缓冲存储器以便显示位图。

[0162] 在块 2012 在共享的存储位置中存储图形帧之后，期望向第二 OS 通知更新的图形信息可用。在块 2020，传送文件描述符，向被编译用于第二 OS 并且处于第二 OS 内的活动执行中的控制台应用程序指示共享的存储位置。在一些实现方式中，文件描述符包括共享的存储位置的指示。在其它实现方式中，文件描述符包括附加信息，如指示更新的图形信息对于正被镜像的应用程序的可用性的标志。

[0163] 如上所述，控制台应用程序可以是 X-Windows 或者是在第二计算环境中的显示器的窗口内显示的类似类型的应用程序。在块 2024，控制台应用程序根据文件描述符访问共享的存储位置处的更新的图形信息（例如，位图），并且将来自共享的存储位置的图形帧显示给第二计算环境的第二显示器上的第一应用程序的第二应用程序显示。在一些实施例中，在第一和第二计算环境二者的显示器上基本上并发地显示应用程序的更新的图形信息。

[0164] 图 21 示出了根据各个实施例的、提供镜像的用户交互空间的图形上下文的交叉环境呈现的另一例示性方法的流程图 2100。如在图 20 中，方法 2100 开始于块 2104，使用第一操作系统计算对被编译用于第一操作系统并且处于第一操作系统内的活动的执行中的第一应用程序的一组表面的更新。在块 2108，将更新的该组表面存储在（例如，在共享的内核上并发地运行的）第一操作系统和第二操作系统二者可访问的共享的存储位置。

[0165] 在块 2112，利用第一操作系统呈现更新的该组表面以生成第一图形帧。然后，在块 2116，可以使用第一操作系统将第一图形帧显示给第一计算环境的第一显示器上的第一应用程序的第一应用程序显示。例如，移动 OS130 呈现更新的应用程序图形并且将更新的图形显示给移动设备 110 的（多个）显示器 116。

[0166] 在块 2108 在共享的存储器中存储更新的该组表面之后的任何时间，期望向第二 OS 通知更新的图形信息可用。在块 2120，传送文件描述符，向被编译用于第二操作系统并且处于第二操作系统内的活动执行中的控制台应用程序指示共享的存储位置。特别地，如在图 20 的方法 2000 中在共享的存储器中存储的信息是未被呈现的表面信息（例如，几何原语）而不是呈现后的比特。

[0167] 相应地，在块 2124，通过第二操作系统（例如，根据文件描述符经由控制台应用程序）呈现来自共享的存储位置的更新的该组表面以生成基本上与第一图形帧相同的第二图形帧。在块 2128，经由第二操作系统的控制台应用程序将第二图形帧显示给第二计算环境的第二显示器上第一应用程序的第二应用程序显示，使得第二应用程序显示基本上与第一应用程序显示相同。

[0168] 值得注意的，在第一和第二操作系统二者上复制呈现时，可能涉及附加开销。然而，在许多情况下，这种附加开销是值得的。例如，当不同计算环境的显示器具有略微不同的特征时，可能期望在适合于各个显示器的各个单独的渲染上下文中呈现更新的图形信息。

[0169] 图 20 和图 21 的方法描述了在镜像的图形上下文中利用在第一 OS 中运行的应用程序的活动显示进行图形上下文的交叉环境镜像。然而，这些方法可以用于其中在图形上

下文中没有主动地显示应用程序的情况下。例如，图形上下文可以是显示第一 OS 的主屏幕或者其它特征(例如，搜索屏幕等等)。在这些情况下，通过第一 OS 的组件更新图形上下文的表面信息，并且可以执行图 20 和图 21 的方法的其它步骤以便在第二应用程序显示中提供镜像的图形上下文。

[0170] 特别地，可以与应用程序的交叉环境呈现并发地采用图形上下文的交叉环境镜像。例如，可以使用根据图 20 或图 21 的方法，使用上述的应用程序的交叉环境呈现的技术，在第二用户环境中显示在第一 OS 上运行的应用程序的同时，将移动设备的用户交互空间的活动的图形上下文镜像到第二用户环境。为了进行例示，参照图 8，可以在第一控制台窗口 882 内显示移动 OS 的用户交互空间，而在显示器 844 上在桌面 OS 的用户交互空间内的第二控制台窗口 884 内显示移动 OS 应用程序。

[0171] 可以以与在图 13、14、18 和 / 或 19 中图示的对交叉环境应用程序提供用户界面支持基本上相同的方式执行对镜像的图形上下文提供用户交互支持。具体地，可以从第二 OS 的控制台应用程序向虚拟输入设备提供输入事件。第一 OS 可以通过扩展的或者非扩展的输入队列接收来自虚拟输入设备的输入事件。

[0172] 交叉环境重定向实施例

[0173] 上述技术提供了通过第二操作系统的用户交互空间对第一操作系统的应用程序和图形上下文的交叉环境用户交互支持。为了便于透明交叉环境使用模型，实施例针对从第二操作系统的用户交互空间对第一操作系统的应用程序和 / 或镜像的上下文提供访问。

[0174] 参照回到图 8，用户可以通过包括移动设备上的交互组件(即，(多个)触摸屏显示器 116，(多个)其它 I/O 设备 118)的第一用户交互空间与第一 OS(即，移动 OS，Android) 进行交互。用户还可以通过包括辅助终端环境 840 的显示器 844 的第二用户交互空间与第二 OS(即，桌面 OS，Hydroid) 进行交互。如上所述，可用于桌面 OS160(即，被编译用于桌面 OS160 并且加载到桌面 OS160 的执行环境中) 的一组应用程序可以与可用于移动 OS130 的一组应用程序不同。实施例针对于通过在桌面 OS160 的用户交互空间的菜单内提供用于移动 OS130 上可用的应用程序的菜单图标或者菜单列表条目而使得在桌面 OS160 的用户交互空间内可访问移动 OS130 的应用程序。

[0175] 图 22 图示了根据各个实施例的交叉环境重定向的方面。在图 22 图示的计算环境 2200 中，用户通过桌面 OS 用户交互空间 2202 与桌面 OS160 进行交互。在桌面 OS 用户交互空间 2202 内，菜单栏 2220 包括可用的应用程序的图标或者列表。为了启动应用程序，用户从菜单栏、或者菜单栏 2220 的下拉或弹出列表中选择应用程序名称或者图标。传统地，菜单栏 2220 仅仅包括在桌面 OS160 上可用的应用程序的菜单条目或者图标。例如，菜单条目 2222、2224、2226，和 / 或 2228 可以是桌面 OS160 上可用的应用程序(例如，被编译用于桌面 OS160 并且被加载到桌面 OS160 的执行环境中)。本发明的实施例针对提供从桌面 OS 用户交互空间 2202 对移动 OS130 的应用程序和 / 或图形上下文的交叉环境访问。例如，菜单条目 2232、2234、2236、22367 和 / 或 2238 可以指示在移动 OS130 上可用的应用程序和 / 或移动 OS130 的图形上下文。

[0176] 将桌面 OS 用户交互空间 2202 显示在与桌面 OS160 相关联的用户交互空间(例如，辅助终端环境 840)内的显示器上。桌面 OS160 的菜单栏 2220 包括与被编译用于桌面 OS160 并且在桌面 OS160 上加载(例如被编译用于 Hydroid/Linux 并且被加载到 Hydroid OS 的执

行环境内）的应用程序相关联的菜单条目 2222、2224，2226，和 / 或 2228。菜单栏 2220 还包括与被编译用于移动 OS130 并且在移动 OS130 上加载（例如被编译用于 Android 并且在 Android 执行环境内加载）的应用程序相关联的菜单条目 2234、2236，2237，和 / 或 2238。当用户选择菜单条目 2234、2236，和 / 或 2238 之一时，在移动 OS130 上启动相关联的应用程序并且在桌面 OS160 的控制台窗口内进行显示，例如，在桌面 OS 用户交互空间 2202 的窗口 2216 内显示。菜单条目 2232 可以与移动 OS 图形上下文相关联，使得如果选择了菜单条目 2232，则在桌面 OS160 的控制台窗口内显示移动 OS 的图形上下文。

[0177] 图 23 图示了为构建桌面 OS160 的菜单栏 2220 可以采用的处理流程 2300。在处理流程 2300 的步骤 2302，桌面 OS160 在移动 OS130 中查询可用应用程序的列表。在一个实施例中，桌面 OS160 的系统服务或者启动器（launcher）应用程序在移动 OS130 的服务中查询所有可启动的移动 OS 应用程序快捷方式。移动 OS130 利用可用的应用程序的列表（即，用于可用的移动 OS 应用程序的可启动的快捷方式）进行响应。可用应用程序的列表可以包括在移动 OS130 上所有可用的应用程序（所有在移动 OS130 上加载的并且可执行的应用程序）或者可用的移动 OS 应用程序的子集。例如，列表可以包括在移动 OS GUI 的（多个）应用程序菜单屏幕上出现的所有应用程序。在步骤 2304，桌面 OS160 接收来自移动 OS130 的应用程序的列表。由移动 OS130 返回的应用程序的列表包括列出的每个应用程序的应用程序分组名称，并且还可以包括列出的每个应用程序的应用程序名称和图标。

[0178] 桌面 OS160 通过在块 2306、2308 和 2310 上进行迭代对应用程序列表的每个应用程序在菜单栏 2220 中创建菜单条目。对于每个应用程序，桌面 OS160 在块 2306 对菜单栏 2220 中的应用程序的图标进行实例化，在块 2308 将图标与桌面 OS160 的控制台应用程序相关联，并且在块 2310 将指示应用程序的分组名称的参数与图标相关联。使用上述的交叉环境呈现的实施例，控制台应用程序在桌面 OS160 上运行并且在桌面 OS160 内显示应用程序的图形信息。以此方式，当用户选择菜单条目时，在桌面 OS160 上启动控制台应用程序，并且将应用程序的分组名称传送给控制台应用程序。

[0179] 桌面 OS160 可以以各种方式显示与移动 OS130 的应用程序相关联的菜单条目。菜单条目可以被显示在菜单栏 2220 中，或者被显示在当选择指示移动 OS 应用程序是可用的菜单条目时出现的下拉菜单中。在菜单栏 2220 或者下拉菜单中，可以使用图标或者仅仅应用程序名称来显示菜单条目。在一个实施例中，桌面 OS160 为移动 OS 应用程序显示单独的菜单栏。在另一个实施例中，与移动 OS 应用程序相关联的菜单条目与桌面 OS 应用程序的菜单条目并排或者混合出现在桌面 OS 菜单栏 2220 内。可选地，移动 OS 菜单条目可以处于定界符 2240 隔开的菜单栏 2220 的区域 2230 中或者以其它方式可被识别为包括移动 OS 菜单条目。菜单栏 2220 可以包括移动设备自身的活动显示的菜单条目，即，根据图 20 和 / 或图 21 的方法用户可以在桌面 OS 的用户环境内选择显示移动 OS 的用户交互空间的菜单条目。在一个实施例中，在可用应用程序的列表中返回移动 OS 的主屏幕应用程序并且向其提供相关联的菜单条目。

[0180] 当用户选择与移动 OS 应用程序相关联的菜单条目时，桌面 OS160 启动与菜单条目相关联的控制台应用程序并且将应用程序的分组名称传送给控制台应用程序。控制台应用程序在桌面 OS 用户交互空间 2202 内显示窗口（即，控制台应用程序在桌面 OS 的图形系统内进行显示）。控制台应用程序向移动 OS130 发送请求以启动应用程序（即，请求移动 OS130

启动其分组名称作为执行参数被提供给控制台应用程序的应用程序）并且通过控制台应用程序显示应用程序的图形帧。应用程序可以是或者可以不是当前正在移动 OS130 上运行的。如果应用程序当前正在移动 OS130 上运行，则可以将应用程序的显示从移动 OS130 移动到桌面 OS 用户交互空间 2202，或者同时将其显示在移动设备的显示器和用户交互空间 2202 二者上。使用上述的交叉环境呈现和交叉环境用户界面支持技术的任一种可以为应用程序完成应用程序图形的显示和用户交互支持。

[0181] 图 24 图示了响应于用户在桌面 OS GUI880 的菜单栏 2220 上选择与移动 OS 相关联的菜单条目，移动 OS130 启动应用程序所遵循的处理流程 2400。处理流程 2400 开始于块 2402，移动 OS130 从桌面 OS160 接收请求以启动被编译用于移动 OS 并且在移动 OS 的执行环境内加载的应用程序，以供在桌面 OS 上进行显示。在块 2404，移动 OS 分配未使用的虚拟显示器 ID。例如，移动 OS 的图形系统可以保存虚拟显示器 ID 的列表并且将未使用的虚拟显示器 ID 分配给第一应用程序的处理。在块 2406，移动 OS 在移动 OS 内启动第一应用程序（即，在移动 OS 上运行）。在块 2408，移动 OS130 将第一应用程序的刷新通知与虚拟显示器相关联。例如，移动 OS 的图形服务器可以保存应用程序的列表和它们相关联的虚拟显示器。在块 2410 和 2412，移动 OS 通过监视第一应用程序的应用程序图形并且在第一应用程序的应用程序图形信息被更新时向桌面 OS 的控制台应用程序进行通知，来保持第一应用程序的图形信息。块 2410 和 2412 可以对应于根据图 10、12、16 和 / 或 17 的方法的保持交叉环境应用程序的应用程序图形。

[0182] 出于例示和描述的目的，已经提出了前面的描述。此外，该描述不是旨在将本发明的实施例限制为在此公开的形式。尽管以上已经讨论了多个示例性方面和实施例，但本领域技术人员将认识到其某些变型、修改、置换、增加、和子组合。

[0183] 可以通过能够执行对应功能的任何合适的手段来执行上述的方法的各个操作。这些手段可以包括（多个）各种硬件和 / 或软件组件和 / 或（多个）模块，包括但不限于电路、专用集成电路（ASIC），或者处理器。

[0184] 可以由被设计用于执行在此描述的功能的通用处理器、数字信号处理器（DSP）、ASIC、现场可编程门阵列信号 (FPGA)，或者其它可编程逻辑器件（PLD）、离散的门电路，或者晶体管逻辑电路、离散的硬件组件，或者其任何组合来实现或者执行所述的各种例示性的逻辑块、模块，以及电路。通用的处理器可以是微处理器，但是作为替换，该处理器可以是任何市场上可得到的处理器、控制器、微控制器，或者状态机。处理器还可以被实现为计算器件的组合，例如，DSP 和处理器的组合，多个微处理器，一个或多个连同 DSP 核的处理器，或者任何其它这样的配置。

[0185] 可以在硬件、由处理器执行的软件模块，或者二者的组合中直接体现连同本公开描述的方法或者算法的步骤。软件模块可以驻留在任何形式的有形的存储介质中。可以使用的一些存储介质的示例包括随机存取存储器（RAM）、只读存储器（ROM）、闪存存储器、EPROM 存储器、EEPROM 存储器、寄存器、硬盘、可移除盘、CD-ROM 等等。存储介质可以耦接到处理器使得处理器可以从存储介质中读取信息并且将信息写入到存储介质。作为替换，存储介质可以被集成到处理器中。软件模块可以是单指令，或者许多指令，并且可以分布到若干不同的代码段上、不同的程序中，以及多个存储介质上。

[0186] 在此公开的方法包括用于实现所描述的方法的一个或多个动作。方法和 / 或动作

可以与另一互换，而不脱离权利要求的范围。换言之，除非指定了动作的特定顺序，可以修改特定动作的顺序和 / 或用途，而不脱离权利要求的范围。

[0187] 可以以硬件、软件、固件或者其任何组合来实现所描述的功能。如以软件实现，所述功能可以被存储为在有形的计算机可读介质上的一个或多个指令。存储介质可以是能够由计算机访问的任何可用的有形介质。通过示例而不是限制，这种计算机可读的介质可以包括 RAM、ROM、EEPROM、CD-ROM、或者其它光盘存储装置、磁盘存储装置，或者其它磁盘存储设备，或者任何其它的可以用于以指令或者数据结构的形式承载或存储期望的程序代码并且可以通过计算机访问的有形介质。如在此使用的，磁盘和光盘包括致密盘（CD）、激光盘、光盘、数字多用途盘（DVD）、软盘，和蓝光〇 R 盘，其中磁盘（disk）通常以磁方式重现数据，而光盘 (disc) 以激光光学地重现数据。

[0188] 由此，计算机程序产品可以执行在此提出的操作。例如，这种计算机程序产品可以是其上有形存储（和 / 或编码）了指令的计算机可读的有形介质，指令可通过一个或多个处理器执行以执行在此描述的操作。计算机程序产品可以包括包装材料。

[0189] 软件或者指令还可以通过传输介质进行传输。例如，可以使用传输介质，诸如同轴电缆、光缆、双绞线、数字用户 (DSL)，或者诸如红外、无线电或者微波之类的无线技术，从网站、服务器，或者其它远程源传输软件。

[0190] 此外，可以下载和 / 或以其它方式由用户终端和 / 或基站（在可应用时）获得用于执行在此描述的方法和技术的模块和 / 或其它合适的部件。例如，这种设备可以耦接到服务器以便于传递用于执行在此描述的方法的部件。可替换地，可以经由存储部件（例如，RAM、ROM、物理存储介质，诸如 CD 或者软盘等等）来提供在此描述的各种方法，使得在将存储部件耦接到或者提供给设备时，用户终端和 / 或基站可以获得各种方法。此外，可以利用向设备提供在此描述的方法和技术的任何其它合适的技术。

[0191] 其它示例和实现方式在本公开和所附的权利要求的范围和精神内。例如，由于软件的特性，可以使用处理器、硬件、固件、硬连线，或者这些中任何组合执行的软件来实现上述的功能。实现功能的特征还可以位于各种位置，包括被分布使得在不同的物理位置处实现一部分功能。此外，如在此使用的，包括在权利要求中，如在以“至少一个”开始的一列条目中使用的“或”指示分离性的列表使得例如“A，B 或者 C 中的至少一个”的列表意味着 A 或 B 或 C 或 AB 或 AC 或 BC 或 ABC（即，A 和 B 和 C）。此外，用语“示例性”不是意味着所描述的示例是优选的或者比其它示例更好。

[0192] 可以做出对在此描述的技术的各种改变、替换和更改，而不脱离由所附权利要求定义的教导的技术。此外，本公开和权利要求的范围不限于上述的处理、机器、制造、物质的合成、手段、方法，和动作的特定方面。可以利用与在此描述的对应方面执行基本相同功能或者实现基本相同结果的目前存在的或者随后开发的处理、机器、制造、物质的合成、手段、方法或者动作。相应地，所附的权利要求在它们的范围内包括这种处理、机器、制造、物质的合成、手段、方法或者动作。

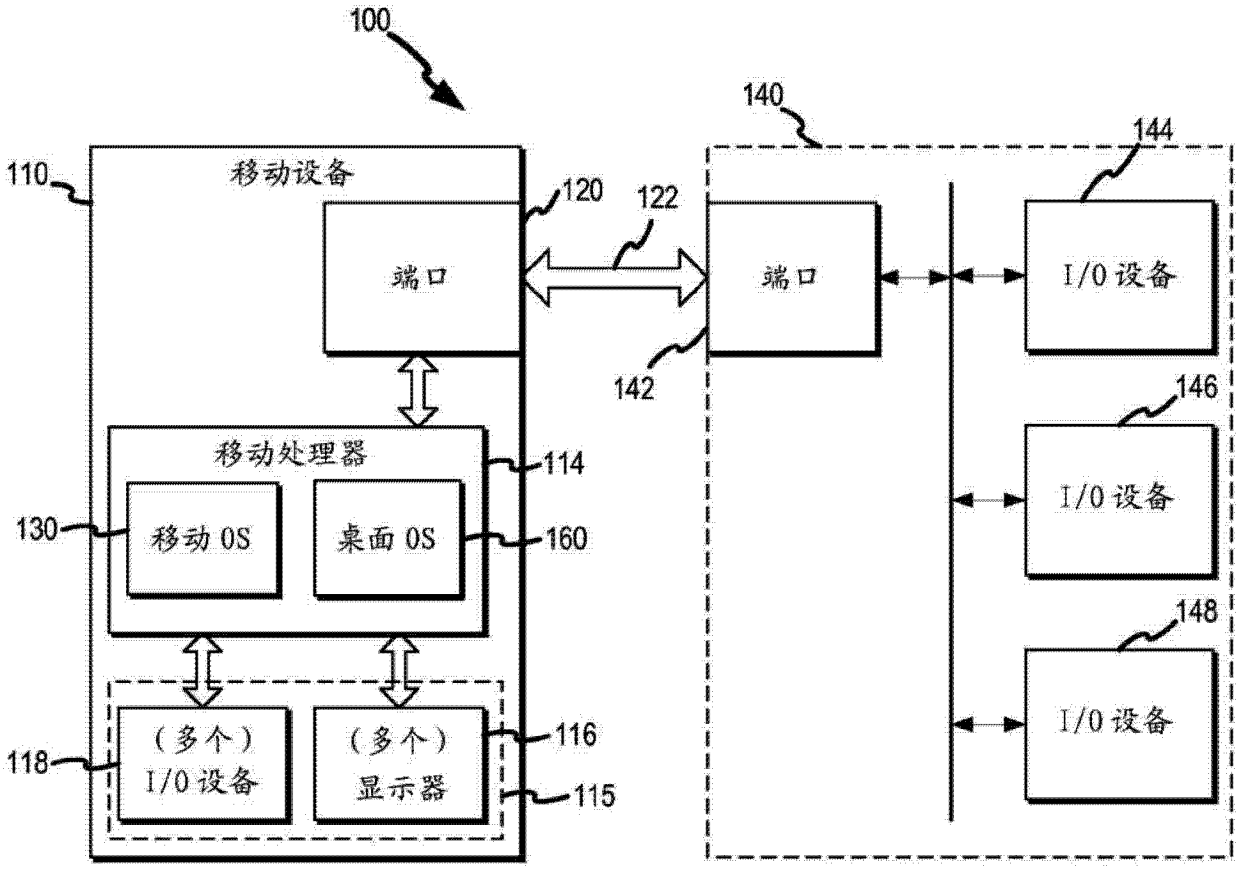

100
110
移动设备
120
122
140
144
端口
端口
I/O 设备
142
146
移动处理器
114
130
移动 OS
桌面 OS
160
I/O 设备
148
I/O 设备
118
（多个）
I/O 设备
（多个）
显示器
116
115

图 1

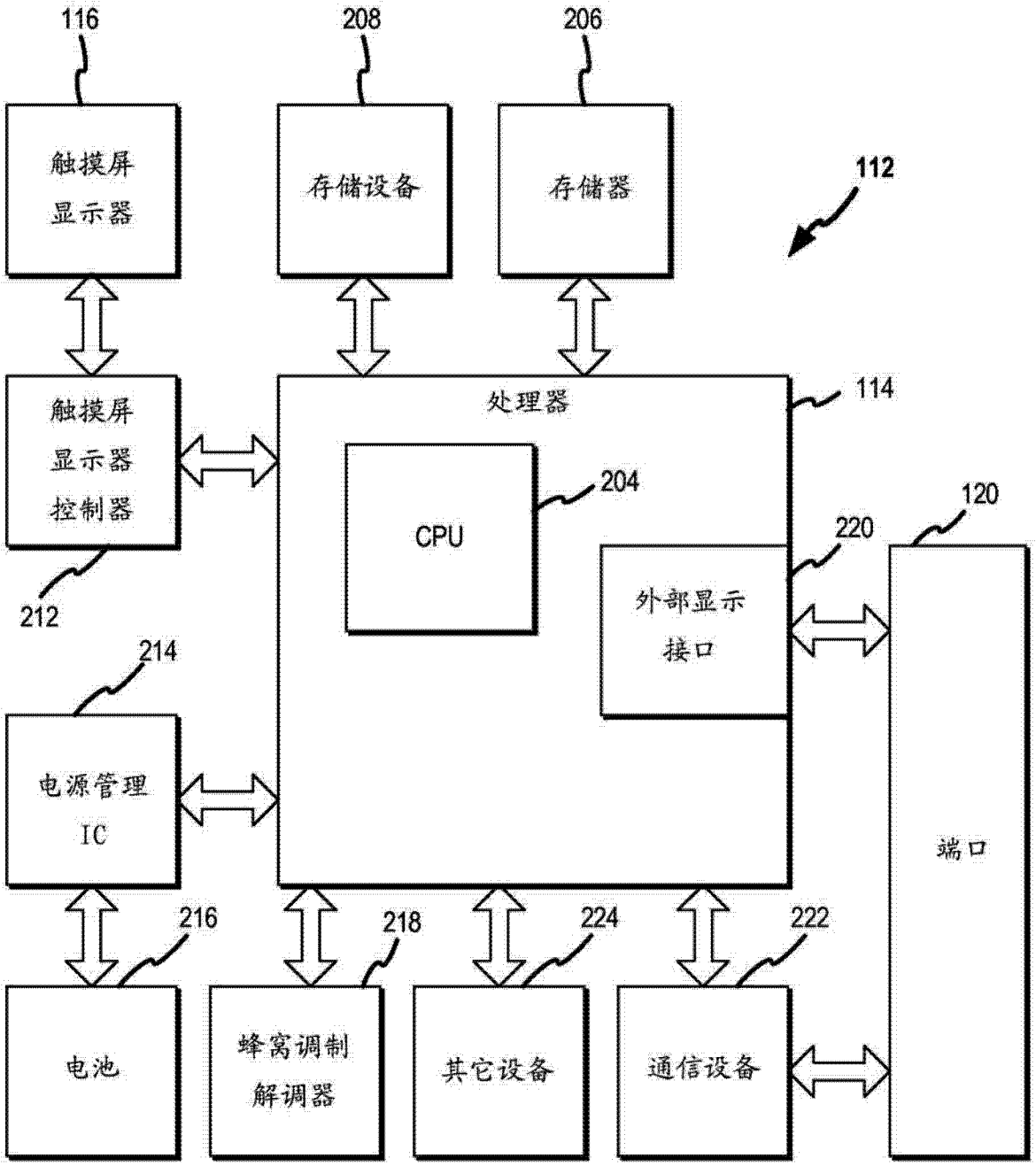

116
208
206
触摸屏
显示器
存储设备
存储器
112
触摸屏
显示器
控制器
处理器
114
CPU
204
120
220
外部显示
接口
212
214
电源管理
IC
端口
216
218
224
222
电池
蜂窝调制
解调器
其它设备
通信设备

图 2

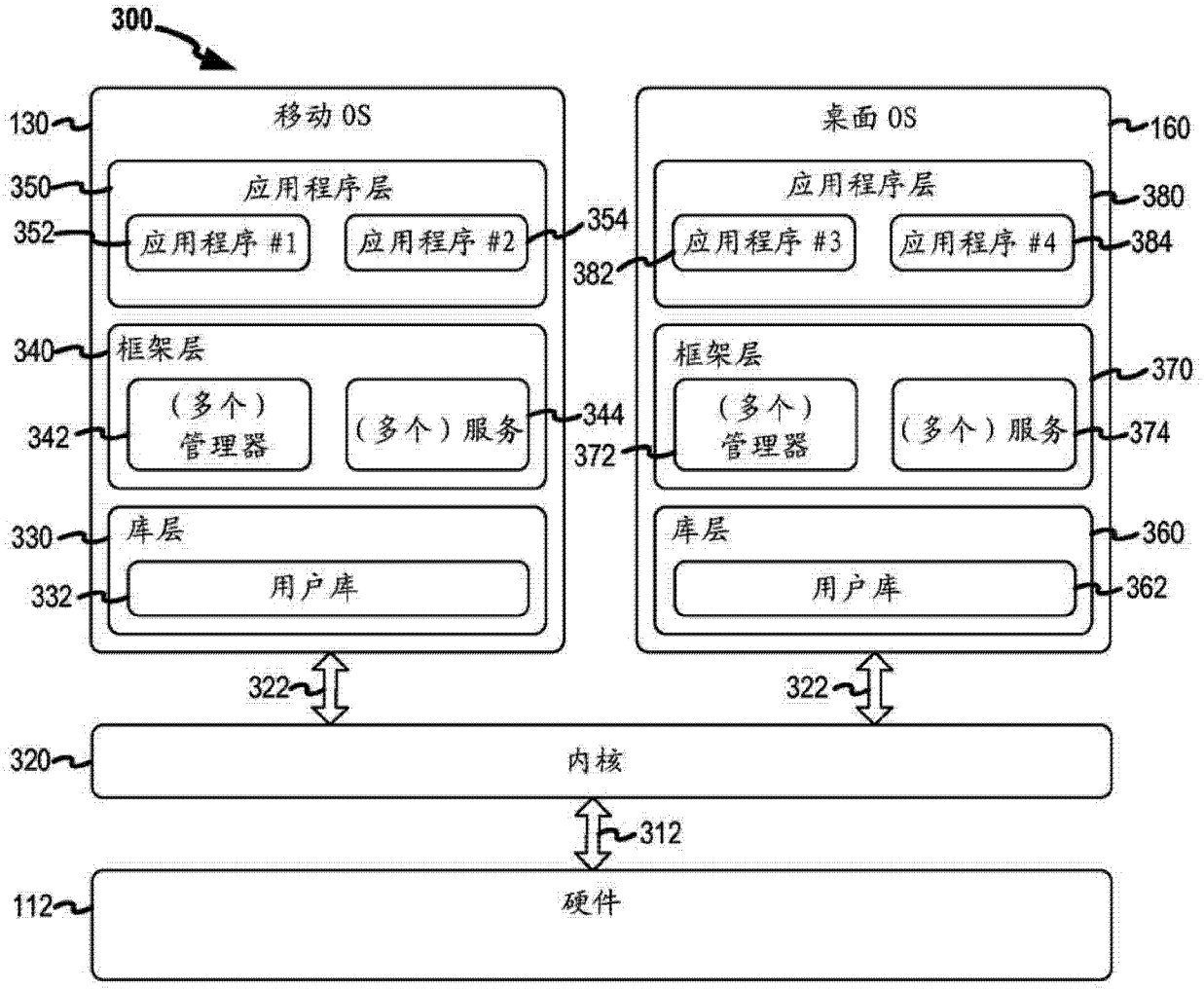

300
移动 OS
130
应用程序层
350
应用程序 #1
352
应用程序 #2
354
框架层
340
(多个)
管理器
342
(多个)服务
344
库层
330
用户库
332
桌面 OS
160
应用程序层
380
应用程序 #3
382
应用程序 #4
384
框架层
370
(多个)
管理器
372
(多个)服务
374
库层
360
用户库
362
322
322
320
内核
312
112
硬件

图 3

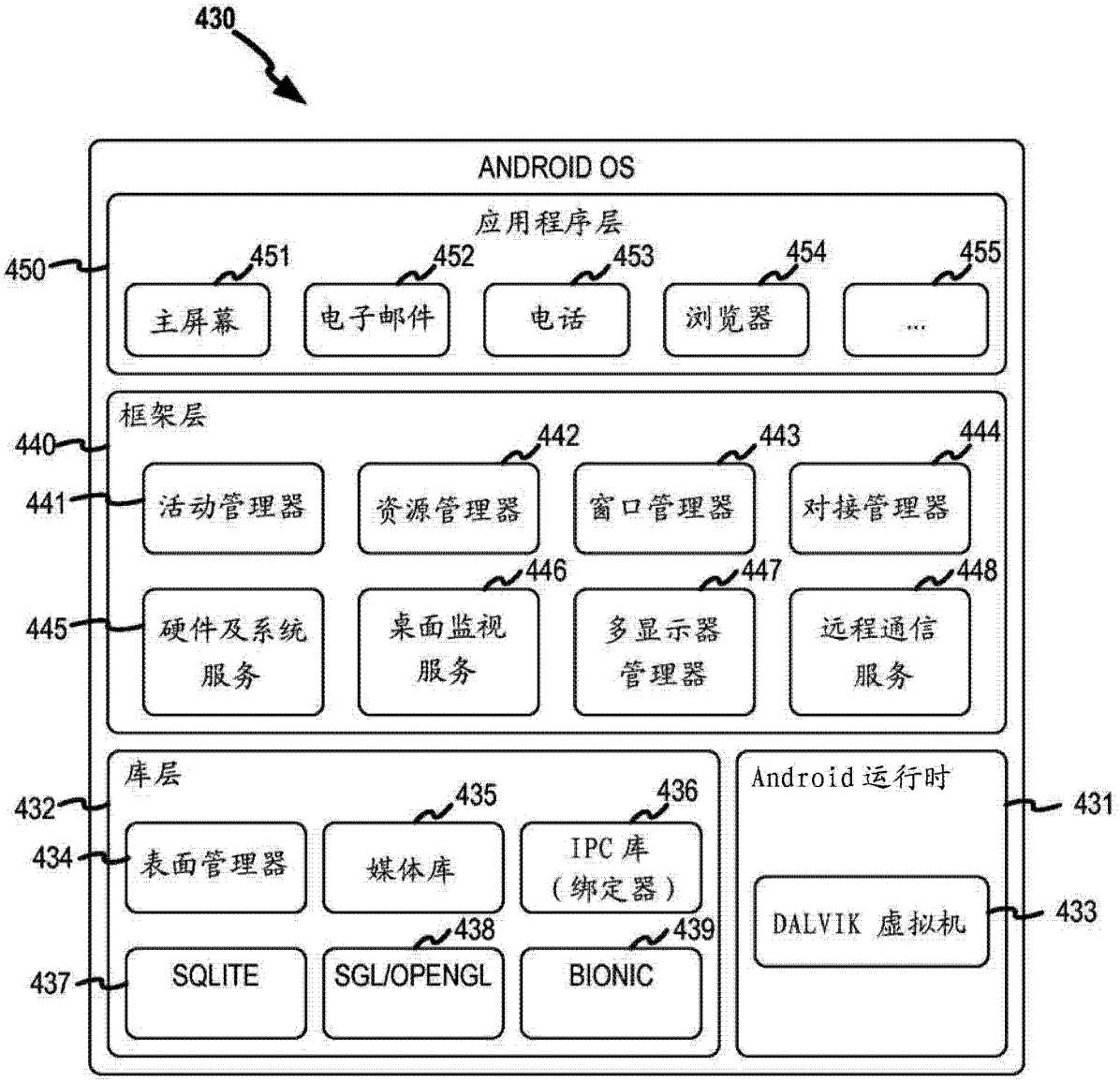

430
ANDROID OS
应用程序层
450
451
452
453
454
455
主屏幕
电子邮件
电话
浏览器
...
框架层
440
441
442
443
444
活动管理器
资源管理器
窗口管理器
对接管理器
445
446
447
448
硬件及系统服务
桌面监视服务
多显示器管理器
远程通信服务
库层
432
435
436
434
表面管理器
媒体库
IPC 库（绑定器）
438
439
437
SQLITE
SGL/OPENGL
BIONIC
Android 运行时
431
DALVIK 虚拟机
433

图 4

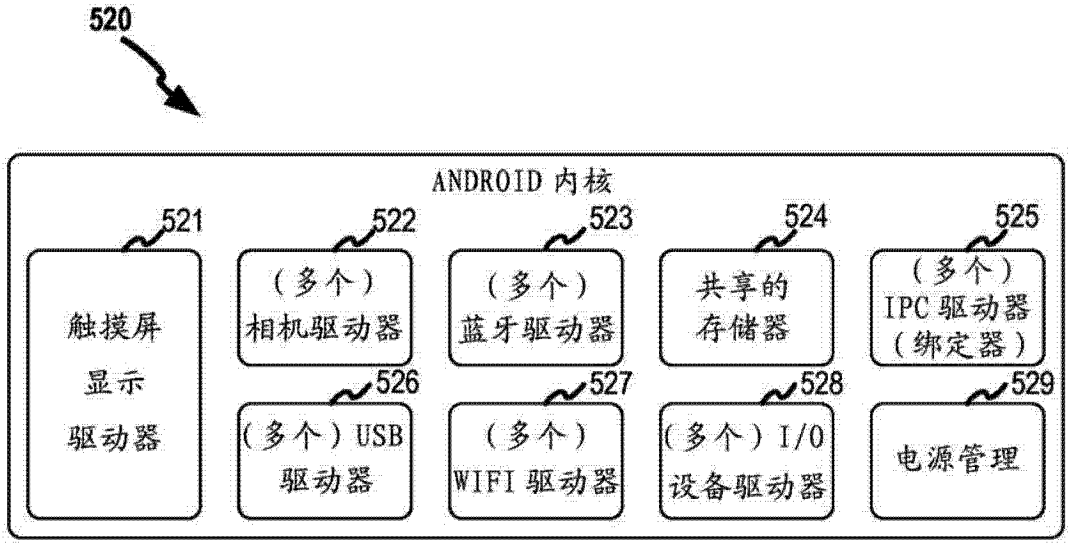

520
ANDROID 内核
521
触摸屏
显示
驱动器
522
（多个）
相机驱动器
523
（多个）
蓝牙驱动器
524
共享的
存储器
525
（多个）
IPC 驱动器
（绑定器）
526
（多个）USB
驱动器
527
（多个）
WIFI 驱动器
528
（多个）I/O
设备驱动器
529
电源管理

图 5

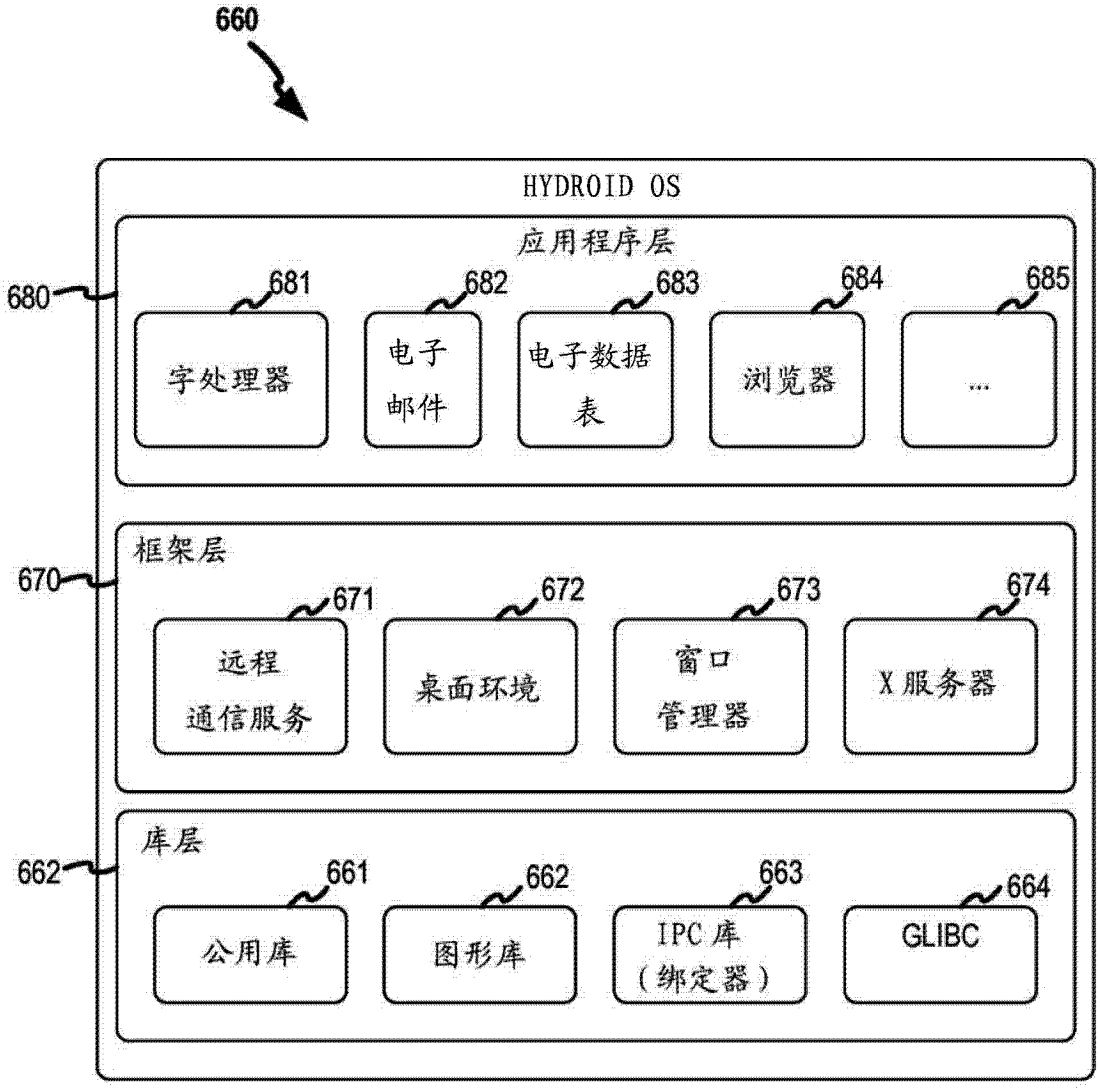

660
HYDROID OS
应用程序层
680
681
682
683
684
685
字处理器
电子邮件
电子数据表
浏览器
...
框架层
670
671
672
673
674
远程通信服务
桌面环境
窗口管理器
X 服务器
库层
662
661
662
663
664
公用库
图形库
IPC 库（绑定器）
GLIBC

图 6

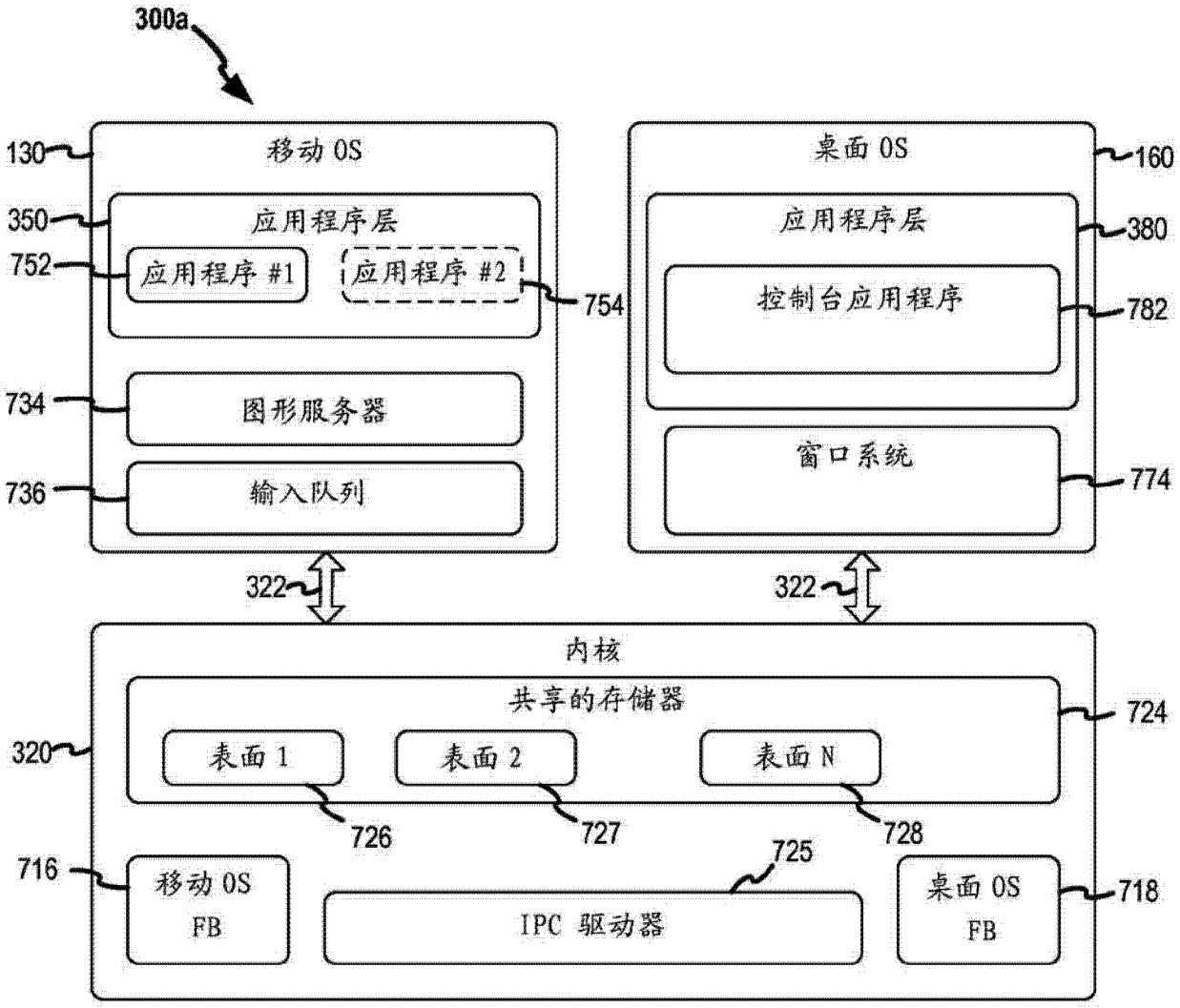

300a
130
移动 OS
350
应用程序层
752
应用程序 #1
应用程序 #2
754
734
图形服务器
736
输入队列
160
桌面 OS
380
应用程序层
控制台应用程序
782
窗口系统
774
322
322
内核
724
共享的存储器
320
表面 1
表面 2
表面 N
726
727
728
725
716
移动 OS FB
IPC 驱动器
桌面 OS FB
718

图 7

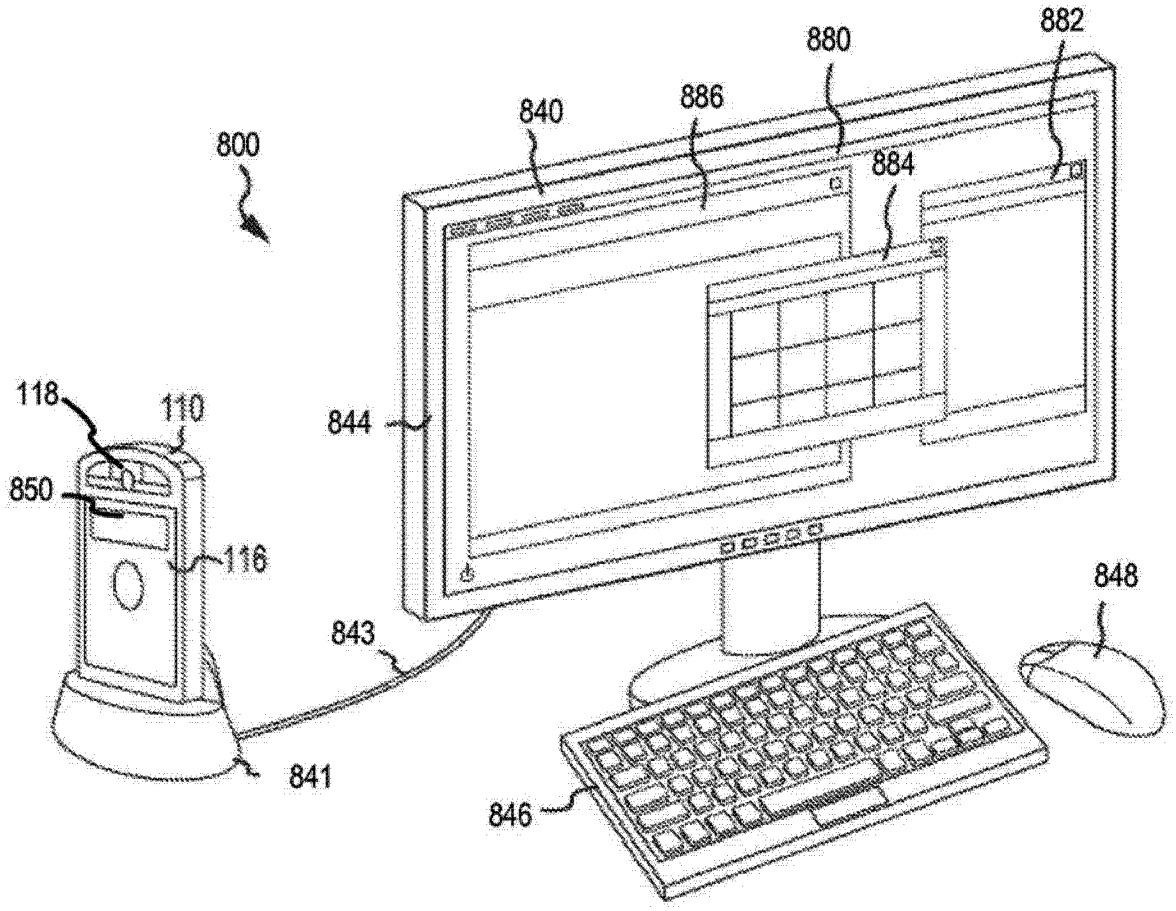


图 8

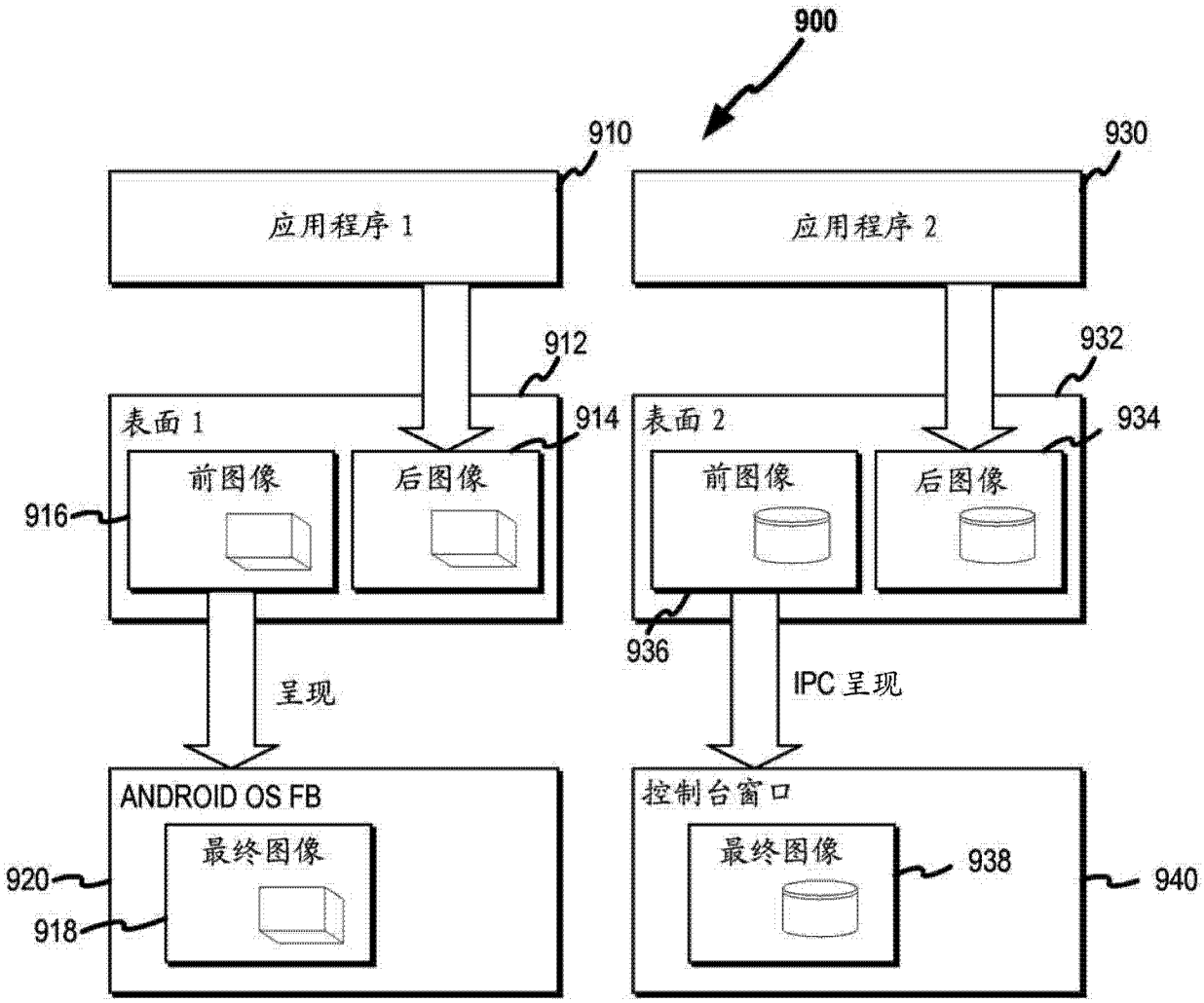

900
910
930
应用程序 1
应用程序 2
912
932
表面 1
914
表面 2
934
前图像
后图像
前图像
后图像
916
936
呈现
IPC 呈现
ANDROID OS FB
控制台窗口
最终图像
最终图像
920
938
940
918

图 9

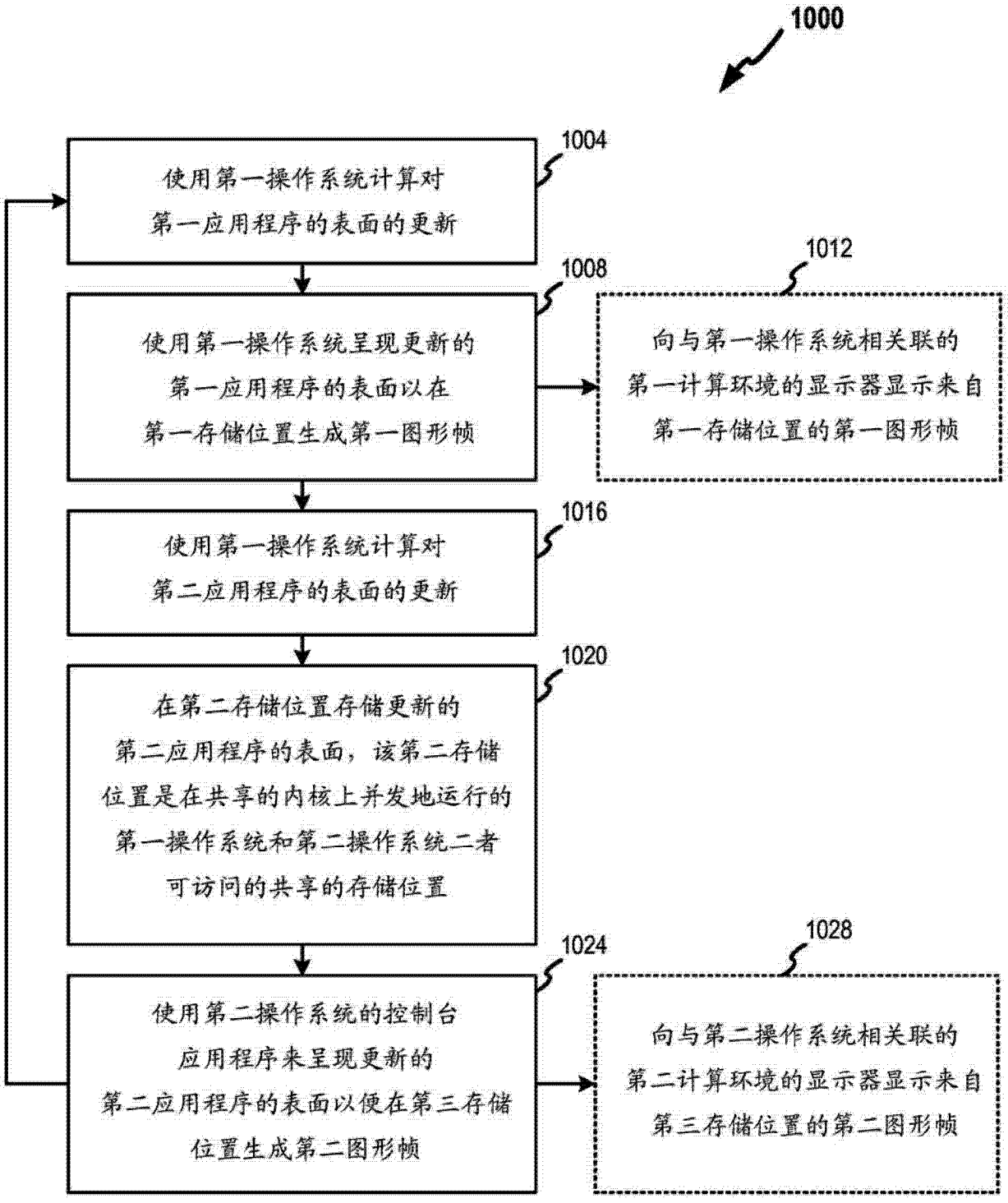

1000
1004
使用第一操作系统计算对
第一应用程序的表面的更新
1008
使用第一操作系统呈现更新的
第一应用程序的表面以在
第一存储位置生成第一图形帧
1012
向与第一操作系统相关联的
第一计算环境的显示器显示来自
第一存储位置的第一图形帧
1016
使用第一操作系统计算对
第二应用程序的表面的更新
1020
在第二存储位置存储更新的
第二应用程序的表面，该第二存储
位置是在共享的内核上并发地运行的
第一操作系统和第二操作系统二者
可访问的共享的存储位置
1024
使用第二操作系统的控制台
应用程序来呈现更新的
第二应用程序的表面以便在第三存储
位置生成第二图形帧
1028
向与第二操作系统相关联的
第二计算环境的显示器显示来自
第三存储位置的第二图形帧

图 10

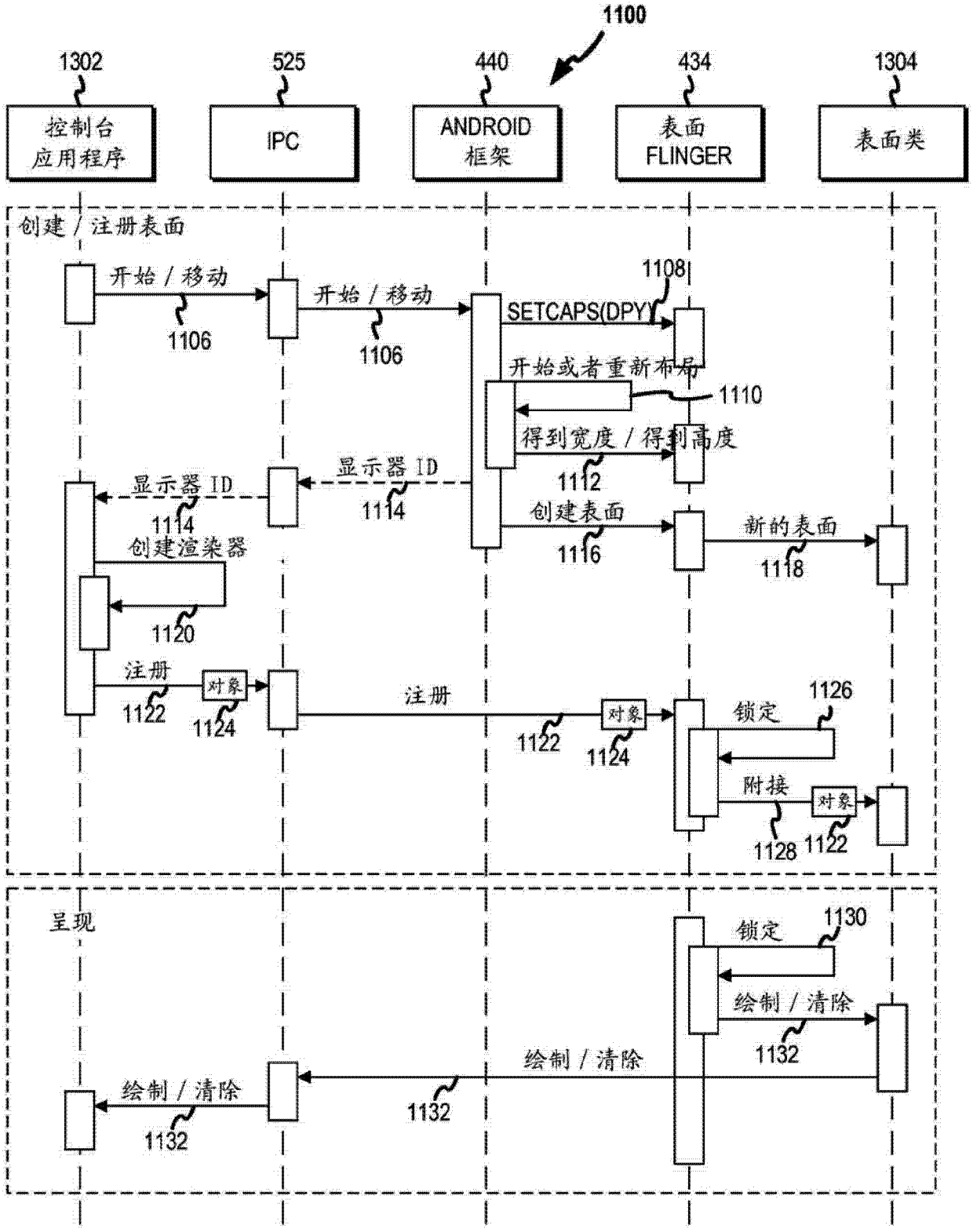

1100
1302
控制台
应用程序
525
IPC
440
ANDROID
框架
434
表面
FLINGER
1304
表面类
创建 / 注册表面
开始 / 移动
1106
开始 / 移动
1106
1108
SETCAPS(DPY)
开始或者重新布局
1110
得到宽度 / 得到高度
1112
显示器 ID
1114
显示器 ID
1114
创建表面
1116
新的表面
1118
创建渲染器
1120
注册
对象
1122
1124
注册
对象
1122
1124
1126
锁定
附接
对象
1128
1122
呈现
1130
锁定
绘制 / 清除
1132
绘制 / 清除
1132
绘制 / 清除
1132

图 11

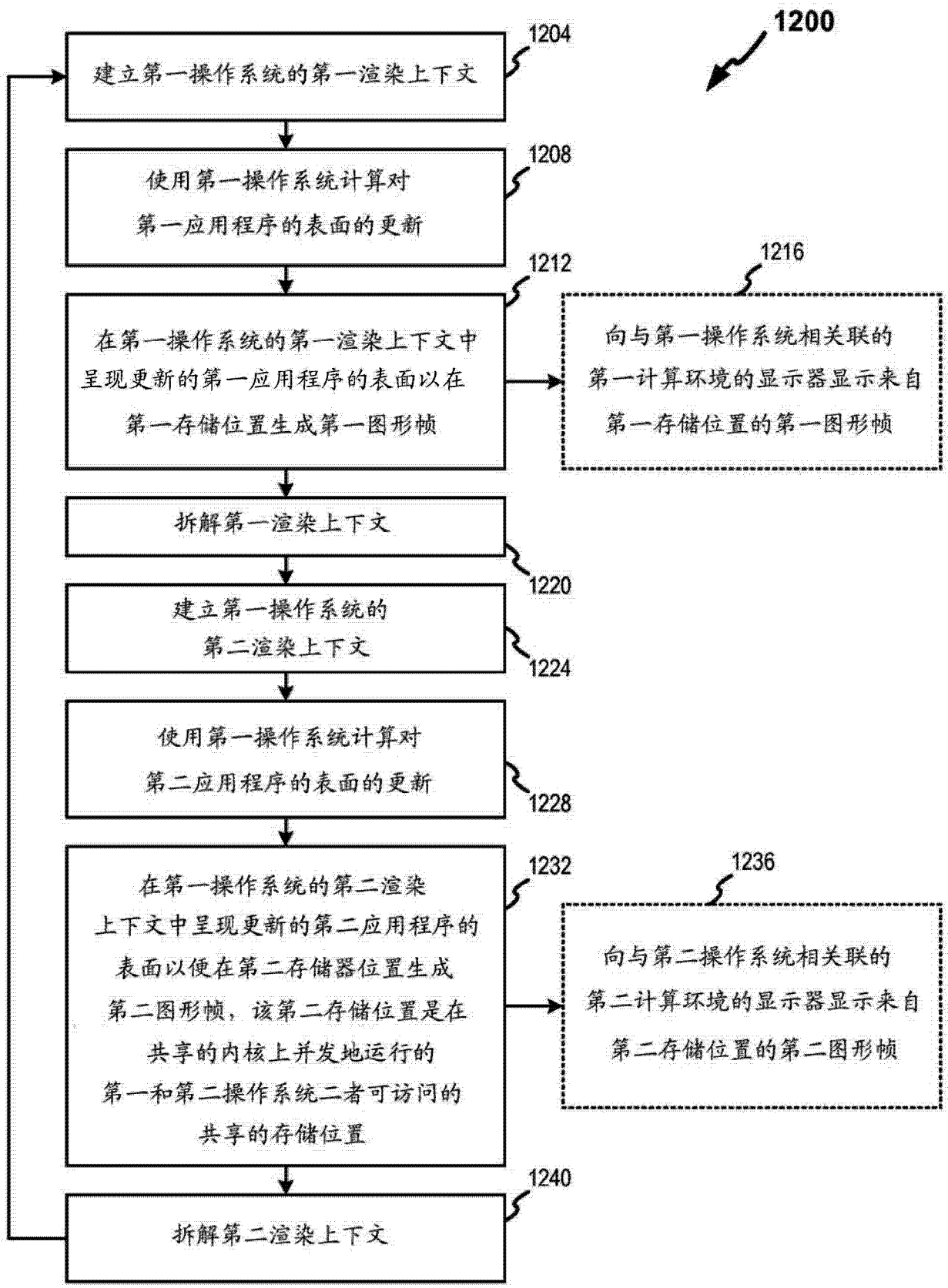

1200
建立第一操作系统的第一渲染上下文
1204
使用第一操作系统计算对
第一应用程序的表面的更新
1208
在第一操作系统的第一渲染上下文中
呈现更新的第一应用程序的表面以在
第一存储位置生成第一图形帧
1212
向与第一操作系统相关联的
第一计算环境的显示器显示来自
第一存储位置的第一图形帧
1216
拆解第一渲染上下文
1220
建立第一操作系统的
第二渲染上下文
1224
使用第一操作系统计算对
第二应用程序的表面的更新
1228
在第一操作系统的第二渲染
上下文中呈现更新的第二应用程序的
表面以便在第二存储器位置生成
第二图形帧，该第二存储位置是在
共享的内核上并发地运行的
第一和第二操作系统二者可访问的
共享的存储位置
1232
向与第二操作系统相关联的
第二计算环境的显示器显示来自
第二存储位置的第二图形帧
1236
拆解第二渲染上下文
1240

图 12

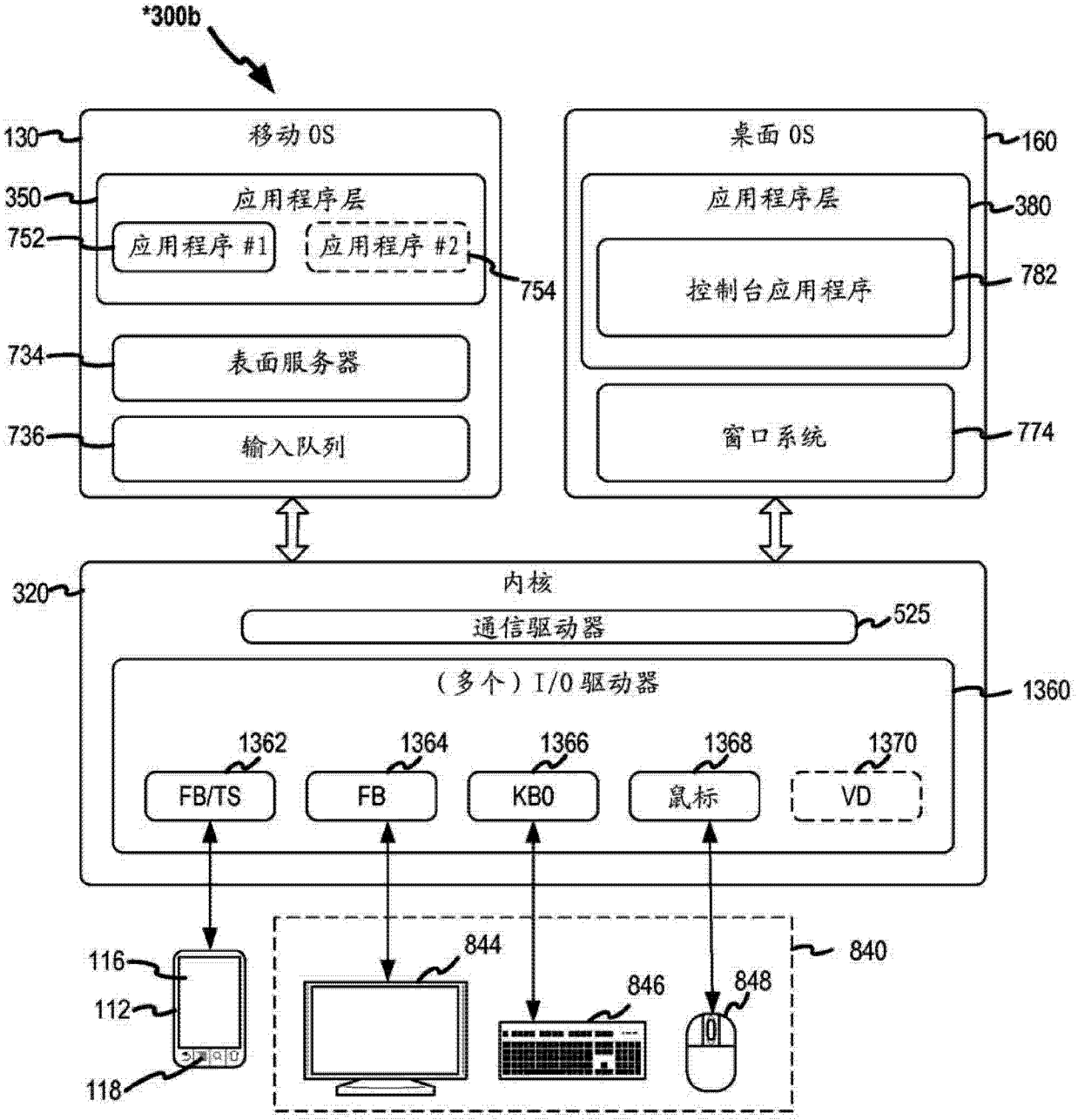

*300b
130
移动 OS
350
应用程序层
752
应用程序 #1
应用程序 #2
754
734
表面服务器
736
输入队列
桌面 OS
160
应用程序层
380
控制台应用程序
782
窗口系统
774
320
内核
通信驱动器
525
(多个) I/O 驱动器
1360
1362
1364
1366
1368
1370
FB/TS
FB
KB0
鼠标
VD
116
112
118
844
846
848
840

图 13

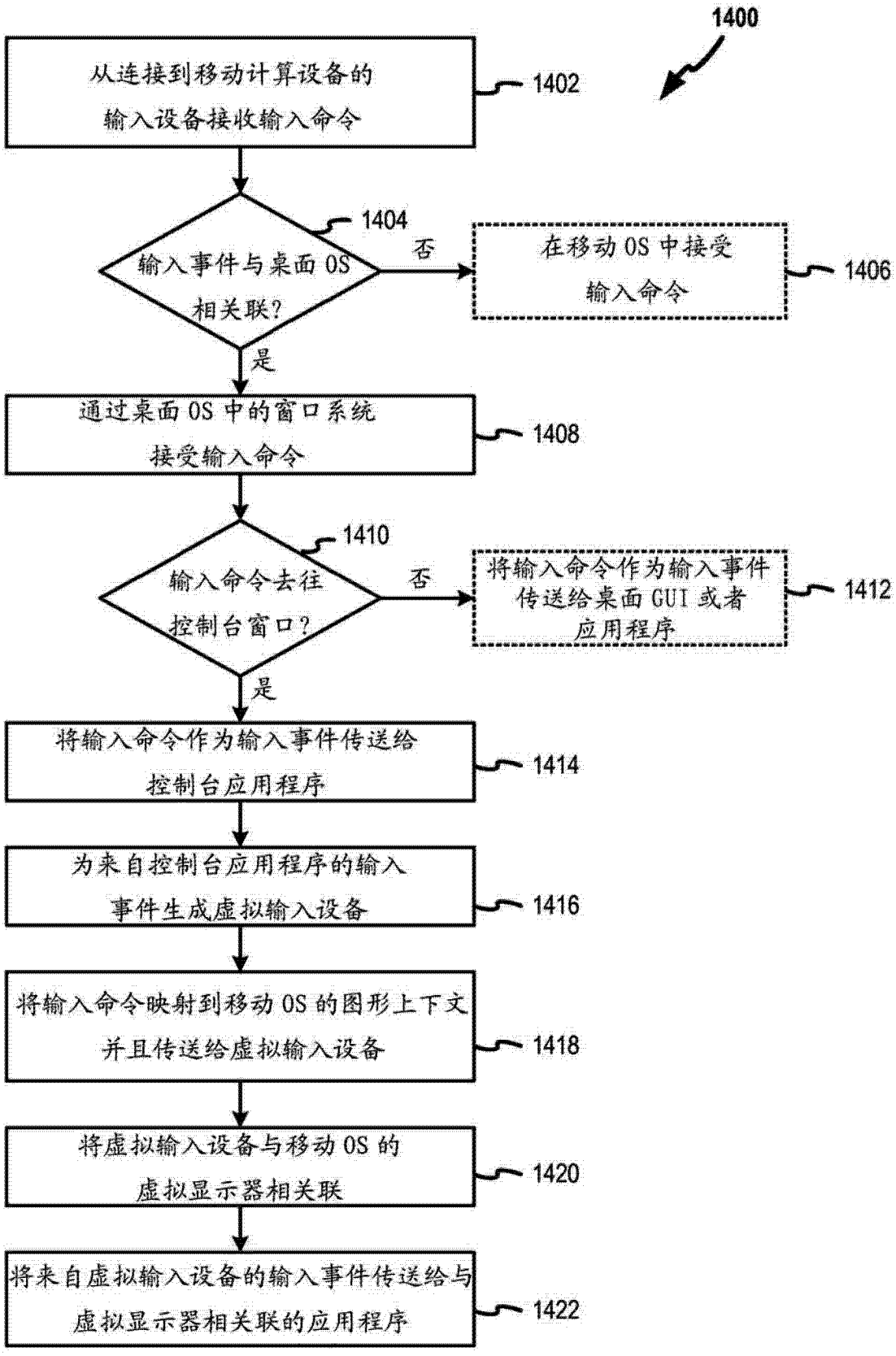

1400
从连接到移动计算设备的输入设备接收输入命令
1402
输入事件与桌面 OS 相关联?
1404
否
在移动 OS 中接受输入命令
1406
是
通过桌面 OS 中的窗口系统接受输入命令
1408
输入命令去往控制台窗口?
1410
否
将输入命令作为输入事件传送给桌面 GUI 或者应用程序
1412
是
将输入命令作为输入事件传送给控制台应用程序
1414
为来自控制台应用程序的输入事件生成虚拟输入设备
1416
将输入命令映射到移动 OS 的图形上下文并且传送给虚拟输入设备
1418
将虚拟输入设备与移动 OS 的虚拟显示器相关联
1420
将来自虚拟输入设备的输入事件传送给与虚拟显示器相关联的应用程序
1422

图 14

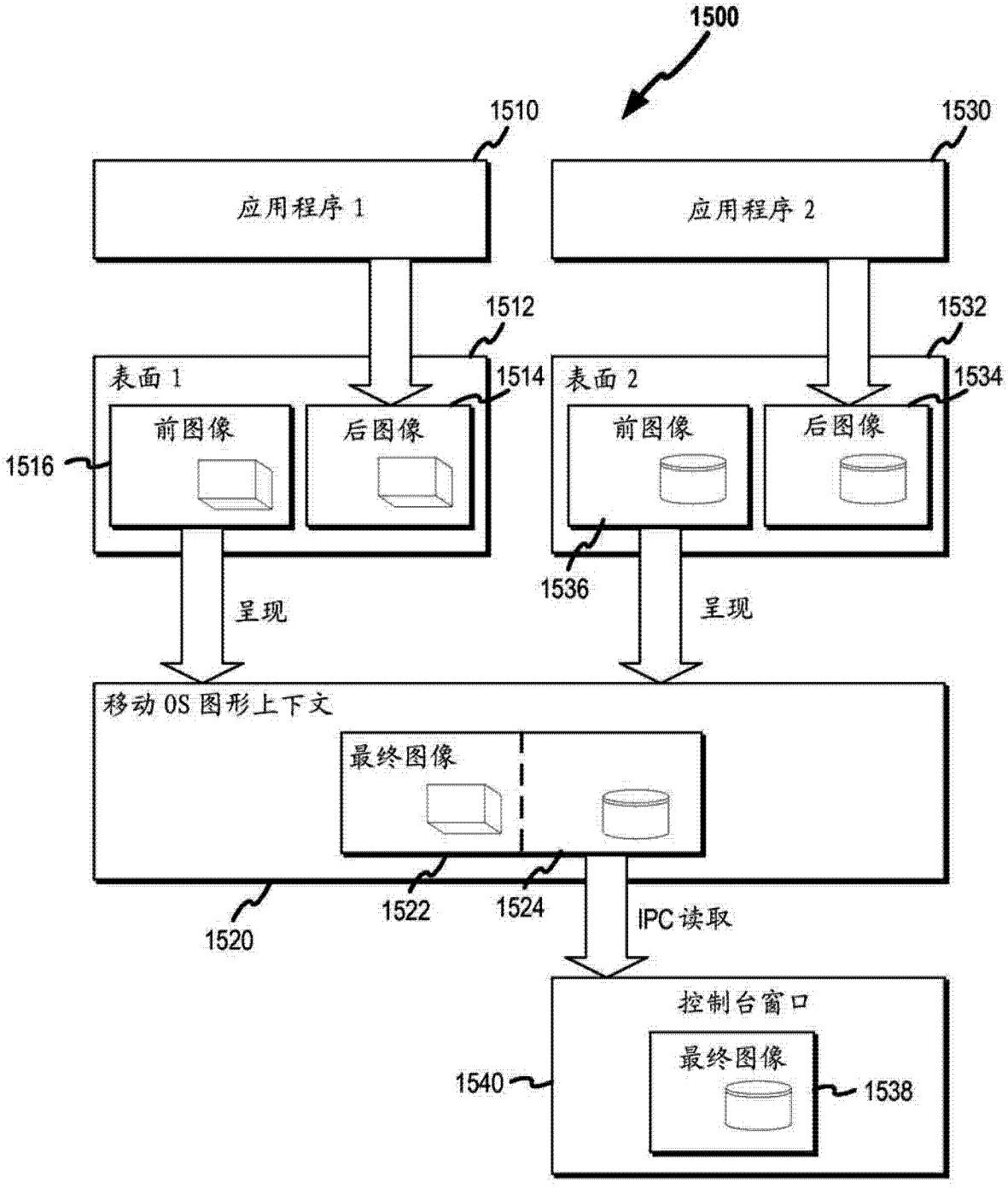

1500
1510
1530
应用程序 1
应用程序 2
1512
1532
表面 1
表面 2
1514
1534
前图像
后图像
前图像
后图像
1516
1536
呈现
呈现
移动 OS 图形上下文
最终图像
1520
1522
1524
IPC 读取
控制台窗口
最终图像
1540
1538

图 15

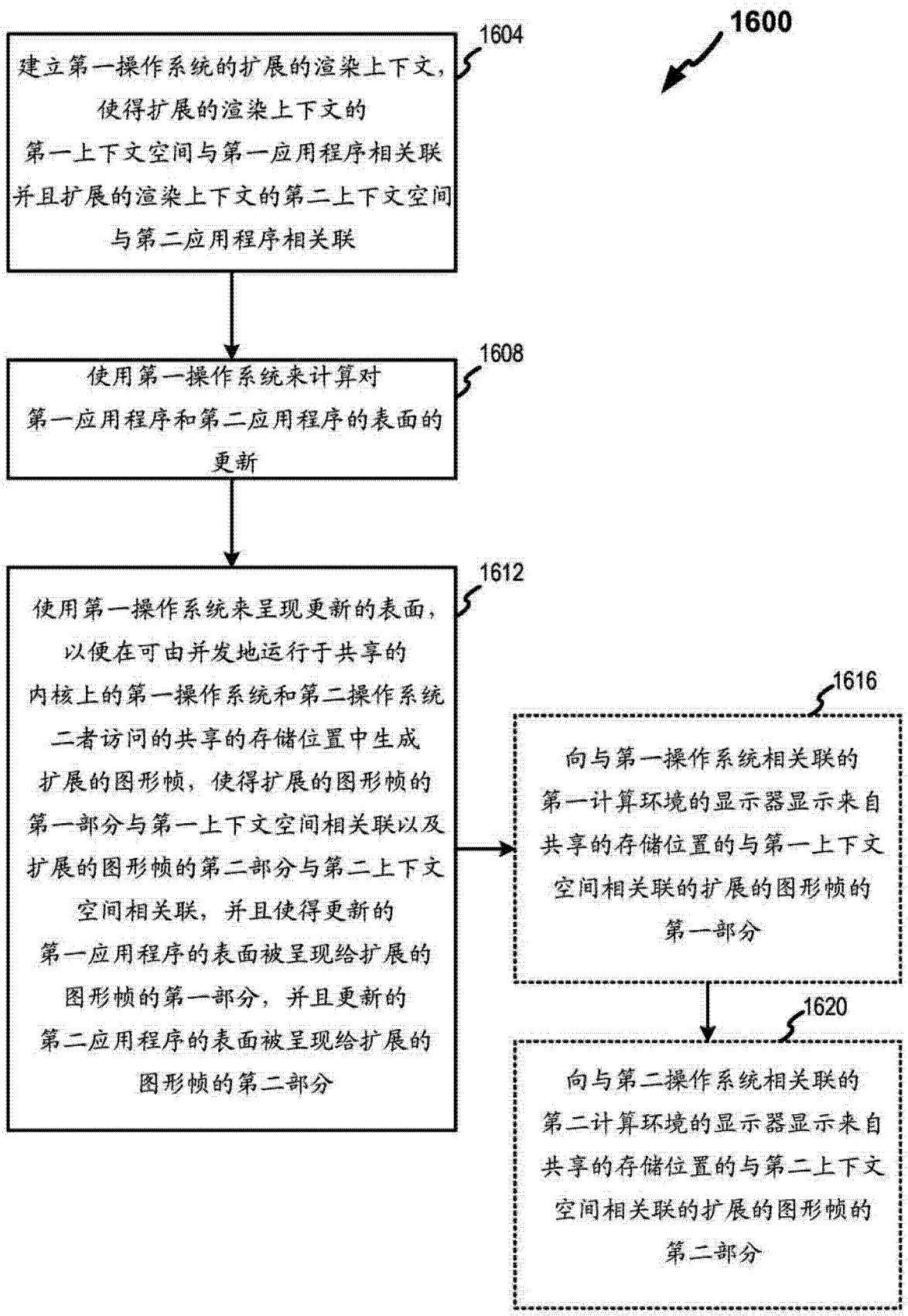

1600
1604
建立第一操作系统的扩展的渲染上下文，使得扩展的渲染上下文的第一上下文空间与第一应用程序相关联并且扩展的渲染上下文的第二上下文空间与第二应用程序相关联
1608
使用第一操作系统来计算对第一应用程序和第二应用程序的表面的更新
1612
使用第一操作系统来呈现更新的表面，以便在可由并发地运行于共享的内核上的第一操作系统和第二操作系统二者访问的共享的存储位置中生成扩展的图形帧，使得扩展的图形帧的第一部分与第一上下文空间相关联以及扩展的图形帧的第二部分与第二上下文空间相关联，并且使得更新的第一应用程序的表面被呈现给扩展的图形帧的第一部分，并且更新的第二应用程序的表面被呈现给扩展的图形帧的第二部分
1616
向与第一操作系统相关联的第一计算环境的显示器显示来自共享的存储位置的与第一上下文空间相关联的扩展的图形帧的第一部分
1620
向与第二操作系统相关联的第二计算环境的显示器显示来自共享的存储位置的与第二上下文空间相关联的扩展的图形帧的第二部分

图 16

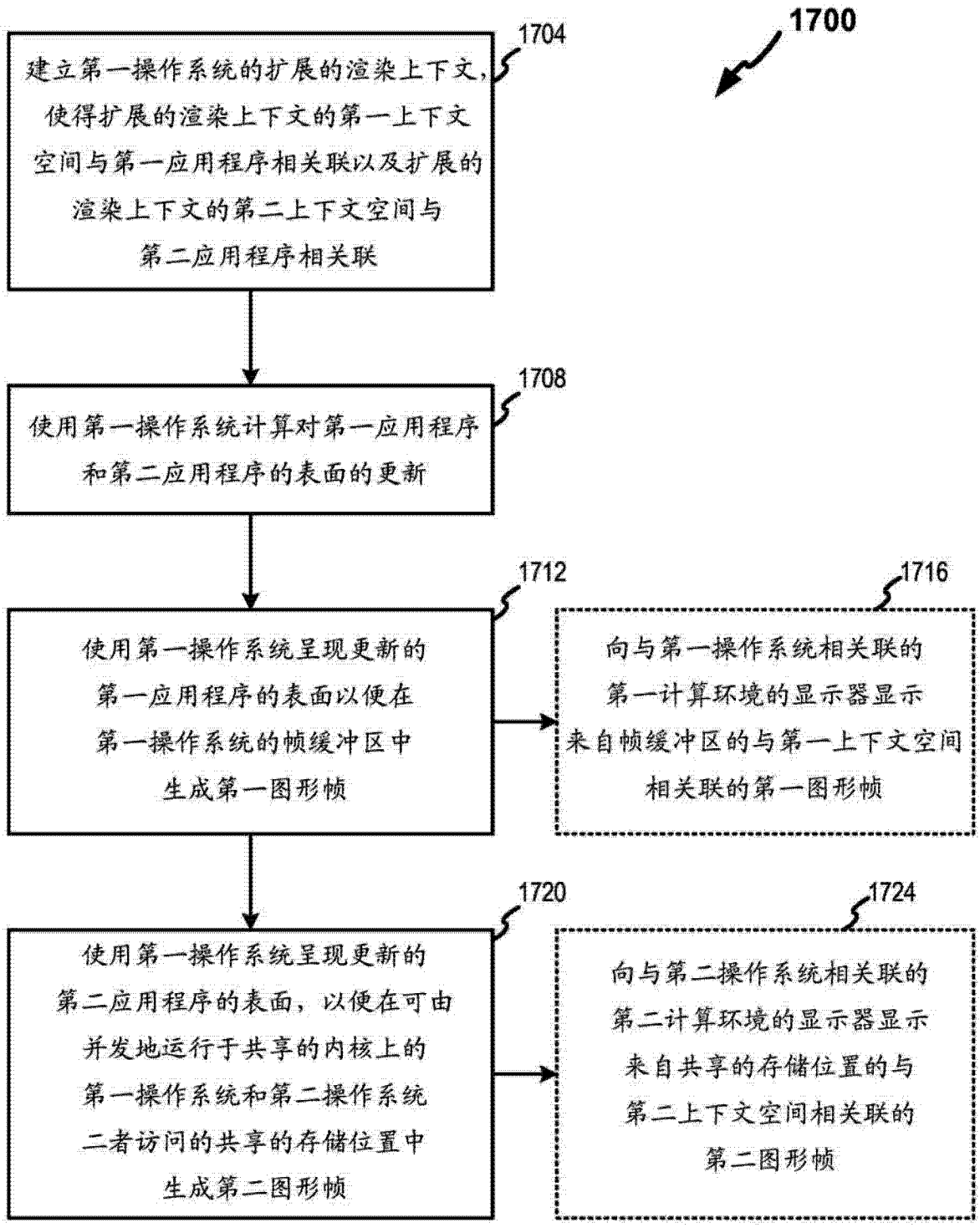

1700
1704
建立第一操作系统的扩展的渲染上下文，使得扩展的渲染上下文的第一上下文空间与第一应用程序相关联以及扩展的渲染上下文的第二上下文空间与第二应用程序相关联
1708
使用第一操作系统计算对第一应用程序和第二应用程序的表面的更新
1712
使用第一操作系统呈现更新的第一应用程序的表面以便在第一操作系统的帧缓冲区中生成第一图形帧
1716
向与第一操作系统相关联的第一计算环境的显示器显示来自帧缓冲区的与第一上下文空间相关联的第一图形帧
1720
使用第一操作系统呈现更新的第二应用程序的表面，以便在可由并发地运行于共享的内核上的第一操作系统和第二操作系统二者访问的共享的存储位置中生成第二图形帧
1724
向与第二操作系统相关联的第二计算环境的显示器显示来自共享的存储位置的与第二上下文空间相关联的第二图形帧

图 17

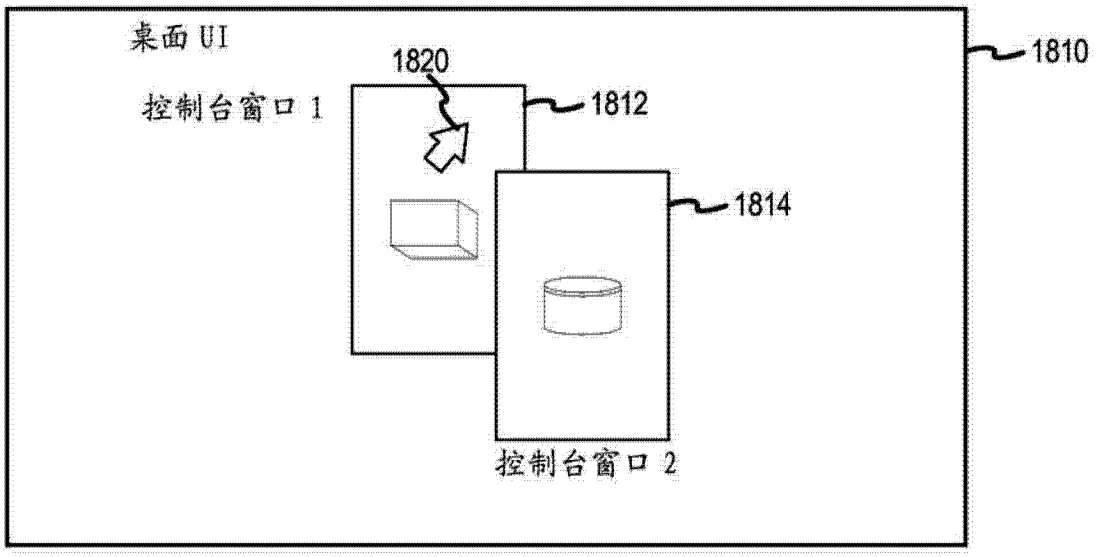

桌面 UI
1820
1810
控制台窗口 1
1812
1814
控制台窗口 2

图 18a

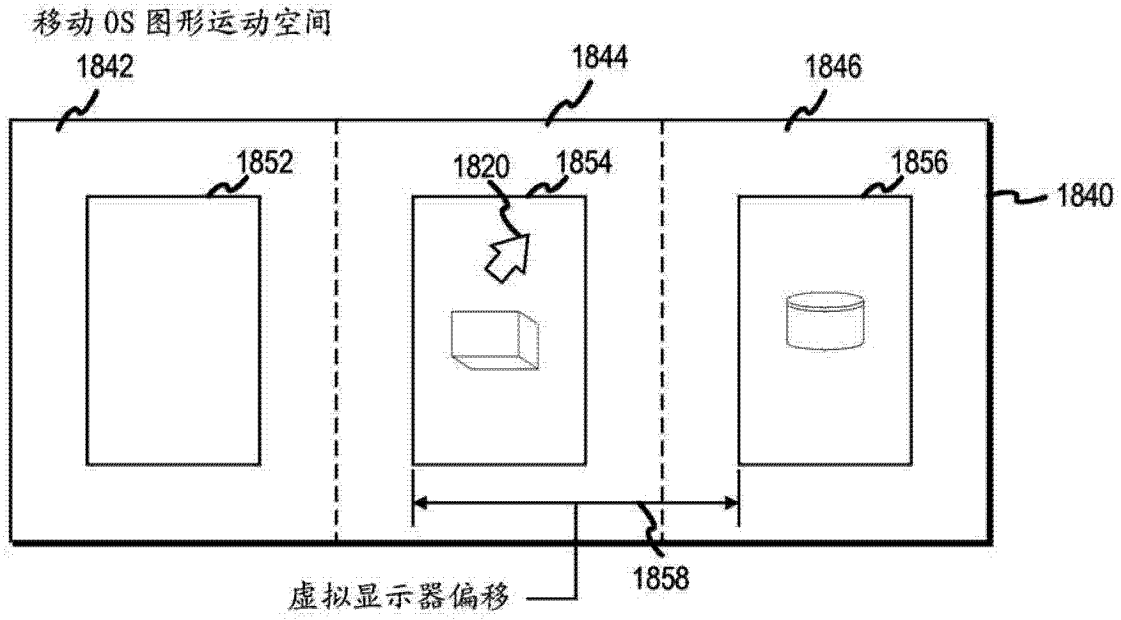

移动 OS 图形运动空间
1842
1844
1846
1852
1820
1854
1856
1840
1858
虚拟显示器偏移

图 18b

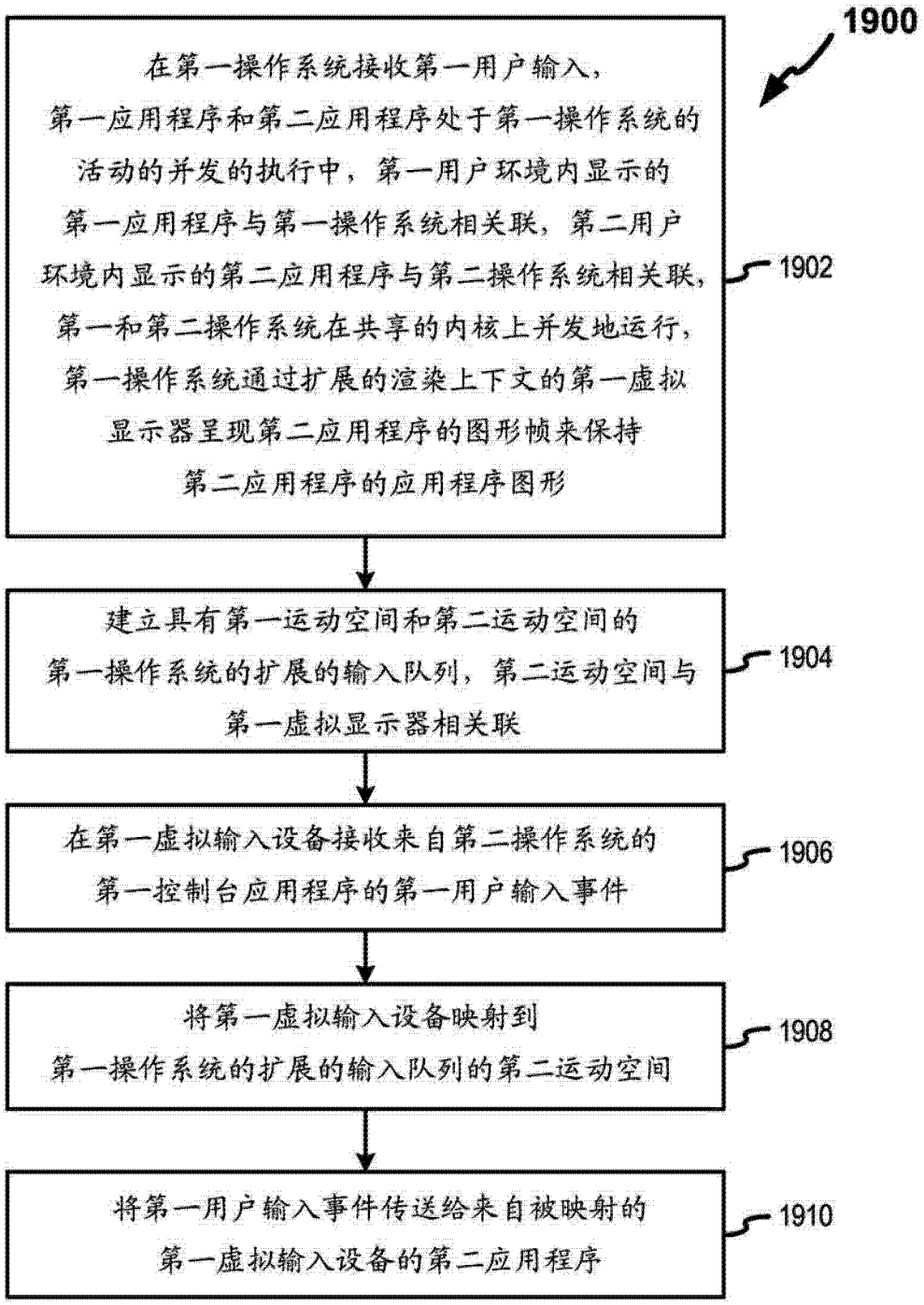

1900
在第一操作系统接收第一用户输入，
第一应用程序和第二应用程序处于第一操作系统的
活动的并发的执行中，第一用户环境内显示的
第一应用程序与第一操作系统相关联，第二用户
环境内显示的第二应用程序与第二操作系统相关联，
第一和第二操作系统在共享的内核上并发地运行，
第一操作系统通过扩展的渲染上下文的第一虚拟
显示器呈现第二应用程序的图形帧来保持
第二应用程序的应用程序图形
1902
建立具有第一运动空间和第二运动空间的
第一操作系统的扩展的输入队列，第二运动空间与
第一虚拟显示器相关联
1904
在第一虚拟输入设备接收来自第二操作系统的
第一控制台应用程序的第一用户输入事件
1906
将第一虚拟输入设备映射到
第一操作系统的扩展的输入队列的第二运动空间
1908
将第一用户输入事件传送给来自被映射的
第一虚拟输入设备的第二应用程序
1910

图 19

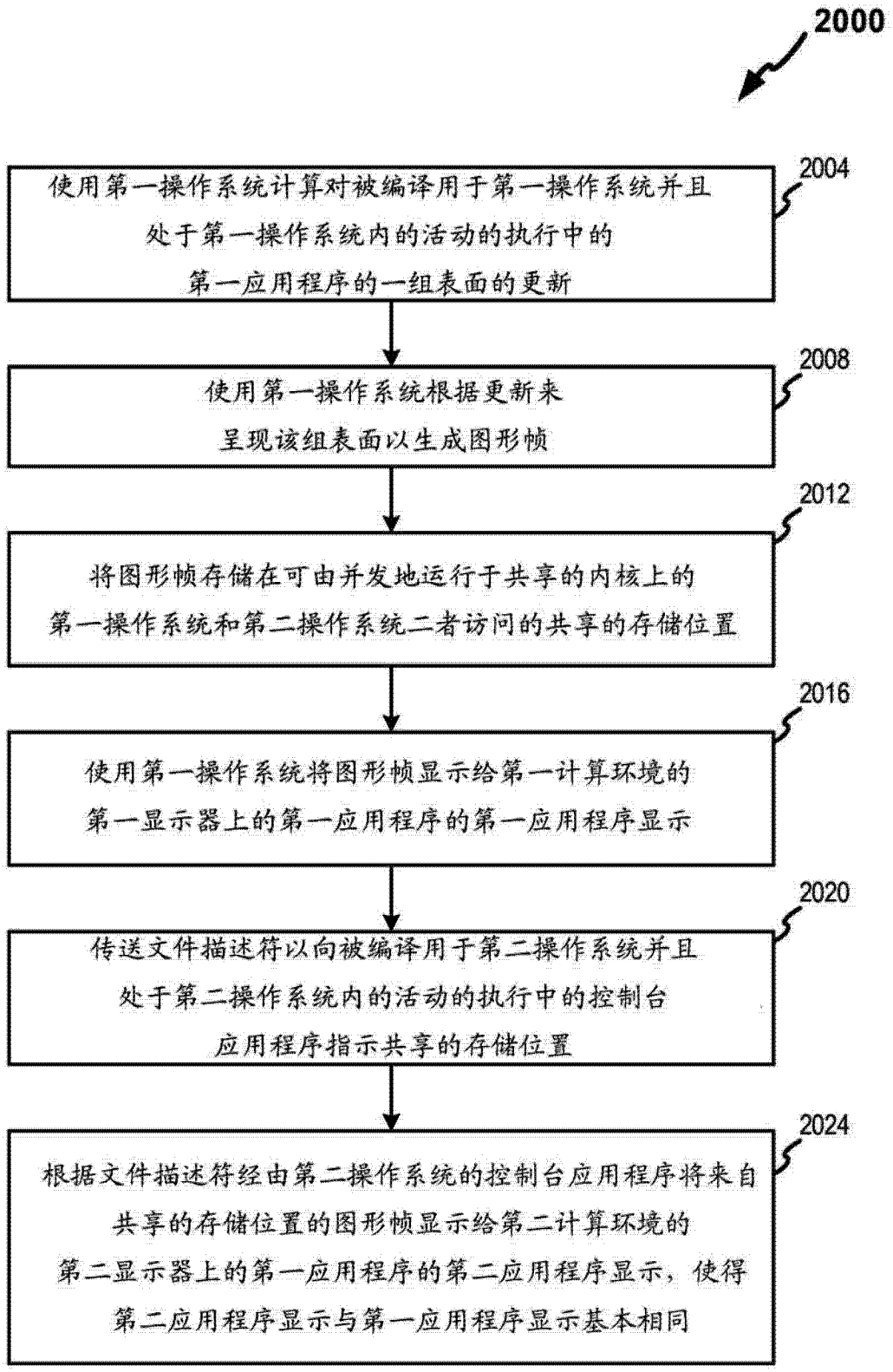

2000
使用第一操作系统计算对被编译用于第一操作系统并且处于第一操作系统内的活动的执行中的第一应用程序的一组表面的更新
2004
使用第一操作系统根据更新来呈现该组表面以生成图形帧
2008
将图形帧存储在可由并发地运行于共享的内核上的第一操作系统和第二操作系统二者访问的共享的存储位置
2012
使用第一操作系统将图形帧显示给第一计算环境的第一显示器上的第一应用程序的第一应用程序显示
2016
传送文件描述符以向被编译用于第二操作系统并且处于第二操作系统内的活动的执行中的控制台应用程序指示共享的存储位置
2020
根据文件描述符经由第二操作系统的控制台应用程序将来自共享的存储位置的图形帧显示给第二计算环境的第二显示器上的第一应用程序的第二应用程序显示，使得第二应用程序显示与第一应用程序显示基本相同
2024

图 20

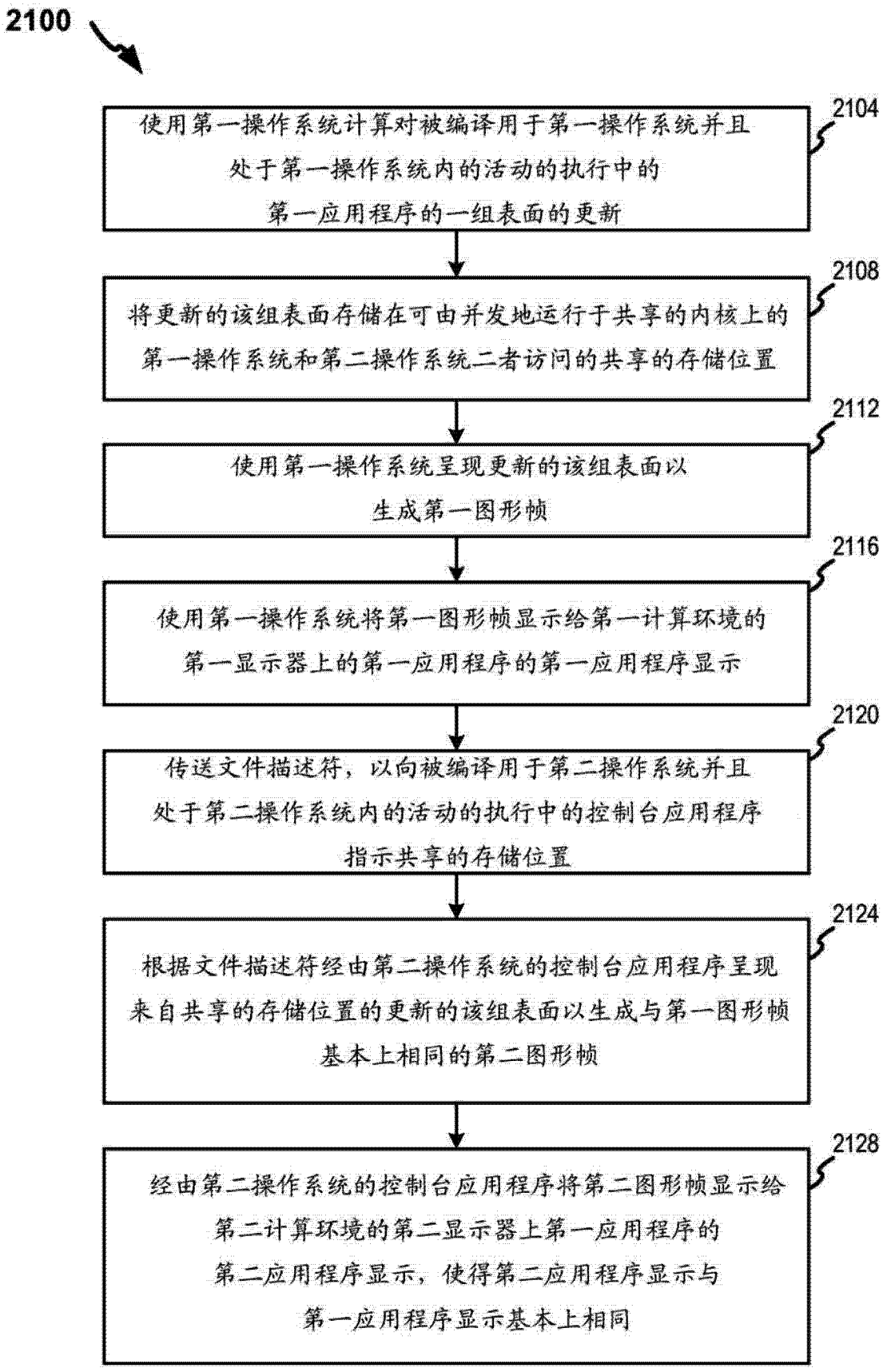

2100
使用第一操作系统计算对被编译用于第一操作系统并且处于第一操作系统内的活动的执行中的第一应用程序的一组表面的更新
2104
将更新的该组表面存储在可由并发地运行于共享的内核上的第一操作系统和第二操作系统二者访问的共享的存储位置
2108
使用第一操作系统呈现更新的该组表面以生成第一图形帧
2112
使用第一操作系统将第一图形帧显示给第一计算环境的第一显示器上的第一应用程序的第一应用程序显示
2116
传送文件描述符，以向被编译用于第二操作系统并且处于第二操作系统内的活动的执行中的控制台应用程序指示共享的存储位置
2120
根据文件描述符经由第二操作系统的控制台应用程序呈现来自共享的存储位置的更新的该组表面以生成与第一图形帧基本上相同的第二图形帧
2124
经由第二操作系统的控制台应用程序将第二图形帧显示给第二计算环境的第二显示器上第一应用程序的第二应用程序显示，使得第二应用程序显示与第一应用程序显示基本上相同
2128

图 21

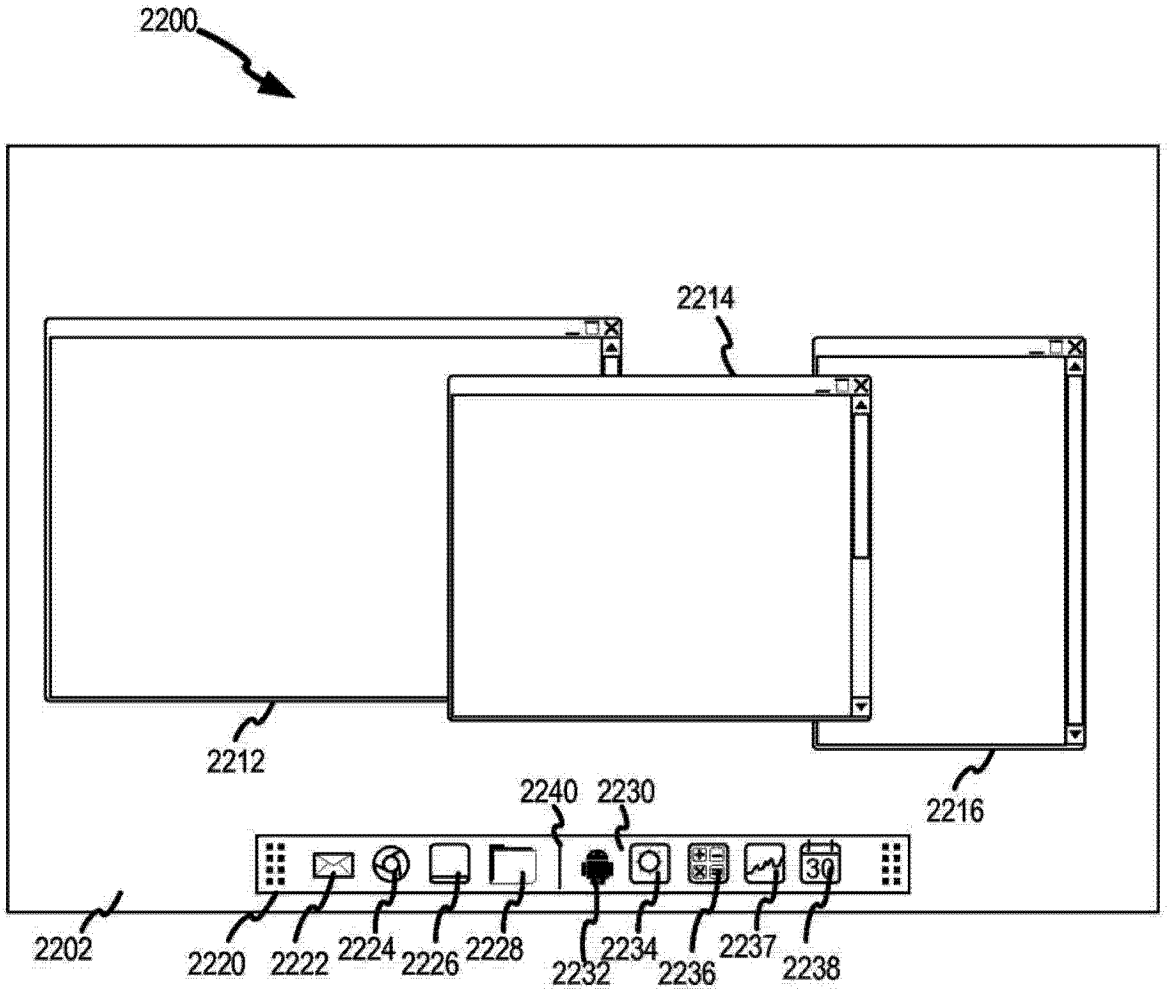

2200
2214
2212
2216
2240
2230
2202
2220
2222
2224
2226
2228
2232
2234
2236
2237
2238
30

图 22

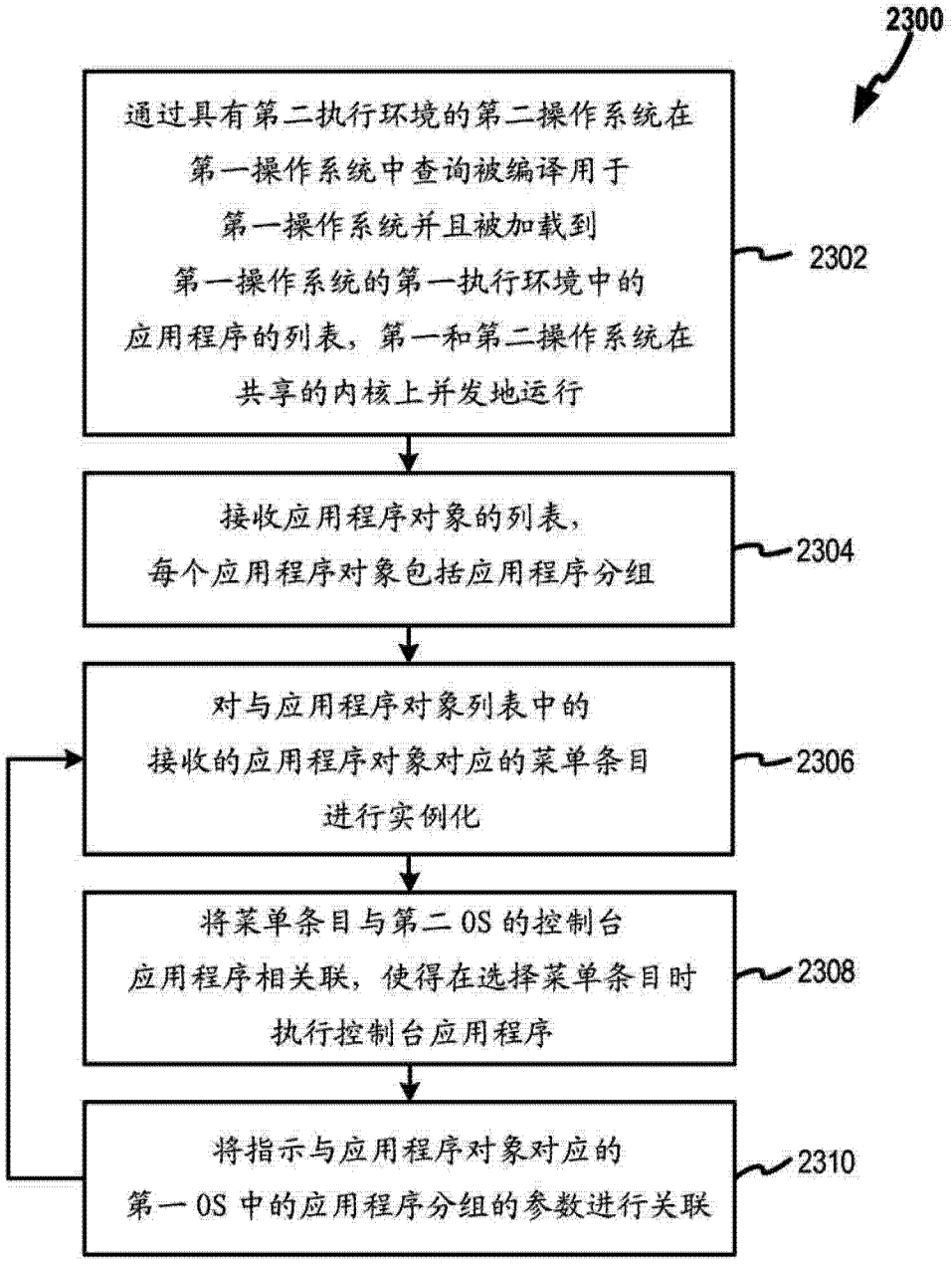

2300
通过具有第二执行环境的第二操作系统在第一操作系统中查询被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境中的应用程序的列表，第一和第二操作系统在共享的内核上并发地运行
2302
接收应用程序对象的列表，每个应用程序对象包括应用程序分组
2304
对与应用程序对象列表中的接收的应用程序对象对应的菜单条目进行实例化
2306
将菜单条目与第二 OS 的控制台应用程序相关联，使得在选择菜单条目时执行控制台应用程序
2308
将指示与应用程序对象对应的第一 OS 中的应用程序分组的参数进行关联
2310

图 23

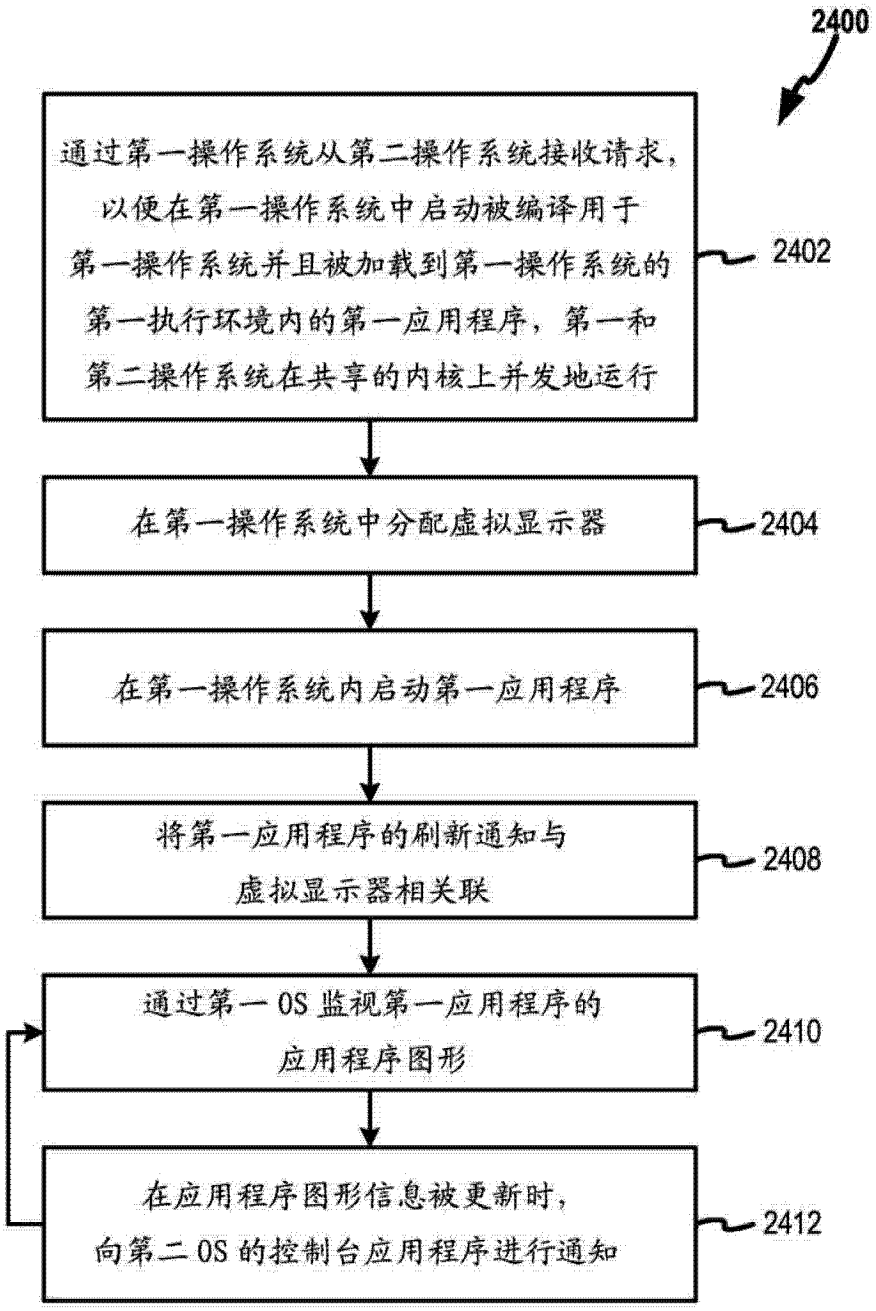

2400
通过第一操作系统从第二操作系统接收请求，以便在第一操作系统中启动被编译用于第一操作系统并且被加载到第一操作系统的第一执行环境内的第一应用程序，第一和第二操作系统在共享的内核上并发地运行
2402
在第一操作系统中分配虚拟显示器
2404
在第一操作系统内启动第一应用程序
2406
将第一应用程序的刷新通知与虚拟显示器相关联
2408
通过第一 OS 监视第一应用程序的应用程序图形
2410
在应用程序图形信息被更新时，向第二 OS 的控制台应用程序进行通知
2412

图 24