(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 104303149 A
(43)申请公布日 2015.01.21

(21)申请号 201380010455.5
(22)申请日 2013.02.20
(30)优先权数据
61/602,287 2012.02.23 US
13/418,597 2012.03.13 US
(85)PCT国际申请进入国家阶段日
2014.08.21
(86)PCT国际申请的申请数据
PCT/US2013/026873 2013.02.20
(87)PCT国际申请的公布数据
WO2013/126415 EN 2013.08.29
(71)申请人 高通股份有限公司
地址 美国加利福尼亚州
(72)发明人 N·S·加盖什 G·L·卡拉维
V·维加雅拉佳 T·A·厄尔默
J·H·斯塔布斯 A·J·弗朗茨
W·加布里尔
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 亓云
(51)Int.Cl.
G06F 9/48(2006.01)
G06F 1/32(2006.01)

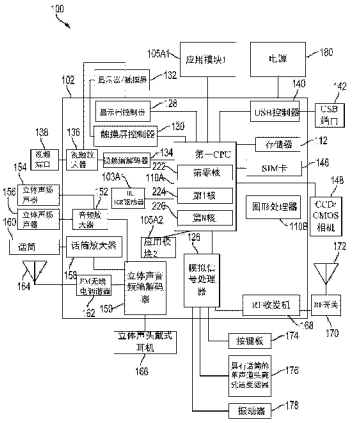
权利要求书4页 说明书34页 附图30页

(54)发明名称

用于调度便携式计算设备中的请求的方法和系统

(57)摘要

一种用于在便携式计算设备内的资源间管理请求的方法和系统，包括调度器从客户端接收数据以用于调度多个请求。每个请求标识至少一个资源和所请求的截止期限。接着，来自该客户端的数据被调度器存储在数据库中。调度器随后基于这些请求中的所请求截止期限以及基于该便携式计算设备内的资源的当前状态来确定用于处理这些请求的时间和顺序。调度器随后在所确定的时间并根据所确定的顺序将这些请求传达给各资源。调度器可以自行斟酌以响应于从客户端接收到新请求命令而将一请求调度在其所请求的截止期限之后。调度器可以允许与休眠处理器状态相对应的休眠集将处理器断电。



CN 104303149 A

1. 一种用于在便携式计算设备内的资源间管理请求的方法，所述方法包括：

从客户端接收数据以用于调度多个请求，每个请求标识至少一个资源和所请求的截止期限；

将来自所述客户端的所述数据存储到数据库中；

基于所述请求中的所请求截止期限以及基于所述便携式计算设备内的资源的当前状态来确定用于处理所述请求的时间和顺序；以及

在所确定的时间并根据所确定的顺序将所述请求传达给所述资源。

2. 如权利要求 1 所述的方法，其特征在于，进一步包括：

响应于接收到新请求命令将一请求调度在其所请求的截止期限之后。

3. 如权利要求 1 所述的方法，其特征在于，进一步包括：

允许与休眠处理器状态相对应的休眠集将处理器断电，从而在所述处理器退出所述休眠处理器状态时所述处理器能继续对一个或多个请求进行工作。

4. 如权利要求 1 所述的方法，其特征在于，进一步包括：

在经调度休眠状态期间接收非预期请求；

响应于接收到所述非预期请求，确定是否需要一个或多个经调度请求来响应所述非预期请求。

5. 如权利要求 4 所述的方法，其特征在于，进一步包括：

如果在所述经调度休眠状态期间不需要一个或多个经调度请求来响应所述非预期请求，则取消一个或多个经调度请求。

6. 如权利要求 5 所述的方法，其特征在于，进一步包括：

重新调度所取消的经调度请求中的一个或多个请求以在下一经调度活跃状态中发生。

7. 如权利要求 1 所述的方法，其特征在于，进一步包括：

接收撤销调度先前经调度请求的命令。

8. 如权利要求 7 所述的方法，其特征在于，进一步包括：

响应于接收到所述撤销调度先前经调度请求的命令而锁定资源。

9. 如权利要求 8 所述的方法，其特征在于，进一步包括：

响应于接收到所述撤销调度先前经调度请求的命令而用资源完成对经调度请求的处理。

10. 如权利要求 1 所述的方法，其特征在于，所述便携式计算设备包括移动电话、个人数字助理、寻呼机、智能电话、导航设备、和具有无线连接或链路的手持计算机中的至少一者。

11. 一种用于在便携式计算设备内的资源间管理请求的计算机系统，所述系统包括：

处理器，能操作用于：

从客户端接收数据以用于调度多个请求，每个请求标识至少一个资源和所请求的截止期限；

将来自所述客户端的所述数据存储到数据库中；

基于所述请求中的所请求截止期限以及基于所述便携式计算设备内的资源的当前状态来确定用于处理所述请求的时间和顺序；以及

在所确定的时间并根据所确定的顺序将所述请求传达给所述资源。

12. 如权利要求 11 所述的系统，其特征在于，所述处理器进一步能操作用于：

响应于接收到新请求命令将一请求调度在其所请求的截止期限之后。

13. 如权利要求 12 所述的系统，其特征在于，所述处理器进一步能操作用于：

允许与休眠处理器状态相对应的休眠集将处理器断电，从而在所述处理器退出所述休眠处理器状态时所述处理器能继续对一个或多个请求进行工作。

14. 如权利要求 11 所述的系统，其特征在于，所述处理器进一步能操作用于：

在经调度休眠状态期间接收非预期请求；

响应于接收到所述非预期请求，确定是否需要一个或多个经调度请求来响应所述非预期请求。

15. 如权利要求 14 所述的系统，其特征在于，所述处理器进一步能操作用于：

如果在所述经调度休眠状态期间不需要一个或多个经调度请求来响应所述非预期请求，则取消一个或多个经调度请求。

16. 如权利要求 15 所述的系统，其特征在于，所述处理器进一步能操作用于：

重新调度所取消的经调度请求中的一个或多个以在下一经调度活跃状态中发生。

17. 如权利要求 11 所述的系统，其特征在于，所述处理器进一步能操作用于：

接收撤销调度先前经调度请求的命令。

18. 如权利要求 17 所述的系统，其特征在于，所述处理器进一步能操作用于：

响应于接收到所述撤销调度先前经调度请求的命令而锁定资源。

19. 如权利要求 18 所述的系统，其特征在于，所述处理器进一步能操作用于：

响应于接收到所述撤销调度先前经调度请求的命令而用资源完成对经调度请求的处理。

20. 如权利要求 11 所述的系统，其特征在于，所述便携式计算设备包括移动电话、个人数字助理、寻呼机、智能电话、导航设备、和具有无线连接或链路的手持计算机中的至少一者。

21. 一种用于在便携式计算设备内的资源间管理请求的计算机系统，所述系统包括：

用于从客户端接收数据以用于调度多个请求的装置，每个请求标识至少一个资源和所请求的截止期限；

用于将来自所述客户端的所述数据存储到数据库中的装置；

用于基于所述请求中的所请求截止期限以及基于所述便携式计算设备内的资源的当前状态来确定用于处理所述请求的时间和顺序的装置；以及

用于在所确定的时间并根据所确定的顺序将所述请求传达给所述资源的装置。

22. 如权利要求 21 所述的系统，其特征在于，进一步包括：

用于响应于接收到新请求命令将一请求调度在其所请求的截止期限之后的装置。

23. 如权利要求 22 所述的系统，其特征在于，进一步包括：

用于允许与休眠处理器状态相对应的休眠集将处理器断电，从而在所述处理器退出所述休眠处理器状态时所述处理器能继续对一个或多个请求进行工作的装置。

24. 如权利要求 21 所述的系统，其特征在于，进一步包括：

用于在经调度休眠状态期间接收非预期请求的装置；

用于响应于接收到所述非预期请求，确定是否需要一个或多个经调度请求来响应所述

非预期请求的装置。

25. 如权利要求 24 所述的系统，其特征在于，进一步包括：

用于如果在所述经调度休眠状态期间不需要一个或多个经调度请求来响应所述非预期请求，则取消一个或多个经调度请求的装置，随后

26. 如权利要求 25 所述的方法，其特征在于，进一步包括：

用于重新调度所取消的经调度请求中的一个或多个以在下一经调度活跃状态中发生的装置。

27. 如权利要求 21 所述的系统，其特征在于，进一步包括：

用于接收撤销调度先前经调度请求的命令的装置。

28. 如权利要求 27 所述的系统，其特征在于，进一步包括：

用于响应于接收到所述撤销调度先前经调度请求的命令而锁定资源的装置。

29. 如权利要求 28 所述的系统，其特征在于，进一步包括：

响应于接收到所述撤销调度先前经调度请求的命令而用资源完成对经调度请求的处理。

30. 如权利要求 21 所述的系统，其特征在于，所述便携式计算设备包括移动电话、个人数字助理、寻呼机、智能电话、导航设备、和具有无线连接或链路的手持计算机中的至少一者。

31. 一种包括计算机可使用介质的计算机程序产品，所述计算机可使用介质具有实施于其中的计算机可读程序代码，所述计算机可读程序代码适配成被执行以实现一种用于在便携式计算设备内的资源间管理请求的方法，所述方法包括：

从客户端接收数据以用于调度多个请求，每个请求标识至少一个资源和所请求的截止期限；

将来自所述客户端的所述数据存储到数据库中；

基于所述请求中的所请求截止期限以及基于所述便携式计算设备内的资源的当前状态来确定用于处理所述请求的时间和顺序；以及

在所确定的时间并根据所确定的顺序将所述请求传达给所述资源。

32. 如权利要求 31 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

响应于接收到新请求命令将一请求调度在其所请求的截止期限之后。

33. 如权利要求 31 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

允许与休眠处理器状态相对应的休眠集使处理器断电，从而在所述处理器退出所述休眠处理器状态时所述处理器能继续对一个或多个请求进行工作。

34. 如权利要求 31 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

在经调度休眠状态期间接收非预期请求；

响应于接收到所述非预期请求，确定是否需要一个或多个经调度请求来响应所述非预期请求。

35. 如权利要求 34 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法

进一步包括：

如果在所述经调度休眠状态期间不需要一个或多个经调度请求来响应所述非预期请求，则取消一个或多个经调度请求。

36. 如权利要求 35 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

重新调度所取消的经调度请求中的一个或多个以在下一经调度活跃状态中发生。

37. 如权利要求 31 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

接收撤销调度先前经调度请求的命令。

38. 如权利要求 37 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

响应于接收到所述撤销调度先前经调度请求的命令而锁定资源。

39. 如权利要求 38 所述的计算机程序产品，其特征在于，所述程序代码实现所述方法进一步包括：

响应于接收到所述撤销调度先前经调度请求的命令而用资源完成对经调度请求的处理。

40. 如权利要求 31 所述的计算机程序产品，其特征在于，所述便携式计算设备包括移动电话、个人数字助理、寻呼机、智能电话、导航设备、和具有无线连接或链路的手持计算机中的至少一者。

用于调度便携式计算设备中的请求的方法和系统

[0001] 优先权和相关申请声明

[0002] 本申请根据35 U.S.C.§119(e)要求于2012年2月23日提交的题为“METHOD AND SYSTEM FOR SCHEDULING REQUESTS IN A PORTABLE COMPUTING DEVICE(用于调度便携式计算设备中的请求的方法和系统)”的美国临时专利申请S/N.61/602,287的优先权。该申请的全部内容通过引用纳入于此。

[0003] 相关技术描述

[0004] 便携式计算设备(“PCD”)正变得越来越流行。这些设备可包括蜂窝电话、便携式/个人数字助理(“PDA”)、便携式游戏控制台、便携式导航单元、掌上型计算机、以及其他便携式电子设备。这些设备中的每个设备可具有主功能。例如,蜂窝电话一般具有接收和传送电话呼叫的主功能。

[0005] 除这些设备的主功能之外,许多设备包括外围功能。例如,蜂窝电话可包括以上所述的进行蜂窝电话呼叫的主功能,以及相机、摄像机、全球定位系统(GPS)导航、web浏览、发送和接收电子邮件、发送和接收文本消息、以及按即讲能力等外围功能。随着PCD功能性的增加,支持此类功能性所需的计算或处理能力也随之增加。可通过在PCD中增加处理器的数量来增加处理能力。随着计算能力和处理器数量的增加,存在着对有效管理各处理器的更大的需求。

[0006] 诸如以上描述的那些之类的功能可实施在可被称为资源的各种相应的硬件和软件元件中。处理器可以在软件(诸如应用程序)控制下在各种时间请求各种资源。在多处理器PCD中,第一处理器可以控制与受第二处理器控制的资源所不同的资源。在常规技术中,跨各处理器来高效地管理资源以节省由这些资源所消耗的功率可能是非常复杂和困难的。

[0007] 概述

[0008] 公开了用于在便携式计算设备内的资源间管理请求的方法和系统。该方法和系统包括调度器从客户端接收数据以用于调度多个请求。每个请求标识至少一个资源和所请求的截止期限。接着,来自客户端的数据被调度器存储在数据库中。调度器随后基于这些请求中的所请求截止期限以及基于该便携式计算设备内的资源的当前状态来确定用于处理这些请求的时间和顺序。调度器随后在所确定的时间并根据所确定的顺序将这些请求传达给各资源。

[0009] 调度器可以自行斟酌以响应于从客户端接收到新请求命令而将一请求调度在其所请求的截止期限之后。调度器可允许与休眠处理器状态相对应的休眠集使处理器断电,从而该处理器可在该处理器退出休眠处理器状态时继续对一个或多个经调度请求进行工作。如果调度器在经调度休眠状态期间接收到非预期请求,则其可确定是否需要一个或多个经调度请求来响应于此非预期请求。如果不需要一个或多个经调度请求来响应于在经调度休眠状态期间的此非预期请求,则调度器可取消一个或多个经调度请求并随后重新调度所取消的经调度请求中的一个或多个请求以在下一经调度活跃状态中发生。

[0010] 附图简述

[0011] 在附图中，除非另行指出，否则相似的附图标记贯穿各视图指示相似的部分。对于带有字母字符命名的参考标号（诸如，“102A”或“102B”），该字母字符命名可区分同一附图中存在的两个相似部件或元素。在意图使一参考标号涵盖所有附图中具有相同参考标号的所有部件时，可略去参考标号的字母字符命名。

[0012] 图 1 是解说用于便携式计算设备（“PCD”）中的分布式资源管理的系统的示例性元件的功能框图；

[0013] 图 2 是解说其中第一处理器需要请求受第二处理器控制的资源的实例的示例的功能框图；

[0014] 图 3 是管理 PCD 的资源的节点架构的第一方面的示图；

[0015] 图 4 是 PCD 的一组示例性资源的有向无环资源图；

[0016] 图 5 是管理 PCD 的资源的节点架构的第二方面的一般示图；

[0017] 图 6 是管理 PCD 的资源的节点架构的第二方面的具体示图；

[0018] 图 7 是解说用于创建用于管理 PCD 的资源的节点架构的方法的流程图；

[0019] 图 8 是解说用于创建用于管理 PCD 的资源的节点架构的方法的接续流程图；

[0020] 图 9 是解说图 7-8 的用于在 PCD 的软件架构中接收节点结构数据的子方法或例程的流程图；

[0021] 图 10 是解说图 7-8 的用于在 PCD 的软件架构中创建节点的子方法或例程的流程图；

[0022] 图 11 是解说图 10 的用于在 PCD 的软件架构中创建客户端的子方法或例程的流程图；

[0023] 图 12 是解说用于在用于 PCD 的软件架构中创建对资源的客户端请求的方法的流程图；

[0024] 图 13 解说了两个处理器之间的通信路径，这两个处理器各自控制其自身资源图的资源；

[0025] 图 14 是示出控制器、资源功率管理器、主处理器、低级驱动器、共享资源和本地资源间的关系的功能框图；

[0026] 图 15 是解说关于控制器和触发集的细节的功能框图；

[0027] 图 16 解说了处理器的示例性活跃 - 休眠触发集；

[0028] 图 17 是解说用于管理触发集以及以其他方式将处理器从第一应用状态（诸如苏醒状态）转变成第二应用状态（诸如休眠状态）的方法的逻辑流程图；

[0029] 图 18 是解说用于管理触发集以及以其他方式将处理器从第二应用状态（诸如休眠状态）转变成第三应用状态（诸如苏醒状态）的方法的逻辑流程图；

[0030] 图 19 是控制器缓冲存储器的功能框图；

[0031] 图 20 是解说用于将处理器从第一应用状态（诸如苏醒状态）转变成第二应用状态（诸如休眠状态）的替换方法的逻辑流程图；

[0032] 图 21 是解说节点调度器、调度器数据库、以及资源功率管理器和其它节点架构系统元件之间的关系的功能框图；

[0033] 图 22 是解说调度器数据库的示例性内容的示图；

[0034] 图 23 解说了演示客户端、客户端请求、调度器以及定时器之间的关系的示例性时

序图；

[0035] 图 24 解说了演示客户端、客户端请求、调度器、定时器以及跟踪休眠集的控制器之间的关系的示例性时序图；

[0036] 图 25 解说了当一个或多个用户线程 2301 通过调用调度器 2101 在框 1205A、1205B 中创建两个请求 675 时的示例性时序图；

[0037] 图 26 解说了当用户线程经由客户端可以确定它不再想要一特定请求得到处理并且发布一撤销调度请求命令时的示例性时序图；

[0038] 图 27 解说了一示例性时序图，其中已经调度了请求并且用户线程已经发布该请求，但是在一撤销调度请求命令已被发布时调度器尚未开始处理前述请求；

[0039] 图 28 解说了一示例性时序图，其中已经调度了一请求并且用户线程 2301 已经发布该请求，并且在一撤销调度请求命令已被发布时调度器 2101 已经开始处理前述请求；

[0040] 图 29 解说了简单场景，其中单个应用程序正在运行并且在活跃状态期间处理一个或多个请求后，CPU 可以进入休眠状态，在此休眠状态中调度器使用新的请求功能 / 特征；

[0041] 图 30 解说了调度器如何管理在经调度的休眠状态期间的非预期唤醒或中断的示例性场景。

[0042] 详细描述

[0043] 措辞“示例性”在本文中用于表示“用作示例、实例或解说”。本文中描述为“示例性”的任何方面不必被解释为优于或胜过其他方面。

[0044] 在本描述中，术语“应用”还可包括具有诸如：对象代码、脚本、字节码、标记语言文件和补丁之类的可执行内容的文件。此外，本文中引述的“应用”还可包括本质上不可执行的文件，诸如可能需要被打开的文档或需要被访问的其他数据文件。

[0045] 术语“内容”也可包括具有可执行内容的文件，可执行内容诸如：对象代码、脚本、字节码、标记语言文件和补丁。此外，本文中引述的“内容”也可包括本质上不可执行的文件，诸如可能需要被打开的文档或需要被访问的其他数据文件。

[0046] 如在本描述中所使用的，术语“组件”、“数据库”、“模块”、“系统”和类似术语旨在引述计算机相关实体，任其是硬件、固件、硬件与软件的组合、软件，还是执行中的软件。例如，组件可以是但不限于在处理器上运行的进程、处理器、对象、可执行件、执行的线程、程序、和 / 或计算机。作为解说，计算设备上运行的应用和计算设备两者都可以是组件。一个或多个组件可驻留在进程和 / 或执行的线程内，并且组件可本地化在一台计算机上和 / 或分布在两台或更多台计算机之间。另外，这些组件可从其上存储有各种数据结构的各种计算机可读介质来执行。各组件可借助于本地和 / 或远程过程来通信，诸如根据具有一个或多个数据分组的信号（例如，来自借助于该信号与本地系统、分布式系统中的另一组件交互、和 / 或跨诸如因特网等网络与其它系统交互的一个组件的数据）。

[0047] 在本描述中，术语“通信设备”、“无线设备”、“无线电话”、“无线通信设备”和“无线手持机”被可互换地使用。随着第三代（“3G”）和第四代（“4G”）无线技术的到来，更大的带宽可用性已经使得能够实现具有更多样化的无线能力的更便携的计算设备。

[0048] 在本描述中，术语“便携式计算设备”（“PCD”）用于描述基于有限容量的电源（诸如电池）进行操作的任何设备。尽管电池运行的 PCD 已经被使用了数十年，但可充电电池

方面的技术进步结合第三代（“3G”）和第四代（“4G”）无线技术的到来，已经使得能够实现具有多种能力的多种 PCD。因此，PCD 可以是蜂窝电话、卫星电话、寻呼机、个人数字助理（“PDA”）、智能电话、导航设备、智能本或阅读器、媒体播放器、上述设备的组合、以及具有无线连接的膝上型计算机等。

[0049] 图 1 是无线电话形式的 PCD 100 的示例性非限定性方面的功能框图，该 PCD 100 用来实现用于便携式计算设备中的分布式资源管理的方法和系统。如所示，PCD 100 包括片上系统 102，该片上系统 102 具有多核中央处理单元（“CPU”）110A、图形处理器 110B、以及模拟信号处理器 126。如本领域普通技术人员已知的，这些处理器 110A、110B、126 可被一起耦合在一个或多个系统总线或另一互连架构上。

[0050] CPU 110A 可包括第零核 222、第一核 224 等、直到第 N 核 226，如本领域普通技术人员所理解的。在替换性实施例中，如本领域普通技术人员所理解的，也可使用一个或多个数字信号处理器（“DSP”）来替代 CPU 110A 和图形处理器 110B。此外，在替代实施例中，可包括两个或更多个多核处理器。

[0051] 如图 1 中所解说的，显示器控制器 128 和触摸屏控制器 130 耦合至多核 CPU 110A。片上系统 102 外部的触摸屏显示器 132 被耦合至显示控制器 128 和触摸屏控制器 130。PCD 100 中还包括视频编码器 / 解码器（“编解码器”）134，例如逐行倒相制（“PAL”）编码器、顺序传送彩色与存储（“SECAM”）编码器、国家电视系统委员会（“NTSC”）编码器或耦合到多核中央处理单元（“CPU”）110A 的任何其他类型的视频编码器 134。视频放大器 136 耦合至视频编码器 134 和触摸屏显示器 132。视频端口 138 被耦合至视频放大器 136。如图 2 中所描绘的，通用串行总线（“USB”）控制器 140 被耦合至 CPU 110A。而且，USB 端口 142 被耦合至 USB 控制器 140。订户身份模块 (SIM) 卡 146 也可被耦合至 CPU 110A。此外，如图 1 中所示，数码相机 148 可被耦合至 CPU 110A。在示例性方面，数字相机 148 是电荷耦合器件（“CCD”）相机或互补金属氧化物半导体（“CMOS”）相机。

[0052] 如图 1 中进一步解说的，立体声音频编解码器 150 可被耦合至模拟信号处理器 126。此外，音频放大器 152 可被耦合至立体声音频编解码器 150。在示例性方面中，第一立体声扬声器 154 和第二立体声扬声器 156 被耦合至音频放大器 152。图 1 示出了话筒放大器 158 也可被耦合至立体声音频编解码器 150。另外，话筒 160 可被耦合至话筒放大器 158。在特定方面中，调频（“FM”）无线电调谐器 162 可被耦合至立体声音频编解码器 150。同样，FM 天线 164 被耦合至 FM 无线电调谐器 162。此外，立体声头戴式受话机 166 可被耦合至立体声音频编解码器 150。

[0053] 图 1 进一步指示了射频（“RF”）收发机 168 可被耦合至模拟信号处理器 126。RF 开关 170 可被耦合至 RF 收发机 168 和 RF 天线 172。如图 1 中所示，按键板 174 可被耦合至模拟信号处理器 126。同样，带话筒的单声道头戴式送受话器 176 可被耦合至模拟信号处理器 126。此外，振动器设备 178 可被耦合至模拟信号处理器 126。图 1 还示出了电源 180（例如电池）被耦合至片上系统 102。在一特定方面，电源 180 包括可再充电电池或来源于连接到交流（“AC”）电源的 AC-DC 变换器的直流（“DC”）供电电源。

[0054] PCD 100 的上述元件中的某些可包括硬件，而另一些可包括软件，并且还有一些可包括硬件和软件的组合。术语“资源”在本文被用来指代可由处理器控制的任何这样的元素，而不管是硬件、软件或其组合。资源可在一个方面被定义为这种元件的功能性的包封。

除非另行指出，否则术语“处理器”或“主处理器”在本文被用来指代处理器，诸如 CPU 110、图形处理器 110B、模拟信号处理器 126、或在软件、固件或类似控制逻辑的控制下操作的任何其他处理器、控制器或类似元件。

[0055] 如下文更详细地描述的，资源的一示例是在处理器上执行的软件元素。在处理器上执行的线程（诸如举例而言与正在执行的应用程序有关的线程）可通过致使对资源发布“请求”来访问资源。如以下所描述的，资源请求是通过在本公开中被称为“框架”的基于软件的系统来处理的。

[0056] 术语“客户端”在本公开中被广泛用来指对请求资源的功能产生影响的元素。因此，如本文中对这些术语的使用那样，出于发布资源请求的目的，线程可创建或使用客户端。应当注意，在一些实例中，资源可创建或使用客户端，从而使得资源可引发针对另一资源发布资源请求。如以下将进一步详细描述的，由于发起请求的资源和被请求的资源之间的依赖关系，这样的其它资源可在本文中被称为“依赖性”资源。资源和客户端可由存储器中的数据结构来表示。

[0057] 由于资源受多处理器 PCD 100 中的特定处理器的控制，因此并不是 PCD 100 中的每个处理器都能够访问 PCD 100 中的每个资源。图 2 解说了其中可能希望 PCD 100 中的第一处理器 202 发布对受 PCD 100 中的第二处理器 206 控制的资源功率管理器 157 的资源请求 203 的实例的示例。注意，第一处理器 202 还可控制多个资源 105A1、105A2。类似地，第二处理器 206 可控制多个附加资源 105B1、105B2。

[0058] 在其中第一处理器 202 正执行关于例如视频播放器应用程序的线程 208 的实例中，线程 208 可要求调整第一处理器 202 的一个或多个工作参数，这些工作参数增强第一处理器 202 的性能。（尽管出于清楚起见，线程 208 和资源功率管理器 157 被概念性地解说为驻留在其各自相应的处理器 202 和 206 中，但本领域普通技术人员理解此类元件可根据充分理解的计算原理由处理器在该处理器的存储器空间中执行或另行操作的。）此类工作参数可包括例如，时钟速度和总线速度。

[0059] 例如，各种处理器可以使用相同的总线时钟，但是仅其中一个处理器具有对总线时钟的直接（硬件级）控制。提高时钟速度可以导致例如视频播放器应用程序的更好性能，因为视频的回放一般相比其它某些任务是处理能力更为密集的任务。因为处理能力一般以百万条指令每秒（“MIPS”）来表达，所以线程 208 可以发布对某个数目的 MIPS 的要求。资源功率管理器 157 可包括一算法，该算法响应于对指定数目的 MIPS 的请求来引起信号 210 上的变化，信号 210 可以表示时钟速度、总线速度或促进第一处理器 202 以所请求的 MIPS 等级来工作的其它参数。

[0060] 线程通过因总线或协议而异的应用程序接口 (API) 来访问资源功率管理器 157 可以是可能的，第一处理器 202 可以通过该 API 与第二处理器 206 通信。然而，以下描述的框架可以提供比因资源而异和因总线而异的 API 更为统一的处置资源请求的方式。如下所述，经由该框架，以统一方式发布和服务于资源请求，而不考虑该请求是针对受从其发布该资源请求的同一处理器控制的资源，还是针对受不同处理器控制的资源。受从其发布资源请求的同一处理器控制的资源可被称为“原生”资源。由除了从其发布资源请求的处理器以外的处理器控制的资源在本文中可被称为“远程资源”或“分布式资源”。

[0061] 另外，发布针对远程资源的请求招致时间延迟或等待时间形式的处理开销。即，在

处理器之间发送与资源请求有关的一个或多个消息需要一定量的时间。在一些实例中，单个资源请求可以导致多个处理器间消息。

[0062] 图 3 是包括代表 PCD 100 的软件或硬件（或两者）的功能框的示图。线“A”左边的各框表示 PCD 100 的受 CPU 110A 控制的资源。此类资源可包括：还一般化地称为第一硬件元素（硬件元素 #1）的 CPU 110A 自身；还一般化地称为第二硬件元素（硬件元素 #2）的用于 CPU 110A 的时钟 442；还一般化地称为第三硬件元素（硬件元素 #3）的总线仲裁器或调度器 422；还一般化地称为第一软件元素（软件元素 #1）的总线程序 A-444A；还一般化地称为第二软件元素（软件元素 #2）的总线程序 B-444B；还一般化地称为第三软件元素（软件元素 #3）的时钟程序 AHB；以及由软件元素监视的被一般地指示为键按压 448 的动作或功能。

[0063] CPU 110A 控制或者具有对以上引述的资源的访问，因为这些资源在 CPU 110A 的存储器空间内，并且没有其它将会禁止 CPU 110A 访问这些资源的限制（诸如安全性限制）存在。例如，CPU 110A 可以能够控制或访问这些资源的硬件寄存器。应该注意，PCD 100 可包括控制或具有对除了以上引述的资源的资源访问的其它 CPU 110（参见例如图 2）。

[0064] 可包括计算机指令库的框架管理器 440 管理封装这些资源的功能性的节点。即，可访问这些节点以间接访问这些资源。出于便利，封装了资源的功能性的节点可在本文中被称为包括、包含、具有（等等）该资源。每个节点包括一个或多个资源。可以软件代码、固件、或者类似介质来定义各节点，并且各节点在 PCD 100 的操作期间在例如存储器 112（图 1）中被实例化为数据结构。

[0065] 节点 601 可在启动、上电、初始化、引导等序列期间或者在 PCD 100 的操作期间的其它任何适合时间被实例化。应注意，本文对实例化资源、对资源发出请求、或者另行与资源交互的引述应被理解为意指与包括该资源的节点交互。对于本公开的其余部分，将如以下参照图 5 所描述的那样用参考标号 601 来标示普适或非特定的节点。

[0066] 例如，节点 601 可包括具有与第一硬件元素或即中央处理单元 110 一般性对应的单个资源的第一节点 602。在本公开所描述的软件架构的情况下，可为节点 601 的每个资源提供包括一个或多个字母数字字符的唯一性名称。在图 3 中所解说的示例性实施例中，第一节点 602 的资源已经被指派有资源名称“/core/cpu”。该示例性资源名称一般对应于本领域普通技术人员所知的常规文件命名结构。然而，如本领域普通技术人员所认识到的，包含字母数字字符和 / 或符号的任何其他组合的其它类型的资源名称也完全在本公开的范围之内。

[0067] 节点 601 可进一步包括，例如具有多个资源的第二节点 622。在这一示例性实施例中，第二节点 622 具有包括与总线仲裁器或调度器 422 相对应的单个硬件元素的第一资源。第二节点 622 的第二资源包括与总线程序 A 的第一软件元素 444A 一般性对应的软件元素。第二节点 622 的第三资源包括与总线程序 B 的第二软件元素 444B 一般性对应的另一软件元素。本领域普通技术人员认识到，给定节点 601 的任何组合的和任何数目的资源和资源类型也完全在本公开的范围之内。

[0068] 图 3 还解说了与这两个软件元素 448、450 的动作或功能一般性对应的第一客户端 648。在图 3 中所解说的示例性实施例中，第一客户端 648 一般对应于可在由便携式计算设备 100 支持的特定应用程序模块 105 内发生的键按压动作。然而，本领域普通技术人员认

识到，除键按压之外的其他的软件元素动作和 / 或功能也完全在本公开的范围之内。关于客户端请求 648 及其各自的创建的进一步细节将在下面结合图 11 来描述。

[0069] 图 3 还解说了特定架构性元素之间的关系。例如，图 3 解说了客户端 648 和第一节点 602 之间的关系。具体而言，第一客户端 648 可生成用虚线解说的客户端请求 675A，该客户端请求 675A 由包括资源“/core/cpu”的第一节点 602 管理或处置。典型地，有预定或设定数目的类型的客户端请求 675。客户端请求 675 将在下面进一步结合图 11 详细描述。

[0070] 图 3 中显示的其他关系包括用虚线 680 解说的依存性。依存性是另一节点 601 的各个资源之间的关系。依存性关系通常指示第一资源 (A) 依赖于可为该第一资源 (A) 提供信息的第二资源 (B)。该信息可以是由第二资源 (B) 执行的操作的结果或者其可简单地包括该第一资源 (A) 所需的状态信息，或是其任何组合。该第一资源 (A) 和第二资源 (B) 可以是同一节点 601 的一部分或者它们可以是不同节点 601 的部分。应注意，这些客户端请求 675 可不仅源自执行线程（诸如在以上描述的键按压动作的示例中那样），还可源自其它节点 601。为了从依存节点 601 获得信息，节点 601 可以向其依存节点 601 发布客户端请求 675。因此，指示依存性的虚线 680 还可指示潜在客户端请求 675 的方向。

[0071] 在图 3 中，第一节点 602 如由依存性箭头 680B 所指示地依存于第二节点 622，该依存性箭头 680B 起始自第一节点 602 并且延伸到 622 处的第二节点。图 3 还解说了第一节点 602 还可依存于第三节点 642，如由依存性箭头 680A 所解说的。图 3 还解说了第二节点 622 依存于第四节点 646，如由依存性箭头 680C 所解说的。本领域普通技术人员认识到，用图 3 的虚箭头解说的依存性 680 本质上仅仅是示例性的并且各个节点 601 之间的其他依存性组合也在本公开的范围之内。

[0072] 框架管理器 440 负责维护如上所述的关系，包括但不限于图 3 所解说的客户端请求 675 和依存性 680。一些此类关系（诸如依存性）藉由资源及其节点 601 已经在 PCD 100 的软件代码中定义的方式存在于 PCD 启动时间（即，上电、初始化、引导等），框架管理器 440 在此种启动时间对其进行访问以开始节点实例化过程。其它此类关系（诸如客户端请求 675）在节点 601 已经被实例化之后发生，诸如在其中应用程序调动资源的应用程序线程的执行期间。无论客户端请求 675 源自执行中的应用程序线程或除节点 601 以外的类似元件（例如客户端请求 675A），还是源自节点 601，客户端请求 675 均被定向通过框架管理器 440。框架管理器 440 在各节点 601 间引导信息传递。在概念上，框架管理器 440 用作多个线程可以藉以基本上与各节点 601 并发通信的矩阵。尽管不同的线程可以涉及不同的数据，但同一框架管理器软件代码可服务多个线程。

[0073] 如以下进一步详细描述的，框架管理器 440 可以在节点 601 的依存性节点一被初始化（即在任何给定节点 601 的依存性 680 已经被解析时）就实例化该节点 601。框架管理器 440 尝试实例化已经在 PCD 100 的软件架构中定义的所有节点 601。当支持依存性的资源存在或者处于准备好处置与依存性 680 有关的信息的就绪状态中时，依存性 680 就是完整的或已解析的。

[0074] 例如，如果包括单个资源“/clk/cpu”的第三节点 642 尚未被实例化，那么包括单个资源“/core/cpu”的第一节点 602 就因在第一节点 602 和第三节点 642 之间存在的依存性关系 680A 而不可被框架管理器 440 实例化。一旦第三节点 642 已经由框架管理器 440 实例化，那么由于依存性关系 680A，框架管理器 440 就可实例化第二节点 602。

[0075] 如果因为特定节点601的依存性680中的一个或多个依存性不完整或未解析而导致框架管理器440不能实例化该特定节点601，则框架管理器440将继续运行或执行与已被成功实例化的那些节点601相对应的步骤。框架管理器440通常将跳过对可能因不完整的依存性（其中依存资源尚未被创建）故而不存在的特定节点601的调用，并且将反映该不完整状态的消息返回给该调用。

[0076] 在诸如图1中所解说的多核环境中，框架管理器440可在分别的核（诸如图1的第零、第一和第N核222、224、和226）上创建或实例化各节点601。一般可在多核环境中在分别的核上且并行地创建各节点601，只要这些节点601不彼此依存并且如果特定节点的所有对应的依存性如以下描述地为完整的即可。在多处理器环境中，节点601可在各种处理器上被创建或实例化，诸如图1的CPU 110A、图形处理器110B等。即，一些节点601可存在于一个处理器的存储器空间中，而其它节点601可存在于另一处理器的存储器空间中。然而，应注意，一个处理器上的节点601对于另一处理器上的节点601可以不是仅经由框架管理器440即可访问的。

[0077] 类似于以上描述的（主）框架管理器440的远程框架管理器300可与框架管理器440并行存在。远程框架管理器300与框架管理器440协同或一起工作以协调在不同处理器上的节点601之间的处理器间信息传递。即，远程框架管理器300帮助框架管理器440在所涉及的各节点601存在于不同处理器上的实例中维护上述关系，诸如依存性和客户端请求。

[0078] 因此，经由框架管理器440和300的组合效果，或许并不可使一个处理器上的节点601对另一其它处理器上的节点601呈可访问。此外，框架管理器440和300的组合可以执行本公开中归于框架管理器440的所有功能，无论所涉及的节点601是存在于同一处理器还是不同处理器上。在此种多处理器实施例中，框架管理器300和440所包含的软件的个体副本可驻留在每个处理器的域中。因此，每个处理器具有对同一框架管理器软件的访问。

[0079] 图4方便地以有向无环图（“DAG”）400的形式重新组织了上述节点602、622、642和646。图400是定义上述软件架构的另一种方式。在图论的语汇中，图400的顶点对应于节点601，图400的边对应于客户端请求675，而毗邻节点或顶点表示资源依存性。本领域普通技术人员将认识到，图400作为依存性的结果是有向图，并且因为框架管理器440防止定义在其中资源A依存于资源B并且资源B依存于资源A的循环故而是无环的。即，框架管理器440将不实例化被（错误地）定义为彼此依存的两个节点601。

[0080] 该图的无环属性对于防止死锁是重要的，因为如下所述，每个节点601在其被访问时被锁定（在事务处理意义上）。假使在第一线程要访问并锁定两个节点601之一同时第二线程要访问并锁定这两个节点601中的另一个的实例中这两个节点601彼此依存，则这两个线程均将被挂起。

[0081] 然而，在其中软件开发者或所涉及的其它此类人员在定义软件架构时认为希望要在该软件架构中定义两个彼此依存的资源的相对较罕有的实例中，这两个（或更多个）资源可被包括在与彼此相同的节点601中。同一节点中的两个资源共享相同锁定状态。至少部分地出于这一原因，软件开发者或其它此类人员可选择定义多资源节点，诸如该架构中的节点622。

[0082] 尽管本公开可能出于清楚和便利性的原因，引述“节点”601而非节点601的“资

源”，但应该理解，客户端请求可被定向至指定资源而非节点。换句话说，节点 601（其如上所述地可以是封装一个或多个资源的功能性的数据结构）从客户端或客户端请求的其它发布者（诸如另一节点 601）的角度来看可以是透明的。从客户端的角度看，请求是针对资源而非节点来发布的。类似地，从客户端的角度看，状态查询、事件、或者该架构的其它元素与资源而非节点相关联。

[0083] 诸如示例性图 400 之类的资源图对于理解根据依存性的节点 601 的实例化而言是有用的，以下关于图 6-10 对此进行描述。叶节点（诸如节点 642 和 646）在非叶节点之前被实例化，因为叶节点没有依存性。一般来说，节点 601 必须在依存于它的节点可被实例化之前被实例化。此外，可以看出，服务于资源请求对应于遍历有向无环图，在其中顶点对应于节点 601，边对应于客户端请求 675，而毗邻节点或顶点表示资源依存性。

[0084] 在多处理器 PCD 100 中，第一处理器可具有对第一资源图中的第一组节点 601 的访问或能够控制第一组节点 601，而第二处理器可具有对第二资源图中的第二组节点 601 的访问或能够控制第二组节点 601，其中第一和第二资源图不共享任何资源，即它们是资源互斥的。即，在此种环境中，每个处理器具有其自身的资源图，其定义其它处理器所不可访问的资源和其它元素间的关系。本公开的分布式资源管理涉及在其中两个或更多个处理器各自具有对其自身资源图中的资源的访问并且不具有对其它处理器的资源图中的资源的访问的实例中维护上述关系，诸如依存性和客户端请求。

[0085] 在一些实施例中，以上引述的对资源访问的限制可受硬件配置所限。即，处理器可能不具有它能藉以影响硬件设备（诸如寄存器）的手段，因为该硬件设备受另一处理器的存储器空间控制或处于另一处理器的存储器空间中。替换地，或者附加地，出于诸如使得处理器被暴露于安全性风险（例如，可能感染另一处理器的病毒）的情况最小化之类的原因，对资源访问的限制可在软件中施加。

[0086] 图 5 是用于管理图 1 的 PCD 100 的资源的系统的软件架构 500B1 的另一方面的一般示图。出于清楚目的在 PCD 100 的上下文以及其中所涉及的所有资源和其它元素由同一处理器控制（即，它们被包括在同一资源图中）的架构中描述了这一方面。在该一般示图中，尚未给每个节点 601 的这一个或多个资源提供唯一性名称。图 5 的节点或资源图 500B1 仅包括节点 601、客户端 648、事件 690、和由架构或框架管理器 440 支持的查询功能 695。已经用椭圆形以及具有表示节点 601 内的资源之间的各个依存性的具体方向的箭头 680 来解说每个节点 601。

[0087] 图 5 还解说了第一节点 601A 的客户端 648 可如何向第一节点 601A 发布客户端请求 675。在发布这些客户端请求 675 之后，第二节点 601B 可触发事件 690 或者提供对查询 695 的响应，其中与事件 690 和查询 695 相对应的消息流回到客户端 648。

[0088] 图 6 是用于管理图 1 的 PCD 100 的资源的系统的软件架构 500B1 的上述方面的更为具体的示图。图 6 解说了仅包括具有具体的的、但为示例性的资源名称的各节点 601、以及与图 3 的诸项相对应的客户端 648、事件 690、和查询功能 695 的节点或资源图 500B2。已经用椭圆形以及具有表示节点 601 内的资源之间的各个依存性的具体方向的箭头 680 来解说每个节点 601。

[0089] 例如，第一节点 602 具有依存性箭头 680B 以指示第一节点 602 依存于第二节点 622 的这三个资源。类似地，包括第二软件元素 444B 且在图 11C 中用参考字母“C”一般性

标示的第三资源“/bus/ahb/sysB/”具有指示该第三资源 (C) 依存于第四节点 646 的该单个“/clk/sys/ahb”资源的依存性箭头 680C。

[0090] 图 6 还解说了来自各节点 601 的输出数据，输出数据可包括一个或多个事件 690 或查询功能 695。查询功能 695 类似于事件 690。查询功能 695 可具有可以是或可以不是唯一性的查询句柄。该查询功能一般不被外部地标识并且一般而言其不具有状态。查询功能 695 可被用来确定节点 601 的特定资源的状态。查询功能 695 和事件 690 可具有与建成的客户端 648 的关系并且这些关系由方向性箭头 697 表示以指示来自各个事件 690 和查询功能 695 的信息被传递给特定客户端 648。

[0091] 图 5-6 的节点或资源图 500B 表示存在于在处理器控制下的存储器中并且由框架管理器 440 管理的关系。节点或资源图 500B 可由框架管理器 440 自动地生成以作为有用工具来用于标识由框架管理器 440 所管理的各个元素之间的关系以及用于由软件团队进行故障诊断。

[0092] 图 7 是解说用于创建或实例化用于管理 PCD 100 的（诸）资源的软件结构的方法 1000A 的流程图。出于清楚目的，是在在其中所涉及的所有资源和其它元素由同一处理器控制（即，它们被包括在同一资源图中）的架构的上下文中描述了这一方法。

[0093] 框 1005 是用于管理 PCD 100 的资源的方法或过程 1000 的第一例程。在框 1005 中，例程可由框架管理器 440 执行或运行以接收节点结构数据。该节点结构数据可包括勾勒特定节点 601 与其它诸节点 601 可具有的依存性的依存性阵列。关于节点结构数据及此例程或子方法 1005 的进一步细节将在下面结合图 9 进一步详细描述。

[0094] 接着，在框 1010 中，框架管理器 440 可审阅作为在框 1005 中接收到的节点结构数据的一部分的该依存性数据。在判决框 1015 中，框架管理器 440 可确定该节点结构数据是否定义叶节点 601。叶节点 601 一般意味着要基于该节点结构数据来创建的节点不具有任何依存性，诸如图 3-4 中的节点 642 和 646。如果对判决框 1015 的询问是肯定的（这意味着用于创建当前节点的该节点结构数据不具有任何依存性），则框架管理器 440 继续行至例程框 1025。

[0095] 如果对判决框 1015 的询问是否定的，则跟随“否”分支到判决框 1020，其中框架管理器确定该节点结构数据内的所有硬依存性是否都存在。硬依存性可包括其中没有它则资源不能存在的依存性。同时，软依存性可包括其中资源可作为任选性步骤来使用依存资源的依存性。软依存性意味着具有软依存性的节点 601 或该节点 601 的资源甚至可在该软依存性并不存在时在该节点架构内被创建或实例化。

[0096] 软依存性的示例可包括对于包含多个资源的面向资源的节点 601 的操作而言并非必不可少的优化特征。框架管理器 440 可因节点或资源的所有硬依存性均存在而创建或实例化该节点或资源，甚至在那些有未被创建的软依存性的节点或资源的软依存性不存在时亦是如此。回调特征可被用来引用该软依存性，以便当该软依存性对框架管理器 440 变得可用时，框架管理器 440 将向引用该软依存性的每个回调作出关于这些软依存性现在可用的通知。

[0097] 如果对判决框 1020 的询问是否定的，则跟随“否”分支到框 1027，其中该节点结构数据由框架管理器 440 存储在诸如存储器之类的临时存储中并且框架管理器 440 创建与该未经实例化的节点相关联的回调特征。

[0098] 如果对判决框 1015 的询问是肯定的，则跟随“是”分支到例程 1025，其中基于在例程框 1005 中接收到的节点结构数据来创建或实例化节点 601。例程框 1025 的进一步细节将在下面结合图 9 描述。接着，在框 1030 中，框架管理器 440 使用新创建节点 601 的（诸）唯一性资源名称来公布此新创建节点 601，从而其他节点 601 可向 / 从此新创建节点 601 发送信息或接收信息。

[0099] 现在参考作为图 7 的接续流程图的图 8，在框 1035 中，框架管理器 440 向依存于此新创建节点 601 的其它节点 601 作出此新创建节点 601 已经被实例化并且准备好接收或传送信息的通知。根据一个示例性方面，当依存节点（如图 5 的节点 601B）被创建时立即触发通知，即，通知被递归地执行。所以如果图 5 的节点 601B 被构建，则节点 601A 立即得到通知。该通知可允许节点 601A 被构建（因为节点 601B 本是节点 601A 的最终依存性）。节点 601B 的构建可使其它诸节点 601 得到通知，依此类推。直到依存于节点 601B 的最终资源完整了，节点 601B 才变得完整。

[0100] 稍微更复杂一些的第二实现是要将所有通知放到单独通知队列上，并且然后在单个时间点开始遍历该队列，即，这些通知被迭代地执行。因此当图 5 的节点 601B 被构建时，对节点 601A 的通知被推送到列表上。然后该列表被执行并且节点 601A 得到通知。这使对（除节点 601A 之外的、未在图 5 中解说的）其他附加节点 601 的通知也被放在同一列表上，并且在向节点 601A 的通知被发送之后随后发送这些通知。直到与节点 601B 和节点 601A 相关联的所有工作都已经完成之后，对（除对节点 601A 的通知之外的）其它节点 601 的通知才会发生。

[0101] 逻辑上，这两个实现是等同的，但是它们在被实现时具有不同的存储器消耗特性。递归实现是简单的但可能消耗任意量的栈空间，其中栈消耗是依存性图的深度的函数。迭代实现稍微更复杂些并且需要更多一点的静态存储器（通知列表），但是不管依存性图（诸如图 5 所解说）的深度如何，栈使用是恒定的。

[0102] 同样，框 1035 中的对节点创建的通知不限于其它节点。它也可被内部地用于别名构建。不只是其他节点，系统 500A 中的任何任意性元素均可使用此相同机制来请求在节点变得可用时得到通知。节点和非节点两者皆可使用相同通知机制。

[0103] 在判决框 1040 中，框架管理器 440 确定基于当前节点 601 的创建，是否有其它节点 601 或软依存性现在得到释放以进行创建或实例化。判决框 1040 一般性地确定是否有资源因为最近已经经历创建或实例化的当前节点已经满足了某些依存性关系 680 而可被创建。

[0104] 如果对判决框 1040 的询问是肯定的，那么跟随“是”分支回到例程框 1025，其中由于刚刚被创建的节点 601 满足了依存性，得到释放的节点 601 现在可被创建或实例化。

[0105] 如果对判决框 1040 的询问是否定的，则跟随“否”分支到例程框 1045，其中框架管理器 440 可管理如图 2 中所解说的软件架构的元素之间的通信。接着，在框 1050 中，框架管理器 440 可继续通过使用与特定资源相关联的资源名称来对资源所采取的动作进行日志录入或记录。可在由框架管理器 440 或由框架管理器 440 管理的任何元素（诸如有资源、节点 601、客户端 648、事件 695、和查询功能 697）采取的任何动作之后由框架管理器 440 执行框 1045。框 1045 示出该节点架构的另一方面，其中框架管理器 440 可维护关于活动的流水日志，该日志将由每个元素执行的动作根据由创建了特定元素（诸如节点 601 的

资源）的作者提供的唯一性标识符或名称来列出。

[0106] 与现有技术相比，框 1050 中的列出指派给系统的每个资源的唯一性名称的关于活动的日志录入是独特的并且可提供诸如在调试和错误故障诊断中使用的显著优势。节点架构 500A 的另一独特方面在于，分别的团队可独立于彼此地工作于不同硬件和 / 或软件元素，其中每个团队将能够使用唯一性且易于跟踪的资源名称，而无需创建用于转译由其他团队和 / 或原始设备制造商 (OEM) 指派的不那么有意义且通常令人困惑的资源名称的表。

[0107] 接着，在判决框 1055 中，框架管理器 440 确定由框架管理器 440 记录的活动日志是否已经被请求。如果对判决框 1055 的询问是否定的，则跟随“否”分支到此过程的结束，其中此过程返回到例程 1005。如果对判决框 1055 的询问是肯定的，则跟随“是”分支到框 1060，其中框架管理器 440 向输出设备（诸如打印机或显示屏和 / 或其两者）发送包括有意义的资源名称和由这些资源名称执行的各个动作的活动日志。该过程然后返回到如上所述的例程框 1005。

[0108] 图 9 是解说图 7 的用于接收定义 PCD 100 的软件架构的节点结构数据的子方法或例程 1005 的流程图。该接收方法可在任何合适时间发生，诸如举例而言，当 PCD 100 启动或初始化时。在此种实例中，当处理器在根据该架构准备实例化节点 601 期间从存储器读取相应的软件代码时，接收节点结构数据。

[0109] 框 1105 是图 7 的子方法或例程 1005 中的第一步骤。在框 1105 中，框架管理器 440 可接收给软件或硬件元素（诸如图 7 的 CPU 110 和时钟 442）的唯一性名称。如先前讨论的，节点 601 必须引用至少一个资源。每个资源在系统 500A 中具有唯一性名称。系统 500A 内的每个元素可用唯一性名称来标识。每个元素具有从字符视角来看唯一性的名称。换言之，一般而言，系统 500A 内没有两个元素具有相同名称。根据本系统的示例性方面，节点 601 的资源一般可具有跨该系统唯一性的名称，但是不要求客户端或事件名称是唯一性的，尽管视希望它们可以是唯一性的。

[0110] 为方便起见，可采用常规的使用正斜杠“/”字符以创建唯一性名称的树文件命名结构或文件命名“隐喻”，诸如但不限于用于 CPU 110 的“/core/cpu”和用于时钟 442 的“/clk/cpu”。然而，如本领域普通技术人员所认识到的，包含任何其他的字母数字字符和 / 或符号组合的其它类型的资源名称也完全在本公开的范围之内。

[0111] 接着，在框 1110 中，框架管理器 440 可为与正被创建的节点 601 的一个或多个资源相关联的一个或多个驱动器功能接收数据。驱动器功能一般包括要由特定节点 601 的一个或多个资源完成的动作。例如，在图 6 中，节点 602 的资源 /core/cpu 的驱动器功能可请求其为提供已被请求的所请求处理量所需的总线带宽量和 CPU 时钟频率。将经由节点 642 和节点 622 中的资源的客户端来作出这些请求。节点 642 中的 /clk/cpu 的驱动器功能将通常负责根据其接收自节点 602 的 /core/cpu 资源的请求来实际设定物理时钟频率。

[0112] 在框 1115 中，框架管理器 440 可接收节点属性数据。该节点属性数据一般包括定义节点策略的数据，节点策略诸如有安全性（是否可经由用户空间应用来访问该节点）、可远程性（是否可从该系统中的其他处理器访问该节点）和可访问性（该资源是否能支持多个并发客户端）之类。框架管理器 440 还可定义允许资源超驰默认框架行为（诸如请求评估或日志录入策略）的属性。

[0113] 接着，在框 1120 中，框架管理器 440 可为正被创建的特定节点 601 接收定制的用

户数据。该用户数据可包括本领域普通技术人员关于“C”编程语言所理解的空“star”字段。用户数据也被本领域普通技术人员称为“trust me(信任我)”字段。示例性定制用户数据可包括但不限于,诸如频率表之类的表、寄存器映射图等。在框 1120 中接收到的用户数据不被系统 500B 引用,但是允许对资源的定制(若该定制不被框架管理器 440 所识别或完全支持)。该用户数据结构是“C”编程语言中的旨在被扩展用于特定或专门用途的基类。

[0114] 本领域普通技术人员认识到,用于扩展特定类的专门用途的其他种类的数据结构也在本公开的范围之内。例如,在“C++”(C 加加)编程语言中,等效结构可包括将变为用于节点 601 内的资源的扩展机制的关键字“public”。

[0115] 接着,在框 1125 中,框架管理器 440 可接收依存性阵列数据。该依存性阵列数据可包括正被创建的节点 601 所依存于的一个或多个资源 601 的唯一性且具体的名称。例如,假使图 6 的第一节点 602 正在被创建,那么在此框 1125 中,依存性阵列数据可包括第一节点 602 所依存于的第二节点 622 的这三个资源和第三节点 642 的这单个资源的资源名称。

[0116] 接着,在框 1130 中,框架管理器 440 可接收资源阵列数据。该资源阵列数据可包括用于正被创建的当前节点的参数,诸如与图 7B-7C 的第一节点 602 有关的参数(假使该第一节点 602 正被创建)。该资源阵列数据可包括以下数据的一者或多者 :其他资源的名称 ;单位 ;最大值 ;资源属性 ;插件数据 ;以及类似于框 1120 的定制用户数据的任何定制的资源数据。该插件数据一般标识检索自软件库的功能并且通常列出可由正被创建的特定节点或特定多个节点支持的客户端类型。该插件数据还允许对客户端创建和拆毁的定制。在框 1130 之后,该过程返回到图 7 的框 1010。

[0117] 在图 9 中,已经用虚线解说了属性数据框 1115、定制用户数据框 1120、和依存性阵列数据框 1125,以指示这些特定步骤是任选的并且对于任何给定节点 601 而言均非必需的。同时,已经用实线解说了唯一性名称框 1105、驱动器功能框 1110、和资源阵列数据框 1130,以指示例程 1005 的这些步骤一般而言对于创建节点 601 是重要的。

[0118] 图 10 是解说图 7 的用于在 PCD 100 中的软件架构中创建节点的子方法或例程 1025 的流程图。例程框 1205 是用于根据一个示例性实施例来实例化或创建节点 601 的子方法或例程 1025 中的第一例程。在例程框 1205 中,在该步骤中创建与正被实例化的节点 601 相关联的一个或多个客户端 648。关于例程框 1205 的进一步细节将在下面结合图 11 更详细地描述。

[0119] 在框 1210 中,框架管理器可创建或实例化与框 705 的节点结构数据相对应的这一个或多个资源。接着,在框 1215 中,框架管理器 440 可激活在例程框 1005 的例程框 1110 中接收到的驱动器功能。根据一个示例性方面,可使用在例程框 1005 的资源阵列数据框 1130 中接收到的最大值来激活这些驱动器功能。根据另一优选示例性方面,可用连同来自例程 1005 的节点结构数据一起传递的任选的初始值来激活每个驱动器功能。如果初始数据未提供,那么以 0 即最小值来初始化该驱动器功能。通常也以使得知道驱动器功能正被初始化的方式来激活该驱动器功能。这使得资源能够执行专门用于初始化的任何操作,但无需在正常或例程操作期间被执行。该过程随后返回到图 7 的步骤 1030。

[0120] 图 11 是解说图 10 的用于在 PCD 100 的软件架构中创建或实例化客户端 648 的子方法或例程 1205 的流程图。框 1305 是其中创建一个或多个资源 601 的客户端 648 的例程框 1205 的第一步骤。在框 1205 中,框架管理器 440 接收被指派给正被创建的客户端 648

的名称。类似于资源名称，客户端 648 的名称可包括任何类型的字母数字和 / 或符号。

[0121] 接着，在框 1310 中，如果对正被创建的客户端 648 有任何特定的定制，那么定制的用户数据可由框架管理器 440 接收。已经用虚线解说了框 1310 以指示该步骤是任选的。框 1310 的定制用户数据类似于以上结合节点 601 的资源的创建来讨论的定制用户数据。

[0122] 在框 1315 中，框架管理器 440 接收指派给正被创建的特定客户端的客户端类型类别。本文中的客户端类型类别可包括四种类型之一 :(a) 要求的、(b) 冲激、(c) 向量、以及 (d) 等时的。可取决于正由系统 101 管理的资源和取决于依赖于节点 601 的资源的应用程序来扩展客户端类型类别列表。

[0123] 要求的类别一般对应于对从被要求的客户端 648 传递到特定资源 601 的标量值的处理。例如，要求的请求可包括某个数目的百万条指令每秒 (MIPS)。同时，冲激类别一般对应于对在某个时段内完成某个活动（对起始时间或停止时间没有任何指定）的请求的处理。

[0124] 等时类别一般对应于对通常重复发生并且具有定义明确的起始时间和定义明确的结束时间的动作的请求。向量类别一般对应于通常作为被串行或并行地要求的多个动作的一部分的数据阵列。

[0125] 接着，在框 1320 中，框架管理器 440 接收指示客户端 648 是已经被指定为同步的还是异步的数据。同步客户端 648 是通常要求框架管理器 440 锁定节点 601 的资源直至该资源 601 返回数据和关于资源 601 已经完成了来自该同步客户端 648 的所请求任务的指示的客户端。

[0126] 另一方面，异步客户端 648 可由被框架管理器 440 访问的一个或多个线程并行地处置。框架管理器 440 可创建对线程的回调并且可在该回调已经被相应线程执行时返回值。本领域普通技术人员认识到，异步客户端 648 并不像同步客户端 648 在该同步客户端 648 的任务正被执行时那样锁定资源。

[0127] 在框 1320 之后，在判决框 1325 中，框架管理器 440 确定由客户端 645 标识出的资源是否可用。如果对判决框 1325 的询问是否定的，则跟随“否”分支到框 1330，其中空值或消息被返回给用户，以指示该客户端 648 此时不能被创建。

[0128] 如果对判决框 1325 的询问是肯定的，则跟随“是”分支到判决框 1335，其中框架管理器 440 确定由客户端 648 标识出的每个资源是否都支持在框 1310 中提供的客户端类型。如果对判决框 1335 的询问是否定的，则跟随“否”分支回到框 1330，其中返回空值或消息，以指示客户端 648 此时不能被创建。

[0129] 如果对判决框 1335 的询问是肯定的，则跟随“是”分支到框 1340，其中框架管理器 440 在存储器中创建或实例化客户端 648。接着，在框 1345 中，如果在框 1310 中接收到任何定制用户数据（诸如任选自变量），则可将这些任选自变量随其各自的资源映射到特定节点 601。接着，在框 1350 中，新创建的客户端 645 被耦合到其对应的处于空闲状态或所请求状态中的一个或多个资源，如上所述。该过程随后返回到图 10 的框 1210。

[0130] 图 12 是解说用于在 PCD 100 的软件架构中创建对资源 601 的客户端请求 675 的方法 1400 的流程图。一般在以上结合图 7-11 所描述的客户端和节点创建（实例化）之后执行方法 1400。

[0131] 框 1405 是用于创建对资源 601 的客户端请求 675 的方法 1400 中的第一步骤。方

法 1400 将描述框架管理器 440 如何处置以下三种类型的客户端请求 675 :(a) 要求的、(b) 冲激、以及 (c) 向量。如以上提及的请求 675 的名称所暗示的,客户端请求 675 一般对应于被创建的且以上描述的客户端类型。

[0132] 在框 1405 中,框架管理器 440 可接收与诸如以上提及的三者之一之类的特定客户端请求 675 相关联的数据,这三者为 :(a) 要求的、(b) 冲激、以及 (c) 向量。与要求的请求相关联的数据一般包括从被要求的客户端 648 传递到特定资源 601 的标量值。例如,要求的请求可包括某个数目的百万条指令每秒 (MIPS)。冲激请求包括对在某个时间段内完成某个活动 (对起始时间或停止时间没有任何指定) 的请求。

[0133] 用于向量请求的数据一般包括被要求串行或并行完成的多个动作的阵列。向量请求可包括任意性长度的值。向量请求通常具有大小值和值阵列。节点 601 的每个资源可被扩展成具有指针字段以支持向量请求。在“C”编程语言中,如本领域普通技术人员所理解,该指针字段由 union 函数支持。

[0134] 接着,在框 1410 中,框架管理器 440 通过由以上结合图 11 所描述的方法创建的客户端 648 来发布请求。接着,在框 1415 中,如果请求是要求的类型或向量类型,那么框架管理器 440 对正通过该客户端传递的请求数据进行双重缓冲。如果该请求是冲激类型,那么框 1415 被框架管理器 440 跳过。

[0135] 对于要求的请求,在该框 1415 中,来自在先请求的值被维护在存储器中,从而框架管理器 440 可确定在先前请求值与当前请求值集合之间是否有任何差异。对于向量请求,在先请求通常不被维护在存储器中,尽管如特定实现所希望地,节点 601 的资源可维护这些先前请求。因此,框 1415 对于向量类型的请求是任选的。

[0136] 在框 1420 中,框架管理器 440 计算先前请求值集合与当前请求值集合之间的 Δ 或即差异。在判决框 1425 中,框架管理器确定当前请求值集合是否等同于先前请求值集合。换言之,框架管理器 440 确定当前请求值集合和先前请求值集合之间是否存在差异。如果在请求值的当前集合和先前集合之间没有差异,则跟随“是”分支 (跳过框 1430 到框 1470) 到框 1475,其中该过程结束。

[0137] 如果对判决框 1425 的询问是否定的 (这意味着请求值集合相对于先前请求值集合而言是不同的),则跟随“否”分支到判决框 1430。

[0138] 在判决框 1430 中,框架管理器 440 确定该当前请求是否是异步请求。如果对判决框 1430 的询问是否定的,则跟随“否”分支到框 1440,在其中与客户端请求 675 相对应的资源 601 被框架管理器 440 锁定。如果对判决框 1430 的询问是肯定的 (这意味着该当前请求是异步请求类型),则跟随“是”分支到框 1435,其中该请求可被推送到另一线程上并且可由另一核来执行 (若框架管理器 440 当前管理着如图 1 那样的多核系统)。已经用虚线来解说框 1435 以指示该步骤可以是任选的 (若 PCD 100 是单核中央处理系统)。

[0139] 接着,在框 1440 中,与请求 675 相对应的资源 601 被框架管理器 440 锁定。接着,在框 1445 中,资源 601 执行一般与在图 9 的框 1130 中接收到的资源阵列数据的插件数据相对应的更新功能。该更新功能一般包括负责鉴于新客户端请求的新资源状态的功能。

[0140] 该更新功能将其先前状态与该客户端请求中的所请求状态相比较。如果所请求状态大于先前状态,那么该更新功能将执行该客户端请求。然而,如果所请求状态等于或小于该当前的并且资源正以之来操作的状态,那么该客户端请求将不被执行,以便提高效率,因

为老的状态达成或满足所请求的状态。更新功能从客户端取得新请求并且将其与所有其他活跃请求聚集以确定资源的新状态。

[0141] 作为示例，多个客户端可能正在请求总线时钟频率。总线时钟的更新功能通常将取所有这些客户端请求的最大值并且将其用作该总线时钟的新的期望状态。并非所有资源将使用相同更新功能，尽管有一些更新功能将被多个资源使用。一些共用更新功能是要取诸客户端请求的最大值、取诸客户端请求的最小值、以及对客户端请求求和。或者，诸资源在其资源需要以某种独特的方式来聚集请求的情况下可定义其自己的定制更新功能。

[0142] 接着，在框 1450 中，框架管理器 440 将该数据传递给与客户端 648 相对应的资源，从而该资源可执行专门用于节点 601 的资源的驱动器功能。驱动器功能应用由更新功能所计算出的资源状态。这可能造成需要更新硬件设置、发布对依存资源的请求、调用旧式功能或上述各项的某种组合。

[0143] 在先前示例中，该更新功能计算了所请求的总线时钟频率。驱动器功能可接收该所请求的频率并且其可更新时钟频率控制 HW（硬件）以该频率来运行。注意，有时对于驱动器功能而言，要满足更新功能所已经计算出的确切请求状态是不可能的。在该情形中，该驱动器功能可选取最好程度地满足该请求的频率。例如，总线时钟 HW 可能仅能够以 128 MHz 和 160 MHz 运行，但是所请求状态或许是 150 MHz。在该情况下，驱动器功能应以 160 MHz 运行，因为该频率超过了所请求状态。

[0144] 接着，在框 1455 中，框架管理器 440 从已经在框 1450 中执行了驱动器功能的资源接收状态控制。接着，在框 1460 中，事件 690（若已对该资源定义了该事件）可被触发，从而数据被传递回到与事件 690 相对应的客户端 648。事件可在另一线程中被处理。这可使对所锁定资源花费的时间量最小化并且允许在如图 1 中所解说的多核系统中进行并行操作。

[0145] 可以类似于如何可对该方法 1400 中所述的资源定义请求的方式来对资源定义一个或多个事件 690。换句话说，该事件创建过程可很大程度上与客户端创建过程并行。对于事件而言有所不同的是，要定义仅在越过某些阈值时获触发的事件是可行的。

[0146] 对仅基于阈值来获触发的事件的这种定义允许通知资源何时变得被过度订阅（其具有比其所能支持的更多的并发用户），这指示系统过载状况，或通知资源何时变低 / 消失，这可允许其他事物被关闭，复原在系统变得被过度订阅时被禁用了的功能性，等等。因为可用阈值来进行事件注册，所以将系统在有事件通知之际不得不进行的工作量减少到只在有某事物真有必要时才发生。在每次状态变化之际登记事件也是可行的。

[0147] 接着，在任选框 1465 中，如果正被处理的请求是向量请求，那么通常执行该任选框 1465。任选框 1465 通常包括检查或确定以评价向量指针是否仍然位于用户传递到该向量中的同一数据上。如果对此任选框 1465 的询问是肯定的（这意味着该指针仍正指向由用户传递到该向量中的同一数据），则该指针被清除，从而对老数据的引用不被维持。该任选框 1465 通常被执行以考虑到向量请求（与冲激请求和要求的请求对比而言）正被处理时的以上描述的双重缓冲框 1415。

[0148] 接着，在框 1470 中，框架 440 解锁所请求资源，从而可由特定节点 601 的当前的但如今得到释放的所请求资源来处置其他客户端请求 648。该过程随后返回到第一框 1405 以接收下一客户端请求。

[0149] 上述方法和数据结构基本上如同它们适用于单处理器 PCD 100 一样可适用于多

处理器PCD 100。然而，远程框架300（图3）可以提供可增强多处理器实施例中的操作的附加特征。

[0150] 例如，远程框架300可有利地使得处理器间通信的细节对于应用程序员或类似人员呈透明。因此，例如应用程序可以定义发布对目标资源的请求的客户端，而无需必须在客户端定义中包括控制该资源的处理器域的任何标识。相反，远程框架300确保该请求将到达目标资源，而无论是哪个处理器控制该客户端以及是哪个处理器控制目标资源。

[0151] 另外，远程框架300管理处理器间通信，从而例如，应用程序不需要包括与协议或者处理器之间的通信路径（例如总线）的其它方面有关的任何指令。此外，因为不同的处理器间通信路径可以使用不同的协议，因此远程框架300允许资源定义指定协议连同资源的其它方面。以下关于图13描述了涉及分布式资源管理的这些以及其它特征。

[0152] 图13解说了其中受第一处理器（未示出）控制的第一资源1302用作对应于受第二处理器（未示出）控制的第二资源1304的分布式或远程资源的示例或实例。术语“分布式资源”或者“远程资源”在本公开中被用于指代一个处理器上的、与另一处理器上的“原生”资源相对应的资源。这一示例中的第二资源1304用作对于第二处理器而言的原生资源。

[0153] 分布式资源被用作访问对应的原生资源的手段。在这一示例中，术语“资源”可与术语“节点”互换地使用，因为应理解，资源可包括在节点内。

[0154] 断线1301解说了受第一处理器控制的资源（线1301左边）和受第二处理器控制的资源（线1301右边）之间的划分。第一资源1302是受第一处理器控制的两个或更多个资源之一。一个此种资源可以是第一资源1302所依存于的协议资源1306。类似地，第二资源1304是受第二处理器控制的两个或更多个资源之一。一个此种资源可以是第二资源1304所依存于的协议资源1308。

[0155] 第一和第二资源1302和1306还可用以上一般化地关于资源或节点描述的相同方式依存于附加资源，但是出于清楚目的此类附加资源并未在图13中示出。注意，受第一处理器控制的资源由第一资源图（即，有向无环图）来定义，并且受第二处理器控制的资源由不与第一资源图共享任何资源的第二此种资源图来定义。

[0156] 在其各自相应处理器的控制下，第一和第二资源1302和1304能够经由通信路径1303来进行信息通信。通信路径1303表示第一和第二处理器之间的物理介质以及用于经由该介质来通信的传输协议的一个或多个层的组合。因此，第一资源1302和第二资源1304之间的任何通信均必须符合这些协议。协议资源1306和1308定义协议或者可以指向库（未示出）中的协议定义。远程框架300和（主）框架440彼此协同操作以管理这些资源和其间的通信。如下所述，在第一处理器的控制下，客户端1312可以发出对第一资源1302的一个或多个资源请求。第一资源1302使用对应的第二资源1304的功能性以服务该资源请求。

[0157] 在不同的应用状态中，可能需要或希望处理器请求不同的资源配置或资源状态。例如，总线资源可控制总线时钟的速度。在一个应用状态中，处理器可请求允许该处理器以例如1亿条指令每秒（MIPS）的速率来操作的总线时钟，而在另一应用状态中，处理器可请求允许其以例如1.5亿条指令每秒（150 MIPS）的速率来操作的总线时钟。在处理器准备要进入为休眠状态的应用状态的情形中，该处理器可请求零MIPS的总线时钟。

[0158] 类似地，在由正执行第一应用程序的处理器定义的一个应用状态中，处理器可请求 100 MIPS，而在由正执行第二应用程序的该处理器定义的另一应用状态中，该处理器可请求 150 MIPS。同样地，在由正并发地执行特定数目个应用程序的处理器定义的一个应用状态中，该处理器可请求 100 MIPS，而在由正并发地执行不同数目个应用程序的该处理器定义的第二应用状态中，该处理器可请求 150 MIPS。应理解，上文引述的总线时钟仅旨在作为可由发出布源请求的处理器配置的资源的示例，并且数字“100”和“150”旨在作为处理速度的任意性示例。

[0159] 资源配置或状态可被编组成资源状态集。资源状态集定义了由处理器在某一处理器应用状态中一并使用的一个或多个资源的配置或状态。例如，某一资源状态集可包括由总线时钟资源向处理器提供某个数目的 MIPS 的处理速度的配置或状态信息，以及由解码器（即，资源的另一示例）向处理器提供解码功能的配置或状态信息。

[0160] 图 14 是解说形成系统 103 的控制器 101、资源功率管理器 157、主处理器 110、126、低级驱动器 103、共享资源 105A-C、以及本地资源 105D-H 之间的关系的功能性框图。图 14 还解说了触摸屏 132 可如何耦合至触摸屏驱动器 / 控制器 130。触摸屏驱动器 / 控制器 130 可耦合至第一主处理器 110A 的时钟码 113A。

[0161] 系统 103 可按照使资源等待时间最小化的方式在由处理器 110 所想要的资源状态集之间切换。术语“资源等待时间”指的是在主处理器 110、126 开始使控制器 101 和资源功率管理器 157 准备要转变到另一资源状态集的时间与该资源集的资源变成被配置成指定状态并准备好被该处理器使用的时间之间发生的延迟或等待时间。如下所述，资源状态集可被宽泛地分类成：活跃资源状态集，其中向处理器提供被配置成协助该处理器执行应用程序以及以其他方式提供处理能力的资源；以及休眠资源状态，其中仅向处理器提供协助该处理器维持休眠状态（即，其中处理器不在执行应用程序或以其他方式提供处理能力的状态）的资源，。尽管处于休眠状态的处理器可维持低级功能，但该处理器不执行软件，即如本领域普通技术人员所能理解的应用程序。应该理解，下文描述的“下一活跃状态”特征可被应用于任何资源状态集之间的转变，而不管它们可能是活跃集还是休眠集。

[0162] 在图 14 中示出的示例性实施例中，第一主处理器 110A 可耦合至资源功率管理器 157 和控制器 101。控制器 101 可耦合至第一主处理器 110A 的时钟码 113A。控制器 101 可包括一个或多个低级驱动器 103。这一个或多个低级驱动器 103 可负责与一个或多个共享资源 105A-C 通信。共享资源 105A-C 可包括支持主处理器 110 的任务或功能的任何类型的设备。共享资源 105A-C 可包括诸如其他处理器的时钟之类的设备以及如图形处理器、解码器等单功能元件、以及类似资源。

[0163] 共享资源 105A-C 可耦合至一个或多个本地资源 105D-H。这一个或多个本地资源 105D-H 可与共享资源 105A-C 类似，因为它们可包括支持或协助主处理器 110 的任务或功能的任何类型的设备。本地资源 105D-H 可包括诸如其他处理器的时钟之类的设备以及如图形处理器、解码器等单功能元件及类似资源。本地资源 105D-H 可包括叶节点。叶节点被本领域普通技术人员理解为通常不引用或包括其他从属资源 105 的本地资源 105D-H。

[0164] 控制器 101 可负责管理从这一个或多个主处理器 110、126 发布的请求。例如，控制器 101 可管理源自第一主处理器 110A 的请求。第一主处理器 110A 可响应于操作者操纵触摸屏 132 而发布这一请求。触摸屏 132 可向触摸屏驱动器 / 控制器 130 发布信号。触摸

屏驱动器 / 控制器 130 可进而向第一主处理器 110A 的时钟码 113A 发布信号。

[0165] 控制器 101 还可负责管理特定处理器 110 的休眠状态。在进入休眠状态之前，处理器 110 将提供用于管理休眠状态的信息。用于管理休眠状态的信息包括休眠状态进入和休眠状态退出。这一用于管理休眠状态的信息将在下文被称为触发和资源状态。资源状态集可包括用于按照支持处理器的休眠状态的方式来配置一个或多个资源的资源信息。

[0166] 触发可定义致使处理器 110 或进入休眠状态或离开休眠状态的事件。触发将一般地引用包含在控制器 101 内或可由控制器 101 访问的资源状态。资源状态定义了特定处理器 110 所需的资源 105 所想要的状态。在示例性实施例中，每个处理器 110 可以向控制器 101 提供至少两个资源状态集 :活跃资源状态集和休眠资源状态集。

[0167] 然而，在其他实施例中，处理器 110 可提供作为单一活跃集或单一休眠集的补充的资源状态集，或者与单一活跃集和单一休眠集不同的资源状态集。这样的其他资源状态集可与上述的处理器应用状态中的一个或多个相对应。即，对于任何应用状态，处理器可提供相应的资源状态集。

[0168] 在示例性实施例中，活跃资源状态集可定义当处理器 110 在活跃地执行处理功能并且向其资源 105 要求动作 / 功能时资源 105 的状态。休眠资源状态集可定义当处理器 110 处于休眠或空闲状态时资源 105 的状态。下文将结合图 15 来描述关于触发和资源状态的进一步细节。

[0169] 图 15 是解说关于控制器 101、资源集 304 和触发集 314 的细节的功能框图。如先前所述的，控制器 101 可包括由 PCD 100 的处理器 110、126 中的一个或多个执行的软件。控制器 101 可将信息存储在存储器 112 中或存储在控制器 101 内的区域中，诸如本地存储，如本领域普通技术人员所能理解的。这一信息可包括资源表 302，资源表 302 包括被指派给由控制器 101 服务的每一主处理器 110 的资源集 304。这一信息还可包括触发集 314，触发集 314 也被指派给每一主处理器 110 并且其对于每一主处理器 110 而言可以是唯一性的。

[0170] 每一资源集 304 一般包括与由特定主处理器 110 所想要的资源 105 的状态有关的信息。指派给特定主处理器 110 的每一资源集 304 可包括活跃资源集 306 和休眠资源集 308。活跃资源集 306 可定义或描述当特定主处理器 110 是活跃的或正常运作的情况下资源 105 的状态。休眠资源集 308 可定义或描述在特定主处理器处于休眠或休止的状态时资源 105 的状态，如本领域普通技术人员所能理解的。每一资源集 304 还可包括附加集，诸如在图 15 所示的示例性实施例中被指派给第一主处理器 110 的“集合 1”和“集合 2”。

[0171] 作为示例，如图 15 中所解说的第一主处理器 (A)110A 的活跃资源集 306 已将以下值赋给其资源 105 中的每一个 :给第一共享资源 (SR#1)105A 的值为 1 ;给第二共享资源 (SR#2)105B 的值为 1 ;给第 N 共享资源 (SR#N)105C 的值为 1 ;而给第一本地资源 (LR#1)105D 的四个值为 1、0、1 和 1。

[0172] 如前所述，资源 105 的状态不限于单一的值，并且可包括多个值。此外，资源的状态可包括数个不同类型的参数中的任何类型的参数。例如，状态可指定数百兆赫作为可充当资源 105 的特定时钟的时钟速度的数量。

[0173] 作为另一示例，如图 15 所解说的第一主处理器 (A)110A 的休眠资源集 308A 将以下值赋给其资源 105 中的每一个 :对于第一共享资源 (SR#1)105A，这一资源已被赋值为 0 ;第二共享资源 (SR#2)105B 具有 0 的赋值 ;而第 N 共享资源 (SR#N)105C 具有 0 的赋值。第

一本地资源 (LR#1)105D 可具有 0、1、0 和 0 的赋值。

[0174] 指派给特定主处理器 110 的每一触发集 314 可包括至少三个字段 :中断字段 316 ;“来自集合”318 ;以及“前往集合”320。触发集 314 的这三个字段中的每一个还可包括对应的三列的集合 :触发起始列 322 ;清除列 324 ;以及计时器列 326。

[0175] 中断字段 316 描述可由资源功率管理器 157 生成和 / 或检测的动作或活动。中断字段 316 可一般地被表征为可允许控制器 101 基于由 RPM 157 检测到的触发事件来选择特定处理器 110 所想要的特定资源集 304 的“触发事件”。控制器 101 对资源集 304 的选择可避免在上文背景部分中描述的耗时的软件握手。

[0176] 回顾图 15 的针对第一主处理器 (A)110A 的第一触发集 (触发集 #1),按照列的次序来讨论该集合的各个字段。从触发集 314A 的第一列开始,触发起始列 322 在其第一行中具有与中断字段 316 相对应的列为“解码中断”的动作。

[0177] 如前所述,中断字段 316 可定义致使控制器 101 响应于检测到触发起始字段 322 而激活资源集 304 的状态的参数。在图 15 所解说的示例性实施例中,中断字段 316A 已经被定义或描述为“解码中断”,这意味着当资源功率管理器 110 检测到“解码中断”时,诸如当 PCD 100 正在解码视频时,则这一事件可警告控制器 101 要查看“触发起始”列下第一列 322A1 中的“来自集合”字段 318。

[0178] “来自集合”字段 318 可包括表明对于正被控制器 101 查看的特定主处理器 110 而言当前资源集 304 应该是什么的值。字段 318 可将资源集 304 按照其标识符来列出,诸如“活跃集”、“休眠集”,或者按集合编号来列出,如“集合 1”或“集合 2”。字段 320 还可包括例如星号之类的“通配符”。

[0179] “来自集合”字段 318 中的通配符指定可致使控制器 101 检索上一所知被特定主处理器 101 使用的活跃资源集 304。在图 15 所解说的示例性实施例中,“来自集合”行 318A 和触发起始列 322A1 具有星号或通配符的值。

[0180] “前往集合”320 如同“来自集合”318 那样可包括资源集 304 的按照其标识符的列表,诸如“活跃集”、“休眠集”,或按集合编号的列表,如“集合 1”或“集合 2”。字段 320 还可包括如星号之类的意指上一被处理器 110 使用的资源集 304 的“通配符”。在图 15 所解说的示例性实施例中,“前往集合”字段 320A 和触发起始字段列 322A1 具有值“集合 1”,其是第一资源集 304A 的列 310A 中列出的资源集 1。

[0181] 对于图 15 所解说的示例,当解码中断事件被 RPM 157 检测到时,它警告控制器 101。控制器 101 查看第一主处理器 110 的第一触发集。因为触发起始列 322A1 列出了匹配值 (解码中断),所以控制器 101 查看“来自集合”字段 318A 并且确定该值是通配符值或星号。控制器 101 随后查看“前往”字段 320A,其具有指定特定资源集 304A 的值“集合 1”。基于控制器 101 所查看到的这一信息,控制器 101 将把第一主处理器 110A 的当前资源集 304A 从其当前集合切换至资源集“集合 1”。资源集 1 被列在指派给第一主处理器 110A 的资源集 304A 的列 310A 中。

[0182] 此外,当 RPM 157 或控制器 101 检测到诸如在第一触发集的清除列 324A1 中所解说的“不解码”事件时,则控制器 101 随后将查看“来自集合”字段 318A 并确定该值包括“集合 1”。控制器 101 随后将查看“前往集合”字段 320,其在该示例中具有为通配符或星号的值。这意味着控制器 101 将把第一主处理器 110A 的资源集 304A 从“集合 1”资源集切换成

处理器 110A 使用的上一活跃资源集。

[0183] 触发集的计时器字段 326 可表明控制器 101 可使用特定资源集 304 的时间量。因此对于图 15 所解说的示例性实施例，对于第一触发集的计时器字段 326A1，该字段具有 3 毫秒的值。这意味着，当解码中断事件与第一触发集的触发起始字段 322A1 匹配时，则控制器 101 只使用“前往集合”字段 320A 中指定的资源集 304 达 3 毫秒的时间段。在其他示例性实施例中，可能发生或存在的情况是 :计时器字段 326 中没有信息，或者值被定义成与指示不存在针对这一转变的计时器触发 326 以及该转变仅适用于“不解码”字段的值相对应。在诸如图 15 所解说的其中定义了计时器字段——计时器字段 326A1 和 326A2 的情形中，则在计时器字段 326 和清除字段 324 之间无论首先发生哪个事件，通常都将发起该转变。

[0184] 图 16 解说了处理器 110 的示例性活跃 - 休眠触发集 314。在这一示例性实施例中，第一列 322 中的中断字段 316 定义了“关闭”事件作为要对特定处理器 110 发起休眠集 308(图 15) 的动作。“关闭”事件可包括如操作者选择用于关闭 PCD 100 的开 / 关按钮之类的动作。

[0185] 在图 16 中的示例性实施例中，当检测到“关闭”事件时，控制器 101 将当前活跃资源集 306 转变到休眠集 308。休眠集 308 被列在图 15 中的表 302 的主资源集 304 中。

[0186] 当控制器 101 从 RPM 157 接收到已发生了“引起”事件 (诸如 PCD 100 的操作者发起的上电事件) 的消息时，则控制器将基于触发集 314 的“前往集合”字段 320 中列出的通配符或星号值将处理器 110 从其休眠集 308 转变到上一活跃资源集 304。

[0187] 如上所述，系统 103 不限于活跃集和休眠集 306、308。系统 103 可被用于针对除了如图 15 所解说的进入休眠状态或退出休眠状态之外的其他事件而在资源集 304 之间切换。

[0188] 图 17 是解说用于管理将处理器 110 置于休眠状态的触发集 314 的方法 1700 的逻辑流程图。框 1705 是方法 1700 的第一步骤。在框 1705 中，每一处理器 110 可基于来自 PCD 100 的先前使用情形的数据按需更新其资源集 304 以及其在控制器 101(图 1-2) 中的触发集 314。

[0189] 在框 1710 中，处理器 110 可请求 RPM 157(图 14) 生成给控制器 101 的关闭信号。在框 1715 中，RPM 157 可将关闭信号发送至控制器 101。

[0190] 控制器 101 可在框 1720 中接收关闭信号，并且激活触发集 314，其中触发集 314 可如图 16 所解说地被指派给关闭事件。在图 16 所解说的示例性实施例中，将关闭信号对照着触发集 314 的中断字段 316 来进行匹配。触发集 314 引导控制器 101 访问如在“前往集合”字段 320 中所指示的休眠集 308。在框 1725 中，控制器 101 可立即将确认信号发送至 RPM 157，而控制器 101 继续激活匹配于此关闭信号事件的触发集 314 所引用的资源集 304。

[0191] 在框 1730 中，对于每一匹配的触发集 314，诸如在图 16 所解说的对应中断字段 316 中列出“关闭”事件的匹配触发集 314，控制器 101 可将当前资源集 304 切换到休眠集 308，诸如图 15 的主处理器 110A 的第一资源集 305A 的休眠集 308A。

[0192] 接着，在框 1735 中，控制器 101 可向低级驱动器 103 发布休眠请求状态，如图 14 中所解说的。低级驱动器 103 可将所请求的状态传递至对应的资源 105。

[0193] 在框 1740 中，每一资源 105 可向控制器 101 和 RPM 157 发出关闭信号确认。方法 1700 随后可结束。

[0194] 图 18 是解说用于管理将处理器 110 从休眠状态置于活跃状态的触发集 314 的方

法 1800 的逻辑流程图。框 1205 是方法 1800 的第一步骤。在框 1805 中，用 RPM 157 检测到苏醒条件或苏醒事件，或者直接由控制器 101 检测到苏醒事件，其中控制器 101 可具有其自己的中断控制器（未解说）。示例性实施例可被设计成使得苏醒中断可以是 RPM 157 不能检测到的。在这样的示例性实施例中，控制器 101 可使用其中断控制器来检测它们，并且使它们“匹配”于主处理器 110 的休眠集要求。

[0195] 接着，在框 1810 中，RPM 157 可将苏醒信号发送给控制器 101。在框 1815 中，控制器 101 可从 RPM 157 接收苏醒信号，并且激活匹配于该苏醒信号的一个或多个触发集 314。例如，控制器 101 可将苏醒信号与图 16 的触发集 314 的“活跃”列中的中断字段 316 中列出的“引起”事件相匹配。在图 16 的示例性实施例中，活跃列 324 中的“前往字段”320 将控制器引导至被当前处理器 110 使用过的上一资源集 304。

[0196] 因此在框 1820 中，控制器 101 将基于这一匹配的触发集 314 来改变处理器 110 的当前资源集 304。本领域普通技术人员将明白，控制器 101 将在在如图 15 所示的其维护的所有触发集之间循环。

[0197] 接着，在框 1825 中，控制器 101 可向 RPM 157 发送标识哪些主处理器 110 已经从休眠状态中被唤醒的苏醒确认。接着，在框 1830 中，具有匹配的苏醒触发集 314 的每一处理器 110 从休眠状态被释放，并且凭借 RPM 157 的供电被恢复到其活跃状态。方法 1800 随后结束。

[0198] 图 19-20 解说了在本说明书中被称为“下一活跃资源状态集”或“下一活跃集”的另一特征。下一活跃集的一个示例是下一苏醒集。下一苏醒集或其他下一活跃集可按照上文参考图 18 和控制器 101 在苏醒事件发生之际切换至的资源集 304 所述的相同方式来使用。

[0199] 图 19 与图 15 类似之处在于它表示存储在控制器 101 中的信息。在示例性实施例中，控制器 101 可包括三个存储器缓冲，在本说明书中出于方便的考虑被称为“A”存储器缓冲 702、“B”存储器缓冲 704 和“C”存储器缓冲 706。

[0200] 图 20 是与图 17 类似的逻辑流程图，在于其解说了一种用于将处理器置于休眠状态的方法 800。框 2005 是方法 800 的第一步骤，并且与上文参照图 17 所描述的框 1705 类似。框 2005 指示，处理器 110 不仅可更新活跃或苏醒资源状态集和休眠资源状态集，还可更新下一苏醒资源状态集。如图 20 所示，处理器可致使活跃集被存储在控制器 101 的“A”缓冲 702（图 19）中，休眠集被存储在控制器 101 的“B”缓冲 704（图 19）中，而下一苏醒集被存储在控制器 101 的“C”缓冲 706（图 19）中。框 2005 的其他方面与上文参照框 1705 所描述的相同，并因而此处不再描述。

[0201] 框 2010、2015、2020、2025、2030、2035 和 2040 分别与图 17 的框 1710、1715、1720、1725、1730、1735 和 1740 相同，并因此在此处不再描述。注意，当处理器开始关闭时，它处于与“A”缓冲 702（图 19）中存储的苏醒集相对应的苏醒应用状态。处理器随后按照上文参考图 17 所描述的相同方式进入与存储在“B”缓冲 704（图 19）中的休眠集相对应的休眠应用状态。处理器从休眠应用状态苏醒（图 18）为与存储在“C”缓冲 706（图 19）中的下一苏醒集相对应的下一苏醒应用状态。通过将下一苏醒集更新预先存储在“C”缓冲 706 中并且尽快地应用它们，控制器 101 就能够在苏醒事件发生之际立即开始配置由该下一苏醒集指定的资源，从而有助于使资源等待时间最小化。

[0202] 图 21 是解说节点调度器 2101、资源功率管理器（“RPM”）157、以及其它节点架构系统元件之间的关系的功能框图。具体地，调度器 2102 被耦合至调度器数据库 2103；资源功率管理器 157；CPU 繁忙监视器（“CBM”）资源 2109；休眠低功率资源 2107；定时器 2105；图 3 的框架管理器 440；以及各种资源 105A-N。以上已经结合图 14-15 描述了 RPM 157。RPM 157 与控制器 101（参见图 15）一起工作以维护和监视包括用于维护 PCD 100 的各种处理器 110 的状态的活跃和休眠集的资源集 304。

[0203] 以上结合图 3-6 和 14-16 描述了资源 105A-105N。资源 105 可包括硬件和 / 或软件。资源 105 可以接收由调度器 2101 管理的请求 675。

[0204] 调度器 2101 是如以上结合图 3 描述的框架的一部分。调度器 2101 被耦合至框架管理器 440。调度器 2101 提供指定由客户端 648 发布的以资源 105 为目的地的请求 675（参见图 3-5，其解说了以上结合框架管理器 440 描述的客户端 648 和请求 675 的细节）基于该请求中指定的定时截止期限何时将被应用的排队特征。调度器 2101 的定时由定时器 2105 来跟踪。

[0205] 在由客户端 648 作出经调度的请求 675 之后，则调用方应用或实体可随后继续其自己的处理。调度器 2101 将随后基于由客户端 648 提供的截止期限来确定执行请求 675 的最佳方式。

[0206] 调度器 2101 查看每个经调度请求 675 并确定该经调度请求 675 将花费多长时间来执行。调度器 2101 还执行退避计算。调度器 2101 与定时器 2105 和 / 或与处理器 110 的休眠循环一起工作以理解哪些经调度请求 675 可被应用，以及它们何时应当发生以满足由客户端 648 或调用方实体规定的所请求截止期限。

[0207] 定时器 2105 可包括时钟，该时钟包括以 T 滴答时间来运行的 32 位定时器，如由本领域普通技术人员所理解的。可采用其它时钟而不会脱离本公开。例如，可以使用 64 位定时器系统，如由本领域普通技术人员所理解的。对于 32 位定时器，调度器 2101 可通过查看由定时器 2105 跟踪的当前时间与在经调度请求 675 中规定的所请求时间之差来确定请求 675 的时间是在过去还是将来。

[0208] 为了用 32 位定时器系统来检测迟到请求 675/ 在遥远的将来的请求，期望时间必须小于时间分辨率的一半，如果该差小于 UINT_MAX(0 X 8000 0000)，则其被调度器 2101 认为是在将来的。如果所请求时间大于或等于 UINT_MAX，则此所请求时间被调度器 2101 认为是发生在过去并且调度器 101 通常立即处理此种请求 675。

[0209] 以上描述的由调度器 2101 确定的定时被调度器 2101 存储在调度器数据库 2103 中。调度器数据库 2103 维护与请求 675 可被执行的各种方式以及当请求 675 按特定顺序发生时它们所需要的相应定时有关的信息。换句话说，调度器 2101 可跟踪各种选项及其由调度器 2101 根据不同情景服务于请求 675 所会花费的相应时间并将这些值存储到调度器数据库 2103 中。以下将结合图 22 来描述关于调度器数据库 2103 的进一步细节。

[0210] 休眠低功率资源（“LPR”）2107 通过查看数据库 2103 来确定在系统 100 从休眠状态苏醒时需要触发什么资源 105。休眠 LPR 2107 可被表征为应用编程接口 (API)。休眠 LPR 2107 将发布它确定将出现在由控制器 101 和 RPM 157 管理的资源集 304 的下一苏醒集中的那些请求 675。

[0211] CPU 繁忙监视器（“CBM”）资源 2109 由调度器 2101 建立。CBM 资源 2100 跟踪服

务由调度器 675 管理的请求 675 的所有 CPU 110 的状态。对 CBM 资源 2109 的任何非零请求 675 指示发布请求 675 的客户端 648 或用户线程 2301(参见图 23)预期 CPU 110 在请求 675 的执行期间保持繁忙。CBM 资源 2109 是二进制资源,这意味着只要有任何客户端 648 处于繁忙,CPU 110 就被 CMB 资源 2109 认为是繁忙的,并且 CPU 110 直至最后的客户端 675 完成非零或繁忙请求才被表征为自由或非繁忙。此外,关于调度器 2101 和 CBM 资源 2109 之间的操作和关系的进一步细节将在以下结合图 29 来描述。

[0212] 图 22 是解说调度器数据库 2103 的示例性内容的示图。如以上结合图 21 所述,调度器数据库 2103 维护与请求 675 可被执行的各种方式以及当请求 675 按特定顺序发生时它们需要的相应定时有关的信息。换句话说,调度器 2101 可跟踪各种选项及其由调度器 2101 根据不同情景服务于请求 675 所会花费的相应时间并将这些值存储到调度器数据库 2103 中。

[0213] 调度器数据库 2103 可包括但不限于以下信息:线程配置数据 2205;调度器等待时间 2210;最小调度器 Δ2215;请求等待时间 2220;分叉前瞻 Δ2225;分叉等待时间 2230;交汇等待时间 2235;休眠苏醒转换等待时间 2245;时间队列等待时间;低功率资源(“LPR”)进入等待时间 2250;LPR 退出等待时间 2255;LPR 现在 Δ2260;经调度链表 2270;请求状态 2275;请求时间 2280;启动时间 2285;迟到概率 2287;回调通知 2290;最新通知状态 2293;以及请求迟滞 2295。

[0214] 线程配置数据 2205 可跟踪来自源自客户端 648 和请求 675 的线程的各种信息。调度器等待时间 2210 可以按定时器 2105 测得的滴答来衡量处置调度器代码所要花费的时间。最小调度器 Δ2215 可以衡量一请求 675 因其在太遥远的将来以致于不能被当前超时所处置而将被重新调度的时间。

[0215] 请求等待时间 2220 可以衡量在若对资源 105 发布了经调度请求 675 的情况下所使用的默认经调度请求等待时间。请求等待时间 2220 可包括从发布完全异步请求 675 到完成的预期时间。分叉前瞻 Δ2225 可以衡量要添加至休眠苏醒时间的时间,以确定什么请求 675 可有资格被分叉(fork)。这一 Δ2225 可被用于对休眠集在从休眠状态苏醒时可能必须执行的工作进行补偿。

[0216] 分叉等待时间 2230 是默认经调度分叉等待时间,其可在对资源 105 发布了经调度请求但是该资源 105 不支持分叉请求等待时间查询的情况下使用。分叉等待时间 2230 包括从发布经分叉请求 675 到其返回时的时间。对分叉等待时间 2230 的请求仅对于可分叉的资源 105 是有效的。

[0217] 交汇等待时间 2235 可包括分叉操作完成直至资源 105 被交汇并且经分叉请求退役时的时间。交汇等待时间 2235 还是默认经调度分叉等待时间,其可在对资源 105 发出来经调度请求 675 但是该资源 105 不支持交汇等待时间查询的情况下被使用。

[0218] 休眠苏醒转换等待时间 2240 衡量休眠集退出休眠状态所花费的时间,定时器 2105 激发所需的时间、以及当多个请求 675 被交汇时所需的任何时间。时间队列等待时间 2245 可以衡量定时器 2105 调用定时器功能所花费的时间。

[0219] 低功率资源(“LPR”)进入等待时间 2250 可以包括在使用 LPR 进入功能时用于开销的默认时间。LPR 退出等待时间 2255 可以包括在使用 LPR 退出功能时用于开销的默认时间。LPR 现在 Δ2260 可以包括 LPR 进入功能重新调度定时器 2105 的默认时间。

[0220] 调度器数据库 2103 的其余数据可被表征为请求定义 2265，如图 22 中所解说的。经调度链表 2270 可以跟踪正等待运行的客户端 648。请求状态 2275 可以跟踪各种请求 675 的状态，诸如等待、处理、被分叉、被交汇等。请求时间 2280 可以包括在其中请求 675 必须被完成的时间。启动时间 2285 可以跟踪启动从客户端 648 发布的请求 675 所需的时间。

[0221] 迟到概率参数 2287 在是时候由调度器 2101 调度请求 675 时向调度器 2101 提供灵活性。如果迟到概率参数 2287 被设为等于 0，则这意味着请求 675 可能不能容忍关于由调度器 2101 对其进行调度的任何迟滞。如果其被设为等于 1，则调度器 2101 可允许请求 675 迟到——如果此种迟滞将帮助调度器 2101 处理其当前的请求 675 集合的话。调度器 2101 还可以跟踪 0 和 1 之间的值，作为可接受的迟滞率。因此，0 和 1 之间的值可被设置，从而客户端将接受有一半请求是迟到的。0 和 1 之间的另一值可被设置，从而可接受所有请求都是迟到的。依此类推。

[0222] 回调通知 2290 跟踪何时已经完成对发布了请求 675 的客户端 648 的回调。最新通知状态 2293 跟踪回调通知是已经按时发布；已经推迟地发布；或者该请求已撤销调度。并且请求迟滞 2295 可以跟踪请求 675 是并未迟到的时间，请求 675 迟到了的时间，和 / 或迟到请求 675 结束与它被请求之间的时间总和。调度器数据库 2103 并不限于这些参数。调度器数据库 2103 可以跟踪其它参数，如由本领域普通技术人员所理解的。

[0223] 现在参照图 23，这一附图解说了演示客户端 648、客户端请求 675、调度器 2101 以及定时器 2105 之间的关系的示例性时序图 2300。图 23 的 Y 轴对应于时间。关于客户端 648 和请求 675 的基线讨论请参见之前图 3-6 的讨论。

[0224] 由用户线程 2301 建立的客户端 648 想要使对资源 105（未解说）的经调度请求 675 发生在时间 X 的特定请求截止期限处。用户线程 2301 在客户端 648 已经在框架 300 内被创建后发布该请求 675。

[0225] 请求 675 可包括任何类型的请求，诸如但不限于，向量请求、标量请求、或者其它任何标准同步请求，如先前所述。调度器 2101 将来自请求 675 的信息置于数据库 2103 中。来自请求 675 的该信息可包括但不限于，以上在图 22 中结合数据库 2103 描述的信息。

[0226] 具体地，调度器 2101 确定应在何时执行每个请求 675，从而所有请求 675 截至其各自相应的截止期限之前被完成或执行。调度器 2101 还可为每个请求 675 计算退避。调度器 2101 在数据库 2103 中维护该信息。如先前所述，数据库 2103 维护可与请求 675 可被执行的各种方式以及当请求 675 按特定顺序发生时它们需要的相应定时有关的信息。换句话说，调度器 2101 可跟踪各种选项及其由调度器 2101 根据不同情景来服务于请求 675 所会花费的相应时间并将这些值存储到调度器数据库 2103 中。

[0227] 在此之后，在客户端框 675 处，用户线程 2301 或者请求方实体可以继续其自身对其它数据或功能的处理。调度器 2101 确定这一个或多个资源 105 必须被唤醒以处理一个或多个请求 675 的时间，并且随后调度器 2101 在阶段 2305A 向定时器 2015 发布定时命令。

[0228] 在阶段 2310，定时器 2015 期满或命中由调度器 2101 规定的时间并向调度器 2101 发布信号。在阶段 2310，调度器 2101 查询数据库 2103 并确定其需要处理的一个或多个请求 675。尽管调度器 2101 正执行数据库 2103 中列出的针对特定阶段 2310 的请求 675，但调度器 2101 正测量并跟踪各请求 675 是否正被准时处理或者它们是否正推迟地运行。在这一阶段 2310，调度器 2101 还确定各请求 675 完成的相应历时并将这一信息记录到数据库

2103 中。

[0229] 在阶段 2310 处调度器 2101 还可以确定要延迟某些请求 675 的处理或执行，并且调度器 2101 可随后向定时器 2105 发布附加定时命令（“设立期望定时器事件”），从而如在数据库 2103 中列出的这些其它请求 675 可在稍后时间发生。

[0230] 调度器 2101 将尝试兑现或执行数据库 2103 中的所有请求 675，即使调度器 2101 需要比其被请求的截止期限（如由客户端 648 根据用户线程 2301 规定的）更晚地运行某些请求 675。

[0231] 当调度器 2101 确定资源 105 完成了请求 675A，则它生成客户端通知或者回调 2315A。在这一客户端通知 2315A 中，调度器 2101 将通知客户端 648 请求 675A 是已经准时执行（截至所请求的截止期限）还是它是在所请求的截止期限之后被执行的。

[0232] 每个请求 675 还可包括附加信息，诸如迟滞概率 2287，如以上结合图 22 所描述的。这一迟滞概率参数 2287 可在是时候由调度器 2101 调度请求 675 时向调度器 2101 提供灵活性。

[0233] 如果迟滞概率参数 2287 被设为等于 0，则这意味着请求 675 可能不能容忍关于由调度器 2101 对其进行的调度的任何迟滞。同时，如果迟滞概率参数 2287 被设为等于 0xffffffff 中的 0x8000 0000(50％)，则这意味着请求 675 可以容忍请求 675 被调度器 2101 处理每 100 次有 50 次迟滞 (1％)。迟滞概率参数 2287 允许调度器 2101 以一定灵活性来调度请求 675。

[0234] 例如，设想一请求 675 可能要花费至少 10 毫秒 (ms) 才能由资源 105 完成。在一些情景中，该请求 675 在 99％的时间里可能只要花费 1ms 就能由一资源 105 完成。如果针对这一请求 675 的迟滞概率参数被设为等于 0，则调度器 2101 必须将该请求调度在离其请求的截止期限至少 10 秒开外，从而请求 675 能准时完成。

[0235] 然而，如果迟滞概率参数 2287 被设为 0x2900000（意指 1％），则调度器 2101 可以自行斟酌以将此请求 675 调度在离其请求的截止期限 1 秒开外，从而如果其历时仅持续 1ms 则请求 675 可以准时完成。但是如果其历时超过 1ms（诸如如上所述的最差情形 10ms 场景），则请求 675 可能完成得比其所请求的截止期限晚得多。

[0236] 调度器 2103 还可为每个请求 675 维护该最差情形值。该最差情形值通常包括请求启动时间 2285（参见以上结合图 22 讨论的数据库参数）的最保守估计，从而请求 675 截至规定截止期限前完成。如先前所提及的，每个请求 675 具有由客户端 648 经由用户线程 2301 规定的截止期限。作为默认，当迟滞概率参数 2287 并未用请求 675 中的值来规定或置备时，则调度器 101 将始终对请求 675 使用最差情形值。

[0237] 现在参照图 24，这一附图解说了演示客户端 648、客户端请求 675、调度器 2101、定时器 2105 以及跟踪休眠集的控制器 101 之间的关系的示例性时序图 2400。图 24 的 Y 轴对应于时间。关于客户端 648 和请求 675 的描述请参见之前图 3-6 的讨论。还参见图 15，解说了如先前以上所述地跟踪休眠集的控制器 101 的进一步细节。

[0238] 客户端 648 经由用户线程 2301 想要作出对资源 105（未解说）的请求 675。客户端 648 想要请求 675 发生在规定时间处。请求 675 可包括任何类型的请求 198，诸如但不限于，向量请求、标量请求、或者其它任何标准同步请求。

[0239] 调度器 2101 将来自请求 675 的信息置于数据库 2103 中。如上所提及地，调度器

2101 推测应在何时执行每个请求 675，从而所有提交的请求 675 截至其相应截止期限被完成或执行。调度器 2101 在这一阶段计算退避。

[0240] 在此之后，在请求框 675 处，用户线程 2301 或者请求方实体可以继续其自身对其它数据或功能的处理。调度器 2101 确定这一个或多个资源 105（未解说）必须被唤醒以处理请求 675 的时间，并且随后调度器 101 在阶段 2305A 向定时器 860 发布定时命令。

[0241] 休眠事件被调度为在阶段 2427 处发生。在阶段 2410，控制器 101（或查阅控制器 101 的资源状态 304 的调度器 2101）将在阶段 2427 处的休眠事件或休眠循环以及在阶段 2430 处的其预计苏醒时间存储在数据库 2103 中。在阶段 2415，调度器 2101 可将资源集 304 的下一苏醒集（参见图 15）传达给资源功率管理器（“RPM”）157，从而 RPM 157 将知晓一旦已经在阶段 2427 完成休眠循环，应该使用哪个苏醒集。

[0242] 调度器 2101 可包括现在应该使之成为资源集 304 的下一苏醒集的一部分的一个或多个请求 675，从而当系统在阶段 2430 走出休眠状态时这些请求 675 得到执行。在其中这些下一苏醒集被传达给 RPM 157 的阶段 2415 处的 RPM 通信之后，在阶段 2420，调度器 2101 发布命令“处置请求”675。该命令处置请求 675 使 RPM 157 包括这一阶段的任何后续请求 675 以成为由控制器 101 管理的资源集 304 的下一苏醒集的一部分。

[0243] 如先前所提及的，数据库 2103 在高等级跟踪客户端 648、资源 105 和请求 675。数据库 2103 还可跟踪作为请求 675 一部分的所请求的截止期限。

[0244] 数据库 2103 还可维持由调度器 2101 作出的计算，这些计算包括每个请求 675 需要完成的时间、每个请求 675 需要启动的时间、每个请求 675 的结束时间、运行请求 675 的历时、执行对始发客户端或用户线程 2301 的通知或回调所需的历时、所有处置的历时、以及在请求 675 被执行之前的任何定时器 2015 的历时。

[0245] 如果存在多个请求 675，则数据库 2103 跟踪由调度器 2101 对在多个请求 675 之间可能发生的定时上的任何交迭以及因为该交迭导致的定时上的任何偏移作出计算。具体地，以上提及的高等级数据可由上述图 22 中列出的信息来跟踪。

[0246] 资源 105 可向数据库 2103 提供关于某些请求 675 可能要花费多久（也称为等待时间）才由资源 105 或另一资源 105 执行的信息。调度器 101 可针对以上结合图 22 描述的至少三种类型的等待时间数据来查询资源 105。请求等待时间 2220、分叉等待时间 2230、以及交汇等待时间 2235。如果资源 105 不提供任何等待时间的值，则将使用资源的默认值。

[0247] 如果资源 105 宁可直接处置请求 675 而不愿涉及调度器 2101，则它可响应于来自调度器 2101 的请求等待时间命令而返回 0 的等待时间值。一旦接收到作为等待时间的该 0 值，调度器 2101 就可随后丢弃对请求 675 的处置，从而资源 105 直接处置请求 675。

[0248] 资源 105 可以提供超出来自调度器 2101 的等待时间请求中所请求的之外的附加信息。例如，资源 105 可以通知调度器 2101 该资源对于将由该资源 105 服务的实际请求（由客户端 648 经由用户线程 2301 发布）支持至少两种工作模式。资源 105 可以向调度器 2101 提供关于这两种工作模式中的每一种工作模式的等待时间的数据。

[0249] 在阶段 2425，就在阶段 2427 的休眠状态之前，经由控制器 101 处置休眠状态的调度器 2101 将调用休眠调度器低功率资源（“LPR”）2107。该休眠 LPR2107 通过查看数据库 2103 来确定在阶段 2427 系统从休眠状态中苏醒时需要触发什么资源 105。休眠 LPR 2107 可被表征为应用编程接口 (API)。

[0250] 休眠 LPR 2107 将发出其确定要出现在由控制器 101 管理的资源集 304 的下一苏醒集中的那些请求 675。在这一阶段，可以实现极大的能量节省，因为休眠 LPR 2107 正在预先规划当包括处理器 110 的系统在阶段 2430 退出休眠状态时需要被处理的请求 675。

[0251] 具体地，就在阶段 2425 的休眠状态之前，休眠 LPR 2104 将标识哪些资源 105 已经调度了可能被提前并在阶段 2430 退出休眠状态之际就要处理且可被置于资源集 304 的下一苏醒集中的请求 675。

[0252] 请求 675 将作为可分叉请求 675 被发布，并且休眠 LPR 2107 将随后通知控制器 101 和调度器 2101 关于调度器 2101 执行交汇功能的新截止期限。阶段 2427 处的休眠状态可随后正常行进。当系统或处理器 110 在阶段 2430 苏醒时，RPM 157 将信令通知该交汇并且 LPR 的退出功能将信令通知调度器 2101 以交汇这些请求 675。当在阶段 2430 退出休眠状态时，调度器 2101 可以完成在使用资源 105 之前需要的任何最后工作，并且随后调动经调度的客户端的完成回调或客户端通知 2315A。

[0253] 图 25 解说了当一个或多个用户线程 2301 通过调用调度器 2101 在框 1205A、1205B 中创建两个请求 675 时的示例性场景。参见以上描述的用于创建请求 675 的图 12。

[0254] 在框 1205A 创建了第一请求作为第一客户端 648A，而框 1205B 创建了第二请求作为第二客户端 648B。用户线程 2301 向调度器 2101 指示来自客户端 648A、648B 的这两个请求 675A1、675B 将是经调度请求 675 并且来自第一客户端 648A 的第一请求 675A1 将在时间上相对于来自第二客户端 648B 的第二请求 675B 而言更早完成。

[0255] 用户线程 2301 随后发出请求 675A1、675B，如由框 675A1、675B 所表示的。尽管来自第一客户端 648A 的第一请求 675A1 将在来自第二客户端 648B 的第二请求 675B 之前发生，但用户线程 2301 可按任何次序发布这些请求 675。因此，如图 25 中所解说的，来自第二客户端的第二请求 675B 是在阶段 2501 处在阶段 2502 处来自第一客户端的第一请求 675A1 之前发布的。

[0256] 调度器 2101 在这一阶段将来自所发布的请求 675157B1、157B2 的数据记录到数据库 2103 中，并通过与定时器 2105 一起工作来建立请求 675 将得到处理的时间。具体地，调度器 2101 向定时器 2105 通知某些触发 314，如以上结合图 15-16 所述。触发 314 指示应该在何时启动每个请求 675A1、675B。

[0257] 当在阶段 2505A 定时器期满（可对应于以上提及的由调度器 2101 传达给定时器的触发 314），调度器 2101 就可在框 2101A 处理在框 2510A1 的第一请求 675A1。随后在框 2515A，调度器 2101 可以发布第一请求 675A1 已由调度器 2101 完成的客户端通知。

[0258] 在框 2510B，调度器 2101 可以随后处理第二请求 675B。随后在框 2515B，调度器 2101 可以发布第二请求 675B 已由调度器 2101 完成的客户端通知。如先前所提及的，第一请求 675A1 在第二请求 675B 之前被处理，因为第一请求 675A1 的完成或结束时间比第二请求 675B 的完成或结束时间更早。

[0259] 根据一个示例性实施例，调度器 2101 在先来先服务的基础上管理请求 675。这就意味着如果两个或更多个请求 675 具有相同的期望结束时间或完成时间，则调度器 2101 可以处理并完成首先到达调度器 2101 以进行调度的请求 675。稍后抵达调度器 2101 的请求 675 将在首先到达调度器 2101 的请求 675 之后的时间完成，即使这两个请求 675 可能具有相同完成截止期限。

[0260] 所以对于图 25 中解说的示例性实施例，如果第一和第二请求 675A1、675B 两者都具有相同期望完成时间，则第二请求 675B 将在第一请求 675A1 之前已经完成，因为第二请求 675B 是在第一请求 675A1 到达调度器 2101 之前首先就到达的。由此得出：框 2510A1 和 2515A 以及框 2510B 和 2515B 将是相反的，以反映第二请求 675B 的完成在第一请求 675A1 的完成之前。

[0261] 然而，调度器 2101 并不限于先来先服务协议的这一示例性实施例。在其它示例性实施例中，可向调度器 2101 提供允许其根据请求可能完成得多快和 / 或其完成可能如何影响便携式计算设备 100 的功率节省来调度请求 675 的功能性。

[0262] 在一些示例性实施例中，请求 675 之间在定时上没有交迭。例如，如果要求第一请求 675A1 以 10ms 退避在 100ms 的假想时间截止期限完成，并且如果要求第二请求 675B 以 10ms 退避在时间 150ms 完成，则对于这些定时在这两个请求 675A1、675B 的调度上没有交迭。将向此种示例性场景中的每个请求 675 提供其自身的定时器事件，从而将建立两个定时器事件以处理这两个非交迭请求 675A1、675B。

[0263] 通常，当多个请求 675 之间在时间上有一定交迭时，多个请求 675 可被坍缩并由调度器 2101 处理。可以定时以及处理请求 675 所需的系统开销的形式来定义请求 675 之间的交迭。

[0264] 例如，如果两个请求 675 的所请求截止期限显著远离彼此，则调度器 2101 可以确定并发地或顺序地处理这两个请求 675 而非创建两个单独的定时器事件是否更为高效。换句话说，如果调度器 2101 确定并发地或者顺序地调度两个请求 675 将比用定时器 2105 创建两个单独的定时器事件更为高效，则调度器 2101 可以超驰所请求的完成截止期限。

[0265] 由调度器 2101 作出的关于请求 675 何时应在所请求的截止期限之前被处理的这一效率确定可以考虑 PCD 100 的功耗。如果调度器 2101 确定与所请求的截止期限（其中 PCD 100 可能具有较少功率来处理一个或多个请求 675）相比，PCD 100 当前具有足够功率来处理这些请求 675，则调度器 2101 可以将这些请求 675 调度得比其所请求的截止期限更早。调度器 2101 可以通过确定特定请求 675 是否与“ON”（导通）状态有关或者特定请求 675 是否与“OFF”（关闭）状态有关来跟踪请求 675 的功耗。

[0266] 图 26 解说了当用户线程 2301 经由客户端 648A 可以确定它不再想要特定请求 675 得到处理时的示例性时序图 2600。此类不希望或不想要的请求 675 被表征为撤销调度的请求 675。

[0267] 关于撤销调度请求 675 有两种情形：(a) 尚未被调度器 2101 处理的请求，以及 (b) 当前正被调度器 2101 处理的请求 675。图 26-27 解说了已经被创建但尚未被调度器 2102 处理的请求 675。同时，图 28 解说了已经被创建但正被调度器 2101 处理并且随后客户端 648 稍后在请求 675 正被处理时发布撤销调度请求命令的请求 675。

[0268] 回头参照图 26，客户端 648A 在阶段 1205A 被创建，如由框 648A 所表示的。在由客户端 648A 发布请求 675 之前，在阶段 2610 发布撤销调度请求命令。在这一场景中，调度器 2101 通常可以简单地将不希望的请求 675 从其存储在数据库 2103 中的经调度请求 675 列表中移除。在阶段 2615，可发布新的即时请求而无需其正被调度（如所解说的）或者另一经调度请求可被启动（在客户端上重复 1205A 和 648A）。

[0269] 图 27 解说了时序图 2700，其中已经调度了请求 675 并且用户线程 2301 已经在阶

段 2705 发布该请求 675，但是在已经发布撤销调度的请求命令 2605 时调度器 2101 尚未开始处理前述请求 675。具体地，经调度客户端 648A 在对应于图 12 的框 1205A 被创建，如上所述。

[0270] 随后在阶段 2705，客户端 648A 发布该请求 675。在阶段 2305A，具有其关于这一个或多个请求 675 的启动时间的数据库 2103 将随后被更新。随后在阶段 2305B，撤销调度请求命令 2605 已经由客户端 648A 发布。在此刻，数据库 2103 可由调度器 2102 来更新，其中请求 675 可从数据库 2103 中被移除。请求 675 可从数据库 2103 中被移除是因为请求 675 尚未被调度器 2101 处理（即，请求 675 尚未被发送给其所请求的资源 105）。在这一阶段，可以启动另一即时或经调度请求。

[0271] 图 28 解说了示例性时序图 2800，其中已经调度了请求 675 并且用户线程 2301 已经发布了该请求 675，并且调度器 2101 已经开始处理该请求 675。换句话说，调度器 2101 已经开始处理现在不被客户端 648 或用户线程 2301 所希望的请求 675。

[0272] 根据这一示例性实施例，当用户线程 2301 或客户端 648 在阶段 2805 发布撤销调度请求命令 2605 时，调度器 2101 发布对资源 105 的锁定，如由框 2815 所解说的，其将撤销调度请求命令封锁在外，如由框 2810 所指示的。以此方式，调度器 2101 在框 2505 中完成经调度请求 675，而无视在阶段 2805 发布的撤销调度请求命令 2605。

[0273] 因此在图 28 的这一示例性实施例中，响应于阶段 2805 处的撤销调度请求命令 2605，用户线程 2301 或客户端 648 可以接收客户端通知 2510 以及针对撤销调度请求命令 2605 的关于请求 675 已经被处理 / 完成的返回值。

[0274] 在其中请求 675 已经被置于将由休眠 LPR 2107 完成的资源集 304 的下一苏醒集中，并且用户线程 2301 已经发布撤销调度请求命令 2605，并且直至系统苏醒才发布客户端通知 2510 的场景中，调度器 2101 可以继承请求 675 的优先级，这允许调度器 2101 立即对请求 675 进行动作。

[0275] 这成为允许用户线程 2301 和调度器 2101 立即完成请求 675 而非等待其所请求的截止期限的同步点。调度器 2101 将向用户线程 2301 发回通知，其指示请求 675 已经被完成并且因此撤销调度请求命令 2605 也完成。

[0276] 图 29 解说了简单场景，其中单个应用程序正在运行并且在活跃状态期间处理一个或多个请求 675 后，CPU 110 可以进入休眠状态 2902。Y 轴可以表示电压或电流而 X 轴可以表示时间，诸如以毫秒为单位。

[0277] 在阶段 1 和 2 之间，单个应用正在运行并且 CPU 110 处于活跃并且具有如 5 毫伏的电压。随后 CPU 110 进入 X 轴时间线上阶段 2 和 3 之间的休眠状态 2902，在其中具有零电压。类似地，在 X 轴时间线上的阶段 3 和 4 之间，CPU 110 处于活跃并且具有诸如 5 毫伏的电压。

[0278] 在阶段 1.1，用户线程 2301（未解说）可以向 CPU 繁忙监视器（“CBM”）资源 2109 发布请求，如以上在图 21 中所描述的。CBM 资源 2109 是调度器 2101 的一部分和 / 或由其建立。对 CBM 资源 2109 的任何非零请求 675 指示发布请求 675 的客户端 648 或用户线程 2301 预期 CPU 110 在该请求 675 的执行期间保持繁忙。CBM 资源 2109 是二进制资源，这意味着只要有任何客户端 648 处于繁忙，CPU 110 就被 CMB 资源 2109 认为是繁忙的，并且直至最后的客户端 675 完成非零或繁忙请求，才将 CPU 110 表征为自由或非繁忙的。

[0279] 在阶段 1.2，可抑制资源请求 675 可由调度器 2101 向资源 105 发布。可抑制请求 675 是从可抑制客户端 648 发布的资源请求 675，并且这些请求 675 具有当 CPU 110 进入空闲或者进入休眠状态 2902 时它们不需要被兑现的属性。换句话说，可抑制请求 675 在 CPU 110 进入休眠状态 2902 时可以被“关断”。

[0280] 可抑制请求 675 与所要求的请求 675 直接相反，后者在所有时间均要求被 CPU 110 兑现。通过在阶段 2 发布可抑制请求 675，就不需要在进入休眠状态 2902 之前显式地取消请求 675。

[0281] 在资源 105 被 RPM 157 服务的特定情形中，可抑制请求 675 不被添加至资源集 304 的休眠集，并且仅被添加至资源集 304 的活跃集。同时，所要求的请求 675 被添加至资源集 304 的活跃和休眠集两者（参见图 15），如由本领域普通技术人员所理解的。

[0282] 在阶段 2.1，一旦请求 675 由资源 105 完成，则资源 105 向调度器 2101 的 CBM 资源 2109 发回通知，其指示资源 105 预期“不久”将进入空闲。阶段 2.1 处的资源开始进行与请求 675 的完成相关联的清除工作。

[0283] 在阶段 2.2，就在休眠状态 2902 之前，客户端 648 或用户线程 2301 可以使用自行斟酌功能 / 特征来调度新请求 2111，并且向调度器 2101 指示其需要新请求 2111 在 X 轴上的阶段 3 处得到处理。自行斟酌功能或特征可包括作为该新请求 2111 的一部分的数据。

[0284] 自行斟酌功能通知调度器 2101 客户端 648 不指望正被完成的先前请求 675 或当前请求 675 将持续到直至新请求 2111 生效为止。换句话说，在有了此附加的自行斟酌功能或特征作为新请求 2111 的一部分的情况下，调度器 2101 将认识到正被处理的当前客户端请求 675 不需要在新的经调度请求 2111 生效时是完成的或者继续其处理。有了这一自行斟酌功能，调度器 2101 就有按调度器 2101 自行斟酌来完成当前请求 675 的自由而非要求。

[0285] 在阶段 2，调度器 2101 可以查阅正被 CBM 资源 2109 跟踪的 CPU 110 的当前状态。如果调度器 2101 注意到如由 CBM 资源 2109 所跟踪的那样 CPU 110 不处于繁忙状态，则调度器 2101 可以允许休眠状态（经由图 15 的控制器 101 及其资源集）将 CPU 110 关闭达阶段 2 和 3 之间的空闲或休眠状态而无需取消用自行斟酌功能发布的新请求 2111。以此方式，RPM 157 经由资源集 304 的休眠集（参见图 15）在进入实际休眠状态 2902 之前不需要发布任何取消请求 675。

[0286] 总之，调度器 2101 可观察由 CBM 资源 2109 跟踪的繁忙状态以确定是否预期 CPU 110 不久后有休眠状态。有了对新请求 2111 的自行斟酌功能，如果预期不久后将发生休眠，则调度器 2101 能够决定是否要结束 / 取消当前请求 675 并调度新请求 2111。

[0287] 替换地，如果调度器 2101 认识到当前请求 675 和新请求 2111 是类似的或相同的，则调度器 2101 可以决定不取消 / 结束当前请求 675 并且不兑现新请求 2111。这将允许 RPM 157 经由控制器 100 的休眠集来关闭 CPU 110 并随后打开 CPU 110 从而当前请求 675 在 CPU 被打开时得到处理。

[0288] 新请求 2111 的这一自行斟酌功能 / 特征允许调度器 2101 决定是否要取消当前请求 675 或发出新请求 2111。调度器 2101 的这一自行斟酌功能以及通过查看由 CBM 资源 2109 跟踪的繁忙状态来预计休眠状态的能力增加了系统 2100 的效率并节省了功率。功率可以得到节省是因为与不允许调度器 2101 对于其从客户端 648 接收的请求 675 具有自行斟酌的系统相比，调度器 2101 不需要执行那么多的工作。

[0289] 工作因调度器 2101 使用 CBM 资源 2109 和新请求 2111 中的自行斟酌功能而得以减少，因为调度器 2101 不需要对由 RPM 157 和控制器 101（参见图 15）管理的资源集的休眠集和活跃集调整任何参数。不需要对休眠集和活跃集的任何参数进行调整是因为调度器 2101 能够确定 CPU 110 的相继活跃状态彼此相似并且不要求对活跃或休眠集进行任何改变。

[0290] 无需调整由 RPM 157 和控制器 101 管理的资源集的活跃或休眠集，本发明的系统 2100 已经实现了 15％数量级的效率（即，与不允许调度器 2101 对于兑现 / 取消当前和新请求 675 具有自行斟酌的系统相比，已经实现了 45ms 中的 7ms 或以上的数量级的时间节省）。

[0291] 如先前所提及的，调度器 2101 能够用 CBM 资源 2109 来提高系统 2100 的效率。CBM 资源 2109 是二进制指示符，其指示在系统 2100 中是否正发生允许调度器 2101 推断休眠状态将不会在相对不久发生的事情。

[0292] 当 CBM 资源 2109 指示绝大多数或所有系统元素不处于“繁忙”（这通常意指还未为资源 105 调度大量工作并且那些资源 105 即将完全没有请求 675），则调度器 2101 可以预计休眠状态 2902 相对不久即将发生。通过观察由 CBM 资源 2109 跟踪的状态，调度器 2101 可以取决于调度器 2101 是否预计将发生休眠状态 2902 来作出关于资源是否应该被断电的决定。

[0293] 例如，如果调度器 2101 注意到所有资源 105 当前正注册非繁忙状态，则调度器 2101 可以预计休眠状态 2902 将发生并且调度器 2101 可以允许休眠状态 2902（经由具有其资源集的 RPM 157 和控制器 101）将一个或多个资源 101 断电，从而调度器 2101 不需要执行这些任务并因此节省工作。

[0294] 在另一示例中，如果调度器 2101 注意到第一资源 105A 正注册非繁忙状态（即，第一资源 105A 不活跃），而第二资源 105B 正注册繁忙状态并且如果调度器 2101 确定第一资源 105A 不被调度用于任何即将到来的请求 675，则调度器 2101 可以关闭非繁忙的第一资源 105A 以便节省功率。这一关闭非繁忙第一资源 105A 的动作可节省总体功率，因为由于第二资源 105B 正注册繁忙状态而预计不会有休眠状态 2902。如果允许非繁忙第一资源 105A 维持在“On”但处于非活跃状态，则可能浪费功率。

[0295] 图 30 解说了调度器 2101 如何管理在阶段 3015 和 3030 之间发生的经调度休眠状态期间的非预期唤醒或中断 3002B 的示例性场景。X 轴可以表示以毫秒计的时间，而 Y 轴可以表示电流或电压。

[0296] 调度器 2101 在 CPU 110 的繁忙或活跃状态 3002A 的阶段 3005 开始处管理对一个或多个资源 105 的一个或多个请求 675。在阶段 3015 进入休眠状态之前，调度器 2101 在查阅数据库 2103 后认识到在为阶段 3030 调度的下一经调度繁忙或活跃状态 3002C，当前活跃的资源的当前状态将很可能不会改变。

[0297] 因此，在阶段 3015 之前，调度器 2101 通过不履行关闭或关断资源 105 并更新资源集 304 的相应休眠和活跃集的过程来避免额外工作并节省能量。因此在阶段 3015，调度器 2101 允许所有资源 105 进入休眠状态并允许每个资源 105 的休眠集关闭相应资源 105。一旦进入阶段 3015，调度器 2101 预期系统在阶段 3030 苏醒。

[0298] 换句话说，调度器 2101 在阶段 3020 不会预期苏醒 3002B。然而，在阶段 3020，调度器 2101 的确接收引起在阶段 3020 处的苏醒和非预期活跃状态 3002B 的非预期中断。调

度器 2101 可以通过比较数据库 2103 中存在的定时数据来确定活跃集 3002B 是非预期的。具体地，如果调度器 2101 注意到当前活跃状态 3002B 已经在时间上出现得比阶段 3030 处的经调度苏醒时间早得多，则调度器 2101 将认识到此第二活跃状态 3002B 是不为系统所预期或期望的。

[0299] 在这一阶段 3020，调度器 2101 确定被调度为要在系统退出在阶段 3015 开始的先前经调度休眠状态时呈活跃的资源 105 对于当前中断或苏醒事件 3002B 是不需要的。因此，调度器 2101 可以向资源 105 发布取消请求 3007，从而资源 105 在这一非期望的第二活跃状态 3002B（其中资源 105 不在被使用）期间被关闭。

[0300] 接着，在阶段 3025 处的第二非预期活跃状态 3002B 的结束处，调度器 2101 可以与休眠 LPR 2107 通信以便调度新请求 2111 从而打开资源 105，从而请求 2111 在第三活跃状态 3002C（先前由调度器 2101 预期 / 调度的）处变为下一活跃苏醒集 304 的一部分。以此方式，调度器 2101 具有推断或重新调度中断状态（诸如图 30 的非预期的第二活跃状态 3002B）可能不需要的请求 675 的智能。

[0301] 例如，假设图 30 中的资源 105 是 RF 调制解调器。调度器 2101 按其自行斟酌已经决定其将允许资源 105 的休眠集关闭 RF 调制解调器，从而 RF 调制解调器将对于为阶段 3030 调度的下一预期的活跃状态 3002C 立即变为活跃。

[0302] 然而，调度器 2101 在阶段 3020 得知导致在此阶段 3020 的非预期苏醒的中断与不需要 RF 调制解调器的应用程序相关联。调度器 2101 可以随后发布对 RF 调制解调器的取消请求 2111，从而它在阶段 3020 的这一不期望的活跃状态 3002B 期间不执行与其先前请求 675 相对应的其经调度工作。调度器 2101 使用休眠 LPR 2107 将随后为阶段 3030 处的预期且先前经调度的活跃状态 3002C 中的下一苏醒集（“NAS”）304 调度与 RF 调制解调器相关联的新请求 2111。

[0303] 鉴于以上公开内容，本领域普通技术人员能够例如基于本说明书中的流程图和相关联的描述来编写计算机代码或标识恰适的硬件和 / 或其它逻辑或电路系统来实现分布式资源管理系统和方法而没有困难。因此，并不认为对特定程序代码指令集或详细硬件设备的公开是充分理解如何作出并使用分布式资源管理系统和方法所必需的。所要求保护的计算机实现的过程的发明性的功能性在以上描述中结合可解说各种过程流程的附图更为详细地进行了解释。此外，处理器 110、126、202、206 等结合存储器 112 及其中存储的指令可用作用于执行本文中所描述的一个或多个方法步骤的装置。

[0304] 在一个或多个示例性方面中，所描述的功能可在硬件、软件、固件或其任何组合中实现。如果在软件中实现，则各功能可以作为一条或多条指令或代码存储在计算机可读介质上或藉其进行传送。计算机可读介质包括计算机存储介质和通信介质两者，这些介质包括促成计算机程序从一地向另一地转移的任何介质。存储介质可以是能被计算机访问的任何可用介质。以示例而非限定的方式，此类计算机可读介质可以包括 RAM、ROM、EEPROM、CD-ROM 或其它光盘存储、磁盘存储或其它光学或磁存储设备、或者可用以携带或者存储指令或数据结构形式的期望程序代码且可由计算机访问的任何其它介质。本文中所使用的术语“盘”或“碟”包括但不限于压缩碟（“CD”）、激光碟、光碟、数字多用碟（“DVD”）、软盘以及蓝光碟。上述组合应当也被包括在计算机可读介质的范围内。

[0305] 尽管已详细解说和描述了精选的方面，但是将可理解，可在其中作出各种替换和

变更而不会脱离本公开如所附权利要求所定义的精神和范围。

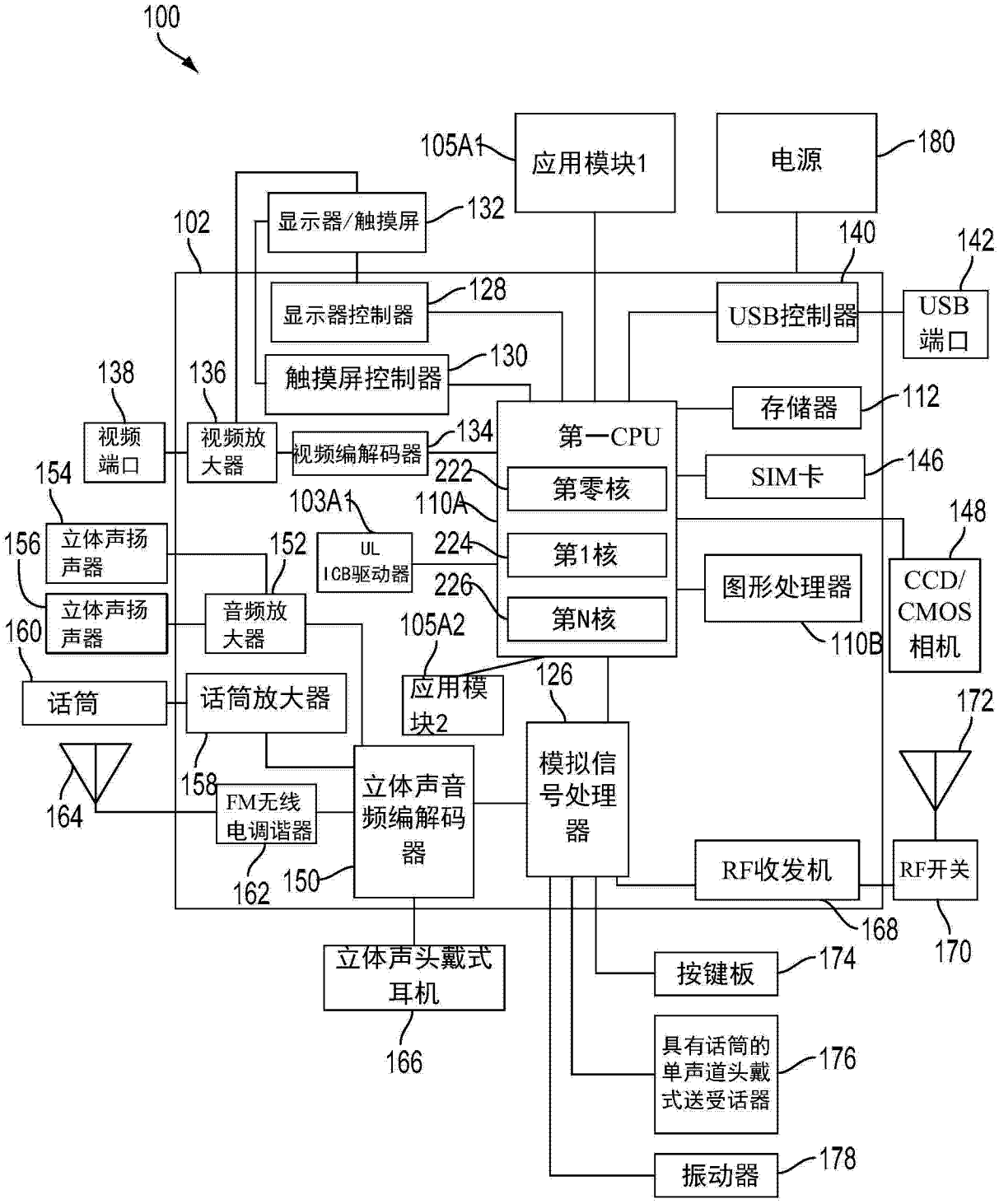

100
105A1
应用模块1
电源
180
显示器/触摸屏
132
102
140
142
显示器控制器
128
USB控制器
USB端口
触摸屏控制器
130
138
136
存储器
112
视频端口
视频放大器
视频编解码器
134
第一CPU
154
222
第零核
SIM卡
146
103A1
110A
148
156
UL ICB驱动器
152
224
第1核
CCD/CMOS相机
立体声扬声器
226
图形处理器
160
音频放大器
105A2
第N核
立体声扬声器
110B
话筒
话筒放大器
应用模块2
126
172
158
模拟信号处理器
164
FM无线电调谐器
立体声音频编解码器
RF收发机
RF开关
162
150
168
170
立体声头戴式耳机
按键板
174
具有话筒的单声道头戴式送受话器
176
166
振动器
178

图 1

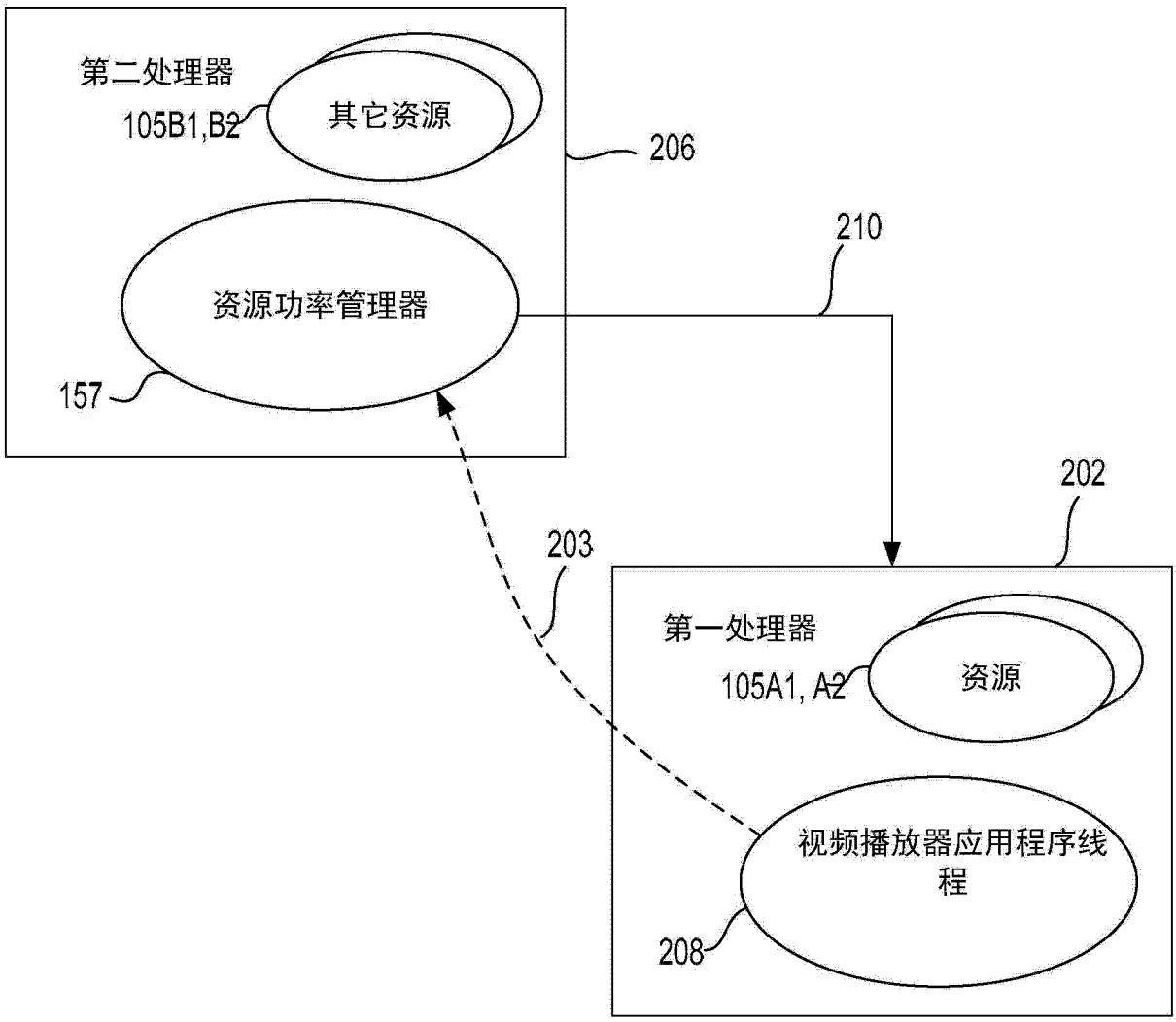

第二处理器
105B1,B2
其它资源
206
210
资源功率管理器
157
202
203
第一处理器
105A1, A2
资源
视频播放器应用程序线程
208

图 2

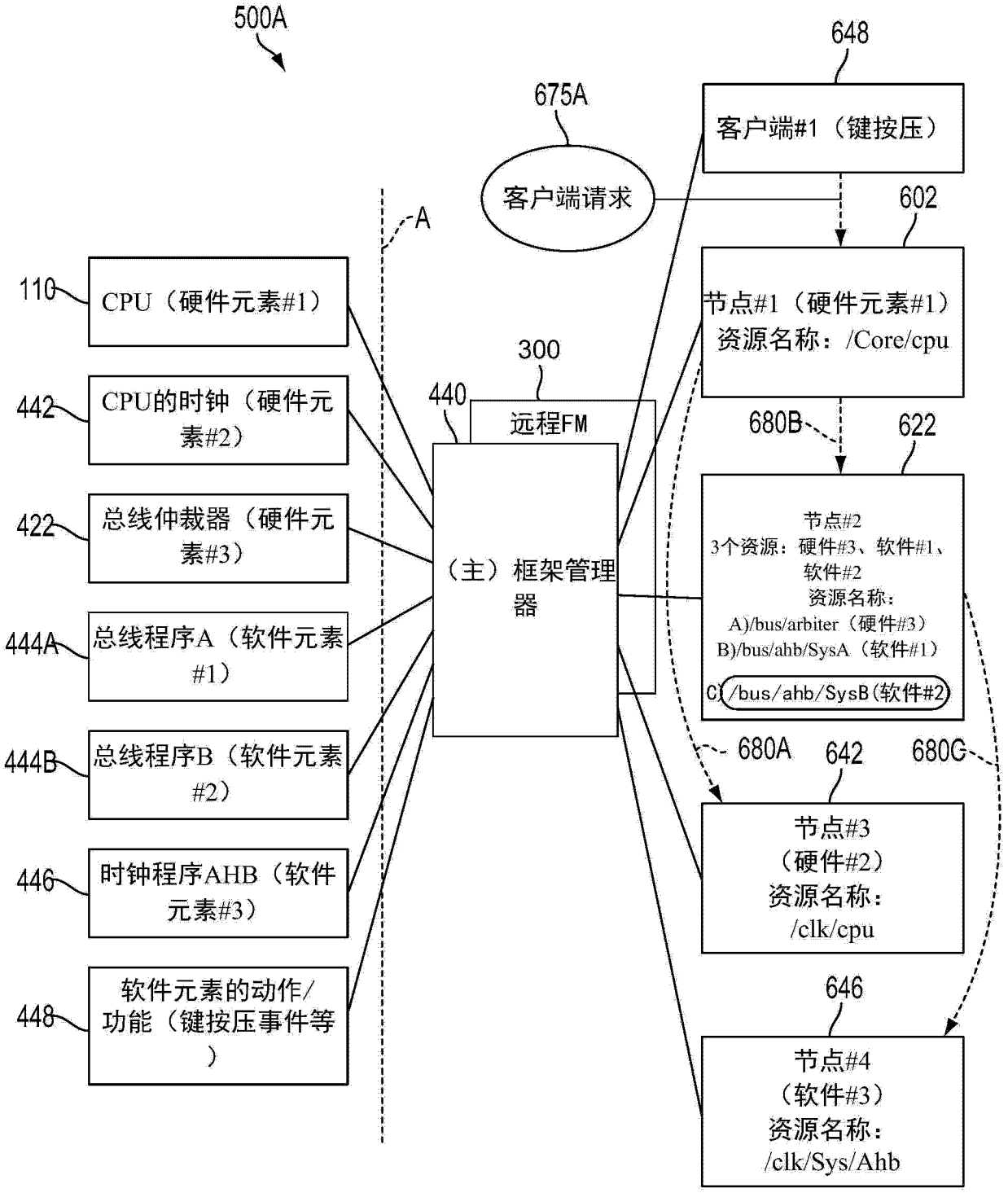

500A
675A
客户端请求
648
客户端#1（键按压）
602
节点#1（硬件元素#1）
资源名称：/Core/cpu
A
110
CPU（硬件元素#1）
300
440
远程FM
442
CPU的时钟（硬件元素#2）
422
总线仲裁器（硬件元素#3）
444A
总线程序A（软件元素#1）
444B
总线程序B（软件元素#2）
446
时钟程序AHB（软件元素#3）
448
软件元素的动作/功能（键按压事件等）
（主）框架管理器
680B
622
节点#2
3个资源：硬件#3、软件#1、软件#2
资源名称：
A)/bus/arbiter（硬件#3）
B)/bus/ahb/SysA（软件#1）
C)/bus/ahb/SysB(软件#2)
680A
642
680C
节点#3
（硬件#2）
资源名称：
/clk/cpu
646
节点#4
（软件#3）
资源名称：
/clk/Sys/Ahb

图 3

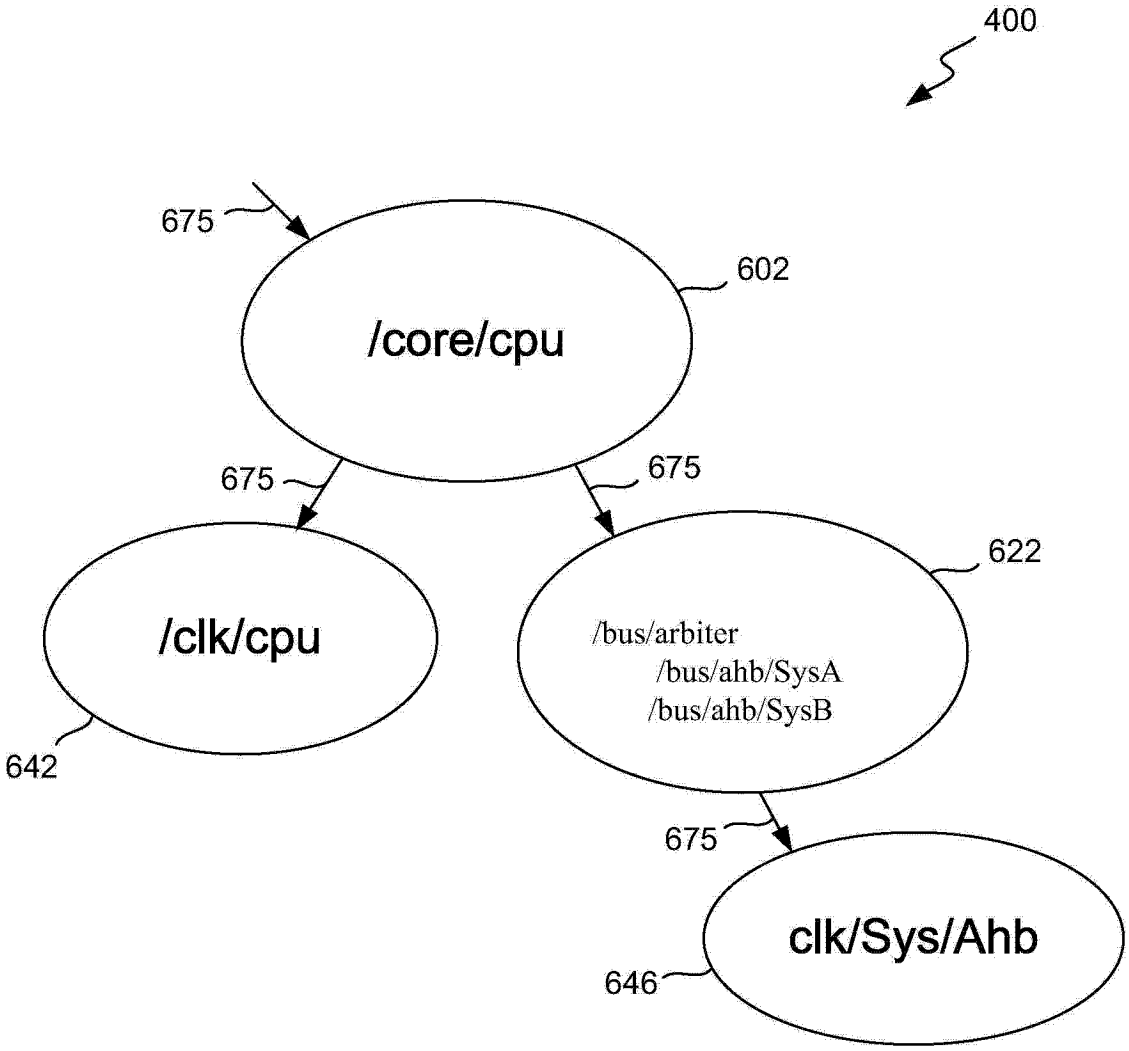

400
675
602
/core/cpu
675
675
/clk/cpu
642
622
/bus/arbiter
/bus/ahb/SysA
/bus/ahb/SysB
675
clk/Sys/Ahb
646

图 4

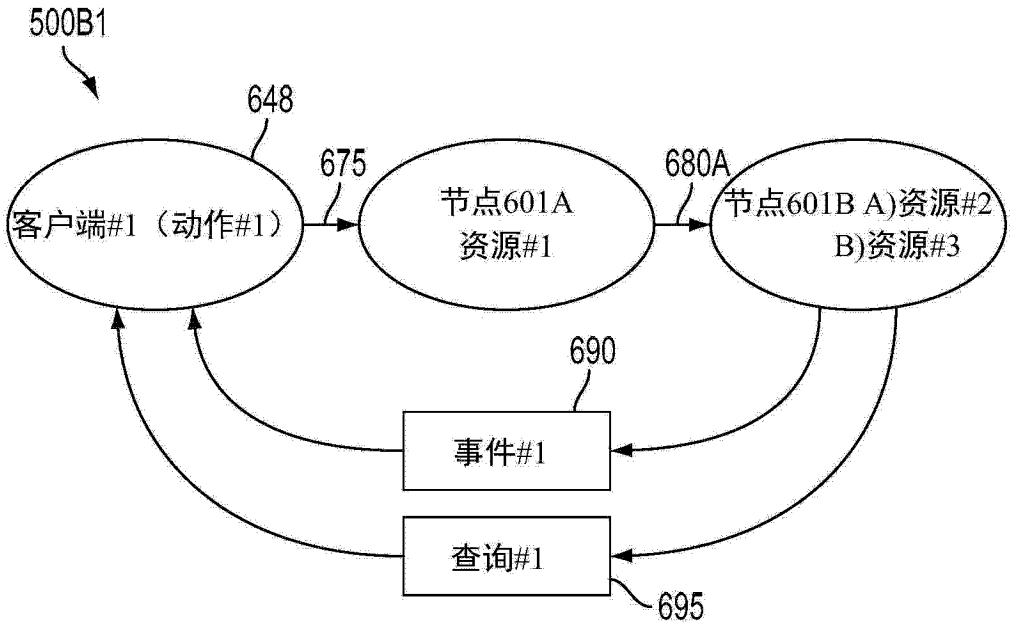

500B1
648
675
680A
客户端#1（动作#1）
节点601A
资源#1
节点601B A)资源#2
B)资源#3
690
事件#1
查询#1
695

图 5

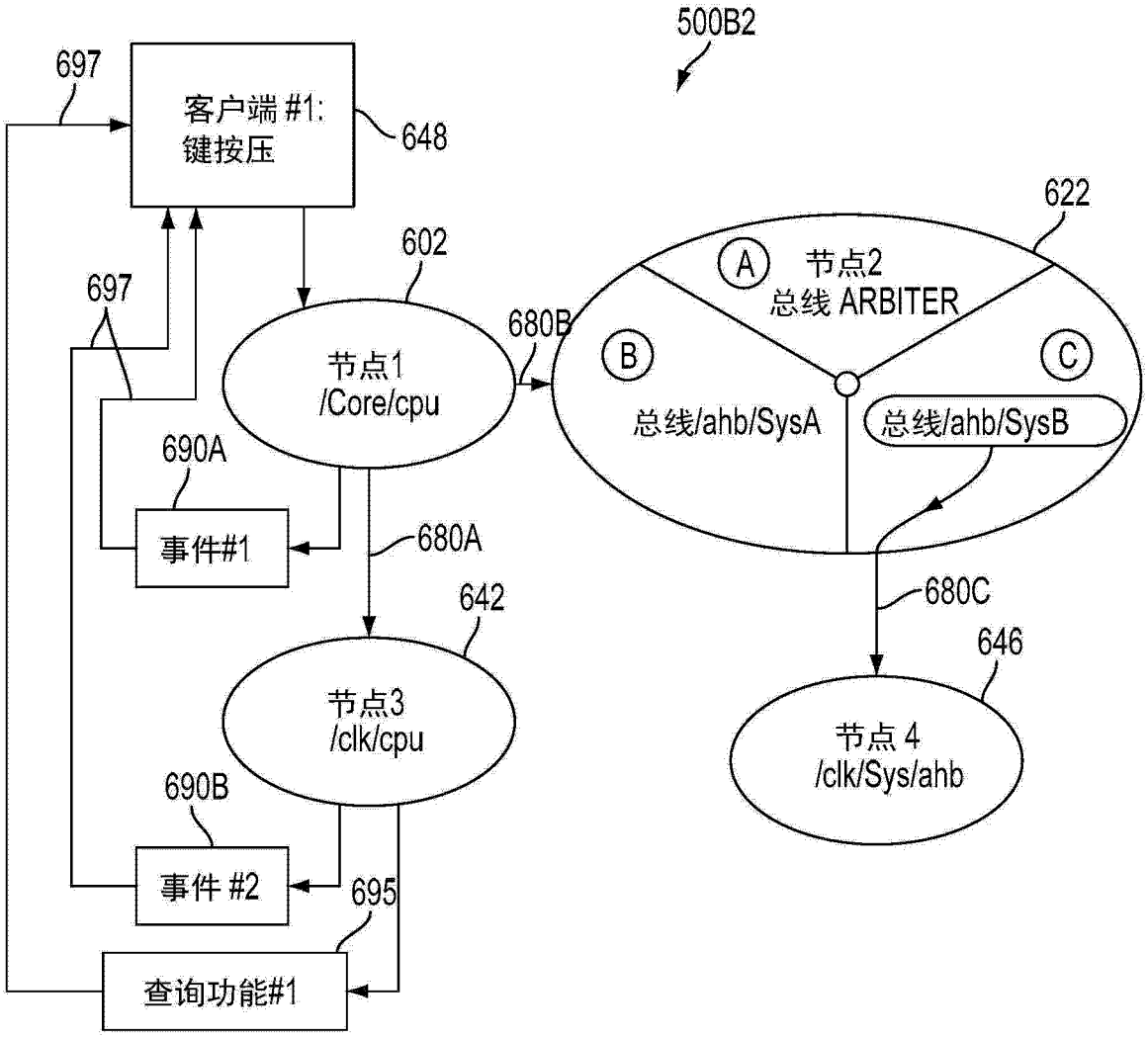

500B2
697
客户端 #1:
键按压
648
622
602
697
节点1
/Core/cpu
680B
A
节点2
总线 ARBITER
B
C
总线/ahb/SysA
总线/ahb/SysB
690A
事件#1
680A
642
680C
646
节点3
/clk/cpu
节点 4
/clk/Sys/ahb
690B
事件 #2
695
查询功能#1

图 6

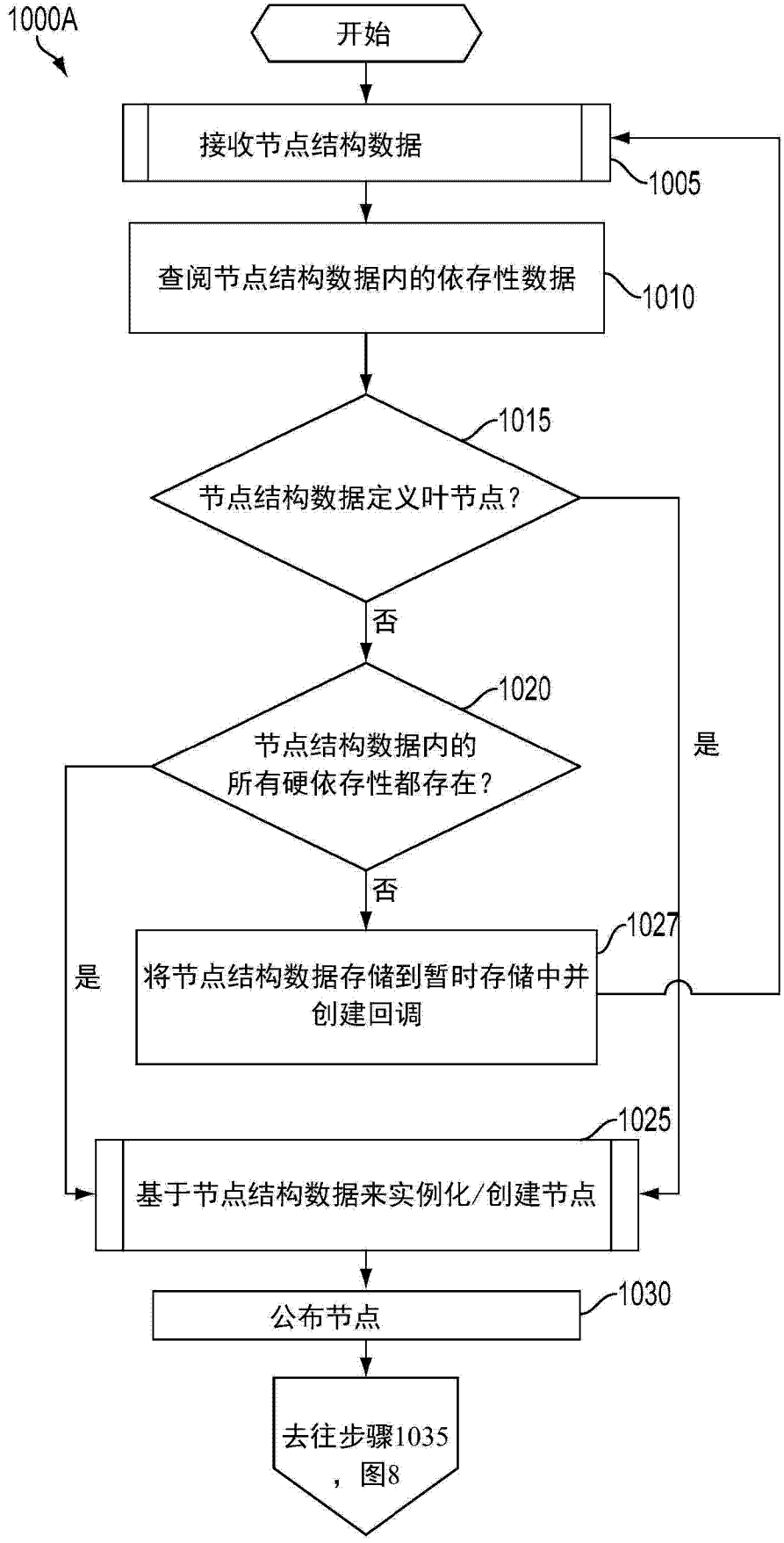

1000A
开始
接收节点结构数据
1005
查阅节点结构数据内的依存性数据
1010
1015
节点结构数据定义叶节点？
否
是
1020
节点结构数据内的
所有硬依存性都存在？
否
是
1027
将节点结构数据存储到暂时存储中并
创建回调
1025
基于节点结构数据来实例化/创建节点
1030
公布节点
去往步骤1035
，图8

图 7

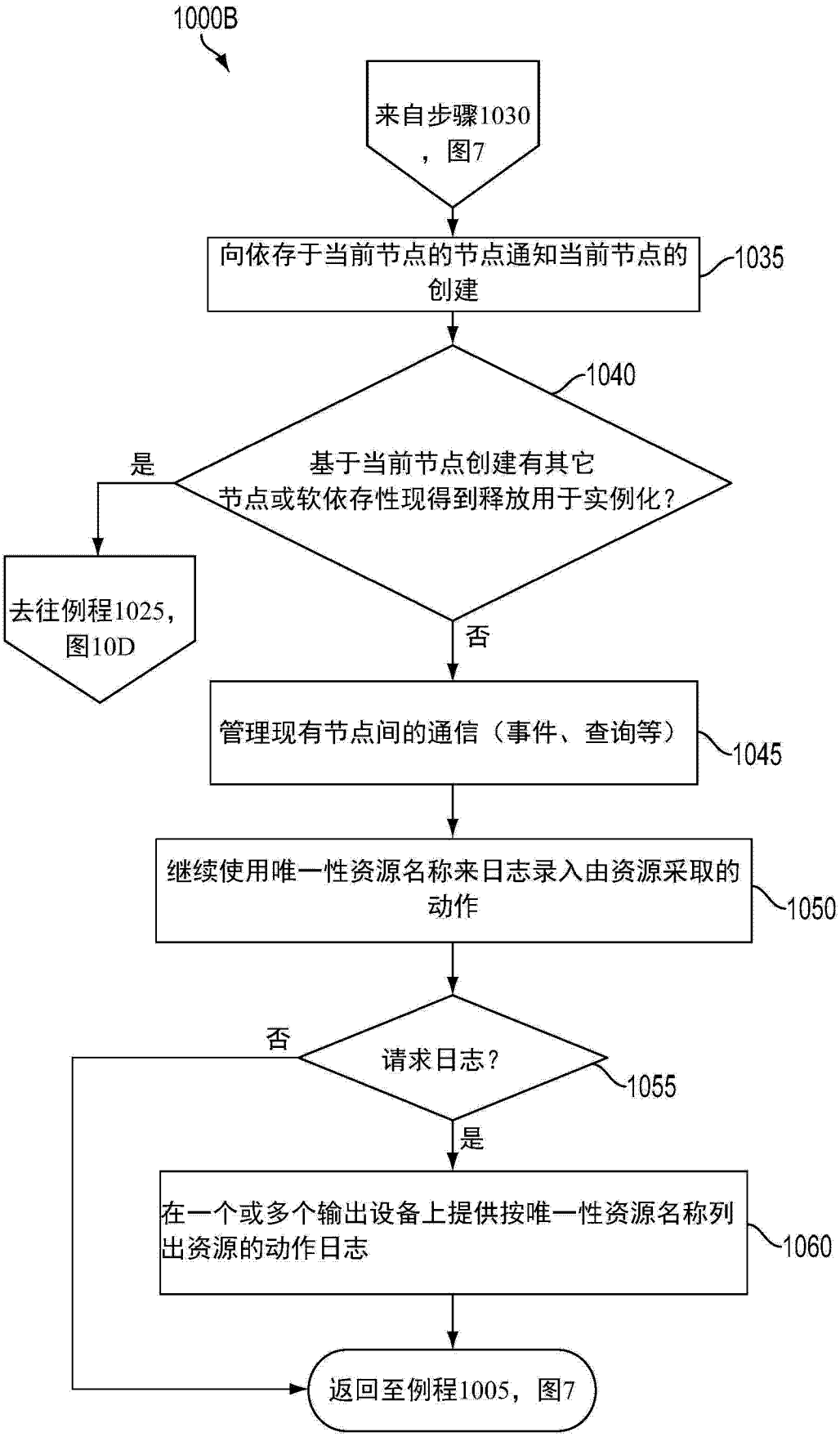

1000B
来自步骤1030，图7
向依存于当前节点的节点通知当前节点的创建
1035
1040
基于当前节点创建有其它节点或软依存性现得到释放用于实例化？
是
去往例程1025，图10D
否
管理现有节点间的通信（事件、查询等）
1045
继续使用唯一性资源名称来日志录入由资源采取的动作
1050
否
请求日志？
1055
是
在一个或多个输出设备上提供按唯一性资源名称列出资源的动作日志
1060
返回至例程1005，图7

图 8

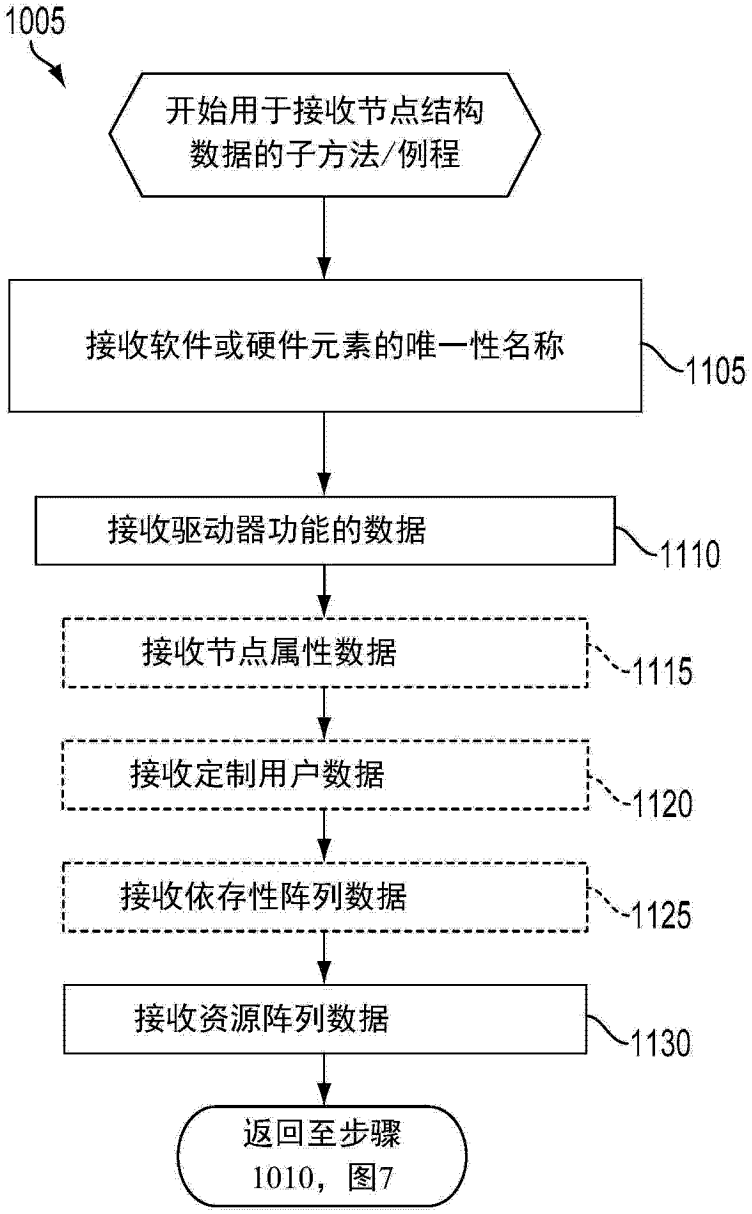

1005
开始用于接收节点结构数据的子方法/例程
接收软件或硬件元素的唯一性名称
1105
接收驱动器功能的数据
1110
接收节点属性数据
1115
接收定制用户数据
1120
接收依存性阵列数据
1125
接收资源阵列数据
1130
返回至步骤1010，图7

图 9

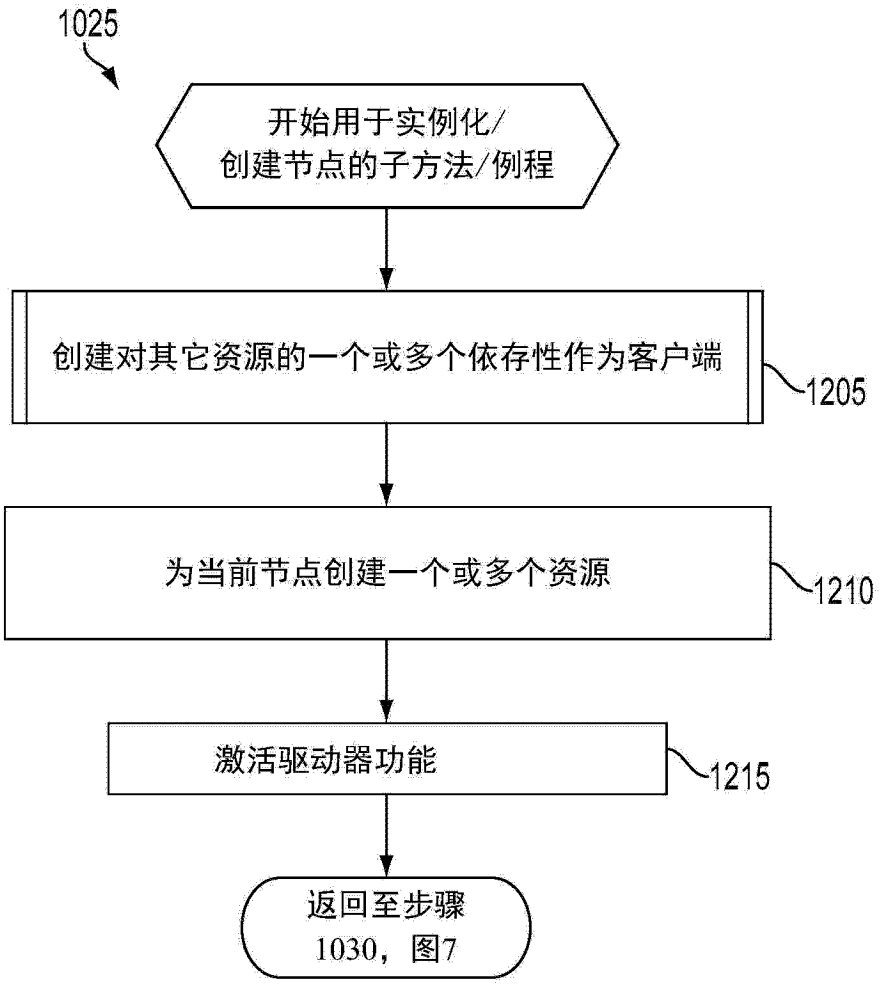

1025
开始用于实例化/
创建节点的子方法/例程
创建对其它资源的一个或多个依存性作为客户端
1205
为当前节点创建一个或多个资源
1210
激活驱动器功能
1215
返回至步骤
1030，图7

图 10

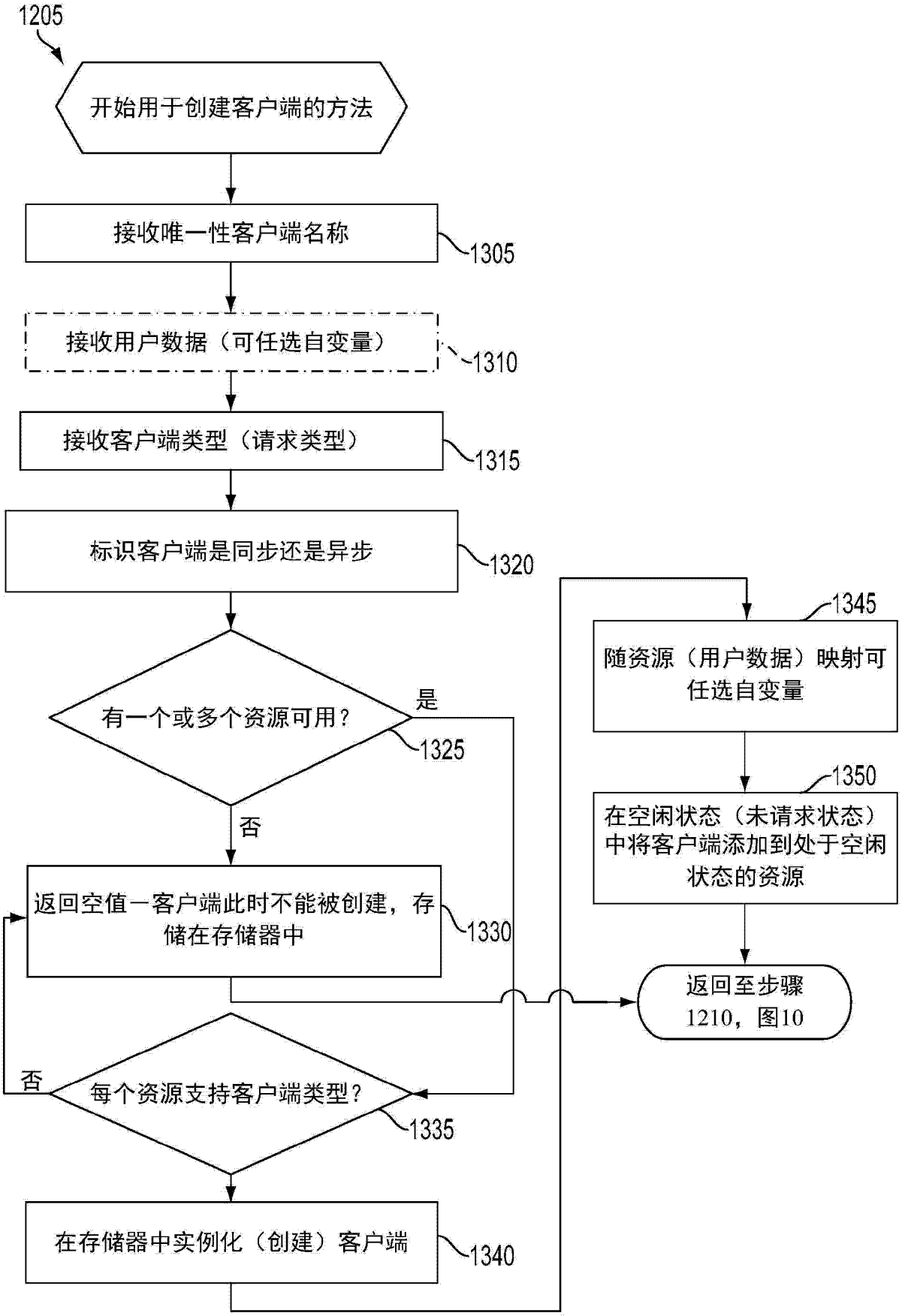

1205
开始用于创建客户端的方法
接收唯一性客户端名称
1305
接收用户数据（可任选自变量）
1310
接收客户端类型（请求类型）
1315
标识客户端是同步还是异步
1320
有一个或多个资源可用？
是
1325
否
返回空值—客户端此时不能被创建，存储在存储器中
1330
每个资源支持客户端类型？
否
1335
在存储器中实例化（创建）客户端
1340
1345
随资源（用户数据）映射可任选自变量
1350
在空闲状态（未请求状态）中将客户端添加到处于空闲状态的资源
返回至步骤1210，图10

图 11

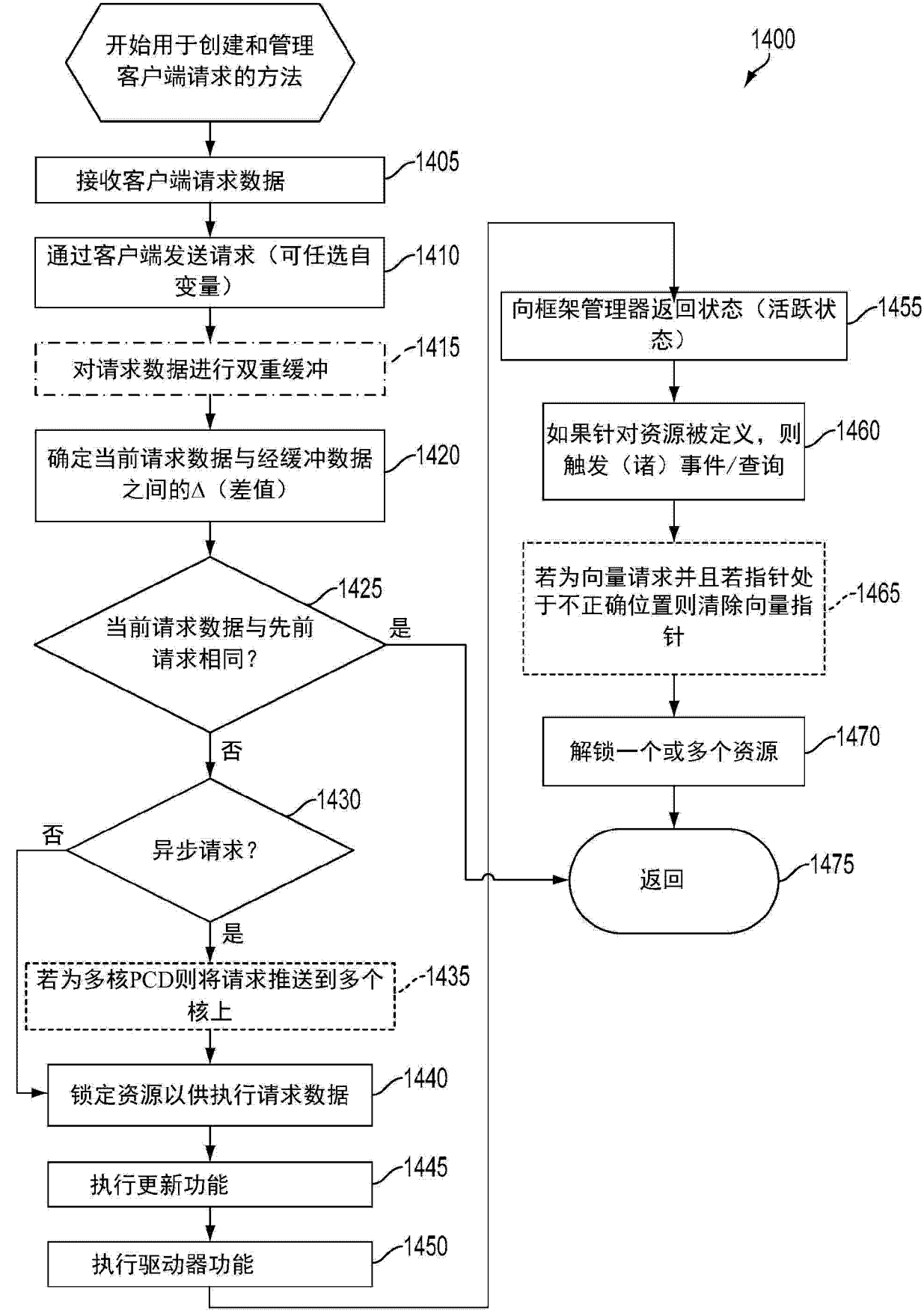

开始用于创建和管理客户端请求的方法
1400
接收客户端请求数据
1405
通过客户端发送请求（可任选自变量）
1410
对请求数据进行双重缓冲
1415
确定当前请求数据与经缓冲数据之间的Δ（差值）
1420
当前请求数据与先前请求相同？
1425
是
否
异步请求？
1430
否
是
若为多核PCD则将请求推送到多个核上
1435
锁定资源以供执行请求数据
1440
执行更新功能
1445
执行驱动器功能
1450
向框架管理器返回状态（活跃状态）
1455
如果针对资源被定义，则触发（诸）事件/查询
1460
若为向量请求并且若指针处于不正确位置则清除向量指针
1465
解锁一个或多个资源
1470
返回
1475

图 12

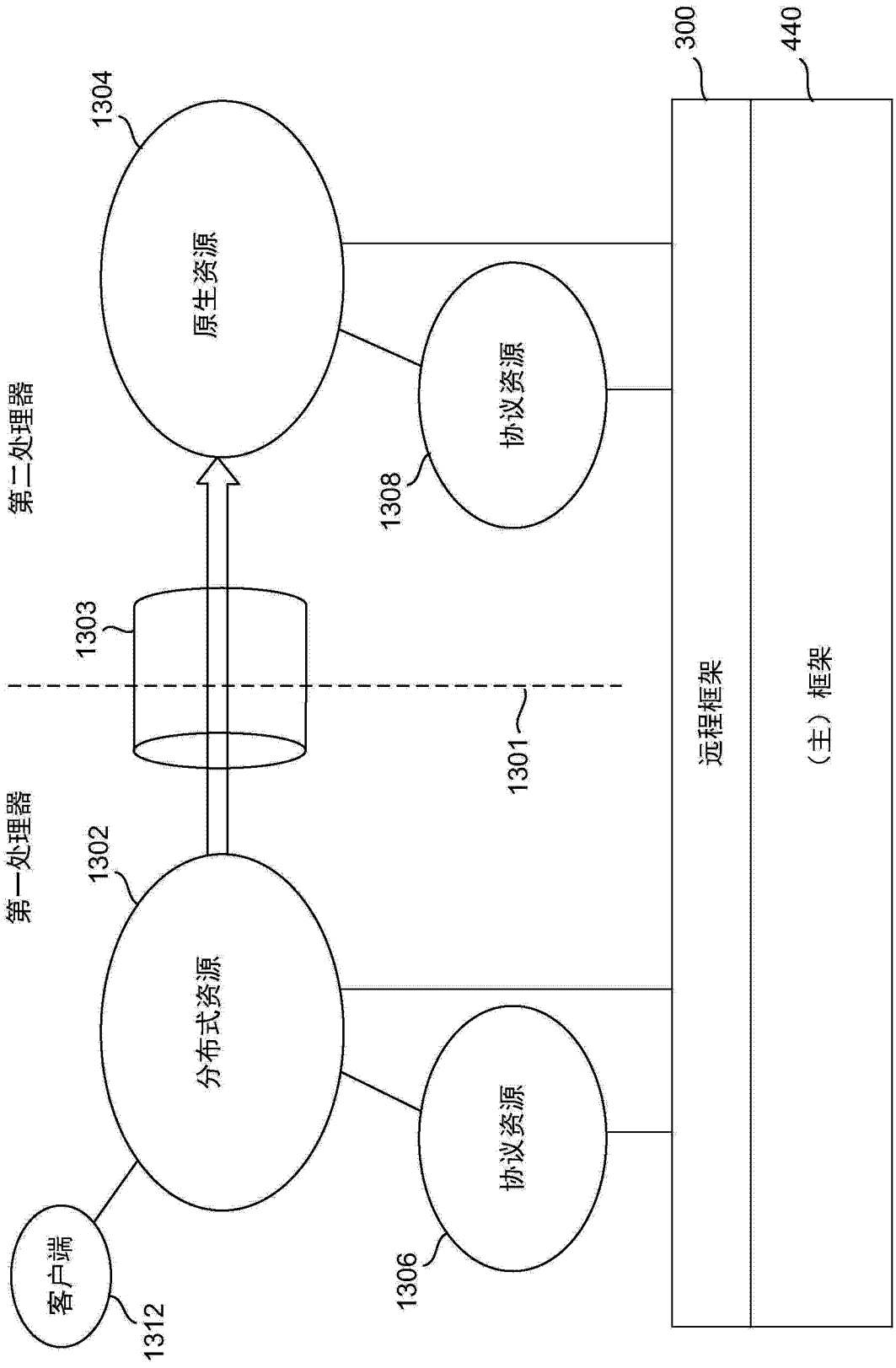

300
440
1304
原生资源
协议资源
1308
第二处理器
1303
1301
第一处理器
1302
分布式资源
协议资源
1306
客户端
1312
远程框架
（主）框架

图 13

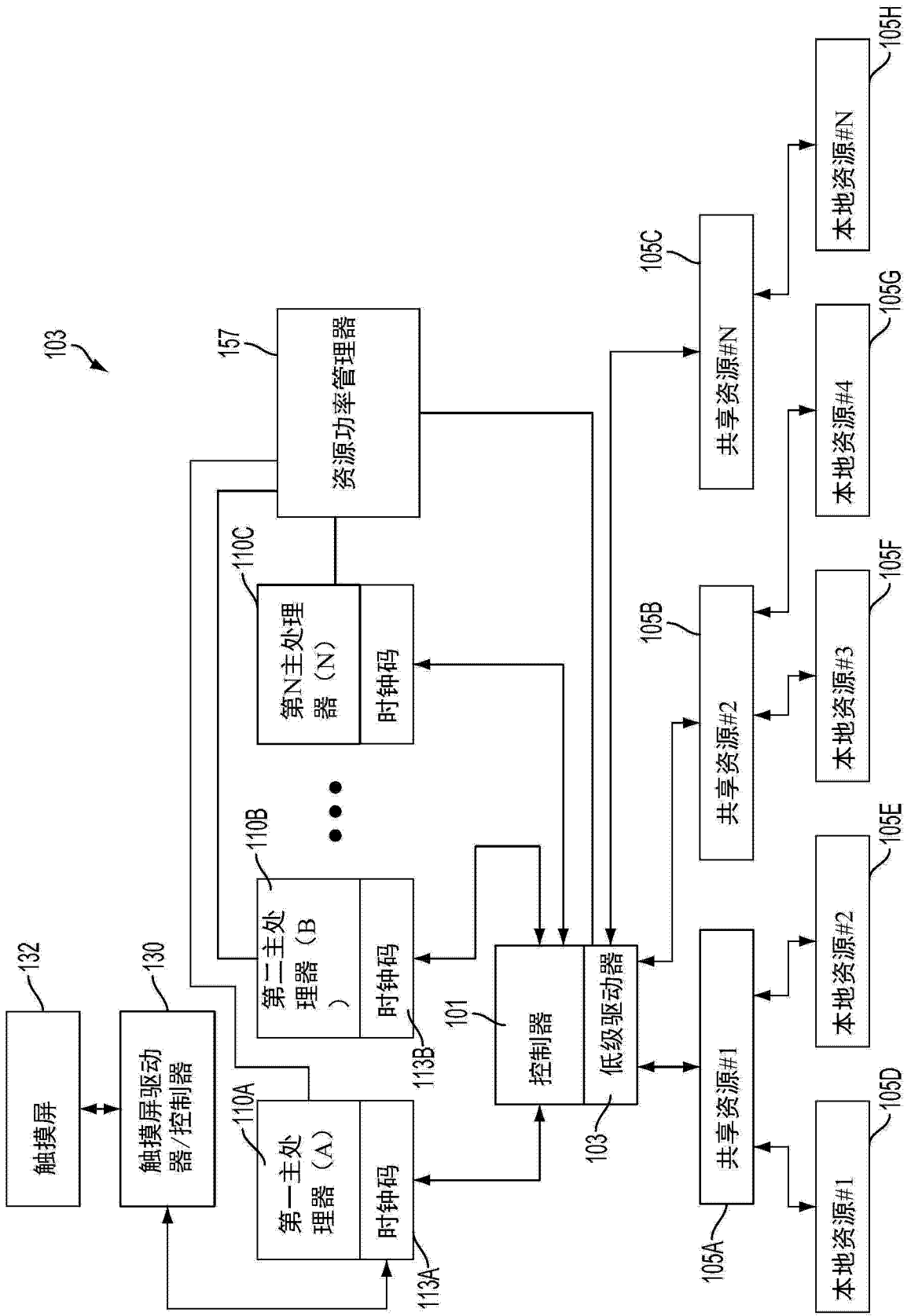

103
157
资源功率管理器
110C
第N主处理器（N）
时钟码
110B
第二主处理器（B）
时钟码
113B
110A
第一主处理器（A）
时钟码
113A
101
控制器
低级驱动器
103
132
触摸屏
130
触摸屏驱动器/控制器
105C
共享资源#N
105B
共享资源#2
105A
共享资源#1
105H
本地资源#N
105G
本地资源#4
105F
本地资源#3
105E
本地资源#2
105D
本地资源#1

图 14

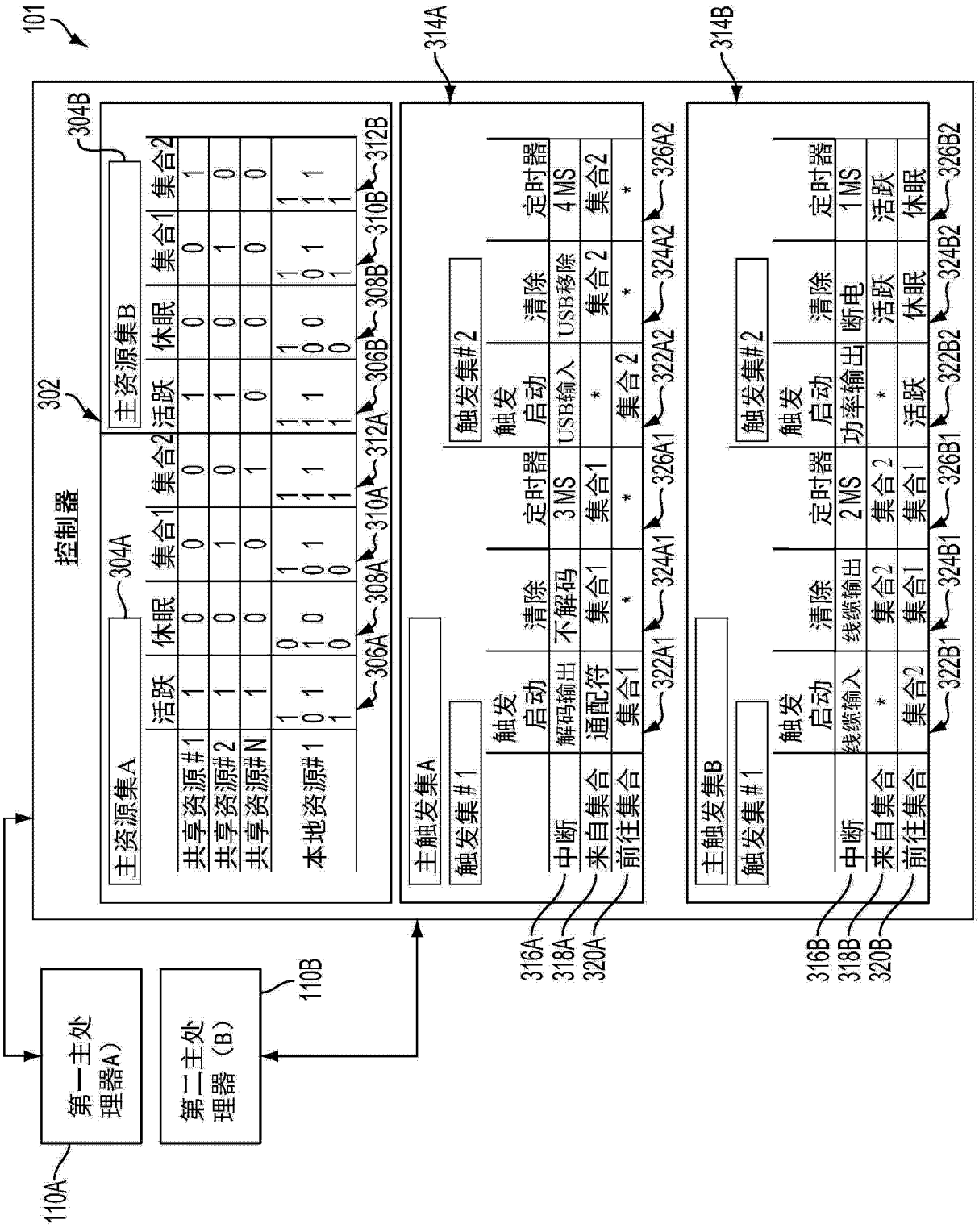

101
控制器
302
主资源集A
304A
活跃
休眠
集合1
集合2
共享资源#1
共享资源#2
共享资源#N
本地资源#1
306A
308A
310A
312A
主资源集B
304B
306B
308B
310B
312B
主触发集A
触发集#1
触发启动
清除
定时器
中断
来自集合
前往集合
解码输出
不解码
通配符
3 MS
USB输入
USB移除
4 MS
触发集#2
322A1
324A1
326A1
322A2
324A2
326A2
314A
316A
318A
320A
主触发集B
线缆输入
线缆输出
2 MS
功率输出
断电
1 MS
322B1
324B1
326B1
322B2
324B2
326B2
314B
316B
318B
320B
第一主处理器A)
第二主处理器（B）
110A
110B

图 15

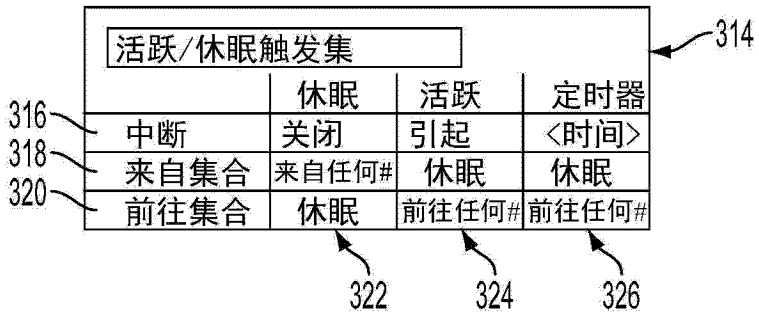

活跃/休眠触发集
314
休眠
活跃
定时器
316
中断
关闭
引起
<时间>
318
来自集合
来自任何#
休眠
休眠
320
前往集合
休眠
前往任何#
前往任何#
322
324
326

图 16

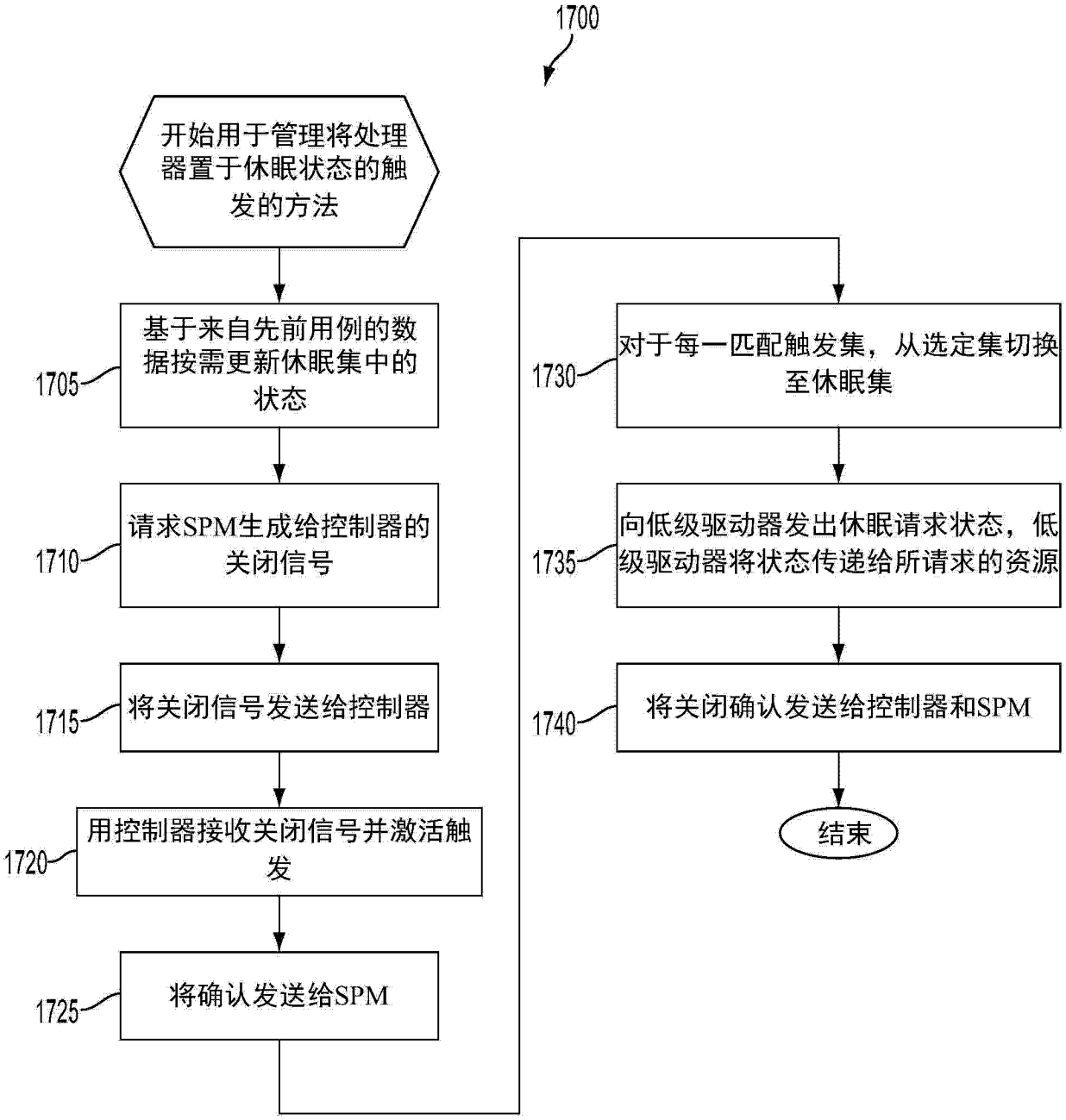

1700
开始用于管理将处理器置于休眠状态的触发的方法
1705
基于来自先前用例的数据按需更新休眠集中的状态
1710
请求SPM生成给控制器的关闭信号
1715
将关闭信号发送给控制器
1720
用控制器接收关闭信号并激活触发
1725
将确认发送给SPM
1730
对于每一匹配触发集，从选定集切换至休眠集
1735
向低级驱动器发出休眠请求状态，低级驱动器将状态传递给所请求的资源
1740
将关闭确认发送给控制器和SPM
结束

图 17

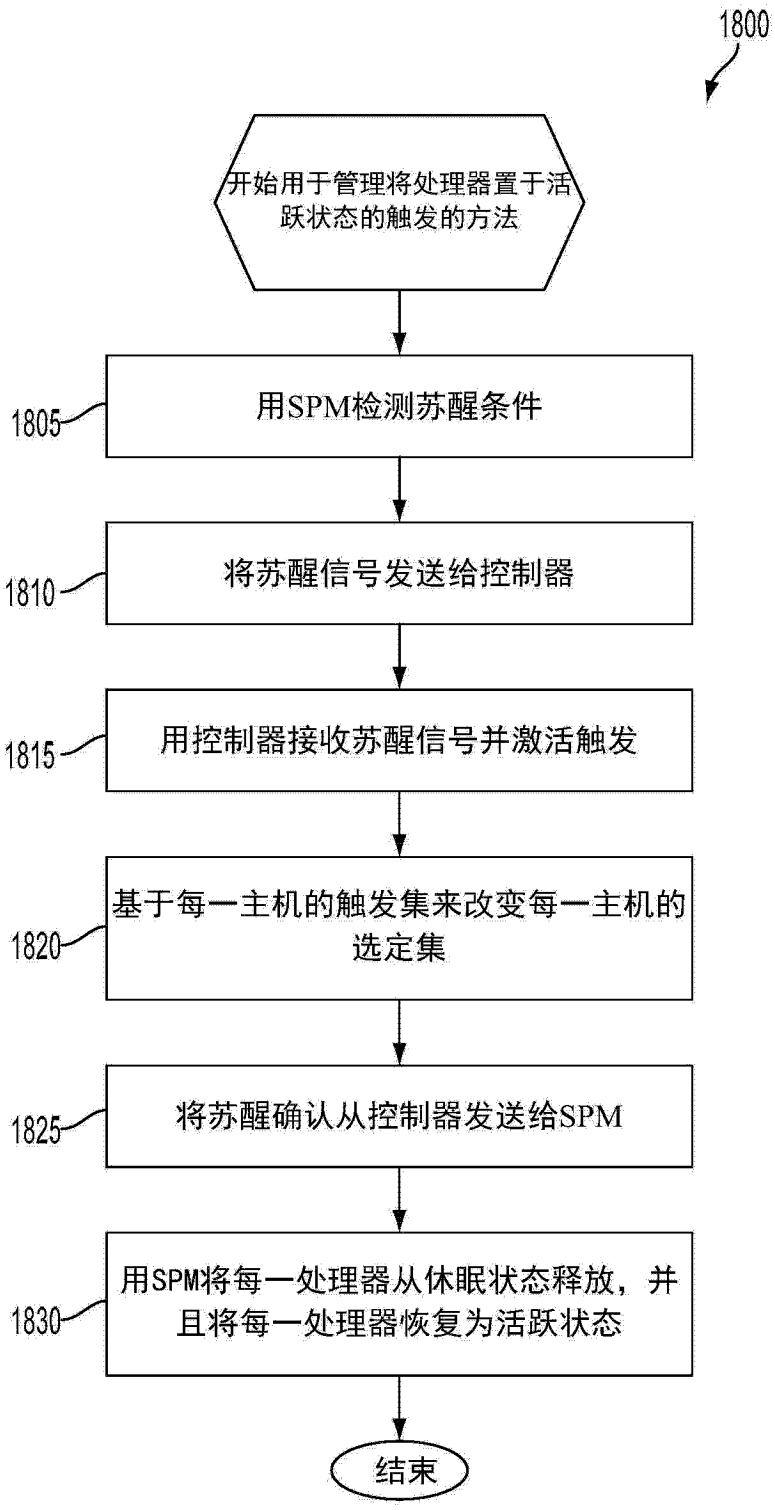

1800
开始用于管理将处理器置于活跃状态的触发的方法
1805
用SPM检测苏醒条件
1810
将苏醒信号发送给控制器
1815
用控制器接收苏醒信号并激活触发
1820
基于每一主机的触发集来改变每一主机的选定集
1825
将苏醒确认从控制器发送给SPM
1830
用SPM将每一处理器从休眠状态释放，并且将每一处理器恢复为活跃状态
结束

图 18

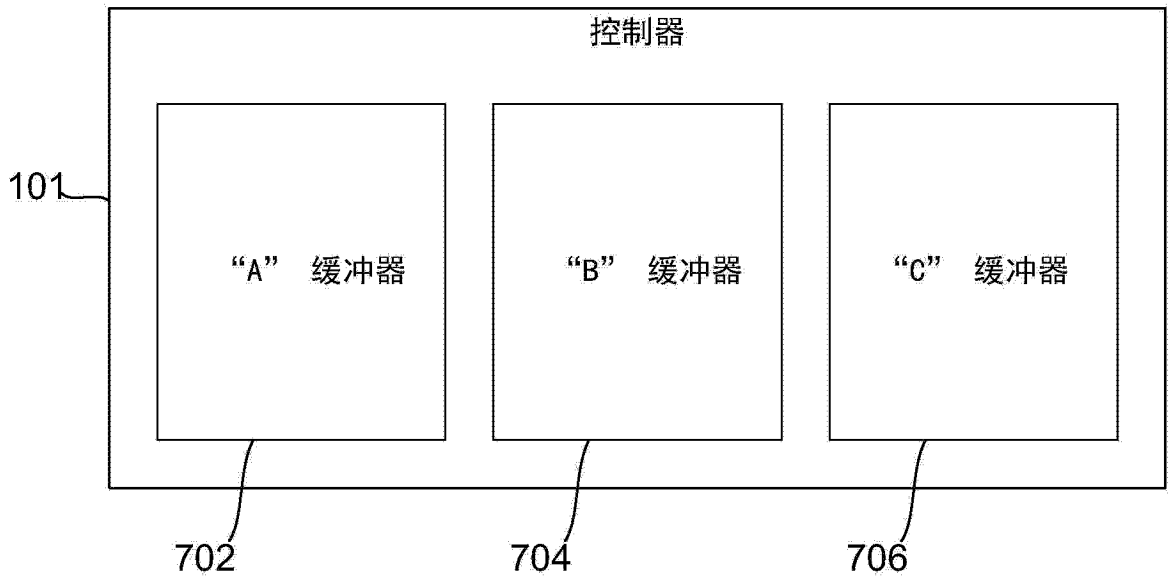


图 19

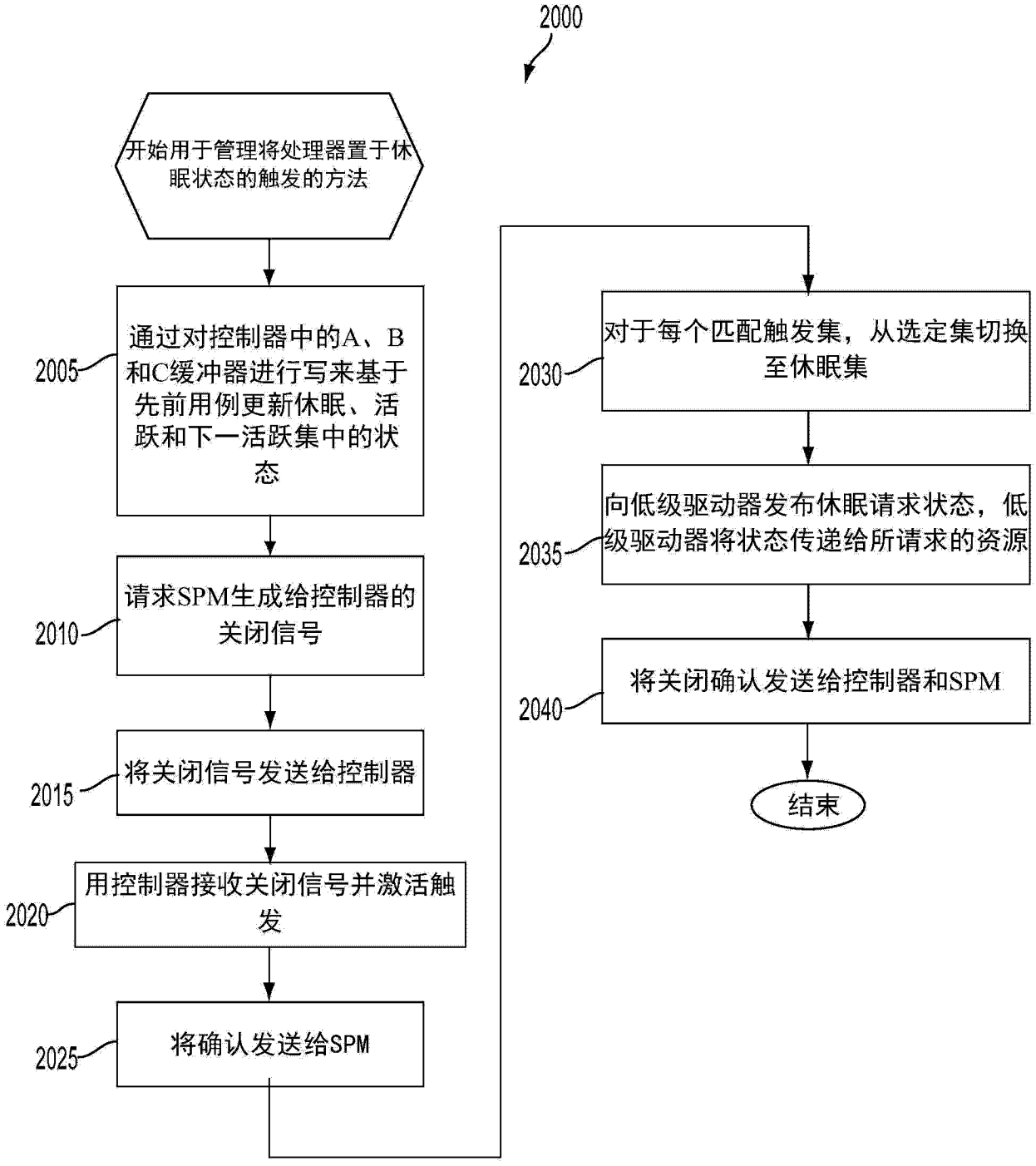

2000
开始用于管理将处理器置于休眠状态的触发的方法
2005
通过对控制器中的A、B和C缓冲器进行写来基于先前用例更新休眠、活跃和下一活跃集中的状态
2010
请求SPM生成给控制器的关闭信号
2015
将关闭信号发送给控制器
2020
用控制器接收关闭信号并激活触发
2025
将确认发送给SPM
2030
对于每个匹配触发集，从选定集切换至休眠集
2035
向低级驱动器发布休眠请求状态，低级驱动器将状态传递给所请求的资源
2040
将关闭确认发送给控制器和SPM
结束

图 20

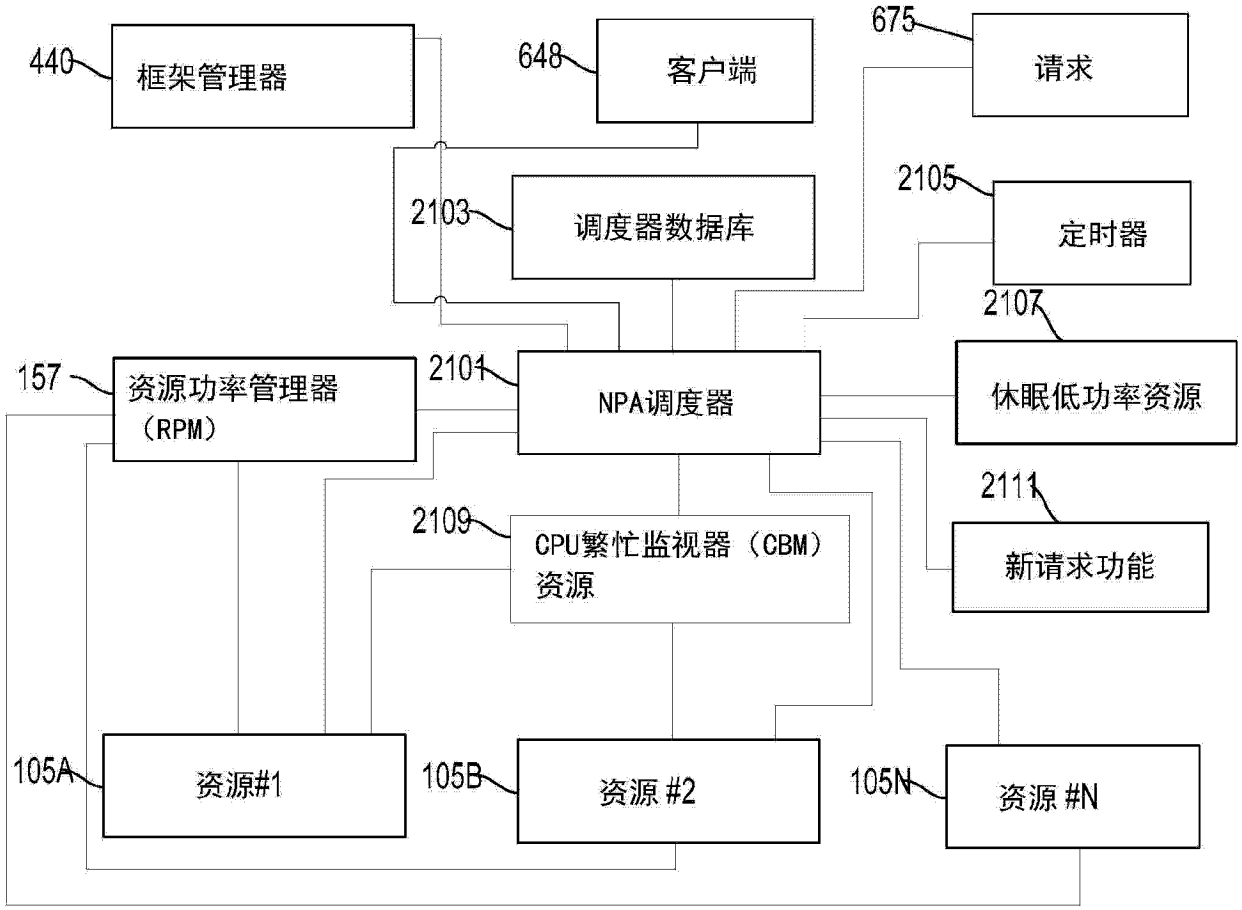
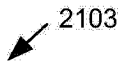

440
框架管理器
648
客户端
675
请求
2103
调度器数据库
2105
定时器
2107
休眠低功率资源
157
资源功率管理器
（RPM）
2101
NPA调度器
2111
新请求功能
2109
CPU繁忙监视器（CBM）
资源
105A
资源#1
105B
资源 #2
105N
资源 #N

图 21



线程配置数据 - 2205
调度器等待时间 - 2210
最小调度器Δ - 2215
请求等待时间 - 2220
分叉前瞻Δ - 2225
分叉等待时间 - 2230
交汇等待时间 - 2235
休眠苏醒转换等待时间 - 2240
时间队列等待时间 - 2245
LPR进入等待时间 - 2250
LPR退出等待时间 - 2255
LPR现在Δ -2260
调度资源定义： - 2265
经调度链表 -2270
请求状态 - 2275
请求时间 - 2280
启动时间 -2285
迟到概率 - 2287
回调通知 - 2290
最新通知状态 - 2293
请求迟滞 - 2295

图 22

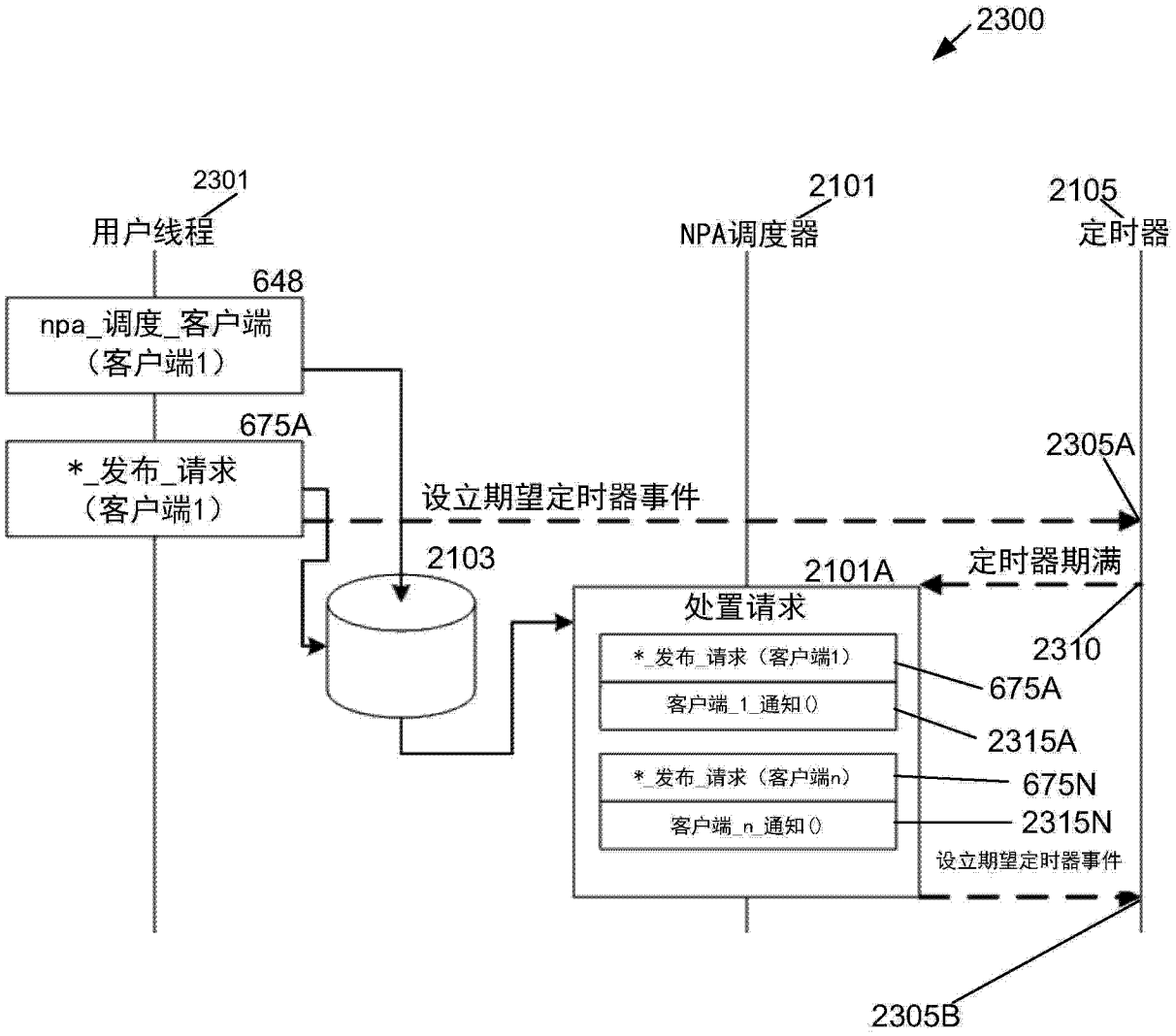

2300
2301
用户线程
2101
NPA调度器
2105
定时器
648
npa_调度_客户端
（客户端1）
675A
*_发布_请求
（客户端1）
设立期望定时器事件
2305A
2103
2101A
定时器期满
处置请求
*_发布_请求（客户端1）
客户端_1_通知()
2310
675A
2315A
*_发布_请求（客户端n）
客户端_n_通知()
675N
2315N
设立期望定时器事件
2305B

图 23

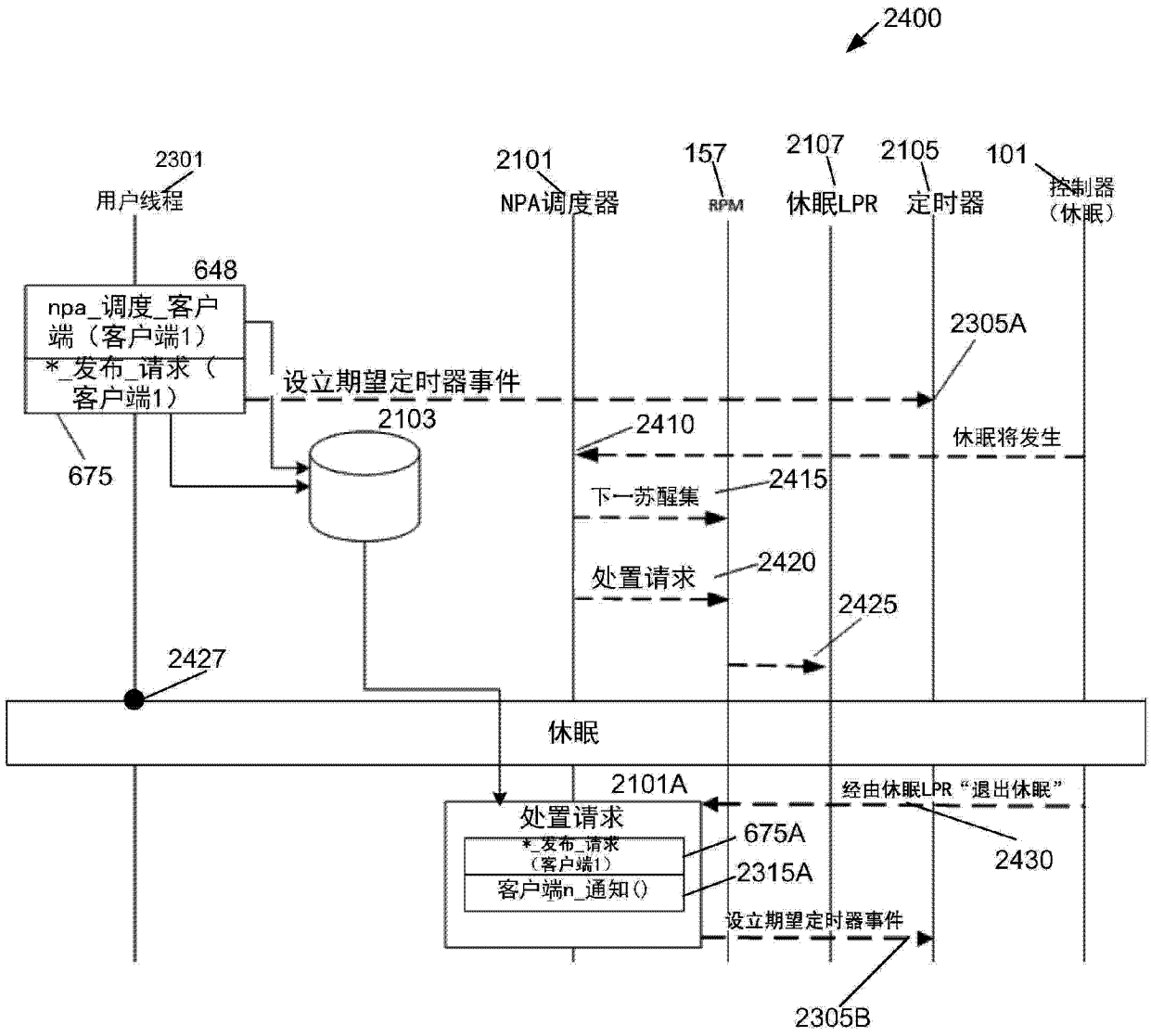

2400
2301
用户线程
2101
NPA调度器
157
RPM
2107
休眠LPR
2105
定时器
101
控制器（休眠）
648
npa_调度_客户端（客户端1）
*_发布_请求（客户端1）
675
设立期望定时器事件
2305A
2103
2410
休眠将发生
下一苏醒集
2415
处置请求
2420
2425
2427
休眠
2101A
处置请求
*_发布_请求（客户端1）
客户端n_通知()
675A
2315A
经由休眠LPR“退出休眠”
2430
设立期望定时器事件
2305B

图 24

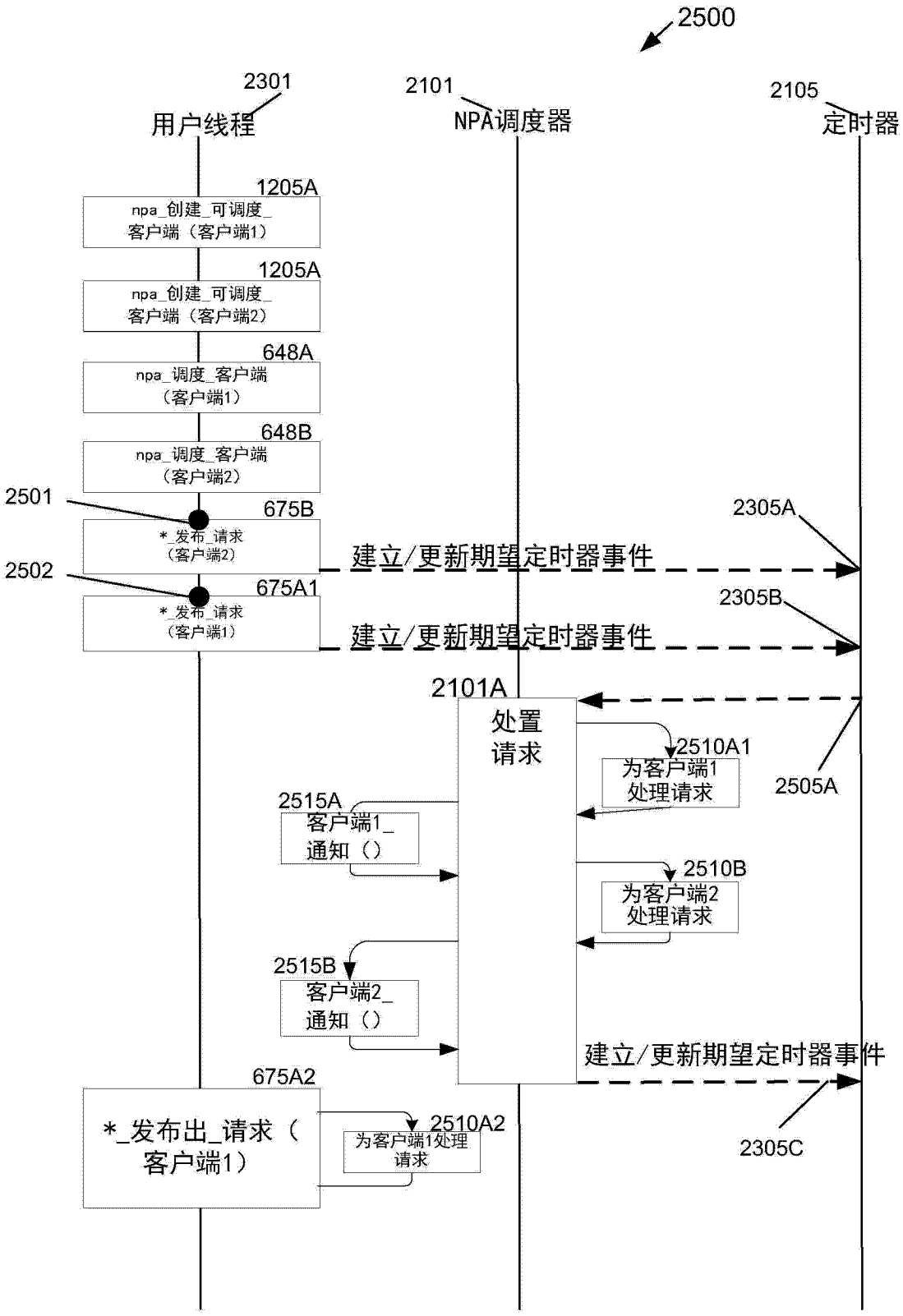

2500
2301
用户线程
2101
NPA调度器
2105
定时器
1205A
npa_创建_可调度_
客户端（客户端1）
1205A
npa_创建_可调度_
客户端（客户端2）
648A
npa_调度_客户端
（客户端1）
648B
npa_调度_客户端
（客户端2）
2501
675B
*_发布_请求
（客户端2）
2305A
建立/更新期望定时器事件
2502
675A1
*_发布_请求
（客户端1）
2305B
建立/更新期望定时器事件
2101A
处置
请求
2510A1
为客户端1
处理请求
2505A
2515A
客户端1_
通知（）
2510B
为客户端2
处理请求
2515B
客户端2_
通知（）
建立/更新期望定时器事件
675A2
*_发布出_请求（
客户端1）
2510A2
为客户端1处理
请求
2305C

图 25

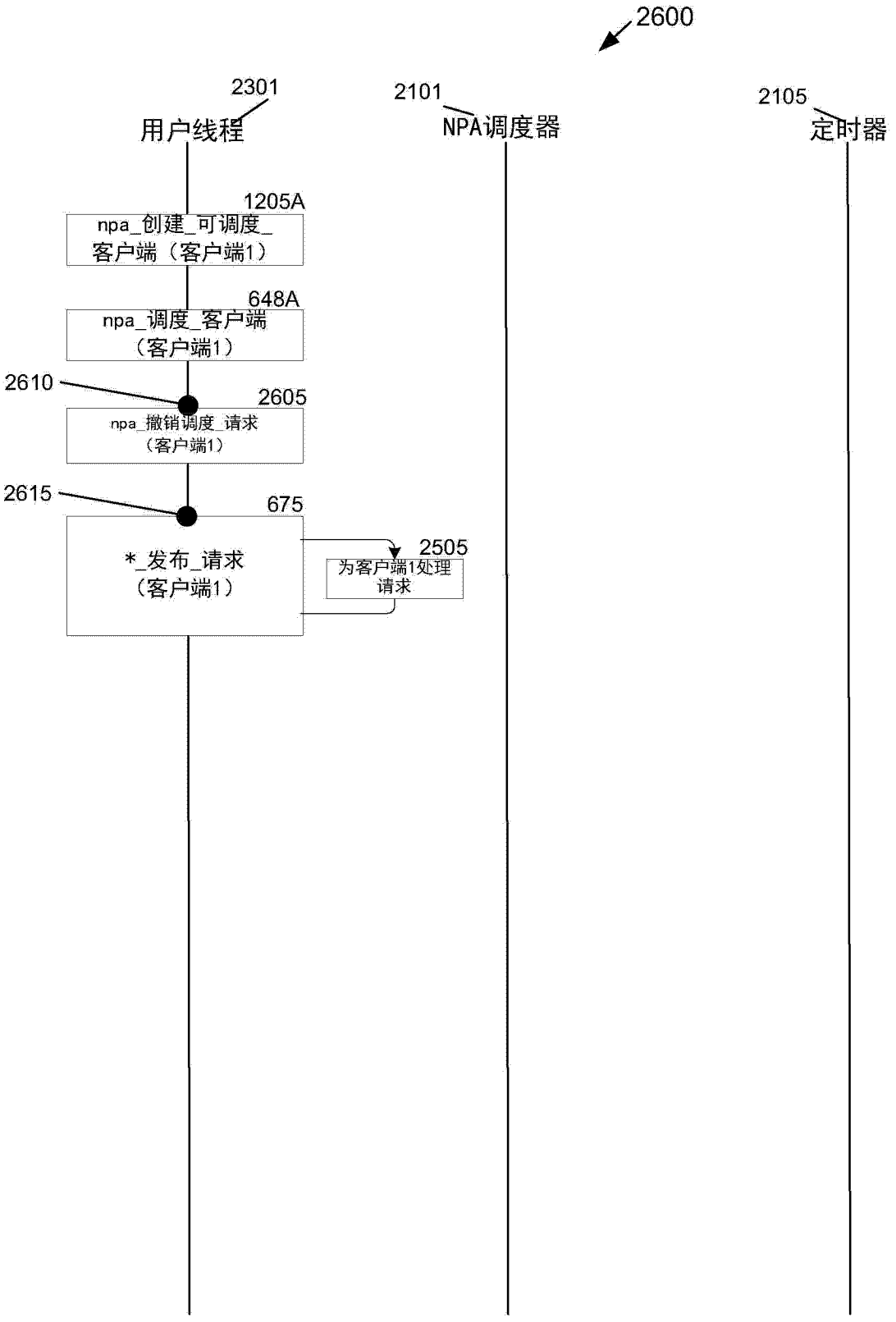

2600
2301
用户线程
2101
NPA调度器
2105
定时器
1205A
npa_创建_可调度_
客户端（客户端1）
648A
npa_调度_客户端
（客户端1）
2610
2605
npa_撤销调度_请求
（客户端1）
2615
675
*_发布_请求
（客户端1）
2505
为客户端1处理
请求

图 26

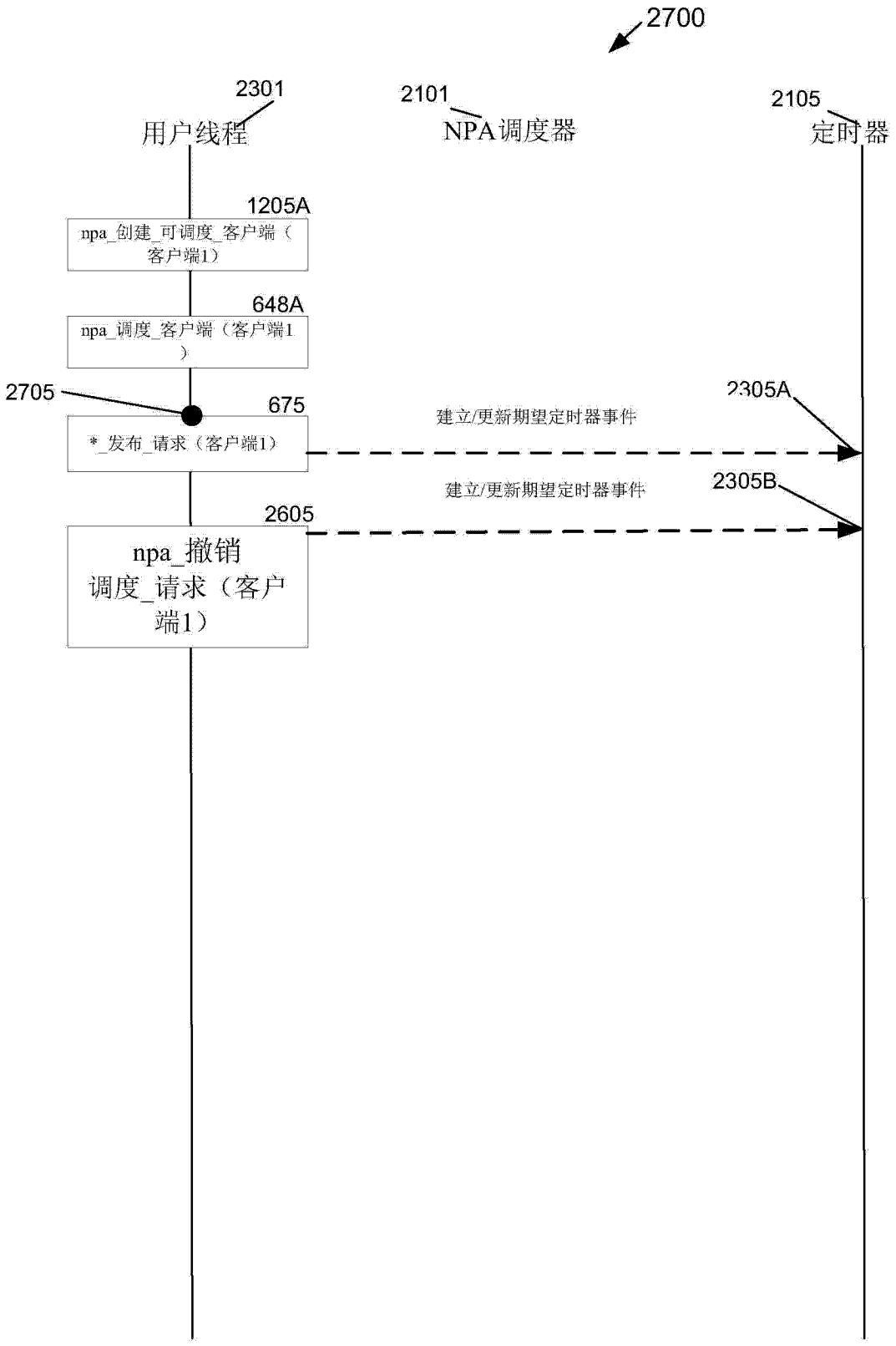

2700
2301
用户线程
2101
NPA调度器
2105
定时器
1205A
npa_创建_可调度_客户端（客户端1）
648A
npa_调度_客户端（客户端1）
2705
675
*_发布_请求（客户端1）
2305A
建立/更新期望定时器事件
2305B
建立/更新期望定时器事件
2605
npa_撤销调度_请求（客户端1）

图 27

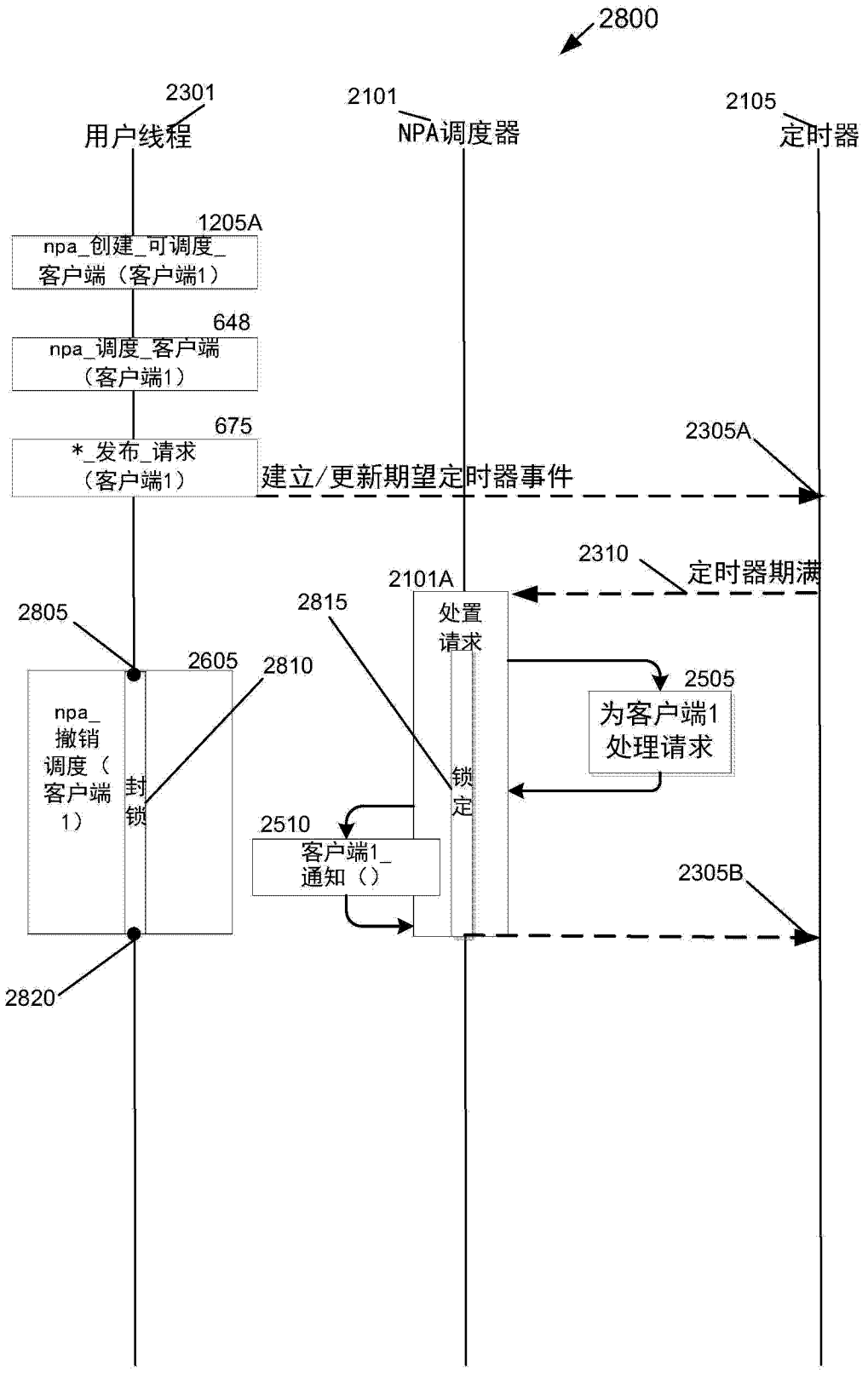

2800
2301
用户线程
2101
NPA调度器
2105
定时器
1205A
npa_创建_可调度_
客户端（客户端1）
648
npa_调度_客户端
（客户端1）
675
*_发布_请求
（客户端1）
2305A
建立/更新期望定时器事件
2310
定时器期满
2101A
处置
请求
2815
2805
2605
2810
2505
为客户端1
处理请求
npa_
撤销
调度（
客户端
1）
封
锁
锁
定
2510
客户端1_
通知（）
2305B
2820

图 28

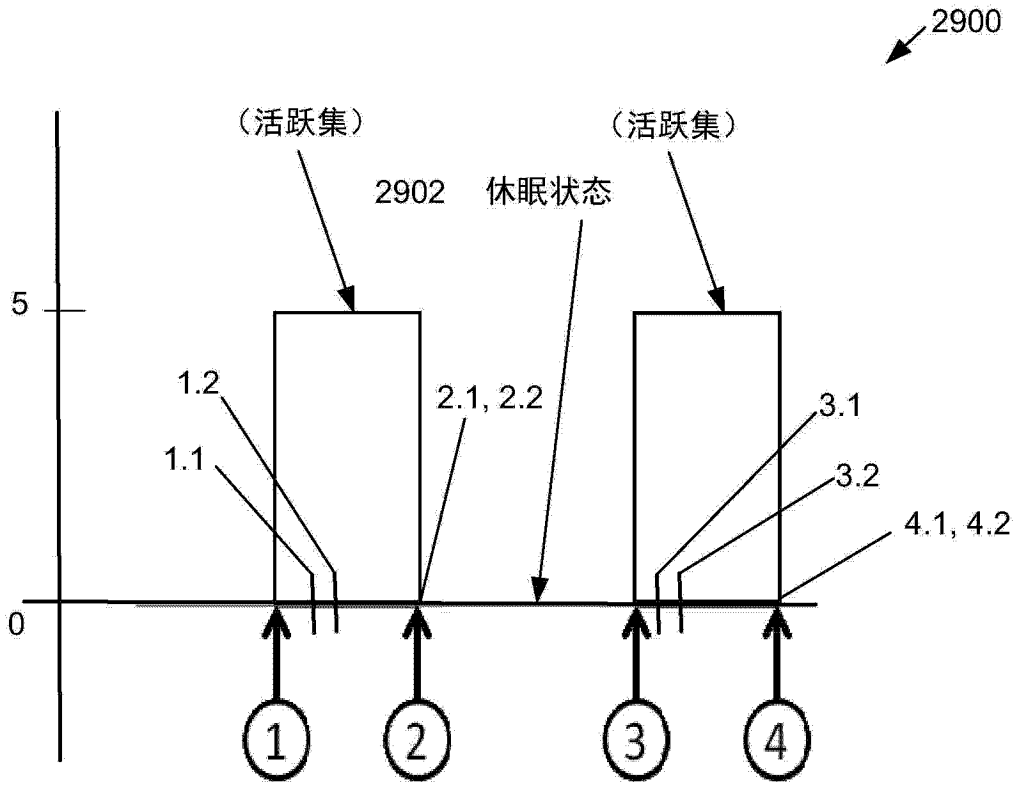


图 29

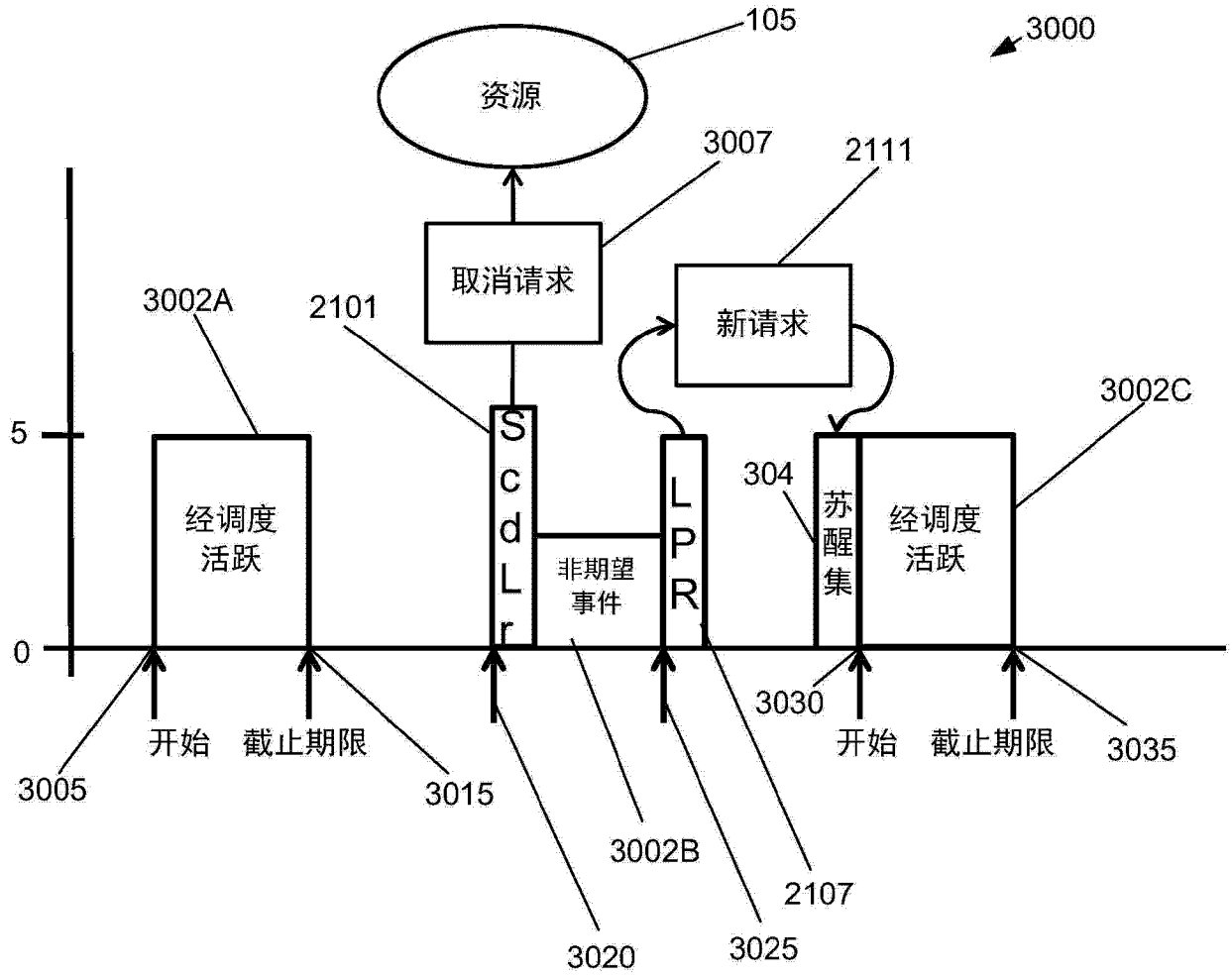

105
3000
资源
3007
2111
取消请求
新请求
3002A
2101
3002C
5
0
经调度
活跃
S
c
d
L
r
非期望
事件
L
P
R
304
苏
醒
集
经调度
活跃
3005
开始
截止期限
3015
3020
3002B
3025
2107
3030
开始
截止期限
3035

图 30