(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117950935 A
(43)申请公布日 2024.04.30

(21)申请号 202211345647.9

(22)申请日 2022.10.31

(71)申请人 荣耀终端有限公司
地址 518040 广东省深圳市福田区香蜜湖街道红荔西路8089号深业中城6号楼A单元3401

(72)发明人 相超 刘鹏程

(74)专利代理机构 北京润泽恒知识产权代理有限公司 11319
专利代理师 王洪

(51)Int.Cl.
G06F 11/30(2006.01)

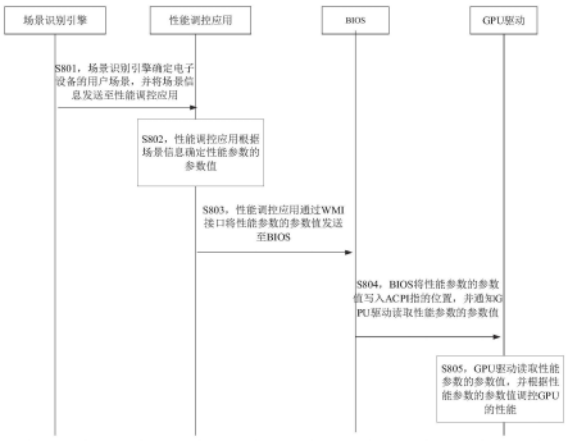
权利要求书2页 说明书25页 附图10页

(54)发明名称

一种性能调控方法及电子设备

(57)摘要

本申请提供了一种性能调控方法及电子设备，涉及计算机技术领域。通过该方案，可以根据电子设备当前的用户场景确定电子设备的性能需求，根据电子设备的性能需求调节电子设备的性能参数，该性能参数包括CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗和所述GPU的DB功耗中的至少一种，使显卡根据当前的TDP、TPP、TGP和/或DB功耗进行性能调控，从而可以根据场景提供更合理的CPU和GPU性能，不仅更好地满足用户在各种场景下的性能需求，而且可以相对地提高设备的续航，从而提高用户体验。



CN 117950935 A

1.一种性能调控方法,其特征在于,应用于电子设备,所述电子设备包括中央处理器CPU和图形处理器GPU,所述方法包括:

响应于用户操作启动第一应用,在启动所述第一应用之前,所述电子设备的第一性能参数为第一参数值,所述第一性能参数包括所述CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗、所述GPU的DB功耗中的至少一种;

启动所述第一应用之后,在第一时间点,所述电子设备的第一性能参数调节为第二参数值,所述第二参数值和所述第一参数值不同;

响应于用户操作,关闭所述第一应用,启动第二应用;

启动所述第二应用之后,在第二时间点,将所述电子设备的第一性能参数调节为第三参数值,所述第三参数值和所述第二参数值不同。

2.根据权利要求1所述的性能调控方法,其特征在于,所述方法还包括:

响应于用户操作,启动第三应用,在启动所述第三应用之前,所述电子设备的第二性能参数为第四参数值,所述第二性能参数包括所述GPU的Dx值;

启动所述第三应用之后,在第三时间点,将所述电子设备的第二性能参数调节为第五参数值,所述第五参数值与所述第四参数值不同;

响应于用户操作,关闭所述第三应用,启动第四应用;

启动所述第四应用之后,在第四时间点,将所述电子设备的第二性能参数调节为第六参数值,所述第六参数值和所述第五参数值不同。

3.根据权利要求1所述的性能调控方法,其特征在于,所述方法还包括:

在第五时间点之前,所述电子设备的第二性能参数为第七参数值,所述第二性能参数包括所述GPU的Dx值,所述第五时间点在所述第一时间点和/或所述第二时间点之后;

在第五时间点,将所述电子设备的第二性能参数调节为第八参数值,所述第八参数值与所述第七参数值不同;

所述电子设备在第五时间点的剩余电池电量小于在所述第一时间点和/或所述第二时间点的剩余电池电量,在所述第五时间点所述电子设备未插电;或者

所述电子设备在第五时间点的温度大于在所述第一时间点和/或所述第二时间点的温度。

4.根据权利要求1-3中的任意一项所述的性能调控方法,其特征在于,所述方法还包括:

根据所述第应用确定所述电子设备所处的用户场景,根据所述电子设备所处的用户场景确定所述第二参数值的大小;

根据所述第二应用确定所述电子设备所处的用户场景,根据所述电子设备所处的用户场景确定所述第三参数值的大小。

5.根据权利要求1所述的性能调控方法,其特征在于,所述方法还包括:

将所述第二参数值发送至GPU驱动,所述GPU驱动根据所述第二参数值控制所述GPU的性能;

将所述第三参数值发送至所述GPU驱动,所述GPU驱动根据所述第三参数值控制所述GPU的性能。

6.根据权利要求5所述的性能调控方法,所述电子设备还包括BIOS和ACPI,

将所述第二参数值发送至GPU驱动,所述GPU驱动根据所述第二参数值控制所述GPU的性能,包括:

通过WMI接口将所述第二参数值传输至所述BIOS;

所述BIOS将所述所述第二参数值写入所述ACPI指定的第一性能参数存储位置,并通过所述GPU驱动读取所述第二参数值;

所述GPU驱动读取所述第二参数值,并根据所述第二参数值控制所述GPU的性能;

将所述第三参数值发送至GPU驱动,所述GPU驱动根据所述第三参数值控制所述GPU的性能,包括:

通过WMI接口将所述第三参数值传输至所述BIOS;

所述BIOS将所述所述第三参数值写入所述ACPI指定的第一性能参数存储位置,并通过所述GPU驱动读取所述第三参数值;

所述GPU驱动读取所述第三参数值,并根据所述第二参数值控制所述GPU的性能。

7.根据权利要求2所述的性能调控方法,其特征在于,所述方法还包括:

将所述第五参数值发送至GPU驱动,所述GPU驱动根据所述第五参数值控制所述GPU的性能;

将所述第六参数值发送至所述GPU驱动,所述GPU驱动根据所述第六参数值控制所述GPU的性能。

8.根据权利要求7所述的性能调控方法,所述电子设备还包括BIOS和ACPI,

将所述第五参数值发送至GPU驱动,所述GPU驱动根据所述第五参数值控制所述GPU的性能,包括:

通过WMI接口将所述第五参数值传输至所述BIOS;

所述BIOS将所述所述第五参数值写入所述ACPI指定的第二性能参数存储位置,并通过所述GPU驱动读取所述第五参数值;

所述GPU驱动读取所述第五参数值,并根据所述第五参数值控制所述GPU的性能;

将所述第六参数值发送至GPU驱动,所述GPU驱动根据所述第六参数值控制所述GPU的性能,包括:

通过WMI接口将所述第六参数值传输至所述BIOS;

所述BIOS将所述所述第六参数值写入所述ACPI指定的第二性能参数存储位置,并通过所述GPU驱动读取所述第六参数值;

所述GPU驱动读取所述第六参数值,并根据所述第六参数值控制所述GPU的性能。

9.一种电子设备,其特征在于,包括:存储器和处理器,所述存储器和所述处理器耦合;所述存储器存储有程序指令,所述程序指令由所述处理器执行时,使得所述电子设备执行如权利要求1-8中任意一项所述的性能调控方法。

10.一种计算机可读存储介质,其特征在于,包括计算机程序,其特征在于,当所述计算机程序在电子设备上运行时,使得所述电子设备执行如权利要求1-8中任意一项所述的性能调控方法。

一种性能调控方法及电子设备

技术领域

[0001] 本申请涉及智能终端技术领域,尤其涉及一种性能调控方法及电子设备。

背景技术

[0002] 随着电子技术的不断发展,笔记本电脑或便携式电脑(Laptop)等电子设备作为人们日常生活工作中的常用设备得到了广泛发展。目前市场上存在各种各样配置的笔记本电脑,例如许多笔记本电脑为了提供更高的性能都配置了独立显卡。独立显卡的增加一般意味着设备功耗的增加,而鉴于笔记本电脑的移动属性,续航一直都是用户关注焦点。如何优化笔记本电脑或便携式电脑(Laptop)等电子设备的性能、续航、发热等一直是用户关注的焦点。

发明内容

[0003] 为了解决上述技术问题,本申请提供一种参性能调控方法及电子设备。在该性能调控方法中,可以根据电子设备当前的用户场景调节电子设备的性能参数,使显卡根据当前的TDP、TPP、TGP和/或DB功耗进行性能调控,从而可以根据场景提供更合理的CPU和GPU性能,不仅更好地满足用户在各种场景下的性能需求,而且可以相对地提高设备的续航,从而提高用户体验。

[0004] 第一方面,本申请提供一种性能调控方法,应用于电子设备,所述电子设备包括中央处理器CPU和图形处理器GPU,所述方法包括:

[0005] 响应于用户操作启动第一应用,在启动所述第一应用之前,所述电子设备的第一性能参数为第一参数值,所述第一性能参数包括所述CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗、所述GPU的DB功耗中的至少一种;

[0006] 启动所述第一应用之后,在第一时间点,所述电子设备的第一性能参数调节为第二参数值,所述第二参数值与所述第一参数值不同;

[0007] 响应于用户操作,关闭所述第一应用,启动第二应用;

[0008] 启动所述第二应用之后,在第二时间点,将所述电子设备的第一性能参数调节为第三参数值,所述第三参数值与所述第二参数值不同。

[0009] 根据第一方面,可以根据电子设备当前运行的应用确定电子设备的性能需求,根据电子设备的性能需求调节电子设备的第一性能参数,该第一性能参数包括CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗和所述GPU的DB功耗中的至少一种,使显卡根据当前的TDP、TPP、TGP和/或DB功耗进行性能调控,从而可以根据场景提供更合理的CPU和GPU性能,不仅更好地满足用户在各种场景下的性能需求,而且可以相对地提高设备的续航,从而提高用户体验。

[0010] 根据第一方面,或者以上第一方面的任意一种实现方式,所述方法还包括:

[0011] 响应于用户操作,启动第三应用,在启动所述第三应用之前,所述电子设备的第二性能参数为第四参数值,所述第二性能参数包括所述GPU的Dx值;

[0012] 启动所述第三应用之后，在第三时间点，将所述电子设备的第二性能参数调节为第五参数值，所述第五参数值与所述第四参数值不同；

[0013] 响应于用户操作，关闭所述第三应用，启动第四应用；

[0014] 启动所述第四应用之后，在第四时间点，将所述电子设备的第二性能参数调节为第六参数值，所述第六参数值和所述第五参数值不同。

[0015] 如此设置，可以根据电子设备当前的用户场景确定电子设备的性能需求，根据电子设备的性能需求调节电子设备的第二性能参数，该第二性能参数包括GPU的Dx值，使显卡根据当前Dx值对应的功耗进行性能调控，从而可以根据场景提供更合理的GPU性能，不仅更好地满足用户在各种场景下的性能需求，而且可以相对地提高设备的续航，从而提高用户体验。

[0016] 根据第一方面，或者以上第一方面的任意一种实现方式，所述方法还包括：

[0017] 在第五时间点之前，所述电子设备的第二性能参数为第七参数值，所述第二性能参数包括所述GPU的Dx值，所述第五时间点在所述第一时间点和/或所述第二时间点之后；

[0018] 在第五时间点，将所述电子设备的第二性能参数调节为第八参数值，所述第八参数值与所述第七参数值不同；

[0019] 所述电子设备在第五时间点的剩余电池电量小于在所述第一时间点和/或所述第二时间点的剩余电池电量，在所述第五时间点所述电子设备未插电；或者

[0020] 所述电子设备在第五时间点的温度大于在所述第一时间点和/或所述第二时间点的温度。

[0021] 如此设置可以在电子设备温度过高或电量不足时通过限制GPU功耗来降低设备发热或电量消耗，提高设备续航。

[0022] 根据第一方面，或者以上第一方面的任意一种实现方式，所述方法还包括：

[0023] 根据所述第一应用确定所述电子设备所处的用户场景，根据所述电子设备所处的用户场景确定所述第二参数值的大小；

[0024] 根据所述第二应用确定所述电子设备所处的用户场景，根据所述电子设备所处的用户场景确定所述第三参数值的大小。

[0025] 如此设置，可以根据电子设备当前的用户场景调节电子设备的性能参数，使显卡根据当前的TDP、TPP、TGP和/或DB功耗进行性能调控，从而可以根据场景提供更合理的CPU和GPU性能，不仅更好地满足用户在各种场景下的性能需求，而且可以相对地提高设备的续航，从而提高用户体验。

[0026] 根据第一方面，或者以上第一方面的任意一种实现方式，所述方法还包括：

[0027] 将所述第二参数值发送至GPU驱动，所述GPU驱动根据所述第二参数值控制所述GPU的性能；

[0028] 将所述第三参数值发送至所述GPU驱动，所述GPU驱动根据所述第三参数值控制所述GPU的性能。

[0029] 如此设置，可以使显卡根据当前的TDP、TPP、TGP和/或DB功耗进行性能调控，从而可以根据场景提供更合理的CPU和GPU性能，不仅更好地满足用户在各种场景下的性能需求，而且可以相对地提高设备的续航，从而提高用户体验。

[0030] 根据第一方面，或者以上第一方面的任意一种实现方式，所述电子设备还包括

BIOS和ACPI，

[0031] 将所述第二参数值发送至GPU驱动，所述GPU驱动根据所述第二参数值控制所述GPU的性能，包括：

[0032] 通过WMI接口将所述第二参数值传输至所述BIOS；

[0033] 所述BIOS将所述所述第二参数值写入所述ACPI指定的第一性能参数存储位置，并通过所述GPU驱动读取所述第二参数值；

[0034] 所述GPU驱动读取所述第二参数值，并根据所述第二参数值控制所述GPU的性能；

[0035] 将所述第三参数值发送至GPU驱动，所述GPU驱动根据所述第三参数值控制所述GPU的性能，包括：

[0036] 通过WMI接口将所述第三参数值传输至所述BIOS；

[0037] 所述BIOS将所述所述第三参数值写入所述ACPI指定的第一性能参数存储位置，并通过所述GPU驱动读取所述第二参数值；

[0038] 所述GPU驱动读取所述第三参数值，并根据所述第三参数值控制所述GPU的性能。

[0039] 如此设置可以通过WMI接口和BIOS将第一性能参数的当前参数值发送至GPU驱动，保证GPU驱动稳定及时获取当前参数值，并根据当前参数值进行性能调控。

[0040] 根据第一方面，或者以上第一方面的任意一种实现方式，所述方法还包括：

[0041] 将所述第五参数值发送至GPU驱动，所述GPU驱动根据所述第五参数值控制所述GPU的性能；

[0042] 将所述第六参数值发送至所述GPU驱动，所述GPU驱动根据所述第六参数值控制所述GPU的性能。

[0043] 如此设置，可以使显卡根据当前的Dx值进行性能调控，从而可以根据场景提供更合理的CPU和GPU性能，不仅更好地满足用户在各种场景下的性能需求，而且可以相对地提高设备的续航，从而提高用户体验。

[0044] 根据第一方面，或者以上第一方面的任意一种实现方式，所述电子设备还包括BIOS和ACPI，

[0045] 将所述第五参数值发送至GPU驱动，所述GPU驱动根据所述第五参数值控制所述GPU的性能，包括：

[0046] 通过WMI接口将所述第五参数值传输至所述BIOS；

[0047] 所述BIOS将所述所述第五参数值写入所述ACPI指定的第二性能参数存储位置，并通过所述GPU驱动读取所述第三参数值；

[0048] 所述GPU驱动读取所述第五参数值，并根据所述第五参数值控制所述GPU的性能；

[0049] 将所述第六参数值发送至GPU驱动，所述GPU驱动根据所述第六参数值控制所述GPU的性能，包括：

[0050] 通过WMI接口将所述第六参数值传输至所述BIOS；

[0051] 所述BIOS将所述所述第六参数值写入所述ACPI指定的第二性能参数存储位置，并通过所述GPU驱动读取所述第六参数值；

[0052] 所述GPU驱动读取所述第六参数值，并根据所述第六参数值控制所述GPU的性能。

[0053] 如此设置可以通过WMI接口和BIOS将第二性能参数的当前参数值发送至GPU驱动，保证GPU驱动稳定及时获取当前参数值，并根据当前参数值进行性能调控。

[0054] 第二方面，本申请提供一种电子设备，包括：存储器和处理器，所述存储器和所述处理器耦合；所述存储器存储有程序指令，所述程序指令由所述处理器执行时，使得所述电子设备执行第一方面或第一方面的任意可能的实现方式中的性能调控方法。

[0055] 第三方面，本申请提供了一种计算机可读介质，用于存储计算机程序，该计算机程序包括用于执行第一方面或第一方面的任意可能的实现方式中的方法的指令。

[0056] 第四方面，本申请提供了一种计算机程序，该计算机程序包括用于执行第一方面或第一方面的任意可能的实现方式中的方法的指令。

[0057] 第五方面，本申请提供了一种芯片，该芯片包括处理电路、收发管脚。其中，该收发管脚、和该处理电路通过内部连接通路互相通信，该处理电路执行第一方面或第一方面的任一种可能的实现方式中的方法，以控制接收管脚接收信号，以控制发送管脚发送信号。

附图说明

[0058] 图1为示例性示出的目前一种温度监控方法的模块交互示意图；

[0059] 图2为示例性示出的电子设备的硬件结构示意图；

[0060] 图3为示例性示出的电子设备的软件结构示意图；

[0061] 图4为本申请实施例提供的软件模块间的交互示意图；

[0062] 图5为本申请实施例提供的一种信号交互示意图；

[0063] 图6为本申请实施例提供的一种界面图；

[0064] 图7为本申请实施例提供的又一种信号交互示意图；

[0065] 图8为本申请实施例提供的性能调控方法的流程图；

[0066] 图9为本申请实施例提供的性能调控方法的示例调控过程；

[0067] 图10为本申请实施例提供的性能调控方法的又一示例调控过程；

[0068] 图11示出了本申请实施例的一种装置的示意性框图。

具体实施方式

[0069] 下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

[0070] 本文中术语“和/或”，仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。

[0071] 本申请实施例的说明书和权利要求书中的术语“第一”和“第二”等是用于区别不同的对象，而不是用于描述对象的特定顺序。例如，第一目标对象和第二目标对象等是用于区别不同的目标对象，而不是用于描述目标对象的特定顺序。

[0072] 在本申请实施例中，“示例性的”或者“例如”等词用于表示作例子、例证或说明。本申请实施例中被描述为“示例性的”或者“例如”的任何实施例或设计方案不应被解释为比其它实施例或设计方案更优选或更具优势。确切而言，使用“示例性的”或者“例如”等词旨在以具体方式呈现相关概念。

[0073] 在本申请实施例的描述中，除非另有说明，“多个”的含义是指两个或两个以上。例

如,多个处理单元是指两个或两个以上的处理单元;多个系统是指两个或两个以上的系统。

[0074] 为了下述各实施例的描述清楚简洁,首先给出相关概念或技术的简要介绍:

[0075] 主板BIOS:Basic Input Output System,基本输入输出系统。BIOS是一组固化到计算机内主板上一个ROM芯片上的程序,它保存着计算机最重要的基本输入输出的程序、系统设置信息、开机上电自检程序和系统启动自举程序。BIOS主要功能是为计算机提供最底层的、最直接的硬件设置和控制。BIOS提供给操作系统的接口包括BS(Boot Services,启动服务)和RT(Runtime Services,运行时服务)。

[0076] 显卡BIOS:BIOS又称VGA BIOS或Video BIOS,主要用于存放显示芯片与驱动程序之间的控制程序,其控制着显卡各种工作状态,包括核心工作频率,显存工作频率,功耗限制、工作电压和显存时序等核心参数。显卡BIOS芯片用来保存显卡BIOS程序,和主板BIOS一样,显卡BIOS是储存在BIOS芯片中的,而不是储存在磁盘中。

[0077] NVAPI:NVAPI为NVIDIA(英伟达/辉达)厂商提供的供第三方调用的接口用于监控显卡的温度、性能参数等。

[0078] ACPI:Advanced Configuration and Power Interface,高级配置与电源接口,ACPI是系统硬件/固件(BIOS)与OS(操作系统)和OS应用程序之间的接口。换言之BIOS通过统一的接口提供操作系统到硬件的通讯,ACPI就是BIOS中这种统一接口的一种实现方式。ACPI为操作系统(OS)提供两类数据结构:数据表和定义块。它们是OS(操作系统)与BIOS/EFI固件进行交互的接口。数据表存储原始数据,并由设备驱动程序使用。定义块由解释程序可执行的字节码组成。其中的数据表部分由固件(BIOS/EFI)负责生成,用于描述系统硬件的各个方面的信息。ACPI接口中包含了很多预定义的表格,这些表格的定义存储在BIOS芯片中,由BIOS生成到内存中提交给操作系统。

[0079] ACPI Control Method:ACPI控制方法,类似于C语言中的函数,AML的函数叫做Method。根据ACPI的规范,BIOS提供了一些标准的Method给OS调用,例如_PTS,_WAK等等。控制方法定义OS如何执行一个简单的硬件任务。例如,OS调用控制方法(ControlMethod)去读取一个高温区的温度。

[0080] WMI:windows management instrumentation,Windows管理规范,为从计算机系统、网络或企业获取管理数据的任何本地或远程应用程序或脚本提供统一接口。该统一接口的设计使得WMI客户端应用程序和脚本不必调用各种各样的操作系统应用程序编程接口(API)。

[0081] NVIDIADynamic Boost(动态增强):是一项新技术,它可以更智能的为笔记本电脑的CPU和GPU分配功率,从而使笔记本电脑在有限的功耗前提下,提升功耗利用效率,从而提升笔记本电脑的整体性能。

[0082] Dx-Notify:NVIDIA(英伟达)显卡预留的性能控制接口,Dx-Notify支持D1-D5共5阶的性能控制,每一阶对应的GPU性能不同,可以在Video BIOS设置D1-D5对应的性能,D1性能最强,D5性能最弱。

[0083] SCI:System Control Interrupt,系统控制中断。专门用于ACPI电源管理的一个IRQ(中断请求),需要OS(操作系统)支持。

[0084] SMI:System Management Interrupt,系统管理中断,使用系统进入SMM(System Management Mode,系统管理模式)的特殊中断。

[0085] SMM:System Management Mode,系统管理模式,是一个对所有Intel处理器都统一的标准体系结构特性,系统管理模式(SMM)提供与传统IA-32架构中的系统管理中断(SMI)处理程序相同的执行环境。

[0086] TGP:Total Graphics Power,一般指整张显卡的性能/功耗。

[0087] TDP:CPU Thermal Design Power,散热设计功耗,其含义是“当处理器达到最大负荷的时候,所释放出的热量”(单位为瓦(W)),是反应一颗处理器热量释放的指标,应用上,TDP是反映处理器热量释放的指标,是电脑的冷却系统必须有能力驱散的最大热量限度。

[0088] TPP功耗:Total Processor Power,全处理器功耗,对于笔记本而言,一般特指CPU功耗+GPU功耗。

[0089] DB功耗:显卡的动态功耗,即在显卡的TGP功耗上进行动态调节的上下限,例如TGP功耗为40W,DB功耗为15W,则显卡驱动根据CPU当前功耗/实际功耗可以将显卡功耗限制在25W至55W之间。

[0090] 单烤/双烤:烤机就是电脑/计算机的稳定性测试,单烤就是软件(例如AIDA64Extreme)单独运行某个部件的压力测试,如GPU、FPU单烤,双烤就是软件让两种测试同时进行,一般指CPU和GPU满负荷工作。

[0091] 地址信息:用于表示预先申请的内存的位置信息,CPU/程序通过地址信息可以获知预先申请的内存的具体位置,将诸如GPU温度、CPU温度、主板温度、风扇转速等参数存储在所述内存中,地址信息的表示方法有多种,所述地址信息可以是内存的起始位置与长度,也可以是内存的起始位置和末位置,凡是可以用于表示内存的位置的地址信息均适用于本申请施例。

[0092] 焦点窗口(focus window),指拥有焦点的窗口。焦点窗口是唯一可以接收键盘输入的窗口。焦点窗口的确定方式与系统的焦点模式(focus mode)关联。焦点窗口的顶层窗口被称为活动窗口(active window)。同一时间只有一个窗口可以是活动窗口。焦点窗口大概率为用户当前需要使用的窗口。

[0093] 焦点模式,可用于决定鼠标如何使一个窗口获得焦点。一般地,焦点模式可包括三种,分别为:

[0094] (1)点击聚焦(click-to focus),在这种模式下,鼠标点击的窗口即可获得焦点。即当鼠标点击一个可以获得焦点的窗口的任意位置,即可激活该窗口,该窗口便被置于所有窗口的最前面,并接收键盘输入。当鼠标点击其他窗口时,该窗口会失去焦点。

[0095] (2)焦点跟随鼠标(focus-follow-mouse),在这种模式下,鼠标下的窗口可以获取焦点。即当鼠标移到一个可以获得焦点的窗口的范围内,用户不需要点击窗口的某个地方就可以激活这个窗口,接收键盘输入,但该窗口不一定被置于所有窗口的最前面。当鼠标移出这个窗口的范围时,这个窗口也会随之失去焦点。

[0096] (3)草率聚焦(sloppy focus),这种焦点模式与focus-follow-mouse比较类似:当鼠标移到一个可以获得焦点的窗口的范围内,用户不需要点击窗口的某个地方就可以激活这个窗口,接收键盘输入,但该窗口不一定被置于所有窗口的最前面。与focus-follow-mouse不同的是,当鼠标移出这个窗口范围时,焦点并不会随之改变,只有当鼠标移动到别的可以接收焦点的窗口时,系统焦点才改变。

[0097] 进程包括多个线程,线程可以创建窗口。焦点进程为创建焦点窗口的线程所属的

进程。

[0098] 笔记本电脑或便携式电脑(Laptop)的所有芯片或处理器中,CPU(中央处理单元)和GPU(图形处理单元)是功耗最大,性能最难控制的芯片,如何控制CPU、GPU的性能、功耗、发热一直笔记本电脑或便携式电脑(Laptop)性能调控的难点,本申请实施例的目的使基于诸如笔记本电脑等电子设备的用户场景控制显卡性能和功耗,从而实现对整机性能和功耗的控制。

[0099] 图1为示例性示出的目前一种性能调控方法的模块交互示意图。在图1所示示例中,以调节NV(即NVIDIA)显卡的Dx值(即Dx-Notify中的Dx值)为例说明目前的显卡性能调控流程。

[0100] 在本申请中显卡、GPU、图形处理单元或图形处理器具有相同的含义,表示设备中的相同部件。显卡可以分为集成显卡和独立显卡。在本申请实施例中以调节独立显卡的性能为例进行说明。但应当理解的是,本申请实施例公开的性能调控方法也适用于集成显卡。此外,在本申请实施例中以NV显卡为例进行说明,但是本申请实施例公开的性能调控方法也适用于其它显卡。

[0101] 如图1所示,目前电子设备的显卡性能调控方法包括:

[0102] S101,控制器(EC)从显卡外部的温度传感器读取显卡温度。

[0103] 示例性地,控制器(EC)通过例如系统管理总线(“SMBus”)与显卡外部的温度传感器通信连接,并通过系统管理总线(“SMBus”)对温度传感器或其它设备进行轮询来从显卡外部的温度传感器读取独显卡温度。

[0104] S102,控制器(EC)根据显卡温度确定显卡的Dx值。

[0105] 控制器根据预先设定的温度与Dx值的关系,基于显卡当前温度确定显卡当前的Dx值。例如在显卡当前温度超过D4对应的上限温度时,控制器(EC)确定将显卡Dx值调节为D5,以降低显卡性能,从而降低显卡温度。

[0106] S103,控制器(EC)通知BIOS切换显卡Dx值。

[0107] 示例性地,控制器(EC)通过62/66IO端口传递数据。BIOS和控制器(EC)预先定义SMI(System Management Interrupt,系统管理中断)中断号,BIOS或者说更具体地BIOS运行时服务通过62/66IO端口向控制器(EC)获取中断号。控制器(EC)确定显卡的Dx值后会产生SCI中断,BIOS运行时服务调用中断号对应的中断服务函数再次通过62/66IO端口读取显卡Dx值。

[0108] S104,BIOS修改Dx值,并通知显卡驱动读取Dx值。

[0109] 示例性地,ACPI中指定了内部存储器(即内存)中存储Dx值的存储位置,BIOS读取到Dx值后将该存储位置Dx数值修改为最新数值,然后通知显卡驱动读取最新的Dx值。

[0110] S105,显卡驱动读取Dx值,并根据Dx值对应的功耗控制显卡。

[0111] 具体地,BIOS在修改Dx值后会产生SCI中断,显卡驱动接收到SCI中断(也即事件通知)后调用显卡控制方法(ACPI中定义的显卡control method)读取Dx的当前数值。

[0112] 当读取到Dx的当前数值后,则根据Dx当前数值对应的功耗控制显卡性能。示例性例如当前Dx值为D3,对应的显卡功耗为15W,则显卡驱动控制显卡将其最大功耗限制在15W。

[0113] 至于Dx的当前数值对应的具体功耗可以从显卡BIOS中获取。

[0114] 虽然上述显卡性能调控方法可以调控显卡性能,但是只会根据显卡的温度被动进

行显卡性能控制,并不会根据场景和需求主动进行显卡性能控制,很多情况下都是设备或显卡温度过高后才进行性能调控,以降低温度,这给用户的体验并不是很好。此外,Dx控制显卡上限值只支持5阶,调节范围比较固定,无法根据具体场景调节显卡的性能参数,这导致部分场景无法给用户提供更好的性能。例如在对CPU性能需要较大的场景(例如编程)中,根据显卡温度通过Dx调节显卡性能可能导致虽然显卡性能需求降低,但是仍然占用较大的功耗指标,这导致笔记本等设备无法提供足够的CPU性能。

[0115] 此外,目前一般常用性能调控方法还有在设备设计时根据散热和性能等条件,在BIOS中直接设定TPP&TGP&TDP的数值,显卡在工作过程中根据TPP/TGP/TDP设定数值,自动调整性能或者当游戏/GPU单烤/双烤等需要GPU大性能情况显卡驱动自动调整性能,但在此过程中TPP/TGP/TDP参数并不会进行改变,换言之CPU、显卡性能参数不会根据场景变化,只会根据芯片的功耗进行性能改变,这种方法无法适应不同场景对CPU和显卡不同的性能需求。

[0116] 此外,目前还有通过WMI接口设置显卡的TPP和TGP来进行显卡性能控制的方法,这种方法同样TPP、TGP和DB值直接设定,不会根据电子设备当前所处的用户场景动态调整,适用范围较小,并且显卡性能控制精确度会受CPU实际功率影响,并不稳定。

[0117] 基于上述原因,本申请实施例提供了一种性能调控方法,通过对设备当前的场景进行识别,当需要进行性能调控时根据具体场景设定TPP、TGP和/或TDP的具体参数或Dx值,使显卡根据当前的TPP、TGP和/或TDP参数或Dx值进行性能调控,从而可以根据场景提供更合理的CPU和GPU性能,不仅更好地满足用户在各种场景下的性能需求,而且可以相对地提高设备的续航,从而提高用户体验。下面对本申请实施例提供的性能调控方法进行详细说明。

[0118] 图2为示例性示出的根据本申请实施例的电子设备的硬件结构示意图。应该理解的是,图2所示电子设备100仅是电子设备的一个范例,并且电子设备100可以具有比图中所示的更多的或者更少的部件,可以组合两个或多个的部件,或者可以具有不同的部件配置。图2中所示出的各种部件可以在包括一个或多个信号处理和/或专用集成电路在内的硬件、软件、或硬件和软件的组合中实现。

[0119] 电子设备100可以包括:处理器110,内部存储器122,外部存储器121,嵌入式控制器(EC)130、独立图形处理器140、BIOS固件闪存150,天线1,无线通信模块160,显示屏170,风扇180,键盘181,温度传感器190等。

[0120] 处理器110例如为中央处理单元(central processing unit,CPU),其可以包括一个或多个处理核心。处理器110还可以包括:控制器,存储器控制器,图形处理器(graphics processing unit,GPU,集成显卡)等。

[0121] 处理器110中还可以设置存储器,用于存储指令和数据。在一些实施例中,处理器110中的存储器为高速缓冲存储器或寄存器。

[0122] 在一些实施例中,处理器110可以包括接口总线,其可以包括一个或多个接口。接口可以包括系统管理总线(“SMBus”),低引脚数(LPC)总线,串行外设接口(“SPI”)、高清音频(“HDA”)总线、串行高级技术附件(“SATA”)总线、标准化互连(例如,PCIe)和/或通用串行总线(universal serial bus,USB)接口等。

[0123] 可以理解的是,本申请实施例示意的各模块间的接口连接关系,只是示意性说明,

并不构成对电子设备100的结构限定。在本申请另一些实施例中，电子设备100也可以采用上述实施例中不同的接口连接方式，或多种接口连接方式的组合。

[0124] 无线通信模块160可以提供应用在电子设备100上的包括无线局域网(wireless local area networks，WLAN)(如无线保真(wireless fidelity，Wi-Fi)网络)，蓝牙(bluetooth，BT)等无线通信的解决方案。在一些实施例中，电子设备100的天线1和无线通信模块160耦合，使得电子设备100可以通过无线通信技术与网络以及其他设备通信。

[0125] 电子设备100通过独立图形处理器(即独立显卡或独立GPU)140或处理器110集成的图形处理器(即集成显卡或集成GPU)，显示屏170以及处理器110等实现显示功能。图形处理器(GPU)用于执行数学和几何计算，用于图形渲染。电子设备100可包括一个或多个GPU，其执行程序指令以生成或改变显示信息。

[0126] 显示屏170用于显示界面，图像，视频等。显示屏170包括显示面板。

[0127] 内部存储器(或主存储器)120可以包括存储程序区和存储数据区。其中，存储程序区可存储操作系统，至少一个功能所需的应用程序(比如声音播放功能，图像播放功能等)等。存储数据区可存储电子设备100使用过程中所创建的数据等。此外，内部存储器120可以包括高速随机存取存储器(RAM)，例如为DDR(Double Data Rate)RAM，更具体可以为DDR4或DDR5 RAM。内部存储器(或主存储器)120可以通过内存总线与处理器110通信连接。

[0128] 外部存储器121例如为固态磁盘、(“SSD”)或硬盘驱动器(“HDD”)或者闪存器件，通用闪存存储器(universal flash storage，UFS)等。外部存储器121可以通过串行高级技术附件(“SATA”)总线或标准化互连(例如，PCIe)与处理器110通信连接。

[0129] 嵌入式控制器(EC)130用于实现键盘控制，触摸板，电源管理，风扇控制等等的功能。控制器130可以通过温度传感器190采样CPU或GPU温度，然后根据CPU或GPU温度依据算法调控风扇。控制器130可以包含独立运行的软件，存放在自己的非易失性介质中。控制器130读取的外部温度传感器的温度也存储在控制器130内部RAM中。控制器120可以通过例如eSPI或LPC总线与处理器110进行连接和通信。

[0130] 独立图形处理器140用于执行数学和几何计算，用于图形渲染。独立图形处理器140可以在无3D程序运行时进入低功耗睡眠状态。独立图形处理器140睡眠时，电子设备100的图形渲染工作由处理器110中的图形处理器(即集成显卡或集成GPU)完成。独立图形处理器140可以通过标准化互连(例如，PCIe)与处理器110通信连接。独立图形处理器140可以根据显卡的Dx参数或TPP、TGP参数进行性能调控。

[0131] BIOS固件闪存150用于存储BIOS固件，BIOS固件可以为传统BIOS固件，也可以为基于UEFI的BIOS固件。BISO固件闪存150可以采样ROM(只读存储器)或Flash memory(闪存)。BIOS固件包括主板BIOS固件和显卡BIOS固件。

[0132] 风扇180用于为处理器110、独立图形处理器140等进行散热，以降低器件温度，保证设备运行。电子设备100可以包括一个或多个风扇180。示例性地，电子设备100可以在处理器110附近布置一个风扇，在独立图形处理器140附近布置一个风扇。

[0133] 键盘181用于供电子设备100的用户进行输入。

[0134] 温度传感器190用于采集电子设备100的例如处理器110、内部存储器120、独立图形处理器140等器件的温度。电子设备100可以包括一个或多个温度传感器190。示例性地，例如在处理器110外部布置一个或多个温度传感器190，在内部存储器120外部布置一个或

多个温度传感器,在独立图形处理器140外部布置一个或多个温度传感器190。温度传感器190可以通过例如系统管理总线(“SMBus”)与处理器110或嵌入式控制器(EC)130通信连接,从而将采集的温度发送至处理器110或嵌入式控制器(EC)130。

[0135] 电子设备100的软件系统可以采用分层架构,事件驱动架构,微核架构,微服务架构,或云架构。本申请实施例以分层架构的Windows系统为例,示例性说明电子设备100的软件结构。

[0136] 图3是本申请实施例的电子设备100的软件结构框图。

[0137] 电子设备100的分层架构将软件分成若干个层,每一层都有清晰的角色和分工。层与层之间通过软件接口通信。

[0138] 在一些实施例中,将Windows系统分为用户态和内核态。其中,用户态包括应用层以及子系统动态链接库。内核态自下而上分为固件层、硬件抽象层(hardwareabstraction layer,HAL)、内核和驱动层及执行体。

[0139] 如图3所示,应用层包括音乐、视频、游戏、办公、性能调控软件等应用程序。应用层还包括环境子系统、调度引擎、系统支持进程、服务进程等。其中,图中仅示出部分应用程序,应用层还可以包括其他应用程序,例如购物应用、浏览器等,本申请不做限定。

[0140] 环境子系统可以将基本的执行体系统服务的某些子集以特定的形态展示给应用程序,为应用程序提供执行环境。

[0141] 场景识别引擎可以识别电子设备100所处的用户场景,并确定与该用户场景匹配的基础调度策略(也可称为第二调度策略)。

[0142] 调度引擎可以获取电子设备100的负载情况,并结合电子设备100的负载情况及上述基础调度策略确定符合电子设备100实际运行情况的实际调度策略。

[0143] 性能调控应用可以根据电子设备100所处的用户场景对CPU、GPU等进行性能调控。性能调控应用可以为独立应用程序,也可以其它应用(例如计算机管家)集成的软件模块,或者可以集成在操作系统的服务或引擎中。

[0144] 子系统动态链接库包括API模块,该API模块包括Windows API,Windows原生API、NVAPI等。其中,Windows API,Windows原生API均可以为应用程序提供系统调用入口及内部函数支持,区别在于Windows原生API为Windows系统原生的API。例如,Windows API可包括user.dll、kernel.dll,Windows原生API可包括ntdll,dll。其中,user.dll是Windows用户界面接口,可用于执行创建窗口、发送消息等操作。kernel.dll于为应用程序提供访问内核的接口。ntdll.dll是重要的WindowsNT内核级文件,描述了windows本地NTAPI的接口。当Windows启动时,ntdll.dll就驻留在内存中特定的写保护区域,使别的程序无法占用这个内存区域。

[0145] 执行体包括进程管理器、虚拟内存管理器、安全引用监视器、I/O管理器、Windows管理规范(Windows management instrumentation,WMI)、电源管理器、系统事件驱动(operating system event driver,OsEventDriver)节点、系统与芯片驱动(operatingsystem to System on Chip,OS2SOC)节点等。

[0146] 进程管理器用于创建及中止进程和线程。

[0147] 虚拟内存管理器实现“虚拟内存”。虚拟内存管理器也为高速缓存管理器提供基本的支持。

[0148] 安全引用监视器可在本地计算机上执行安全策略,它保护了操作系统资源,执行运行时对象的保护和监视。

[0149] I/O管理器执行独立于设备的输入/输出,并进一步处理调用适当的设备驱动程序。

[0150] 电源管理器可管理所有支持电源状态更改的设备的电源状态更改。

[0151] 系统事件驱动节点可以与内核和驱动层进行交互,例如与显卡驱动进行交互,在确定存在GPU视频解码事件后,向场景识别引擎上报该GPU视频解码事件。

[0152] 系统与芯片驱动节点可供调度引擎向硬件设备发送调整信息,例如向CPU发送调整PL1和PL2的信息。

[0153] 内核和驱动层包括内核以及设备驱动程序。

[0154] 内核是对处理器体系结构的抽象,将执行体与处理器体系结构的差异相隔离,保证系统的可移植性。内核可以进行线程安排和调度、陷阱处理和异常调度、中断处理和调度等。

[0155] 设备驱动程序运行在内核模式下,为I/O系统和相关硬件之间的接口。设备驱动程序可包括显卡驱动、Intel DTT驱动、鼠标驱动、音视频驱动、摄像头驱动、键盘驱动等。例如,显卡驱动可以驱动GPU运行,Intel DTT驱动可以驱动CPU运行。

[0156] HAL是一个核心态模块,可以隐藏各种与硬件有关的细节,例如I/O接口、中断控制器以及多处理器通信机制等,为运行Windows的不同硬件平台提供统一的服务接口,实现多种硬件平台上的可移植性。需要说明的是,为了维护Windows的可移植性,Windows内部组件和用户编写的设备驱动程序并不直接访问硬件,而是通过调用HAL中的例程。

[0157] 固件层可以包括基本输入输出系统(basic input output system,bios),BIOS是一组固化到计算机主板上一个只读存储器(read only memory,ROM)芯片内的程序,它保存着计算机最重要的基本输入输出的程序、开机后自检程序和系统自启动程序,它可从互补金属氧化物半导体(complementary metal oxide semiconductor,CMOS)中读写系统设置的具体信息。其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。

[0158] Intel DTT驱动可以通过BIOS向CPU发送指令的。

[0159] 需要说明的是,本申请实施例仅以Windows系统举例来说明,在其他操作系统中(例如安卓系统,IOS系统等),只要各个功能模块实现的功能和本申请的实施例类似也能实现本申请的方案。

[0160] 可以理解的是,图3示出的软件结构中的层以及各层中包含的部件,并不构成对电子设备100的具体限定。在本申请另一些实施例中,电子设备100可以包括比图示更多或更少的层,以及每个层中可以包括更多或更少的部件,本申请不做限定。

[0161] 图4示出了电子设备100对资源进行调度时软件和硬件的工作流程示意图。

[0162] 如图4所示,应用层的场景识别引擎包括系统探针模块、场景识别模块及基础策略匹配管理器。场景识别模块可分别与系统探针模块及基础策略匹配管理器进行交互。场景识别模块可以向系统探针模块发送用于获取探针状态的请求。系统探针模块可以获取电子设备100的运行状态。例如,系统探针模块可以包括电源状态探针、外设状态探针、进程负载探针、音视频状态探针、系统负载探针及系统事件探针等。

[0163] 其中,电源状态探针可以向内核态订阅电源状态事件,根据内核态反馈的回调函

数确定电源状态，电源状态包括电池(剩余)电量、电源模式等，电源模式可包括交流电源(alternating current，AC)和直流电源(direct current，DC)。例如，电源状态探针可向执行体层的OsEventDriver节点发送订阅电源状态事件的请求，由OsEventDriver节点向执行体层的电源管理器转发该请求。电源管理器可通过该OsEventDriver节点向电源状态探针反馈回调函数。

[0164] 外设状态探针可以向内核态订阅外设事件，根据内核态反馈的回调函数确定外设事件。外设事件包括鼠标滚轮滑动事件、鼠标点击事件、键盘输入事件、麦克风输入事件、摄像头输入事件等。

[0165] 进程负载探针可以向内核态订阅进程负载，根据内核态反馈的回调函数确定进程(例如，第一进程)的负载。

[0166] 系统负载探针可以向内核态订阅系统负载，根据内核态反馈的回调函数确定系统负载。

[0167] 音视频状态探针可向内核态订阅音视频事件，根据内核态反馈的回调函数确定电子设备100当前存在的音视频事件。音视频事件可包括GPU解码事件等。例如，音视频状态探针可以向执行体层的OsEventDriver节点发送用于订阅GPU解码事件的请求，由OsEventDriver节点向内核和驱动层的显卡驱动转发该请求。显卡驱动可以监控GPU的状态，在监控到GPU在进行解码操作后，通过该OsEventDriver节点向音视频状态探针反馈回调函数。

[0168] 系统事件探针可以向内核态订阅系统事件，根据内核态反馈的回调函数确定系统事件。系统事件可包括窗口变化事件、进程创建事件、线程创建事件等。例如，系统事件探针可以向执行体层的OsEventDriver节点发送订阅进程创建事件的请求，由OsEventDriver节点向进程管理器转发该请求。进程管理器可在创建进程后，通过该OsEventDriver节点向系统事件探针反馈回调函数。又例如，系统事件探针还向API模块发送订阅焦点窗口变化事件，API模块可监控电子设备100的焦点窗口是否发生变化，并在监控到焦点窗口发生变化时，向系统事件探针反馈回调函数。

[0169] 可见，系统探针模块通过向内核态订阅电子设备100的各种事件，再根据内核态反馈的回调函数确定电子设备100的运行状态，即得到探针状态。系统探针模块得到探针状态后，可向场景识别模块反馈该探针状态。场景识别模块接收到探针状态后，可根据该探针状态确定电子设备100所处的用户场景。该用户场景可包括视频场景、游戏场景、办公场景及社交场景等。用户场景可以反映用户当前的使用需求。例如，场景识别引擎在识别出焦点窗口为视频应用的窗口时，确定出电子设备100处于视频场景，这说明用户需要使用视频应用观看、浏览视频。又例如，场景识别引擎识在识别出焦点窗口为微信™的聊天窗口时，确定电子设备100处于社交场景。场景识别模块还可向基础策略匹配管理器发送该用户场景。基础策略匹配管理器可根据该用户场景确定基础调度策略(也可称为第二调度策略，具体可以参见下文S301、S302中的说明)。基础策略匹配管理器可向场景识别模块反馈该基础调度策略。场景识别模块可向应用层的调度引擎发送该基础调度策略及用户场景。

[0170] 如图4所示，调度引擎包括负载管控器、芯片策略融合器以及调度执行器。其中，负载管控器可接收场景识别模块发送的基础调度策略及用户场景。负载管控器还可从系统探针模块获取系统负载，并根据系统负载和用户场景对该基础调度策略进行调整，得到实际

调度策略(也可称为第一调度策略,具体可以参见下文S310中的说明)。实际调度策略中包括OS调度策略和第一CPU功耗调度策略(也可称为第一子策略)。其中,负载管控器可向调度执行器发送该OS调度策略,由调度执行器基于该OS调度策略进行调度。OS调度策略用于调整焦点进程的进程优先级及I/O优先级。示例性的,调度执行器可向进程管理器发送调整焦点进程的进程优先级的指令,响应于该指令,进程管理器对焦点进程的进程优先级进行调整。又例如,调度执行器可向I/O管理器发送调整焦点进程的I/O优先级的指令,响应于该指令,I/O管理器对焦点进程的I/O优先级进行调整。

[0171] 负载管控器还可向芯片策略融合器发送第一CPU功耗调度策略,芯片策略融合器可基于CPU的芯片平台类型及该第一CPU功耗调度策略,得到第二CPU功耗调度策略(也可称为第二子策略,具体可以参见下文S317～S325中的说明)。CPU的芯片平台类型主要分为两种,分别为超威半导体公司®(advanced micro devices,AMD)的CPU和英特尔®(Intel®)的CPU,这两类CPU对于CPU功耗的调整方式并不相同,因此需要进行区分。

[0172] 若CPU的芯片平台类型为AMD(也可以称为第一类型),调度执行器可以向电源管理器发送调整EPP的指令,以调整CPU的EPP。另外,调度执行器还可以向OS2SOC驱动节点发送调整PL1、PL2的指令,以调整CPU的PL1和PL2。

[0173] 若CPU的芯片平台类型为英特尔®,调度执行器可以通过WMI插件向Intel DTT驱动发送该第二CPU功耗调度策略,第二CPU功耗调度策略可包括PL1的最小值、PL1的最大值、PL2、PL2的持续时间及EPP,由Intel DTT驱动CPU基于该第二CPU功耗调度策略运行。

[0174] 本申请实施方式所提供的性能调控方法,主要分为两个过程,分别为:(1)确定电子设备所处的用户场景;(2)根据电子设备所处的用户场景确定电子设备的性能参数。下面将结合附图分别说明上述两个过程。

[0175] 下面将以电子设备处于视频播放场景为例,结合图5,对图4所示的电子设备中部分模块的交互过程进行说明。如图5所示,本申请实施例提供的一种性能调控方法,其确定电子设备所处的用户场景的流程如下:

[0176] S501、系统探针模块向OsEventDriver节点发送订阅进程创建事件的请求。

[0177] 如图5所示,场景识别引擎包括系统探针模块,系统探针模块包括系统事件探针。在本申请实施例中,可以由系统事件探针向位于执行体层的OsEventDriver节点发送订阅进程创建事件的请求。其中,订阅进程创建事件的请求也可以称为第一请求。

[0178] 在一种可选的实施方式中,订阅进程创建事件的请求可以携带有进程名称。即场景识别引擎可以仅订阅指定进程的创建事件,减少不相干进程的创建事件的干扰。例如,指定进程可以为视频应用的进程、游戏应用的进程、办公应用的进程、社交应用的进程等等。当然,在其他实施方式中,场景识别引擎也可以不对订阅的进程创建事件做出限制。

[0179] S502、OsEventDriver节点向进程管理器发送订阅进程创建事件的请求。

[0180] 进程创建事件的请求可以参考S501的描述,在此不做赘述。

[0181] 也就是说,场景识别引擎的系统事件探针可以通过OsEventDriver节点向进程管理器发送订阅进程创建事件的请求。

[0182] 可以理解地,OsEventDriver节点会向进程管理器注册一个回调,注册该回调的作用是当进程管理器创建进程后,可以向OsEventDriver节点返回该进程创建事件。

[0183] S503、系统探针模块向OsEventDriver节点发送订阅GPU解码事件的请求。

[0184] 仍然如图4所示，系统探针模块还包括音视频状态探针。在本申请实施例中，可以由系统探针模块的音视频状态探针向OsEventDriver节点发送订阅GPU解码事件的请求。其中，订阅GPU解码事件的请求也可以称为第三请求。

[0185] S504、OsEventDriver节点向显卡驱动发送订阅GPU解码事件的请求。

[0186] 也就是说，场景识别引擎的音视频状态探针可以通过OsEventDriver节点向显卡驱动发送订阅GPU解码事件的请求。同样地，OsEventDriver节点可向显卡驱动注册一个回调，注册该回调的作用是当显卡驱动监控到GPU进行解码操作后，可以向OsEventDriver节点返回该GPU解码事件。

[0187] S505、系统探针模块向API模块发送订阅焦点窗口变化事件的请求。

[0188] API模块可包括由user32.dll实现的windows用户界面接口，该接口可用于创建窗口。在一种可选的实施方式中，可以由系统探针模块的系统事件探针向API模块的windows用户界面接口发送订阅焦点窗口变化事件的请求。其中，订阅焦点窗口变化事件的请求也可以称为第二请求。

[0189] 同样地，该系统事件探针可向API模块注册一个回调，注册该回调的作用是当API模块(的windows用户界面接口)监控到焦点窗口发生变化时，可以向系统事件探针返回该焦点窗口变化事件。

[0190] 焦点窗口为拥有焦点的窗口，大概率为用户当前需要使用的窗口。因此，通过监控焦点窗口，可以确定用户的使用需求。例如，焦点窗口为视频应用的窗口，则表明用户需求浏览、播放视频。又例如，焦点窗口为游戏应用的窗口，则表明用户需求打游戏。通过监控焦点窗口是否发生变化，可以确定用户需求是否发生改变。例如，焦点窗口由视频应用的窗口变为游戏应用的窗口，则表明用户当前的需求由看视频变成了打游戏。

[0191] 需要说明的是，上述S501、S503及S505之间没有严格的先后顺序，其可以按照图5中所示的顺序依次执行，也可以同时执行，也可以按照S503、S501、S505的顺序依次执行、按照S503、S505、S501的顺序依次执行、按照S505、S501、S503的顺序依次执行或者按照S505、S503、S501的顺序依次执行。相应地，S502、S504及S506之间也没有严格的先后顺序，只要满足S502在S501之后执行、S504在S503之后执行以及S506在S505之后执行即可，在此不做具体限制。

[0192] S506、响应于接收到用户开启视频应用的操作，视频应用向进程管理器发送创建进程请求。

[0193] 其中，创建进程请求包括视频应用程序的存储地址。

[0194] 视频应用可以通过API模块的kernel32.dll接口及Ntdll.dll接口向进程管理器发送创建进程的请求(图未示)。

[0195] S507、进程管理器创建视频应用进程。

[0196] 具体的，进程管理器可以通过该存储地址查询到视频应用程序的二进制文件。通过加载视频应用程序的二进制文件，可以创建进程运行的环境，启动视频应用进程。

[0197] 其中，Windows操作系统将一个应用程序的一次运行定义为一个进程。一个进程可以拥有多个线程。窗口是窗口结构的实例，是一种图形用户界面(graphical user interface，GUI)资源，窗口是由线程创建的，线程可以拥有它所创建的所有窗口。在本申请实施例中，电子设备运行视频应用，则进程管理器需创建该视频应用的进程，即视频应用进

程(即第一进程)。视频应用进程包括多个线程,多个线程包括线程1,线程1可用于创建视频应用的主窗口,主窗口为集成有视频应用全部功能按键的窗口。

[0198] S508、进程管理器向OsEventDriver节点上报进程创建事件。

[0199] 其中,进程创建事件可包括进程管理器所创建的进程的名称。在本申请实施例中,该进程的名称为视频应用进程的名称。当然,若进程管理器创建的是其他应用的进程,该进程的名称也对应为其他应用进程的名称。

[0200] 前文已经说明,OsEventDriver节点向进程管理器发送了订阅进程创建事件的请求,且注册了回调。因此,进程管理器在创建视频应用进程后可向OsEventDriver节点上报进程创建事件。

[0201] S509、OsEventDriver节点向系统探针模块上报进程创建事件。

[0202] 其中,关于进程创建事件的描述见S508,在此不再赘述。

[0203] 在本申请实施例中,该OsEventDriver节点可向系统探针模块的系统事件探针上报该进程创建事件。

[0204] S510、系统探针模块向场景识别模块发送进程创建事件。

[0205] S511、响应于线程1的调用请求,API模块创建窗口1。

[0206] 进程管理器创建视频应用进程后,视频应用进程的线程1主动调用API模块的windows用户界面接口创建窗口1。示例性的,如图6中(a)所示,电子设备可以显示窗口601,该窗口601可以为桌面,也可以称为主界面。该窗口601包括视频应用的图标602。电子设备可以接收用户点击该视频应用的图标602的操作,响应于该操作,如图6中的(b)所示,电子设备显示窗口603(即窗口1,也可以称为第一窗口)。在上述过程中,焦点窗口由原本的窗口601变为窗口603。

[0207] S512、API模块向系统探针模块上报焦点窗口事件。

[0208] 在本申请实施例中,API模块的windows用户界面接口创建窗口1后,可以获取第一进程(即焦点进程)的名称及第二进程的名称,第一进程为当前的焦点窗口(即窗口1)对应的进程,第二进程为上一个焦点窗口(例如,窗口2)对应的进程。示例性的,窗口1对应的进程为视频应用进程(第一进程),该进程的名称例如为hlive.exe,窗口2对应的进程为windows程序管理器的进程(第二进程),该进程的名称例如为explorer.exe。由于第一进程的名称与第二进程的名称不一致,API模块确定焦点窗口发生变化,向系统探针模块的系统事件探针上报焦点窗口事件。其中,焦点窗口变化事件包括第一进程(即焦点进程)的名称。示例性的,第一进程为视频应用进程,焦点窗口变化事件携带有视频应用进程的名称。

[0209] 需要说明的是,在电子设备已经启动视频应用的情况下,电子设备可以不用执行S506～S511。在系统探针模块向API模块发送订阅焦点窗口变化事件的请求后,若用户将焦点窗口切换为视频应用的窗口,API模块同样可以检测到焦点窗口发生变化,并向系统探针模块上报焦点窗口事件。

[0210] S513、系统探针模块向场景识别模块发送焦点窗口事件。

[0211] S514、场景识别模块确定第一进程所属的类型为视频类。

[0212] 电子设备可以预先配置有应用名单,场景识别模块可以查询应用名单中是否包括第一进程。若应用名单中包括第一进程,场景识别模块可以确定第一进程所属的类型。其中,应用名单包括每个应用的进程名称及应用所属的类型。示例性的,应用名单可以如表1

所示：

[0213] 表1

[0214]

应用	进程名称	类型
荣耀视频	hlive.exe	视频类
word	word.exe	办公类
射击游戏	shot.exe	游戏类
微信™	wechat.exe	社交类
……	……	……

[0215] 例如，第一进程的名称为hlive.exe，则场景识别模块可以确定第一进程所属的类型为视频类。又例如，第一进程的名称为wechat.exe，则场景识别模块可以确定第一进程所属的类型为社交类。需要说明的是，上述表1仅作为示例，实际上表1还可包括更多应用的进程名称及其所属的类型。

[0216] 需要说明的是，此步骤的目的在于初步判断电子设备所处的用户场景。电子设备所处的用户场景可包括视频场景、游戏场景、社交场景、办公场景、浏览器场景等等。其中，视频场景进一步可包括视频播放场景、视频浏览场景。社交场景进一步可包括文字聊天场景、语音聊天场景、视频聊天场景等。办公场景进一步可包括文档编辑场景、文档浏览场景、视频会议场景等。浏览器场景可包括浏览网页场景及播放视频场景等。

[0217] 在本步骤中，通过第一进程所属的类型，可以确定电子设备所处的用户场景的类型。例如，若确定第一进程所属的类型为视频类，则可以确定电子设备处于视频场景；又例如，若确定第一进程所属的类型为游戏类，则可以确定电子设备处于游戏场景。为了进一步分析用户需求，场景识别模块还可以进一步结合其他参数(例如，外设事件、GPU运行状态等)来分析电子设备所处的具体场景，以达到分析结果更加准确的效果，其具体内容参见后文，在此暂不描述。

[0218] S515，响应于接收到用户播放视频的操作，视频应用向API模块发送视频播放指令。

[0219] 具体的，视频应用可向API模块的DirectX API发送该视频播放指令。该视频播放指令可包括视频的缓存地址。

[0220] S516、API模块读取视频文件。

[0221] API模块可根据视频播放指令中携带的缓存地址，读取对应的视频文件。

[0222] S517、API模块向显卡驱动发送解码指令。

[0223] S518、显卡驱动向GPU发送启动指令。

[0224] S519、GPU进行解码。

[0225] 具体的，GPU可通过GPU video processing引擎对该视频文件进行解码操作。

[0226] S520、GPU向显卡驱动上报解码事件。

[0227] S521、显卡驱动向OsEventDriver节点上报解码事件。

[0228] S522、OsEventDriver节点向系统探针模块上报解码事件。

[0229] 具体的，OsEventDriver节点向系统探针模块的音视频状态探针上报该解码事件。

[0230] S523、系统探针模块向场景识别模块发送解码事件。

[0231] S524、场景识别模块向系统探针模块发送指令1。

[0232] 其中,指令1指示系统探针模块获取第一进程的GPU占用率。该指令1可携带有第一进程的名称。

[0233] S525、系统探针模块向进程管理器发送获取第一进程的GPU占用率的请求。

[0234] 其中,该获取焦点进程的GPU占用率的请求可以包括第一进程的名称。

[0235] 在一种可选的实施方式中,可以由系统探针模块的音视频状态探针向进程管理器发送获取第一进程的GPU占用率的请求。

[0236] S526、进程管理器采集第一进程的GPU占用率。

[0237] 具体的,进程管理器可以通过显卡驱动的图像核心(graphics kernel)接口采集第一进程的GPU占用率。

[0238] S527、进程管理器向系统探针模块发送第一进程的GPU占用率。

[0239] 进程管理器可向系统探针模块的音视频状态探针发送第一进程的GPU占用率。

[0240] S528、系统探针模块向场景识别引擎发送第一进程的GPU占用率。

[0241] S529、场景识别模块判断第一进程的GPU占用率是否大于0。

[0242] 其中,若第一进程的GPU占用率大于0,则执行S130。

[0243] 通过第一进程的GPU占用率,可以确定第一进程在运行过程中是否使用GPU,若第一进程的GPU占用率大于0,则可以认为第一进程在运行过程中使用了GPU;若第一进程的GPU占用率为0,则表明第一进程在运行过程中未使用GPU。

[0244] S530、场景识别模块向系统探针模块发送指令2。

[0245] 其中,指令2指示系统探针模块获取第一进程的GPU引擎。该指令2可携带有第一进程的名称。

[0246] S531、系统探针模块向进程管理器发送获取第一进程的GPU引擎的请求。

[0247] 其中,可以由系统探针模块的音视频状态探针向进程管理器发送获取第一进程的GPU引擎的请求。获取第一进程的GPU引擎的请求包括第一进程的名称。

[0248] GPU引擎包括GPU 3D引擎、GPU copy引擎、GPU video encode引擎、GPU video processing引擎。其中,GPU 3D引擎主要负责处理2D或者3D图形。GPU copy引擎主要用于传输数据。GPU video encode引擎主要用于进行编码操作。GPU video processing引擎主要进行解码操作。在一些实施例中,GPU video processing引擎也可以由GPU video decode引擎代替。

[0249] S532、进程管理器获取第一进程的GPU引擎。

[0250] 具体的,进程管理器可以通过显卡驱动的graphics kernel接口获取第一进程的GPU引擎。

[0251] S533、进程管理器向系统探针模块发送消息1,消息1指示第一进程的GPU引擎为GPU video processing引擎。

[0252] 具体的,进程管理器可向系统探针模块的音视频状态探针发送该消息,再由音视频状态将该消息转发给场景识别模块。

[0253] S534、系统探针模块向场景识别模块发送消息1。

[0254] S535、场景识别模块判断第一进程的GPU引擎是否为GPU video processing引擎。

[0255] 若第一进程的GPU引擎为GPU video processing引擎,则执行S529;若第一进程的GPU引擎不为GPU video processing引擎,则执行S530。

[0256] 在S514步骤中，场景识别引擎已经确定第一进程所属的类型为视频类，即可以确定电子设备处于视频场景。通过步骤S535，场景识别引擎可以确定第一进程通过GPU执行的具体操作，进而确定用户在使用视频应用的具体操作。例如，若第一进程的GPU引擎为GPU video processing引擎，则表明第一进程在使用GPU进行解码操作，可以认为用户在使用视频应用播放视频。又例如，第一进程的GPU引擎不为GPU video processing引擎，则表明第一进程没有在使用GPU进行解码操作，那么用户大概率在视频应用上浏览视频资源，还未播放视频。

[0257] S536、场景识别模块根据第一进程的进程信息确定用户场景为视频播放场景。

[0258] 其中，第一进程的进程信息包括第一进程的名称、第一进程所属的应用类型、第一进程的GPU占用率及第一进程使用的GPU引擎等信息。

[0259] 综合上述内容可知，若第一进程(焦点进程)的类型为视频类、第一进程的GPU占用率大于0且第一进程的GPU引擎为GPU video processing引擎，则可以确定电子设备处于视频播放场景。

[0260] 需要说明的是，上述S501～S536仅以电子设备处于视频场景下的视频播放场景为例进行说明。实际上，电子设备还可以处于其他用户场景(例如，游戏场景、办公场景、社交场景、视频浏览场景、编程场景等)。

[0261] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于游戏类、CPU的电源模式变为游戏模式(game mode)、第一进程的GPU占用率大于0且第一进程的GPU引擎为GPU 3D引擎，则可以确定电子设备处于游戏场景。

[0262] 其中，系统探针模块的电源状态探针可以向电源管理器发送订阅电源模式变化事件的请求。电源管理器可以在电源模块转换为游戏模式(game mode)时，向系统探针模块的电源状态探针上报该电源模式变化事件。如此，场景识别引擎可通过电源模式变化事件确定CPU的电源模式是否为game mode。

[0263] 另外，场景识别引擎获取第一进程的类型的过程可以参阅图5中S501、S502、S505、S506～S514，场景识别引擎判断第一进程的GPU占用率是否大于0且第一进程的GPU引擎是否为GPU 3D引擎的过程参阅S524～S535。区别在于将视频应用更换为游戏应用，在此不再赘述。

[0264] 下面，再结合图7简单说明电子设备处于办公场景时，其确定自身所处的用户场景的流程。需要说明的是，图7所示的流程图与图5所示的流程图，其原理及流程基本详细，下面仅具体说明两者的不同之处，类似之处不再赘述，详情参见图5中相关步骤的说明。图7示出了本申请实施例提供的一种性能调控方法，其确定电子设备所处的用户场景的流程如下：

[0265] S701、系统探针模块向OsEventDriver节点发送订阅进程创建事件的请求。

[0266] S702、OsEventDriver节点向进程管理器发送订阅进程创建事件的请求。

[0267] S703、系统探针模块向OsEventDriver节点发送订阅外设事件的请求。

[0268] 如图4所示，系统探针模块还包括外设状态探针。在本申请实施例中，可以由系统探针模块的外设状态探针向OsEventDriver节点发送订阅外设事件的请求。其中，订阅外设事件的请求也可以称为第四请求。

[0269] 其中，外设事件包括鼠标滚轮滑动、鼠标点击、键盘输入、摄像头输入、麦克风输入

等事件。

[0270] S704、OsEventDriver节点向外设驱动发送订阅外设事件的请求。

[0271] 需要说明的是，外设驱动为所有外设的驱动的统称，例如可以包括鼠标驱动、键盘驱动、摄像头驱动、麦克风驱动等。

[0272] S705、系统探针模块向API模块发送订阅焦点窗口变化事件的请求。

[0273] S706、响应于接收用户开启办公应用的操作，办公应用向进程管理器发送创建办公应用进程的请求。

[0274] 其中，创建办公应用进程的请求可包括办公应用程序的存储地址。

[0275] S707、进程管理器创建办公应用进程。

[0276] 具体的，进程管理器可以通过该存储地址查询到办公应用程序的二进制文件。通过加载办公应用程序的二进制文件，可以创建进程运行的环境，启动视频应用进程。另外，办公应用进程包括线程2，线程2可用于创建办公应用的主窗口。

[0277] S708、进程管理器向OsEventDriver节点上报进程创建事件。

[0278] S709、OsEventDriver节点向系统探针模块上报进程创建事件。

[0279] 其中，该进程创建事件携带有办公应用进程的名称。

[0280] S710、系统探针模块向场景识别模块发送进程创建事件。

[0281] S711、响应于线程2的调用请求，API模块创建办公应用窗口。

[0282] S712、API模块向系统探针模块上报焦点窗口事件。

[0283] 其中，焦点窗口事件中携带有第一进程(焦点进程)的名称。可以理解地，在本申请实施例中，第一进程即为办公应用进程。

[0284] S713、系统探针模块向场景识别模块发送焦点窗口事件。

[0285] S714、场景识别模块确定第一进程所属的类型为办公类。

[0286] 例如，第一进程的名称为word.exe，则可以确定第一进程所属的类型为办公类。

[0287] S715、响应于用户对外设的操作，外设驱动检测到外设事件。

[0288] S216、外设驱动向OsEventDriver节点上报外设事件。

[0289] S717、OsEventDriver节点向系统探针模块发送外设事件。

[0290] S718、系统探针模块向场景识别模块发送外设事件。

[0291] S719、场景识别模块根据外设事件及第一进程所属的类型确定用户场景。

[0292] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于办公类，且外设事件为鼠标滚轮滑动事件或点击事件，则可以确定电子设备具体处于办公场景下的文档浏览场景。又或者，若场景识别引擎确定第一进程(焦点进程)的类型属于办公类，且在接收到键盘输入事件后的预设时间(例如，10秒)内未再次接收到鼠标滚轮滑动事件、鼠标击事件、键盘输入事件，可以确定电子设备具体处于办公场景下的文档浏览场景。

[0293] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于办公类，且接收到键盘输入事件，则可以确定电子设备具体处于办公场景下的文档编辑场景。

[0294] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于办公类，且接收到摄像头输入事件(即摄像头处于开启状态且存在视频流输入)，可以确定

电子设备具体处于办公场景下的视频会议场景。

[0295] 电子设备还可以处于社交场景。社交场景包括三个具体场景，分别为：文字聊天场景、语音聊天场景及视频聊天场景。判断电子设备处于社交场景的原理与判断电子设备处于办公场景的原理类似，在此暂不赘述，下面仅说明判断电子设备处于社交场景需要满足的条件。

[0296] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于社交类，且接收到键盘输入事件，则可以确定电子设备具体处于社交场景下的文字聊天场景。

[0297] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于社交类，且接收到麦克风输入事件且摄像头处于关闭状态，则可以确定电子设备具体处于社交场景下的语音聊天场景。

[0298] 在一种可选的实施方式中，若场景识别引擎确定第一进程(焦点进程)的类型属于社交类，且接收到麦克风输入事件、摄像头输入事件，则可以确定电子设备具体处于社交场景下的视频聊天场景。

[0299] 上述内容说明了如何识别电子设备所处的用户场景，在确定电子设备所处的用户场景后，电子设备还可根据所处的用户场景确定是否需要进行性能调控，当需要进行性能调控时根据所处的用户场景设定TPP、TGP和/或TDP的具体参数值或Dx值，使显卡根据当前的TPP、TGP和/或TDP参数或Dx值进行性能调控，从而可以根据场景提供更合理的CPU和GPU性能，不仅更好地满足用户在各种场景下的性能需求，而且可以相对地提高设备的续航，从而提高用户体验。

[0300] 图8为本申请实施例提供的性能调控方法的流程图。

[0301] 如图8所示，本申请实施例提供的性能调控方法具体包括：

[0302] 步骤S801，场景识别引擎确定电子设备的用户场景，并将场景信息发送至性能调控应用。

[0303] 其中，场景信息用于指示电子设备所处的用户场景。示例性的，电子设备可以预先为不同的用户场景分配唯一标识，该场景信息则可包括用户场景所的唯一标识。例如，该标识(例如为V01)可指示电子设备处于视频播放场景。又例如，该标识(例如为V02)可指示电子设备处于视频浏览场景。

[0304] 关于场景识别模块确定电子设备所处的用户场景的过程，具体参阅S501～S536，在此不再赘述。

[0305] 步骤S802，性能调控应用根据场景信息确定性能参数的参数值。

[0306] 其中，性能参数包括第一性能参数和第二性能参数。第一性能参数包括CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗和所述GPU的DB功耗中的至少一种。第二性能参数包括GPU的Dx值。

[0307] 示例性地，在本申请实施例中，可以通过测试预先确定各种用户场景的性能需求，例如通过测试确定办公场景所需的性能，并根据该性能确定办公场景下的第一性能参数的参数值，例如确定CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗和所述GPU的DB功耗的具体参数值。

[0308] 示例性地，在本申请实施例中，可以通过建立用户动态测试模型来确定各场景下

的性能需求。该动态测试模型可以用于确定电子设备运行各种类型应用时CPU性能、GPU性能等，从而基于这些数据确定电子设备在各场景下的性能需求，进而基于该性能需求确定各种场景下第一性能参数的具体数值。

[0309] 通过在各种场景下采用不同的第一性能参数可以使得电子设备在保证当前应用流畅运行的同时降低设备的功耗。示例性地，例如当电子设备处于办公场景时，因为性能需求较低，因此可以设置较低的TGP、TDP、TPP和DB值，在保证办公应用流畅运行的同时，降低设备功耗，延长设备续航。当电子设备处于游戏场景、性能测试场景、3D渲染场景时，GPU性能需求较大，则可以使TPP、DB、TGP值较大，这样可以是GPU提供更高的性能，满足游戏运行需求。当电子设备处于编程场景时，CPU性能需求较大，GPU需要性能较小，则可以使TPP较大，TGP和DB值较小，从而满足编程需求，并降低设备功耗。换言之，在本申请实施例中，根据电子设备所处的场景动态调整第一性能参数的参数值，从而既保证应用的流畅运行，又降低设备的功耗，延长设备续航。并且与目前的电子设备出厂时设定TDP、TGP或TPP值的调控方式相比，可以更好地满足应用的性能需求，根据具体应用合理分配CPU功耗和GPU功耗，提高用户体验。

[0310] 应当理解的是，对于电子设备而言，基于处理器性能和散热设计TPP一般具有上限值，在本申请实施例中根据电子设备设定TPP、TDP、TGP的具体数值是在TPP上限值之内进行设定。例如电子设备的最大TPP为90W，则可以在性能需求较低的场景中将TPP设置为30W。在性能需要较大的场景中，可以将TPP设置为60W、70W甚至90W等较大数值。并且CPU性能需求较大时使TDP较大，GPU性能需求较大时，使TGP较大。与目前的基于TGP、TDP、TPP的性能调控方式相比可以更灵活的分配CPU和GPU功耗。这是因为采用目前的调控方式，一般直接设定TPP例如为60W，TDP为25W，TGP为35W，由处理器基于TPP、TGP、TDP根据需要自行调节性能，这种方式由于TDP、TGP已经设定使得无法根据需要合理分配CPU和GPU功耗。而采用本实施例的方式，则可以在GPU性能需求较大CPU性能需求较小的场景中将TGP设置为40W，将TDP设置为20W，这样可以更好地满足应用的性能需求。

[0311] 还应当理解的是，一般电子设备都配置有均衡模式和性能模式，在性能模式和均衡模式中性能参数上限值不同，当电子设备处于性能模式时则在性能模式下的第一性能参数的上限值内设定各用户场景下第一性能参数的参数值。当电子设备处于均衡模式时则在均衡模式下的第一性能参数的上限值内设定各用户场景下第一性能参数的参数值。例如在性能模式TGP上限为50W，在均衡模式中TGP上限为30W，则在性能模式下在50W的上限内设定各用户场景的TGP值，在均衡模式下在30W的上限内设定各用户场景的TGP值。

[0312] 进一步地，示例性地，在本申请实施例中，可以通过测试预先确定各种用户场景的性能需求，例如通过测试确定办公场景所需的性能，并根据该性能确定办公场景下的第二性能参数的参数值。通过调节第一性能参数可以更细的调节GPU的性能，为GPU提供更合理的性能。通过调节第二性能参数可以更大幅度或快速地调节GPU性能，以迅速降低(或者增大)GPU性能。

[0313] 需要说明的是，在本申请实施例中可以根据需要根据用户场景的需求调节第一性能参数或第二性能参数。

[0314] 在一种可选的实施方式中，默认Dx设置为D1(最大GPU功耗)，当用户场景发生变化时，通过调节第一性能参数来调节GPU性能。

[0315] 在一种可选的实施方式中，当用户场景需求的GPU性能很小时，可以将Dx值调节为D4或D5，以直接将GPU性能限制在较小值，从而延长设备续航。

[0316] 在一种可选的实施方式中，在当前用户场景的性能需求相对之前用户场景的性能需求增大时，通过调节第一性能参数来使设备提供更大的GPU和CPU性能。

[0317] 在一种可选的实施方式中，当电子设备的剩余电池电量不足时，例如小于设定百分比(例如小于20％)时，以将Dx值调节为D4或D5，以直接将GPU性能限制在较小值，从而延长设备续航。

[0318] 在一种可选的实施方式中，当电子设备的温度过高时，例如大于设定阈值(例如大于90℃)时，以将Dx值调节为D4或D5，以直接将GPU性能限制在较小值，从而延长设备续航。

[0319] 需要说明的是电子设备的温度包括CPU温度、GPU温度、主板温度等，电子设备的温度过高指的是CPU温度、GPU温度、主板温度等中的一个或多个温度过高。

[0320] 步骤S803，性能调控应用通过WMI接口将性能参数的参数值发送至BIOS。

[0321] 当在前述步骤S802中确定性能参数的参数值后，则性能调控应用调用WMI接口将性能参数的参数值发送至BIOS。

[0322] 步骤S804，BIOS将性能参数的参数值写入ACPI指的位置，并通知GPU驱动读取性能参数的参数值。

[0323] 示例性地，ACPI中指定了内部存储器(即内存)中存储第一性能参数和第二性能参数的存储位置，BIOS获取到一性能参数和第二性能参数的参数值后将该存储位置的性能参数修改为最新数值，然后通知GPU驱动读取最新的图形处理器的性能参数。

[0324] 第一性能参数的存储位置和第二性能参数的存储位置可以相同，也可以不同。

[0325] 步骤S805，GPU驱动读取性能参数的参数值，并根据性能参数的参数值调控GPU的性能。

[0326] 具体地，BIOS在修改图形处理器的性能参数的数值后会产生SCI中断，显卡驱动接收到SCI中断(也即事件通知)后调用显卡控制方法(ACPI中定义的显卡control method)读取图形处理器的性能参数的当前数值。

[0327] 当读取到性能参数后，则根据性能参数调控图形处理器的性能。例如当读取到Dx的当前数值后，则根据Dx当前数值对应的功耗控制显卡性能。示例性例如当前Dx值为D3，对应的显卡功耗为15W，则GPU驱动控制显卡将其最大功耗限制在15W。又例如，当读取到TGP、TDP、TPP、DB的当前数值后，根据TGP、TDP、TPP、DB的当前数值控制显卡性能。

[0328] 需要说明的是，当根据TGP、TDP、TPP、DB的当前数值控制显卡性能时，GPU驱动还与CPU驱动通信以获取当前的CPU功耗，并根据当前的TGP、TDP、TPP、DB值调节显卡性能。

[0329] 示例性地，例如当前的TGP为40W，TDP为30W，TPP为70W，DB为15W，则GPU驱动可以更加CPU的实际功耗将GPU功耗限制在25W(40-15)和55W(40+15)之间。即当CPU实际功耗较小时，例如CPU实际功耗为15W，则GPU功耗可以在55W内调节，即可以提供最大55W的GPU功耗。又例如CPU实际功耗为30W时，则GPU功耗可以在25W内调节，即可以提供最大25W的GPU功耗，这样可以保证提供足够的CPU性能。

[0330] 下面结合图9和图10以具体应用为例说明本申请实施例提供的性能调控方法。

[0331] 图9为本申请实施例提供的性能调控方法的示例调控过程。

[0332] 如图9所示，本申请实施例提供的性能调控方法的示例调控过程包括：

[0333] 步骤S901,响应于用于操作启动第一应用,在启动所述第一应用之前,所述电子设备的第一性能参数为第一参数值。

[0334] 示例性地,第一应用例如编程应用,例如各种编程应用。

[0335] 所述第一性能参数包括所述CPU的TDP功耗、所述GPU的TGP功耗、所述GPU的TPP功耗、所述GPU的DB功耗中的至少一种。

[0336] 第一参数值可以是一个参数集合,包含TDP、TPP、TGP、DB的具体数值。示例性地,第一参数值例如为TDP30W、TPP70W、TGP40W,DB15W。通过减小TGP和DB可以有效GPU性能,从而使设备提供充足的CPU性能。

[0337] 步骤S902,场景识别引擎根据第一应用确定电子设备处于第一用户场景。

[0338] 示例性地,第一用户场景为编程场景。关于场景识别模块确定电子设备所处的用户场景的过程,具体参阅S501～S536,在此不再赘述。

[0339] 步骤S903,场景识别引擎确定将第一场景信息发送至性能调控应用。

[0340] 示例性的,电子设备可以预先为不同的用户场景分配唯一标识,该场景信息则可包括用户场景所的唯一标识。

[0341] 步骤S904,在第一时间点,性能调控应用根据第一场景信息将第一性能参数调节为第二参数值,第二参数值与第一参数值不同。

[0342] 第二参数值可以是一个参数集合,包含TDP、TPP、TGP、DB的具体数值。第二参数值与第一参数值不同,指的是第一参数值和第二参数值中,存在至少一个参数的对应数值不同。例如,第一参数值中的TGP数值和第二参数值中的TGP数值不同。

[0343] 示例性地,第二参数值例如为TDP30W、TPP60W、TGP10W,DB5W。

[0344] 通过减小TGP和DB可以有效GPU性能,从而使设备提供充足的CPU性能。

[0345] 步骤S905,性能调控应用通过WMI接口将第二参数值发送至BIOS。

[0346] 步骤S906,BIOS将第二参数值写入ACPI指的位置,并通知GPU驱动读取第二参数值。

[0347] 具体地BIOS将第二参数值写入ACPI指的第一性能参数的存储位置,并通知GPU驱动读取第二参数值。

[0348] 步骤S907,GPU驱动读取第二参数值,并根据第二一参数值调控GPU的性能。

[0349] GPU读取第二参数值的过程参见S805的描述,在此不再赘述。

[0350] 步骤S908,响应于用于操作启动第二应用。

[0351] 示例性地,第二应用例如游戏应用,例如3A大型游戏。

[0352] 步骤S909,场景识别引擎根据第二应用确定电子设备处于第二用户场景。

[0353] 示例性地,第二用户场景为游戏场景。关于场景识别模块确定电子设备所处的用户场景的过程,具体参阅S501～S536,在此不再赘述。

[0354] 步骤S910,场景识别引擎确定将第二场景信息发送至性能调控应用。

[0355] 示例性的,电子设备可以预先为不同的用户场景分配唯一标识,该场景信息则可包括用户场景所的唯一标识。

[0356] 步骤S911,在第二时间点,性能调控应用根据第二场景信息将第一性能参数调节为第三参数值,第三参数值与第二参数值不同。

[0357] 第三参数值可以是一个参数集合,包含TDP、TPP、TGP、DB的具体数值。第三参数值

与第二参数值不同，指的是第三参数值和第二参数值中，存在至少一个参数的对应数值不同。例如，第三参数值中的TGP数值和第二参数值中的TGP数值不同。

[0358] 示例性地，第三参数值例如为TDP30W、TPP80W、TGP50W，DB15W。通过增大TGP和DB，使得电子设备可以提供充足的GPU性能，提高游戏体验。

[0359] 步骤S912，性能调控应用通过WMI接口将第三参数值发送至BIOS。

[0360] 步骤S913，BIOS将第三参数值写入ACPI指的位置，并通知GPU驱动读取第二参数值。

[0361] 具体地BIOS将第三参数值写入ACPI指的第一性能参数的存储位置，并通知GPU驱动读取第三参数值。

[0362] 步骤S914，GPU驱动读取第三参数值，并根据第三参数值调控GPU的性能。

[0363] GPU读取第三参数值的过程参见S805的描述，在此不再赘述。

[0364] 图10为本申请实施例提供的性能调控方法的又一示例调控过程。

[0365] 如图10所示，本申请实施例提供的性能调控方法的示例调控过程包括：

[0366] 步骤S1001，响应于用于操作启动第三应用，在启动所述第三应用之前，所述电子设备的第二性能参数为第四参数值。

[0367] 示例性地，第三应用例如办公应用，例如各种办公软件，例如word。

[0368] 所述第二性能参数包括所述GPU的Dx值。示例性地，所述第四参数值例如为D1。

[0369] 步骤S1002，场景识别引擎根据第三应用确定电子设备处于第三用户场景。

[0370] 示例性地，第三用户场景为办公场景。关于场景识别模块确定电子设备所处的用户场景的过程，具体参阅S501～S536，在此不再赘述。

[0371] 步骤S1003，场景识别引擎确定将第三场景信息发送至性能调控应用。

[0372] 示例性的，电子设备可以预先为不同的用户场景分配唯一标识，该场景信息则可包括用户场景所的唯一标识。

[0373] 步骤S1004，在第三时间点，性能调控应用根据第三场景信息将第二性能参数调节为第五参数值，第五参数值与第四参数值不同。

[0374] 示例性地，第五参数值例如为D4或D5。办公场景下GPU性能需求较低，因此可以通过调节Dx值直接限制GPU性能，以提高设备续航。

[0375] 步骤S1005，性能调控应用通过WMI接口将第五参数值发送至BIOS。

[0376] 步骤S1006，BIOS将第五参数值写入ACPI指的位置，并通知GPU驱动读取第五参数值。

[0377] 具体地BIOS将第五参数值写入ACPI指的第二性能参数的存储位置，并通知GPU驱动读取第五参数值。第二性能参数的存储位置与第一性能参数的存储位置可以相同，也可以不同。

[0378] 步骤S1007，GPU驱动读取第五参数值，并根据第五参数值调控GPU的性能。

[0379] GPU读取第五参数值的过程参见S805的描述，在此不再赘述。

[0380] 步骤S1008，响应于用于操作启动第四应用。

[0381] 示例性地，第四应用例如游戏应用，例如3A大型游戏

[0382] 步骤S1009，场景识别引擎根据第四应用确定电子设备处于第四用户场景。

[0383] 示例性地，第四用户场景为游戏场景。关于场景识别模块确定电子设备所处的用

户场景的过程,具体参阅S501～S536,在此不再赘述。

[0384] 步骤S1010,场景识别引擎确定将第四场景信息发送至性能调控应用。

[0385] 示例性的,电子设备可以预先为不同的用户场景分配唯一标识,该场景信息则可包括用户场景所的唯一标识。

[0386] 步骤S1011,在第四时间点,性能调控应用根据第四场景信息将第二性能参数调节为第六参数值,第六参数值与第五参数值不同。

[0387] 示例性地,第六参数值例如为D1。

[0388] 步骤S1012,性能调控应用通过WMI接口将第六参数值发送至BIOS。

[0389] 步骤S1013,BIOS将第五参数值写入ACPI指的位置,并通知GPU驱动读取第五参数值。

[0390] 具体地BIOS将第五参数值写入ACPI指的第二性能参数的存储位置,并通知GPU驱动读取第五参数值。

[0391] 步骤S1014,GPU驱动读取第五参数值,并根据第五参数值调控GPU的性能。

[0392] GPU读取第五参数值的过程参见S805的描述,在此不再赘述。

[0393] 可以理解的是,电子设备为了实现上述功能,其包含了执行各个功能相应的硬件和/或软件模块。结合本文中所公开的实施例描述的各示例的算法步骤,本申请能够以硬件或硬件和计算机软件的结合形式来实现。某个功能究竟以硬件还是计算机软件驱动硬件的方式来执行,取决于技术方案的特定应用和设计约束条件。本领域技术人员可以结合实施例对每个特定的应用来使用不同方法来实现所描述的功能,但是这种实现不应认为超出本申请的范围。

[0394] 一个示例中,图11示出了本申请实施例的一种装置300的示意性框图。装置300可包括:处理器301和收发器/收发管脚302,可选地,还包括存储器303。

[0395] 装置300的各个组件通过总线304耦合在一起,其中总线304除包括数据总线之外,还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见,在图中将各种总线都称为总线304。

[0396] 可选地,存储器303可以用于前述方法实施例中的指令。该处理器301可用于执行存储器303中的指令,并控制接收管脚接收信号,以及控制发送管脚发送信号。

[0397] 装置300可以是上述方法实施例中的电子设备或电子设备的芯片。

[0398] 其中,上述方法实施例涉及的各步骤的所有相关内容均可以援引到对应功能模块的功能描述,在此不再赘述。

[0399] 上述本申请实施例提供的一种参数监控方法所执行的步骤,也可以由电子设备100中包括的一种芯片系统来执行,其中,该芯片系统可以包括处理器。该芯片系统可以与存储器耦合,使得该芯片系统运行时调用该存储器中存储的计算机程序,实现上述电子设备100执行的步骤。其中,该芯片系统中的处理器可以是应用处理器也可以是非应用处理器的处理器。

[0400] 以上所述,以上实施例仅用以说明本申请的技术方案,而非对其限制;尽管参照前述实施例对本申请进行了详细的说明,本领域的普通技术人员应当理解:其依然可以对前述各实施例所记载的技术方案进行修改,或者对其中部分技术特征进行等同替换;而这些修改或者替换,并不使相应技术方案的本质脱离本申请各实施例技术方案的范围。

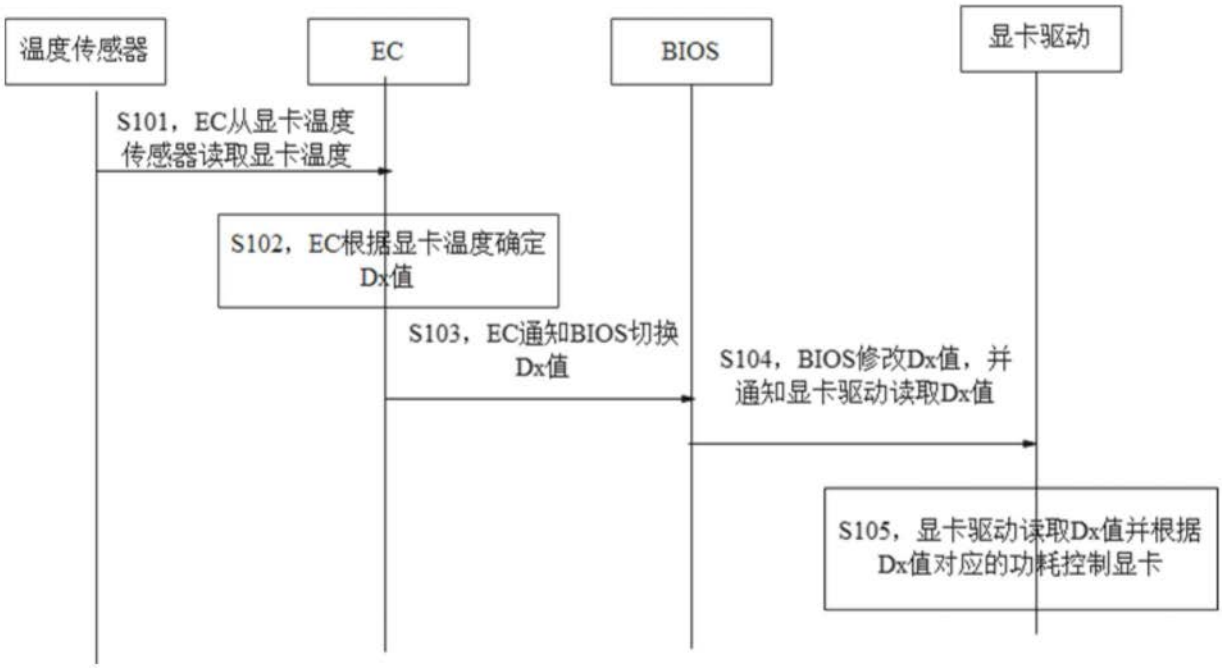

温度传感器
EC
BIOS
显卡驱动
S101，EC从显卡温度传感器读取显卡温度
S102，EC根据显卡温度确定Dx值
S103，EC通知BIOS切换Dx值
S104，BIOS修改Dx值，并通知显卡驱动读取Dx值
S105，显卡驱动读取Dx值并根据Dx值对应的功耗控制显卡

图1

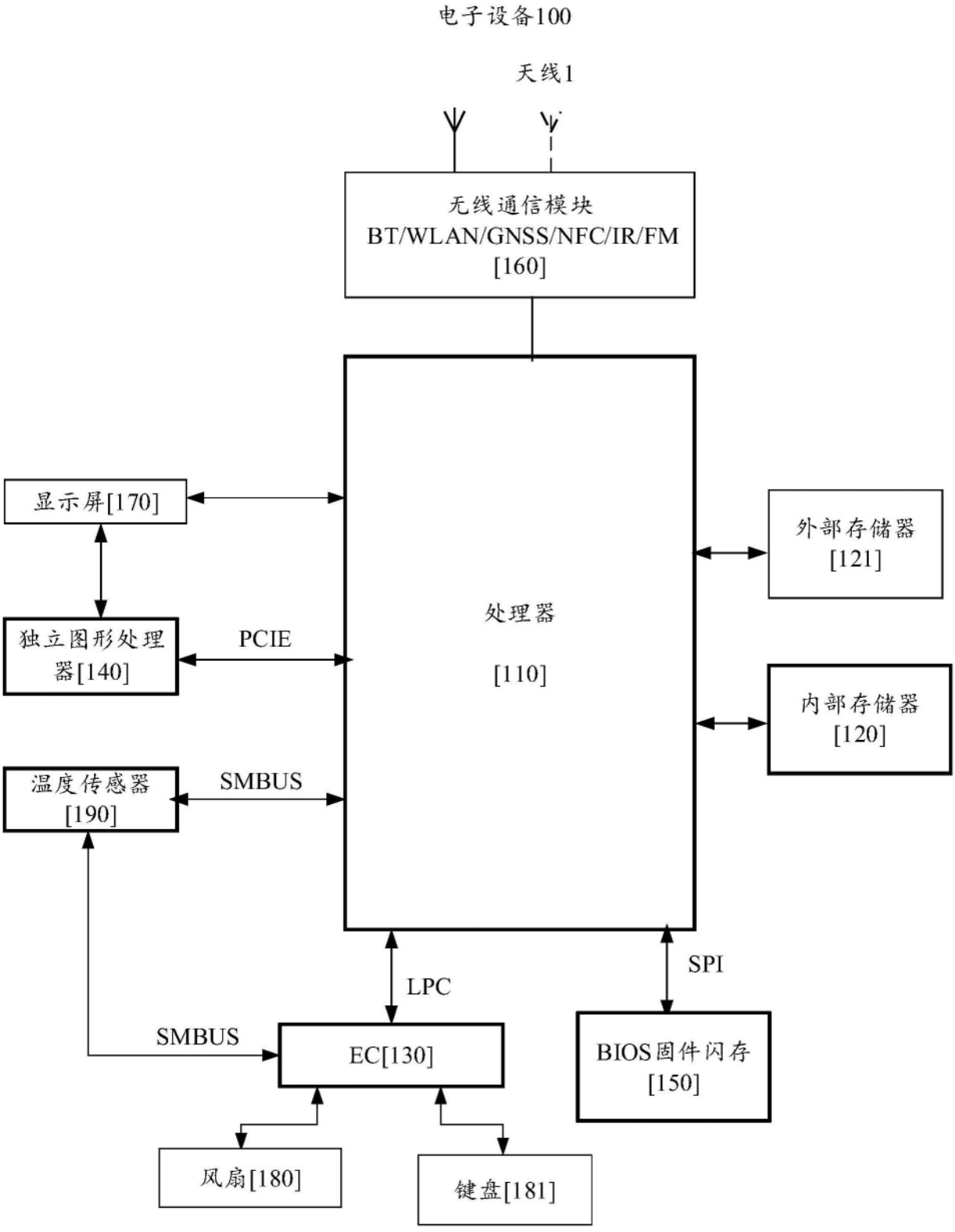

电子设备100
天线1
无线通信模块
BT/WLAN/GNSS/NFC/IR/FM
[160]
显示屏[170]
外部存储器
[121]
处理器
[110]
独立图形处理
器[140]
PCIE
内部存储器
[120]
温度传感器
[190]
SMBUS
LPC
SPI
SMBUS
EC[130]
BIOS固件闪存
[150]
风扇[180]
键盘[181]

图2

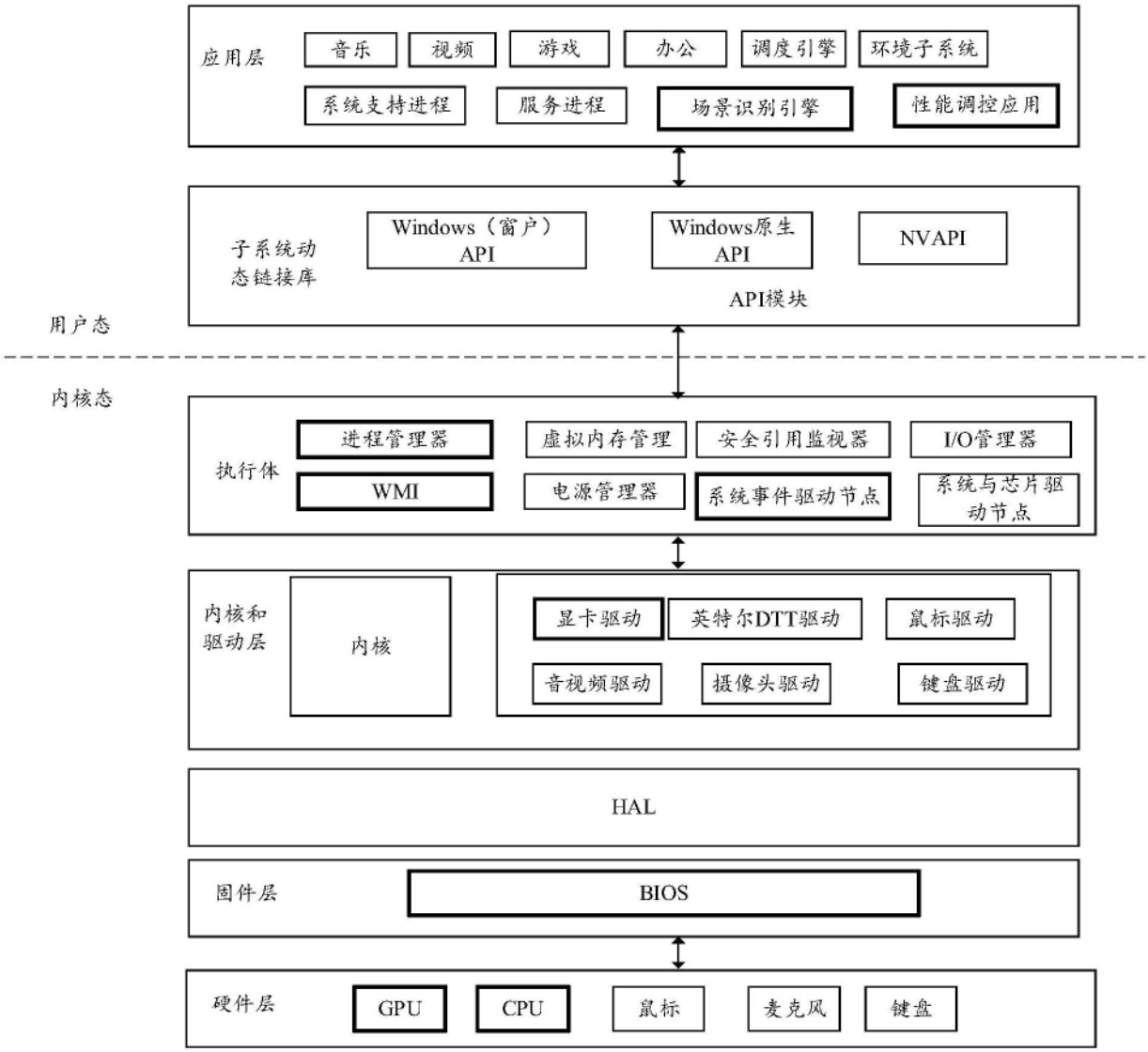

应用层
音乐
视频
游戏
办公
调度引擎
环境子系统
系统支持进程
服务进程
场景识别引擎
性能调控应用
子系统动态链接库
Windows（窗户）API
Windows原生API
NVAPI
API模块
用户态
内核态
执行体
进程管理器
虚拟内存管理
安全引用监视器
I/O管理器
WMI
电源管理器
系统事件驱动节点
系统与芯片驱动节点
内核和驱动层
内核
显卡驱动
英特尔DTT驱动
鼠标驱动
音视频驱动
摄像头驱动
键盘驱动
HAL
固件层
BIOS
硬件层
GPU
CPU
鼠标
麦克风
键盘

图3

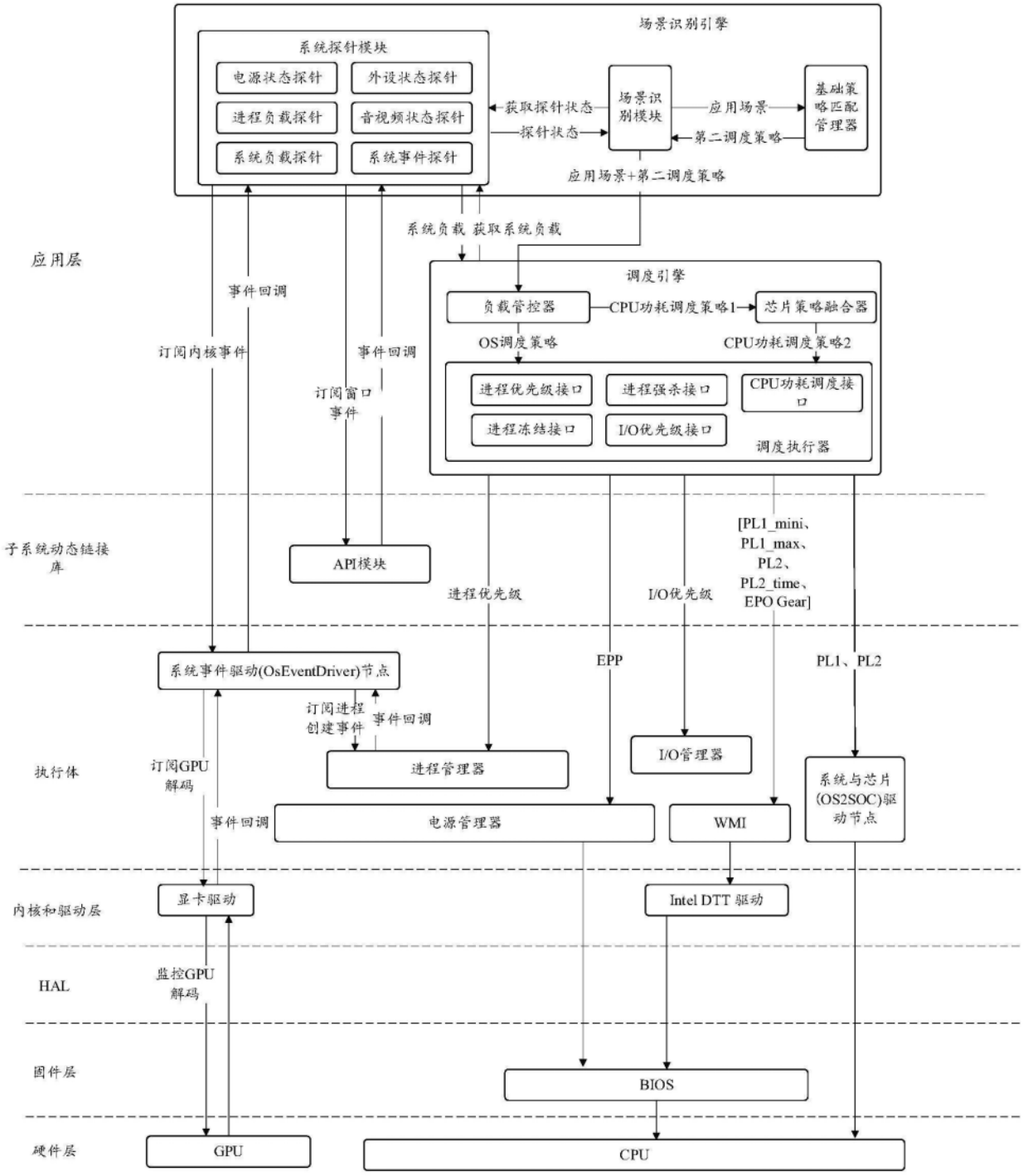

场景识别引擎
系统探针模块
电源状态探针
外设状态探针
进程负载探针
音视频状态探针
系统负载探针
系统事件探针
获取探针状态
探针状态
场景识别模块
应用场景
第二调度策略
基础策略匹配管理器
应用场景+第二调度策略
系统负载 获取系统负载
应用层
事件回调
调度引擎
负载管控器
CPU功耗调度策略1
芯片策略融合器
OS调度策略
CPU功耗调度策略2
订阅内核事件
事件回调
订阅窗口事件
进程优先级接口
进程强杀接口
CPU功耗调度接口
进程冻结接口
I/O优先级接口
调度执行器
子系统动态链接库
API模块
进程优先级
I/O优先级
[PL1_mini、PL1_max、PL2、PL2_time、EPO Gear]
系统事件驱动(OsEventDriver)节点
EPP
PL1、PL2
订阅进程创建事件
事件回调
执行体
订阅GPU解码
进程管理器
I/O管理器
系统与芯片(OS2SOC)驱动节点
事件回调
电源管理器
WMI
内核和驱动层
显卡驱动
Intel DTT 驱动
HAL
监控GPU解码
固件层
BIOS
硬件层
GPU
CPU

图4

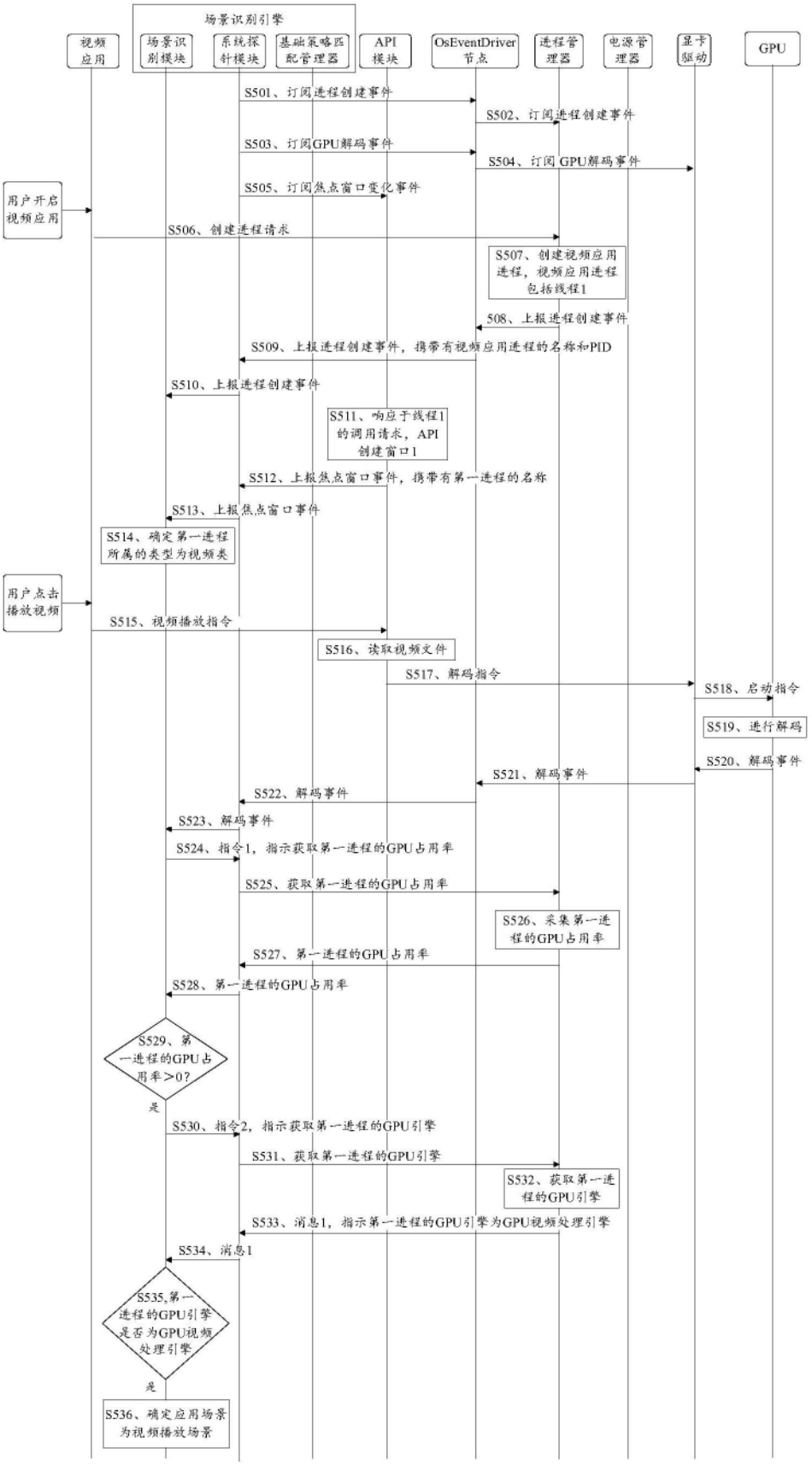

场景识别引擎
视频应用
场景识别模块
系统探针模块
基础策略匹配管理器
API模块
OsEventDriver节点
进程管理器
电源管理器
显卡驱动
GPU
S501、订阅进程创建事件
S502、订阅进程创建事件
S503、订阅GPU解码事件
S504、订阅GPU解码事件
S505、订阅焦点窗口变化事件
用户开启视频应用
S506、创建进程请求
S507、创建视频应用进程，视频应用进程包括线程1
S508、上报进程创建事件
S509、上报进程创建事件，携带有视频应用进程的名称和PID
S510、上报进程创建事件
S511、响应于线程1的调用请求，API创建窗口1
S512、上报焦点窗口事件，携带有第一进程的名称
S513、上报焦点窗口事件
S514、确定第一进程所属的类型为视频类
用户点击播放视频
S515、视频播放指令
S516、读取视频文件
S517、解码指令
S518、启动指令
S519、进行解码
S520、解码事件
S521、解码事件
S522、解码事件
S523、解码事件
S524、指令1，指示获取第一进程的GPU占用率
S525、获取第一进程的GPU占用率
S526、采集第一进程的GPU占用率
S527、第一进程的GPU占用率
S528、第一进程的GPU占用率
S529、第一进程的GPU占用率>0?
是
S530、指令2，指示获取第一进程的GPU引擎
S531、获取第一进程的GPU引擎
S532、获取第一进程的GPU引擎
S533、消息1，指示第一进程的GPU引擎为GPU视频处理引擎
S534、消息1
S535,第一进程的GPU引擎是否为GPU视频处理引擎
是
S536、确定应用场景为视频播放场景

图5

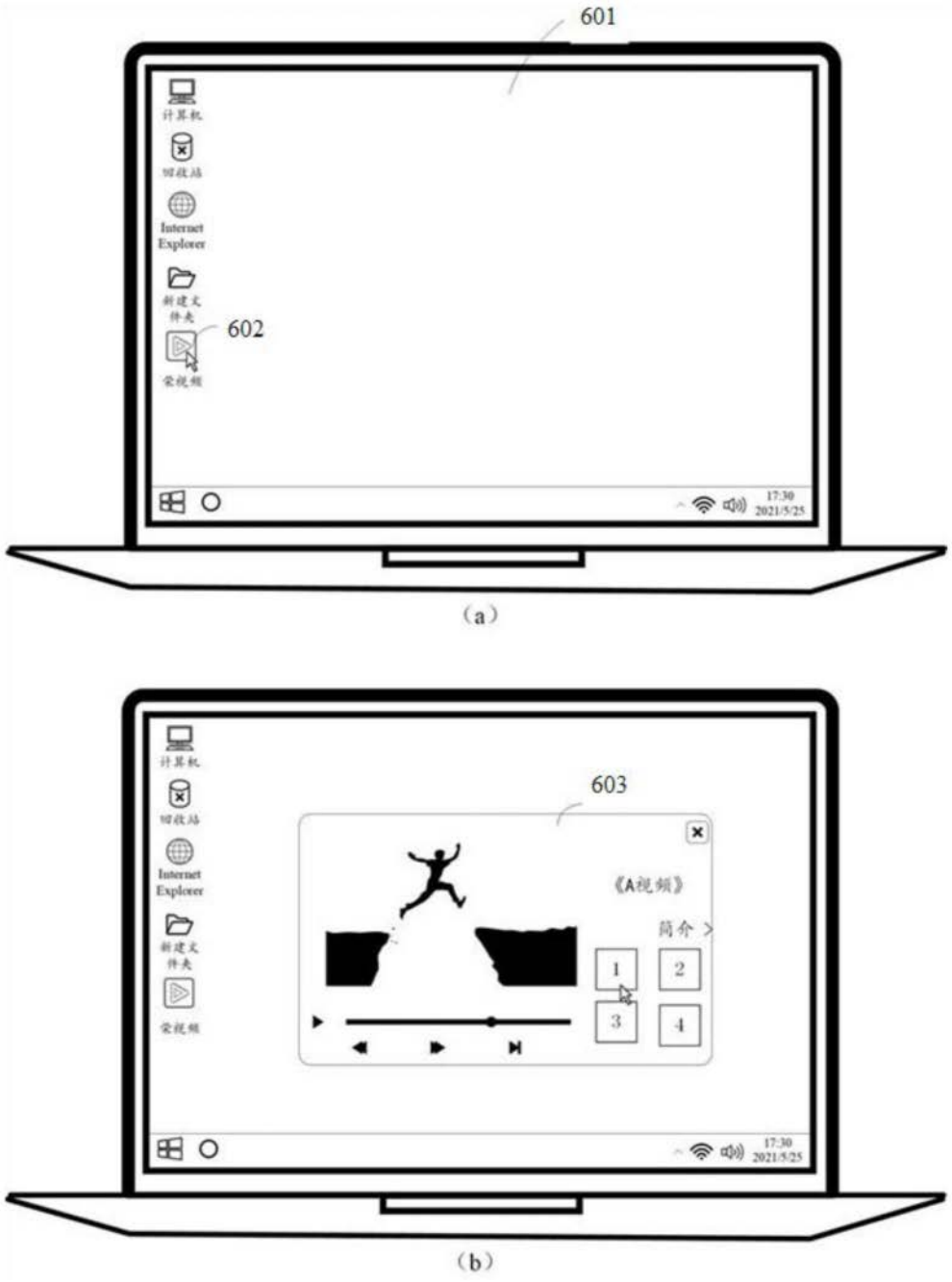

601
计算机
回收站
Internet Explorer
新建文件夹
602
常视频
17:30
2021/5/25
(a)
603
《A视频》
简介 >
1
2
3
4
(b)

图6

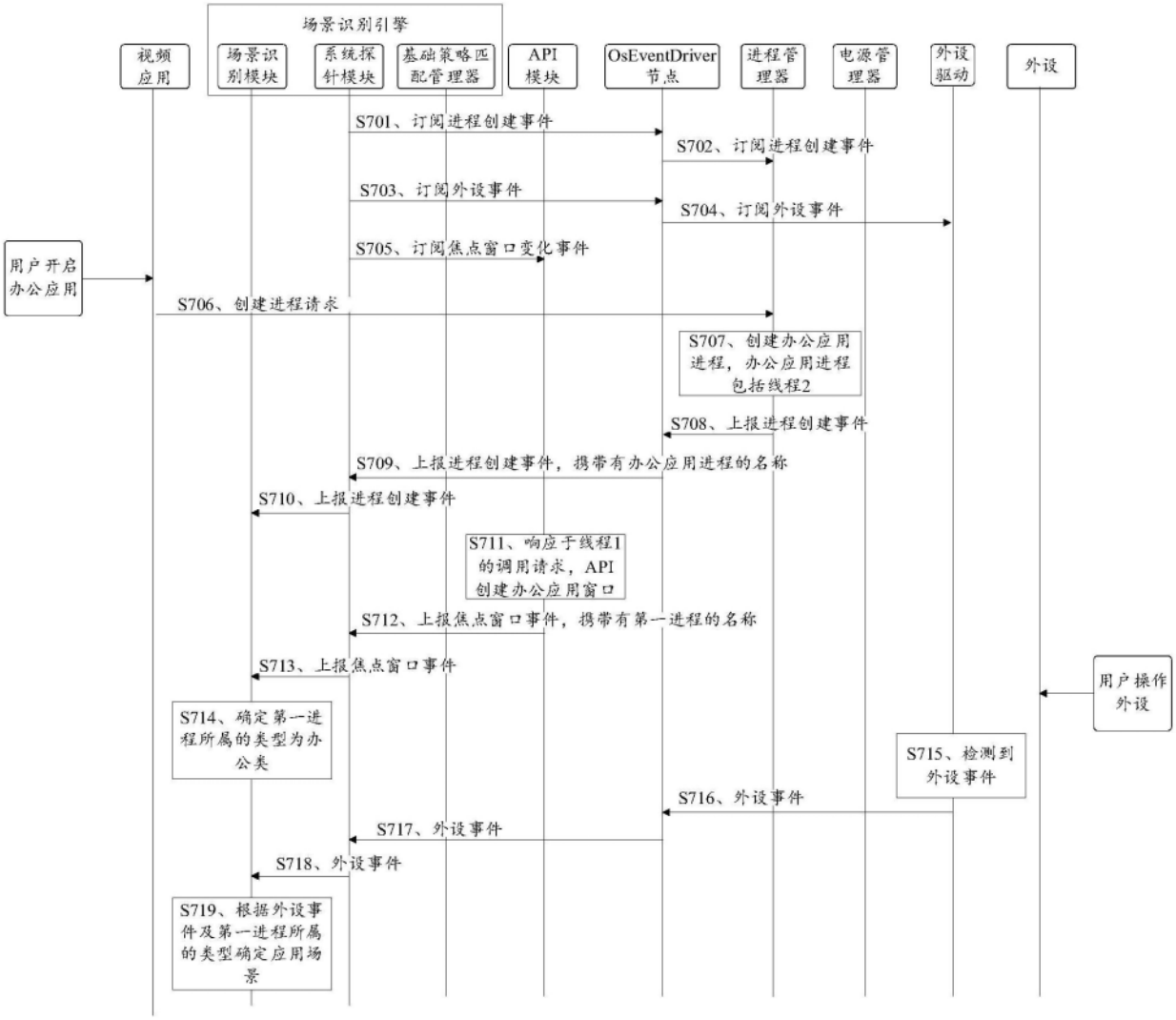

场景识别引擎
视频应用
场景识别模块
系统探针模块
基础策略匹配管理器
API模块
OsEventDriver节点
进程管理器
电源管理器
外设驱动
外设
用户开启办公应用
S701、订阅进程创建事件
S702、订阅进程创建事件
S703、订阅外设事件
S704、订阅外设事件
S705、订阅焦点窗口变化事件
S706、创建进程请求
S707、创建办公应用进程，办公应用进程包括线程2
S708、上报进程创建事件
S709、上报进程创建事件，携带有办公应用进程的名称
S710、上报进程创建事件
S711、响应于线程1的调用请求，API创建办公应用窗口
S712、上报焦点窗口事件，携带有第一进程的名称
S713、上报焦点窗口事件
用户操作外设
S714、确定第一进程所属的类型为办公类
S715、检测到外设事件
S716、外设事件
S717、外设事件
S718、外设事件
S719、根据外设事件及第一进程所属的类型确定应用场景

图7

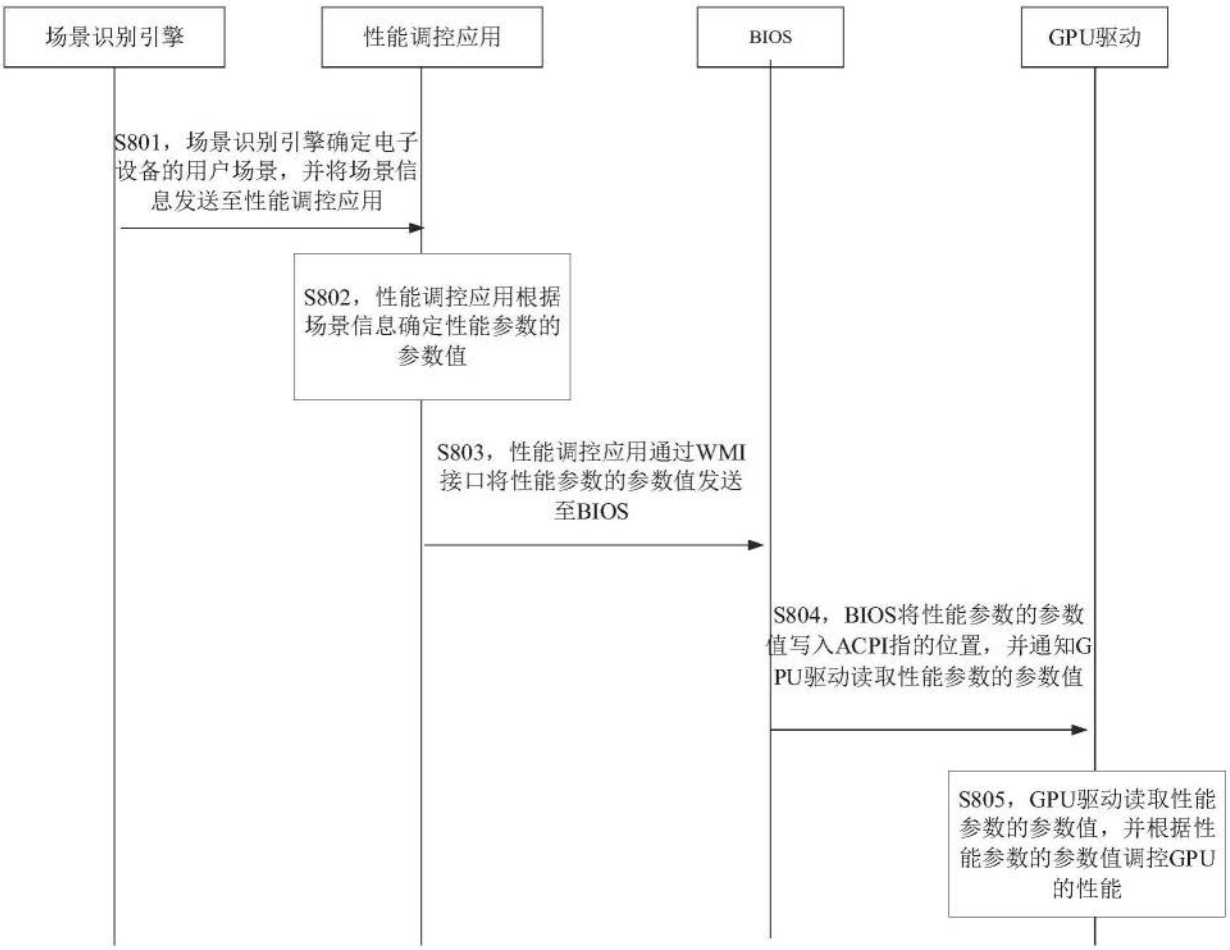

场景识别引擎
性能调控应用
BIOS
GPU驱动
S801，场景识别引擎确定电子设备的用户场景，并将场景信息发送至性能调控应用
S802，性能调控应用根据场景信息确定性能参数的参数值
S803，性能调控应用通过WMI接口将性能参数的参数值发送至BIOS
S804，BIOS将性能参数的参数值写入ACPI指的位置，并通知GPU驱动读取性能参数的参数值
S805，GPU驱动读取性能参数的参数值，并根据性能参数的参数值调控GPU的性能

图8

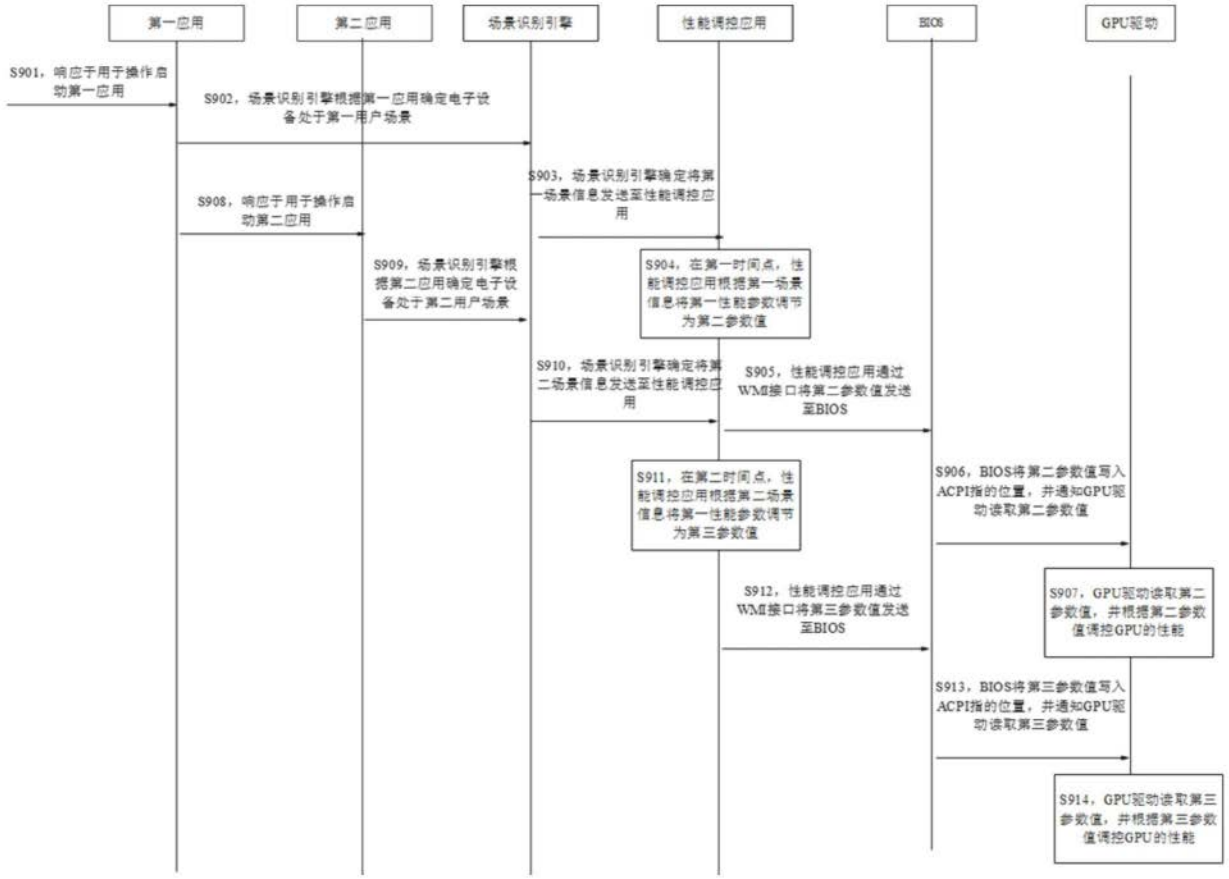

第一应用
第二应用
场景识别引擎
性能调控应用
BIOS
GPU驱动
S901，响应于用于操作启动第一应用
S902，场景识别引擎根据第一应用确定电子设备处于第一用户场景
S903，场景识别引擎确定将第一场景信息发送至性能调控应用
S908，响应于用于操作启动第二应用
S904，在第一时间点，性能调控应用根据第一场景信息将第一性能参数调节为第二参数值
S909，场景识别引擎根据第二应用确定电子设备处于第二用户场景
S910，场景识别引擎确定将第二场景信息发送至性能调控应用
S905，性能调控应用通过WMI接口将第二参数值发送至BIOS
S911，在第二时间点，性能调控应用根据第二场景信息将第一性能参数调节为第三参数值
S906，BIOS将第二参数值写入ACPI指的位置，并通知GPU驱动读取第二参数值
S912，性能调控应用通过WMI接口将第三参数值发送至BIOS
S907，GPU驱动读取第二参数值，并根据第二参数值调控GPU的性能
S913，BIOS将第三参数值写入ACPI指的位置，并通知GPU驱动读取第三参数值
S914，GPU驱动读取第三参数值，并根据第三参数值调控GPU的性能

图9

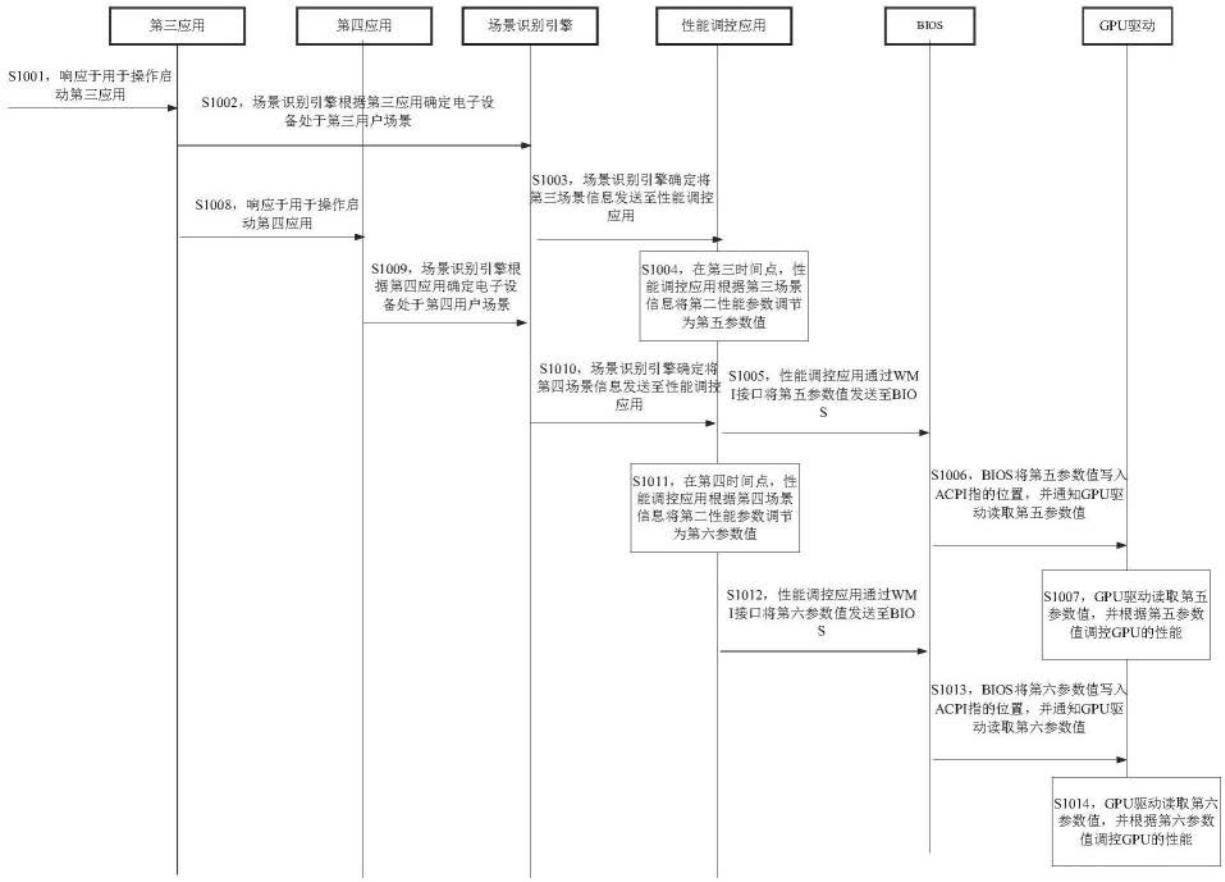

第三应用
第四应用
场景识别引擎
性能调控应用
BIOS
GPU驱动
S1001，响应于用于操作启动第三应用
S1002，场景识别引擎根据第三应用确定电子设备处于第三用户场景
S1003，场景识别引擎确定将第三场景信息发送至性能调控应用
S1008，响应于用于操作启动第四应用
S1004，在第三时间点，性能调控应用根据第三场景信息将第二性能参数调节为第五参数值
S1009，场景识别引擎根据第四应用确定电子设备处于第四用户场景
S1010，场景识别引擎确定将第四场景信息发送至性能调控应用
S1005，性能调控应用通过WMI接口将第五参数值发送至BIOS
S1011，在第四时间点，性能调控应用根据第四场景信息将第二性能参数调节为第六参数值
S1006，BIOS将第五参数值写入ACPI指的位置，并通知GPU驱动读取第五参数值
S1012，性能调控应用通过WMI接口将第六参数值发送至BIOS
S1007，GPU驱动读取第五参数值，并根据第五参数值调控GPU的性能
S1013，BIOS将第六参数值写入ACPI指的位置，并通知GPU驱动读取第六参数值
S1014，GPU驱动读取第六参数值，并根据第六参数值调控GPU的性能

图10

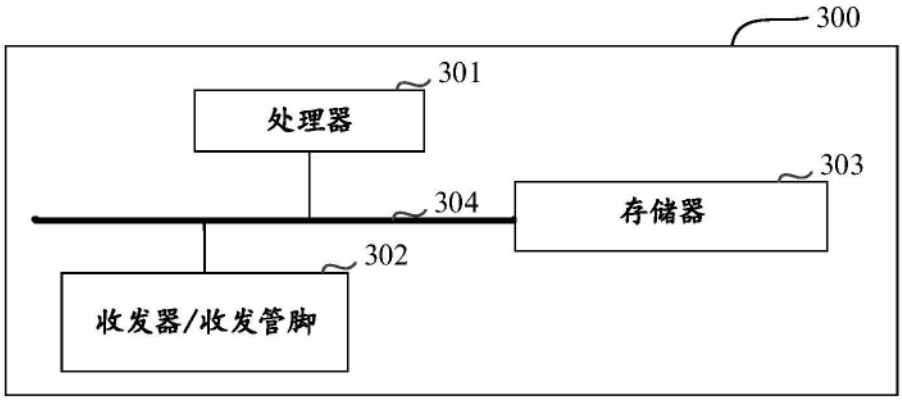

300
301
处理器
303
304
存储器
302
收发器/收发管脚

图11