(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104350478 B
(45)授权公告日 2017.06.16

(21)申请号 201380030019.4

(22)申请日 2013.07.30

(65)同一申请的已公布的文献号
申请公布号 CN 104350478 A

(43)申请公布日 2015.02.11

(30)优先权数据
61/677,390 2012.07.30 US
13/782,433 2013.03.01 US

(85)PCT国际申请进入国家阶段日
2014.12.08

(86)PCT国际申请的申请数据
PCT/US2013/052780 2013.07.30

(87)PCT国际申请的公布数据
WO2014/022431 EN 2014.02.06

(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 S·阿哈玛德

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 袁玥

(51)Int.Cl.
G06F 12/02(2006.01)

(56)对比文件
US 2008162610 A1,2008.07.03,说明书第[0032]段.
US 2007136401 A1,2007.06.14,全文.
US 2007174369 A1,2007.07.26,全文.
US 2003221047 A1,2003.11.27,说明书第[0011],[0015],[0025]-[0028],[0032]-[0042],[0004],[0015],[0058]-[0059]段.
US 2009248766 A1,2009.10.01,全文.

审查员 吴海旋

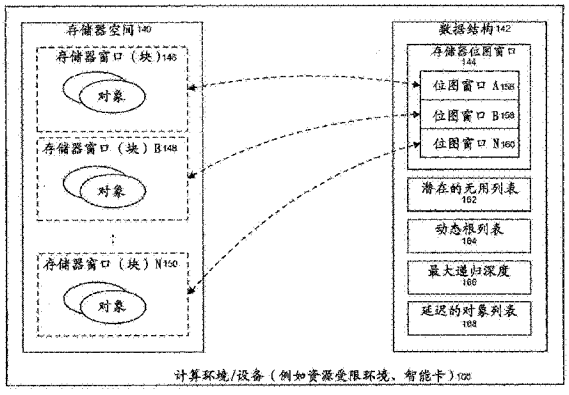
权利要求书2页 说明书6页 附图7页

(54)发明名称

用于使用位图窗口的持久性存储器中对象删除的系统和方法

(57)摘要

一种用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的系统和方法。根据实施例,该系统一般可以与使用持久性存储器的诸如智能卡、Java卡和其他资源受限的环境的计算环境(100)一起使用。根据实施例,该系统包括处理器或计算引擎以及用于软件对象的存储的持久性存储器;以及包括一或多个存储器位图窗口(144)的数据结构(142),每个存储器位图窗口代表持久性存储器(140)中可寻址空间的块(146、148、150);其中该系统在从持久性存储器删除不可获得的对象时使用一或多个存储器位图窗口。



CN 104350478 B

1.一种用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的系统，包括：

计算环境或设备，包括处理器或计算引擎和持久性存储器，所述持久性存储器被划分为多个存储器块，每个存储器块保存一个或多个根对象和一个或多个与之相关联的子对象；

数据结构，能够包括一或多个存储器位图窗口，每个存储器位图窗口代表不同的存储器块；以及

其中所述系统通过以下步骤，使用所述数据结构以标记不可获得的对象以用于从持久性存储器删除：

将在特定存储器块中未被引用的根对象放入潜在的无用列表中，

确定所述根对象的子对象在与所述特定存储器块不同的存储器块中，并且执行以下步骤之一：

如果所述子对象在已经检查过的之前存储器块中，则从潜在的无用列表中移除所述根对象，和

如果所述子对象在尚未被检查的下一存储器块中，则使用所述根对象作为起始点来检查下一存储器块以检查用于删除的对象，重复该放入步骤和确定步骤，直到检查过每个存储器块，和

通过标示对应的位图窗口中的对应比特来标记潜在的无用列表中的用于删除的每个对象。

2.如权利要求1所述的系统，其中所述数据结构包括以下各项中的一或多个：

最大递归深度；以及

延迟的对象列表。

3.如权利要求2所述的系统，其中

所述最大递归深度是限制用于标记阶段的递归深度的可配置值；以及

一旦达到被配置的递归深度，所述延迟的对象列表包括还未被处理的子对象。

4.如权利要求3所述的系统，其中所述对应比特代表存储器块中的地址。

5.一种用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的方法，包括以下步骤：

提供包括处理器或计算引擎和持久性存储器的计算环境或设备，所述持久性存储器被划分为多个存储器块，每个存储器块保存一个或多个根对象和一个或多个与之相关联的子对象；

提供数据结构，该数据结构能够包括一或多个存储器位图窗口，每个存储器位图窗口代表不同的存储器块；

将在特定存储器块中未被引用的根对象放入潜在的无用列表中，

确定所述根对象的子对象在与所述特定存储器块不同的存储器块中，并且执行以下步骤之一：

如果所述子对象在已经检查过的之前存储器块中，则从潜在的无用列表中移除所述根对象，和

如果所述子对象在尚未被检查的下一存储器块中，则使用所述根对象作为起始点来检

查下一存储器块以检查用于删除的对象，重复该放入步骤和确定步骤，直到检查过每个存储器块，和

通过标示对应的位图窗口中的对应比特来标记潜在的无用列表中的用于删除的每个对象。

6.如权利要求5所述的方法，其中所述数据结构进一步包括以下各项中的一或多个：

最大递归深度；以及

延迟的对象列表。

7.如权利要求6所述的方法，其中

所述最大递归深度是限制用于标记阶段的递归深度的可配置值；以及

一旦达到被配置的递归深度，所述延迟的对象列表包括还未被处理的子对象。

8.如权利要求7所述的方法，其中，所述对应比特代表存储器块中的地址。

9.一种用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的系统，包括：

用于提供包括处理器或计算引擎和持久性存储器的计算环境或设备的装置，所述持久性存储器被划分为多个存储器块，每个存储器块保存一个或多个根对象和一个或多个与之相关联的子对象；

用于提供数据结构的装置，所述数据结构能够包括一或多个存储器位图窗口，每个存储器位图窗口代表不同的存储器块；

用于将在特定存储器块中未被引用的根对象放入潜在的无用列表中的装置，

用于确定所述根对象的子对象在与所述特定存储器块不同的存储器块中的装置，以及

用于执行以下步骤之一的装置：

如果所述子对象在已经检查过的之前存储器块中，则从潜在的无用列表中移除所述根对象，和

如果所述子对象在尚未被检查的下一存储器块中，则使用所述根对象作为起始点来检查下一存储器块以检查用于删除的对象，

用于重复该放入步骤和确定步骤，直到检查过每个存储器块的装置，和

用于通过标示对应的位图窗口中的对应比特来标记潜在的无用列表中的用于删除的每个对象的装置。

10.如权利要求9所述的系统，其中，所述数据结构进一步包括以下各项中的一或多个：

最大递归深度；以及

延迟的对象列表。

11.如权利要求10所述的系统，其中

所述最大递归深度是限制用于标记阶段的递归深度的可配置值；以及

一旦达到被配置的递归深度，所述延迟的对象列表包括还未被处理的子对象。

12.如权利要求11所述的系统，其中，所述对应比特代表存储器块中的地址。

用于使用位图窗口的持久性存储器中对象删除的系统和方法

[0001] 版权说明

[0002] 本专利文件中的一部分公开内容包含受版权保护的素材。当专利文献或专利公开出现在可公开获得的美国专利商标局文件或档案中时，版权所有人不反对任何人影印再现专利文献或专利公开，不过保留其它方面的所有版权。

技术领域

[0003] 本发明的实施例主要涉及持久性存储器存储，且特别地涉及资源受限的环境和其它计算环境，以及用于使用代表存储器块的位图窗口从持久性存储器删除不可获得的对象的系统和方法。

背景技术

[0004] 诸如智能卡或类似执行环境或设备的典型资源受限的环境可能具有有限的随机存取存储器资源，并且反而使用持久性存储器作为用于创建和存储软件对象的默认位置。

[0005] 这样的环境需要回收由于从定义的根的集合不可获得而已经而变得无用的对象所使用的持久性存储器。在比如Java SE的其他环境中可能有用的无用收集算法通常依赖于比如直接移动或标记对象的技术。然而，这些技术在资源受限的智能卡或类似执行环境中不再同样有用，因为考虑到在持久性存储器设备用尽之前只可以对持久性存储器设备做出有限数量的数据写入，在持久性存储器中移动或标记对象是耗时的且潜在具有破坏性的操作。

发明内容

[0006] 此处描述的是一种用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的系统和方法。根据实施例，该系统可以一般性地与使用持久性存储器的计算环境(诸如智能卡、Java卡和其他资源受限的环境)一起使用。根据实施例，该系统包括处理器或计算引擎和用于软件对象的存储的持久性存储器；以及可以包括一或多个存储器位图窗口的数据结构，每个位图窗口代表持久性存储器中的可寻址空间的块；其中该系统在从持久性存储器时删除不可获得对象中使用该一或多个存储器位图窗口。

附图说明

[0007] 图1根据实施例图解了使用诸如资源受限的环境或智能卡的持久性存储器的计算环境的示例。

[0008] 图2根据实施例图解了用于在持久性存储器中对象删除的代表存储器块的位图窗口的使用。

[0009] 图3根据实施例进一步图解了用于在持久性存储器中对象删除的位图窗口的使用。

[0010] 图4根据实施例进一步图解了用于在持久性存储器中对象删除的位图窗口的使

用。

[0011] 图5是根据实施例图解了用于在持久性存储器中对象删除的使用位图窗口的算法或方法的流程图。

[0012] 图6是根据实施例进一步图解用于在持久性存储器中对象删除的使用位图窗口的算法或方法的流程图。

[0013] 图7是根据实施例进一步图解用于在持久性存储器中对象删除的使用位图窗口的算法或方法的流程图。

具体实施方式

[0014] 如上所述，典型的资源受限的环境(诸如智能卡或类似执行环境或设备)可能具有有限的随机存取存储器资源，并且反而使用持久性存储器作为用于创建和存储软件对象的默认位置。这样的环境需要回收由于从定义的根的集合不可获得而已经而变得无用的对象所使用的持久性存储器。然而，在其他环境中可能有用但是依赖于比如直接移动或标记对象的技术的无用收集算法在资源受限的智能卡或类似执行环境中不再同样有用，因为考虑到在持久性存储器设备用尽之前只可以对持久性存储器设备做出有限数量的数据写入，在持久性存储器中移动或标记对象是耗时的且潜在具有破坏性的操作。

[0015] 可以被用于解决这个问题的方法的种类的一些示例包括：使用蛮力算法检查每个对象以确定它是否被从系统中的任何其他对象引用；使用引用计数算法，该算法使用引用计数更新对象以使得引用计数为0指示对象未被另一个对象引用；或使用代表存储器中对象的RAM位图。

[0016] 这些方法中的前两种的问题是它们不考虑对于即使简单循环的循环引用。第一种方法也耗时，因为在最差的情况下潜在地不得不针对系统中每一个其他对象检查系统中的每个对象以确定从一个对象到另一个的引用。第三种方法的问题是该方法依赖于解引用对象，使得该系统使用来自对象表而不是真正地址的对象引用；然而这使得整个执行环境变缓，因为很多操作将需要对象地址，并且将不得不完成对话进程以从引用获得地址。

[0017] 根据实施例，此处描述的是用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的系统和方法。根据实施例，该系统可以一般性地与使用持久性存储器的计算环境(诸如智能卡、Java卡和其他资源受限的环境)一起使用。根据实施例，该系统包括处理器或计算引擎和用于软件对象的存储的持久性存储器；以及可以包括一或多个存储器位图窗口的数据结构，每个位图窗口代表持久性存储器中的可寻址空间的块；其中该系统在从持久性存储器中删除不可获得对象时使用该一或多个存储器位图窗口。

[0018] 根据各个实施例，此处描述的系统和方法的一些优点包括能够提供解决循环引用问题的标记-清理无用收集技术的快速版本；不需要解引用对象；同样，直到存储器的真正回收才需要向持久性存储器写入；以及该方法可扩展以使得整体目标存储器面积的变大不增大资源需求。

[0019] 图1根据实施例图解了诸如资源受限的环境或智能卡的使用持久性存储器的计算环境的示例。

[0020] 如图1中所示，根据实施例，诸如资源受限的环境、智能卡或其他环境的计算环境或设备100可以被认为是小型计算设备，该小型计算设备包括可以包括用于执行计算操作

的电路的处理器或计算引擎126;以及诸如随机存取存储器(RAM)120、电可擦可编程只读存储器(EEPROM)122和/或只读存储器(ROM)124的一或多个存储器。

[0021] 根据实施例,该RAM可以包括易失随机存取存储器,该EEPROM可以包括可写非易失存储器,以及该ROM可以包括只读存储器。该RAM可以被用于存储诸如系统堆栈121的数据项和数据结构。该ROM可以包括诸如Java虚拟机的虚拟机108,该虚拟机允许以诸如Java的与平台无关的编程语言所写的应用由智能卡执行;并且也可以包括为访问智能卡的客户提供服务的一或多个应用104、105。根据实施例,应用106也可以位于EEPROM中或在ROM和/或EEPROM中。

[0022] 根据实施例,该系统可以包括卡管理器102或类似逻辑,该卡管理器包括用于管理在计算环境或设备上应用的执行(包括例如允许客户和应用之间的通信)以及从存储器删除对象的代码。例如,如图1中所示,在设备的操作过程中,EEPROM可以包括多个对象112、113、114,这些对象由应用104、105访问(更具体地,应用104访问对象112,应用105访问对象113和114,应用106访问对象114);而已经变得不相连的其他对象115、116不再被任何应用引用。一般希望删除这些未引用的对象以释放存储器空间。

[0023] 通过对可以使用此处描述的系统和方法的一种环境的说明提供图1中图解的环境。根据各个实施例,本发明不限于资源受限的环境或智能卡或以上描述的特定的环境或配置,并且通常可以与可在可写持久性或非易失存储器中存储对象的其他类型的计算环境、设备或系统一起使用。

[0024] 根据实施例,提供用于代表存储器的块的位图窗口的使用的系统和方法。该技术使用可以在非常有限的资源下工作并同时映射大量可寻址空间的算法或方法。该技术也可扩展以使得针对增加的持久性存储器尺寸,即使整体RAM尺寸维持相对不变,该算法或方法将仍然工作。附加地,该算法或方法不降低整体系统性能,而在蛮力、引用计数和解引用技术情况下,系统性能会降低。

[0025] 图2根据实施例图解了用于在持久性存储器中进行对象删除的代表存储器块的位图窗口的使用。

[0026] 如图2所示,根据实施例,系统可以包括数据结构142,该数据结构可以包括一或多个存储器位图窗口144(156、158、160),这些存储器位图窗口代表用于对象的存储的持久性存储器140中的可寻址空间的块146、148、150。该数据结构可以进一步包括潜在的无用列表162,该列表列出在已经处理过的窗口中未被引用的对象;动态根列表164,列出了从之前窗口引用的但是不在之前窗口中的可寻址空间中的、成为下一窗口的根的对象;最大递归深度166,该深度是限制用于标记阶段的递归深度的可配置值;以及延迟的对象列表168,一旦达到所配置的递归深度,该延迟的对象列表包括还未被处理的子对象。

[0027] 图3和图4进一步根据实施例图解了在用于持久性存储器中进行对象删除的位图窗口的使用。

[0028] 根据实施例,为了避免在标记过程中对EEPROM的不必要写入,系统可以在RAM中维持位图。在例如典型的资源受限的环境的某些环境中,RAM是稀有资源;典型的量可能是2k,其为位图存储提供很少空间。举例来说,512字节的位图含有4k比特,且每一比特可以代表4个字节,所以这个尺寸的位图可以被用于代表16k的存储器。

[0029] 根据实施例,在对象删除过程期间,系统可以执行“标记”在存储器中所有可获得

的对象的标记过程。这可以通过跟随从“根”(诸如从小程序表和静态变量)的引用链来完成,以找到可获得对象。例如,小程序对象可以指向另一个对象,该对象又指向其他对象,这些其他对象的每一个也指向其他对象。类似的,静态变量可以指向对象,这些对象又指向其他对象。

[0030] 根据实施例,在标记阶段期间,从窗口的根对象开始,它的子对象被标记。如果子对象位于窗口外部并且在下一窗口中,那么该对象可作为用于下一窗口的根被放到动态根列表中。如果子对象在该窗口内,那么它可以被标记,且该系统可以在该对象的子对象上递归。如果子对象在之前窗口中,那么该系统检查潜在的无用列表并在该对象上递归,并随后将它从潜在的无用列表移除,因为它可从该窗口被寻址。在针对窗口的标记阶段之后,系统寻找未被标记的对象,并将它们放在潜在的无用列表中。这些对象可能还不会真正被认为是无用的,因为它们可从下一窗口被寻址。该系统随后处理之后的窗口。最终,在完成这些窗口之后,潜在的无用列表将包含未被引用且可被认为无用的对象。

[0031] 如图3中所示,根据实施例,存储器空间被划分成窗口。针对窗口初始化位图,其中一比特代表存储器中一个地址。例如,如果以4字节对齐执行存储器管理,那么1比特可以代表在4字节存储器单元中的第一字节。随后,该位图窗口被处理。对于每个窗口,根对象被标记,包括如小程序表169所指示的在当前窗口内标记小程序对象170;标记从静态字段176引用的在窗口中的对象174;并标记来自动态根表的在窗口中的对象178。

[0032] 根据实施例,子对象被随后标记,包括如果在当前窗口内发现子对象171、172,通过标记184在位图中对应比特标记该对象。如果在潜在无用列表中当前窗口外发现子对象180,那么该对象和它的子对象被在当前窗口和潜在无用列表中递归地标记。如果在当前窗口外而不是潜在无用列表中发现子对象,那么,如果该对象在存储器的已经处理过的区域中,则它已经被标记为可获得,且因此它可以被忽略。然而,如果该子对象位于存储器的仍然需要被处理的区域中,那么它被放到动态根列表中。

[0033] 根据实施例,在标记子对象时,如果达到最大递归深度,那么在该层次中的下一对象被放在延迟的对象列表上,并且将在当前层次的处理的结尾处被处理。如果该延迟的对象列表为满,且已经达到最大递归深度,那么在当前窗口中重启标记过程,而不重新设置位图,从而增加在第二和随后循环中的根的数量。一旦在窗口内完成标记阶段,在窗口内的任何未被引用的对象都被放在潜在无用列表上。

[0034] 如图4中所示,过程继续到下一窗口和它的根和子对象185、186、187、188。当所有的窗口已经被处理时,潜在无用列表将仅包括未被引用且因此无用的对象189。根据实施例,这些对象可以被删除以回收已使用的存储器。

[0035] 图5是根据实施例示出用于在持久性存储器中进行对象删除的使用位图窗口的算法或方法190的流程图。

[0036] 如图5所示,在步骤192处,存储空间被划分成窗口。

[0037] 在步骤194处,针对窗口初始化位图,其中一比特代表存储器中的一个地址。

[0038] 在步骤196、198、202处,位图窗口被处理。根据实施例,针对每个窗口,根对象被标记,包括标记在当前窗口内的小程序对象,标记从静态字段引用的在窗口中的对象,以及标记来自动态根表的在窗口中的对象。之后,子对象按照以下进一步详细描述的标准被标记,且位图被更新。当前窗口中的任何未被引用的对象被放到潜在无用列表上。

[0039] 在步骤204处，当所有的窗口都已经被处理时，潜在无用列表将只包括未被引用且因此无用的对象。根据实施例，这些对象可以被删除以回收已被使用的存储器。

[0040] 图6是根据实施例进一步图解用于在持久性存储器中进行对象删除的使用位图窗口的算法或方法210的流程图。特别地，图6根据实施例进一步图解了根和子对象的标记可以如何被执行。

[0041] 如图6中所示，在步骤212处，针对每个窗口，根对象被标记，包括标记在当前窗口内小程序对象、标记从静态字段引用的在窗口中的对象以及标记来自动态根字段的在窗口中的对象。

[0042] 在步骤214处，子对象被标记，包括如果在当前窗口内发现子对象，通过标记在位图中的对应比特标记该对象。如果在当前窗口外在潜在无用列表中发现子对象，那么该对象和它的子对象在当前窗口和潜在列表中被递归地标记。如果子对象被发现在当前窗口外且不在潜在无用列表中，那么，如果该对象在存储器的已经处理过的区域中，它已经被标记为可获得的，且因此可以被忽略。然而，如果该子对象位于存储器的仍然需要被处理的区域中，那么它被放到动态根列表中。

[0043] 在步骤216处，当标记子对象时，如果达到最大递归深度，那么在层次中的下一对象被放在延迟的对象列表上，并且将在当前层次的处理的结尾处被处理。

[0044] 在步骤220处，如果该延迟的对象列表为满，且已经达到最大递归深度，那么在当前窗口中重启标记过程，而不重新设置位图，从而增加在第二和随后循环中的根的数量。

[0045] 在步骤222处，一旦在窗口中完成标记阶段，在窗口内的任何未被引用的对象都被放在潜在无用列表上。如上所描述的，当所有的窗口都已经被处理时，潜在的无用列表将仅包括未被引用且因此无用的对象。根据实施例，这些对象可以被删除以回收被使用的存储器。

[0046] 图7是根据实施例进一步图解用于在持久性存储器中删除对象的使用位图窗口的算法或方法230的流程图。特别地，图7根据实施例图解了之前在图5和6中描述过的每个步骤的组合。

[0047] 总体而言，本发明涉及在一或多个微处理器上运行的、用于使用代表存储器块的位图窗口在持久性存储器中进行对象删除的系统，包括：

[0048] 用于在包括处理器或计算引擎和用于软件对象的存储的持久性存储器的计算环境或设备(诸如资源受限的环境、智能卡或其他环境)处提供数据结构的装置，该数据结构可以包括一或多个位图窗口，每个位图窗口代表持久性存储器中可寻址空间的块。

[0049] 用于在从持久性存储器中删除不可获得的对象时使用一或多个存储器位图窗口的装置。

[0050] 所述系统进一步包括用于在数据结构内提供以下一或多个的装置：

[0051] 潜在无用列表；

[0052] 动态根列表；

[0053] 最大递归深度；以及

[0054] 延迟的对象列表。

[0055] 所述系统进一步包括：

[0056] 该潜在无用列表包括已经被处理的窗口中的未被引用的对象；

[0057] 该动态根列表包括从之前窗口引用的但不在之前窗口中的可寻址空间中的对象，该对象成为下一窗口的根；

[0058] 该最大递归深度是限制用于标记阶段的递归深度的可配置值；以及

[0059] 一旦达到配置的递归深度，该延迟对象列表包括还未被处理的子对象。

[0060] 所述系统进一步包括：

[0061] 用于将用于对象的存储的持久性存储器空间划分为窗口的装置；

[0062] 用于针对窗口初始化位图的装置，此处一比特代表存储器中一个地址；

[0063] 针对每个窗口，标记根，并递归地标记子对象；

[0064] 当在窗口中完成标记阶段时，用于将在该窗口内任意未被引用的对象放在潜在无用列表上的装置；以及

[0065] 用于处理每个窗口以使得潜在无用列表最终包括未被引用且可被认为无用的那些对象的装置。

[0066] 可以使用包括一或多个处理器、存储器和/或根据本公开的启示编程的计算机可读存储介质的一或多个传统通用或专用数字计算机、计算设备、机器或微处理器方便地实现本发明。熟练的编程人员可以基于本公开的启示容易地准备合适的软件代码，如软件领域的技术人员清楚的那样。

[0067] 在一些实施例中，本发明包括计算程序产品，该产品是其上/其中存储有可以被用来编程计算机以执行本发明的任意过程的指令的非易失存储介质或计算机可读介质。该存储介质可以包括但不限于任何种类的盘，包括软盘、光盘、DVD、CD-ROM、微驱动和磁光盘、ROM、RAM、EPROM、EEPROM、DRAM、VRAM、闪存设备、磁或光卡、纳米系统(包括分子存储器IC)或适合存储指令和/或数据的任何种类的介质或设备。

[0068] 本发明的以上描述仅仅被提供用于图解和描述的目的。它并不意图穷尽或限制本发明到所公开的具体形式。对于本领域技术人员而言许多修改和改变将是明显的。实施例被选择和描述从而最好地解释本发明的原则和实际的应用，从而允许本领域技术人员理解针对各个实施例的本发明以及具有适于特定用途的各种修改的本发明。意图由所附权利要求和它们的等价限定本发明的范围。

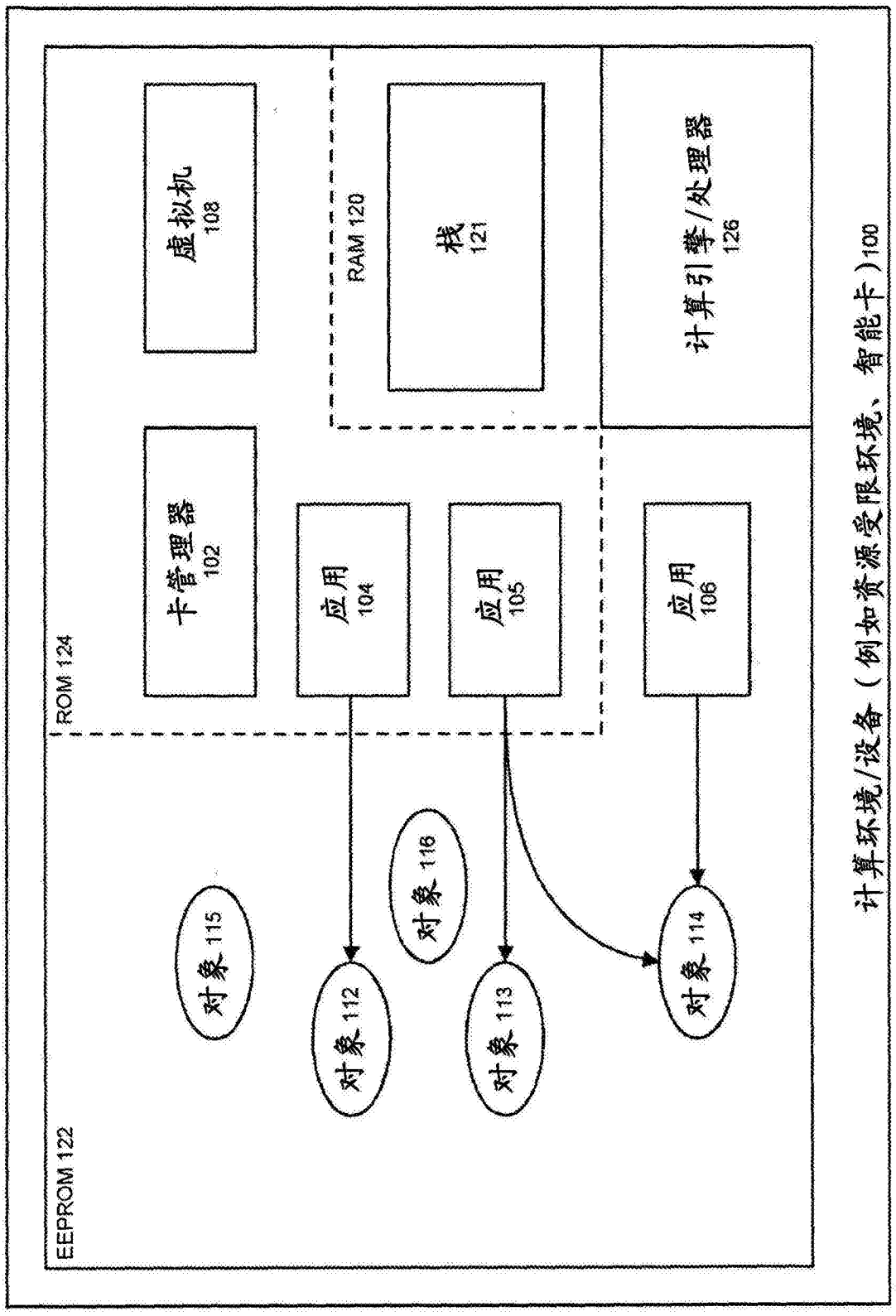

计算环境/设备（例如资源受限环境、智能卡）100
EEPROM 122
ROM 124
RAM 120
虚拟机
108
卡管理器
102
应用
104
应用
105
应用
106
栈
121
计算引擎/处理器
126
对象 115
对象 112
对象 116
对象 113
对象 114

图1

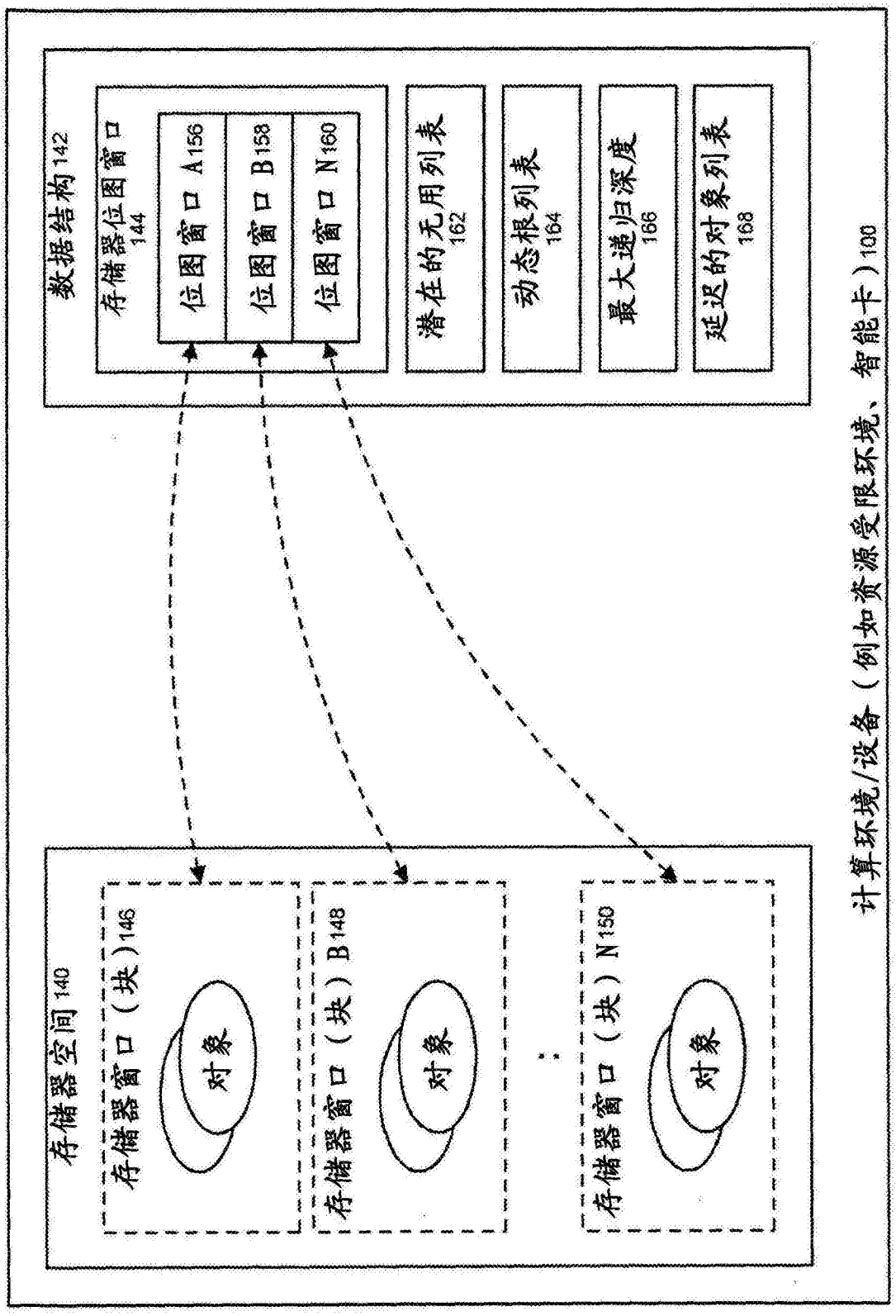

计算环境/设备（例如资源受限环境、智能卡）100
数据结构142
存储器位图窗口144
位图窗口 A156
位图窗口 B158
位图窗口 N160
潜在的无用列表162
动态根列表164
最大递归深度166
延迟的对象列表168
存储器空间140
存储器窗口（块）A146
存储器窗口（块）B148
存储器窗口（块）N150
对象
对象
对象

图2

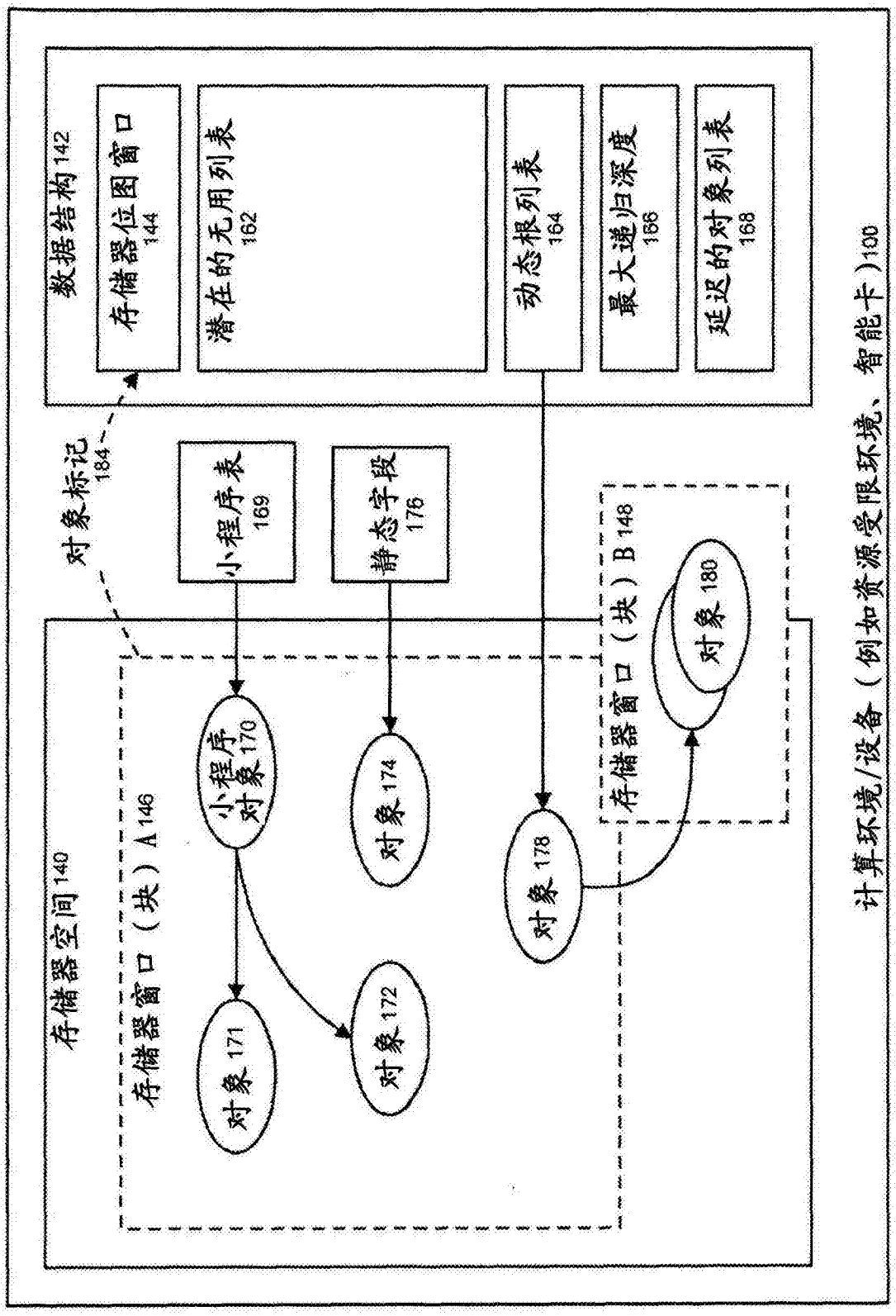

计算环境/设备（例如资源受限环境、智能卡）100
数据结构142
存储器位图窗口144
潜在的无用列表162
动态根列表164
最大递归深度166
延迟的对象列表168
对象标记184
小程序表169
静态字段176
存储器空间140
存储器窗口（块）A146
存储器窗口（块）B148
小程序对象170
对象171
对象172
对象174
对象178
对象180

图3

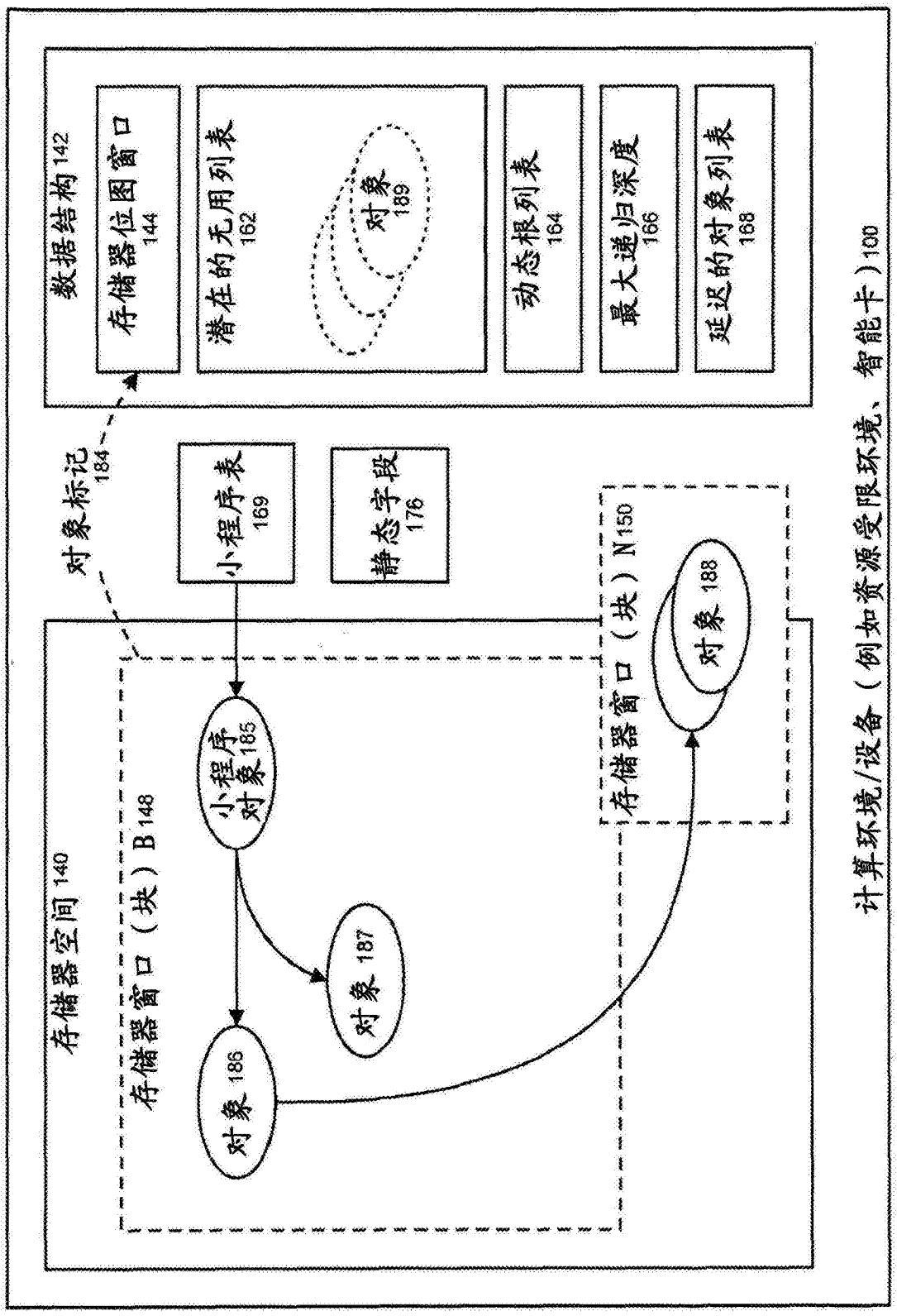

计算环境/设备（例如资源受限环境、智能卡）100
数据结构142
存储器位图窗口144
潜在的无用列表162
对象189
动态根列表164
最大递归深度166
延迟的对象列表168
对象标记184
小程序表169
静态字段176
存储器空间140
存储器窗口（块）B148
小程序对象185
对象186
对象187
存储器窗口（块）N150
对象188

图4

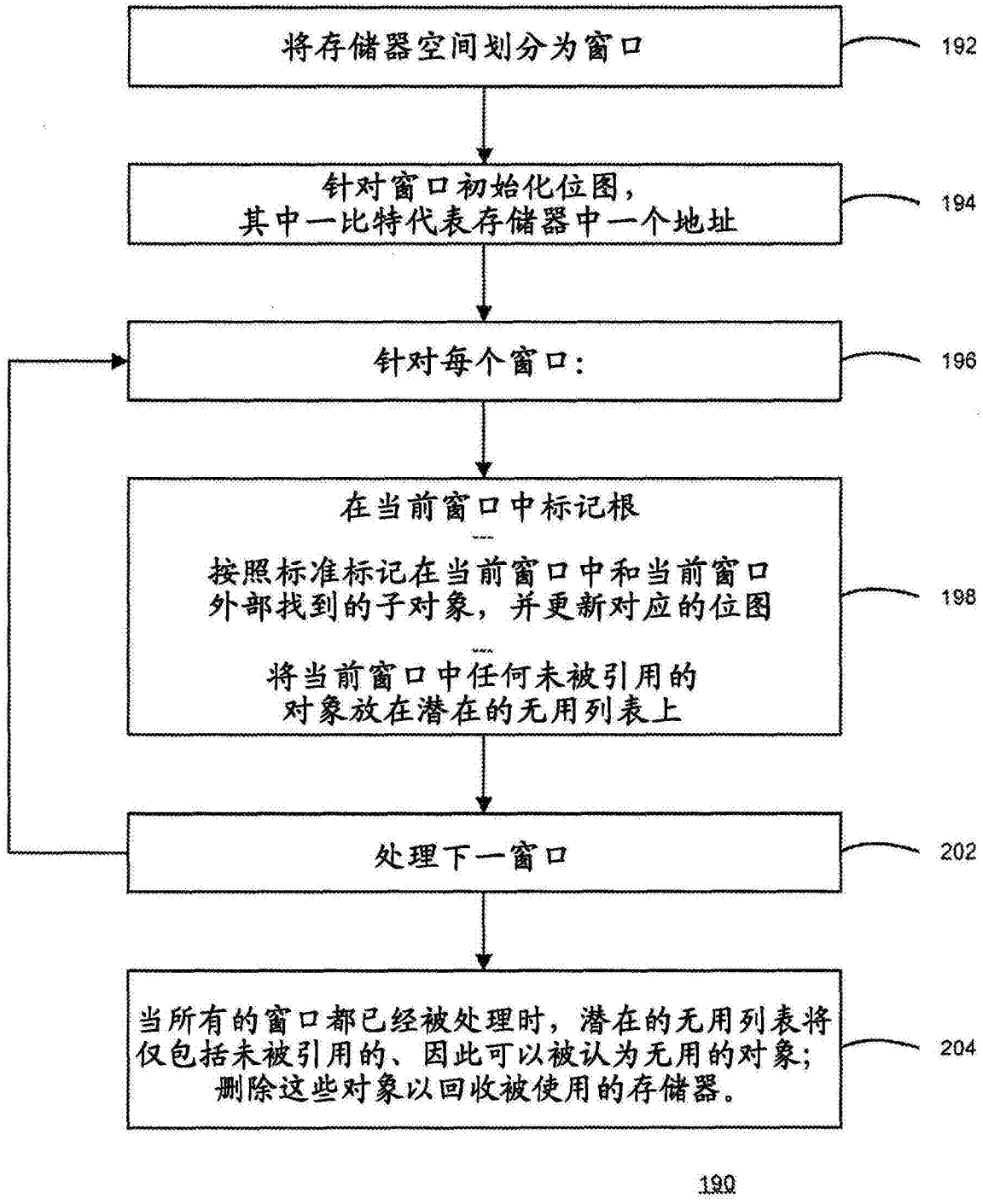

将存储器空间划分为窗口
192
针对窗口初始化位图，
其中一比特代表存储器中一个地址
194
针对每个窗口：
196
在当前窗口中标记根
按照标准标记在当前窗口中和当前窗口外部找到的子对象，并更新对应的位图
将当前窗口中任何未被引用的对象放在潜在的无用列表上
198
处理下一窗口
202
当所有的窗口都已经被处理时，潜在的无用列表将仅包括未被引用的、因此可以被认为无用的对象；删除这些对象以回收被使用的存储器。
204
190

图5

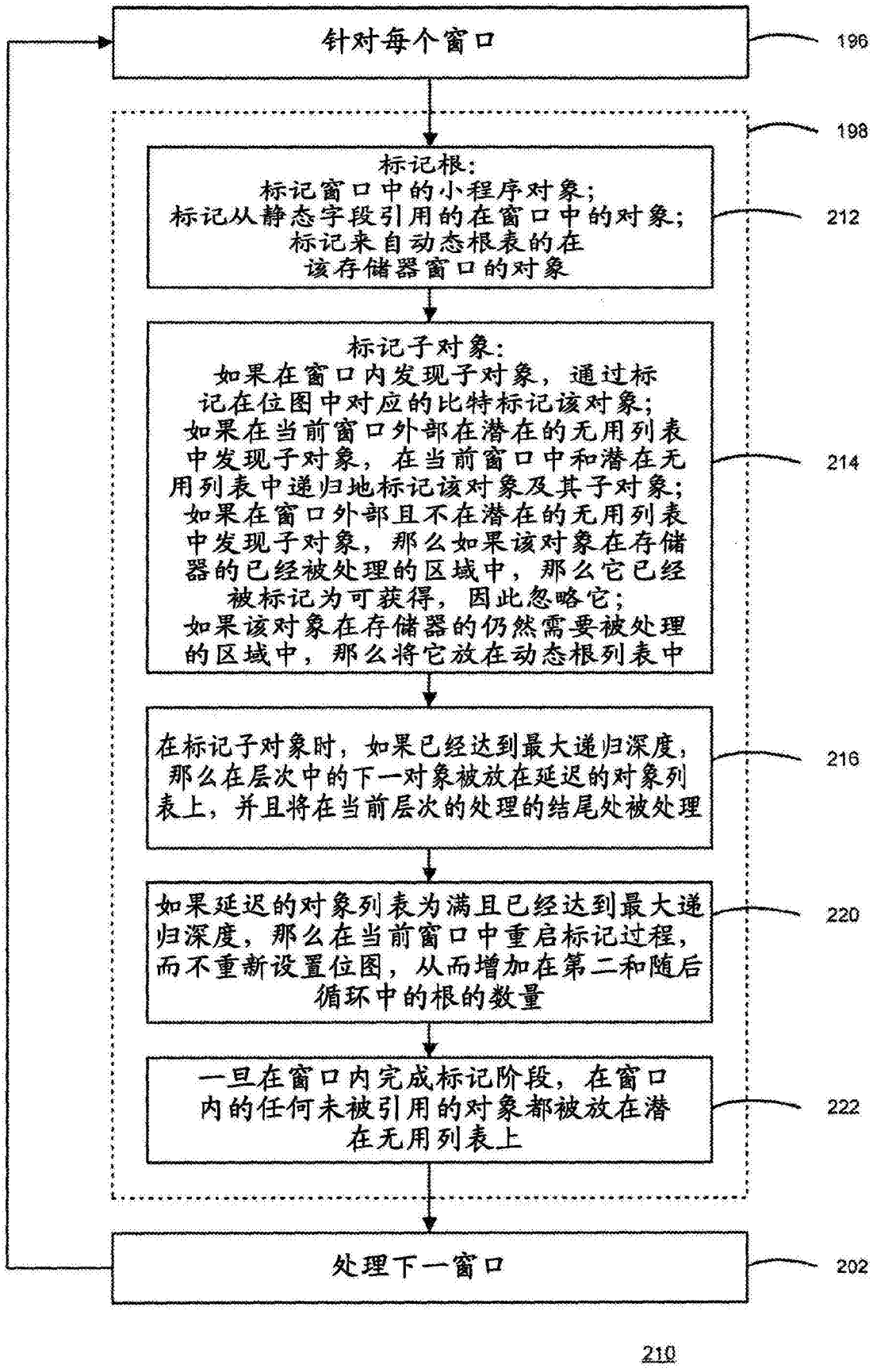

针对每个窗口
196
198
标记根：
标记窗口中的小程序对象；
标记从静态字段引用的在窗口中的对象；
标记来自动态根表的在
该存储器窗口的对象
212
标记子对象：
如果在窗口内发现子对象，通过标记在位图中对应的比特标记该对象；
如果在当前窗口外部在潜在的无用列表中发现子对象，在当前窗口中和潜在无用列表中递归地标记该对象及其子对象；
如果在窗口外部且不在潜在的无用列表中发现子对象，那么如果该对象在存储器的已经被处理的区域中，那么它已经被标记为可获得，因此忽略它；
如果该对象在存储器的仍然需要被处理的区域中，那么将它放在动态根列表中
214
在标记子对象时，如果已经达到最大递归深度，那么在层次中的下一对象被放在延迟的对象列表上，并且将在当前层次的处理的结尾处被处理
216
如果延迟的对象列表为满且已经达到最大递归深度，那么在当前窗口中重启标记过程，而不重新设置位图，从而增加在第二和随后循环中的根的数量
220
一旦在窗口内完成标记阶段，在窗口内的任何未被引用的对象都被放在潜在无用列表上
222
处理下一窗口
202
210

图6

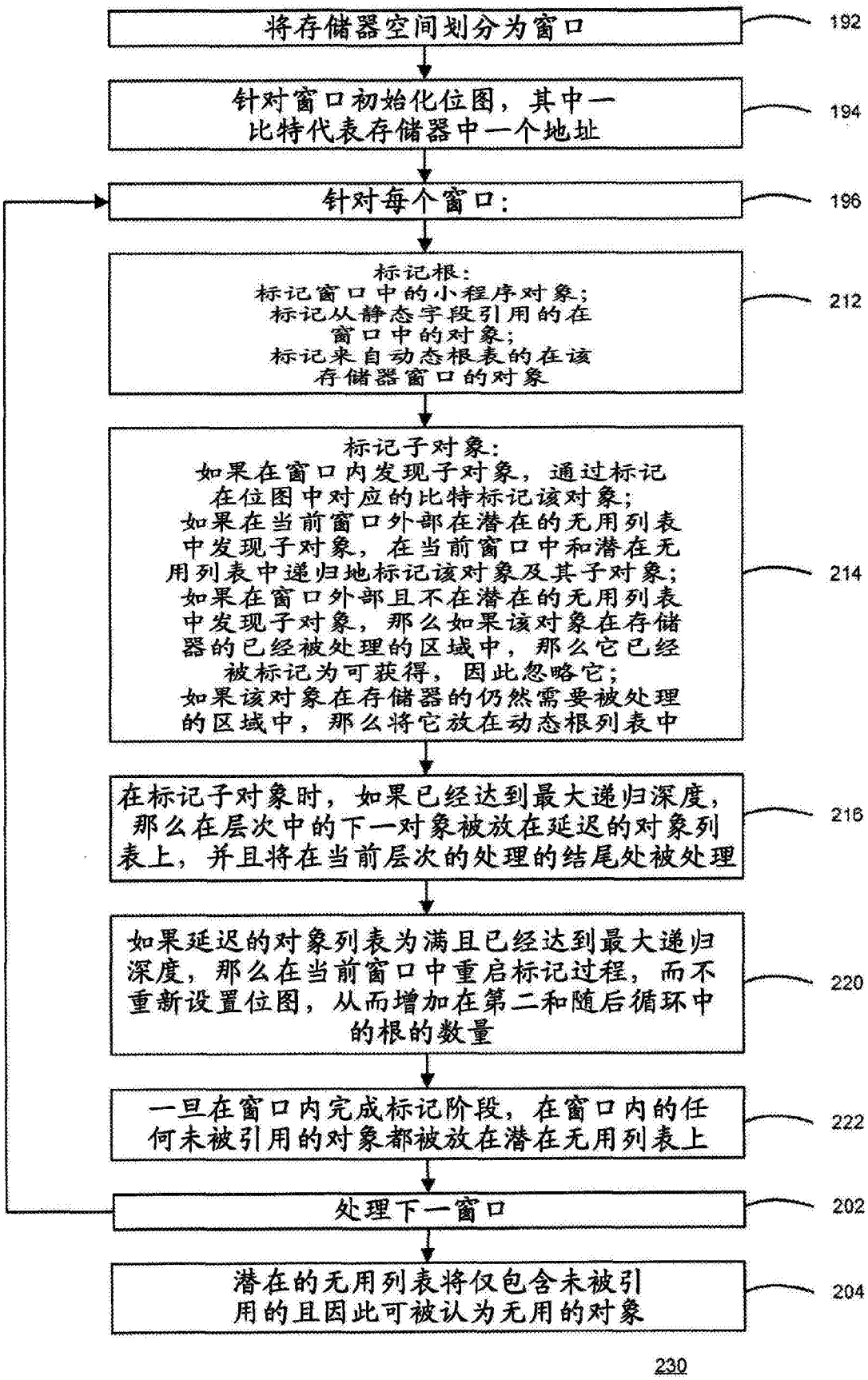

将存储器空间划分为窗口
192
针对窗口初始化位图，其中一比特代表存储器中一个地址
194
针对每个窗口：
196
标记根：
标记窗口中的小程序对象；
标记从静态字段引用的在窗口中的对象；
标记来自动态根表的在该存储器窗口的对象
212
标记子对象：
如果在窗口内发现子对象，通过标记在位图中对应的比特标记该对象；
如果在当前窗口外部在潜在的无用列表中发现子对象，在当前窗口中和潜在无用列表中递归地标记该对象及其子对象；
如果在窗口外部且不在潜在的无用列表中发现子对象，那么如果该对象在存储器的已经被处理的区域中，那么它已经被标记为可获得，因此忽略它；
如果该对象在存储器的仍然需要被处理的区域中，那么将它放在动态根列表中
214
在标记子对象时，如果已经达到最大递归深度，那么在层次中的下一对象被放在延迟的对象列表上，并且将在当前层次的处理的结尾处被处理
216
如果延迟的对象列表为满且已经达到最大递归深度，那么在当前窗口中重启标记过程，而不重新设置位图，从而增加在第二和随后循环中的根的数量
220
一旦在窗口内完成标记阶段，在窗口内的任何未被引用的对象都被放在潜在无用列表上
222
处理下一窗口
202
潜在的无用列表将仅包含未被引用的且因此可被认为无用的对象
204
230

图7