(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 106294199 B
(45)授权公告日 2019.11.05

(21)申请号 201610697224.1

(22)申请日 2012.04.06

(65)同一申请的已公布的文献号
申请公布号 CN 106294199 A

(43)申请公布日 2017.01.04

(30)优先权数据
13/093,724 2011.04.25 US

(62)分案原申请数据
201280020077.4 2012.04.06

(73)专利权人 微软技术许可有限责任公司
地址 美国华盛顿州

(72)发明人 S·卢科 C·C-C·曼

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 顾嘉运

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

(56)对比文件
US 2008098296 A1,2008.04.24,
US 7730016 B2,2010.06.01,
US 7325106 B1,2008.01.29,
MARLOW et al.Making a Fast Curry: Push/Enter vs. Eval/Apply for Higher-order Languages.《PROCEEDINGS OF THE ACM SIGPLAN 2004 INTERNATIONAL CONFERENCE ON FUNCTION PROGRAMMING》.2004,
BOEHM H-J et al.garbage collection in an uncooperative environment.《SOFTWARE PRACTIVE & EXPERIENCE, WILEY & SONS, BOGNOR REGIS,GB》.1988,第18卷(第9期),

审查员 赵鹏翔

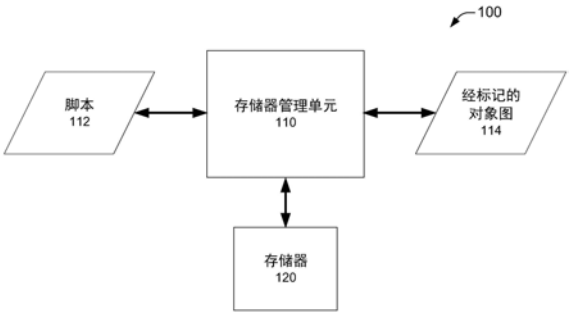
权利要求书2页 说明书9页 附图10页

(54)发明名称

用于存储器管理的保守垃圾收集和经标记的整数

(57)摘要

公开了用于保守垃圾收集的各方面。在一个方面,标识出包含在调用栈中的包括整数和指针的根对象。整数表示被标记并且可以同未经标记的指针表示相区分。跟踪根对象到相应的存储器位置,使得对指针表示执行后续跟踪并对整数表示跳过后续跟踪。然后,释放分配给调用栈不可到达的对象的存储器。在另一方面,标记与调用栈相关联的对象图,并且生成堆,所述堆包括包含在调用栈的经执行部分中的对象。跟踪调用栈的未经执行部分中所包括的对象到堆上的相应存储器位置,使得仅仅对未经标记的指针表示执行后续跟踪。然后,清除与调用栈的未经执行部分不可到达的堆对象相对应的存储器位置。




CN 106294199 B

1.一种用于存储器管理的保守垃圾收集的系统,包括:

至少一个处理器;以及

至少一个存储器,包括:

包括一个或多个根对象的调用栈,每个根对象包括整数表示和指针表示;

与所述调用栈相关联的对象图,所述对象图包括被标记的整数表示或未经标记的指针表示中的至少一个,被标记的整数表示可与未经标记的指针表示区分开来;以及

可在所述至少一个处理器上执行的组件,包括:

扫描组件,标识出在调用栈中的所述一个或多个根对象;

标记组件,标记在所述对象图中至少一个整数表示;

跟踪组件,执行对所述一个或多个根对象中的至少一个到对应的存储器位置的第一跟踪,执行对与未经标记的指针表示相关联的存储器位置的后续跟踪并跳过对与由所述对象图所标识的经标记的整数表示相关联的存储器位置的所述后续跟踪;以及

回收组件,释放分配给所述调用栈不可到达的一组对象的存储器。

2.如权利要求1所述的系统,其特征在于,所述回收组件被配置为将可到达对象保存在堆的原始存储位置中。

3.如权利要求2所述的系统,其特征在于,所述回收组件被配置为根据所述堆内的固定的一组边界来维护堆对象的分隔。

4.如权利要求1所述的系统,其特征在于,所述回收组件被配置为把被所述调用栈调用的对象放置到堆上,并且其中所述回收组件还被配置为基于所述堆的大小是否超过阈值来发起垃圾收集算法。

5.如权利要求1所述的系统,其特征在于,所述调用栈与一脚本的编译相关联。

6.如权利要求1所述的系统,其特征在于,所述调用栈与一对象模型的修改相关联。

7.一种用于存储器管理的保守垃圾收集的方法,包括:

在与调用栈相关联的对象图中标记整数表示,所述调用栈包括一个或多个根对象,所述对象图包括包含被标记的整数表示和未经标记的指针表示的至少一个根对象;

在所述调用栈的执行期间生成对象的堆,所述对象的堆包括调用栈的经执行部分中的根对象;

执行在所述调用栈的未经执行部分中包括的未经执行的根对象到所述堆上的相应存储器位置的第一跟踪;

执行对由所述对象图所标识的未经标记的指针表示相关联的存储器位置的后续跟踪,同时跳过对由所述对象图所标识的经标记的整数表示的所述后续跟踪;以及

清除堆上的与所述调用栈的未经执行部分中的一个或多个根对象不可到达的对象相对应的存储器位置。

8.如权利要求7所述的方法,其特征在于,进一步包括:

在所述调用栈的执行期间监视所述堆的大小,并且清除与对象相对应的存储器位置是由所述堆的大小超出阈值而触发的。

9.如权利要求7所述的方法,其特征在于,其中清除与对象相对应的存储器位置还包括:

在所述堆的原始存储器位置中保存可到达对象的存储,所述可到达对象是所述调用栈

的未经执行的部分可到达的。

10.如权利要求7所述的方法，其特征在于，所述调用栈与一脚本的编译相关联。

11.如权利要求7所述的方法，其特征在于，所述调用栈与一对象模型的修改相关联。

12.如权利要求7所述的方法，其特征在于，在与调用栈相关联的对象图中标记整数表示还包括：

将所述整数表示变换为与指针表示不同的数据结构。

13.如权利要求12所述的方法，其特征在于，所述不同的数据结构的至少一个位表示一个标记。

14.一种用于存储器管理的保守垃圾收集的系统，包括：

至少一个处理器；以及

至少一个存储器，包括：

包括一个或多个根对象的调用栈，每个根对象包括整数表示和指针表示；

与所述调用栈相关联的对象图，所述对象图包括包含被标记的整数表示和未经标记的指针表示的一个或多个根对象，被标记的整数表示可与未经标记的指针表示区分开来；以及

可在所述至少一个处理器上执行的组件，包括：

标记组件，标记与调用栈相关联的对象图，所述被标记的对象图包括表示被标记的整数表示和未经标记的指针表示的调用栈的根对象；

堆组件，在所述调用栈的执行期间生成对象的堆，所述堆包括来自所述调用栈的经执行部分的根对象；

跟踪组件，执行对所述调用栈的未经执行的部分中的未经执行的根对象到所述堆中的对应的存储器位置的第一跟踪，并执行对与由所述对象图所标识的未经标记的指针表示相关联的存储器位置的后续跟踪，同时跳过对与经标记的整数表示相关联的存储器位置的所述后续跟踪；以及

回收组件，清除所述堆中分配给所述调用栈不可到达的对象的存储器。

15.如权利要求14所述的系统，其特征在于，所述组件还包括：

监视组件，在所述调用栈的执行期间监视所述堆的大小；

其中，当所述堆的大小超出阈值时，激活所述回收组件。

16.如权利要求14所述的系统，其特征在于，所述回收组件将所述调用栈的未经执行部分可到达的对象的存储保存在所述堆的原始存储器位置中。

17.如权利要求14所述的系统，其特征在于，所述回收组件根据所述堆内的固定的一组边界来维护所述堆中的对象的分隔。

18.如权利要求14所述的系统，其特征在于，所述调用栈与一脚本的编译相关联。

19.如权利要求14所述的系统，其特征在于，所述调用栈与一对象模型的修改相关联。

20.如权利要求14所述的系统，其特征在于，通过将所述整数表示变换成与指针表示不同的数据结构，所述标记组件对整数表示进行标记。

# 用于存储器管理的保守垃圾收集和经标记的整数

[0001] 本申请是申请日为2012.4.6,申请号为201280020077.4(国际申请号为PCT/US2012/032639),名为“用于存储器管理的保守垃圾收集和经标记的整数”申请的分案申请。

## 背景技术

[0002] 根据与一些常规系统有关的背景技术,应当注意的是,计算设备具有传统上已存储的信息和相关联的应用。为了这些目的,实现有效的存储器管理方案对于实现增加的计算性能而言是合乎需要的。自动存储器管理方案中的开发与手动存储器管理方案相比是特别合乎需要的。垃圾收集器算法例如是自动存储器管理方案,其尝试回收不再被特定程序使用的对象所占用的存储器。

[0003] 跟踪式垃圾收集器是最常见类型的垃圾收集器。跟踪式垃圾收集器首先确定哪些对象是可到达的(或者,可能是可到达的),并然后丢弃所有剩余的对象。可到达对象可以被定义成如下的对象:针对所述对象,在程序环境中存在可直接地或通过来自其他可到达对象的引用而通向其的某个变量。更确切而言,对象一般可以以两种方式来到达。首先,假设被称为根的一组区别(distinguished)的对象的是可到达的。通常,这些对象包括从调用栈内的任何地方被引用的对象(也就是说,当前正被调用的函数中的所有局部变量和参数)以及任何全局变量。其次,从可到达对象被引用的任何对象本身被认为是可到达的。

[0004] 在具有引用类型和未装箱值类型两者的语言中,常规垃圾收集器的一个难题是,垃圾收集器区分调用栈上的哪些变量是常规值(例如整数),并且哪些是引用(例如指针)。这样的区分将在这样的系统中有益于区分目的,因为整数和引用在其他情况下在表示在存储器中时看上去是相似的。就此而言,垃圾收集器将能够使用该区分来判定是否将特定元素视为引用并跟随该引用,或者其是否仅仅是原语值。

[0005] 当今存储器管理方案的上述缺点仅旨在提供常规系统的一些问题的概览,并且不旨在是穷尽性的。在仔细阅读了以下详细描述后,现有技术的其他问题和各非限制性性实施例的对应好处可变得显而易见。

[0006] 概述

[0007] 此处提供了简化的概述以帮助能够对以下更详细的描述和附图中的示例性、非限制性实施例的各方面有基本或大体的理解。然而,本概述并不旨在是详尽的或穷尽的。相反,本概述的唯一目的在于,以简化的形式提出与一些示例性、非限制性实施例相关的一些概念,作为以下各实施例的更详细的描述的序言。

[0008] 根据一个或多个实施例及其相应公开,结合用于存储器管理的保守垃圾收集描述了各个非限制性方面。在这样的一个方面,提供了用于在保守垃圾收集算法内标记整数的方法。该方法可以包括:标识出包含在调用栈中的可包括整数表示和指针表示的根对象。该实施例还可以包括标记整数表示使得经标记的整数表示能够同未经标记的指针表示相区分。该方法还可以包括:跟踪根对象到相应的一组存储器位置。对于该特定实施例,对未经标记的指针表示执行后续跟踪,而对经标记的整数表示跳过后续跟踪。然后,该方法还可以

包括:释放分配给调用栈不可到达的一组对象的存储器。

[0009] 在另一方面,公开了一种被配置为用经标记的整数实现保守垃圾收集算法的存储器管理设备。在这一实施例中,存储器管理设备包括被配置为执行存储在存储器中的计算机可执行组件的处理器。计算机可执行组件包括扫描组件、标记组件、跟踪组件和回收组件。扫描组件被配置为标识出包含在调用栈中的包括整数表示和指针表示的根对象。然后,标记组件被配置为标记整数表示使得经标记的整数表示能够同未经标记的指针表示区分开,而跟踪组件被配置为跟踪根对象到相应的一组存储器位置。对于该实施例,跟踪组件被配置为对未经标记的指针表示执行后续跟踪,其中对经标记整数表示跳过后续跟踪。然后,回收组件被配置为释放分配给调用栈不可到达的一组对象的存储器。

[0010] 在又一方面,公开了用于在保守垃圾收集算法内标记整数的计算机可读存储介质。在这一实施例中,计算机可读存储介质包括用于使得至少一个处理器执行各个动作的计算机可读指令。例如,这样的动作包括:标记与调用栈相关联的对象图并且在调用栈的执行期间生成对象的堆。在此,经标记的对象图中所包括的根对象包括经标记的整数表示和未经标记的指针表示,而对象的堆包括调用栈的经执行部分中所包括的对象。该实施例还包括:跟踪包括在调用栈的未经执行部分中的未经执行的对象到堆上的相应存储器位置。对于该特定实施例,后续跟踪对未经标记的指针表示执行,并且对经标记的指针表示跳过。与堆上不可到达的对象相对应的存储器位置然后被清除,其中不可到达的对象是调用栈的未经执行部分所不能到达的。

[0011] 以下更详细地描述其他实施例和各非限制性性示例、场景和实现。

## 附图说明

[0012] 参考附图进一步描述各非限制性实施例,在附图中:

[0013] 图1示出了根据一实施例的促进实现保守垃圾收集算法的示例性系统;

[0014] 图2是示出了根据一实施例的对象的示例性堆的示图;

[0015] 图3是示出了根据一实施例的示例性对象图的示图;

[0016] 图4是示出了根据一实施例的对存储器的示例性释放的示图;

[0017] 图5是示出了根据一实施例的示例性存储器管理设备的框图;

[0018] 图6是示出根据一实施例的用于实现保守垃圾收集算法的示例性非限制性实施例的流程图;

[0019] 图7是示出根据一实施例的示例性资源管理单元的框图;

[0020] 图8是示出根据一实施例的用于标记整数的示例性非限制性实施例的流程图。

[0021] 图9是表示其中可实现在此处所述的各个实施例的示例性、非限制性联网环境的框图;以及

[0022] 图10是表示其中可实现此处所述的各个实施例的一个或多个方面的示例性、非限制性计算系统或操作环境的框图。

## 具体实施方式

[0023] 概览

[0024] 如背景技术中所述,实现区分调用栈上的整数和指针的垃圾收集算法是合乎需要

的。在各个实施例中，就本机代码兼容性重新设计了存储器管理。在一方面，脚本对象是得到较少管理的对象，而非只是存储器的本机片段，使得各对象间的引用计数被消除。此外，实现了保守垃圾收集算法，其中没有假设是指针的所有事物都是已知的。以公共语言运行时(CLR)垃圾收集器为例，栈被有力地键入。然而，使用本机代码(例如，C代码和脚本代码)，人们不知道栈上有什么。在这一方面，取代利用引用计数模型，所构思的是直接与对象进行交互。

[0025] 在此公开的各个实施例所针对的是，在保守垃圾收集算法内标记整数。为此目的，应当注意，在此所公开的各方面促进对生成伪指针的可能性更小的文档对象模型执行脚本(例如java脚本)。此外，在此公开的各方面减少了伪指针的概率，因为未经标记的指针和经标记的整数是容易区分的。

[0026] 经标记的指针和保守垃圾收集

[0027] 当web浏览体验从具有最少交互性的对信息的单调呈现发展成在客户机侧具有许多交互性的更丰富应用或小程序体验时，若干问题已产生。更一般的，web浏览体验已发展成信息显示以及与显示器上的对象的更丰富的交互性的混合。伴随这种发展的特定挑战基于使原来的文档对象模型(DOM)适应于流畅地处理脚本代码(诸如，java脚本对象)的体验，原来的文档对象模型最初主要是基于客户机上的本机代码为信息的单调呈现而设计的。

[0028] 为了促进更期望的用户体验，提高速度是特别重要的。例如，使用过去的飞出菜单，web体验基于与服务器的通信对延迟进行闪烁。然而，脚本允许小程序修改飞出菜单上的DOM，而无需返回到服务器。由于人们想要在不返回到服务器的情况下对飞出菜单进行更多动作，使脚本代码快速执行已成为挑战。

[0029] 由于用户体验会受到高效地脚本化DOM的较大影响，因此将DOM改变成尽可能快以最大化交互响应是合乎需要的。在过去，由于对包括使任何对象可脚本化的一组接口(例如，iDispatch、iActiveScript等)的对象链接和嵌入(OLE)自动化的使用，脚本引擎和DOM的本机类之间的通信是低劣的。然而，这些方法是慢的，并因此期望改进。因此，在此公开的方面所针对的是通过在保守垃圾收集算法内标记整数来提高脚本执行速度。

[0030] 图1示出了根据一实施例的便于实现保守垃圾收集算法的示例性系统。如图所示，系统100可包括存储器管理单元110，该单元通信地耦合到存储器120。在一方面，存储器管理单元110被配置为实现保守垃圾收集算法来管理存储器120中的存储器空间。此外，存储器管理单元110被配置为生成与脚本112的执行相关联的经标记的对象图114。例如，脚本112可以是对DOM执行的java脚本，其中java脚本包括要求分配存储器120中的存储器空间的各个对象。在特定的实施例中，为了促进对存储器120内整数和指针的区分，对应于整数值的脚本对象被存储器管理单元110标记并且被包括在经标记的对象图114中。

[0031] 在一方面，分配给脚本对象的存储器包括将这些对象存储在堆上。接着参考图2，提供了根据一实施例的对象的示例性堆的框图。如图所示，堆200可包括可用存储器210以及所分配的与各个对象220、230、240、250、260和270相对应的存储器。对于这个特定示例，对象240和270对应于指针值，而对象220、230、250和260对应于整数值。也就是说，对象240是引用了对象230所表示的整数值的指针值，而对象270是引用了对象260所表示的整数值的指针值。

[0032] 就此而言，对应于整数值的脚本对象可以被标记，使得它们可容易地同指针值相

区分。接着参考图3，提供了示出示例性对象图的示图，该示例性对象图促进堆上的经标记/未经标记的对象的映射。如所示那样，对象图300将包括在未经执行的调用栈部分310中的对象映射到堆320。为这些和相关目的，应注意，堆320和可用存储器330一般分别与堆200、可用存储器210类似。

[0033] 在一方面，利用对象图300来确定堆320上的哪些对象是未经执行的调用栈部分310可到达的。也就是说，跟踪包括在未经执行的调用栈部分310中的根对象到堆320中的相应存储器位置，其中与整数值相对应的根对象被标记。于是，所构思的是，仅仅对可到达的未经标记的指针值执行对根对象的后续跟踪，以标识出被那些指针值引用的可到达对象。相反，对可到达的经标记的整数值，跳过后续跟踪。对于该特定示例，由于可到达对象390是对应于指针值的根对象，因此其未被标记。因此，对可到达对象390执行后续跟踪，其标识出与整数值相对应的可到达对象380。然而，对可到达对象340和370跳过后续跟踪，因为这些根对象是对应于经标记的整数值的根对象。

[0034] 应当注意，还可以利用对象图300来标识未经执行的调用栈部分310不可到达的对象。在这个特定示例中，不可到达对象350和360被认为是不可到达的，因为它们不对应于未经执行的调用栈部分310中的根对象，而且它们也没有被可到达的指针对象引用。

[0035] 在标识出不可到达的对象以后，可以清除被分配给这样的对象的存储器。接着参考图4，提供了根据一实施例的对存储器的示例性释放的示图。如所示那样，对象图400包括堆420，其一般与堆200和320类似，其中堆420描绘了对之前分配给不可到达对象350和360的存储器的释放。也就是说，除了可用存储器430以外，堆420现在还包括已释放的存储器450和460。然而，在一方面，将可到达对象440、470、480和490保存在其原始存储位置中(即，堆420没有被损坏(collapse))。

[0036] 接着参考图5，一框图示出了被配置为根据各个方面实现保守垃圾收集算法的示例性存储器管理单元。如图所示，存储器管理单元500可包括处理器组件510、存储器组件520、扫描组件530、标记组件540、跟踪组件540和回收组件560。

[0037] 在一个方面，处理器组件510被配置为执行与执行多个功能中的任何功能相关的计算机可读指令。处理器组件510可以是单个处理器或多个处理器，其专用于分析要从存储器管理单元500传递的信息和/或生成可由存储器组件520、扫描组件530、标记组件540、跟踪组件550和/或回收组件560利用的信息。附加地或可替代地，处理器组件510可被配置为控制存储器管理单元500的一个或多个组件。

[0038] 在另一方面中，存储器组件520被耦合至处理器组件510并被配置为存储由处理器组件510执行的计算机可读指令。存储器组件520还可被配置为存储包括由扫描组件530、标记组件540、跟踪组件550、和/或回收组件560中的任何组件生成的数据在内的多种其它类型的数据中的任何数据。存储器组件520可以被配置为多个不同的配置，包括作为随机存取存储器、带后备电源的存储器、硬盘、磁带等等。还可以在存储器组件520上实现各种特征，如压缩和自动备份，如使用独立驱动器冗余阵列配置。

[0039] 如所示那样，存储器管理单元500还可包括扫描组件530。在这样的实施例中，扫描组件530被配置为标识出调用栈中包含的包括整数表示和指针表示的根对象。在此所构思的是，将这样的调用栈与对特定对象模型执行的脚本相关联。例如，调用栈可以与在运行时期间促进文档对象模型的修改的java脚本的编译相关联。

[0040] 如所示那样，存储器管理单元500还可以包括标记组件540，该标记组件540被配置为标记包括在调用栈中的整数表示。在此，应当注意，整数的标记被执行使得经标记的整数表示能够容易地与未经标记的指针表示相区分。为此目的，还应注意，标记组件540可被配置为以多种方式中的任何方式标记整数。例如，标记组件540可以被配置为将整数表示变换成不同的数据结构。在特定的实施例中，这样的变换可以包括让标记组件540被配置为将所述不同的数据结构中的至少一位专用于标记。

[0041] 在一方面，存储器管理设备500还包括跟踪组件550。在这样的实施例中，跟踪组件550被配置为跟踪由扫描组件530标识出的根对象到相应一组存储器位置。在最初跟踪根对象以后，所构思的是，执行对相关对象的后续跟踪。然而在特定的实施例中，后续跟踪仅仅对未经标记的指针表示执行，并且对经标记的整数表示跳过。

[0042] 在另一方面，存储器管理设备500还包括回收组件560，该回收组件560被配置为释放被分配给调用栈不可到达的一组对象的存储器。在特定的实施例中，回收组件500可以被配置为将调用栈所调用的对象放置到堆上。在这样的实施例中，回收组件560然后还可以被配置为基于堆的大小是否超过了阈值来发起垃圾收集算法。在另一实施例中，回收组件560可被配置为将各可到达对象分别保存在堆的各原始存储位置中。例如，回收组件560可被配置为根据堆内的固定的一组边界来维护各堆对象的分隔。

[0043] 图6是示出根据一实施例的用于实现保守垃圾收集算法的示例性非限制性实施例的流程图。在600，标识出包含在调用栈中的包括整数表示和指针表示的根对象。接着在610，标记整数表示标记使得经标记的整数表示可以同未经标记的指针表示相区分。然后在620，跟踪根对象到相应的一组存储器位置，使得对指针表示执行后续跟踪并对整数表示跳过后续跟踪。然后在630，释放分配给调用栈不可到达的对象的存储器。

[0044] 接着参考图7，示出了被配置为根据各个方面标记整数的示例性资源管理单元的框图。如所示那样，资源管理单元700可包括处理器组件710、存储器组件720、标记组件730、堆组件740、跟踪组件750、释放组件760、监视组件770和制图组件780。

[0045] 类似于存储器管理单元500中的处理器组件510，处理器组件710被配置为执行与执行多个功能中的任何功能相关的计算机可读指令。处理器组件710可以是单个处理器或多个处理器，其专用于分析要从资源管理单元700传递的信息和/或生成可由存储器组件720、标记组件730、堆组件740、跟踪组件750、释放组件760、监视组件770和/或制图组件780使用的信息。附加地或可替代地，处理器组件710可被配置为控制资源管理单元700的一个或多个组件。

[0046] 在另一方面中，存储器组件720被耦合到处理器组件710并被配置为存储由处理器组件710执行的计算机可读指令。存储器组件720还可被配置为存储多种其他类型的数据中的任何数据，包括由标记组件730、堆组件740、跟踪组件750、释放组件760、监视组件770和/或制图(graphing)组件780中的任何组件生成的数据。此处，应注意，存储器组件720与存储器管理单元500中的存储器组件520类似。因此，可以理解，存储器组件520的任何前述特征/配置也适用于存储器组件720。

[0047] 如所示那样，资源管理单元700还可包括标记组件730。在这样的实施例中，标记组件730被配置为标记与调用栈相关联的对象图。在此，应当注意，包括在经标记的对象图中的根对象可以包括经标记的整数表示和未经标记的指针表示。对于一些实施例，还应当注

意，资源管理单元700还可以包括制图组件780，该制图组件780被配置为产生由标记组件730标记的对象图。

[0048] 在一方面，资源管理单元700被配置为管理堆中包括的对象。为了促进这样的管理，资源管理单元700可以包括堆组件740，该堆组件740被配置为在调用栈执行期间生成对象的堆。对于该特定实施例，所构思的是，该对象堆可以包括调用栈的经执行的一部分中所包括的对象。

[0049] 在另一方面，资源管理单元700还可包括跟踪组件750。在这样的实施例中，跟踪组件750被配置为跟踪调用栈的未经执行部分中的未经执行对象到堆上的相应存储器位置。在此，类似于存储器管理设备500中的跟踪组件550，所构思的是，跟踪组件750可以被配置为在最初跟踪调用栈的根对象以后执行相关对象的后续跟踪。此外所构思的是，这样的后续跟踪仅仅对未经标记的指针表示执行，并且对经标记的整数表示跳过。

[0050] 如所示那样，资源管理单元700还可包括释放组件760。在这样的实施例中，释放组件760被配置为清除堆上的与如下对象相对应的存储器位置：所述对象被视为调用栈的未经执行的部分所不能到达的。在一方面，释放组件760还可被配置为将调用栈的未经执行部分可到达的各对象的存储分别保存在堆内的它们的原始存储器位置中。在另一方面，资源管理单元700还可以包括监视组件770，该监视组件770被配置为在调用栈的执行期间监视堆的大小。对于该特定实施例，释放组件760然后可以被配置为基于堆大小超过特定阈值来触发分配给不可到达对象的存储器的清除。

[0051] 图8是示出根据一实施例的用于标记整数的示例性非限制性实施例的流程图。在800，标记与调用栈相关联的对象图标记使得经标记的对象图中所包括的根对象包括经标记的整数表示和未经标记的指针表示。在810，然后在调用栈的执行期间生成对象的堆，其包括调用栈的经执行部分中所包括的对象。接着，在820，跟踪包括在调用栈的未经执行部分中的未经执行的对象到堆上的相应存储器位置。对于该特定实施例，仅仅对指针表示执行后续跟踪，而不对整数表示执行后续跟踪。然后，在830，清除与调用栈的未经执行部分不可到达的堆对象相对应的存储器位置。

[0052] 示例性联网以及分布式环境

[0053] 本领域普通技术人员能够理解，此处所描述的用于在保守垃圾收集算法内标记整数的各个实施例可以结合任何计算机或其他客户机或服务器设备来实现，该任何计算机或其它客户机或服务器设备可作为计算机网络的一部分来部署或者被部署在分布式计算环境中，并且可以连接到任何种类的数据存储。在这一点上，此处描述的各实施例可在具有任何数量的存储器或存储单元的、并且任何数量的应用和进程跨任何数量的存储单元发生的任何计算机系统或环境中实现。这包括但不限于具有部署在具有远程或本地存储的网络环境或分布式计算环境中的服务器计算机和客户机计算机的环境。

[0054] 附图9提供了示例性联网或分布式计算环境的非限制性性示意图。该分布式计算环境包括计算对象或设备910、912等以及计算对象或设备920、922、924、926、928等，这些计算对象或设备可包括如由应用930、932、934、936、938表示的程序、方法、数据存储、可编程逻辑等。可以理解，计算对象或设备910、912等以及计算对象或设备920、922、924、926、928等可包括不同的设备，诸如PDA、音频/视频设备、移动电话、MP3播放器、膝上型计算机等。

[0055] 每个计算对象或设备910、912等以及计算对象或设备920、922、924、926、928等可

经由通信网络940或直接或间接地与一个或多个其他计算对象或设备910、912等以及计算对象或设备920、922、924、926、928等通信。即使在图9中被示为单个元件，但网络940也可包括向图9的系统提供服务的其他计算对象或计算设备，和/或可表示未示出的多个互连网络。每个计算对象或设备910、912等或920、922、924、926、928等还可包含诸如应用930、932、934、936、938之类的应用，该应用可利用API或适用于与根据各实施例提供的来自任何平台的信息作为服务的基础结构进行通信或实现这种基础结构的其他对象、软件、固件和/或硬件。

[0056] 存在支持分布式计算环境的各种系统、组件和网络配置。例如，计算系统可由有线或无线系统、本地网络或广泛分布的网络连接在一起。当前，许多网络被耦合至因特网，后者为广泛分布的计算提供了基础结构并包含许多不同的网络，但任何网络基础结构可用于变得与如各实施例中所描述的技术相关联的示例性通信。

[0057] 由此，可使用诸如客户机/服务器、对等、或混合体系结构之类的网络拓扑结构和网络基础结构的主机。在客户机/服务器体系结构中，尤其在联网系统中，客户机通常是访问另一计算机(例如，服务器)所提供的共享网络资源的计算机。在附图9的图示中，作为非限制性性示例，计算对象或设备920、922、924、926、928等可被认为是客户机和计算对象，或者设备910、912等可被认为是服务器，其中计算对象或设备910、912等提供数据服务，诸如从计算对象或设备920、922、924、926、928等接收数据、存储数据、处理数据、向计算对象或设备920、922、924、926、928发送数据等，但任何计算机都可取决于环境而被认为是客户机、服务器或两者。这些计算设备中的任一个都可以处理数据，或请求可隐含此处一个或多个实施例所描述的存储器管理和相关技术的服务或任务。

[0058] 服务器通常是可通过诸如因特网或无线网络基础结构之类的远程网络或本地网络访问的远程计算机系统。客户机进程可在第一计算机系统中活动，而服务器进程可在第二计算机系统中活动，它们通过通信介质相互通信，由此提供分布式功能并允许多个客户机利用服务器的信息收集能力。按照用户剖析来利用的任何软件对象可以独立地提供或跨多个计算设备或对象分布。

[0059] 例如，在其中通信网络/总线940是因特网的网络环境中，计算对象或设备910、912等可以是计算对象或设备920、922、924、926、928等经由诸如HTTP等多种已知协议中的任一种与其通信的web服务器。如所提及的，计算对象或设备910、912等还可用作计算对象或设备920、922、924、926、928等，或者相反，这可以是分布式计算环境的特性。

[0060] 示例性计算设备

[0061] 如所提到的，在此描述的各个实施例适用于其中实现用于在保守垃圾收集算法内标记整数的基础结构是合乎需要的任何设备。因此，应当理解，结合此处描述的各实施例来使用，即在设备可以结合用所标记的整数实现保守垃圾收集算法来提供某个功能的任何地方构想了各种种类的手持式、便携式和其它计算设备和计算对象。因此，在下面的图10中描述的以下通用远程计算机仅是一个示例，且所公开的主题的各实施例可以用具有网络/总线互操作性和交互的任何客户机来实现。

[0062] 尽管并不是必需的，但各实施例的任意一个可以部分地经由操作系统来实现，以供设备或对象的服务开发者使用，和/或被包括在结合可操作组件来操作的应用软件中。软件可在诸如客户机工作站、服务器或其他设备之类的一个或多个计算机所执行的诸如程序

模块之类的计算机可执行指令的通用上下文中描述。本领域的技术人员可以理解，网络交互可以用各种计算机系统配置和协议来实施。

[0063] 因此，图10示出了其中可实现一个或多个实施例的合适的计算系统环境1000的一个示例，但是上面已经弄清楚，计算系统环境1000仅为合适的计算环境的一个示例，而不旨在对各实施例中的任意一个的使用范围或功能提出任何限制。不应将计算环境1000解释为对示例性操作环境1000中示出的任一组件或其组合有任何依赖性或要求。

[0064] 参考图10，用于实现此处的一个或多个实施例的示例性远程设备可以包括手持式计算机1010形式的通用计算设备。手持式计算机1010的组件可以包括但不限于：处理单元1020、系统存储器1030和将包括系统存储器在内的各种系统组件耦合至处理单元1020的系统总线1021。

[0065] 计算机1010通常包括各种计算机可读介质，并且可以是可由计算机1010访问的任何可用介质。系统存储器1030可包括诸如只读存储器(ROM)和/或随机存取存储器(RAM)之类的易失性和/或非易失性存储器形式的计算机存储介质。作为示例而非限制性，存储器1030还可以包括操作系统、应用程序、其他程序模块、和程序数据。软件也可以包含在数字多功能盘(DVD)、紧致盘(CD)或其他可移动存储上。

[0066] 用户可以通过输入设备1040向计算机1010输入命令和信息。监视器或其他类型的显示设备也经由诸如输出接口1050之类的接口连接到系统总线1021。除监视器之外，计算机还可以包括其他外围输出设备，如扬声器和打印机，它们可以通过输出接口1050连接。

[0067] 计算机1010可使用到一个或多个其他远程计算机(诸如远程计算机1070)的逻辑连接在联网或分布式环境中操作。远程计算机1070可以是个人计算机、服务器、路由器、网络PC、对等设备或其他常见网络节点、或者任何其他远程媒体消费或传输设备，并且可包括以上关于计算机1010所述的任何或全部元件。图10所示的逻辑连接包括诸如局域网(LAN)或广域网(WAN)之类的网络1071，但也可包括其他网络/总线。这些联网环境在家庭、办公室、企业范围的计算机网络、内联网和因特网中是常见的。

[0068] 如上所述，尽管结合各种计算设备、网络和广告体系结构描述了各示例性实施例，但基本概念可被应用于其中期望管理存储器的任何网络系统和任何计算设备或系统。

[0069] 有多种实现此处描述的一个或多个实施例的方式，例如，使应用和服务能使用存储器管理的适当API、工具包、驱动程序代码、操作系统、控件、独立或可下载的软件对象等等。可以从API(或其他软件对象)的观点以及从便于根据所描述的实施例中的一个或多个提供存储器管理的软件或硬件对象来构想各实施例。此处描述的各种实现和实施例可以具有完全采用硬件、部分采用硬件并且部分采用软件、以及采用软件的方面。

[0070] 本文中所使用的词语“示例性”意味着用作示例、实例、或说明。为避免疑惑，本文所公开的主题不限于这些示例。另外，在此所述的被描述为“示例性”的任意方面或设计并不一定要被解释为相比其它方面或设计更优选或有利，也不意味着排除本领域技术人员已知的等效示例性结构和技术。此外，在详细描述或权利要求书中使用术语“包括”、“具有”、“包含”和其他类似词语的程度上，为避免疑惑，这些术语旨在以类似于术语“包括”作为开放的过渡词的方式是包含性的而不排除任何附加或其他元素。

[0071] 如所述的，此处所述的各种技术可结合硬件或软件或，在适当时，以两者的组合来实现。如此处所使用的，术语“组件”、“系统”等同样旨在指计算机相关实体，或者是硬件、硬

件和软件的组合、软件或者是执行中的软件。例如，组件可以是，但不限于是，在处理器上运行的进程、处理器、对象、可执行码、执行的线程、程序和/或计算机。作为说明，在计算机上运行的应用和计算机都可以是组件。一个或多个组件可以驻留在进程和/或执行的线程中，并且组件可以位于一个计算机内和/或分布在两个或更多计算机之间。

[0072] 如前所述的系统已经参考若干组件之间的交互来描述。可以理解，这些系统和组件可包括组件或指定的子组件、某些指定的组件或子组件和/或附加的组件，并且根据上述内容的各种置换和组合。子组件还可作为通信地耦合到其他组件的组件来实现，而不是被包括在父组件内(层次性)。另外，注意到，一个或多个组件也可以合并到提供聚合功能的单个组件中，或者也可以分成多个单独的子组件，并且，可以提供诸如管理层之类的任何一个或更多中间层，以可通信地耦合到这样的子组件，以便提供集成的功能。此处所述的任何组件也可与一个或多个此处未专门描述的但本领域技术人员一般已知的其他组件进行交互。

[0073] 鉴于以上描述的示例性系统，参考各附图的流程图将可以理解依照所公开的主题实现的方法。尽管为了说明简洁起见，按照一系列框示出和描述了方法，但是，应该理解和知道，所要求保护的主题不限于框的次序，因为一些框可以按与此处所描绘和描述的不同的次序进行和/或与其他框并发地进行。尽管经由流程图示出了非顺序或分支的流程，但可以理解，可实现达到相同或相似结果的各种其他分支、流程路径和框的次序。此外，并非全部所示的框都是实现下面所述的方法所必需的。

[0074] 虽然在某些实施例中示出了客户机侧观点，但要出于避免对存在相对应的服务器观点的疑问来理解，反之亦然。类似地，在实施一方法时，可以提供具有存储和被配置为经由一个或多个组件实施该方法的至少一个处理器的相对应的设备。

[0075] 尽管结合各附图的优选实施例描述了各实施例，但可以理解，可以使用其他类似的实施例，或可以对所描述的实施例进行修改和添加来执行相同的功能而不背离本发明。而且，此处描述的各实施例的一个或多个方面可以在多个处理芯片或设备中实现或跨多个处理芯片或设备实现，且存储可以类似地跨多个设备来实现。因此，本发明不应限于任何单个实施例，而是应该根据所附权利要求书的广度和范围来解释。

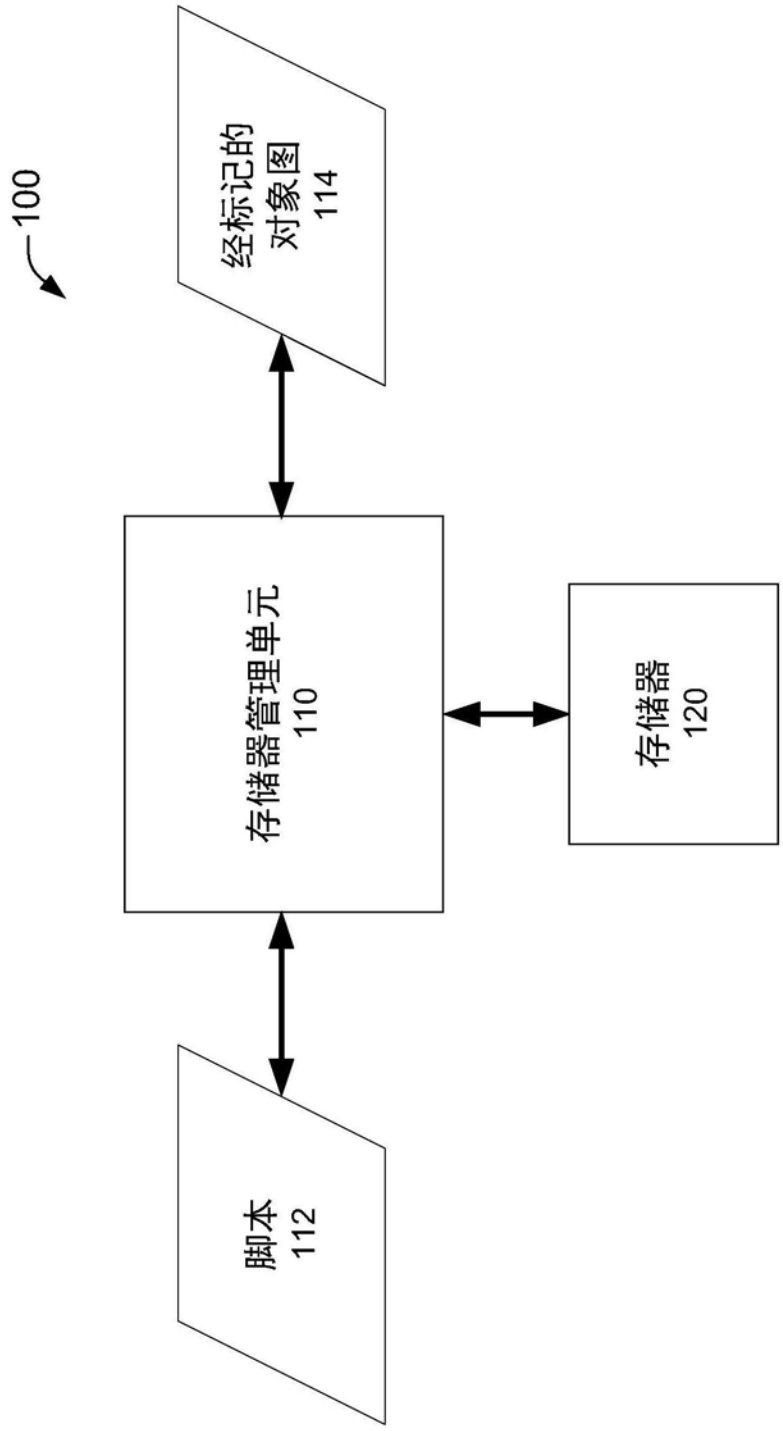



图1

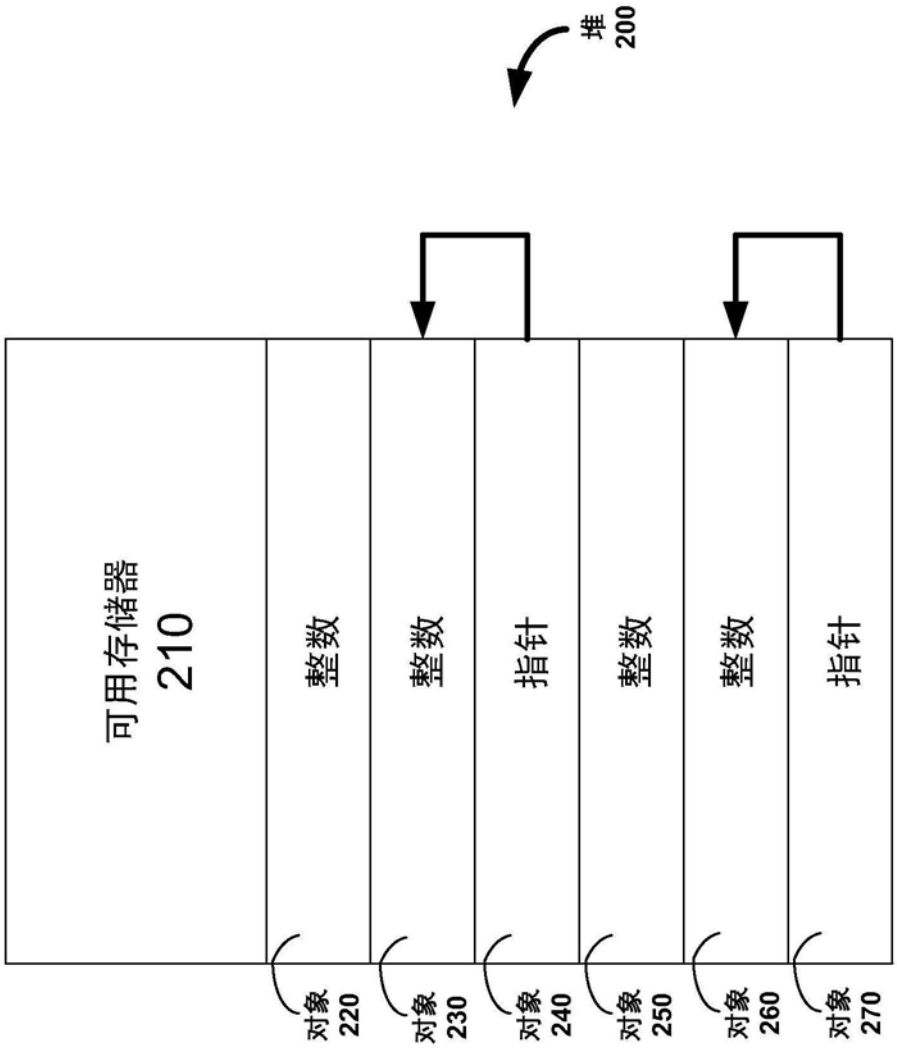

堆
200
可用存储器
210
整数
整数
指针
整数
整数
指针
对象
220
对象
230
对象
240
对象
250
对象
260
对象
270

图2

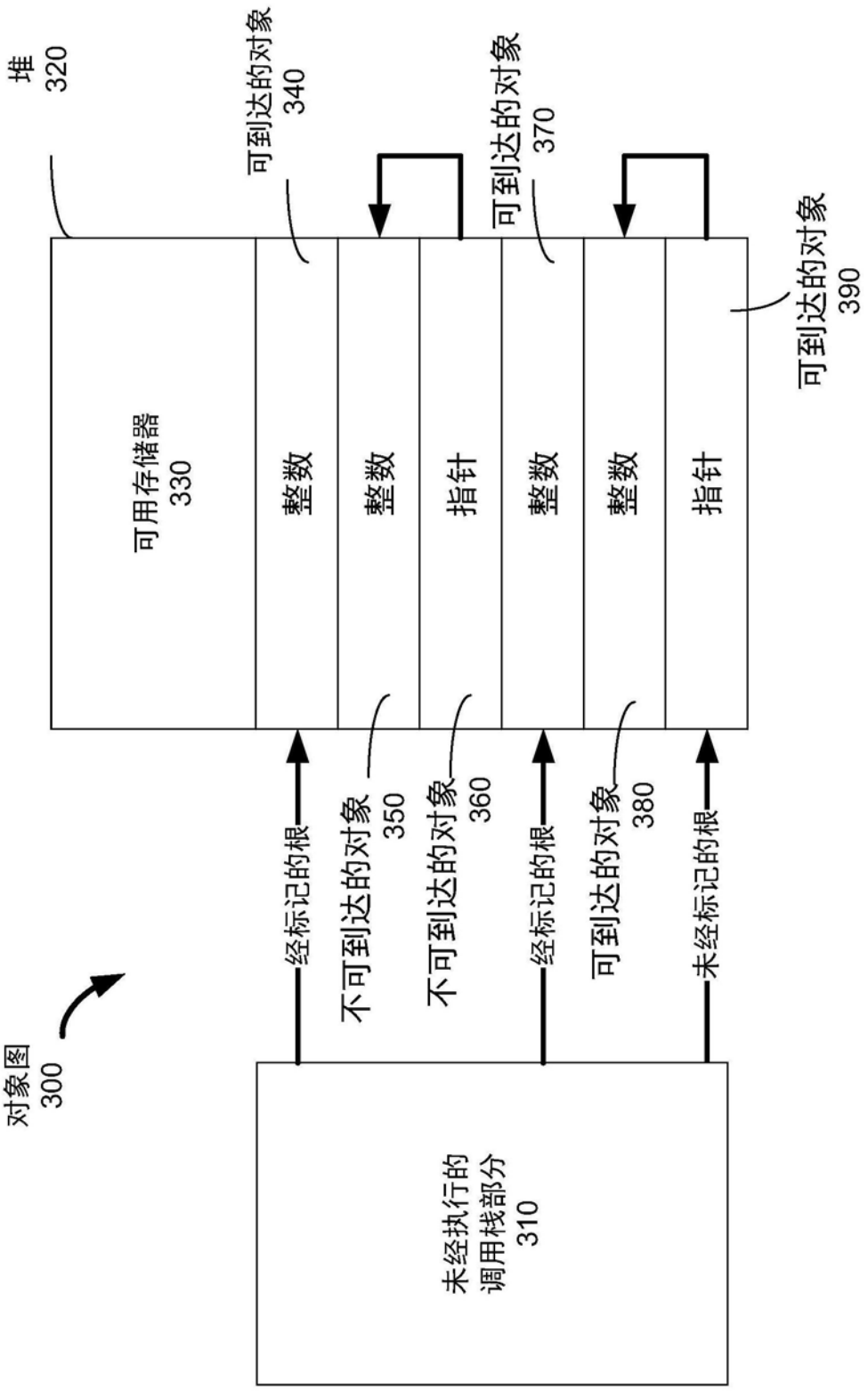

堆
320
可到达的对象
340
可到达的对象
370
可到达的对象
390
可用存储器
330
整数
整数
指针
整数
整数
指针
经标记的根
不可到达的对象
350
不可到达的对象
360
经标记的根
可到达的对象
380
未经标记的根
对象图
300
未经执行的
调用栈部分
310

图3

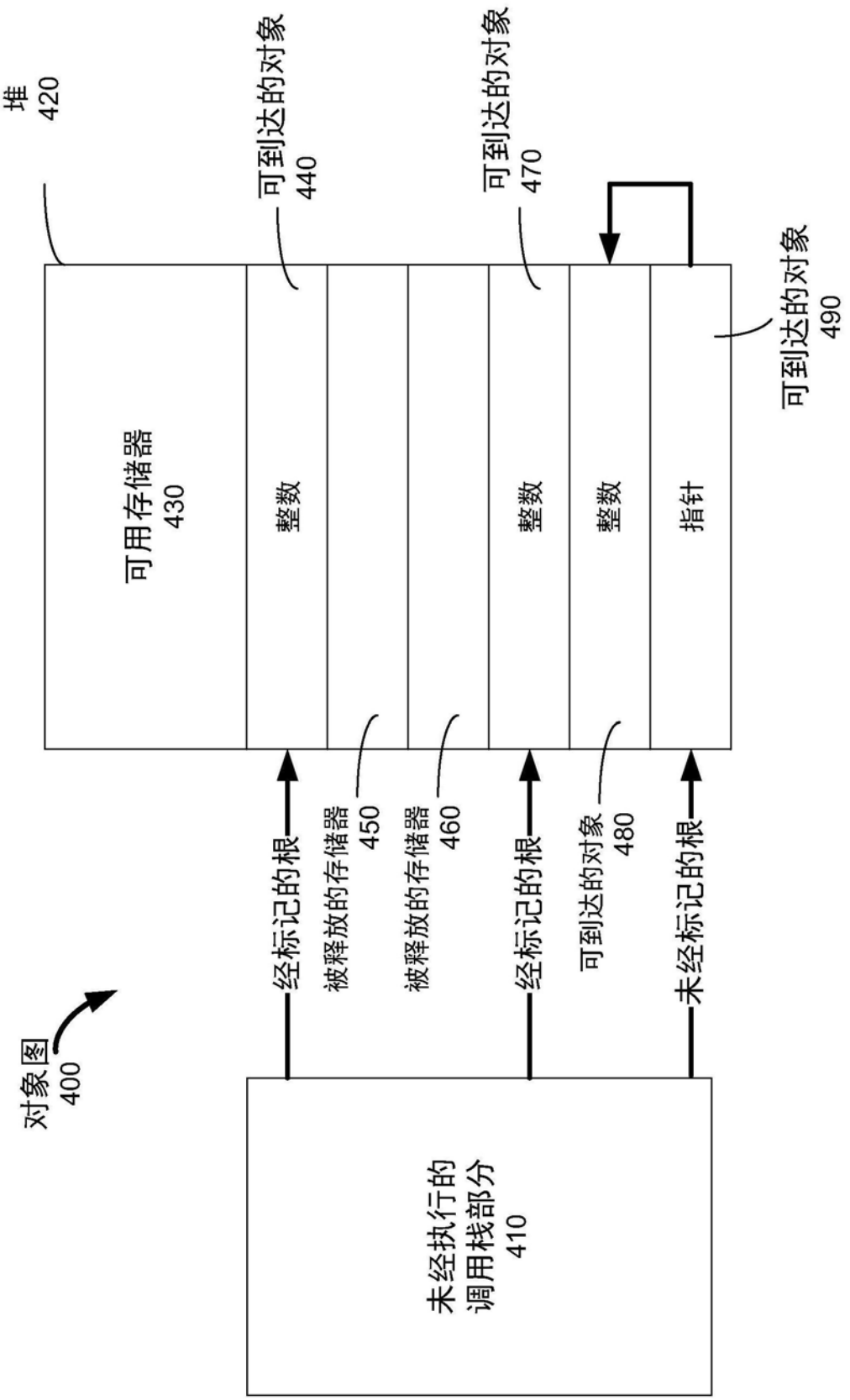

堆
420
可到达的对象
440
可到达的对象
470
可到达的对象
490
可用存储器
430
整数
整数
整数
指针
经标记的根
被释放的存储器
450
被释放的存储器
460
经标记的根
可到达的对象
480
未经标记的根
对象图
400
未经执行的
调用栈部分
410

图4


500
存储器管理设备
处理器组件
510
存储器组件
520
扫描组件
530
标记组件
540
跟踪组件
550
回收组件
560
API: 应用编程接口

图5

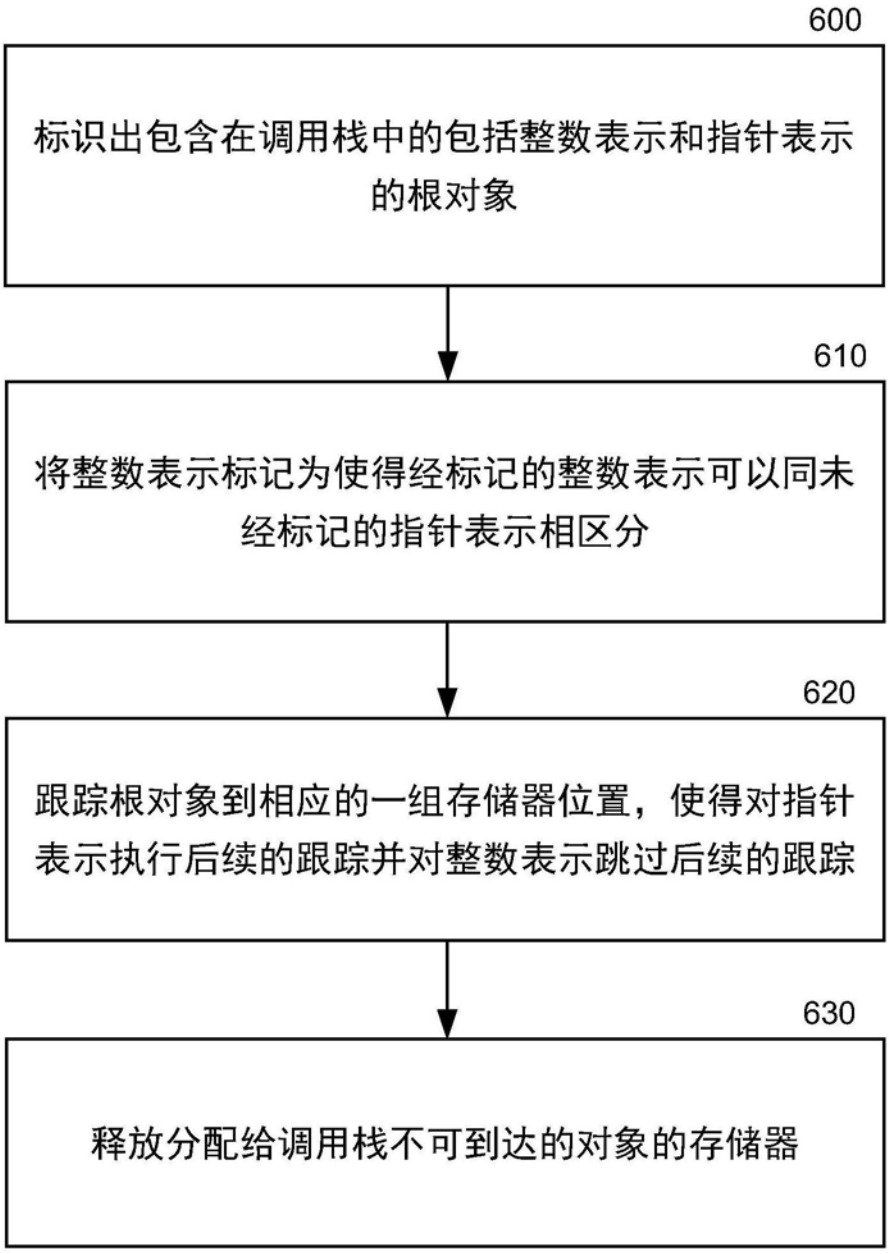

600
标识出包含在调用栈中的包括整数表示和指针表示的根对象
610
将整数表示标记为使得经标记的整数表示可以同未经标记的指针表示相区分
620
跟踪根对象到相应的一组存储器位置，使得对指针表示执行后续的跟踪并对整数表示跳过后续的跟踪
630
释放分配给调用栈不可到达的对象的存储器

图6


700
资源管理单元
处理器组件
710
存储器组件
720
标记组件
730
堆组件
740
跟踪组件
750
释放组件
760
监视组件
770
制图组件
780

图7

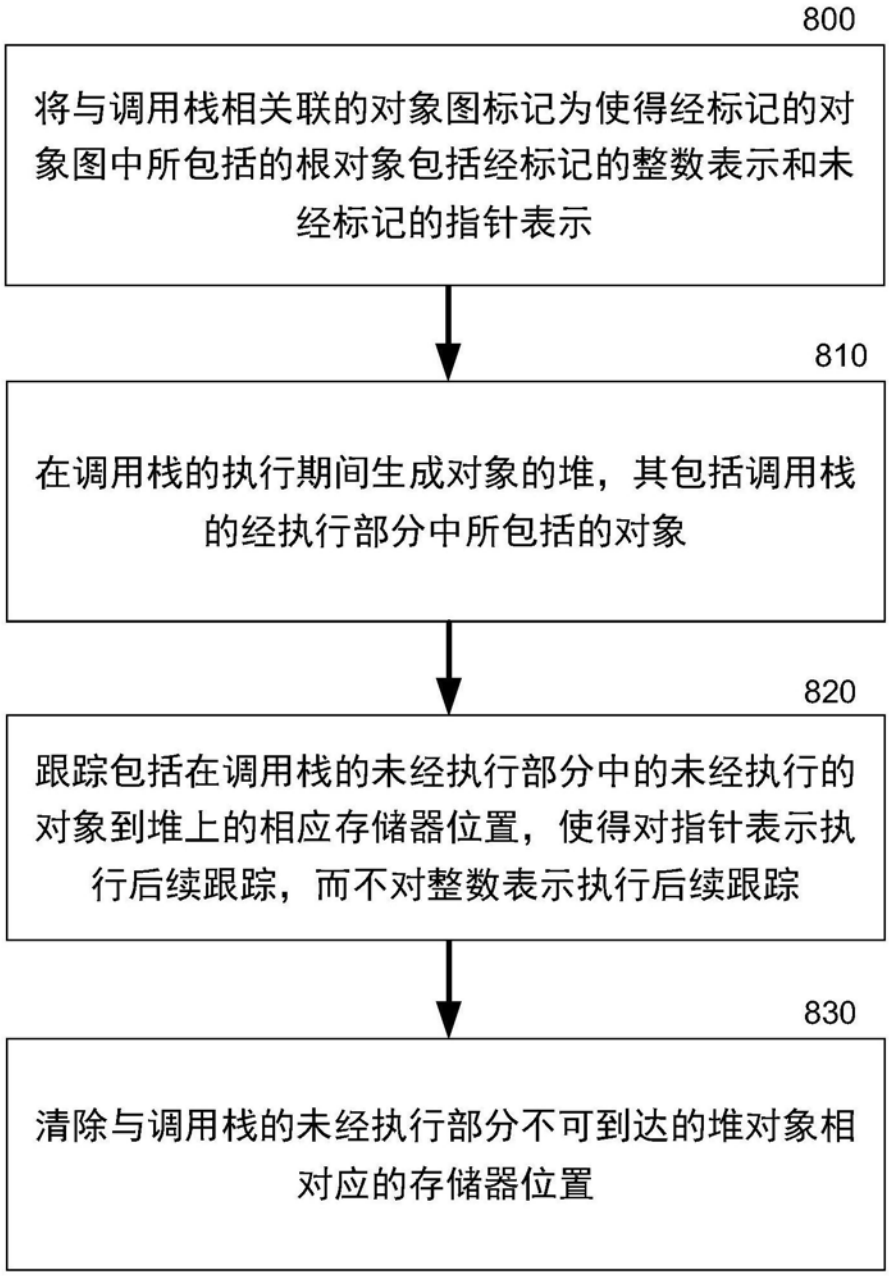

800
将与调用栈相关联的对象图标记为使得经标记的对象图中所包括的根对象包括经标记的整数表示和未经标记的指针表示
810
在调用栈的执行期间生成对象的堆，其包括调用栈的经执行部分中所包括的对象
820
跟踪包括在调用栈的未经执行部分中的未经执行的对象到堆上的相应存储器位置，使得对指针表示执行后续跟踪，而不对整数表示执行后续跟踪
830
清除与调用栈的未经执行部分不可到达的堆对象相对应的存储器位置

图8

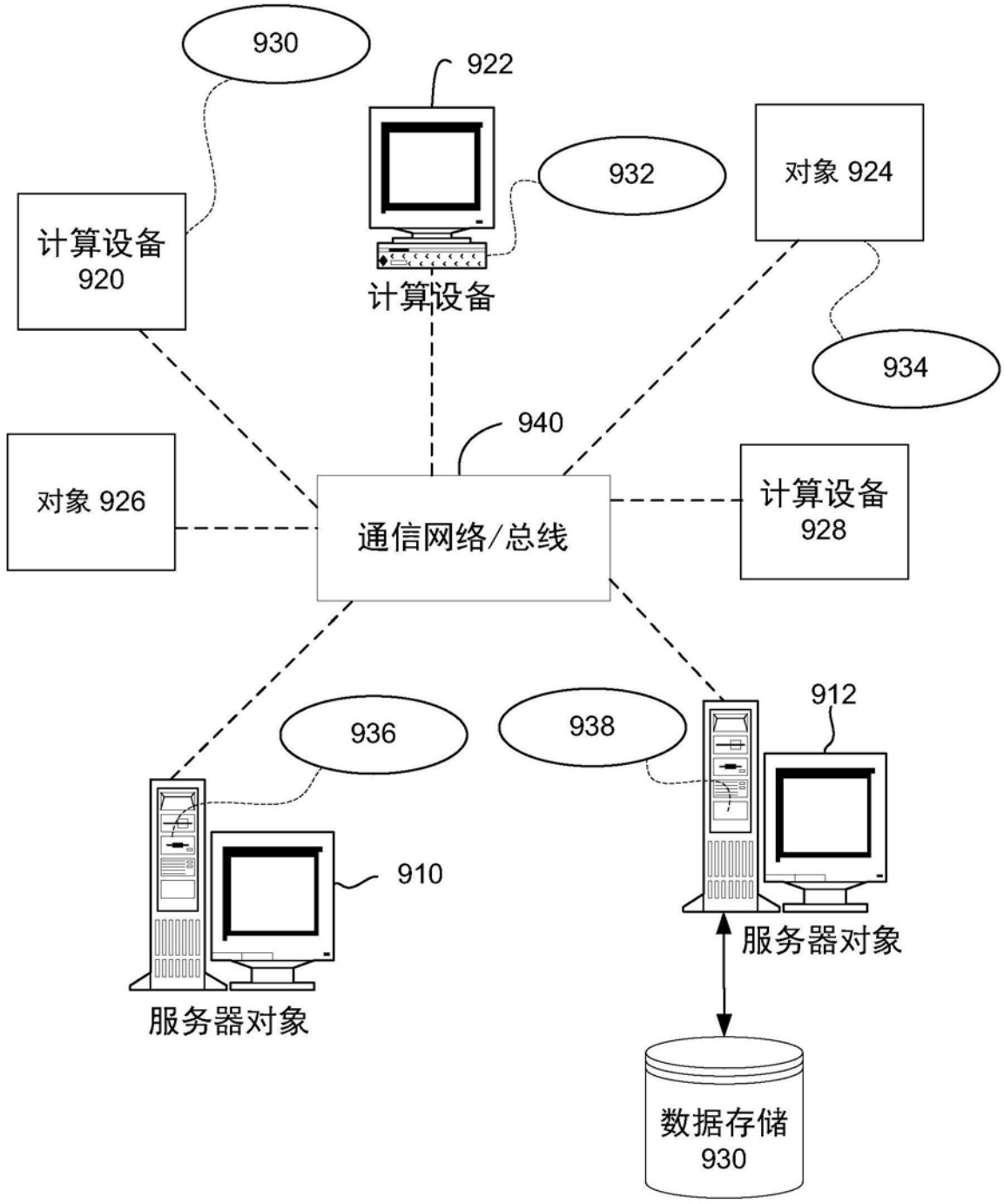

930
922
932
对象 924
计算设备
920
计算设备
934
940
对象 926
通信网络/总线
计算设备
928
912
936
938
910
服务器对象
服务器对象
数据存储
930

图9

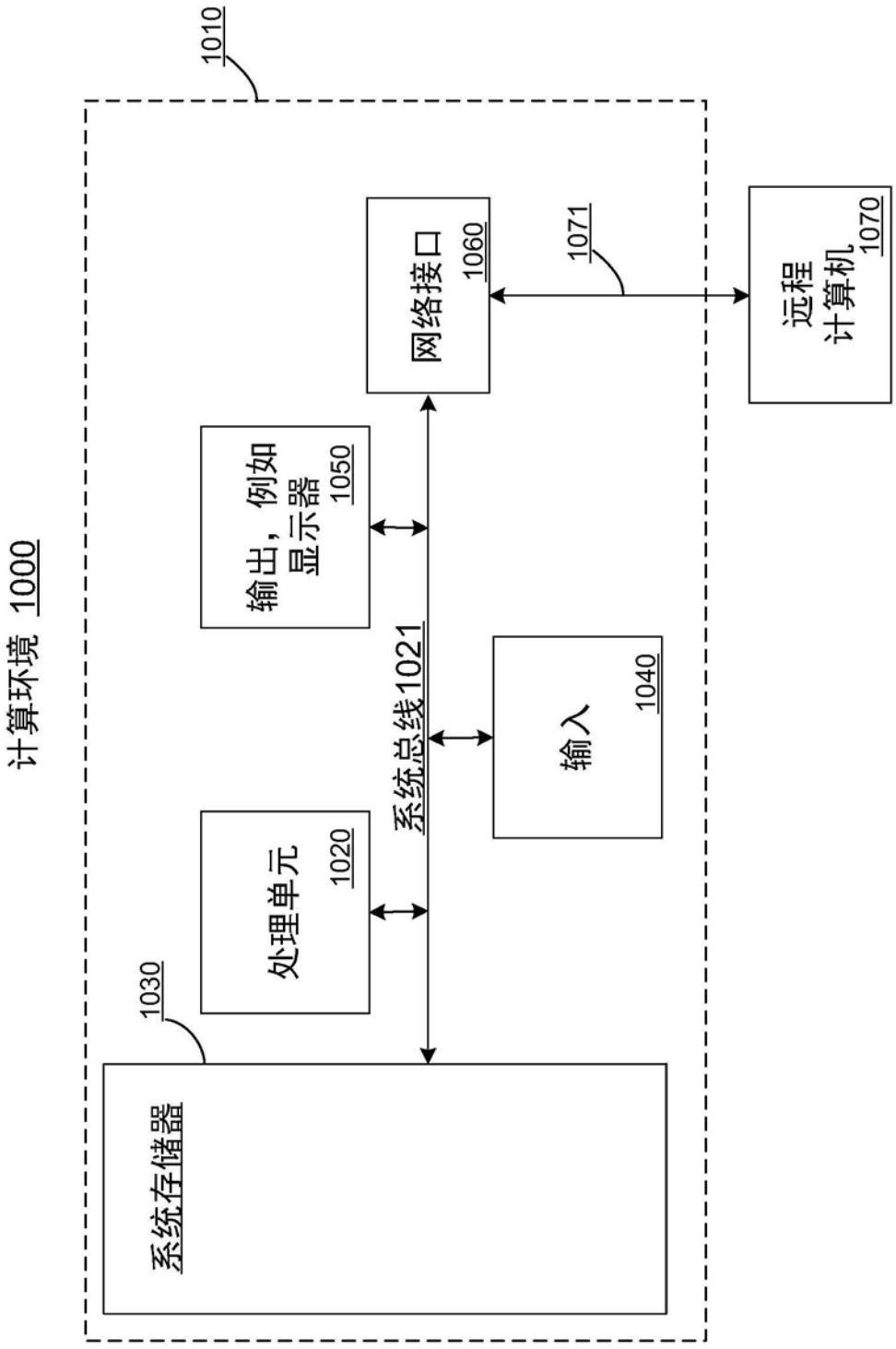

计算环境 1000
1010
网络接口 1060
1071
远程计算机 1070
输出，例如显示器 1050
系统总线1021
输入 1040
处理单元 1020
1030
系统存储器

图10