(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 106255958 B
(45)授权公告日 2019.08.02

(21)申请号 201480078392.1

(22)申请日 2014.10.23

(65)同一申请的已公布的文献号
申请公布号 CN 106255958 A

(43)申请公布日 2016.12.21

(30)优先权数据
14/262,033 2014.04.25 US

(85)PCT国际申请进入国家阶段日
2016.10.25

(86)PCT国际申请的申请数据
PCT/IB2014/065554 2014.10.23

(87)PCT国际申请的公布数据
WO2015/162465 EN 2015.10.29

(73)专利权人 索尼公司
地址 日本东京都

(72)发明人 H·森德斯特伦 A·伊斯贝里 H·格兰 J·K·马丁森

(74)专利代理机构 北京三友知识产权代理有限公司 11127
代理人 李辉 吕俊刚

(51)Int.Cl.
G06F 9/52(2006.01)
G06F 9/48(2006.01)
G06F 9/30(2006.01)
G06F 9/38(2006.01)
G06F 9/46(2006.01)

(56)对比文件
US 2011264898 A1,2011.10.27,
CN 101051282 A,2007.10.10,
CN 101556545 A,2009.10.14,
CN 102681890 A,2012.09.19,
CN 101317160 A,2008.12.03,
梁博.多核结构上的线程级推测关键技术研究.《中国博士学位论文全文数据库 信息科技辑》.2009,(第06期),第I137-3页.

审查员 王佳

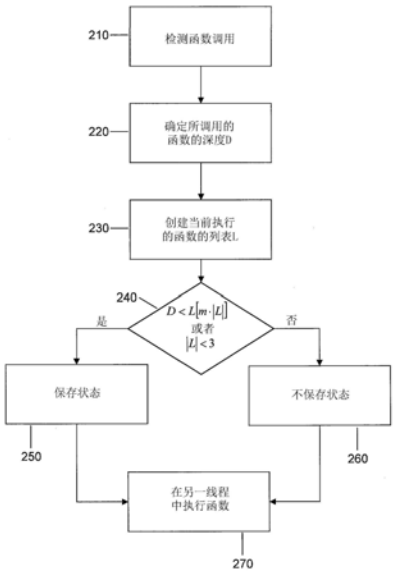
权利要求书2页 说明书5页 附图3页

(54)发明名称

用于执行程序代码的方法和装置

(57)摘要

用于执行程序代码的方法和装置。处理器装置在一个或更多个线程中执行程序代码。所述处理器装置在所述线程中的一个线程中检测函数的调用,并且在另一线程中执行所述函数。而且,所述处理器装置在如下各项之间执行选择:当在所述另一线程中开始所述函数的执行时保存所述处理器装置的状态,与当在所述另一线程中开始所述函数的执行时不保存所述处理器装置的所述状态。响应于与在所述另一线程中所述函数的所述执行有关的冲突,所述处理器装置可以执行至所述处理器装置的最后保存状态的回退,并且在调用所述函数的所述线程中执行所述函数。



CN 106255958 B

1.一种用于执行程序代码的方法,该方法包括以下步骤:

处理器装置在一个或更多个线程中执行程序代码,

所述处理器装置在所述线程中的一个线程中检测函数的调用;

所述处理器装置在另一线程中执行所述函数;以及

所述处理器装置在如下各项之间执行选择:当在所述另一线程中开始所述函数的执行时保存所述处理器装置的状态;当在所述另一线程中开始所述函数的执行时不保存所述处理器装置的所述状态,

其中,所述选择取决于对所述函数的深度与当前正执行的另一函数的深度的比较,其中,所述函数的深度是反映在指定函数内被调用的隶属函数的嵌套程度的参数,

其中,响应于所述函数的深度小于所述另一函数的深度,所述处理器装置选择保存所述状态,并且响应于所述函数的深度等于或大于所述另一函数的深度,所述处理器装置选择不保存所述状态。

2.根据权利要求1所述的方法,所述方法包括以下步骤:

所述处理器装置确定当前正由所述处理器装置执行的多个函数;以及

所述处理器装置创建所述多个函数的深度的经排序的列表;以及

所述处理器装置根据所述另一函数在所述列表中的位置来选择所述另一函数的所述深度。

3.根据权利要求2所述的方法,

其中,所述列表中的所述位置取决于响应于与在所述另一线程中的所述函数的所述执行有关的冲突而执行的、至所述处理器装置的最后保存状态的回退的数量。

4.根据权利要求3所述的方法,

其中,随着所执行的回退的数量增加,所述位置朝着所述列表的、对应于更高深度的端部移动。

5.根据权利要求1所述的方法,

其中,所述程序代码是基于脚本的web应用。

6.一种执行程序代码的装置,该装置包括:

存储器,该存储器能够操作以存储程序代码;以及

处理器装置,该处理器装置能够操作以在一个或更多个线程中执行所述程序代码;其中,所述处理器装置被配置成执行如下操作:

-检测在所述线程中的一个线程中的函数的调用,

-在另一线程中执行所述函数,以及

-在如下各项之间执行选择:当在所述另一线程中开始所述函数的执行时保存所述处理器装置的状态;当在所述另一线程中开始所述函数的执行时不保存所述处理器装置的所述状态,

其中,所述选择取决于对所述函数的深度与当前正执行的另一函数的深度的比较,其中,所述函数的深度是反映在指定函数内被调用的隶属函数的嵌套程度的参数,

其中,所述处理器装置被配置为:响应于所述函数的深度小于所述另一函数的深度,所述处理器装置选择保存所述状态,并且响应于所述函数的深度等于或大于所述另一函数的深度,所述处理器装置选择不保存所述状态。

7.根据权利要求6所述的装置，

其中，所述处理器装置被配置成执行如下操作：

-确定当前正由所述处理器装置执行的多个函数，

-创建所述多个函数的深度的经排序的列表，以及

-根据所述另一函数在所述列表中的位置来选择所述另一函数的所述深度。

8.根据权利要求7所述的装置，

其中，所述列表中的所述位置取决于响应于与在所述另一线程中所述函数的所述执行有关的冲突而执行的、至最后保存状态的回退的数量。

用于执行程序代码的方法和装置

技术领域

[0001] 本发明涉及执行程序代码的方法并且涉及对应装置。

背景技术

[0002] 计算机技术的目前趋势是,计算机系统配备有诸如多处理器核心这样的并行处理能力。而且,应用被移至因特网,例如,作为web应用。在某种程度上,web应用可以被视为针对新应用的多系统平台,因为它们考虑到在各种web浏览器中执行应用,其又可以运行在各种计算系统上。

[0003] 然而,使用并行处理能力出现几个问题。具体来说,有效利用并行处理能力可能需要专门设计的程序代码,而且可能需要显著努力来提供这种“并行化”程序代码。在开发并行化程序代码时可能出现的示例性问题是将一问题分解成并行部分,利用该可用并行处理能力来定标问题(例如:“如果该问题可以在四个处理器核心上以一特定速度处理,那么,当八个处理器核心可用时,性能会加倍吗?”),或者调试该并行化程序代码。

[0004] 对于程序代码开发者来说,按透明方式处理这种问题的一个已知方法是,被称为“线程级推测(Thread-Level Speculation)”的概念。对于线程级推测的情况下,在执行该程序代码期间调用的函数在一新线程中执行,例如,在分离处理器核心上运行。如果这种在新线程中执行该函数造成冲突,那么,执行回退,并接着在现有线程中执行该函数,典型地讲,在由其调用该函数的线程中。

[0005] 然而,无论何时调用函数都需要能够执行回退可能导致显著的存储器使用,因为在开始在该新线程中执行该函数时,该系统的状态需要加以存储。

[0006] 因此,需要一种考虑到在多个线程中有效执行程序代码的技术。

发明内容

[0007] 根据实施方式,提供了一种方法。根据所述方法,处理器装置在一个或更多个线程中执行程序代码。所述程序代码可以是基于脚本的web应用,例如,基于JavaScript。所述处理器装置在所述线程中的一个线程中检测函数的调用,并且在另一线程中执行所述函数。而且,所述处理器装置在如下各项之间执行选择:当在所述另一线程中开始执行所述函数时,保存所述处理器装置的状态,与当在所述另一线程中开始执行所述函数时,不保存所述处理器装置的所述状态。响应于与在所述另一线程中执行所述函数有关的冲突,所述处理器装置可以执行回退至所述处理器装置的最后保存状态,并且在调用所述函数的所述线程中执行该函数。

[0008] 根据实施方式,所述选择取决于所执行的回退的数量。而且,所述选择可以取决于当前由所述处理器装置执行的函数的数量。例如,响应于所述函数数量小于一阈值(例如,三),所述处理器装置可以选择保存所述状态的选项,以及响应于所述函数的数量等于或大于所述阈值,所述处理器装置可以选择不保存所述状态的选项。

[0009] 根据实施方式,所述选择取决于所述函数的深度。具体来说,所述选择可以取决于

对所述函数的深度与当前正执行的另一函数的深度的比较。例如，所述处理器装置可以确定当前由所述处理器装置执行的多个函数，并且创建所述多个函数的深度的经排序的列表。接着，所述处理器装置可以根据所述另一函数在所述列表中的位置来选择所述另一函数的所述深度。所述列表中的所述位置可以取决于所执行的回退的数量。例如，这种相关性可以使得，随着所执行的回退的数量增加，所述列表中的所述位置可以朝着所述列表的、对应于更高深度的端部移动。响应于所述函数的深度小于所述另一函数的深度，所述处理器装置可以选择保存所述状态的选项。响应于所述函数的深度等于或大于所述另一函数的深度，所述处理器装置可以选择不保存所述状态的选项。

[0010] 根据另一实施方式，提供了一种装置。所述装置包括可操作以存储程序代码的存储器。而且，所述装置包括处理器装置，其可操作以在一个或更多个线程中执行所述程序代码。例如，所述处理器装置可以包括具有多个核心的处理器甚或多个单核或多核处理器。而且，所述处理器装置可以支持通过单个处理器核心执行多个线程。所述程序代码可以是基于脚本的web应用，例如，基于JavaScript。所述处理器装置被配置成，在所述线程中的一个线程中检测函数的调用，并且在另一线程中执行所述函数。而且，所述处理器装置被配置成在如下各项之间执行选择：当在所述另一线程中开始执行所述函数时，保存所述处理器装置的状态；与当在所述另一线程中开始执行所述函数时，不保存所述处理器装置的所述状态。根据实施方式，所述处理器装置还可以被配置成，检测与在所述另一线程中执行所述函数有关的冲突，并且响应于这种冲突，执行回退至所述处理器装置的最后保存状态，并且在调用所述函数的所述线程中执行该函数。

[0011] 根据实施方式，所述选择取决于执行回退的数量。而且，所述选择可以取决于当前由所述处理器装置执行的函数的数量。例如，响应于所述函数的数量小于一阈值(例如，三)，所述处理器装置可以选择保存所述状态的选项，而响应于所述函数的数量等于或大于所述阈值，所述处理器装置可以选择不保存所述状态的选项。

[0012] 根据实施方式，所述选择取决于所述函数的深度。具体来说，所述选择可以取决于对所述函数的深度与当前正执行的另一函数的深度的比较。例如，所述处理器装置可以确定当前通过所述处理器装置执行的多个函数，并且创建所述多个函数的深度的经排序的列表。接着，所述处理器装置可以根据所述另一函数在所述列表中的位置来选择所述另一函数的深度。所述列表中的所述位置可以取决于所执行的回退的数量。例如，这种相关性可以使得，随着所执行的回退的数量增加，所述列表中的所述位置可以朝着所述列表的、对应于更高深度的端部移动。响应于所述函数的深度小于所述另一函数的深度，所述处理器装置可以选择保存所述状态的选项。响应于所述函数的深度等于或大于所述另一函数的深度，所述处理器装置可以选择不保存所述状态的选项。

[0013] 下面，参照附图来更详细地描述本发明的上述和另一些实施方式。

附图说明

[0014] 图1示意性地例示了根据本发明的实施方式的装置。

[0015] 图2示出了用于例示根据本发明的实施方式的方法的流程图。

[0016] 图3示出了用于例示根据本发明的实施方式的方法的另一处理的流程图。

具体实施方式

[0017] 下面，对本发明的示例性实施方式进行更详细描述。必须明白的是，下面的描述仅出于例示本发明的原理的目的而给出，而非按限制性意义进行。相反地，本发明的范围仅通过所附权利要求书来限定，而非通过此后描述的示例性实施方式来限制。

[0018] 所例示的实施方式涉及通过具有并行处理能力的处理器装置来执行程序代码。该处理器装置例如可以是多核处理器，即，配备有多个程序核心的处理器，其支持并行执行多个线程，例如，对应处理器核心上的每个线程。而且，该处理器装置可以包括多个单核或多核处理器。更进一步，该处理器装置可以支持通过单一处理器核心并行执行多个线程，例如利用如针对由Intel Corporation制造的某些处理器所提供的超线程技术。

[0019] 该处理器装置可以是各种计算机系统的一部分，举例来说，诸如智能电话或平板计算机这样的移动装置的处理器模块。然而，该处理器装置还可以在其它种类装置中加以利用。

[0020] 图1示意性地例示了这种装置100的示例性结构。在图1的实施例中，假定该装置是支持针对无线网络的无线电连接的移动装置。

[0021] 如所示的，装置100包括无线电接口110。而且，装置100包括配备有多个处理器核心141、142、143、144的处理器140。该多核处理器140是上述处理器装置的示例。

[0022] 而且，该装置包括联接至处理器140的存储器150。该存储器150包括具有要通过处理器140执行的程序代码的程序代码模块160、170。在该例示例中，这些程序代码模块包括应用代码模块160和操作系统代码模块170。

[0023] 该应用代码模块160可以包括要通过处理器140执行的应用的程序代码，如基于脚本的web应用，例如，基于JavaScript。应用代码模块160的程序代码还可以动态地从因特网下载。操作系统代码模块170可以包括该装置的操作软件，包括用于通过处理器控制执行该应用的代码，例如，采用脚本执行引擎的形式。这种脚本执行引擎例如可以是web浏览器应用的一部分。

[0024] 要明白的是，如图1所示的结构仅仅是示例性的，并且该装置100还可以包括没有例示的其它部件，例如，用于实现用户接口或其它通信接口的结构或程序代码模块。而且，要明白的是，所示结构的详细实现可以改变。例如，存储器150可以包括：只读存储器(ROM)、随机存取存储器(RAM)、闪速存储器、磁存储部等。

[0025] 在如下进一步例示的实施例中，处理器装置(例如，多核处理器140)用于执行一应用的程序代码，如在应用代码模块160中的web应用。该程序代码被假定成为，基于脚本的web应用的程序代码，例如，基于JavaScript。为使能够透明使用该处理器装置的并行处理能力，该处理器装置支持线程级推测。

[0026] 该装置被假定成为是事件驱动的。当应用中出现一事件(例如，鼠标点击或其它用户输入)时，这种事件典型地对应于函数调用。当处理器装置遇到应用中的函数调用时，其首先尝试在新线程中运行该函数(即，推测在该新线程中成功执行该函数)。如果存在有关在该新线程中执行该函数的冲突，则该处理器装置返回至推测之前的状态(即，执行回退)，并且在不推测的情况下重新执行该函数，例如，通过在调用该函数的同一线程中执行该函数。为使能够实现这种回退，处理器装置保存推测之前的状态。推测还可以嵌套，即，在一线程中推测地执行的函数可以调用另一函数，其在又一线程中推测地执行。

[0027] 如在此例示的概念基于对于某些应用(例如,基于JavaScript的web应用)来说冲突非常稀少的观察。例如,某一应用的程序代码的执行可以涉及5000次推测,其中仅3%导致回退。因此,对于每一个推测来说,可能不需要保存该状态,而仅针对这些推测的一部分需要保存该状态。在所示的概念中,当开始在新线程中推测执行函数时,这根据由处理器装置所执行的选择来加以考虑。具体来说,处理器装置在其中通过该处理器装置保存开始执行该函数时的状态的第一选项,与其中不通过该处理器装置保存开始执行该函数时的状态的第二选项之间决定。使用后面的选项考虑到避免过度使用存储器资源来保存仅潜在地需要回退的信息。如果在推测执行函数期间出现冲突,并且因选择第二选择而未保存开始推测执行该函数时的状态,则处理器装置可以执行回退至最新保存状态。如下更详细说明的,第一选项与第二选项之间的决定可以基于动态自适应式启发(heuristic)。该启发例如可以考虑到被调用的函数的深度、所执行的回退的数量、以及当前通过该处理器装置执行的其它函数。

[0028] 将在如图2和图3的流程图所示的执行程序代码的方法的背景下,对该启发的示例性实现进行说明。

[0029] 首先,在开始执行该应用的程序代码之前,可以将列表位置指针m的初始值例如设置成m=0.5。而且,可以将表示回退数量的回退计数器nr设置成起始值,例如,nr=1。

[0030] 如果处理器装置检测到调用了函数(如步骤210所示),则其可以应用用于在第一选项(保存该状态)与第二选项(不保存该状态)之间选择的下列处理。

[0031] 在步骤220,处理器装置可以确定所调用的函数的深度D。该深度D是反映在指定函数内被调用的隶属函数的嵌套程度的参数。例如,如果在指定函数内未调用进一步的函数,则该函数具有深度D=1。如果在该函数内调用了隶属函数,则其深度D增加了该隶属函数的深度。

[0032] 在步骤230,处理器装置可以创建当前正由处理器装置(按不同线程)执行的函数的深度D的列表L。在该例示例中,假定列表L按升序创建,即,最低深度D形成列表L的开头,而其它深度D按增加深度D的次序跟随。在典型情况下,该列表还可以包括具有同一值D的多个成员。

[0033] 在步骤240,处理器装置基于列表L而执行检查。具体来说,处理器装置可以检查列表L的长度,即,lLl。如果长度lLl小于阈值(例如,3),则处理器装置继续步骤250,如分支"Y"所示。在步骤250,处理器装置保存该状态,即,选择第一选项。因此,针对初始函数调用来说,例如,第一调用的函数,保存该状态。如进一步所示,处理器装置可以比较所调用的函数的深度D与当前执行的另一函数的深度,即,列表L中的深度。执行该比较的深度处于列表L中的位置m*lLl处,即,对应于列表成员L[m*lLl]。由于上述初始设置的值m,因而,该位置初始接近列表中部,典型地对应于中等深度值。具体来说,处理器装置可以检查所调用的函数的深度D是否小于列表成员L[m*lLl]的值。如果是这种情况,则处理器装置继续步骤250,如分支"Y"所示。在步骤250,处理器装置保存该状态,即,选择第一选项。因此,如果所调用的函数的深度相对较小,则也保存该状态。

[0034] 要不然,如果长度lLl等于或大于该阈值,或者所调用的函数的深度D等于或大于列表成员L[m*lLl]的值,则处理器装置继续步骤260,如分支"N"所示。在步骤260,处理器装置不保存该状态,即,选择第二选项。

[0035] 在步骤270，处理器装置在另一线程中执行所调用的函数，即，执行针对该函数的线程级推测。

[0036] 如上提到，该线程级推测可以导致与在另一线程中执行该函数有关的冲突。通过图3的流程图例示了可以在出现这种冲突时所应用的示例性处理。

[0037] 在步骤310，处理器检测与在另一线程中推测执行该函数有关的冲突。

[0038] 在步骤320，响应于检测到该冲突，处理器装置执行回退至最新存储状态。如果在开始在该另一线程中推测执行该函数时保存该状态，则执行回退至该状态。否则执行回退至更早存储状态。

[0039] 响应于该回退，处理器装置在步骤330递增回退计数器，即，将nr增加至Nr+1。而且，处理器装置递增列表位置指针m。这通过取决于回退计数器的递增值来实现。在该例示例中，列表位置指针m被假定成根据以下增加：

[0040] m:＝m+1/2nr+1。

[0041] 就是说，随着回退数量nr的增加，列表位置指针n朝着列表L的末端移动，即，朝着更高深度。这意味着，在图2的步骤240处选择第一选项的可能性随着回退的出现而增加，由此，允许更有效处理回退。另一方面，如果没有出现回退，则保持在步骤240选择第二选项的某一可能性，由此，允许避免因保存状态而过度使用存储器。

[0042] 如可以看出，如上说明的概念允许在多个线程中有效执行程序代码。由于使用线程级推测，因而，这可以按对于程序代码的开发者来说透明的方式加以实现。另一方面，避免了过度使用存储器资源，以供使能够进行回退。

[0043] 要明白的是，如上说明的概念容易进行各种修改。例如，所例示启发可以按不同方式来修改。举例来说，代替按升序经排序的列表L地，该列表可以按降序排序，并且列表位置指针代替递增地进行递减。而且，该概念可以应用于各种装置中，例如，在固定计算机系统中。而且，该概念不限于特定处理器实现。更进一步，该概念可以结合各类编程语言来使用。

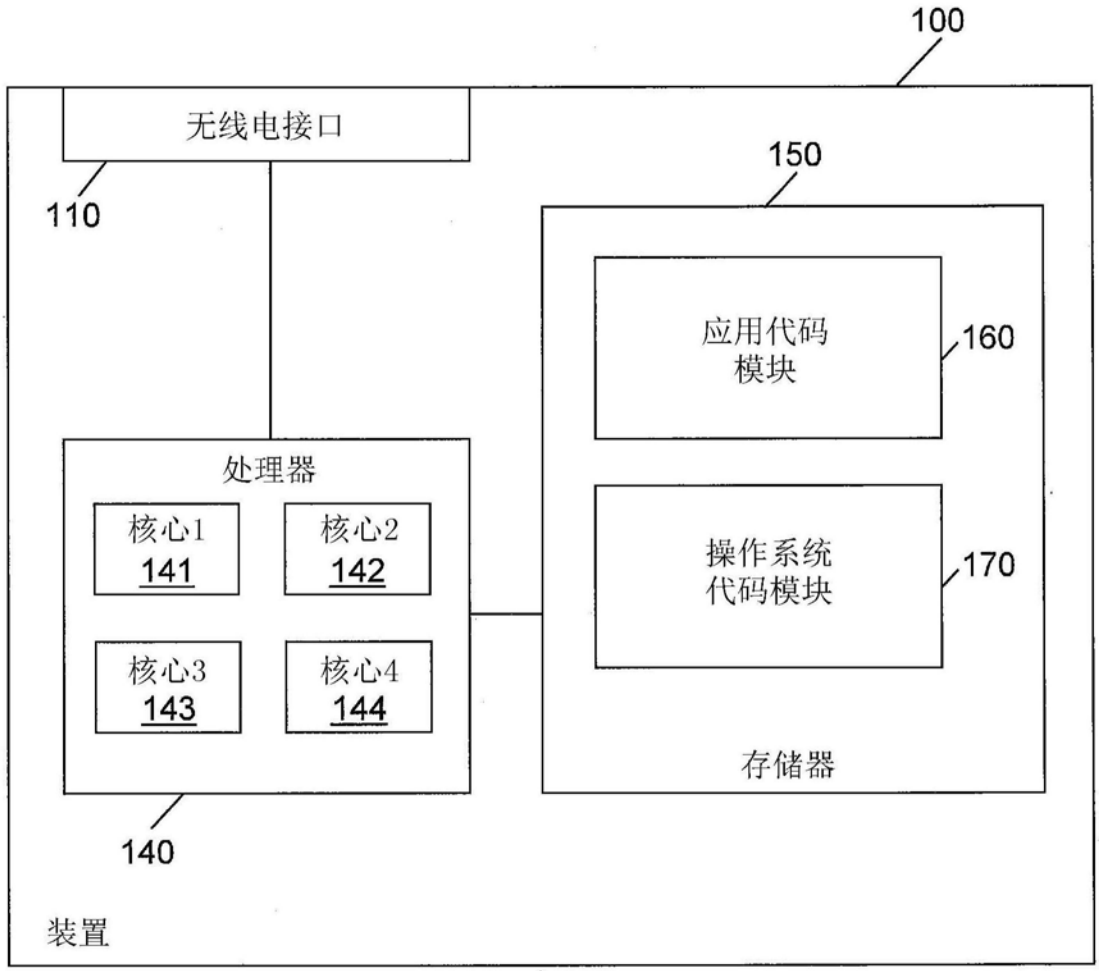

100
无线电接口
110
150
应用代码
模块
160
处理器
核心1
141
核心2
142
操作系统
代码模块
170
核心3
143
核心4
144
存储器
140
装置

图1

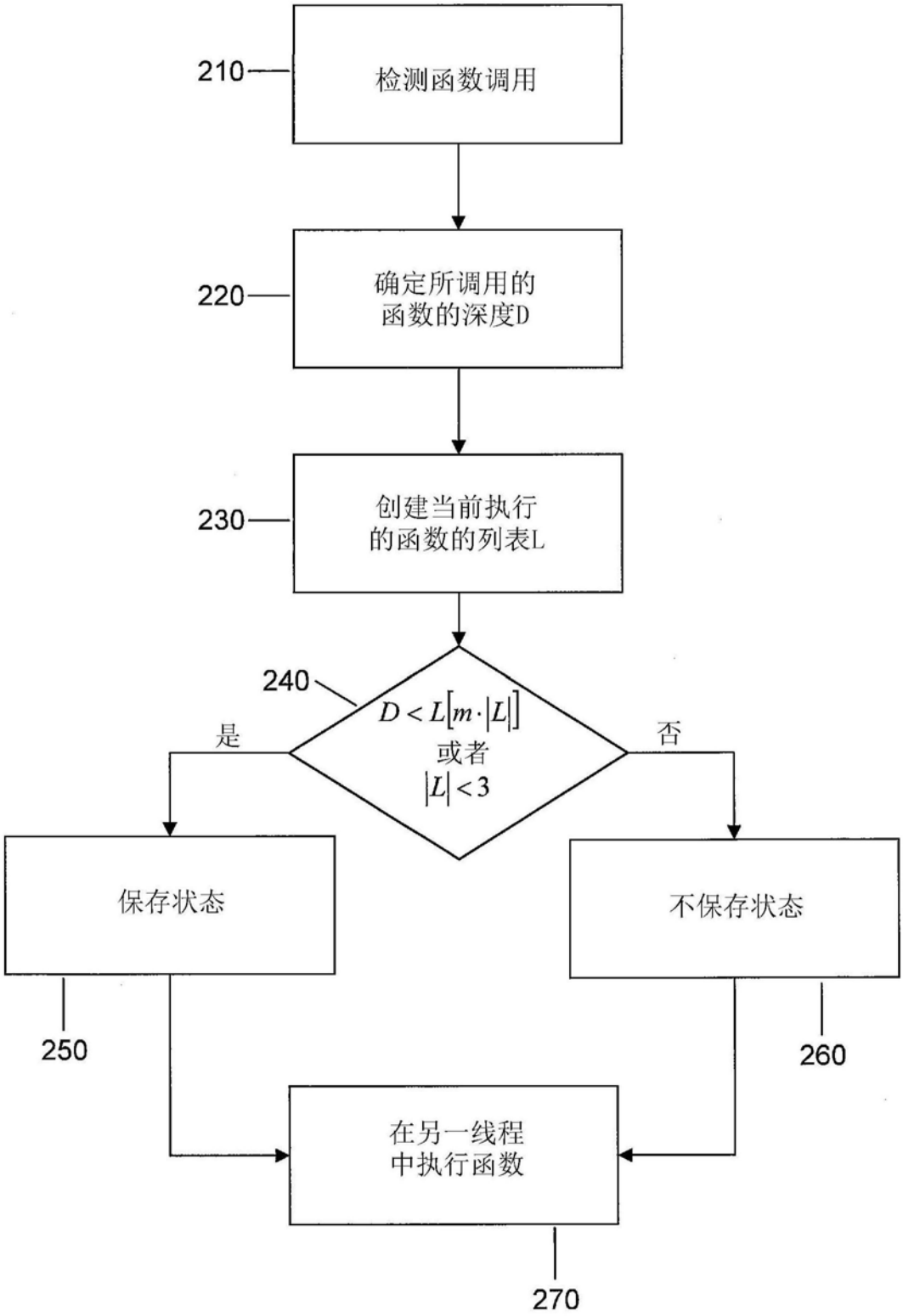

210
检测函数调用
220
确定所调用的
函数的深度D
230
创建当前执行
的函数的列表L
240
$D < L[m \cdot |L|]$
或者
$|L| < 3$
是
否
保存状态
不保存状态
250
260
在另一线程
中执行函数
270

图2

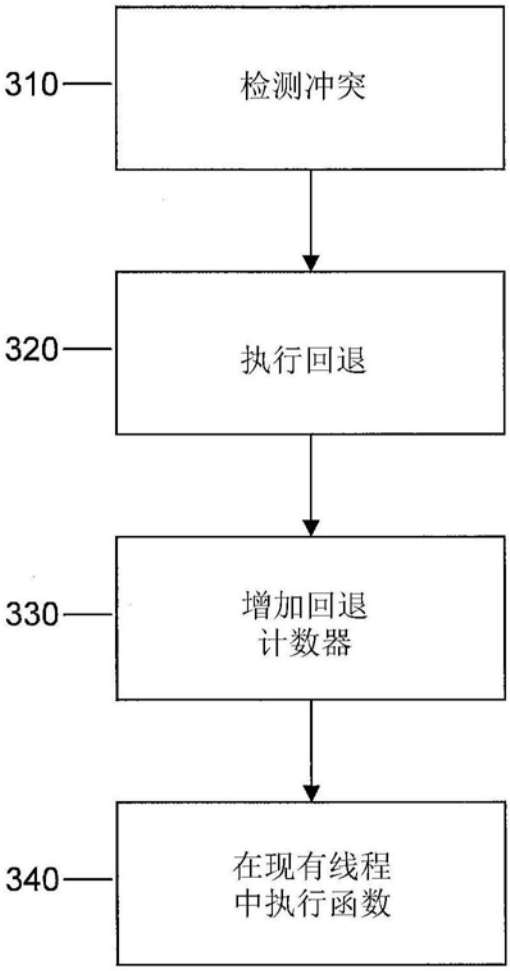


图3