(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 106104465 B
(45)授权公告日 2019.03.01

(21)申请号 201580015800.3
(22)申请日 2015.03.06
(65)同一申请的已公布的文献号
申请公布号 CN 106104465 A
(43)申请公布日 2016.11.09
(30)优先权数据
14/227,003 2014.03.27 US
(85)PCT国际申请进入国家阶段日
2016.09.23
(86)PCT国际申请的申请数据
PCT/EP2015/054731 2015.03.06
(87)PCT国际申请的公布数据
WO2015/144421 EN 2015.10.01
(73)专利权人 国际商业机器公司
地址 美国纽约
(72)发明人 L•C•海勒 J•P•库巴拉
F•Y•布萨巴 J•D•布拉德伯里
M•法雷尔 D•L•奥西塞克
D•格雷纳 T•斯莱格尔
D•W•施密特 C•盖尼
C•雅各比
(74)专利代理机构 北京市中咨律师事务所 11247
代理人 于静 张亚非
(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)
G06F 9/48(2006.01)
G06F 9/50(2006.01)
(56)对比文件
US 4816991 A,1989.03.28,
CN 1906580 A,2007.01.31,
EP 0145960 A2,1985.06.26,
审查员 陈敏

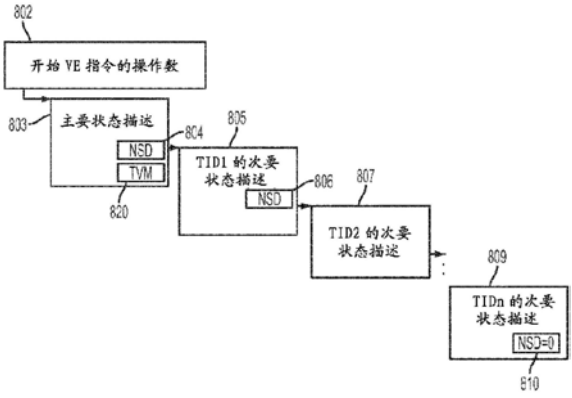
权利要求书1页 说明书14页 附图11页

(54)发明名称

用于在计算机中分派多个线程的开始虚拟执行指令

(57)摘要

各实施例涉及计算机中的多线程。一个方面为一种计算机，所述计算机包括具有核心的配置，所述核心包括物理线程且可在单线程(ST)模式及多线程(MT)模式中操作。所述计算机还包括主机程序，所述主机程序被配置为在所述核心上以所述ST模式执行以发出开始虚拟执行(开始VE)指令，以分派包括客体虚拟机(VM)的客体实体。所述开始VE指令由所述核心执行且包括从由所述开始VE指令指定的位置获得具有客体状态的状态描述。所述执行包括基于所述客体状态来判定所述客体实体包括单个客体线程还是多个客体线程，及基于所述客体状态以及所述客体实体包括单个客体线程还是多个客体线程的判定而在所述MT模式或所述ST模式中开始所述客体线程。



CN 106104465 B

1.一种用于分派配置中的多个线程的计算机实施的方法,所述配置包括被启用以在单线程ST模式及多线程MT模式中操作的核心,所述核心包括物理线程,所述方法包括:

由在所述核心上以所述单线程ST模式执行的主机程序发出开始虚拟执行指令以在所述核心上分派客体实体,所述客体实体包括客体虚拟机VM的全部或一部分,且所述开始虚拟执行指令由所述核心执行,所述执行包括:

从由所述开始虚拟执行指令指定的位置获得第一状态描述,所述第一状态描述具有客体状态;

基于所述客体状态来判定所述客体实体包括单个客体线程还是多个客体线程;

基于所述客体状态且基于判定所述客体实体包括多个客体线程,在所述核心上以所述多线程MT模式开始所述多个客体线程,其中所述多个客体线程彼此独立地执行;及

基于所述客体状态且基于判定所述客体实体包括单个客体线程,在所述核心上以所述单线程ST模式开始所述单个客体线程。

2.如权利要求1所述的方法,其中所述核心包括当所述核心处于所述多线程MT模式中时控制所述物理线程之间的共享资源的使用的计算机指令。

3.如权利要求1所述的方法,其中当所述核心处于所述多线程MT模式中时,所述主机程序将所述客体实体管理为单个逻辑核心。

4.如权利要求1所述的方法,其中由所述主机程序利用线程有效性屏蔽以指示所述客体实体中的一个或多个客体线程的有效性。

5.如权利要求1所述的方法,其中所述方法进一步包括在将控制返回至所述主机程序之前退出所述客体实体中的一个或多个客体线程中的全部客体线程。

6.如权利要求1所述的方法,其中基于所述客体实体包括多个客体线程,在第一状态描述中含有用于一个线程的状态数据,且在额外状态描述中含有用于一个或多个额外线程中的每一者的状态数据。

7.如权利要求1所述的方法,其中基于所述客体实体包括多个客体线程,将全部所述客体线程所共有的状态数据的至少一部分存储在单个位置中。

8.如权利要求6所述的方法,其中所述第一状态描述及所述额外状态描述被存储在环及列表结构中的至少一者中。

9.如权利要求1所述的方法,进一步包括执行无动作退出,所述无动作退出包括基于来自其他客体线程的请求而退出客体线程。

10.一种用于分派配置中的多个线程的系统,其包括适于执行如任一前述方法权利要求所述的方法的所有步骤的装置。

用于在计算机中分派多个线程的开始虚拟执行指令

背景技术

[0001] 本发明一般地涉及多线程(MT),且更具体言之,涉及一种用于在计算机中分派多个线程的开始虚拟执行(开始VE)指令。

[0002] 多线程(MT)提供用于在不需要添加额外核心的情况下增加可在单个物理处理器核心内并行操作的处理器线程的数目的手段。理想地,MT通过使一个或多个线程使用核心硬件的当前未由在同一核心上运行的其他线程使用的部分来提供此增加能力。举例而言,在由高速缓存未命中所引起的延时或一个线程中的其他延迟期间,一个或多个其他线程可利用核心资源,因此增加资源的利用率。实际上,即使此共享导致线程之间的一些干扰且需要某一额外硬件,但MT仍提供使用较少硬件执行每一线程工作的能力,如果每一线程在其自身隔离核心硬件上执行,则将需要较多硬件。常常,当线程之间的硬件资源的共享亦减少计算机系统上的将信息(诸如,来自存储器的数据)提供至两个独特核心的整体压力时,可从MT获得额外益处。

[0003] 通常,尽管MT提供硬件节省,但另一工作线程的添加消耗系统管理程序级别处的使用额外单独核心提供增加能力将需要的相同协调成本。在许多情况下,一旦达成某一按比例调整比率,在工作线程(无论执行于单个核心抑或共享核心上)之间协调资源的开销是显著的且会减小或甚至胜过由运行独立工作线程的能力所得到的益处。亦即,当待管理事物的数目增加时通常存在更多管理开销。

发明内容

[0004] 本发明提供一种如权利要求1所述的计算机实施的方法以及对应的系统和计算机程序。

附图说明

[0005] 当本说明书完结时在权利要求中特定地指出且清楚地要求保护被视为实施例的主题。实施例的前述及其他特征及优点自结合附图而进行的以下详细描述显而易见,在附图中:

[0006] 图1描绘可根据一个实施例实施的计算环境;

[0007] 图2描绘可根据一个实施例实施的物理处理器;

[0008] 图3描绘可根据一个实施例实施的计算环境;

[0009] 图4描绘根据一个实施例的多线程化(MT)逻辑线程的状态描述;

[0010] 图5描绘根据一个实施例的线程有效性屏蔽(TVM)的方块图;

[0011] 图6描绘根据一个实施例的固定偏移状态描述组;

[0012] 图7描绘根据一个实施例的被指定为地址列表的状态描述组;

[0013] 图8描绘根据一个实施例的被指定为链接列表的状态描述组;

[0014] 图9描绘根据一个实施例的被指定为循环列表或环的状态描述组;

[0015] 图10描绘根据一个实施例的核心分派过程;

[0016] 图11描绘根据一个实施例的从虚拟执行的协调退出；

[0017] 图12描绘根据一个实施例的系统控制区的方块图；

[0018] 图13描绘根据一个实施例的用于在多线程化核心之间协调的过程流；及

[0019] 图14描绘根据一个实施例的计算机可读介质。

具体实施方式

[0020] 本文中所描述的实施例可用于减少多线程(MT)环境中的系统管理程序管理开销。如本文所描述，多个线程的管理可在将多个线程作为单个逻辑核心(core)管理的系统管理程序与在多个线程存取物理核心的资源时管理多个线程之间的交互的机器之间分割。此可通过允许系统管理程序基于逻辑核心管理许多的系统管理程序基础设施资源及允许机器基于较细粒度线程管理其他资源而导致显著减少多线程(MT)开销成本。一个实施例包括可由在单个线程(ST)上运行的系统管理程序执行的核心分派指令。核心分派指令的执行(在本文中称为“指定MT的开始VE指令”)可引起构成客体虚拟机(VM)的全部或一部分的多个客体逻辑线程待在单个物理核心上分派。在一实施例中，由系统管理程序用于分派客体的指令指定待分派的客体是单线程化还是多线程化的。

[0021] 本文中所描述的实施例可包括用于管理多线程化逻辑核心的分派的结构，诸如用于指示客体逻辑核心内的哪些逻辑线程当前有效的线程有效性屏蔽及包括状态描述环的状态描述组。另外，主要及次要状态描述及字段类型(例如，主要、核心共享、线程特定)可被实施为在具有多个线程的逻辑核心被分派时允许计算机资源的有效管理。另外，可提供逻辑核心内的所有线程同时退出虚拟执行的协调退出以简化系统管理程序及逻辑核心管理功能两者。

[0022] 各实施例可包括由系统管理程序维护的在本文中称为核心导向式系统控制区(COSCA)的控制结构。COSCA由系统管理程序及机器两者使用以管理可影响客体配置中的多个逻辑处理器的某些功能。COSCA的一个实施例被实现为树状结构，其中叶表示逻辑核心且每一叶含有对应于该核心的线程的列表。COSCA结构可含有允许系统管理程序容易地存取用于特定核心中的所有线程的状态描述的字段(例如，状态描述地址)。

[0023] 如本文所用，术语“线程”指单个指令流及其相关联状态。亦即，在架构级别处，每一逻辑线程表示独立CPU或处理器。在硬件级别处，物理线程是当逻辑线程被分派时结合维持该客体状态而执行与该逻辑线程相关联的指令流。正是机器对该线程状态的维护减少了在系统管理程序级别处所需的管理。可用于由逻辑核心使用的逻辑线程的总数受可用于物理核心的物理线程的总数限制。

[0024] 如本文中所使用，术语“物理核心”指执行一个或多个独立指令流或线程但共享许多基本资源(诸如，执行单元及低级别高速缓存)的硬件处理单元。可以多种方式进行此共享，所述方式包括使每一线程在独立时间使用相同硬件资源，或使资源被逻辑上共享，同时每一物理项被标记线程识别符。线程(例如，常常需要资源A但仅很少需要资源B的一个线程及通常使用资源B而不使用资源A的另一线程)之间的适当整合效果能够改进此共享的效率。如本文中所使用，术语“机器”指包括于物理核心中的硬件以及用于支持物理核心的毫码及其他硬件。

[0025] 如本文中所使用，术语“客体VM”及“客体”可互换地用以指可包括单个CPU或多个

CPU的单个客体配置。如本文中所使用，术语“逻辑核心”指被定义以作为指定MT的开始VE指令的一部分而一起分派的逻辑客体线程或CPU的组。客体VM可由单个逻辑核心(ST或MT)或多个逻辑核心(其中的每一者亦可为ST或MT)组成。

[0026] 如本文中所使用，术语“软件”指系统管理程序(例如，PR/SM或zVM)或客体操作系统或由于开始VE指令而分派的应用程序。

[0027] 如本文中所使用，术语“系统管理程序”及“主机”指管理系统资源且分派客体逻辑处理器以在物理硬件上执行的程序。

[0028] 用于分派客体点至状态描述或状态描述组的开始VE指令的操作数定义该客体处理器或核心的状态。状态描述自身具有指向可视为状态描述的扩展的“卫星(satellite)块”的指针且包括进一步定义该客体核心或处理器的状态的额外信息。如本文中所使用，术语“状态描述”不仅指状态描述自身而且指这些卫星块。图12中描绘了核心导向式系统控制区(COSCA)(这些卫星块中的一者)。

[0029] 现转向图1，大体展示可由例示性实施例实施的计算环境100。计算环境100可(例如)基于由国际商业机器公司(Armonk,New York)提供的z/Architecture。z/Architecture描述于2012年8月的题为“z/Architecture Principles of Operation”的IBM®公开(IBM公开第SA22-7832-09号)中。在一个实例中，基于z/Architecture的计算环境包括由国际商业机器公司(Armonk,New York)提供的eServer zSeries。

[0030] 作为一个实例，计算环境100可包括耦合至系统控制器120的处理器复合体102。处理器复合体102可包括(例如)一个或多个分区104(例如，逻辑分区LP1至LPn)、一个或多个物理核心106(例如，核心1至核心m)及级别0系统管理程序108(例如，逻辑分区管理器)，下文描述其中的每一者。

[0031] 每一逻辑分区104能够充当单独系统。亦即，每一逻辑分区104可被独立地重设，视需要最初加载有操作系统110且以不同程序操作。在逻辑分区104中运行的操作系统110或应用程序可看起来存取完全及完整系统，但实际上存取仅其可获得的一部分。硬件与许可内部代码(通常称为微码或毫码或固件)的组合将程序保持在一个逻辑分区104中以免干扰不同逻辑分区104中的程序。这允许若干不同逻辑分区104以时间分片方式在单个或多个物理核心106上操作。在一实施例中，每一物理核心包括一个或多个中央处理器(在本文中亦称为“物理线程”)。在图1中所展示的实例中，每一逻辑分区104具有驻留操作系统110，其针对一个或多个逻辑分区104而可不同。在每一逻辑分区104中运行的操作系统110为虚拟机或客体配置的一个实例。在一个实施例中，操作系统110为由国际商业机器公司(Armonk,New York)提供的z/OS®操作系统。

[0032] 物理核心106包括被分配至逻辑分区104的物理处理器资源。逻辑分区104可包括一个或多个逻辑处理器，其中的每一者表示分配至分区104的所有或一部分物理处理器资源。物理核心106可专用于特定分区104的逻辑核心，使得基础核心106的物理处理器资源被保留用于该分区104；或与另一分区104的逻辑核心共享，使得基础核心资源的物理处理器资源潜在地可用于另一分区104。

[0033] 在图1中所展示的实施例中，逻辑分区104由级别0系统管理程序108管理，所述级别0系统管理程序由在物理核心106上运行的固件实施。逻辑分区104及系统管理程序108各自包括驻留在与物理核心106相关联的中央存储装置(存储器)的相应部分中的一个或多个

程序。系统管理程序108的一个实例为由国际商业机器公司(Armonk,New York)提供的Processor Resource/System Manager(PR/SM™)。

[0034] 在图1中耦合至中央处理器复合体102的系统控制器120可包括负责在发出请求的不同处理器之间仲裁的集中式逻辑。举例而言,当系统控制器120接收存储器存取请求时,其判定是否允许存取该存储位置,且如果允许,则将存储位置的内容提供至中央处理器复合体102,同时维护该复合体内的处理器之间的存储器一致性。

[0035] 现转向图2,大体展示根据一个实施例的用于实施机器或物理核心(诸如,图1中的物理核心106)的处理电路200的方块图。处理电路200可包括在多处理环境中的多个物理核心中的一个物理核心。图2中所展示的处理电路200包括可将处理电路200耦合至其他核心及外围设备的系统控制器接口单元202。系统控制器接口单元202亦可将Dcache 204(其读取及存储数据值)、Icache 208(其读取程序指令)及高速缓存接口单元206连接至外部存储器、处理器及其他外围设备。

[0036] Icache 208可结合指令取回单元(IFU)210提供指令流的加载,所述指令取回单元预先取回指令且可包括推测性加载及分支预测能力。可将所取回指令提供至指令解码单元(IDU)212以用于解码成指令处理数据。

[0037] IDU 212可将指令提供至发出单元214,所述发出单元可控制指令至各种执行单元(诸如,用于执行一般运算的一个或多个定点单元(FXU)216及用于执行浮点运算的一个或多个浮点单元(FPU)218)的发出。FPU 218可包括二进制浮点单元(BFU)220、十进制浮点单元(DFU)222或任何其他浮点单元。发出单元214亦可经由一个或多个LSU管线耦合至一个或多个加载/存储单元(LSU)228。多个LSU管线被视为用于执行加载及存储以及用于分支的地址产生的执行单元。LSU 228及IFU 210两者可利用转换后备缓冲器(TLB)230以提供用于操作数及指令地址的缓冲转换。

[0038] FXU 216及FPU 218耦合至各种资源,诸如通用寄存器(GPR)224及浮点寄存器(FPR)226。GPR 224及FPR 226通过LSU 228提供用于自Dcache 204加载及存储的数据值的数据值存储。

[0039] 现转向图3,大体展示可由一个实施例实施的计算环境300。图3中所展示的计算环境300类似于图1中所展示的计算环境100,外加在逻辑分区104(标记为LP2)中执行的级别1系统管理程序302。如图3中所展示,级别1系统管理程序302可提供先前关于系统管理程序108(在本文中亦称为“级别0系统管理程序”)描述的相同系统管理程序功能,诸如在标记为LP2的逻辑分区104内的多个操作系统(例如,在虚拟机VM1 304、VM2 306及VM3 308中运行的OS1 314、OS2 312及OS3 310)之间的资源的透明时间分片及这些操作系统的彼此隔离。图3中所展示的实施例包括作为实例的三个虚拟机且其他实施例可基于应用需求而包括更多或更少虚拟机。

[0040] 如图3中所展示,标记为LP1的逻辑分区104具有驻留操作系统110,且标记为LP2的逻辑分区104运行级别1系统管理程序302,级别1系统管理程序302继而建立虚拟机304、306、308的级别1系统管理程序302,所述虚拟机中的每一者运行其自身的驻留操作系统314、312、310。任何数目个逻辑分区104可运行级别1系统管理程序302。在一实施例中,级别1系统管理程序302为由国际商业机器公司(Armonk,New York)提供的z/VM系统管理程序。在各种逻辑分区中运行的常驻操作系统可不同,且当在级别1系统管理程序302下运行时,

单个分区104(例如,LP2)内的驻留操作系统(例如,操作系统314、312、310)亦可不同。在一实施例中,在标记为LP1的逻辑分区104中的操作系统110为由国际商业机器公司(Armonk, New York)提供的z/OS操作系统。在一实施例中,操作系统310及312为Linux且操作系统314为z/OS。

[0041] 当级别1系统管理程序302在逻辑分区104中运行时,其可将由级别0系统管理程序(诸如,系统管理程序108)提供至逻辑分区104的资源的相同虚拟化提供至在虚拟机308、306、304中运行的操作系统310、312、314。当在第一级别处时,每一虚拟机可包括多个虚拟处理器。

[0042] 物理核心106包括可为专用或如针对图1所描述在逻辑分区104LP1、LP2、LP3及LP4之间共享的物理处理器资源。当在一个或多个物理核心上分派逻辑分区LP2时,级别1系统管理程序302可接着在其虚拟机VM1 304、VM2 306及VM3 308之间透明地共享这些资源。在一个实施例中,级别0系统管理程序108使用指定MT的开始VE指令来分派多线程化级别1系统管理程序302,所述多线程化级别1系统管理程序接着使用指定ST的开始VE指令来分派单线程化虚拟机VM1 304、VM2 306及VM3 308。在一不同实施例中,级别0系统管理程序108使用指定ST的开始VE指令来分派单线程化级别1系统管理程序302,所述单线程化级别1系统管理程序接着使用指定MT的开始VE指令来分派多线程化虚拟机VM1 304、VM2 306及VM3 308。在另一实施例中,级别1系统管理程序302及其客体VM 304、306、308两者皆为单线程化的。

[0043] 在客体多处理(MP)环境中,系统管理程序可维护被称为系统控制区(SCA)的控制结构,其由系统管理程序及机器两者使用以管理可影响客体配置中的多个逻辑处理器的某些功能。针对配置或虚拟机中的所有客体处理器,在状态描述中指定相同SCA起点(SCAO)。在一实施例中,此区可包括公用区(一般用于协调全客体配置功能)及单独处理器特定的项。公用区(例如)保存关于客体配置内的哪些虚拟处理器有效的信息。SCA内的单独处理器特定的区可(例如)用于解释或仿真处理器间客体功能(诸如,处理器间中断)或容易地提供指向每一逻辑处理器的相应状态描述的可存取指针。在一实施例中,用于ST的SCA通过针对每一潜在客体线程添加额外线程特定的项而扩展以供MT使用。

[0044] 核心分派的一个实施例可允许在单个线程上运行的系统管理程序使用开始VE指令的变体(有时被称为开始多线程化虚拟执行(开始MVE))在其核心上分派多线程化客体。在多线程化客体中的每一线程可表示客体逻辑中央处理单元(CPU),或客体线程。开始VE指令能够经由状态描述中的控制字段启用在物理核心上的多线程(MT)客体执行。当用于核心分派时,开始VE指令的操作数可指定含有所有客体线程的状态的单个状态描述或各自(例如)表示单个客体线程的状态的一组状态描述。在一实施例中,逻辑核心包括该组状态描述。核心分派需要虚拟执行进入以将逻辑核心及这些客体逻辑线程中的每一者的状态加载至物理核心线程及其线程中。这些线程可为彼此独立地操作的指令流。在各种实施例中,可以若干方式指定状态描述组,包括指定为彼此的固定偏移、指定为状态描述地址或状态描述的列表、或指定为适用于核心的状态描述的循环列表(环),其中该组中的每一状态描述表示单独客体线程。此类技术允许系统管理程序及机器容易存取逻辑核心内的其他线程且允许适用于整个逻辑核心的字段被保持在单个位置中。

[0045] 客体OS可通过发出启用客体中的多线程的MT设定指令简单地利用多线程。这允许

客体OS将这些新线程视为额外独立CPU且如这些新线程在不存在多线程化的情况下来管理它们。另外，客体OS可以以利用这些线程共享核心的事实的方式使用这些线程，或客体OS可使这些线程以相互更相依方式操作。这对于系统管理程序及机器为全部透明的。系统管理程序接着将这些额外线程提供至客体OS，同时系统管理程序自身继续每核心在单个线程上运行且基于核心管理客体MT环境中的许多部分。在题为“Thread Context Preservation in a Multithreading Computer System”的美国专利申请14/226,895中更详细地描述多线程的OS启用。

[0046] 在核心分派的一个实施例中，指定为指定MT的开始VE指令的操作数的状态描述为“主要”状态描述且相关联的客体逻辑线程为“主要”线程。组中的其他状态描述在本文中被称为“次要”状态描述，且如果适用，则适用于次要逻辑线程。当状态描述组被实施为列表或环时，主要状态描述中可存在指向第一次要状态描述的下一状态描述(NSD)字段，第一次要状态描述继而1)指向组中的下一次要状态描述或2)含有指示组的结束的值。列表中的最后状态描述中的NSD值可为主要状态描述的地址，在此状况下，列表形成状态描述环。

[0047] 在非MT实现中，系统管理程序每次在给定物理核心上分派一个客体逻辑处理器(在本文中亦被称为“逻辑线程”)。如果特定逻辑处理器处于无效状态(例如，在停止状态下或在停用等待中)，则系统管理程序将不分派该客体。在MT环境中，核心分派允许系统管理程序同时在核心上分派多个客体线程。为了适应该逻辑核心的状态描述组中的线程中的一者或多者无效的可能性，一个实施例利用主要状态描述中的线程有效性屏蔽(TVM)，自软件观点，其中的每一位指示组中的对应状态描述中的逻辑线程的有效性。

[0048] 在另一实施例中，仅将有效线程包括在状态描述组中且有效性指示为不必要的。在状态描述组中包括无效逻辑线程的一个实施例允许系统管理程序维护与这些无效线程相关联的状态且这些线程可在将来再次变得有效。机器将仅初始化且运行具有有效状态的这些线程。如果组中的至少一个线程为有效的，则系统管理程序将仅分派一个客体逻辑核心。

[0049] 现转向图4，大体展示根据一个实施例的包括客体的架构状态的大部分的逻辑线程的状态描述。在此上下文中，术语“状态描述”不仅包括状态描述自身而且包括充当扩展的卫星块，其指针驻留于状态描述中。如图4中所展示，状态描述400可包括客体通用寄存器(GR)402、存取寄存器(AR)404、控制寄存器(CR)406、客体定时器408(包括时钟比较器及CPU定时器)、客体前缀寄存器410、虚拟CPU编号(VCN)412、程序状态字(PSW)及指令地址(IA)414。另外，状态描述可包括控制信息，诸如拦截控制(IC)位420，其用以指示某些指令(例如，加载程序状态字(LPSW)及失效页表项(IPTE))是否需要拦截主机或在客体指令执行可开始之前是否需要清空客体转换后备缓冲器(TLB)。状态描述亦含有用于定义如图6至图9中所描述的状态描述列表及环的下一状态描述(NSD)422。主要状态描述亦包括如图5中所描述的TVM 430及逻辑分区编号(LPN)432。虚拟CPU编号(VCN)412等效于CPU编号，可被调节以包括MT模式中的线程编号，如题为“Address Expansion and Contraction in a Multithreading Computer System”的美国专利申请14/226,947中所描述。

[0050] 核心内的线程可由二进制线程标识符(TID)识别。为简洁起见，在以下各图中，线程x常常通过术语TIDx来引用，所述术语在此状况下含义为“具有TID x的线程”。

[0051] 现参看图5，大体展示根据一个实施例的线程有效性屏蔽(TVM)520的方块图。如图

5中所展示，TVM 520的位0 530表示状态描述组中的逻辑线程0的有效性，位1 531表示线程1的有效性，位2 532表示线程2的有效性，位3 533表示线程3的有效性，等等，直至表示线程n(与此核心相关联的状态描述组中的最后可能逻辑线程)的有效性的位n 537。TVM可驻留于该组的主要状态描述中。

[0052] 现转向图6，大体展示根据一个实施例的固定偏移状态描述组结构。如图6中所展示，在彼此的固定偏移(N)处指定状态描述组。在此状况下，开始VE指令的操作数602指向逻辑线程0的主要状态描述603。逻辑线程x的次要状态描述605位于主要状态描述之后N个字节的固定偏移处，且逻辑线程y的次要状态描述607位于线程x的次要状态描述之后N个字节处。对于组中的所有线程，此情形继续。可以以若干方式确定组中的线程的数目，包括藉由主要状态描述中的计数或藉由在列表中的最后状态描述地址之后的结束标记。

[0053] 图6可表示两种状况，第一状况为组包括组中的所有逻辑线程(无论其是否有效)的状态描述的状况，且第二状况为仅有效状态描述被包括在组中的状况。在第一状况下，线程x的状态描述605表示线程1的状态，且线程y的状态描述607表示线程2的状态。仅在此第一状况下需要的TVM 620表示这些逻辑线程中的每一者的有效性。在第二状况下，线程x的状态描述605表示第一有效的逻辑次要线程的状态，且逻辑线程y的状态描述607表示第二有效的次要线程的状态。举例而言，如果线程1并不有效且线程2及3皆有效的，则线程x 605将表示线程2且线程y 607将表示线程3。将不存在包括于组中的线程1的状态描述，这是因为线程1为无效的。这相同的两种状况亦可适用于以下图7至图9中所示的实施例，然而仅描述及图示状况1。

[0054] 现转向图7，大体展示根据一个实施例的被指定为列表的状态描述组结构。在此状况下，开始VE指令的操作数702表示状态描述地址的列表，其中所述列表中的第一项704指向线程0的主要状态描述705，所述列表中的第二项706指向线程1的次要状态描述707，所述列表中的第三项708指向线程2的次要状态描述709，等等，针对组中的所有线程继续。TVM 720表示这些线程中的每一者的有效性。

[0055] 现转向图8，大体展示根据一个实施例的被指定为链接列表的状态描述组结构。在此状况下，如在图6中所描绘的状况下，开始VE指令的操作数802指向线程0的主要状态描述803，但实际上用于线程1的次要状态描述805的指针804被提供为主要状态描述中的下一状态描述(NSD)字段804。继而，用于线程2的次要状态描述807的指针806被提供为线程1的次要状态描述中的NSD 806。针对组中的所有线程，此情形将继续，其中最后线程n的状态描述809中的NSD 810被指定为0或指示列表结尾的某一其他独特值。在主要状态描述803中提供的TVM 820表示这些线程中的每一者的有效性。

[0056] 现转向图9，大体展示根据一个实施例的被指定为循环列表或环的状态描述组结构。此状况等同于图8中所展示的状况，因为开始VE指令的操作数902指向线程0的主要状态描述903，其含有用于线程1的次要状态描述905的NSD 904，所述次要状态描述含有用于线程2的次要状态描述907的NSD 906，且针对所有线程此情形继续直至最后线程n。然而，在图9中所展示的实施例中，在线程n的状态描述909中的NSD 910形成循环列表且指回至主要状态描述903。在主要状态描述903中提供的TVM 920表示这些线程中的每一者的有效性。

[0057] 核心分派允许系统管理程序管理在核心级别处的逻辑线程的许多方面。核心分派不仅常常通过将核心的多个线程的虚拟执行的协调推送至机器中而简化线程管理所需的

系统管理程序代码，而且其可减少管理配置中的更多处理器所需的开销。用于逻辑分区(或客体)的优先级管理可继续在逻辑核心级别处进行，从而减少此类管理上的按比例调整压力。系统管理程序自身仍需要管理与逻辑核心相关联的线程集合以确保在重新发出开始VE指令之前其需求(诸如，指令拦截)全部被满足。

[0058] 现参看图10，大体展示根据一个实施例的核心分派过程。如图10中所展示，系统管理程序正在物理核心N 1010及物理线程A 1020上单线程化地运行。在块1022中，系统管理程序发出指定MT的开始VE指令以分派多线程化客体核心。机器判定客体为多线程化的，且在块1024中，使物理线程B及C可用于运行软件。机器将客体状态自线程中的每一者的状态描述加载至对应物理线程中。在图10中所描绘的实施例中，机器使用多个物理线程以执行此功能，亦即，在物理线程A 1020上运行的毫码将客体逻辑线程X的状态加载至物理线程A中，如块1026中所展示。同样地，在物理线程B 1040及C 1060上运行的毫码将客体逻辑线程Y及Z的状态加载至物理线程B及C中，如块1046及1066中所展示。一旦客体状态已加载，在客体逻辑线程X、Y及Z上运行的软件便在物理线程A、B及C上执行，如块1028、1048及1068中所展示。

[0059] 现参看图11，大体展示根据一个实施例的自虚拟执行的协调退出。如图11中所展示，客体逻辑线程X、Y及Z正在物理线程A 1120、B 1140及C 1160上执行客体软件，如块1128、1148及1168中所指示。一个或多个客体线程判定需要自虚拟执行的退出。参看图11，在物理线程B 1140上运行的客体逻辑线程Y判定其必须退出虚拟执行(如块1150中所展示)，从而使机器用信号通知物理线程A 1120及C 1160退出虚拟执行，如块1152中所展示。在块1136、1154及1174中，在物理线程中的每一者上运行的毫码协调自虚拟执行的退出且接着使物理线程B 1140及C 1160不可用于由软件使用，如块1156及1176中所指示。物理线程A 1120上的毫码将主机状态重载至硬件中(如块1138中所展示)，这导致系统管理程序软件在物理线程A上的执行，如块1140中所展示。系统管理程序接着将按需处理任何待决客体拦截及主机中断。

[0060] 图12描绘根据一个实施例的用于包括多个逻辑核心的单个客体配置的核心导向式系统控制区(COSCA)的方块图。图12中所展示的COSCA可用于提供核心内的逻辑线程之间的协调及不同核心上的逻辑线程之间的协调两者。COSCA可包括具有指针的表示整个客体配置的公用区，每一逻辑核心一个指针以便分离核心描述区。每一核心描述包括表示该核心的公用区及用于该核心的一系列邻接的单独线程特定区或线程描述。在另一实施例中，核心描述提供线程描述的位置。所提供位置可为隐含的(例如，它们为包含于核心描述中的列表，或它们可在与核心描述相连的存储块中)。在其他实施例中，可提供指向含有线程描述的其他存储位置的指针。如本文中所使用，术语“指示位置”用于指确定项目(例如，线程描述或COSCA中的其他元素)的位置的这些方式中的任一者，或任何额外方式。此结构维护MT客体配置的树状表示，其促进(特定言之，在系统管理程序级别处)在一些情景下基于核心来管理事物，但其他情景下基于线程或处理器来管理事物。

[0061] 可在客体配置内的所有客体线程的状态描述中的SCA起点(SCAO)字段中提供相同COSCA起点(COSCAO)，且可针对给定核心内的所有线程提供相同核心描述区地址(CDAA)。此实施例的优点为其不需要一些系统管理程序可能难以提供的那样多的邻接实际存储空间。另一实施例可添加额外间接层且使每一核心描述包括用于每一线程特定区的指针的列表，

从而使含有这些区的控制块无需是邻接的。

[0062] 现参看图12,大体展示用于包括两个逻辑核心(每一核心中具有三个逻辑线程)的单个客体配置的COSCA的一个实例实施例。在一实施例中,COSCA包括COSCA公用区1260(在图12中展示为“COSCACA 1260”)、核心描述区1270及核心描述区1280的内容。将与逻辑核心0相关联的状态描述组的主要状态描述1203指定为由系统管理程序使用以分派客体核心0的开始VE指令的操作数1202。另外,将与逻辑核心1相关联的状态描述组的主要状态描述1233指定为用于分派核心1的开始VE指令的操作数1232。“核心0线程0”的此主要状态描述1203含有NSD01 1205,其指向核心0线程1的次要状态描述1213,所述次要状态描述继而含有指向组中的核心0线程2的最终次要状态描述1223的NSD02 1215。类似地,逻辑核心1的状态描述组以核心1线程0的主要状态描述1233开始,所述主要状态描述含有指向核心1线程1的次要状态描述1243的NSD11 1235,所述次要状态描述含有指向核心1线程2的最终次要状态描述1253的NSD12 1245。此客体配置中的所有六个线程的状态描述1203、1213、1223、1233、1243、1253在SCA0 1204、1214、1224、1234、1244、1254中包含相同值,其指向COSCA公用区1260。

[0063] 如图12中所展示的COSCA公用区1260含有用以协调全客体配置功能的核心级别信息。COSCA公用区1260包括指示客体配置内的每一逻辑核心的有效性的SCA核心有效性屏蔽(SCVM)1261,且亦包括每一核心的核心描述区地址(CDAA)1262、1264。SCVM中的位及核心描述地址的阵列两者可通过核心编号来编制索引。指向核心0的核心描述区(CDA)1270的CDAA0 1262包括于COSCA公用区1260中。另外,核心0中的所有线程的状态描述中的CDAA字段1206、1216、1226亦指向核心0的CDA 1270。指向核心1的CDA 1280的CDAA1 1264亦包括于COSCA公用区1260中,且同样地,核心1中的所有线程的状态描述中的CDAA字段1236、1246、1256亦指向核心1的CDA 1280。核心0的核心描述区(CDA)1270含有指示核心0内的每一逻辑线程的有效性的SCA线程有效性屏蔽(STVM0)1271。其亦含有核心0线程0的线程描述区1272、核心0线程1的线程描述区1274及核心0线程2的线程描述区1276。核心1的CDA1280类似地含有STVM1 1281以及核心1线程0的线程描述区1282、核心1线程1的线程描述区1284及核心1线程2的线程描述区1286。这些线程描述区1272、1274、1276、1282、1284、1286中的每一者含有用于对应于该线程描述区的线程(分别为核心0线程0、核心0线程1、核心0线程2、核心1线程0、核心1线程1及核心1线程2)的状态描述地址(SDA)1273、1275、1277、1283、1285、1287。STVM中的位及线程描述区的阵列两者可通过线程标识来编制索引。这些SDA使系统管理程序更容易管理核心内的线程且使机器更容易呈现客体处理器间中断。

[0064] 图13描绘根据一个实施例的用于管理多线程化核心的过程流,其使用图12中所展示的COSCA。在图13中所展示的实例中,在块1302处,在第一物理线程(例如,由状态描述1213定义的核心0线程1)上运行的客体操作系统(OS)已确定其将用信号通知第二逻辑线程或目标线程(例如,由状态描述1253定义的核心1线程2)。在块1304处,客体OS(例如)通过发出处理器间中断指令执行此操作。作为执行处理器间中断指令的一部分,机器使用COSCA来仿真客体处理器间中断指令。因为包括目标逻辑线程的逻辑核心在发信号进行时可能或可能未被分派,所以由机器仿真处理器间中断指令。在块1306处,机器定位(例如,经由SCA0 1214,这是因为处理器间中断指令由逻辑核心0线程1执行)用于客体配置的公用区(例如,COSCA公用区1260),以便存取SCVM(例如,SCVM 1261)以检验目标核心的有效性且获得适当

CDAA(例如,CDAA1 1264,这是因为目标线程系在核心1上)。

[0065] 接下来,在块1308处,机器定位(例如,经由CDA1 1264)用于目标核心的核心描述区(例如,CDA 1280)。机器通过存取核心描述区中的STVM(例如,CDA 1280中的STVM1 1281)而检验目标线程为有效的。在块1310处,机器定位线程描述区(例如,对应于线程2的线程描述区1286,这是因为目标线程为线程2)。在块1312处,将关于中断的信息记录在用于目标线程的线程描述区中(例如,其将发送线程的身份放置至线程描述区1286中)。在块1314处,机器定位(例如,经由线程描述区1286中的SDA12 1287)目标线程的状态描述(例如,核心1TID2的次要状态描述1253)。在块1316处,使中断在目标状态描述中待决(例如,在核心1TID2的状态描述1253中设定IP位1257)。因此,当目标逻辑处理器(例如,核心1线程2)被分派在物理线程上且被启用以用于中断时,机器将向客体操作系统呈现中断(在启用情况下)。如果目标逻辑处理器已在中断变为待决时被分派,则一旦其被启用,其便将采取中断。

[0066] 存在机器亦可利用逻辑核心内的线程具有共同属性的事实的情况。举例而言,核心分派自然地适于使逻辑核心上的所有客体线程驻留在同一LPAR区或分区中。该设计可通过仅必须每核心一个地而非每线程一个地实施与该分区相关联的事物来最小化硬件。另外,亦可简化复杂的控制逻辑(例如,全系统中断的处置),这是因为其必须仅处理单个核心值。

[0067] 在一个实施例中,表示多线程化客体的状态描述的组中的每一字段(或字段内的位)被分类为主要、核心公用或线程特定。主要字段仅驻留在主要状态描述中且适用于逻辑核心中的所有处理器;代表核心的任何线程对主要字段的任何存取必须使用来自相关联主要状态描述的值。此分类用于定义核心的总状态(诸如,线程有效性屏蔽)的字段。核心公用字段在逻辑核心内的所有处理器当中是公用的且此字段在组中的每一状态描述中具有相同值;代表处理器对这些字段中的一者的任何存取可使用来自该组中的任何状态描述的值。此分类用于在整个核心上应用的字段,诸如LP编号。需要系统管理程序维护在所有状态描述中的核心公用字段,但允许机器存取任何线程的状态描述中的此字段,无论哪个提供最佳性能。因为这些字段并不由系统管理程序常常改变但常常由机器每次进入虚拟执行时存取,所以将字段定义为核心公用而非线程特定的允许虚拟执行进入(例如)以使用主要状态描述中的值自主要线程加载次要线程工具。线程特定字段对每一逻辑线程为特定的;代表任何给定线程对这些字段中的一者的任何存取必须使用来自该线程的状态描述的值。此分类用于在线程之间通常是独特的字段,诸如客体前缀。

[0068] 在一实施例中,为了支持核心分派的使用及单线程化地运行的系统管理程序,可提供自虚拟执行的协调退出(VE退出),其中给定核心中的所有客体线程同时退回至ST主机。在协调VE退出的情形中,VE退出的类型可划分成三种类别:(1)与主机操作有关的主机中断;(2)与客体操作有关的主机中断;及(3)客体拦截。主机外部I/O及一些机器检查中断属于VE退出类别(1)。对于此状况,需要所有客体线程退出虚拟执行模式,以便允许主机处置中断。此中断将可能引起主机分派不同客体。如果在以虚拟执行模式运行时发生中断,则可在所有线程上检测到主机中断,使得所有线程可退出虚拟执行模式,或可在单个线程上检测到主机中断,所述单个线程接着用信号通知其他线程它们是否应退出。

[0069] VE退出类别(2)(与客体有关的主机中断)可包括一些机器检查中断(诸如,不可纠正的存储错误)。在非多线程化情景下,这些条件作为主机中断来呈现。在核心分派情况下,

仅存在一个主机线程，但因为这些异常与客体操作有关，所以多个客体线程针对同一主机中断检测到相异及不同的原因是可能的。为适应此情形，对于核心分派，在适当时，将这些主机中断替代地呈现于对应客体状态描述中作为新类型的客体拦截且与下文所描述的类别(3)相同地处置。在一实施例中，归因于客体存储器引用而产生的主机地址转换错误中断亦属于类别(2)，且可呈现为另一新类型的客体拦截。

[0070] 对于VE退出类别(2)及(3)两者(上述)，客体拦截(甚至在客体多线程化环境中)与单个客体线程有关且独立于另一线程的客体执行。进一步可能的，多个客体线程同时认识到此类条件，需要主机处置它们中的全部。通常，当呈现有拦截时，主机将仿真代表客体的一些行为且接着重新分派该同一客体。对于这些状况，因为主机正单个线程化地运行，所以所有客体线程必须在主机可处置拦截(多个)之前退出虚拟执行模式。这可通过等待所有线程自然地退出或通过在一个线程已判定其必须拦截回至主机时用信号通知其他线程退出而实现。此被称为“协调VE退出”。

[0071] 当每一线程判定其必须退出虚拟执行模式时，其进入VE退出，且在初始VE退出同步循环中等待，直至所有其他有效线程亦准备退出。如果实施需要，则其用信号通知其他线程在进入此同步循环之前退出。当在VE退出同步循环中时，仅处置最少中断。为了考虑在无主机中断且无客体拦截应用时需要客体线程退出虚拟执行模式的情景，定义“无动作”拦截以向主机指示不需要代表此客体的拦截动作。

[0072] 一旦所有线程已进入初始VE退出同步循环，客体数据在所有有效状态描述中的存储便可完成。亦即，驻留于硬件中的当前客体状态被保存在对应状态描述中，此逻辑客体线程因此可在稍后被重新分派。在此存储完成之后需要最终VE退出同步点以保证对次要线程状态描述的所有更新在控制返回至系统管理程序(其通常在主要线程上运行)之前完成。一旦VE退出完成，系统管理程序便可处理环中的每一线程以判定是否呈现拦截，且如果呈现，则适当地处置拦截。在如此进行之后，其可接着在物理处理器上重新分派此同一客体逻辑核心或不同客体逻辑核心。

[0073] 技术效果及益处包括在多线程(MT)环境中提供减小的系统管理程序管理开销。可在将多个线程管理为单个逻辑核心的系统管理程序与随着多个线程存取物理核心的资源而管理多个线程之间的交互的机器之间划分多个线程的管理，从而通过允许系统管理程序基于逻辑核心来管理许多系统管理程序基础结构资源且允许机器基于更细微的线程来管理其他资源而导致显著减小的多线程(MT)开销成本。

[0074] 各实施例包括一种提供用于在计算机中分派多个线程的开始虚拟执行指令的系统、方法及计算机程序产品。根据一个方面，一种计算机系统包括配置，所述配置具有被启用以在单线程(ST)模式及多线程(MT)模式中操作的核心。所述核心包括物理线程。所述计算机系统亦包括主机程序，所述主机程序被配置为在所述核心上以所述ST模式执行以发出开始虚拟执行(开始VE)指令以在所述核心上分派客体实体。所述客体实体包括客体虚拟机(VM)的全部或一部分。所述开始VE指令由所述核心执行。所述执行包括从由所述开始VE指令指定的位置获得第一状态描述。所述第一状态描述具有客体状态。所述执行亦包括基于所述客体状态来判定所述客体实体包括单个客体线程还是多个客体线程。基于所述客体状态且基于判定所述客体实体包括多个客体线程，在所述核心上以所述MT模式开始所述客体线程。所述客体线程彼此独立地执行。基于所述客体状态且基于判定所述客体实体包括单

个客体线程，在所述核心上以所述ST模式开始所述客体线程。

[0075] 根据另一方面，提供一种用于分派配置中的多个线程的计算机实施的方法。所述配置包括被启用以在单线程(ST)模式及多线程(MT)模式中操作的核心。所述核心包括物理线程。所述方法包括由在所述核心上以所述ST模式执行的主机程序发出开始虚拟执行(开始VE)指令以在所述核心上分派客体实体。所述客体实体包括客体VM的全部或一部分，且所述开始VE指令由所述核心执行。所述执行包括从由所述开始VE指令指定的位置获得第一状态描述。所述第一状态描述具有客体状态。所述执行亦包括基于所述客体状态来判定所述客体实体包括单个客体线程还是多个客体线程。基于所述客体状态且基于判定所述客体实体包括多个客体线程，所述执行亦包括在所述核心上以所述MT模式开始所述客体线程，其中所述客体线程彼此独立地执行。基于所述客体状态且基于判定所述客体实体包括单个客体线程，所述执行包括在所述核心上以所述ST模式开始所述客体线程。

[0076] 另一方面包括一种用于分派配置中的多个线程的计算机程序产品。所述配置包括被启用以在单线程(ST)模式及多线程(MT)模式中操作的核心。所述核心包括物理线程。所述计算机程序产品包括具有程序指令的计算机可读存储介质，其中所述计算机可读存储介质并非信号，所述程序指令可由处理电路读取以使所述处理电路执行一种方法。所述方法包括由在所述核心上以所述ST模式执行的主机程序发出开始虚拟执行(开始VE)指令以在所述核心上分派客体实体。所述客体实体包括客体VM的全部或一部分，且所述开始VE指令由所述核心执行。所述执行包括从由所述开始VE指令指定的位置获得第一状态描述。所述第一状态描述具有客体状态。所述执行亦包括基于所述客体状态来判定所述客体实体包括单个客体线程还是多个客体线程。基于所述客体状态且基于判定所述客体实体包括多个客体线程，所述执行包括在所述核心上以所述MT模式开始所述客体线程，其中所述客体线程彼此独立地执行。基于所述客体状态且基于判定所述客体实体包括单个客体线程，所述执行包括在所述核心上以所述ST模式开始所述客体线程。

[0077] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括其中所述核心包括当所述核心处于所述MT模式中时控制所述物理线程之间的共享资源的使用的计算机指令。

[0078] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括其中当所述核心处于所述MT模式中时，所述主机程序将所述客体实体的至少一部分管理为单个逻辑核心。

[0079] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括其中由所述主机程序利用线程有效性屏蔽以指示所述客体实体中的所述一个或多个客体线程的有效性。

[0080] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括在将控制返回至所述主机程序之前退出所述客体实体中的所述一个或多个客体线程中的全部客体线程。

[0081] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括其中基于所述客体实体包括多个客体线程，将全部所述客体线程所共有的状态数据的至少一部分存储在单个位置中。

[0082] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括

其中基于所述客体实体包括多个客体线程，在第一状态描述中含有用于一个线程的状态数据，且在额外状态描述中含有用于一个或多个额外线程中的每一者的状态数据。

[0083] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括其中所述第一状态描述及所述额外状态描述被存储在环及列表结构中的至少一者中。

[0084] 除上文所描述的特征中的一者或多者之外，或作为替代例，另外实施例亦可包括执行无动作退出，所述无动作退出包括基于来自其他客体线程的请求而退出客体线程。

[0085] 本文中所用的术语，仅仅是为了描述特定的实施例，而不意图限定本发明。本文中所用的单数形式的“一”和“该”，旨在也包括复数形式，除非上下文中明确地另行指出。还要知道，“包含”和/或“包括”在本说明书中使用时，说明存在所指出的特征、整体、步骤、操作、单元和/或组件，但是并不排除存在或增加一个或多个其它特征、整体、步骤、操作、单元和/或组件，以及/或者它们的组合。

[0086] 以下的权利要求中的对应结构、材料、操作以及所有功能性限定的装置(means)或步骤的等同替换，旨在包括任何用于与在权利要求中具体指出的其它单元相组合地执行该功能的结构、材料或操作。所给出的对一个或多个实施例的描述其目的在于示意和描述，并非是穷尽性的，也并非是要将本发明限定到所表述的形式。对于所属技术领域的普通技术人员来说，显然可以作出许多修改和变型而不偏离本发明的精神和范围。对实施例的选择和描述，是为了最好地解释本发明的原理和实际应用，使所属技术领域的普通技术人员能够明了，本发明可以有适合所要的特定用途的具有各种改变的各种实施例。

[0087] 本发明的各种实施例的描述已为达成说明的目的而呈现，但不意欲为穷尽性的或限于所揭示的实施例。在不脱离所描述实施例的范围及精神的情况下，许多修改及变化对于本领域技术人员将显而易见。本文中所使用的术语被选择以最好地解释实施例的原理、实际应用或对市场中找到的技术的技术改进，或使其他本领域技术人员能够理解本文所揭示的实施例。

[0088] 现参看图14，在一个实例中，计算机程序产品1400包括(例如)一个或多个存储介质1402以在其上存储计算机可读程序代码装置或逻辑1404从而提供及促进本文中所描述的实施例的一个或多个方面，其中所述介质可为有形的及/或非暂时性的。

[0089] 本发明可以是系统、方法和/或计算机程序产品。计算机程序产品可以包括计算机可读存储介质，其上载有用于使处理器实现本发明的各个方面的计算机可读程序指令。

[0090] 计算机可读存储介质可以是可以保持和存储由指令执行设备使用的指令的有形设备。计算机可读存储介质例如可以是但不限于电存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备或者上述的任意合适的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括：便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式压缩盘只读存储器(CD-ROM)、数字多功能盘(DVD)、记忆棒、软盘、机械编码设备、例如其上存储有指令的打孔卡或凹槽内凸起结构、以及上述的任意合适的组合。这里所使用的计算机可读存储介质不被解释为瞬时信号本身，诸如无线电波或者其他自由传播的电磁波、通过波导或其他传输媒介传播的电磁波(例如，通过光纤电缆的光脉冲)、或者通过电线传输的电信号。

[0091] 这里所描述的计算机可读程序指令可以从计算机可读存储介质下载到各个计算/

处理设备，或者通过网络、例如因特网、局域网、广域网和/或无线网下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光纤传输、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配卡或者网络接口从网络接收计算机可读程序指令，并转发该计算机可读程序指令，以供存储在各个计算/处理设备中的计算机可读存储介质中。

[0092] 用于执行本发明操作的计算机程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或者以一种或多种编程语言的任意组合编写的源代码或目标代码，所述编程语言包括面向对象的编程语言—诸如Smalltalk、C++等，以及常规的过程式编程语言—诸如“C”语言或类似的编程语言。计算机可读程序指令可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络—包括局域网(LAN)或广域网(WAN)—连接到用户计算机，或者，可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。在一些实施例中，通过利用计算机可读程序指令的状态信息来个性化定制电子电路，例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA)，该电子电路可以执行计算机可读程序指令，从而实现本发明的各个方面。

[0093] 这里参照根据本发明实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述了本发明的各个方面。应当理解，流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合，都可以由计算机可读程序指令实现。

[0094] 这些计算机可读程序指令可以提供给通用计算机、专用计算机或其它可编程数据处理装置的处理器，从而生产出一种机器，使得这些指令在通过计算机或其它可编程数据处理装置的处理器执行时，产生了实现流程图和/或框图中的一个或多个方框中规定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储介质中，这些指令使得计算机、可编程数据处理装置和/或其他设备以特定方式工作，从而，存储有指令的计算机可读介质则包括一个制造品，其包括实现流程图和/或框图中的一个或多个方框中规定的功能/动作的各个方面的指令。

[0095] 也可以把计算机可读程序指令加载到计算机、其它可编程数据处理装置、或其它设备上，使得在计算机、其它可编程数据处理装置或其它设备上执行一系列操作步骤，以产生计算机实现的过程，从而使得在计算机、其它可编程数据处理装置、或其它设备上执行的指令实现流程图和/或框图中的一个或多个方框中规定的功能/动作。

[0096] 附图中的流程图和框图显示了根据本发明的多个实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分，所述模块、程序段或指令的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

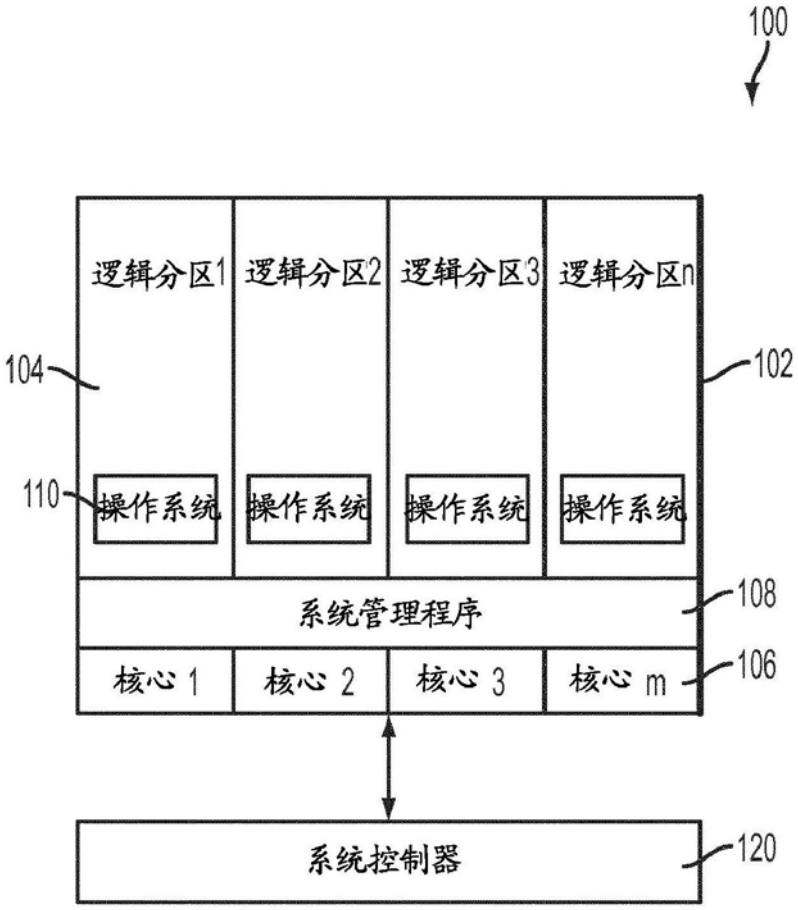

100
逻辑分区1
逻辑分区2
逻辑分区3
逻辑分区n
104
102
110
操作系统
操作系统
操作系统
操作系统
系统管理程序
108
核心 1
核心 2
核心 3
核心 m
106
系统控制器
120

图1

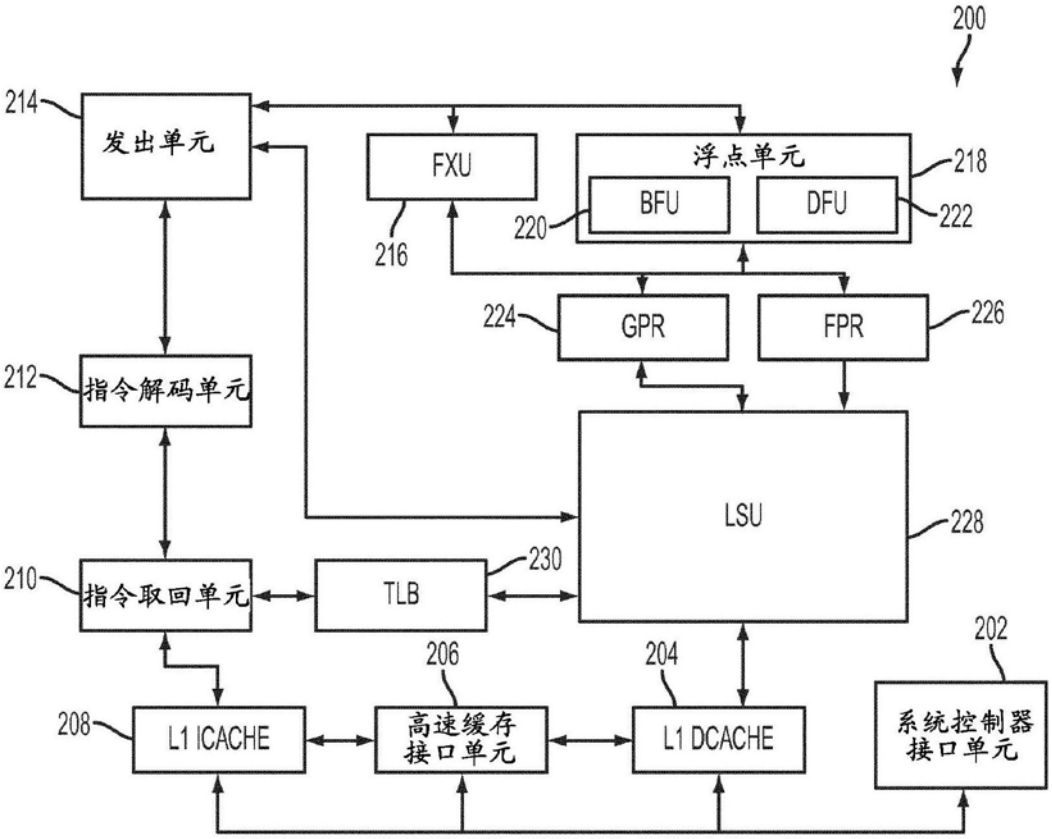

200
214
发出单元
FXU
浮点单元
218
BFU
DFU
220
222
216
224
GPR
FPR
226
212
指令解码单元
LSU
228
210
指令取回单元
TLB
230
206
204
202
208
L1 ICACHE
高速缓存
接口单元
L1 DCACHE
系统控制器
接口单元

图2

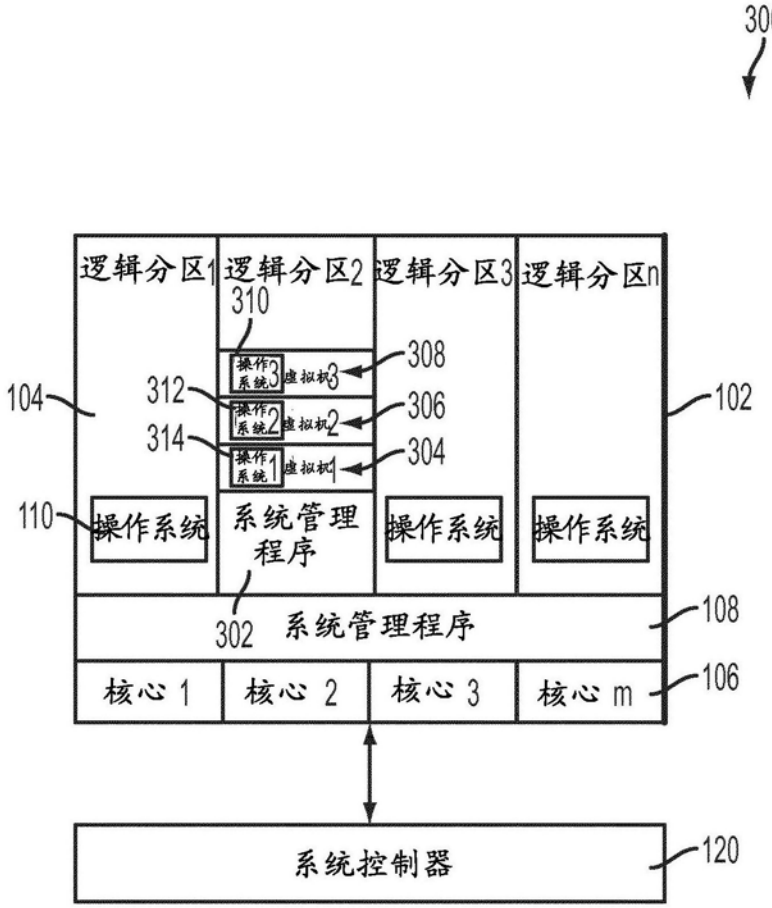

300
逻辑分区1
逻辑分区2
逻辑分区3
逻辑分区n
310
308
312
306
314
304
104
102
操作系统3
虚拟机3
操作系统2
虚拟机2
操作系统1
虚拟机1
110
操作系统
系统管理程序
操作系统
操作系统
302
系统管理程序
108
核心 1
核心 2
核心 3
核心 m
106
系统控制器
120

图3

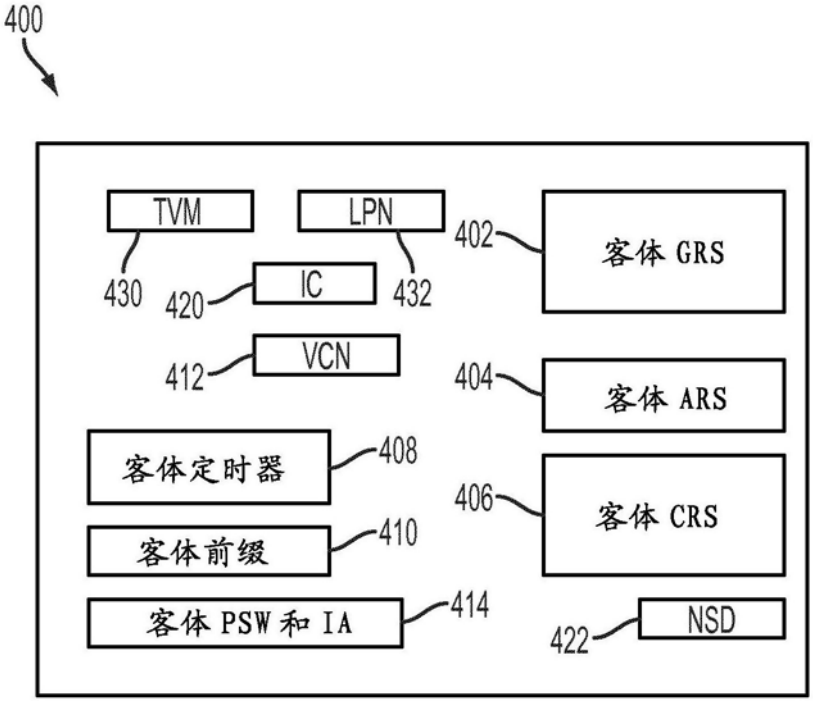

400
TVM
LPN
402
客体 GRS
430
420
IC
432
412
VCN
404
客体 ARS
客体定时器
408
406
客体 CRS
客体前缀
410
客体 PSW 和 IA
414
422
NSD

图4

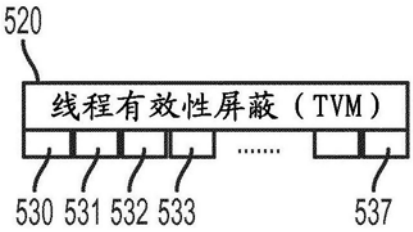

520
线程有效性屏蔽（TVM）
530 531 532 533
537

图5

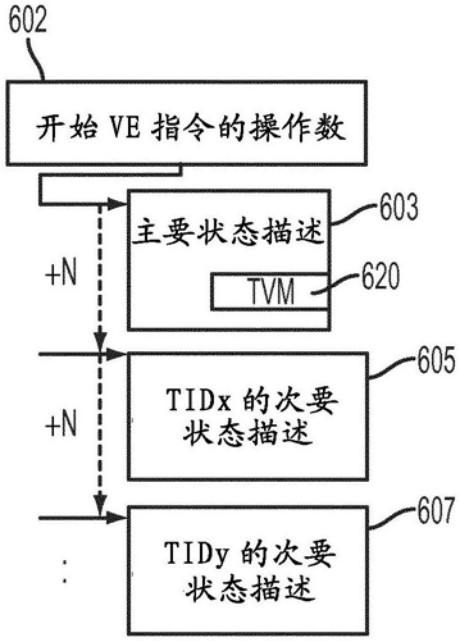

602
开始 VE 指令的操作数
主要状态描述
603
+N
TVM
620
605
TIDx 的次要状态描述
+N
607
TIDy 的次要状态描述

图6

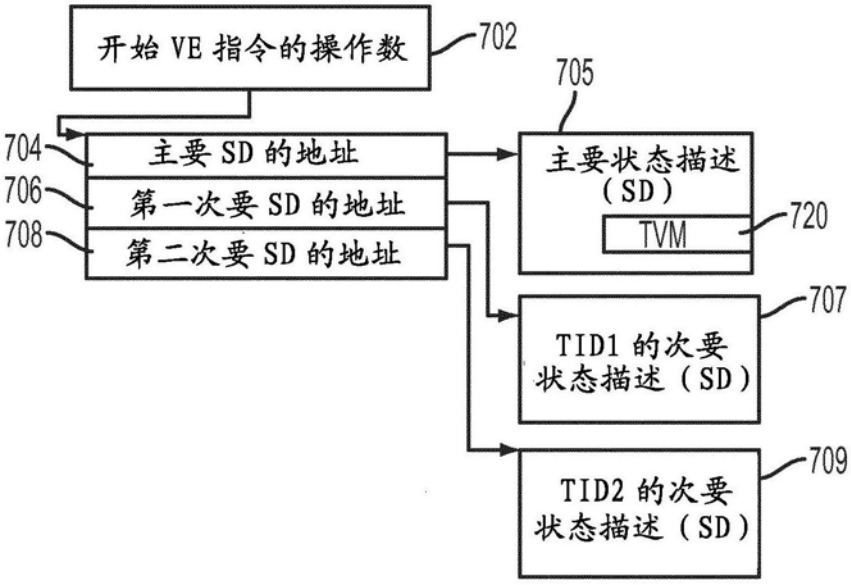

开始 VE 指令的操作数
702
705
704
主要 SD 的地址
706
第一次要 SD 的地址
708
第二次要 SD 的地址
主要状态描述（SD）
TVM
720
707
TID1 的次要状态描述（SD）
709
TID2 的次要状态描述（SD）

图7

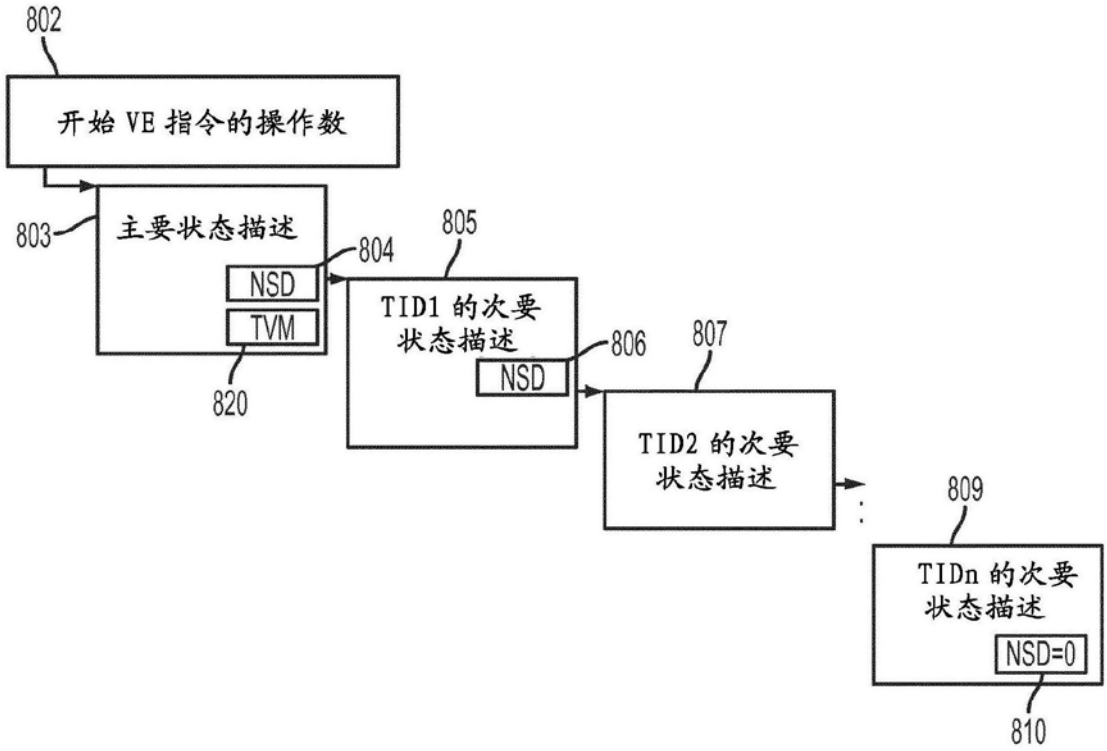

802
开始 VE 指令的操作数
803
主要状态描述
NSD
TVM
804
805
TID1 的次要状态描述
NSD
806
807
TID2 的次要状态描述
809
TIDn 的次要状态描述
NSD=0
810
820

图8

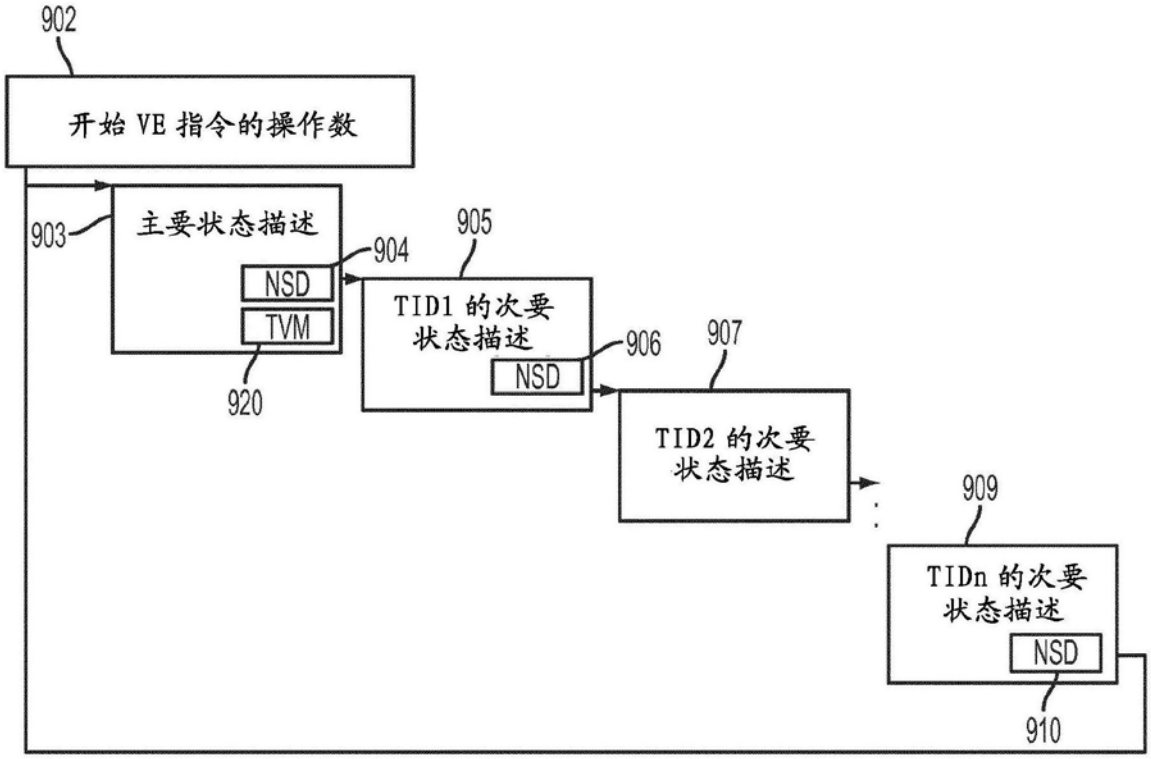

902
开始 VE 指令的操作数
903
主要状态描述
NSD
TVM
904
905
TID1 的次要状态描述
NSD
906
907
TID2 的次要状态描述
909
TIDn 的次要状态描述
NSD
910
920

图9

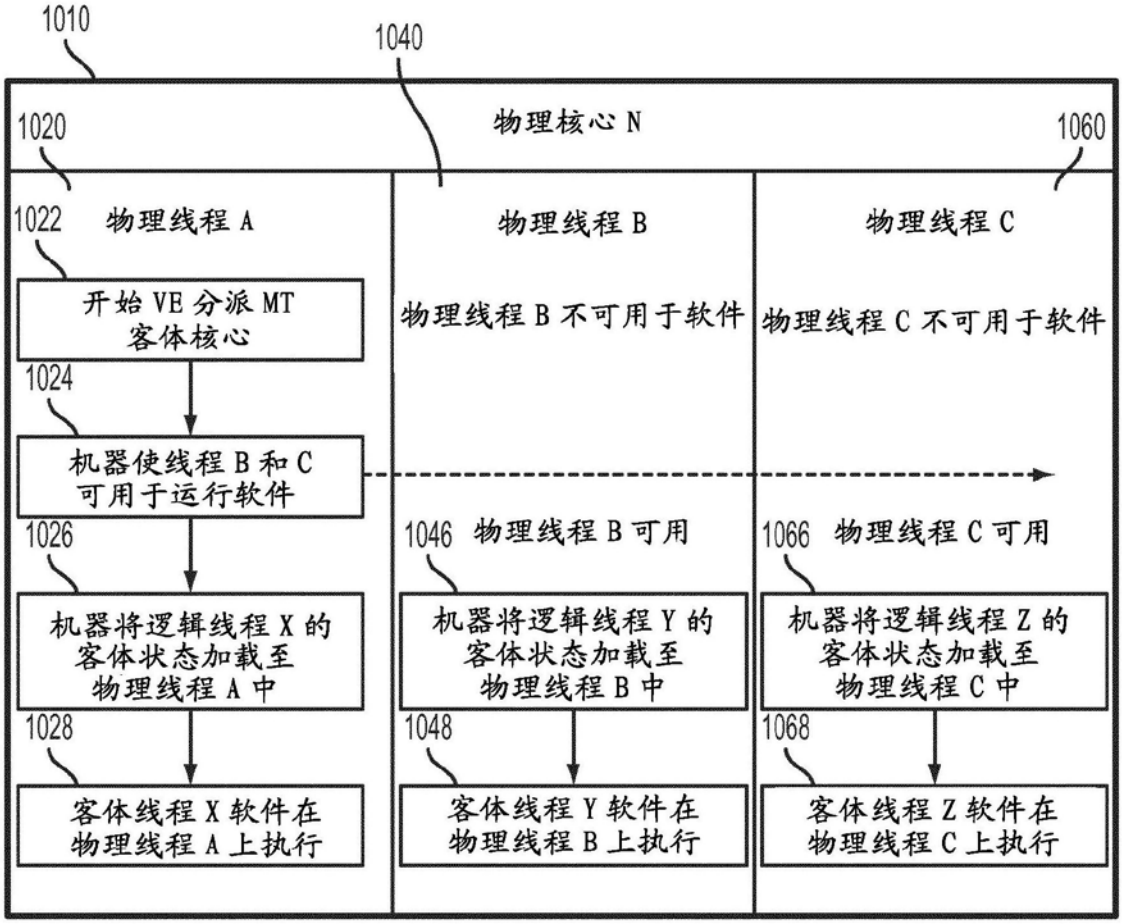

1010
1040
1020
物理核心 N
1060
1022
物理线程 A
物理线程 B
物理线程 C
开始 VE 分派 MT
客体核心
物理线程 B 不可用于软件
物理线程 C 不可用于软件
1024
机器使线程 B 和 C
可用于运行软件
1026
1046
物理线程 B 可用
1066
物理线程 C 可用
机器将逻辑线程 X 的
客体状态加载至
物理线程 A 中
机器将逻辑线程 Y 的
客体状态加载至
物理线程 B 中
机器将逻辑线程 Z 的
客体状态加载至
物理线程 C 中
1028
1048
1068
客体线程 X 软件在
物理线程 A 上执行
客体线程 Y 软件在
物理线程 B 上执行
客体线程 Z 软件在
物理线程 C 上执行

图10

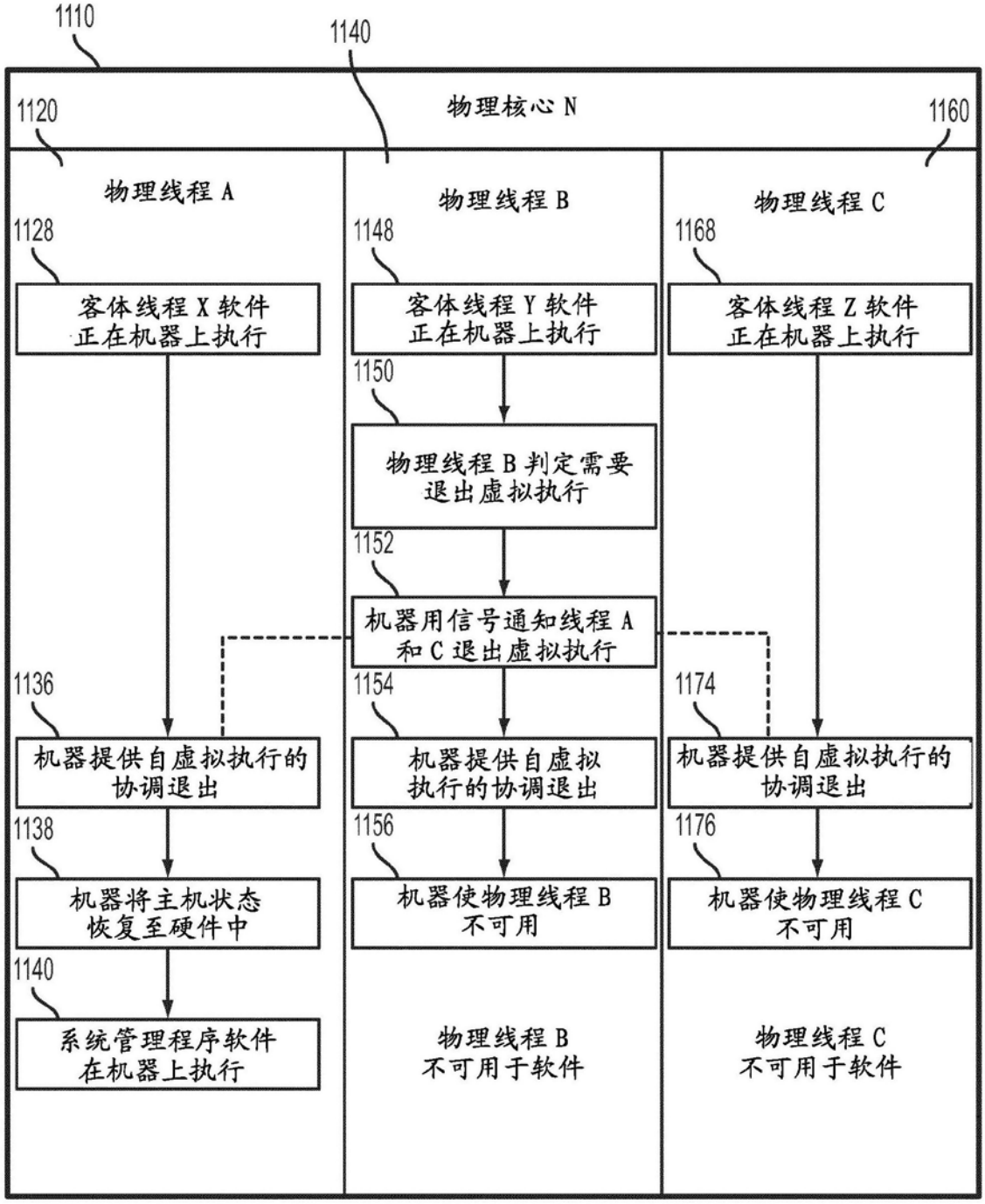

1110
1140
1120
物理核心 N
1160
物理线程 A
物理线程 B
物理线程 C
1128
客体线程 X 软件
正在机器上执行
1148
客体线程 Y 软件
正在机器上执行
1168
客体线程 Z 软件
正在机器上执行
1150
物理线程 B 判定需要
退出虚拟执行
1152
机器用信号通知线程 A
和 C 退出虚拟执行
1136
机器提供自虚拟执行的
协调退出
1154
机器提供自虚拟
执行的协调退出
1174
机器提供自虚拟执行的
协调退出
1138
机器将主机状态
恢复至硬件中
1156
机器使物理线程 B
不可用
1176
机器使物理线程 C
不可用
1140
系统管理程序软件
在机器上执行
物理线程 B
不可用于软件
物理线程 C
不可用于软件

图11

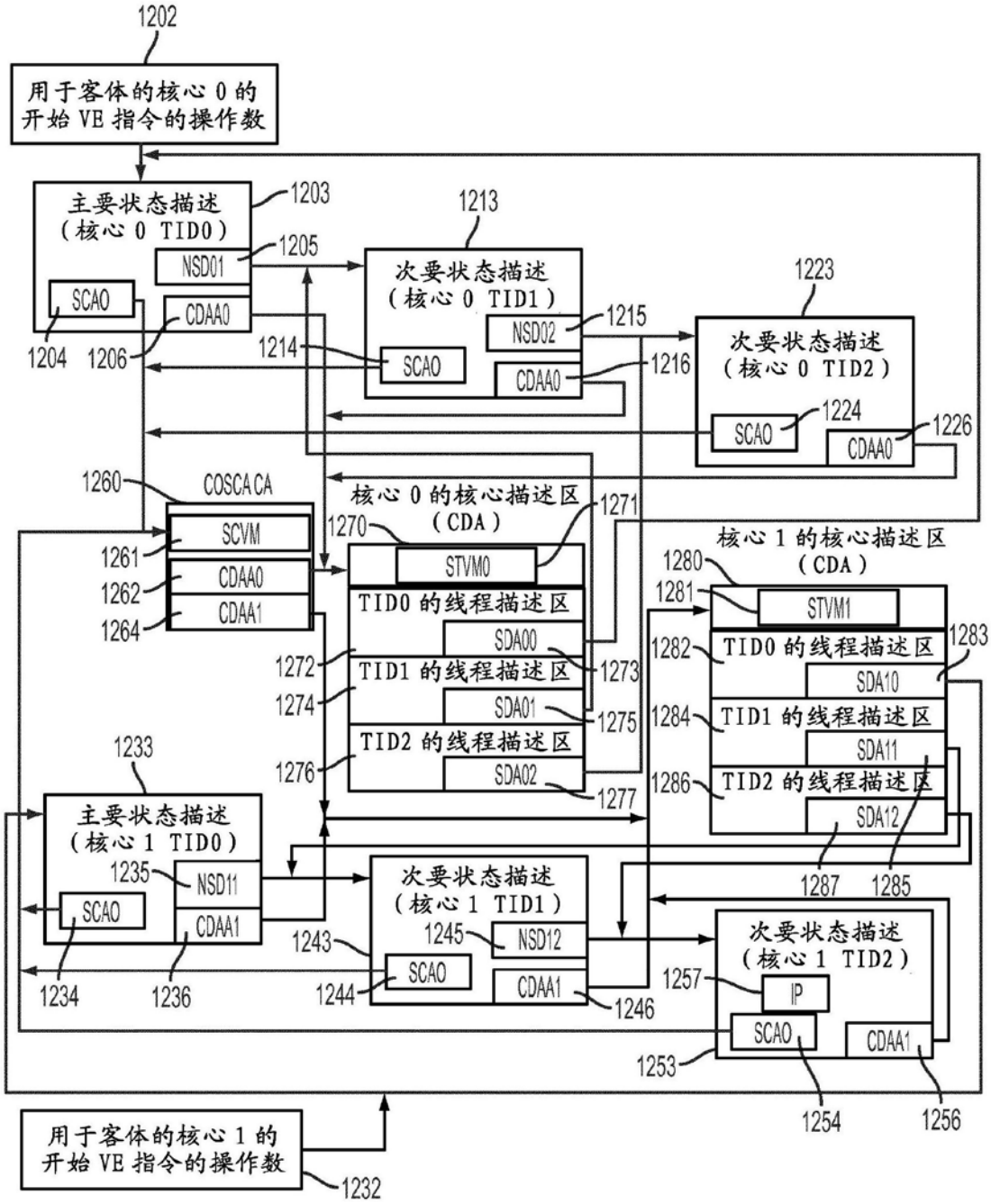

1202
用于客体的核心 0 的开始 VE 指令的操作数
主要状态描述（核心 0 TID0）
1203
NSD01
1205
SCAO
CDAA0
1204
1206
次要状态描述（核心 0 TID1）
1213
NSD02
1215
SCAO
CDAA0
1214
1216
次要状态描述（核心 0 TID2）
1223
SCAO
1224
CDAA0
1226
1260
COSCACA
SCVM
1261
CDAA0
1262
CDAA1
1264
核心 0 的核心描述区（CDA）
1270
1271
STVM0
TID0 的线程描述区
SDA00
1272
1273
TID1 的线程描述区
1274
SDA01
1275
TID2 的线程描述区
1276
SDA02
1277
核心 1 的核心描述区（CDA）
1280
1281
STVM1
1282
TID0 的线程描述区
1283
SDA10
1284
TID1 的线程描述区
SDA11
1286
TID2 的线程描述区
SDA12
1287
1285
1233
主要状态描述（核心 1 TID0）
1235
NSD11
SCAO
CDAA1
1234
1236
次要状态描述（核心 1 TID1）
1243
1245
NSD12
SCAO
CDAA1
1244
1246
次要状态描述（核心 1 TID2）
1257
IP
SCAO
CDAA1
1253
1254
1256
用于客体的核心 1 的开始 VE 指令的操作数
1232

图12

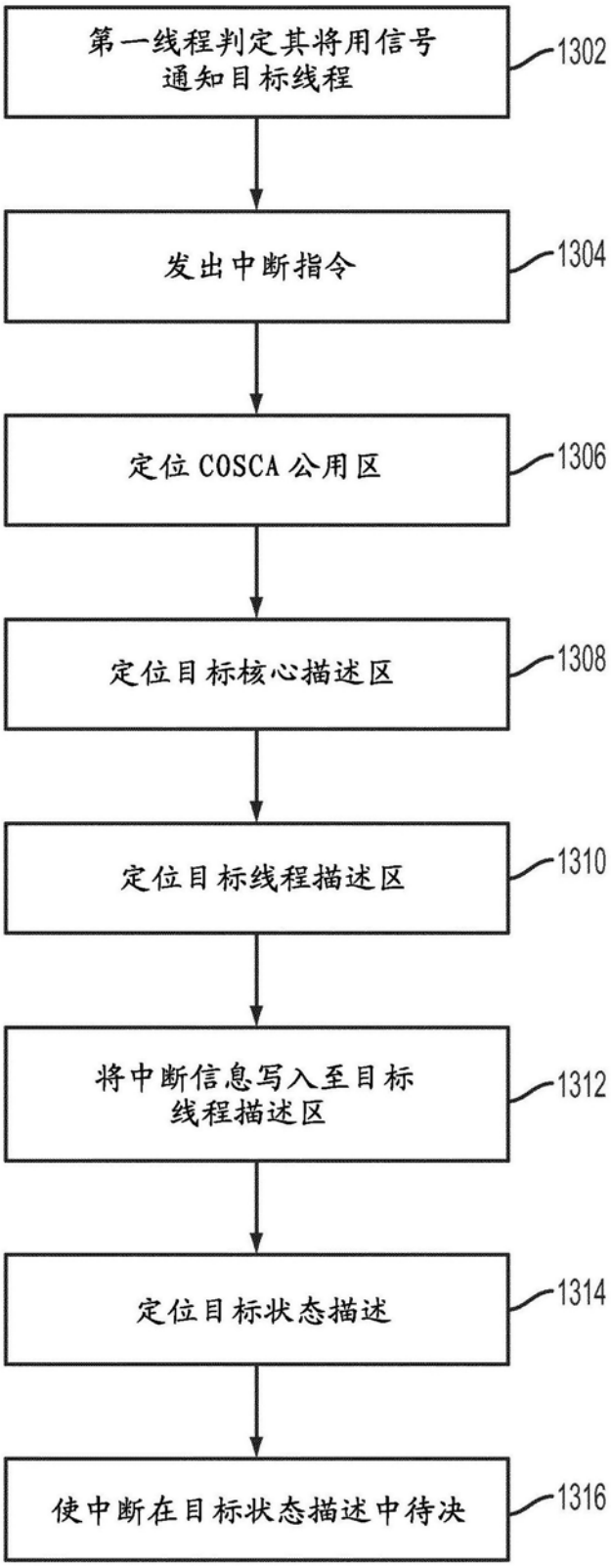

第一线程判定其将用信号通知目标线程
1302
发出中断指令
1304
定位 COSCA 公用区
1306
定位目标核心描述区
1308
定位目标线程描述区
1310
将中断信息写入至目标线程描述区
1312
定位目标状态描述
1314
使中断在目标状态描述中待决
1316

图13

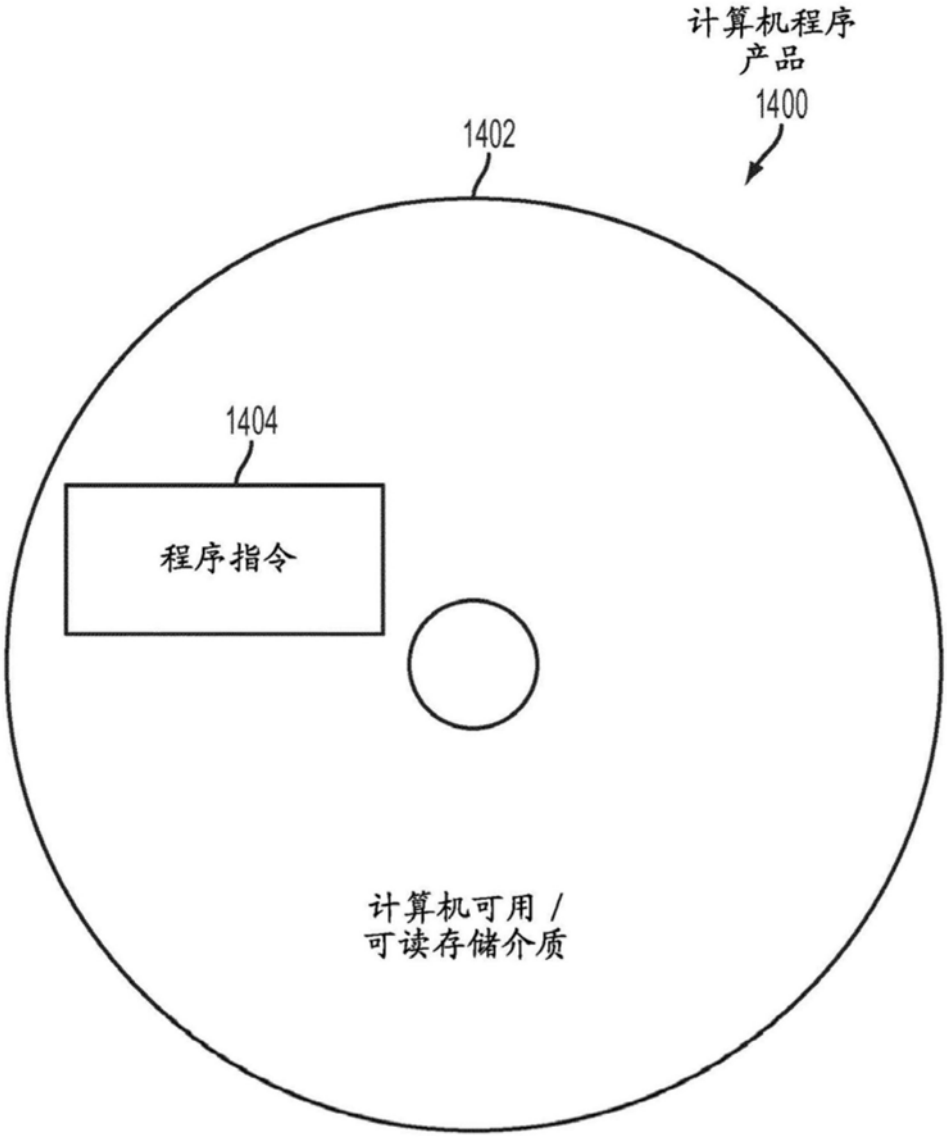


图14