[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)

[12] 发明专利说明书

专利号 ZL 03822194.2

[45] 授权公告日 2007 年 12 月 19 日

[11] 授权公告号 CN 100356328C

[22] 申请日 2003.8.11 [21] 申请号 03822194.2
[30] 优先权
[32] 2002. 9.19 [33] US [31] 10/246,889
[86] 国际申请 PCT/GB2003/003497 2003.8.11
[87] 国际公布 WO2004/027599 英 2004.4.1
[85] 进入国家阶段日期 2005.3.18
[73] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 A·门多萨 J·H·朔普
[56] 参考文献
WO 2001050247A 2001.7.12
审查员 王 丹

[74] 专利代理机构 北京市中咨律师事务所
代理人 于 静 李 峥

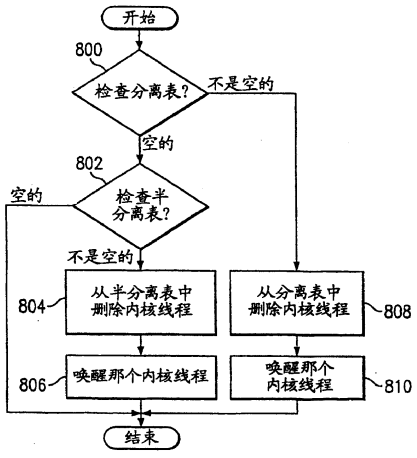
权利要求书 2 页 说明书 11 页 附图 6 页

[54] 发明名称

用于处理数据处理系统中的线程的方法和装置

[57] 摘要

本发明披露了一种用于管理线程的方法、设备和计算机指令。 与一个用户线程相关联的内核线程被检测为不被该用户线程需要。 该内核线程被半分离，其中响应该内核线程是不被需要的，用于该线程的数据不改变堆栈。



1. 一种用于在数据处理系统中管理线程的方法，包括:

检测和一个用户线程相关联的、不被该用户线程需要的内核线程，用户线程有一个相关联的用户线程堆栈；以及

响应内核线程不被需要，通过把内核线程置于一个半分离表中，半分离该内核线程，同时在与用户线程相关联的用户线程堆栈上保持内核线程的数据。

2. 如权利要求 1 所述的方法，还包括:

把所述内核线程与所述用户线程同时置于不可运行的状态。

3. 如权利要求 2 所述的方法，还包括: 响应所述用户线程从睡眠中醒来，从半分离表中删除所述内核线程，其中不需要和所述用户线程的再附加。

4. 如权利要求 2 所述的方法，还包括:

响应所述用户线程从睡眠状态醒来，确定第二内核线程是否被分离而不是半分离；以及

响应第二内核线程被分离，将第二内核线程附加到所述用户线程。

5. 如权利要求 2 所述的方法，其中不可运行的状态是等待状态或睡眠状态之一。

6. 一种用于管理线程的数据处理系统，所述数据处理系统包括:

检测装置，用于检测和一个用户线程相关的、不被该用户线程需要的内核线程，用户线程有一个相关联的用户线程堆栈；以及

半分离装置，用于响应所述内核线程是不被需要的，通过把内核线程置于一个半分离表中，半分离所述内核线程，同时在与用户线程相关联的用户线程堆栈上保持内核线程的数据。

7. 如权利要求 6 所述的数据处理系统，还包括:

放置装置，用于把所述内核线程与所述用户线程同时置于不可运行的状态。

8. 如权利要求 7 所述的数据处理系统，还包括：

删除装置，用于响应所述用户线程从睡眠中醒来，从半分离表中删除所述内核线程，其中不需要和所述用户线程的再附加。

9. 如权利要求 7 所述的数据处理系统，还包括：

确定装置，用于响应所述用户线程从睡眠状态醒来，确定第二内核线程是否被分离而不是半分离；以及

附加装置，用于响应第二内核线程被分离，将第二内核线程附加到所述用户线程。

10. 如权利要求 7 所述的方法，其中不可运行的状态是等待状态或睡眠状态之一。

用于处理数据处理系统中的线程的方法和装置

技术领域

本发明一般涉及一种改进的数据处理系统，更具体地说，涉及用于管理数据处理系统中的线程的方法和装置。

背景技术

线程是中央处理单元(CPU)利用的基本单位。线程通常具有程序计数器、寄存器组和堆栈空间。一个线程和其它的线程共用其代码部分、数据部分和操作系统资源，例如打开的文件和信号。这些组件也被称为“任务”。一些系统在用户级的库中实现用户线程，而不通过系统调用，从而线程转换不需要调用操作系统并对内核产生中断。

在许多方面，线程用和进程相同的方式操作。线程可以处于几个状态之一：就绪、阻塞、睡眠、运行或终止。在用户空间中的用户线程由内核中的内核线程运行。内核线程也被称为“虚拟处理器”。在一些情况下，使用一对一的模型，其中每个用户线程具有一个相应的内核线程。在其它的情况下，使用 M:N 模型，其中许多用户线程在少数内核线程上运行，以便提高性能。对于这个模型，会发生一些情况，例如在一个互斥体上阻塞，其中不再需要内核线程来运行特定的用户线程。互斥体是一种加锁机制，其涉及对用于抓取和释放对象的编程标记的使用。当不能被共享的数据被获得，或者不能被在系统中的其它位置同时进行的处理被开始时，互斥体被设置为“锁定”，这阻塞了使用它的其它企图。当不再需要该数据或者该例程被完成时，互斥体被设置为“非锁定”。如果没有其它的用户线程是当前可运行的，则这个内核线程将使其自己从那个特定的用户线程分离开并进入睡眠状态。

内核线程的分离以便进入睡眠状态引起若干个动作发生。发生的一个动作是，内核线程从用户堆栈转换到其自己的较小的堆栈。此外，内核线程设置信号屏蔽，以便阻塞大部分信号。当该内核线程被再次需要时，这个线程将转换到用户线程的堆栈，并设置多个线程特定的属性，例如信号屏蔽。

本发明认识到，这个当前使用的分离和随后的再附加机制涉及大的性能开销。每个分离或再附加需要系统调用，以便把数据从用户空间复制到内核空间或者从内核空间复制到用户空间。此外，数个锁被在内核库和内核两者中使用，导致可能增加的锁争用。这种类型的分离还涉及潜在的信号处理问题。具体地说，在内核线程阻塞信号之前会出现一个小的窗口，其中内核线程在执行于其小的内核堆栈上时可能会接收到一个信号。本发明还认识到在较大的用户线程堆栈上很好地运行的信号处理器可能使较小的内核线程堆栈溢出，从而破坏存储器和/或使应用程序进行核心转储（core dump）。

性能开销和堆栈溢出这两个问题是单独的问题，但是具有相似的根本原因。这个原因便是“空闲”内核线程的分离。因此，具有一种改进的方法、设备以及计算机指令，用于以减小性能开销并避免堆栈溢出的方式处理空闲的内核线程将是有利的。

发明内容

本发明提供一种用于管理线程的方法、设备和计算机指令。

一方面，本发明提供一种用于在数据处理系统中管理线程的方法，包括：检测和一个用户线程相关联的、不被该用户线程需要的内核线程，用户线程有一个相关联的用户线程堆栈；以及响应内核线程不被需要，通过把内核线程置于一个半分离表中，半分离该内核线程，同时在与用户线程相关联的用户线程堆栈上保持内核线程的数据。

优选地，将放置在半分离表上的内核线程与用户线程同时处于不能运行的状态。

优选地，当用户线程从睡眠中醒来时，该内核线程被从半分离表中删除，使得不需要被再附加到用户线程。

优选地，线程在 AIX 操作系统中执行。

优选地，用于内核线程的数据保留在用户堆栈上，而不需要把数据复制到内核堆栈以便半分离内核线程。

优选地，使用一个库进行对内核线程不被需要的检测和半分离内核线程。

优选地，当用户线程从睡眠状态醒来时，确定第二内核线程是否被分离而不是半分离，并且如果第二内核线程被分离，其被附加到该用户线程上。

优选地，所述不能运行的状态是等待状态或睡眠状态之一。

另一方面，本发明提供一种用于管理线程的数据处理系统，所述数据处理系统包括：检测装置，用于检测和一个用户线程相关的、不被该用户线程需要的内核线程，用户线程有一个相关联的用户线程堆栈；以及半分离装置，用于响应所述内核线程是不被需要的，通过把内核线程置于一个半分离表中，半分离所述内核线程，同时在与用户线程相关联的用户线程堆栈上保持内核线程的数据。

附图说明

如附图所示，下面仅以举例方式参照本发明的优选实施例说明本发明其中：

图 1 表示一种数据处理系统，其中可以按照本发明的优选实施例实施本发明；

图 2 是可以实施本发明的数据处理系统的方框图；

图 3 是表示在处理内核线程中使用的组件的示意图；

图 4A-4C 是在半分离表中使用的数据结构的示例；

图 5 是用于处理阻塞调用的已知的方法的流程图；

图 6 是用于管理空闲线程的方法的流程图；

图 7 是用于唤醒内核线程的已知的方法的流程图；以及

图 8 是用于处理内核线程的方法的流程图。

具体实施方式

现在参看附图，特别参看图 1，给出了按照本发明的优选实施例可以实施本发明的数据处理系统的图示表示。所示的计算机 100 包括系统单元 102、视频显示终端 104、键盘 106、存储装置 108，存储装置 108 可以包括软盘驱动器和其它类型的永久的和可移动的存储介质、以及鼠标 110。计算机 100 可以包括附加的输入装置，诸如例如游戏杆、触摸垫、触摸屏、轨迹球、麦克风等。计算机 100 可以使用任何适合的计算机来实现，例如 IBM eServer 计算机或者 IntelliStation 计算机，它们是位于纽约，阿蒙克的国际商业机器公司的产品。虽然给出的图示给出了一种计算机，但是本发明的其它实施例可以用其它类型的数据处理系统例如网络计算机来实施。计算机 100 优选地包括图形用户接口(GUI)，其可以借助于驻留在运行于计算机 100 内的计算机可读的介质中的系统软件来实现。

现在参见图 2，示出了本发明可以被实现于其内的数据处理系统的方框图。数据处理系统 200 是计算机例如图 1 的计算机 100 的一个例子，实现本发明的处理的代码或指令可以位于其上。数据处理系统 200 使用外围组件互连(PCI)局部总线体系结构。虽然所示的例子使用 PCI 总线，其它的总线体系结构例如加速图形端口(AGP)和工业标准结构(ISA)也可被使用。处理器 202 和主存储器 204 通过 PCI 桥 208 和 PCI 局部总线 206 相连。PCI 桥 208 还可以包括用于处理器 202 的集成的存储器控制器和高速缓冲存储器。在所示的例子中，局域网(LAN)适配器 210、小型计算机接口 SCSI 主机总线适配器 212、以及扩展总线接口 214 通过直接组件连接和 PCI 局部总线 206 相连。与此相反，音频适配器 216、图形适配器 218、以及音频/视频适配器 219 通过被插入扩展槽中的内插板卡和 PCI 总线 206 相连。扩展总线接口 214 提供用于键盘以及鼠标适配器 220、调制解调器 222 以及附加存储器 224 的连接。SCSI 主机总线适配器 212 提供了用于硬盘驱动器

226、磁带驱动器 228 以及 CD-ROM 驱动器 230 的连接。典型的 PCI 局部总线实现支持 3 个或 4 个 PCI 扩展槽或内插连接器。

一个操作系统在处理器 202 上运行并被用于协调和提供对图 2 中的数据处理系统 200 内的各种组件的控制。该操作系统可以是市场上可得到的操作系统，例如高级交互执行体(AIX)或 Windows XP。AIX 是 UNIX 的一个版本，并且可以从国际商业机器公司得到。Windows XP 可以从微软公司得到。用于操作系统和应用的指令或程序被设置在存储装置例如硬盘驱动器 226 上，并可被装入主存储器 204 中，以便由处理器 202 执行。

本领域的普通技术人员应当理解，图 2 中的硬件可以根据实施方案而改变。除了图 2 所示的硬件之外或代替图 2 所示的硬件，也可以使用其它的内部硬件或外围设备，例如闪速只读存储器(ROM)、等同的非易失性存储器、或光盘驱动器等。此外，本发明的处理可以应用于多处理器数据处理系统。

例如，数据处理系统 200，如果可选择地被配置为网络计算机，可以不包括 SCSI 主机总线适配器 212、硬盘驱动器 226、磁带驱动器 228 以及 CD-ROM 驱动器 230。在这种情况下，计算机，其被恰当地称为客户计算机，包括某种类型的网络通信接口，例如 LAN 适配器 210、调制解调器 222 或其类似物。作为另一个例子，数据处理系统 200 可以是一种独立的系统，其被配置为不依赖于某些类型的网络通信接口就可以启动，不论数据处理系统 200 是否包括某些类型的网络通信接口。作为另一个例子，数据处理系统 200 可以是个人数字助理(PDA)，其配置有 ROM 和/或闪速 ROM，用于提供用于存储操作系统文件和/或用户产生的数据的非易失性存储器。

图 2 所示的例子和上述的例子不意味着体系结构上的限制。例如，除了采取 PDA 的形式之外，数据处理系统 200 还可以是笔记本计算机或手持计算机。数据处理系统 200 还可以是信息站或 Web 设备。

本发明的处理由使用计算机执行的指令的处理器来执行，所述指令可以位于存储器例如主存储器 204、存储器 224 或一个或几个外围设备 226

－230 中。

现在参看图 3，一个按照本发明的优选实施例示出了在处理内核线程中使用的组件的示意图。在这个例子中，用户线程 300、302、304、306 和 308 位于用户空间 310 内，而内核线程 312、314 和 316 位于内核空间 318 内。这个例子中的这些线程遵从 M:N 模型，其中许多用户线程在少数内核线程上运行，以便提高性能。

当前，用户线程 300 正在由内核线程 312 运行，用户线程 304 正在由内核线程 314 运行，用户线程 306 正在由内核线程 316 运行。对这些用户线程执行的操作位于用户线程堆栈 320 内。每个用户线程和一个用户线程堆栈相关联。内核线程 312，314 和 316 具有位于用户线程堆栈 320 中的数据。用户线程堆栈 320 内的特定堆栈是和正由内核线程运行的用户线程相关联的堆栈。

一般地说，如果内核线程例如内核线程 312 对于运行用户线程例如用户线程 300 来说不再被需要，则内核线程 312 将分离其自身，并和用户线程 300 一道进入睡眠状态。在睡眠状态下的内核线程一般被置于分离表 322 内，所述分离表由 Pthread 库 324 管理。Pthread 库 324 是一个可动态加载的库，其被用于 AIX 中。随着从用户线程 300 分离，用于内核线程 312 的信息，所述信息表示当前堆栈指针，该指针指向用户线程堆栈 320 内的用户线程堆栈，被修改为指向为它的内核线程在内核线程堆栈 326 中保留的区域。当用户线程 300 从睡眠状态醒来时，内核线程 312 可以被从分离表 322 中删除，并再附加到用户线程 300。可替换地，如果内核线程 312 为不可得到的，则可以将分离表 322 中的另一个可以得到的内核线程附加到用户线程 300。

按照本发明的优选实施例，当用户线程 300 进入不需要内核线程 312 的状态例如睡眠状态时，借助于 Pthread 库 324，内核线程 312 被置于半分离表 328 中，而不被置于分离表 322 中。对于半分离，内核线程 312 不改变其堆栈或信号屏蔽。而是，内核线程 312 被置于半分离表 328 上，并和用户线程 300 同时睡眠。在这个同时睡眠状态下，内核线程 312 保持诸

如和其同时睡眠的用户线程的堆栈和信号屏蔽的信息。这个信息也被称为“用户线程特定属性”。以这种同时睡眠，内核线程 312 把自身识别为可用于运行其它的用户线程，但是最好是首先使用分离表 322 上的一个线程。优先选择被给予运行和内核线程 312 相关联的用户线程，即用户线程 300。根据特定的实施方案，半分离表 328 可以用不同的方式来实现。在这个例子中，这个表被实现为线程结构的链接的内核。

如果用户线程 300 从睡眠状态醒来，内核线程 312 将把自身从半分离表 328 中删除，并继续运行用户线程 300。这种机制提供一个途径，其中唤醒 Pthread 库 324 中的一个线程只涉及较低的等待时间。当从被争夺的互斥体、条件变量或信号醒来时，这种机制是有用的，因为跟随这些事件而执行的动作通常必须在程序的其余部分可以进行之前被完成。

如果在后来的时间需要半分离表 328 中的内核线程来运行另外的用户线程，则借助于从当前用户线程的状态分离到新的用户线程的状态的而不分离内核线程，可以避免因保持分离而导致的性能处罚。例如，如果内核线程 312 在半分离表 328 上，并且另一个用户线程例如用户线程 302 需要内核线程 312 以运行该线程，通过将用户线程 300 的堆栈内的属性改变为和用户线程 302 的那些属性匹配，和用户线程 300 相关联的堆栈可被附加到用户线程 302。例如，堆栈指针和信号屏蔽可以被从用于用户线程 300 的堆栈指针和信号屏蔽改变为用于用户线程 302 的堆栈指针和信号屏蔽。

结果，只有当和内核线程相关联的用户线程退出或结束时，该内核线程才完全分离。因而，在大多数情况下，避免了和通常的分离相关联的性能开销。

参见图 4A-4C，其中按照本发明的优选实施例示出了在半分离表中使用的数据结构。这些数据结构可用于实现半分离表，诸如图 3 中的半分离表 328。

在图 4A 中，使用链接表指向不同的内核线程。例如，表项 400 和 402 被用于指向不同的内核线程。表头标记 406 标识着链接表的开始。表项 400 包含前一个指针 408 和下一个指针 410。表项 402 包含前一个指针 412 和

下一个指针 414。这些指针用于指向链接表内的前一个表项和下一个表项。此外，表项 400 包括指针 416，表项 402 包括指针 418，这些指针指向内核线程，诸如内核线程 420 和 422。

接着，在图 4B 中，在表结构中使用的信息被合并在线程结构中。在这个例子中，表头标记 430 指向开头，或第一个线程，内核线程 432。内核线程 432 包括有下一个指针 434，指针 434 指向内核线程 436。内核线程 432 还包括前一个指针 438，指针 438 指向表中的某个前面的内核线程。内核线程 436 包括前一个指针 440，指针 440 指回内核线程 432。内核线程 436 中的下一个指针 442 指向表中的下一个内核线程。这个例子是给出的例子中的一种优选的表结构。

在图 4C 中，使用数组 450 指向不同的内核线程。表头标记 452 指向数组 450 的开始，数组 450 包括指针 454，456 和 458。这些指针分别指向内核线程 460，462 和 464。提供这些例子仅仅用于说明可以如何实现半分离表。根据特定的实现方案，其它类型的结构，例如树，也可以被使用。

现在参看图 5，给出了用于处理阻塞调用的已知方法的流程图。图 5 中所示的方法可以被实现在一个库中，例如图 3 中的 Pthread 库 324。阻塞调用是可以引起用户级的线程从运行或可运行状态改变到等待状态或睡眠状态的另一状态的任何调用。

该方法以检测潜在阻塞调用开始（步骤 500）。该方法从用户线程分离内核线程（步骤 502）。步骤 502 需要从用户线程堆栈向内核线程堆栈复制信息以及其它的操作的开销，所述的其它的操作诸如将信号屏蔽改变为阻塞信号。接着，该方法查找新的用户线程以便运行（步骤 504）。如果发现可运行的线程，则该内核线程被附加到该新的用户线程（步骤 506）。步骤 506 涉及这样的开销，诸如从内核线程堆栈向用户线程堆栈复制数据以及设置信号屏蔽。新的用户线程被运行（步骤 508），此后该方法结束。

再次参看步骤 504，如果没有可运行的线程，内核线程便被置于分离表上，并成为睡眠状态（步骤 510）。此后，该方法等待，以便检测成为可运行的用户线程（步骤 512）。此后，内核线程被附加到作为可运行的

用户线程被检测到的用户线程（步骤 514）。然后，该用户线程由内核线程运行（步骤516），此后，该方法结束。

现在参见图 6，按照本发明的优选实施例示出了用于管理空闲线程的方法的流程图。图 6 所示的方法可以在库中实现，例如图 3 所示的 Pthread 库 324。

该方法从检测潜在的阻塞调用开始（步 600）。这种潜在的阻塞调用是一种把用户级线程置于等待或睡眠状态的调用。正在睡眠或者正在等待的线程被称为“不可运行的”线程。然后该方法寻找新的用户线程以便运行（步骤 602）。如果发现了可运行的用户线程，该方法将内核线程转换到该新的用户线程（步骤 604）。步骤 604 的转换可以通过使用用户级调度器来实现。根据特定的实施方案，这个转换可能需要或者不需要内核线程的分离和再附加。用户线程由该内核线程运行（步骤 606），此后该方法结束。

再次回到步骤 602，如果没有发现可运行的用户线程，内核线程被置于半分离的睡眠状态（步骤 608）。在这种类型的睡眠中，内核线程被置于半分离表上，然后进入睡眠状态。和该内核线程相关联的用户线程与该内核线程一起被保持或被列于半分离表中。进行这种关联以便指出优先使用这个特定的内核线程来运行该用户线程，而不是运行另一个用户线程，除非必需这样做。

此后，对一个事件进行检测和处理（步骤 610）。如果该事件是与该内核线程相关联的用户线程成为是可运行的，则该用户线程由该内核线程运行（步骤612），此后该方法结束。再次参看步骤610，如果和该内核线程相关联的用户线程退出或者不需要该内核线程，则该内核线程被置于分离表上并进入睡眠状态（步骤 614），此后该方法结束。更具体地说，步骤614 进入图 5 所示的以步骤510 开始的一系列的步骤。

返回步骤 610，如果事件是任何其它的事件，则该方法查找用户线程以便运行（步骤616）。如果未找到可运行的线程，则该方法返回步骤608。否则，如果另一个用户线程成为可运行的，则该方法把该内核线程转换到

成为可运行的那个用户线程（步骤 618）。这个步骤可以包括两个操作，即分离和附加。可替换地，可以使用把内核线程从当前的用户线程转换到新的用户线程的单个的操作。然后用户线程以内核线程运行（步骤 620），然后该方法结束。

现在参见图 7，其中示出了用于唤醒内核线程的已知方法的流程图。图 7 所示的方法可以在一个库中实现，例如图 3 所示的 Pthread 库 324。在本图中所示的方法允许线程等待或睡眠，直到一个用户线程被检测为是可运行的。

该方法由检查用于内核线程的分离表开始（步骤 700）。如果分离表不是空的，则从分离表中删除一个内核线程（步骤 702）。该内核线程被唤醒（步骤 704），此后该方法结束。再次参看步骤 700，如果分离表是空的，则该方法结束。

现在转向图 8，示出了按照本发明的优选实施例用于处理内核线程的流程图。图 8 中所示的方法可以在一个库中实现，例如图 3 中的 Pthread 库 324。该方法在一个使另一个内核线程成为可运行的内核线程中启动。

该方法以检查用于内核线程的分离表开始（步骤 800）。如果分离表是空的，该方法便检查用于内核线程的半分离表（步骤 802）。如果半分离表是空的，则该方法结束。否则，如果半分离表不是空的，则从半分离表中删除一个内核线程（步骤 804）。从半分离表中被删除的这个内核线程被唤醒（步骤 806），此后该方法结束。而后，该内核线程运行。

再次参看步骤 800，如果分离表不是空的，则从分离表中删除一个内核线程（步骤 808）。该内核线程被唤醒（步骤 810），此后该方法结束。本发明的机制仍然使用分离表，因为分离而不是半分离在某些情况下可能是需要的。这个表对于已有的用户线程可能是需要的。在这种情况下，当唤醒用户线程时，从分离表中选择内核线程优先于从半分离表中选择内核线程。

因而，本发明提供一种用于管理空闲的内核线程的改进的方法、设备和计算机指令。本发明的机制避免了当前内核线程分离和随后再附加所涉

及的开销。此外，本发明的机制还避免了可能发生的堆栈溢出。这些优点是通过使用半分离方法提供的。这个方法涉及把内核线程置于半分离表中，并且然后将内核线程置于睡眠状态。执行这些步骤不需要在使内核线程和用户线程分离时一般所需的那些步骤。

重要的是注意到，虽然在一个功能完全的数据处理系统的环境下说明了本发明，本领域的普通技术人员应当理解，本发明的方法能够以指令的计算机可读介质的形式以及各种形式发布，并且，不管实际上用于进行发布的信号承载介质的具体类型是什么本发明都同样地适用。计算机可读介质的例子包括可记录型的介质例如软盘、硬盘驱动器、RAM、CD-ROM、DVD-ROM；以及传输型的介质，例如数字和模拟通信链路、使用诸如例如射频和光波传输的传输形式的有线或无线通信链路。计算机可读介质可以采取编码格式的形式，其被解码以便实际用于特定的数据处理系统中。

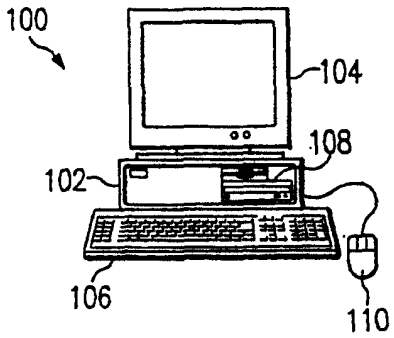

100
104
108
102
106
110

图 1

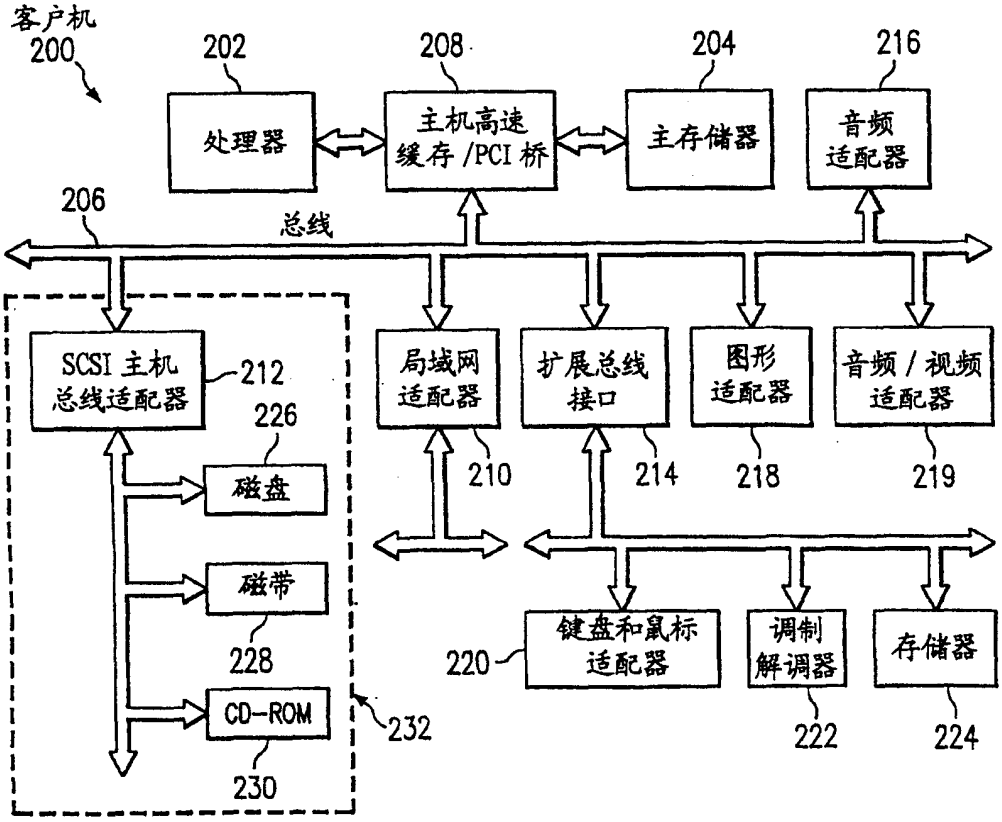

客户机
200
202
208
204
216
处理器
主机高速
缓存 /PCI 桥
主存储器
音频
适配器
206
总线
SCSI 主机
总线适配器
212
226
磁盘
磁带
228
CD-ROM
230
232
局域网
适配器
210
扩展总线
接口
214
图形
适配器
218
音频 / 视频
适配器
219
220
键盘和鼠标
适配器
调制
解调器
222
存储器
224

图 2

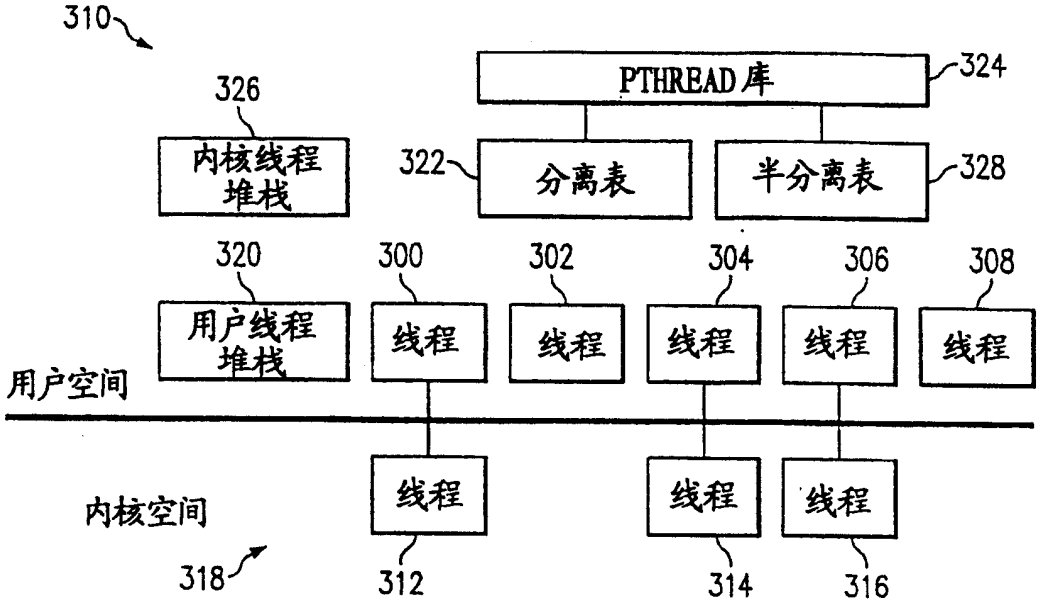

310
PTHREAD库
324
326
内核线程堆栈
322
分离表
半分离表
328
320
300
302
304
306
308
用户线程堆栈
线程
线程
线程
线程
线程
用户空间
线程
线程
线程
内核空间
318
312
314
316

图 3

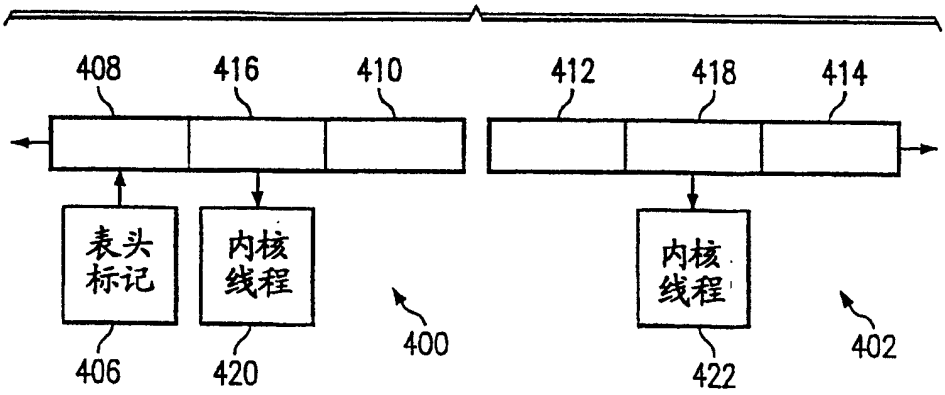

408
416
410
412
418
414
表头标记
内核线程
内核线程
400
402
406
420
422

图 4A

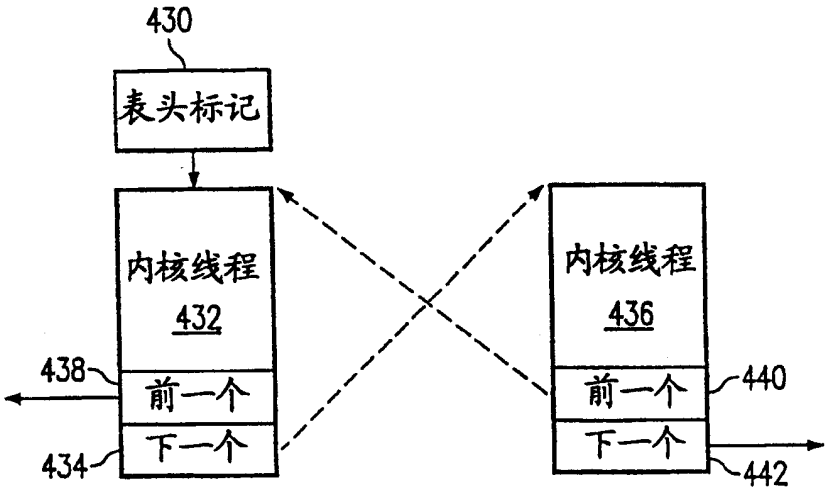

430
表头标记
内核线程
432
438
前一个
下一个
434
内核线程
436
前一个
440
下一个
442

图 4B

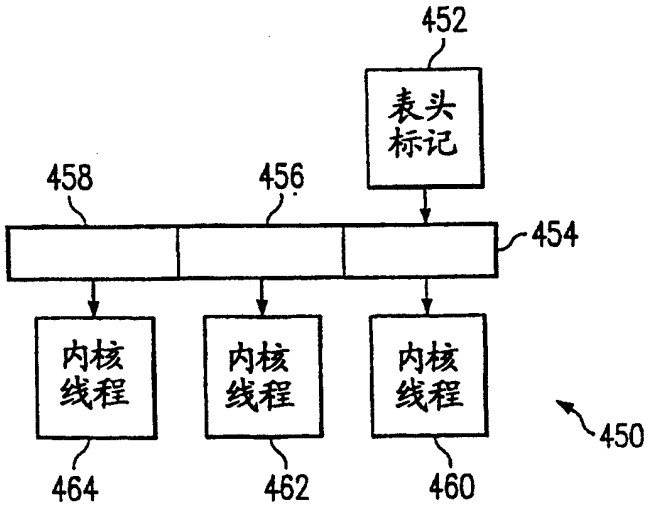

452
表头
标记
458
456
454
内核
线程
内核
线程
内核
线程
450
464
462
460

图 4C

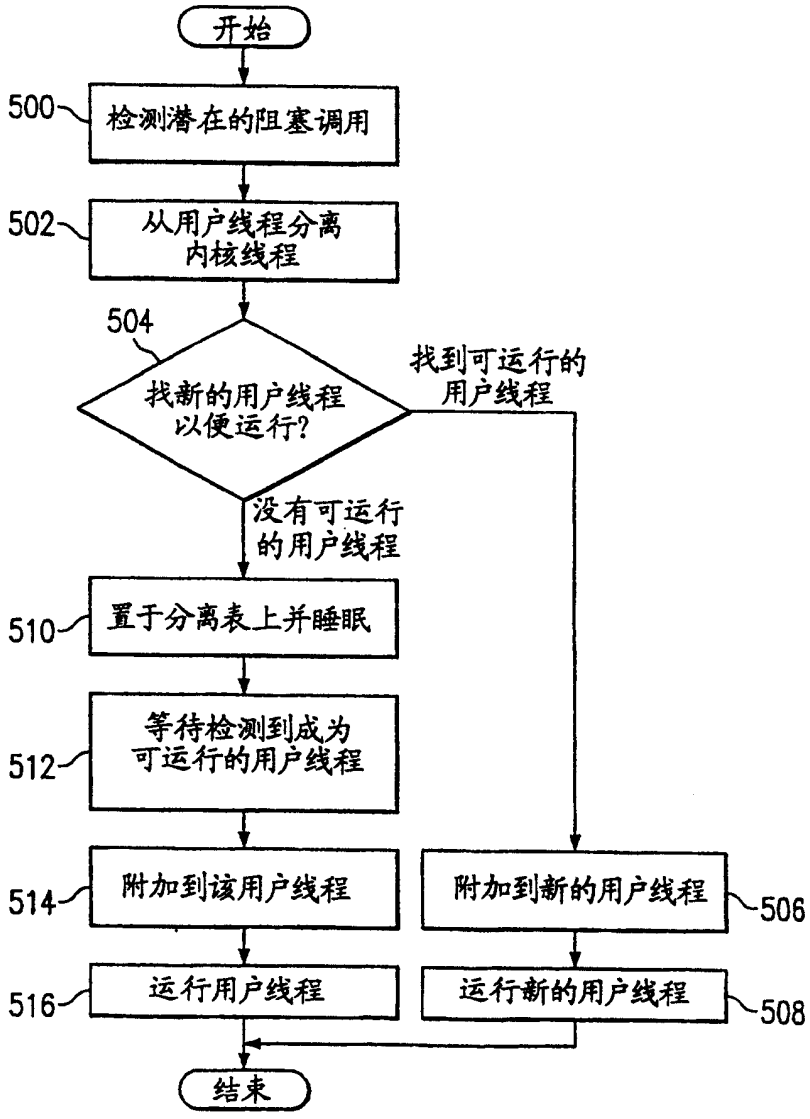

开始
500
检测潜在的阻塞调用
502
从用户线程分离
内核线程
504
找新的用户线程
以便运行?
找到可运行的
用户线程
没有可运行
的用户线程
510
置于分离表上并睡眠
512
等待检测到成为
可运行的用户线程
514
附加到该用户线程
附加到新的用户线程
506
516
运行用户线程
运行新的用户线程
508
结束

图 5

(现有技术)

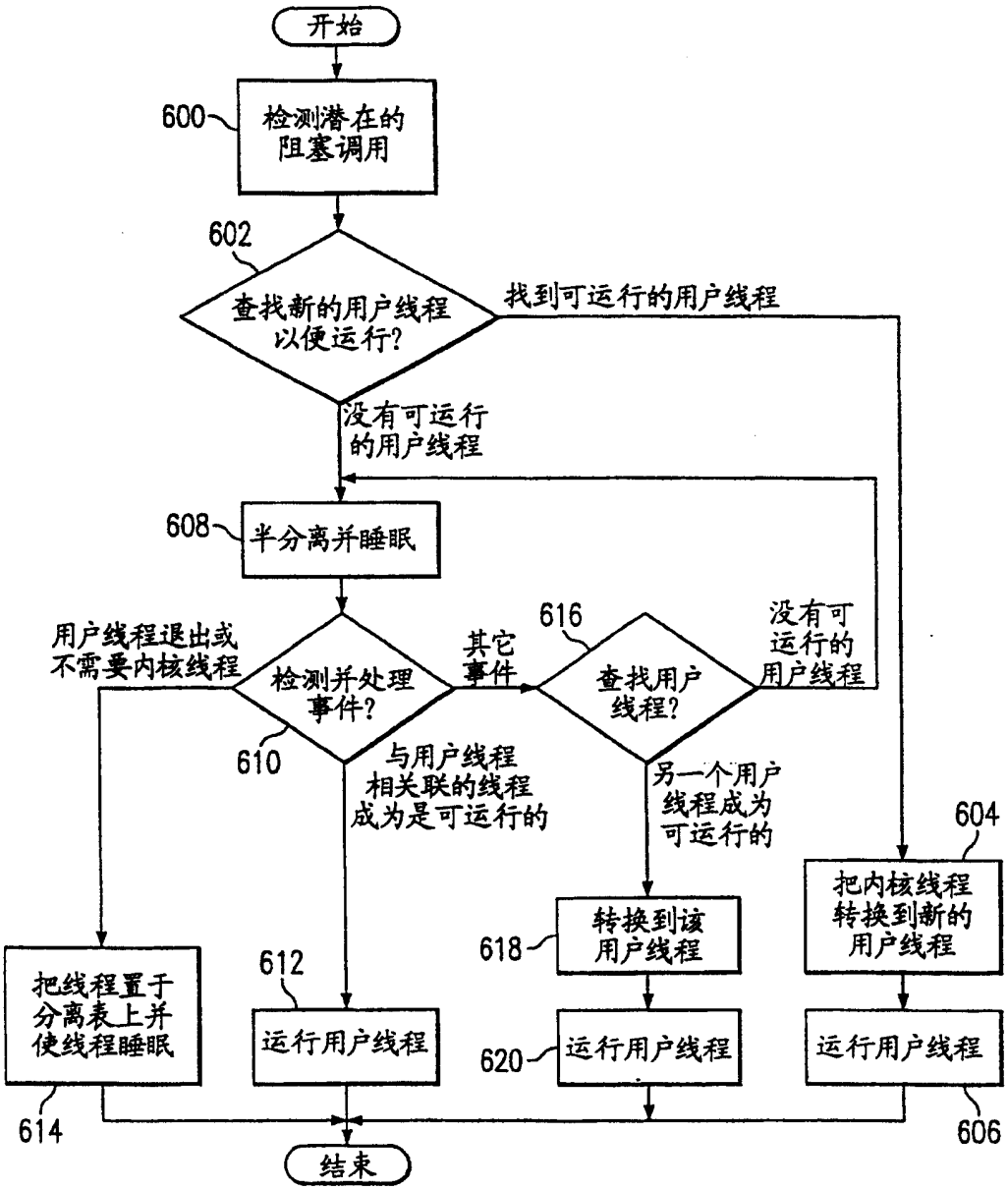

开始
600
检测潜在的阻塞调用
602
查找新的用户线程以便运行?
找到可运行的用户线程
没有可运行的用户线程
608
半分离并睡眠
用户线程退出或不需要内核线程
检测并处理事件?
610
其它事件
616
查找用户线程?
没有可运行的用户线程
与用户线程相关联的线程成为是可运行的
另一个用户线程成为可运行的
604
把内核线程转换到新的用户线程
618
转换到该用户线程
把线程置于分离表上并使线程睡眠
612
运行用户线程
620
运行用户线程
运行用户线程
614
606
结束

图 6

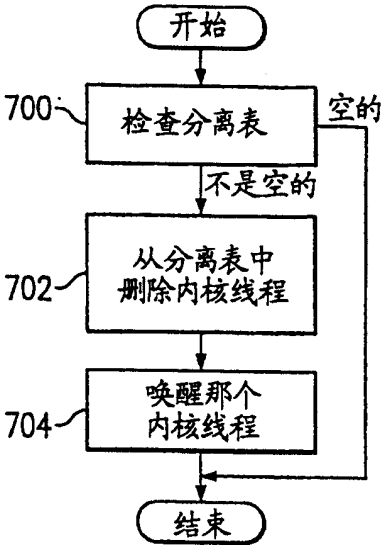

开始
700
检查分离表
空的
不是空的
从分离表中
删除内核线程
702
唤醒那个
内核线程
704
结束

图 7
(现有技术)

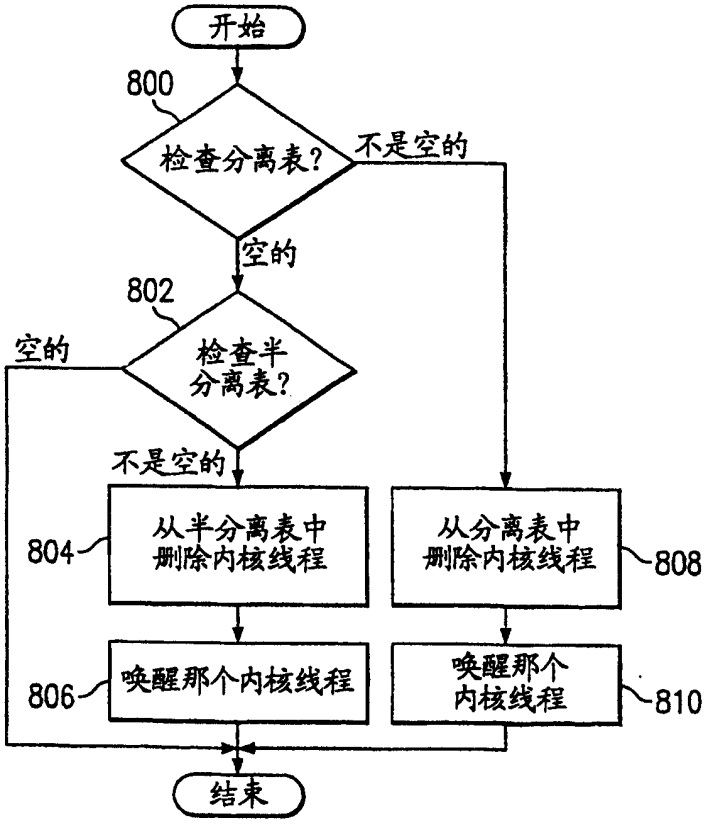

开始
800
检查分离表?
不是空的
空的
802
检查半
分离表?
空的
不是空的
从半分离表中
删除内核线程
804
从分离表中
删除内核线程
808
唤醒那个内核线程
806
唤醒那个
内核线程
810
结束

图 8