



[12] 发明专利申请公开说明书

[21] 申请号 03122588.8

[43] 公开日 2004 年 1 月 14 日

[11] 公开号 CN1467624A

[22] 申请日 2003.4.21 [21] 申请号 03122588.8

[30] 优先权

[32] 2002. 6. 20 [33] US [31] 10/177453

[71] 申请人 惠普公司

地址 美国加利福尼亚州

[72] 发明人 L·文卡塔苏布拉马尼安

[74] 专利代理机构 中国专利代理(香港)有限公司

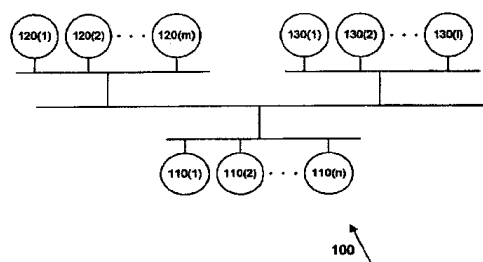
代理人 栾本生 王 勇

权利要求书 2 页 说明书 8 页 附图 2 页

[54] 发明名称 多线程系统中的回旋-让渡

[57] 摘要

提供用于多线程系统中的回旋-让渡的技术。将每个等待锁的线程连接到多个回旋-让渡处理器中的一个回旋-让渡处理器,使得不然就会被这些等待线程用于回旋-让渡循环其它处理器能被其他线程用于其它用途。进一步,在一个预定的时间期间,根据一个线程在该预定的时间期间内经历的自愿上下文转换的数目,将该线程连接到一个回旋-让渡处理器或从该回旋-让渡处理器中去除。



1. 一种用于多线程系统中的方法，包含以下步骤：

在一个线程正在等待一个被另一个线程(110)获得的锁的期间，
将被标识为回旋-让渡处理器(130, 204)的处理器(120, 204)专用于
5 运行这个线程(110)；

一个获得在第一处理器(120, 204)上运行一个程序部分的第一
锁的第一线程(110)；其中运行该程序部分需要第一锁；和

一个第二线程(110)，在等待获得该第一锁的期间执行以下步骤：

在回旋-让渡处理器(130, 204)上运行；

10 尝试获得第一锁；和

如果第二线程(110)得不到第一锁，则第二线程(110)
重复运行和尝试的步骤。

2. 如权利要求1的方法，其中，第二线程(110)在预定的时间期
间内重复运行和尝试的步骤。

15 3. 如权利要求1的方法，其中，第二线程(110)在第一次在回旋
-让渡处理器(130, 204)上运行之前，执行以下步骤：

在第一处理器(120, 204)上运行；和

尝试获得锁；和

如果第二线程(110)得不到第一锁，则第二线程(110)重复在
20 第一处理器(120, 204)上运行和尝试获得锁的步骤。

4. 如权利要求3的方法，其中，第二线程(110)在预定的时间期
间内重复在第一处理器(120, 204)上运行和尝试获得锁的步骤。

5. 如权利要求1的方法，其中，如果第二线程(110)得不到第一
锁，则第二线程(110)执行让出回旋-让渡处理器(130, 204)的步
25 骤。

6. 如权利要求1的方法，其中，回旋-让渡处理器(130, 204)
是从多个专用于运行等待线程(110)的处理器(120, 204)中选择的。

7. 如权利要求1的方法，其中，每次第二线程(110)在回旋-让
渡处理器(130, 204)上运行时，从多个专用于运行等待线程(110)
30 的处理器(120, 204)中选择回旋-让渡处理器(130, 204)。

8. 如权利要求1的方法，还包含的步骤是，根据第三线程(110)
在预定时间期间内经历的上下文转换的数目，允许第三线程(110)在

回旋-让渡处理器(130, 204)上运行。

9. 如权利要求1的方法, 其中, 根据第二线程(110)在预定时间期间内经历的上下文转换的数目, 第二线程(110)退出在回旋-让渡处理器(130, 204)上的运行。

- 5 10. 如权利要求1的方法, 其中, 在第二线程(110)等待获得第一锁期间, 第二线程(110)独占地在回旋-让渡处理器(130, 204)上运行。

多线程系统中的回旋-让渡

技术领域

- 5 本发明总体涉及多线程系统，更具体来说，涉及这种系统中的回旋-让渡(spin-yielding)。

背景技术

- 10 在多线程程序中，需要保护程序的某些部分和/或变量不被多个线程访问。在一种使用封锁机制的方法中，线程在访问受保护部分时获得一个锁，由此在该线程保持该锁的时间内防止其它线程访问同一部分。当该线程释放该锁时，另一个线程可以获得该锁，以访问该受保护部分并防止其它线程访问同一部分。

- 15 可以将等待锁的线程放入一个睡觉队列(sleep queue)，在这个等待线程做好执行准备时将其转移到一个运行队列。然而，在睡觉队列与运行队列之间的转移一般是代价高的。在已经开发出来的有些方法中，等待线程在被转移到睡觉队列之前先执行一些回旋-让渡循环。在这些方法中，一个等待队列在等待锁的同时，在一个处理器上回旋(spin)一些周期，然后再尝试获得锁。如果该线程不能获得锁，则该线程让渡(yield)该处理器，即将处理器让给其它线程使用。该等待线
20 程重复回旋-让渡循环，直到次数达到一个阈值，这时才将该等待线程放入睡觉队列中。经验显示，等待线程一般经过几次尝试就能获得锁，而不被转移到睡觉对列中。

- 25 然而，在诸如锁竞争严重和/或线程长时间把持锁的应用等各种情况中，回旋-让渡线程在许多情况下最终要被放到睡觉队列。在回旋-让渡的时期中，执行回旋-让渡循环的处理器不能完成其它有用的工作，回旋-让渡线程可能产生大量昂贵的系统调用。在具有许多处理器的对称处理器系统中，可以说就有这个效果。

根据以上叙述，需要提供解决上述缺陷和相关问题的机制。

发明内容

- 30 本发明在各种实施例中提供用于多线程系统中的回旋-让渡的技术。在一个实施例中，一个具有 n 个线程的过程在 m 个处理器和 1 个“回旋-让渡”(spin-yield)处理器上运行。将每个等待锁的线程连

接到 (bound to) 这 1 个回旋-让渡处理器中的一个回旋-让渡处理器，使得不然就会被这些等待线程用于回旋-让渡循环的 m 个处理器能被其他线程用于其它用途。进一步，在一个预定的时间期间，根据一个线程在该预定的时间期间内经历的自愿上下文转换的数目，将该线程连接到 5 一个回旋-让渡处理器或从该回旋-让渡处理器中去除。

附图说明

本发明是通过各附图中的例子非限定性地解释的，附图中相同的标注号表示类似的部件。

图 1 表示一个可以在其中实现本发明的实施例的系统；

10 图 2 表示一个可以在其中实现本发明的实施例的计算机系统。

具体实施方式

在以下的说明中，为了解释的需要，陈述了众多的具体细节，以便于透彻地理解本发明。然而，对于该领域的熟练人员来说，不用这些具体细节就可以实现本发明。在其他例子中，为了突出本发明，用框图 15 图来表示众所周知的结构和器件。

图 1 表示一个可以在其中实现本发明实施例的系统 100。系统 100 包括多个线程、处理器、和“回旋-让渡”处理器，为了表示的需要，将它们分别称为线程 110 (1)-110 (n)、处理器 120 (1)-120 (m)、以及回旋-让渡处理器 130 (1)-130 (1)。线程 110、处理器 120、 20 以及回旋-让渡处理器 130 相互之间通过相应的总线通信。整数 n 、 m 、1 表示线程 110 的数量、处理器 120 的数量和回旋-让渡处理器 130 的数量不同并且可以是随机的。

线程

线程 110 是程序中能独立于其它部分执行的一部分。每个线程 110 25 与一个线程标识符 (TTD) 相关联，多个线程 110 可能是一个具有过程标识符 (PID) 的过程的一部分。因为创建一个线程 110 比创建一个全面的过程在 CPU 时间和其它资源方面的代价较低，所以可以将线程 110 称为“轻量级过程”。在同一个过程中的各线程 110 共享分配给该过程的系统资源，如存储器。线程 110 见于各种应用程序，诸如 C 和 C++ 30 应用，Java 应用，Java 虚拟机，包括例如 web 服务器、数据库服务器、BtoB 逻辑等等的企业应用。通常，一个过程在一个处理器上运行。然而，一个有多个线程 110 的过程可以在多个处理器上运行，这是因为

一个线程 110 可在一个处理器上运行。执行准备就绪的线程 110 待在运行队列中，调度程序根据该队列为该线程安排运行时间。执行准备没有就绪的线程 110 待在睡觉队列中，在执行准备就绪时被放入运行队列。

- 5 视各种应用而定，多个线程 110 可以同时地访问一个程序的不同部分。然而，只有一个线程可以访问“受保护的”变量、代码片断或部分程序部分。受保护的部分一般是程序的一个重要程序段，对该程序段的不加控制的修改会产生不利的后果。金融账户余额的变量就是一例包含在受保护部分中的受保护变量。如果同时访问这个变量并对其进行修改，会导致不准确和不利的后果。当某线程访问一个受保护部分时，线程获得一个对应该部分的锁，由此能防止其它线程同时访问该相同部分，因为需要该锁才能访问该受保护部分。封锁机制是运行应用程序的操作系统通常执行的。

- 15 为了方便解释，将持有锁的线程称为线程 110L，而将等待锁的线程称为 110W。在一个实施例中，在一个等待线程-例如线程 110w(1) 在一个处理器 120 上回旋-让渡的时间已经超过一个可设置的阈值 t_1 后，就将该线程 110w(1) 连接到一个专用的回旋-让渡处理器 130-例如处理器 130(1)。这个阈值 t_1 是变化的，可以低至 0。阈值 t_1 等于 0 时，表示一旦线程 110w(1) 开始等待锁，就将线程 110w(1) 连接到一个回旋-让渡处理器 130。阈值 t_1 不等于 0 时，表示等线程 110w(1) 在一个处理器 120 上回旋-让渡的时间超过阈值 t_1 后，再将线程 110w(1) 连接到一个回旋-让渡处理器 130。用户可以根据各种因素来决定阈值 t_1 的值，这些因素诸如是，用户有关该系统的经验、线程 110W 获得锁之前所经历的回旋-让渡循环的平均次数、不在等待锁的有可能作其他工作的线程 110 的数量，等等。例如，如果用户确定线程 110W(1) 将需要很长时间才能获得锁，并且/或者线程 110W(1) 在其上回旋的处理器 120(1) 不然的话能作其他工作，则用户可以设定一个较低的阈值 t_1 。这使线程 110W(1) 能在短时间内被连接到一个回旋-让渡处理器 130。然而，如果短时间内就能获得锁并且/或者系统能提供用于回旋-让渡的处理器 120(1)，则用户可以设定一个较高的阈值 t_1 。阈值 t_1 可以以实际时间记，例如以微秒、秒为单位，或者用其它时间单位，例如线程 110W(1) 已经回旋-让渡的循环次数、处理器循环(processor

cycles)次数、上下文转换(context switch)的数量等等。

在一个实施例中，回旋指的是执行一些空操作(NOP)程序语句，让渡指的是释放处理器供其它线程使用。术语“回旋-让渡”是个习惯用语。一个回旋-让渡循环包括回旋、获锁失败、让出处理器、以及——通常在循环的结尾——被转移到运行队列。例如，线程 110W(1)在等待锁时，先回旋一个时间，然后尝试获得锁。如果该线程 110W(1)获得锁，则该线程 110W(1)变成一个有锁并且阻止其它线程获得该同一个锁的线程 110L。然而，如果该线程 110W(1)不能获得锁，则该线程 110W(1)让出回旋-让渡处理器 130(1)供另一个正在等待的线程 110W 使用，并被转移到运行队列中，等待再次在一个回旋-让渡处理器上运行或回旋。线程 110W(1)然后若干次地重复进行回旋、尝试获锁、让出处理器、以及被转移到运行队列，直到达到阈值 t2 时，线程 110W(1)被放到睡觉队列中。类似于以上讨论的阈值 t1，阈值 t2 可以以实际时间记，或者用其它时间单位。在一个实施例中，线程 110W(1)在同一个回旋-让渡处理器-例如处理器 130(1)上重复回旋-让渡循环。就是说，线程 110W(1)每次从运行队列再次返回去回旋时，线程 110W(1)在同一个回旋-让渡处理器 130(1)上回旋。在备择实施例中，如果某线程 110W 被连接到一组回旋-让渡处理器，则线程 110W(1)每次从运行队列再次返回去回旋时，线程 110W(1)在该组中此时可能是合适的——一个回旋-让渡处理器 130(1)上回旋。可选作用于回旋-让渡的较好的候选的例子，是在其上运行的线程数少的处理器 130。在一个实施例中，把线程 110W 连接到回旋-让渡处理器 130，要调用一个对功能 MPCTL 的系统调用，这个功能普遍见于 Unix 操作系统。功能 MPCTL 指定一个在一个处理器或一组处理器上运行的线程——如果它受操作系统支持的话。当某个线程 110W 被连接到一个或一组回旋-让渡处理器 130 时，该线程 110W 独占地在处理器 130 上运行，不许在任何处理器 120 上运行，这就使处理器 120 能执行其它与回旋-让渡无关的任务。这就改善了系统性能。

处理器

处理器 120 是计算机中普遍存在的，可称为计算机的大脑。一般来说，处理器 120 执行内存中存储的指令，控制逻辑，处理信息，执行算术和逻辑运算，等等。线程 110 可以在处理器 120 上运行。调度程

序安排线程 110 在特定的处理器 120 或 130 上运行。

回旋-让渡处理器

回旋-让渡处理器 130 类似于处理器 120, 但是是专用于回旋-让渡的。就是说, 回旋-让渡处理器 130 服务于所有正在等待锁的线程 110W。回旋-让渡处理器的个数 1 最小为 1, 可以随各种因素而变化, 这些因素例如包括, 系统中可用的处理器的总数, 与处理器个数相比在运行的线程 110 的个数, 可用作回旋-让渡处理器的“自由”处理器的个数等等。一般来说, 等待线程 110W 的个数越多, 被提供用作回旋-让渡处理器 130 的处理器 120 的个数就越多。用回旋-让渡处理的多个处理器 130, 能允许多个线程 110W 同时地回旋-让渡。例如, 线程 110W(1) 在处理器 130(1) 上回旋-让渡, 线程 110W(2) 在处理器 130(2) 上回旋-让渡, 线程 110W(3) 在处理器 130(3) 上回旋-让渡, 是这三个处理器 130(1)、130(2)、130(3) 而不是一个处理器 130, 服务于三个线程 110W(1)、110W(2)、110W(3) 的全部数量的回旋-让渡循环。

可以用不同的方式选择作为回旋-让渡处理器 120 的处理器 130, 这些方式例如包括, 随机地选择一个处理器 120, 或者选择一个在选择时相对自由的处理器 120-例如被安排比另一个处理器 120 执行较少任务的处理器, 等等。

自愿上下文转换

自愿上下文转换发生在某处理器上运行的线程自愿让出该处理器供其它线程使用时。这通常发生在线程为睡觉(sleep)、让出(yield)等等而进行系统调用时。在一个实施例, 根据例如线程 110(1) 的一个线程在预定的时间期间内经历的自愿上下文转换的数目, 将线程 110(1) 连接到例如处理器 130(1) 的一个回旋-让渡处理器或从该回旋-让渡处理器中去除。例如, 如果在从时间 t1 到时间 t2 的 10 秒钟内, 线程 110(1) 经历的自愿上下文转换的数目多于一个阈值, 例如 50 个上下文转换, 则将线程 110(1) 连接到一个例如处理器 130(1) 的回旋-让渡处理器。类似地, 如果线程 110(1) 已经被连接到一个例如处理器 130(1) 的回旋-让渡处理器, 并且在预定的时间期间内经历了不到 50 个上下文转换, 则将线程从处理器 130(1) 释放。在一个实施例, 线程 110(1) 被释放后, 就被转移到运行队列, 以便被安排到自由处理器 120 上执行。自愿上下文转换的数目一般由操作系统保存。

自愿上下文转换被用作一个例子。上述技术可以被用于在例如为线程保留的时间片到期、线程被另一个具有更高优先权的线程或过程占先时由操作系统或其他适当的单元引起的自愿上下文转换。

5 在一个实施例中，提供诸如时间间隔、阈值、回旋-让渡处理器数等各种参数作为程序使用的变量。线程要利用这些变量来按照上述技术连接(bind)或脱离回旋-让渡处理器 130。

改善系统性能

本文所述技术能改善系统性能。尽管进行了两个试验，一个没有专用回旋-让渡处理器，一个有专用回旋-让渡处理器；有专用回旋-让渡
10 处理器的试验显示出更好的系统性能。在两个实验中，使用了 150 个线程 110 和 8 个处理器，其中 50 个线程被安排作为线程 110W，即它们在等待获得锁，因此执行回旋-让渡循环。为了解释方便，将其余 100 个不在回旋-让渡的线程 100 称作线程 110N，执行一组示例性的任务。在没有专用回旋-让渡处理器的第一个实验中 100 个线程 110N 和 50 个
15 线程 110W 在 8 个处理器上运行它们的任务。按照 10 次运行的平均，这些线程 110N 和 110M 执行它们的任务需要大约 269 秒钟。在第二个实验中，这 8 个处理器中的两个被专门用作回旋-让渡处理器 130，因此剩下 6 个处理器 120。在 50 个线程 110W 在两个回旋-让渡处理器 130 上回旋-让渡的同时，100 个线程 110N 在六个处理器 120 上执行相同
20 的示例性任务。结果，按照 10 次运行的平均，100 个线程 110N 和 50 个线程 110M 完成同样的任务需要大约 169 秒，而不是 269 秒。

计算机系统概述

图 2 表示一个可以在其中实现本发明的实施例的计算机系统 200
25 的框图。例如，计算机系统 200 可以被实现得包括系统 100，以按照上述技术运行程序和执行功能等等。在一个实施例中，计算机系统 200 包括一个中央处理单元(CPU) 204，随机存取存储器(RAM) 208、只读存储器(ROM) 212、存储器 216 和通信接口 220，它们都连接到总线 224。

CPU 204 控制逻辑，处理信息，协调计算机系统 200 内部的活动。在一个实施例中，CPU 204 通过例如协调数据从输入设备到显示设备
30 232 的移动而执行存储在 RAM 208 和 ROM 212 中的指令。CPU 204 可包括一个或多个处理器。

RAM 208 通常被称作主存储器，它暂存信息和 CPU 204 要执行的指

令。RAM 208 中的信息，可以从输入设备 228 获得，或者由 CPU 204 生成，作为被 CPU 204 执行的指令所需要的算术过程的一个部分。

ROM 212 存储信息和指令，这些信息和指令一旦在 ROM 芯片中存储，就是只能读的，不能修改和删除。在一个实施例中，ROM 212 存储用于计算机系统 200 的配置和初始操作的命令。

存储器 216，诸如软盘、盘驱动器或磁带驱动器，能长久地存储供计算机系统 200 使用的信息。

通信接口 220 使计算机系统 200 能与其它计算机或设备接合。通信接口 220 例如可以使调制解调器、集成业务数字网 (ISDN) 卡、局域网 (LAN) 端口等等。所属技术领域的熟练人员知道，调制解调器或 ISDN 卡通过电话线提供数据通信，而 LAN 端口则通过 LAN 提供数据通信。通信接口 220 也可以允许无线通信。

总线 224 可以是任何用于传送供计算机系统 200 使用的信息的通信机制。在图 2 的例子中，总线 224 是一个在 CPU 204、RAM 208、ROM 212、存储器 216 和通信接口 220 之间传送数据的介质。

计算机系统 200 一般连接到输入设备 228、显示设备 223 和光标控制器 236。输入设备 228 例如包括字母数字和其它键的键盘，向 CPU 204 传送信息和命令。显示设备 223 例如阴极射线管 (CRT)，向计算机系统 200 的用户显示信息。光标控制器 236 例如鼠标、跟踪球或光标方向键，向 CPU 204 传送方向信息和命令并控制显示设备 223 上光标的移动。

计算机系统 200 可以通过一个或多个网络与其它计算机和设备通信。例如，计算机系统 200 利用通信接口 220 通过网络 240 与另一个连接到打印机 248 的计算机 244 通信，或者通过万维网 252 与服务器 256 通信。万维网 252 一般也称作“因特网”。或者，计算机系统 200 还可以通过网络 240 与因特网 252 通信。

计算机系统 200 可以被用来实现上述的技术。在各种实施例中，CPU 204 通过执行装入 RAM 208 中的指令来执行上述技术的步骤。在备择实施例中，可以用硬导线连接的电路来取代软件指令或者与软件指令组合，以实现上述技术。因此，本发明的实施例决不限于任一软件、固件、硬件、电路或者它们组合。

CPU 204 所执行的指令可以存储在一个或多个计算机可读介质中或

者通过计算机可读介质传输，计算机可读介质指的是计算机能从中读取信息的任何介质。计算机可读介质例如可以是软盘、CD-ROM、CD-RAM、DVD-ROM、DVD-RAM、或任何其他光学介质、纸带、穿孔卡片、或任何其它有孔状图案(patterns of holes)的物理介质、RAM、ROM、EPROM、或任何其他存储芯片或盒式磁带。计算机可读介质可以是同轴电缆、铜线、光线、声波或电磁波、电容或电感耦合等等。举例来说，要被 CPU 204 执行的指令一开始以一个或多个软件程序的形式存储在通过总线 224 与计算机系统 200 相接合的 CD-ROM 中。计算机系统 200 把这些指令加载到 RAM 208 中，执行一些指令，然后把一些指令通过通信接口 220、调制解调器、电话线发送到网络，例如网络 240、因特网 252 等。远程计算机通过网络电缆接收数据，执行所接收的指令，并把数据发送到计算机系统 200，以便存储在存储器 216 中。

以上结合具体实施例说明了本发明。然而很明显，在不偏离本发明的精神和范围的情况下可以作出各种修改和改变。因此，具体说明和附图应被视为是解释性的而不是限制性的。

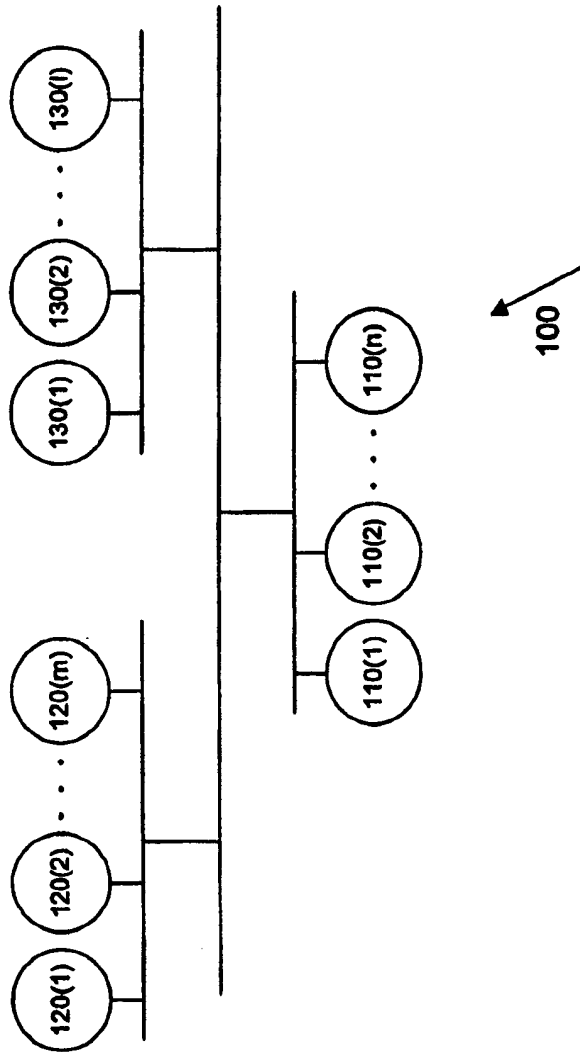


图 1

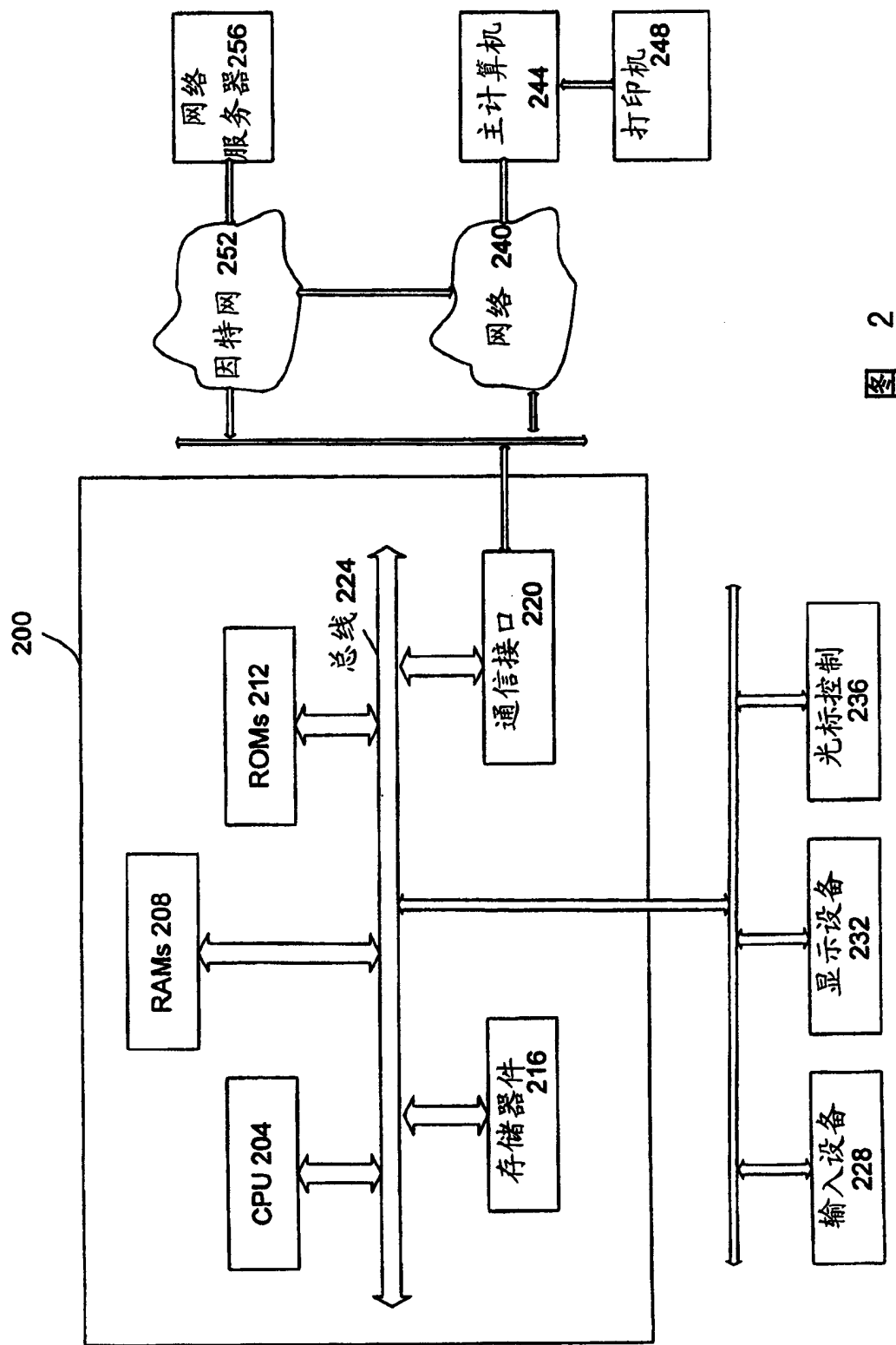


图 2