

[19] 中华人民共和国国家知识产权局

[51] Int. Cl<sup>7</sup>

G06F 9/30

G06F 9/38



# [12] 发明专利申请公开说明书

[21] 申请号 03156991.9

[43] 公开日 2004年4月21日

[11] 公开号 CN 1490717A

[22] 申请日 2003.9.17 [21] 申请号 03156991.9

[30] 优先权

[32] 2002.9.17 [33] US [31] 10/244,414

[71] 申请人 国际商业机器公司

地址 美国纽约

[72] 发明人 E·R·奥尔特曼 R·奈尔

J·K·奥布赖恩

K·M·奥布赖恩 P·H·奥登

D·A·普莱纳 S·W·萨特耶

[74] 专利代理机构 北京市中咨律师事务所

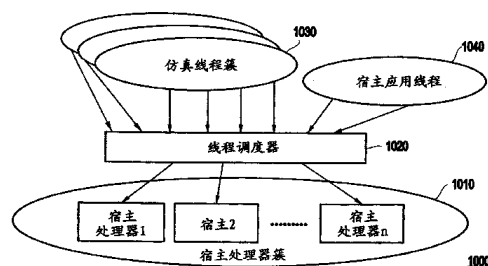
代理人 李 峥 于 静

权利要求书3页 说明书14页 附图9页

[54] 发明名称 在多处理器宿主系统上进行多处理器仿真的方法和系统

[57] 摘要

一种用于在宿主计算系统上执行为目标指令集体系结构编写的多处理器程序的方法(和系统),该宿主计算系统具有多个被设计来处理第二指令集体系结构的指令的处理器,该方法包括将被设计用来在目标计算系统的处理器上运行的程序的每个部分表示为将在宿主计算系统上被执行的一个或多个程序线程。



ISSN 1008-4274

1. 一种在一个宿主计算系统上执行为目标指令集体系结构编写的程序的方法，该宿主计算系统具有多个被设计用来处理一个第二指令集体系结构的指令的处理器，该方法包括：

将设计用来在目标计算系统的一个处理器上运行的该程序的每个部分表示为将在宿主计算系统上执行的一个或多个程序线程。

2. 根据权利要求1所述的方法，其特征在于，进一步包括：  
调度每个线程，以便在宿主系统的一个或多个处理器上执行。

3. 根据权利要求1所述的方法，其特征在于，进一步包括：

将目标系统的不是特定于目标系统的一个处理器的功能表示为将由宿主系统上的一个或多个处理器执行的一个或多个线程。

4. 根据权利要求2所述的方法，其特征在于，进一步包括：

将目标系统的不是特定于目标系统的一个处理器的功能表示为将由宿主系统上的一个或多个处理器执行的一个或多个线程。

5. 根据权利要求1所述的方法，其特征在于，线程的数量超过该宿主系统中的处理器的数量。

6. 根据权利要求1所述的方法，其特征在于，当一个给定指令组被解释了一个预先确定的次数之后，将该指令组从该第一个指令集转换到该第二个指令集。

7. 根据权利要求6所述的方法，其特征在于，该解释处理过程被作为多处理器系统中的一个线程执行。

8. 根据权利要求1所述的方法，其特征在于，在一个给定时间所述处理器的多个功能被表示为多个线程。

9. 根据权利要求1所述的方法，其特征在于，进一步包括：  
动态创建线程。

10. 根据权利要求1所述的方法，其特征在于，进一步包括：  
同时确定一个第一线程的转换和执行一个第二线程。

11. 根据权利要求1所述的方法，其特征在于，进一步包括：  
同时转换一个第一线程和执行一个第二线程。
12. 根据权利要求1所述的方法，其特征在于，进行所述线程到所述处理器的多对多映射。
13. 根据权利要求1所述的方法，其特征在于，当仿真线程正在被执行时，一个宿主应用正在运行。
14. 一种用于一个多处理器系统的仿真的方法，包括：  
将目标系统的多个部分表示为用于在一个宿主系统上执行的多个线程；以及  
调度所述每个线程，以便在所述宿主系统上执行。
15. 一种用于一个多处理器系统的仿真的方法，包括：  
将一个目标系统的多个硬件资源映射到多个软件线程；  
将所述多个软件线程映射到一个宿主系统的多个硬件资源；  
将该目标系统的状态信息映射到该宿主系统的一个存储器；以及  
通过将多个仿真任务划分为大量线程，而改善仿真的性能。
16. 一种用于一个多处理器系统的仿真的系统，包括：  
用于将一个目标系统的多个硬件资源映射到多个软件线程的装置；  
用于将所述多个软件线程映射到一个宿主系统的多个硬件资源的装置；  
用于将该目标系统的状态信息映射到该宿主系统的一个存储器的装置；以及  
用于通过将多个仿真任务划分为大量线程改善仿真的性能的装置。
17. 一种用于在一个宿主计算系统上执行为目标指令集体系统结构编写的程序的系统，该宿主计算系统具有多个被设计来处理一个第二指令集体系统结构的指令的处理器，该系统包括：  
一个表示单元，用于将被设计用来在该目标计算系统的一个处理器上运行的程序的每个部分表示为将在该宿主计算系统被执行的一个或多个程序线程。

18. 一种用于一个多处理器系统的仿真的系统，包括：

一个第一映射模块，用于将一个目标系统的多个硬件资源映射到多个软件线程；

一个第二映射模块，用于将所述多个软件线程映射到一个宿主系统的多个硬件资源；

一个第三映射模块，用于将该目标系统的状态信息映射到该宿主系统的一个存储器；以及

一个划分模块，用于将多个仿真任务划分为大量线程。

19. 一种用于多处理器系统的宿主计算系统的线程处理结构，包括：

一个用于保持多个线程的线程池；

一个线程处理器，用于访问所述宿主计算系统的存储器，并决定选择该线程池中的哪个线程以进行仿真；

一个线程生成器，用于创建新线程，并将所述新线程放入所述线程池中；以及

一个线程调度器，用于调度保持在所述线程池中的所述多个线程，所述调度器扫描等待的多个线程，并按照优先次序将下一个线程分配给所述多处理器系统的一个可用处理器。

20. 一种以有形的方式体现了机器可读指令的程序的信号承载介质，该程序可由数字处理设备执行，以实行在一个宿主计算系统上执行为目标指令集体系统结构编写的程序的方法，该宿主计算系统具有多个被设计来处理一个第二指令集体系统结构的指令的处理器，所述方法包括：

将被设计用来在一个目标计算系统的一个处理器上运行的程序的每个部分表示为将在该宿主计算系统上被执行的一个或多个程序线程。

## 在多处理器宿主系统上进行多处理器 仿真的方法和系统

### 技术领域

本发明一般地涉及计算机系统，具体地涉及一种在一个多重处理计算机系统（multiprocessing computing system）上再现另一个多重处理计算系统的行为的方法（和系统）。

### 技术背景

人们很早就认识到，需要在一个计算机系统中仿真（emulate）另一个计算机系统的行为。为此提出过几种方案。在作为参考包含于此的美国专利 No. 5,832,205 中有对这些技术的综述。

美国专利 No. 5,832,205 的解决方案包括一个组合的硬件/软件方案，它在一个处理器上进行对另一个处理器的指令集的仿真。该方案允许硬件设计纳入有助于目标指令集的执行的特征。然而，由于同一个原因，它无法同样有效地仿真所有系统。

作为参考包含于此的 SimOS（例如，参见 Stephen A. Herrod, “Using Complete Machine Simulation to Understand Computer System Behavior”，博士论文，斯坦福大学，1998 年 2 月）和 SimICS（例如，参见 Peter S. Magnusson, “A Design For Efficient Simulation of a Multiprocessor”，第一届国际计算机和电信系统的建模、分析和仿真（MASCOTS）研讨会会议记录，La Jolla，加利福尼亚，1993 年 1 月，69-78 页）是无需特殊硬件特征而能进行仿真的例子。然而，它们的性能却不如美国专利 No. 5,832,205 的有效。

一般而言，这些系统使用了多级转换。已有人描述过这样的技术（例

如，参见 Jim Turley, “Alpha Runs x86 Code with FX!32”, 1996 年 3 月 5 日, Microprocessor Report), 其中转换的程度(范围)根据代码执行的程度(范围)而变化。

然而,今天的计算机系统包含不止一个处理器(如与单处理器系统相对的多处理器系统)。传统技术尚未很好地解决这些多处理器系统的仿真。

另外,除了在这样的系统上仿真多个处理器以外,其它方面需要仿真的有处理器之间的各种形式的通信,以及决定多个处理器对存储器位置访问顺序的规则。

SimOS 和 SimICS 都试图仿真多处理器系统的行为。但是它们没有使用多处理器系统作为宿主(host)计算系统。

因而,传统技术没有解决在多处理器系统中的多处理器仿真的问题。

这就是说,传统技术(及指令集结构)常局限于(并指的是)被仿真单个处理器系统,而今天的大多数系统是多处理器系统,尤其大系统(如超出个人计算机(PC)领域之外的)更是如此。因此,被用于以一个处理器简单地仿真另一个处理器的技术在多处理器系统环境中无法使用。就是说,当有多个处理器存在时,为单处理器系统设计的传统仿真技术不起作用。

## 发明内容

鉴于传统方法和结构的上述及其它问题、障碍和缺点,本发明的一个目标是提供一种方法和结构,其中利用某种处理器指令集和存储器结构,使得一个多处理器系统可以有效地仿真使用其它处理器指令的另一个多处理器系统的行为。

根据本发明的第一个方面,一种用于在具有被设计用来处理第二指令集结构的指令的多个处理器的宿主计算系统上执行为目标指令集结构所编写的程序的方法(和系统),包括,将设计用来在目标计算系统的处理器上运行的程序的每个部分表示为将在宿主计算系统上执行的一个或多个程

序线程 (thread)。

根据本发明的第二个方面, 一个系统 (和方法), 包括: 将目标系统的硬件资源映射到软件线程的装置; 将线程映射到宿主系统的硬件资源的装置; 将目标系统的状态信息映射到宿主系统的存储器中的装置; 以及通过将仿真任务划分为大量线程而提高仿真性能的装置。

根据本发明的第三个方面, 一种多处理器系统的宿主计算机的线程处理结构, 包括: 一个线程池, 用于保持线程; 一个线程处理器中, 用于访问宿主系统的存储器, 并决定从线程池中选择哪个线程用于仿真; 一个线程生成器, 用于创建新线程, 并将所述新线程放入线程池; 以及一个线程调度器, 用于对线程池中保持的线程进行调度, 该调度器扫描正在等待的线程并按照优先级次序将下一个线程分配给可用的处理器。

根据本发明的第四个方面, 一种以有形的方式体现机器可读指令的程序的信号承载介质, 该程序由数字处理设备执行来完成在具有被设计用来处理第二指令集结构的指令的多个处理器的宿主计算系统上执行为目标指令集结构而编写的程序的方法, 该方法包括把设计用来在目标计算系统的一个处理器上运行的程序的每一部分表现为在宿主计算系统上执行的一个或多个程序线程。

使用本发明的独有的和非显而易见的方面, 对具有多个处理器的系统的仿真可以被高效地进行。进一步地, 本发明使用宿主计算机进行该仿真。

此外, 本发明的一个关键特征是摆脱了宾客系统 (guest system) 是一件硬件的概念。代之以, 本发明将宾客系统当作一个软件。

这样, 宾客 (guest) 被更抽象地视为具有多个并行的、需要执行的线程, 然后这些线程被映射到宿主机的硬件资源中。这从本质上消除了通常把程序中的并行性映射到宾客的硬件、然后宾客的硬件由宿主 (机) 的硬件仿真的中间步骤。本发明消除该中间步骤, 即使知道在宾客上可能实际上存在硬件, 其可能已映射到宿主的硬件。

因而, 本发明消除了将宾客 (机) 的应用的软件线程映射到宾客的硬件的步骤。其后, 宾客的这些线程中的每一个都被调度, 用于在宿主的一

个或多个处理器上运行。

本发明的另一个优点是建造和调试这样一个系统更容易了，因为不再需要操心使宾客机的硬件细节正确。

#### 附图说明

通过下面参照附图对本发明的优选实施例的详细描述，会使本发明的上述和其它的目的、方面和优点得到更好的理解，在附图中：

图 1 示出一个目标多处理器计算系统 100，包括多个处理单元、一个存储器子系统、一个一致性总线（coherent bus）互连，和一个输入/输出（I/O）处理器；

图 2 示出在图 1 所示系统 100 上执行的各种指令的一个分类方案 200；

图 3 示出一个宿主多处理器计算系统 300，包括多个宿主处理单元，一个宿主存储器子系统，一个一致性宿主总线互连，和一个宿主 I/O 处理器；

图 4 示出将目标系统 100 中的各种资源映射到 400 宿主系统 300 的存储器中；

图 5 示出宿主系统 300 上的一个线程处理软件结构 500；

图 6 示出一个系统，其中存储器访问使用特定于线程的存储器可以被更快地进行；

图 7 示出一个系统 700，用于将目标系统的功能简单地映射到宿主系统的各线程中；

图 8 示出一个系统 800，用于从图 7 所示线程到多重处理宿主系统中的处理器的普通映射（trivial mapping）；

图 9 示出一个系统 900，用于将图 7 所示线程更有效地映射到多重处理宿主系统中的处理器；

图 10 示出一个更一般的系统 1000，其可以被映射到一个多重处理宿主系统上；

图 11 示出一个仿真方案 1100，它将转换（结果）缓存以便重用（reuse）。



图 12 示出一个系统 1200，用于产生并行转换线程；

图 13 示出一个系统 1300，它是图 9 的增强，以容纳附加的转换线程；  
以及

图 14 示出一种信号承载介质 1400（例如存储介质），用于保存按照本发明的程序的步骤。

### 具体实施方式

现在参照附图，更具体地说参照图 1 至 14，其中示出了根据本发明的方法和结构的优选实施例。

图 1 示出了一个用于被仿真的一般多重处理系统 100。它包括多个处理器 110A 至 110D，其中每一个都可能有自己的局部高速缓存（local caches），并通过某互连网络 120 与存储器分层体系结构（memory hierarchy）130 连接，该存储器分层体系结构可能包括由主存储器（未示出）支持（备份）的附加的高速缓存级别。该系统也可以通过 I/O 处理器 140 访问包括盘（磁盘、光盘）和通信网络的 I/O 设备，该 I/O 处理器将来自系统的进向请求格式化为设备能理解的形式。显然，该系统不限于如图所示的 4 个处理器等，而是事实上可以使用任何数量的处理器等。

图 1 中的每个处理器 110A-110D 都可以被看作好像正在执行影响到系统 100 的状态的指令。每个指令的效果可按如图 2 的方案 200 所示进行分类。

例如，一个指令可以根据其是否只影响执行它的处理器的局部资源还是影响在所有处理器中所共享的资源，而宽泛地划分为“局部资源指令”或“共享资源指令”。局部资源的例子有每个处理器局部的通用寄存器、浮点寄存器、处理器状态寄存器和控制寄存器。共享资源可以包括存储器和 I/O 设备。

对共享资源的仿真必须格外小心，因为多个处理器可能试图在一个给定的期间访问这些资源。重要的是，在仿真的系统中共享资源的访问的顺序应当与被仿真的系统中可能发生的顺序相同。

为了对此进行有效的管理，共享资源指令进一步被划分为 (a) “排它指令”，意即它们访问由执行处理器独占（排它）使用的共享资源，(b) “共享读指令”，意即该指令所使用的共享资源只被读取而不被改变，以及 (c) “通信指令”，其包括所有其它共享资源指令。之所以称为通信指令，是因为它们通常被用于从一个处理器到另一个处理器之间传递信息，如一个处理器写入一个值，而另一个或多个处理器读取该值。

图 3 显示了一个希望在其上进行仿真的宿主系统 300。它在物理结构上与图 1 的目标系统 100 相似，尽管某些细节可能不同。

例如，宿主处理器的指令集可能与目标处理器的不同。处理器 310A 至 310D 的数量和存储器大小也可能不同。访问共享资源的互连网络 320 在形式上和功能上都可能不同。我们将假设宿主系统被配置为 SMP (symmetric multiprocessing, 对称多重处理) 配置。这一点隐含的意思之一是宿主 (机) 中的所有处理器 310A 至 310D 都将访问相同的存储器，并在访问存储器位置时具有大体相似的延迟。图中也示出了宿主 I/O 处理器 340，它与图 1 所示处理器 140 相似。

目标系统中每个资源的状态是通过分配宿主系统 300 的存储器的区域而被建模的。这里假设仿真是在共享虚拟存储器操作系统环境中进行的。这提供了可以容纳目标系统的所有真实资源的宿主存储器大小 (尺寸)。

图 4 示出了宿主 (机) 300 的虚拟存储器 (VM) 400 的分解，它仿真了目标系统 100 的各种资源。VM 400 包括共享资源 410 (如目标真实存储器)、处理器局部资源 420 (如通用寄存器、浮点寄存器、程序计数器、控制寄存器等)、I/O 局部资源和仿真程序存储器。

我们注意到，64 位虚拟寻址宿主 (机) 完全可以适应数十甚至数百千兆字节的真实存储器，以及数百个处理器的局部资源，而仍然有足够的可用寻址能力用于仿真程序本身。

除了共享存储器 SMP，我们还将假设操作系统支持多线程。此种支持的一个例子是 Unix® 操作系统中的 p-threads 软件包 (例如，参见 Bradford Nichols 等人的 “Pthreads Programming: A POSIX Standard for Better

Multiprocessing,” (O’Reilly Nutshell), 1996 年 9 月)。这样的软件包 (package) 允许在软件中创建并行执行的多个程序流, 同时也允许在这些程序流之间安全地共享变量。此外, 这些软件包通常也提供了工具 (手段) 来繁育 (spawn) 新线程, “杀死” (终止) 线程, 以及中断或唤醒线程。

我们注意到, 存在如图 1 至 4 所示的系统, 本发明可以示例性地在其上实施。确实, 本发明的一个目的就是实施本发明, 而无需更改宿主系统中的物理硬件 (如插入任何硬件修改), 从而根据本发明进行仿真。

图 5 示出了根据本发明的一个线程处理系统 500。系统 500 包括一个线程处理器 510、一个线程生成器 520、一个线程池 530 和一个线程调度器 (scheduler) 400。

如图 5 所示, 线程处理器 (引擎) 510 决定从线程池 530 中选择哪个线程用于仿真, 以此处理 (调度) 线程池 530 中保持的线程。

在处理线程的操作中, 有时线程处理器决定必须创建一些新线程。因而, 线程被线程生成器 520 创建, 并被放置在线程池 530 中。线程调度器 540 扫描等待的线程, 并按优先次序将下一个线程分配给一个可用的处理器。

线程在一个处理器中的执行涉及到读宿主虚拟存储器 400 中的某些位置, 并修改这些或其它位置。因此, 线程处理器与宿主虚拟存储器 400 交互, 从而也把宾客系统的存储器映射到宿主的存储器中。对于该线程处理器可用的只是宿主虚拟存储器, 因此会发生这种映射。

我们注意到, 如果图 5 所示的模型不可用, 则在传统系统中所发生的将是事先确定什么线程存在 (例如, 假设每一个宿主处理器是一个线程), 然后使用宿主上可用的 (线程) 进行一到一的映射。如上所述, 这种技术存在许多问题。

因此, 本发明使用了创造性的线程处理器 500, 以决定需要创建和调度什么线程。

假设一个大的线程池 500, 随着可用来处理这些线程的处理器数量的增加, 系统的效率提高。然而, 这种效率可能会受到如图 2 中所定义的

通信类型的指令的数量的限制。

即使目标系统的整个真实存储器是共享的，也常常能够将该存储器进一步划分为三个类别，这三个类别与图 2 所示共享资源指令的三个子类相对应。

这些区域是 (a) “排它访问区域”，(b) “只读区域”，和 (c) “其它共享区域”。排它访问区域是只能由单个线程访问的那些区域。只读区域可以被多个线程访问，但从来不被修改。因此，复制这些区域，并将一个拷贝（复本）包括作为该线程的局部排它区域的一部分是允许的。

其它共享区域应当以不同方式对待。例如，如果为了有效的局部访问而制作了复本，则重要的是一个线程所作的更改应当正确地传达给所有其他可能正在访问或将在未来访问该相同区域的线程。

图 6 示出了使用特定于线程的存储器 610 如何可以使存储器访问的速度更快。就是说，线程处理器 510 可以访问线程指定（局部）存储器而进行快速访问，而宿主虚拟存储器的共享部分是以受保护访问方式被访问的。

因此，为了提高效率，存储器可以被划分为多个（例如两个）部分。第一个部分是通信最少的部分，而第二个部分是在线程之间有大量通信的部分。

因而，如果存储器的一些部分是专用于每个线程的，则可以使这些部分成为快速访问存储器，而需要彼此“对话”（例如，需要被共享）且可能不需要快速访问（例如，由于必须检查它们的权限，以确定是否允许这种访问等）的线程则被形成宿主虚拟存储器的一个共享部分。因此，通过把存储器划分为两个部分，就可实现更快的总的存储器访问速度。

图 7 示出了把仿真一个目标多重处理系统的任务（功能）直接映射 700 到宿主系统的线程。

每个处理器 710A、710B、710C、710D 等及其资源都分别被仿真作为线程 720A、720B、720C、720D 等。同时示出的还有 I/O 处理器 730 和 I/O 线程 740。显然，本发明不要求处理仅仅是与传统意义上的处理器有关的处理，而是也包括 I/O 处理器、IBM 390® 系统的通道、某些系统中的协

处理器等。

另外，提供了一个系统线程 750，它包括目标系统的所有不是特定于处理器的功能 760，以及所有仿真系统本身的功能，包括处理线程创建、线程间通信、调试和性能监视方面的任务。

我们注意到，图 7 所示概念可以在单个处理器上实现，其中一个单个处理器处理来自宿主的（多个）线程。就是说，使用单个处理器，可以认为该概念是一个多重程序设计系统（multiprogramming system），其中在单个处理器上不断地发生各种线程之间的切换。仿真系统本身处于包含前述线程软件包的共享存储器 SMP 操作系统之下。

图 8 示出了一个多处理器系统，其中的线程软件包可被编写为将每个线程 820A、820B、820C、820D 等映射到宿主处理器 810A、810B、810C、810D 等中的一个（与上述单处理器的情况形成对比）。此外还示出了 I/O 线程 840 被映射到宿主处理器 810E，以及系统线程 850 被映射到宿主处理器 810F。

图 8 所示方法的优点是，进行仿真的各宿主处理器之间的物理通信限于各线程本身之间所发生的通信。由于线程紧密地映射到该被仿真的目标系统的结构，因而宿主的通信行为与目标系统的通信行为是相似的。

然而，伴随图 8 所示此种方法的一个缺点（例如，该方法暗示了线程和宿主处理器之间的一对一的关系）是宿主系统有可能未充分利用。就是说，在每个宿主处理器都分别专用于一个单个线程的系统中，如果被仿真的目标处理器中的一个处于空闲状态，则相应的宿主处理器也不会被充分利用。

该技术的另一个缺点是其可伸缩性。如果宿主系统具有比目标系统多很多的处理器，则很多额外的处理器就不能被适当利用。相反，如果宿主系统具有较少的处理器，则线程到处理器的一对一映射将只能通过将多个目标处理器映射到相同线程来实现。

图 9 示出了一个系统 900，它提供了避免上述某些问题的解决方案，其中包括一个宿主处理器簇（cluster）910、一个线程调度器 920、一个仿

真线程簇 930。正如下面将要讨论的，由于图 9 所示系统采取行动平衡各宿主处理器的负载，更高的效率得以产生。确实，可能会有某些期间，其中某些处理器完全空闲而某些则完全超负荷。包括线程调度器 920 的图 9 所示系统用于使负载平滑（平衡）。更具体地说，线程调度器 920 决定何时把哪个线程放在何处（例如哪个宿主处理器），从而使负载均衡最优化。

这样，如前所述，当线程的数量增多时，动态线程映射相对于静态映射的优势将增大。因而，有可能重新构造仿真系统，以提供更多的并行线程，而不是如图 7 所示的每个处理器一个线程的方案。

图 10 示出如何把一个更一般的系统 1000 映射（仿真）到多重处理宿主系统上。系统 1000 可以包括一个宿主处理器簇 1010、一个线程调度器 1020、一个或多个仿真线程簇 1030 和多个宿主应用线程 1040。

就是说，本发明不仅可用于多重处理主机，也可用于如图 10 所示类似于 IBM 390<sup>®</sup> 系统的主机，其中有一个簇配置，它具有多个不同的相互之间通信的多处理器。因此，即使这样的系统也可以在上述类型的多重处理宿主上被仿真。

此外，本发明系统不仅限于仿真。就是说，该特定系统不仅限于进行仿真，而是例如，应用（如 Linux）可以直接在该宿主上运行，此时应用并没有处于仿真之中而是在宿主上作本机运行。在这种情况下，宿主应用线程 1040 可以在宿主上运行，并由线程调度器 1020 调度/管理（该线程调度器也管理仿真线程簇的线程）。因此，本发明不仅对于仿真，而且对于直接在宿主上运行应用都有极大的用处。

我们注意到，Herrod 的关于 SimOS 的上述文章和美国专利 No. 5,832,205 已指出，仿真系统的性能可以通过如下进行缓存而得到极大提高。

就是说，如果预期一组指令将被执行数次，则首先将其由目标指令集转换为宿主指令集，然后将转换结果（translation）保存在一个称为“转换高速缓存”的特殊存储器区域中。当此后遇到此组指令的地址时，则直接执行本机宿主指令。通过避免重新取目标指令和重新转换这些指令，指

令组的执行显著加快。通过分析这组指令并使产生的转换结果最优化，可以获得进一步的益处。此种最优化的一些例子可参见作为参考包含于此的美国专利 No. 5, 832, 205。

通过缓存转换结果而获得的益处既依赖于转换指令组所需要的努力，也依赖于被转换的（指令）组最终被执行的次数。由于对大多数代码而言，后者是无法预测的，因而使用了试探法（heuristics）来决定潜在的转换候选对象。

图 11 显出了执行以上操作的一个方法 1100，一个简单的试探法是记录一个给定指令组过去执行的次数，并在该计数超过预定阈值时转换该（指令）组。

图 11 示出一个转换表 1110，它是由下一次被执行的指令组的地址索引的。

在步骤 1120 中，如果在转换表 1110 中有相应于此地址的有效的条目，则它指向已转换的本机指令应当从中读取并执行的位置。

如果没有如步骤 1120 所确定的有效条目，则目标指令直接被解释（步骤 1130），并且与该指令组相关联的计数器被递增（步骤 1140）。

如果该计数器超过了一个阈值（步骤 1150），例如如果该组已被解释了 5 次，则该指令组即被调度进行转换（步骤 1160）。

如果在步骤 1120 中，确定该指令已经被转换（即判断结果为“是”），则通过访问转换高速缓存 1175，在步骤 1170 中执行缓存的指令组转换结果。

然后，在步骤 1180 中，确定下一个要被仿真的指令组。

在如美国专利 No. 5,832,205 所描述的系统，被仿真的线程执行转换或是在发现临界条件满足之时，或是在下次执行程序组之前。线程执行转换所花费的时间本来可能被用于开始执行下一组指令，因此它代表了系统的一种开销。

一个更有效的方法是线程简单地把指令组放入转换池（如 1190），并继续下一组指令的执行。这样，当转换已经被完成时，转换的目标（object）

被放入转换高速缓存，带有转换表 1110 中的一个指针，正如下面将讨论的图 12 所示的。

图 12 示出了用于产生并行转换线程的系统 1200。

在图 12 中，另一个称为转换池管理器（即图 12 所示的 1210）的线程，监视转换池（图 11 所示的 1190），以挑选需要转换的指令组，并独立于处理器线程的执行。进一步讲，转换池管理器 1210 本身也不需要执行转换。

由于转换一组指令的处理过程大体上独立于转换另一组指令的处理过程，转换池管理器 1210 可以繁育（spawn）几个线程，其中每个线程都对来自转换池的一组指令执行转换，如图 12 所示。

在图 12 中，转换池管理器 1210 从转换池 1190 中选择一个指令组进行转换。转换池管理器 1210 更新转换表 1110，然后轮流将线程提供给转换线程调度器 1220。转换线程调度器调度转换线程 1230A、1230B、1230C、1230D 等，并将它们写入转换高速缓存 1240。

此系统/操作的效果本质上是将仿真任务进一步划分为可由宿主多重处理系统更好地利用的独立平行线程。因而，图 12 的系统利用了转换特性，把它纳入其中，并将它映射到本发明系统的框架中。因而，这种缓存转换结果（并实际上执行该转换）的特性加强了由本发明系统进行的仿真（以及所处理的线程类型）。

图 13 示出了系统 1300，它包括一个宿主处理器簇 1310、一个线程调度器 1320 和一个仿真线程簇 1330。图 13 是图 9 的修改视图，它具有可以提高宿主系统的性能和利用率的附加的线程。这种附加线程的益处进一步延伸到如图 10 所示的多簇大型机仿真系统。因而，尽管图 13 为了清楚和简洁的缘故，只示例性地示出了一个仿真线程簇，可以如图 10 所示提供多个这样的仿真线程簇，以及如图 10 所示的各宿主应用线程 1040。

除了如上所述的硬件/软件环境，本发明的另一个不同方面包括一个由计算机实施的、用于完成上述方法的方法。作为一个例子，该方法可以在如上讨论的特定环境中实施。

例如，这样一种方法可以通过操作多处理器系统中的、如数字数据处



理器所体现的计算机，以执行机器可读的指令序列而实现。这些指令可以存在（驻留）于各类信号承载介质中。

因此，本发明的这个方面被指向编程产品，包括以有形的方式体现机器可读的指令的程序的信号承载介质，该指令程序可以由包含处理器/中央处理单元（CPU）和上述硬件的多重处理系统中的一个或多个数字数据处理器执行，以实施本发明的方法。

例如，该信号承载介质可以包括 CPU 中包含的 RAM，如由快速访问存储装置所代表的。可替代地，该指令也可以被包含在其它信号存储介质中，如包含在可由 CPU 直接或间接访问的磁性数据存储盘 1400（图 14）中。

不管是包含在磁盘 1400 中，还是包含在计算机/CPU 中，或包含在其它地方，该指令可以被存储在各种机器可读的数据存储介质中，如 DASD 存储（如传统的“硬盘驱动器”或 RAID 阵列）、磁带、电子只读存储器（如 ROM、EPROM 或 EEPROM）、光存储设备（如 CD-ROM、WORM、DVD、数字光带等）、纸质“穿孔”卡或其它适当的信号承载介质，包括传输介质，如数字的和模拟的和通信链路和无线链路。在本发明的一个说明性实施例中，机器可读的指令包括软件目标代码，该目标代码是由诸如 C 等语言编译而成的。

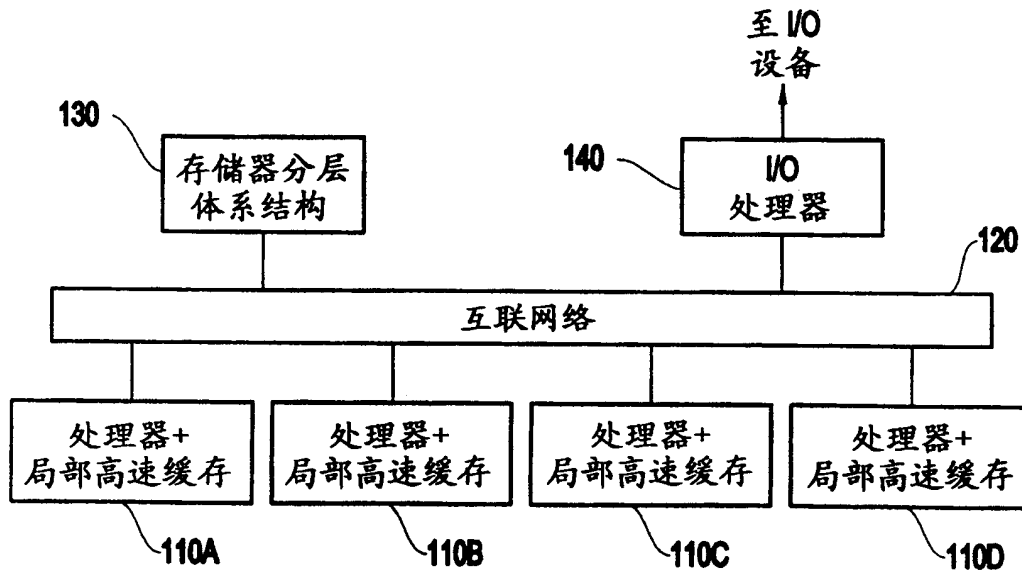
尽管本发明被按照几个优选实施例进行了说明，但本领域的专业人士能够认识到，实施本发明时，可以在所附权利要求的精神和范围内进行修改。

我们注意到，如上所述，本发明有很多益处，包括有效的仿真。此外，本发明可以被用作进行有助于负载均衡的“虚拟化”（virtualization）的基础。虚拟化可以采取各种形式，包括负载均衡。例如，虚拟化也可用于容错，其中在一个具有一对一映射（或其它映射方案）的系统中，如果一个处理器发生故障，系统仍可以继续工作，因为发生故障的系统可以得到抽象，并可以有一个较小的宿主处理器池。因而，另一个处理器可以承担起发生故障的处理器职责。

本发明的另一个可能应用是在节能方面。就是说，当确定在多处理器系统中消耗了过多能量后，某些处理器可以关闭，而整个仿真系统的功能可以在较低的能量水平上维持。在这种情况下，被关闭的processors的功能可以被转移到另一个处理器上。通过本发明做到这一点比较容易，因为每个处理器被视为一个线程，而不是线程到处理器的一对一映射。

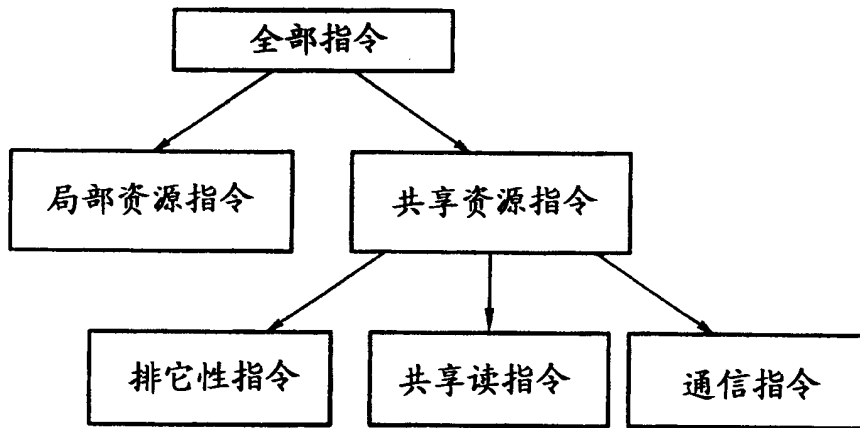
就是说，本发明的方案是多对多映射，而不是上述 Herrod 文章中 SimOS 技术的多对一映射，或美国专利 No. 5,832,205 的一对一映射方案。

此外，应当注意，申请人的意图是涵盖所有权利要求元素的等同物，即在获得权力的过程中进行了修改。



100

图1



200

图2

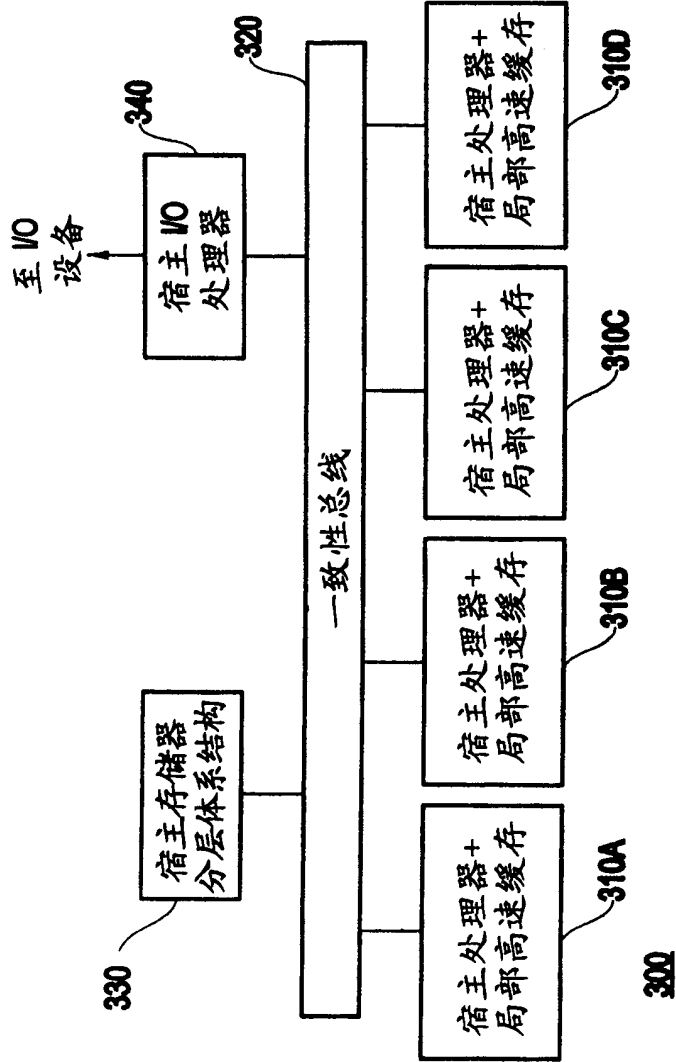


图3

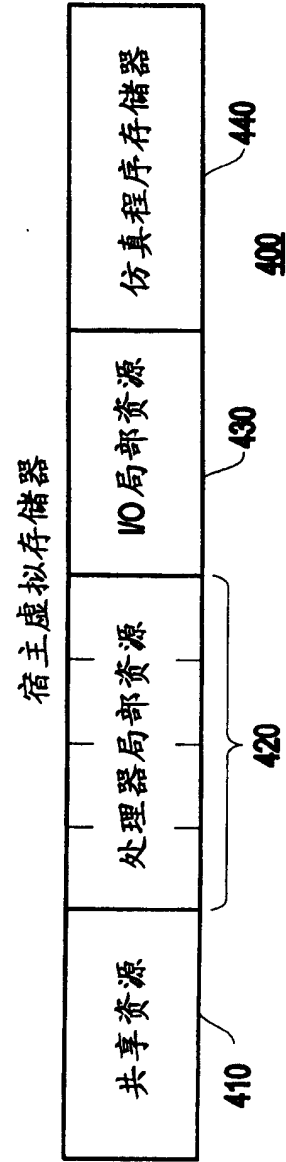


图4

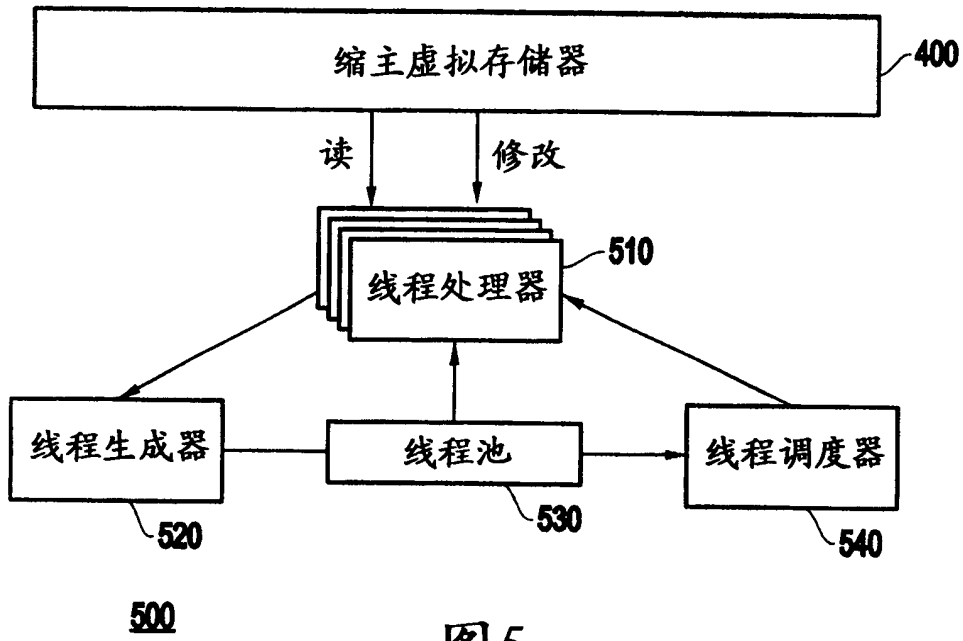


图5

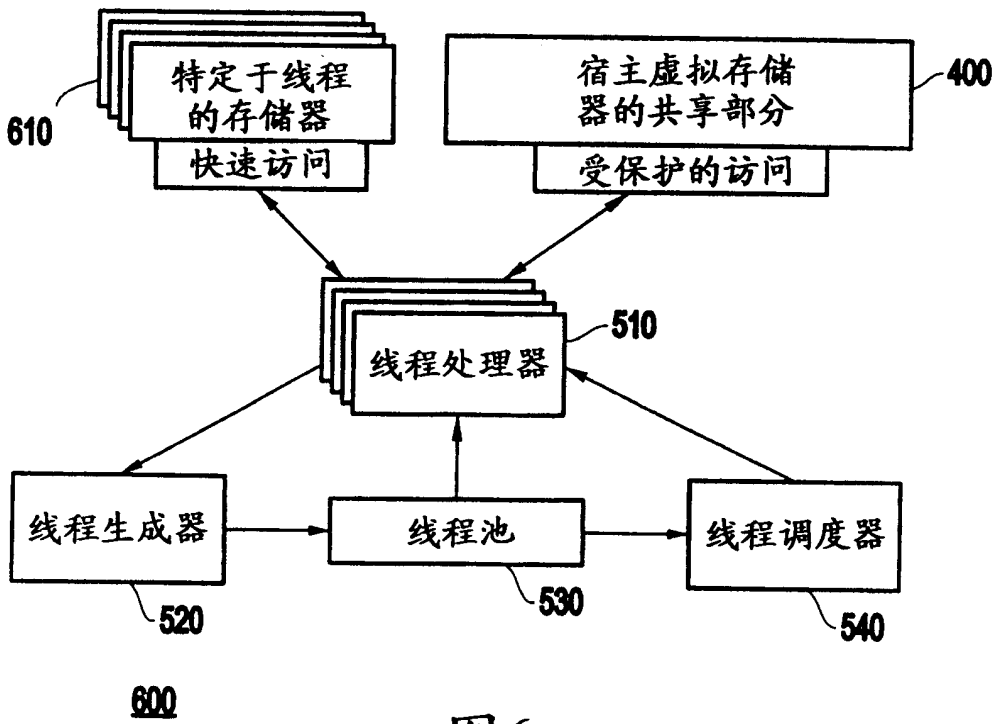
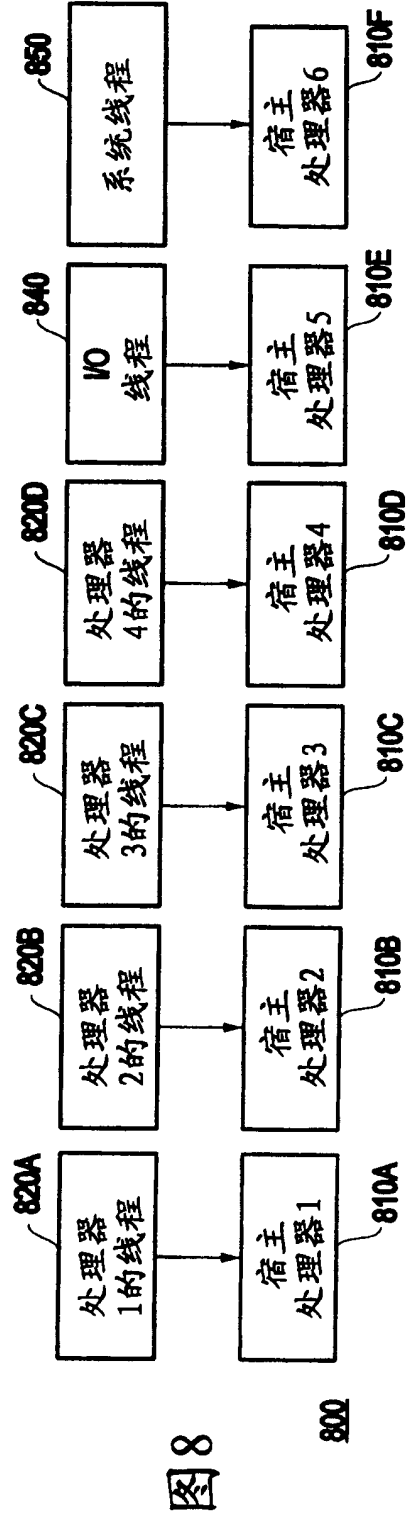
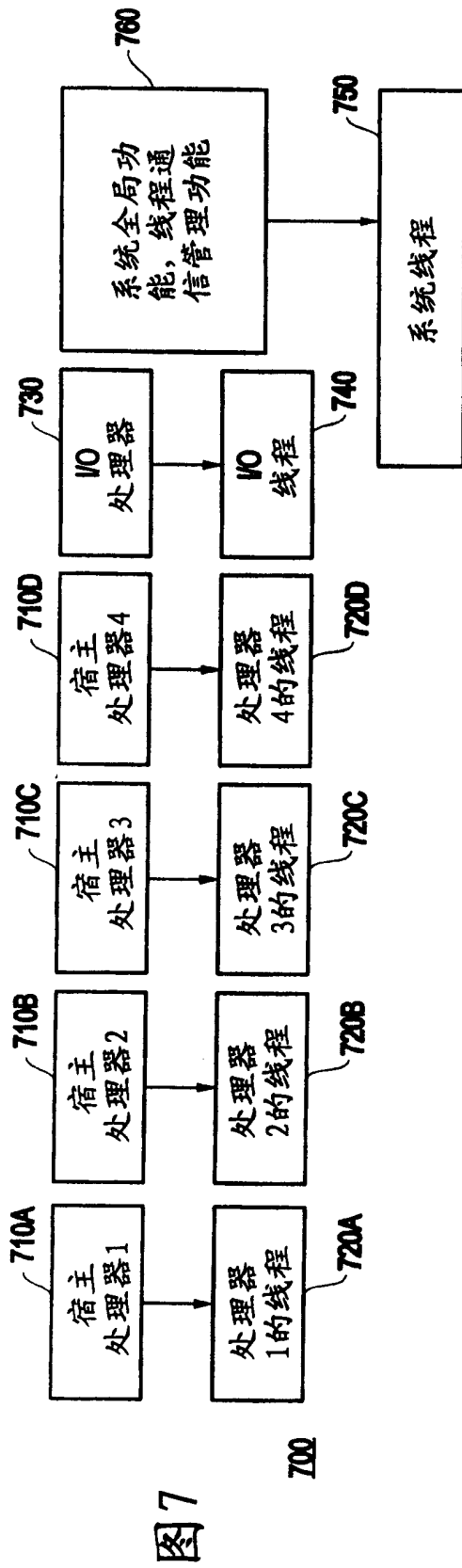


图6



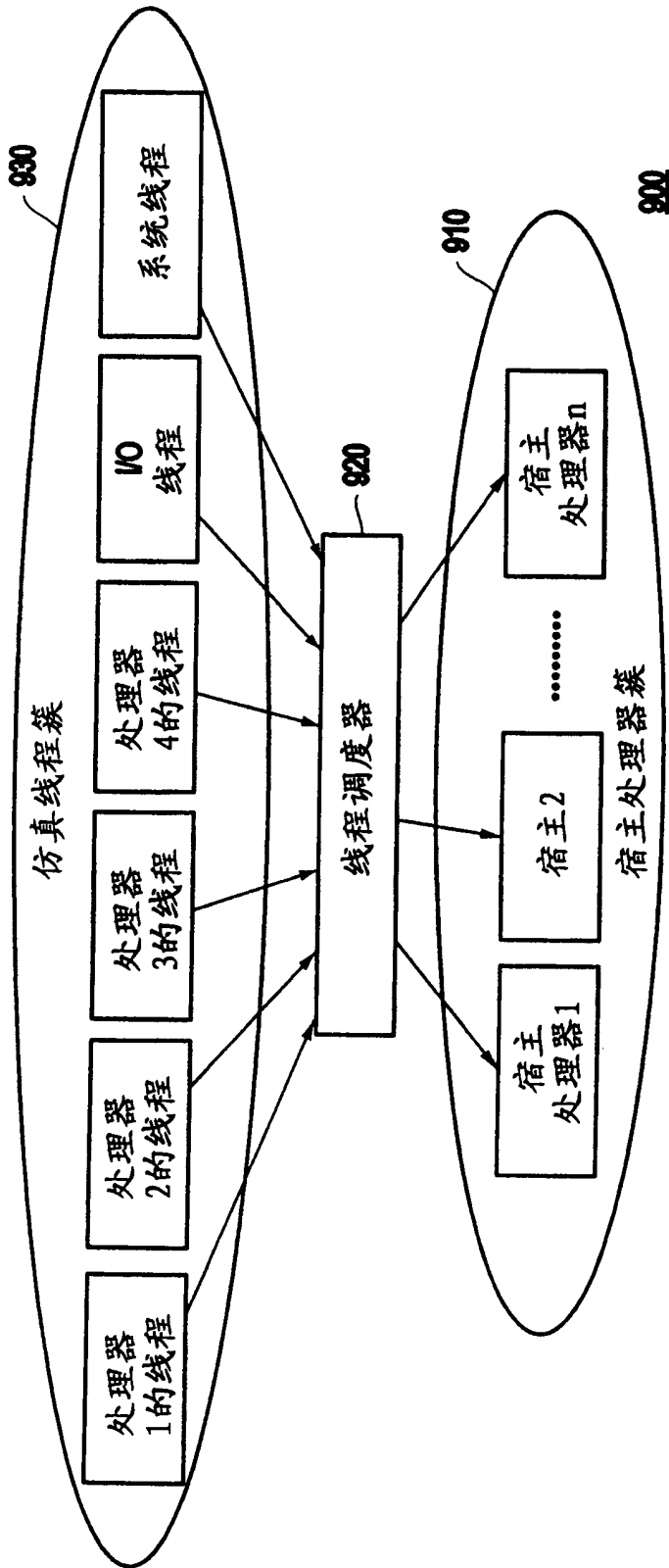


图9

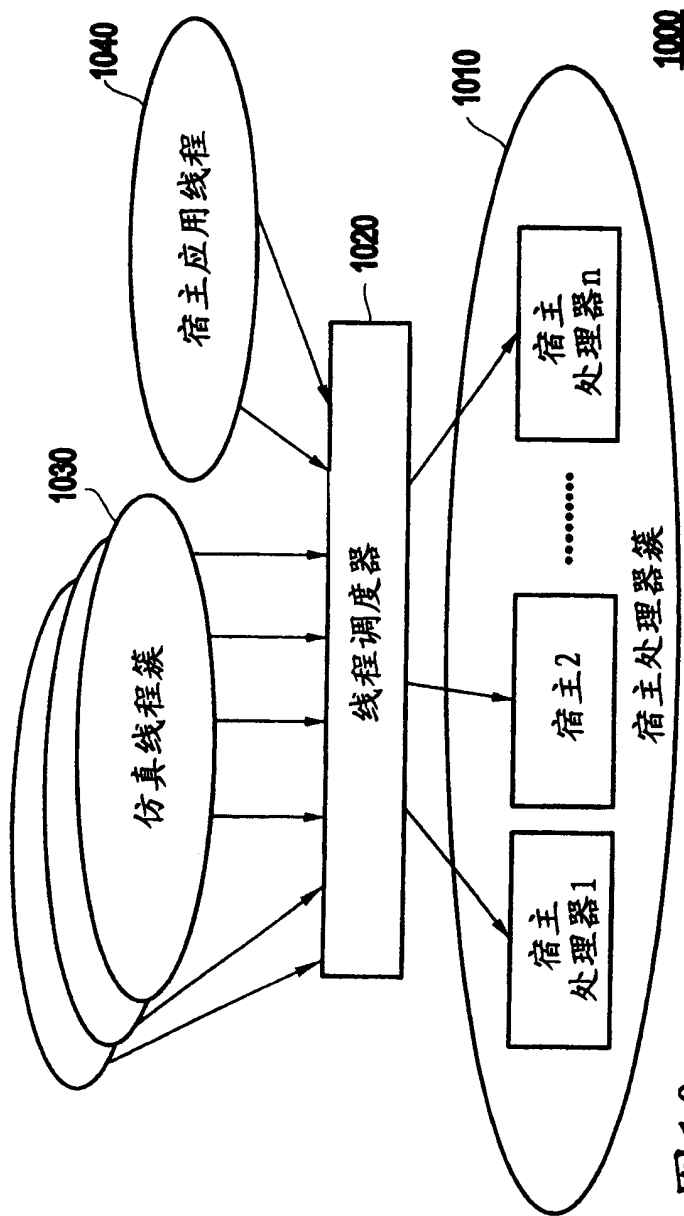


图10



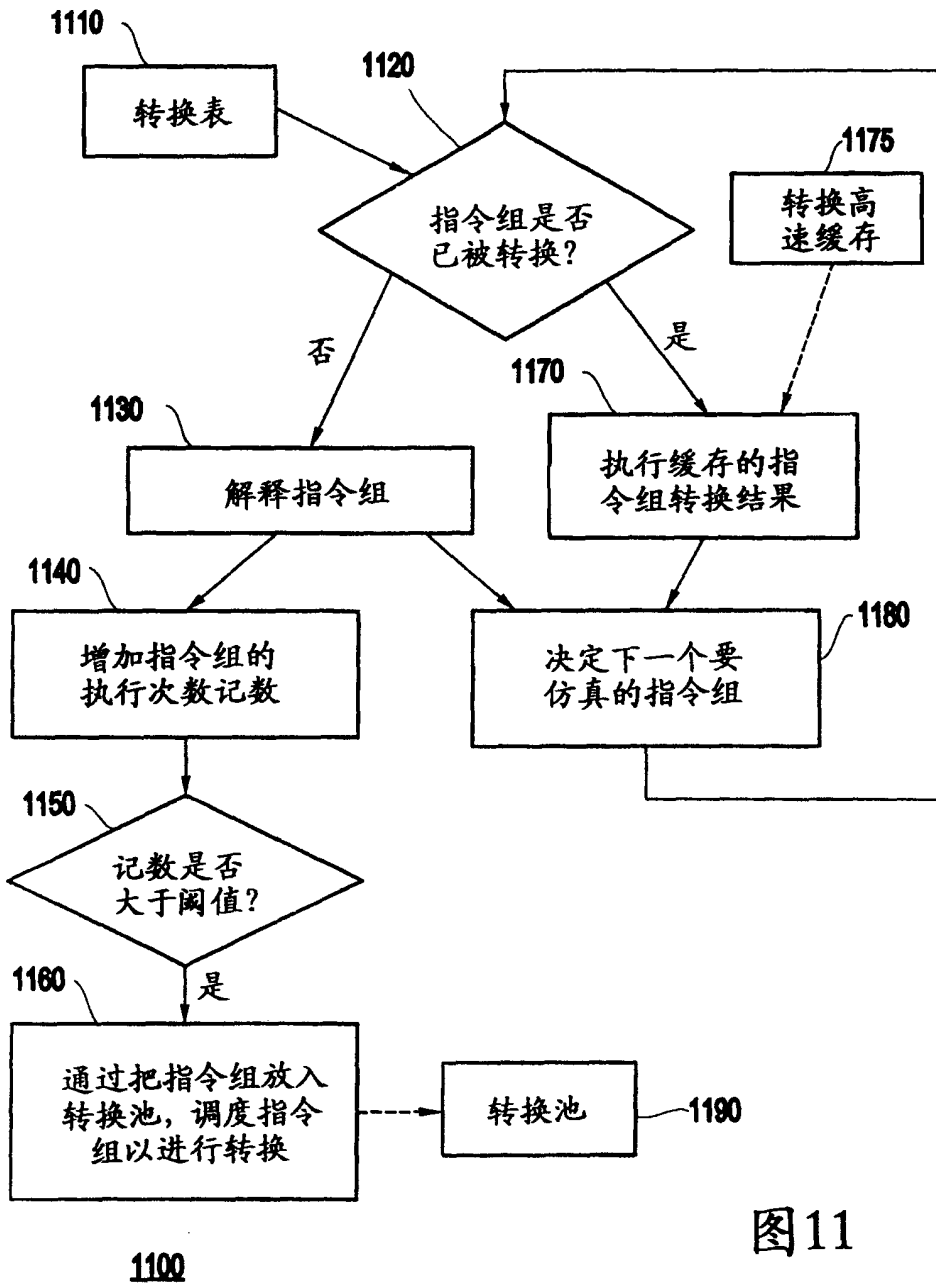


图11

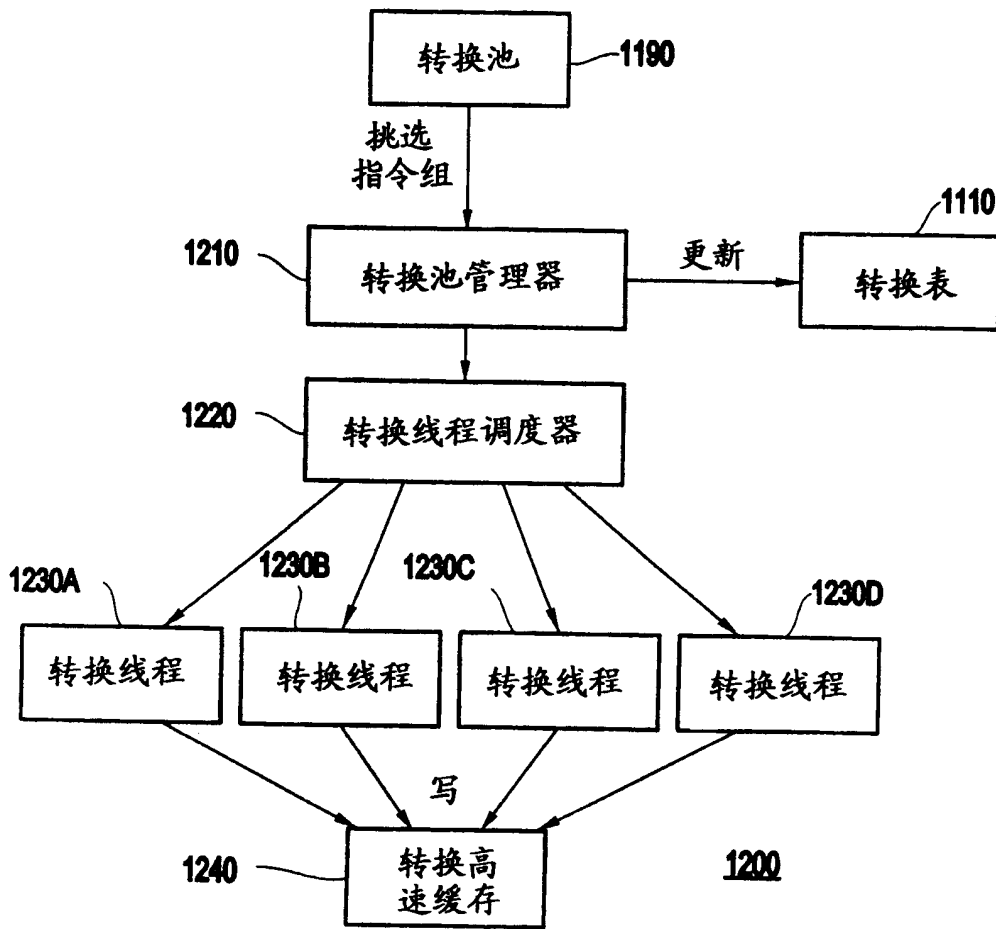


图12

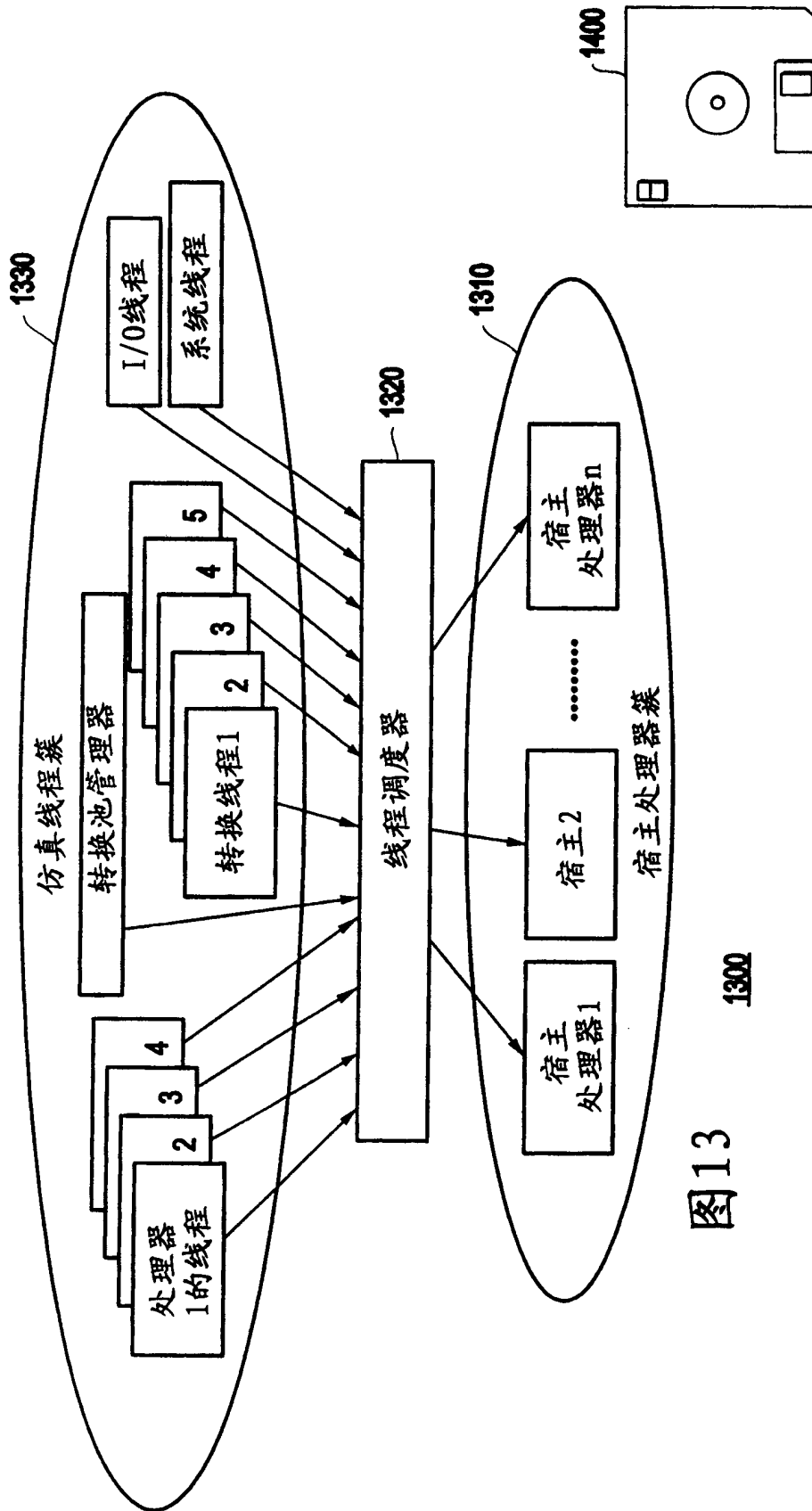


图13

1300

图14