[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/50
G06F 13/14 G06F 15/16

[12] 发明专利申请公开说明书

[21] 申请号 03145156.X

[43] 公开日 2004 年 1 月 21 日

[11] 公开号 CN 1469246A

[22] 申请日 2003.6.16 [21] 申请号 03145156.X
[30] 优先权
[32] 2002.6.20 [33] US [31] 10/177,410
[71] 申请人 国际商业机器公司
地址 美国纽约
[72] 发明人 L·B·布伦纳 D·J·伯迪克

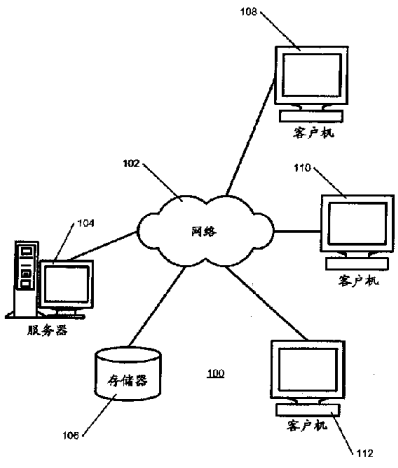
[74] 专利代理机构 北京市中咨律师事务所
代理人 于 静 李 峥

权利要求书 4 页 说明书 10 页 附图 8 页

[54] 发明名称 对多处理器系统进行负荷平衡的装置和方法

[57] 摘要

本发明公开了一种对多处理器系统进行负荷平衡的方法，系统和装置，所述多处理器系统包括多个处理器运行队列，每个运行队列用于保持待处理的线程。本发明对每个创建的线程分配一个优先级并且将创建的线程放置于运行队列中，分配的优先级将线程归类到一个区域中；如归类到一个运行队列中的一个区域中的线程数多于另一运行队列中的同一区域中的线程数，系统就是负荷不平衡的；如果系统是负荷不平衡的，可通过将在该区域中较高线程数的运行队列中的线程转移到在该区域中较具有低线程数的运行队列中来平衡此系统。



ISSN1008-4274

1.一种对多处理器系统进行负荷平衡的方法，所述系统的构成包括多个处理器运行队列，每个运行队列用于保持待处理的线程，此方法的构成包括:

对每个创建的线程分配一个优先级并且将创建的线程放置于运行队列中，分配的优先级将线程归类到一个区域中;

判断系统是否负荷不平衡，如归类到一个运行队列中的一个区域中的线程数多于另一运行队列中的同一区域中的线程数，系统就是负荷不平衡的；以及

对系统进行负荷平衡处理，如果系统是负荷不平衡的，可通过将在该区域中具有较高线程数的运行队列中的线程转移到在该区域中具有较低线程数的运行队列中来平衡此系统。

2.如权利要求 1 所述的方法，其中按类来组织每个队列的线程。

3.如权利要求 2 所述的方法，其中根据线程所属于的多处理器系统的用户，来确定线程所属于的类。

4.如权利要求 3 所述的方法，其中分配的优先级是线程属于的类的优先级。

5.如权利要求 4 所述的方法，其中当类中的线程受到处理时，类优先级减小，而当类中的线程在容许的范围内未受到处理时，其类优先级增加。

6.如权利要求 5 所述的方法，其中当类优先级减小或增加时，线程可重新归类到另一个区域。

7.如权利要求 6 所述的方法，其中如果一个运行队列的一个区域中的线程数超过另一个运行队列的同一个区域中的线程数一个预定的数目以上，则系统是负荷不平衡的。

8.一种在计算机可读介质上用来对多处理器系统进行负荷平衡的计算机程序产品，所述系统的构成包括多个处理器运行队列，每个运行队列用于保持待处理的线程，此计算机程序的构成包括:

对每个创建的线程分配一个优先级并且将创建的线程放置于运行队列中的代码单元，分配的优先级将线程归类到一个区域；

判断系统是否负荷不平衡的代码单元，如归类到一个运行队列中的一个区域中的线程数多于另一运行队列中的同一区域中的线程数，系统就是负荷不平衡的；以及

对系统进行负荷平衡处理的代码单元，如果系统是负荷不平衡的，可通过将在该区域中具有较高线程数的运行队列中的线程转移到在该区域中具有较低线程数的运行队列中来平衡此系统。

9.如权利要求 8 所述的计算机程序产品，其中按类来组织每个队列的线程。

10. 如权利要求 9 所述的计算机程序产品，其中根据线程所属于的多处理器系统的用户，来确定线程所属于的类。

11. 如权利要求 10 所述的计算机程序产品，其中分配的优先级是线程属于的类的优先级。

12. 如权利要求 11 所述的计算机程序产品，其中当类中的线程受到处理时，类优先级减小，而当类中的线程在容许的范围内未受到处理时，其类优先级增加。

13. 如权利要求 12 所述的计算机程序产品，其中当类优先级减小或增加时，线程可重新归类到另一个区域。

14. 如权利要求 13 所述的计算机程序产品，其中如果一个运行队列的一个区域中的线程数超过另一个运行队列的同一个区域中的线程数一个预定的数目以上，则系统是负荷不平衡的。

15. 一种对多处理器系统进行负荷平衡的装置，所述系统的构成包括多个处理器运行队列，每个运行队列用于保持待处理的线程，此装置的构成包括：

对每个创建的线程分配一个优先级并且将创建的线程放置于运行队列中的装置，分配的优先级将线程归类到一个区域中；

判断系统是否负荷不平衡的装置，如归类到一个运行队列中的一个区

域中的线程数多于另一运行队列中的同一区域中的线程数，系统就是负荷不平衡的；以及

对系统进行负荷平衡处理的装置，如果系统是负荷不平衡的，可通过将在该区域中具有较高线程数的运行队列的线程转移到在该区域中具有较低线程数的运行队列中来平衡此系统。

16. 如权利要求 15 所述的装置，其中按类来组织每个队列的线程。

17. 如权利要求 16 所述的装置，其中根据线程所属于的多处理器系统的用户，来确定线程所属于的类。

18. 如权利要求 17 所述的装置，其中分配的优先级是线程属于的类的优先级。

19. 如权利要求 18 所述的装置，其中当类中的线程受到处理时，类优先级减小，而当类中的线程在容许的范围内未受到处理时，其类优先级增加。

20. 如权利要求 19 所述的装置，其中当类优先级减小或增加时，线程可重新归类到另一个区域。

21. 如权利要求 20 所述的装置，其中如果一个运行队列的一个区域中的线程数超过另一个运行队列的同一个区域中的线程数一个预定的数目以上，则系统是负荷不平衡的。

22. 一种可自行进行负荷平衡处理的多处理器系统，所述系统的构成包括多个处理器运行队列，每个运行队列用于保持待处理的线程，此系统的构成包括：

至少一个用来存储代码数据的存储系统；以及至少一个用来处理代码数据以便进行以下处理的处理器：对每个创建的线程分配优先级并且将创建的线程放置于运行队列中，分配的优先级将线程归类到一个区域中；判断系统是否负荷不平衡，如归类到一个运行队列中的一个区域中的线程数多于另一运行队列中的同一区域中的线程数，系统就是负荷不平衡的；以及对系统进行负荷平衡处理，如果系统是负荷不平衡的，可通过将在该区域中具有较高线程数的运行队列中的线程转移到在该区域中具有较低线程

数的运行队列中来平衡此系统。

23. 如权利要求 22 所述的多处理器系统，其中按类来组织每个队列的线程。

24. 如权利要求 23 所述的多处理器系统，根据线程所属于的多处理器系统的用户，来确定线程所属于的类。

25. 如权利要求 24 所述的多处理器系统，其中分配的优先级是线程属于的类的优先级。

26. 如权利要求 25 所述的多处理器系统，其中当类中的线程受到处理时，类优先级减小，而当类中的线程在容许的范围内未受到处理时，其类优先级增加。

27. 如权利要求 26 所述的多处理器系统，其中当类优先级减小或增加时，线程可重新归类到另一个区域。

28. 如权利要求 27 所述的多处理器系统，其中如果一个运行队列的一个区域中的线程数超过另一个运行队列的同一个区域中的线程数一个预定的数目以上，则系统是负荷不平衡的。

对多处理器系统进行负荷平衡的装置和方法

技术领域

本发明系针对计算机系统中的资源分配。更具体言之，本发明系针对根据系统管理员设定的某些限制或使用策略对计算机系统的资源进行负荷平衡的方法和装置。

背景技术

在任何给定的处理时间，可能有多个用户进程或线程等待在计算机系统的处理器或 CPU 上执行。为了最好地利用系统的 CPU，就必须采用可以对等待执行的进程或线程进行适当排队的有效机制。为完成这一任务，多数计算机系统采用的机制是利用调度器来完成该任务。

注意，进程是一个程序。当一个程序执行时，不严格地称其是一个任务。在多数操作系统中，在任务和程序之间存在一对一的关系。然而，有些操作系统容许将一个程序分割成为多个任务或线程。这种系统称为多线程操作系统。为简单起见，下面线程和进程可以互换使用。

调度器是一个可以协调计算机系统的共享资源（比如，CPU）的使用的软件程序。调度器通常使用先进先出（即 FIFO）、轮转法或后进先出（LIFO）、优先级排队、树等等算法或多个算法的组合来进行调度。基本上，如果一个计算机系统具有 3 个 CPU（CPU_1、CPU_2 和 CPU_3），每个 CPU 将相应地具有一个就绪待处理队列或运行队列。如果用来对运行队列分配进程的算法是轮转法并且如果生成的最后进程是分配与 CPU_2 相关联的队列，则生成的下一个进程将分配 CPU_3 的队列。而生成的下一个进程将分配与 CPU_1 相关联的队列，依此类推。这样，调度器的设计可使每个进程公平共享计算机系统的资源。

有时系统管理员可能希望不同的进程对资源的共享是不同的。在此场合，与调度器相结合使用工作负荷管理器(WLM)。WLM 对每个进程分配一个数字表示一个进程可利用计算机系统的资源的时间的百分比。每当一个进程使用资源时，减小其被分配的数字。这一方案容许具有较低分配数字的进程仍然可以利用资源。

在某些时候，将这一概念应用于各类进程。在此情况下，一类进程可以是属于一组特定用户的一组进程。于是，恰好与给一个进程分配一个数字的情况一样，当一类进程利用资源时，分配给此类进程的数字减小。同样，这一方案也是保证具有较低分配数字的那些类进程有机会利用资源。

然而，问题是有时在一个例如具有两类进程（每一类进程都具有不同的分配数字）的两处理器系统中，最终是一个处理器处理其队列中的一类进程，而另一处理器处理其队列中的另一类进程。当出现这种情况时，计算机系统的两个处理器的利用效率可能不如系统管理员希望的那样高。特别是，一个运行队列中的进程将得到比容许其得到的更多的处理时间。

这样，需要有一种系统、装置和方法来保证系统的每个队列永远包含分配数字不同的各类进程的混合以保证系统管理员建立的处理器使用策略可以得到坚持。

发明内容

本发明可提供一种集成系统任务调度器和工作负荷管理器的方法、系统和装置。调度器用于向线程分配默认优先级和将线程置于运行队列中，而工作负荷管理器用于执行系统管理员设定的策略。策略之一可以是使不同类的线程类获得系统 CPU 时间的百分比不同。这一策略在多类线程在运行队列中尽可能均匀分布时可以可靠地实现。

在一个具体实施方式中，按类组织线程，每一类由一组相关的线程构成。每一类都与遵循使用策略的优先级相关联。此优先级用于修改分配给此类中的每个线程的调度优先级。根据类优先级的值，类和该类中的每个线程可处于多个优先级区域或范围之一中。本发明周期性地判断一运行队

列中一个优先级区域内的线程数另一运行队列中该优先级区域内的线程数相比是否超过一个预先确定的数值。如果是这样，系统就被认为是负荷失衡。本发明就力图均衡此系统的负荷，从线程数大的运行队列中移动该优先级区域内的一个线程送到线程数较小的运行队列。如果此系统在最高优先级区域的负荷是平衡的，本发明将通过检查了解系统在次高优先级区域的负荷是否平衡，依此类推。

附图说明

本发明的特征性的新特点在后附的权利要求中列出。然而，本发明本身，以及其优选使用模式、进一步的目的及优点可参照结合下列附图的实施例的详细描述了解得最为清楚。

图 1 为示出根据本发明的分布式数据处理系统的示例性框图。

图 2 为根据本发明的服务器装置的示例性框图。

图 3 为根据本发明的客户机的示例性框图。

图 4 示出可由系统管理员设计的资源使用策略。

图 5 示出了类优先级如何影响运行队列中的线程。

图 6 示出了由 3 组不同的用户共享的计算机系统的运行队列。

图 7 为可用来实现本发明的软件程序的流程图。

具体实施方式

下面参照图 1 对可实现本发明的数据处理系统的网络的图示予以说明。网络数据处理系统 100 是其中可以应用本发明的计算机的网络。网络数据处理系统 100 包含网络 102，网络 102 是用来在网络数据处理系统 100 之内互联的各种装置和计算机之间提供通信链路的媒体。网络 102 可包含各种连接，例如，有线通信链路、无线通信链路或光纤缆线。

在示例中，服务器 104 以及存储单元 106 与网络 102 连接。此外，客户机 108、110 和 112 与网络 102 相连。这些客户机 108、110 和 112 可以是，例如，个人计算机或网络计算机。在示例中，服务器 104 向客户机 108、

110 和 112 提供数据，如引导文件、操作系统映像及应用。客户机 108、110 和 112 是服务器 104 的客户机。网络数据处理系统 100 可包括附加的服务器、客户机和其他未示出的装置。在示例中，网络数据处理系统 100 是由网络 102 代表使用 TCP/IP 协议集互相通信的全世界网络和网关集合的因特网。在因特网的中心是在主节点或主计算机之间的主干高速数据通信线路，由成千上万个传送数据和消息的商用、政府、教育及其他计算机系统组成。当然，网络数据处理系统 100 也可以由不同类型的网络实现，比如像内联网、局域网（LAN）或广域网（WAN）。图 1 的意图是作为一个示例，而不是对本发明的体系结构的限制。

下面参照图 2，图 2 示出的是根据本发明的优选实施例的数据处理系统的框图，此系统，可以作为服务器，例如图 1 中的服务器 104。数据处理系统 200 可以是对称的多处理器(SMP)系统，包含多个与系统总线 206 连接的处理器 202 和 204。另一种方案是采用单处理器系统。与系统总线 206 相连接的还有可提供与局部存储器 209 的接口的存储器控制器/高速缓存 208。I/O 总线桥 210 与系统总线 206 相连接并提供与 I/O 总线 212 的接口。存储器控制器/高速缓存 208 和 I/O 总线桥 210 可如图所示地集成。

与 I/O 总线 212 连接的外围部件互连（PCI）总线桥 214 可提供与 PCI 局部总线 216 的接口。可以有多个调制解调器与 PCI 局部总线 216 相连接。典型的 PCI 总线设备可支持 4 个 PCI 扩展槽或附加连接器。至图 1 的网络计算机 108、110 和 112 的通信链路可通过藉助附件电路板连接到 PCI 局部总线 216 的调制解调器 218 和网络适配器 220 来提供。附加的 PCI 总线桥 222 和 224 可提供对附加的 PCI 局部总线 226 和 228 的接口，从这些接口附加的调制解调器或者网络适配器可以得到支持。以这种方式，数据处理系统 200 允许与多个网络计算机相连接。存储器映像图形适配器 230 和硬盘 232 也可以或者直接或者间接地与 I/O 总线 212 相连接，如图所示

本技术领域的人员可以理解，图 2 所示的硬件是可以变化的。比如，其他的外围设备，如光盘驱动器等等，也可以用来代替所示的硬件或者与其一起使用。上述的示例并不意味着对本发明进行系统结构上的限制。

图 2 所示的数据处理系统也可以是，例如，IBM 的 e-Server pSeries 系统，纽约阿芒口国际商业机器公司的产品，可运行先进交互执行(AIX)操作系统或 LINUX 操作系统。

下面参考图 3，图中示出可以应用本发明的数据处理系统的框图。数据处理系统 300 是客户计算机的一个示例。数据处理系统 300 采用外围部件互连（PCI）局部总线系统结构。虽然示出的示例采用 PCI 总线，但是也可以采用其他的总线结构，如加速图形端口(AGP)和工业标准体系结构（ISA）。处理器 302 和主存储器 304 通过 PCI 桥 308 与 PCI 局部总线 306 相连接。PCI 桥 308 也可包含应用于处理器 302 的集成存储器控制器和高速缓冲存储器。至 PCI 局部总线 306 的附的连接，可以通过元件直接互连或通过附加电路板来实现。在上述示例中，局域网(LAN)适配器 310、SCSI 主机总线适配器 312 和扩展总线接口 314 通过元件直接互连连接到 PCI 局部总线 306。与此相对，音频适配器 316、图形适配器 318 和音频/视频适配器 319 通过插入扩展槽的附加电路板与 PCI 局部总线 306 相连接。扩展总线接口 314 可提供到键盘和鼠标适配器 320、调制解调器 322 和附加存储器 324 的连接。小型计算机系统接口(SCSI)主机总线适配器 312 可提供到硬盘驱动器 326、磁带驱动器 328 和 CD-ROM 驱动器 330 的连接。典型的 PCI 局部总线装置可以支持三个或者四个扩展槽或附加连接器。

操作系统在处理器 302 上运行并且用来协调和提供对图 3 中的数据处理系统 300 内的各种部件的控制。操作系统可以是市售的操作系统，如微软公司出售的 Windows 2000。面向对象的编程系统，如 Java，可以与操作系统配合运行并提供从在数据处理系统 300 上执行的程序或应用对操作系统的调用。“Java”是 Sun 微系统公司的商标。操作系统的指令、面向对象的操作系统、应用或程序存储在存储装置，如硬盘驱动器 326 中，并且装载到主存储器 304 由处理器 302 执行。

本领域的人员可以理解，图 3 所示的硬件可随实现而改变。其他的内部硬件或外设，如闪速 ROM(或等效的非易失性存储器)或光盘驱动器等等，也可以用来代替图 3 所示的硬件或者与其一起使用。同样，本发明的

进程也可应用到多处理器数据处理系统。

另外一个例子，数据处理系统 300 可以是一个独立系统，配置成为可以引导而无需依靠某种网络通信接口，不管数据处理系统 300 是否包含某种网络通信接口。再举一个例子，数据处理系统 300 可以是个人数字助理（PDA），其配置为具有 ROM 和/或闪速 ROM 存储器，用来提供存储操作系统文件和/或用户生成的数据的非易失性存储器。

图 3 中的示例和上述的示例并不代表对本发明的体系结构的限制。比如，数据处理系统 300 除了采取 PDA 形式之外，也可以是笔记本计算机或手持计算机。数据处理系统 300 也可以是信息站设备或 web 设备。

本发明可以提供一种装置、系统和方法来保证多处理器系统的每个运行队列包含不同线程类中的进程，每个类具有不同的优先级数，从而保证建立的使用策略可以得到遵守。对于图 1 的客户机 108、110 和 112 或对于服务器 104 或对于服务器 104 及客户机 108、110 和 112 两者，本发明也可以是局部性质的。因此，本发明可以驻留在计算机系统所使用的任何数据存储介质（即软盘、CD 盘、硬盘、ROM、RAM 等等）之上。

图 4 示出可由系统管理员设计的资源使用策略。如图 4A 所示，此资源使用策略的开发是针对一个大学中的 3 个系（例如物理系、化学系和数学系）共享的计算机系统（例如服务器）。根据此策略，物理系 400 的用户 402 得到 60%的计算机系统 CPU 时间，化学系 410 的用户 412 得到 30%的计算机系统 CPU 时间，而数学系 420 的用户 422 得到 10%的计算机系统 CPU 时间。为了互相区别各组线程，将其标记为类。具体说，属于物理系的用户的线程标记为 A 类，属于化学系的用户的线程标记为 B 类，而属于数学系的用户的线程标记为 C 类。

此资源使用策略的一个附加策略是区域分布策略。此区域分布策略示于图 4B。其中显示出两个区域，区域 $_1$ 和区域 $_2$。这一分布策略将优先级在 0－14 之间的所有类归类到区域 $_1$ 中，而将优先级在 15－30 之间的所有类归类到区域 $_2$ 中。这样，新创建的属于数学系用户的线程将落在区域 $_1$ 中，而属于物理系或化学系用户的线程将落在区域 $_2$ 中。

与现有技术中一样，生成的每个线程具有默认的优先级。此默认优先级对于所有的线程都相同。然而，当线程置于运行队列中时，其优先级根据其所属的类进行调节。例如，对 A 类线程的默认优先级将加上 30。与此类似，对 B 类线程的默认优先级将加上 20，而对 C 类线程的默认优先级将加上 10。这样，运行队列中的线程的总优先级为：$P_{总}=P_{默认}+P_{类}$。

图 5 示出类优先级如何影响运行队列中的线程。在图 5 中，处理器的优先级空间 500 分割为优先级数 0、10、20 和 30。还示出 A 类线程 520、B 类线程 540 和 C 类线程 560。线程的 $P_{默认}$ 分配零（0）值。于是，所有的线程都从同一优先级零（0）开始。将优先级增量ΔA 530，30 加在 A 类线程之上，将优先级增量ΔB550，20 加在 B 类线程之上，而将优先级增量ΔC 570，10 加在 C 类线程之上。于是，处理器对待 A 类线程将会优先于 B 类线程。与此类似，处理器对待 B 类线程将会优先于 C 类线程。

在现有技术中，在线程受到处理时，其优先级也减小。这一方案容许运行队列中的所有线程可以公平地得到处理器的注意。在本发明中也采用同一方案。即在线程受到处理时，其 $P_{默认}$ 将减小。这容许类内的线程互相公平地竞争。此外，在一类线程受到处理时，类优先级也同样地减小。例如，由于 A 类线程具有比 B 类线程和 C 类线程高的优先级，A 类线程将在其他两类之前处理。但是，在 A 类线程受到处理时，A 类优先级 30 也将减小，直到达到 20 或 20 以下。此时，A 类和 B 类线程两者都将受到处理。两类的优先级将减小到 10 或 10 以下，到那时所有三类中的线程将受到处理。

在类优先级减小时，类可能从一个区域进入另一个区域。于是，类优先级可看作是流动的。当类中的线程受到处理时，优先级减小，而当线程未受到处理时，优先级增加，如在使用策略中所示。

图 6 示出共享的计算机系统的运行队列。其中假设计算机系统具有两个 CPU 和两个运行队列，每个运行队列都与一个 CPU（即 CPU_0 600 和 CPU_1 650）有联系。还假设每一类的优先级增量已经加到线程上。系统中存在三类线程。具体地说，与 CPU_0 600 相联系的运行队列既包含 A 类 610

的线程，也包含 B 类 620 的线程。与 CPU_1 相联系的运行队列包含 C 类 630 的线程。

在运行中，当创建一个线程时，工作负荷管理器根据其属于的用户判断线程的类，而调度器利用放置算法（比如，轮转法）将线程放置于运行队列中。当线程受到处理脱离运行队列并且新的线程放置到运行队列中时，可能出现线程按照图 6 所示分布到系统的场合。

图 6 中线程的分布并不理想，因为由系统管理员设定的使用策略（见图 4A）将不会得到坚持。例如，由于 C 类线程是与 CPU_1 650 相联系的运行队列中的唯一的线程，这些线程将得到 50%的系统处理时间。显然，这比容许其得到的 10%高得多。一种保证其一点儿都不会得到多于容许其得到的处理时间的方法是使运行队列中存在另一类线程。这将促使两类线程之间的竞争。的确，线程的理想分布是使每个运行队列中包含所有三个不同类的线程。

保证建立的使用策略得到坚持的一种方法是周期地检查每个运行队列来判断其中是否至少存在不同的两类线程（即一个优先级较高的类和一个优先级较低的类）。如不是如此，则应该在该运行队列中放置一个具有比现在存在于其中的类优先级高的不同类的线程。另外，如果尽管在运行队列中具有这样不同的两类线程，但两类中的任何一类或两类仍然得到高于容许的处理时间，那就需要在运行队列中再放置一个具有更高的类优先级的不同的类的线程。

为了检查运行队列中的所有的线程来判断其中是否每一个都包含至少两类不同的线程，需要花费很多时间并且使 CPU 高强度运行。结果，本发明每 0.1 秒检查线程的一部分以便了解其是否分布于全部运行队列中。本发明利用此前引入的区域概念(见图 4B)来判断对哪一部分线程进行调查。具体说，本发明将在每一个运行队列中包括相同数目的特定区域中的线程的系统定义为良好平衡系统。如果两个 A 类线程、两个 B 类线程和两个 C 类线程存在于两个处理器中的每一个的运行队列中（见图 6B），此系统将是平衡系统。此系统之所以是平衡系统是因为在两个运行队列中都有四个

区域$_2$中的线程和两个区域$_1$中的线程。

如果系统不是平衡系统（即，如果一个运行队列比另一个运行队列具有更多的某一特定区域中的线程），则线程将从具有较大线程数的运行队列转移到在该区域中具有较小线程数的运行队列。应该指出，此系统是很强的动态系统。就是说， 在任何时刻，一个线程可以放置到运行队列中，而一个或多个线程可以受到处理脱离运行队列，而一个类或多个类可从一个区域转移到另一个区域等。于是，因为这一动态机制，本发明在一个时候只处理一个区域，并且当系统不平衡时只有一个线程从一个运行队列中转移到另一个运行队列。

再参考图 6A。当本发明检查区域$_2$时，将会发现在该区域中有八（8）个线程，不过其全部都是在与 CPU_0 610 相联系的运行队列中。因此，系统将被确定为不平衡的。结果，一个线程将从与 CPU_0 相联系的运行队列转移到与 CPU_1 相联系的运行队列，并且该过程结束。

在此场合，转移的线程是 A 类还是 B 类线程实际上没什么关系，因为两类都早在区域$_2$中。如果系统是平衡的并且在此进程中所有的 B 类线程从与 CPU_0 相联系的运行队列转移到与 CPU_1 相联系的运行队列，则 B 类线程可能会启动取得大量的处理时间（因为其类优先级高于 C 类线程）。当发生这种情况时，其类优先级将开始减小。如果其类优先级减小到足够使其类优先级现在进入区域$_1$，系统将会再变成为不平衡的。此时，A 类线程将从一个运行队列转移到另一个运行队列以使系统重新变为平衡系统。

在使区域$_2$平衡之后，区域$_1$将受到检查。在此场合，将会发现系统是非平衡系统，因为所有的 C 类线程（请注意 C 类是区域$_1$中唯一的一类）处于与 CPU_1 相联系的运行队列中。结果，C 类线程将每次转移一个直到两个运行队列都包含相同数目的 C 类线程。

在我们的示例中，理想的线程分布是示于图 6B 中的情况。在图 6B 中，每个运行队列都包含相同数目的各类线程。于是，假设线程既没有放置于运行队列中，也没有进行处理脱离运行队列，则系统将会永远保持平衡。如果一个类从一个区域转移到另一个区域，该类中的所有线程将会发生迁

移。因此，系统将会继续保持平衡。

图 7 为可用来实现本发明的软件程序的流程图。当计算机系统打开或重置时，此软件程序启动（步骤 700）。此程序监视每个运行队列中的每个类中的线程数。此程序做到这一点的方法是通过对运行队列每秒取样 100 次以便得到在每个运行队列中每个类中的平均线程数。之后，每 0.1 秒此过程利用线程类来判断一个运行队列最高区域中的线程数（例如区域 $_2$）平均来讲是否比另一运行队列中同一区域（即区域 $_2$）中的线程数大 1.5 个线程。如果是，则此系统将被认为是不平衡的。此过程将尝试通过将在该区域中具有最高线程数的运行队列中的一个线程转移到在该区域中具有最低线程数的运行队列中来平衡此系统。

如前所述，因为此系统具有很强的动态性，此过程将不能保证此系统的确是平衡的（即此过程将不能保证在每个运行队列中在该区域都存在相同数目的线程）。此外，因为这需要花费很多时间并且使 CPU 高强度运行来保证从所有区域来看此系统是平衡的，此过程将在此中止。

这样，此过程永远从最高区域开始。如果最高区域是不平衡的，则此过程将尝试平衡此区域并在该处中止。如果最高区域是平衡的，则此过程将会调查次高区域。每个区域都将受到调查，从最高的到最低的，直到一个运行队列的线程数比另一运行队列的中的线程数大 1.5 个线程。当发生这种情况时，将把具有最高线程数的运行队列的一个线程转移到具有最低线程数的运行队列中而此过程将结束。如果在所有的运行队列中所有的区域中的线程数都相等，则此过程将结束而不将任何线程从一个运行队列转移到另一个运行队列（步骤 700－714）。

对本发明的上述描述的目的在于显示和描述，而并不企图包罗一切或局限于所公开的形式的本发明。对于本技术领域的人士各种改变和变形是显而易见的。本实施方式的选择和描述是为了最好地说明本发明的原理，实际应用，并且使本技术领域的其他人士理解本发明的可适应预期的各种实际利用的各种改型的各种实施方式。

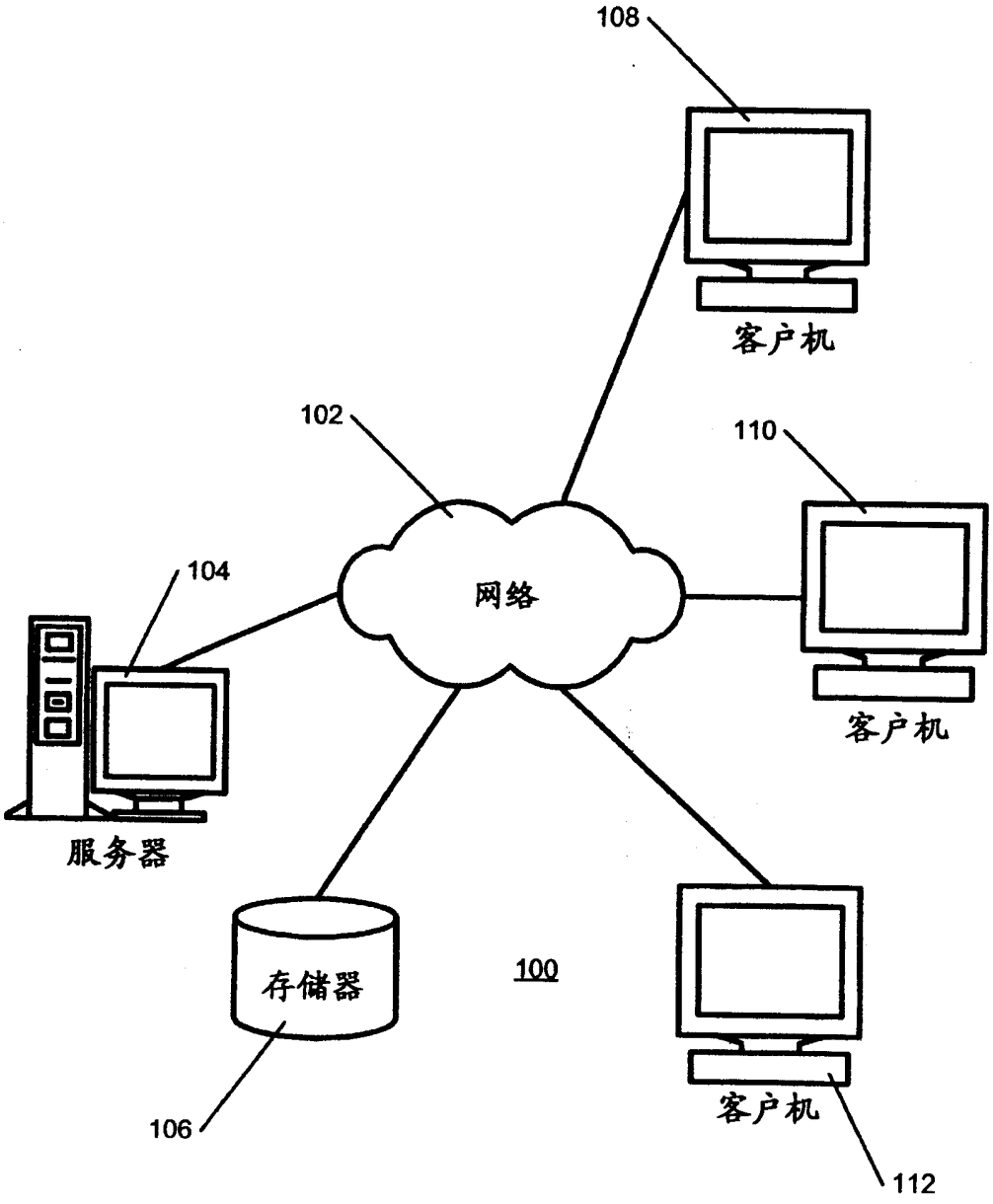


图1

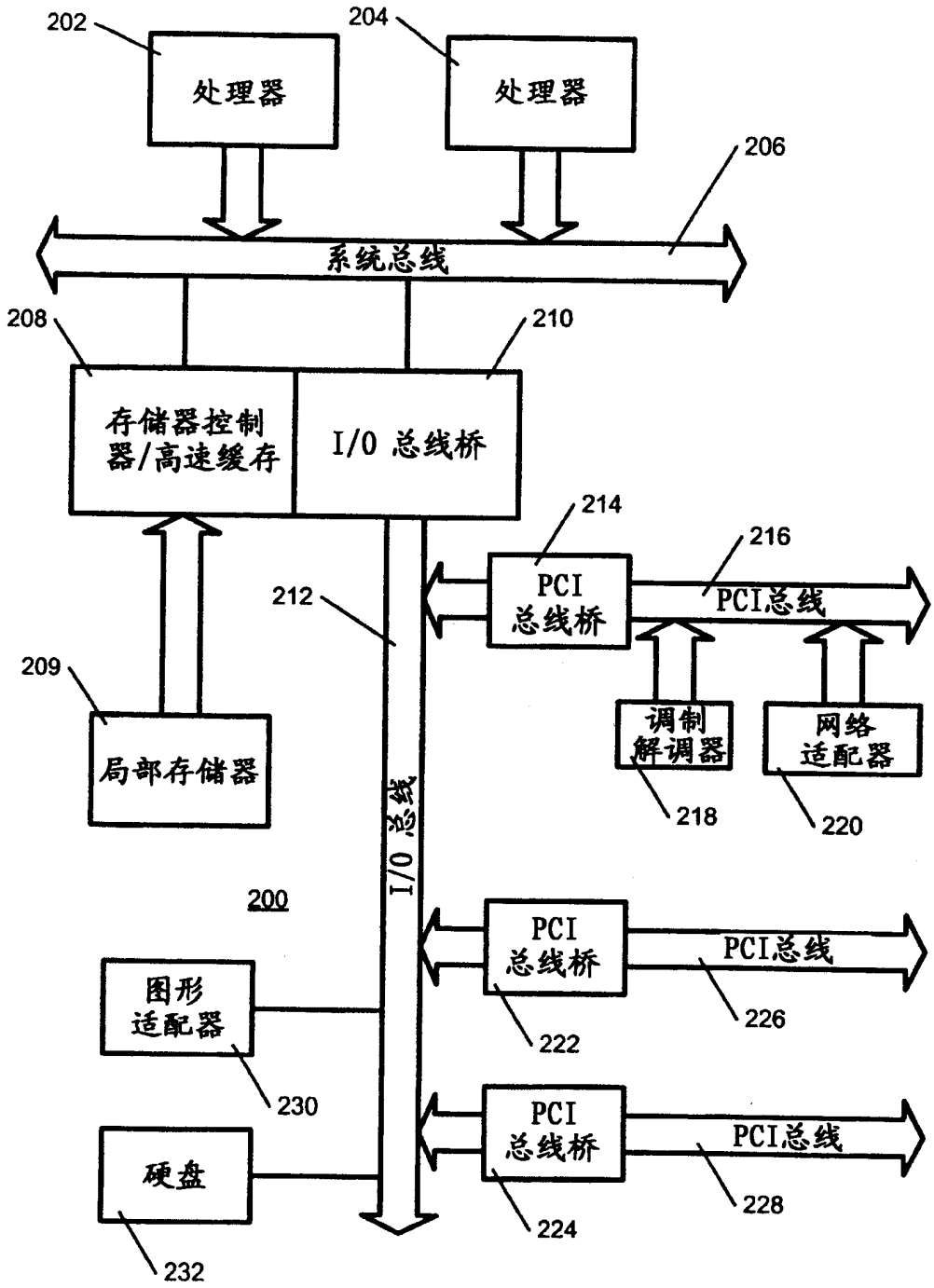

202
处理器
204
处理器
206
系统总线
208
存储器控制器/高速缓存
210
I/O 总线桥
214
PCI 总线桥
216
PCI总线
212
209
局部存储器
调制解调器
网络适配器
218
220
I/O 总线
200
PCI 总线桥
PCI总线
图形适配器
222
226
230
PCI 总线桥
PCI总线
硬盘
224
228
232

图2

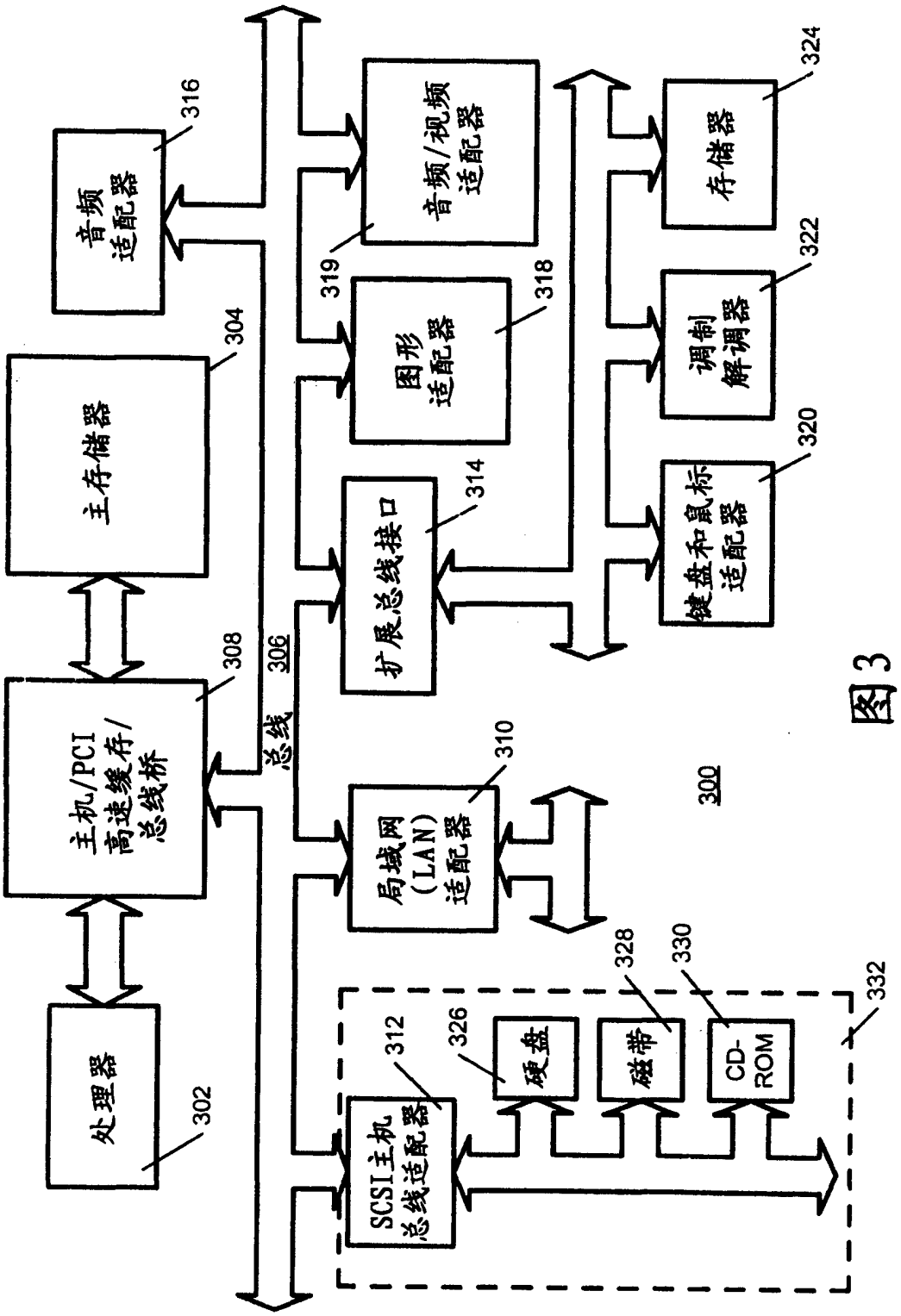
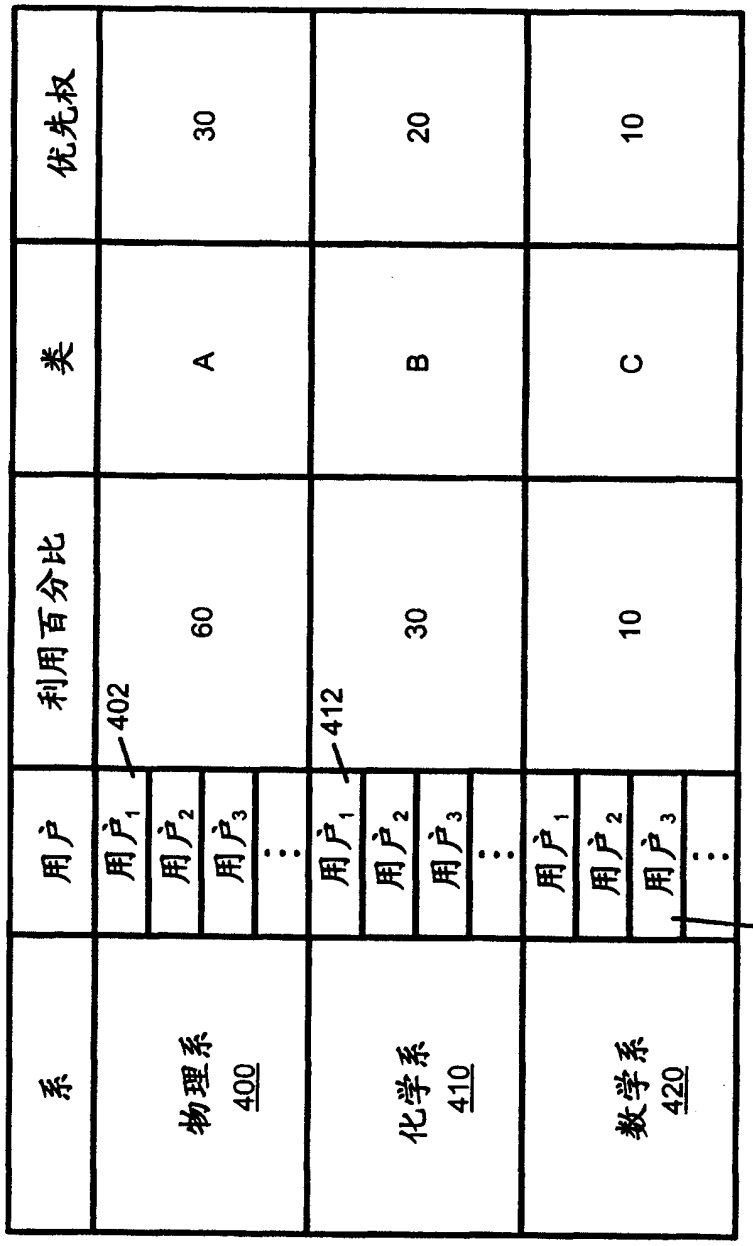

处理器
302
主机/PCI
高速缓存/
总线桥
308
主存储器
304
音频
适配器
316
总线
306
扩展总线接口
314
图形
适配器
318
音频/视频
适配器
319
局域网
(LAN)
适配器
310
SCSI主机
总线适配器
312
硬盘
326
磁带
328
CD-
ROM
330
332
键盘和鼠标
适配器
320
调制
解调器
322
存储器
324
300

图3



系	用户	利用百分比	类	优先权
物理系 400	用户1	60	A	30
	用户2			
	用户3			
	⋮			
化学系 410	用户1	30	B	20
	用户2			
	用户3			
	⋮			
数学系 420	用户1	10	C	10
	用户2			
	用户3			
	⋮			

402
412
422

图4A

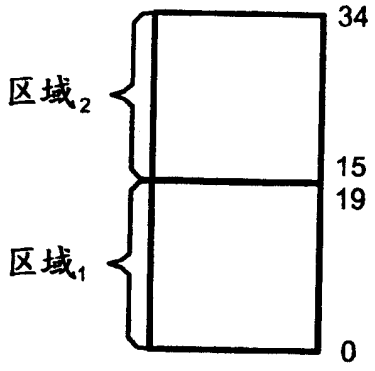

34
区域2
15
19
区域1
0

图4B

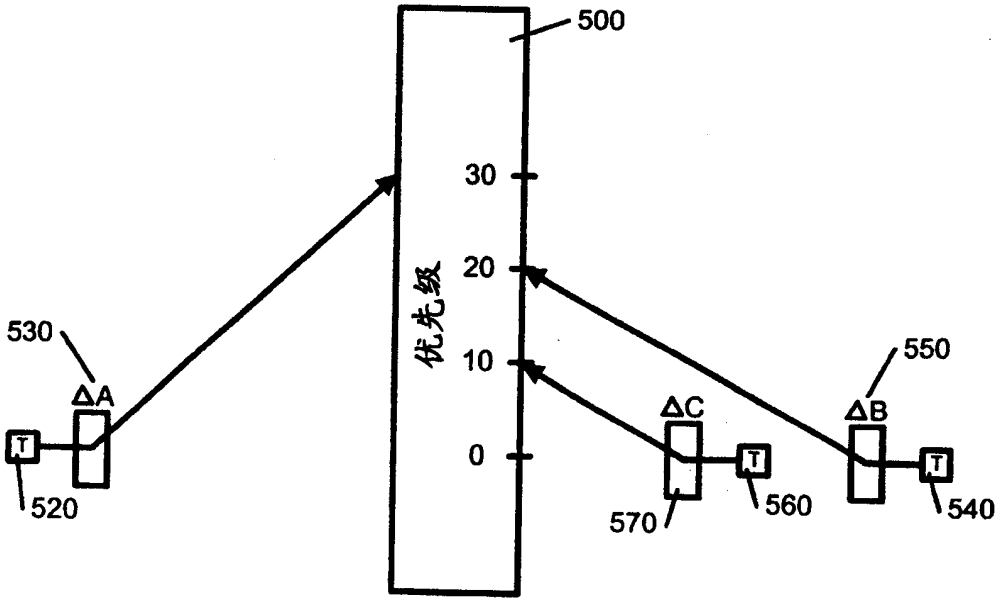

500
30
20
优先级
10
0
530
ΔA
T
520
ΔC
T
570
560
550
ΔB
T
540

图5

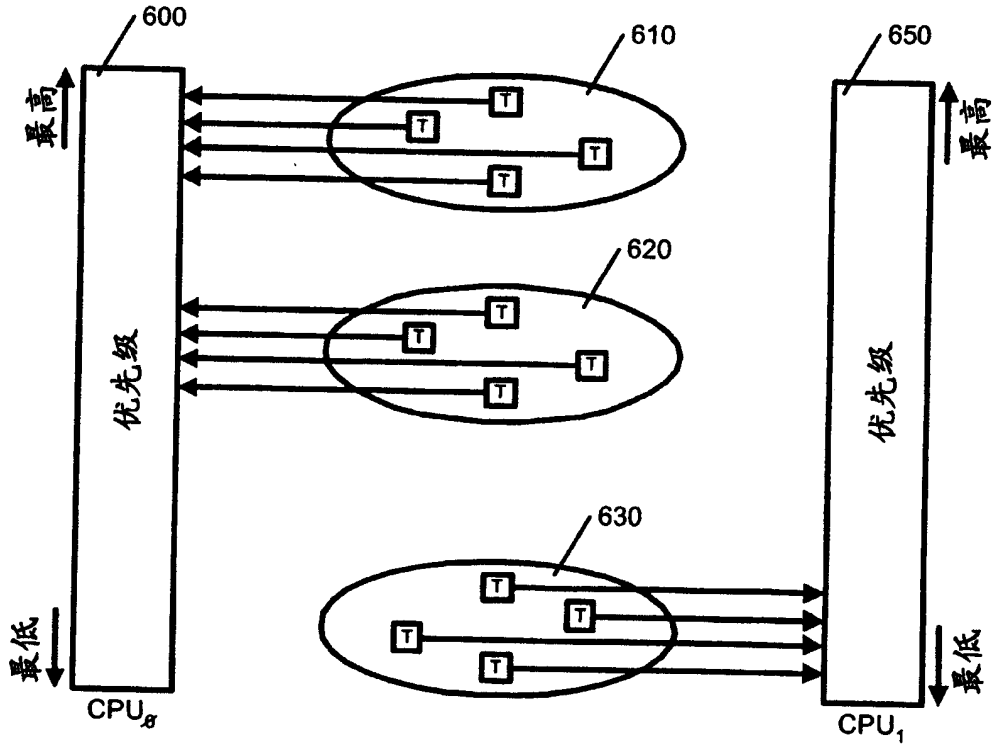


图6A

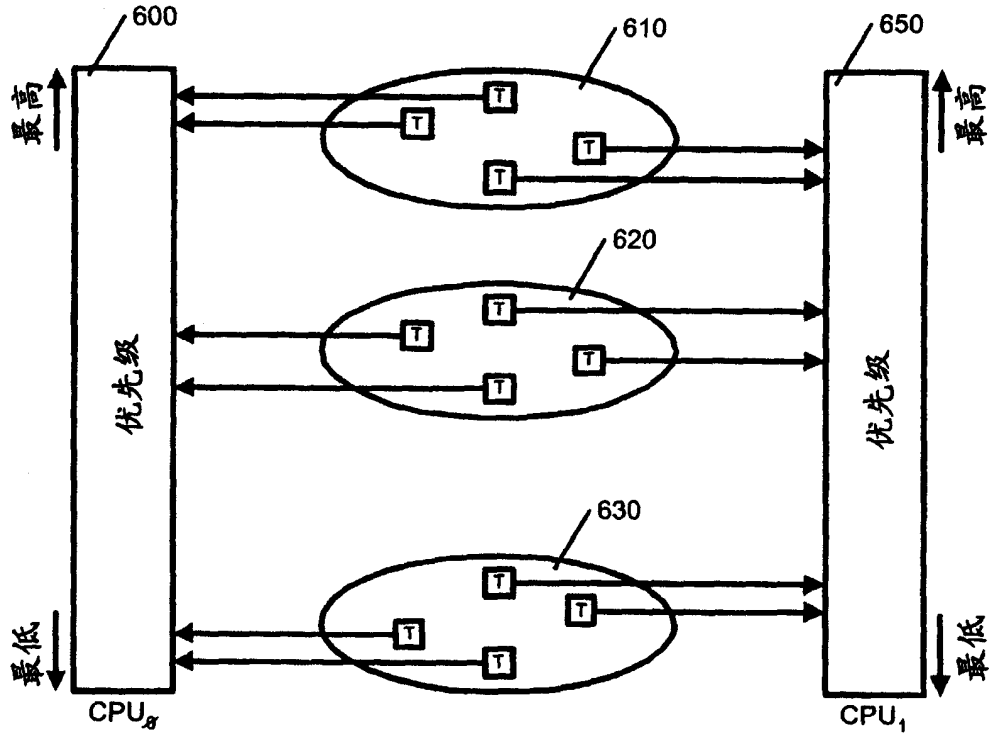


图6B

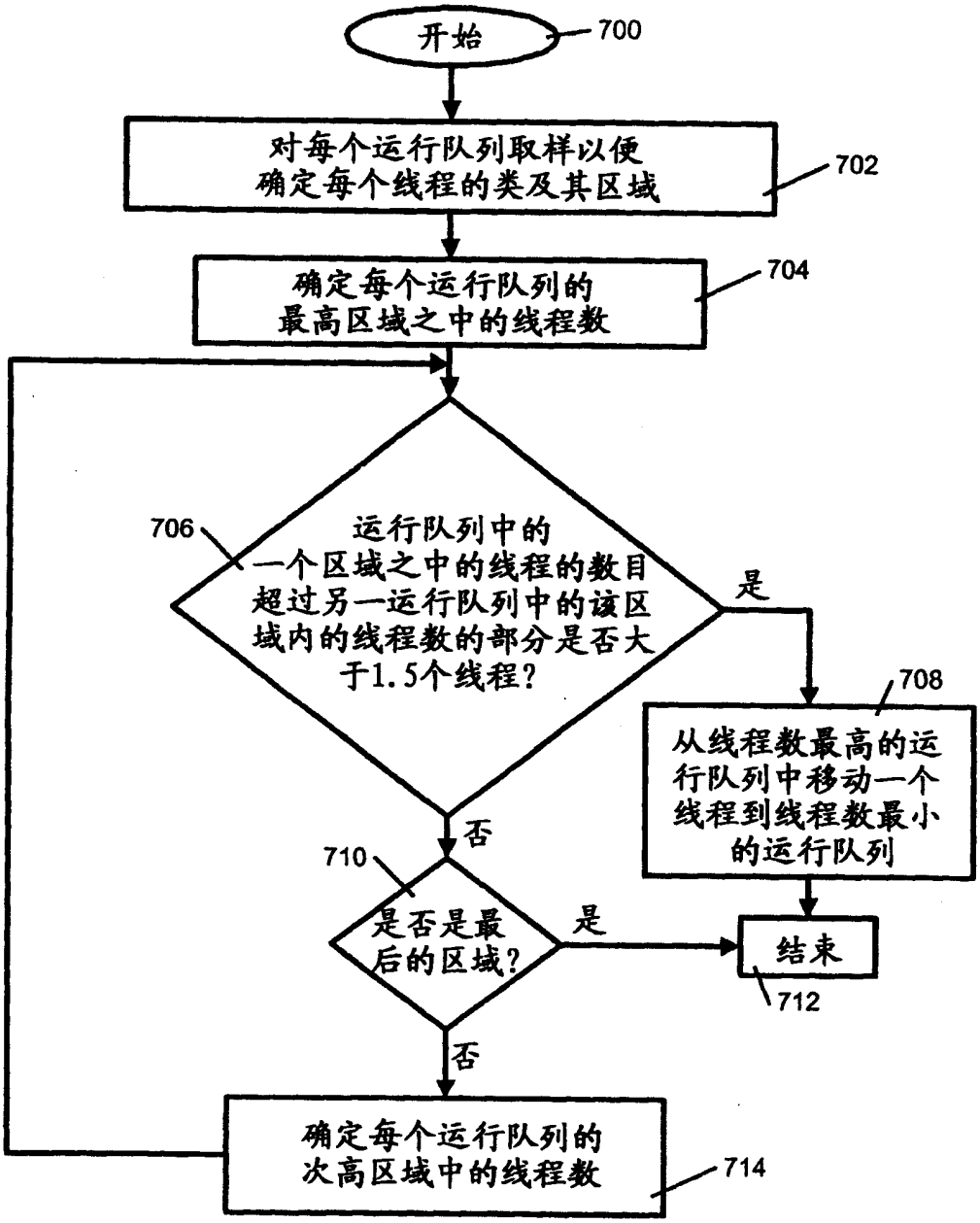

开始
700
对每个运行队列取样以便
确定每个线程的类及其区域
702
确定每个运行队列的
最高区域之中的线程数
704
706
运行队列中的
一个区域之中的线程的数目
超过另一运行队列中的该区
域内的线程数的部分是否大
于1.5个线程?
是
708
从线程数最高的运
行队列中移动一个
线程到线程数最小
的运行队列
否
710
是否是最
后的区域?
是
结束
712
否
确定每个运行队列的
次高区域中的线程数
714

图7