(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104094223 A
(43)申请公布日 2014.10.08

(21)申请号 201380008222.1
(22)申请日 2013.01.24
(30)优先权数据
13/366,999 2012.02.06 US
(85)PCT国际申请进入国家阶段日
2014.08.06
(86)PCT国际申请的申请数据
PCT/EP2013/051285 2013.01.24
(87)PCT国际申请的公布数据
WO2013/117434 EN 2013.08.15
(71)申请人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 B.R. 普拉斯基
A. 布尤克托苏诺格卢
V. 斯里尼瓦桑
(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 黄玫
(51)Int.Cl.
G06F 9/38(2006.01)

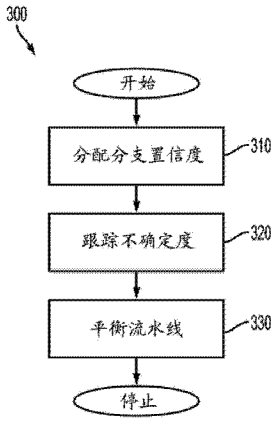
权利要求书1页 说明书9页 附图5页
按照条约第19条修改的权利要求书1页

(54)发明名称

通过指令不确定度的多线程处理器指令平衡

(57)摘要

一种用于指令执行的计算机系统包括具有流水线的处理器。所述系统被配置为:执行包括以下步骤的方法:在所述流水线中取回多条指令,其中,所述多条指令包括多条分支指令;对于所述多条分支指令中的每一条,将分支不确定度分配给所述多条分支指令中的每一条;对于所述多条指令中的每一条,分配指令不确定度,其为较旧的未判决的分支的分支不确定度的总和;以及在所述流水线中基于指令不确定度的当前总和来平衡所述指令。



CN 104094223 A

1. 一种用于流水线中的指令执行的计算机实现的方法，所述方法包括：

在所述流水线中取回多条指令，其中，所述多条指令包括多条分支指令；

对于所述多条分支指令中的每一条，将分支不确定度分配给所述多条分支指令中的每一条；

对于所述多条指令中的每一条，分配指令不确定度，其为较旧的未判决的分支的分支不确定度的总和；以及

在所述流水线中基于指令不确定度的当前总和来平衡所述指令。

2. 如权利要求 1 所述的方法，还包括：分配用于与所述多条指令的多个指令子集关联的多个线程中的每一个的多个不确定度。

3. 如权利要求 2 所述的方法，还包括：根据所述多个不确定度来计算多个不确定度比例。

4. 如权利要求 2 所述的方法，还包括：在所述流水线中调度所述多条指令。

5. 如权利要求 2 所述的方法，其中，与来自所述多个不确定度的比例成比例地调度指令。

6. 如权利要求 1 所述的方法，还包括：计算总不确定度，其中，所述总不确定度是用于所述多条指令中的每一个的所述指令不确定度的总和。

7. 如权利要求 1 所述的方法，还包括：保留用于所述多条指令的子集的发布队列空间。

8. 如权利要求 1 所述的方法，还包括：响应于第二子集指令变为可用于调度而从所述流水线清除第一指令子集。

9. 一种系统，包括适用于执行如任何前述方法权利要求所述的方法的所有步骤的装置。

10. 一种计算机程序，包括用于执行如任何前述方法权利要求所述的方法的所有步骤的指令。

通过指令不确定度的多线程处理器指令平衡

技术领域

[0001] 本发明涉及多线程处理，更具体地说，涉及用于通过指令不确定度在多线程处理器内的指令平衡的系统、方法和计算机程序产品。

背景技术

[0002] 在多线程流水线设计中存在很多目标，包括但不限于尽可能快地运行每个单独线程以及优化每瓦特的工作单元。流水线是并行性的一种具体形式，其中，可以在同一硬件上交错若干指令的执行。无论流水线设计的目标如何，对于程序的每个线程，都可能遇到多分支指令。当线程经历分支时，预测分支将采取哪个方向，然后沿着所预测的路径来执行线程。典型地按每分支预测表项 2 比特饱和计数器来实现分支预测，其中，状态是 :00(强不采用)，弱不采用 (01)、弱采用 (10) 和强采用 (11)。在已判决的所采用的分支上，计数器增加 ;然而，在到达“11”的状态时，计数器在该值饱和。在已判决的不采用的分支上，计数器减少 ;然而，在到达状态“00”时，计数器在该值饱和。对于状态“10”和状态“11”，将分支预测为采用，而对于状态“00”和“01”，将分支预测为不采用。该饱和计数器为分支提供方向 ;然而，关于分支的精度没有声明。例如，在 for 循环上，除了失败情况之外，每次都采用分支。在失败迭代时，置信度计数器从“11”降级为“10”。下一次遇到 for 循环，将再次采用该分支。故此，在“11”的状态下，对于遇到除了失败之外的所有情况采用分支。在“10”的状态时，将再次采用分支。故此，强状态并不指示比弱状态更置信的预测。在这种进行预测的方案中未考虑的是分支预测的精度。在多线程处理环境中，多线程可以同时执行工作。故此，多线程可以运行在流水线中，每个线程遇到多分支，并且通过流水线传输所预测的状态 (强 / 弱采用 / 不采用)，从而在分支判决时，可以更新分支表内的预测状态。此外，在多线程处理器中，存在各线程之间所共享的资源。这些动态划分的资源是基于先到先服务的。虽然动态资源允许硅面积和功率优化，但存在动态资源的很多性能限制，例如线程挂起 (hogging)，其产生性能限制因素。例如，当一个线程 X 停止 (stall) 时，另一线程 Y 有能力用上大量流水线资源。线程 Y 可能正创建扔掉 (throw-away) 工作 (例如，下行到所确定的分支错误路径) 并且从 X 获取 X 可能对于未扔掉的工作而在近期未来中使用的资源。

发明内容

[0003] 示例性实施例包括一种用于指令执行的计算机系统，所述系统包括 :处理器，具有流水线，其中，所述系统被配置为 :执行包括以下步骤的方法 :在所述流水线中取回多条指令，其中，所述多条指令包括多条分支指令 ;对于所述多条分支指令中的每一条，将分支不确定度分配给所述多条分支指令中的每一条 ;对于所述多条指令中的每一条，分配指令不确定度，其为较旧的未判决的分支的分支不确定度的总和 ;以及在所述流水线中基于指令不确定度的当前总和来平衡所述指令。

[0004] 附加示例性实施例包括一种用于流水线中的指令执行的计算机实现的方法，所述方法包括 :在所述流水线中获取多条指令，其中，所述多条指令包括多条分支指令 ;对于所

述多条分支指令中的每一条，将分支不确定度分配给所述多条分支指令中的每一条；对于所述多条指令中的每一条，分配指令不确定度，其为较旧的未判决的分支的分支不确定度的总和；以及在所述流水线中基于指令不确定度的当前总和来平衡所述指令。

[0005] 其它示例性实施例包括一种用于流水线中的指令执行的计算机程序产品。所述计算机程序产品包括：有形存储介质，可由处理电路读取，并且存储用于所述处理电路执行的指令，以用于执行方法。所述方法包括：在所述流水线中取回多条指令，其中，所述多条指令包括多条分支指令；对于所述多条分支指令中的每一条，将分支不确定度分配给所述多条分支指令中的每一条；对于所述多条指令中的每一条，分配指令不确定度，其为较旧的未判决的分支的分支不确定度的总和；以及在所述流水线中基于指令不确定度的当前总和来平衡所述指令。

[0006] 通过本发明的的技术来实现附加特征和优点。本发明的其它实施例和方面在此详细描述并且被看作一部分所要求的本发明。为了更好地理解本发明的优点和特征，参照描述和附图。

附图说明

[0007] 在说明书的结尾处的权利要求中特定地指出并且独特地要求被看作本发明的主题内容。从结合附图的以下详细描述，本发明的以上和其它特征和优点是清楚的，其中：

[0008] 图 1 示出用于通过指令不确定度在多线程处理器内进行指令平衡的系统 100 的框图；

[0009] 图 2 示出根据示例性实施例的可以实现示例性指令平衡方法的处理器指令流水线系统的框图；

[0010] 图 3 示出用于通过指令不确定度在多线程处理器内进行指令平衡的方法的流程图；

[0011] 图 4 示出可以实现为跟踪流水线中的不确定度的表的示例；

[0012] 图 5 示意性示出根据示例性实施例的可以实现的静态水印的示例；

[0013] 图 6 示意性示出根据示例性实施例的可以实现的来自发布队列的刷新 (flush) 的示例；以及

[0014] 图 7 示出在具有作为制造物的有形介质中实施的计算机程序代码逻辑的计算机可读 / 可用的介质上的计算机程序产品的示例。

具体实施方式

[0015] 在示例性实施例中，系统和方法在处理器中实现同时多线程运行，以允许处理器内核实现每晶体管更高的吞吐量（即更多工作）。在给定的流水线中，在此所描述的系统和方法不仅计算每条所预测的路径的方向，而且还计算用于每条所预测的路径的置信度 / 精度。此外，跟踪对于所预测的分支的不确定度并且通过每个所预测的分支传输对于所预测的分支的不确定度，以便关于从每个线程执行指令的顺序进行判断。通过允许线程“全空闲”并且只要存在可用的资源，就要求如每个线程期望的那样多的资源，资源在追求实现最大可能吞吐量的意义上并未在各线程之间得以划分。增加吞吐量公正性的一个区域是在分支的区域周围。每当给定的线程更有可能（相对于其它线程）对于给定的分支得到不正确

地预测的方向 / 目标时，就应在资源正消耗 / 请求时回退，并且允许其它（即更高置信的）线程要求其资源的共享。在单线程处理器中，所有资源专用于单线程。在同时多线程 (SMT) 处理器中，这些资源在并行执行线程之间是“共享”的。一些资源是静态地划分的，而其它资源是动态地划分的。动态地划分的资源是基于先到先服务的。虽然动态资源允许优化，但存在很多动态资源的性能限制。例如，当一个线程 X 停止达几个周期时，另一线程 Y 有能力利用大量流水线资源。在克服停止时，线程 Y 可能正创建扔掉 (throw-away) 工作（例如，下行到分支错误路径）并且从线程 X 获取线程 X 可能对于未扔掉的工作而使用的资源。

[0016] 图 1 示出用于通过指令不确定度在多线程处理器内进行指令平衡的系统 100 的框图。可以以硬件、软件（例如固件）或其组合来实现在此所描述的方法。在示例性实施例中，以作为专用或通用数字计算机（例如个人计算机、工作站、小型计算机或大型计算机）的微处理器的一部分的硬件来实现在此所描述的方法。系统 100 因此包括通用计算机 101。

[0017] 在示例性实施例中，关于硬件架构，如图 1 所示，计算机 101 包括处理器 105、耦合到存储器控制器 115 的存储器 110 以及以通信方式经由本地输入 / 输出控制器 135 而耦合的一个或多个输入和 / 或输出 (I/O) 设备 140、145（或外设）。输入 / 输出控制器 135 可以是例如但不限于一个或多个总线或其它本领域公知的有线或无线连接。输入 / 输出控制器 135 可以具有附加元件（其为了简化而省略），例如控制器、缓冲器（高速缓存）、驱动器、转发器和接收器，以使得能够进行通信。此外，本地接口可以包括地址、控制和 / 或数据连接，以使得在前述组件之间能够进行适当的通信。

[0018] 处理器 105 是用于执行特别是在存储器 110 中存储的软件的硬件设备。处理器 105 可以是任何定制的或商用的处理器、中央处理单元 (CPU)、与计算机 101 关联的若干处理器之间的辅助处理器、基于半导体的微处理器（以微芯片或芯片组的形式）、宏处理器或通常用于执行指令的任何设备。

[0019] 存储器 110 可以包括易失性存储器元件（例如随机存取存储器 (RAM，例如 DRAM、SRAM、SDRAM 等)）和非易失性存储器元件（例如 ROM，可擦除可编程只读存储器 (EPROM)、电可擦除可编程只读存储器 (EEPROM)、可编程只读存储器 (PROM)、磁带、压缩盘只读存储器 (CD-ROM)、磁盘、软盘、盒式磁盘、盒式磁带等）中的任一或组合。此外，存储器 110 可以包括电子、磁、光和 / 或其它类型的存储介质。注意，存储器 110 可以具有分布式架构，其中，各个组件位于彼此远离，但可以由处理器 105 存取。

[0020] 存储器 110 中的指令可以包括一个或多个分离程序，其中的每一个包括用于实现逻辑功能的可执行指令的有序列表。在图 1 的示例中，存储器 110 中的指令适合操作系统 (OS)111。操作系统 111 本质上控制其它计算机程序的执行，并且提供排程、输入 - 输出控制、文件和数据管理、存储器管理和通信控制以及有关的服务。

[0021] 在示例性实施例中，传统键盘 150 和鼠标 155 可以耦合到输入 / 输出控制器 135。其它输出设备（例如 I/O 设备 140、145）可以包括输入设备，例如但不限于打印机、扫描仪、麦克风等。最后，I/O 设备 140、145 可以还包括传递输入和输出的设备，例如但不限于网络接口卡 (NIC) 或调制器 / 解调器（用于存取其它文件、设备、系统或网络）、射频 (RF) 或其它收发器、电话接口、桥接器、路由器等。系统 100 可以还包括耦合到显示器 130 的显示控制器 125。在示例性实施例中，系统 100 可以还包括网络接口 160，用于耦合到网络 165。网络 165 可以是基于 IP 的网络，用于经由宽带连接在计算机 101 与任意的外部服务器、客户

机等之间的通信。网络 165 在计算机 101 与外部系统之间发送并且接收数据。在示例性实施例中，网络 165 可以是由服务提供商管制的受管理 IP 网络。可以例如使用无线协议和技术（例如 WiFi、WiMax 等）以无线方式来实现网络 165。网络 165 也可以是分组交换网络，例如局域网、广域网、城域网、因特网或其它相似类型的网络环境。网络 165 可以是固定无线网络、无线局域网 (LAN)、无线广域网 (WAN)、个域网 (PAN)、虚拟专用网络 (VPN)、内联网或其它合适的网络系统，并且包括用于接收并且发送信号的装备。

[0022] 如果计算机 101 是 PC、工作站、智能设备等，则存储器 110 中的指令可以还包括基本输入输出系统 (BIOS)（为了简化而省略）。BIOS 是基本软件例程的集合，其在启动时初始化并且测试硬件，启动 OS 111，并且支持各硬件设备之间的数据传送。BIOS 存储在 ROM 中，从而当激活计算机 101 时可以执行 BIOS。

[0023] 当计算机 101 处于操作中时，处理器 105 被配置为：执行存储器 110 内所存储的指令，以将数据传递出入存储器 110，并且通常依照指令来控制计算机 101 的操作。

[0024] 在示例性实施例中，在指令平衡方法以硬件实现的情况下，可以通过本领域均公知的以下技术中的任一或组合来实现在此所描述的指令平衡方法：具有用于在数据信号上实现逻辑功能的逻辑门的分立式逻辑电路、具有适当组合逻辑门的专用集成电路 (ASIC)、可编程门阵列 (PGA)、现场可编程门阵列 (FPGA) 等。

[0025] 图 2 示出根据示例性实施例的可以实现示例性指令平衡方法的处理器的无序指令流水线系统 200 的框图。如在此所描述的那样，流水线是并行性的一种具体形式，其中，可以在同一硬件上交错若干指令的执行。在示例性实施例中，流水线系统包括七级：(1) 指令取回 210；(2) 解码和指令分组 220；(3) 调度到发布队列中 230；(4) 无序发布 240；(5a) 执行寄存器修改的固定点单元 250；(5b) 存取高速缓存的加载 / 存储单元 255；(6) 有序完成表 260；(7) 包括寄存器提交和存储器存储的架构更新 270。在示例性实施例中，指令取回 210 级取回待处理的指令。解码和指令分组 220 级对指令进行解码并且确定哪些指令依赖于哪些指令。调度到发布队列中 230 将指令以及关联依赖性放置到发布队列中。无序发布队列 240 定义哪些指令待发送到执行架构功能的执行单元。固定点单元 250 执行寄存器到寄存器的操作。加载 / 存储单元 255 将获取寄存器值并且将其放置到存储器 / 高速缓存中，或获取存储器 / 高速缓存的中内容并且将其放置到寄存器中。有序完成 260 定义有序手段，以将指令发送到架构更新 270 级。架构更新 270 将寄存器提交到存储器 / 高速缓存，并且允许无序寄存器文件映射器释放重命名的寄存器。

[0026] 图 3 示出用于通过指令不确定度在多线程处理器内进行指令平衡的方法 300 的流程图。在块 310，系统 100 分配分支置信度。分支预测器可以实现很多类型的计数器，包括但不限于一位预测器或两位饱和计数器。为了说明，讨论两位饱和计数器。应理解，在其它示例性实施例中预期其它类型的预测计数器。对于所给定的预测分支的分支方向，可以表示若干状态。例如，所表示的状态可以是：11 强采用、10 弱采用、01 弱不采用、00 强不采用。在示例性实施例中，通过保持 x 位计数器正确并且 x 位计数器错误，可以为分支预测建立正确性的运行平均。每当分支预测正确（独立于采用或不采用）时，系统 100 增加“正确”计数器。每当预测错误时，系统 100 增加“错误”计数器。每当计数器在其最大值处饱和时，系统 100 将两个计数器除以二。例如，可以通过将每个计数器右移一位来完成除以二。在示例性实施例中，所预测的分支的置信度存储在具有方向预测的分支预测阵列中。每当进

行预测时，还取得不确定度。在一个实施例中，不确定度可以由下式给出：

[0027] 1/Confidence = Wrong/(Correct+Wrong)

[0028] 在块 320，系统 100 跟踪所预测的分支的不确定度。在示例性实施例中，当用于分支的不确定度具有大于 0 的错误计数时，则存在关于所预测的分支的不确定度的级别。不确定度指示：在最近的过去，按照计数器，预测已经错误。无论所给定的分支的不确定度如何，关于按照分支自身的不确定度在流水线中执行分支都没有问题；其为在所讨论的分支之后的指令。

[0029] 例如，如果所给定的分支具有“1”的不确定度，则在该分支之后下行于流水线 200 而发送的每条指令也被看作具有“1”的不确定度。故此，如果例如在该分支之后下行于流水线 200 而发送十条指令，则所给定的线程已经向下发送值为 10 的不确定度。如果在第一分支之后，存在四条无分支指令，然后存在后随另外五条指令的具有“2”的不确定度的另一分支，则用于该线程的流水线 200 中的总不确定度是“20”。在该示例中，可以理解，第一分支并不将不确定度添加到流水线 200。然而，接下来的四个无分支指令均具有“1”的不确定度（总不确定度于此＝“4”）。第二分支下行于第一分支的不确定度路径。第二分支因此将又一个不确定度添加到不确定度计数（总值现在为“5”）。在第二分支之后的指令均具有用于第二分支的“2”以及用于第一分支的“1”的不确定度，对于在第二分支之后的每条指令加和为“3”不确定度的值。不确定度是处于“3”的值的五条指令，总共 15 不确定度。对于十条指令共计，存在“20”的不确定度。继续于该示例，若第一分支正确地判决（第二分支仍待判决），则用于在第一分支之后的十条指令的流水线中的不确定度从“20”下降到“10”，因为第一分支在流水线 200 中不再提供不确定度。反之，若第二分支正确地判决（第一分支仍待判决），则不确定度也从“20”下降到“10”，因为值为“2”的最后五条指令均不再将不确定度添加到流水线 200。若两个分支都正确地判决，则流水线中的不确定度的级别是“0”，因为在流水线中不存在尚未判决的分支。

[0030] 图 4 示出可以实现为跟踪流水线 200 中的不确定度的表 400 的示例。在示例中，该表包括置信度值列 410，该列的每一行表示用于在流水线 200 中碰到分支时的指令的当前不确定度值总和。在如何跟踪用于无序跟踪 260 的指令组的示例中的“三元组”的实现从而指令可以按比每周期一条指令更快的速度按顺序进行架构式更新 270。对于所给定的示例，在三元组内，存在指令空隙 (slot)0、1 和 2。分支在任何空隙中受支持；然而，空隙 1 或 2 中的分支结束所给定的三元组内的指令。在第一行 420 中，指令 Instr 在流水线 200 中具有“0”不确定度。“0”的不确定度示出于 Instr 之下。在第二行 430 中，指令碰到具有“1”的不确定度的分支 Br(1)。用于指令的不确定度现在是“1”，其在 Instr 之下示出为“1”。在第二行 430 中，指令然后碰到具有不确定度“3”的分支 Br(3)。与指令关联的不确定度现在是在行 440 处列 410 中所放置的“4”。在行 440 中，指令在流水线 200 中具有在 Instr 之下的不确定度“4”。因此，在行 450 中，列 410 保持在不确定度“4”。然而，在行 450 中，指令碰到具有不确定度“2”的另一分支 Br(2)。与指令关联的不确定度现在是“6”。故此，在行 460 中，列 410 现在读取随流水线 200 中的指令保持的不确定度“6”。故此，应理解，用于三元组（即每个行）的不确定度是（三元组总和值＊三元组中的指令的数量）+（最旧的分支不确定度＊比三元组中的较旧的分支更年轻的指令的数量）。

[0031] 在块 330，系统 100 平衡流水线 200。在示例性实施例中，为了通过最有可能从每

个线程完成的指令来保持流水线 200 平衡，流水线 200 调度来自每个线程的指令 230，从而流水线 200 中的后调度的每线程的不确定度的级别得以平衡。关于无序发布引擎，每个线程具有其自身的使用期优先级和判断，在其之间，在两个线程之间发布的线程是基于同样保持不确定度的级别平衡的。以此方式，指令发送到最有可能执行工作的流水线中，而不是在稍后时间点被清除出流水线。具有较高置信度（较小不确定度）的指令因此被给定优先级。

[0032] 在流水线 200 中，排队指令用于发布。与传统流水线相似，流水线 200 按顺序取回 210、解码 220 并且调度 230 指令。然而，也与传统流水线相似，指令一般无序排队并且发布 240。在示例性实施例中，指令的发布考虑指令的不确定度以及指令的使用期。在平衡流水线 200 中，系统 100 考虑指令位置的不确定度，而在调度指令的同时，尽可能均匀地平衡流水线中的不确定度，同时还增加有可能执行影响架构化状态的工作的指令的数量。在示例性实施例中，为了保持流水线中的指令的不确定度相等，计算不确定度的比例并且根据该比例来调度指令。例如，如果线程 X 具有 50 的不确定度总和，线程 Y 具有 25 的不确定度总和，则不确定度的比例是 2/1。根据该比例，对于线程 Y 调度对于所调度的线程 X 的每条指令的两倍那么多的指令。以此方式，虽然 X 的不确定度将继续增加超过 50，但 Y 将以比 X 增加超过 50 更快的速度增加超过 25。故此，两个线程的流水线内的不确定度朝向不确定度均衡的点移动。通过达到均衡点，在统计上很可能最佳最小工作量将被清除出流水线 200。可以理解，因为分支可以判决并且不确定度可以改变，所以比例可以在任何时间改变。例如，对于线程 X 的不确定度可以达到 60，对于线程 Y 的不确定度可以达到 40，在此情况下，比例变为 3/2。在此情况下，对于线程 X 的每两条指令，将再次被调度对于线程 Y 的三条指令，以平衡流水线 200 中的不确定度并且增加用于影响机器的架构状态的待执行的工作的机会。在另一示例中，分支可以对于线程 X 判决，将不确定度降低到 30，由此具有 3/4 的比例。在此情况下，对于线程 X 的每四条指令，将再次被调度对于线程 Y 的三条指令，以平衡流水线 200 中的不确定度并且增加用于影响机器的架构状态的待执行的工作的机会。

[0033] 如在此所描述的那样，流水线平衡中的另一考量是考虑待发布的指令的使用期。传统上，流水线发布在发布队列中具有有效数据的最旧指令。在示例性实施例中，基于指令的使用期以及指令的不确定度来调度指令。对于每个线程，对于发布有效的最旧指令检查流水线 200 中到该点为止的指令的不确定度的总和，来自具有最低不确定度的给定线程的指令是待发布的指令。在不确定度在各线程间相等的情况下，于是可以应用用于发布的各线程之间的循环 (round-robin) 选择。例如，如果线程 X 使得其待发布的最旧的有效指令被定义为不确定度的总和为 30，且线程 Y 使得其待发布的最旧的有效指令被定义为不确定度的总和为 10，则发布用于线程 Y 的指令，如上所述。当在给定周期上多于一条的指令待发布时，对于在该给定周期上待由指令发布填充的每条指令空隙重复比较。

[0034] 在示例性实施例中，在此所描述的系统和方法实现静态水印方案。图 5 示意性示出根据示例性实施例的可以实现的静态水印发布队列 500 的示例。假定其它线程停止，静态水印建立用于待调度的一个线程的指令数量的阈值，从而给定的线程 B 在另一线程 A 没有可用于放置到发布队列中的任何指令的时间帧期间不挂起发布队列。在示例中，线程 A 和线程 B 都具有作为待调度的取回的功能的指令。在示例中，因为例如线程 A 可能经历很长的延迟指令取回，所以线程 B 拿走大部分资源。故此，大量线程 B 指令调度到发布队列

中，而 A 是取回待解码并且放置在发布队列中的附加指令的关闭 (off) 指令。传统上，发布队列将允许线程 B 继续使用资源。在示例性实施例中，设置阈值以确保通过为线程 A 保留发布队列中的空间而同样可以调度一些线程 A 指令。一旦用于线程 B 的指令达到水印级别 510（发布队列的给定百分比），就停止将指令从线程 B 调度到队列中。即使线程 A 并不准备调度指令，其也具有调度指令到的发布队列中所保留的 N 个空间。当线程 A 从长延迟操作恢复时，对于线程 A 的指令在发布队列中将存在可用的点。示例示出头指针 520 和尾指针 530。头指针 520 是循环发布队列的头，并且定义队列中的最旧指令，尾指针 530 是定义下一指令插入到队列中的什么地方的附加指针。在示例性实施例中，尾指针 530 指向在水印阈值 510 之后的第一空隙。B 被允许继续将指令调度到发布队列的尾指针位置中，直到到达水印 510。此时，尾指针移动一个点穿过水印，以表示下一指令待调度到的发布队列的位置。在完成长延迟操作之后的线程 A 的指令现在具有调度到的发布队列中的空隙。在该方案中，虽然不确定度的比例很有可能从均衡偏移离开，但水印已经防止该比例变得比其已经变为的更过度。

[0035] 在示例性实施例中，在此所描述的系统和方法可以实现从发布清除的方案。图 6 示意性示出根据示例性实施例的可以实现的从发放队列 600 清除的示例。可以从发布队列清除指令，从而在发布队列中存在所调度的指令的不确定度的平衡。在示例中，线程 A 和线程 B 都具有待调度的发布队列中的指令。因为例如线程 A 可能经历长延迟指令取回操作，所以线程 B 拿走大部分资源。故此，大量线程 B 指令调度到将两个线程的不确定度比例移动离开均衡的发布队列中。线程 A 将停止多长是未知的，故此，线程 B 可以通过使用整个发布队列来得到附加性能，因为其将另外在此保持未使用，对于线程 A，没有可用于放置到发布队列中的指令。在示例性实施例中，当用于线程 A 的指令变得可用于发布队列中的插入时，流水线 200 可以从发布队列清除线程 B 的 N 个更年轻的指令，并且在调度用于线程 A 的指令之后重新调度它们，这平衡了发布队列不确定度。头指针 610 是定义发布队列中的最旧指令的循环发布队列的头。尾指针 620 定义指令要插入到发布队列中的下一空隙。给定数量的 B 可以从队列被清除，B 指针可以移动得更靠近头指针 610，以建立待清除的多个线程 B 指令，如在此所描述的那样。在示例性实施例中，为了避免重新取回用于线程 B 的指令，指令可以保存在“虚假”（虚拟）指令队列中，其可以是在被调度 230 之前保存来自指令取回的指令的队列的扩展。

[0036] 如上所述，可以通过计算机实现的处理和用于实践这些处理的装置来对实施例进行实施。实施例可以包括具有包含在作为制造物的有形介质中实施的指令的计算机程序代码逻辑 704 的计算机可读 / 可用的介质 702 上的如图 7 中所描述的计算机程序产品 700。用于计算机可读 / 可用的介质 702 的示例性制造物可以包括软盘、CD-ROM、硬盘驱动器、通用串行总线 (USB) 闪速驱动器或任何其它计算机可读存储介质，其中，当计算机程序代码逻辑 704 加载到计算机中并且由计算机执行时，计算机变为用于实践本发明的装置。实施例包括计算机程序代码逻辑 704，例如，是否存储在存储介质中，加载到计算机中和 / 或由计算机执行，或通过一些传输介质（例如通过电引线或缆线，通过光纤或经由电磁辐射）而发送，其中，当计算机程序代码逻辑 704 加载到计算机中并且由计算机执行时，计算机变为实践本发明的装置。当在通用微处理器上实现时，计算机程序代码逻辑 704 分段配置微处理器以创建特定逻辑电路。

[0037] 所属技术领域的技术人员知道，本发明可以实现为系统、方法或计算机程序产品。因此，本公开可以具体实现为以下形式，即：可以是完全的硬件、也可以是完全的软件（包括固件、驻留软件、微代码等），还可以是硬件和软件结合的形式，本文一般称为“电路”、“模块”或“系统”。此外，在一些实施例中，本发明还可以实现为在一个或多个计算机可读介质中的计算机程序产品的形式，该计算机可读介质中包含计算机可读的程序代码。

[0038] 可以采用一个或多个计算机可读的介质的任意组合。计算机可读介质可以是计算机可读信号介质或者计算机可读存储介质。计算机可读存储介质例如可以是——但不限于——电、磁、光、电磁、红外线、或半导体的系统、装置或器件，或者任意以上的组合。计算机可读存储介质的更具体的例子（非穷举的列表）包括：具有一个或多个导线的电连接、便携式计算机磁盘、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)、光纤、便携式紧凑磁盘只读存储器 (CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。在本文件中，计算机可读存储介质可以是任何包含或存储程序的有形介质，该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

[0039] 计算机可读信号介质可以包括例如在基带中或作为载波的一部分具有在其中实施的计算机可读程序代码的传输数据信号。该传输信号可以采用任何多种形式，包括但不限于电磁、光或其任何合适的组合。计算机可读信号介质可以是并非计算机可读存储介质并且可以传递、传输、传送用于由指令执行系统、装置或设备使用或与之结合的程序的任何计算机可读介质。

[0040] 体现在计算机可读介质上的程序代码可以用任何适当的介质传输，所述介质包括但不限于：无线、有线、光缆、RF 等，或上述的任意合适的组合。

[0041] 可以以一种或多种程序设计语言或其组合来编写用于执行本发明操作的计算机程序代码，所述程序设计语言包括面向对象的程序设计语言—诸如 Java、Smalltalk、C++，还包括常规的过程式程序设计语言—诸如”C”语言或类似的程序设计语言。程序代码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）。

[0042] 本文中将参照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 / 或框图描述本发明。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，都可以由计算机程序指令实现。这些计算机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，这些计算机程序指令通过计算机或其它可编程数据处理装置执行，产生了实现流程图和 / 或框图中的方框中规定的功能 / 操作的装置。

[0043] 也可以把这些计算机程序指令存储在能使得计算机或其它可编程数据处理装置以特定方式工作的计算机可读介质中，这样，存储在计算机可读介质中的指令就产生出一个包括实现流程图和 / 或框图中的方框中规定的功能 / 操作的指令装置 (instruction means) 的制造品 (manufacture)。

[0044] 也可以把计算机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，从而使得在计算机或其它可编程装置上执行的指令能够提供实现流程图和／或框图中的方框中规定的功能／操作的过程。

[0045] 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、程序段或代码的一部分，所述模块、程序段或代码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。也应当注意，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操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。

[0046] 在此使用的术语仅是为了描述特定实施例，且不旨在限制本发明。如在此使用的，单数形式“一”、“一个”和“该”也旨在包括复数形式，除非上下文另外清楚地指明。还将理解，当在说明书中使用时，术语“包括”和／或“包含”指明存在所述的特征、整体、步骤、操作、元件和／或组件，但不排除存在或附加一个或多个其他特征、整体、步骤、操作、元件和／或组件。

[0047] 所附权利要求书中的所有装置或步骤加功能元件的相应结构、材料、操作以及等价物，如有的话，旨在包括用于结合如特别要求保护的其他所要求保护的元件来执行所述功能的任何结构、材料或操作。呈现本发明的说明是为了示出和描述的作用，但不是穷尽性的或将本发明限制于所公开的形式。许多修改和变化对本领域普通技术人员来说是明显的，且不脱离本发明的范围。选择和描述实施例是为了最佳地解释本发明的原理和实际应用，并使得本领域普通技术人员能针对适于考虑的特定用途的具有各种修改的各种实施例理解本发明。

[0048] 在此所描述的流程图仅为一个示例。在不脱离本发明的精神的情况下，可以存在对于在此所描述的该图或步骤（或操作）的很多变形。例如，可以按不同的顺序来执行步骤，或可以添加、删除或修改步骤。所有这些变形被看作本发明的一部分。

[0049] 虽然已经描述了本发明优选实施例，但本领域技术人员应理解，在现在和将来，可以进行落入所附权利要求的范围内的各种改进和增强。这些权利要求应理解为保持首先描述的本发明的正确保护范围。

[0050] 在此所描述的流程图仅为一个示例。在不脱离本发明的精神的情况下，可以存在对于在此所描述的该图或步骤（或操作）的很多变形。例如，可以按不同的顺序来执行步骤，或可以添加、删除或修改步骤。所有这些变形被看作本发明的一部分。

[0051] 虽然已经描述了本发明优选实施例，但本领域技术人员应理解，在现在和将来，可以进行落入所附权利要求的范围内的各种改进和增强。这些权利要求应理解为保持首先描述的本发明的正确保护范围。

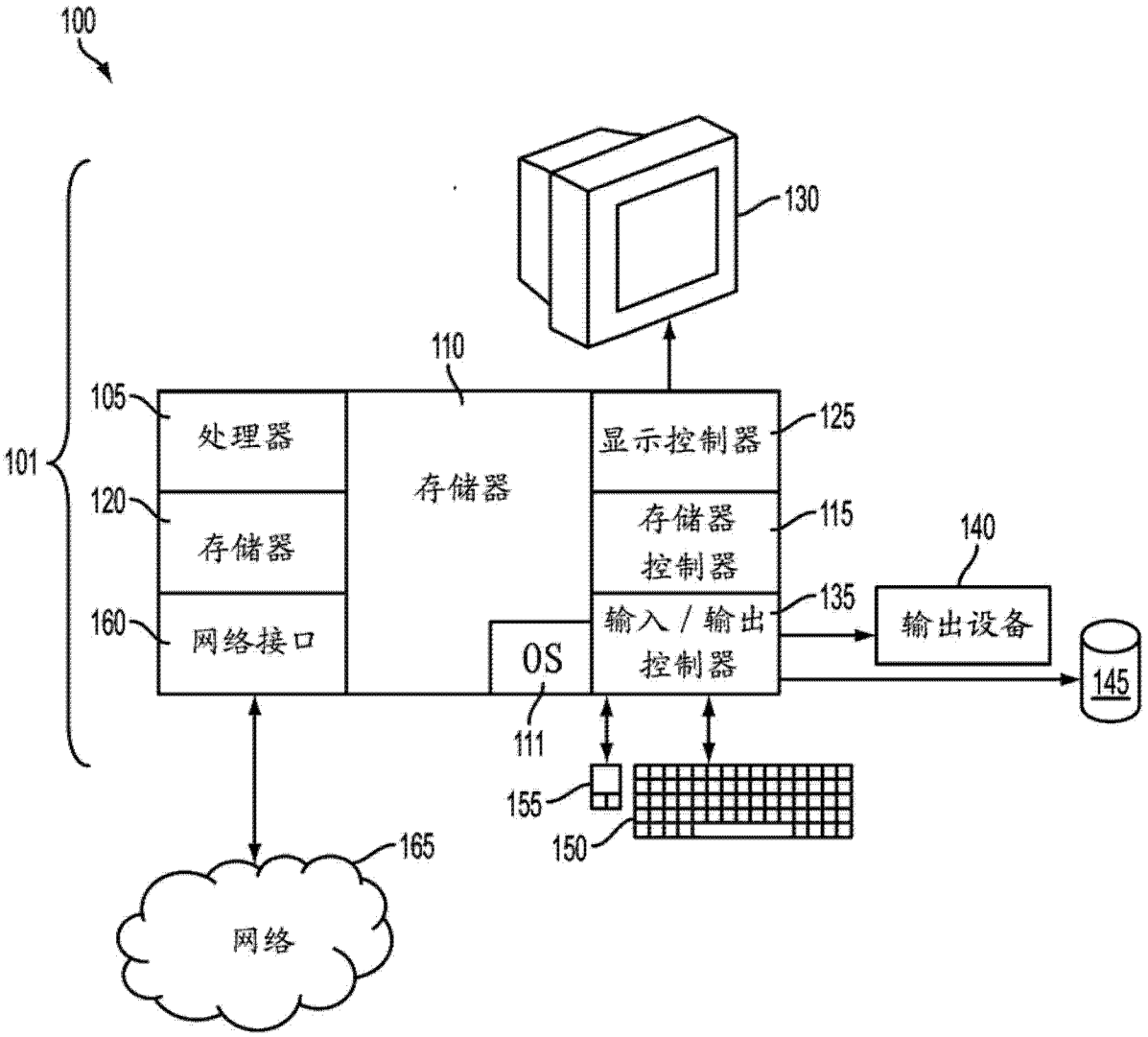

100
101
105
处理器
120
存储器
160
网络接口
110
存储器
111
OS
130
125
显示控制器
115
存储器
控制器
135
输入 / 输出
控制器
140
输出设备
145
155
150
165
网络

图 1

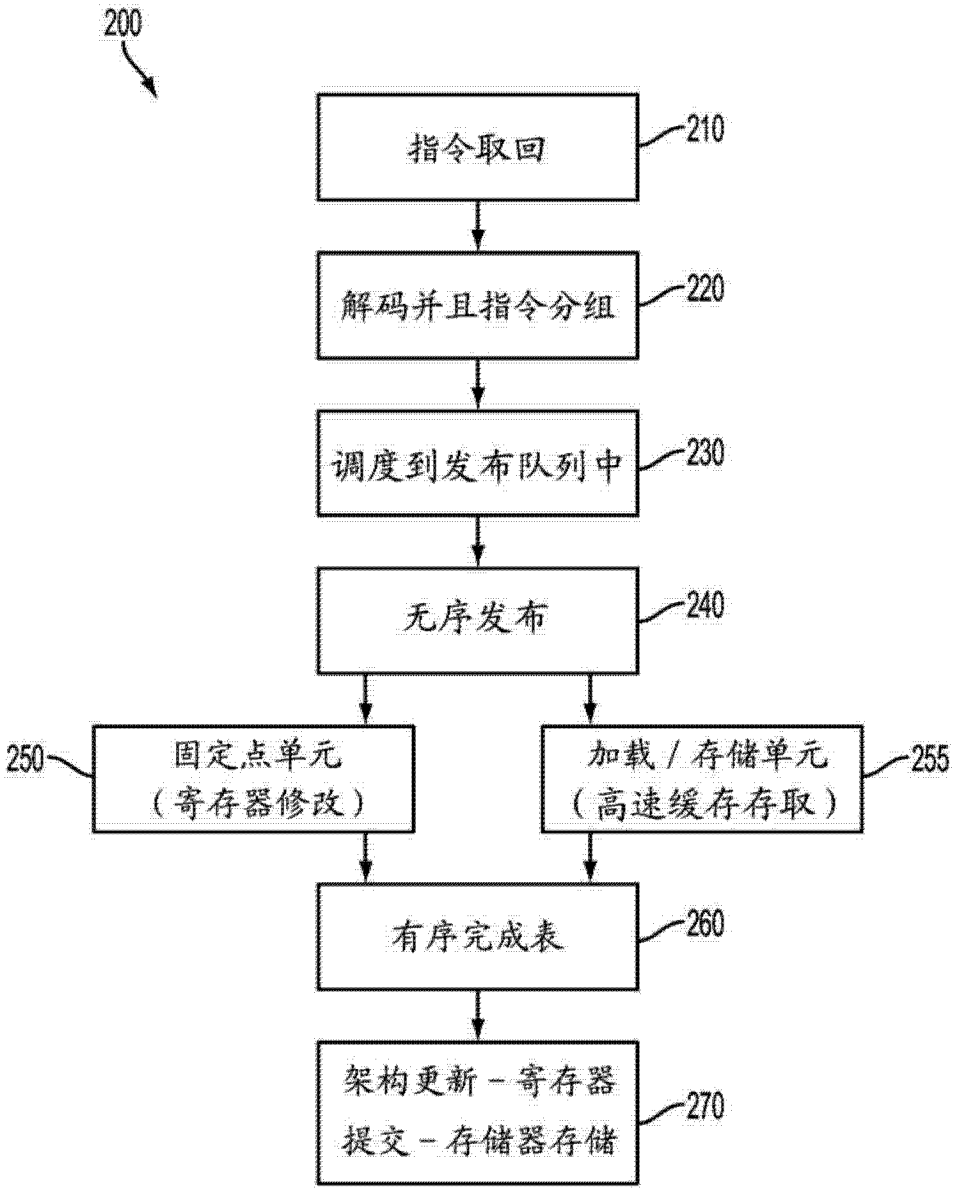

200
指令取回
210
解码并且指令分组
220
调度到发布队列中
230
无序发布
240
250
固定点单元
（寄存器修改）
加载 / 存储单元
（高速缓存存取）
255
有序完成表
260
架构更新 - 寄存器
提交 - 存储器存储
270

图 2

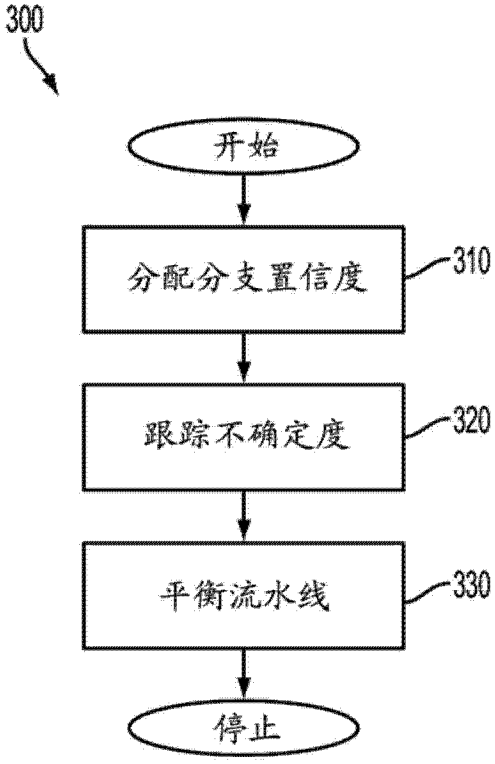


图 3

400

410

420	0	INSTR 0	INSTR 0	INSTR 0
430	0	BR (1) 0	INSTR 1	BR (3) 1
440	4	INSTR 4	INSTR 4	INSTR 4
450	4	INSTR 4	BR (2) 4	N/A -
460	6	INSTR 6	INSTR 6	INSTR 6

图 4

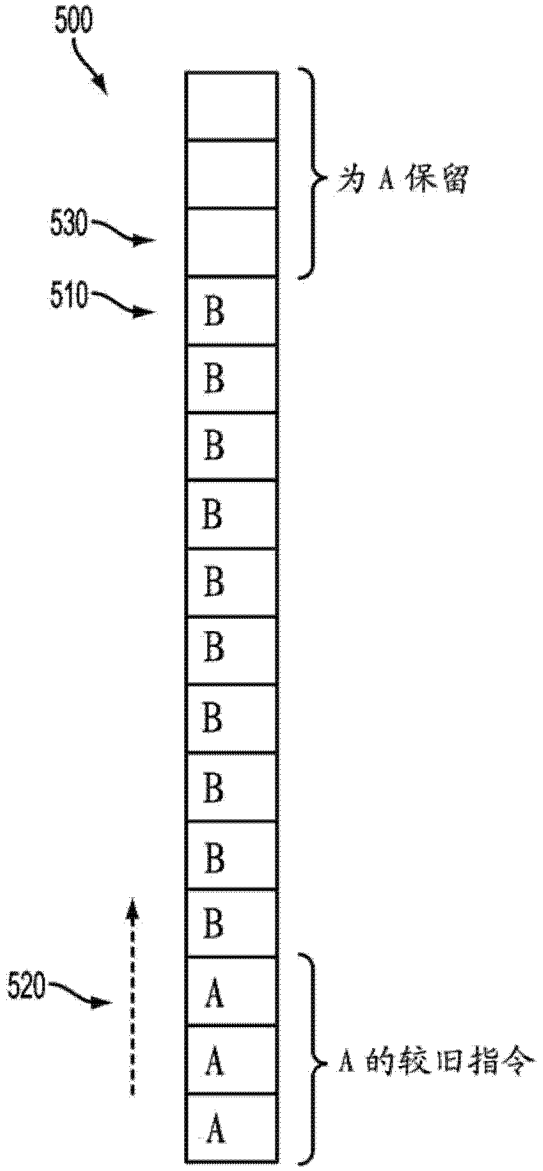

500
为 A 保留
530
510
B
B
B
B
B
B
B
B
B
B
B
520
A
A
A
A 的较旧指令

图 5

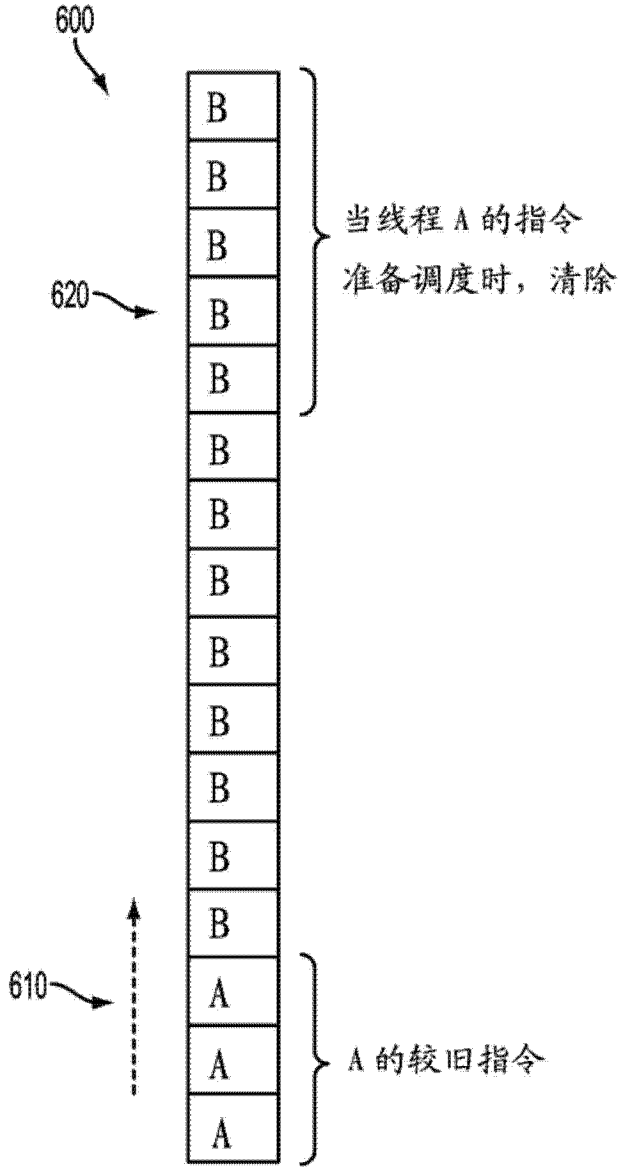

600
B
B
B
B
B
当线程 A 的指令
准备调度时，清除
620
B
B
B
B
B
B
B
B
B
B
610
A
A
A
A 的较旧指令

图 6

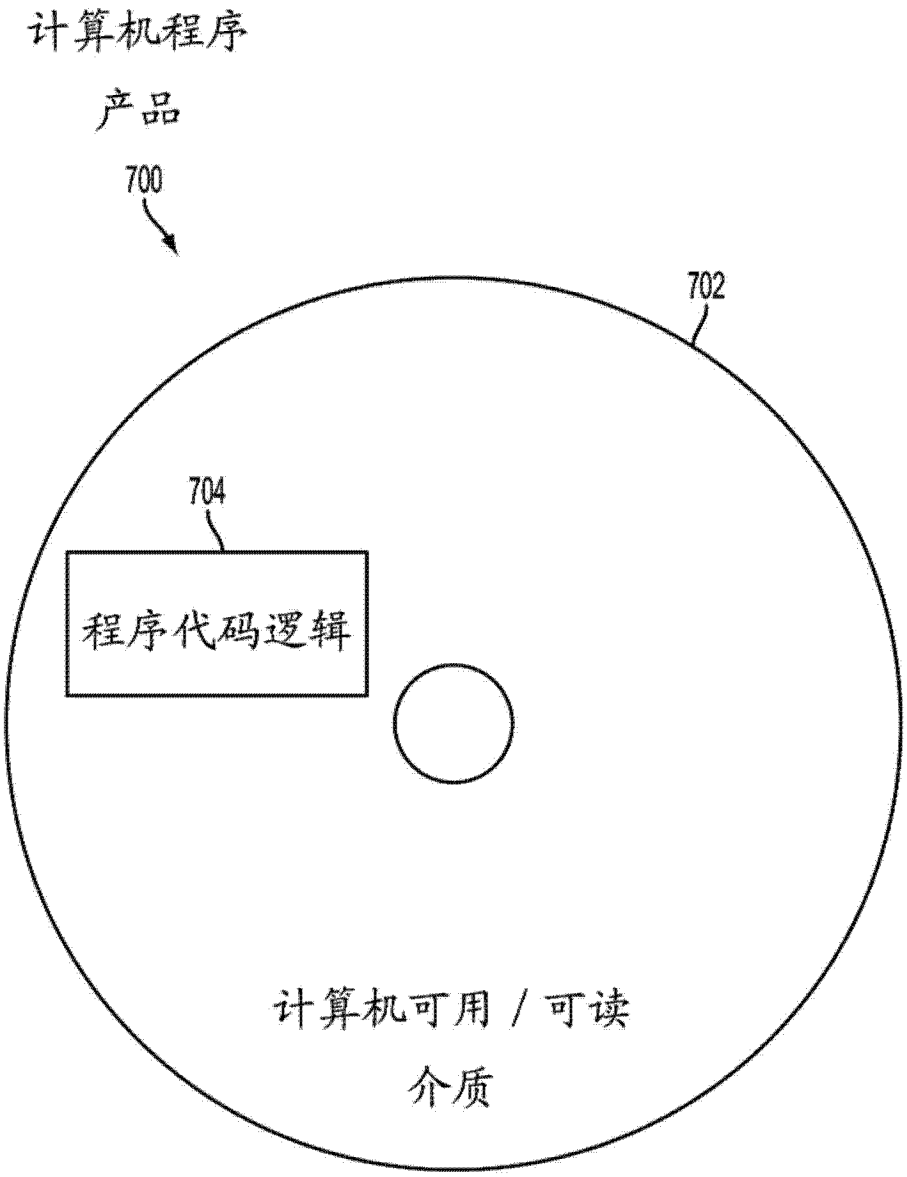


图 7

1. 一种用于流水线中的指令执行的计算机实现的方法，所述方法包括：

在所述流水线中从多个线程取回多条指令，其中，所述多条指令包括多条分支指令；

对于所述多条分支指令中的每一条，将分支不确定度分配给所述多条分支指令中的每一条；

对于所述多条指令中的每一条，分配指令不确定度，其为较旧的未判决的分支的分支不确定度的总和，其中分配给一线程中的第一指令的指令不确定度不同于分配给所述线程中的第二指令的指令不确定度；以及

在所述流水线中基于指令不确定度的当前总和来平衡所述多个线程之间的指令。

2. 如权利要求 1 所述的方法，还包括：分配用于所述多个线程中的每一个的多个不确定度。

3. 如权利要求 2 所述的方法，还包括：根据所述多个不确定度来计算多个不确定度比例。

4. 如权利要求 2 所述的方法，还包括：在所述流水线中调度所述多条指令。

5. 如权利要求 2 所述的方法，其中，与来自所述多个不确定度的比例成比例地调度指令。

6. 如权利要求 1 所述的方法，还包括：计算总不确定度，其中，所述总不确定度是用于所述多条指令中的每一个的所述指令不确定度的总和。

7. 如权利要求 1 所述的方法，还包括：保留用于所述多条指令的子集的发布队列空间。

8. 如权利要求 1 所述的方法，还包括：响应于第二子集指令变为可用于调度而从所述流水线清除第一指令子集。

9. 一种系统，包括适用于执行如任何前述方法权利要求所述的方法的所有步骤的装置。

10. 一种计算机程序，包括用于执行如任何前述方法权利要求所述的方法的所有步骤的指令。