

(12) 发明专利申请

(10) 申请公布号 CN 103218259 A

(43) 申请公布日 2013.07.24

(21) 申请号 201310018618.6

(51) Int. Cl.

(22) 申请日 2013.01.18

G06F 9/48 (2006.01)

G06F 15/163 (2006.01)

(30) 优先权数据

13/353,150 2012.01.18 US

13/353,155 2012.01.18 US

(71) 申请人 辉达公司

地址 美国加利福尼亚州

(72) 发明人 卡里姆·M·阿夫达利亚

兰基·V·姗 杰尔姆·F·小杜鲁克

蒂莫西·约翰·珀塞尔

坦莫尼·曼德尔 广田源太郎

(74) 专利代理机构 北京市磐华律师事务所

11336

代理人 徐丁峰 魏宁

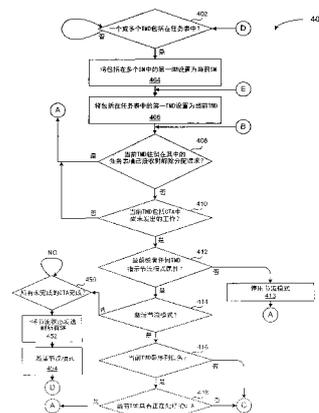
权利要求书2页 说明书14页 附图8页

(54) 发明名称

计算任务的调度和执行

(57) 摘要

本发明的一个实施例阐述了用于选择包括在多个处理器中的第一处理器以接收与计算任务相关的工作的技术。该技术涉及分析多个处理器中的每一个处理器的状态数据以识别已经指派了一个计算任务并且有资格接收与该一个计算任务相关的工作的一个或多个处理器，从识别为有资格的一个或多个处理器的每一个接收指示处理器接收新任务的能力的可用性值，基于从该一个或多个处理器接收的可用性值来选择第一处理器以接收与该一个计算任务相关的工作，并且经由协作线程阵列 (CTA) 向该第一处理器发出与该一个计算任务相关的工作。



1. 一种用于选择包括在多个处理器中的第一处理器以接收与计算任务相关的工作的计算机实现的方法,所述方法包括:

分析所述多个处理器中的每个处理器的状态数据以识别已被指派一个计算任务并且有资格接收与所述一个计算任务相关的工作的一个或多个处理器;

从识别为有资格的所述一个或多个处理器的每一个接收指示所述处理器接收新工作的能力的可用性值;

基于从所述一个或多个处理器所接收的所述可用性值来选择第一处理器以接收与所述一个计算任务相关的工作;以及

经由协作线程阵列 (CTA) 向所述第一处理器发出与所述一个计算任务相关的所述工作。

2. 根据权利要求 1 所述的计算机实现的方法,其中当与所述一个计算任务相关联的所述状态数据已由处理器所接收并确认时,处理器被识别为有资格的。

3. 根据权利要求 1 所述的计算机实现的方法,其中当所述一个计算任务与数目大于或等于由所述一个计算任务所指示的每 CTA 工作项的阈值数目的未完成工作项相关联时,处理器被识别为有资格的。

4. 根据权利要求 1 所述的计算机实现的方法,其中当已出现超时期,并且与所述一个计算任务相关联的未完成工作项的数目不超过由所述一个计算任务所指示的每 CTA 工作项的阈值数目时,处理器被识别为有资格的。

5. 根据权利要求 1 所述的计算机实现的方法,其中当所述一个计算任务指示节流模式应被激活并且所述多个处理器正操作在所述节流模式中时,处理器被识别为有资格的,以及其中在所述节流模式中,所述第一处理器包括在所述多个处理器的受限子集中并且在所述受限子集内的每个处理器均被允许访问大于存储器第二部分的存储器第一部分,所述存储器第二部分通常在非节流模式中处理计算任务时对所述多个处理器中的每个处理器可用。

6. 一种用于将计算任务指派给包括在多个处理器中的第一处理器的计算机实现的方法,所述方法包括:

分析多个计算任务中的每个计算任务以识别有资格指派给所述第一处理器的一个或多个计算任务,其中每个计算任务均列出在第一表中并且与优先级值和指示所述计算任务添加到所述第一表的时间的分配顺序相关联;

基于所述优先级值和所述分配顺序中的至少一个来从所识别的一个或多个计算任务中选择第一计算任务;以及

指派所述第一计算任务给所述第一处理器用于执行。

7. 根据权利要求 6 所述的计算机实现的方法,其中当与计算任务相关联的解除分配请求尚未发出时,所述计算任务被识别为有资格的。

8. 根据权利要求 6 所述的计算机实现的方法,其中当计算任务包括尚未经由协作线程阵列 (CTA) 发出到所述多个处理器中的任何所述处理器的工作时,所述计算任务被识别为有资格的。

9. 根据权利要求 6 所述的计算机实现的方法,其中当必须在节流模式中对计算任务进行处理时,所述计算任务被识别为有资格的,以及其中在所述节流模式中,所述第一处理器

包括在所述多个处理器的受限子集中并且在所述受限子集内的每个处理器均被允许访问大于存储器第二部分的存储器第一部分,所述存储器第二部分通常在非节流模式中处理计算任务时对所述多个处理器中的每个处理器可用。

10. 根据权利要求 6 所述的计算机实现的方法,其中当计算任务要求在任何给定时间仅能执行一个 CTA 并且没有与所述计算任务相关联的 CTA 当前正在由所述多个处理器中的任何所述处理器所执行时,所述计算任务被识别为有资格的。

11. 根据权利要求 6 所述的计算机实现的方法,其中当与计算任务相关联的亲规则和规则不禁止与所述计算任务相关联的任何所述 CTA 由所述第一处理器所执行时,所述计算任务被识别为有资格的。

12. 根据权利要求 6 所述的计算机实现的方法,其中当与计算任务相关联的所执行的 CTA 的数目尚未达到阈值时,所述计算任务被识别为有资格的。

计算任务的调度和执行

技术领域

[0001] 本发明大体涉及计算任务,并且更具体地,涉及计算任务的调度和执行。

背景技术

[0002] 用于在多处理器系统中执行的计算任务的常规调度依赖于应用程序或驱动程序。在计算任务的执行期间,允许驱动程序调度计算任务而所需的驱动程序和多处理器之间的交互可能延迟计算任务的执行。

[0003] 因此,本领域需要的是用于基于处理资源和可用计算任务的优先级来动态调度计算任务用于执行的系统和方法。重要的是,调度机制不应取决于或要求软件或驱动程序交互。

发明内容

[0004] 本发明的一个实施例阐述了用于选择包括在多个处理器中的第一处理器以接收与计算任务相关的工作的方法。该方法涉及分析多个处理器中的每个处理器的状态数据以识别已经被指派一个计算任务并且有资格接收与所述一个计算任务相关的工作的一个或多个处理器,从识别为有资格的所述一个或多个处理器中的每一个接收指示处理器接收新任务的能力的可用性值,基于从所述一个或多个处理器所接收的可用性值来选择第一处理器以接收与所述一个计算任务相关的工作,并且经由协作线程阵列 (CTA) 将与所述一个计算任务相关的工作发出给第一处理器。

[0005] 本发明的另一个实施例阐述了用于将计算任务指派给包括在多个处理器中的第一处理器的方法。该方法涉及分析多个计算任务中的每个计算任务来识别有资格用于指派给第一处理器的一个或多个计算任务,其中每个计算任务均在第一表中列出,并且与优先级值和指示将所述计算任务添加到第一表的时间的分配顺序相关联。该技术进一步涉及基于优先级值和分配顺序中的至少一个来从所识别的一个或多个计算任务中选择第一计算任务,并且将第一计算任务指派给第一处理器用于执行。

[0006] 进一步的实施例提供了非暂时性计算机可读介质和计算机系统以实现上述所阐述的各个方法。

附图说明

[0007] 因此,可以详细地理解本发明的上述特征,并且可以参考实施例得到对如上面所简要概括的本发明更具体的描述,其中一些实施例在附图中示出。然而,应当注意的是,附图仅示出了本发明的典型实施例,因此不应被认为是对其范围的限制,本发明可以具有其他等效的实施例。

[0008] 图 1 是示出了配置为实现本发明的一个或多个方面的计算机系统的框图。

[0009] 图 2 是根据本发明的一个实施例的,用于图 1 的计算机系统的并行处理子系统的框图。

[0010] 图 3A 是根据本发明的一个实施例的,图 2 的任务 / 工作单元的框图。

[0011] 图 3B 是根据本发明的一个实施例的,在图 2 的并行处理单元之一内的通用处理集群的框图。

[0012] 图 3C 是根据本发明的一个实施例的,图 3B 的流多处理器的一部分的框图。

[0013] 图 4A-4B 示出了根据本发明的一个实施例的,用于将任务指派给图 3A-3C 的流多处理器 (SM) 的方法。

[0014] 图 5 示出了根据本发明的一个实施例的,用于选择 SM 以接收与任务相关的工作的方法。

具体实施方式

[0015] 在下面的描述中,将阐述大量的特定细节以提供对本发明更透彻的理解。然而,本领域的技术人员应该清楚,本发明可以在没有或多个这些特定细节的情况下得以实施。

[0016] 系统概述

[0017] 图 1 为示出了配置为实现本发明的一个或多个方面的计算机系统 100 的框图。计算机系统 100 包括中央处理单元 (CPU) 102 和经由可以包括存储器桥 105 的互连路径通信的系统存储器 104。存储器桥 105 可以是例如北桥芯片,经由总线或其他通信路径 106 (例如超传输 (HyperTransport) 链路) 连接到 I/O (输入 / 输出) 桥 107。I/O 桥 107,其可以是例如南桥芯片,从一个或多个用户输入设备 108 (例如键盘、鼠标) 接收用户输入并且经由通信路径 106 和存储器桥 105 将所述输入转发到 CPU102。并行处理子系统 112 经由总线或第二通信路径 113 (例如外围部件互连 (PCI) Express、加速图形端口或超传输链路) 耦合到存储器桥 105;在一个实施例中,并行处理子系统 112 是将像素传递到显示设备 110 (例如传统的基于阴极射线管或液晶显示器的监视器) 的图形子系统。系统盘 114 也连接到 I/O 桥 107。交换器 116 提供 I/O 桥 107 与诸如网络适配器 118 以及各种插卡 120 和 121 的其他部件之间的连接。其他部件 (未明确示出),包括通用串行总线 (USB) 或其他端口连接、压缩磁盘 (CD) 驱动器、数字视频光盘 (DVD) 驱动器、胶片录制设备及类似部件,也可以连接到 I/O 桥 107。图 1 所示的各种通信路径包括特殊命名的通信路径 106 和 113 可以使用任何适合的协议实现,诸如 PCI-Express、AGP (加速图形端口)、超传输或者任何其他总线或点到点通信协议,并且如本领域已知的,不同设备间的连接可使用不同协议。

[0018] 在一个实施例中,并行处理子系统 112 包含经优化用于图形和视频处理的电路,包括例如视频输出电路,并且构成图形处理单元 (GPU)。在另一个实施例中,并行处理子系统 112 包含经优化用于通用处理的电路,同时保留底层 (underlying) 的计算架构,本文将更详细地进行描述。在又一个实施例中,可以将并行处理子系统 112 与一个或多个其他系统元件集成在单个子系统中,诸如结合存储器桥 105、CPU102 以及 I/O 桥 107,以形成片上系统 (SoC)。

[0019] 应该理解,本文所示系统是示例性的,并且变化和修改都是可能的。连接拓扑,包括桥的数量和布置、CPU102 的数量以及并行处理子系统 112 的数量,可根据需要修改。例如,在一些实施例中,系统存储器 104 直接连接到 CPU102 而不是通过桥,并且其他设备经由存储器桥 105 和 CPU102 与系统存储器 104 通信。在其他替代性拓扑中,并行处理子系统 112

连接到 I/O 桥 107 或直接连接到 CPU102,而不是连接到存储器桥 105。而在其他实施例中, I/O 桥 107 和存储器桥 105 可能被集成到单个芯片上而不是作为一个或多个分立设备存在。大型实施例可以包括两个或两个以上的 CPU102 以及两个或两个以上的并行处理系统 112。本文所示的特定部件是可选的;例如,任意数量的插卡或外围设备都可能得到支持。在一些实施例中,交换器 116 被去掉,网络适配器 118 和插卡 120、121 直接连接到 I/O 桥 107。

[0020] 图 2 示出了根据本发明一个实施例的并行处理子系统 112。如图所示,并行处理子系统 112 包括一个或多个并行处理单元 (PPU) 202,每个并行处理单元 202 都耦合到本地并行处理 (PP) 存储器 204。通常,并行处理子系统包括 U 个 PPU,其中 $U \geq 1$ 。(本文中,类似对象的多个实体以标识该对象的参考数字和需要时标识所述实体的括号中的数字来表示。)PPU202 和并行处理存储器 204 可使用一个或多个集成电路设备来实现,诸如可编程处理器、专用集成电路 (ASIC) 或存储器设备,或者以任何其他技术可行的方式来实现。

[0021] 再参考图 1 以及图 2,在一些实施例中,并行处理子系统 112 中的一些或所有 PPU202 是具有渲染管线的图形处理器,它可以配置为实施与下述相关的各种操作:经由存储器桥 105 和第二通信路径 113 从 CPU102 和 / 或系统存储器 104 所提供的图形数据生成像素数据,与本地并行处理存储器 204(可被用作图形存储器,包括例如常用帧缓冲区 (buffer)) 交互以存储和更新像素数据,传递像素数据到显示设备 110 等等。在一些实施例中,并行处理子系统 112 可包括一个或多个作为图形处理器而操作的 PPU202 以及包括一个或多个用于通用计算的其他 PPU202。这些 PPU 可以是相同的或不同的,并且每个 PPU 均可具有专用并行处理存储器设备或不具有专用的并行处理存储器设备。并行处理子系统 112 中的一个或多个 PPU202 可输出数据到显示设备 110,或者并行处理子系统 112 中的每个 PPU202 均可输出数据到一个或多个显示设备 110。

[0022] 在操作中,CPU102 是计算机系统 100 的主处理器,控制和协调其他系统部件的操作。具体地,CPU102 发出控制 PPU202 的操作的命令。在一些实施例中,CPU102 为每个 PPU202 写入命令流到数据结构中(在图 1 或图 2 中未明确示出),所述数据结构可位于系统存储器 104、并行处理存储器 204、或 CPU102 和 PPU202 都可访问的其他存储位置中。将指向每个数据结构的指针写到入栈缓冲区 (pushbuffer) 以发起对数据结构中的命令流的处理。PPU202 从一个或多个入栈缓冲区读取命令流,然后相对于 CPU102 的操作异步地执行命令。可以经由设备驱动程序 103 由应用程序为每个入栈缓冲区指定执行优先级以控制对不同入栈缓冲区的调度。

[0023] 现在返回参考图 2 和图 1,每个 PPU202 均包括经由连接到存储器桥 105(或者,在一个替代性实施例中,直接连接到 CPU102) 的通信路径 113 与计算机系统 100 的其余部分通信的 I/O(输入 / 输出)单元 205。PPU202 到计算机系统 100 的其余部分的连接也可以变化。在一些实施例中,并行处理子系统 112 可作为插卡来实现,所述插卡可被插入到计算机系统 100 的扩展槽中。在其他实施例中,PPU202 可以和诸如存储器桥 105 或 I/O 桥 107 的总线桥一起集成在单个芯片上。而在其他实施例中,PPU202 的一些或所有元件可以和 CPU102 一起集成在单个芯片上。

[0024] 在一个实施例中,通信路径 113 是 PCI-EXPRESS 链路,如本领域所知的,其中专用通道被分配到每个 PPU202。也可以使用其他通信路径。I/O 单元 205 生成用于在通信路径 113 上传输的数据包(或其他信号),并且还从通信路径 113 接收所有传入的数据包(或其

他信号),将传入的数据包引导到PPU202的适当部件。例如,可将与处理任务相关的命令引导到主机接口206,而可将与存储器操作相关的命令(例如,对并行处理存储器204的读取或写入)引导到存储器交叉开关单元210。主机接口206读取每个入栈缓冲区,并且将存储在入栈缓冲区中的命令流输出到前端212。

[0025] 有利地,每个PPU202都实现高度并行处理架构。如详细示出的,PPU202(0)包括处理集群阵列230,该阵列230包括C个通用处理集群(GPC)208,其中 $C \geq 1$ 。每个GPC208都能够并发执行大量的(例如,几百或几千)线程,其中每个线程均是程序的实例(instance)。在各种应用中,可分配不同的GPC208用于处理不同类型的程序或用于执行不同类型的计算。取决于因每种类型的程序或计算所产生的工作量,GPC208的分配可以变化。

[0026] GPC208从任务/工作单元207内的工作分布单元接收所要执行的处理任务。所述工作分布单元接收指向编码为任务元数据(TMD)并存储在存储器中的处理任务的指针。指向TMD的指针包括在存储为入栈缓冲区并由前端单元212从主机接口206接收的命令流中。可被编码为TMD的处理任务可包括在所要处理的数据阵列内的索引,以及定义数据将被如何处理(例如,什么程序将被执行)的状态参数和命令。任务/工作单元207从前端212接收任务并确保在每一个TMD所指定的处理发起前,将GPC208配置为有效状态。可以为每个TMD指定用来调度处理任务的执行的优先级。还可从处理集群阵列230接收处理任务。可选地,TMD可包括控制是否将TMD添加到处理任务列表(或指向处理任务的指针列表)的头部或尾部的参数,从而提供除优先级以外的另一级别的控制。

[0027] 存储器接口214包括D个分区单元215,每个分区单元215均直接耦合到一部分并行处理存储器204,其中 $D \geq 1$ 。如所示的,分区单元215的数量一般等于动态随机存取存储器(DRAM)220的数量。在其他实施例中,分区单元215的数量也可以不等于存储器设备的数量。本领域的技术人员应该理解DRAM220可以用其他合适的存储设备来替代并且可以是一般常规的设计。因此省略了详细描述。诸如帧缓冲区或纹理映射图的渲染目标可以跨DRAM220加以存储,这允许分区单元215并行写入每个渲染目标的各部分以有效地使用并行处理存储器204的可用带宽。

[0028] 任意一个GPC208都可以处理要被写到并行处理存储器204内的任意DRAM220的数据。交叉开关单元210配置为路由每个GPC208的输出到任意分区单元215的输入或到另一个GPC208用于进一步处理。GPC208通过交叉开关单元210与存储器接口214通信,以对各种外部存储器设备进行读取或写入。在一个实施例中,交叉开关单元210具有到存储器接口214的连接以和I/O单元205通信,以及到本地并行处理存储器204的连接,从而使在不同GPC208内的处理内核能够与系统存储器104或对于PPU202而言非本地的其他存储器通信。在图2所示的实施例中,交叉开关单元210直接与I/O单元205连接。交叉开关单元210可使用虚拟信道来分开GPC208与分区单元215之间的业务流。

[0029] 另外,GPC208可被编程以执行与种类繁多的应用相关的处理任务,包括但不限于,线性和非线性数据变换、视频和/或音频数据过滤、建模操作(例如,应用物理定律以确定对象的位置、速率和其他属性)、图像渲染操作(例如,曲面细分(tessellation)着色、顶点着色、几何着色、和/或像素着色程序)等等。PPU202可将数据从系统存储器104和/或本地并行处理存储器204转移到内部(片上)存储器中,处理所述数据,并且将结果数据写回到系统存储器104和/或本地并行处理存储器204,其中这样的数据可以由其他系统部件访

问,所述其他系统部件包括 CPU102 或另一个并行处理子系统 112。

[0030] PPU202 可配备有任意容量 (amount) 的本地并行处理存储器 204,包括没有本地存储器,并且可以以任意组合方式使用本地存储器和系统存储器。例如,在统一存储器架构 (UMA) 实施例中,PPU202 可以是图形处理器。在这样的实施例中,将不提供或几乎不提供专用的图形 (并行处理) 存储器,并且 PPU202 会以排他或几乎排他的方式使用系统存储器。在 UMA 实施例中,PPU202 可集成到桥式芯片中或处理器芯片中,或作为具有高速链路 (例如,PCI-EXPRESS) 的分立芯片提供,所述高速链路经由桥式芯片或其他通信手段将 PPU202 连接到系统存储器。

[0031] 如上所述,在并行处理子系统 112 中可以包括任意数量的 PPU202。例如,可在单个插卡上提供多个 PPU202、或可将多个插卡连接到通信路径 113、或可将一个或多个 PPU202 集成到桥式芯片中。在多 PPU 系统中的 PPU202 可以彼此相同或不同。例如,不同的 PPU202 可能具有不同数量的处理内核、不同容量的本地并行处理存储器等等。在存在多个 PPU202 的情况下,可并行操作那些 PPU 从而以高于单个 PPU202 所可能达到的吞吐量来处理数据。包含一个或多个 PPU202 的系统可以以各种配置和形式因素来实现,包括台式电脑、笔记本电脑或手持式个人计算机、服务器、工作站、游戏控制台、嵌入式系统等等。

[0032] 多个并发任务调度

[0033] 可以在 GPC208 上并发执行多个处理任务并且处理任务在执行期间可以生成一个或多个“子”处理任务。任务 / 工作单元 207 接收任务并动态调度处理任务和子处理任务以由 GPC208 执行。

[0034] 图 3A 为根据本发明一个实施例的图 2 的任务 / 工作单元 207 的框图。任务 / 工作单元 207 包括任务管理单元 300 和工作分布单元 340,以及状态 304 (下面结合图 4A-4B 详细描述其内容)。任务管理单元 300 基于执行优先级级别来组织所要调度的任务。对于每个优先级级别,任务管理单元 300 将指向与任务相对应的 TMD322 的指针列表存储在调度器表 321 中,其中所述列表可以实现为链表。可以将 TMD322 存储在 PP 存储器 204 或系统存储器 104 中。任务管理单元 300 接受任务并将任务存储在调度器表 321 中的速度与任务管理单元 300 调度任务以执行的速度是解耦的。因此,任务管理单元 300 可以在调度任务之前收集若干任务。如本文所进一步详细描述的,每个 TMD322 均包括与在 PPU202 内处置 TMD322 的方式相关的状态 324。

[0035] 工作分布单元 340 包括具有槽的任务表 345,每个槽均可以被用于正在执行的任务的 TMD322 所占用。当任务表 345 中有空闲槽时,任务管理单元 300 可以调度任务以执行。当没有空闲槽时,未占用槽的较高优先级任务可以驱逐占用槽的较低优先级任务。当任务被驱逐时,该任务被停止,并且如果该任务的执行没有完成,则将指向该任务的指针添加到所要调度的任务指针列表以使得任务的执行稍后将恢复。在一些实施例中,恢复任务的位置存储在任务的 TMD322 中。当生成子处理任务时,在任务的执行期间,将指向该子任务的指针添加到所要调度的任务指针列表。可以由在处理集群阵列 230 中执行的 TMD322 生成子任务。如本文所进一步描述的,工作分布单元 340 还包括流多处理器 (SM) 状态 342,其存储状态数据用于包括在 PPU202 中的每个 SM310。

[0036] 不同于由任务 / 工作单元 207 从前端 212 接收的任务,子任务从处理集群阵列 230 接收。子任务不被插入帧缓冲区或传输到前端。当生成子任务或将用于子任务的数据存储

在存储器中时不通知 CPU102。通过帧缓冲区提供的任务与子任务之间的另一个区别是通过帧缓冲区提供的任务由应用程序来定义而子任务是在任务执行期间自动生成的。

[0037] 任务处理概述

[0038] 图 3B 为根据本发明一个实施例的在图 2 的 PPU202 之一内的 GPC208 的框图。每个 GPC208 均可配置为并行执行大量线程，其中术语“线程”是指在特定输入数据集上执行的特定程序的实例。在一些实施例中，单指令、多数据 (SIMD) 指令发出技术用于在不提供多个独立指令单元的情况下支持大量线程的并行执行。在其他实施例中，单指令、多线程 (SIMT) 技术用于使用配置为向 GPC208 中的每一个内的处理引擎集发出指令的共有指令单元来支持大量一般来说同步的线程的并行执行。不同于所有处理引擎通常都执行相同指令的 SIMD 执行机制，SIMT 执行通过给定线程程序允许不同线程更容易跟随分散执行路径。本领域普通技术人员应该理解 SIMD 处理机制代表 SIMT 处理机制的功能子集。

[0039] 经由将处理任务分布到流多处理器 (SM) 310 的管线管理器 305 来有利地控制 GPC208 的操作。管线管理器 305 还可配置为通过为由 SM310 所输出的处理数据指定目的地来控制工作分布交叉开关 330。

[0040] 在一个实施例中，每个 GPC208 均包括 M 个 SM310，其中 $M \geq 1$ ，每个 SM310 均配置为处理一个或多个线程组。另外，如本领域已知的，每个 SM310 均有利地包括可以管线化的相同的功能执行单元集（例如执行单元和加载 - 存储单元 - 作为 Exec 单元 302 和 LSU303 在图 3C 中示出），其允许在前一个指令完成之前发出新指令。可提供功能执行单元的任意组合。在一个实施例中，功能单元支持各种各样的操作，包括整数和浮点运算（例如加法和乘法）、比较操作、布尔操作 (AND、OR、XOR)、移位和各种代数函数的计算（例如平面插值、三角函数、指数函数和对数函数等等）；以及相同的功能单元硬件可均衡的用来 (be leveraged to) 实施不同的操作。

[0041] 如本文之前所定义的，传输到特定 GPC208 的一系列指令构成线程，以及跨 SM310 内的并行处理引擎（未示出）的某一数量的并发执行线程的集合在本文中称为“warp”或“线程组”。如本文所使用的，“线程组”是指对不同输入数据并发执行相同程序的一组线程，所述组的一个线程被指派到 SM310 内的不同处理引擎。线程组可以包括比 SM310 内的处理引擎数量少的线程，在这种情况下一些处理引擎将在该线程组正在被处理的周期期间处于闲置状态。线程组还可以包括比 SM310 内的处理引擎数量多的线程，在这种情况下处理将在连续的时钟周期内发生。因为每个 SM310 均可以并发支持多达 G 个线程组，结果是在任意给定时间在 GPC208 中可以执行多达 $G * M$ 个线程组。

[0042] 此外，多个相关线程组可以在 SM310 内同时活动（在执行的不同阶段）。该线程组集合在本文中称为“协作线程阵列”（“CTA”）或“线程阵列”。特定 CTA 的大小等于 $m * k$ ，其中 k 是线程组中并发执行线程的数量并且通常是 SM310 内的并行处理引擎数量的整数倍，以及 m 是 SM310 内同时活动的线程组的数量。CTA 的大小一般由编程者以及可用于 CTA 的硬件资源诸如存储器或寄存器的容量来确定。

[0043] 每个 SM310 均包含一级 (L1) 高速缓存（图 3C 所示）或使用用于实施加载和存储操作的 SM310 外部的相应 L1 高速缓存中的空间。每个 SM310 都还有权访问在所有 GPC208 之间共享并且可用于在线程之间转移数据的二级 (L2) 高速缓存。最后，SM310 还有权访问片外“全局”存储器，所述“全局”存储器可以包括例如并行处理存储器 204 和 / 或系统存储

器 104。应该理解,PPU202 外部的任意存储器均可用作全局存储器。此外,一点五级 (L1.5) 高速缓存 335 可以包括在 GPC208 内,其配置为接收并保持由 SM310 所请求的经由存储器接口 214 从存储器获取的数据,包括指令、标准 (uniform) 数据和常数数据,并将所请求的数据提供给 SM310。在 GPC208 中具有多个 SM310 的实施例有利地共享了高速缓存在 L1.5 高速缓存 335 中的共有指令和数据。

[0044] 每个 GPC208 均可以包括配置为将虚拟地址映射到物理地址中的存储器管理单元 (MMU) 328。在其他实施例中,MMU328 可以驻留在存储器接口 214 内。MMU328 包括用于将虚拟地址映射到像素块 (tile) 的物理地址的页表条目 (PTE) 集和可选地包括高速缓存线索索引。MMU328 可以包括地址转换后备缓冲区 (TLB) 或可以驻留在多处理器 SM310 或 L1 高速缓存或 GPC208 内的高速缓存。物理地址经处理以分布表面数据访问位置来允许高效请求在分区单元 215 之间交错。高速缓存线索索引可用于确定用于高速缓存线的请求是否命中或未命中。

[0045] 在图形和计算应用中,GPC208 可配置为使得每个 SM310 均耦合到用于实施纹理映射操作例如确定纹理样本位置、读出纹理数据以及过滤该纹理数据的纹理单元 315。从内部纹理 L1 高速缓存 (未示出) 或者在一些实施例中从 SM310 内的 L1 高速缓存读出纹理数据并根据需要在所有 GPC208 之间共享的 L2 高速缓存、并行处理存储器 204 或系统存储器 104 中获取纹理数据。为了经由交叉开关单元 210 将所处理的任務提供给另一个 GPC208 用于进一步处理或为了将所处理的任務存储在 L2 高速缓存、并行处理存储器 204 或系统存储器 104 中,每个 SM310 均将所处理的任務输出到工作分布交叉开关 330。preROP (预光栅操作) 325 配置为从 SM310 接收数据、将数据引导到分区单元 215 内的 ROP 单元以及针对颜色混合实施优化、组织像素颜色数据和实施地址转译。

[0046] 应该理解本文所述的内核架构是示例性的并且各种变化和修改都是可能的。任意数量的处理单元例如 SM310 或纹理单元 315、preROP325 均可以包括在 GPC208 内。进一步地,如图 2 所示,PPU202 可以包括任意数量的 GPC208,所述 GPC208 有利地在功能上彼此相似以使得执行行为不取决于哪个 GPC208 接收特定处理任务。进一步地,每个 GPC208 有利地均使用单独的和各异的处理单元、L1 高速缓存来独立于其他 GPC208 操作以为一个或多个应用程序执行任务。

[0047] 本领域普通技术人员应该理解图 1、2、3A 和 3B 所描述的架构决不限本发明的范围并且在不脱离本发明范围的情况下本文所教导的技术可以在任意经适当配置的处理单元上实现,所述处理单元包括但不限于一个或多个 CPU、一个或多个多核 CPU、一个或多个 PPU202、一个或多个 GPC208、一个或多个图形或专用处理单元等等。

[0048] 在本发明的实施例中,使用 PPU202 或计算系统的其他处理器以使用线程阵列执行通用计算是可取的。为线程阵列中的每个线程均指派在线程的执行期间对于线程可访问的唯一的线程标识符 (“线程 ID”)。可被定义为一维或多维数值的线程 ID 控制线程处理行为的各方面。例如,线程 ID 可用于确定线程将要处理输入数据集的哪部分和 / 或确定线程将要产生或写输出数据集的哪部分。

[0049] 每线程指令序列可包括定义代表性线程和线程阵列的一个或多个其他线程之间的协作行为的至少一个指令。例如,每线程指令序列可能包括在序列中的特定点处挂起用于代表性线程的操作执行直到诸如其他线程的一个或多个到达该特定点的时间为止的指

令、用于代表性线程将数据存储在其他线程的一个或多个有权访问的共享存储器中的指令、用于代表性线程自动读出和更新存储在其他线程的一个或多个基于它们的线程 ID 有权访问的共享存储器中的数据中的指令等等。CTA 程序还可以包括计算数据将从其读出的共享存储器中的地址的指令,该地址是线程 ID 的函数。通过定义合适的函数并提供同步技术,可以以可预测的方式由 CTA 的一个线程将数据写入共享存储器中的给定位置并由同一个 CTA 的不同线程从该位置读出数据。因此,数据在线程之间共享的任意期望形式可以得到支持,以及 CTA 中的任意线程可以与同一个 CTA 中的任意其他线程分享数据。如果存在数据在 CTA 的线程之间的共享,则其范围由 CTA 程序确定;因此,应该理解在使用 CTA 的特定应用中,CTA 的线程可能会或可能不会真正互相分享数据,这取决于 CTA 程序,术语“CTA”和“线程阵列”在本文作为同义词使用。

[0050] 图 3C 为根据本发明一个实施例的图 3B 的 SM310 的框图。SM310 包括配置为经由 L1.5 高速缓存 335 从存储器接收指令和常数的指令 L1 高速缓存 370。warp 调度器和指令单元 312 从指令 L1 缓冲 370 接收指令和常数并根据该指令和常数控制本地寄存器堆 304 和 SM310 功能单元。SM310 功能单元包括 N 个 exec(执行或处理)单元 302 和 P 个加载-存储单元 (LSU) 303。

[0051] SM310 提供具有不同级别的可访问性的片上(内部)数据存储。特殊寄存器(未示出)对于 LSU303 可读但不可写并且用于存储定义每个线程的“位置”的参数。在一个实施例中,特殊寄存器包括每线程(或 SM310 内的每 exec 单元 302) 一个的存储线程 ID 的寄存器;每个线程 ID 寄存器仅由各自的 exec 单元 302 可访问。特殊寄存器还可以包括附加寄存器,其对于执行由 TMD322 所代表的同一个处理任务的所有线程(或由所有 LSU303) 可读,其存储 CTA 标识符、CTA 维数、CTA 所属网格(grid) 的维数(或队列位置,如果 TMD322 编码队列任务而不是网格任务的话)、以及 CTA 被指派到的 TMD322 的标识符。

[0052] 如果 TMD322 是网格 TMD,则 TMD322 的执行会启动和执行固定数量的 CTA 以处理存储在队列 525 中的固定量的数据。将 CTA 的数量指定为网格宽度、高度和深度的乘积。可以将固定量的数据存储于 TMD322 中或 TMD322 可以存储指向将由 CTA 所处理的数据的指针。TMD322 还存储由 CTA 所执行的程序的开始地址。

[0053] 如果 TMD322 是队列 TMD,那么使用 TMD322 的队列特点,这意味着将要被处理的数据量不一定是固定的。队列条目存储用于由指派到 TMD322 的 CTA 所处理的数据。队列条目还可以代表在线程执行期间由另一个 TMD322 所生成的子任务,从而提供嵌套并行性。通常线程或包括线程的 CTA 的执行被挂起直到子任务的执行完成。在一些实施例中,被挂起的线程或 CTA 保存它们的程序状态、将数据写到表示线程或 CTA 的继续部分的队列 TMD 并且随后退出,以便允许其它线程或 CTA 运行。可以将队列存储在 TMD322 中或与 TMD322 分开存储,在该情况下 TMD322 存储指向该队列的指针。有利地,当代表子任务的 TMD322 正在执行时可以将由子任务所生成的数据写到队列。队列可以实现为循环队列以使得数据的总量不限于队列的大小。

[0054] 属于网格的 CTA 具有指示网格内各自 CTA 的位置的隐含网格宽度、高度和深度参数。在初始化期间响应于经由前端 212 从设备驱动程序 103 所接收的命令来写特殊寄存器并且在处理任务的执行期间特殊寄存器不改变。前端 212 调度每个处理任务用于执行。每个 CTA 均与特定 TMD322 相关联用于一个或多个任务的并发执行。此外,单个 GPC208 可以

并发执行多个任务。

[0055] 绑定到任务的参数存储器（未示出）存储可由该任务的任意线程（或任意 LSU303）读取但不可由其写入的运行时间参数（常数）。在一个实施例中，设备驱动程序 103 在引导 SM310 开始执行使用参数的任务之前将这些参数提供给参数存储器。任意 CTA 内的任意线程（或 SM310 内的任意 exec 单元 302）均可以通过存储器接口 214 访问全局存储器。可以将全局存储器的各部分存储在 L1 高速缓存 320 中。

[0056] 每个线程均将本地寄存器堆 304 用作暂存空间；每个寄存器被分配以专用于一个线程，并且在本地寄存器堆 304 的任意一个中的数据仅对于寄存器被分配到的线程可访问。本地寄存器堆 304 可以实现为物理上或逻辑上分为 P 个通道的寄存器堆，每个通道具有一定数量的条目（其中每个条目可以存储例如 32 位字）。将一个通道指派到 N 个 exec 单元中和 P 个下载 - 存储单元 LSU303 的每一个，并且利用用于执行同一个程序的不同线程的数据来填充不同通道中的相应条目以帮助 SIMD 执行。可以将通道的不同部分分配到 G 个并发线程组中的不同线程组，以使得本地寄存器堆 304 中的给定条目仅对于特定线程可访问。在一个实施例中，保留本地寄存器堆 304 内的某些条目用于存储线程标识符，这实现特殊寄存器之一。此外，标准 L1 高速缓存 375 存储用于 N 个 exec 单元 302 和 P 个下载 - 存储单元 LSU303 的每个通道的标准或常数值。

[0057] 共享存储器 306 对于单个 CTA 内的线程可访问；换言之，共享存储器 306 中的任意位置对于同一个 CTA 内的任意线程（或对于 SM310 内的任意处理引擎）可访问。共享存储器 306 可以实现为具有允许任意处理引擎对共享存储器中的任意位置读取或写入的互连的共享寄存器堆或共享片上高速缓存存储器。在其他实施例中，共享状态空间可能映射到片外存储器的每 CTA 区域上并被高速缓存在 L1 高速缓存 320 中。参数存储器可以实现为在实现共享存储器 306 的同一个共享寄存器堆或共享高速缓存存储器内的指定部分，或者实现为 LSU303 对其具有只读访问权限的单独的共享寄存器堆或片上高速缓存存储器。在一个实施例中，实现参数存储器的区域还用于存储 CTA ID 和任务 ID，以及 CTA 和网格维数或队列位置，这实现特殊寄存器的各部分。SM310 中的每个 LSU303 均耦合到统一地址映射单元 352，统一地址映射单元 352 将为在统一存储器空间中所指定的加载和存储指令所提供的地址转换为每个相异存储器空间中的地址。因此，指令可以用于通过指定统一存储器空间中的地址来访问本地、共享或全局存储器空间中的任意一个。

[0058] 每个 SM310 中的 L1 高速缓存 320 可以用于高速缓存私有的每线程本地数据还有每应用全局数据。在一些实施例中，可以将每 CTA 共享数据高速缓存在 L1 高速缓存 320 中。LSU303 经由存储器和高速缓存互连 380 耦合到共享存储器 306 和 L1 高速缓存 320。

[0059] 计算任务的调度和执行

[0060] 图 4A-4B 示出了根据本发明的一个实施例的，用于将任务指派给图 3A-3C 的 SM310 的方法。尽管结合图 1-3C 的系统描述了方法步骤，但本领域普通技术人员将理解配置为以任何顺序实施该方法步骤的任何系统均在本发明的范围内。

[0061] 如图所示，方法 400 在步骤 402 开始，其中 WDU340 确定是否一个或多个 TMD322 包括在图 3A 的任务表 345 中。在步骤 404，WDU340 将包括在多个 SM 中的第一 SM（例如，包括在 PPU202 内的 SM310）设置为当前 SM。在步骤 406，WDU340 将包括在任务表 345 中的第一 TMD322 设置为当前 TMD。

[0062] 在步骤 408, WDU340 确定是否当前 TMD 驻留在其中的任务表 345 槽已经接收到解除分配请求。在步骤 408, 如果 WDU340 确定当前 TMD 驻留在其中的任务表槽已经接收到解除分配请求, 那么当前 TMD 不应该是任何 SM310。相应地, 方法 400 前进到步骤 428, 其中 WDU340 将包括在任务表 345 中的下一个 TMD322 设置为当前 TMD。进而, 方法 400 前进回到上述步骤 408。

[0063] 相反, 在步骤 408, 如果 WDU340 确定当前 TMD 驻留在其中的任务表槽并未接收到解除分配请求, 那么方法 400 前进到步骤 410。

[0064] 在步骤 410, WDU340 确定当前 TMD 是否包括 CTA 中尚未发出的工作。在步骤 410, 如果 WDU340 确定当前 TMD 不包括 CTA 中尚未发出的工作, 那么方法 400 前进到上述步骤 428。否则, 方法 400 前进到步骤 412。

[0065] 在一个实施例中, 每个 TMD322 均包括当该 TMD322 正被调度用于执行时, 由例如任务管理单元 300 和工作分布单元 340 所设置的准静态状态。每个 TMD322 均还包括当执行该 TMD322 时被更新的动态状态, 例如当发生用于 TMD322 的 CTA 启动和完成时。

[0066] 存在包括在 TMD322 中的很多状态部件, 其与在 PPU202 内处置 TMD322 的方式相关。在一个实施例中, TMD322 包括用于跟踪包括在尚未完成的 TMD322 中的工作项的数目的状态。在一些情况下, 连同指定在最终启动 CTA 用于执行之前所允许等待阈值时间量以积累最小所需数目的工作项的状态 (以下称为“合并超时 (coalescing timeout)”) 一起, TMD322 还可包括指定需要包括在被发出到 SM310 的每个 CTA 中的工作项的最小数目的状态 (以下称为“合并规则 (coalescing rules)”)。当 TMD 指定 M 个工作项 / 每 CTA 时, 则由每个 CTA 读取 N 个工作项。例如, 可能有多个将工作项写到队列 TMD 的 TMD, 其中队列 TMD 的每个 CTA 均处理 N 个工作项。这就将 N 个分离的工作项“合并”到一个 CTA 中。然而, 多个 TMD 有可能并不生成被 N 所整除的数目的工作项, 其导致留下未完成的部分工作项集。为了规避以上所述, 在一个实施例中, TMD 包括允许利用 M 个工作项来启动 CTA 的超时值, 其中 $M < N$ 。把 M 的值当做 CTA 的输入, 并且写入与 CTA 相关联的指令以处理 M 个工作项或者 N 个工作项, 其取决于 M 的值。

[0067] TMD322 还包括指定 TMD322 执行优先级级别的状态, 所述执行优先级级别例如在 1-10 的数字范围内的优先级级别, 其中最低的数字表示最高的执行优先级级别。TMD322 还包括指示 TMD322 驻留在其中的任务表 345 中的槽在由任务管理单元 300 调度后是否是有效槽——也就是其中尚未请求对 TMD322 的解除分配的状态。如以下结合图 4A-4B 所详细描述, TMD322 还可包括用于 SM 亲和规则 (affinity rule) 的状态, 其指定可将 TMD322 指派到 PPU202 中的哪个 SM310 上。每个 TMD322 均还可包括指示是否 TMD322 仅当任务 / 工作单元 207 正操作于“节流模式”时才可执行的状态, 所述“节流模式”涉及有权访问由包括在 PPU202 中的 SM310 可访问的所有共享存储器的单个 CTA。在一个实施例中, 当 WDU340 在节流和非节流模式之间切换时, 切换模式的状况 (status) 存储在状态 304 中并且由 WDU340 进行更新。每个 TMD322 均还可包括指定 TMD322 为序列任务并且因此在任意给定时间可使至多一个 CTA “正在处理 (in flight)” (即, 正由 SM310 所执行) 的状态。

[0068] 在步骤 412, WDU340 确定是否任务表 345 中的任何 TMD322 指示节流模式属性。在步骤 412, 如果 WDU340 确定任何 TMD 指示节流模式属性, 则方法 400 前进到步骤 414 以确定在任务 / 工作单元 207 内是否激活了节流模式。在步骤 414, 如果 WDU340 确定在任务 / 工

作单元 207 内未激活节流模式,则方法 400 前进到步骤 450。如图所示,在步骤 450,WDU340 等待直到所有未完成的 TMD322 均被执行,即当激活时未指示节流模式的 TMD322。随后方法 400 前进到步骤 452,其中 WDU340 将节流状态发送到 SM310 的每一个。在一个实施例中,用于每个 SM310 的节流状态均既包括指示 SM310 可访问的共享存储器的一部分的大小的值,还包括共享存储器该部分开始处的基址。因此,当使能较少的 SM310 时,对每个 SM310 来说,指示共享存储器的一部分的大小的值增加。相反,当使能较多的 SM310 时,对每个 SM310 来说,指示共享存储器的一部分的大小的值减少。

[0069] 在步骤 454,WDU340 激活节流模式,于是方法 400 前进回到步骤 402。WDU340 继续在节流模式下操作直到步骤 412 为假,即直到 WDU340 确定不再有包括在任务表 345 中的 TMD322 指示节流模式属性。相应地,WDU340 在步骤 413 停用节流模式,于是方法 400 在步骤 416 处重新开始。

[0070] 在步骤 416,WDU340 确定当前 TMD 是否为序列任务。在步骤 416,如果 WDU340 确定当前 TMD 是序列任务,那么方法 400 前进到步骤 418,其中 WDU340 确定当前 TMD 是否具有正在处理的 CTA,即当前正由 SM310 所执行的 CTA。在步骤 418,如果 WDU340 确定当前 TMD 具有正在处理的 CTA,那么方法 400 前进到上述步骤 428。否则,方法 400 前进到如下所述的步骤 420。

[0071] 现在返回参考步骤 416,如果 WDU340 确定当前 TMD 不是序列任务,那么方法 400 前进到步骤 419。在步骤 419,WDU340 确定当前 TMD322 的启动配额,如果有的话,是否被满足。在一个实施例中,每个 TMD322 均既包括启动配额使能位,也包括启动配额值。当启动配额使能位设置为“真”时,WDU340 确定是否已经启动相当于启动配额值的若干 CTA。相应地,在步骤 419,如果 WDU340 确定 TMD322 的启动配额,如果有的话,已被满足,那么方法 400 前进到步骤 460。

[0072] 在步骤 460,WDU340 解析任务表 345 并且选择具有与当前 TMD 相同的优先级的 TMD322,于是 WDU340 将所选择的 TMD322 设置为当前 TMD322。随后方法 400 前进到步骤 402。

[0073] 现在返回参考步骤 419,如果 WDU340 确定 TMD322 的启动配额未被满足,或对于 TMD322 未指定启动配额,那么方法前进到步骤 420。

[0074] 在步骤 420,WDU340 确定当前 TMD 的亲规则和节流模式参数是否禁止将当前 TMD 指派给当前 SM。在步骤 420,如果 WDU340 确定当前 TMD 的亲规则和节流模式参数禁止将当前 TMD 指派给当前 SM,那么方法 400 前进到步骤 428,如上所述。否则,在步骤 424,WDU340 将当前 TMD 添加到与当前 SM 相对应的任务列表。

[0075] 在步骤 426,WDU340 确定在任务表中是否包括附加的 TMD322。在步骤 426,如果 WDU340 确定在任务表 345 中包括附加的 TMD322,那么方法 400 前进到步骤 428,如上所述。这样,包括在任务表 345 中的每个 TMD322 均与当前 SM 相比较以确定哪一个 TMD322 最有资格被指派给当前 SM,如以下在步骤 434 中所述。

[0076] 然而,在步骤 426,如果 WDU340 确定在任务表 345 中不包括附加的 TMD322,那么所有 TMD322 均已经与当前 SM 相比较,并且因此方法 400 前进到步骤 430。在步骤 430,WDU340 基于与包括在任务列表中的每个 TMD322 相关联的执行优先级值来执行初级类别的任务列表。在步骤 432,WDU340 基于与包括在任务列表中的每个 TMD322 相关联的时间戳值来执行

次级类别的任务列表,其中时间戳值表示 TMD322 插入到任务表 345 中的时间。在一个实施例中,时间戳值维护在状态 304 中或者可作为列包括在任务表 345 内。

[0077] 在一些实施例中,代替时间戳,WDU340 维护包括在任务表 345 中的槽的分类列表,其中每次分配或解除分配新任务时分别插入或删除在列表中的条目。因此,槽的分类列表保持为有组织的并且仅在每次分配或删除任务时被重新分类,使得具有最高优先级值的最旧的 TMD322 可被容易地识别并且指派给当前 SM,如以下在步骤 434 所述。

[0078] 在步骤 434, WDU340 将具有最高优先级值和最旧时间戳值的 TMD322 指派给当前 SM。在一个实施例中,当 TMD322 在步骤 434 被指派给当前 SM 时,当前 SM 具有由 WDU340 设置并且存储在 SM 状态 342 中的与其相关联的状态。此后,当在当前 SM 上执行与指派给当前 SM 的 TMD322 相对应的 CTA 时, WDU340 修改状态,如以下结合图 5 所详细描述。在一个实施例中,状态包括若干属性,其包括指示有资格的 TMD 是否被指派给当前 SM 的“TASK_ASSIGN”。状态还可包括“STATE_SYNC”属性,其指示 WDU340 是正在等待发出 TMD322 状态更新到当前 SM,还是正在等待当前 SM 确认状态更新,如以下在步骤 438 所进一步详细描述的。状态还可包括“CTA_LAUNCH”属性,其指示当前 SM 准备好从步骤 434 的 TMD322 接收并且执行 CTA(受限于具有接收和执行 CTA 的能力的当前 SM)。其他状态可用于导出用于当前 SM 的 CTA 可用性值,如以下结合图 5 所述,其表示 WDU340 可立即启动到当前 SM 的附加的 CTA 的数量(即,在 WDU340 从当前 SM 接收到任何更多 CTA 完成信息之前)。

[0079] 在步骤 436, WDU340 确定不是当前 TMD 的 TMD322 是否之前被指派给当前 SM。在步骤 436,如果 WDU340 确定不是当前 TMD 的 TMD322 之前被指派给当前 SM,那么方法 400 前进到步骤 438,其中 WDU340 将与当前 TMD 相关联的状态数据发送给当前 SM。否则,方法 400 前进到步骤 440。

[0080] 在步骤 440, WDU340 确定在多个 SM310 中是否包括附加的 SM310。在步骤 440,如果 WDU340 确定在多个 SM310 中包括附加的 SM310,那么方法 400 前进到步骤 442,其中 WDU340 将包括在多个 SM310 中的下一个 SM310 设置为当前 SM。然而,如果 WDU340 在步骤 440 确定在多个 SM 中不包括附加的 SM,那么方法 400 前进回到步骤 402,并且根据本文的技术重复方法 400。

[0081] 因此,在方法 400 的结束处,如果包括在任务表 345 中的 TMD322 存在的话,则取决于例如所述 TMD322 的状态数据,已对 0 个或多个 SM310 指派了 TMD322。结合持续地将不同的 TMD322 指派给不同的 SM310,工作分布单元 340 还配置为持续地选择 SM,应向所述 SM 发出来自指派给一个 SM 的 TMD322 的 CTA,下面结合图 5 对其加以描述。

[0082] 图 5 示出了根据本发明的一个实施例的,用于选择 SM310 以接收与任务相关的工作的方法 500。尽管结合了图 1-3C 的系统来描述方法步骤,但是本领域技术人员将理解,配置为以任何顺序实施方法步骤的任何系统都落在本发明的范围之内。

[0083] 如图所示,方法 500 在步骤 502 开始,其中 WDU340 从包括在 PPU202 中的每个 SM310 接收以下指示:如果有与 SM310 相关联的 TMD322 的话,SM310 是否有资格从该 TMD322 接收 CTA。在一个实施例中,该指示以“ready(准备就绪)”状况的形式进行传送,其源自于与 SM310 相关联的并且存储在图 3A 的 SM 状态 342 中的状态。在一个示例中,如果 SM310 已经被指派了 TMD322(例如,根据以上结合图 4A-4B 的所述的方法步骤 400)并且与 TMD322 相关联的状态已经发送到 SM310 并且由其确认(例如,根据方法 400 的方法步骤 438),则

SM310 被确定为准备就绪。还可基于 WDU340 是否正操作在以上结合图 4A-4B 所述的节流模式来确定 SM310 为使能或禁用。指派给 SM310 的 TMD322 要求本文所述的节流模式是活动的并且任务 / 工作单元 207 实际上正操作在节流模式。可基于指派给 SM310 的 TMD322 是否满足任何合并规则来进一步确定 SM310 为准备就绪。例如, 指派给 SM310 的 TMD322 可指示在 CTA 发出到 SM310 之前, 最少 8 个未完成的工作项必须包括在例如与 TMD322 相关联的工作项队列中。而且, 为了规避包括在 TMD322 中的未完成工作项的数量大于 0 但不会超过每 CTA 的未完成工作项的阈值最小值数量的情况, 可实现如以上结合图 4A-4B 所述的合并超时。假设 TMD322 和 / 或 SM310 满足结合步骤 502 所述的附加的资格要求, 则当合并超时发生时, SM310 成为有资格从 TMD322 接收 CTA。

[0084] 在步骤 506, WDU340 确定负载平衡模式或循环模式是否是活动的。在一个实施例中, 由存储在任务 / 工作单元 207 的状态 304 中的单个位值来管理活动模式。

[0085] 在步骤 508, WDU340 从有资格的 SM310 中的每一个接收 CTA 可用性值。在一个实施例中, CTA 可用性值是指 SM310 所具有的接受和执行附加 CTA 的总体能力的数值。该数量由每个 SM310 计算, 并且基于例如正由 SM310 所执行的 CTA 的当前数量、最近指派给 SM 的任务的每 CTA 资源需求以及 SM310 可用的空闲资源的总量等等。

[0086] 在步骤 510, WDU340 基于 CTA 可用性值来执行一类有资格的 SM310。在步骤 512, WDU340 确定是否两个或两个以上 SM310 共享同一个最高 CTA 可用性值。在步骤 512, 如果 WDU340 确定两个或两个以上 SM310 共享同一个最高 CTA 可用性值, 那么方法 500 前进到步骤 514, 其中 WDU340 基于固定的 SM 优先级列表来选择这两个或两个以上 SM310 中的一个。在一个实施例中, 固定的优先级列表包括在任务 / 工作单元 207 的状态 304 中。

[0087] 现在返回参考步骤 512, 如果 WDU340 确定两个或两个以上 SM310 不共享同一个最高 CTA 可用性值, 那么方法 500 前进到步骤 516, 其中 WDU 利用最高 CTA 可用性值来选择 SM310。

[0088] 在步骤 518, WDU340 向所选择的 SM310 发出指派给所选择的 SM310 的 TMD322 的 CTA。随后方法 500 前进回到步骤 502, 其中重复方法步骤 500 使得只要存在至少一个指派给一个或多个 SM310 的 TMD322, 则 WDU340 就持续地将 CTA 发出到该一个或多个 SM310 并且包括尚未由任何 SM310 所执行的工作。

[0089] 现在返回参考步骤 506, 如果 WDU340 确定任务 / 工作单元 207 的活动模式指示了循环模式, 那么方法 500 前进到步骤 520。在步骤 520, WDU340 从在步骤 502 所确定的有资格的 SM310 中选择数字上的下一个 SM310。在一个实施例中, WDU340 维护 CTA 被发出给其的最后一个 SM 的状态 304 中的识别值。这样, WDU340 可通过持续地向具有数字上的下一个 SM 识别值的 SM 发出 CTA 并且相应地更新状态 304 中的识别值来实现循环技术。

[0090] 可将本发明的一个实施例实现为用于与计算机系统一起使用的程序产品。程序产品的程序定义了实施例的功能 (包括本文所述的方法) 并且可被包含在各种各样的计算机可读存储介质上。示例性的计算机可读存储介质包括但不限于: (i) 永久地将信息存储在其上的非可写存储介质 (例如, 诸如由 CD-ROM 驱动器可读的光盘只读存储器 (CD-ROM)、闪存、只读存储器 (ROM) 芯片或任何类型的固态非易失性半导体存储器的计算机内的只读存储器设备); 以及 (ii) 将可更改信息存储在其上的可写存储介质 (例如, 软盘驱动器内的软盘、硬盘驱动器或者任何类型的固态随机存取半导体存储器)。

[0091] 上文已经参考特定实施例描述了本发明。然而，本领域普通技术人员将理解，在不脱离如在所附的权利要求中所阐述的本发明的更宽的精神和范围的情况下可做出各种修改和改变。因此，前述描述和附图视为示例性的而不是限制性的意义。

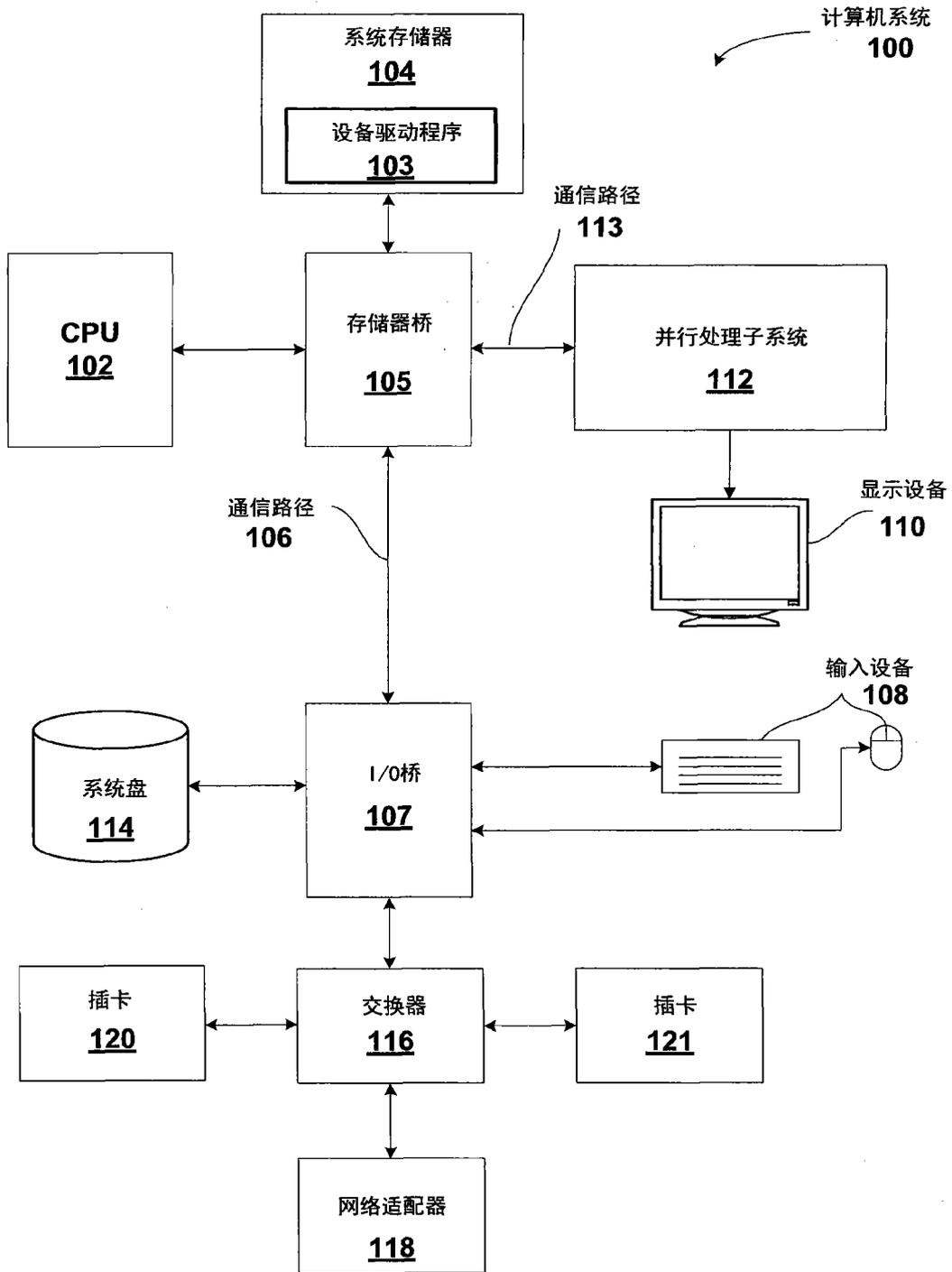


图 1

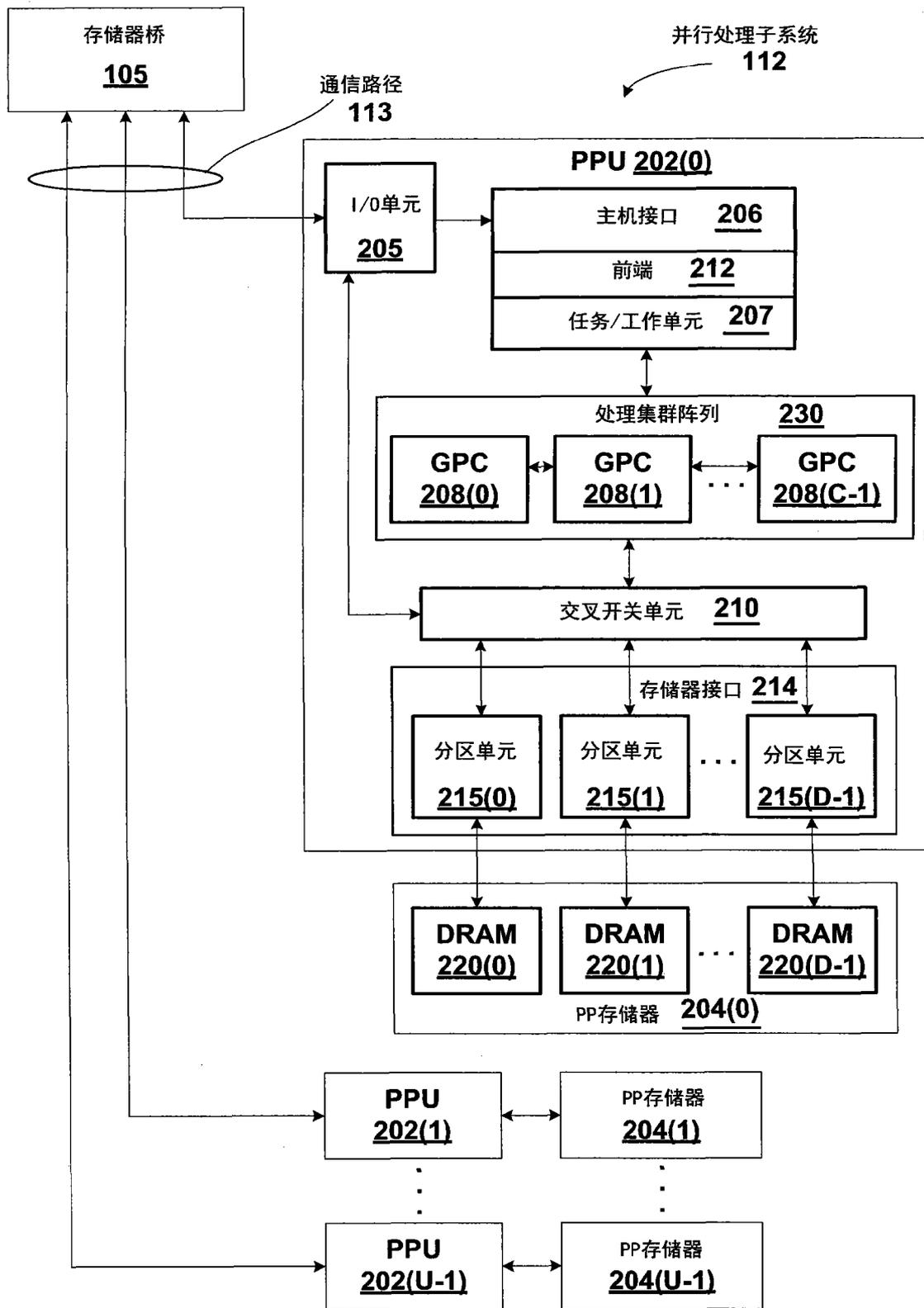


图 2

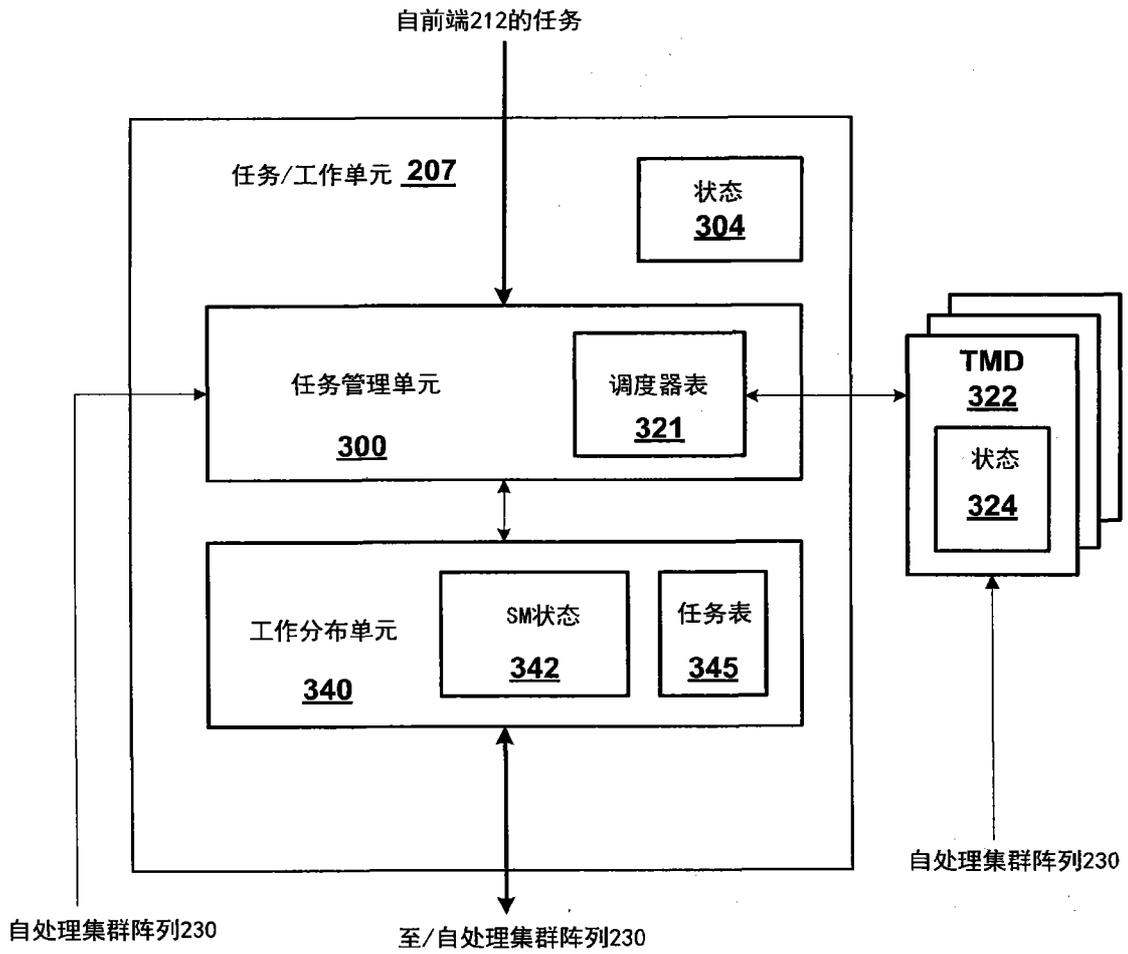


图 3A

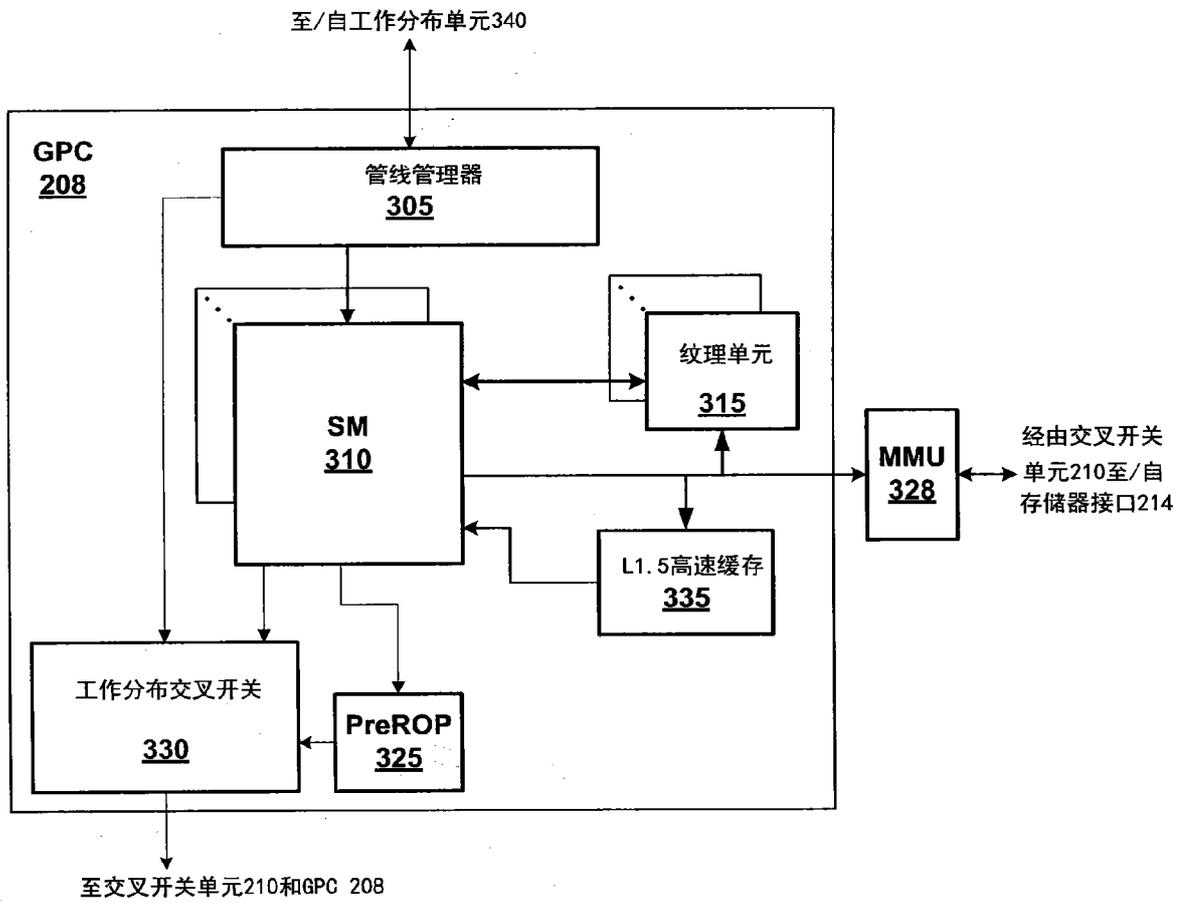


图 3B

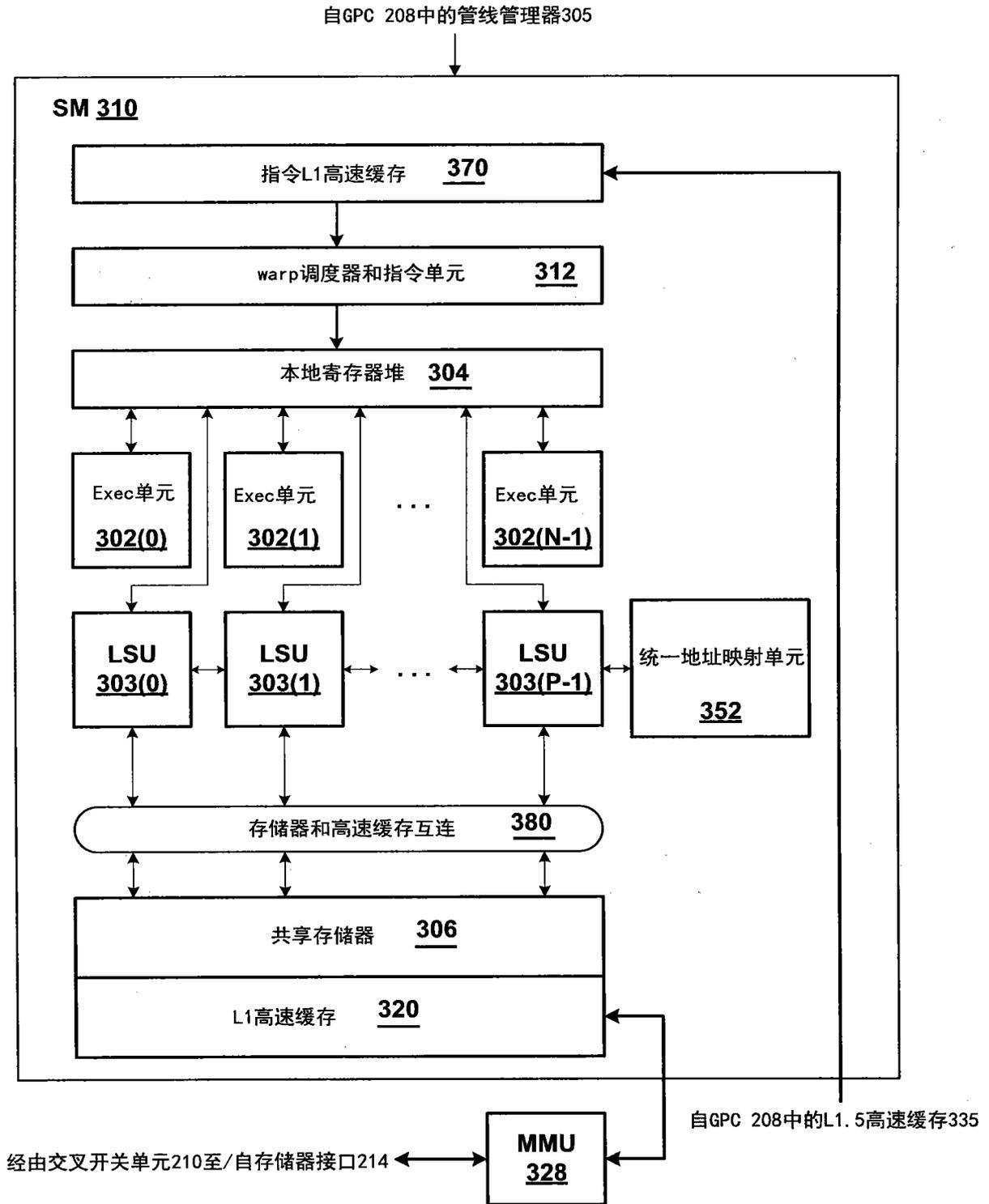


图 3C

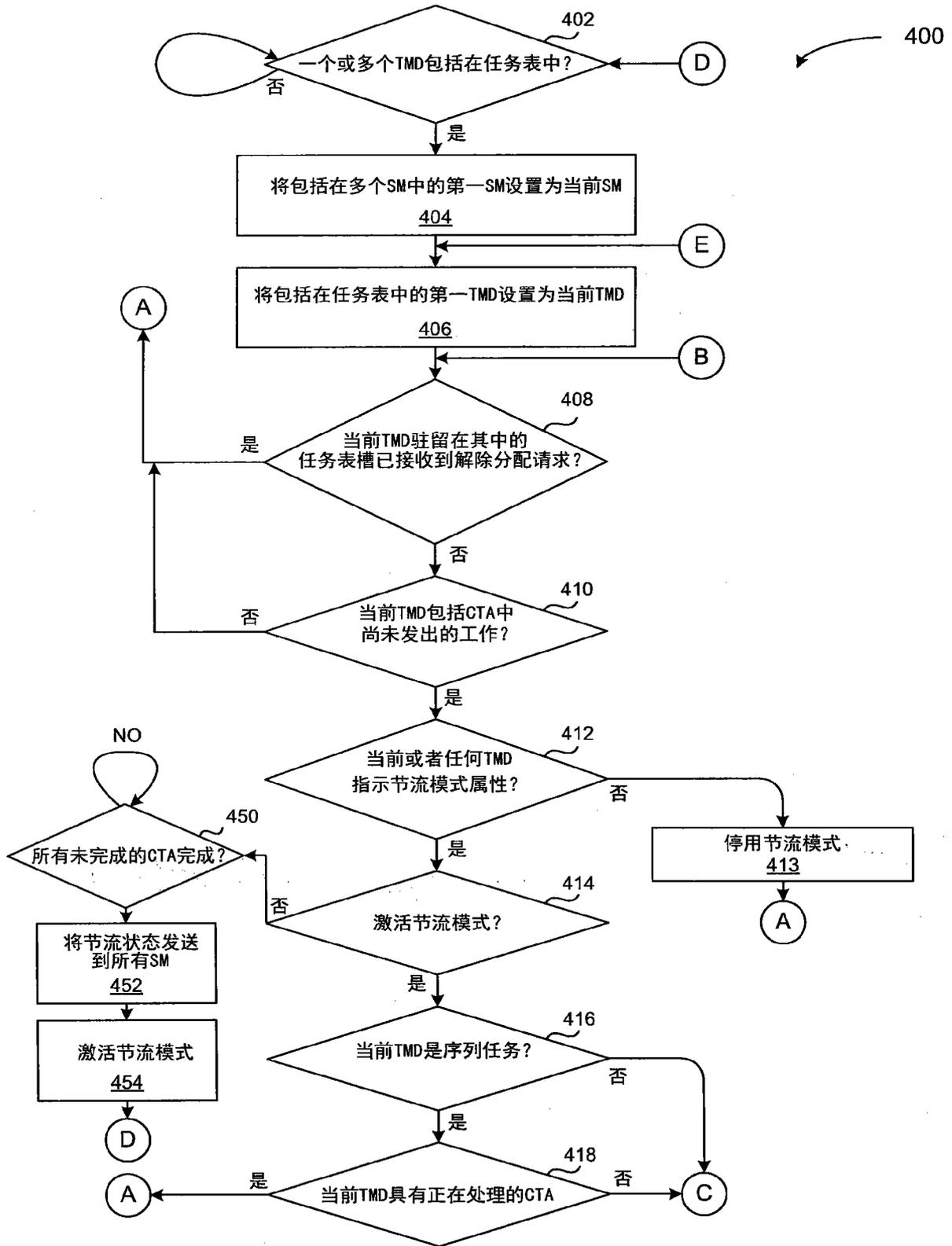


图 4A

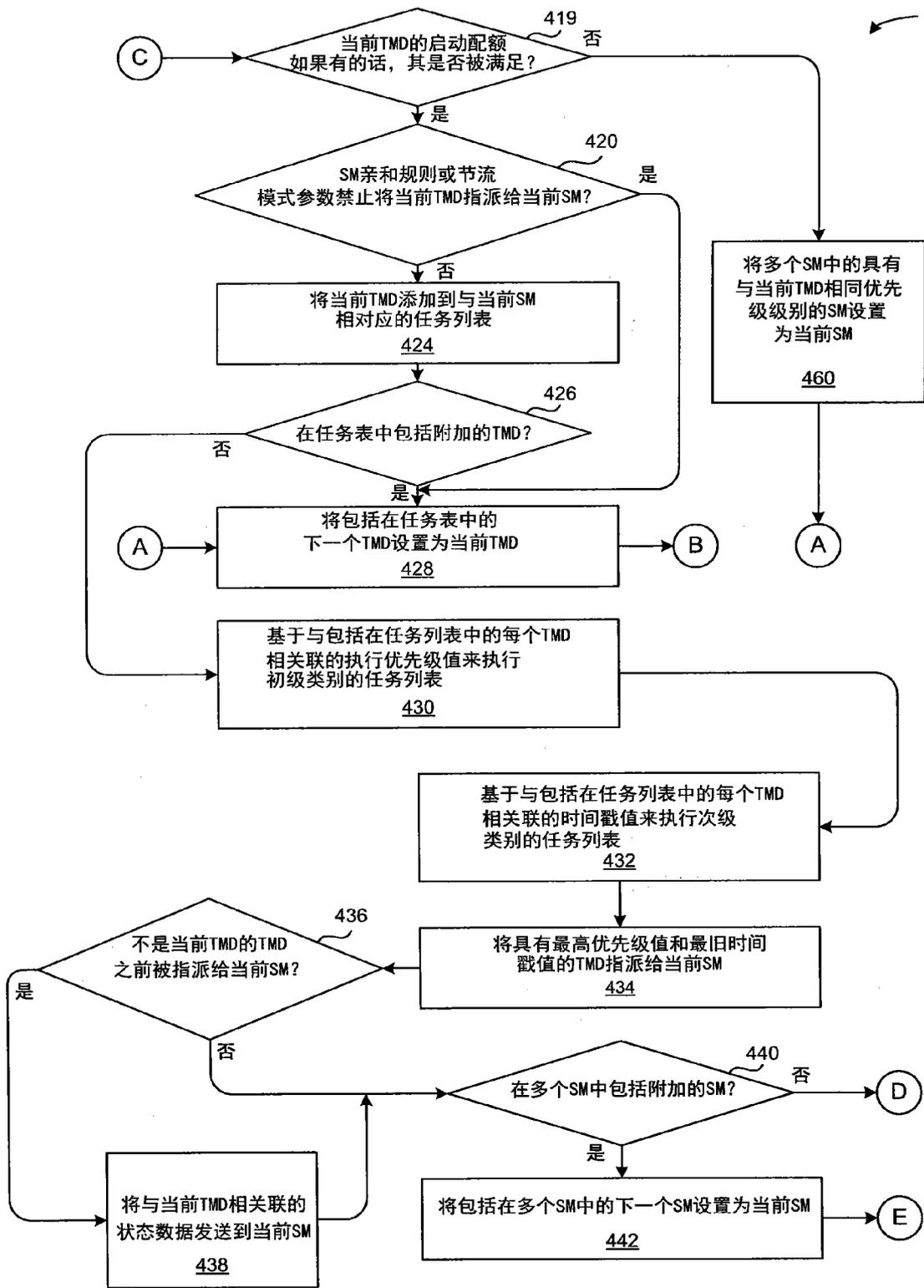


图 4B

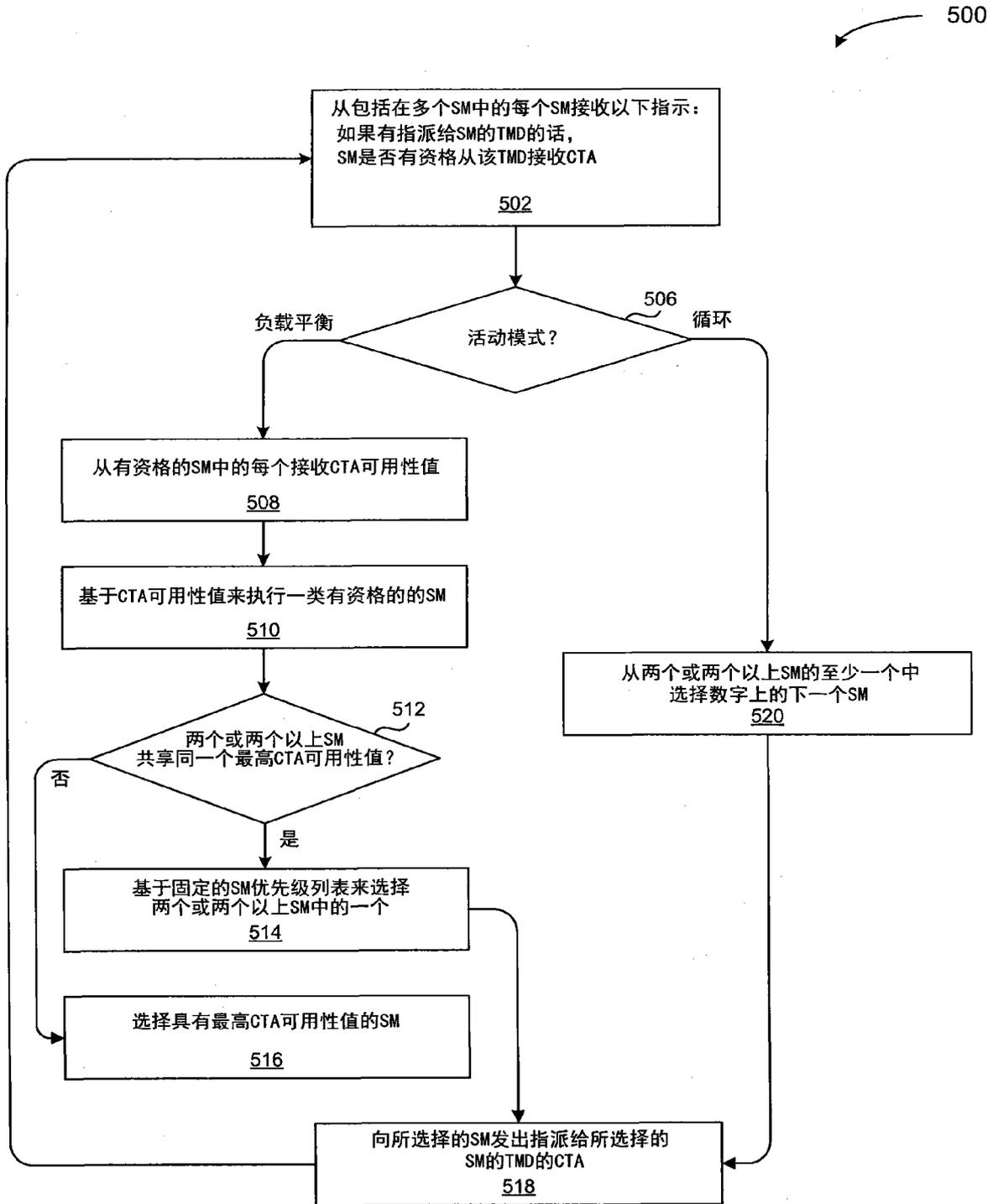


图 5