(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 111985629 A
(43)申请公布日 2020.11.24

(21)申请号 202010323930.6

(22)申请日 2020.04.22

(30)优先权数据
16/418,799 2019.05.21 US

(71)申请人 辉达公司
地址 美国加利福尼亚州

(72)发明人 E·埃布拉希米 A·佐菲卡
S·帕尔

(74)专利代理机构 北京市磐华律师事务所 11336
代理人 高伟

(51)Int.Cl.
G06N 3/063(2006.01)
G06N 3/04(2006.01)
G06N 3/08(2006.01)

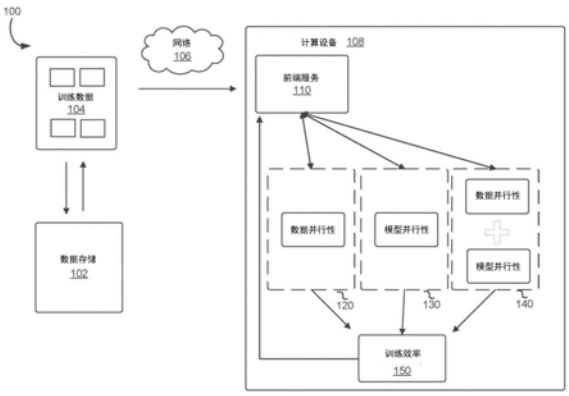
权利要求书3页 说明书21页 附图13页

(54)发明名称

用于训练神经网络的并行化策略

(57)摘要

本发明公开了一种用于训练神经网络的并行化策略。给定用于训练深度学习(DL)模型的多个设备(例如,图形处理单元),训练神经网络以系统地找到交叉点的系统和方法,其指示在特定系统上优化DL模型的训练时应实现哪个并行化策略,以实现最大的效率增益。



CN 111985629 A

1.一种处理器,包括:

一个或更多个算术逻辑单元(ALU),用于使用一个或更多个神经网络推断信息,其中所述一个或更多个神经网络已经使用训练数据并行性和模型并行性的组合进行了训练,以达到训练效率的级别。

2.根据权利要求1所述的处理器,其中所述一个或更多个神经网络已经通过增加数据并行性直到达到训练效率的中间级别以及增加模型并行性直到达到训练效率的所述级别进行了训练。

3.根据权利要求2所述的处理器,其中所述训练效率的所述中间级别至少部分地基于与使用训练数据并行性相关联的训练时间来测量,并且所述训练效率的所述级别至少部分地基于与使用训练数据并行性和模型并行性的组合相关联的训练时间来测量。

4.根据权利要求3所述的处理器,其中所述一个或更多个神经网络已通过以下步骤进行了训练:

比较与使用训练数据并行性相关联的训练时间和与使用训练数据并行性和模型并行性的所述组合相关联的训练时间;以及

基于所述比较,使用训练数据并行性和模型并行性的所述组合。

5.根据权利要求1所述的处理器,其中所述一个或更多个神经网络已通过以下步骤进行了训练:

获得与所述一个或更多个神经网络相关联的配置参数;以及

基于所获得的信息,使用训练数据并行性和模型并行性的所述组合。

6.根据权利要求4所述的处理器,其中所述一个或更多个神经网络已经通过响应于较慢的训练时间减少训练模型并行性进行了训练。

7.一种系统,包括:

一个或更多个计算机,其具有一个或更多个处理器,用于使用第一数量的训练数据线程训练一个或更多个神经网络,以达到训练效率的第一级别,并且所述一个或更多个神经网络的第二数量的部分使用所述第一数量的训练数据线程并行训练,以达到训练效率的第二级别。

8.根据权利要求7所述的系统,其中所述第一数量的训练数据线程被拆分为要在所述一个或更多个处理器之间分布以训练所述一个或更多个神经网络的子集。

9.根据权利要求7所述的系统,其中并行训练的所述一个或更多个神经网络的所述第二数量的部分被拆分为要在所述一个或更多个处理器之间分布以训练所述一个或更多个神经网络的组件。

10.根据权利要求7所述的系统,其中具有所述一个或更多个处理器的所述一个或更多个计算机通过增加所述第一数量的训练数据线程直到达到所述训练效率的第一级别,来进一步训练所述一个或更多个神经网络。

11.根据权利要求7所述的系统,其中具有所述一个或更多个处理器的所述一个或更多个计算机通过以下步骤来进一步训练所述一个或更多个神经网络:

比较与使用所述第一数量的训练数据线程相关联的训练时间和与使用所述一个或更多个神经网络的所述第二数量的部分相关联的训练时间,所述一个或更多个神经网络的所述第二数量的部分使用所述第一数量的训练数据线程并行训练;以及

基于所述比较，使用所述一个或更多个神经网络的所述第二数量的部分，所述一个或更多个神经网络的所述第二数量的部分使用所述第一数量的训练数据线程并行训练。

12.根据权利要求7所述的系统，其中响应于达到所述训练效率的第一级别，使用所述第一数量的训练数据线程并行训练的所述一个或更多个神经网络的所述第二数量的部分增加。

13.根据权利要求7所述的系统，其中所述训练效率的第一级别和第二级别至少部分地基于针对所述系统的功耗的信息。

14.一种机器可读介质，其上存有指令集，所述指令集如果由一个或更多个处理器执行，则使得所述一个或更多个处理器至少：

使用第一数量的并行训练数据线程训练神经网络，导致训练效率的第一级别；以及

使用所述第一数量的并行训练数据线程并行训练所述神经网络的第二数量的部分，导致训练效率的第二级别。

15.根据权利要求14所述的机器可读介质，其中所述训练效率的第一级别至少部分地基于与使用所述第一数量的并行训练数据线程训练所述神经网络相关联的训练时间。

16.根据权利要求14所述的机器可读介质，其中所述训练效率的第二级别指示，与使用所述神经网络的所述第二数量的部分结合使用所述第一数量的并行训练数据线程并行训练所述神经网络相关联的训练时间。

17.根据权利要求14所述的机器可读介质，其中所述指令集进一步使得所述一个或更多个处理器，至少通过调节所述第一数量的并行训练数据线程来训练所述神经网络，直到达到所述训练效率的第一级别。

18.根据权利要求14所述的机器可读介质，其中所述指令集进一步使得所述一个或更多个处理器至少通过以下步骤训练所述神经网络：

比较与使用所述第一数量的并行训练数据线程相关联的训练时间和与使用所述第一数量的并行训练数据线程且并行使用所述神经网络的所述第二数量的部分相关联的训练时间；以及

基于所述比较，使用所述神经网络的所述第二数量的部分且并行使用所述第一数量的并行训练数据线程。

19.根据权利要求14所述的机器可读介质，其中所述指令集进一步使得所述一个或更多个处理器，在使用所述神经网络的所述第二数量的部分且并行使用所述第一数量的并行训练数据线程之前，至少使用所述第一数量的并行训练数据线程训练所述神经网络。

20.一种方法，包括：

确定第一数量的并行训练数据线程，其用于训练神经网络至达到或高于训练效率的第一级别；以及

确定所述神经网络的第二数量的部分，用于使用所述第一数量的并行训练数据线程并行训练至达到或高于训练效率的第二级别。

21.根据权利要求20所述的方法，其中所述第一数量的并行训练数据线程被拆分为要在计算机系统的一个或更多个处理器之间分布以训练所述神经网络的子集。

22.根据权利要求20所述的方法，其中要并行训练的所述神经网络的所述第二数量的部分被拆分为要在计算机系统的一个或更多个处理器之间分布以训练所述神经网络的组

件。

23.根据权利要求20所述的方法，其中训练所述神经网络的所述第一数量的并行训练数据线程增加，直到所述神经网络被训练至达到或高于所述训练效率的第一级别，其中所述训练效率的第一级别至少部分地基于与使用所述第一数量的并行训练数据线程训练所述神经网络相关联的训练加速。

24.根据权利要求23所述的方法，其中响应于所述神经网络被训练至达到或高于所述训练效率的第一级别，确定使用所述第一数量的并行训练数据线程并行训练的所述神经网络的所述第二数量的部分。

25.根据权利要求20所述的方法，其中使用所述第一数量的并行训练数据线程并行训练的所述神经网络的第二数量的部分，至少部分地基于与使用所述第二数量的部分训练所述神经网络相关联的训练加速。

用于训练神经网络的并行化策略

背景技术

[0001] 深度学习(DL)和其他机器学习模型的使用正在普遍增长,并且用于训练它们的数据集的大小继续增加,这反过来导致了越来越长的训练时间。随着机器学习变得越来越复杂,模型变得越来越大,并且用于训练这些模型的数据集包含更多信息。因此,鉴于需要采用的设备数量也增加了,具有更大数据集的训练机器学习模型开始影响训练时间。确保增加数量的设备之间的准确性所必需的同步和通信开销进一步限制了总体训练时间。用于处理大型数据集的常规技术导致效率低下,随着数据集变得越来越大以及模型变得更加复杂,效率低下变得更加明显。

附图说明

[0002] 将参考附图描述各种技术,其中:

[0003] 图1示出了其中实现根据本公开的实施例的系统的示意图;

[0004] 图2示出了根据一个实施例的表示利用一批输入训练神经网络的随机批梯度下降过程的示意图;

[0005] 图3示出了根据一个实施例的同步随机梯度下降过程的示意图,该过程表示使用多个计算设备利用一批输入对神经网络的数据并行训练。

[0006] 图4示出了根据一个实施例的使用两个计算设备利用一批输入对网络的数据并行训练的通用深度网络的示意图。

[0007] 图5示出了根据一个实施例的使用两个计算设备利用一批输入对网络的模型并行训练的通用深度网络的示意图;

[0008] 图6示出了根据一个实施例的使用数据并行训练和模型并行训练两者的通用深度网络的示意图;

[0009] 图7示出了根据一个实施例的用于训练神经网络的过程的说明性示例;

[0010] 图8示出了根据一个实施例的用于训练神经网络的过程的另一说明性示例;

[0011] 图9示出了根据一个实施例的并行处理单元(“PPU”)的示例;

[0012] 图10示出了根据一个实施例的通用处理集群(“GPC”)的示例;

[0013] 图11示出了根据一个实施例的内存分区单元的示例;

[0014] 图12示出了根据一个实施例的流式多处理器的示例;以及

[0015] 图13示出了根据一个实施例的可以在其中实现各种示例的计算机系统。

具体实施方式

[0016] 在一个实施例中,利用根据本公开实现的系统和方法来实现大型复杂的深度学习(DL)模型的效率增益(例如,改进训练时间)。DL模型的一个示例是神经网络。在一个实施例中,在本文中确定和描述了使用模型并行性(parallelism)(MP)和数据并行性(DP)两者来配置神经网络训练的改进方式。在一个实施例中,DP涉及使用训练数据集的独立子集(被称为小批(mini-batches))在独立工作器(worker)(例如,设备、处理器)上使用模型的副本。

MP涉及在多个工作器(例如,设备、处理器)之间分发正在训练的神经网络的组件,因此每个工作器具有正在训练的整个神经网络中的不同组件。在一个实施例中,MP涉及在处理同一小批时在多个工作器之间拆分(splitting)模型。在一个实施例中,训练数据集的子集(例如,小批)是馈送给每个工作器的一批输入。在一个实施例中,训练步骤中所有小批的集合是全局批(global batch)。在一个实施例中,工作器包括被组织为一组设备的一个以上设备或被组织为一组处理器以形成单个单元的一个以上处理器。在一个实施例中,工作器是一个设备或一个处理器。在一个实施例中,通过组合MP和DP的使用来确定训练神经网络的改进配置。在一个实施例中,采用MP和DP的组合以最小化可用硬件上给定神经网络的端到端训练时间。

[0017] 在一个实施例中,增加使用DP的并行性的数量(例如,不同工作器/设备/处理器使用的小批的数量),直到效率增益不再显著(例如,由一组重要性标准所定义的)。在这一点上,在一个实施例中,引入了MP,并且增加了与使用DP相结合的引入的MP并行性的量(例如,正被训练的神经网络被拆分成的组件的数量),以实现更大的效率增益(例如,改进的训练时间)。

[0018] 在一个实施例中,用于训练神经网络的数据集被细分为子集(例如,小批),并且跨诸如图形处理单元(GPU)(例如,使用DP)之类的多个计算设备处理用于形成全局批的小批的集合。在一个实施例中,多个张量处理单元(TPU)用于处理子集。随着全局批大小在数量或大小上的缩放(scale)/增加,训练给定神经网络的整体训练时间开始变得效率低下。在一个实施例中,随着全局批大小的增加,使用DP准确地训练神经网络所花费的时间量开始花费更长的时间。这样,在一个实施例中,第二类型的并行性(例如,MP)被实现为与DP结合工作以创建混合(hybrid)并行性(例如,DP加MP)。在一个实施例中,使用该混合并行性训练神经网络允许存在于给定工作器上的模型的每个副本都可以并行化模型,从而现在,每个工作器包括多个计算设备,而不是每个工作器是一个计算设备。在一个实施例中,每个神经网络将具有唯一的尺度(scale),在该尺度上通过引入MP与DP结合工作来训练网络并改进训练时间,可以克服DP的缩放和统计效率下降。因此,在一个实施例中,当缩放全局批大小时,混合并行性训练提供比单独使用DP更好的效率增益(例如,在最小化端到端训练时间上更有效)。在一个实施例中,本文描述的系统和方法使用所测量的信息来解析地确定战略,该信息被测量以指示单独DP的缩放效率低下和单独MP的加速(speedup),并且当为给定神经网络选择了要使用的正确机制时,将其用于选择运行的GPU的数量。

[0019] 在一个实施例中,使用本文描述的技术来确定何时实现使用MP与DP的混合并行性方法,以克服单独DP训练在缩放时(例如,当全局批大小的增加超出DP可以继续有效地处理以保持准确性的范围时)所经历的固有缩放效率损失和统计效率损失。在一个实施例中,与使用DP训练神经网络有关的端到端训练时间被监视和分析,以了解缩放效率损失和统计效率损失如何影响训练可扩展性(scalability),并且本文描述的技术指示添加MP允许针对给定神经网络实现训练加速,这对于混合并行化策略的整体可扩展性至关重要。拆分数据集(例如,使用DP)和拆分网络(例如,使用MP)每个都可分别提供效率增益,但是对于每种并行化方式,都有一个点,其中效率增益被组合并行训练的结果所需的开销所抵消。在一个实施例中,当DP中的全局批的大小增长到仅DP的训练加速显着下降的点(例如,训练时间更长)时,MP可以与DP结合使用,以改进超出单独DP所能实现的训练时间。

[0020] 在前面和下面的描述中，描述了各种技术。为了说明的目的，阐述了特定的配置和细节以便提供对实现技术的可能方式的透彻理解。然而，还将显而易见的是，以下描述的技术可以在没有具体细节的情况下以不同的配置来实践。此外，可以省略或简化众所周知的特征，以避免使所描述的技术模糊。

[0021] 图1示出了其中实现了根据本公开的实施例的系统100的示意图。在一个实施例中，系统100是包括数据存储102或数据存储服务/设备的计算环境。数据存储102可以被配置为存储数据、元数据、训练数据、训练数据线程等。在一个实施例中，训练数据包括图像数据或视频数据的集合，该图像数据或视频数据当由一个或更多个神经网络获得时，出于面部识别目的训练一个或更多个神经网络。在一个实施例中，训练数据包括语音数据或音频数据，该语音数据或音频数据当由一个或更多个神经网络获得时，训练一个或更多个神经网络以进行语音识别。在一个实施例中，数据存储102存储用于训练神经网络的训练数据104(例如，训练数据线程)。在一个实施例中，神经网络也被称为深度网络或简称为本文所述的网络。训练数据104可以包括诸如数组或矩阵之类的数据结构。训练数据104的子集可以被称为小批，并且小批的集合可以被称为全局批。在一个实施例中，用于训练神经网络的这些小批可以是重叠的或适当的子集(非重叠的)。在一个实施例中，数据存储102被配置为通过经由网络106(经由有线或无线网络)将训练数据104发送到计算设备108而满足获得训练数据104的请求。

[0022] 在一个实施例中，计算设备108是图形处理单元(GPU)，或者在另一个实施例中，其表示包括多个GPU的多个计算设备。在一个实施例中，计算设备108表示神经网络，其中神经网络使用训练数据104来训练网络。在一个实施例中，计算设备108被配置为接收或获得输入训练数据104，并确定要实现的并行化策略，使得计算设备108被训练为最大化训练效率。输入训练数据104可以由计算设备108经由网络106从系统100外部或内部的数据存储102、另一计算设备、计算服务、虚拟机、虚拟服务(图1中未示出)接收。

[0023] 在一个实施例中，计算设备108包括前端服务110。前端服务110可以是被配置为接收或获得输入训练数据104的计算机系统或其组件。在一个实施例中，前端服务110是包括一个或更多个处理器和用于存储指令的内存的服务前端，这些指令作为执行的结果使前端服务110执行本文所述的各种过程和方法。在一个实施例中，一个或更多个处理器中的每一个是逻辑处理器，其中每个逻辑处理器能够同时执行其自己的指令流。在一个实施例中，前端服务110配置有可执行代码，以提交和接收网络应用程序编程接口(API)请求。在一个实施例中，前端服务110是计算服务/设备，例如服务器计算机系统、服务器集群、虚拟计算机系统、虚拟服务器、虚拟运行时环境、容器环境、无服务器执行环境、服务托管系统或与计算设备108相关联并与其一起使用以接收输入训练数据104的任何合适的计算实体。

[0024] 在一个实施例中，前端服务110监视训练效率150，该训练效率150指示在使用DP 120、MP 130或DP和MP的组合140中的至少一个进行训练时计算设备108的训练时间、功耗和/或其他信息。即，在一个实施例中，前端服务110处理输入训练数据104并发送指示或指令集，以使至少一种类型的并行性被实现以训练计算设备108。在一个实施例中，所应用的一种类型的并行性是数据并行性(DP)120，其中通过使用训练数据104的独立子集，通过使用独立计算设备上的模型网络的副本来训练计算设备108。在一个实施例中，并且如上所述，训练数据104的子集被称为小批。在一个实施例中，所应用的另一种类型的并行性是模

型并行性(MP)130,其中当在相同的小批上工作时,模型跨多个工作器被拆分。在一个实施例中,所应用的另一种类型的并行性是混合并行性,其包括使用DP和MP两者来训练计算设备108。

[0025] 在一个实施例中,监视这些并行性策略中的每一个的训练效率150,并且将与训练效率150有关的信息作为对计算设备108的反馈提供给前端服务110或与计算设备108相关联的另一服务,以基于各种训练测试或预定的信息,确定在给定训练数据104和已接收的训练数据104的量的情况下,哪种并行性方法可用于最大化计算设备108的训练效率150。在一个实施例中,训练效率150是存储与训练时间有关的值或信息的数据存储、数据存储设备、缓冲区或消息队列。此外,在一个实施例中,训练效率150是与计算设备108分离但保持经由网络106与计算设备108的通信的数据存储。

[0026] 在一个实施例中,前端服务110从计算设备108中被移除,并且基于整数线性编程的工具(例如,DLPlacer)是与计算设备108结合运行的软件,使得当执行该工具时,最佳地放置工具以最大化训练加速(例如,最大化训练效率)。在一个实施例中,该工具被实现为当单独使用DP 120时产生指示训练效率150的结果,并且进一步向计算设备108提供信息以指示应将神经网络的多少部分进行拆分(例如,何时使用MP以及使用多少MP),以当训练数据104的全局批大小缩放到超出DP能够单独有效处理的范围时,提供最大效率。在一个实施例中,该工具被实现为产生结果,以指示单独MP的训练效率150,并且进一步向计算设备108提供信息以指示是否引入DP以及实现DP与MP。

[0027] 在一个实施例中,基于整数线性编程的工具向计算设备108指示如何最大化资源利用率。基于整数线性编程的工具可以提取网络中操作之间的并行性。在一个实施例中,作为使用基于整数线性编程的工具的结果,还可以最小化在节点之间移动数据的通信开销所需的资源。在一个实施例中,使用该工具,神经网络被表达为数据流图(DFG)。在一个实施例中,DFP的节点对应于计算操作和单向边缘,其中边缘示出了操作之间的依赖性。在一个实施例中,每个节点具有操作期望的执行时间和内存占用。在一个实施例中,边缘权重对应于在其中连接的操作之间交换的字节数。在一个实施例中,通过在计算设备(例如,GPU)上对模型进行性能分析(profile)来接收节点和边缘权重。在一个实施例中,可以解析地计算节点和边缘权重。在一个实施例中,该工具通过向硬件图(例如,放置),时间表以及激活、权重和梯度的通信路由提供计算DFG操作的分配来减少每步训练时间。在一个实施例中,当每个操作被映射到单个且唯一的设备、满足所有操作之间的依赖关系、在执行期间并置(co-located)的操作不重叠且总内存量不超过设备内存容量时,该工具返回正确的解决方案。在一个实施例中,在满足上面列出的所有条件的同时,该工具还假设两个操作(其被并置于设备上,被背对背执行)在一个操作的终止与另一个操作的开始之间没有任何延迟,在带宽为B且延迟为L的链路上传送大小为S的数据块需要(S/B+L)时间,并且设备之间的张量通信与计算重叠。在一个实施例中,基于这些假设,该工具预测给定MP解决方案的训练加速,然后将MP应用于与DP结合使用以更好地训练网络。

[0028] 如上所述,在一个实施例中,使用该工具或前端服务110的计算设备108确定使用DP 120、MP 130或DP和MP的组合140中的哪一个。在一个实施例中,随着全局批大小的继续增长,处理训练数据104的增加所需的处理器的量也随之增长。在一个实施例中,作为结果并且在某个点处,DP 120可以到达临界点(tipping point),在该临界点处,DP 120开始经

历与训练计算设备108有关的更长的训练时间(例如,训练慢下来)。在一个实施例中,该临界点也被称为交叉点(crossover-point)、阈值或最大效率值。在这一点上,在一个实施例中,计算设备108应用另一种类型的并行性,使得在引入这种其他类型的并行性之后训练时间被减少或改进。在一个实施例中,通过将训练时间与如果继续单独使用DP 120来训练计算设备108将要花费的训练时间进行比较以及与如果计算设备108使用DP和MP的组合140将要花费的训练时间进行比较来识别训练时间的改进。

[0029] 在一个实施例中,使用训练DP和MP的组合140来训练计算设备108,以达到又一个级别的训练效率。即,在一个实施例中,通过识别DP的临界点(例如,允许DP 120在不经历较长训练时间的情况下训练网络的训练数据或训练数据线程的最大量),确定何时应用混合并行化(对于各种神经网络,混合并行化优于单独DP通常在不同的尺度上发生的。在一个实施例中,如果在使用DP和MP的组合140来训练计算设备108时,未能满足该特定级别的训练效率,则MP减少,并且仅使用DP 120和减少MP 130或者仅单独使用DP 120来训练计算设备108。在一个实施例中,减少了DP 120而不是MP 130。

[0030] 这样,在一个实施例中,训练效率150基于来自与训练计算设备108相关联的训练时间的测量。在一个实施例中,某水平的训练效率指示与使用训练数据DP和MP的组合140训练计算设备108相关联的训练时间等于或低于与仅使用DP训练计算设备108相关联的训练时间,并且因此该信息可用于使前端服务110或与计算设备108相关联的基于整数线性编程的工具使用DP和MP 140来应用该混合并行化策略,直到训练计算设备108达到另一最大效率为止。

[0031] 图2示出了根据一个实施例的表示利用一批输入训练神经网络的随机批梯度下降过程的示意图200。在神经网络训练中,在一个实施例中,首先一批输入通过网络前向传播(前向传递)202,使得针对该批输入中的每个输入执行损失计算。然后损失通过网络反向传播(反向传递)204,从而执行梯度计算。在一个实施例中,然后将批的梯度合在一起并取平均值,并进一步用于更新权重206。在一个实施例中,该批输入具有被选择的大小,使得用于训练的特定设备的所有计算资源可以充分利用。在一个实施例中,该整个过程被称为表示训练神经网络的随机批梯度下降。即,在一个实施例中,一个前向传递202和反向传递204一起对与权重206相关联的梯度的更新被称为训练步(training step)。在一个实施例中,假设所有输入被处理一次,则运行通过训练数据集的单个迭代涉及多个步(step)并且被称为时期(epoch)。在一个实施例中,训练过程(如图2所示)运行多个时期,直到达到一些预定的或最佳的训练精度。

[0032] 图3示出了根据一个实施例的表示使用多个计算设备利用一批输入的DP训练神经网络的同步随机梯度下降过程的示意图300。在一个实施例中,当跨多个计算设备/工作器(例如,GPU 0、GPU 1,...,GPU N-1)复制模型参数(例如,权重)全集时,使用DP加快了训练速度。如图3所示,每个工作器在不同批的输入上通过前向传递302和反向传递304独立地工作。在至少一个实施例中,然后在每个工作器之间传送梯度并将其取平均。此后,在一个实施例中,然后每个工作器将相同的梯度值集应用于模型权重306,并且使用全约(all-reduce)通信308来完成每个工作器之间的梯度的任何通信。在一个实施例中,在每次迭代(通过使用对所有梯度计算的平均值)后执行模型更新。在一个实施例中,将每个工作器的该批输入标识为小批,并且在训练步中所有小批的集合是全局批。

[0033] 图4示出了根据一个实施例的使用两个计算设备(例如,2个GPU)利用一批输入的DP训练网络的通用深度网络的示意图400。在一个实施例中,每个计算设备为数据集的子集(或者也被称为来自全局批输入的小批输入)计算误差和梯度。如上所述,数据集可以由诸如数组或矩阵之类的数据结构组成,其中数据结构中的每个元素都用于训练神经网络。训练数据可以包括标记数据,例如图像数据、视频数据、语音数据、音频数据、文本等。在一个实施例中,然后使用外围组件互连快速(PCI-e)或计算设备之间的任何其他通信协议在两个计算设备之间传送参数和梯度。

[0034] 在一个实施例中,如图4中所示的示意图400仅示出了两个计算设备;但是,两个以上的计算设备被实现为训练深度网络。如本文所述,随着全局批大小的继续增加或缩放,单独DP实现可能开始经历效率低下(例如,更长的训练时间),并且实现整个本公开内容以及在图5中进一步描述的诸如MP之类的附加并行性技术可以提高训练深度网络的效率增益。

[0035] 图5示出了根据一个实施例的使用两个计算设备利用一批输入的MP训练网络的通用深度网络的示意图500。在一个实施例中,通过将示意图500的不同操作或放置在不同设备上来拆分网络。传统上,MP训练用于其参数无法放入单个设备的内存的网络。然而,在一个实施例中,MP在整个网络适合于一个设备的情况下提供每步训练加速。在一个实施例中,如图5所示,在各种设备上同时执行独立操作。对于许多网络而言,在多个设备之间拆分数据流图(DFG)并非易事。在一个实施例中,关于与在设备之间移动数据有关的计算节点之间的通信的开销是相当大的,因此有时其可能超过MP所提供的任何收益。因此,在一个实施例中,对用于网络的DFG进行了划分,并且应当考虑许多考虑因素,例如针对这些设备中的每个设备的计算强度、设备间的带宽以及甚至网络拓扑。

[0036] 图6示出了根据一个实施例的使用DP和MP两者正在训练的通用深度网络的示意图600。在一个实施例中,当全局批大小被缩放时,一起使用DP和MP两者可以改进与训练神经网络相关联的端到端训练时间。在一个实施例中,存在至少三个因素,其导致神经网络的端到端训练时间的改变:每步的平均时间(T)、每个时期的步数(S)和收敛到期望的精度所需的时期的数量(E)。(上面参考图2的描述讨论了训练步和时期的定义。)在一实施例中,训练时间的总量(例如,收敛的时间(C))表示为:

[0037] $C = T \times S \times E$ (1)

[0038] 在一个实施例中,T主要由计算效率来确定,即在给定相同的训练设置、算法和小批大小的情况下,T仅取决于设备的计算能力。在一个实施例中,处理更好性能的硬件设备提供较小的T值。然而,在一个实施例中,S取决于训练数据集中的输入量,并且还取决于全局批大小。在一个实施例中,数据集中的每个输入或每个项每个时期被处理一次。因此,在一个实施例中,每个时期的步数(S)等于数据集中的项目数,除以全局批大小(例如,训练输入数据的大小)。在一个实施例中,收敛的时期数(E)取决于全局批大小和其他训练超参数(hyper-parameters)。

[0039] 在一个实施例中,在将MP与DP组合并应用于训练网络之前,监视、计算和/或分析单独使用DP的训练时间。在一个实施例中,与在单个设备上进行训练相比,使用N路(N-way)数据并行性(SU_N)来利用单独使用DP加快训练速度,其表示为:

[0040] $$SU_N = \frac{T_1}{T_N} \times \frac{S_1}{S_N} \times \frac{E_1}{E_N} \qquad (2)$$

[0041] 在一个实施例中，T_1是当仅使用单个设备进行训练时每步的平均训练时间，而T_N是当使用N个数据并行设备(每个设备具有恒定的小批大小)时每步的时间。在一个实施例中，T_N总是大于T_1，这主要是由于以下事实：在DP中，在每个设备执行了前向传递和反向传递之后，使用全约通信(参见对图3的描述)在设备之间交换梯度。在一个实施例中，由$\frac{T_1}{T_N}$表示的通信开销将永远不会大于一，并且通常小于一。在一个实施例中，该比率$\frac{T_1}{T_N}$是N路DP的缩放效率(SE_N)。

[0042] 在一个实施例中，S_1表示在使用单个设备的情况下每个时期所需的总步数，而S_N是在使用N个设备时每个时期所需的步数。在一个实施例中，在使用单个设备的情况下，全局批大小等于小批大小。在一个实施例中，每个设备的N路数据并行性以其自己的小批数据执行独立的步。因此，在一个实施例中，全局批大小是每个设备的小批大小的N倍。因此，在一个实施例中，$\frac{S_1}{S_N}$也等于N。

[0043] 在一个实施例中，E_1表示在使用单个设备的情况下收敛的时期数。在一个实施例中，E_N是当使用N数量的设备时所需的时期数。在一个实施例中，在具有较大的全局批大小(高于数量N)的情况下，对来自较大数量的训练样本的梯度取平均。在一个实施例中，作为平均的结果，其导致模型过度拟合(over-fitting)以及被吸引到局部极小值或鞍点(saddle points)的趋势。在一个实施例中，然后网络将需要更多的时期来收敛。这样，在一个实施例中，$\frac{E_1}{E_N}$通常小于一。因此，方程2可以简化为：

[0044] $$SU_N = SE_N \times N \times \frac{E_1}{E_N} \quad (3)$$

[0045] 在一个实施例中，当用设备数量(N)训练时，SE_N和$\frac{E_1}{E_N}$两者都减小。在一个实施例中，在具有更大的全局批大小的情况下，应用超参数调整以努力最小化收敛所需的时期数量的增加。然而，在一个实施例中，对于任何特定的网络，超过某个全局批大小，收敛所需的时期数量迅速增加，而与超参数调整无关。

[0046] 在一个实施例中，还在给定神经网络上监视、计算和/或分析MP训练时间。如上所述，仅使用单独MP不能很好地缩放，因此不能被认为是具有大量设备和数据集的广泛适用的策略。但是，出于参考目的，本文中可描述使用MP的训练时间。即，在一个实施例中，使用MP使多个设备能够同时对同一小批执行操作。在一个实施例中，使用MP减少单个步的训练步时间(例如，方程1中的项“T”)。在一个实施例中，来自M路MP的加速由SU^M表示。在一个实施例中，通过在多个设备之间拆分网络并测量每步执行时间来测量M路MP的测量。在一个实施例中，应注意，SU^M加速已经包括与跨多个设备放置的相关操作之间的数据移动有关的通信开销。

[0047] 如前所述，在一个实施例中，仅使用MP时，全局批大小保持不变。因此，在一个实施例中，每个时期的步数(例如，方程1中的项“S”)和收敛所需的时期数(例如，方程1中的项“E”)保持不变。在一个实施例中，当改善SU^M时，它通过仅减小方程1中的项T(而其他两项保

持不变)来最小化收敛时间。在一个实施例中,单独MP不被认为是广泛适用的可扩展并行化策略。然而,在一个实施例中并且如贯穿本公开所描述的,MP与DP组合以将训练可扩展性扩展到超出单独DP和单独MP可提供的限制。

[0048] 在一个实施例中,应用混合并行性策略(例如,使用DP和MP的组合)以实现训练神经网络的最大效率(例如,改进训练时间)。如上所述,在一个实施例中,使用如方程3所提供的N路DP来获得加速。在一个实施例中,假设使用N路DP将系统扩展至多达N个设备,并且所达到的训练加速等于或超过最大效率,并且如果其他设备变为可用于训练,则可以单独使用DP或使用混合方法来训练神经网络。在一个实施例中,网络识别何时单独使用DP,以及何时将DP与MP结合以获得最大可能的训练加速(例如,最大化效率或训练时间)。在一个实施例中,通过单独应用DP,与使用一个设备相比,使用附加设备(例如,M×N)的加速为(用M×N代替方程3中的N):

[0049] $$SU_{M\times N} = \mathrm{SU}_{M\times N} \times M \times N \times \frac{E_1}{E_{M\times N}} \qquad (4)$$

[0050] 在一个实施例中,比较M×N路DP的加速(方程4)和N路DP的加速(方程3)。在一个实施例中,与N路相比,具有M×N路DP的系统的缩放效率通常较低,因此在大量设备之间发生全约通信,并且因此导致训练时间变慢。在一个实施例中,取决于N、M的值以及系统配置,全约通信可能跨越较慢的节点间链路。在一个实施例中,这导致关于全约时间的增加并且减小了$SE_{M\times N}$。此外,在一个实施例中,由于在M×N个设备处全局批大小较大(以保持恒定的小批大小),所以与N路DP相比,每个时期的步数减少了因子M。甚至进一步,在一个实施例中,所需的时期数$E_{M\times N}$大于或等于E_N。在一个实施例中,随着DP训练中使用的设备数量的增加,这些因子都提供较低的效率。

[0051] 在一个实施例中,M×N个设备在N路DP的混合并行化策略中的应用,其中每个工作器使用M路MP,每个工作器的每步加速用SU^M表示。因此,总训练时间/加速表示为:

[0052] $$SU_N^M = SU^M \times SE_N \times N \times \frac{E_1}{E_N} \quad (5)$$

[0053] 在一个实施例中,在具有M路模型并行工作器的混合N路DP与具有单个GPU工作器的N路DP之间进行比较。在一个实施例中,全局批大小将保持不变。在一个实施例中,在M×N个设备配置中,将M个设备中的每个设备分组为单个数据并行工作器。在一个实施例中,每个时期的步数不改变,因为N和$\frac{E_1}{E_N}$处的N路DP的每个时期的步数也保持相同。这样,在一个实施例中,当比较方程3和方程5时,通过MP实现的每步加速将总训练加速增加了因子SU^M。

[0054] 在一个实施例中,将方程4和方程5代入方程6,并且作为结果,确定指示使用混合并行化(例如,使用DP和MP的组合)比单独进行DP缩放更好的条件。在一个实施例中,方程6指示,如果从MP(对于给定模型并行化步)获得的加速足够大以克服分别由于通信增加、同步开销和全局批大小增加而引起的缩放和统计效率损失,则混合并行化(例如,使用DP和MP的组合)策略的应用改进了网络训练时间。

$$SU_N^M > SU_{M\times N}$$

[0055] $$SU^M \times SE_N \times N \times \frac{E_1}{E_N} > SE_{M\times N} \times M \times N \times \frac{E_1}{E_{M\times N}} \quad (6)$$

$$SU^M > M \times \frac{SE_{M\times N}}{SE_N} \times \frac{E_N}{E_{M\times N}}$$

[0056] 在一个实施例中，在系统中，当分别使用两个和四个GPU时，MP的实现在训练效率方面提供了45％和65％的改善。在一个实施例中，仅使用DP是很好的策略，其对于多达32个设备很好地缩放，此后训练时间或训练加速的改进变慢。在一个实施例中，对于该示例，在给定64个设备时的缩放和统计效率损失的情况下，这使得混合32路DP和2路MP混合并行化策略能够比64路DP更好地执行。

[0057] 在一个示例中，类似地，当设备从32个设备扩展到128个设备时，混合16路DP和4路MP混合策略才胜过仅使用DP。但是，在此示例中，此混合策略的性能比32路DP和2路MP的混合策略效率低。在一个实施例中，4路MP的每步加速(SU^4)不能像两路MP的每步加速SU^2(每个数据并行工作器使用两台机器时)那样有效地克服权衡(每个数据并行工作器使用四台机器)。因此，在一个实施例中，相对于任何设备计数的相对改善是在选择并行化策略之前提供的一个因素，这对于扩展到更大数量的设备时所获得的训练加速至关重要。在一个实施例中，大多数情况下，选择取决于如上所述的神经网络的属性和系统配置参数，因此没有单一解决方案或一个大小适合所有的解决方案以有效地扩展多设备训练。

[0058] 图7示出了根据一个实施例的用于训练神经网络的过程700的说明性示例。即，在一个实施例中，使用第一数量的并行训练数据线程来训练神经网络，这导致训练效率的第一级别702。在一个实施例中，第一数量的并行训练数据线程被称为如上文关于图1所述的全局批大小。例如，通过在多个并行工作的计算设备(例如，DP)上使用全局批来训练神经网络。在一个实施例中，训练效率的第一级别指示当使用DP训练神经网络时的训练时间。在一个实施例中，训练效率的第一级别也被称为训练效率的中间级别。

[0059] 在一个实施例中，考虑到网络的配置参数和全局批的大小，使用DP训练神经网络，这导致训练效率的第一级别。在一个实施例中，使用DP训练神经网络(其导致效率的第一级别)是已经达到最大训练效率的指示。如果进一步增加DP，则由于合并了并行运行的所有设备(节点)的结果而导致的额外开销，网络将导致效率降低。在一个实施例中，不是达到最大效率，而是在给定有限数量的可用于训练网络的设备(节点)的情况下，效率不能再变得更好时，达到训练效率的第一级别。例如，尽管网络可以通过单独使用DP来获得某种训练效率的增益，但是增益可能很小，因此网络可以确定通过引入MP(例如，此处增益更显著)获得更大的增益。如结合图1所描述的，在一个实施例中，结合神经网络来运行前端服务或软件工具(例如，DLPlacer)，其确定何时使用DP训练网络达到该效率的第一级别。

[0060] 在一个实施例中，一旦已满足效率的第一级别，然后前端服务或软件工具(例如，DLPlacer)确定使用神经网络的第二数量的部分并行地使用第一数量的并行训练数据线程训练神经网络，导致训练效率的第二级别704。在一个实施例中，现在结合使用第一数量的并行训练数据线程和并行的神经网络的多个部分(例如，使用DP和MP两者)来训练神经网

络。在一个实施例中，如以上结合图1所描述的，确定并应用神经网络的该多个部分，使得训练神经网络以达到效率的第二级别。在一个实施例中，通过分析性地测试神经网络的多个不同部分并进一步应用不同数量的部分，来执行对使用DP和MP的最佳组合来训练网络的确定。在一个实施例中，尽管测试是分析性地执行的，但是测试还是非分析性地执行的。

[0061] 在一个实施例中，如以上关于图6所描述的，如果从MP获得的加速(对于给定的模型并行化步)足够大以克服因单独使用DP时增加的通信、同步开销和全局批大小增加而导致的缩放和统计效率损失，则采用混合并行化(例如，使用DP和MP的组合)策略允许以效率的第二级别(这指示训练时间有所改进)训练神经网络。

[0062] 作为描述图7的过程的示例，用于训练神经网络的数据集首先被拆分为四个子集，其中获得效率的第一级别(例如，最大效率)：T1、T2、T3和T4。在一个实施例中，随后，当满足效率的第一级别时，神经网络的部分被拆分并应用，使得训练达到效率的第二级别(例如，另一最大效率)。在一个实施例中，神经网络采用使用关于用于训练网络的最有效方式的MP和DP的组合的策略来实现。在一个实施例中，通过将神经网络拆分为三个部分：M1、M2和M3来有效地训练神经网络。在一个实施例中，然后混合并行性方法指示T1、T2、T3和T4中的每一个被用来训练M1、M2和M3中的每一个。在此示例中，使用12个处理器(四个训练子集乘以三个模型组件)进行训练将是最有效的，其中一个用于训练子集和模型组件的组合。也就是说，在一个实施例中，至少部分地基于神经网络的训练效率，通过改变训练数据集在多个计算设备之间如何分布以及在多个计算设备之间如何拆分和分发神经网络的组件两者，来有效地训练神经网络。

[0063] 图8示出了根据一个实施例的用于训练神经网络的过程800的另一说明性示例。也就是说，在一个实施例中，通过首先确定第一数量的并行训练数据线程(例如，使用DP)，来训练神经网络至达到或高于训练效率的第一级别(例如，最大效率)802。在一个实施例中，如图1中所示，训练数据线程是全局批大小。在一个实施例中，过程800描述了神经网络，该神经网络通过调整第一类型的并行性(例如，DP)来进行训练，以至少部分基于指示训练效率的第一训练测试集来训练神经网络，直至达到训练效率的第一级别。在一个实施例中，执行该操作直到神经网络达到使用增加的全局批的大小对网络的任何进一步训练导致训练变慢的点为止。在一个实施例中，与所有设备(节点)传送结果所需的开销达到开始变得低效的点。在一个实施例中，如关于图7所描述的，神经网络被训练至达到或高于的训练效率的第一级别不是最大效率，而是与引入MP相比网络获得的效率增益更低的级别。

[0064] 如图8所示，在一个实施例中，通过确定要并行训练(例如，使用MP)的神经网络的第二数量的部分同时使用第一数量的并行训练数据线程，来进一步训练神经网络至达到或高于效率的第二级别804。在一个实施例中，通过实现第二类型的并行性(例如，包括DP和MP两者的组合的混合并行性)来进一步训练神经网络，以至少部分地基于指示训练效率的另一级别的第二训练测试集来训练神经网络。在一个实施例中，如以上关于图1和图6所指示的，执行训练测试以确定要引入的MP的数量，并进一步实现以与DP结合使用，以改进训练网络的训练时间。在一个实施例中，分析地或非分析地进行测试以确定要引入的MP的数量。在一个实施例中，如果应用MP(对于给定模型并行化步)的测试结果足够大，可以克服因单独使用DP时增加的通信、同步开销和全局批大小增加而导致的缩放和统计效率损失，则采用混合并行化(例如，使用DP和MP的组合)策略可以改进网络训练时间。在一个实施例中，如关

于图1所描述的，训练效率指示训练时间并且被测量且表示为一个值，使得当其被分析时，向用户或单独的系统提供关于神经网络是否被有效训练的信息。

[0065] 图9示出了根据一个实施例的并行处理单元(“PPU”)900。在一个实施例中，PPU 900配置有机器可读代码，该机器可读代码如果由PPU执行，则使得PPU执行贯穿本公开描述的某些或全部过程和技术。在一个实施例中，PPU 900是在一个或更多个集成电路器件上实现的，并且利用多线程作为被设计为在多个线程上并行处理计算机可读指令(也被称为机器可读指令或简单指令)的延迟隐藏技术的多线程处理器。在一个实施例中，线程是指执行线程，并且是被配置为由PPU 900执行的指令集的实例。在一个实施例中，PPU 900是被配置为实现图形渲染管线的图像处理单元(“GPU”)，该图形渲染管线用于处理三维(“3D”)图形数据以便生成二维(“2D”)图像数据，以在显示设备(例如，液晶显示器(LCD)设备)上显示。在一个实施例中，PPU 900被用于执行诸如线性代数运算和机器学习运算之类的计算。图9仅出于示例性目的示出了示例并行处理器，并且应被解释为在本公开的范围内考虑的处理器架构的非限制性示例，并且可以采用任何适当的处理器来对其进行补充和/或替代。

[0066] 在一实施例中，一个或更多个PPU被配置为加速高性能计算(“HPC”)、数据中心和机器学习应用。在一个实施例中，PPU 900被配置为加速深度学习系统和应用，包括以下非限制性示例：自主车辆平台、深度学习、高精度语音、图像、文本识别系统、智能视频分析、分子模拟、药物发现、疾病诊断、天气预报、大数据分析、天文学、分子动力学模拟、财务建模、机器人技术、工厂自动化、实时语言翻译、在线搜索优化以及个性化用户推荐等。

[0067] 在一个实施例中，PPU 900包括输入/输出(“I/O”)单元906、前端单元910、调度器单元912、工作分配单元914、集线器916、交叉开关(“Xbar”)920、一个或更多个通用处理集群(“GPC”)918和一个或更多个分区单元922。在一个实施例中，PPU 900经由一个或更多个高速GPU互连908连接到主机处理器或其他PPU 900。在一个实施例中，PPU 900经由互连908连接到主机处理器或其他外围设备。在一个实施例中，PPU 900连接到包括一个或更多个内存设备904的本地内存。在一个实施例中，本地内存包括一个或更多个动态随机存取存储器(“DRAM”)设备。在一个实施例中，一个或更多个DRAM设备被配置为和/或可配置为高带宽存储器(“HBM”)子系统，并且每个设备内堆叠有多个DRAM管芯。

[0068] 高速GPU互连908可指代基于有线的多通道通信链路，系统使用该链路来缩放，并包括与一个或更多个CPU结合的一个或更多个PPU 900、支持PPU 900和CPU之间的高速缓存一致性以及CPU主控。在一个实施例中，高速GPU互连908通过集线器916向/从PPU 900的其他单元(例如，一个或更多个复制引擎、视频编码器、视频解码器、电源管理单元以及未在图9中明确示出的其他组件)发送数据和/或命令。

[0069] 在一个实施例中，I/O单元906被配置为通过系统总线902从主机处理器(图9中未示出)发送和接收通信(例如，命令、数据)。在一个实施例中，I/O单元906直接经由系统总线902或通过一个或更多个中间设备(例如，内存桥)与主机处理器通信。在一个实施例中，I/O单元906可以经由系统总线902与一个或更多个其他处理器(例如，一个或更多个PPU 900)通信。在一个实施例中，I/O单元906实现用于通过PCIe总线进行通信的外围组件互连高速(“PCIe”)接口。在一个实施例中，I/O单元906实现用于与外部设备通信的接口。

[0070] 在一个实施例中，I/O单元906对经由系统总线902接收的分组进行解码。在一个实施例中，至少一些分组表示被配置为使PPU 900执行各种操作的命令。在一个实施例中，如

命令指定的那样，I/O单元906将经解码的命令发送到PPU 900的各种其他单元。在一个实施例中，命令被发送到前端单元910和/或被发送到集线器916或PPU 900的其他单元，例如一个或更多个复制引擎、视频编码器、视频解码器、电源管理单元等(在图9中未明确示出)。在一个实施例中，I/O单元906被配置为在PPU 900的各个逻辑单元之间和之中路由通信。

[0071] 在一个实施例中，由主机处理器执行的程序对缓冲区中的命令流进行编码，该缓冲区将工作负载提供给PPU 900以进行处理。在一个实施例中，工作负载包括指令和要由那些指令处理的数据。在一个实施例中，缓冲区是内存中主机处理器和PPU 900两者都可访问(例如，读/写)的区域—主机接口单元可被配置为经由由I/O单元906通过系统总线902发送的内存请求来访问连接到系统总线902的系统内存中的缓冲区。在一个实施例中，主机处理器将命令流写入缓冲区，然后将指向命令流的开始的指针发送给PPU 900，以使前端单元910接收指向一个或更多个命令流的指针，并管理一个或更多个流、从流中读取命令并将命令转发到PPU 900的各个单元。

[0072] 在一个实施例中，前端单元910耦合到调度器单元912，该调度器单元912配置各个GPC 918以处理由一个或更多个流定义的任务。在一个实施例中，调度器单元912被配置成跟踪与由调度器单元912管理的各种任务有关的状态信息，其中状态信息可以指示任务被分配给哪个GPC 918、任务是活动的还是不活动的、与任务相关联的优先级等。在一个实施例中，调度器单元912管理在一个或更多个GPC 918上的多个任务的执行。

[0073] 在一个实施例中，调度器单元912耦合到工作分配单元914，该工作分配单元914被配置为分派任务以在GPC 918上执行。在一个实施例中，工作分配单元914跟踪从调度器单元912接收的多个被调度任务，并且工作分配单元914为GPC 918中的每一个管理待处理(pending)任务池和活动任务池。在一个实施例中，待处理任务池包括多个时隙(例如，32个时隙)，这些时隙包含被指派为由特定GPC 918处理的任务；活动任务池可包括用于由GPC 918正积极处理的任务的多个时隙(例如，4个时隙)，使得当GPC 918完成任务的执行时，该任务从GPC 918的活动任务池中被逐出，并且来自待处理任务池的其他任务之一被选择并调度，以在GPC 918上执行。在一个实施例中，如果活动任务在GPC 918上是空闲的，例如在等待解决数据依赖性时，则活动任务从GPC 918中被逐出并返回到待处理任务池，同时待处理任务池中的另一个任务被选择并调度，以在GPC 918上执行。

[0074] 在一个实施例中，工作分配单元914经由XBar 920与一个或更多个GPC 918通信。在一个实施例中，XBar 920是将PPU 900的许多单元耦合到PPU 900的其他单元的互连网络，并且可以被配置为将工作分配单元914耦合到特定的GPC 918。尽管未明确示出，但PPU 900的一个或更多个其他单元也可以经由集线器916连接到XBar 920。

[0075] 任务由调度器单元912管理，并由工作分配单元914分派给GPC 918。GPC 918被配置为处理任务并生成结果。结果可以被GPC 918中的其他任务消耗、可以经由XBar 920路由到不同的GPC 918或者可以存储在内存904中。结果可以经由分区单元922写入内存904，分区单元922实现用于从内存904读取数据和向内存904写入数据的内存接口。结果可以经由高速GPU互连908发送到另一个PPU 904或CPU。在一个实施例中，PPU 900包括数量为U个分区单元922，其等于耦合到PPU 900的独立且不同的内存设备904的数量。下面将结合图11更详细地描述分区单元922。

[0076] 在一个实施例中，主机处理器执行驱动器内核，该驱动器内核实现了应用程序编

程接口(API),其使在主机处理器上执行的一个或更多个应用程序能够调度操作以在PPU 900上执行。在一个实施例中,PPU 900同时执行多个计算应用程序,并且PPU 900为多个计算应用程序提供隔离、服务质量("QoS")和独立的地址空间。在一个实施例中,应用程序生成指令(例如,以API调用的形式),该指令使驱动器内核生成一个或更多个任务以由PPU 900执行,并且驱动器内核将任务输出至正由PPU 900处理的一个或更多个流。在一个实施例中,每个任务包括一个或更多个相关线程组,其可以被称为线程束(warp)。在一个实施例中,线程束包括可以并行执行的多个相关线程(例如,32个线程)。在一个实施例中,协作线程可以指包括执行任务并且通过共享内存交换数据的指令的多个线程。根据一个实施例,结合图11更详细地描述了线程和协作线程。

[0077] 图10示出了根据一个实施例的诸如图9的PPU 900所示的GPC之类的GPC 1000。在一个实施例中,每个GPC 1000包括用于处理任务的多个硬件单元,并且每个GPC 1000包括管线管理器1002、预光栅操作单元("PROP")1004、光栅引擎1008、工作分配交叉开关("WDX")1016、内存管理单元("MMU")1018、一个或更多个数据处理集群("DPC")1006以及各部分的任何适当组合。将理解的是,图10的GPC 1000可以包括代替或附加于图10所示的单元的其他硬件单元。

[0078] 在一实施例中,GPC 1000的操作由管线管理器1002控制。管线管理器1002管理一个或更多个DPC 1006的配置,以处理分配给GPC 1000的任务。在一个实施例中,管线管理器1002配置一个或更多个DPC 1006中的至少一个以实现图形渲染管线的至少一部分。在一个实施例中,DPC1006被配置为在可编程流式多处理器("SM")1014上执行顶点着色器程序。管线管理器1002被配置为将从工作分配接收的分组路由到GPC 1000内的适当逻辑单元,并且在一个实施例中,某些分组可以被路由到PROP1004和/或光栅引擎1008中的固定功能硬件单元,而其他分组可以被路由到DPC 1006以由图元引擎1012或SM 1014进行处理。在一个实施例中,管线管理器1002配置一个或更多个DPC 1006中的至少一个以实现神经网络模型和/或计算管线。

[0079] 在一个实施例中,PROP单元1004被配置为将由光栅引擎1008和DPC1006生成的数据路由到内存分区单元中的光栅操作("ROP")单元,如上更详细地描述的。在一个实施例中,PROP单元1004被配置为执行用于颜色混合的优化、组织像素数据、执行地址转换等。在一个实施例中,光栅引擎1008包括被配置为执行各种光栅操作的多个固定功能硬件单元,并且光栅引擎1008包括设置引擎、粗光栅引擎、剔除引擎、裁剪引擎、精细光栅引擎、图块合并引擎及其任何合适的组合。在一个实施例中,设置引擎接收变换后的顶点并生成与由顶点定义的几何图元相关联的平面方程;平面方程被发送到粗光栅引擎以生成图元的覆盖信息(例如,图块的x,y覆盖掩码);粗光栅引擎的输出被发送到剔除引擎,其中与z测试(z-test)失败的图元相关联的片段被剔除,并被发送到裁剪引擎,其中位于视锥体范围之外的片段将被裁剪掉。在一个实施例中,经过裁剪和剔除后留下来的片段被传递给精细光栅引擎,以基于由设置引擎生成的平面方程生成像素片段的属性。在一个实施例中,光栅引擎1008的输出包括要由任何合适的实体(例如,由在DPC 1006内实现的片段着色器)处理的片段。

[0080] 在一个实施例中,GPC 1000中包括的每个DPC 1006包括M-管道(M-Pipe)控制器("MPC")1010;图元引擎1012;一个或更多个SM 1014;及其任何合适的组合。在一个实施例

中,MPC 1010控制DPC 1006的操作,将从管线管理器1002接收的分组路由到DPC 1006中的适当单元。在一个实施例中,与顶点相关联的分组被路由到图元引擎1012,图元引擎1012被配置为从内存中获取与顶点相关联的顶点属性;相反,与着色器程序相关联的分组被发送到SM 1014。

[0081] 在一个实施例中,SM 1014包括可编程流式处理器,其被配置为处理由多个线程表示的任务。在一个实施例中,SM 1014是多线程的并且被配置为同时执行来自特定线程组的多个线程(例如,32个线程),并且实现SIMD(单指令、多数据)架构,其中线程组(例如,线程束)中的每个线程被配置为基于相同的指令集来处理不同的数据集。在一个实施例中,线程组中的所有线程执行相同的指令。在一个实施例中,SM 1014实现SIMT(单指令、多线程)架构,其中线程组中的每个线程被配置为基于相同的指令集来处理不同的数据集,但是其中在执行期间允许线程组中的各个线程发散。在一个实施例中,为每个线程束维护程序计数器、调用栈和执行状态,从而当线程束内的线程发散时,实现线程束和线程束内的串行执行之间的并发。在另一个实施例中,为每个个体线程维护程序计数器、调用栈和执行状态,从而实现线程束之内和线程束之间的所有线程之间的同等的并发。在一个实施例中,为每个个体线程维护执行状态,并且为了更好的效率,执行相同指令的线程可以收敛并且并行地执行。在一个实施例中,下面将更详细地描述SM 1014。

[0082] 在一个实施例中,MMU 1018在GPC 1000和内存分区单元之间提供接口,并且MMU 1018提供虚拟地址到物理地址的转换、内存保护以及内存请求的仲裁。在一个实施例中,MMU 1018提供一个或更多个转换后备缓冲区(“TLB”),用于执行虚拟地址到内存中的物理地址的转换。

[0083] 图11示出了根据一个实施例的PPU的内存分区单元。在一个实施例中,内存分区单元1100包括光栅操作(“ROP”)单元1102;二级(“L2”)高速缓存1104;内存接口1106以及其任何合适的组合。内存接口806耦合到内存。内存接口1106可以实现32、64、128、1024位数据总线等,用于高速数据传输。在一个实施例中,PPU包括U个内存接口1106,每对分区单元1100一个内存接口1106,其中每对分区单元1100连接到相对应的内存设备。例如,PPU可以连接到多达Y个内存设备,例如高带宽内存堆叠或图形双数据速率版本5同步动态随机存取存储器(“GDDR5 SDRAM”)。

[0084] 在一个实施例中,内存接口1106实现了HBM2内存接口,并且Y等于U的一半。在一个实施例中,HBM2内存堆叠与PPU位于同一物理封装上,与常规的GDDR5 SDRAM系统相比,节省了大量的功率和面积。在一个实施例中,每个HBM2堆叠包括四个内存管芯,并且Y等于4,而HBM2堆叠包括每个管芯两个128位通道,总共8个通道,并且数据总线宽度为1024位。

[0085] 在一个实施例中,内存支持单纠错双检错(“SECDED”)纠错码(“ECC”)以保护数据。ECC为对数据损坏敏感的计算应用程序提供了更高的可靠性。在PPU处理非常大的数据集和/或长时间运行应用程序的大规模集群计算环境中,可靠性尤其重要。

[0086] 在一个实施例中,PPU实现多级内存层次结构。在一个实施例中,内存分区单元1000支持统一内存以为CPU和PPU内存提供单个统一虚拟地址空间,从而实现虚拟内存系统之间的数据共享。在一个实施例中,追踪PPU对位于其他处理器上的内存的访问频率,以确保将内存页移动到更频繁地访问页面的PPU的物理内存。在一个实施例中,高速GPU互连908支持地址转换服务,该地址转换服务允许PPU直接访问CPU的页表并提供由PPU对CPU内存的

完全访问。

[0087] 在一个实施例中,复制引擎在多个PPU之间或PPU与CPU之间传输数据。在一个实施例中,复制引擎可以为未被映射到页表中的地址生成页面错误,并且然后内存分区单元1100为页面错误提供服务,将地址映射到页表中,之后复制引擎执行传输。在一个实施例中,为多个处理器之间的多个复制引擎操作固定内存(即,不可分页),从而显著减少了可用的内存。在一个实施例中,由于硬件页面故障,可以将地址传递给复制引擎,而不必考虑内存页是否驻留,并且复制过程是透明的。

[0088] 根据一个实施例,来自图9的内存或其他系统内存的数据由内存分区单元1100提取并存储在L2高速缓存1104中,该L2高速缓存1104位于芯片上并且在各个GPC之间共享。在一个实施例中,每个内存分区单元1100包括与对应的内存设备相关联的L2高速缓存1060的至少一部分。在一个实施例中,在GPC内的各个单元中实现较低级别的高速缓存。在一个实施例中,每个SM 1140可以实现一级(“L1”)高速缓存,其中L1高速缓存是专用于特定SM 1140的私有内存,并且从L2高速缓存1104获取数据并存储在每个L1高速缓存中,以用于在SM 1140的功能单元中进行处理。在一个实施例中,L2高速缓存1104耦合到内存接口1106和XBar 920。

[0089] 在一个实施例中,ROP单元1102执行与像素颜色有关的图形光栅操作,例如颜色压缩、像素混合等。在一个实施例中,ROP单元1102结合光栅引擎1125实施深度测试,从光栅引擎1125的剔除引擎接收与像素片段相关联的样本位置的深度。在一个实施例中,针对与片段相关联的样本位置,测试相对于深度缓冲区中的相应深度的深度。在一个实施例中,如果片段通过了针对样本位置的深度测试,则ROP单元1102更新深度缓冲区,并将深度测试的结果发送到光栅引擎1125。将会理解,分区单元1100的数量可以与GPC的数量不同,因此,在一个实施例中,每个ROP单元1102可以耦合到每个GPC。在一个实施例中,ROP单元1102跟踪从不同的GPC接收到的分组,并确定通过Xbar将ROP单元1102生成的结果路由到哪个。

[0090] 图12示出了根据一个实施例的诸如图10的流式多处理器之类的流式多处理器。在一个实施例中,SM 1200包括:指令高速缓存1202;一个或更多个调度器单元1204;寄存器文件1208;一个或更多个处理核心1210;一个或更多个特殊功能单元(“SFU”)1212;一个或更多个加载/存储单元(“LSU”)1214;互连网络1216;共享内存/L1高速缓存1218;及其任何合适的组合。在一个实施例中,工作分配单元分派任务以在PPU的GPC上执行,并且每个任务被分配给GPC内的特定DPC,并且如果该任务与着色器程序相关联,则该任务被分配给SM 1200。在一个实施例中,调度器单元1204从工作分配单元接收任务,并管理针对分配给SM 1200的一个或更多个线程块的指令调度。在一个实施例中,调度器单元1204调度线程块以作为并行线程的线程束执行,其中每个线程块被分配至少一个线程束。在一个实施例中,每个线程束执行线程。在一个实施例中,调度器单元1204管理多个不同的线程块,将线程束分配给不同的线程块,并且然后在每个时钟周期期间将来自多个不同的协作组的指令分派给各个功能单元(例如,核心1210、SFU 1212和LSU 1214)。

[0091] 协作组可以指用于组织通信线程组的编程模型,该编程模型允许开发人员表达正在通信线程的粒度,从而能够更丰富的表达,更有效的并行分解。在一个实施例中,协作启动API支持线程块之间的同步以执行并行算法。在一个实施例中,常规编程模型的应用程序提供了用于同步协作线程的单个简单构造:跨线程块的所有线程的屏障(例如,

syncthreads()函数)。但是,编程人员通常希望以小于线程块粒度的粒度定义线程组,并在定义的组内进行同步,以实现更高的性能、设计灵活性以及以集体的组范围的功能接口的形式来复用软件。协作组使得编程人员能够以子块(即小至单个线程)粒度和多块粒度明确地定义线程组,并能够对协作组中的线程执行诸如同步的集体操作。编程模型支持跨软件边界的清晰组合,因此库和实用程序功能可以在其本地上下文中安全地同步,而不必对收敛进行假设。协作组图元启用了新的协作并行模式,包括生产者-消费者并行,机会主义并行以及整个线程块网格上的全局同步。

[0092] 在一个实施例中,分配单元1206被配置为将指令发送到一个或更多个功能单元,并且调度器单元1204包括两个分派单元1206,这两个分派单元1206使得能够在每个时钟周期期间分派来自相同线程束的两个不同的指令。在一个实施例中,每个调度器单元1204包括单个分派单元1206或附加分派单元1206。

[0093] 在一个实施例中,每个SM 1200包括寄存器文件1208,该寄存器文件1208为SM 1200的功能单元提供一组寄存器。在一个实施例中,寄存器文件1208在每个功能单元之间进行划分,使得每个功能单元被分配寄存器文件1208的专用部分。在一个实施例中,寄存器文件1208在正由SM 1200执行的不同的线程束之间进行划分,并且寄存器文件1208为连接到功能单元的数据路径的操作数提供临时存储。在一个实施例中,每个SM 1200包括多个L个处理核心1210。在一个实施例中,SM 1200包括大量(例如,128个或更多个)不同的处理核心1210。在一个实施例中,每个核心1210包括全管线、单精度、双精度和/或混合精度处理单元,其包括浮点算术逻辑单元和整数算术逻辑单元。在一个实施例中,浮点算术逻辑单元实现用于浮点算术的IEEE 754-2008标准。在一个实施例中,核心1210包括64个单精度(32位)浮点核心、64个整数核心、32个双精度(64位)浮点核心和8个张量核心。

[0094] 根据一个实施例,张量核心被配置为执行矩阵运算。在一个实施例中,一个或更多个张量核心被包括在核心1210中。在一个实施例中,张量核心被配置为执行深度学习矩阵算术,例如用于神经网络训练和推理的卷积运算。在一个实施例中,每个张量核心对4×4矩阵运算,并且执行矩阵乘法和累加运算D=A×B+C,其中A、B、C和D是4×4矩阵。

[0095] 在一个实施例中,矩阵乘法输入A和B是16位浮点矩阵,并且累加矩阵C和D是16位浮点或32位浮点矩阵。在一个实施例中,张量核心对16位浮点输入数据进行32位浮点累加运算。在一个实施例中,该16位浮点乘法需要64次运算,并产生全精度乘积,然后使用32位浮点加法将其与其他中间乘积进行累加,以进行4×4×4矩阵乘法。在一个实施例中,张量核心用于执行由这些较小的元素构成的更大的二维或更高维度的矩阵运算。在一个实施例中,诸如CUDA 9C++API之类的API公开专门的矩阵加载、矩阵乘法和累加以及矩阵存储操作,以有效地使用来自CUDA-C++程序的张量核心。在一个实施例中,在CUDA级别上,线程束级(warp-level)接口假定跨越线程束的所有32个线程的16×16大小的矩阵。

[0096] 在一个实施例中,每个SM 1200包括执行特定函数(例如,属性评估、倒数平方根等)的M个SFU 1212。在一个实施例中,SFU 1212包括被配置为遍历层次树数据结构的树遍历单元。在一个实施例中,SFU 1212包括被配置为执行纹理贴图过滤操作的纹理单元。在一个实施例中,纹理单元被配置为从内存加载纹理贴图(例如,纹理像素的2D阵列),并且对纹理贴图进行采样以产生采样的纹理值,以用于由SM 1200执行的着色器程序中。在一个实施例中,纹理贴图被存储在共享内存/L1高速缓存中。根据一个实施例,纹理单元实现纹理操

作,例如使用mip贴图(例如,细节水平不同的纹理贴图)进行过滤操作。在一个实施例中,每个SM 1200包括两个纹理单元。

[0097] 在一个实施例中,每个SM 1200包括实现共享内存/L1高速缓存1106和寄存器文件1208之间的加载和存储操作的N个LSU 1154。在一个实施例中,每个SM 1200包括互连网络1216,其将每个功能单元连接到寄存器文件1208并且将LSU 1214连接到寄存器文件1208、共享内存/L1高速缓存1218。在一个实施例中,互连网络1216是交叉开关,该交叉开关可被配置为将任何功能单元连接到寄存器文件1208中的任何寄存器,并将LSU1214连接到寄存器文件和共享内存/L1高速缓存1218中的内存位置。

[0098] 在一个实施例中,共享内存/L1高速缓存1218是片上内存的阵列,其允许SM 1200与图元引擎之间以及SM 1200中的线程之间的数据存储和通信。在一个实施例中,共享内存/L1高速缓存1218包括128KB的存储容量,并且位于从SM 1200到分区单元的路径中。在一个实施例中,共享内存/L1高速缓存1218用于高速缓存读取和写入。共享内存/L1高速缓存1218、L2高速缓存和内存中的一个或更多个是后备存储。

[0099] 在一个实施例中,将数据高速缓存和共享内存功能组合到单个内存块中,为两种类型的内存访问提供了改进的性能。在一个实施例中,该容量被不使用共享内存的程序用作或不用作高速缓存,例如如果共享内存被配置为使用一半容量,则纹理和加载/存储操作可以使用剩余的容量。根据一个实施例,在共享内存/L1高速缓存1218内的集成使共享内存/L1高速缓存1218能够用作用于流传输数据的高吞吐量管道,同时提供对频繁重用的数据的高带宽和低延迟访问。当被配置用于通用并行计算时,与图形处理相比,可以使用更简单的配置。在一个实施例中,固定功能图形处理单元被绕过,从而创建了更简单的编程模型。在一个实施例中,在通用并行计算配置中,工作分配单元将线程的块直接分配和分发给DPC。根据一个实施例,块中的线程执行相同的程序,在计算中使用唯一的线程ID来确保每个线程生成唯一的结果,使用SM 1200执行程序并执行计算,使用共享内存/L1高速缓存1218在线程之间进行通信,以及使用LSU 1214通过共享内存/L1高速缓存1218和内存分区单元来读取和写入全局内存。在一个实施例中,当被配置用于通用并行计算时,SM 1200写入命令,调度器单元可以使用该命令来在DPC上启动新工作。

[0100] 在一个实施例中,PPU被包括在或耦合到台式计算机、膝上型计算机、平板计算机、服务器、超级计算机、智能电话(例如,无线、手持设备)、个人数字助理(“PDA”)、数码相机、车辆、头戴式显示器、手持式电子设备等。在一个实施例中,PPU体现在单个半导体衬底上。在一个实施例中,PPU与一个或更多个其他器件(例如,附加的PPU、内存、精简指令集计算机(“RISC”)CPU、内存管理单元(“MMU”)、数模转换器(“DAC”)等)一起被包括在片上系统(“SoC”)中。

[0101] 在一个实施例中,PPU可以被包括在包括一个或更多个内存设备的图形卡上。图形卡可以被配置为与台式计算机主板上的PCIe插槽接合。在另一个实施例中,PPU可以是主板的芯片组中包括的集成图形处理单元(“iGPU”)。

[0102] 图13示出了根据一个实施例的可以在其中实现各种架构和/或功能的计算机系统1300。在一个实施例中,计算机系统1300被配置为实现贯穿本公开描述的各种过程和方法。

[0103] 在一个实施例中,计算机系统1300包括至少一个中央处理单元1302,其连接到使用任何适当协议实现的通信总线1310,所述适当协议例如,PCI(外围组件互连)、PCI-

Express、AGP(加速图形端口)、超传输(HyperTransport)或任何其他总线或一个或更多个点对点通信协议。在一个实施例中,计算机系统1300包括主内存1304和控制逻辑(例如,实现为硬件、软件或其组合),并且数据被存储在主内存1304中,其可以采取随机存取存储器(“RAM”)的形式。在一个实施例中,网络接口子系统1322提供到其他计算设备和网络的接口,用于从计算机系统1300的其他系统接收数据以及将数据发送到其他系统。

[0104] 在一个实施例中,计算机系统1300包括输入设备1308、并行处理系统1312和显示设备1306,显示设备1306可以使用常规的CRT(阴极射线管)、LCD(液晶显示器)、LED(发光二极管)、等离子显示器或其他合适的显示技术实现。在一个实施例中,用户输入从诸如键盘、鼠标、触摸板、麦克风等的输入设备1308接收。在一个实施例中,每个前述模块可以位于单个半导体平台上以形成处理系统。

[0105] 在本说明书中,单个半导体平台可以指唯一的单一的基于半导体的集成电路或芯片。应当注意,术语“单个半导体平台”还可以指具有增加的连通性的多芯片模块,其模拟芯片上的操作,并且相对于利用传统的中央处理单元(“CPU”)和总线实现方式进行了实质性的改进。当然,根据用户的需求,各种模块也可以分开放置或以半导体平台的各种组合放置。

[0106] 在一个实施例中,以机器可读的可执行代码或计算机控制逻辑算法的形式的计算机程序被存储在主内存1304和/或辅助存储器中。根据一个实施例,计算机程序如果由一个或更多个处理器执行,则使系统1300能够执行各种功能。内存1304、存储器和/或任何其他存储是计算机可读介质的可能示例。辅助存储可以指任何合适的存储设备或系统,例如硬盘驱动器和/或可移除存储驱动器、代表软盘驱动器、磁带驱动器、光盘驱动器、数字多功能盘(“DVD”)驱动器、记录设备、通用串行总线(“USB”)闪存。

[0107] 在一个实施例中,各个先前附图的架构和/或功能在以下项的上下文中实现:中央处理器1302;并行处理系统1312;具有中央处理器1302和并行处理系统1312两者的能力的至少一部分的集成电路;芯片组(例如,被设计为作为执行相关功能的单元工作并出售的集成电路组等);以及集成电路的任何合适组合。

[0108] 在一个实施例中,各个先前附图的架构和/或功能在以下项的上下文中实现:通用计算机系统、电路板系统、专用于娱乐目的的游戏机系统、专用系统等。在一个实施例中,计算机系统1300可以采取台式计算机、膝上型计算机、平板计算机、服务器、超级计算机、智能电话(例如,无线手持设备)、个人数字助理(“PDA”)、数码相机、车辆、头戴式显示器、手持式电子设备、移动电话设备、电视、工作站、游戏机、嵌入式系统和/或任何其他类型的逻辑的形式。

[0109] 在一个实施例中,并行处理系统1312包括多个PPU 1314和相关联的内存1316。在一个实施例中,PPU经由互连1318和交换机1320或多路复用器连接到主机处理器或其他外围设备。在一个实施例中,并行处理系统1312在可并行化的PPU 1314上分发计算任务—例如作为跨多个GPU线程块的计算任务分发的一部分。在一个实施例中,尽管在某些或全部PPU

[0110] 1314之间共享和访问内存(例如,用于读取和/或写入访问),但是这种共享内存可能会导致相对于使用本地内存和驻留在PPU上的寄存器的性能损失。在一个实施例中,通过使用诸如__syncthreads()之类的命令来同步PPU 1314的操作,该命令要求块中的所有线

程(例如,跨多个PPU 1314执行的)在继续之前到达代码的某个执行点。

[0111] 因此,说明书和附图应被认为是说明性的而不是限制性的。但是,很明显,在不脱离权利要求书所阐明的本发明的更广泛精神和范围的前提下,可以对其做出各种修改和改变。

[0112] 其他变型在本公开的精神内。因此,尽管所公开的技术易于进行各种修改和替代构造,但是在附图中示出了某些说明性实施例并且已经在上面进行了详细描述。然而,应当理解,无意将本发明限制为所公开的一种或更多种特定形式,而是相反,意图涵盖落入本发明的精神和范围内的所有修改、替代构造以及等同形式,如所附权利要求所定义的本发明。

[0113] 除非本文另外指出或与上下文明显矛盾,否则在描述所公开的实施例的上下文中(特别是在所附权利要求的上下文中),术语“一”和“一个”和“该”以及类似词的使用应当被解释为涵盖单数形式和复数形式。除非另有说明,否则术语“包含”、“具有”、“包括”和“含有”应当解释为开放式术语(即,意思是“包括但不限于”)。术语“连接的”在未经修改并且指的是物理连接时,应理解为部分或全部包含在、连接到或连接在一起,即使有任何介入。除非在此另外指出,否则本文中对数值范围的记载仅旨在用作分别指代落入该范围内的每个单独的值的简写方法,并且每个单独的值都被并入说明书中,就好像其在本文中被单独叙述一样。除非上下文另外指出或上下文相矛盾,否则术语“集”(例如“项目集”)或“子集”的使用应解释为包括一个或更多个成员的非空集合。此外,除非上下文另外指出或上下文相矛盾,否则相应集的术语“子集”不一定表示相应集的适当子集,而是子集和相应集可以相等。

[0114] 除非以其他方式明确指出或与上下文明显矛盾,否则诸如“A、B和C中的至少一个”或“A、B和C中的至少一个”形式的短语的组合语言在上下文中以其他方式理解为通常用于表示项目、术语等,其可以是A或B或C,或A和B以及C的集合的任何非空子集。例如,在在具有三个成员的集合的说明性示例中,连词短语“A、B、和C中的至少一个”和“A、B和C中的至少一个”是指以下任意集合:{A},{B},{C},{A,B},{A,C},{B,C},{A,B,C}。因此,这种连词语言通常不意图暗示某些实施例要求A中的至少一个、B中的至少一个和C中的至少一个中的每个都存在。另外,除非另有说明或与上下文矛盾,否则术语“多个”表示复数的状态(例如,“多个项目”表示多个项目)。多个中项目的数量至少是两个,但是当明确地或通过上下文指示时可以更多。此外,除非另有说明或从上下文可以清楚地看出,否则短语“基于”是指“至少部分地基于”而不是“仅基于”。

[0115] 除非本文另外指出或与上下文明显矛盾,否则本文描述的过程的操作可以任何合适的顺序执行。在一个实施例中,诸如本文所述的那些过程(或其变形和/或其组合)的过程在配置有可执行指令的一个或更多个计算机系统的控制下执行,并且被实现为通过硬件或其组合在一个或更多个处理器上共同执行的代码(例如,可执行指令、一个或更多个计算机程序或一个或更多个应用程序)。在一个实施例中,例如代码以计算机程序的形式存储在计算机可读存储介质上,该计算机程序包括可由一个或更多个处理器执行的多个指令。在一个实施例中,计算机可读存储介质是不包括暂时性信号(例如,传播的瞬态电或电磁传输)但包括暂时性信号的收发器中的非暂时性数据存储电路(例如,缓冲区、高速缓存和队列)的非暂时性计算机可读存储介质。在一个实施例中,代码(例如,可执行代码或源代码)被存储在其上存储有可执行指令的一个或更多个非暂时性计算机可读存储介质(或用于存储可

执行指令的其他内存)的集合上,该可执行指令在由计算机系统的一个或更多个处理器执行(即,由于被执行)时,导致该计算机系统执行本文所述的操作。在一个实施例中,该非暂时性计算机可读存储介质的集合包括多个非暂时性计算机可读存储介质,并且多个非暂时性计算机可读存储介质中的一个或更多个个体非暂时性存储介质缺少所有代码,而多个非暂时性计算机可读存储介质共同存储所有代码。在一个实施例中,可执行指令被执行,使得不同的指令被不同的处理器执行—例如非暂时性计算机可读存储介质存储指令,并且主CPU执行某些指令,而图形处理器单元执行其他指令。在一个实施例中,计算机系统的不同组件具有单独的处理器,并且不同的处理器执行指令的不同子集。

[0116] 在一实施例中,相应地,计算机系统被配置为实现单独地或共同地执行本文所述的过程的操作的一个或更多个服务,并且这样的计算机系统被配置有能够使操作执行的适用的硬件和/或软件。此外,实现本公开的实施例的计算机系统是单个设备,并且在另一个实施例中,其是包括以不同方式操作的多个设备的分布式计算机系统,使得该分布式计算机系统执行本文所述的操作,并且使得单个设备无法执行所有操作。

[0117] 除非另有要求,否则本文提供的任何和所有示例或示例性语言(例如,“例如”)的使用仅旨在更好地阐明本发明的实施例,并且不构成对本发明范围的限制。说明书中的任何语言都不应解释为指示任何未要求保护的要素对于实践本发明必不可少。

[0118] 本文描述了本公开的实施例,包括发明人已知的用于实施本发明的最佳模式。通过阅读前述说明,那些实施例的变型对于本领域普通技术人员而言将变得显而易见。发明人期望熟练的技术人员适当地采用这样的变型,并且发明人希望以不同于本文具体描述的方式实践本公开的实施例。因此,本公开的范围包括适用法律所允许的所附权利要求中记载的主题的所有修改和等同物。此外,除非本文另外指出或与上下文明显矛盾,否则上述元素在其所有可能的变化中的任何组合均被本公开的范围涵盖。

[0119] 本文引用的所有参考文献,包括出版物、专利申请和专利,均以引用的方式并入本文,其程度如同每个参考文献被单独且具体地指示为以引用的方式并入本文中一样。

[0120] 在说明书和权利要求书中,可以使用术语“耦合”和“连接”及其派生词。应当理解,这些术语可能不旨在作为彼此的同义词。而是,在特定示例中,“连接”或“耦合”可用于指示两个或更多个元素彼此直接或间接物理或电接触。“耦合”也可能意味着两个或多个元素彼此不直接接触,但仍彼此协作或交互。

[0121] 除非另外特别说明,否则可以理解,在整个说明书中,诸如“处理”、“计算”、“运算”、“确定”等的术语是指计算机或计算系统或类似的电子计算设备的动作和/或过程,该类似的电子计算设备将计算系统的寄存器和/或内存中表示为物理量(例如,电子量)的数据操纵和/或转换为类似地表示为计算系统的内存、寄存器或其他此类信息存储、传输或显示设备中的物理量的其他数据。

[0122] 以类似的方式,术语“处理器”可以指处理来自寄存器和/或内存的电子数据并将该电子数据转换成可以存储在寄存器和/或内存中的其他电子数据的任何设备或设备的一部分。作为非限制性示例,“处理器”可以是中央处理单元(CPU)或图形处理单元(GPU)。“计算平台”可以包括一个或更多个处理器。如本文所使用的,“软件”过程可以包括例如随时间执行工作的软件和/或硬件实体,诸如任务、线程和智能代理。而且,每个过程可以指代连续地或间歇地、顺序地或并行地执行指令的多个过程。术语“系统”和“方法”在本文中可互换

地使用，只要该系统可以体现一种或更多种方法并且该方法可以被认为是系统。

[0123] 在本文件中，可以参考获得、获取、接收模拟或数字数据或将其输入子系统、计算机系统或计算机实现的机器中。可以以多种方式来完成获得、获取、接收或输入模拟和数字数据的过程，例如通过接收作为函数调用或对应用程序编程接口的调用的参数的数据。在一些实现方式中，获得、获取、接收或输入模拟或数字数据的过程可以通过经由串行或并行接口传输数据来完成。在另一个实现方式中，可以通过经由计算机网络将数据从提供实体传输到获取实体来完成获得、获取、接收或输入模拟或数字数据的过程。也可以参考提供、输出、传送、发送或呈现模拟或数字数据。在各种示例中，提供、输出、传送、发送或呈现模拟或数字数据的过程可以通过将数据作为函数调用的输入或输出参数、应用程序编程接口或进程间通信机制的参数进行传输来完成。

[0124] 尽管上面的讨论阐述了所描述的技术的示例实现，但是其他架构可以用于实现所描述的功能，并且意图在本公开的范围内。此外，尽管出于讨论目的在上面定义了具体的职责分配，但是根据情况，各种功能和职责可能以不同的方式分配和划分。

[0125] 此外，尽管已经以特定于结构特征和/或方法动作的语言描述了主题，但应理解，所附权利要求书中定义的主题不必限于所描述的特定特征或动作。而是，特定特征和动作被公开为实现权利要求的示例性形式。

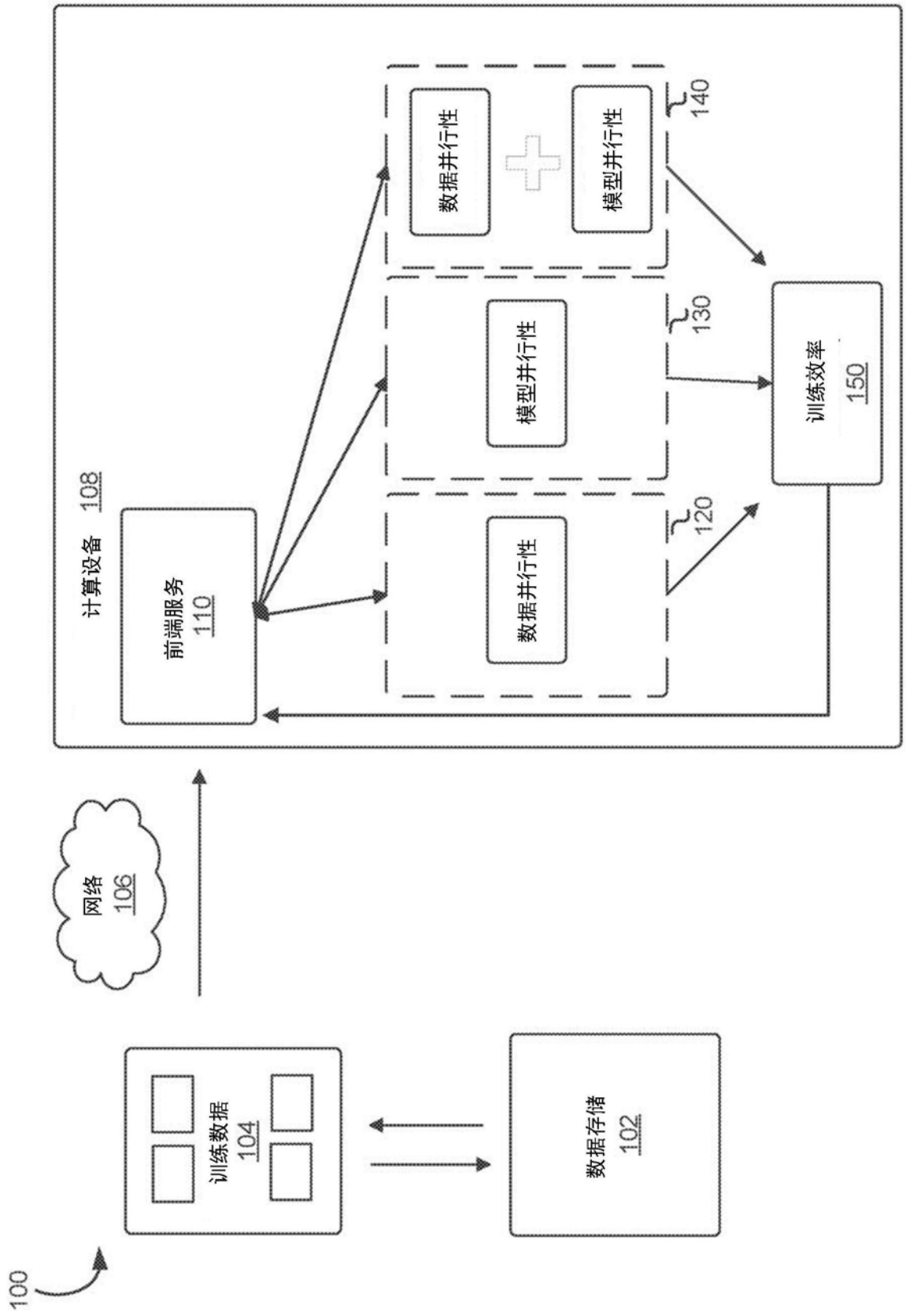

100
训练数据
104
数据存储
102
网络
106
计算设备
108
前端服务
110
数据并行性
模型并行性
120
模型并行性
数据并行性
130
140
训练效率
150

图1

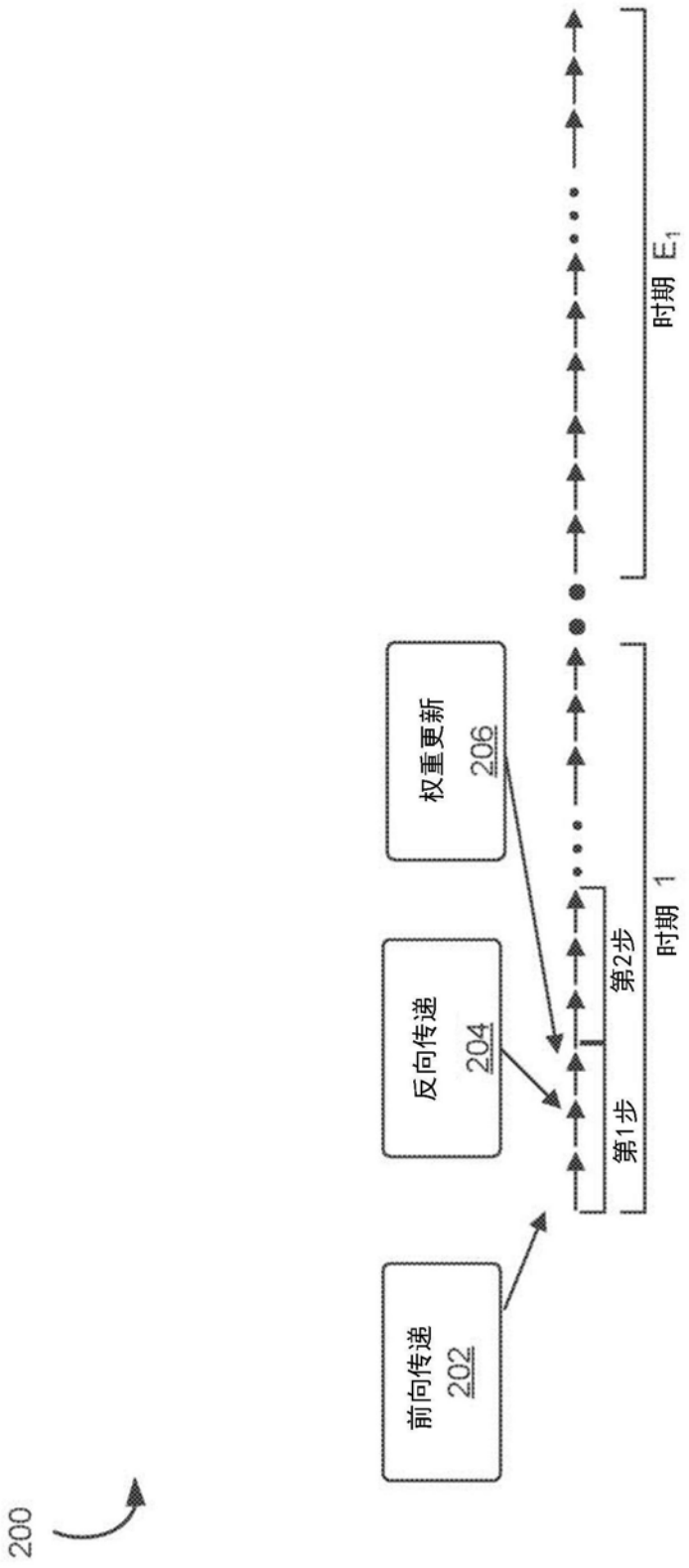


图2

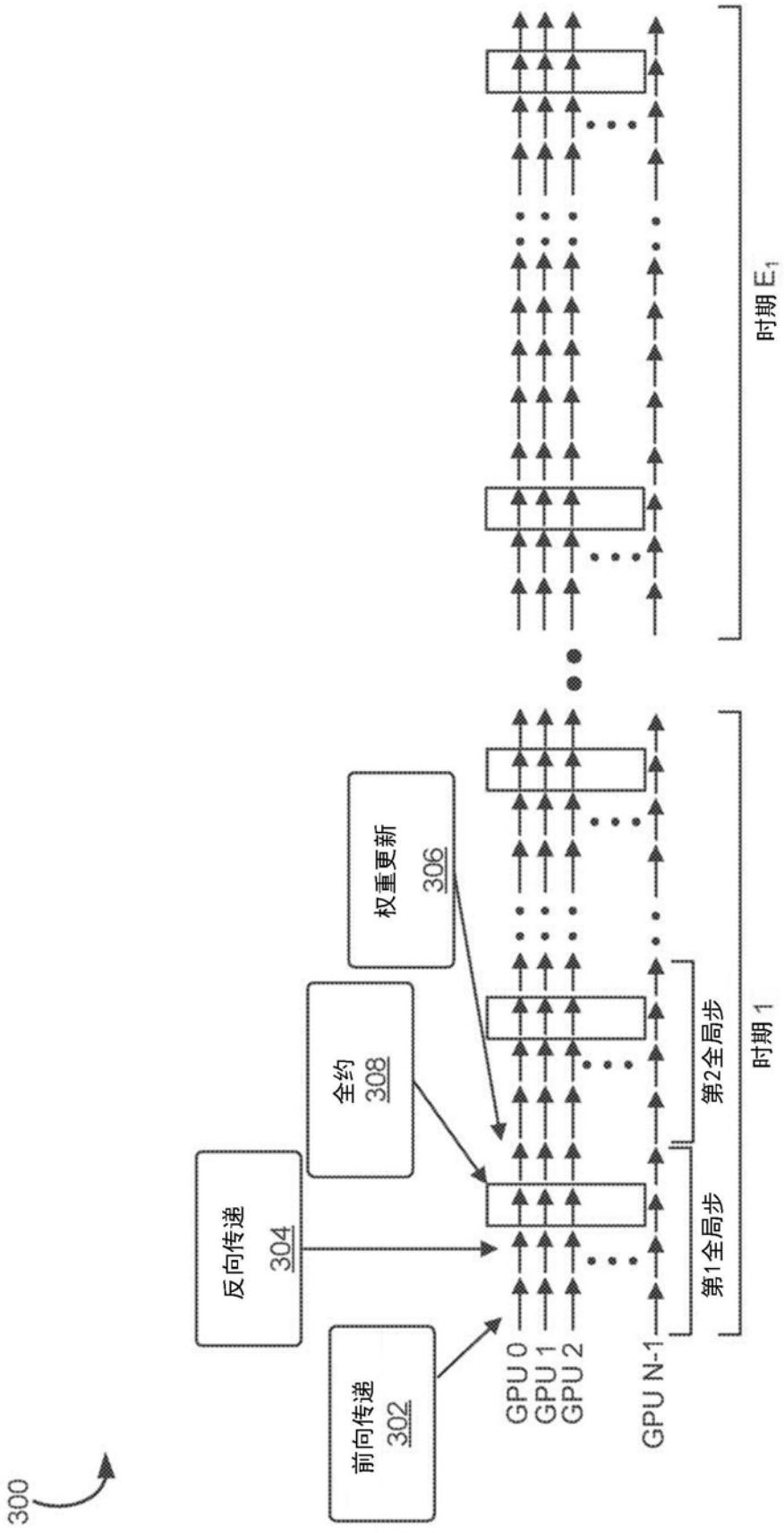

300
前向传递
302
反向传递
304
全约
308
权重更新
306
GPU 0
GPU 1
GPU 2
GPU N-1
第1全局步
第2全局步
时期 1
时期 E_1

图3

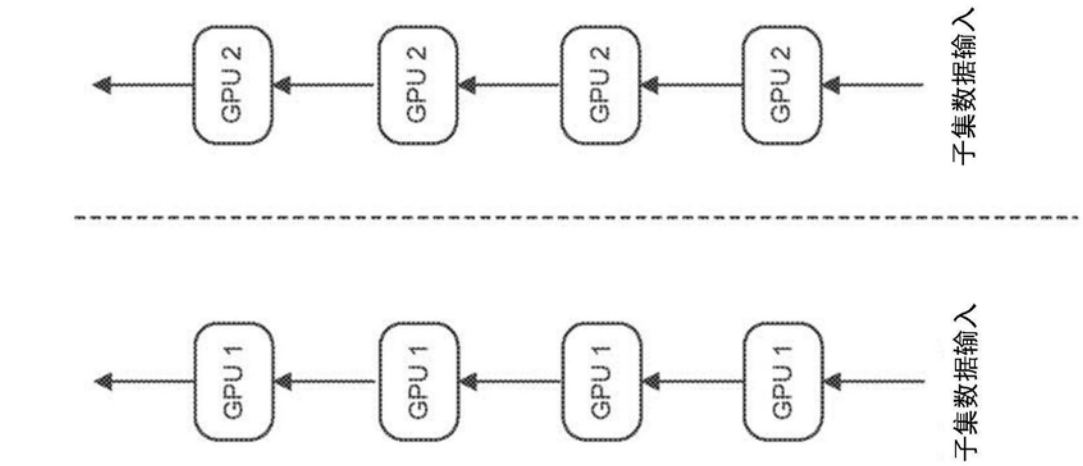


图4

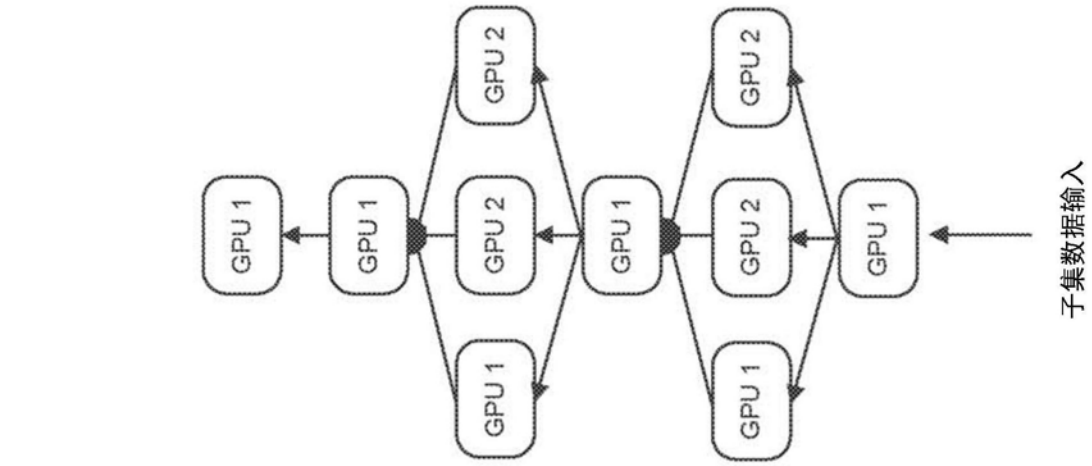


图5

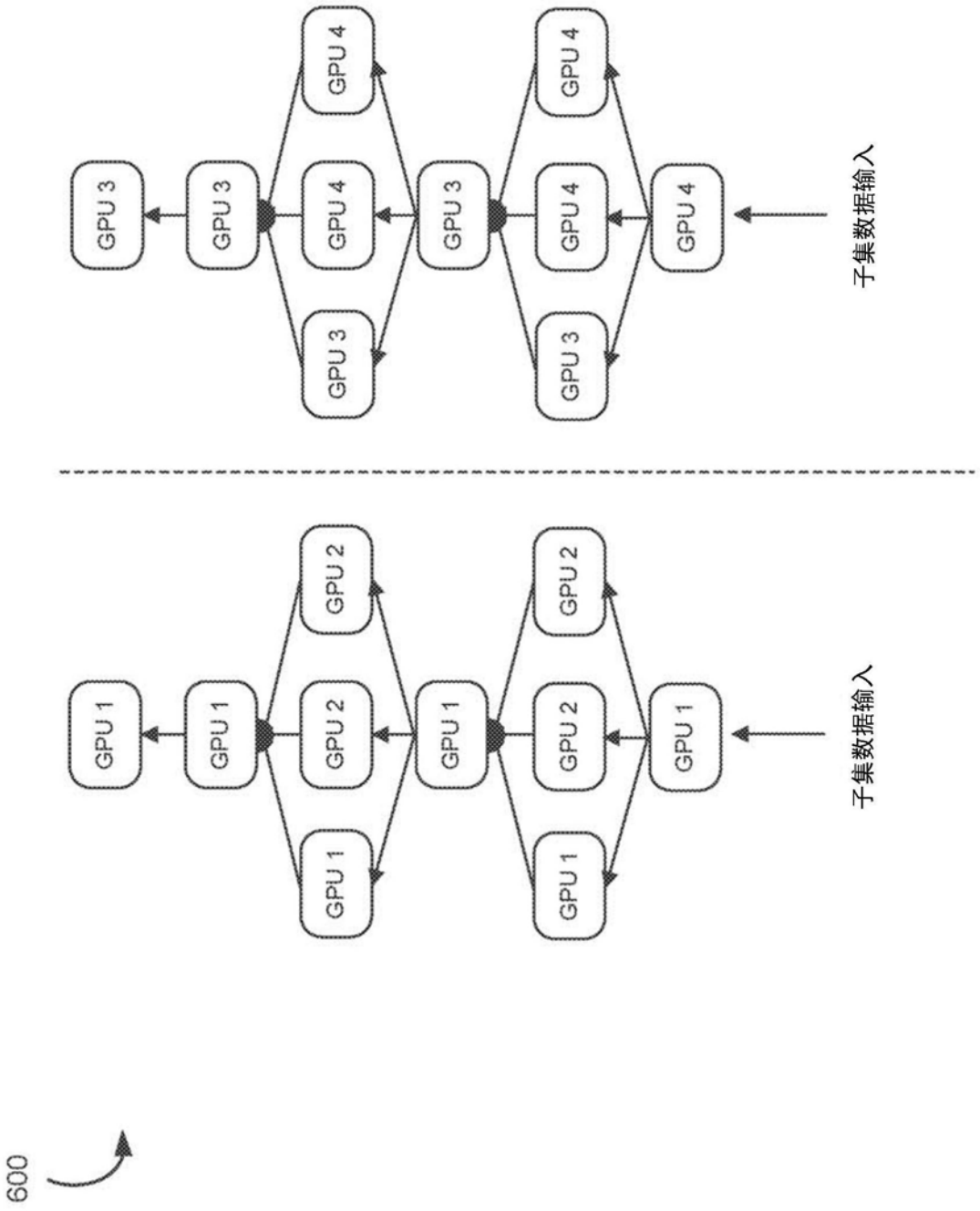

GPU 3
GPU 3
GPU 4
GPU 4
GPU 3
GPU 3
GPU 4
GPU 4
GPU 4
子集数据输入
GPU 1
GPU 1
GPU 2
GPU 2
GPU 1
GPU 1
GPU 2
GPU 2
GPU 1
子集数据输入
600

图6

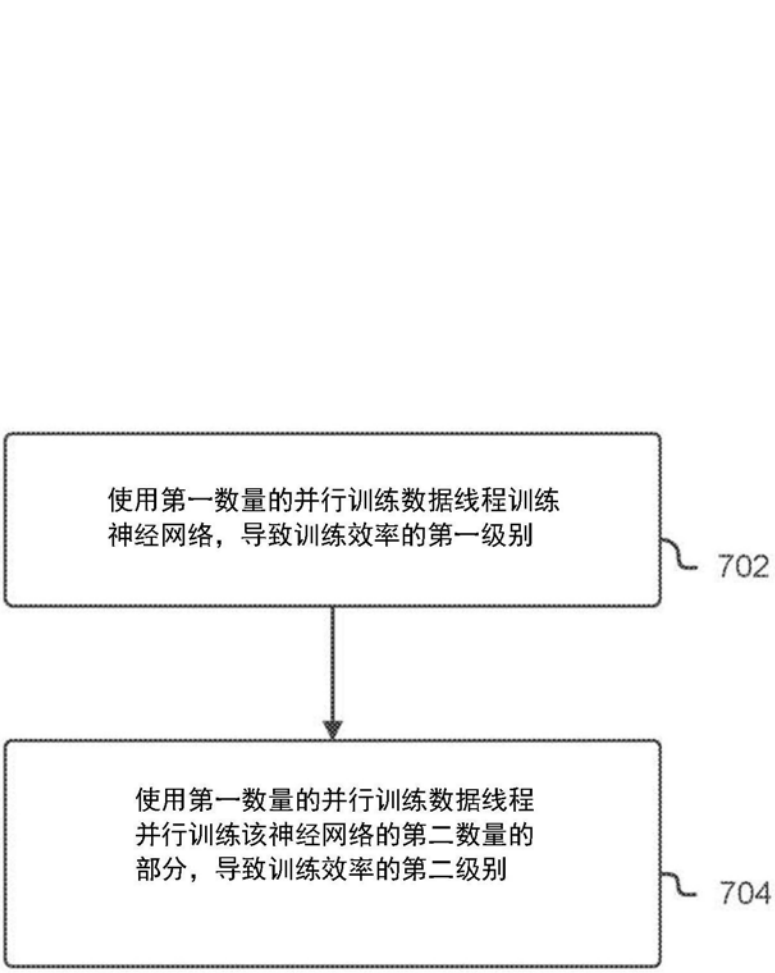

700
使用第一数量的并行训练数据线程训练神经网络，导致训练效率的第一级别
702
使用第一数量的并行训练数据线程并行训练该神经网络的第二数量的部分，导致训练效率的第二级别
704

图7

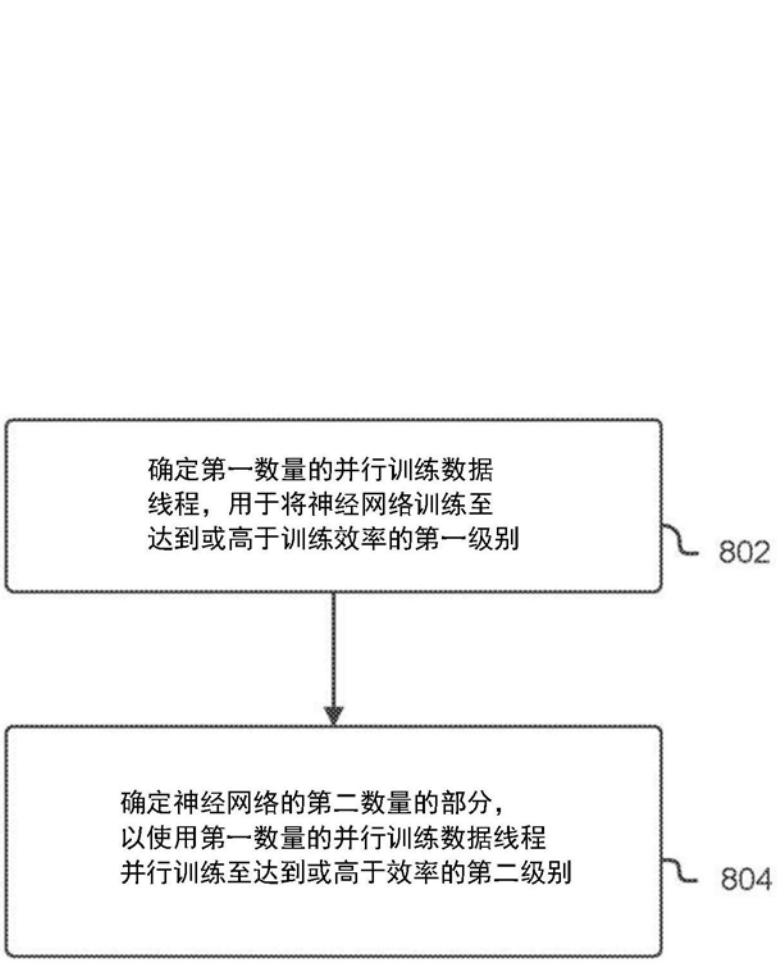

800
确定第一数量的并行训练数据线程，用于将神经网络训练至达到或高于训练效率的第一级别
802
确定神经网络的第二数量的部分，以使用第一数量的并行训练数据线程并行训练至达到或高于效率的第二级别
804

图8

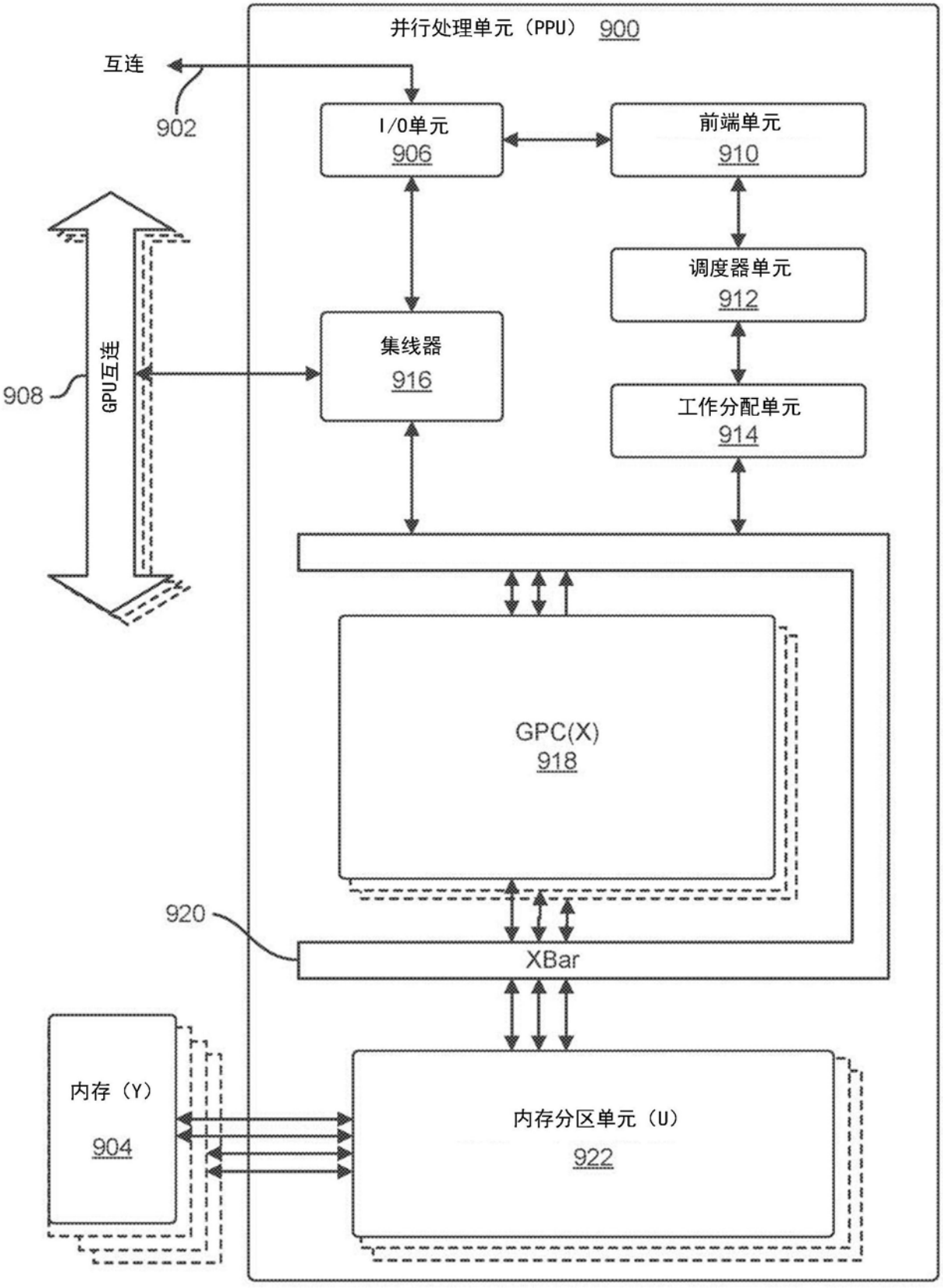

并行处理单元（PPU） 900
互连
902
I/O单元
906
前端单元
910
调度器单元
912
集线器
916
工作分配单元
914
908
GPU互连
GPC(X)
918
920
XBar
内存（Y）
904
内存分区单元（U）
922

图9

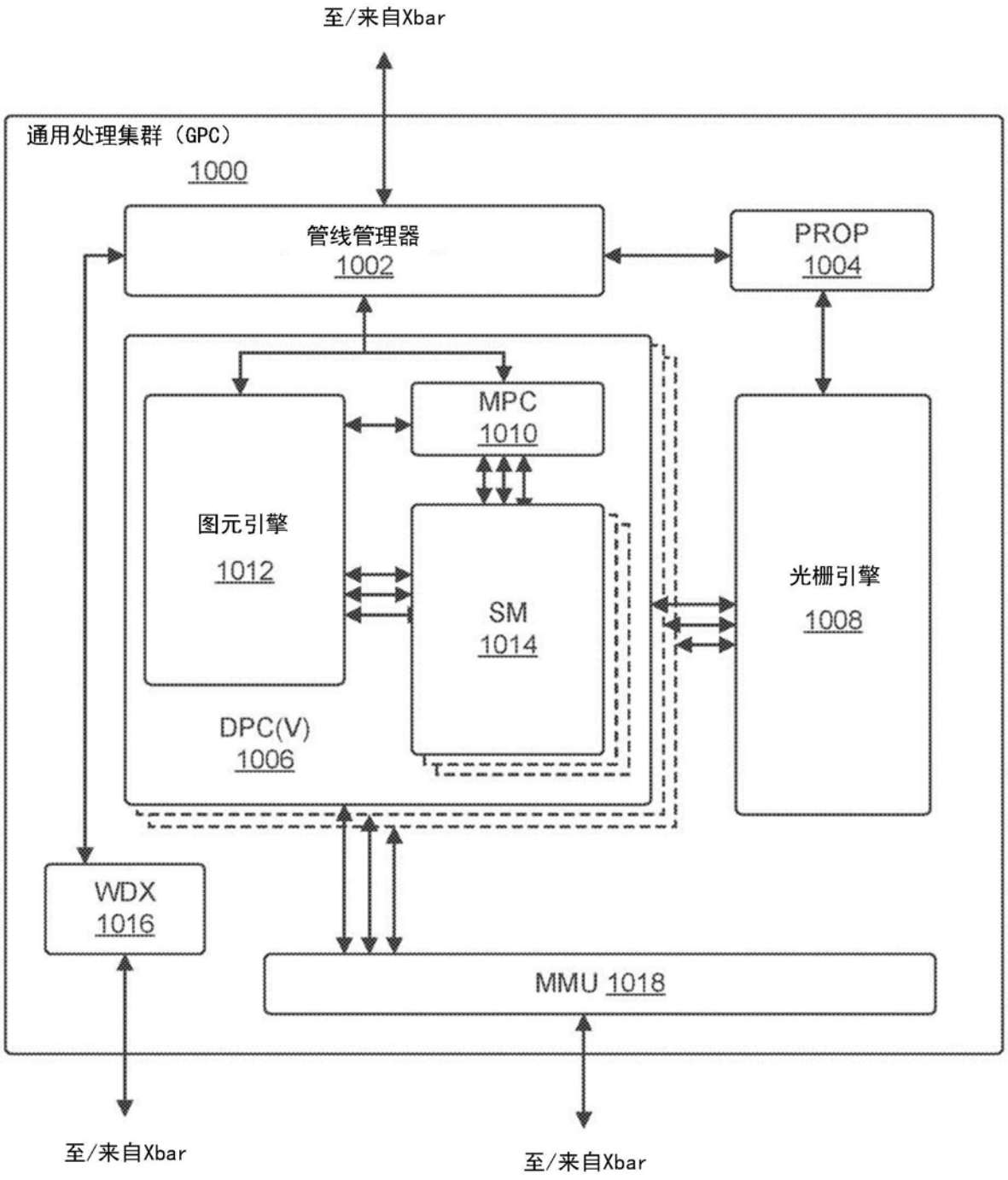

至/来自Xbar
通用处理集群（GPC）
1000
管线管理器
1002
PROP
1004
MPC
1010
图元引擎
1012
SM
1014
光栅引擎
1008
DPC(V)
1006
WDX
1016
MMU 1018
至/来自Xbar
至/来自Xbar

图10

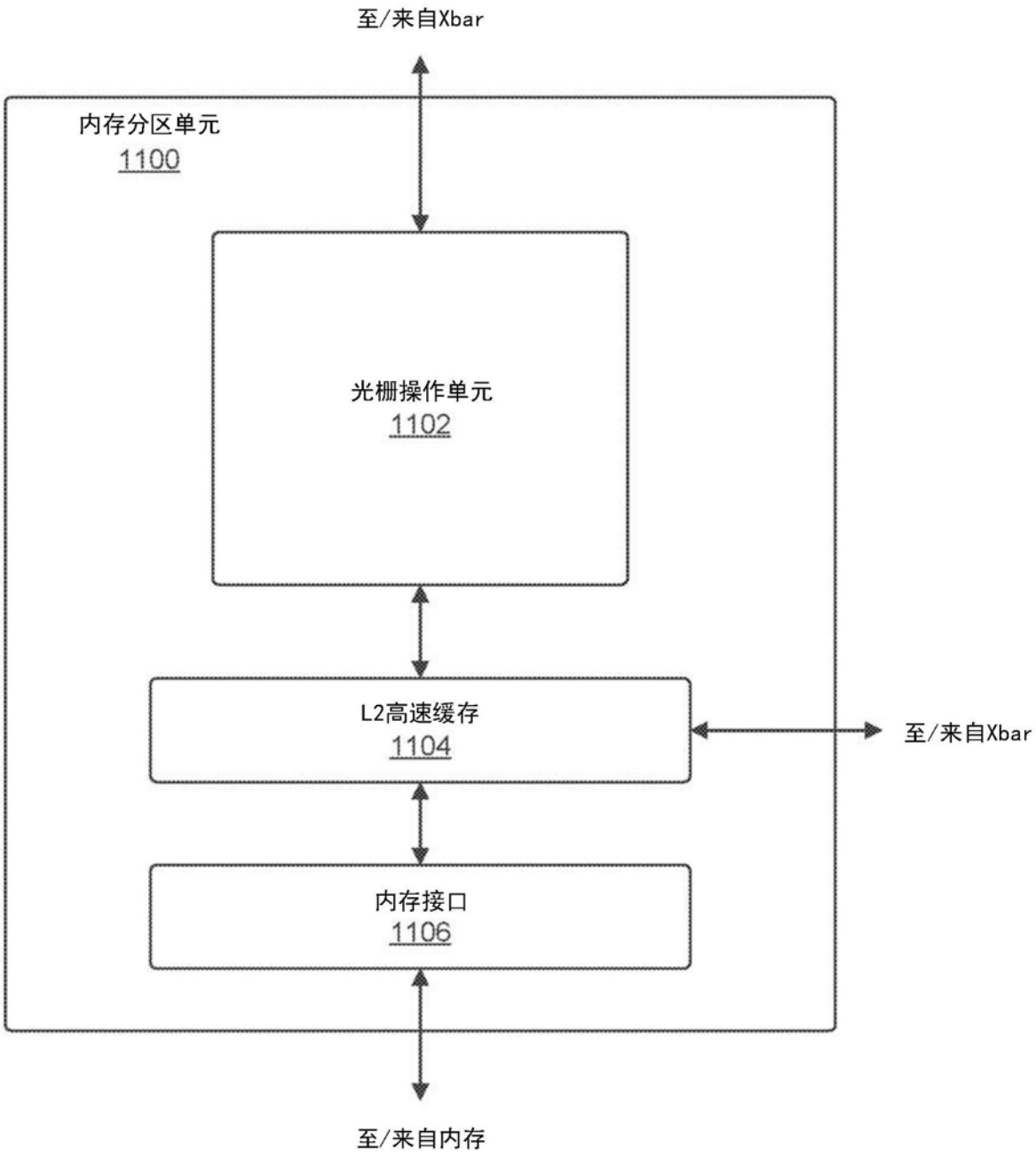

至/来自Xbar
内存分区单元
1100
光栅操作单元
1102
L2高速缓存
1104
至/来自Xbar
内存接口
1106
至/来自内存

图11

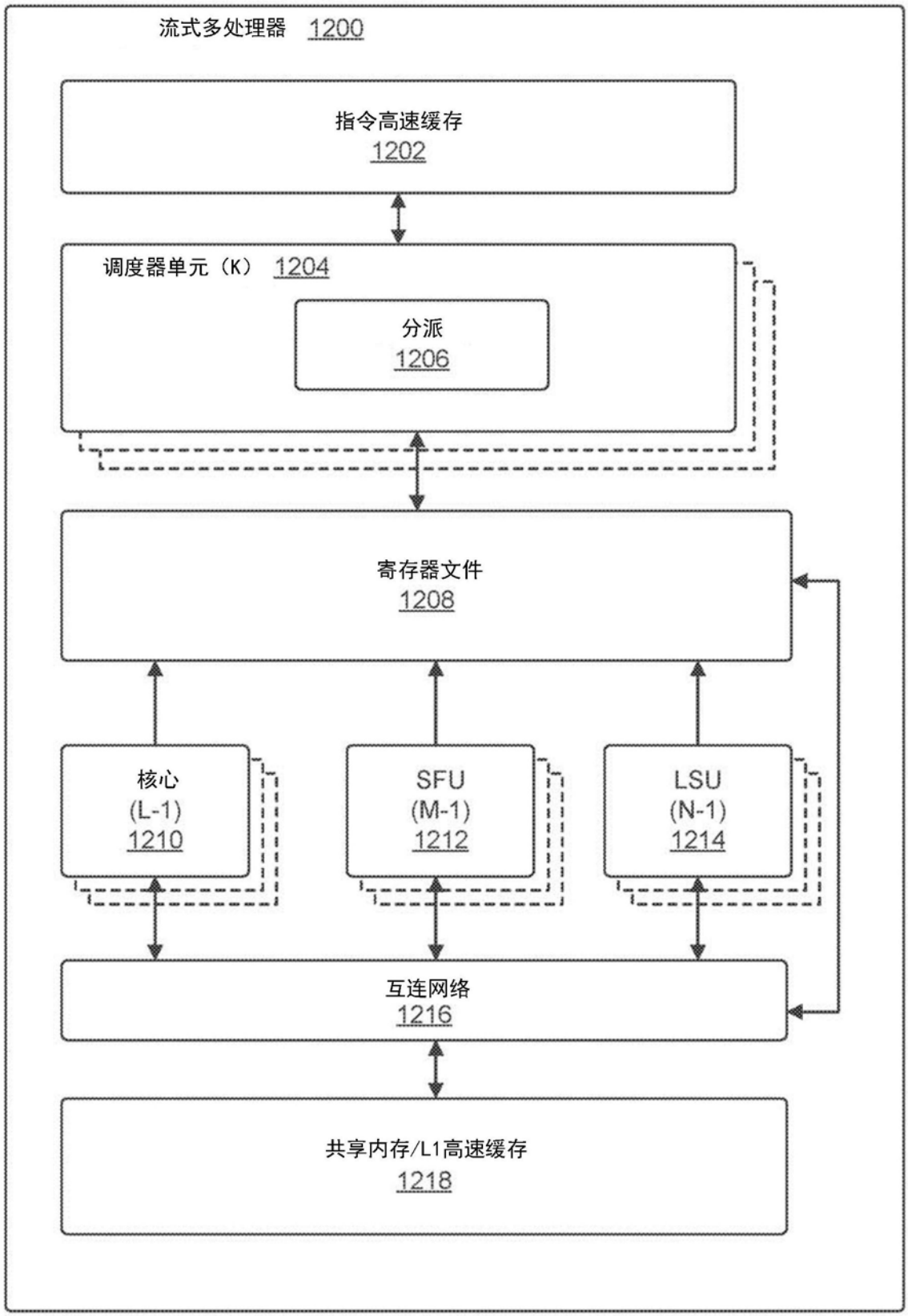

流式多处理器 1200
指令高速缓存
1202
调度器单元（K） 1204
分派
1206
寄存器文件
1208
核心
(L-1)
1210
SFU
(M-1)
1212
LSU
(N-1)
1214
互连网络
1216
共享内存/L1高速缓存
1218

图12

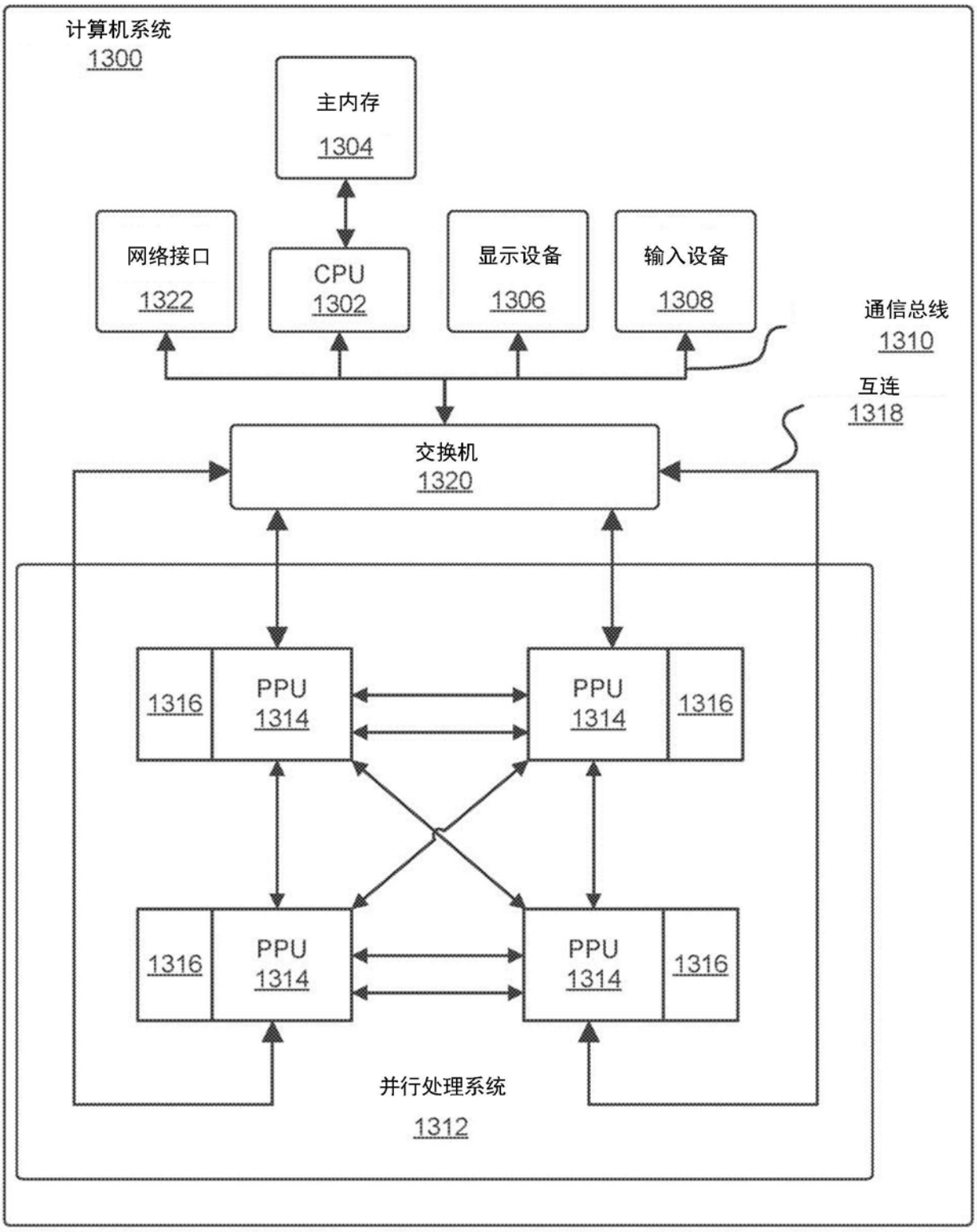

计算机系统
1300
主内存
1304
网络接口
1322
CPU
1302
显示设备
1306
输入设备
1308
通信总线
1310
互连
1318
交换机
1320
1316
PPU
1314
PPU
1314
1316
1316
PPU
1314
PPU
1314
1316
并行处理系统
1312

图13