



(12)发明专利申请

(10)申请公布号 CN 111078415 A

(43)申请公布日 2020.04.28

(21)申请号 201911319986.8

(22)申请日 2019.12.19

(71)申请人 北京奇艺世纪科技有限公司
地址 100080 北京市海淀区北一街2号鸿城
拓展大厦10、11层

(72)发明人 张吉

(74)专利代理机构 北京润泽恒知识产权代理有
限公司 11319

代理人 莎日娜

(51)Int.Cl.
G06F 9/50(2006.01)

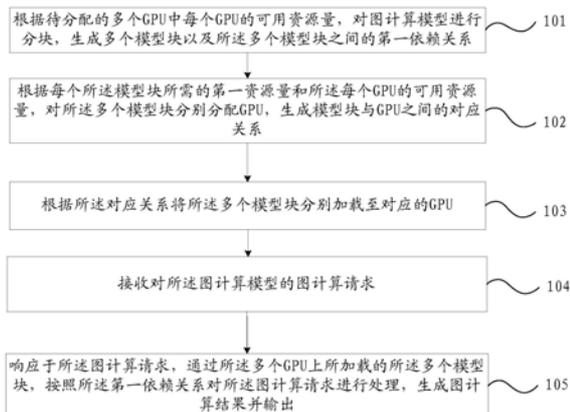
权利要求书3页 说明书12页 附图3页

(54)发明名称

数据处理方法、装置、服务器及计算机可读
存储介质

(57)摘要

本发明实施例提供了一种数据处理方法、装
置、服务器、计算机可读存储介质,该方法包括:
根据待分配的多个GPU中每个GPU的可用资源量,
对图计算模型进行分块,生成多个模型块以及多
个模型块之间的第一依赖关系;根据每个模型块
所需的第一资源量和每个GPU的可用资源量,对
多个模型块分别分配GPU,生成模型块与GPU之
间的对应关系;根据对应关系将多个模型块分别
加载至对应的GPU;接收对图计算模型的图计算请
求;响应于图计算请求,通过多个GPU上所加载的
多个模型块,按照第一依赖关系对图计算请求进
行处理,生成图计算结果并输出。本发明实现了
对大型图计算模型的推理计算。



1. 一种数据处理方法,其特征在于,包括:

根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系;

根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,其中,所述对应关系中模型块所需的第一资源量小于或等于该模型块所对应的GPU的可用资源量;

根据所述对应关系将所述多个模型块分别加载至对应的GPU;

接收对所述图计算模型的图计算请求;

响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出。

2. 根据权利要求1所述的方法,其特征在于,所述根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系,包括:

获取图计算模型中多个第一节点之间的第二依赖关系;

获取所述图计算模型所需的第二资源量;

根据所述第二资源量、所述多个GPU的数量以及每个GPU的可用资源量,按照所述第二依赖关系对所述图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系。

3. 根据权利要求2所述的方法,其特征在于,所述根据所述第二资源量、所述多个GPU的数量以及每个GPU的可用资源量,按照所述第二依赖关系对所述图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系,包括:

计算所述第二资源量和所述多个GPU的数量的比值,生成第三资源量,其中,所述第三资源量小于或等于所述每个GPU的可用资源量;

将所述图计算模型中的多个第一节点转换为有向无环图中的多个第二节点;

根据所述第三资源量和每个所述第二节点对应的第四资源量,按照所述第二依赖关系,对所述多个第二节点进行分割,生成多个子图以及所述多个子图之间的第三依赖关系,其中,每个子图对应的第五资源量小于或等于所述第三资源量;

按照所述多个第一节点与所述多个第二节点之间的转换关系,对所述多个第一节点进行分割,生成与所述多个子图对应的多个模型块,以及所述多个模型块之间的第一依赖关系,其中,任意一个子图对应的所述第五资源量为该子图对应的模型块所需的第一资源量。

4. 根据权利要求1所述的方法,其特征在于,所述根据所述对应关系将所述多个模型块分别加载至对应的GPU,包括:

对于所述多个模型块中的任意一个目标模型块,将所述目标模型块加载至与所述目标模型块对应的目标GPU;

若所述目标GPU在加载所述目标模型块之后所剩余的可用资源量大于或等于所述目标模型块所需的第一资源量,则将所述目标模型块再次加载至所述目标GPU。

5. 根据权利要求1所述的方法,其特征在于,所述响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出,包括:

响应于所述图计算请求,按照所述第一依赖关系,通过所述多个GPU依次运行各自加载的所述多个模型块,并将所述多个模型块中前一个模型块的计算的中间结果输入至下一个模型块进行计算,直至最后一个模型块对输入的中间结果进行计算,来生成图计算结果并输出。

6. 一种数据处理装置,其特征在于,包括:

分块模块,用于根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系;

分配模块,用于根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,其中,所述对应关系中模型块所需的第一资源量小于或等于该模型块所对应的GPU的可用资源量;

加载模块,用于根据所述对应关系将所述多个模型块分别加载至对应的GPU;

接收模块,用于接收对所述图计算模型的图计算请求;

处理模块,用于响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出。

7. 根据权利要求6所述的装置,其特征在于,所述分块模块包括:

第一获取子模块,用于获取图计算模型中多个第一节点之间的第二依赖关系;

第二获取子模块,用于获取所述图计算模型所需的第二资源量;

分块子模块,用于根据所述第二资源量、所述多个GPU的数量以及每个GPU的可用资源量,按照所述第二依赖关系对所述图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系。

8. 根据权利要求7所述的装置,其特征在于,所述分块子模块包括:

计算单元,用于计算所述第二资源量和所述多个GPU的数量的比值,生成第三资源量,其中,所述第三资源量小于或等于所述每个GPU的可用资源量;

转换单元,用于将所述图计算模型中的多个第一节点转换为有向无环图中的多个第二节点;

第一分割单元,用于根据所述第三资源量和每个所述第二节点对应的第四资源量,按照所述第二依赖关系,对所述多个第二节点进行分割,生成多个子图以及所述多个子图之间的第三依赖关系,其中,每个子图对应的第五资源量小于或等于所述第三资源量;

第二分割单元,用于按照所述多个第一节点与所述多个第二节点之间的转换关系,对所述多个第一节点进行分割,生成与所述多个子图对应的多个模型块,以及所述多个模型块之间的第一依赖关系,其中,任意一个子图对应的所述第五资源量为该子图对应的模型块所需的第一资源量。

9. 根据权利要求6所述的装置,其特征在于,所述加载模块包括:

第一加载子模块,用于对于所述多个模型块中的任意一个目标模型块,将所述目标模型块加载至与所述目标模型块对应的目标GPU;

第二加载子模块,用于若所述目标GPU在加载所述目标模型块之后所剩余的可用资源量大于或等于所述目标模型块所需的第一资源量,则将所述目标模型块再次加载至所述目标GPU。

10. 根据权利要求6所述的装置,其特征在于,所述处理模块包括:

计算子模块,用于响应于所述图计算请求,按照所述第一依赖关系,通过所述多个GPU依次运行各自加载的所述多个模型块,并将所述多个模型块中前一个模型块的计算的中间结果输入至下一个模型块进行计算,直至最后一个模型块对输入的中间结果进行计算,来生成图计算结果并输出。

11.一种服务器,其特征在于,包括:处理器、通信接口、存储器和通信总线,其中,处理器,通信接口,存储器通过通信总线完成相互间的通信;

存储器,用于存放计算机程序;

处理器,用于执行存储器上所存放的程序时,实现如权利要求1至5中任一项所述的数据处理方法的步骤。

12.一种计算机可读存储介质,其上存储有计算机程序,其特征在于,该程序被处理器执行时实现如权利要求1至5中任一项所述的数据处理方法中的步骤。

数据处理方法、装置、服务器及计算机可读存储介质

技术领域

[0001] 本发明涉及云计算技术领域,特别是涉及一种数据处理方法、装置、服务器及计算机可读存储介质。

背景技术

[0002] 目前,人工智能模型的体积越来越大,推理计算模型(一般为图计算模型)所需的计算资源较多,而GPU(Graphics Processing Unit,图形处理器,俗称显卡)设备所提供的资源较少。

[0003] GPU在利用图计算模型进行推理计算时,需要将整个模型加载至显存中,再进行推理计算。那么当该图计算模型较大时,则GPU设备较难完成对大模型的加载,从而无法完成图计算。

[0004] 因此,目前需要本领域技术人员迫切解决的一个技术问题就是:如何对大型图计算模型完成推理计算。

发明内容

[0005] 本发明实施例提供了一种数据处理方法、装置、服务器及计算机可读存储介质,以解决相关技术中难以对大型图计算模型完成推理计算的问题。

[0006] 为了解决上述问题,根据本发明实施例的一个方面,本发明公开了一种数据处理方法,包括:

[0007] 根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系;

[0008] 根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,其中,所述对应关系中模型块所需的第一资源量小于或等于该模型块所对应的GPU的可用资源量;

[0009] 根据所述对应关系将所述多个模型块分别加载至对应的GPU;

[0010] 接收对所述图计算模型的图计算请求;

[0011] 响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出。

[0012] 根据本发明实施例的另一方面,本发明还公开了一种数据处理装置,包括:

[0013] 分块模块,用于根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系;

[0014] 分配模块,用于根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,其中,所述对应关系中模型块所需的第一资源量小于或等于该模型块所对应的GPU的可用资源量;

[0015] 加载模块,用于根据所述对应关系将所述多个模型块分别加载至对应的GPU;

[0016] 接收模块,用于接收对所述图计算模型的图计算请求;

[0017] 处理模块,用于响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出。

[0018] 根据本发明实施例的又一方面,本发明还公开了一种服务器,包括:处理器、通信接口、存储器和通信总线,其中,处理器,通信接口,存储器通过通信总线完成相互间的通信;存储器,用于存放计算机程序;处理器,用于执行存储器上所存放的程序时,实现上述任意一项所述的数据处理方法的步骤。

[0019] 根据本发明实施例的再一方面,本发明还公开了一种计算机可读存储介质,所述计算机可读存储介质中存储有指令,当其在计算机上运行时,使得计算机执行上述任意一项所述的数据处理方法中的步骤。

[0020] 根据本发明实施例的又一方面,本发明还公开了一种包含指令的计算机程序产品,当其在计算机上运行时,使得计算机执行上述任一所述的数据处理方法。

[0021] 在本发明实施例中,根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系,并根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,使得一个大的图计算模型被分割为多个模型块而分别加载至不同的GPU,由于一个GPU所加载的模型块所需要的资源量是小于该GPU可用资源量的,使得单个GPU加载一个大型图计算模型的部分模型,因此,实现了多个GPU对一个大型图计算模型的加载,并在接收到对图计算模型的图计算请求后,可以通过多个GPU上所分别加载的该图计算模型的多个模型块,来按照第一依赖关系进行计算,使得生成的图计算结果与直接采用图计算模型进行计算所生成的图计算结果是一致的,实现了对大型图计算模型的推理计算,突破了单个GPU的资源限制。

附图说明

[0022] 图1是本发明的一种数据处理系统实施例的结构框图;

[0023] 图2是本发明的一种数据处理方法实施例的步骤流程图;

[0024] 图3是本发明一个实施例的有向无环图的示意图;

[0025] 图4是本发明的一种数据处理装置实施例的结构框图;

[0026] 图5是本发明的一种服务器实施例的结构框图。

具体实施方式

[0027] 为使本发明的上述目的、特征和优点能够更加明显易懂,下面结合附图和具体实施方式对本发明作进一步详细的说明。

[0028] 参照图1,示出了本发明的一个数据处理系统实施例的框图,该系统可以应用于服务器。

[0029] 该系统可以包括图分割模块、GPU计算模块(具体为图1中的各个GPU,例如GPU0、GPU1、GPU2、GPUN)以及GPU通讯模块。

[0030] 其中,由于系统中包括多个GPU,且分别用于加载同一个大的图计算模型的不同模型块,因此,为了确保图计算的准确性,需要使用GPU通讯模块来实现不同GPU之间的数据通讯。

[0031] GPU通讯模块,用于负责在多卡(即多个显卡(多个GPU))计算时,来在不同GPU之间进行通讯,其分为如下方式:

[0032] 一:多个GPU部署在同一台服务器上,即单机多卡模式下,GPU通讯模块可以使用如图1所示的PCIE(peripheral component interconnect express,是一种高速串行计算机扩展总线标准),或NVLink(是英伟达(NVIDIA)开发并推出的一种总线及其通信协议)来实现多卡之间进行通讯,完成多个显卡的联合运算。

[0033] 二:多个GPU部署在不同服务器上,当多个GPU需要采用网络通讯的方式时,则GPU通讯模块可以使用如图1所示的RDMA(Remote Direct Memory Access,远程直接数据存取)来实现多卡之间的网络通讯,完成不同服务器上的多个显卡的联合运算。

[0034] 图分割模块,主要用于对接收的模型文件(即图计算模型的文件)以及GPU的资源描述信息,并利用这两种信息来对该模型文件进行分块,生成多个模型块,并为每个模型块分配GPU;

[0035] GPU计算模块,这里为图1示意性示出的4个GPU,则分别用于对分配的模型块进行加载,并在系统接收到调用方的图计算请求时,4个GPU利用各自加载的模型块进行联合运算,将运算结果返回给调用方。

[0036] 关于上述系统中各个模块的具体实现步骤,可以参照后续的各个数据处理方法的实施例的描述。

[0037] 下面基于图1所示的数据处理系统,来对本发明各个实施例的数据处理方法作详细阐述。

[0038] 参照图2,示出了本发明的一种数据处理方法实施例的步骤流程图,该方法可以应用于服务器,该方法具体可以包括如下步骤:

[0039] 步骤101,根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系;

[0040] 在一个示例中,如图1所示,图分割模块可以接收到图计算模型的文件以及GPU资源描述信息(这里包括待分配的多个GPU中每个GPU的可用资源量)。

[0041] 其中,该多个GPU的可用资源量可以相同或者不同,此外,多个GPU各自支持的最大显存也是可以相同或者不同的。

[0042] 此外,由于在对大的图计算模型进行加载时,受一个GPU的显存限制,因此,这里,对于任意一个GPU,其可用资源量对应的资源参数可以是显存信息。即GPU资源描述信息描述了每个GPU的可用显存。为了便于理解,后文实施例以可用资源量为可用显存为例进行说明。

[0043] 另外,由于这里的每个GPU在加载本发明的模型块之前可能正加载有其他模型,或者因为其他原因导致部分显存被占用,因此,这里的每个GPU的可用显存是小于或等于相应GPU所支持的最大显存的。

[0044] 由于图计算模型是推理计算模型,推理计算不需要反向传播梯度,因此,图计算模型中第N层的计算逻辑依赖于N-1层的计算逻辑的输出结果,不存在N-1层的计算逻辑依赖第N层的计算逻辑的输出结果的情况。即前一层逻辑在计算时不需要借助于后一层逻辑的数据输入。因此,图分割模块可以参照各个GPU的可用显存来对图计算模型进行分块,从而将单个模型拆分成小模型集合(即多个模型块),使得各个模型块所需的显存不超过GPU的

可用显存。相应的,在分块后,也会形成多个模型块之间的依赖关系(例如模型块4的输入依赖于模型块3的输出,模型块3的输入依赖于模型块2的输入,模型块2的输入依赖于模型块1的输出的这种依赖关系)。

[0045] 其中,不同模型块所需要的显存可以相同或者不同。

[0046] 在一个示例中,当多个GPU的可用显存相同时,则这里各个模型块所需的显存均不超过该GPU的可用显存;

[0047] 在另一个示例中,当多个GPU的可用显存不完全相同时,则这里各个模型块所需要的显存均不超过多个GPU的可用显存的均值,或者,这里各个模型块所需要的显存均不超过多个GPU的可用显存的最小值。

[0048] 步骤102,根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系;

[0049] 其中,所述对应关系中模型块所需的第一资源量小于或等于该模型块所对应的GPU的可用资源量;

[0050] 在图1的示例中,图分割模块可以获取每个模型块所需要的显存,在获取该显存信息时,可以根据图计算模型中每一层计算逻辑的参数描述信息,来确定各层所需要的显存,而每一模型块包括至少一层计算逻辑,从而可以将一个模型块所包括的每个计算逻辑所需要的显存进行累加,得到一个模型块所需要的显存,同理,同理后文实施例提及的在获取该图计算模型所需要的显存时,也是对各层计算逻辑所需要的显存进行累加得到,其中,每层计算逻辑所需要的显存则是依据该层计算逻辑的参数描述信息来确定。

[0051] 图分割模块可以根据每个模型块所需要的显存,以及每个GPU的可用显存,来对多个模型块分配GPU。

[0052] 需要说明的是,一个模型块只分配到一个GPU,而不同模型块可以分配到同一个或不同的GPU,即一个GPU可以加载单个模型块或多个不同的模型块。具体加载单个模型块还是不同的模型块,取决于该GPU的可用显存以及其所需要加载的模型块所需要的显存。

[0053] 经过模型块的分配之后,可以生成模型块与GPU之间的对应关系,在该对应关系中,任意一个模型块所需要的显存是小于或等于该模型块所分配至的GPU(即对应关系中所对应的GPU)的可用显存的。

[0054] 步骤103,根据所述对应关系将所述多个模型块分别加载至对应的GPU;

[0055] 其中,在图1的示例中,图分割模块可以将对应关系中的各个模型块分别部署至对应的各个GPU。那么GPU计算模块,即图1中的4个GPU就可以分别加载各自所部署的模型块,实现多个GPU对一个大的图计算模型的加载,从而完成大模型的推理计算。

[0056] 步骤104,接收对所述图计算模型的图计算请求;

[0057] 其中,在图1的示例中,图分割模块可以接收来自调用方,例如客户端的图计算请求,其中,该图计算请求可以包括待计算的源数据以及对该源数据进行计算的图计算模型的标识信息。这里的图计算模型即为步骤101~步骤103所分块的图计算模型。

[0058] 步骤105,响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出。

[0059] 其中,图分割模块可以响应于该图计算请求,来通过图1中4个GPU所加载的例如4个模型块来按照所述第一依赖关系,来对图计算请求进行处理,生成图计算结果并输出该

图计算结果。

[0060] 例如按照上述第一依赖关系, GPU0上所加载的模型块1对源数据进行计算, 生成中间结果1; GPU0将中间结果1输出给GPU1, 由GPU1上所加载的模型块2对中间结果1进行计算, 生成中间结果2; GPU1将中间结果2输出给GPU2, 由GPU2上所加载的模型块3对中间结果2进行计算, 生成中间结果3; GPU2将中间结果3输出给GPUN, 由GPUN上所加载的模型块4对中间结果3进行计算, 生成图计算结果。

[0061] 多个模型块运行后对图计算请求进行计算得到的图计算结果与分块前的原始的图计算模型对该图计算请求进行计算得到的图计算结果是相同的。

[0062] 此外, 关于多个GPU联合计算来对图计算请求进行响应时, 可以由图分割模块来控制多个GPU的模型块运算后的中间结果的传输, 也可以在步骤103之后, 图分割模块将多个模型块之间的第一依赖关系下发给上述4个GPU, 从由各个GPU自动控制不同GPU之间的中间结果的传输。

[0063] 在传统技术中, 由于单个图计算模型需要的计算资源, 超出单个GPU所提供的计算资源, 这种限制为GPU运行机制与GPU物理设备限制, 因此, 单GPU设备无法突破显存限制的瓶颈。

[0064] 那么为了解决上述问题, 在本发明实施例中, 根据待分配的多个GPU中每个GPU的可用资源量, 对图计算模型进行分块, 生成多个模型块以及所述多个模型块之间的第一依赖关系, 并根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量, 对所述多个模型块分别分配GPU, 生成模型块与GPU之间的对应关系, 使得一个大的图计算模型被分割为多个模型块而分别加载至不同的GPU, 由于一个GPU所加载的模型块所需要的资源量是小于该GPU可用资源量的, 使得单个GPU加载一个大型图计算模型的部分模型, 因此, 实现了多个GPU对一个大型图计算模型的加载, 并在接收到对图计算模型的图计算请求后, 可以通过多个GPU上所分别加载的该图计算模型的多个模型块, 来按照第一依赖关系进行计算, 使得生成的图计算结果与直接采用图计算模型进行计算所生成的图计算结果是一致的, 实现了对大型图计算模型的推理计算, 突破了单个GPU的资源限制。

[0065] 可选地, 在执行步骤101时, 可以通过S201~S203来实现:

[0066] S201, 获取图计算模型中多个第一节点之间的第二依赖关系;

[0067] S202, 获取所述图计算模型所需的第二资源量;

[0068] S203, 根据所述第二资源量、所述多个GPU的数量以及每个GPU的可用资源量, 按照所述第二依赖关系对所述图计算模型进行分块, 生成多个模型块以及所述多个模型块之间的第一依赖关系。

[0069] 具体而言, 在S201中, 图计算模型中的每一个计算逻辑可以看作一个节点, 其中, 循环计算逻辑可以看作一个循环节点(例如for循环的10次计算逻辑看作一个节点), 由于图计算模型无反向传播, 因此, 可以利用拓扑排序算法来获取图计算模型的多个第一节点之间的第二依赖关系。

[0070] 在S202中, 该步骤的具体实现方法可以参照上文, 这里不再赘述。这里的第二资源量为根据图计算模型中每个计算逻辑的参数描述信息所估计的每个计算逻辑所需要的资源量的累加和。

[0071] 在S203中, 在对图计算模型进行分块时, 可以按照多个第一节点之间的第二依赖

关系来进行顺序分块,每一个模型块包括至少一个第一节点。而至于每次分割生成一个模型块时,按照顺序将几个第一节点分为一组构成一个模型块,则取决于该第二资源量、GPU的数量,以及每个GPU的可用资源量。

[0072] 由于图计算模型中的多个第一节点之间存在第二依赖关系,那么按照该第二依赖关系对该模型进行分块之后,所生成的多个模型块之间的第一依赖关系也是遵循于该第二依赖关系的。

[0073] 在S203中对图计算模型进行分块时,没有限制具体的分块方式,可以采用类似平均的方式进行分块(即不同模型块所需要的资源量是基本相同的,且该资源量均小于多个GPU中单个GPU的可用资源量的最小值),也可以采用灵活的分块方式,即不同模型块所需要的资源量可以是参照每个GPU的可用资源量而灵活分割的。

[0074] 在本发明实施例中,由于图计算模型中的节点之间存在一定的依赖关系,因此,为了确保分割后的多个模型块对图计算请求的计算结果,与原图计算模型对该图计算请求的计算结果的一致性,可以按照该依赖关系来对图计算模型进行分块,并且,在分块时,具体将多少个节点以及哪些节点作为一个模型块,可以参照待分配的多个GPU的数量,以及每个GPU的可用资源量,对图计算模型合理分割得到多个模型块,以及多个模型块之间的第一依赖关系。

[0075] 可选地,在执行S203时,可以通过S301~S304来实现:

[0076] S301,计算所述第二资源量和所述多个GPU的数量的比值,生成第三资源量,其中,所述第三资源量小于或等于所述每个GPU的可用资源量;

[0077] S302,将所述图计算模型中的多个第一节点转换为有向无环图中的多个第二节点;

[0078] S303,根据所述第三资源量和每个所述第二节点对应的第四资源量,按照所述第二依赖关系,对所述多个第二节点进行分割,生成多个子图以及所述多个子图之间的第三依赖关系,其中,每个子图对应的第五资源量小于或等于所述第三资源量;

[0079] S304,按照所述多个第一节点与所述多个第二节点之间的转换关系,对所述多个第一节点进行分割,生成与所述多个子图对应的多个模型块,以及所述多个模型块之间的第一依赖关系,其中,任意一个子图对应的所述第五资源量为该子图对应的模型块所需的第一资源量。

[0080] 具体而言,在S301中,不同GPU的可用资源量可以相同或不同。

[0081] 其中,可以计算图计算模型所需要的总资源量(即第二资源量) m ,和GPU数量 n 的比值,生成第三资源量 $p=m/n$,第三资源量 p 即为平均每个GPU所需要提供的资源量。该第三资源量 p 是小于或等于任意一个GPU的可用资源量的。

[0082] 在S302中,可以将图计算模型的多个第一节点抽象为有向无环图(其中,图计算模型中对应循环计算逻辑的带循环的第一节点可抽象为有向无环图中的单个第二节点,由于需要图计算模型中多个第一节点之间的第二依赖关系来进行图推理计算,因此可以将多个第一节点转换为有向无环图中的多个第二节点,其中,图计算模型中的每一个第一节点(是逻辑代码)可以转回为有向无环图中的一个第二节点(是一个节点数字),转换前后的两组节点是一一对应关系。

[0083] 关于将图计算模型转换为有向无环图的方法可以采用传统技术中的任意一种方

法,这里不再赘述。

[0084] 在一个示例中,图3示出了本发明实施例的图计算模型所转换成的有向无环图的示意图,该有向无环图包括10个第二节点。10个节点之间的箭头方向是遵循图计算模型的多个节点之间的第二依赖关系的。可以理解的是,箭头方向表示数据传递方向,或者对数据的计算顺序。

[0085] 在S303中,关于一个第二节点对应的第四资源量即为该第二节点对应的转换前的第一节点所需要的资源量(例如一个计算逻辑所需要的资源量),该第四资源量的获取方式与上述第三资源量的获取原理是类似的,这里不再赘述。本步骤中,可以将有向无环图中的单个节点或多个节点抽象为一个计算单元。即一个模型块在有向无环图中表示为一个计算单元。

[0086] 因此,需要对图3所示的有向无环图中的10个节点进行划分,在划分时需要按照箭头方向(即第二依赖关系)进行顺序划分。

[0087] 在进行划分时,可以采用拓扑排序的方式来对10个节点进行划分,考虑到输入节点(即1号节点)的入度为零,输出节点(10号节点)的出度为零(其中,入度指代一个节点的输入节点的个数,出度表示一个节点的输出节点的个数),因此,可以以输入节点为起始点,输出节点为终止点,在拓扑排序的过程中累加单个节点所需的资源量(这里1号~10号节点所对应的第四资源量依次为 $m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, m_{10}$,且图计算模型所需要的总资源量 m 等于上述10个第四资源量的累加和),在累加过程中,当累加资源量接近于该第三资源量 p 时,则将该部分排序的第二节点分割为一组,生成一个子图,直至拓扑排序结束,可以生成多个子图,以及多个子图之间的第三依赖关系,其中,该第三依赖关系也是遵循多个第一节点之间的第二依赖关系的,实质上第三依赖关系与第一依赖关系是相同的,只不过第三依赖关系中所指向的对象是子图,而第一依赖关系中所指向的对象是模型块,而子图就是模型块的有向无环图的表示。

[0088] 其中,该子图对应的资源量即为分割前累加得到的资源量。例如图3中生成了4个子图,分别为子图1、子图2、子图3、子图 n ,子图1对应的资源量 $q_1 = m_1 + m_2 + m_3$,子图2对应的资源量 $q_2 = m_4 + m_5 + m_6 + m_7$,子图3对应的资源量 $q_3 = m_8 + m_9$,子图 n 对应的资源量 $q_n = m_{10}$ 。由于在每次分割前都是累加资源量接近与第三资源量 p 时分割的,因此, q_1, q_2, q_3, q_n 均小于或等于 p 。这里4个子图之间的依赖关系为,子图 n 的输入依赖于子图3的输出,子图3的输入依赖于子图2的输入,子图2的输入依赖于子图1的输出的这种依赖关系

[0089] 这里的每个子图即为上述每个计算单元。

[0090] 其中,每个子图包括至少一个第二节点。

[0091] 在S304中,可以将图计算模型的多个第一节点,分割为多个模型块,且模型块1、模型块2、模型块3、模型块4分别对应于子图1、子图2、子图3、子图 n 。

[0092] 在本发明实施例中,采用类似平均分配的方法利用GPU数目对该图计算模型所需要的总资源量作平均分配,得到第三资源量,并通过将图计算模型转换为有向无环图,来实现对图计算模型的分块,使得每次分块得到的一个模型块所需要的资源量都是接近且小于或等于该第三资源量的,在遵循图计算模型的节点之间的依赖关系的前提下,实现了对多个模型块的合理划分。

[0093] 那么在本实施例中,在执行步骤102时,在对所述多个模型块中的任意一个目标模

型块分配目标GPU时,可以根据所述目标模型块所需的第五资源量,将所述目标模型块分配至所述多个GPU中的目标GPU,其中,所述目标GPU的可用资源量大于或等于所述第五资源量。

[0094] 在本发明实施例中,在对一个模型块分配GPU时,由于参考了GPU的可用资源量以及该模型块所需要的资源量,并将模块块分配至可用资源量大于该模型块的所需资源量的GPU中,使得分配有该模型块的GPU中的资源足够多地能够用于对分配的模型块进行加载,确保了图计算模型被分块后的每个模型块都可以被相应的GPU完成加载,不存在资源不够的情况,确保了图计算模型的加载和使用。

[0095] 可选地,在一个实施例中,为了提升对图计算模型的加载效率,在执行步骤103时,对于所述多个模型块中的任意一个目标模型块,可以将所述目标模型块加载至与所述目标模型块对应的目标GPU。也就是说,可以对一个GPU只分配一个模型块来进行加载,这样多个模型块可以被并行加载,从而提升了图计算模型的加载效率。

[0096] 可选地,在另一个实施例中,为了提升GPU的利用率,在执行步骤103时,对于所述多个模型块中的任意一个目标模型块,可以将所述目标模型块加载至与所述目标模型块对应的目标GPU,并在将所述目标模型块加载至与所述目标模型块对应的目标GPU之后,若所述目标GPU在加载所述目标模型块之后所剩余的可用资源量大于或等于所述目标模型块所需的第一资源量,则还可以将所述目标模型块再次加载至所述目标GPU。

[0097] 例如,将模型块1加载至GPU0之后,若该GPU0还有剩余资源,且该剩余资源量大于模型块1所需要的资源量 m_1 ,则可以继续加载模型块1至该GPU0,直至该GPU0的剩余资源量小于 m_1 ,这样可以使得单个GPU加载相同的多个模型块(例如加载了两个模型块1),那么该GPU0则可以同时支持该模型块1对应的计算逻辑的两次计算请求,从而提升了对GPU的资源利用率。

[0098] 在本发明实施例中,在一个GPU加载了所分配的模型块之后,如果该GPU的可用资源量大于该模型块所需要的资源量,则可以将该模型块再次加载至GPU,即使得同一个GPU加载有至少两个相同的模型块,由于一个模型块的运行可以完成一次该模型块对应的计算逻辑的图计算,因此,使得同一个GPU加载至少两个相同的模型块,可以提升对该GPU的资源利用率,以及提升图计算请求的响应效率。

[0099] 此外,本发明实施例的方法的GPU计算模块使用Nvidia MPS(多进程服务技)技术或组成pipeline(线性通信模型),来将GPU所加载的模型块服务化,那么当一个GPU加载有多个相同模型块时,则GPU计算模块根据加载同一模型块的数量来启动相应数量的服务,例如上述GPU0对同一个模型块1加载了2次,因此,这里需要启动2个服务,服务数量等于被加载的模型块的数量,这样,才符合MPS以及pipeline计算的要求,进而提高GPU的资源利用率以及小模型集合的计算效率。

[0100] 可选地,在执行步骤105时,可以响应于所述图计算请求,按照所述第一依赖关系,通过所述多个GPU依次运行各自加载的所述多个模型块,并将所述多个模型块中前一个模型块的计算的中间结果输入至下一个模型块进行计算,直至最后一个模型块对输入的中间结果进行计算,来生成图计算结果并输出。

[0101] 其中,关于多个GPU联合计算来对图计算请求进行响应时,可以由图分割模块来控制多个GPU的模型块运算后的中间结果的传输,也可以在步骤103之后,图分割模块将多个

模型块之间的第一依赖关系下发给上述4个GPU,从由各个GPU自动控制不同GPU之间的中间结果的传输。

[0102] 在本发明实施例中,按照图计算模型所分割的多个模型块之间的第一依赖关系的顺序,来由多个GPU依次运行各自加载的所述多个模型块,使得多个模型块的运行顺序符合所述第一依赖关系,并将前一个模型块计算的中间结果传输给下一个模型块进行计算,直至最后一个模型块对输入的中间结果计算,使得最后一个模型块所生成的计算结果即为图计算结果,那么分割后的多个模型块对图计算请求进行计算后生成的图计算结果则可以与完成的未分割的图计算模型计算后生成的图计算结果相同,确保了图计算模型分块加载后,运算结果与分块加载之前的运算结果相同。

[0103] 可选地,在步骤103之后,图分割模块对有向无环图进行分割后,可以将各个模型块的分割节点的编号下发给相应的加载各个模型块的GPU。

[0104] 例如图计算模型的多个第一节点的编号与图3所示的有向无环图的多个节点的编号一致,即图计算模型块包括图3所示的10个第一节点,则在步骤103之后,在系统初始化时,图分割模块可以将模型块1的终止节点:节点2和节点3的编号下发给加载有模型块2(其中,模型块2依赖于模型块1)的GPU1,并将节点4依赖于节点2和节点3,以及节点5依赖于节点3的这个依赖关系下发给GPU1;类似的,将模型块2的终止节点:节点6和节点7的编号下发给GPU3,并将节点8依赖于节点6和节点7的依赖关系下发给GPU3;将模型块3的终止节点,即节点9的编号下发给GPU_n,并将节点10依赖于节点9的依赖关系下发给GPU_n。

[0105] 也就是说,图分割模块在对图计算模型分块后,可以记录分块时的切割节点(第一节点)的编号信息,并记录对每个模型块所分配的GPU的GPU编号,以及将切割节点之间的依赖关系(属于第二依赖关系中的一部分)下发给加载有各个切割节点的GPU。

[0106] 那么图分割模块在接收到图计算请求之后,可以确定图计算模型,然后,将加载有该图计算模型的首个模型块的GPU的地址(这里为图1所示的GPU0的地址信息)返回给客户端,然后,客户端根据该地址向GPU0发送图计算请求(包括待计算的源数据),那么模型块1对该源数据进行计算得到节点2输出的结果2和节点3输出的结果3;然后,GPU1可以根据下发的切割节点之间的依赖关系,从GPU0处获取结果2和结果3,并将结果2和结果3均传入给模型块2的节点4,以及结果3传入给模型块2的节点5,模型块2根据接收的结果2和结果3进行运算,生成节点6输出的结果6以及节点7输出的结果7;然后,GPU2可以根据下发的切割节点之间的依赖关系,从GPU1处获取结果6和结果7,并由模型块3对结果6和结果7进行运算,生成节点9输出的结果9;接着,GPU_n可以根据下发的切割节点之间的依赖关系,从GPU2处获取到结果9,并由模型块4对结果9进行运算,生成图计算结果,由GPU_n将该图计算结果输出给客户端。

[0107] 在本发明实施例中,图分割模型在对图计算模型分割成多个模型块之后,可以记录各个模型块之间的切割节点的信息,以及切割节点之间的依赖关系,并下发给加载有各个模型块的各个GPU,从而在由各个GPU对图计算请求进行响应时,能够利用多个模型块进行联合运算,使得运算得到的图计算结果,与采用未分割的图计算模型计算得到的结果是相同的,确保了不同GPU之间的数据传输和数据计算的准确性。

[0108] 那么在不同GPU之间进行数据通信时,针对同一台物理机上的多个GPU,可以使用PCIE通讯或NVLink技术,来实现不同GPU之间的通讯;而对于不同物理机上的GPU,如果需要

进行网络连接,则可以使用RDMA技术完成物理机的相互访问,以完成多个GPU的有效通讯。

[0109] 需要说明的是,对于方法实施例,为了简单描述,故将其都表述为一系列的动作组合,但是本领域技术人员应该知悉,本发明实施例并不受所描述的动作顺序的限制,因为依据本发明实施例,某些步骤可以采用其他顺序或者同时进行。其次,本领域技术人员也应该知悉,说明书中所描述的实施例均属于优选实施例,所涉及的动作并不一定是本发明实施例所必须的。

[0110] 与上述本发明实施例所提供的上述数据处理方法相对应,参照图4,示出了本发明的一种数据处理装置实施例的结构框图,具体可以包括如下模块:

[0111] 分块模块41,用于根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系;

[0112] 分配模块42,用于根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,其中,所述对应关系中模型块所需的第一资源量小于或等于该模型块所对应的GPU的可用资源量;

[0113] 加载模块43,用于根据所述对应关系将所述多个模型块分别加载至对应的GPU;

[0114] 接收模块44,用于接收对所述图计算模型的图计算请求;

[0115] 处理模块45,用于响应于所述图计算请求,通过所述多个GPU上所加载的所述多个模型块,按照所述第一依赖关系对所述图计算请求进行处理,生成图计算结果并输出。

[0116] 可选地,所述分块模块41包括:

[0117] 第一获取子模块,用于获取图计算模型中多个第一节点之间的第二依赖关系;

[0118] 第二获取子模块,用于获取所述图计算模型所需的第二资源量;

[0119] 分块子模块,用于根据所述第二资源量、所述多个GPU的数量以及每个GPU的可用资源量,按照所述第二依赖关系对所述图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系。

[0120] 可选地,所述分块子模块包括:

[0121] 计算单元,用于计算所述第二资源量和所述多个GPU的数量的比值,生成第三资源量,其中,所述第三资源量小于或等于所述每个GPU的可用资源量;

[0122] 转换单元,用于将所述图计算模型中的多个第一节点转换为有向无环图中的多个第二节点;

[0123] 第一分割单元,用于根据所述第三资源量和每个所述第二节点对应的第四资源量,按照所述第二依赖关系,对所述多个第二节点进行分割,生成多个子图以及所述多个子图之间的第三依赖关系,其中,每个子图对应的第五资源量小于或等于所述第三资源量;

[0124] 第二分割单元,用于按照所述多个第一节点与所述多个第二节点之间的转换关系,对所述多个第一节点进行分割,生成与所述多个子图对应的多个模型块,以及所述多个模型块之间的第一依赖关系,其中,任意一个子图对应的所述第五资源量为该子图对应的模型块所需的第一资源量。

[0125] 可选地,所述加载模块43包括:

[0126] 第一加载子模块,用于对于所述多个模型块中的任意一个目标模型块,将所述目标模型块加载至与所述目标模型块对应的目标GPU;

[0127] 第二加载子模块,用于若所述目标GPU在加载所述目标模型块之后所剩余的可用

资源量大于或等于所述目标模型块所需的第一资源量,则将所述目标模型块再次加载至所述目标GPU。

[0128] 可选地,所述处理模块45包括:

[0129] 计算子模块,用于响应于所述图计算请求,按照所述第一依赖关系,通过所述多个GPU依次运行各自加载的所述多个模型块,并将所述多个模型块中前一个模型块的计算的中间结果输入至下一个模型块进行计算,直至最后一个模型块对输入的中间结果进行计算,来生成图计算结果并输出。

[0130] 在本发明实施例中,根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及所述多个模型块之间的第一依赖关系,并根据每个所述模型块所需的第一资源量和所述每个GPU的可用资源量,对所述多个模型块分别分配GPU,生成模型块与GPU之间的对应关系,使得一个大的图计算模型被分割为多个模型块而分别加载至不同的GPU,由于一个GPU所加载的模型块所需要的资源量是小于该GPU可用资源量的,使得单个GPU加载一个大型图计算模型的部分模型,因此,实现了多个GPU对一个大型图计算模型的加载,并在接收到对图计算模型的图计算请求后,可以通过多个GPU上所分别加载的该图计算模型的多个模型块,来按照第一依赖关系进行计算,使得生成的图计算结果与直接采用图计算模型进行计算所生成的图计算结果是一致的,实现了对大型图计算模型的推理计算,突破了单个GPU的资源限制。

[0131] 对于装置实施例而言,由于其与方法实施例基本相似,所以描述的比较简单,相关之处参见方法实施例的部分说明即可。

[0132] 根据本发明的又一个实施例,本发明还提供了一种服务器,如图5所示,包括处理器501、通信接口502、存储器503和通信总线504,其中,处理器501,通信接口502,存储器503通过通信总线504完成相互间的通信;

[0133] 存储器503,用于存放计算机程序;

[0134] 处理器501,用于执行存储器503上所存放的程序时,实现如下步骤:

[0135] 根据待分配的多个GPU中每个GPU的可用资源量,对图计算模型进行分块,生成多个模型块以及多个模型块之间的第一依赖关系;

[0136] 根据每个模型块所需的第一资源量和每个GPU的可用资源量,对多个模型块分别分配GPU,生成模型块与GPU之间的对应关系;

[0137] 根据对应关系将多个模型块分别加载至对应的GPU;

[0138] 接收对图计算模型的图计算请求;

[0139] 响应于图计算请求,通过多个GPU上所加载的多个模型块,按照第一依赖关系对图计算请求进行处理,生成图计算结果并输出。

[0140] 上述服务器提到的通信总线504可以是外设部件互连标准(Peripheral Component Interconnect,简称PCI)总线或扩展工业标准结构(Extended Industry Standard Architecture,简称EISA)总线等。该通信总线504可以分为地址总线、数据总线、控制总线等。为便于表示,图中仅用一条粗线表示,但并不表示仅有一根总线或一种类型的总线。

[0141] 通信接口502用于上述服务器与其他设备之间的通信。

[0142] 存储器503可以包括随机存取存储器(Random Access Memory,简称RAM),也可以

包括非易失性存储器 (non-volatile memory), 例如至少一个磁盘存储器。可选的, 存储器 503 还可以是至少一个位于远离前述处理器的存储装置。

[0143] 上述的处理器 501 可以是通用处理器, 包括中央处理器 (Central Processing Unit, 简称 CPU)、网络处理器 (Network Processor, 简称 NP) 等; 还可以是数字信号处理器 (Digital Signal Processing, 简称 DSP)、专用集成电路 (Application Specific Integrated Circuit, 简称 ASIC)、现场可编程门阵列 (Field-Programmable Gate Array, 简称 FPGA) 或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件。

[0144] 根据本发明的再一个实施例, 本发明还提供了一种计算机可读存储介质, 该计算机可读存储介质中存储有指令, 当其在计算机上运行时, 使得计算机执行上述任一实施例所述的数据处理方法中的步骤。

[0145] 在本发明提供的又一实施例中, 还提供了一种包含指令的计算机程序产品, 当其在计算机上运行时, 使得计算机执行上述任一实施例所述的数据处理方法。

[0146] 在上述实施例中, 可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时, 可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行所述计算机程序指令时, 全部或部分地产生按照本发明实施例所述的流程或功能。所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中, 或者从一个计算机可读存储介质向另一个计算机可读存储介质传输, 例如, 所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线 (例如同轴电缆、光纤、数字用户线 (DSL)) 或无线 (例如红外、无线、微波等) 方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质, (例如, 软盘、硬盘、磁带)、光介质 (例如, DVD)、或者半导体介质 (例如固态硬盘 Solid State Disk (SSD)) 等。

[0147] 需要说明的是, 在本文中, 诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来, 而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺序。而且, 术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含, 从而使得包括一系列要素的过程、方法、物品或者设备不仅包括那些要素, 而且还包括没有明确列出的其他要素, 或者是还包括为这种过程、方法、物品或者设备所固有的要素。在没有更多限制的情况下, 由语句“包括一个……”限定的要素, 并不排除在包括所述要素的过程、方法、物品或者设备中还存在另外的相同要素。

[0148] 本说明书中的各个实施例均采用相关的方式描述, 各个实施例之间相同相似的部分互相参见即可, 每个实施例重点说明的都是与其他实施例的不同之处。尤其, 对于系统实施例而言, 由于其基本相似于方法实施例, 所以描述的比较简单, 相关之处参见方法实施例的部分说明即可。

[0149] 以上所述仅为本发明的较佳实施例而已, 并非用于限定本发明的保护范围。凡在本发明的精神和原则之内所作的任何修改、等同替换、改进等, 均包含在本发明的保护范围内。

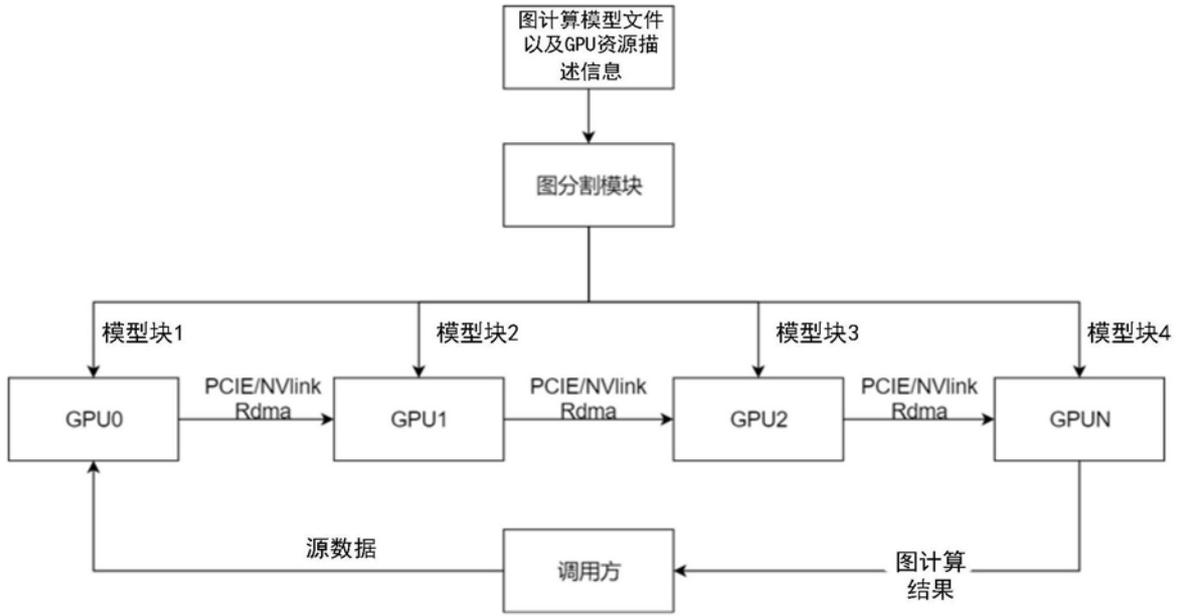


图1

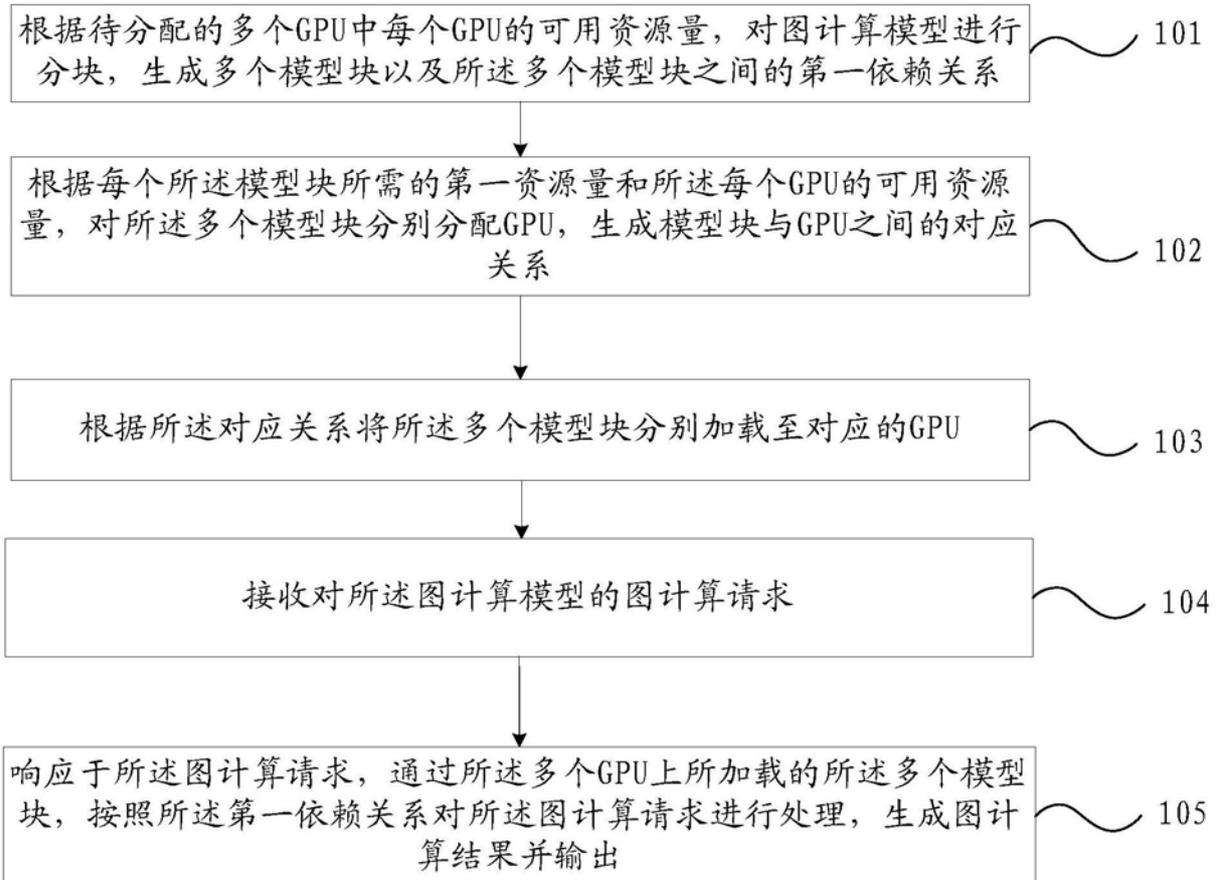


图2

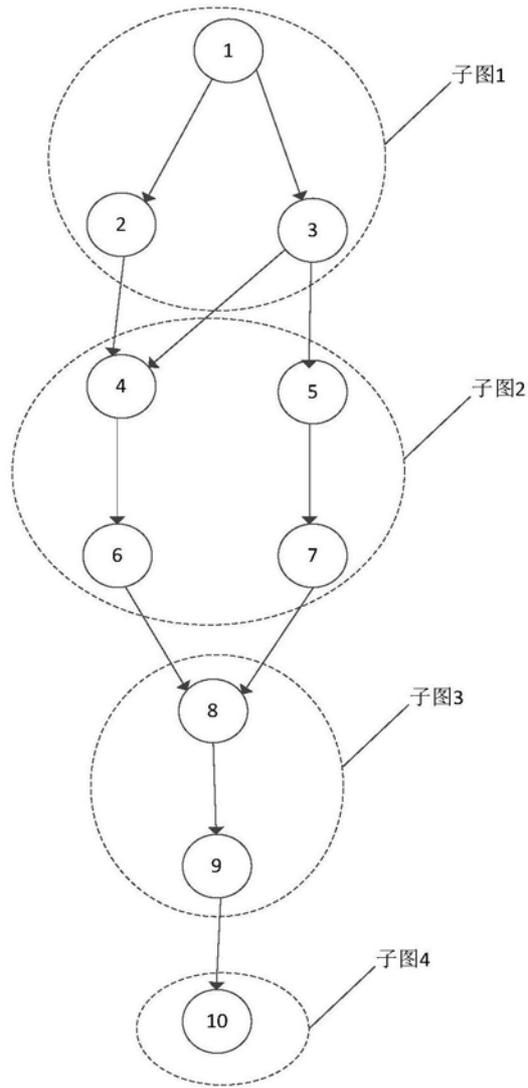


图3



图4

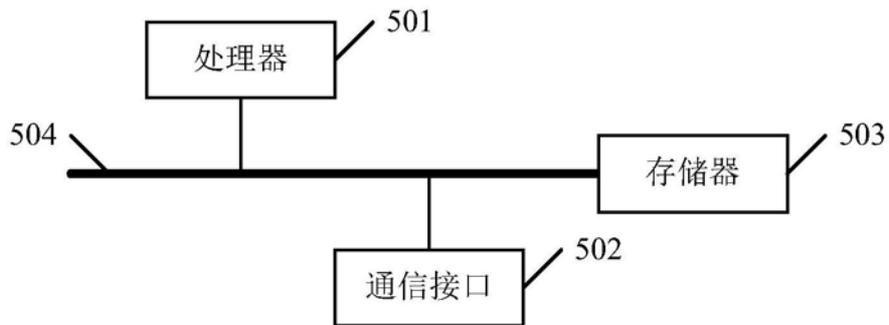


图5