

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2024 年 4 月 25 日 (25.04.2024)



(10) 国际公布号
WO 2024/082679 A1

(51) 国际专利分类号:
G06N 3/04 (2023.01) **G06F 8/30** (2018.01)
G06N 3/08 (2023.01)

(21) 国际申请号: PCT/CN2023/102301

(22) 国际申请日: 2023 年 6 月 26 日 (26.06.2023)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
202211277782.4 2022 年 10 月 19 日 (19.10.2022) CN

(71) 申请人: 华为技术有限公司 (HUAWEI TECHNOLOGIES CO., LTD.) [CN/CN]; 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。

(72) 发明人: 焦建兵 (JIAO, Jianbing); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。许世峰 (XU, Shifeng); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。林嘉树 (LIN, Jiashu); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。周剑 (ZHOU, Jian); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。

(74) 代理人: 北京润泽恒知识产权代理有限公司 (BEIJING RUN ZEHENG INTELLECTUAL PROPERTY LAW FIRM); 中国北京市海淀区中关村南大街甲 18 号北京国际 C 座 6 层 606, Beijing 100081 (CN)。

(81) 指定国(除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG,

(54) Title: METHOD AND APPARATUS FOR PROCESSING COMPUTATIONAL GRAPH

(54) 发明名称: 计算图的处理方法和装置

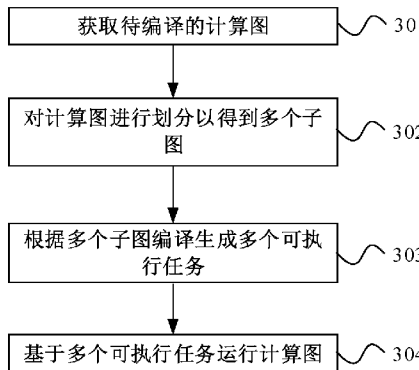


图 3

- 301 Acquire a computational graph to be compiled
302 Divide the computational graph to obtain a plurality of sub-graphs
303 Perform compilation according to the plurality of sub-graphs to generate a plurality of executable tasks
304 Run the computational graph on the basis of the plurality of executable tasks

(57) Abstract: Provided in the present application are a method and apparatus for processing a computational graph. The method for processing a computational graph in the present application comprises: acquiring a computational graph to be compiled, wherein a dynamic shape is used for input data of the computational graph, and the computational graph comprises a plurality of operators; dividing the computational graph to obtain a plurality of sub-graphs, wherein any sub-graph comprises at least one of the operators in the computational graph; performing compilation according to the plurality of sub-graphs to generate a plurality of executable tasks; and running the computational graph on the basis of the plurality of executable tasks. The present application can achieve software compiling and efficient execution for a network model in a dynamic shape, thereby implementing a program execution method, in

BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

- (84) 指定国(除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第21条(3))。

which a computational graph is taken as a core, in a dynamic shape.

(57) 摘要: 本申请提供一种计算图的处理方法和装置。本申请计算图的处理方法, 包括: 获取待编译的计算图, 所述计算图的输入数据采用动态shape, 所述计算图包括多个算子; 对所述计算图进行划分以得到多个子图, 任意一个所述子图包括所述计算图中的至少一个所述算子; 根据所述多个子图编译生成多个可执行任务; 基于所述多个可执行任务运行所述计算图。本申请可以解决动态shape网络模型的软件编译和高效执行, 实现了动态shape下以计算图为核心的程序执行方法。

说明书

发明名称: 计算图的处理方法和装置

[0001] 本申请要求于2022年10月19日提交中国专利局、申请号为202211277782.4、申请名称为“计算图的处理方法和装置”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

[0002] 本申请涉及神经网络技术，尤其涉及一种计算图的处理方法和装置。

背景技术

[0003] 动态shape的某一个或者某几个维度 $\text{dim} = -1$ ，这在编译时是未知的，只有实际运行时才知道具体的dim值。例如： $[10, -1, 20, 30]$ 和 $[10, -1, -1, 30]$ ，其中分别有一个或一个以上的维度是 -1 。与此相对应的，静态shape的每一个维度均是已知值，例如， $[10, 10, 20, 30]$ 。

[0004] 神经网络模型中，当输入输出是动态shape时，如何高效的编译和运行神经网络模型，是业界努力解决的一个课题。

[0005] 发明内容

[0006] 本申请提供一种计算图的处理方法和装置，以解决动态shape网络模型的软件编译和高效执行，实现了动态shape下以计算图为核心的程序执行方法。

[0007] 第一方面，本申请提供一种计算图的处理方法，包括：获取待编译的计算图，所述计算图的输入数据采用动态shape，所述计算图包括多个算子；对所述计算图进行划分以得到多个子图，任意一个所述子图包括所述计算图中的至少一个所述算子；根据所述多个子图编译生成多个可执行任务；基于所述多个可执行任务运行所述计算图。

[0008] 本申请实施例，在动态shape下，以子图为基础调度执行单位的图调度软件解决方案(区别以node为单位的stream调度机制)，以解决动态shape网络模型的软件编译和高效执行，实现了动态shape下以计算图为核心的程序执行方法。

[0009] 计算图的input/output采用动态shape，即，input/output中，用于表示像素组成结构的shape为动态shape，其中有一个或多个维度的值为 -1 ，在编译阶段该 -1 对

应的维度是未知的。计算图可以包括多个算子(亦即Node)，例如图1所示计算图中，每一层对应一个算子，则该计算图包括11个算子。

[0010] 根据算子的特性，可以对input/output进行切分，例如，卷积算子可以支持对input/outputinput/output进行连续多等分切分，也可以支持对input/outputinput/output进行跳跃切分，这样可以将input/outputinput/output切分成多份数据，从而减少每一份的数据量，再增加算子的个数，通过并发执行提升执行效率。

[0011] 本申请实施例划分子图的策略可以包括将切分方式相同且连续的算子归入同一子图中，也可以将切分方式不同且连续的算子归入同一个子图。以切分方式相同为例，计算图共有10个算子1-10，其中，算子1-3所支持的切分方式相同，那么算子1-3可以归入同一个子图，算子4和6虽然切分方式相同，但是算子4和6不是连续的，因此只能将算子4归入一个子图，算子5归入另一个子图，算子6归入又一个子图，算子7-10所支持的切分方式相同，那么算子7-10又可以归入同一个子图。由此可知，计算机划分得到的多个子图中，可以有只包含一个算子的子图，也可以有包含多个算子的子图，其中当第一子图包括n个算子时，n个算子连续排列，且n个算子所支持的输入数据切分方式相同， $n \geq 1$ ，第一子图为多个子图中的任意一个。

[0012] 本申请实施例中，可以对多个子图分别进行静态编译，以得到多个线程任务；然后获取待处理数据；根据待处理数据对多个线程任务进行动态编译，以得到多个可执行任务。由此可见，对计算图的编译过程可以包括静态编译和动态编译两部分，其中，

[0013] 静态编译过程中：获取第一子图内的引擎总数，第一子图为多个子图中的任意一个，第一子图包括m个算子， $m \geq 1$ ；根据第一子图内的引擎总数确定N， $N > 1$ 且 $\geq$ 引擎总数，N表示可并发运行的线程数量；获取N个线程任务，任意一个线程任务包括m个算子的切分块，N个线程任务对应N个线程。

[0014] 本申请实施例中，可以在编译态根据硬件资源情况，将计算图划分成几个子图，实现shape无关的计算图静态处理，生成线程任务。将静态图相关的执行步骤提前在运行前执行，可以降低运行时编译开销。

- [0015] 可选的，可以在第一子图的开头和结尾插入连接算子。连接算子可以包括in label, AT-start, AT-end, outlabel等，这些算子可以自定义，只是达成连接功能。
- [0016] 可选的，可以对第一子图包括的m个算子进行优化。该优化可以包括各种融合优化，单算子优化，常量折叠优化，dtype, format优化等。
- [0017] 可选的，可以对N个线程任务进行缓存操作。可以对子图内的多个算子，进行Cache操作(例如，prefetch, invalid, writeback等)。
- [0018] 动态编译过程中：获取待处理数据的动态shape；根据待处理数据的动态shape对多个线程任务中的未知参数进行更新，以得到多个可执行任务。
- [0019] 本申请实施例中，可以根据实际的动态shape计算线程任务的并发参数，并更新线程任务中的动态shape和线程(thread)并发相关参数(例如刷新线程任务里跟shape中-1相关的部分)，最终生成可执行的task，生成最终的可执行任务。动态编译和动态执行在运行时，可以将host任务和device任务在host和device流水执行，以提升执行效率。
- [0020] 此外，可以根据待处理数据的动态shape实时计算获取threadnum，threadnum表示待处理数据的切分数量。
- [0021] 在一种可能的实现方式中，将动态shape代入预设公式以得到threadnum，预设公式用于表示动态shape和threadnum之间的对应关系。
- [0022] 亦可以称作cache策略，在子图并发执行时，所有数据不出cache，利用cache的带宽提升子图执行性能。子图内所有node thread并发时，每个thread上节点的input+output+workspace+prefetch的内存消耗最大值之和 $\leq$ cache大小。由于编译态不知道shape实际大小，设置shape为变量使用拟合或者插值等方式，预估出L2cache资源的变量公式，以便在运行获取实际shape时，快速带入变量公式，算出切分的threadnum的最小值。
- [0023] 在一种可能的实现方式中，以第一子图内的引擎满负荷工作为目的获取threadnum。
- [0024] 亦可以称作计算资源策略，在子图并发执行时，各个引擎满规格跑计算资源。同一类型引擎的blockdim=芯片core的整数倍；引擎bound的计算资源满负荷

跑，引擎非bound的计算资源不阻塞bound计算资源的运行；在编译态，根据拟合的shape，评估出引擎并发的最佳流水，给出符合最佳流水的threadnum的建议值；结合cache资源的策略，在运行态实际综合出threadnum值。

- [0025] 以复用可执行任务的方式，基于多个可执行任务调度第一子图包括的threadnum个子任务，threadnum表示第一子图的待处理数据的切分数量，每个子任务包括m个算子， $m \geq 1$ ，第一子图为多个子图中的任意一个。
- [0026] 本申请实施例以子图为单位下发可执行任务，并发调度执行以子图为单位的任务，可以提升资源的并发效率。
- [0027] 在一种可能的实现方式中，以运行总时长最短为目的，给多个可执行任务分配带宽。
- [0028] 亦可以称作Qos策略，根据上文最佳流水调整thread subtask的Qos值，子图并发执行时，通过协调各计算资源和带宽的优先级，保证整图的高效执行。
- [0029] 第二方面，本申请提供一种处理装置，包括：获取模块，用于获取待编译的计算图，所述计算图的输入数据采用动态shape，所述计算图包括多个算子；划分模块，用于对所述计算图进行划分以得到多个子图，任意一个所述子图包括所述计算图中的至少一个所述算子；编译模块，用于根据所述多个子图编译生成多个可执行任务；运行模块，用于基于所述多个可执行任务运行所述计算图。
- [0030] 在一种可能的实现方式中，所述划分模块，具体用于获取所述多个算子的切分信息，所述切分信息用于指示对应算子所支持的输入数据切分方式；根据所述多个算子的切分信息获取所述多个子图。
- [0031] 在一种可能的实现方式中，当第一子图包括n个算子时，所述n个算子连续排列，且所述n个算子所支持的输入数据切分方式相同， $n > 1$ ，所述第一子图为所述多个子图中的任意一个。
- [0032] 在一种可能的实现方式中，所述编译模块，具体用于对所述多个子图分别进行静态编译，以得到多个线程任务；获取待处理数据；根据所述待处理数据对所述多个线程任务进行动态编译，以得到所述多个可执行任务。
- [0033] 在一种可能的实现方式中，所述编译模块，具体用于获取第一子图内的引擎总数，所述第一子图为所述多个子图中的任意一个，所述第一子图包括m个算子

, $m \geq 1$; 根据所述第一子图内的引擎总数确定 N , $N > 1$, N 表示可并发运行的线程数量; 获取 N 个线程任务, 任意一个所述线程任务包括 m 个结构体, 所述 N 个线程任务对应 N 个线程。

[0034] 在一种可能的实现方式中, 所述编译模块, 具体用于获取所述待处理数据的动态shape; 根据所述待处理数据的动态shape对所述多个线程任务中的未知参数进行更新, 以得到所述多个可执行任务。

[0035] 在一种可能的实现方式中, 所述编译模块, 还用于根据所述待处理数据的动态shape获取threadnum, threadnum表示所述待处理数据的切分数量。

[0036] 在一种可能的实现方式中, 所述编译模块, 具体用于将所述动态shape代入预设公式以得到所述threadnum, 所述预设公式用于表示所述动态shape和所述threadnum之间的对应关系。

[0037] 在一种可能的实现方式中, 所述编译模块, 具体用于以所述第一子图内的引擎满负荷工作为目的获取所述threadnum。

[0038] 在一种可能的实现方式中, 所述编译模块, 还用于在所述第一子图的开头和结尾插入连接算子。

[0039] 在一种可能的实现方式中, 所述编译模块, 还用于对所述第一子图包括的所述 m 个算子进行优化。

[0040] 在一种可能的实现方式中, 所述编译模块, 还用于对所述 N 个线程任务进行缓存操作。

[0041] 在一种可能的实现方式中, 所述运行模块, 具体用于以复用可执行任务的方式, 基于所述多个可执行任务调度第一子图包括的threadnum个子任务, threadnum表示所述第一子图的待处理数据的切分数量, 每个所述子任务包括 m 个算子, $m \geq 1$, 所述第一子图为所述多个子图中的任意一个。

[0042] 在一种可能的实现方式中, 所述运行模块, 还用于以运行总时长最短为目的, 给所述多个可执行任务分配带宽。

[0043] 第三方面, 本申请提供一种设备, 包括: 一个或多个处理器; 存储器, 用于存储一个或多个程序; 当所述一个或多个程序被所述一个或多个处理器执行, 使得所述一个或多个处理器实现如上述第一方面中任一项所述的方法。

[0044] 第四方面，本申请提供一种计算机可读存储介质，包括计算机程序，所述计算机程序在计算机上被执行时，使得所述计算机执行上述第一方面中任一项所述的方法。

[0045] 第五方面，本申请提供一种计算机程序产品，所述计算机程序产品包括计算机程序代码，当所述计算机程序代码在计算机上运行时，使得计算机执行上述第一方面中任一项所述的方法。

附图说明

[0046] 图1为计算图的一个示例性的示意图；

[0047] 图2为本申请的硬件系统的一个示例性的结构图；

[0048] 图3为本申请计算图的处理方法的过程300的流程图；

[0049] 图4为子图切分的示意图；

[0050] 图5为连接算子示意图；

[0051] 图6为动态编译示意图；

[0052] 图7为子图运行的实施过程示意图；

[0053] 图8为本申请实施例的处理装置800的结构示意图。

具体实施方式

[0054] 为使本申请的目的、技术方案和优点更加清楚，下面将结合本申请中的附图，对本申请中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

[0055] 本申请的说明书实施例和权利要求书及附图中的术语“第一”、“第二”等仅用于区分描述的目的，而不能理解为指示或暗示相对重要性，也不能理解为指示或暗示顺序。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含，例如，包含了一系列步骤或单元。方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0056] 应当理解,在本申请中,“至少一个(项)”是指一个或者多个,“多个”是指两个或两个以上。“和/或”,用于描述关联对象的关联关系,表示可以存在三种关系,例如,“A和/或B”可以表示:只存在A,只存在B以及同时存在A和B三种情况,其中A,B可以是单数或者复数。字符“/”一般表示前后关联对象是一种“或”的关系。“以下至少一项(个)”或其类似表达,是指这些项中的任意组合,包括单项(个)或复数项(个)的任意组合。例如,a,b或c中的至少一项(个),可以表示:a,b,c,“a和b”,“a和c”,“b和c”,或“a和b和c”,其中a,b,c可以是单个,也可以是多个。

[0057] 本申请实施例涉及神经网络的应用,为了便于理解,下面先对相关名词或术语进行解释说明:

[0058] 1、神经网络

[0059] 神经网络(neural network, NN)是机器学习模型,神经网络可以由神经单元组成的,神经单元可以是指以 x_s 和截距1为输入的运算单元,该运算单元的输出可以为:

$$h_{w,b}(x) = f(W^T x) = f(\sum_{s=1}^n W_s x_s + b) \quad (1-1)$$

[0060] 其中, $s=1, 2, \dots, n$, n 为大于1的自然数, W_s 为 x_s 的权重, b 为神经单元的偏置。 f 为神经单元的激活函数(activation functions),用于将非线性特性引入神经网络中,来将神经单元中的输入信号转换为输出信号。该激活函数的输出信号可以作为下一层卷积层的输入。激活函数可以是ReLU等非线性函数。神经网络是将许多个上述单一的神经单元联结在一起形成的网络,即一个神经单元的输出可以是另一个神经单元的输入。每个神经单元的输入可以与前一层的局部感受野(local receptive field)相连,来提取局部感受野的特征,局部感受野可以由若干个神经单元组成的区域。

[0061] 2、多层感知器(multi-layer perception, MLP)

[0062] MLP是一种简单的深度神经网络(deep neural network, DNN)(不同层之间是全连接的),也称多层神经网络,可以理解为具有很多层隐含层的神经网络,这里的“很多”并没有特别的度量标准。从DNN按不同层的位置划分,DNN内部的神经网络可以分为三类:输入层,隐含层,输出层。一般来说第一层是输入

层，最后一层是输出层，中间的层数都是隐含层。层与层之间是全连接的，也就是说，第*i*层的任意一个神经元一定与第*i*+1层的任意一个神经元相连。虽然DNN看起来很复杂，但是就每一层的工作来说，其实并不复杂，简单来说就是如下线性关系表达式： $\vec{y} = \alpha(\mathbf{W}\vec{x} + \vec{b})$ ，其中， $\vec{x}$ 是输入向量， $\vec{y}$ 是输出向量， $\vec{b}$ 是偏移向量， $\mathbf{W}$ 是权重矩阵(也称系数)， $\alpha()$ 是激活函数。每一层仅仅是对输入向量 $\vec{x}$ 经过如此简单的操作得到输出向量 $\vec{y}$ 。由于DNN层数多，则系数 $\mathbf{W}$ 和偏移向量 $\vec{b}$ 的数量也就很多了。这些参数在DNN中的定义如下所述：以系数 $\mathbf{W}$ 为例：假设在一个三层的DNN中，第二层的第4个神经元到第三层的第2个神经元的线性系数定义为 w_{24}^3 。上标3代表系数 $\mathbf{W}$ 所在的层数，而下标对应的是输出的第三层索引2和输入的第二层索引4。总结就是：第*L*-1层的第*k*个神经元到第*L*层的第*j*个神经元的系数定义为 w_{jk}^L 。需要注意的是，输入层是没有 $\mathbf{W}$ 参数的。在深度神经网络中，更多的隐含层让网络更能够刻画现实世界中的复杂情形。理论上而言，参数越多的模型复杂度越高，“容量”也就越大，也就意味着它能完成更复杂的学习任务。训练深度神经网络的也就是学习权重矩阵的过程，其最终目的是得到训练好的深度神经网络的所有层的权重矩阵(由很多层的向量 $\mathbf{W}$ 形成的权重矩阵)。

[0063] 3、卷积神经网络

[0064] 卷积神经网络(convolutional neuron network, CNN)是一种带有卷积结构的深度神经网络，是一种深度学习(deep learning)架构，深度学习架构是指通过机器学习的算法，在不同的抽象层级上进行多个层次的学习。作为一种深度学习架构，CNN是一种前馈(feed-forward)人工神经网络，该前馈人工神经网络中的各个神经元可以对输入其中的图像作出响应。卷积神经网络包含了一个由卷积层和池化层构成的特征抽取器。该特征抽取器可以看作是滤波器，卷积过程可以看作是使用一个可训练的滤波器与一个输入的图像或者卷积特征平面(feature map)做卷积。

[0065] 卷积层是指卷积神经网络中对输入信号进行卷积处理的神经元层。卷积层可以包括很多个卷积算子，卷积算子也称为核，其在图像处理中的作用相当于一

个从输入图像矩阵中提取特定信息的过滤器，卷积算子本质上可以是一个权重矩阵，这个权重矩阵通常被预先定义，在对图像进行卷积操作的过程中，权重矩阵通常在输入图像上沿着水平方向一个像素接着一个像素(或两个像素接着两个像素……这取决于步长stride的取值)的进行处理，从而完成从图像中提取特定特征的工作。该权重矩阵的大小应该与图像的大小相关，需要注意的是，权重矩阵的纵深维度(depth dimension)和输入图像的纵深维度是相同的，在进行卷积运算的过程中，权重矩阵会延伸到输入图像的整个深度。因此，和一个单一的权重矩阵进行卷积会产生一个单一纵深维度的卷积化输出，但是大多数情况下不使用单一权重矩阵，而是应用多个尺寸(行×列)相同的权重矩阵，即多个同型矩阵。每个权重矩阵的输出被堆叠起来形成卷积图像的纵深维度，这里的维度可以理解为由上面所述的“多个”来决定。不同的权重矩阵可以用来提取图像中不同的特征，例如一个权重矩阵用来提取图像边缘信息，另一个权重矩阵用来提取图像的特定颜色，又一个权重矩阵用来对图像中不需要的噪点进行模糊化等。该多个权重矩阵尺寸(行×列)相同，经过该多个尺寸相同的权重矩阵提取后的特征图的尺寸也相同，再将提取到的多个尺寸相同的特征图合并形成卷积运算的输出。这些权重矩阵中的权重值在实际应用中需要经过大量的训练得到，通过训练得到的权重值形成的各个权重矩阵可以用来从输入图像中提取信息，从而使得卷积神经网络进行正确的预测。当卷积神经网络有多个卷积层的时候，初始的卷积层往往提取较多的一般特征，该一般特征也可以称之为低级别的特征；随着卷积神经网络深度的加深，越往后的卷积层提取到的特征越来越复杂，比如高级别的语义之类的特征，语义越高的特征越适用于待解决的问题。

[0066] 由于常常需要减少训练参数的数量，因此卷积层之后常常需要周期性的引入池化层，可以是一层卷积层后面跟一层池化层，也可以是多层卷积层后面接一层或多层池化层。在图像处理过程中，池化层的唯一目的就是减少图像的空间大小。池化层可以包括平均池化算子和/或最大池化算子，以用于对输入图像进行采样得到较小尺寸的图像。平均池化算子可以在特定范围内对图像中的像素值进行计算产生平均值作为平均池化的结果。最大池化算子可以在特定范围内

取该范围内值最大的像素作为最大池化的结果。另外，就像卷积层中用权重矩阵的大小应该与图像尺寸相关一样，池化层中的运算符也应该与图像的大小相关。通过池化层处理后输出的图像尺寸可以小于输入池化层的图像的尺寸，池化层输出的图像中每个像素点表示输入池化层的图像的对应子区域的平均值或最大值。

[0067] 在经过卷积层/池化层的处理后，卷积神经网络还不足以输出所需要的输出信息。因为如前所述，卷积层/池化层只会提取特征，并减少输入图像带来的参数。然而为了生成最终的输出信息(所需要的类信息或其他相关信息)，卷积神经网络需要利用神经网络层来生成一个或者一组所需要的类的数量的输出。因此，在神经网络层中可以包括多层隐含层，该多层隐含层中所包含的参数可以根据具体的任务类型的相关训练数据进行预先训练得到，例如该任务类型可以包括图像识别，图像分类，图像超分辨率重建等等。

[0068] 可选的，在神经网络层中的多层隐含层之后，还包括整个卷积神经网络的输出层，该输出层具有类似分类交叉熵的损失函数，具体用于计算预测误差，一旦整个卷积神经网络的前向传播完成，反向传播就会开始更新前面提到的各层的权重值以及偏差，以减少卷积神经网络的损失，及卷积神经网络通过输出层输出的结果和理想结果之间的误差。

[0069] 4、循环神经网络

[0070] 循环神经网络(recurrent neural networks, RNN)是用来处理序列数据的。在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，而对于每一层层内之间的各个节点是无连接的。这种普通的神经网络虽然解决了很多难题，但是却仍然对很多问题却无能无力。例如，你要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。RNN之所以称为循环神经网络，即一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐含层本层之间的节点不再无连接而是有连接的，并且隐含层的输入不仅包括输入层的输出还包括上一时刻隐含层的输出。理论上，RNN能够对任何长度的序列数据进行处理。对于RNN的训练和对传统的CNN或DNN的

训练一样。同样使用误差反向传播算法，不过有一点区别：即，如果将RNN进行网络展开，那么其中的参数，如 W ，是共享的；而如上举例上述的传统神经网络却不是这样。并且在使用梯度下降算法中，每一步的输出不仅依赖当前步的网络，还依赖前面若干步网络的状态。该学习算法称为基于时间的反向传播算法(Back propagation Through Time, BPTT)。

[0071] 既然已经有了卷积神经网络，为什么还要循环神经网络？原因很简单，在卷积神经网络中，有一个前提假设是：元素之间是相互独立的，输入与输出也是独立的，比如猫和狗。但现实世界中，很多元素都是相互连接的，比如股票随时间的变化，再比如一个人说了：我喜欢旅游，其中最喜欢的地方是云南，以后有机会一定要去。这里填空，人类应该都知道是填“云南”。因为人类会根据上下文的内容进行推断，但如何让机器做到这一步？RNN就应运而生了。RNN旨在让机器像人一样拥有记忆的能力。因此，RNN的输出就需要依赖当前的输入信息和历史的记忆信息。

[0072] 5、损失函数

[0073] 在训练深度神经网络的过程中，因为希望深度神经网络的输出尽可能的接近真正想要预测的值，所以可以通过比较当前网络的预测值和真正想要的目标值，再根据两者之间的差异情况来更新每一层神经网络的权重向量(当然，在第一次更新之前通常会有初始化的过程，即为深度神经网络中的各层预先配置参数)，比如，如果网络的预测值高了，就调整权重向量让它预测低一些，不断的调整，直到深度神经网络能够预测出真正想要的目标值或与真正想要的目标值非常接近的值。因此，就需要预先定义“如何比较预测值和目标值之间的差异”，这便是损失函数(loss function)或目标函数(objective function)，它们是用于衡量预测值和目标值的差异的重要方程。其中，以损失函数举例，损失函数的输出值(loss)越高表示差异越大，那么深度神经网络的训练就变成了尽可能缩小这个loss的过程。

[0074] 6、反向传播算法

[0075] 卷积神经网络可以采用误差反向传播(back propagation, BP)算法在训练过程中修正初始的超分辨率模型中参数的大小，使得超分辨率模型的重建误差损失越

来越小。具体地，前向传递输入信号直至输出会产生误差损失，通过反向传播误差损失信息来更新初始的超分辨率模型中参数，从而使误差损失收敛。反向传播算法是以误差损失为主导的反向传播运动，旨在得到最优的超分辨率模型的参数，例如权重矩阵。

[0076] 7、生成式对抗网络

[0077] 生成式对抗网络(generative adversarial networks, GAN)是一种深度学习模型。该模型中至少包括两个模块：一个模块是生成模型(Generative Model)，另一个模块是判别模型(Discriminative Model)，通过这两个模块互相博弈学习，从而产生更好的输出。生成模型和判别模型都可以是神经网络，具体可以是深度神经网络，或者卷积神经网络。GAN的基本原理如下：以生成图片的GAN为例，假设有两个网络，G(Generator)和D(Discriminator)，其中G是一个生成图片的网络，它接收一个随机的噪声 z ，通过这个噪声生成图片，记做 $G(z)$ ；D是一个判别网络，用于判别一张图片是不是“真实的”。它的输入参数是 x ， x 代表一张图片，输出 $D(x)$ 代表 x 为真实图片的概率，如果为1，就代表100%是真实的图片，如果为0，就代表不可能是真实的图片。在对该生成式对抗网络进行训练的过程中，生成网络G的目标就是尽可能生成真实的图片去欺骗判别网络D，而判别网络D的目标就是尽量把G生成的图片和真实的图片区分开来。这样，G和D就构成了一个动态的“博弈”过程，也即“生成式对抗网络”中的“对抗”。最后博弈的结果，在理想的状态下，G可以生成足以“以假乱真”的图片 $G(z)$ ，而D难以判定G生成的图片究竟是不是真实的，即 $D(G(z))=0.5$ 。这样就得到了一个优异的生成模型G，它可以用来生成图片。

[0078] 8、计算图

[0079] 本申请实施例中，神经网络模型可以使用计算图表达各层(layer)以及layer之间的连接关系。图1为计算图的一个示例性的示意图，如图1所示，该计算图包括6个卷积层，5个激活层ReLU，卷积核的尺寸为 3×3 。亦即，该计算图总共包括11个算子(亦称作节点(Node))，一个算子对应前述一个层(例如卷积层或激活层)。按照从左向右的顺序，前一层的输出可以作为后一层的输入，其中第一个算子的输入亦即该计算图的输入，最后一个算子的输出亦即该计算图的输出。

需要说明的是，计算图可以采用上文任意一种神经网络的拓扑结构，对此不做具体限定。

[0080] 计算图在使用中可以实现一些功能，例如，识别图像、智能翻译、智能推荐等。而在实施前述功能之前，需要对计算图进行编译，生成可执行表达，进而在硬件环境中运行前述可执行表达，以执行计算图的功能。需要说明的是，可执行表达可以是文件，也可以是程序的数据结构等，本申请实施例对此不做具体限定。

[0081] 9、动态shape

[0082] 本申请实施例中，计算图的输入/输出可以采用动态shape。动态shape(亦称作dynamic tensor/dynamic shape/unknown shape)，其表示输入/输出的shape中，某一个或者某几个维度为-1(即 $\text{dim}=-1$)，动态shape在编译时是未知的，只有在实际运行时才知道具体的dim值。例如：

[0083] 1)[10,-1,20,30],[10,-1,-1,30]，其中有一个或一个以上的维度是-1，则是动态shape；

[0084] 2)[10,10,20,30]，其中每一个维度都是已知值，则是静态(known)shape。

[0085] 在计算机图中，可以使用shape的概念，用以表示输入/输出(input/output)在内存中的排布，亦可以理解为是input/outputinput/output的组成结构，本申请实施例中，input/output可以是图片，也可以是语音、文字等，对此不做具体限定。示例性的，以图片输入为例，二维图片的尺寸可以表示成 100×50 ，三维图片用红绿蓝表示可以表示成 $3 \times 100 \times 50$ ，此外加上图片的数量100，可以表示成 $100 \times 3 \times 100 \times 50$ 。shape可以认为是输入的图片的像素点在内存中的排布结构。

[0086] 动态shape，可以理解为每次需要处理的图片的像素是不同的，例如，这次是 100×50 ，下次是 10×5 ，再下次是 1000×500 。也可以理解为每次需要处理的语句不同，例如，句子的长度、字符等不同。以及，待处理对象的规格是不固定的。

[0087] 示例性的，100张图片，每张图片由红黄蓝组成，尺寸为 100×50 ，则可以表示成shape[100,3,100,50]，此时计算图的输入数据包括实际数据(所有像素点的像素值)，以及前述shape(表示所有多像素点的排放结构)。计算图的输出数据也包括

实际数据(所有像素点的像素值), 以及shape(表示所有多像素点的排放结构), 此时shape与输入数据的shape可以相同, 也可以不同, 例如, shape[50,3,100,50]。

[0088] 图2为本申请的硬件系统的一个示例性的结构图, 如图2所示, 主机(host)上的处理程序将编译好的模型下发到设备(device), 由device上的神经网络处理器/芯片/中央处理器(central processing unit, CPU)进行神经网络模型的执行。模型的编译可以离线在其他个人计算机(personal computer, PC)机器上进行编译, 也可以在host机器上进行编译。

[0089] 需要说明的是, 本申请的host和device可以集成于同一个电子设备中, 共同完成存储器操作, 该电子设备具备存储能力; host和device也可以分别设置于不同的电子设备上, 例如, host设置于手机、平板电脑等用户设备上, device设置于神经网络处理器上。

[0090] 图3为本申请计算图的处理方法的过程300的流程图, 如图3所示, 过程300可以由上述主机和设备共同执行。过程300描述为一系列的步骤或操作, 应当理解的是, 过程300可以以各种顺序执行和/或同时发生, 不限于图3所示的执行顺序。过程300包括如下步骤:

[0091] 步骤301、获取待编译的计算图。

[0092] 本申请实施例中, 计算图的输入/输出(input/output)采用动态shape, 即, input/output中, 用于表示像素组成结构的shape为动态shape, 其中有一个或多个维度的值为-1, 在编译阶段该-1对应的维度是未知的。

[0093] 如上文所述, 计算图可以包括多个算子(亦即Node), 例如图1所示计算图中, 每一层对应一个算子, 则该计算图包括11个算子。

[0094] 步骤302、对计算图进行划分以得到多个子图。

[0095] 本申请实施例中, 可以获取计算图中包括的多个算子的切分信息, 该切分信息用于指示对应算子所支持的input/output切分方式; 再根据多个算子的切分信息特征划分成多个子图。

[0096] 根据算子的特性, 可以对input/output进行切分, 例如, 卷积算子可以支持对input/output进行连续多等分切分, 也可以支持对input/output进行跳跃切分, 这

样可以将input/output切分成多份数据，从而减少每一份的数据量，再增加算子的个数，通过并发执行提升执行效率。

[0097] 本申请实施例划分子图的策略可以包括将切分方式相同且连续的算子归入同一子图中，也可以将切分方式不同且连续的算子归入同一个子图。以切分方式相同为例，计算图共有10个算子1-10，其中，算子1-3所支持的切分方式相同，那么算子1-3可以归入同一个子图，算子4和6虽然切分方式相同，但是算子4和6不是连续的，因此只能将算子4归入一个子图，算子5归入另一个子图，算子6归入又一个子图，算子7-10所支持的切分方式相同，那么算子7-10又可以归入同一个子图。由此可知，计算机划分得到的多个子图中，可以有只包含一个算子的子图，也可以有包含多个算子的子图，其中当第一子图包括 n 个算子时， n 个算子连续排列，且 n 个算子所支持的输入数据切分方式相同， $n \geq 1$ ，第一子图为多个子图中的任意一个。

[0098] 步骤303、根据多个子图编译生成多个可执行任务。

[0099] 本申请实施例中，可以对多个子图分别进行静态编译，以得到多个线程任务；然后获取待处理数据；根据待处理数据对多个线程任务进行动态编译，以得到多个可执行任务。由此可见，对计算图的编译过程可以包括静态编译和动态编译两部分，其中，

[0100] 静态编译过程中：获取第一子图内的引擎总数，第一子图为多个子图中的任意一个，第一子图包括 m 个算子， $m \geq 1$ ；根据第一子图内的引擎总数确定 N ， $N > 1$ 且 $\geq$ 引擎总数， N 表示可并发运行的线程数量；获取 N 个线程任务，任意一个线程任务包括 m 个算子的切分块， N 个线程任务对应 N 个线程。

[0101] 本申请实施例中，可以在编译态根据硬件资源情况，将计算图划分成几个子图，实现shape无关的计算图静态处理，生成线程任务。将静态图相关的执行步骤提前在运行前执行，可以降低运行时编译开销。

[0102] 可选的，可以在第一子图的开头和结尾插入连接算子。连接算子可以包括in label, AT-start, AT-end, outlabel等，这些算子可以自定义，只是达成连接功能。

- [0103] 可选的，可以对第一子图包括的m个算子进行优化。该优化可以包括各种融合优化，单算子优化，常量折叠优化，dtype，format优化等。
- [0104] 可选的，可以对N个线程任务进行缓存操作。可以对子图内的多个算子，进行缓存(Cache)操作(例如，prefetch，invalid，writeback等)。
- [0105] 动态编译过程中：获取待处理数据的动态shape；根据待处理数据的动态shape对多个线程任务中的未知参数进行更新，以得到多个可执行任务。
- [0106] 本申请实施例中，可以根据实际的动态shape计算线程任务的并发参数，并更新线程任务中的动态shape和线程(thread)并发相关参数(例如刷新线程任务里跟shape中-1相关的部分)，最终生成可执行的task，生成最终的可执行任务。动态编译和动态执行在运行时，可以将host任务和device任务在host和device流水执行，以提升执行效率。
- [0107] 此外，可以根据待处理数据的动态shape实时计算获取待处理数据的切分数量(threadnum)。
- [0108] 在一种可能的实现方式中，将动态shape代入预设公式以得到threadnum，预设公式用于表示动态shape和threadnum之间的对应关系。
- [0109] 亦可以称作cache策略，在子图并发执行时，所有数据不出cache，利用cache的带宽提升子图执行性能。子图内所有node thread并发时，每个thread上节点的input+output+workspace+prefetch的内存消耗最大值之和 $\leq$ cache大小。由于编译态不知道shape实际大小，设置shape为变量使用拟合或者插值等方式，预估出L2cache资源的变量公式，以便在运行获取实际shape时，快速带入变量公式，算出切分的threadnum的最小值。
- [0110] 在一种可能的实现方式中，以第一子图内的引擎满负荷工作为目的获取threadnum。
- [0111] 亦可以称作计算资源策略，在子图并发执行时，各个引擎满规格跑计算资源。同一类型引擎的blockdim=芯片core的整数倍；引擎bound的计算资源满负荷跑，引擎非bound的计算资源不阻塞bound计算资源的运行；在编译态，根据拟合的shape，评估出引擎并发的最佳流水，给出符合最佳流水的threadnum的建议值；结合cache资源的策略，在运行态实际综合出threadnum值。

[0112] 步骤304、基于多个可执行任务运行计算图。

[0113] 以复用可执行任务的方式，基于多个可执行任务调度第一子图包括的threadnum个子任务，threadnum表示第一子图的待处理数据的切分数量，每个子任务包括m个算子， $m \geq 1$ ，第一子图为多个子图中的任意一个。

[0114] 本申请实施例以子图为单位下发可执行任务，并发调度执行以子图为单位的任务，可以提升资源的并发效率。

[0115] 在一种可能的实现方式中，以运行总时长最短为目的，给多个可执行任务分配带宽。

[0116] 亦可以称作Qos策略，根据上文最佳流水调整thread subtask的Qos值，子图并发执行时，通过协调各计算资源和带宽的优先级，保证整图的高效执行。

[0117] 本申请实施例，在动态shape下，以子图为基础调度执行单位的图调度软件解决方案(区别以node为单位的stream调度机制)，以解决动态shape网络模型的软件编译和高效执行，实现了动态shape下以计算图为核心的程序执行方法。

[0118] 以下采用一个具体的实施例对上述方案进行说明。

[0119] 一、静态编译

[0120] 静态编译中，圈定子图范围(scope)策略，给定一个计算图，根据切分信息确定scope范围(根据切分信息找到一致切分轴的范围)，按照自动线程化(threading)方式(连续等分方式，在不能均分的情况下，尾快可以和非尾快不相同)进行切分，固化窗口尺寸(window size)大小(运行时devcie上同时执行的thread数)，将threadnum作为变量，对资源进行拟合评估，在编译态评估出快速公式，在运行态根据实际的shape代入快速公式算出切分的threadnum；评估资源包括但不限于：cache资源，计算资源等。

[0121] 图4为子图切分的示意图，如图4所示，该示例中的子图包括4个算子(从上到下依次为conv、vec、conv、vec)，该子图的input/output可以切分成8份(份数的获取方式可参照下文)，则可以由8个thread处理8份数据的处理，每个线程对应的算子相同。

[0122] 1、预估策略：针对划定的scope子图范围，在编译态进行预估，减少运行态计算时长消耗：

[0123] (1)固化thread window size = N (N ≥ 子图内引擎总数)

[0124] 通过thread windows size个thread表达自动threading下的线程子任务结构(thread subtask context)任务，即便在不知道实际threadnum情况下，利用自动threading的方法将子图级task表达出来。超过或低于thread windows size时，在动态编译时对子图内的task进行调整。

[0125] (2)cache策略：子图并发时，所有数据不出cache，利用cache的带宽提升子图执行性能。子图内所有节点(node)thread并发时，每个thread上节点的input+output+workspace+prefetch的内存消耗最大值之和 ≤ cache大小。由于编译态不知道shape实际大小，设置shape为变量使用拟合或者插值等方式，预估出L2cache资源的变量公式，以便在运行获取实际shape时，快速带入变量公式，算出切分的threadnum的最小值。由于从内存读写性能比较差，希望输入输出可以从cache进行读写，提高读写效率。

[0126] (3)计算资源策略：子图并发执行时，各个引擎满规格跑计算资源。同一类型引擎的blockdim = 芯片core的整数倍；引擎bound的计算资源满负荷跑，引擎非bound的计算资源不阻塞bound计算资源的运行；在编译态，根据拟合的shape，评估出引擎并发的最佳流水，给出符合最佳流水的threadnum的建议值；结合cache资源的策略，在运行态实际综合出threadnum值。由于各个引擎的满负荷度不一样，可以找出最佳的一种情况，该方法可以包括寻优找到最佳值，应理解还可以采用其他方法(例如，对所有任务进行流水排布方式；根据引擎数确定计算资源并发数，并发thread数 ≥ 引擎数，等策略)，对此不做限定。

[0127] 上述(2)和(3)是和/或关系，即可以只采用(2)或(3)得到threadnum，也可以采用(2)和(3)得到threadnum。

[0128] 2、Task编译和生成

[0129] (1)以子图为单位，进行子图内算子优化，包括各种融合优化，单算子优化，常量折叠优化，dtype，format优化等；

[0130] (2)对动态shape子图内的多thread节点，进行Cache操作(prefetch, invalid, writeback)；

[0131] (3)虽然thread数尚未确定，发明了自动threading技术，

[0132] a)在子图的开头和结尾插入inlabel, AT-start, AT-end, outlabel等连接算子。如图5(图5为连接算子示意图)所示, inlabel和outlabel表示子图的开始和结束, 算子名称可以任意命名, AT-start, AT-end是每个thread的开始和结束算子, AT-start, AT-end描述符中包含threadid(第几个thread)和threadnum(总共几个thread), outlabel里记录了总共需要执行多少各threadnum, 每执行完一个thread, 对outlabel中减1, 当outlabel中threadnum减为0时, 该子图执行完成。

[0133] b)只需要表达thread window size个thread, 多余的thread数=threadnum-thread window size, 通过执行时复用, 从而可以在编译态固定表达出threadnum个task任务。当某一个thread执行完成后, 将threadnum-thread window size中的一个thread替换进thread window size里执行完成的thread, 以达成复用目的。

[0134] (4)shape相关的参数更新动态编译中实现。

[0135] 二、动态编译

[0136] 如图6所示, 动态编译

[0137] (1)Host侧Subgraph子图间任务并发流水。子图在host先进行动态编译, 完成推导shape信息, 根据前述算法确定子图内thread数, 分配内存, 更新子图任务中在编译态尚未获取的信息(只有在动态编译, 获取到实际shape时, 才可以计算出实际的值), 将device需要的信息通过host to device进行搬运, 并发起线程任务的launch动态, 激发device开始执行任务。Host上多个子图间可以流水的方式并发执行。在前一个子图inferredshape出最后一个node的shape后, 下一个子图就获取到了第一个node的shape, 这样第二个子图的host就开始执行下去了。因此多个子图可以流水执行下去。

[0138] (2)Host侧subgraph子图内node任务并发流水。子图内存在多个算子, 每个算子在host执行的操作类似, 只是下一个算子需要依赖上一个算子的shape, 因此在获取到上一个算子的shape后, 下一个算子就可以流水执行下去。

[0139] (3)执行时host侧刷新subgraph内动态shape在编译态未知部分。动态shape在编译态时, 因为未知shape, 所以很多执行的信息无法确定, 在拿到shape后, 可以将执行信息的实际值刷新到task里面。

[0140] (4)动态shape算子(下一个算子的shape依赖以前一个算子的输出值才可以确定,此时这个算子称为动态shape算子)不放在子图范围内,需要通过host和device的交互获取shape等信息。以单算子或者独立子图的方式执行。

[0141] 三、动态执行

[0142] 如图7所示,通过软硬件配合的方式完成子图的执行过程,软件负责子图的调度,具体流程如下:

[0143] (1)硬件将子图预加载进cache,并通知device调度软件启动执行,device调度软件读取初始就绪(前序节点计数为0)的任务。

[0144] (2)device调度软件将就绪的任务推送到硬件执行装置中执行。

[0145] (3)硬件执行装置从cache中读取任务描述符,执行就绪任务,包括调度到多个异构引擎的执行。

[0146] (4)硬件执行子图任务完成,通知mcu软件。

[0147] (5)device调度软件取当前执行完成任务的后序任务,将后序任务的前序节点计数--,当计数为0,推送该任务到硬件执行装置执行。

[0148] (6)2~5循环往复,直到子图执行完成。

[0149] 对于子图编译提供的thread windows size个task任务序列,device调度在执行完某个thread task任务序列后,将整个thread替换新的thread,并修改新thread context里的thread id等信息。直至所有thread执行完成。

[0150] 此外,根据上面的最佳流水调整thread subtask的Qos值,子图并发执行时,通过协调各计算资源和带宽的优先级,保证整图的高效执行,8个thread可以同时并行,这样会抢内存的带宽,QoS可以按照策略分配带宽,带宽是指计算机有很多总线,总线带宽越高,性能越好。

[0151] 图8为本申请实施例的处理装置800的结构示意图。该处理装置800可以用于上文的硬件系统,其包括:获取模块801、划分模块802、编译模块803和运行模块804。其中,

[0152] 获取模块801,用于获取待编译的计算图,所述计算图的输入数据采用动态shape,所述计算图包括多个算子;划分模块802,用于对所述计算图进行划分以得到多个子图,任意一个所述子图包括所述计算图中的至少一个所述算子;编

译模块803，用于根据所述多个子图编译生成多个可执行任务；运行模块804，用于基于所述多个可执行任务运行所述计算图。

[0153] 在一种可能的实现方式中，所述划分模块802，具体用于获取所述多个算子的切分信息，所述切分信息用于指示对应算子所支持的输入数据切分方式；根据所述多个算子的切分信息获取所述多个子图。

[0154] 在一种可能的实现方式中，当第一子图包括 n 个算子时，所述 n 个算子连续排列，且所述 n 个算子所支持的输入数据切分方式相同， $n > 1$ ，所述第一子图为所述多个子图中的任意一个。

[0155] 在一种可能的实现方式中，所述编译模块803，具体用于对所述多个子图分别进行静态编译，以得到多个线程任务；获取待处理数据；根据所述待处理数据对所述多个线程任务进行动态编译，以得到所述多个可执行任务。

[0156] 在一种可能的实现方式中，所述编译模块803，具体用于获取第一子图内的引擎总数，所述第一子图为所述多个子图中的任意一个，所述第一子图包括 m 个算子， $m \geq 1$ ；根据所述第一子图内的引擎总数确定 N ， $N > 1$ ， N 表示可并发运行的线程数量；获取 N 个线程任务，任意一个所述线程任务包括 m 个结构体，所述 N 个线程任务对应 N 个线程。

[0157] 在一种可能的实现方式中，所述编译模块803，具体用于获取所述待处理数据的动态shape；根据所述待处理数据的动态shape对所述多个线程任务中的未知参数进行更新，以得到所述多个可执行任务。

[0158] 在一种可能的实现方式中，所述编译模块803，还用于根据所述待处理数据的动态shape获取threadnum，threadnum表示所述待处理数据的切分数量。

[0159] 在一种可能的实现方式中，所述编译模块803，具体用于将所述动态shape代入预设公式以得到所述threadnum，所述预设公式用于表示所述动态shape和所述threadnum之间的对应关系。

[0160] 在一种可能的实现方式中，所述编译模块803，具体用于以所述第一子图内的引擎满负荷工作为目的获取所述threadnum。

[0161] 在一种可能的实现方式中，所述编译模块803，还用于在所述第一子图的开头和结尾插入连接算子。

- [0162] 在一种可能的实现方式中，所述编译模块803，还用于对所述第一子图包括的所述 m 个算子进行优化。
- [0163] 在一种可能的实现方式中，所述编译模块803，还用于对所述 N 个线程任务进行缓存操作。
- [0164] 在一种可能的实现方式中，所述运行模块804，具体用于以复用可执行任务的方式，基于所述多个可执行任务调度第一子图包括的 threadnum 个子任务， threadnum 表示所述第一子图的待处理数据的切分数量，每个所述子任务包括 m 个算子， $m \geq 1$ ，所述第一子图为所述多个子图中的任意一个。
- [0165] 在一种可能的实现方式中，所述运行模块804，还用于以运行总时长最短为目的，给所述多个可执行任务分配带宽。
- [0166] 本实施例的装置，可以用于执行图3所示方法实施例的技术方案，其实现原理和技术效果类似，此处不再赘述。
- [0167] 在实现过程中，上述方法实施例的各步骤可以通过处理器中的硬件的集成电路或者软件形式的指令完成。处理器可以是通用处理器、数字信号处理器(digital signal processor,DSP)、特定应用集成电路(application-specific integrated circuit, ASIC)、现场可编程门阵列(field programmable gate array,FPGA)或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。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。本申请实施例公开的方法的步骤可以直接体现为硬件编码处理器执行完成，或者用编码处理器中的硬件及软件模块组合执行完成。软件模块可以位于随机存储器，闪存、只读存储器，可编程只读存储器或者电可擦写可编程存储器、寄存器等本领域成熟的存储介质中。该存储介质位于存储器，处理器读取存储器中的信息，结合其硬件完成上述方法的步骤。
- [0168] 上述各实施例中提及的存储器可以是易失性存储器或非易失性存储器，或可包括易失性和非易失性存储器两者。其中，非易失性存储器可以是只读存储器(read-only memory, ROM)、可编程只读存储器(programmable ROM, PROM)、可擦除可编程只读存储器(erasable PROM, EPROM)、电可擦除可编程只读存储器(electrically EPROM, EEPROM)或闪存。易失性存储器可以是随机存

取存储器(random access memory, RAM), 其用作外部高速缓存。通过示例性但不是限制性说明, 许多形式的RAM可用, 例如静态随机存取存储器(static RAM, SRAM)、动态随机存取存储器(dynamic RAM, DRAM)、同步动态随机存取存储器(synchronous DRAM, SDRAM)、双倍数据速率同步动态随机存取存储器(double data rate SDRAM, DDR SDRAM)、增强型同步动态随机存取存储器(enhanced SDRAM, ESDRAM)、同步连接动态随机存取存储器(synclink DRAM, SLDRAM)和直接内存总线随机存取存储器(direct rambus RAM, DR RAM)。应注意, 本文描述的系统和方法的存储器旨在包括但不限于这些和任意其它适合类型的存储器。

[0169] 本领域普通技术人员可以意识到, 结合本文中所公开的实施例描述的各示例的单元及算法步骤, 能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行, 取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能, 但是这种实现不应认为超出本申请的范围。

[0170] 所属领域的技术人员可以清楚地了解到, 为描述的方便和简洁, 上述描述的系统、装置和单元的具体工作过程, 可以参考前述方法实施例中的对应过程, 在此不再赘述。

[0171] 在本申请所提供的几个实施例中, 应该理解到, 所揭露的系统、装置和方法, 可以通过其它的方式实现。例如, 以上所描述的装置实施例仅仅是示意性的, 例如, 所述单元的划分, 仅仅为一种逻辑功能划分, 实际实现时可以有另外的划分方式, 例如多个单元或组件可以结合或者可以集成到另一个系统, 或一些特征可以忽略, 或不执行。另一点, 所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口, 装置或单元的间接耦合或通信连接, 可以是电性, 机械或其它的形式。

[0172] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的, 作为单元显示的部件可以是或者也可以不是物理单元, 即可以位于一个地方, 或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

- [0173] 另外，在本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。
- [0174] 所述功能如果以软件功能单元的形式实现并作为独立的产品销售或使用，可以存储在一个计算机可读取存储介质中。基于这样的理解，本申请的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备(个人计算机，服务器，或者网络设备等)执行本申请各个实施例所述方法的全部或部分步骤。而前述的存储介质包括：U盘、移动硬盘、只读存储器(read-only memory, ROM)、随机存取存储器(random access memory, RAM)、磁碟或者光盘等各种可以存储程序代码的介质。
- [0175] 以上所述，仅为本申请的具体实施方式，但本申请的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本申请揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应以所述权利要求的保护范围为准。

权利要求书

- [权利要求 1] 一种计算图的处理方法，其特征在于，包括：
获取待编译的计算图，所述计算图的输入数据采用动态shape，所述计算图包括多个算子；
对所述计算图进行划分以得到多个子图，任意一个所述子图包括所述计算图中的至少一个所述算子；
根据所述多个子图编译生成多个可执行任务；
基于所述多个可执行任务运行所述计算图。
- [权利要求 2] 根据权利要求1所述的方法，其特征在于，所述对所述计算图进行划分以得到多个子图，包括：
获取所述多个算子的切分信息，所述切分信息用于指示对应算子所支持的输入数据切分方式；
根据所述多个算子的切分信息获取所述多个子图。
- [权利要求 3] 根据权利要求2所述的方法，其特征在于，当第一子图包括 n 个算子时，所述 n 个算子连续排列，且所述 n 个算子所支持的输入数据切分方式相同， $n > 1$ ，所述第一子图为所述多个子图中的任意一个。
- [权利要求 4] 根据权利要求1-3中任一项所述的方法，其特征在于，所述根据所述多个子图编译生成多个可执行任务，包括：
对所述多个子图分别进行静态编译，以得到多个线程任务；
获取待处理数据；
根据所述待处理数据对所述多个线程任务进行动态编译，以得到所述多个可执行任务。
- [权利要求 5] 根据权利要求4所述的方法，其特征在于，所述对所述多个子图分别进行静态编译，以得到多个线程任务，包括：
获取第一子图内的引擎总数，所述第一子图为所述多个子图中的任意一个，所述第一子图包括 m 个算子， $m \geq 1$ ；
根据所述第一子图内的引擎总数确定 N ， $N > 1$ ， N 表示可并发运行的线程数量；

获取N个线程任务，任意一个所述线程任务包括m个结构体，所述N个线程任务对应N个线程。

[权利要求 6] 根据权利要求4或5所述的方法，其特征在于，所述根据所述待处理数据对所述多个线程任务进行动态编译，以得到所述多个可执行任务，包括：

获取所述待处理数据的动态shape；

根据所述待处理数据的动态shape对所述多个线程任务中的未知参数进行更新，以得到所述多个可执行任务。

[权利要求 7] 根据权利要求6所述的方法，其特征在于，还包括：

根据所述待处理数据的动态shape获取threadnum，threadnum表示所述待处理数据的切分数量。

[权利要求 8] 根据权利要求7所述的方法，其特征在于，所述根据所述待处理数据的动态shape获取threadnum，包括：

将所述动态shape代入预设公式以得到所述threadnum，所述预设公式用于表示所述动态shape和所述threadnum之间的对应关系。

[权利要求 9] 根据权利要求7或8所述的方法，其特征在于，所述根据所述待处理数据的动态shape获取threadnum，包括：

以所述第一子图内的引擎满负荷工作为目的获取所述threadnum。

[权利要求 10] 根据权利要求5-9中任一项所述的方法，其特征在于，还包括：在所述第一子图的开头和结尾插入连接算子。

[权利要求 11] 根据权利要求5-10中任一项所述的方法，其特征在于，还包括：对所述第一子图包括的所述m个算子进行优化。

[权利要求 12] 根据权利要求5-11中任一项所述的方法，其特征在于，还包括：对所述N个线程任务进行缓存操作。

[权利要求 13] 根据权利要求1-12中任一项所述的方法，其特征在于，所述基于所述多个可执行任务运行所述计算图，包括：
以复用可执行任务的方式，基于所述多个可执行任务调度第一子图包括的threadnum个子任务，threadnum表示所述第一子图的待处理

数据的切分数量，每个所述子任务包括 m 个算子， $m \geq 1$ ，所述第一子图为所述多个子图中的任意一个。

[权利要求 14]

根据权利要求13所述的方法，其特征在于，还包括：

以运行总时长最短为目的，给所述多个可执行任务分配带宽。

[权利要求 15]

一种处理装置，其特征在于，包括：

获取模块，用于获取待编译的计算图，所述计算图的输入数据采用动态shape，所述计算图包括多个算子；

划分模块，用于对所述计算图进行划分以得到多个子图，任意一个所述子图包括所述计算图中的至少一个所述算子；

编译模块，用于根据所述多个子图编译生成多个可执行任务；

运行模块，用于基于所述多个可执行任务运行所述计算图。

[权利要求 16]

根据权利要求15所述的装置，其特征在于，所述划分模块，具体用于获取所述多个算子的切分信息，所述切分信息用于指示对应算子所支持的输入数据切分方式；根据所述多个算子的切分信息获取所述多个子图。

[权利要求 17]

根据权利要求16所述的装置，其特征在于，当第一子图包括 n 个算子时，所述 n 个算子连续排列，且所述 n 个算子所支持的输入数据切分方式相同， $n > 1$ ，所述第一子图为所述多个子图中的任意一个。

[权利要求 18]

根据权利要求15-17中任一项所述的装置，其特征在于，所述编译模块，具体用于对所述多个子图分别进行静态编译，以得到多个线程任务；获取待处理数据；根据所述待处理数据对所述多个线程任务进行动态编译，以得到所述多个可执行任务。

[权利要求 19]

根据权利要求18所述的装置，其特征在于，所述编译模块，具体用于获取第一子图内的引擎总数，所述第一子图为所述多个子图中的任意一个，所述第一子图包括 m 个算子， $m \geq 1$ ；根据所述第一子图内的引擎总数确定 N ， $N > 1$ ， N 表示可并发运行的线程数量；获取 N 个线程任务，任意一个所述线程任务包括 m 个结构体，所述 N 个线程任务对应 N 个线程。

- [权利要求 20] 根据权利要求18或19所述的装置，其特征在于，所述编译模块，具体用于获取所述待处理数据的动态shape；根据所述待处理数据的动态shape对所述多个线程任务中的未知参数进行更新，以得到所述多个可执行任务。
- [权利要求 21] 根据权利要求20所述的装置，其特征在于，所述编译模块，还用于根据所述待处理数据的动态shape获取threadnum，threadnum表示所述待处理数据的切分数量。
- [权利要求 22] 根据权利要求21所述的装置，其特征在于，所述编译模块，具体用于将所述动态shape代入预设公式以得到所述threadnum，所述预设公式用于表示所述动态shape和所述threadnum之间的对应关系。
- [权利要求 23] 根据权利要求21或22所述的装置，其特征在于，所述编译模块，具体用于以所述第一子图内的引擎满负荷工作为目的获取所述threadnum。
- [权利要求 24] 根据权利要求19–23中任一项所述的装置，其特征在于，所述编译模块，还用于在所述第一子图的开头和结尾插入连接算子。
- [权利要求 25] 根据权利要求19–24中任一项所述的装置，其特征在于，所述编译模块，还用于对所述第一子图包括的所述m个算子进行优化。
- [权利要求 26] 根据权利要求19–25中任一项所述的装置，其特征在于，所述编译模块，还用于对所述N个线程任务进行缓存操作。
- [权利要求 27] 根据权利要求15–26中任一项所述的装置，其特征在于，所述运行模块，具体用于以复用可执行任务的方式，基于所述多个可执行任务调度第一子图包括的threadnum个子任务，threadnum表示所述第一子图的待处理数据的切分数量，每个所述子任务包括m个算子， $m \geq 1$ ，所述第一子图为所述多个子图中的任意一个。
- [权利要求 28] 根据权利要求27所述的装置，其特征在于，所述运行模块，还用于以运行总时长最短为目的，给所述多个可执行任务分配带宽。
- [权利要求 29] 一种设备，其特征在于，包括：
一个或多个处理器；

存储器，用于存储一个或多个程序；

当所述一个或多个程序被所述一个或多个处理器执行，使得所述一个或多个处理器实现如权利要求1-14中任一项所述的方法。

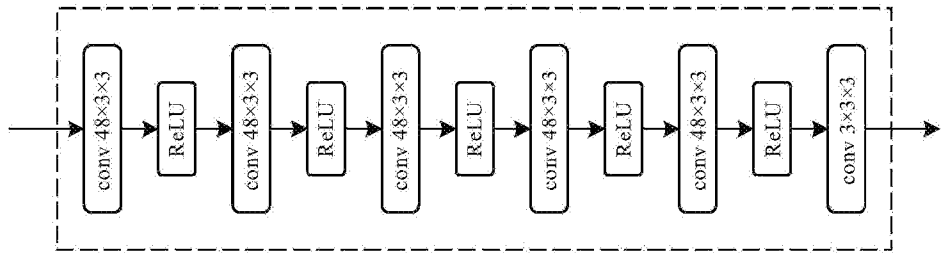
[权利要求 30]

一种计算机可读存储介质，其特征在于，包括计算机程序，所述计算机程序在计算机上被执行时，使得所述计算机执行权利要求1-14中任一项所述的方法。

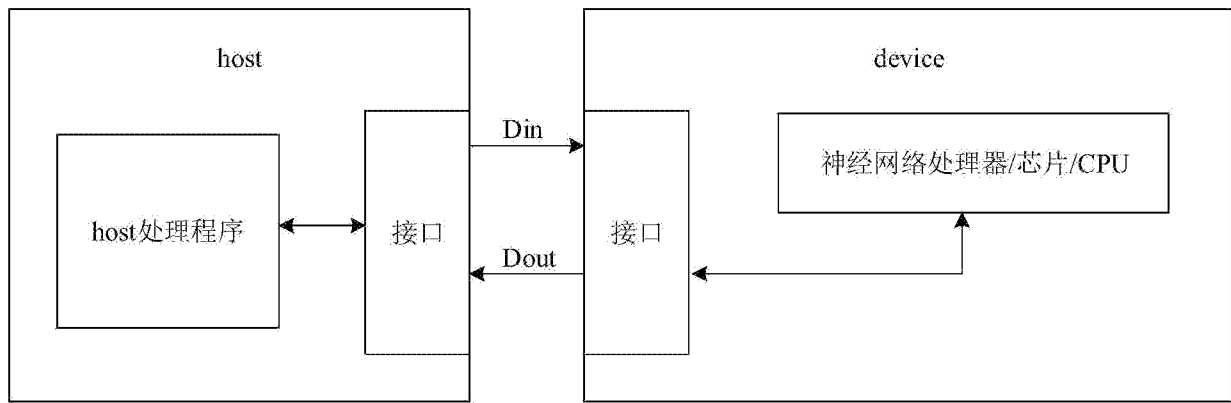
[权利要求 31]

一种计算机程序产品，其特征在于，所述计算机程序产品包括计算机程序代码，当所述计算机程序代码在计算机上运行时，使得计算机执行权利要求1-14中任一项所述的方法。

[图 1]

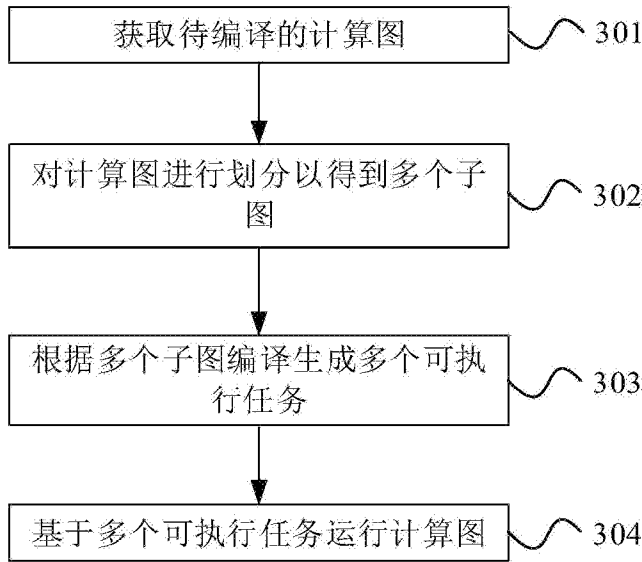


[图 2]

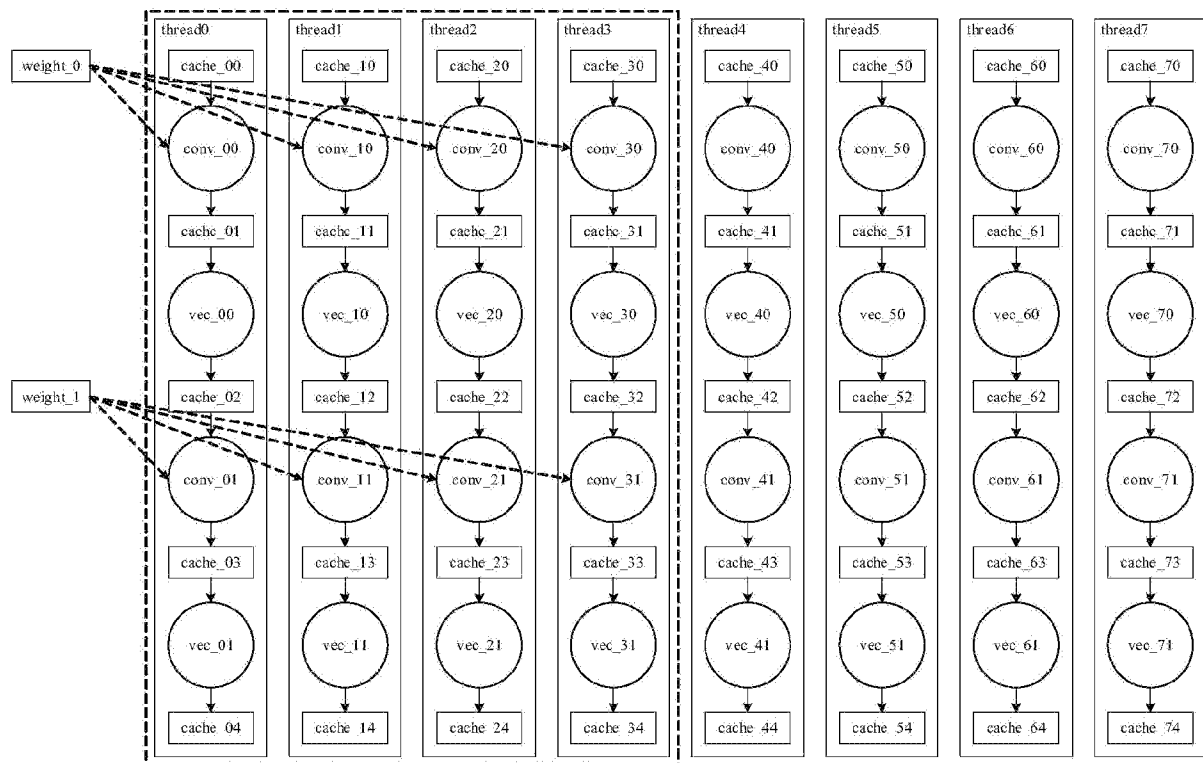


[图 3]

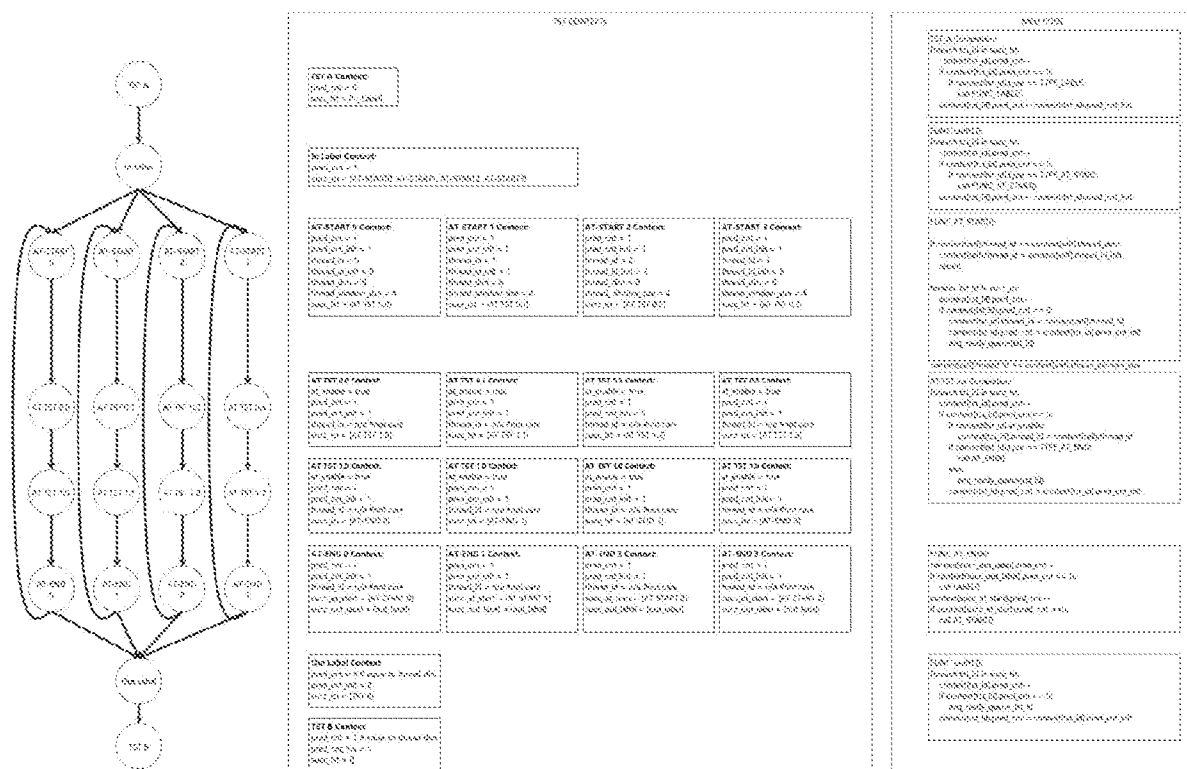
300



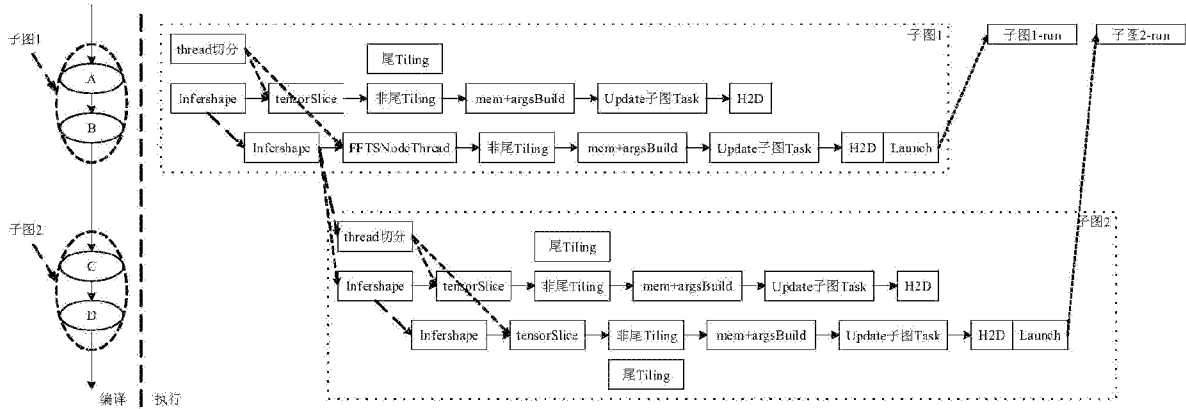
[图 4]



[图 5]

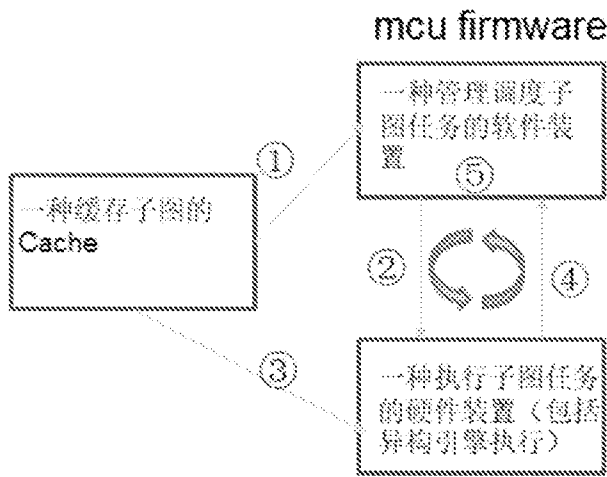


[图 6]

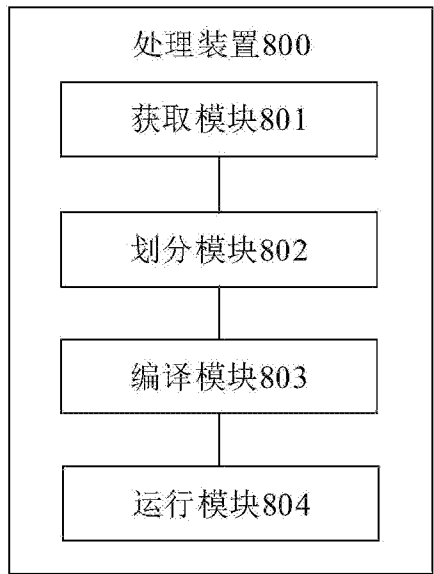


细则 26,
28.08.2023

[图 7]



[图 8]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2023/102301

A. CLASSIFICATION OF SUBJECT MATTER

G06N3/04(2023.01)i; G06N3/08(2023.01)i; G06F8/30(2018.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06N G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS, VEN, CNTXT, WOTXT, USTXT, EPTXT, CNKI, IEEE: 计算图, 算子, 子图, 编译, 输入数据, 张量, 动态, 形状, 优化, 调度, 线程, graph, sub graph, operator, compile, input data, tensor, shape, optimization, thread

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 114580653 A (ALIBABA CLOUD COMPUTING LTD.) 03 June 2022 (2022-06-03) description, paragraphs 28-62 and 68-72	1-31
A	CN 114385181 A (PHYTIUM TECHNOLOGY CO., LTD.) 22 April 2022 (2022-04-22) entire document	1-31
A	CN 111338635 A (TENCENT TECHNOLOGY (SHENZHEN) CO., LTD.) 26 June 2020 (2020-06-26) entire document	1-31
A	US 2019303762 A1 (XILINX, INC.) 03 October 2019 (2019-10-03) entire document	1-31



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“D” document cited by the applicant in the international application

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

04 August 2023

Date of mailing of the international search report

17 August 2023

Name and mailing address of the ISA/CN

China National Intellectual Property Administration (ISA/
CN)
China No. 6, Xitucheng Road, Jimenqiao, Haidian District,
Beijing 100088

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2023/102301

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	114580653	A	03 June 2022	None			
CN	114385181	A	22 April 2022	None			
CN	111338635	A	26 June 2020	None			
US	2019303762	A1	03 October 2019	US	11514324	B2	29 November 2022
				CN	110321999	A	11 October 2019

国际检索报告

国际申请号
PCT/CN2023/102301

A. 主题的分类 G06N3/04(2023.01)i; G06N3/08(2023.01)i; G06F8/30(2018.01)i 按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类																	
B. 检索领域 检索的最低限度文献(标明分类系统和分类号) IPC: G06N G06F 包含在检索领域中的除最低限度文献以外的检索文献 在国际检索时查阅的电子数据库(数据库的名称, 和使用的检索词(如使用)) CNABS, VEN, CNTXT, WOTXT, USTXT, EPTXT, CNKI, IEEE: 计算图, 算子, 子图, 编译, 输入数据, 张量, 动态, 形状, 优化, 调度, 线程, graph, sub graph, operator, compile, input data, tensor, shape, optimization, thread																	
C. 相关文件 <table> <tr> <th>类 型*</th> <th>引用文件, 必要时, 指明相关段落</th> <th>相关的权利要求</th> </tr> <tr> <td>X</td> <td>CN 114580653 A (阿里云计算有限公司) 2022年6月3日 (2022 - 06 - 03) 说明书第28-62, 68-72段</td> <td>1-31</td> </tr> <tr> <td>A</td> <td>CN 114385181 A (飞腾信息技术有限公司) 2022年4月22日 (2022 - 04 - 22) 全文</td> <td>1-31</td> </tr> <tr> <td>A</td> <td>CN 111338635 A (腾讯科技(深圳)有限公司) 2020年6月26日 (2020 - 06 - 26) 全文</td> <td>1-31</td> </tr> <tr> <td>A</td> <td>US 2019303762 A1 (XILINX, INC.) 2019年10月3日 (2019 - 10 - 03) 全文</td> <td>1-31</td> </tr> </table>			类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求	X	CN 114580653 A (阿里云计算有限公司) 2022年6月3日 (2022 - 06 - 03) 说明书第28-62, 68-72段	1-31	A	CN 114385181 A (飞腾信息技术有限公司) 2022年4月22日 (2022 - 04 - 22) 全文	1-31	A	CN 111338635 A (腾讯科技(深圳)有限公司) 2020年6月26日 (2020 - 06 - 26) 全文	1-31	A	US 2019303762 A1 (XILINX, INC.) 2019年10月3日 (2019 - 10 - 03) 全文	1-31
类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求															
X	CN 114580653 A (阿里云计算有限公司) 2022年6月3日 (2022 - 06 - 03) 说明书第28-62, 68-72段	1-31															
A	CN 114385181 A (飞腾信息技术有限公司) 2022年4月22日 (2022 - 04 - 22) 全文	1-31															
A	CN 111338635 A (腾讯科技(深圳)有限公司) 2020年6月26日 (2020 - 06 - 26) 全文	1-31															
A	US 2019303762 A1 (XILINX, INC.) 2019年10月3日 (2019 - 10 - 03) 全文	1-31															
☐ 其余文件在C栏的续页中列出。 ☒ 见同族专利附件。																	
* 引用文件的具体类型: “A” 认为不特别相关的表示了现有技术一般状态的文件 “D” 申请人在国际申请中引证的文件 “E” 在国际申请日的当天或之后公布的在先申请或专利 “L” 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件(如具体说明的) “O” 涉及口头公开、使用、展览或其他方式公开的文件 “P” 公布日先于国际申请日但迟于所要求的优先权日的文件 “T” 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件 “X” 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性 “Y” 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性 “&” 同族专利的文件																	
国际检索实际完成的日期 2023年8月4日		国际检索报告邮寄日期 2023年8月17日															
ISA/CN的名称和邮寄地址 中国国家知识产权局 中国北京市海淀区蓟门桥西土城路6号 100088		授权官员 宋朝 电话号码 (+86) 010-53961349															

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2023/102301

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
CN	114580653	A	2022年6月3日	无			
CN	114385181	A	2022年4月22日	无			
CN	111338635	A	2020年6月26日	无			
US	2019303762	A1	2019年10月3日	US	11514324	B2	2022年11月29日
				CN	110321999	A	2019年10月11日