

计算流图调度方案的生成方法、装置、电子设备及计算机可读存储介质

本申请要求了 2021 年 6 月 3 日提交的、申请号为 202110620358.4、发明名称为“计算流图调度方案的生成方法、装置、电子设备及计算机可读存储介质”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

本公开涉及计算流图调度领域，尤其涉及一种计算流图调度方案的生成方法、装置、电子设备及计算机可读存储介质

背景技术

深度学习 (DL) 模型可以表示为有向无环图 (DAG)，图中的顶点表示模型中的计算操作，有向边表示不同计算操作之间的数据流。

在硬件上部署 DL 模型一般可分为两种场景：训练模型和推理模型。在两种场景中均需要决定执行 DL 模型的调度方案，调度方案内容包括：DAG 中顶点的执行顺序，各个顶点被执行时使用的计算设备和资源量，顶点被执行后产出的数据使用的存储设备和资源量等。

在训练模型的场景中，一般使用如 HBM(High Bandwidth Memory, 高带宽内存)等带宽很高的存储介质，数据传输速度一般不构成性能瓶颈。而在推理模型的场景中，由于推理芯片一般使用 DDR (Double Data Rate SDRAM, 双倍速率同步动态随机存储器)等带宽相对有限的存储介质，数据传输速度就成为影响推理性能的一个重要因素。

当前对 DL 模型的计算调度算法的主要方向有：

顶点融合：将计算图中具有数据依赖关系的顶点融合成为一个顶点，即上游顶点的输出数据会在产出之后直接给到下游顶点被消耗掉，而不会在存储资源上做缓存操作，这样就可以减少原有两顶点之间的数据搬运耗时。但是该方案一般是通过专家经验将 DAG 中指定类型的计算顶点进行融合，并根据融合结果对 DAG 进行重写，基于重写得到 DAG 的拓扑排序进行顶点计算顺序的排布，非常依赖专家经验，并且不是对所有模型结构都适用。

多设备分配：根据顶点的计算和存储特性，将其指定到不同的计算设备和存储设备上执行，以提高各设备的计算利用率，并减少设备间数据搬运的消耗。但是该方案不改变 DAG 中原有顶点的计算顺序，也不能提高模型执行过程中的计算并行度。

顶点复制：对计算需求低但存储需求较大的顶点结果进行重新计算，来为其他更常被复用的顶点输出数据预留更多的高速缓存空间，以此减少整个 DAG 执行过程中数据在低速缓存和高速缓存之间搬运的总耗时。这种方法相当于在将 DAG 中某个顶点复制之后插入到了其他位置。但是该方案并不能提升模型执行过程中的计算并行度。相反由于在原有 DAG 中加入了新的顶点，还提升了整个模型的计算消耗。

发明内容

提供该发明内容部分以便以简要的形式介绍构思，这些构思将在后面的具体实施

方式部分被详细描述。该发明内容部分并不旨在标识要求保护的技术方案的关键特征或必要特征，也不旨在用于限制所要求的保护的技术方案的范围。

为了解决现有技术中的上述技术问题，本公开实施例提出如下技术方案：

第一方面，本公开实施例提供一种计算流图调度方案的生成方法，包括：

一种计算流图调度方案的生成方法，其特征在于，包括：

将原始计算流图中的原始顶点进行分组得到第一计算流图，所述每个分组作为第一计算流图中的一个顶点，所述顶点为所述原始计算流图中的至少一个原始顶点所形成集合；

根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N ， N 为大于等于 1 的整数；

复制 N 张所述第一计算流图得到第二计算流图；

在所述第二计算流图中加入辅助顶点得到第三计算流图；

根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题；

求解所述整数线性规划问题得到所述第三计算流图的调度方案；

简化所述第三计算流图的调度方案，形成所述第二流图的调度方案。

进一步的，所述将原始计算流图中的原始顶点进行分组得到第一计算流图，包括：

根据所述原始计算流图中的原始顶点的输入数据和输出数据将所述原始计算流图中的原始顶点进行分组得到第一计算流图。

进一步的，所述根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N ，包括：

获取所述第一计算流图的顶点的最大存储需求；

根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N 。

进一步的，所述根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N ，包括：

根据以下公式计算所述计算单元的数量 N ： $2^{\lceil \log_2 \lceil M/m \rceil \rceil}$ ；

其中， M 表示所述最大存储需求， m 表示单个计算单元的存储空间大小。

进一步的，所述将所述第一计算流图复制数量 N 个得到第二计算流图，包括：

将所述第一计算流图复制所述数据 N 个；

将所述数量 N 个第一计算流图组合生成第二计算流图；其中所述第二计算流图用于对多批数据并行处理。

进一步的，所述辅助顶点包括：表示所述原始计算流图的输入数据读取操作的第一辅助顶点、表示所述原始计算流图顶点的中间结果计算操作的第二辅助顶点以及表

示所述第二计算流图中计算终止操作的第三辅助顶点。

进一步的, 所述根据第三计算流图构建与所述第三计算流图对应的整数线性规划问题, 包括:

求 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的取值, 使得以下多项式的取值最小:

$$\sum_{t=1}^T \sum_{i=1}^N (L_{t,i} + S_{t,i}) C_i$$

其中, i 表示第三计算流图中顶点的编号, t 表示时间步骤, $R_{t,i}$ 表示是否在第 t 个时间步骤计算第 i 个顶点的结果; $S_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果存储到低速缓存中; $L_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果从低速缓存上读取到计算单元的缓存中; $F_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果在计算单元的缓存中所占用的空间释放; C_i 表示将第 i 个顶点的计算结果在低速缓存和计算单元的缓存之间传输所需要的消耗; 其中, 所述 $R_{t,i}=0$ 或 1, $S_{t,i}=0$ 或 1, $L_{t,i}=0$ 或 1, $F_{t,i}=0$ 或 1, 其中 0 表示不执行对应的操作, 1 表示执行对应的操作; T 和 N 为大于 1 的整数; 其中, 所述整数线性规划问题还包括所述 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的限制条件, 所述限制条件由所述计算单元的硬件性能确定。

进一步的, 所述求解所述整数线性规划问题得到所述第三计算流图的调度方案, 包括:

对所述整数线性规划问题进行编码;

对所述编码进行求解得到所述第三计算流图中的顶点的执行顺序。

进一步的, 所述简化所述第三计算流图的调度方案, 形成所述第二流图的调度方案, 包括:

将所述第三计算流图的调度方案中的辅助顶点删除得到所述第二流图的调度方案。

进一步的, 所述方法还包括:

根据计算单元的数量以及所述数量 N 确定所述调度方案中每个顶点处理的数据量。

第二方面, 本公开实施例提供一种计算流图调度方案的生成装置, 包括:

第一计算流图生成模块, 用于将原始计算流图中的顶点进行分组得到第一计算流图, 所述每个分组作为第一计算流图中的一个顶点, 所述顶点为所述原始计算流图中的至少一个原始顶点所形成集合;

计算单元数量确定模块,用于根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N , N 为大于等于 1 的整数;

第二计算流图生成模块,用于复制 N 张所述第一计算流图得到第二计算流图;

第三计算流图生成模块,用于在所述第二计算流图中加入辅助顶点得到第三计算流图;

整数线性规划问题构建模块,用于根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题;

整数线性规划问题求解模块,用于求解所述整数线性规划问题得到所述第三计算流图的调度方案;

简化模块,用于简化所述第三计算流图的调度方案,形成所述第二流图的调度方案。

进一步的,所述第一计算流图生成模块,还用于:根据所述原始计算流图中的原始顶点的输入数据和输出数据将所述原始计算流图中的原始顶点进行分组得到第一计算流图。

进一步的,所述计算单元数量确定模块,还用于:获取所述第一计算流图的顶点的最大存储需求;根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N 。

进一步的,所述计算单元数量确定模块,还用于:根据以下公式计算所述计算单元的数量 N : $2^{\lceil \log_2 \lceil M/m \rceil \rceil}$; 其中, M 表示所述最大存储需求, m 表示单个计算单元的存储空间大小。

进一步的,所述第二计算流图生成模块,还用于:将所述第一计算流图复制所述数据 N 个;将所述数量 N 个第一计算流图组合生成第二计算流图;其中所述第二计算流图用于对多批数据并行处理。

进一步的,所述辅助顶点包括:表示所述原始计算流图的输入数据读取操作的第一辅助顶点、表示所述原始计算流图顶点的中间结果计算操作的第二辅助顶点以及表示所述第二计算流图中计算终止操作的第三辅助顶点。

进一步的,所述整数线性规划问题构建模块,还用于:求 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的取值,使得以下多项式的取值最小:

$$\sum_{t=1}^T \sum_{i=1}^N (L_{t,i} + S_{t,i}) C_i$$

其中, i 表示第三计算流图中顶点的编号, t 表示时间步骤, $R_{t,i}$ 表示是否在第 t 个时间步骤计算第 i 个顶点的结果; $S_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计

算结果存储到低速缓存中； $L_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果从低速缓存上读取到计算单元的缓存中； $F_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果在计算单元的缓存中所占用的空间释放； C_i 表示将第 i 个顶点的计算结果在低速缓存和计算单元的缓存之间传输所需要的消耗；其中，所述 $R_{t,i}=0$ 或 1 ， $S_{t,i}=0$ 或 1 ， $L_{t,i}=0$ 或 1 ， $F_{t,i}=0$ 或 1 ，其中 0 表示不执行对应的操作， 1 表示执行对应的操作； T 和 N 为大于 1 的整数；其中，所述整数线性规划问题还包括所述 $R_{t,i}$ ， $S_{t,i}$ ， $L_{t,i}$ 和 $F_{t,i}$ 的限制条件，所述限制条件由所述计算单元的硬件性能确定。

进一步的，所述整数线性规划问题求解模块，还用于：对所述整数线性规划问题进行编码；对所述编码进行求解得到所述第三计算流图中的顶点的执行顺序。进一步的，所述简化模块，还用于：将所述第三计算流图的调度方案中的辅助顶点删除得到所述第二流图的调度方案。

进一步的，所述计算流图调度方案的生成装置，还用于：根据计算单元的数量以及所述数量 N 确定所述调度方案中每个顶点处理的数据量。

第三方面，本公开实施例提供一种电子设备，包括：存储器，用于存储计算机可读指令；以及一个或多个处理器，用于运行所述计算机可读指令，使得所述处理器运行时实现前述第一方面中的任一所述的方法。

第四方面，本公开实施例提供一种计算机可读存储介质，该非暂态计算机可读存储介质存储计算机指令，该计算机指令用于使计算机执行前述第一方面中的任一所述的方法。

第五方面，本公开实施例提供一种计算机程序产品，包括计算机指令，当所述计算机指令被计算设备执行时，所述计算设备可以执行前述第一方面中的任一所述的方法。

本公开实施例公开了一种计算流图调度方案的生成方法、装置、电子设备和计算机可读存储介质。其中该计算流图调度方案的生成方法包括：将原始计算流图中的顶点进行分组得到第一计算流图，其中所述第一计算流图中的顶点为所述原始计算流图中的顶点所形成的至少一个集合；根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N ， N 为大于等于 1 的整数；复制 N 张所述第一计算流图得到第二计算流图；在所述第二计算流图中加入辅助顶点得到第三计算流图；根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题；求解所述整数线性规划问题得到所述第三计算流图的调度方案；简化所述第三计算流图的调度方案，形成所述第二流图的调度方案。上述计算流图调度方案的生成方法通过将原始计算流图转换成第三计算流图并构建整数线性规划问题以求解调度方案，解决了现有技术中数据重用率低或并行度低的技术问题。

上述说明仅是本公开技术方案的概述，为了能更清楚了解本公开的技术手段，而可依照说明书的内容予以实施，并且为了让本公开的上述和其他目的、特征和优点能够

更明显易懂，以下特举较佳实施例，并配合附图，详细说明如下。

附图说明

结合附图并参考以下具体实施方式，本公开各实施例的上述和其他特征、优点及方面将变得更加明显。贯穿附图中，相同或相似的附图标记表示相同或相似的元素。应当理解附图是示意性的，原件和元素不一定按照比例绘制。

图1为本公开实施例中的计算流图调度方案的生成方法的流程示意图；

图2为本公开实施例中的原始计算流图的示例示意图；

图3为本公开实施例中的第一计算流图的示意图；

图4为本公开实施例中的计算流图调度方案的生成方法的进一步流程示意图；

图5为本公开实施例中第二计算流图的示例示意图；

图6为本公开实施例中第三计算流图的示例示意图；

图7为本公开实施例中第三计算流图中的顶点的执行顺序的示意图；

图8为本公开实施例中第二流图的调度方案的示意图。

具体实施方式

下面将参照附图更详细地描述本公开的实施例。虽然附图中显示了本公开的某些实施例，然而应当理解的是，本公开可以通过各种形式来实现，而且不应该被解释为限于这里阐述的实施例，相反提供这些实施例是为了更加透彻和完整地理解本公开。应当理解的是，本公开的附图及实施例仅用于示例性作用，并非用于限制本公开的保护范围。

应当理解，本公开的方法实施方式中记载的各个步骤可以按照不同的顺序执行，和/或并行执行。此外，方法实施方式可以包括附加的步骤和/或省略执行示出的步骤。本公开的范围在此方面不受限制。

本文使用的术语“包括”及其变形是开放性包括，即“包括但不限于”。术语“基于”是“至少部分地基于”。术语“一个实施例”表示“至少一个实施例”；术语“另一实施例”表示“至少一个另外的实施例”；术语“一些实施例”表示“至少一些实施例”。其他术语的相关定义将在下文描述中给出。

需要注意，本公开中提及的“第一”、“第二”等概念仅用于对不同的装置、模块或单元进行区分，并非用于限定这些装置、模块或单元所执行的功能的顺序或者相互依存关系。

需要注意，本公开中提及的“一个”、“多个”的修饰是示意性而非限制性的，本领域技术人员应当理解，除非在上下文另有明确指出，否则应该理解为“一个或多个”。

本公开实施方式中的多个装置之间所交互的消息或者信息的名称仅用于说明性的目的，而并不是用于对这些消息或信息的范围进行限制。

图 1 为本公开实施例提供的计算流图调度方案的生成方法的流程示意图。

所述计算流图调度方案的生成方法，用于生成 DL 模型的计算流图中顶点的执行顺序，各个顶点被执行时使用的计算设备和资源量，顶点被执行后产生的数据使用的存储设备和资源量等。

如图 1 所示，该方法包括如下步骤：

步骤 S101，将原始计算流图中的原始顶点进行分组得到第一计算流图，所述每个分组作为第一计算流图中的一个顶点，所述顶点为所述原始计算流图中的至少一个原始顶点所形成集合。

如图 2 所示，为原始计算流图的一个示例。在该实例中，原始计算流图中包括多个原始顶点，每个原始顶点分别表示一种计算或操作，如卷积运算、激活操作、加法运算、池化操作等等，顶点之间的有向边表示顶点之间的数据流动方向。

在该步骤中，将原始计算流图中的原始顶点根据一定的规则或者按照预设的算法进行分组，被分配在同一组的所述原始顶点融合为一个融合顶点，所述融合顶点作为第一计算流图中的顶点。其中，所述融合顶点为原始计算流图中的至少一个原始顶点所形成的集合，即在所述集合中的所述原始顶点所表示的计算/操作，以及所述原始顶点之间的有向边均作为第一计算流图中的顶点内的计算/操作以及数据流向。

其中，所述将原始计算流图中的原始顶点进行分组得到第一计算流图，包括：根据所述原始计算流图中的所述原始顶点的输入数据和输出数据将所述原始计算流图中的所述原始顶点进行分组得到第一计算流图。在该实施例中，可以根据输入数据和输出数据的依赖关系将原始顶点进行分组得到第一计算流图的一个顶点。

所述分组的标准还可以包括原始顶点的计算资源需求。如连续的原始顶点所需要的计算资源相同，如连续的多个原始顶点所需要的计算资源均为 2 份计算资源(包括计算单元和存储空间等)或 4 份计算资源，则可以将这些连续的多个原始顶点分在一组中以形成第一计算流图的顶点。

所述分组的标准还可以包括原始顶点是否可以并行执行计算或操作，如在原始计算流图中，位于同一个原始顶点之后的两个分支中的原始顶点，如果这两个分支中的原始顶点所需要的计算资源的需求变化不大，则可以将两个分支中的原始顶点分在一组中以形成第一计算流图的顶点。

示例性的，如图 3 所述为 resnet50 的原始顶点分组之后所形成的第一计算流图。其中 resnet50 网络的原始顶点按照预设的标准分为 4 个顶点，分别为 group1、group2、group3 和 group4。所述 4 个顶点具有数据依赖性，即 group1 的输出数据为 group2 的输入数据，group2 的输出数据为 group3 的输入数据，group3 的输出数据为 group4 的输入数据，group4 的输出数据即为 resnet50 的输出数据。

本公开中，对分组的具体标准不做限定，可以采用不同分组标准对应的不同分组算法对所述原始计算流图中的原始顶点进行分组，以得到第一计算流图中的顶点。

返回图 1，所述计算流图调度方案的生成方法，还包括：步骤 S102，根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N，N 为大于等于 1 的整数。

可选的，所述第一计算流图中的顶点的存储资源需求包括所述第一计算流图中的

顶点的各个计算环节的存储资源需求，如输入数据的存储需求、中间计算结果的存储需求以及输出数据的存储需求。

可选的，所述步骤 S102 进一步包括：

步骤 S401，获取所述第一计算流图的顶点的最大存储需求；

步骤 S402，根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N。

在步骤 S401 中，获取第一计算流图的顶点的最大存储需求。可选的，所述最大需求为上述输入数据的存储需求、中间计算结果的存储需求以及输出数据的存储需求中最大的存储需求。

示例性的，如上述 resnet50 的示例，其中各顶点的存储需求如下表所示：

Group	输入数据 (KB)	中间计算结果存储 (KB)	输出数据 (KB)
1	294	3528	784
2	784	1764	392
3	392	882	196
4	196	441	2

如果存储资源能够满足最大的存储需求，则存储资源可以满足其他顶点的存储需求。因此在还步骤中首先获取第一计算流图的顶点的最大存储需求。在上述示例中，最大存储需求为顶点 Group1 的中间计算结果 3528KB。

每个深度学习模型都需要硬件资源对模型进行图调度。在一个示例中，使用申请人自行研发的 NPU STCP920 对 resnet50 模型进行图调度，NPU 上有 8 个可进行高效矩阵乘和卷积运算的计算单元，每个计算单元各有独占的 1280 KB 的一级高速缓存，8 个计算单元共享空间足够大的二级低速缓存。则在示例中，计算单元的存储资源为 1280KB，则在步骤 S402 中，根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N。

其中，可选的，可以根据存储所述最大存储需求所需要的存储资源来确定所述数量 N，如要存储 3528KB 的数据，至少需要 3 个存储资源，则可以确定所需要的计算单元的所述数量 N 为 3。

但是，如果 N=3，则上述 8 个计算单元处理数据时，会有 2 个计算单元闲置。为此，可选的，所述步骤 S102 进一步包括：

根据以下公式计算所述计算单元的数量 N： $2^{\lceil \log_2 \lceil M/m \rceil \rceil}$ ；

其中，M 表示所述最大存储需求，m 表示单个计算单元的存储空间大小。

根据上述示例，最大存储需求为 3528KB，单个计算单元的存储资源为 1280KB，带入上述公式中，可以计算得到 N=4。即，需要 4 个计算单元处理一批数据。

返回图 1，所述计算流图调度方案的生成方法，还包括：步骤 S103，复制 N 张所述第一计算流图得到第二计算流图。

在该步骤中，将步骤 S101 中得到的第一计算流图复制 N 份。所述 N 份第一计算

流图可以并行对 N 批数据进行处理。可以理解的，第一计算流图表示对数据进行处理逻辑，而在实际执行数据处理时，第一计算流图的每个顶点所做的处理由对应的硬件如计算单元来执行。

其中，所述步骤 S103 进一步包括：

将所述第一计算流图复制所述数量 N 个；

将所述 N 个第一计算流图组合生成第二计算流图；其中所述第二计算流图用于对多批数据并行处理。

即在复制出 N 个第一计算流图之后，将 N 个第一计算流图进行组合生成第二计算流图。其个，所述组合包括将所述 N 个第一计算流图的顶点作为所述第二计算流图的顶点，将所述 N 个第一计算流图中的顶点之间的有向边作为所述第二计算流图中的顶点的有向边。如图 5 为第二计算流图的示例示意图。

返回图 1，所述计算流图调度方案的生成方法，还包括：步骤 S104，在所述第二计算流图中加入辅助顶点得到第三计算流图。

其中，所述辅助顶点包括：表示所述原始计算流图的输入数据读取操作的第一辅助顶点、表示对所述原始计算流图顶点的中间结果计算操作的第二辅助顶点以及表示所述第二计算流图中计算终止操作的第三辅助顶点。

在现有的方案中，深度学习模型按照计算操作为顶点的方式产出的原始计算流图只关注输入数据在模型中的计算过程，而忽略了模型本身的参数数据对模型执行过程中对计算和存储需求的影响。在该步骤中，在第二计算流图中添加辅助顶点，为原始计算流图补充了在模型执行过程中的生命周期（如第一辅助顶点表示模型计算的开始、第二辅助顶点表示模型的中间执行过程、第三辅助接点表示模型计算的终止）、存储空间占用等模型参数数据信息，为后续的模型调度方案设计提供更多完备信息。这些信息可以帮助设计者更方便地产出模型调度方案，并减少分析不同模型调度方案的可行性和比较性能优劣上的工作量。

如图 6 所示为第三计算流图的示例示意图。其中，除了第二计算流图中的顶点之外的顶点，均为辅助顶点。如 `batch1 input` 和 `group1 weight` 为表示原始计算流图的输入数据读取操作的第一辅助顶点，其中 `batch1 input` 表示第一份样本数据的输入数据读取，`group1 weight` 表示模型的权重数据的读取。其他第一辅助顶点依此类推，不再赘述。其中 `batch1 group1 internal` 表示第一份样本数据在 `group1` 顶点中的中间结果计算操作，为所述第二辅助顶点，其他第二辅助顶点依此类推，不再赘述。其中，`termination` 表示所述第二计算流图中计算终止操作的第三辅助顶点。

返回图 1，所述计算流图调度方案的生成方法，还包括：步骤 S105，根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题。

整数规划问题中目标函数和约束条件皆为线性的最优化问题。而整数线性规划问题为要求所有的未知量都为整数的线性规划问题（integer linear programming, ILP）。即通过将计算过程中的消耗作为未知量构造目标函数，通过将硬件资源的性能作为约束条件，能够构造出整数线性规划问题，而这个整数线性规划问题的解，即为调度方案。

示例性的，所述步骤 S105 包括：

求 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的取值, 使得以下多项式的取值最小:

$$\sum_{t=1}^T \sum_{i=1}^N (L_{t,i} + S_{t,i}) C_i \quad (\text{公式 1})$$

其中, i 表示第三计算流图中顶点的编号, t 表示时间步骤, $R_{t,i}$ 表示是否在第 t 个时间步骤计算第 i 个顶点的结果; $S_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果存储到低速缓存中; $L_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果从低速缓存上读取到计算单元的缓存中; $F_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果在计算单元的缓存中所占用的空间释放; C_i 表示将第 i 个顶点的计算结果在低速缓存和计算单元的缓存之间传输所需要的消耗; 其中, 所述 $R_{t,i}=0$ 或 1 , $S_{t,i}=0$ 或 1 , $L_{t,i}=0$ 或 1 , $F_{t,i}=0$ 或 1 , 其中 0 表示不执行对应的操作, 1 表示执行对应的操作; T 和 N 为大于 1 的整数; 其中, 所述整数线性规划问题还包括所述 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的限制条件, 所述限制条件由所述计算单元的硬件性能确定。

以上述示例为例, 根据原始计算流图的结构以及所述 NPU STCP920 的硬件特性, 可以得到 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的限制条件如下:

$$\begin{aligned} \sum_t R_{t,i} &= 1, i \in \{1, \dots, N\} \\ \sum_t S_{t,i} &\leq 1, i \in \{1, \dots, N\} \\ S_{t,i} &\leq \sum_{k=1}^t R_{k,i} - \sum_{k=1}^{t-1} F_{k,i}, t \in \{2, \dots, T\}, i \in \{1, \dots, N\} \\ S_{t,i} &\leq \sum_{k=1}^t R_{k,i}, t \in \{2, \dots, T\}, i \in \{1, \dots, N\} \\ L_{t,i} &\leq \sum_{k=1}^{t-1} S_{k,i}, t \in \{2, \dots, T\}, i \in \{1, \dots, N\} \\ 0 &\leq \sum_{k=1}^t (R_{k,i} + L_{k,i} - F_{k,i}) - \sum_{k=1}^{t-1} S_{k,i} \leq 1, t \in \{2, \dots, T-1\}, i \in \{1, \dots, N\} \\ \sum_{k=1}^T (R_{k,i} + L_{k,i} - F_{k,i}) - \sum_{k=1}^{T-1} S_{k,i} &= 0, i \in \{1, \dots, N\} \\ R_{t,j} &\leq \sum_{k=1}^{t-1} R_{k,i} - \sum_{k=1}^{t-1} S_{k,i} + \sum_{k=1}^t L_{k,i} - \sum_{k=1}^{t-1} F_{k,i}, t \in \{1, \dots, T-1\}, i \in \{1, \dots, N\}, \langle i, j \rangle \in DAG3 \\ S_{T,i} &= 0, i \in \{1, \dots, N\} \\ \sum_{i=1}^N \left(\sum_{k=1}^{t-1} (R_{k,i} + S_{k,i} - F_{k,i}) + \sum_{k=1}^t L_{k,i} \right) &\leq B * 1280, t \in \{2, \dots, T\} \end{aligned}$$

$$\sum_{i=1}^N \left(\sum_{k=1}^t (R_{k,i} + S_{k,i} - F_{k,i} + L_{k,i}) \right) \leq B * 1280, t \in \{2, \dots, T\}$$

上述构建整数线性规划问题的方法，使用了二元整数规划问题的构建方法。在实际应用中，还可以使用其他整数规划问题的构建方法。如多元整数规划问题的构建方法，如使用 R_i , S_i , L_i , F_i 来表示对顶点 i 进行对应操作的时间步骤，即 $R_i, S_i, L_i, F_i \in \{0, 1, \dots, T\}^4$, $R_i = t$ 表示在时间步骤 t 对顶点 i 执行计算操作， $R_i = 0$ 表示调度方案中不对顶点 i 执行计算操作，其他操作的定义类似，从而可以对应的构建出非二元的整数线性规划问题，在此不再赘述。

将模型调度问题使用上述数学公式表示出来，并设定了明确的优化目标，使设计者可以使用最优化理论中的数学方法进行调度方案的设计和优化。

返回图 1，所述计算流图调度方案的生成方法，还包括：步骤 S106，求解所述整数线性规划问题得到所述第三计算流图的调度方案。

在得到目标函数(如上述公式 1)以及限制条件之后，可以计算得到该目标函数在该限制条件下的解。步骤 S106 为求解所述整数线性规划问题得到使得目标函数最小的解的过程，即为第三计算流图的调度方案。

可选的，所述步骤 S106 包括：

对所述整数线性规划问题进行编码；

对所述编码进行求解得到所述第三计算流图中的顶点的执行顺序。

即将步骤 S105 中构建的目标函数以及限制条件进行编码，之后通过运行编码求解得到第三计算流图中的顶点的执行顺序。

可选的，求解整数线性规划问题可以使用现有的工具包进行求解。如使用 python 扩展包 pulp 对上述问题进行编码和求解，pulp 是针对线性规划问题开发的一个 python 扩展包，其提供了一种描述线性规划问题的语言规范，以及为各种线性规划问题的求解器封装了可以通过 python 来调用的接口。可以理解的，还可以通过其他任意编程语言对构建出的整数线性规划问题进行编码。可以使用任意的具有求解线性规划问题能力的软件对编码后的整数线性规划问题进行求解，在此不再赘述。

如图 7 所示为求解所述整数线性规划问题得到的第三计算流图中的顶点的执行顺序的示意图。其中，由于第一辅助顶点仅仅表示数据的输入，因此第一辅助顶点的执行顺序与其对应的顶点是相关的，即读取数据之后才会开始计算，其执行顺序对第三计算流图中其他顶点的执行顺序没有影响，仅仅用于构建上述整数线性规划问题，因此未示出第一辅助顶点。

返回图 1，所述计算流图调度方案的生成方法，还包括：步骤 S107，简化所述第三计算流图的调度方案，形成所述第二流图的调度方案。

由于所述第三计算流图中包括了辅助顶点，其仅仅是为了调度方案提供完备的信息。因此在实际使用时，这些顶点可以去掉。因此，所述步骤 S107 包括：

将所述第三计算流图的调度方案中的辅助顶点删除得到所述第二流图的调度方案。如图 8 所示为简化第三计算流图的调度方案后得到的第二流图的调度方案。所述

第二流图的调度方案即为最终的调度方案。

为了实际所使用的硬件设备能够发挥最大的性能。所述计算流图调度方案的生成方法，还包括：

根据计算单元的数量以及所述数量 N 确定所述调度方案中每个顶点处理的数据量。

如上述示例中，第二计算流图的调度方案为使用 4 个处理单元处理 4 批数据，但是 NPU 中包含 8 个计算单元，因此为了发挥最大的算力，可以将调度方案中的每个顶点处理的数据量增加一倍。可以理解的，每个顶点表示处理数据的逻辑，实际处理数据的是与所述顶点对应的计算单元。

上述实施例公开了一种计算流图调度方案的生成方法，该计算流图调度方案的生成方法包括：将原始计算流图中的顶点进行分组得到第一计算流图，其中所述第一计算流图中的顶点为所述原始计算流图中的顶点所形成的至少一个集合；根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N ， N 为大于等于 1 的整数；复制 N 张所述第一计算流图得到第二计算流图；在所述第二计算流图中加入辅助顶点得到第三计算流图；根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题；求解所述整数线性规划问题得到所述第三计算流图的调度方案；简化所述第三计算流图的调度方案，形成所述第二流图的调度方案。上述计算流图调度方案的生成方法通过将原始计算流图转换成第三计算流图并构建整数线性规划问题以求解调度方案，解决了现有技术中数据重用率低或并行度低的技术问题。

通过上述实施例可以看出：传统的模型调度方案设计需要设计者拥有丰富的 DL 模型优化经验，并且能够深入理解当前需要调度的 DL 模型的结构特性。而使用本公开提出的自动化算法可以规避这种对于专家经验的依赖。人工进行模型调度方案的设计，需要设计者花费大量时间对不同调度方案进行验证和效果对比，这个过程需要大量的时间和人力。而使用本公开提出的自动化算法可以在短时间内产出 DL 模型的调度方案，极大节省人力和时间成本。上文提到，对不同结构的 DL 模型，由于设计者自身的经验限制，以及对模型特性的理解深浅不一，不同的设计者可能会产出性能各异的模型调度方案，且难以证明其产出的调度方案是否为最优。使用本公开提出的自动化方法可以稳定地对不同结构的 DL 模型产出全局最优的调度方案。

另外，传统的计算流图调度方案设计只针对单批数据在计算流图上单次的运行的调度优化问题，不能解决当数据量较小时，计算流图中某个顶点计算并行度过低造成计算资源浪费的问题。本发明提出的方法通过计算流图的复制合并将问题转化为多批数据在计算流图上多次运行的调度优化问题，扩展了可行调度方案集合的边界，从而能够在更大的调度方案空间中寻找所有顶点都具有较高计算并行度的调度方案，减少计算资源浪费，以此提高计算流图执行的整体性能。

本公开实施例该提供一种计算流图调度方案的生成装置，包括：

第一计算流图生成模块，用于将原始计算流图中的原始顶点进行分组得到第一计算流图，所述每个分组作为第一计算流图中的一个顶点，所述顶点为所述原始计算流图中的至少一个原始顶点所形成集合；

计算单元数量确定模块，用于根据所述第一计算流图中的顶点的存储资源需求以

及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N , N 为大于等于 1 的整数;

第二计算流图生成模块, 用于复制 N 张所述第一计算流图得到第二计算流图;

第三计算流图生成模块, 用于在所述第二计算流图中加入辅助顶点得到第三计算流图;

整数线性规划问题构建模块, 用于根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题;

整数线性规划问题求解模块, 用于求解所述整数线性规划问题得到所述第三计算流图的调度方案;

简化模块, 用于简化所述第三计算流图的调度方案, 形成所述第二流图的调度方案。

进一步的, 所述第一计算流图生成模块, 还用于: 根据所述原始计算流图中的原始顶点的输入数据和输出数据将所述原始计算流图中的原始顶点进行分组得到第一计算流图。

进一步的, 所述计算单元数量确定模块, 还用于: 获取所述第一计算流图的顶点的最大存储需求; 根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N 。

进一步的, 所述计算单元数量确定模块, 还用于: 根据以下公式计算所述计算单元的数量 N : $2^{\lceil \log_2 \lceil M/m \rceil \rceil}$; 其中, M 表示所述最大存储需求, m 表示单个计算单元的存储空间大小。

进一步的, 所述第二计算流图生成模块, 还用于: 将所述第一计算流图复制所述数据 N 个; 将所述数量 N 个第一计算流图组合生成第二计算流图; 其中所述第二计算流图用于对多批数据并行处理。

进一步的, 所述辅助顶点包括: 表示所述原始计算流图的输入数据读取操作的第一辅助顶点、表示所述原始计算流图顶点的中间结果计算操作的第二辅助顶点以及表示所述第二计算流图中计算终止操作的第三辅助顶点。

进一步的, 所述整数线性规划问题构建模块, 还用于: 求 $R_{t,i}$, $S_{t,i}$, $L_{t,i}$ 和 $F_{t,i}$ 的取值, 使得以下多项式的取值最小:

$$\sum_{t=1}^T \sum_{i=1}^N (L_{t,i} + S_{t,i}) C_i$$

其中, i 表示第三计算流图中顶点的编号, t 表示时间步骤, $R_{t,i}$ 表示是否在第 t 个时间步骤计算第 i 个顶点的结果; $S_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果存储到低速缓存中; $L_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果

从低速缓存上读取到计算单元的缓存中； $F_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果在计算单元的缓存中所占用的空间释放； C_i 表示将第 i 个顶点的计算结果在低速缓存和计算单元的缓存之间传输所需要的消耗；其中，所述 $R_{t,i}=0$ 或 1 ， $S_{t,i}=0$ 或 1 ， $L_{t,i}=0$ 或 1 ， $F_{t,i}=0$ 或 1 ，其中 0 表示不执行对应的操作， 1 表示执行对应的操作； T 和 N 为大于 1 的整数；其中，所述整数线性规划问题还包括所述 $R_{t,i}$ ， $S_{t,i}$ ， $L_{t,i}$ 和 $F_{t,i}$ 的限制条件，所述限制条件由所述计算单元的硬件性能确定。

进一步的，所述整数线性规划问题求解模块，还用于：对所述整数线性规划问题进行编码；对所述编码进行求解得到所述第三计算流图中的顶点的执行顺序。进一步的，所述简化模块，还用于：将所述第三计算流图的调度方案中的辅助顶点删除得到所述第二流图的调度方案。

进一步的，所述计算流图调度方案的生成装置，还用于：根据计算单元的数量以及所述数量 N 确定所述调度方案中每个顶点处理的数据量。

本公开实施例还提供一种电子设备，包括：存储器，用于存储计算机可读指令；以及一个或多个处理器，用于运行所述计算机可读指令，使得所述处理器运行时实现上述实施例中的任一所述计算流图调度方案的生成方法。

本公开实施例还提供一种非暂态计算机可读存储介质，该非暂态计算机可读存储介质存储计算机指令，该计算机指令用于使计算机执行前述实施例中的任一所述计算流图调度方案的生成方法。

本公开实施例还提供一种计算机程序产品，其中，包括计算机指令，当所述计算机指令被计算设备执行时，所述计算设备可以执行前述实施例中的任一所述计算流图调度方案的生成方法。

本公开附图中的流程图和框图，图示了按照本公开各种实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段、或代码的一部分，该模块、程序段、或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意，在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个接连地表示的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或操作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

描述于本公开实施例中所涉及到的单元可以通过软件的方式实现，也可以通过硬件的方式来实现。其中，单元的名称在某种情况下并不构成对该单元本身的限定。

本文中以上描述的功能可以至少部分地由一个或多个硬件逻辑部件来执行。例如，非限制性地，可以使用的示范类型的硬件逻辑部件包括：现场可编程门阵列（FPGA）、专用集成电路（ASIC）、专用标准产品（ASSP）、片上系统（SOC）、复杂可编程逻辑设备（CPLD）等等。

在本公开的上下文中，机器可读介质可以是有形的介质，其可以包含或存储以供指令

执行系统、装置或设备使用或与指令执行系统、装置或设备结合地使用的程序。机器可读介质可以是机器可读信号介质或机器可读储存介质。机器可读介质可以包括但不限于电子的、磁性的、光学的、电磁的、红外的、或半导体系统、装置或设备，或者上述内容的任何合适组合。机器可读存储介质的更具体示例会包括基于一个或多个线的电气连接、便携式计算机盘、硬盘、随机存取存储器（RAM）、只读存储器（ROM）、可擦除可编程只读存储器（EPROM 或快闪存储器）、光纤、便捷式紧凑盘只读存储器（CD-ROM）、光学储存设备、磁储存设备、或上述内容的任何合适组合。

权 利 要 求

1、一种计算流图调度方案的生成方法，其特征在于，包括：

将原始计算流图中的原始顶点进行分组得到第一计算流图，所述每个分组作为第一计算流图中的一个顶点，所述顶点为所述原始计算流图中的至少一个原始顶点所形成集合；

根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N ， N 为大于等于 1 的整数；

复制 N 张所述第一计算流图得到第二计算流图；

在所述第二计算流图中加入辅助顶点得到第三计算流图；

根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题；

求解所述整数线性规划问题得到所述第三计算流图的调度方案；

简化所述第三计算流图的调度方案，形成所述第二流图的调度方案。

2、如权利要求 1 所述的计算流图的调度方案生成方法，其特征在于，所述将原始计算流图中的原始顶点进行分组得到第一计算流图，包括：

根据所述原始计算流图中的原始顶点的输入数据和输出数据将所述原始计算流图中的原始顶点进行分组得到第一计算流图。

3、如权利要求 1 或 2 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N ，包括：

获取所述第一计算流图的顶点的最大存储需求；

根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N 。

4、如权利要求 3 所述的计算流图的调度方案生成方法，其特征在于，所述根据所述最大存储需求以及所述计算单元的存储资源计算并行处理单批计算数据所需要的计算单元的数量 N ，包括：

根据以下公式计算所述计算单元的数量 N ： $2^{\lceil \log_2 \lceil M/m \rceil \rceil}$ ；

其中， M 表示所述最大存储需求， m 表示单个计算单元的存储空间大小。

5、如权利要求 1-4 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述将所述第一计算流图复制数量 N 个得到第二计算流图，包括：

将所述第一计算流图复制所述数据 N 个；

将所述数量 N 个第一计算流图组合生成第二计算流图；其中所述第二计算流图用于对多批数据并行处理。

6、如权利要求 1-5 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述辅助顶点包括：

表示所述原始计算流图的输入数据读取操作的第一辅助顶点、表示所述原始计算流图顶点的中间结果计算操作的第二辅助顶点以及表示所述第二计算流图中计算终止操作的第三辅助顶点。

7、如权利要求 1-6 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述根据第三计算流图构建与所述第三计算流图对应的整数线性规划问题，包括：

求 $R_{t,i}$ ， $S_{t,i}$ ， $L_{t,i}$ 和 $F_{t,i}$ 的取值，使得以下多项式的取值最小：

$$\sum_{t=1}^T \sum_{i=1}^N (L_{t,i} + S_{t,i}) C_i$$

其中， i 表示第三计算流图中顶点的编号， t 表示时间步骤， $R_{t,i}$ 表示是否在第 t 个时间步骤计算第 i 个顶点的结果； $S_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果存储到低速缓存中； $L_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果从低速缓存上读取到计算单元的缓存中； $F_{t,i}$ 表示在第 t 个时间步骤是否将第 i 个顶点的计算结果在计算单元的缓存中所占用的空间释放； C_i 表示将第 i 个顶点的计算结果在低速缓存和计算单元的缓存之间传输所需要的消耗；其中，所述 $R_{t,i}=0$ 或 1 ， $S_{t,i}=0$ 或 1 ， $L_{t,i}=0$ 或 1 ， $F_{t,i}=0$ 或 1 ，其中 0 表示不执行对应的操作， 1 表示执行对应的操作； T 和 N 为大于 1 的整数；其中，所述整数线性规划问题还包括所述 $R_{t,i}$ ， $S_{t,i}$ ， $L_{t,i}$ 和 $F_{t,i}$ 的限制条件，所述限制条件由所述计算单元的硬件性能确定。

8、如权利要求 1-7 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述求解所述整数线性规划问题得到所述第三计算流图的调度方案，包括：

对所述整数线性规划问题进行编码；

对所述编码进行求解得到所述第三计算流图中的顶点的执行顺序。

9、如权利要求 1-8 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述简化所述第三计算流图的调度方案，形成所述第二流图的调度方案，包括：

将所述第三计算流图的调度方案中的辅助顶点删除得到所述第二流图的调度方案。

10、如权利要求 1-8 中任一项所述的计算流图的调度方案生成方法，其特征在于，所述方法还包括：

根据计算单元的数量以及所述数量 N 确定所述调度方案中每个顶点处理的数据量。

11、一种计算流图调度方案的生成装置，包括：

第一计算流图生成模块，用于将原始计算流图中的原始顶点进行分组得到第一计算流图，所述每个分组作为第一计算流图中的一个顶点，所述顶点为所述原始计算流图中的至少一个顶点所形成的集合；

计算单元数量确定模块，用于根据所述第一计算流图中的顶点的存储资源需求以及计算单元的存储资源确定并行处理单批计算数据所需要的计算单元的数量 N ， N 为大于等于 1 的整数；

第二计算流图生成模块，用于复制 N 张所述第一计算流图得到第二计算流图；

第三计算流图生成模块,用于在所述第二计算流图中加入辅助顶点得到第三计算流图;

整数线性规划问题构建模块,用于根据所述第三计算流图构建与所述第三计算流图对应的整数线性规划问题;

整数线性规划问题构建模块求解模块,用于求解所述整数线性规划问题得到所述第三计算流图的调度方案;

简化模块,用于简化所述第三计算流图的调度方案,形成所述第二流图的调度方案。

12、根据权利要求 11 所述的装置,其特征在于,所述第一计算流图生成模块,还用于:根据所述原始计算流图中的原始顶点的输入数据和输出数据将所述原始计算流图中的原始顶点进行分组得到第一计算流图。

13、一种电子设备,包括:存储器,用于存储计算机可读指令;以及一个或多个处理器,用于运行所述计算机可读指令,使得所述处理器运行时实现权利要求 1-10 中任一项所述的方法。

14、一种计算机可读存储介质,该非暂态计算机可读存储介质存储计算机指令,该计算机指令用于使计算机执行权利要求 1-10 中任一项所述的方法。

15、一种计算机程序产品,包括计算机指令,当所述计算机指令被计算设备执行时,所述计算设备可以执行权利要求 1-10 中任一项所述的方法。