

模型编译方法、装置、设备和存储介质

本申请要求于 2023 年 04 月 25 日提交中国专利局、申请号为 202310478439.4、发明名称为“模型编译方法、装置、设备和存储介质”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。

5

技术领域

本申请涉及人工智能技术领域，尤其涉及一种模型编译方法、装置、设备和存储介质。

背景技术

10

目前，机器学习被广泛地应用于图像处理、自然语言处理、搜索、推荐及广告等领域中。机器学习的计算任务往往需要消耗一定的计算机算力，常常被部署在诸如图像处理器（Graphics Processing Unit,简称 GPU）等执行器上进行计算。

15

具体实施过程中，机器学习模型从高层建模到底层部署需要进行模型编译，即将机器学习模型对应的高级代码语言翻译为底层硬件所支持的底层机器语言。在进行模型编译的过程中，机器学习模型对应的高级代码语言先转换成是模型对应的计算图的计算表达，通过对计算图执行相应的优化策略，生成最终的底层机器语言，比如在 GPU 上的运行的可执行程序。其中，对计算图执行的优化策略决定了机器学习模型的任务执行效率。

20

相关技术中，模型编译结果运行时，往往存在 CPU 调度计算图中各计算子图对应的执行指令给 GPU 时的调度开销大、GPU 资源利用不充分等问题，无法满足用户对机器学习的计算任务的性能要求。

发明内容

本申请实施例提供一种模型编译方法、装置、设备和存储介质,用以提升编译得到的可执行程序文件在被执行器执行时的执行效率以及执行器的资源利用率。

25

第一方面，本申请实施例提供一种模型编译方法，所述方法包括：获取待编译的模型对应的计算图中的目标计算子图，所述目标计算子图中包含多个访存密集型算子及相应数据，所述目标计算子图是所述计算图中包含的一个计算子图；根据所述模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是指依次采用所述多个访存密集型算子对相应数据进行计算的任务，所述计算任务并行度指示每个线程能够并行执行的计算任务数量，所述目标线程是多个线程中任一个；针对任一组计算任务，根据所述多个访存密集型算子的计算依赖关系，对所述任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，所述重排使得同一访存密集型算子对应的不同计算任务被并行排列；根据所述重排后的多个计算任务，生成所述目标线程与所述任一组计算任务对应的执行指令序列；根据所述目标线程与所述任一组计算任务对应的执行指令序列，确定所述执行器执行所述模型的可执行程序文件，所述可执行程序文件中包含所述计算图包含的多个计算子图各自对应的执行指令序列。

30

第二方面，本申请实施例提供一种模型编译装置，所述装置包括：获取模块，用于获取待编译的模型对应的计算图中的目标计算子图，所述目标计算子图中包含多个访存密集型算子及相应数据，所述目标计算子图是所述计算图中包含的一个计算子图；编译模块，用于根据所述模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是指依次采用所述多个访存密集型算子对相应数据进行计算的任务，所述计算任务并行度指示每个线程能够并行执行的计算任务数量，所述目标线程是多个线程中任一个；针对任一组计算任务，根据所述多个访存密集型算子的计算依赖关系，对所述任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，所述重排使得同一访存密集型算子对应的不同计算任务被并行排列；根据所述重排后的多个计算任务，生成所述目标线程与所述任一组计算任务对应的执行指令序列；根据所述目标线程与所述任一组计算任务对应的执行指令序列，确定所述执行器执行所述模型的可执行程序文件，所述可执行程序文件中包含所述计算图包含的多个计算子图各自对应的执行指令序列。

第三方面，本申请实施例提供一种电子设备，包括：存储器、处理器、通信接口；其中，所述存储器上存储有可执行代码，当所述可执行代码被所述处理器执行时，使所述处理器执行如第一方面所述的模型编译方法。

第四方面，本申请实施例提供了一种非暂时性机器可读存储介质，所述非暂时性机器可读存储介质上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器至少可以实现如第一方面所述的模型编译方法。

附图说明

为了更清楚地说明本申请实施例中的技术方案，下面将对实施例描述中所需要使用的附图作一简单地介绍，显而易见地，下面描述中的附图是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

- 图1为本申请实施例提供了一种模型编译方法的流程图；
- 图2为本申请实施例提供了一种计算任务的分组示意图；
- 图3为本申请实施例提供的另一种模型编译方法的流程图；
- 图4为本申请实施例提供了一种执行指令序列的执行情形示意图；
- 图5为本申请实施例提供的又一种模型编译方法的流程图；
- 图6为本申请实施例提供的又一种模型编译方法的流程图；
- 图7为本申请实施例提供了一种计算子图的示意图；
- 图8为本申请实施例提供了一种模型编译装置的结构示意图；
- 图9为本实施例提供了一种电子设备的结构示意图。

具体实施方式

为使本申请实施例的目的、技术方案和优点更加清楚，下面将结合本申请实施例中的附图，对本申请实施例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。

下面结合附图对本申请的一些实施方式作详细说明。在各实施例之间不冲突的情况下，下述的实施例及实施例中的特征可以相互组合。

5 另外，下述各方法实施例中的步骤时序仅为一种举例，而非严格限定。

需要说明的是，本申请实施例中所涉及的用户信息（包括但不限于用户设备信息、用户个人信息等）和数据（包括但不限于用于分析的数据、存储的数据、展示的数据等），均为经用户授权或者经过各方充分授权的信息和数据，并且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准，并提供有相应的操作入口，供用户选择授权或者拒绝。

10 为便于理解，以下先对本申请实施例涉及的几个概念进行说明。

模型编译，是指将机器学习模型对应的机器学习算法从开发形态转换为部署形态的过程。其中，机器学习算法的开发形态是指以接近自然语言和数学公式表达的高级编程语言所编写的源代码，机器学习算法的部署形态是指底层机器能够执行的程序代码。通俗来说，模型编译是一种高级代码语言向底层机器语言转换的过程，其目的在于将机器学习模型部署在特定的硬件环境之中。

计算图，是一种表示机器学习模型推理过程的计算逻辑与状态的工具。某一机器学习模型对应的计算图，包含有多个计算子图，每个计算子图中包含有算子和相应数据（通常以张量形式表示）。在模型编译过程中，计算图是高级代码语言向底层机器语言转换时的中间表达（Intermediate Representation，简称 IR），通过对计算图执行相应的优化策略，得到最终编译的底层机器语言，也即底层硬件的可执行程序文件。

实际应用中，根据可执行程序文件中各算子对应的计算指令和访存指令的比例，将计算子图包含的算子分为计算密集型算子和访存密集型算子。其中，计算密集型算子是指计算指令和访存指令的比例大于设定阈值的算子，比如：矩阵乘法、卷积等；访存密集型算子是指计算指令和访存指令的比例小于等于设定阈值的算子，比如：逐点计算（element-wise）、归约（reduce）计算等。本实施例中的模型编译方法，用于对访存密集型算子相关的执行指令编译过程进行优化。简单来说，在计算密集型算子中，某个数据可能会被反复使用多次，比如矩阵乘法中，矩阵中某一个点会被执行多次不同的乘法计算；在访存密集型算子中，某个数据可能仅被使用一次，比如两个矩阵的逐点相加计算中，一个点仅被执行一次加法计算。

图1为本申请实施例提供的一种模型编译方法的流程图，如图1所示，该方法包括如下步骤：

101、获取待编译的模型对应的计算图中的目标计算子图，目标计算子图中包含多个访存密集型算子及相应数据，目标计算子图是模型对应的计算图中包含的一个计算子图。

35 102、根据模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是依次采用多个访存密集型算子对相应数据进行计算的任务，计算任务并行度指示每个线程能够并行执行的计算任务数量，目标线程是多个线程中任一个。

103、针对任一组计算任务，根据多个访存密集型算子的计算依赖关系，对任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，重排使得同一访存密集型算子对应的不同计算任务被并行排列。

5 104、根据重排后的多个计算任务，生成目标线程与任一组计算任务对应的执行指令序列。

105、根据目标线程与任一组计算任务对应的执行指令序列，确定执行器执行模型的可执行程序文件，可执行程序文件中包含计算图包含的多个计算子图各自对应的执行指令序列。

本实施例中，模型编译方法可由 CPU 执行，CPU 上设置有具有相应模型编译功能的编译器。在一实施方式中，执行器可以是图像处理器（Graphics Processing Unit,简称 GPU）等加速器，以下以执行器为 GPU 为例进行说明，但并不以此为限。

实际应用中，GPU 基于其上有限的片上存储空间（包括共享内存 shared memory 和寄存器 register），在执行模型对应的可执行程序文件时，往往会由于计算子图中计算密集型算子和访存密集型算子的资源使用冲突，导致模型对应的可执行程序的运行效率低下，GPU 的资源使用不充分。

具体而言，计算子图中的计算密集型算子具有数据局部性，为了取得较高的数据重用效率，通常会使用 GPU 上较多的片上存储空间存储数据计算结果，以便这些计算结果再被多个其他计算访问，这一特性不可避免的会造成 GPU 中仅有少量的片上存储空间可被访存密集型算子利用。可以理解的是，GPU 中少量的片上存储空间仅能提供少量的线程，而访存密集型算子往往需要较多的线程来提高对 GPU 片外存储空间的访存效率。因此，综合来说，计算密集型算子和访存密集型算子的资源使用冲突在于，GPU 有限的片上存储空间不能同时既为计算密集型算子提供足够的用于存储数据计算结果的片上存储空间，又为访存密集型算子提供足够多的线程以达到较高的线程并行度。

为解决上述至少一个技术问题，本实施例中，通过对线程内访存密集性算子对应的计算任务进行顺序重排，使得在 GPU 内线程数量保持不变的情况下，编码得到的可执行程序文件在被执行时，访存密集型算子对应的计算任务的整体并行度更高，从而有效提升模型对应的可执行程序的运行效率。由于对访存密集型算子对应的计算任务的顺序重排，在模型对应的可执行程序文件中表现为对计算任务对应的执行指令序列的重排，因此，本方案可以理解为通过提高可执行程序文件中各线程需要执行的计算任务的执行指令序列并行度，使得在线程并行度不变的情况下，可执行程序被执行时计算任务整体的并行度更高。以下对本实施例中的模型编译方法展开进行说明。

在进行模型编译时，待编译的模型对应的高级代码语言在被发送给 CPU 后，CPU 中的编译器可以根据预设规则自动生成模型对应的计算图。任一模型对应的计算图包含有多个不同的计算子图，具体实施过程中，CPU 中的编译器确定任一计算子图对应的执行指令序列的过程是相似的，本实施例以目标计算子图为例，对确定计算子图对应的执行指令序列进行说明。其中，目标计算子图可以是待编译模型对应的计算图所包含的多个计算子图中的任一个。

需要说明的是，本实施例中不对生成计算子图对应的执行指令序列的顺序进行限制，

每个计算子图可单独生成其对应的执行指令序列。

概括来说，在确定目标计算子图对应的编译结果（也即指令执行序列）时，先确定目标计算子图对应的计算任务数量；再确定每个线程分别需要执行的多个计算任务；之后对每个线程内分配到的计算任务进行分组，并对每组计算任务内的计算任务进行顺序重排，
5 以得到重排后的各组计算任务分别对应的执行指令序列；最后，基于各线程中每组计算任务分别对应的执行指令序列，确定目标计算子图对应的编译结果。

展开来说，目标计算子图中包含有多个访存密集型算子和相应数据，本实施例中，将依次采用多个访存密集型算子对相应数据进行计算的任务称为计算任务。比如：假设目标计算子图存在访存密集型算子“+”和“-”，相应数据为3个维度均为8*8的矩阵A、B和
10 C，对应的计算逻辑为A+B-C，则该目标计算子图对应的任务数量为64，分别为计算任务1、计算任务2、...、计算任务64。

假设待编译的模型在编译后得到的可执行程序文件由GPU-i执行，也即待编译的模型在编译后要部署在GPU-i上，则编译时CPU上的编译器能够获取到GPU-i对应的配置信息，比如：GPU-i上可用于执行访存密集型算子对应的计算任务的线程数量、线程标识和
15 每个线程的计算任务并行度等。其中，线程的计算任务并行度可以理解为线程能够同时执行的计算任务的数量。基于GPU-i的配置信息，可以确定GPU-i上各线程需要串行执行的多组计算任务。

举例来说，承接于上述对目标计算子图的假设，若GPU-i中设有4个线程：线程1、线程2、线程3和线程4，则可将目标计算子图对应的64个计算任务平均分配给4个线程，
20 例如线程1分配到计算任务1-16，线程2分配到计算任务17-32，线程3分配到计算任务33-48，线程4分配到计算任务49-64。实际应用中，也可采用非平均分配的方式，将目标计算子图对应的多个计算任务分配给GPU-i中的相应线程。

之后，针对4个线程中的任一个线程（称为目标线程），根据目标线程对应的计算任务并行度，对目标线程中的16个计算任务进行分组。图2为本申请实施例提供的一种计算任务的分组示意图，当线程1至线程4的计算任务并行度为2时，以线程1和线程2分别需要执行的16个计算任务为例进行示意，如图2所示，线程1需要执行的计算任务1-16以及线程2需要执行的计算任务17-32，均被划分为8组，每组计算任务中包含2个计算任务；当线程1至线程4的计算任务并行度为4时，以线程3需要执行的16个计算任务为例进行示意，如图2所示，线程3需要执行的计算任务33-48被划分为4组，每组计算任务中包含4个计算任务；当线程1至线程4的计算任务并行度为8时，以线程4需要执行的16个计算任务为例进行示意，如图2所示，线程4需要执行的计算任务49-64被划分为2组，每组计算任务中包含8个计算任务。基于图2所示意的计算任务分组情况可知，
25 不同的计算任务并行度对应于不同的分组结果。

需要说明的是，本实施例中，目标线程执行划分得到的多组计算任务时，串行执行。
35 目标线程在执行同一组计算任务中的多个计算任务时，并行执行。所谓串行是指先执行完一组计算任务后，再执行下一组计算任务；所谓并行是指同时执行同一组计算任务中的多个计算任务。

为实现同组计算任务中的多个计算任务高效执行，本实施例中，针对一组计算任务，

根据多个访存密集型算子之间的计算依赖关系，对多个计算任务进行了顺序重排。其中，计算依赖关系可以理解为访存密集型算子的执行先后顺序以及相应数据的关联关系，以 $A+B-C$ 为例，访存密集型算子“-”在访存密集型算子“+”之后执行，且访存密集型算子“-”对 C 进行计算时依赖于“ $A+B$ ”的计算结果，则认为访存密集型算子“-”与访存密集型算子“+”之间存在计算依赖关系。

需要强调的是，本实施例中对计算任务的顺序重排的目的是使得同一访存密集型算子对应的不同计算任务被并行排列。举例来说，假设线程1内存在一组计算任务，该组计算任务包含2个计算任务分别为 $a1+b1-c1$ 和 $a2+b2-c2$ ，其中 $a1+b1$ 的计算结果为 $d1$ ， $a2+b2$ 的计算结果为 $d2$ ，则在对这两个任务进行顺序重排时，将访存密集型算子“+”对应的计算任务“ $a1+b1$ ”与计算任务“ $a2+b2$ ”并行排列，将访存密集型算子“-”对应的计算任务“ $d1-c1$ ”与计算任务“ $d2-c2$ ”并行排列。

针对上述线程1中的2个计算任务，重排后生成的执行指令序列可以为“ $a1+b1$ 的计算指令， $a2+b2$ 的计算指令， $d1-c1$ 的计算指令， $d2-c2$ 的计算指令”。由于 $a1+b1$ 与 $a2+b2$ 之间不存在计算依赖关系，所以在连续发送 $a1+b1$ 的计算指令和 $a2+b2$ 的计算指令后， $a1+b1$ 和 $a2+b2$ 均能够执行，可以实现指令并行执行的效果，提高指令的执行效率。

可以理解的是，如果计算任务 $a1+b1-c1$ 和 $a2+b2-c2$ 不进行计算任务的顺序重排，则其对应的执行指令序列为“ $a1+b1$ 的计算指令， $d1-c1$ 的计算指令， $a2+b2$ 的计算指令， $d2-c2$ 的计算指令”。实际执行过程中，在发送 $a1+b1$ 的计算指令后，需要得到 $d1$ 后才能继续发送 $d1-c1$ 的计算指令，之后类似地再发送 $a1+b1$ 的计算指令以及 $d1-c1$ 的计算指令。这种方式会导致计算指令之间存在时间空隙，比如：发送 $a1+b1$ 的计算指令后，等待 $a1+b1$ 计算完成得到 $d1$ 的时间空隙，从而不能充分利用GPU-i的片上存储资源，造成资源浪费且计算效率低下。

从可执行程序文件被执行时计算任务整体的并行度的角度来看，上述计算任务并行度为2的线程1，其可以并行执行2个计算任务，若GPU-i中的4个线程对应的计算任务并行度均为2，则GPU-i在执行可执行程序时，计算任务整体的并行度为 $2*4=8$ 。相较于原本的4个线程对应的计算任务整体并行度为4的情形，通过本实施例中的方法得到的各组计算任务生成的执行指令序列在被执行时，同一访存密集型算子对应的计算指令能够被并发执行，计算任务整体的并行度得到有效提高，进而计算指令的执行效率也有所提高。

最后，根据目标线程与任一组计算任务对应的执行指令序列，确定执行器执行模型的可执行程序文件，其中，可执行程序文件中包含计算图包含的多个计算子图各自对应的执行指令序列。

本实施例中，通过对线程需要执行的每组计算任务中同一访存密集型算子对应的不同计算任务进行并行排列，使得得到的执行指令序列具有较高的指令并行度，最终生成的可执行程序文件在被执行时，能够并行地发送并执行多个执行指令，以同时对同一访存密集型算子对应的不同计算任务进行并行计算，能够在执行器中的线程数量保持不变的情况下提高可执行程序文件的运行效率。

上述实施例从计算任务并行度的角度考虑，提供了提高可执行程序文件的运行效率的模型编译方案。实际应用中，影响可执行程序文件的运行效率的因素还包括：指令被执行

过程中的资源调度、指令的调度开销等。以下实施例从这些因素考虑，提供其他的模型编译方案。

图3为本申请实施例提供的另一种模型编译方法的流程图，如图3所示，该方法包括如下步骤：

5 301、获取待编译的模型对应的计算图中的目标计算子图，目标计算子图中包含多个访存密集型算子及相应数据，目标计算子图是模型对应的计算图中包含的一个计算子图。

302、根据模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是依次采用多个访存密集型算子对相应数据进行计算的任务，计算任务并行度指示每个线程能够并行执行的计算任务数量，目标线程是多个线程中任一个。

10 303、针对任一组计算任务，根据多个访存密集型算子的计算依赖关系，对任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，重排使得同一访存密集型算子对应的不同计算任务被并行排列。

304、根据重排后的多个计算任务，生成目标线程与任一组计算任务对应的执行指令序列。

15 305、在目标线程与任一组计算任务对应的执行指令序列前，插入第一数据拷贝指令，第一数据拷贝指令用于将任一组计算任务的下一组计算任务对应的数据从执行器的片外存储空间拷贝到执行器的片上存储空间的第一缓冲区中，以使得在执行下一组计算任务时从第一缓冲区读取对应的数据。

20 306、根据目标线程与任一组计算任务对应的执行指令序列以及第一数据拷贝指令，确定执行器执行模型的可执行程序文件，可执行程序文件中包含计算图包含的多个计算子图各自对应的执行指令序列。

其中，步骤301至步骤304的具体实施过程可参考前述实施例，本实施例中不再赘述。

25 本实施例中，仍以执行器为GPU为例进行说明。实际应用中，GPU的存储空间可分为片上存储空间和片外存储空间。其中，片上存储空间包括共享内存shared memory和寄存器，是指线程块内可共享的片上存储器；片外存储空间对应于GPU的全局显存global memory（也可称为主存），对全局线程可见。

30 GPU可执行的模型对应的可执行程序文件中，某一组计算任务对应的执行指令序列，通常包含有与“读-计算-写”相关的指令。其中，“读”是指从片上存储空间的缓冲区或者寄存器读取该组计算任务对应的数据；“计算”是指根据从缓冲区读取的数据执行与访存密集型算子对应的计算；“写”是指将计算结果写入片上存储空间的寄存器。

35 GPU在执行任一组计算任务对应的执行指令序列之前，通常需要先将该组计算任务对应的数据从片外存储空间拷贝到片上存储空间的缓冲区，以便在执行该组计算任务对应的执行指令序列时，从缓冲区读取相应数据。实际应用中，各组计算任务对应的数据拷贝时间会影响各组计算任务对应执行指令序列的执行效率，若拷贝时间过长，则需要在长时间等待后，才能执行计算任务对应的执行指令序列。

本实施例中，为了加快可执行程序文件的运行效率，引入了“双缓冲”机制。通俗来说，就是在GPU的片上存储空间中设置了两个缓冲区。通过这两个缓冲区的设置，能够

在执行目标线程中当前组的计算任务时，预先将后一组计算任务对应的数据从片外存储空间拷贝到片上存储空间的缓冲区，也即将后一组计算任务对应的数据拷贝过程前置，从而在执行后一组计算任务时，直接从片上存储空间的缓冲区读取各组计算任务对应数据即可，无需再等待从片外存储空间拷贝后一组计算任务对应数据到片上存储空间的缓冲区。

5 基于双缓冲区的设置，能够节约拷贝计算任务相关数据到片上存储空间时的等待时间，使得执行指令的运行效率更高。

一般地，目标线程需要串行执行多组计算任务，按照其对应的执行顺序，可以将多组计算任务分为第一个执行任务（也可称为第 1 组执行任务）和第 k 组执行任务，其中 k 为大于 1 的整数。

10 当步骤 305 中的任一组其他任务对应于第 k 组执行任务中时，可以直接在第 k 组执行任务对应的执行指令序列前，插入第一数据拷贝指令。第一数据拷贝指令用于将第 k+1 组计算任务对应的数据从 GPU 的片外存储空间拷贝到 GPU 的片上存储空间的第一缓冲区中，以使得在执行第 k+1 组计算任务时从第一缓冲区读取对应的数据。

15 当步骤 305 中的任一组计算任务对应于第 1 组计算任务时，则在第一数据拷贝指令前，插入第二数据拷贝指令，第二数据拷贝指令用于将第 1 组计算任务对应的数据从 GPU 的片外存储空间拷贝到 GPU 的片上存储空间的第二缓冲区中。

需要说明的，如果每组计算任务中包含有多个访存密集型算子，则在拷贝各组计算任务相应数据到片上存储空间的缓冲区时，按照访存密集型算子的计算依赖关系，依次拷贝同组计算任务中与同一访存密集型算子相关的数据到片上存储空间的缓冲区。

20 具体实施过程中，上述根据第一拷贝指令，拷贝到第一缓冲区中的第 k+1 组计算任务对应的数据为第 k+1 组计算任务中第一个访存密集型算子所对应的数据；上述根据第二拷贝指令，拷贝到第二缓冲区中的第 1 组计算任务对应的数据为第 1 组计算任务中第一个访存密集型算子所对应的数据。

25 对于非第一个访存密集型算子对应的数据，在一实施方式中，其对应的拷贝指令可以与第一个访存密集型算子的拷贝指令类似，被前置到上一组计算任务的执行指令序列中，或者在当前组计算任务的执行指令序列中，本实施例对此不作限制。

以下将通过举例，更加清楚的说明拷贝指令与某一组计算任务的执行指令序列的关系。在举例过程中，为了更清晰的表达，将一组计算任务对应的执行指令序列拆解为了相应的读取指令、计算指令、写入指令。

30 以图 2 中示意的线程 1 为例进行说明。如图 2 所示，线程 1 需要串行执行分组 1 至分组 8 中的计算任务，假设分组 1 中包括： $a_1+b_1-c_1$ 和 $a_2+b_2-c_2$ 这两个计算任务，分组 2 中包括： $a_3+b_3-c_3$ 和 $a_4+b_4-c_4$ 这两个计算任务，分组 3 中包括： $a_5+b_5-c_5$ 和 $a_6+b_6-c_6$ 这两个计算任务。其中， a_1+b_1 的计算结果为 d_1 ， a_2+b_2 的计算结果为 d_2 ， a_3+b_3 的计算结果为 d_3 ， a_4+b_4 的计算结果为 d_4 ， a_5+b_5 的计算结果为 d_5 ， a_6+b_6 的计算结果为 d_6 。以此类推，分组 4 至分组 8 中分别包含的两个计算任务不再进行举例。

35 假设 GPU 的片上存储空间中设置的两个缓冲区分别为 buffer0 和 buffer1，在引入“双缓冲”机制的情况下，线程 1 中的分组 1、分组 2、分组 3 对应的执行指令序列分别为：

线程 1 中分组 1 对应的执行指令序列：1) 将数据 a_1 、 b_1 、 a_2 和 b_2 从片外存储空间拷

贝到 buffer0 的拷贝指令；2) 将数据 a3、b3、a4 和 b4 从片外存储空间拷贝到 buffer1 的拷贝指令；3) 从 buffer0 中读取数据 a1、b1、a2 和 b2 的读取指令；4) 计算 a1+b1 的计算指令，计算 a2+b2 的计算指令；5) 将计算结果 d1、d2 写入寄存器的写入指令；6) 从寄存器读取计算结果 d1、d2 的读取指令；7) 将数据 c1、c2 从片外存储空间拷贝到 buffer0 的拷贝指令；8) 从 buffer0 读取 c1、c2 的读取指令；9) 计算 d1+c1 的计算指令，计算 d2+c2 的计算指令；(10) 将计算结果写入片外存储空间的写入指令。

其中，在执行完线程 1 中的分组 1 对应的执行指令序列后，buffer0 为空。

线程 1 中分组 2 对应的执行指令序列：1) 将数据 a5、b5、a6 和 b6 从片外存储空间拷贝到 buffer0 的拷贝指令；2) 从 buffer1 中读取数据 a3、b3、a4 和 b4 的读取指令；3) 计算 a3+b3 的计算指令，计算 a4+b4 的计算指令；4) 将计算结果 d3、d4 写入寄存器的写入指令；5) 从寄存器读取计算结果 d3、d4 的读取指令；6) 将数据 c3、c4 从片外存储空间拷贝到 buffer1 的拷贝指令；7) 从 buffer1 读取 c3、c4 的读取指令；8) 计算 d3+c3 的计算指令，计算 d4+c4 的计算指令；(9) 将计算结果写入片外存储空间的写入指令。

其中，在执行完线程 1 中的分组 2 对应的执行指令序列后，buffer1 为空。

线程 1 中的分组 3 至分组 7 对应的执行指令序列与分组 2 对应的执行指令序列类似，在此不再赘述。由于分组 8 之后，并没有对应的后一分组，因此分组 8 对应的执行指令序列中并没有将后一组中第一个访存密集型算子所对应的数据拷贝到缓冲区的拷贝指令。

另外，通过上述举例可以发现，本实施例中的第一缓冲区和第二缓冲区并不是固定的指示某一特定的缓冲区，而是交替更新的，比如分组 1 的指令序列中第一缓冲区指示 buffer1，分组 2 的指示序列中第一缓冲区指示 buffer0。

假设将线程 1 所要执行的各组计算任务中第一个访存密集型算子对应数据从片外存储空间拷贝到片上存储空间的缓冲区上的时间均为 t_1 ，各组计算任务对应的执行指令序列的执行时间均为 t_2 ，则图 4 能够更清楚的反映出本实施例中的“双缓冲”机制对模型对应的可执行程序文件的运行效率的提高。

图 4 为本申请实施例提供的一种执行指令序列的执行情形示意图。图 4 中示意了执行 3 组计算任务对应的指令序列需要消耗的时间，如图 4 所示，在非双缓冲机制下，由于仅有一个缓冲区，每次只能拷贝一组计算任务相关的数据，因此，对于下一组计算任务来说，只能等待上一组计算任务的执行指令序列执行完成，缓冲区空闲才能进行对应的数据拷贝，执行完 3 组计算任务对应的指令序列需要消耗的时间为 $3 * (t_1 + t_2)$ 。而在双缓冲机制下，在拷贝分组 1 的计算任务对应的数据的同时还可以拷贝分组 2 的计算任务对应的数据，在执行分组 2 对应的执行指令序列的同时还可以拷贝分组 3 的计算任务对应的数据，执行完 3 组计算任务对应的指令序列需要消耗的时间为 $t_1 + 3 * t_2$ ，明显小于非双重机制下的执行时间。

因此，本实施例中，通过引入双缓冲机制，能够将不同组计算任务的计算和片外存储空间向片内存储空间的数据拷贝重叠起来，有效的提高访存密集型算子的访存效率，提高模型对应可执行程序文件的运行效率。

图 5 为本申请实施例提供的又一种模型编译方法的流程图，如图 5 所示，该方法包括如下步骤：

501、获取待编译的模型对应的计算图中的目标计算子图，目标计算子图中包含多个访存密集型算子及相应数据，目标计算子图是模型对应的计算图中包含的一个计算子图。

502、根据模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是依次采用多个访存密集型算子对相应数据进行计算的任务，计算任务并行度指示每个线程能够并行执行的计算任务数量，目标线程是多个线程中任一个。

503、针对任一组计算任务，根据多个访存密集型算子的计算依赖关系，对任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，重排使得同一访存密集型算子对应的不同计算任务被并行排列。

504、根据重排后的多个计算任务，生成目标线程与任一组计算任务对应的执行指令序列。

505、根据多个线程在目标计算子图下各自执行的多组计算任务分别对应的执行指令序列，确定目标计算子图对应的执行指令序列；融合计算图中多个计算子图分别对应的执行指令序列，以确定模型的可执行程序文件。

其中，步骤 501 至步骤 504 的具体实施过程可参考前述实施例，本实施例中不再赘述。

本实施例中，在生成模型对应的可执行程序文件时，首先，生成每一组计算任务对应的执行指令序列；然后，基于线程需执行的各组计算任务对应的执行指令序列，确定线程对应的执行指令序列；之后，基于计算子图对应的各线程的执行指令序列，确定计算子图对应的执行指令序列；最后，融合模型的计算图对应的各计算子图的执行指令序列，生成模型的可执行程序文件。

通常在生成模型对应的可执行程序文件后，需要对可执行程序文件进行调试，以获取运行性能较优的可执行程序文件。可以理解的是，如果直接对可执行程序文件进行调试，其对应的调试复杂度可能是指数级的，无法在一定时间内完成调试，确定出运行性能较优的可执行程序文件。

由于计算子图与计算子图之间往往是独立的，因此，本实施例中，以计算子图对应的执行指令序列为可执行程序文件的基本调试单位，通过对各个计算子图对应的执行指令序列分别进行调试，确定各个计算子图对应的运行性能较优的执行指令序列，最终将各个计算子图对应的运行性能较优的执行指令序列融合为可执行程序文件。

具体实施过程中，每个线程的计算任务并行度可进行灵活配置，基于前述实施例中的介绍，不同即计算任务并行度会有不同的计算任务分组结果，进而也会生成不同的执行指令序列。

因此，作为一种执行指令序列调试方式，可以先获取目标计算子图在多种计算任务并行度下对应的多个执行指令序列；然后，根据执行器分别执行多个执行指令序列时的运行性能，确定目标计算子图对应的目标执行指令序列；最后，融合多个计算子图分别对应的目标执行指令序列，以确定模型的可执行程序文件。

其中，执行器执行多个执行指令序列时的运行性能可以通过执行时间来度量。比如，目标计算子图在多种计算任务并行度下对应的多个执行指令序列分别为：执行指令序列 1 和执行指令序列 2，执行器执行该执行指令序列 1 的执行时间为 T1，执行该执行指令序列

2 的执行时间为 T_2 ，若 $T_1 > T_2$ ，则确定执行指令序列 2 为目标计算子图对应的目标执行指令序列。

本实施例中，以计算子图对应的执行指令序列为基本调试单位，每次需要被执行器执行测试的执行指令序列相较于可执行程序文件对应的全部执行指令序列来说是少量的，因此可以在短时间内，确定每个计算子图对应的目标执行指令序列，不仅能够得到运行性能较优的可行性程序文件，还能够提高调试效率。

图 6 为本申请实施例提供的又一种模型编译方法的流程图，如图 6 所示，该方法包括如下步骤：

601、获取待编译的模型对应的计算图中的目标计算子图，目标计算子图中包含多个访存密集型算子及相应数据，目标计算子图是模型对应的计算图中包含的一个计算子图。

602、根据模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是指依次采用多个访存密集型算子对相应数据进行计算的任务，计算任务并行度指示每个线程能够并行执行的计算任务数量，目标线程是多个线程中任一个。

603、针对任一组计算任务，根据多个访存密集型算子的计算依赖关系，对任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，重排使得同一访存密集型算子对应的不同计算任务被并行排列。

604、根据重排后的多个计算任务，生成目标线程与任一组计算任务对应的执行指令序列。

605、根据多个线程在目标计算子图下各自执行的多组计算任务分别对应的执行指令序列，确定目标计算子图对应的执行指令序列。

606、分别确定多个计算子图到达初始计算子图的距离；根据距离对多个计算子图进行分组，其中，同一组计算子图中各计算子图对应的距离相同；在每组计算子图对应的执行指令序列之后插入全局同步指令，以确定模型的可执行程序文件。

其中，步骤 601 至步骤 605 的具体实施过程可参考前述实施例，本实施例中不再赘述。

一般地，在确定计算子图对应的执行指令序列之后，会将每个计算子图对应的执行指令序列以函数的形式分别进行存储，该函数也可称为 GPU kernel。GPU 执行可执行程序文件，实际上是 CPU 将各计算子图分别对应的 GPU kernel 发送给 GPU 执行。

上述将各计算子图对应的 GPU kernel 分别进行存储的方式，在 GPU 执行可执行程序文件时，CPU 既要判断何时将哪一个计算子图对应的 GPU kernel 发送给 GPU 执行，又要频繁的调度各计算子图对应的 GPU kernel，会产生较大的非计算开销，也即调度开销。并且各计算子图分别对应于不同的 GPU kernel，还会由于 GPU kernel 带来的全局同步，造成一定的额外开销，其中，全局同步意味着 GPU kernel 对应的所有执行指令序列执行完毕，不再需要使用片上存储空间。另外，由于处理 GPU 任务的 CPU 通常因为数据处理等任务处于忙碌状态，且模型对应的可执行程序文件对应大量的 GPU kernel 又需要分开执行，会导致忙碌的 CPU 无法及时的调度这些 GPU kernel，而产生一定的性能损耗。

为解决上述至少一个技术问题，本实施例中，在确定模型计算图包含的多个计算子图分别对应的执行指令序列后，先分别确定多个计算子图到达初始计算子图的距离，其中初

始计算子图为计算图的输入数据对应的计算子图，也即计算图的第一个计算子图；然后，根据距离对多个计算子图进行分组，将对应的距离相同的计算子图归为同一组；最后，在每组计算子图对应的执行指令序列之后插入全局同步指令，以确定模型的可执行程序文件，其中，模型的可执行程序文件为一个 GPU kernel。

5 图 7 为本申请实施例提供的一种计算子图的示意图，结合图 7 举例说明本实施例中模型的可执行程序文件的确定过程。如图 7 所示，假设待编译的模型对应的计算图包含有 6 个计算子图，分别为计算子图 A、计算子图 B、计算子图 C、计算子图 D、计算子图 E 和计算子图 F（图 7 中用 A、B、C、D、E、F 表示），其中计算子图 A 为初始计算子图。

10 从图 7 中可以看出，一个计算子图到达初始计算子图的距离可能不止一个。以计算子图 C 为例，当计算子图 C 按照“A→C”的路径时，对应的到达初始计算子图的距离为 1；当计算子图 C 按照“A→D→F→C”的路径时，对应的到达初始计算子图的距离为 3。针对这种情形，本实施例中，确定计算子图 C 到达初始计算子图的距离为 3。

由此可以确定图 7 中计算子图 A、计算子图 B、计算子图 C、计算子图 D、计算子图 E 和计算子图 F 到达初始计算子图（即计算子图 A）的距离分别为：0、1、3、1、4、2。
15 由于计算子图 B 和计算子图 D 对应的距离均为 1，因此将计算子图 B 和计算子图 D 归为 1 组，计算子图 A、计算子图 C、计算子图 E 和计算子图 F 分别单独为 1 组。

最后，在 5 组计算子图对应的执行指令序列之后插入全局同步指令，以确定模型的可执行程序文件。其中，计算子图 E 对应分组作为最后一个分组，可以选择在其对应的执行指令序列之后插入或不插入全局同步指令。由于计算子图 B 和计算子图 D 属于同一分组，
20 在分组后插入全局同步指令后，相当于计算子图 B 和计算子图 D 对应的执行指令执行序列，总共需要进行一次全局同步，而上述每个计算子图分别对应一个 GPU kernel 的情形，则需要计算子图 B 和计算子图 D 对应的执行指令执行序列分别进行一次全局同步。因此，本实施例中的模型编译方法能够有效减少全局同步的数量，并且属于同一组的计算子图数量越多，这种效果会越明显。

25 本实施例中，模型的可执行程序文件为一个 GPU kernel，即每组计算子图对应的执行指令序列和全局同步指令作为一个函数进行存储，这样 GPU 在需要执行模型的可执行程序文件时，CPU 只需将模型对应的一个 GPU kernel 发送给 GPU 即可，CPU 无需判断在何时发送哪一个 GPU kernel 给 GPU，也不需要频繁调度 GPU kernel，从而能够有效减少非计算开销，以及 CPU 调度 GPU kernel 产生的性能损耗。另外，由于对各计算子图进行了
30 分组，并只在各分组之后插入了全局同步指令，从而能够减少全局同步的数量，提高计算效率，也即可执行程序文件的运行效率。

实际应用中，前述实施例中的模型编译方案可以相互结合，以实现更好的模型编译效果。

35 另外，需要说明的是，前述实施例所提供的模型编译方法中，虽然都是针对访存密集型算子相关计算任务的执行指令生成过程进行描述，但并不意味着模型对应的完整的可执行程序文件中仅包含与访存密集型算子相关计算任务的执行指令，模型对应的完整的可执行程序文件中还包含有与计算密集型算子相关计算任务的执行指令。

以下将详细描述本申请的一个或多个实施例的模型编译装置。本领域技术人员可以理

解，这些装置均可使用市售的硬件组件通过本方案所教导的步骤进行配置来构成。

图8为本申请实施例提供的一种模型编译装置的结构示意图，如图8所示，该装置包括：获取模块11和编译模块12。

5 获取模块11，用于获取待编译的模型对应的计算图中的目标计算子图，所述目标计算子图中包含多个访存密集型算子及相应数据，所述目标计算子图是所述计算图中包含的一个计算子图。

10 编译模块12，用于根据所述模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是指依次采用所述多个访存密集型算子对相应数据进行计算的任务，所述计算任务并行度指示每个线程能够并行执行的计算任务数量，所述目标线程是多个线程中任一个；针对任一组计算任务，根据所述多个访存密集型算子的计算依赖关系，对所述任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，所述重排使得同一访存密集型算子对应的不同计算任务被并行排列；根据所述重排后的多个计算任务，生成所述目标线程与所述任一组计算任务对应的执行指令序列；根据所述15 目标线程与所述任一组计算任务对应的执行指令序列，确定所述执行器执行所述模型的可执行程序文件，所述可执行程序文件中包含所述计算图包含的多个计算子图各自对应的执行指令序列。

20 在一实施方式中，所述编译模块12，还用于在所述目标线程与所述任一组计算任务对应的执行指令序列前，插入第一数据拷贝指令，所述第一数据拷贝指令用于将所述任一组计算任务的下一组计算任务对应的数据从所述执行器的片外存储空间拷贝到所述执行器的片上存储空间的第一缓冲区中，以使得在执行所述下一组计算任务时从所述第一缓冲区读取对应的数据。

25 在一实施方式中，所述编译模块12，具体用于若所述任一组计算任务是所述多组计算任务中的第一个计算任务，则在所述第一数据拷贝指令前，插入第二数据拷贝指令，所述第二数据拷贝指令用于将所述任一组计算任务对应的数据从所述执行器的片外存储空间拷贝到所述执行器的片上存储空间的第二缓冲区中。

30 其中，拷贝到所述第一缓冲区中的所述下一组计算任务对应的数据为所述下一组计算任务中第一个访存密集型算子所对应的数据；拷贝到所述第二缓冲区中的所述任一组计算任务对应的数据为所述任一组计算任务中第一个访存密集型算子所对应的数据。

30 在一实施方式中，所述编译模块12，还具体用于根据所述多个线程在所述目标计算子图下各自执行的多组计算任务分别对应的执行指令序列，确定所述目标计算子图对应的执行指令序列；融合所述计算图中多个计算子图分别对应的执行指令序列，以确定所述模型的可执行程序文件。

35 在一实施方式中，所述编译模块12，还具体用于获取所述目标计算子图在多种计算任务并行度下对应的多个执行指令序列；根据所述执行器分别执行所述多个执行指令序列时的运行性能，确定所述目标计算子图对应的目标执行指令序列；融合所述多个计算子图分别对应的目标执行指令序列，以确定所述模型的可执行程序文件。

在一实施方式中，所述编译模块12，还具体用于分别确定所述多个计算子图到达初始

计算子图的距离；根据所述距离对所述多个计算子图进行分组，其中，同一组计算子图中各计算子图对应的所述距离相同；在每组计算子图对应的执行指令序列之后插入全局同步指令，以确定所述模型的可执行程序文件。

5 本申请实施例提供的模型编译方案中，针对模型对应的计算子图中与访存密集型算子相关的计算任务进行执行指令优化编译。在对待编译的模型进行编译时，对模型对应的计算图所包含的多个计算子图分别进行编译。计算子图可以分为两类，一类是对应于访存密集型算子的计算子图，一类是对应于计算密集型算子的计算子图，本申请实施例中针对访存密集型算子对应的计算子图进行编译优化。具体地，在针对包含多个访存密集型算子及相应数据的目标计算子图进行编译时，根据模型的执行器（比如 GPU）中设定的多个线程和每个线程的计算任务并行度，确定每个线程需要执行的多个计算任务，并将每个线程需要执行的多个计算任务进行分组，确定多个线程中的任一个线程（称为目标线程）分别需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是指依次采用多个访存密集型算子对相应数据进行计算的任务，计算任务并行度指示每个线程能够并行执行的计算任务数量。接着，针对目标线程中的任一组计算任务，根据多个访存密集型算子的计算依赖关系，对任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，并根据重排后的多个计算任务，生成目标线程与任一组计算任务对应的执行指令序列，其中，重排是为了使同一访存密集型算子对应的不同计算任务被并行排列。最后，根据目标线程与任一组计算任务对应的执行指令序列，确定执行器执行所述模型的可执行程序文件，其中，可执行程序文件中包含计算图包含的多个计算子图各自对应的执行指令序列，即在该可执行程序文件中包含了模型的完整的执行指令，仅需要将该可执行程序文件一次性提供给执行器即可。

25 本实施例中，通过对线程需要执行的每组计算任务中同一访存密集型算子对应的不同计算任务进行并行排列，使得得到的执行指令序列具有较高的指令并行度，最终生成的可执行程序文件在被执行时，能够并行地发送并执行多个执行指令，以同时对同一访存密集型算子对应的不同计算任务进行并行计算，能够在执行器中的线程数量保持不变的情况下提高可执行程序文件的运行效率，充分利用执行器的资源。

图 8 所示的通信装置可以执行前述实施例中通信系统所执行的各个步骤，本实施例未详细描述的部分，可参考前述实施例的相关说明，在此不再赘述。

30 在一个可能的设计中，上述图 8 所示二维码识别装置的结构可实现为一电子设备。如图 9 所示，该电子设备可以包括：存储器 21、处理器 22、通信接口 23。其中，存储器 21 上存储有可执行代码，当所述可执行代码被处理器 22 执行时，使处理器 22 至少可以实现如前述实施例中提供的模型编译方法。

35 另外，本申请实施例提供了一种非暂时性机器可读存储介质，所述非暂时性机器可读存储介质上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器至少可以实现如前述实施例中提供的模型编译方法。

以上所描述的装置实施例仅仅是示意性的，其中所述作为分离部件说明的网元可以是或者也可以不是物理上分开的。可以根据实际的需要选择其中的部分或者全部模块来实现本实施例方案的目的。本领域普通技术人员在不付出创造性的劳动的情况下，即可以理解

并实施。

通过以上的实施方式的描述，本领域的技术人员可以清楚地了解到各实施方式可借助加必需的通用硬件平台的方式来实现，当然也可以通过硬件和软件结合的方式来实现。基于这样的理解，上述技术方案本质上或者说对现有技术做出贡献的部分可以以计算机产品的形式体现出来，本申请可采用在一个或多个其中包含有计算机空闲程序代码的计算机空闲存储介质（包括但不限于磁盘存储器、CD-ROM、光学存储器等）上实施的计算机程序产品的形式。

最后应说明的是：以上实施例仅用以说明本申请的技术方案，而非对其限制；尽管参照前述实施例对本申请进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本申请各实施例技术方案的精神和范围。

15

权 利 要 求

- 1、一种模型编译方法，所述方法包括：
- 获取待编译的模型对应的计算图中的目标计算子图，所述目标计算子图中包含多个访存密集型算子及相应数据，所述目标计算子图是所述计算图中包含的一个计算子图；
- 5 根据所述模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，每个计算任务是指依次采用所述多个访存密集型算子对相应数据进行计算的任务，所述计算任务并行度指示每个线程能够并行执行的计算任务数量，所述目标线程是多个线程中任一个；
- 针对任一组计算任务，根据所述多个访存密集型算子的计算依赖关系，对所述任一组
- 10 计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，所述重排使得同一访存密集型算子对应的不同计算任务被并行排列；
- 根据所述重排后的多个计算任务，生成所述目标线程与所述任一组计算任务对应的执行指令序列；
- 根据所述目标线程与所述任一组计算任务对应的执行指令序列，确定所述执行器执行
- 15 所述模型的可执行程序文件，所述可执行程序文件中包含所述计算图包含的多个计算子图各自对应的执行指令序列。
- 2、根据权利要求 1 所述的方法，所述方法还包括：
- 在所述目标线程与所述任一组计算任务对应的执行指令序列前，插入第一数据拷贝指令，所述第一数据拷贝指令用于将所述任一组计算任务的下一组计算任务对应的数据从所
- 20 述执行器的片外存储空间拷贝到所述执行器的片上存储空间的第一缓冲区中，以使得在执行所述下一组计算任务时从所述第一缓冲区读取对应的数据。
- 3、根据权利要求 2 所述的方法，所述方法还包括：
- 若所述任一组计算任务是所述多组计算任务中的第一个计算任务，则在所述第一数据拷贝指令前，插入第二数据拷贝指令，所述第二数据拷贝指令用于将所述任一组计算任务
- 25 对应的数据从所述执行器的片外存储空间拷贝到所述执行器的片上存储空间的第二缓冲区中。
- 4、根据权利要求 3 所述的方法，拷贝到所述第一缓冲区中的所述下一组计算任务对应的数据为所述下一组计算任务中第一个访存密集型算子所对应的数据；拷贝到所述第二缓冲区中的所述任一组计算任务对应的数据为所述任一组计算任务中第一个访存密集型
- 30 算子所对应的数据。
- 5、根据权利要求 1 至 4 中任一项所述的方法，所述根据所述目标线程与所述任一组计算任务对应的执行指令序列，确定所述执行器执行所述模型的可执行程序文件，包括：
- 根据所述多个线程在所述目标计算子图下各自执行的多组计算任务分别对应的执行指令序列，确定所述目标计算子图对应的执行指令序列；
- 35 融合所述计算图中多个计算子图分别对应的执行指令序列，以确定所述模型的可执行程序文件。
- 6、根据权利要求 5 所述的方法，所述方法还包括：
- 获取所述目标计算子图在多种计算任务并行度下对应的多个执行指令序列；

根据所述执行器分别执行所述多个执行指令序列时的运行性能，确定所述目标计算子图对应的目标执行指令序列；

所述融合所述计算图中多个计算子图分别对应的执行指令序列，以确定所述模型的可执行程序文件，包括：

5 融合所述多个计算子图分别对应的目标执行指令序列，以确定所述模型的可执行程序文件。

7、根据权利要求 5 所述的方法，所述融合所述计算图中多个计算子图分别对应的执行指令序列，以确定所述模型的可执行程序文件，包括：

分别确定所述多个计算子图到达初始计算子图的距离；

10 根据所述距离对所述多个计算子图进行分组，其中，同一组计算子图中各计算子图对应的所述距离相同；

在每组计算子图对应的执行指令序列之后插入全局同步指令，以确定所述模型的可执行程序文件。

8、一种模型编译装置，包括：

15 获取模块，用于获取待编译的模型对应的计算图中的目标计算子图，所述目标计算子图中包含多个访存密集型算子及相应数据，所述目标计算子图是所述计算图中包含的一个计算子图；

编译模块，用于根据所述模型的执行器中设定的多个线程和每个线程的计算任务并行度，确定目标线程需串行执行的多组计算任务，其中，每组计算任务中包含多个计算任务，
20 每个计算任务是指依次采用所述多个访存密集型算子对相应数据进行计算的任务，所述计算任务并行度指示每个线程能够并行执行的计算任务数量，所述目标线程是多个线程中任一个；针对任一组计算任务，根据所述多个访存密集型算子的计算依赖关系，对所述任一组计算任务中包含的多个计算任务进行顺序重排，以得到重排后的多个计算任务，其中，所述重排使得同一访存密集型算子对应的不同计算任务被并行排列；根据所述重排后的多
25 个计算任务，生成所述目标线程与所述任一组计算任务对应的执行指令序列；根据所述目标线程与所述任一组计算任务对应的执行指令序列，确定所述执行器执行所述模型的可执行程序文件，所述可执行程序文件中包含所述计算图包含的多个计算子图各自对应的执行指令序列。

9、一种电子设备，包括：存储器、处理器、通信接口；其中，所述存储器上存储有可执行代码，当所述可执行代码被所述处理器执行时，使所述处理器执行如权利要求 1 至 7 中任一项所述的模型编译方法。
30

10、一种非暂时性机器可读存储介质，所述非暂时性机器可读存储介质上存储有可执行代码，当所述可执行代码被电子设备的处理器执行时，使所述处理器执行如权利要求 1 至 7 中任一项所述的模型编译方法。
35