(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112306675 A
(43)申请公布日 2021.02.02

(21)申请号 202011084416.8

(22)申请日 2020.10.12

(71)申请人 网络通信与安全紫金山实验室
地址 211100 江苏省南京市江宁区秣周东路9号

(72)发明人 陈国海 马海波 黄永明 尤肖虎

(74)专利代理机构 江苏圣典律师事务所 32237
代理人 贺翔

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

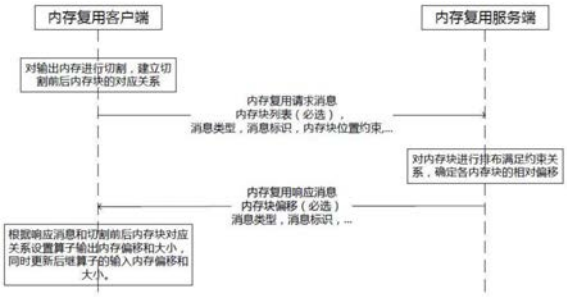
权利要求书2页 说明书10页 附图9页

(54)发明名称

数据处理方法、相关设备以及计算机可读存储介质

(57)摘要

本发明公开了一种数据处理方法、相关设备以及计算机可读存储介质，属于数据处理领域，其中数据处理方法包括：内存复用客户端对算子的输出内存进行切割，并建立切割后的多个内存块和切割前内存块的对应关系；所述内存复用客户端向内存复用服务器发出内存复用请求；所述内存复用客户端接收所述内存复用服务器的响应消息；所述内存复用客户端根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移；通过将算子的输入输出缓存分割成更小块的内存需求，这样更加容易去填补内存复用时的内存空洞，降低深度学习时模型算子输入输出缓存复用的内存总需求。



CN 112306675 A

1.一种数据处理方法,其特征在于,包括:

内存复用客户端对算子的输出内存进行切割,并建立切割后的多个内存块和切割前内存块的对应关系;

所述内存复用客户端向内存复用服务器发出内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;

所述内存复用客户端接收所述内存复用服务器的响应消息;其中所述响应消息包含待排布内存块的相对偏移;

所述内存复用客户端根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

2.根据权利要求1所述的数据处理方法,其特征在于:所述内存块列表包括切割后待排布的内存块的大小,待排布的内存块的标识,待排布内存块的生命起止时间,待排布内存块的位置约束关系。

3.根据权利要求1所述的数据处理方法,其特征在于:所述待排布内存块的相对偏移由所述内存复用服务器根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放所确定。

4.根据权利要求1所述的数据处理方法,其特征在于:所述内存复用请求还包括消息类型和请求标识;所述响应消息还包括消息类型和请求标识。

5.一种内存复用客户端,其特征在于,包括:

第一处理单元,用于对算子的输出内存进行切割,并建立切割后的多个内存块和切割前内存块的对应关系;

第一发送单元,用于向内存复用服务器发出内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;

第一接收单元,用于接收所述内存复用服务器的响应消息;其中所述响应消息包含待排布内存块的相对偏移;

第二处理单元,用于根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

6.根据权利要求5所述的内存复用客户端,其特征在于:所述内存块列表包括切割后待排布的内存块的大小,待排布的内存块的标识,待排布内存块的生命起止时间,待排布内存块的位置约束关系。

7.根据权利要求5所述的内存复用客户端,其特征在于:所述待排布内存块的相对偏移由所述内存复用服务器根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放所确定。

8.一种内存复用服务器,其特征在于,包括:

第二接收单元,用于接收内存复用客户端发出的内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;

第三处理单元,用于根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放,且保证各个内存块间不重叠从而确定待排布内存块的相对偏移;

第二发送单元，用于发送响应消息，其中所述响应消息包含待排布内存块的相对偏移。

9.根据权利要求6所述的内存复用服务器，其特征在于：所述内存块列表包括切割后待排布的内存块的大小，待排布的内存块的标识，待排布内存块的生命起止时间，待排布内存块的位置约束关系。

10.根据权利要求6所述的内存复用服务器，其特征在于：所述内存复用客户端用于对算子的输出内存进行切割，并建立切割后的多个内存块和切割前内存块的对应关系。

11.根据权利要求8所述的内存复用服务器，其特征在于：所述内存复用客户端还用于根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移，同时设置后缀算子的一个或者多个内存块的偏移和大小。

12.一种计算机可读存储介质，所述计算机可读存储介质存储有计算机程序，其特征在于，所述计算机程序被处理器执行时实现如权利要求1至3任一项所述的一种减少边缘计算内存占用的数据处理方法的步骤。

数据处理方法、相关设备以及计算机可读存储介质

技术领域

[0001] 本发明涉及数据处理领域，特别涉及一种能够减少边缘计算内存占用的数据处理方法、相关设备以及计算机可读存储介质。

背景技术

[0002] 近年伴随人工智能兴起，涌现出许许多多深度学习的框架(如TensorFlow，PyTorc，Caffe等)，各种深度学习模型不断涌出，深度学习在各行各业的应用给人们的生活带来了很多便利。比如车牌识别，各种语言间在线转换等。

[0003] 深度学习的神经网络模型如图1所示，最左边是输入层，最右边是输出层，中间可以理解为计算节点，形成一个层次概念。层上的节点越多，层次越深，深度学习的逻辑就越复杂。

[0004] 计算节点(或者称为算子)需要输入的数据进行计算，理所当然数据是需要存储，称这些存储为算子的输入缓存，算子计算完成后，数据需要输出，存储输出数据的缓存称为算子的输出缓存，输入输出缓存的数量有算子功能定义，比方说加法，将两个矩阵相加就需要有两个输入缓存，如果是三个输入矩阵就需要三个输入缓存；缓存的大小有输入数据的数量和数据类型共同决定。

[0005] 由于参与计算的计算节点多，需要处理的数据量大，整个计算需要的数据存储数量很大，通常以GB计算，这对计算系统提出了严格的要求。为适应手机上进行深度学习Google推出TensorFlow Lite，通过权重量化(将32bit的权重使用8bit表示)，从而减少模型存储时的大小和计算时的内存需求。因此需要对深度学习模型进行输入输出内存进行复用以减少需要的总的内存需求。

[0006] 内存复用的含义：就是将一块内存作为多个算子的输入或输出内存，由于算子的执行时顺序执行的(可以存在多个并行的顺序执行)，因此在不重叠的生命周期可以将一块内存块作为多个算子的输入输出缓存，形成内存复用。

[0007] Apache MXNet是一个深度学习框架，专为提高效率和灵活性而设计，给出了一个简单有效的启发方法，先确定需要的内存块的数量，然后计算每一个内存块的大小，具体如下。

[0008] 核心思想：允许变量(输入输出缓存)在不重叠的生命周期中共享内存，对每一块内存设计一个引用计数器并且着色(同意颜色表明是同一块内存)，当引用计数器等于0时表明这块内存可以被回收到内存块资源池，可以供其它算子使用。

[0009] 参考图2，确定需要内存块的数量过程如下，其中A是输入算子，需要独立的内存(并且输出复用输入)，每一个算子的输出缓存是后继算子的输入缓存，因此整个计算内存块数量的过程只需要考虑算子的输出缓存的分配。

[0010] Step1：算子B需要一块输出缓存，供算子C和F做输入缓存，内存池为空，因此需要新分配一块内存(红色)，引用计数为2(需要作为算子C和F的输入缓存)；

[0011] Step2：执行算子C，C算子需要一块输出缓存，内存池为空，分配一绿色的内存，引

用计数为1,同时红色内存块的引用计数减一(算子C执行完成);

[0012] Step3:执行算子F,F算子需要一块输出缓存,此时内存池为空,分配一蓝色内存,引用计数为1.算子F执行完成后红色内存块的引用计数减去1,此时引用计数为0,因此将红色内存块放入到内存块池;

[0013] Step4:执行算子E,需要输出缓存,缓存池中有一块红色的内存,因此无需申请新的内存块,E执行完成后,算子C的输出内存(绿色)块引用计数减一,计数等于0,放入内存块池;

[0014] Step5:执行算子G,根据算子特性,算子G的输出内存可以复用输入内存(即红色内存块),无需从内存池中分配新的内存块,算子G执行完成后,算子F的输出缓存块(蓝色)的引用计数减一。

[0015] 因此,不考虑初始的输入内存块,整个模型的推理执行需要3个内存块,分别是红色内存块,绿色内存块和蓝色内存块。

[0016] 然后根据每种颜色的内存块作为各个算子的输出缓存的最大值确定每个内存块的大小,所有内存块的总和就是算子执行过程中需要的总内存。

[0017] 由以上可以看到绿色和红色间存在一个1M的空间,而蓝色和紫色的生命周期是不重叠的,显然是浪费了部分存储空间。

发明内容

[0018] 为了解决现有的数据在运算过程中内存占用率大的问题,本发明提供一种能够减少边缘计算过程中内存占用的方法以及相关设备。

[0019] 为了实现上述目的,本发明第一方面提供一种数据处理方法,包括:

[0020] 内存复用客户端对算子的输出内存进行切割,并建立切割后的多个内存块和切割前内存块的对应关系;

[0021] 所述内存复用客户端向内存复用服务器发出内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;

[0022] 所述内存复用客户端接收所述内存复用服务器的响应消息;其中所述响应消息包含待排布内存块的相对偏移;

[0023] 所述内存复用客户端根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

[0024] 可选的,所述内存块列表包括切割后待排布的内存块的大小,待排布的内存块的标识,待排布内存块的生命起止时间,待排布内存块的位置约束关系。

[0025] 可选的,所述内存复用请求还包括消息类型和请求标识;所述响应消息还包括消息类型和请求标识。

[0026] 在上述数据处理方法中,可选的,所述待排布内存块的相对偏移由所述内存复用服务器根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放所确定。

[0027] 第二方面,本发明提供一种内存复用客户端,包括:

[0028] 第一处理单元,用于对算子的输出内存进行切割,并建立切割后的多个内存块和切割前内存块的对应关系;

[0029] 第一发送单元,用于向内存复用服务器发出内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;

[0030] 第一接收单元,用于接收所述内存复用服务器的响应消息;其中所述响应消息包含待排布内存块的相对偏移;

[0031] 第二处理单元,用于根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

[0032] 在上述内存复用客户端中,可选的,所述内存块列表包括切割后待排布的内存块的大小,待排布的内存块的标识,待排布内存块的生命起止时间,待排布内存块的位置约束关系。

[0033] 在上述内存复用客户端中,可选的,所述待排布内存块的相对偏移由所述内存复用服务器根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放所确定。

[0034] 第三方面,本发明提供一种内存复用服务器,包括:

[0035] 第二接收单元,用于接收内存复用客户端发出的内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;

[0036] 第三处理单元,用于根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放,且保证各个内存块间不重叠从而确定待排布内存块的相对偏移;

[0037] 第二发送单元,用于发送响应消息,其中所述响应消息包含待排布内存块的相对偏移。

[0038] 在上述内存复用服务器中,可选的,所述内存块列表包括切割后待排布的内存块的大小,待排布的内存块的标识,待排布内存块的生命起止时间,待排布内存块的位置约束关系。

[0039] 在上述内存复用服务器中,可选的,所述内存复用客户端用于对算子的输出内存进行切割,并建立切割后的多个内存块和切割前内存块的对应关系。

[0040] 在上述内存复用服务器中,可选的,所述内存复用客户端还用于根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

[0041] 本发明相对于现有技术的有益效果是:通过将算子的输入输出缓存分割成更小块的内存需求,这样更加容易去填补内存复用时的内存空洞,降低深度学习时模型算子输入输出缓存复用的内存总需求。相比分割前,内存复用中的空洞更少,内存复用的效率更高,需要的总内存更少,缓存与DDR主存间的数据交互更少,CPU等待时间更少,有利于提升CPU的使用率。

附图说明

[0042] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的实施例,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,还可以根据提供的附图获得其他的附图。

[0043] 图1是深度神经网络的模型图;

[0044] 图2是现有深度神经网络数据处理的流程图；

[0045] 图3是本发明中数据处理方法的流程图；

[0046] 图4是本发明中内存复用的示意图；

[0047] 图5是位置约束关系的示意图；

[0048] 图6和图7是Concat算子数据处理的示意图；

[0049] 图8是不重叠算法的示意图；

[0050] 图9是地址映射的示意图；

[0051] 图10和图11是边缘计算设备在离线模式时的数据处理流程图；

[0052] 图12是本发明中的数据处理方法应用在MEP或者MEC的APP中或者是专用硬件中流程图；

[0053] 图13是对算子的输出内存进行切割的示意图；

[0054] 图14是对算子的输出内存进行切割的流程图；

[0055] 图15是本发明对切割后的内存进行布放图；

[0056] 图16是本发明中的数据处理方法应用移动设备中流程图；

[0057] 图17是本发明中的数据处理方法应用(云)服务器中流程图；

[0058] 图18是本发明中内存复用客户端的结构图；

[0059] 图19是本发明中内存复用服务器的结构图。

具体实施方式

[0060] 下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅是本发明的一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0061] 参照图3，本实施例提供一种数据处理方法，包括以下步骤：

[0062] 步骤1，内存复用客户端(下面简称客户端)对算子的输出内存进行切割，按照最小可切割的内存块对可切割的内存进行切割，同时建立切割后的多个内存块和切割前内存块的对应关系；输出待复用的内存块列表，待复用的内存块列表包含切割后待排布的内存块的大小，待排布的内存块的标识，待排布内存块的生命起止时间，待排布内存块的位置约束关系；

[0063] 举例如下，如图4所示，算子B每次从输入缓存中取1K*1K矩阵8字节数据，也就是每次取1MB数据，可以将所需要的4MB缓存划分为4个待排布的内存块，每个内存块划分为1MB。缓存B(切割前)对应切割后的4个内存：缓存B1，缓存B2，缓存B3，缓存B4。

[0064] 在此需要说明的是，如图5所示，位置约束关系是指部分内存块存在必须相连的要求，通常发生在模型算子优化时，如Concat算子用于将2个或者更多个内存块数据合成一个大的内存块数据，如果输入是连续存放的，Concat算子此时就可以优化掉以节省深度学习进行推理时的时间。

[0065] 如图6所示，Concat算子需要将缓存A和缓存B中的数据读出写入到一块新的缓存C，在缓存C中数据相对位置不变，此时可以不申请缓存C，不读数据也不写数据节省大量操作，从而提升效率，直接将缓存A和缓存B连续存放即可，如图7所示。

[0066] 步骤2,客户端向内存复用服务器(下面简称服务器)发出内存复用请求,请求包含待复用的内存块列表/消息类型/请求标识,内存块列表为必选项,其余为可选项;列表项中包含内存块ID,内存块大小,生命周期起始和终止值,与其它内存块的位置约束等。

[0067] 应当理解的是,内存块标识,请求标识和消息类型可以是数字/文本或是数字+文本。

[0068] 步骤3,服务器接收到请求后,对内存块列表中的内存块使用不重叠算法确定内存块的偏移和大小;然后向客户端发送响应消息,响应消息中包含待排布内存块的相对偏移(必选)/消息类型/消息标识。

[0069] 不重叠算法介绍如下,如图8所示,1是已经布放好的内存,2布放的位置有A/B/C/D4种,A和C在内存空间上没有重叠,BD在生命周期上没有重叠。

[0070] 步骤4,客户端接收到响应消息后,参考切割后内存块和切割前内存块的对应关系和响应消息中的内存块的偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

[0071] 举例如下:响应消息中包含4个内存块的偏移{{B1,0x00000000},{B2,0x00100000},{B3,0x00200000},{B4,0x00300000}}。客户端检索对应关系是缓存B,每一块分割后的内存大小是1MB,缓存B对应算子A的第一个输出,其后缀是算子B的第二个输入。

[0072] 第一种设置方法:给算子A的第一个输出和算子B的第二个输入分别下4个偏移和大小{{1,0x00000000,0x100000},{2,0x00100000,0x100000},{3,0x00200000,0x100000},{4,0x00300000,0x100000}}。

[0073] 第二种设置方法:鉴于这4块内存是连续的,可以只设置一个偏移和大小。给算子A的第一个输出和算子B的第二个输入分别下1个偏移和大小{{1,0x00000000,0x400000}。如果分割后内存块不是依次连续递增的,需要建立读写地址映射表,判断读写的地址,然后给地址加上合适的偏移值,举例如下。

[0074] 服务器的响应中数据如下,{{1,0x00800000,0x100000},{2,0x00700000,0x100000},{3,0x00600000,0x100000},{4,0x00A00000,0x100000}}。

[0075] 算子读写数据的操作。由于数据的读写被分配到多块内存块中,算子的读写数据时需要进行地址的转换,这是非常容易做到的。举例如下,使用上面4个分割内存块为例,建立下面的转换关系,如图9所示。当访问范围大于等于0且小于0x100000时对应的偏移值是0x00800000,访问范围大于等于0x200000小于0x300000时对应的偏移值是0x00700000,对应到内存访问的地址就是从0x00A00000开始的存储。

[0076] 这里叠加偏移值是一个开销非常小的操作,相比于矩阵类的乘法和加法运算,开销可以忽略不计。

[0077] 参照图10,实际使用中可以在边缘计算设备之外先按照本发明的方法生成离线执行模型(各个输入输出的内存的多个内存块的相对偏移已经计算完成,存储在离线模型文件中)。

[0078] 参照图11,可以根据算子的输入输出的数据类型/需要访问的次数和一次基本操作涉及到的数据个数设置可以分割的内存块的大小和数量,内存复用客户端从请求对应的算子得到合理划分的指导,然后对内存块进行划分,再使用内存复用算法进行相对偏移的计算。

[0079] 该数据处理方法通过将算子的输入输出缓存分割成更小块的内存需求，这样更加容易去填补内存复用时的内存空洞，降低深度学习时模型算子输入输出缓存复用的内存总需求。相比分割前，内存复用中的空洞更少，内存复用的效率更高，需要的总内存更少，缓存与DDR主存间的数据交互更少，CPU等待时间更少，有利于提升CPU的使用率。

[0080] 以上为本申请中数据处理方法的原理，下面结合具体的应用对本申请中的数据处理方法做进一步介绍。需要说明的是，本发明中的客户端和服务器可以是下面的任何一种：云服务器，虚拟机，容器，一个进程，一个线程，函数，一段代码块。本发明可以在个人移动设备，计算云，多接入方式的边缘计算设备上实施，也可以是在一种设备上生成离线计算模型，在第二种设备执行，生成离线计算模型时将模型中算子的输入输出缓存分割然后实施内存复用算法。

[0081] 案例1：在多接入边缘计算上说明本发明。

[0082] 本实施例在MEP或者MEC(Multiple(mobile)Edge Computation)的APP中或者是专用硬件中使用本发明。

[0083] 如图12所示，APP中客户端和服务器可以是两个线程或者是一个线程中的两个函数，下面以两个函数说明。

[0084] 参照图13，函数A被调用时对输入内存进行切割，假设按照1MB进行切割。函数A中需要对4块内存块分配存储空间，4块内存分别是M1(2MB)，M2(1MB)，M3(1MB)，M4(2MB)，其中要求M2，M3，M4必须连续存放，并且M2的偏移大于M3的偏移。

[0085] 参照图14，同时假设M1是算子A的输出，算子B的第二个输入。

[0086] 步骤101：函数A将M1分割成M1_1(1MB)和M1_2(1MB)，M1内存块对应M1_1和M1_2，函数A使用函数参数构造请求消息，其中内存块列表如表1所示：

[0087] 表1

[0088]

内存块 ID	内存块大小	生命周期起始	生命周期终止	与其它内存块位置关系
M1_1	1MB	3	4	
M1_2	1MB	3	4	
M2	2MB	0	3	在 M3 上方，相邻
M3	1MB	1	2	在 M4 上方，相邻
M4	2MB	0	4	在 M3 下方，相邻

[0089] 步骤102：函数A调用函数B。内存块列表可以选用C++中的Vector或是List存放，具体内容参考表1，可选参数为请求标识省略，可选参数消息类型(请求内存复用，通过枚举值等表示)。

[0090] 步骤103：函数B根据入参中内存块列表，对内存块进行布放，保证各个内存块间不重叠，然后将布放结果使用MemResVec返回。针对上述的内存块列表，一种可能的布放结果如图15右侧所示。

[0091] 可以看到M1_1和M1_2不连续，两者间放置内存块M2，总共需要6MB内存；如果不将M1内存分割，布放结果可能上图左侧所示，需要7MB的内存。

[0092] 右侧返回的结果如下{{M4,0x00000000,0x00200000},{M3,0x00200000,0x00100000},{M1_1,0x00200000,0x00100000},{M2,0x00300000,0x00200000},{M1_2,0x00500000,0x00100000}}。

[0093] 步骤104：函数A在函数B调用返回后，参考分割前后内存对应关系，设置算子A的输出缓存和算子B的第二个输入缓存的多个内存块的偏移和大小。简化起见这里只描述算子A的输出缓存和算子B的第二个输入缓存。

[0094] 算子A的输出缓存{{out,1,0x00200000,0x00100000},{out,1,0x00500000,0x00100000}}，算子B的第二个输入缓存设置{{in,2,0x00200000,0x00100000},{in,2,0x00500000,0x00100000}}。其中in/out表示是算子的输入还是输入，第二个数字表示是算子的第几个输入输出。算子配置完成后在执行时按照数据的偏移加上对应的偏移值将数据写入到指定的内存块或者是从指定的内存块读出数据进行处理。

[0095] 具体案例二：在移动设备上说明本发明。

[0096] 在消费品中使用本发明，可以是笔记本电脑，平板或者是手机。

[0097] 参照图16，消费品中客户端和服务器可以是两个线程或者是一个线程中的两个函数，下面以两个线程说明。

[0098] 参照图13，线程A被调用时对输入内存进行切割，假设按照1MB进行切割。线程A中有4块内存需要分配空间，分别是M1(2MB)，M2(1MB)，M3(1MB)，M4(2MB)，其中要求M2，M3，M4必须连续存放，并且M2的偏移大于M3的偏移。

[0099] 参照图14，同时假设M1是算子A的输出，算子B的第二个输入。

[0100] 步骤201：线程A将M1分割成M1_1(1MB)和M1_2(1MB)，M1内存块对应M1_1和M1_2，线程A通过线程间通信发送内存复用请求消息，其中内存块列表如表2所示：

[0101] 表2

[0102]

内存块 ID	内存块大小	生命周期起始	生命周期终止	与其它内存块位置关系
M1_1	1MB	3	4	
M1_2	1MB	3	4	
M2	2MB	0	3	在 M3 上方，相邻
M3	1MB	1	2	在 M4 上方，相邻
M4	2MB	0	4	在 M3 下方，相邻

[0103] 步骤202：线程A通过线程间通信传递请求消息给线程B。内存块列表可以选用C++中的Vector或是List存放，具体内容参考表2，可选参数为请求标识省略，可选参数消息类型(请求内存复用，通过枚举值等表示)。结构化的数据通过序列化方法形成字节流发送，接收侧再反序列化获得序列化前的数据。

[0104] 步骤203:线程B根据消息中内存块列表,对内存块进行布放,保证各个内存块间不重叠,然后将布放结果使用MemResVec,序列化,然后通过线程间通信发送给线程A。针对上述的内存块列表,一种可能的布放结果如图15右侧所示。

[0105] 可以看到M1_1和M1_2不连续,两者间放置内存块M2,总共需要6MB内存;如果不将M1内存分割,布放结果可能上图左侧所示,需要7MB的内存。

[0106] 右侧返回的结果如下{{M4,0x00000000,0x00200000},{M3,0x00200000,0x00100000},{M1_1,0x00200000,0x00100000},{M2,0x00300000,0x00200000},{M1_2,0x00500000,0x00100000}}。

[0107] 步骤204:线程A在获得线程B响应消息后,参考分割前后内存对应关系,设置算子A的输出缓存和算子B的第二个输入缓存的多个内存块的偏移和大小。简化起见这里只描述算子A的输出缓存和算子B的第二个输入缓存。

[0108] 算子A的输出缓存{{out,1,0x00200000,0x00100000},{out,1,0x00500000,0x00100000}},算子B的第二个输入缓存设置{{in,2,0x00200000,0x00100000},{in,2,0x00500000,0x00100000}}。其中in/out表示是算子的输入还是输入,第二个数字表示是算子的第几个输入输出。算子配置完成后在执行时按照数据的偏移加上对应的偏移值将数据写入到指定的内存块或者是从指定的内存块读出数据进行处理。

[0109] 具体案例三:在(云)服务器上说明本发明。

[0110] 在(云)服务器上使用本发明,包含虚机,容器,或是非虚拟化的服务器。

[0111] 参照图17,客户端和服务器可以是两个线程(进程)或者是一个线程中的两个函数,以两个进程说明。

[0112] 参照图13,进程A被调用时对输入内存进行切割,假设按照1MB进行切割。进程A中有4块内存需要分配空间,分别是M1(2MB),M2(1MB),M3(1MB),M4(2MB),其中要求M2,M3,M4必须连续存放,并且M2的偏移大于M3的偏移。

[0113] 如图14所示,同时假设M1是算子A的输出,算子B的第二个输入。

[0114] 步骤301:进程A将M1分割成M1_1(1MB)和M1_2(1MB),M1内存块对应M1_1和M1_2,线程A通过线程间通信发送内存复用请求消息,其中内存块列表如表3所示:

[0115] 表3

[0116]

内存块 ID	内存块大小	生命周期起始	生命周期终止	与其它内存块位置关系
M1_1	1MB	3	4	
M1_2	1MB	3	4	
M2	2MB	0	3	在 M3 上方，相邻
M3	1MB	1	2	在 M4 上方，相邻
M4	2MB	0	4	在 M3 下方，相邻

[0117] 步骤302:进程A通过进程间通信传递请求消息给线程B。内存块列表可以选用C++

中的Vector或是List存放，具体内容参考表3，可选参数为请求标识省略，可选参数消息类型(请求内存复用，通过枚举值等表示)。结构化的数据通过序列化方法形成字节流发送，接收侧再反序列化获得序列化前的数据。

[0118] 步骤303：进程B根据消息中内存块列表，对内存块进行布放，保证各个内存块间不重叠，然后将布放结果使用MemResVec，序列化，然后通过进程间通信发送给进程A。针对上述的内存块列表，一种可能的布放结果如图15右侧所示。

[0119] 可以看到M1_1和M1_2不连续，两者间放置内存块M2，总共需要6MB内存；如果不将M1内存分割，布放结果可能上图左侧所示，需要7MB的内存。

[0120] 右侧返回的结果如下{{M4,0x00000000,0x00200000},{M3,0x00200000,0x00100000},{M1_1,0x00200000,0x00100000},{M2,0x00300000,0x00200000},{M1_2,0x00500000,0x00100000}}。

[0121] 步骤304：进程A在获得进程B响应消息后，参考分割前后内存对应关系，设置算子A的输出缓存和算子B的第二个输入缓存的多个内存块的偏移和大小。简化起见这里只描述算子A的输出缓存和算子B的第二个输入缓存。

[0122] 算子A的输出缓存{{out,1,0x00200000,0x00100000},{out,1,0x00500000,0x00100000}}，算子B的第二个输入缓存设置{{in,2,0x00200000,0x00100000},{in,2,0x00500000,0x00100000}}。其中in/out表示是算子的输入还是输入，第二个数字表示是算子的第几个输入输出。算子配置完成后在执行时按照数据的偏移加上对应的偏移值将数据写入到指定的内存块或者是从指定的内存块读出数据进行处理。

[0123] 本实施例中的数据处理方法通过将算子的输入输出缓存分割成更小块的内存需求，这样更加容易去填补内存复用时的内存空洞，降低深度学习时模型算子输入输出缓存复用的内存总需求。相比分割前，内存复用中的空洞更少，内存复用的效率更高，需要的总内存更少，缓存与DDR主存间的数据交互更少，CPU等待时间更少，有利于提升CPU的使用率。

[0124] 在一些实施例中，本发明还提供一种内存复用客户端，如图18所示，该内存复用客户端包括：

[0125] 第一处理单元101，用于对算子的输出内存进行切割，并建立切割后的多个内存块和切割前内存块的对应关系；具体的处理过程在上述数据处理方法的步骤1中已经详细的描述过，在此不再赘述。

[0126] 第一发送单元102，用于向内存复用服务器发出内存复用请求；其中，所述内存复用请求包括待复用的内存块列表；内存块列表包括切割后待排布的内存块的大小，待排布的内存块的标识，待排布内存块的生命起止时间，待排布内存块的位置约束关系。具体的处理过程在上述数据处理方法的步骤2中已经详细的描述过，在此不再赘述。

[0127] 第一接收单元103，用于接收所述内存复用服务器的响应消息；其中所述响应消息包含待排布内存块的相对偏移；所述待排布内存块的相对偏移由所述内存复用服务器对所述内存块列表中的内存块进行布放，且保证各个内存块间不重叠所确定。具体的处理过程在上述数据处理方法的步骤3中已经详细的描述过，在此不再赘述。

[0128] 第二处理单元104，用于根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移，同时设置后缀算子的一个或者多个内存块的偏移和大小。具体的处理过程在上述数据处理方法的步骤4中已经详

细的描述过,在此不再赘述。

[0129] 在另外一些实施例中,本发明提供一种内存复用服务器,如图19所示,该内存复用服务器包括:

[0130] 第二接收单元201,用于接收内存复用客户端发出的内存复用请求;其中,所述内存复用请求包括待复用的内存块列表;其中,所述内存块列表包括切割后待排布的内存块的大小,待排布的内存块的标识,待排布内存块的生命起止时间,待排布内存块的位置约束关系。

[0131] 第三处理单元202,用于根据所述内存块列表中的信息对所述内存块列表中的内存块进行布放,且保证各个内存块间不重叠从而确定待排布内存块的相对偏移;具体的数据处理过程在上述数据处理方法的步骤3中已经详细的描述过,在此不再赘述。

[0132] 第二发送单元203,用于发送响应消息,其中所述响应消息包含待排布内存块的相对偏移。

[0133] 另外,所述内存复用客户端用于对算子的输出内存进行切割,并建立切割后的多个内存块和切割前内存块的对应关系。

[0134] 另外,所述内存复用客户端还用于根据切割后内存块和切割前内存块的对应关系和响应消息中的内存块的相对偏移设置算子输出内存的一个或者多个偏移,同时设置后缀算子的一个或者多个内存块的偏移和大小。

[0135] 另外,本发明实施例还提供一种计算机可读存储介质,其中,该计算机可读存储介质可存储有程序,该程序执行时包括上述方法实施例中记载的任何一种数据处理方法的部分或全部步骤。

[0136] 另外,在本发明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。上述集成的单元既可以采用硬件的形式实现,也可以采用软件功能单元的形式实现。

[0137] 所述集成的单元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时,可以存储在一个计算机可读取存储器中。基于这样的理解,本发明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全部或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器中,包括若干指令用以使得一台计算机设备(可为个人计算机、服务器或者网络设备等)执行本发明各个实施例所述方法的全部或部分步骤。而前述的存储器包括:U盘、只读存储器(ROM,Read-Only Memory)、随机存取存储器(RAM,Random Access Memory)、移动硬盘、磁碟或者光盘等各种可以存储程序代码的介质。

[0138] 本领域普通技术人员可以理解上述实施例的各种方法中的全部或部分步骤是可以通过程序来指令相关的硬件来完成,该程序可以存储于一计算机可读存储器中,存储器可以包括:闪存盘、只读存储器(英文:Read-Only Memory,简称:ROM)、随机存取器(英文:Random Access Memory,简称:RAM)、磁盘或光盘等。

[0139] 以上参照附图描述了根据本发明的实施例的用于实现服务链的方法的示例性流程图。应指出的是,以上描述中包括的大量细节仅是对本发明的示例性说明,而不是对本发明的限制。在本发明的其他实施例中,该方法可具有更多、更少或不同的步骤,且各步骤之间的顺序、包含、功能等关系可以与所描述和图示的不同。

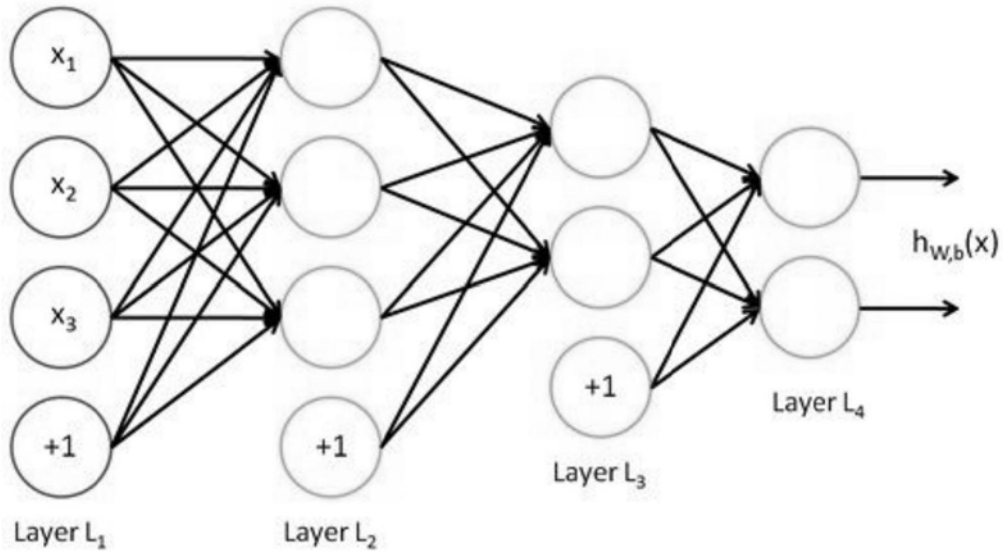

x_1
x_2
x_3
+1
+1
+1
$h_{W,b}(x)$
Layer L_1
Layer L_2
Layer L_3
Layer L_4

图1

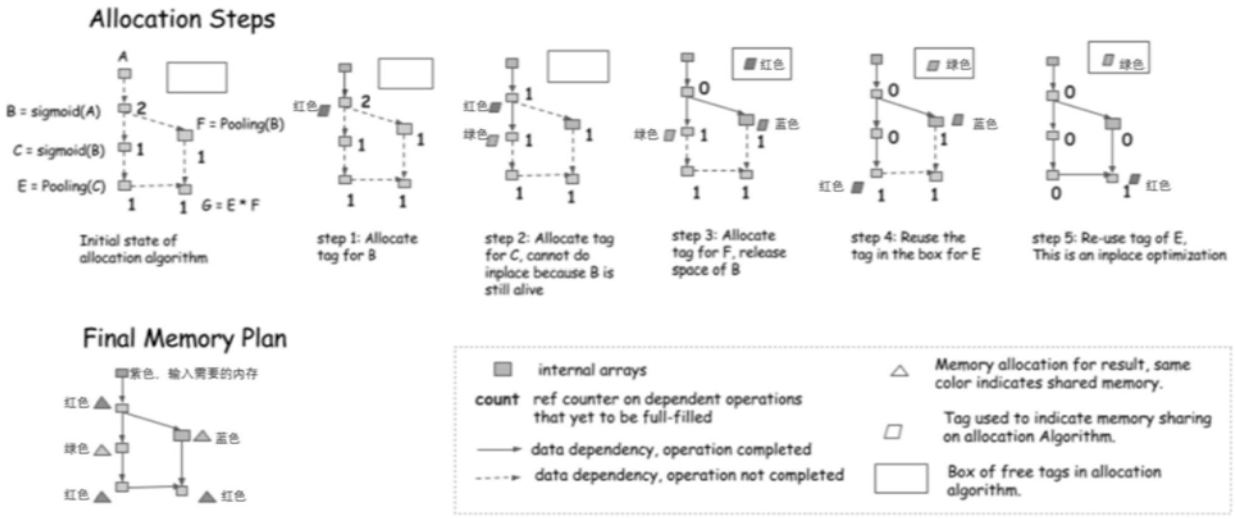

Allocation Steps
B = sigmoid(A)
C = sigmoid(B)
E = Pooling(C)
F = Pooling(B)
G = E * F
Initial state of allocation algorithm
step 1: Allocate tag for B
step 2: Allocate tag for C, cannot do inplace because B is still alive
step 3: Allocate tag for F, release space of B
step 4: Reuse the tag in the box for E
step 5: Re-use tag of E, This is an inplace optimization
Final Memory Plan
internal arrays
count ref counter on dependent operations that yet to be full-filled
data dependency, operation completed
data dependency, operation not completed
Memory allocation for result, same color indicates shared memory.
Tag used to indicate memory sharing on allocation Algorithm.
Box of free tags in allocation algorithm.

图2

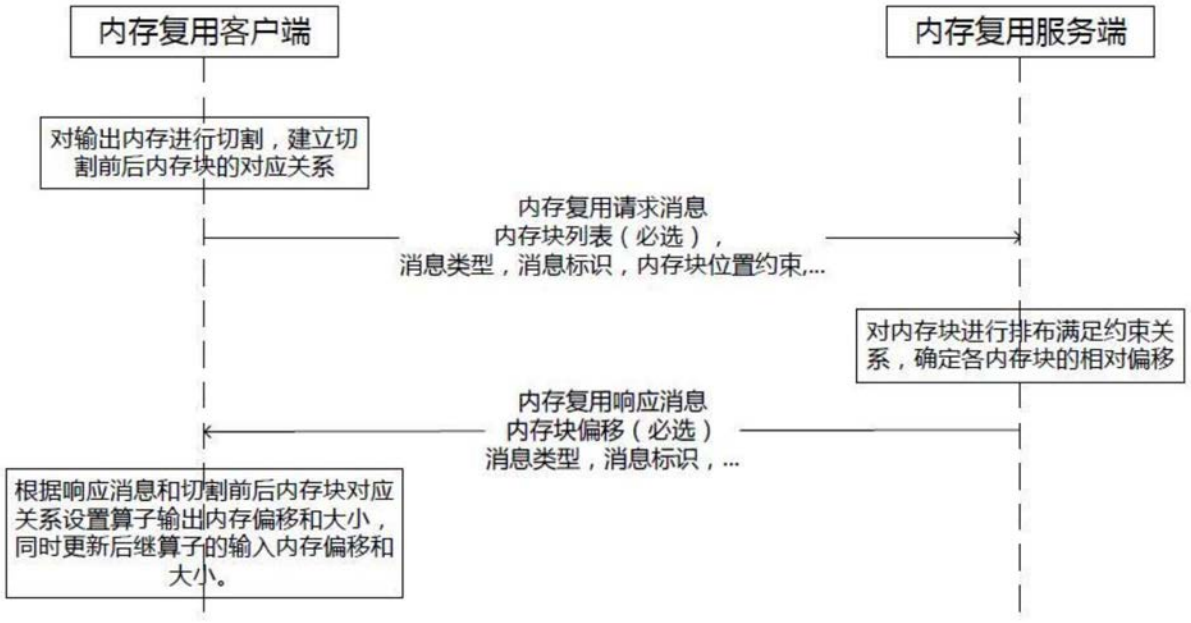

内存复用客户端
内存复用服务端
对输出内存进行切割，建立切割前后内存块的对应关系
内存复用请求消息
内存块列表（必选），
消息类型，消息标识，内存块位置约束,...
对内存块进行排布满足约束关系，确定各内存块的相对偏移
内存复用响应消息
内存块偏移（必选）
消息类型，消息标识，...
根据响应消息和切割前后内存块对应关系设置算子输出内存偏移和大小，同时更新后继算子的输入内存偏移和大小。

图3


缓存
算子A
缓存B
4MB
算子B
缓存

图4

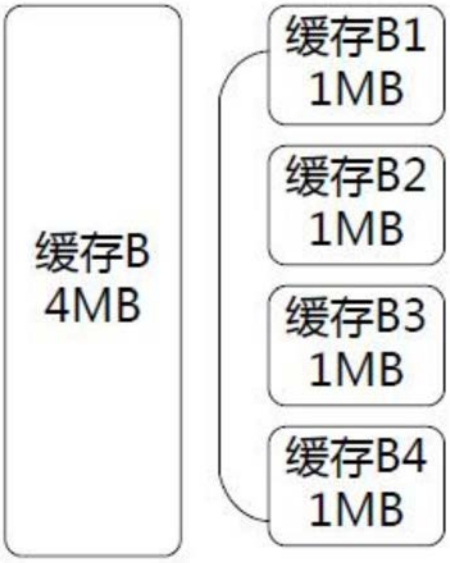

缓存B
4MB
缓存B1
1MB
缓存B2
1MB
缓存B3
1MB
缓存B4
1MB

图5

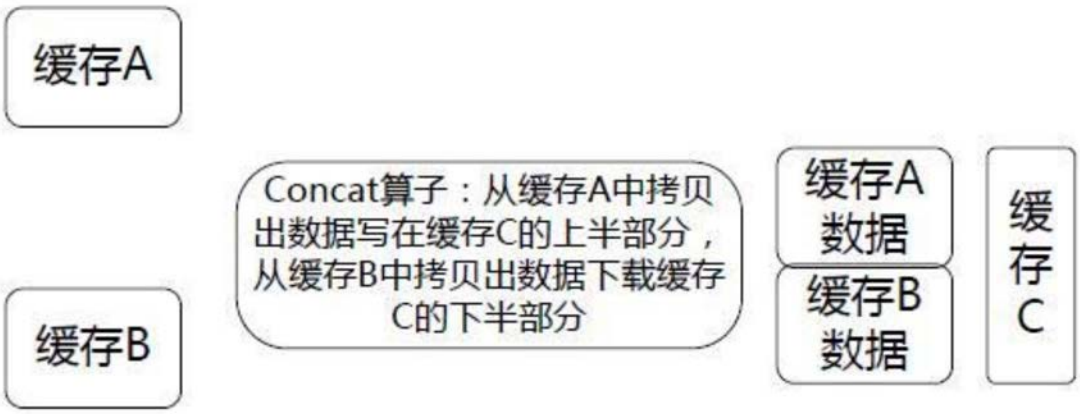

缓存A
缓存B
Concat算子：从缓存A中拷贝出数据写在缓存C的上半部分，从缓存B中拷贝出数据下载缓存C的下半部分
缓存A数据
缓存B数据
缓存C

图6


缓存A
缓存B
Concat算子
不执行任何操作
缓存A
缓存B

图7

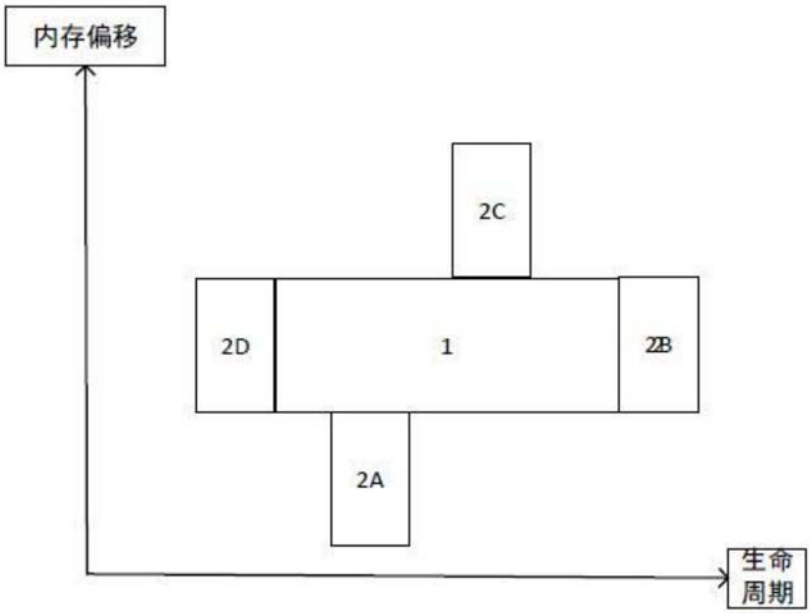

内存偏移
2C
2D
1
2B
2A
生命周期

图8

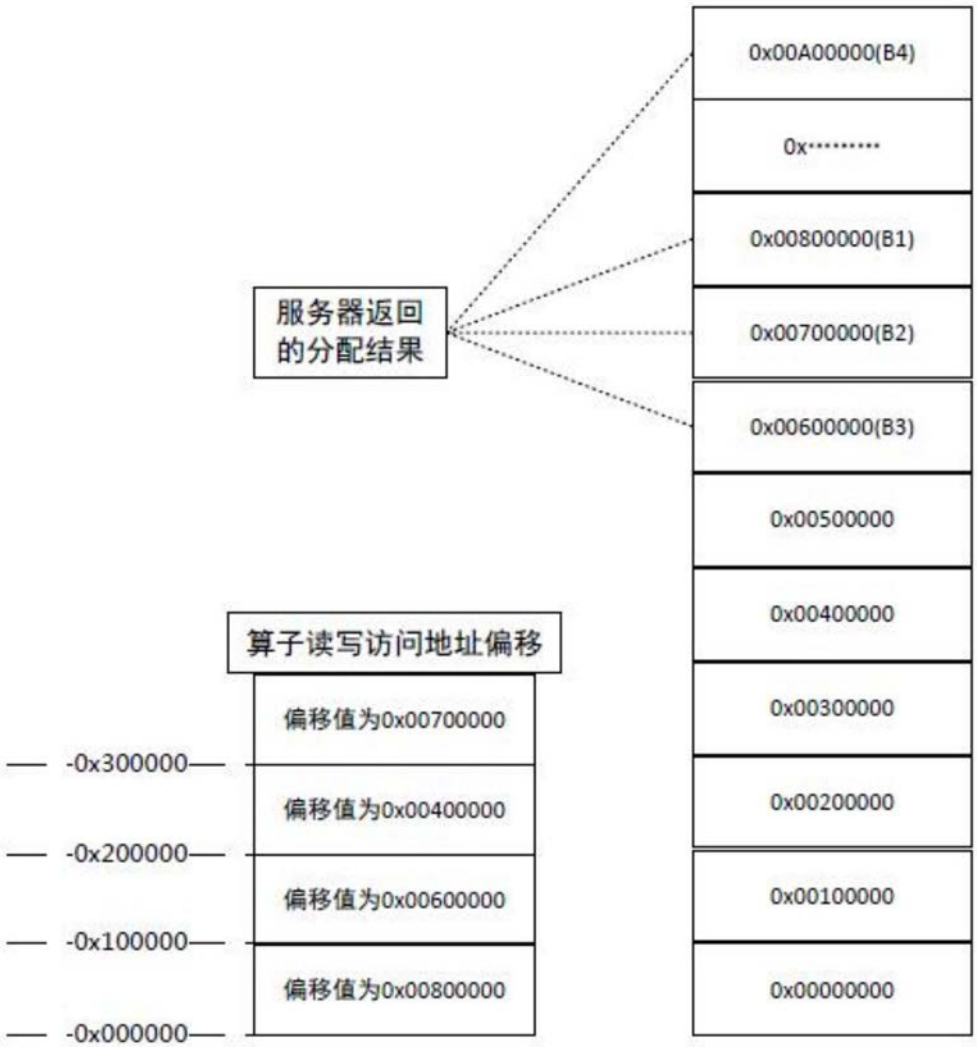

0x00A00000(B4)
0x·····
0x00800000(B1)
0x00700000(B2)
0x00600000(B3)
0x00500000
0x00400000
0x00300000
0x00200000
0x00100000
0x00000000
服务器返回的分配结果
算子读写访问地址偏移
偏移值为0x00700000
偏移值为0x00400000
偏移值为0x00600000
偏移值为0x00800000
-0x300000
-0x200000
-0x100000
-0x000000

图9


服务器A
生成本发明生成深度学习的离线推理模型
离线推理模型文件
边缘计算执行器（APP，专用硬件）

图10

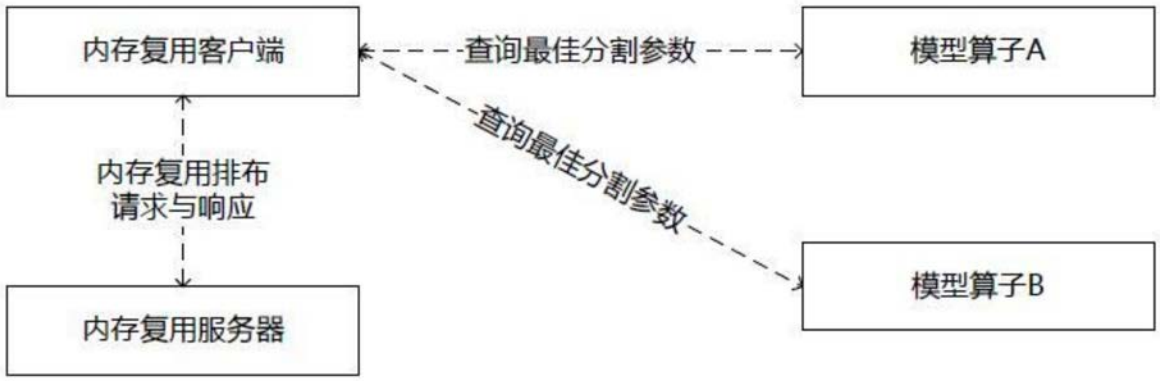

内存复用客户端
查询最佳分割参数
模型算子A
查询最佳分割参数
内存复用排布
请求与响应
模型算子B
内存复用服务器

图11

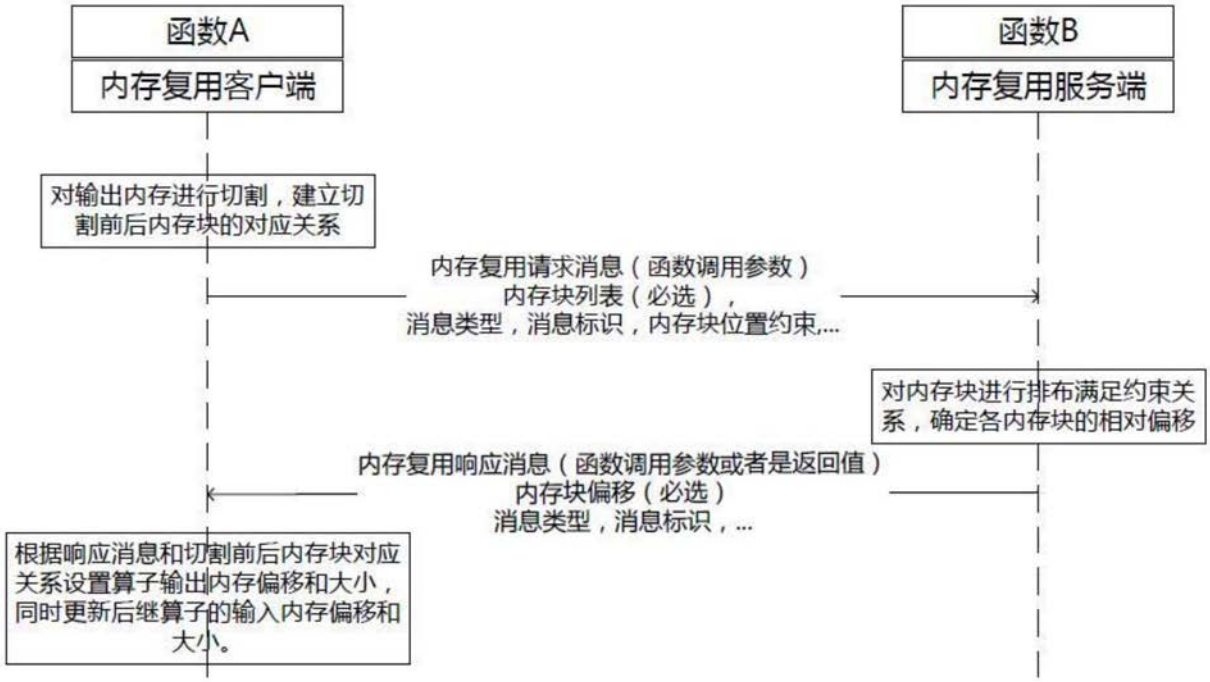

函数A
内存复用客户端
函数B
内存复用服务端
对输出内存进行切割，建立切割前后内存块的对应关系
内存复用请求消息（函数调用参数）
内存块列表（必选），
消息类型，消息标识，内存块位置约束,...
对内存块进行排布满足约束关系，确定各内存块的相对偏移
内存复用响应消息（函数调用参数或者是返回值）
内存块偏移（必选）
消息类型，消息标识，...
根据响应消息和切割前后内存块对应关系设置算子输出内存偏移和大小，同时更新后继算子的输入内存偏移和大小。

图12

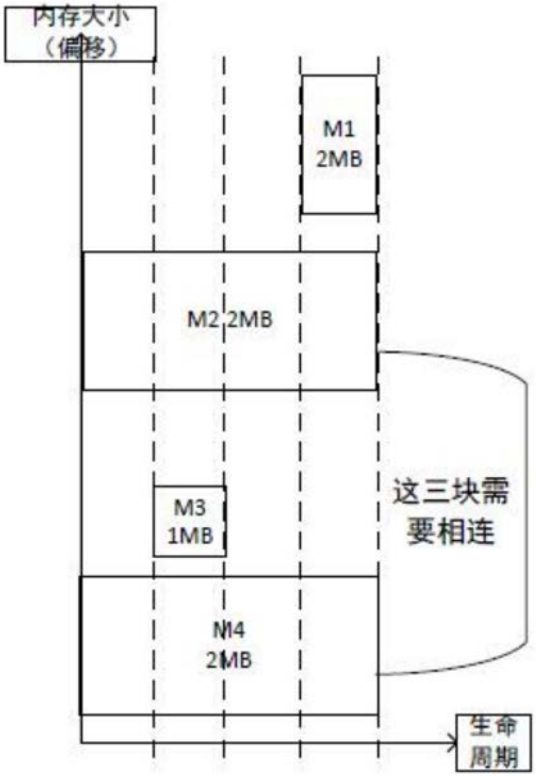

内存大小
（偏移）
M1
2MB
M2 2MB
M3
1MB
M4
2MB
这三块需
要相连
生命
周期

图13


缓存M4
2MB
算子A
缓存M1
2MB
算子B

图14

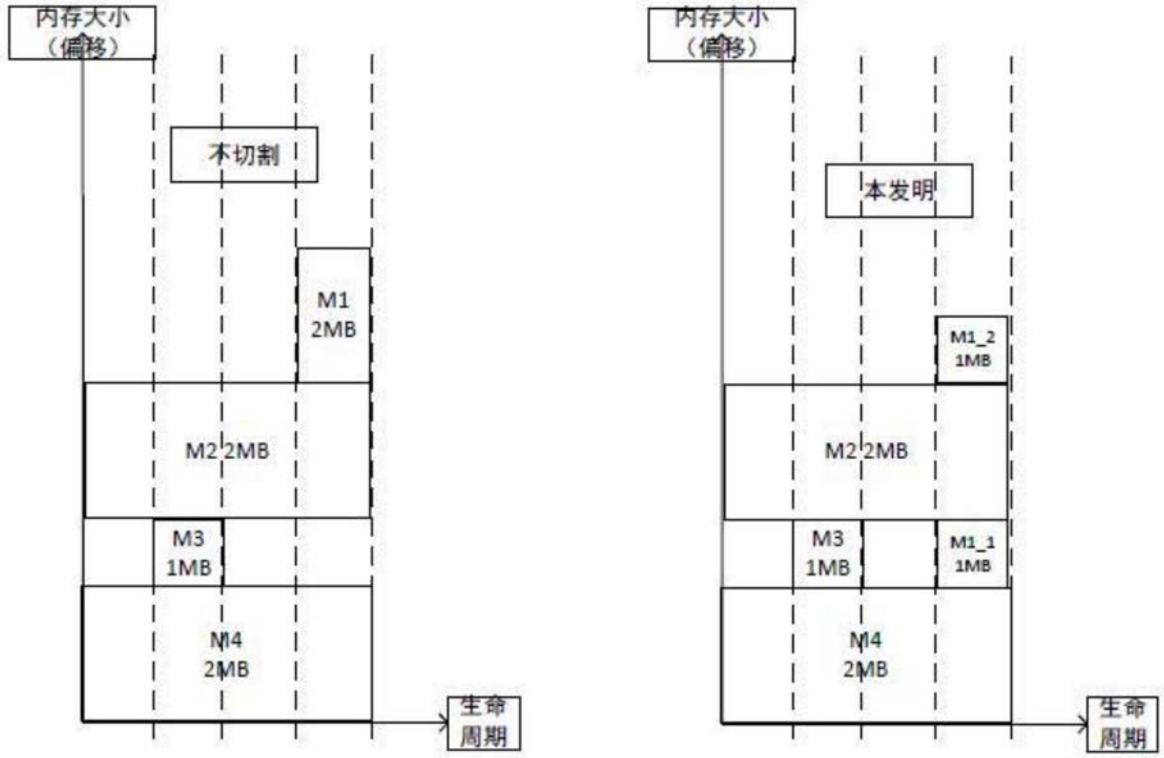

内存大小
（偏移）
本切割
M1
2MB
M2 2MB
M3
1MB
M4
2MB
生命
周期
内存大小
（偏移）
本发明
M1_2
1MB
M2 2MB
M3
1MB
M1_1
1MB
M4
2MB
生命
周期

图15

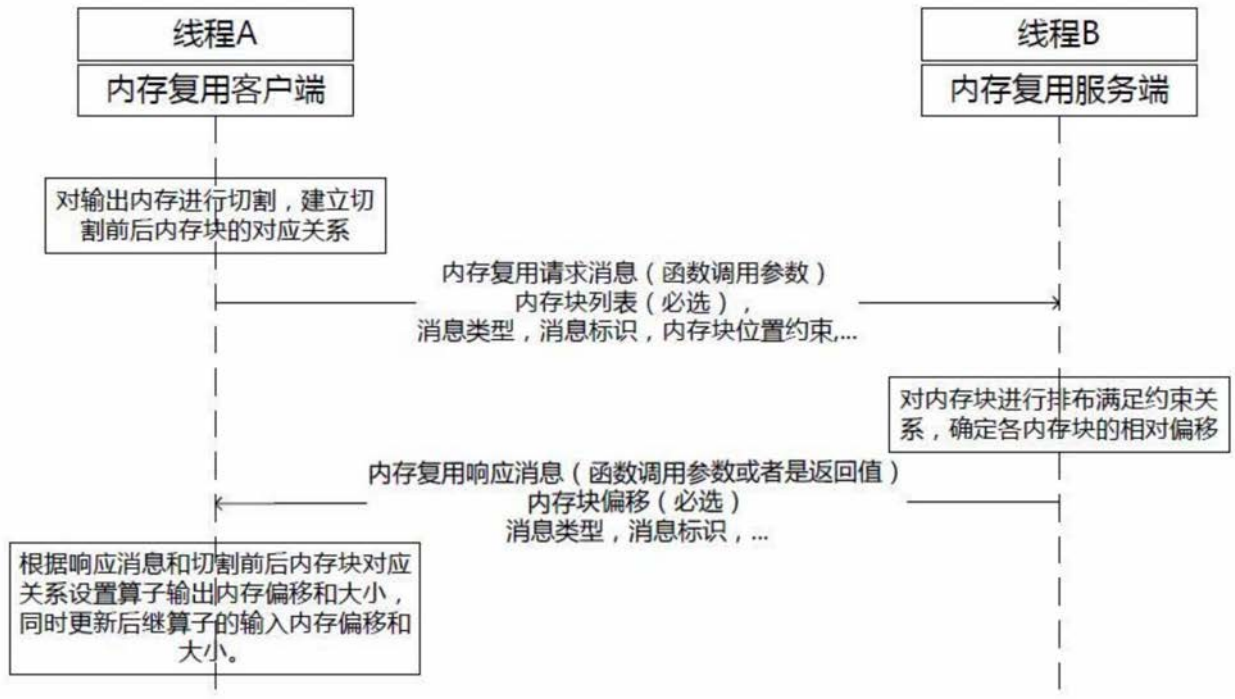

线程A
内存复用客户端
线程B
内存复用服务端
对输出内存进行切割，建立切割前后内存块的对应关系
内存复用请求消息（函数调用参数）
内存块列表（必选），
消息类型，消息标识，内存块位置约束,...
对内存块进行排布满足约束关系，确定各内存块的相对偏移
内存复用响应消息（函数调用参数或者是返回值）
内存块偏移（必选）
消息类型，消息标识，...
根据响应消息和切割前后内存块对应关系设置算子输出内存偏移和大小，同时更新后继算子的输入内存偏移和大小。

图16

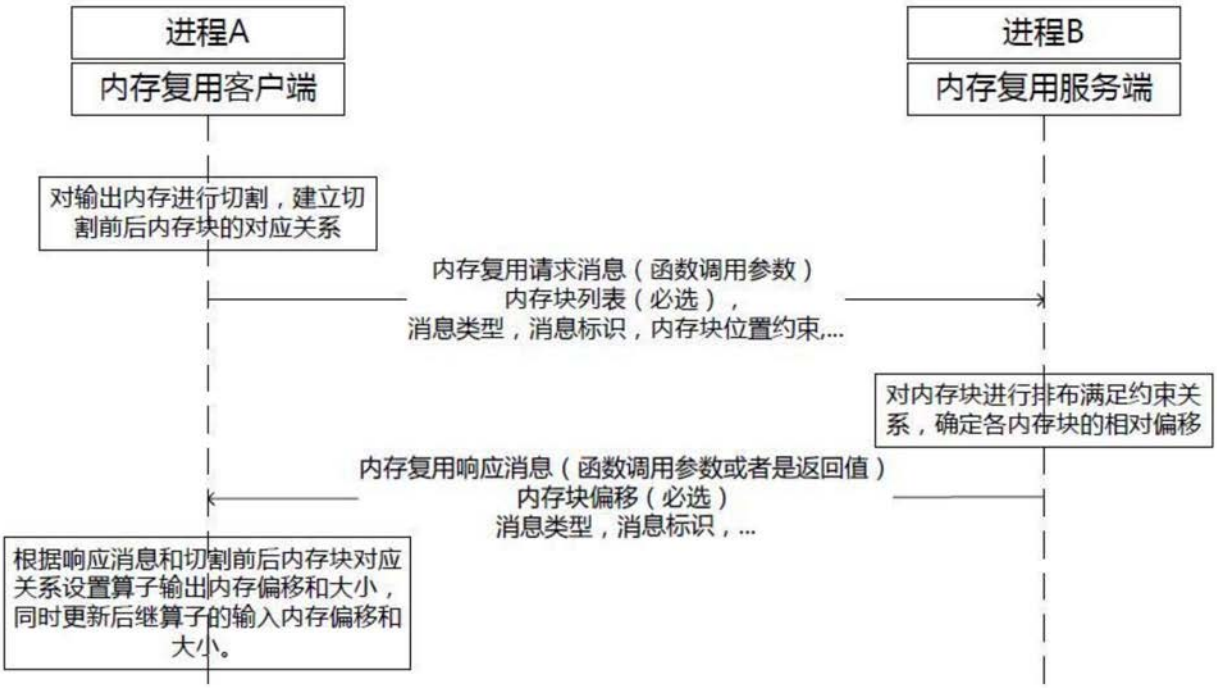

进程A
内存复用客户端
进程B
内存复用服务端
对输出内存进行切割，建立切割前后内存块的对应关系
内存复用请求消息（函数调用参数）
内存块列表（必选），
消息类型，消息标识，内存块位置约束...
对内存块进行排布满足约束关系，确定各内存块的相对偏移
内存复用响应消息（函数调用参数或者是返回值）
内存块偏移（必选）
消息类型，消息标识，...
根据响应消息和切割前后内存块对应关系设置算子输出内存偏移和大小，同时更新后继算子的输入内存偏移和大小。

图17


101
第一处理单元
102
第一发送单元
103
第一接收单元
104
第二处理单元
内存复用客户端

图18

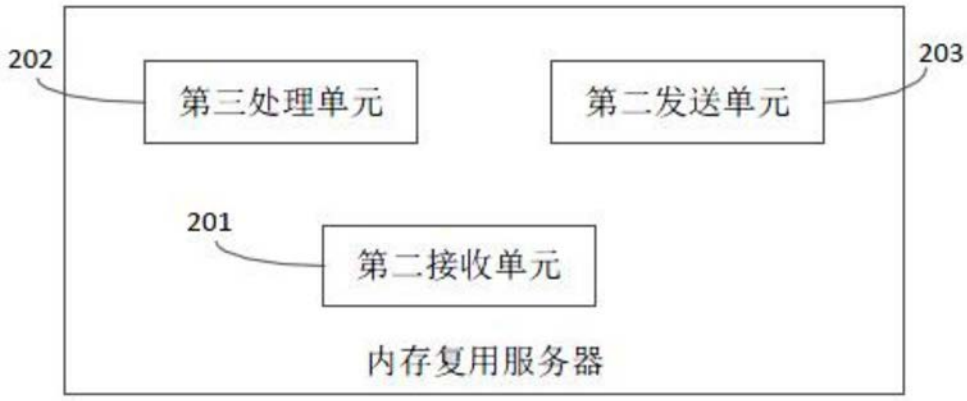


图19