(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110727911 B
(45)授权公告日 2022.09.02

(21)申请号 201810783363.5

(22)申请日 2018.07.17

(65)同一申请的已公布的文献号
申请公布号 CN 110727911 A

(43)申请公布日 2020.01.24

(73)专利权人 展讯通信(上海)有限公司
地址 201203 上海市浦东新区浦东张江高科技园区祖冲之路2288弄展讯中心1号楼

(72)发明人 杰里米·布兰斯科姆

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
专利代理师 朱薇蕾 吴敏

(51)Int.Cl.
G06F 17/16(2006.01)

(56)对比文件
CN 108037908 A,2018.05.15

审查员 苏星晔

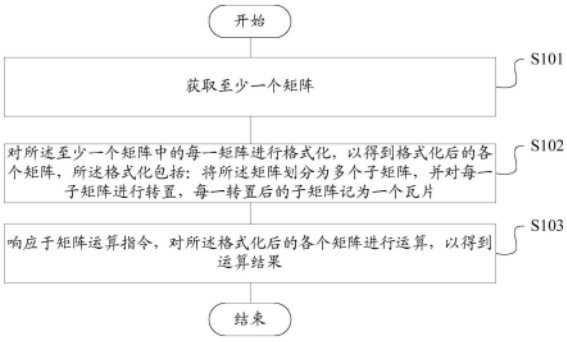
权利要求书2页 说明书13页 附图14页

(54)发明名称

一种矩阵的运算方法及装置、存储介质、终端

(57)摘要

一种矩阵的运算方法及装置、存储介质、终端，所述方法包括：获取至少一个矩阵；对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，所述格式化包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片；响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果。通过本发明提供的方案能够基于更高效的数据格式进行矩阵运算，尤其在多进程场景中能够极大减少数据运算量，优化计算机到内存的传输带宽需求，降低运算期间的执行延迟。



CN 110727911 B

1.一种矩阵的运算方法,其特征在于,包括:

获取至少一个矩阵;

对所述至少一个矩阵中的每一矩阵进行格式化,以得到格式化后的各个矩阵,所述格式化包括:将所述矩阵划分为多个子矩阵,并对每一子矩阵进行转置,每一转置后的子矩阵记为一个瓦片;

响应于矩阵运算指令,对所述格式化后的各个矩阵进行运算,以得到运算结果;

其中,所述格式化还包括:沿行方向或列方向依次获取各个瓦片,并存储得到多个次级子矩阵,每一次级子矩阵包括阵列排布的至少一个瓦片,其中,所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵。

2.根据权利要求1所述的矩阵的运算方法,其特征在于,所述获取至少一个矩阵包括:获取第一矩阵和第二矩阵;所述响应于矩阵运算指令,对所述格式化后的各个矩阵进行运算,以得到运算结果包括:响应于矩阵相乘运算指令,对格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算,以得到所述运算结果。

3.根据权利要求2所述的矩阵的运算方法,其特征在于,所述矩阵相乘运算是以所述瓦片为单位执行的。

4.根据权利要求1所述的矩阵的运算方法,其特征在于,所述转置操作是基于寄存器实现的。

5.根据权利要求1所述的矩阵的运算方法,其特征在于,所述子矩阵包括的元素为4×4个。

6.一种矩阵的运算装置,其特征在于,包括:

获取模块,用于获取至少一个矩阵;

格式化模块,用于对所述至少一个矩阵中的每一矩阵进行格式化,以得到格式化后的各个矩阵,所述格式化包括:将所述矩阵划分为多个子矩阵,并对每一子矩阵进行转置,每一转置后的子矩阵记为一个瓦片;

运算模块,响应于矩阵运算指令,对所述格式化后的各个矩阵进行运算,以得到运算结果;

其中,所述格式化还包括:沿行方向或列方向依次获取各个瓦片,并存储得到多个次级子矩阵,每一次级子矩阵包括阵列排布的至少一个瓦片,其中,所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵。

7.根据权利要求6所述的矩阵的运算装置,其特征在于,所述获取模块包括:获取子模块,用于获取第一矩阵和第二矩阵;所述运算模块包括:矩阵相乘运算子模块,响应于矩阵相乘运算指令,对格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算,以得到所述运算结果。

8.根据权利要求7所述的矩阵的运算装置,其特征在于,所述矩阵相乘运算是以所述瓦片为单位执行的。

9.根据权利要求6所述的矩阵的运算装置,其特征在于,所述转置操作是基于寄存器实现的。

10.根据权利要求6所述的矩阵的运算装置,其特征在于,所述子矩阵包括的元素为4×4个。

11.一种存储介质,其上存储有计算机指令,其特征在于,所述计算机指令在处理器上运行时执行权利要求1至5任一项所述方法的步骤。

12.一种终端,包括存储器和处理器,所述存储器上存储有能够在所述处理器上运行的计算机指令,其特征在于,所述处理器运行所述计算机指令时执行权利要求1至5任一项所述方法的步骤。

一种矩阵的运算方法及装置、存储介质、终端

技术领域

[0001] 本发明涉及计算机技术领域，具体地涉及一种矩阵的运算方法及装置、存储介质、终端。

背景技术

[0002] 现有计算机在存储数据时，通常会将数据以矩阵的形式进行存储，数据在矩阵内是以线性字节阵列的形式排布的。而计算机在执行运算操作时，通常会以分步形式发出一个或多个指令来实现相应的数据处理。

[0003] 以线性字节阵列为例，现有的运算操作执行方案不会考虑到优化计算机到内存的传输带宽需求的方式，数据计算量大，尤其在多进程时极易导致执行延迟，使得现有的矩阵运算计算量大、时延严重。

发明内容

[0004] 本发明解决的技术问题是如何有效提高矩阵运算效率，减少数据运算量，降低运算期间的执行延迟。

[0005] 为解决上述技术问题，本发明实施例提供一种矩阵的运算方法，包括：获取至少一个矩阵；对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，所述格式化包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片；响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果。

[0006] 可选的，所述格式化还包括：沿行方向或列方向依次获取各个瓦片，并存储得到多个次级子矩阵，每一次级子矩阵包括阵列排布的至少一个瓦片，其中，所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵。

[0007] 可选的，所述获取至少一个矩阵包括：获取第一矩阵和第二矩阵；所述响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果包括：响应于矩阵相乘运算指令，对格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算，以得到所述运算结果。

[0008] 可选的，所述矩阵相乘运算是以所述瓦片为单位执行的。

[0009] 可选的，所述转置操作是基于寄存器实现的。

[0010] 可选的，所述子矩阵包括的元素为4×4个。

[0011] 为解决上述技术问题，本发明实施例还提供一种矩阵的运算装置，包括：获取模块，用于获取至少一个矩阵；格式化模块，用于对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，所述格式化包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片；运算模块，响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果。

[0012] 可选的，所述格式化还包括：沿行方向或列方向依次获取各个瓦片，并存储得到多

个次级子矩阵，每一次级子矩阵包括阵列排布的至少一个瓦片，其中，所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵。

[0013] 可选的，所述获取模块包括：获取子模块，用于获取第一矩阵和第二矩阵；所述运算模块包括：矩阵相乘运算子模块，响应于矩阵相乘运算指令，对格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算，以得到所述运算结果。

[0014] 可选的，所述矩阵相乘运算是以所述瓦片为单位执行的。

[0015] 可选的，所述转置操作是基于寄存器实现的。

[0016] 可选的，所述子矩阵包括的元素为4×4个。

[0017] 为解决上述技术问题，本发明实施例还提供一种存储介质，其上存储有计算机指令，所述计算机指令运行时执行上述方法的步骤。

[0018] 为解决上述技术问题，本发明实施例还提供一种终端，包括存储器和处理器，所述存储器上存储有能够在所述处理器上运行的计算机指令，所述处理器运行所述计算机指令时执行上述方法的步骤。

[0019] 与现有技术相比，本发明实施例的技术方案具有以下有益效果：

[0020] 本发明实施例提供一种矩阵的运算方法，包括：获取至少一个矩阵；对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，所述格式化包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片；响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果。较之现有的矩阵运算方式，通过在实际执行运算操作前对待处理的矩阵进行格式化处理，本发明实施例的方案能够有效减少矩阵运算时的内核消耗，优化寄存器利用率。本领域技术人员理解，采用本发明实施例的方案，能够基于更高效的数据格式进行矩阵运算，尤其在多进程场景中能够极大减少数据运算量，优化运算期间从计算机到内存的传输带宽需求，降低运算期间的执行延迟。

[0021] 进一步，所述格式化还包括：沿行方向或列方向依次获取各个瓦片，并存储得到多个次级子矩阵，每一次级子矩阵包括阵列排布的至少一个瓦片，其中，所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵，以提供一种更高效的数据格式，实现跨线程的数据并行化。本领域技术人员理解，采用本发明实施例的方案，由于运算时是以格式化后的矩阵为输入数据进行矩阵运算的，而格式化后的数据是以瓦片或次级子矩阵为单位存储的，从而最小化存储必须被读取和写入的频率，以提高数据访问效率。

附图说明

[0022] 图1是本发明实施例的一种矩阵的运算方法的流程图；

[0023] 图2是图1中步骤S102的一种具体实施方式的流程图；

[0024] 图3是本发明实施例的一种格式化操作的典型应用场景的示意图；

[0025] 图4是本发明实施例的一种矩阵相乘运算方法的流程图；

[0026] 图5是本发明实施例所述矩阵相乘运算的一个典型应用场景的示意图；

[0027] 图6是本发明实施例的一种次级子矩阵的示意图；

[0028] 图7是本发明实施例的一种采用图6所示的次级子矩阵进行矩阵相乘运算的示意图；

[0029] 图8是本发明所述矩阵相乘运算的另一个典型应用场景的示意图；

[0030] 图9是图8所示应用场景中对每一切片进行矩阵相乘运算的流程框图；

[0031] 图10是本发明实施例的一个典型应用场景的张量切片描述符及对应的张量切片的示意图；

[0032] 图11是本发明实施例的另一个典型的应用场景的运算描述符及对应的张量切片的示意图；

[0033] 图12是采用图11所示应用场景的矩阵运算流程框图；

[0034] 图13是本发明实施例的又一个典型应用场景的阶层式运算的示意图；

[0035] 图14是图13的一个变化的应用场景的阶层式运算的示意图；

[0036] 图15和图16是本发明实施例的又一个典型应用场景的递归运算在不同的运算阶段的结构框图；

[0037] 图17是本发明实施例的一种矩阵的运算装置的结构示意图。

具体实施方式

[0038] 本领域技术人员理解，如背景技术所言，现有的计算机在进行数据处理时存在数据计算量大、延迟严重等问题，导致矩阵运算的运算效率低、运算期间计算机到内存的带宽大。

[0039] 为了解决上述技术问题，本发明实施例提供一种矩阵的运算方法，包括：获取至少一个矩阵；对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，所述格式化包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片；响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果。

[0040] 本领域技术人员理解，通过在实际执行运算操作前对待处理的矩阵进行格式化处理，本发明实施例的方案能够有效减少矩阵运算时的内核消耗，优化寄存器利用率。

[0041] 进一步地，采用本发明实施例的方案，能够基于更高效的数据格式进行矩阵运算，尤其在多进程场景中能够极大减少数据运算量，优化运算期间从计算机到内存的传输带宽需求，降低运算期间的执行延迟。

[0042] 为使本发明的上述目的、特征和有益效果能够更为明显易懂，下面结合附图对本发明的具体实施例做详细的说明。

[0043] 图1是本发明实施例的一种矩阵的运算方法的流程图。其中，所述矩阵可以包括以阵列形式存储于计算机(如计算机的内存(memory))的数据，如线性字节阵列等，还可以包括以其他形式存储并需要以矩阵形式进行运算的数据；所述运算可以包括矩阵相乘，还可以包括其他的逻辑运算形式。

[0044] 具体地，参考图1，在本实施例中，所述矩阵的运算方法可以包括如下步骤：

[0045] 步骤S101，获取至少一个矩阵。

[0046] 步骤S102，对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，其中，所述格式化可以包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片。

[0047] 步骤S103，响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运

算结果。

[0048] 更为具体地，结合图1和图2，所述步骤S102中的格式化操作还可以包括如下步骤：

[0049] 步骤S1021，沿行方向或列方向依次获取各个瓦片。

[0050] 步骤S1022，存储所述瓦片得到多个次级子矩阵，每一次级子矩阵包括阵列排布的至少一个瓦片，其中，所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵。

[0051] 其中，步骤S1021中沿行方向或列方向获取的操作是在所述矩阵的范围内进行的。具体而言，沿行方向获取是在所述矩阵的范围内，在前一行依次获取各个瓦片后，再转至下一行依次获取各个瓦片，直至获取矩阵范围内的全部瓦片。本文中，其他“沿行方向”指的也是类似的含义，也即在适当的范围内按照先行后列的顺序进行操作。相应地，“沿列方向”是在适当的范围内按照先列后行的顺序进行操作。

[0052] 进一步地，所述格式化后的矩阵可以以覆盖的形式存储于所述矩阵的存储地址。例如，在对取自内存的矩阵进行格式化处理后，可以原路存回所述内存。或者，所述格式化后的矩阵也可以与所述矩阵并行地存储于所述内存的不同地址上。

[0053] 或者，所述格式化后的矩阵也可以缓存于计算机的寄存器(register)中，以利于后续运算时的快速读取。

[0054] 作为一个非限制性实施例，通过转置获取的所述矩阵，能够将所述矩阵包括的单元进行行列互换操作，从而获得转置后的矩阵，所述转置操作可以是基于所述寄存器实现的。

[0055] 在一个典型的应用场景中，参考图3，接下来以对图3示出的矩阵10进行格式化操作为例作具体阐述。其中，所述矩阵10为存储于内存中的线性字节阵列。

[0056] 具体地，所述矩阵10的步幅(stride)为N，且N≥1；所述矩阵10划分为B行N列，也即共包括B×N(以下和附图中简化为BN)个单元(cell)a；每一单元a内包括4个元素(Vloads)，例如，第0行第0列的单元a包括{0,1,2,3}四个元素，第B-1行第N-1列的单元a包括{BN-4,BN-3,BN-2,BN-1}四个元素；本场景示出的矩阵10为行主序矩阵，也即所述矩阵10的读方向为行方向(在图3中以箭头方向标示)。为了简化，图3中每个单元a上仅列出该单元a包括的第一个元素。

[0057] 进一步地，结合图1至图3，在对所述矩阵10进行格式化操作时，首先将所述矩阵10划分为多个4×4的子矩阵20(图3仅示例性地示出划分出的其中一个子矩阵20)，并沿所述矩阵10的行方向读取划分出的所述子矩阵20并写入寄存器。其中，所述寄存器的写方向如图3所示。

[0058] 需要指出的是，本实施例的子矩阵20和矩阵10都是以单元a为最小描述单位的。

[0059] 进一步地，对所述子矩阵20进行转置操作。例如，对于4×4的子矩阵20，可以采用四个寄存器对其进行转置操作，以获取图3所示的转置后的子矩阵30。其中，关于基于寄存器进行转置操作的具体流程可以参考现有基于寄存器进行的转置操作方案，在此不予赘述。

[0060] 其中，所述子矩阵20和转置后的子矩阵(也即所述瓦片)30均包括V0至V3四行。

[0061] 进一步地，在对划分获得的所有子矩阵20均完成转置操作后，可以沿行方向线性存储各瓦片30至所述内存，以获取所述格式化后的矩阵。此时，所述格式化后的矩阵也是行主序的矩阵。

[0062] 作为一个变化例，当需要获取列主序的转置后的矩阵时，可以沿列方向线性存储各瓦片至所述内存，以获取列主序的转置后的矩阵。

[0063] 在实际应用中，行主序的转置后的矩阵(或者行主序的瓦片)可以是由行主序的矩阵转置获得的；列主序的转置后的矩阵(或列主序的瓦片)可以是由行主序或列主序的矩阵转置获得的。

[0064] 本领域技术人员理解，所述寄存器可以用于转置和格式化所述矩阵10。例如，在后续运算时，可以将所述寄存器作为格式化后的矩阵的临时数据源，还可以作为运算结果的临时累加器。

[0065] 进一步地，以获得行主序的转置后的矩阵为例，在基于所述寄存器获得各瓦片后，可以沿行方向依次获取各瓦片，并存储至所述内存以获得多个次级子矩阵。

[0066] 作为一个非限制性实施例，当每一次级子矩阵包括一个瓦片时，可以认为所述瓦片就是沿行方向线性存储至所述内存的，当然，存储的最小单位为瓦片，而非单元。

[0067] 作为另一个非限制性实施例，当每一次级子矩阵(quadrille)包括多个瓦片时，例如，每个次级子矩阵均包括m×n个瓦片，其中，m＞1且n＞1，此时，对于每一次级子矩阵，均包括沿行方向线性(也即，在m×n的范围内先行方向后列方向的顺序)存储的m×n个瓦片，而所述多个次级子矩阵整体上在所述内存内是沿行方向线性存储的。需要指出的是，本示例中，m可以等于n，或者，m也可以不等于n。

[0068] 相应的，为了获得列主序的转置后的矩阵，可以沿列方向阵列存储所述瓦片。

[0069] 由此，本实施例的方案提供了一种更高效率的矩阵格式化方案，通过将原始矩阵包括的元素格式化为以瓦片为最小单位的形式，能够最小化存储必须被读取和写入的频率，降低运算时的数据运算量，优化计算机到内存的带宽。

[0070] 进一步地，在本实施例中，对转置后的矩阵的运算均是以所述瓦片为单位执行的。

[0071] 接下来以矩阵相乘运算为例做具体阐述。也即，在本场景中，所述矩阵相乘运算是以所述瓦片为单位执行的。

[0072] 具体地，参考图4，在本场景中，所述矩阵相乘运算方法可以包括如下步骤：

[0073] 步骤S1011，获取第一矩阵和第二矩阵。

[0074] 步骤S1021’，分别对所述第一矩阵和第二矩阵进行格式化，以得到格式化后的第一矩阵和格式化后的第二矩阵。

[0075] 步骤S1031，响应于矩阵相乘运算指令，对所述格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算，以得到所述运算结果。

[0076] 其中，所述第一矩阵和第二矩阵可以为相互独立的两个矩阵。

[0077] 关于所述步骤S1021’中对各矩阵的格式化操作的具体内容，可以参考上述图3中的相关描述，在此不予赘述。

[0078] 以所述格式化后的第一矩阵为行主序矩阵，格式化后的第二矩阵为列主序矩阵为例，相应的，所述第一矩阵可以为行主序矩阵，所述第二矩阵可以为行主序矩阵或列主序矩阵。

[0079] 在一个典型的应用场景中，参考图5，所述格式化后的第一矩阵X包括M×N个沿行方向阵列排布的次级子矩阵，为了简化，图5中以每一次级子矩阵包括一个瓦片30为例进行阐述，因而，图5示出的格式化后的第一矩阵X包括M×N个沿行方向阵列排布的瓦片30。其

中,每一行可以记为一个切片(slice)30'。

[0080] 类似的,所述格式化后的第二矩阵Y包括N×Q个沿列方向阵列排布的瓦片30。其中,每一列可以记为一个切片30'。

[0081] 则所述格式化后的第一矩阵X和所述格式化后的第二矩阵Y的矩阵相乘运算的输出Z包括M×Q个沿行方向阵列排布的瓦片30,其中,输出Z中的瓦片30为格式化后的第一矩阵X和格式化后的第二矩阵Y各自的瓦片30的矩阵相乘结果。

[0082] 由此,较之现有技术直接对所述第一矩阵和第二矩阵进行矩阵相乘运算的实施方式,在本应用场景中,能够以小常数C(约为2至4)倍少量增加格式化成本(平方级)的代价显著降低(约为现有技术的1/2倍)矩阵相乘运算时的内核消耗(立方级)。

[0083] 具体地,采用现有技术进行矩阵相乘运算的格式化成本和内核消耗可以基于如下公式表示:

[0084] K×N^2+N^3;

[0085] 其中,K为转置与内核迭代的成本比,N为所述第一矩阵的列数(即第二矩阵的行数),N^2为N的平方,N^3为N的三次方;

[0086] 相应的,基于本应用场景进行矩阵相乘运算的格式化成本和内核消耗可以基于如下公式表示:

[0087] C×K×N^2+(1/2)N^3;

[0088] 由于N>>2(C×K-1),所以,可以得出不等式:

[0089] C×K×N^2+(1/2)N^3<<K×N^2+N^3;

[0090] 也即,基于本应用场景进行的矩阵相乘运算可以极大的减少运算量。

[0091] 进一步,较之现有的线性点方案,采用本实施例的矩阵运算能够有效减少每一乘积项的带宽负担,从而优化寄存器的利用率(较现有技术可优化近2至4倍)。

[0092] 进一步,在对某一行切片和某一列切片进行矩阵相乘运算时,该行切片和列切片包括的瓦片30均存储于寄存器中,从而最小化存储必须被读取和写入的频率,以提高数据访问效率。

[0093] 进一步,基于本实施例的方案还可以实现跨线程的数据并行化。例如,可以按切片分配初始的格式化后的第一矩阵X和格式化后的第二矩阵Y。又例如,在对所述格式化后的第一矩阵X包括的行切片进行矩阵相乘运算期间,可以将所述格式化后的第二矩阵Y包括的列切片共享在缓存中,以进一步最小化存储必须被读取和写入的频率。

[0094] 进一步地,在获取所述输出Z后,所述输出Z也可以是以次级子矩阵的形式存储于所述内存中的,其中,所述次级子矩阵可以包括至少一个阵列排布的瓦片。

[0095] 进一步地,所述矩阵运算指令可以是基于改进的RISC微处理器(Advanced RISC Machines,简称ARM)的无符号点积UDOT指令发送的。

[0096] 接下来以所述次级子矩阵包括多个阵列排布的瓦片为例进行具体阐述。

[0097] 在一个典型的应用场景中,参考图6,所述次级子矩阵40可以包括沿行方向阵列存储的多个瓦片30,多个所述次级子矩阵40沿行方向(即图6所示的内部维度方向)线性存储于所述内存,以形成所述格式化后的矩阵。

[0098] 具体地,类似于瓦片30的行方向和列方向,在本场景中,可以分别从内部维度和外部维度两个维度描述所述次级子矩阵40,其中,对于行主序的次级子矩阵,所述内部维度类

似于行方向，所述外部维度类似于列方向；对于列主序的次级子矩阵，则所述内部维度类似于列方向，所述外部维度类似于行方向。

[0099] 相应的，K_Q为每一次级子矩阵40在内部维度上包括的瓦片30的数量；K_T为每一瓦片30在内部维度上包括的单元a(图未示)的数量；M_Q为每一次级子矩阵40在外部维度上包括的瓦片30的数量；M_T为每一瓦片30在外部维度上包括的单元a的数量。

[0100] 在实际应用中，当所述矩阵运算指令为内积运算指令时，可以以所述内部维度为准进行相应运算，而所述外部维度独立于所述内部维度。

[0101] 在本场景中，参考图6中的虚线箭头方向，对于每一次级子矩阵40，所述$K_Q \times M_Q$个瓦片30沿先内部维度后外部维度的方向(类似于先行后列的方向)存储形成所述次级子矩阵40，而多个次级子矩阵40沿内部维度的方向线性存储于所述内存中。

[0102] 进一步地，对于每一瓦片30，其可以包括4×4个单元a，也即$M_T=K_T=4$。

[0103] 在实际应用中，本应用场景示出的次级子矩阵40可以作为上述图5所示矩阵相乘运算的输入，也即，所述格式化后的第一矩阵X和格式化后的第二矩阵Y均可以本场景示出的所述次级子矩阵40为基础进行矩阵相乘运算。

[0104] 需要指出的是，本实施例所述次级子矩阵40(包括所述瓦片30)仍为固定的底层物理数据组织。在根据所述矩阵运算指令执行相应的矩阵运算时，所述次级子矩阵40可以适用于其他不同的逻辑维度，所述逻辑维度可以是根据运算维度和工作分区确定的，以在需要的时候实现零内边距(padding)。

[0105] 在另一个典型的应用场景中，继续参考图6，针对单指令多数据流(Single Instruction Multiple Data，简称SIMD)结构，基于本实施例的方案同样可以实现基于次级子矩阵40的矩阵相乘优化。

[0106] 具体地，在基于所述次级子矩阵40进行矩阵相乘运算时，假设左侧运算项(即上述图5示出的格式化后的第一矩阵X)的外部维度为M_Q，右侧运算项(即上述图5示出的格式化后的第二矩阵Y)的外部维度为N_Q。不考虑一般性损失，假设$K_Q=1$，也即通过循环迭代求和的方式来跨越内部维度。

[0107] 在对所述次级子矩阵40进行矩阵相乘运算时，每次内存访问的运算量f可以基于如下公式表示：

[0108] $f=M_QN_Q/(M_Q+N_Q)$；

[0109] 而需要按切片进行矩阵相乘和加和运算的寄存器(如向量寄存器)的数量g可以基于如下公式表示：

[0110] $g=M_TM_QN_Q+M_Q+N_Q$；

[0111] 其中，假设一向量寄存器可以容纳M_T个元素，所述M_T个元素可以为一个瓦片30。

[0112] 当进行矩阵相乘运算时采用的向量寄存器的数量为32个时，为了最大化所述数值f，采用本实施例的理想的对称解决方案的每一次级子矩阵40的外部维度(包括M_Q和N_Q)均为2个瓦片30。而采用本实施例的理想的不对称解决方案的每一次级子矩阵40的外部维度分别为$M_Q=3$，$N_Q=2$；或$M_Q=2$，$N_Q=3$。

[0113] 较之上述图5所示的基本情形(即$M_Q=N_Q=1$)，上述两种解决方案均可以提供至少两倍的每次内存访问的运算量，其中，不对称解决方案的运算量甚至可以达到先前的2.4倍的密集程度。

[0114] 需要指出的是，以图6所示次级子矩阵40为基础的应用场景在前述图5所示以瓦片30为基础进行矩阵运算的基础上构建了一个更高级别的改进方案，图5所示的基础方案能够有效优化$M_T \times N_T$个元素的矩阵运算量，所述改进方案还能够有效优化包含$M_Q \times N_Q$个瓦片30的次级子矩阵40的矩阵运算量。

[0115] 接下来以基于ARM架构的次级子矩阵的矩阵相乘运算为例作具体阐述。

[0116] 在一个典型的应用场景中，结合图5和图7，假设M_Q为3个瓦片30，M_T和K_T均为4个单元a，K_Q为1个瓦片30，N_Q为2个瓦片30。也即，所述格式化后的第一矩阵X中的每一切片可以重新定义为包括多个由1×3个瓦片30组成的次级子矩阵40，所述格式化后的第二矩阵Y中的每一切片可以重新定义为包括多个由2×1个瓦片30组成的次级子矩阵40。其中，每一瓦片30包括4×4个单元a。

[0117] 在本场景中，需要3+2+4×3×2＝29个向量寄存器，在进行矩阵相乘运算时，本场景每次内存访问读取的运算项数量是每个次级子矩阵40仅包括一个瓦片30的场景中读取的运算项数量的2.4倍。

[0118] 进一步地，在本场景中，最终获得的每一切片的矩阵相乘的输出Z包括3×2个4×4的瓦片30。

[0119] 在另一个典型的应用场景中，参考图8，本实施例所述方案还可以应用于除常规CPU之外的其他运算环境。例如，可以假设采用本实施例的方案有组织地进行矩阵相乘运算。

[0120] 具体地，在本场景中，假设M_Q为M_Q个瓦片30，M_T和K_T均为W个单元a，K_Q为1个瓦片30，N_Q为N_Q个瓦片30。也即，所述格式化后的第一矩阵X中的每一切片可以重新定义为包括多个由$1 \times M_Q$个瓦片30组成的次级子矩阵40，所述格式化后的第二矩阵Y中的每一切片可以重新定义为包括多个由$N_Q \times 1$个瓦片30组成的次级子矩阵40。其中，每一瓦片40包括W×W个单元a。

[0121] 换言之，所述格式化后的第一矩阵X和格式化后的第二矩阵Y共包括M×K×N个单元a。

[0122] 进一步地，在本场景中，最终获得的每一切片的矩阵相乘的输出Z包括$M_Q \times N_Q$个W×W的瓦片30。

[0123] 进一步地，结合图8和图9，在没有额外(或特别)约束的情况下，对称的解决方案(即，令$M_Q＝N_Q$)可能获得最佳的计算机到内存的传输带宽比例。

[0124] 具体地，在对图8示出的所述格式化后的第一矩阵X的第一行切片和所述格式化后的第二矩阵Y的第一列切片执行矩阵相乘运算时，可以执行操作s1，以分别从内存50读取所述第一行切片和第一列切片，读取带宽可以基于如下公式计算获得：

[0125] $\text{Read BW}＝2N_QW^2/N_Q^2W＝2W/N_Q$；

[0126] 其中，Read BW为所述读取带宽；所述内存50的容量为$2N_QW^2$。

[0127] 进一步地，可以执行操作s2，以将读取的数据进行输入缓存。

[0128] 其中，在将所述读取的数据提交至待执行运算操作的设备之前，这些数据的全部或部分可以已经完成本实施例所述的格式化操作。也即，读取至所述内存50的数据已经是格式化后的矩阵。

[0129] 或者，也可以将原始数据直接读取至所述设备，由所述设备执行操作s3，以进行额

外的格式化操作，然后再进行后续的矩阵运算操作。

[0130] 进一步地，可以执行操作s4，以对所述格式化后的第一矩阵X的第一行切片和所述格式化后的第二矩阵Y的第一列切片执行快速矩计算(fast compute)，获得计算结果60，所述计算结果60也是以W×W瓦片30为单位的。

[0131] 进一步地，执行操作s5，以将获取的计算结果60输出至缓冲记忆装置。例如，可以将获取的计算结果60沿行方向或列方向依次存储至内存70中，所述内存70的容量为$N_Q^2W^2$个单元a。

[0132] 进一步地，执行操作s6，以读取并对所述计算结果60进行快速后处理(fast post-compute)操作，以获取处理后的计算结果60'。优选地，所述快速后处理可以用于将所述计算结果60格式化为符合输出内存的存储格式的计算结果60'。

[0133] 进一步地，执行操作s7，以沿行方向或列方向将所述处理后的计算结果60'写至预设存储区域。其中，写入带宽可以基于如下公式计算获得：

[0134] $Write\_BW=N_Q^2W^2(s)/N_Q^2W(K/W)=W^2(S)/K$；

[0135] 其中，Write_BW为所述写入带宽，S为所述单元a包括的元素数量。

[0136] 进一步地，本实施例的方案还可以适用于基于张量切片(tensor slice)进行多进程构建和执行的应用场景，基于本实施例方案提供的灵活的数据结构(即格式化后的矩阵)能够有效提高本应用场景的数据处理效率。

[0137] 具体地，结合图10至图13，在本应用场景中，从张量切片的角度着手，对调度本实施例的方案获取的格式化后的矩阵的数据结构和后续矩阵运算的操作过程作具体阐述，本应用场景可以适于任何常规或特殊架构。

[0138] 更为具体地，在本应用场景中，所述格式化后的第一矩阵X和格式化后的第二矩阵Y可以概念化为张量切片结构，其中，张量可以理解为矩阵的多维抽象形式。

[0139] 进一步地，可以通过张量切片描述符来携带用于指示这些张量切片的组织信息(例如，所述瓦片和次级子矩阵的维度等信息)。

[0140] 进一步地，所述张量切片描述符可以与张量切片一一对应，不同的张量切片描述符可以相互关联以表示更普遍的张量结构。例如，一个简单的张量结构可以是一个矩阵(即一个二维张量)，其被分割成可管理的行或列的组合。

[0141] 进一步地，所述张量切片描述符还可以包含格式化相关的其他附加信息。不同张量切片的张量切片描述符可以包含不同的其他附加信息。

[0142] 进一步地，参考图10，所述张量切片描述符可以包括张量切片描述符1、张量切片描述符2、…、张量切片描述符D，其中，D＞1。每一张量切片描述符对应一张量切片。

[0143] 进一步地，所述张量切片描述符可以用于指示输入数据的数据结构和相关操作信息这两大类内容，其中，所述输入数据的数据结构可以为一组有规律的结构信息，所述相关操作信息可以用于指示运算逻辑。

[0144] 作为一个非限制性实施例，所述张量切片描述符可以包括的信息有：底层元素的维度，所述底层元素可以包括的底层数据等；瓦片30和次级子矩阵40的定义，如外部维度和内部维度的定义；切片数据的头部的相关信息；下一个张量切片描述符(或描述符的尾部)的相关信息；数据的逻辑行/列标识符；元数据70的相关信息；有效和前进指示符；压缩配置。

[0145] 例如，基于所述张量切片描述符1包括的信息，可以确定对应的张量切片包括阵列排布的多个次级子矩阵40，其中，每一次级子矩阵40包括阵列排布的多个瓦片30，其中，每一瓦片30包括$W^1\times W^1$个单元a(图未示)。由此，$M^1_T=K^1_T=W^1$，$K^1_Q=1$。

[0146] 进一步地，底层数据元素80可以以最佳顺序局部性存储，从而确保压缩配置可以涵盖数据和可选的元数据70。

[0147] 类似的，基于所述张量切片描述符2包括的信息，可以确定对应的张量切片包括阵列排布的多个次级子矩阵40，其中，每一次级子矩阵40包括阵列排布的多个瓦片30，其中，每一瓦片30包括$W^2\times W^2$个单元a(图未示)。由此，$M^2_T=K^2_T=W^2$，$K^2_Q=1$。

[0148] 类似的，基于所述张量切片描述符D包括的信息，可以确定对应的张量切片包括阵列排布的多个次级子矩阵40，其中，每一次级子矩阵40包括阵列排布的多个瓦片30，其中，每一瓦片30包括$W^D\times W^D$个单元a(图未示)。由此，$M^D_T=K^D_T=W^D$，$K^D_Q=1$。

[0149] 进一步地，所述张量切片描述符D还可以包括完成确认信息。

[0150] 本领域技术人员理解，基于本应用场景的方案，所述张量切片可以在整体上指示数据值的底层数据结构(包括矩阵或张量)。

[0151] 进一步，优化的顺序存储利于应用压缩配置，同时保持最佳的数据局部化。

[0152] 进一步，对于每一张量切片(可简称为切片)，通过描述符来描述该张量切片，并且，所述张量切片包括维度化的数据，如针对该切片的次级子矩阵、瓦片、元素维度等，其中，不同的切片可以具有不同的维度。

[0153] 进一步，所述张量切片描述符可以分散的存储在内存的不同地址上，所述不同地址相互独立，如基于连续的描述符表的形式存储。或者，也可以定位在距离其对应的切片预设长度的位置，如可以存储在切片头。

[0154] 进一步，对于每一连续的张量切片，可以通过所述张量切片的维度描述和/或(可选的)指针来计算其位置。

[0155] 进一步，所述张量切片可以与一唯一的行列标识符相关联，当其存在于更大、更具代表性的结构时，所述行列标识符可以和张量切片的逻辑位置相映射。本关联关系的应用可能影响到计算和计算结果的存储。

[0156] 进一步，所述张量切片可以由多个平行的工作线程构建，所述多个平行的工作线程各自构建的张量切片可以以任意顺序添加到所述张量切片的描述符表，直至全部构建完毕。

[0157] 进一步，通过利用已完成和待决的指示，可以进一步开发流水线式的平行运算方式。

[0158] 进一步，通过遍历所述描述符表(也可称为描述符列表)直至所述描述符表的底部，或者基于已完成-待决指示，可以获取并处理待处理的切片。

[0159] 进一步，内存地址可以是实体的，也可以是虚拟的，与系统映射相一致。

[0160] 进一步，所述描述符可以引用分离的描述符表，所述分离的描述符表是分层设计的，以指示更高维度的结构。

[0161] 进一步，底层的瓦片的维度可以不是正方的，例如，所述瓦片根据最优的瓦片尺寸可以是$W\times H$的维度，且$W\neq H$。

[0162] 在另一个典型的应用场景中，参考图11，前述描述符的概念可以进一步应用于构

建运算描述符,所述运算描述符用于定义在一组给定的张量中的运算逻辑。

[0163] 具体地,运算描述符1可以包括:运算符;操作数地址(或结构相关的信息);输出内存的地址和格式;消耗线程管理;运算完成条件和状态。

[0164] 进一步地,所述运算描述符可以关联一个或多个张量切片描述符。

[0165] 例如,在图11所示的应用场景中,所述运算描述符1可以关联张量切片描述符1、张量切片描述符2和张量切片描述符3。其中,根据所述张量切片描述符1,可以确定对应的切片,该切片包括多个线性排布的1瓦片×2瓦片的次级子矩阵40,每一瓦片包括$W^1\times W^1$个单元a;根据所述张量切片描述符2,可以确定对应的切片,该切片包括多个线性排布的1瓦片×1瓦片的次级子矩阵40,每一瓦片30包括$W^2\times W^2$个单元a,也即,在该切片中次级子矩阵40即为瓦片30;根据所述张量切片描述符3,可以确定对应的切片,该切片包括多个线性排布的1瓦片×2瓦片的次级子矩阵40,每一瓦片30包括$W^3\times W^3$个单元a。

[0166] 对于所述张量切片描述符3对应的切片,该切片可以定义内边距(padding)。优选地,所述内边距的数值可以为零,以获得更优的底层瓦片尺寸。

[0167] 进一步地,也可以通过所述描述符的层次来描述运算及其运算符。

[0168] 进一步地,在递归计算的场景中,对于原始的运算符,基于本场景的运算结果可以是可消耗的,也即,前一次的运算结果可以是后一次运算的输入数据。

[0169] 进一步,所述运算操作符可以是由多个工作线程生成的。或者,所述运算操作符可以被多个工作线程消耗或由所述多个工作线程生成。

[0170] 进一步,所述线程可以为SW(Solid Works)线程,也可以为其他实施(如自定义HW)中的独立进程。

[0171] 进一步地,结合图10至图12,可以通过执行根线程(root thread)获取所述运算描述符1,可以通过执行生产者线程(producer thread)1获取所述张量切片描述符1,可以通过执行生产者线程2获取所述张量切片描述符2,可以通过执行生产者线程3获取所述张量切片描述符3。

[0172] 进一步地,可以通过执行使用者线程(consumer thread)1和使用者线程2获取所述运算描述符1指示的运算逻辑。

[0173] 进一步地,通过执行所述使用者线程1,可以基于所述运算逻辑对所述张量切片描述符1关联的格式化后的矩阵的每一切片和所述张量切片描述符3关联的格式化后的矩阵的每一切片进行矩阵运算,获取对应的计算结果。

[0174] 类似的,通过执行所述使用者线程2,可以基于所述运算逻辑对所述张量切片描述符2关联的格式化后的矩阵的每一切片和所述张量切片描述符3关联的格式化后的矩阵的每一切片进行矩阵运算,获取对应的计算结果。

[0175] 进一步地,所述使用者线程可以被静态分配,或基于可用资源中的冲裁协议进行动态调度。

[0176] 接下来,结合图13对阶层式(hierarchical)运算结构作具体阐述。

[0177] 具体地,通过执行根运算(root operation),可以获取运算描述符1。例如,所述运算描述符1可以为矩阵加法运算。并且,基于所述运算描述符1可以确定对应的运算对象。

[0178] 进一步地,继续获取运算描述符2及其对应的运算对象。例如,所述运算描述符2可以为矩阵相乘运算。

[0179] 由此，在执行运算时，先对张量切片描述符1和张量切片描述符2各自对应的切片(可以参考上述图11所示的张量切片描述符1和张量切片描述符2各自对应的切片)进行矩阵相乘运算；将运算结果与张量切片描述符3对应的切片(可以参考上述图11所示的张量切片描述符3对应的切片)再进行矩阵相乘运算，以获取第一计算结果。

[0180] 类似的，获取运算描述符3及其对应的运算对象。例如，所述运算描述F也可以为矩阵相乘运算。

[0181] 由此，在执行运算时，可以对张量切片描述符4对应的切片和张量切片描述符5对应的切片进行矩阵相乘运算，以获取第二计算结果。

[0182] 然后，根据所述运算描述符1指示的运算逻辑对所述第一计算结果和第二计算结果进行矩阵相加运算，以获取最终的计算结果。

[0183] 其中，张量切片描述符1对应的切片、张量切片描述符2对应的切片、张量切片描述符3对应的切片、张量切片描述符4对应的切片以及张量切片描述符5对应的切片包括的数据均预先采用本实施例的方案进行了格式化处理。

[0184] 进一步地，在本场景中，所述运算可以根据运算顺序和运算对象的关联性被阶层化的嵌套和描述。

[0185] 进一步地，在完成运算后或执行过程中，获取的计算结果(如所述第一计算结果、第二计算结果或最终的计算结果)可以替换原始的运算对象进行存储。

[0186] 作为一个变化例，参考图14，可以将所述运算描述符3对应的计算结果(即前述第二计算结果)作为下一次运算的运算对象。也即，与上述图13所示场景的区别在于，本变化例中，张量切片描述符6对应的切片是上一次基于运算描述符3执行运算获取的运算结果，由此实现迭代的运算流程，最终获得的张量切片描述符6和张量切片描述符7各自对应的切片即为所述运算描述符3对应的运算结果。

[0187] 在本变化例中，用于暂存运算结果的输出缓冲区可以预先分配给执行结构，并由执行单元填充。

[0188] 在又一个典型的应用场景中，参考图15和图16，本实施例的方案还可以适用于递归运算场景，用于多次调度相同的操作，消耗自己的输出直至满足预设限制条件。其中，图15为初始状态下递归运算的结构框图，图16为递归运算期间，执行下一次递归运算之前的结构框图。

[0189] 具体地，在本场景中，假设张量切片描述符1对应的切片为矩阵X包括的切片，假设张量切片描述符2对应的切片为矩阵Y_{i+1}包括的切片，假设张量切片描述符3对应的切片为矩阵Y_i包括的切片。

[0190] 假设所述运算描述符1指示的运算逻辑为$Y_{i+1}=Y_i\times X$，直至$i=n$，$n>1$。

[0191] 在本场景中，运算对象可以同时作为输入和输出。

[0192] 在图15所示的初始状态下，尚不存在矩阵Y_{i+1}的输出。

[0193] 在第一次执行运算时，获取所述矩阵Y_{i+1}的运算结果并存储至张量切片描述符3对应的缓存。然后，循环引用所述张量切片描述符3对应的缓存中存储的数据以生成下一次迭代。

[0194] 由此，对比图15和图16，所述张量切片描述符1对应的切片恒为矩阵X包括的切片，所述张量切片描述符2对应的缓存可以被循环再用于生成$X\times X$的运算结果(也即上一次递

归运算的运算结果)。

[0195] 或者,也可以回收利用(reclaim)一个新分配的张量切片描述符及其对应的底层切片数据缓存来实现本场景的迭代操作。

[0196] 进一步地,在本场景中,所述递归运算可以持续执行直至i＝n,其中,n可以为预设数值。

[0197] 或者,所述递归运算的完成条件也可以是根据运算值和运算结果的函数确定的。

[0198] 图17是本发明实施例的一种矩阵的运算装置的结构示意图。本领域技术人员理解,本实施例所述矩阵的运算装置90用于实施上述图1至图16所示实施例中所述的方法技术方案。

[0199] 具体地,在本实施例中,所述矩阵的运算装置90可以包括:获取模块91,用于获取至少一个矩阵;格式化模块92,用于对所述至少一个矩阵中的每一矩阵进行格式化,以得到格式化后的各个矩阵,所述格式化包括:将所述矩阵划分为多个子矩阵,并对每一子矩阵进行转置,每一转置后的子矩阵记为一个瓦片;运算模块93,响应于矩阵运算指令,对所述格式化后的各个矩阵进行运算,以得到运算结果。

[0200] 进一步地,所述格式化还可以包括:沿行方向或列方向依次获取各个瓦片,并存储得到多个次级子矩阵,每一次级子矩阵包括阵列排布的至少一个瓦片,其中,所述格式化后的矩阵包括沿行方向或列方向的多个次级子矩阵。

[0201] 进一步地,所述获取模块91可以包括:获取子模块911,用于获取第一矩阵和第二矩阵;所述运算模块93可以包括:矩阵相乘运算子模块931,响应于矩阵相乘运算指令,对格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算,以得到所述运算结果。

[0202] 进一步地,所述矩阵相乘运算是以所述瓦片为单位执行的。

[0203] 进一步地,所述转置操作是基于寄存器实现的。

[0204] 进一步地,所述子矩阵包括的元素为4×4个。

[0205] 关于所述矩阵的运算装置90的工作原理、工作方式的更多内容,可以参照上述图1至图16中的相关描述,这里不再赘述。

[0206] 进一步地,本发明实施例还公开一种存储介质,其上存储有计算机指令,所述计算机指令运行时执行上述图1至图16所示实施例中所述的方法技术方案。优选地,所述存储介质可以包括诸如非挥发性(non-volatile)存储器或者非瞬态(non-transitory)存储器等计算机可读存储介质。所述存储介质可以包括ROM、RAM、磁盘或光盘等。

[0207] 进一步地,本发明实施例还公开一种终端,包括存储器和处理器,所述存储器上存储有能够在所述处理器上运行的计算机指令,所述处理器运行所述计算机指令时执行上述图1至图16所示实施例中所述的方法技术方案。优选地,所述终端可以是计算设备。

[0208] 虽然本发明披露如上,但本发明并非限定于此。任何本领域技术人员,在不脱离本发明的精神和范围内,均可作各种更动与修改,因此本发明的保护范围应当以权利要求所限定的范围为准。

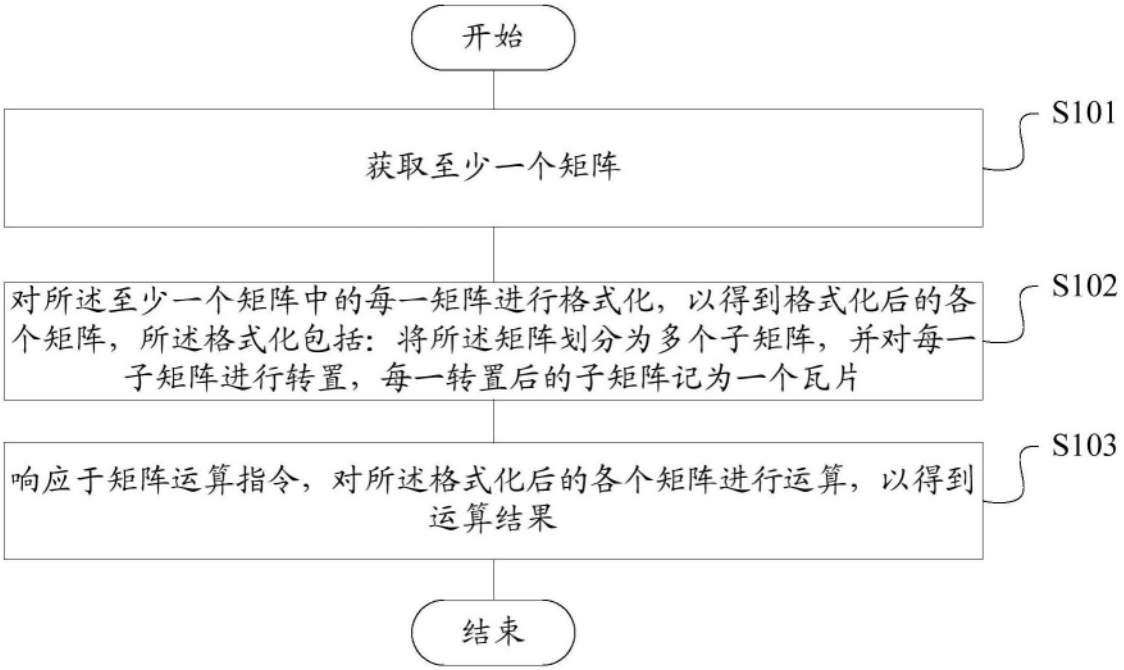

开始
获取至少一个矩阵
S101
对所述至少一个矩阵中的每一矩阵进行格式化，以得到格式化后的各个矩阵，所述格式化包括：将所述矩阵划分为多个子矩阵，并对每一子矩阵进行转置，每一转置后的子矩阵记为一个瓦片
S102
响应于矩阵运算指令，对所述格式化后的各个矩阵进行运算，以得到运算结果
S103
结束

图1

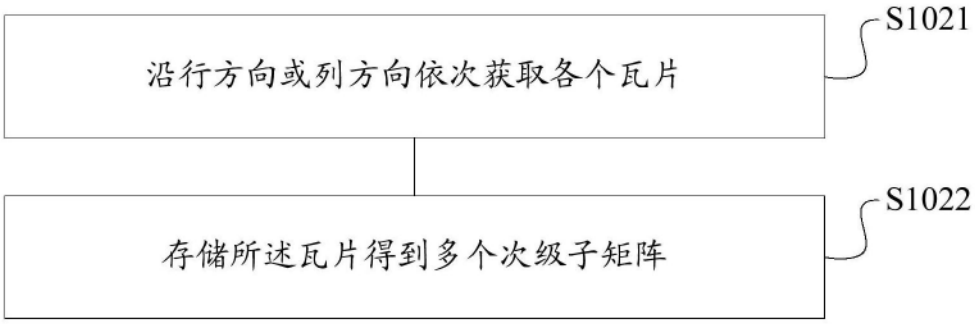

沿行方向或列方向依次获取各个瓦片
S1021
存储所述瓦片得到多个次级子矩阵
S1022

图2

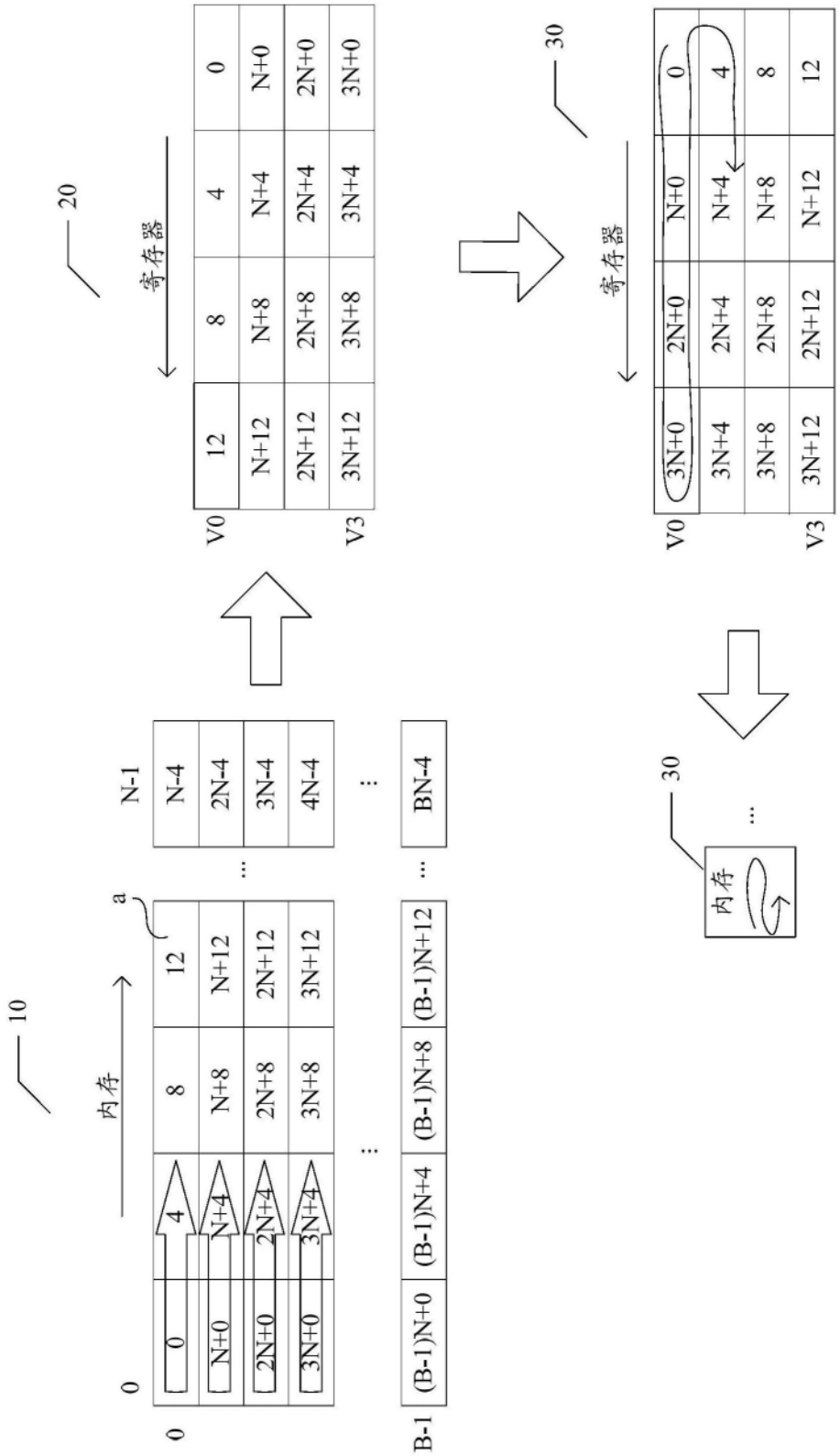

10
内存
0
4
8
12
a
0
N+0
N+4
N+8
N+12
2N+0
2N+4
2N+8
2N+12
3N+0
3N+4
3N+8
3N+12
(B-1)N+0
(B-1)N+4
(B-1)N+8
(B-1)N+12
B-1
N-1
N-4
2N-4
3N-4
4N-4
BN-4
20
寄存器
V0
V3
12
8
4
0
N+12
N+8
N+4
N+0
2N+12
2N+8
2N+4
2N+0
3N+12
3N+8
3N+4
3N+0
30
寄存器
V0
V3
3N+0
2N+0
N+0
0
3N+4
2N+4
N+4
4
3N+8
2N+8
N+8
8
3N+12
2N+12
N+12
12
30
内存

图3

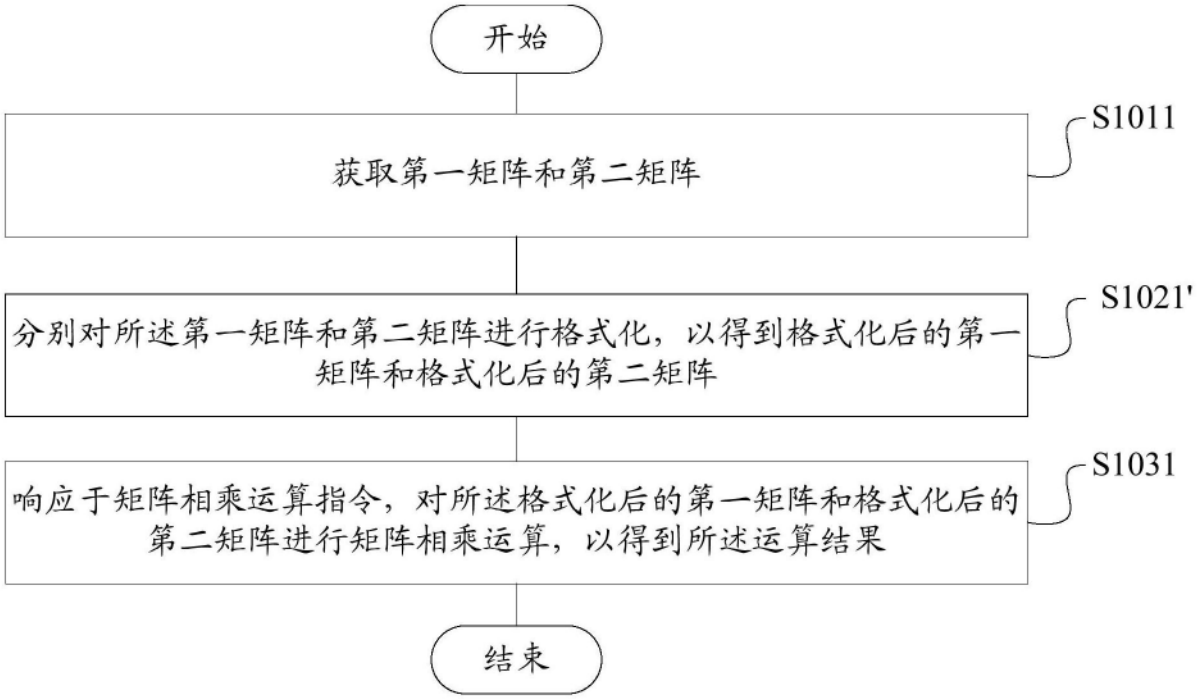

开始
获取第一矩阵和第二矩阵
S1011
分别对所述第一矩阵和第二矩阵进行格式化，以得到格式化后的第一矩阵和格式化后的第二矩阵
S1021'
响应于矩阵相乘运算指令，对所述格式化后的第一矩阵和格式化后的第二矩阵进行矩阵相乘运算，以得到所述运算结果
S1031
结束

图4

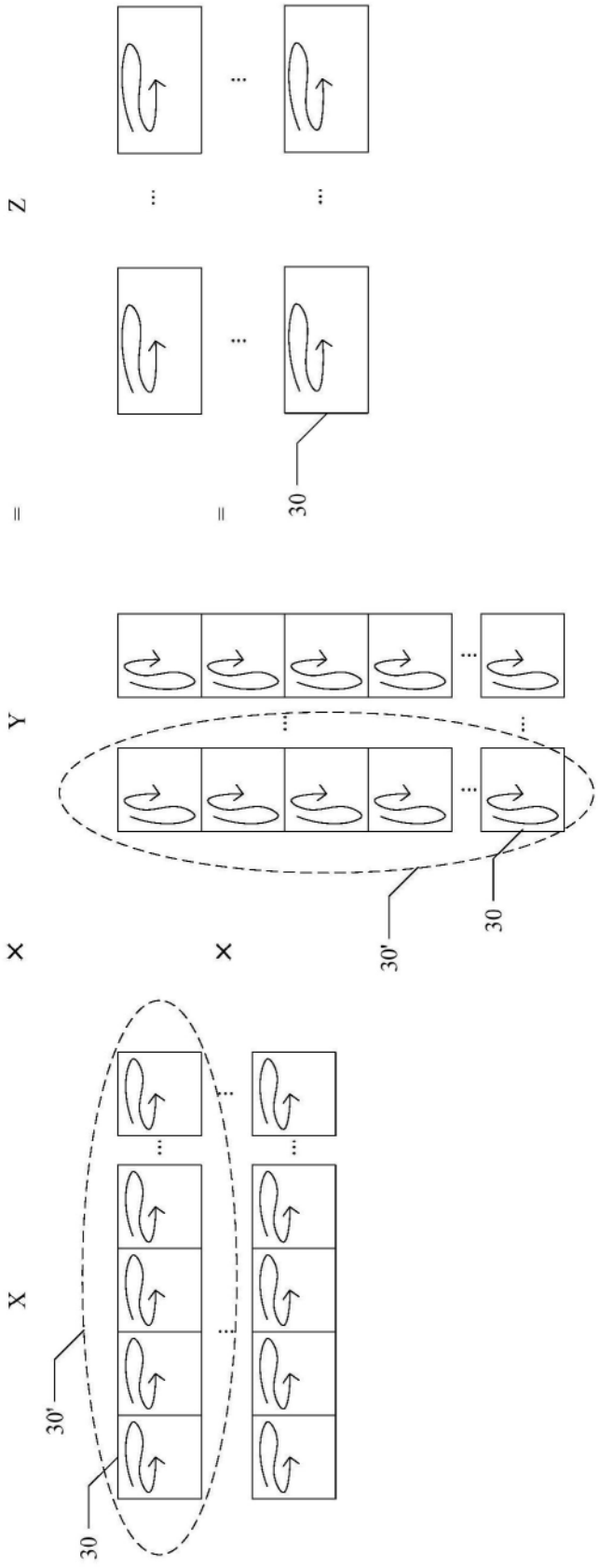


图5

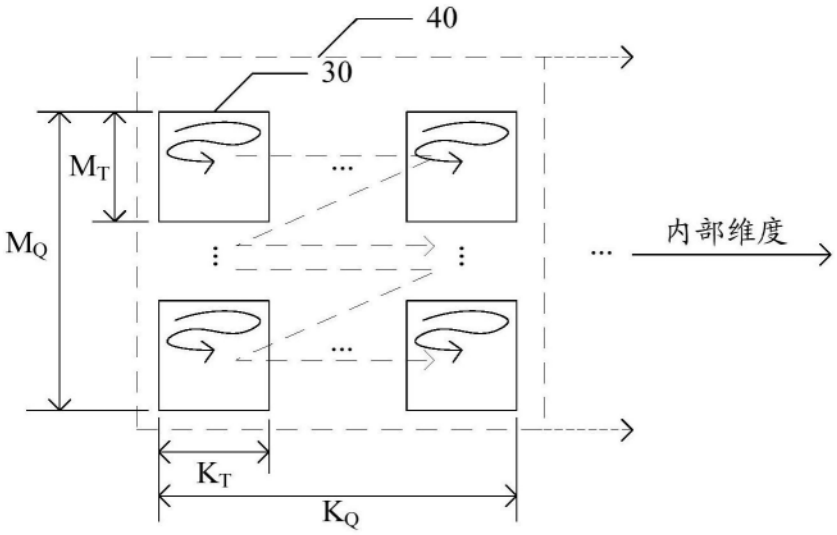


图6

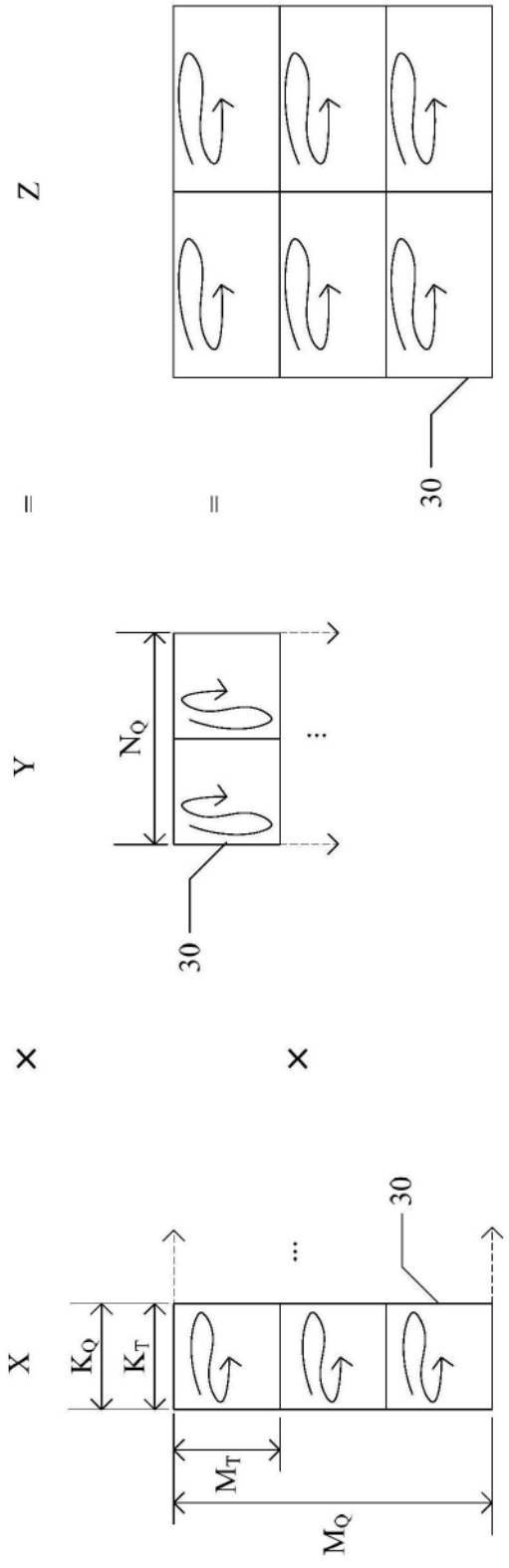


图7

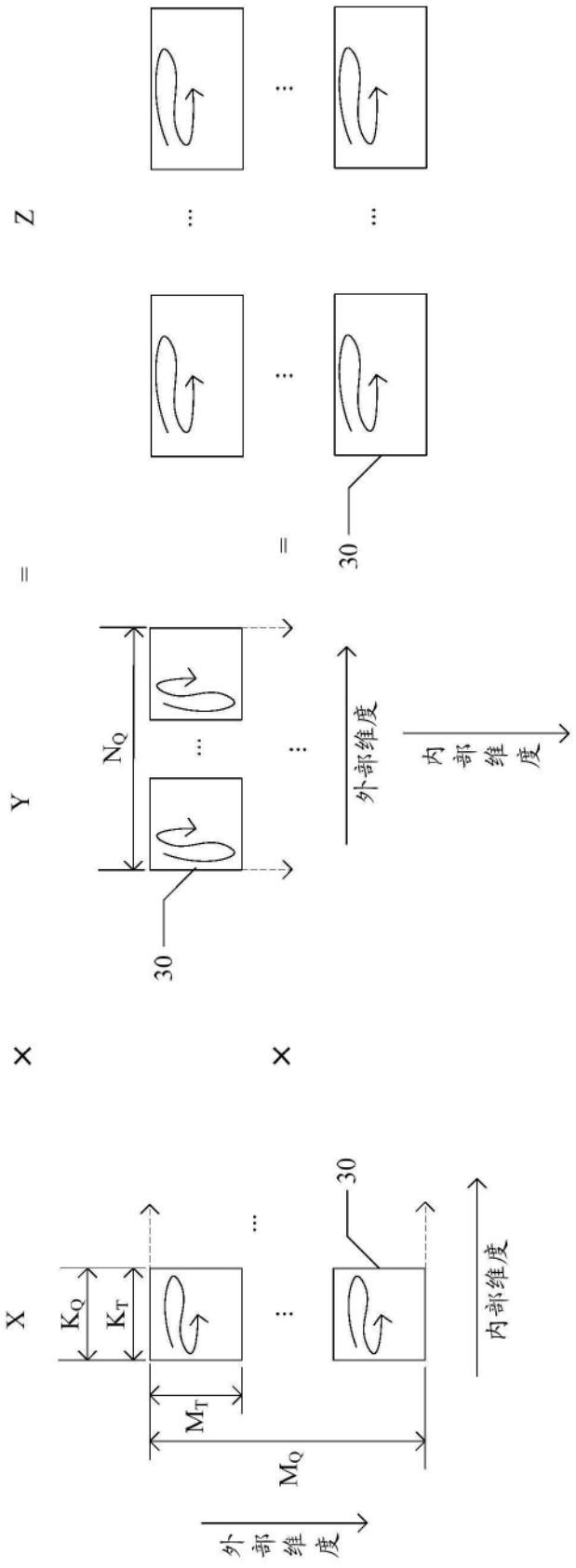


图8

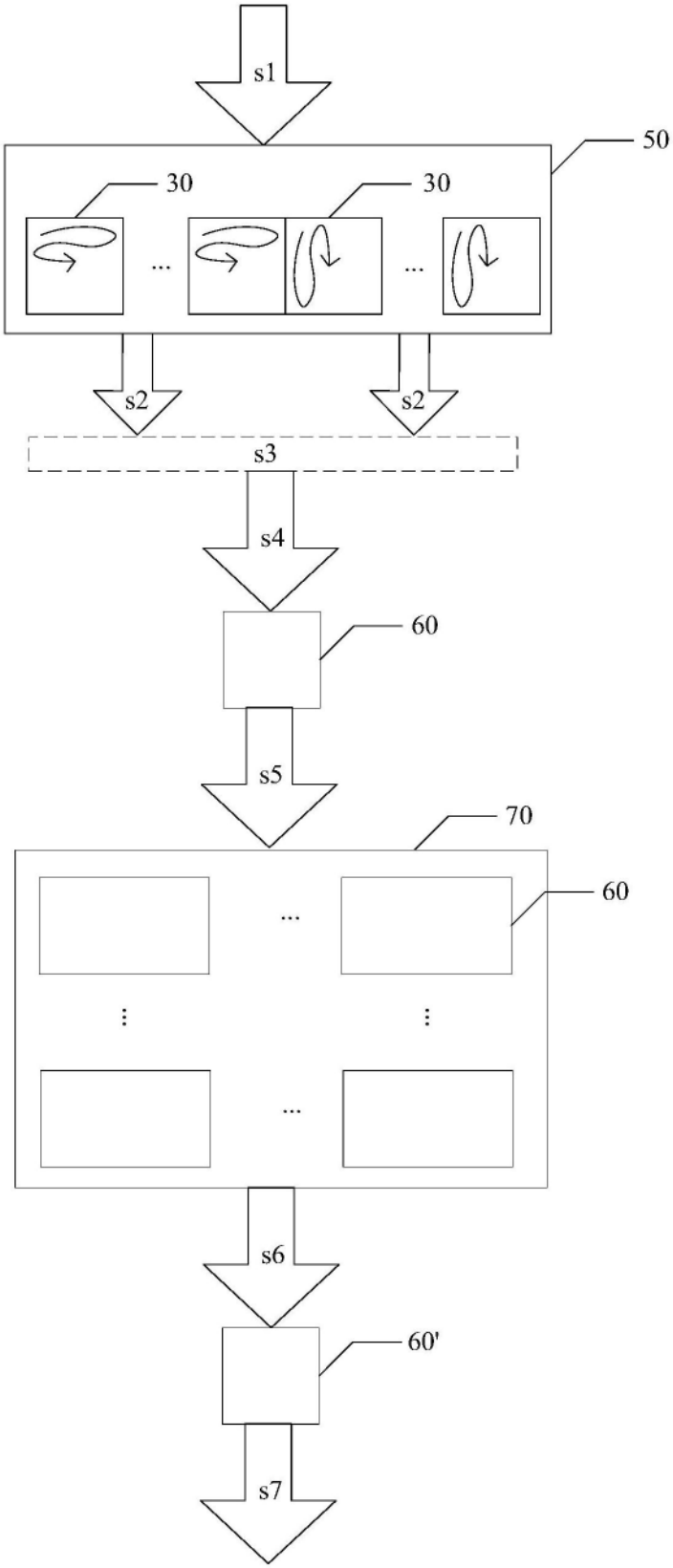


图9

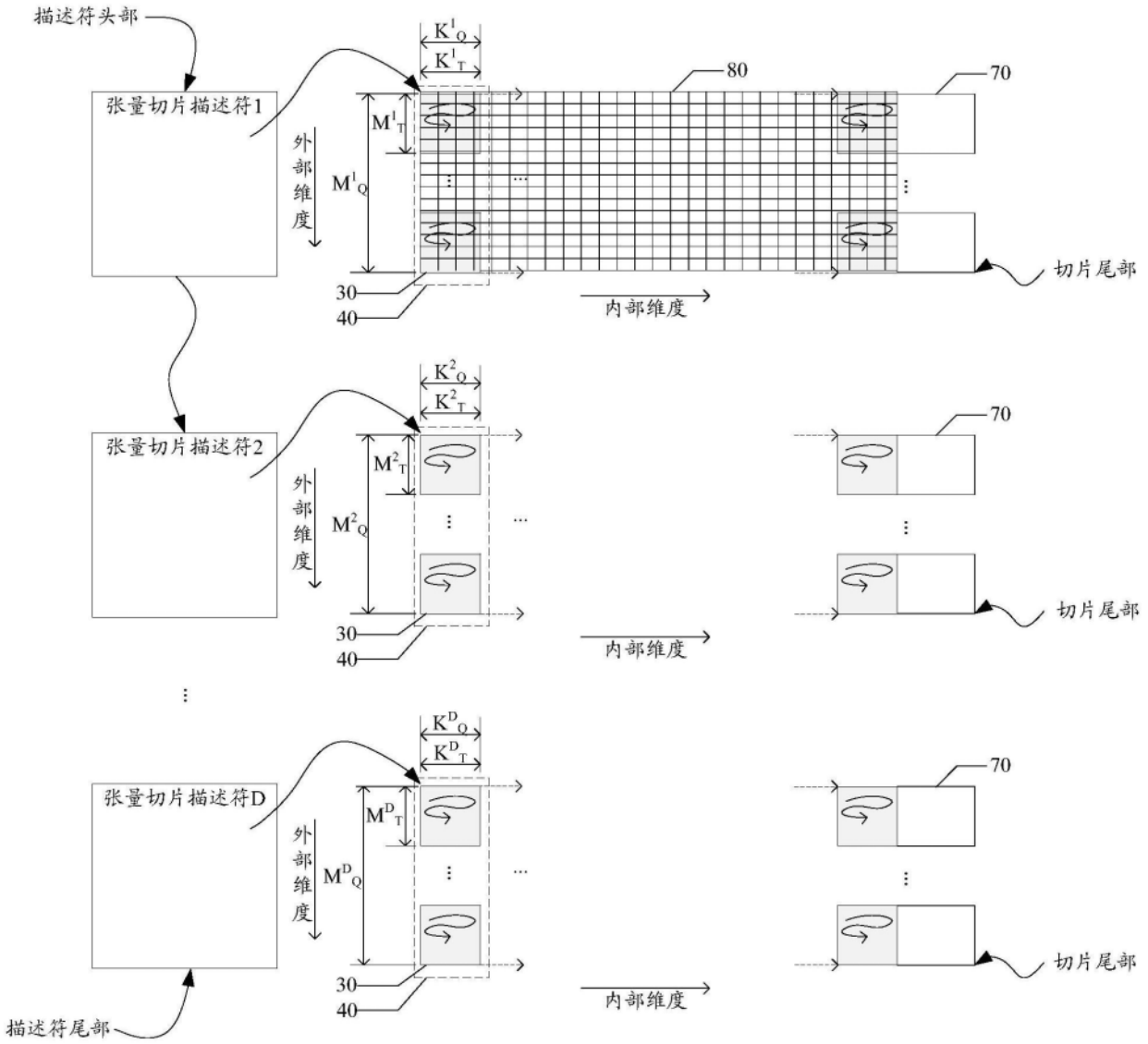

描述符头部
张量切片描述符1
张量切片描述符2
张量切片描述符D
描述符尾部
外部维度
内部维度
80
70
30
40
切片尾部

图10

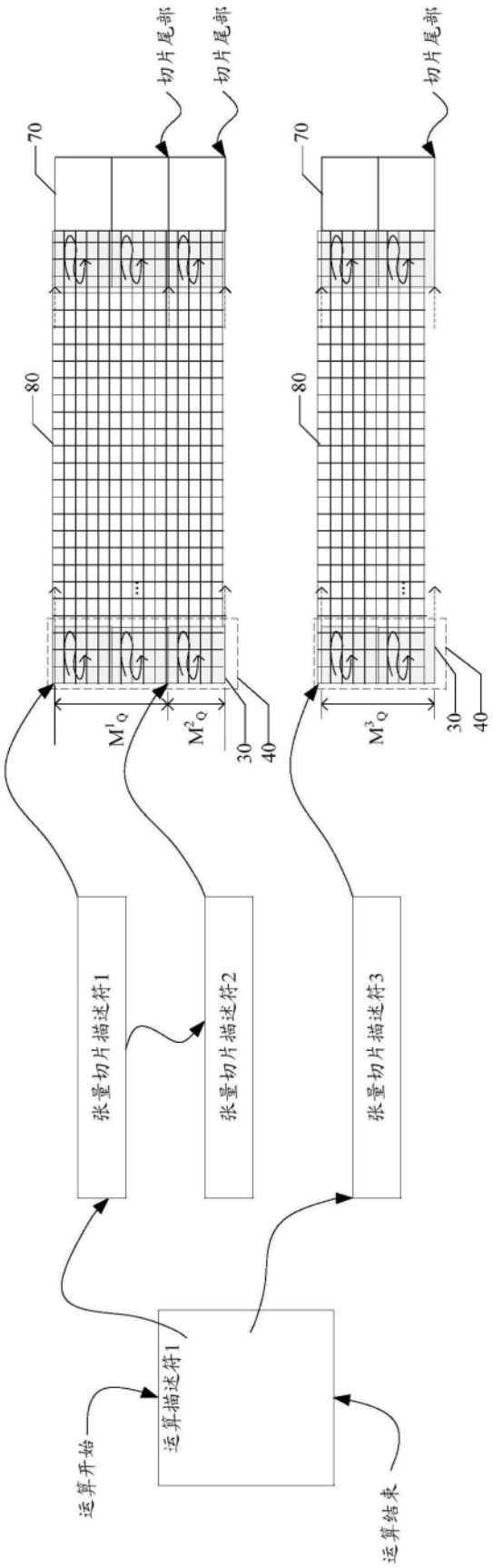

切片尾部
切片尾部
切片尾部
70
70
80
80
M^1_Q
M^2_Q
M^3_Q
30
40
30
40
张量切片描述符1
张量切片描述符2
张量切片描述符3
运算描述符1
运算开始
运算结束

图11

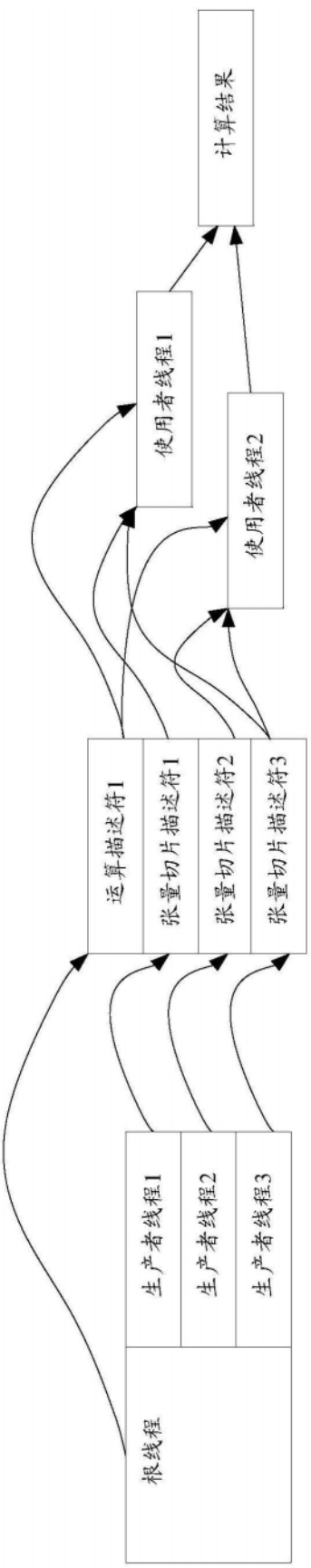

计算结果
使用者线程1
使用者线程2
运算描述符1
张量切片描述符1
张量切片描述符2
张量切片描述符3
生产者线程1
生产者线程2
生产者线程3
根线程

图12

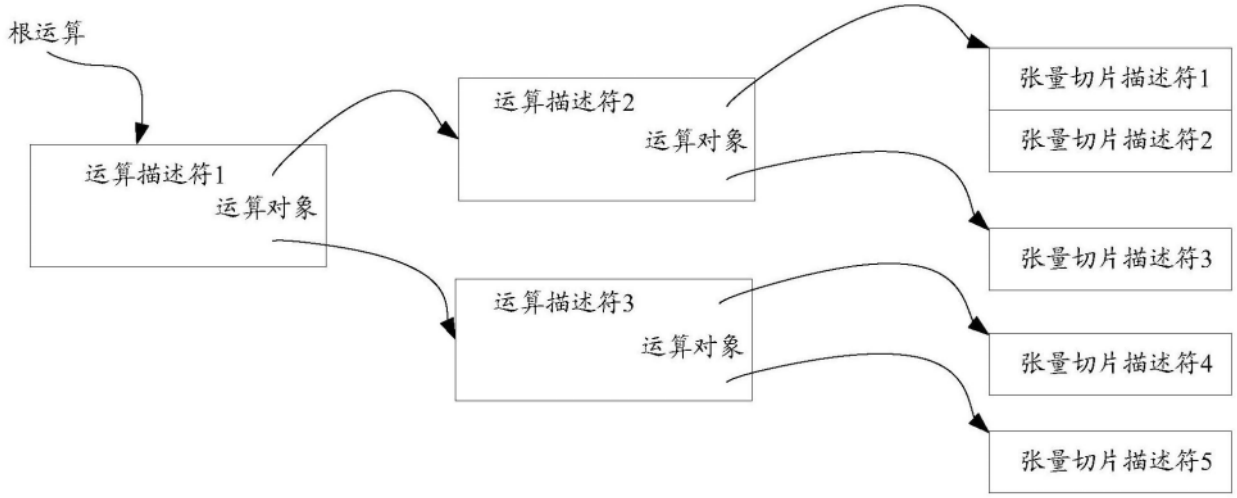

根运算
运算描述符1
运算对象
运算描述符2
运算对象
运算描述符3
运算对象
张量切片描述符1
张量切片描述符2
张量切片描述符3
张量切片描述符4
张量切片描述符5

图13

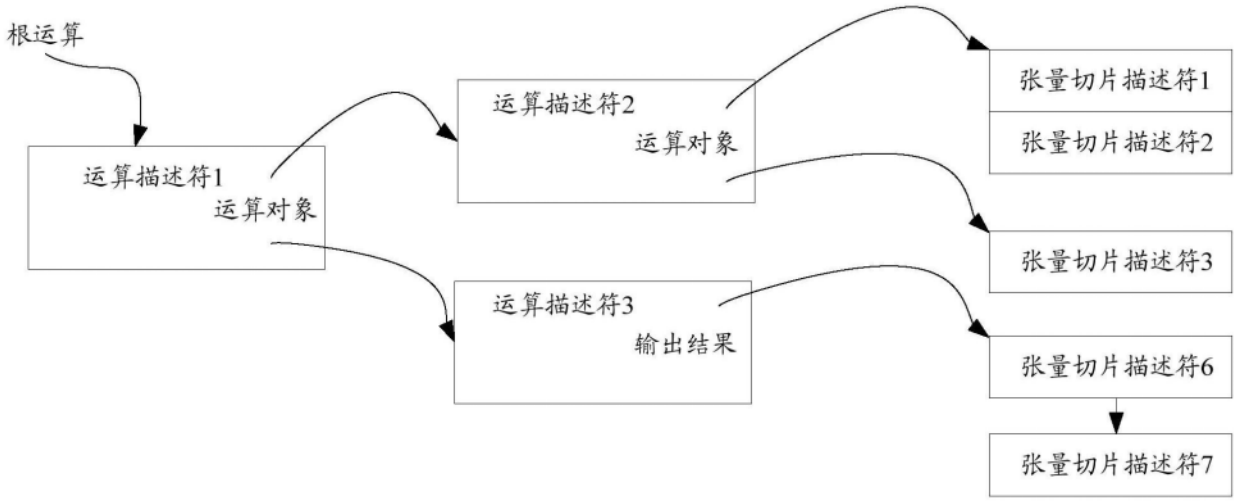

根运算
运算描述符1
运算对象
运算描述符2
运算对象
运算描述符3
输出结果
张量切片描述符1
张量切片描述符2
张量切片描述符3
张量切片描述符6
张量切片描述符7

图14

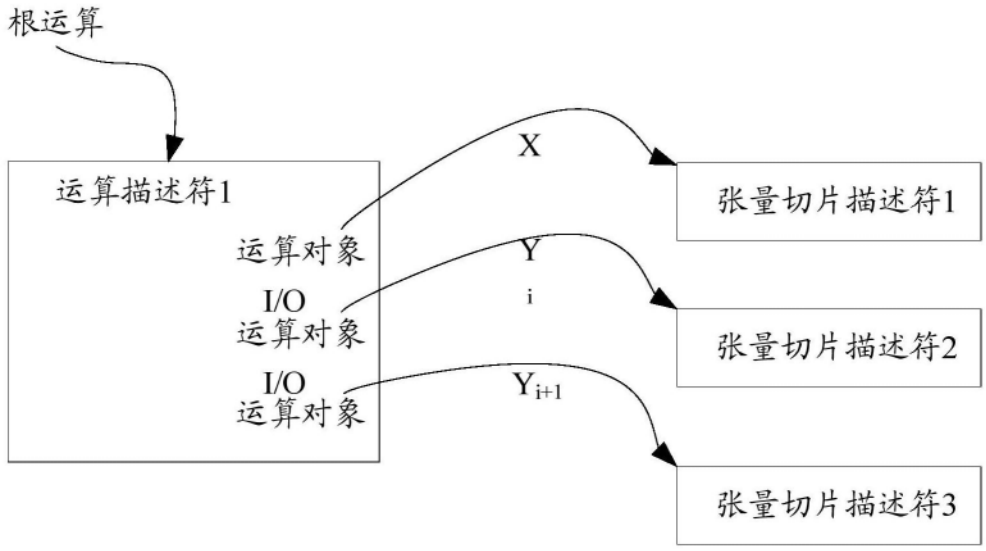

根运算
运算描述符1
运算对象
I/O
运算对象
I/O
运算对象
X
Y
i
Y
i+1
张量切片描述符1
张量切片描述符2
张量切片描述符3

图15

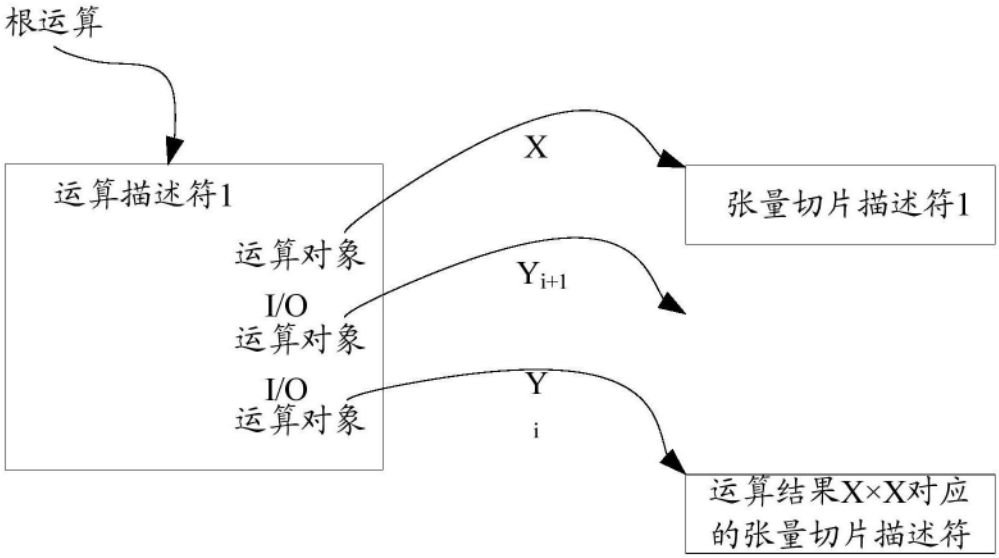

根运算
运算描述符1
运算对象
I/O
运算对象
I/O
运算对象
X
Y
i+1
Y
i
张量切片描述符1
运算结果X×X对应
的张量切片描述符

图16

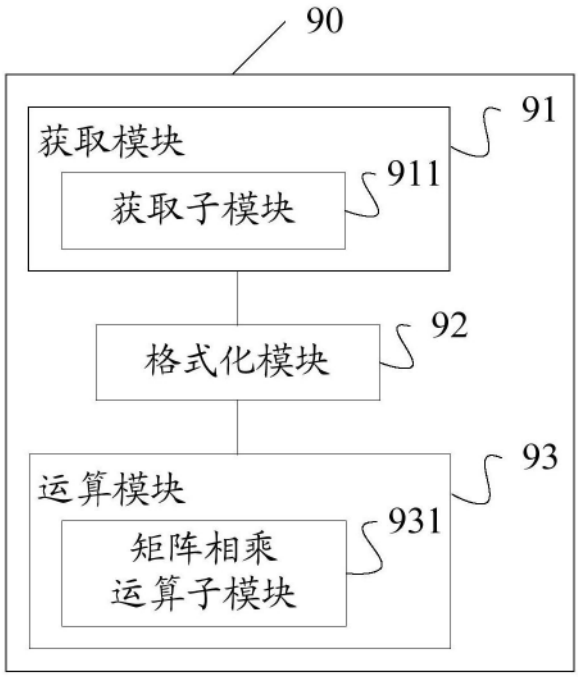


图17