

(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
2023 年 3 月 23 日 (23.03.2023)



(10) 国际公布号
WO 2023/041002 A1

- (51) 国际专利分类号:
G06N 3/063 (2006.01)
- (21) 国际申请号: PCT/CN2022/119198
- (22) 国际申请日: 2022 年 9 月 16 日 (16.09.2022)
- (25) 申请语言: 中文
- (26) 公布语言: 中文
- (30) 优先权:
202111091356.7 2021年9月17日 (17.09.2021) 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)。 中国科学技术大学 (UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA) [CN/CN]; 中国安徽省合肥市包河区金寨路96号, Anhui 230026 (CN)。
- (72) 发明人: 田腾(TIAN, Teng); 中国安徽省合肥市包河区金寨路96号, Anhui 230026 (CN)。 鲁芳敏

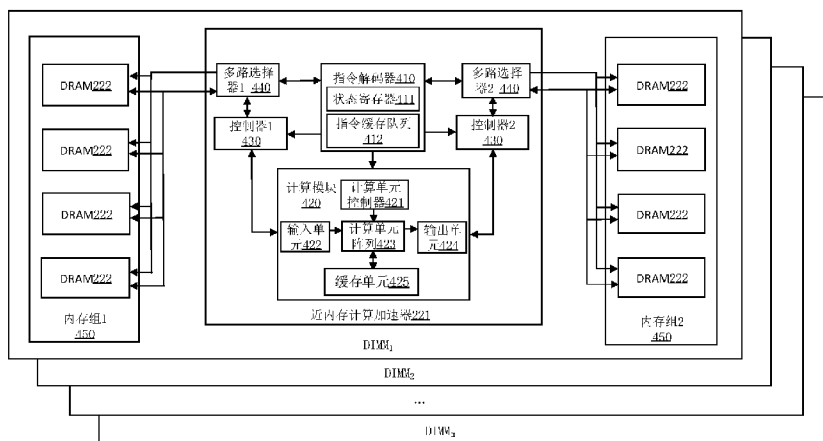
(LU, Fangmin); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 赵乐天(ZHAO, Letian); 中国安徽省合肥市包河区金寨路96号, Anhui 230026 (CN)。 王笑天(WANG, Xiaotian); 中国安徽省合肥市包河区金寨路96号, Anhui 230026 (CN)。 吴巍(WU, Wei); 中国安徽省合肥市包河区金寨路96号, Anhui 230026 (CN)。 金西(JIN, Xi); 中国安徽省合肥市包河区金寨路96号, Anhui 230026 (CN)。

(74) 代理人: 广州三环专利商标代理有限公司 (SCIHEAD IP LAW FIRM); 中国广东省广州市越秀区先烈中路80号汇华商贸大厦1508室, Guangdong 510070 (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, 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,

(54) Title: NEAR MEMORY COMPUTING ACCELERATOR, DUAL IN-LINE MEMORY MODULE AND COMPUTING DEVICE

(54) 发明名称: 近内存计算加速器、双列直插式内存模块以及计算设备



221 Near memory computing accelerator
410 Instruction decoder
411 Status register
412 Instruction cache queue
420 Computing module
421 Computing unit controller
422 Input unit
423 Computing unit array
424 Output unit
425 Cache unit
430 Controller
440 Multi-path selector
450 Memory group

图 5

(57) Abstract: A near memory computing accelerator, a dual in-line memory module and a computing device. The near memory computing accelerator in the present embodiment comprises a plurality of controllers and a computing module, wherein the plurality of controllers are used for acquiring, in parallel, a plurality of pieces of data from a plurality of memory blocks, and one of the plurality of controllers is used for acquiring data from one of the plurality of memory blocks; and the computing module is connected to the plurality of controllers, and is used for calculating the plurality of pieces of data. The near memory computing accelerator and the plurality of memory blocks are located on the same dual in-line memory module (DIMM). By means of the near memory computing accelerator, the acceleration effect can be improved when the data volume is relatively large or the number of vector dimensions of data is relatively large.

LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK,
MN, 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, GH, GM, KE, LR, LS, MW, MZ,
NA, RW, 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, 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:

一 包括国际检索报告 (条约第21条(3))。

(57) 摘要: 一种近内存计算加速器、双列直插式内存模块以及计算设备。本实施例的近内存计算加速器包括多个控制器以及计算模块。所述多个控制器用于并行获取多个内存块中的多个数据, 其中, 所述多个控制器中的一个控制器用于获取所述多个内存块中的一个内存块的数据。所述计算模块分别连接所述多个控制器, 并用于对所述多个数据进行计算, 其中, 所述近内存计算加速器以及所述多个内存块位于同一块双列直插式内存模块DIMM上。上述近内存计算加速器在数据量较多或者数据的向量维度较多时都能够提升加速效果。

近内存计算加速器、双列直插式内存模块以及计算设备

本申请要求于 2021 年 9 月 17 日提交中国专利局、申请号为 202111091356.7、申请名称为“近内存计算加速器、双列直插式内存模块以及计算设备”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

本申请涉及存储领域，尤其涉及一种近内存计算加速器、双列直插式内存模块以及计算设备。

背景技术

存储层通常包括高速缓存（cache）、内存（DRAM memory）和非易失性存储器（NVM）。在传统计算机系统中，数据从非易失性存储器逐级加载到高速缓存中，然后由处理器（central processing unit, CPU）对其进行计算。由于数据需要从非易失性存储器逐级加载到高速缓存（cache）中，因此，CPU 计算消耗数据的速度远高于内存读取数据的速度，严重制约着 CPU 发挥其应有的性能。

为了解决上述问题，现有技术提供了一种近内存计算加速系统，能够在内存附近设置具备计算能力的缓存芯片，由缓存芯片从内存取出数据，并对从内存取出数据进行计算，从而有效缩短数据从存取到计算的延迟。但是，现有的近内存计算加速系统只对向量维度较多的数据具有较好的加速作用，应用场景比较受限。

发明内容

为了解决上述问题，本申请提供了一种近内存计算加速器、双列直插式内存模块以及计算设备，能够在数据量较多或者数据的向量维度较多时都能提升加速效果。

第一方面，提供了一种近内存计算加速器，包括：

多个控制器，用于并行获取多个内存块中的多个数据，其中，所述多个控制器中的一个控制器用于获取所述多个内存块中的一个内存块的数据；

计算模块，分别连接所述多个控制器，用于对所述多个数据进行计算，其中，

所述近内存计算加速器以及所述多个内存块位于同一块双列直插式内存模块 DIMM 上。

上述方案中，近内存计算加速器中的计算模块通过多个控制器并发地从多个内存块获取多个数据，并对多个数据并行进行计算，因此，能够在数据量较多或者数据的向量维度较多时都能够提升加速效果。

在一些可能的设计中，所述计算包括：加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

上述方案中，近内存计算加速器的采用专有的指令集设计，指令集支持加法计算、乘法计算、除法计算、查找计算以及非线性计算等等计算，精简易用，无需复杂的解码设计，有利于减少芯片面积、功耗和成本。

在一些可能的设计中，所述加速器还包括指令解码器，所述指令解码器分别连接所述多

个控制器，

所述指令解码器用于对通用处理器 CPU 发送的指令进行解码，得到解码结果，所述解码结果用于指示待进行计算的所述多个数据的地址以及计算类型；

所述多个控制器用于根据所述多个数据的地址并行从所述多个内存块中获取所述多个数据；

所述计算模块用于根据所述计算类型对所述多个数据执行相应的计算。

在一些可能的设计中，所述计算模块包括缓存单元，所述缓存单元用于缓存对所述多个数据进行计算得到的计算结果。

上述方案中，可以在缓存单元中缓存对所述多个数据进行计算得到的计算结果，在进行累加时，可以从缓存单元中读取计算结果继续进行累加，从而减少读取内存的次数，提高了数据处理的速度。

第二方面，提供了一种双列直插式内存模块 DIMM，包括：

多个内存块，用于存储多个数据，其中，所述多个内存块中的一个内存块用于存储所述多个数据中的一个数据；

近内存计算加速器，分别连接所述多个内存块，用于从所述多个内存块中并行获取所述多个数据，对所述多个数据进行计算。

在一些可能的设计中，所述计算包括：加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

在一些可能的设计中，所述近内存计算加速器，包括：

多个控制器，用于并行获取多个内存块中的多个数据，其中，所述多个控制器中的一个控制器用于获取所述多个内存块中的一个内存块的数据；

计算模块，分别连接所述多个控制器，用于对所述多个数据进行计算。

在一些可能的设计中，所述近内存计算加速器还包括指令解码器，所述指令解码器分别连接所述多个控制器，

所述指令解码器用于对通用处理器 CPU 发送的指令进行解码，得到解码结果，所述解码结果用于指示待进行计算的所述多个数据的地址以及计算类型；

所述多个控制器用于根据所述多个数据的地址并行从所述多个内存块中获取所述多个数据；

所述计算模块用于根据所述计算类型对所述多个数据执行相应的计算。

在一些可能的设计中，所述计算模块包括缓存单元，所述缓存单元用于缓存对所述多个数据进行计算得到的计算结果。

第三方面，提供了一种计算设备，包括：

通用处理器 CPU，用于发送近内存计算指令；

双列直插式内存模块 DIMM，包括多个内存块以及近内存计算加速器，所述近内存计算加速器用于：

接收所述近内存计算指令，所述近内存计算指令用于指示待执行近内存计算的多个数据的地址以及计算类型；

根据所述多个数据的地址从所述多个内存块中并行获取所述多个数据；

根据所述计算类型对所述多个数据执行计算。

在一些可能的设计中，所述计算包括：加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

在一些可能的设计中，所述近内存计算加速器包括缓存单元，所述缓存单元用于缓存对所述多个数据进行计算得到的计算结果。

在一些可能的设计中，所述多个内存块被配置为允许所述近内存计算加速器访问时，禁止所述 CPU 访问。

在一些可能的设计中，所述 DIMM 还包括与所述多个内存块均不相同的一个或者多个内存块，所述一个或者多个内存块被配置为允许所述 CPU 访问时，禁止所述近内存计算加速器访问。

在一些可能的设计中，所述 DIMM 被配置为允许所述近内存计算加速器访问所述多个内存块中的一个内存块时，所述 CPU 同步访问所述一个或者多个内存块。

在一些可能的设计中，所述近内存计算加速器满足以下条件中的一个或者多个；

所述近内存计算加速器访问所述多个内存块中的一个内存块的能耗低于所述 CPU 访问所述一个或者多个内存块的能耗。

第四方面，提供了一种近内存计算方法，所述方法包括：

接收近内存计算指令，所述近内存计算指令用于指示待执行近内存计算的多个数据的地址以及计算类型；

根据所述多个数据的地址从多个内存块中并行获取所述多个数据；

根据所述计算类型对所述多个数据执行近内存计算。

在一些可能的设计中，所述近内存计算包括：加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

附图说明

为了更清楚地说明本申请实施例或背景技术中的技术方案，下面将对本申请实施例或背景技术中所需要使用的附图进行说明。

图 1 是本申请涉及的一种近内存计算加速系统的结构示意图；

图 2 是图 1 所示的近内存计算加速系统的运作机制的示意图；

图 3 是本申请提供的一种计算设备的示意图；

图 4 是本申请提供的一种 CPU 的结构示意图；

图 5 是本申请提供的一种 DIMM 的结构示意图；

图 6 是本申请提供的一种近内存计算指令的结构示意图；

图 7 是图 1 所示的近内存计算加速系统和图 3 所示的计算设备在各个场景下的加速效果的对比示意图；

图 8 是图 3 所示的计算设备分别在向量维度为 16 以及向量维度为 1024 时在不同的内存组数量下的加速效果的示意图；

图 9 是本申请提供的一种近内存计算方法的流程示意图。

具体实施方式

参见图 1，图 1 是本申请本申请涉及的一种近内存计算加速系统的结构示意图。如图 1

所示,所述近内存计算加速系统包括 CPU 110、多个双列直插式内存模块(dual inline memory modules, DIMM) 120 以及 DDR 总线 130。其中, CPU 110 和 DIMM 120 之间可以通过 DDR 总线 130 进行通信。

CPU 110 是计算中心,具有较高的计算能力,可以包括用于缓存数据的高速缓存。DIMM 120 包括动态随机存取存储器(dynamic random access memory, DRAM) 121 以及缓存芯片 122。缓存芯片 122 包括内存控制器和向量计算单元(NMP core)。其中, DRAM 121 用于存储数据。缓存芯片 122 中的本地内存控制器根据有限状态机控制逻辑,将高层次的 CPU 的指令翻译成低层次的 DRAM 121 的 C/A 命令,以将数据从 DRAM 121 中读取至缓存芯片 122 中的向量计算单元中执行计算任务,并将该计算任务的计算结果写回 DRAM 121 中。由此可见,如果利用 CPU 110 进行计算, CPU 110 需要将 DIMM 120 的 DRAM 121 中的数据(从 NVM 中读取的)通过 DDR 总线 130 读取到 CPU 110 的高速缓存中才能进行计算。而缓存芯片 122 是设置在 DIMM 120 中的,如果利用缓存芯片 122 进行计算,缓存芯片 122 可以直接访问 DIMM 120 的 DRAM 121 中的数据(从 NVM 中读取的)并进行计算,有效缩短数据从读取到计算的延迟。

下面将结合图 2 对图 1 所示的近内存计算加速系统运作机制进行详细的说明。

假设近内存计算加速系统中 DIMM 的数量为 m 个,每个 DIMM 中缓存芯片 122 的数量均为一个,计算任务涉及的从 NVM 中读取的数据包括数据 1、数据 2、数据 3、...、数据 n 。其中,数据 1 可以包括(子向量 1、子向量 2、...、子向量 m),数据 2 可以包括(子向量 1、子向量 2、...、子向量 m),...,数据 n 可以包括(子向量 1、子向量 2、...、子向量 m)。并且,数据 1 的子向量 1,数据 2 的子向量 1,...,数据 n 的子向量 1 均必须存储于 DIMM₁ 中;数据 1 的子向量 2,数据 2 的子向量 2,...,数据 n 的子向量 2 均必须存储于 DIMM₂;...;数据 1 的子向量 m ,数据 2 的子向量 m ,...,数据 n 的子向量 m 均必须存储于 DIMM _{m} 。DIMM₁ 中缓存芯片 1 对数据 1 的子向量 1,数据 2 的子向量 1,...,数据 n 的子向量 1 进行处理,得到计算结果的子向量 1; DIMM₂ 中缓存芯片 2 对数据 1 的子向量 2,数据 2 的子向量 2,...,数据 n 的子向量 2 进行处理,得到计算结果的子向量 2;...; DIMM _{m} 中缓存芯片 m 对数据 1 的子向量 m ,数据 2 的子向量 m ,...,数据 n 的子向量 m 进行处理,得到计算结果的子向量 m 。

在上述方案中,单个数据的各个子向量必须要分布存储在各个 DIMM 中,才能达到加速的效果。但是,当单个数据的向量维度不足以分布到各个 DIMM 的时候,例如,当单个数据的向量维度为 1 的时候,则只有 DIMM₁ 中的缓存芯片 1 进行工作,其他 DIMM 中的缓存芯片(例如,缓存芯片 2 至 m)都处于空闲状态,加速的效果将会受到影响。

参见图 3,图 3 是本申请提供的一种计算设备的示意图。如图 3 所示,本申请涉及的计算设备包括: CPU 210、一个或者多个 DIMM 220。其中, CPU 210 和 DIMM 220 之间可以通过总线 230 之间进行通信。

CPU 210 的体系结构可以分为两类:一类为精简指令集计算机(reduced instruction set computer, RISC),每条指令仅用于完成一项简单操作。对于经常使用的简单操作, RISC 结构的处理器可以以更快的速度执行指令,对不常用的操作, RISC 结构的处理器常通过组合指令来完成。另一类为复杂指令系统计算机(complex instruction set computer, CISC),每个指令可执行若干简单操作,例如从存储器读取、存储、和计算操作,全部集于单一复杂指令之中。CISC 结构的处理器指令系统比较丰富,有专用指令来完成特定的功能。在一具体的

实施例中,如图4所示,CPU 210 用于处理 DIMM 220 中存放的指令和数据。在一些实施例中,指令可以包括一个或多个指令格式。指令格式可以指示各种字段(位的数量、位的位置等)来指定待执行的数据处理的操作和操作数,其中该操作将在该操作数上被执行。一些指令格式可以进一步由指令模板(或子格式)定义。CPU 210 包括存储器地址寄存器 301、存储器数据寄存器 302、程序计数器 303、指令寄存器 304、指令译码器 305、操作控制器 307、计算单元 308、通用寄存器 309、累加器 310、程序状态寄存器 311、时序电路 312 以及处理器总线 313。其中,处理器总线 313 也可以是数据总线、电源总线、控制总线或者状态信号总线等。

存储器地址寄存器 301 用于保存 CPU 210 当前所要访问的 DIMM 220 中的内存的地址。

存储器数据寄存器 302 用于保存 CPU 210 从该地址中读取或写入的数据和读取或写入的指令,以便弥补 CPU 和存储器之间存在的操作速度上的差异。

程序计数器 303 用于存放下一条指令的地址,当指令顺序执行时,每取一条指令后程序计数器 303 自动加上一条指令的字节数。当遇到转移指令时,程序计数器 303 通过转移指令中的地址码字段来指定下一条指令的地址。

时序电路 312 通过固定时钟为各部件提供一个时间基准,CPU 210 执行一条指令的时间为一个指令周期。

指令寄存器 304 用于保存当前正在执行的指令。指令包括操作码和地址码两个字段,操作码部分由指令译码器 305 对操作码部分进行译码,以产生指令所要求操作的控制电位。操作控制器 307 可以根据指令译码器输出的控制电位信号和时序电路 312 产生的时序信号,生成各种操作控制信号,控制 CPU 210 其余部件完成取指令和执行指令的操作。

微指令是 CPU 210 执行指令的最小单位,一条指令可以是单个微指令,也可以由若干个微指令组成。由多条微指令组合的指令称为复杂指令,可以使用各种不同的机制来实现指令译码器 305 对复杂指令的解码。具体的解码机制包括但不限于查找表,硬件实现,可编程逻辑阵列(PLA),微码只读存储器(ROM)等方式。在一个实施例中,可以将复杂指令对应的微指令的步骤存储在微码只读存储器 306 中,指令译码器 305 在译码的过程中,可以从微码只读存储器 306 中查询并得到组成复杂指令的微指令的操作码和地址码,依次对微指令的操作码部分进行译码,产生微指令所要求操作的控制电位。

操作控制器 307 具有多个缓冲器,可以根据指令的种类,将解码后的指令送往各自的保留站中保存下来。并根据计算单元的硬件电路的状态和各指令能否提前执行的具体情况分析后,将能提前执行的指令调度给相应计算单元 308 执行。在这期间对指令流进行重新排序,以使指令流水线行进并被平滑调度。示例性地,对于整数计算的指令,操作控制器 307 可以使用整数保留站保存指令,并分配给整数计算单元执行计算;二对于浮点数计算的指令,操作控制器 307 可以使用浮点数保留站保存指令,并分配给浮点数计算单元执行计算。

通用寄存器组 309 用于根据指令的地址码,保存地址码对应的数据。计算单元 308 用于接收操作控制器 307 的操作控制信号并对保存在通用寄存器组 309 中的数据执行计算,包括算术计算(包括加减乘数等基本计算及其附加计算)和逻辑计算(包括移位、逻辑测试或两个值比较)。计算中产生的临时变量存放在累加器 310 中,产生的状态的信息存放在程序状态字寄存 311 中,例如计算结果进/借位标志(C)、计算结果溢出标志(O)、计算结果为零标志(Z)、计算结果为负标志(N)、计算结果符号标志(S)等。程序状态字寄存器还用来保存中断和计算设备工作状态等信息,以便 CPU 210 及时了解机器运行状态和程序运行状态。

计算单元 308 中包含多种不同的电路模块，可以分别用于执行不同的指令。例如，整数计算单元 3081 以及浮点数计算单元 3082，分别用于对整数和浮点数进行算术计算和逻辑计算。

应理解，图 4 所示的 CPU 可以包括更多或者更少的部件，或者，图 4 中的多个部件可以集成为一个部件，此处不作具体限定。

如图 5 所示，DIMM 是由一个或者多个近内存计算加速器 221 以及多个 DRAM 222 组成的模块。

DRAM 222 是一种半导体存储器，与大部分随机存取存储器 (random access memory, RAM) 一样，属于一种易失性存储器 (volatile memory) 设备。DIMM 中的 DRAM 222 的数量通常为多个。为了方便对 DRAM 222 进行管理，可以引用内存块的概念。内存块可以包括第一内存块以及第二内存块，其中，第一内存块被配置为允许所述近内存计算加速器访问时，禁止所述 CPU 访问；第二内存块被配置为禁止近内存计算加速器访问时，允许 CPU 访问。并且，第一内存块被近内存计算加速器访问的同时，CPU 可以同步访问第二内存块。这样，CPU 与近内存计算加速器可以同时访问各自的内存地址空间，互不干扰，解决了现有技术 CPU 与近内存计算加速器无法同时访问内存，在近内存计算期间严重影响系统其余应用处理效率的问题。其中，内存块可以是多个 DRAM 222 组成的内存组 (Rank)，也可以是将单个 DRAM 222 进行划分得到的内存库 (bank)，也可以是 DRAM 等等。举个例子说明，DIMM 的正反两面都具有多个 DRAM 222，可以将同一面的多个 DRAM 组成一个内存组。单个 DRAM 222 可以分成 8 个或 16 个内存库。可以理解，上述例子都仅仅作为具体的示例，在实际应用中，也可以将 DIMM 的正反两面的 DRAM 222 分成 4 个内存组，8 个内存组或者更少或更多内存组，也可以将单个 DRAM 222 可以分成 2 个内存库，4 个内存库或者更少或更多内存库等等，此处不作具体限定。

近内存计算加速器 221 分别与 DIMM 内的内存块相连，可并行访问第一内存块，从而增加内部等效数据访问带宽，并且，由于数据集中在近内存计算加速器 221 上处理，支持跨多个内存块数据访问与处理，数据分配无需限制在同一内存块上。

在一具体的实施例中，近内存计算加速器 221 包括指令解码器 410、计算模块 420、多个控制器 430 以及多个多路选择器 440。

指令解码器 410 用于接收 CPU 通过 DDR 总线发送的指令，对 CPU 发送的指令进行解码，得到解码结果，所述解码结果用于指示待进行计算的所述多个数据的地址以及计算类型。在一更具体的实施例中，指令解码器 410 包括状态寄存器 411 以及指令缓存队列 412。其中，指令包括普通指令以及近内存计算指令。如果指令是普通指令，指令解码器 410 将普通指令通过多路选择器 440 传递给对应的 DRAM 222。如果指令是近内存计算指令，指令解码器 410 阻止该近内存计算指令到达 DRAM 222，并将该近内存计算指令保存在指令缓存队列 412 中。状态寄存器 411 为内存可寻址的空间，当 CPU 向该地址发送读取请求，指令解码器 410 随即向 CPU 返回状态寄存器 411 中存储的近内存计算的工作状态。

计算模块 420 用于根据所述计算类型对所述多个数据执行相应的计算。计算模块 420 包括计算单元控制器 421、输入单元 422、计算单元阵列 423、输出单元 424 以及缓存单元 425。其中，计算单元控制器 421 用于控制计算单元阵列 423 执行近内存计算指令对相应的数据处理操作。输入单元 422 用于缓存从 DRAM 222 中读取准备用于执行近内存计算指令的数据。输出单元 424 用于缓存计算单元阵列 423 执行近内存计算指令之后得到的计算结果。缓存单元

425 用于存储执行近内存计算指令的数据处理过程中需要重复利用的数据。缓存单元 425 的容量可以根据需要进行设置，例如，可以是 4 千字节。计算单元阵列 423 总的吞吐带宽与 DIMM 条内所有内存组 450 并发访问数据总带宽相同，来自一个或多个内存组 450 的数据集中在计算单元阵列 423 进行处理。

控制器 430 接受指令解码器 410 发送过来的所述多个数据的地址和计算模块 420 输出的计算结果，生成对 DRAM 222 读写数据的操作命令。控制器 430 的数量为多个，例如，两个，三个或者更多。在一具体的实施例中，控制器 430 的数量可以与内存组 450 的数量相同，即，每个内存组 450 均具有独立的控制器 430。或者，控制器 430 的数量可以与内存库的数量相同，即，每个内存库均具有独立的控制器 430。或者，控制器 430 的数量可以与 DRAM 222 的数量相同，即，每个 DRAM 222 均具有独立的控制器 430。

多路选择器 440 用于根据指令解码器 410 的控制信号，选择将控制器 430 或者 CPU 的访问命令发送给 DRAM 222，以及，从 DRAM 222 中获取需要发送给控制器 430 以及 CPU 的数据。在一具体的实施中，多路选择器 440 的数量可以与控制器 430 的数量相同。在其他的实施例中，控制器 430 的数量也可以多于或者少于多路选择器 440 的数量，此处不作具体限定。

可以理解，图 5 所示 DIMM 仅仅作为一个具体的实例，在实际应用中，DIMM 可以包括更多的内存组、更多的近内存计算加速器，并且，近内存计算加速器可以包括更多或者更少的部件，此处不作具体限定。

近内存计算指令是 CPU 发送给 DIMM 的，属于 DIMM 支持的近内存计算指令集中的指令。其中，近内存计算指令集可以包括一条或者多条近内存计算指令。不同的近内存计算指令集可以支持不同的近内存计算指令。近内存计算指令集中指令的内容以及指令的数量等等均可以根据用户的需求进行设置。当近内存计算指令集越复杂，则 DIMM 支持的近内存计算越复杂，但是，DIMM 的电路的设计越复杂，计算的效率越低；相反，当近内存计算指令集越简单，则 DIMM 支持的近内存计算越简单，但是，DIMM 的电路的设计越简单，计算的效率越高。在一具体的实施例中，近内存计算指令集支持的计算包括加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

近内存计算指令通常包括操作码、子操作码、非线性操作码、向量维度、输入地址 1、输入地址 2，输出地址，常数 1 以及常数 2。其中，操作码用于指示可以进行近内存计算的计算类型，例如，操作码可以指示加法计算、乘法计算、除法计算以及查找计算等等计算类型中的一种或者多种。子操作码用于指示每种计算类型包括的子计算类型，例如，当操作码为加法时，子操作码可以包括（从内存读，写回缓存）、（从缓存读，写回缓存）、（从缓存读，写回内存）、（从内存读，写回内存）等等子计算类型中的一种或者多种。当操作码为乘法时，子操作码可以包括向量对应元素积、向量数乘、向量内积等等子计算类型中的一种或者多种。当操作码为除法时，子操作码可以包括向量对应元素相除、向量按元素除以常数、常数按元素除以向量等等子计算类型中的一种或者多种。当操作码为查找时，子操作码可以包括归纳和匹配等等子计算类型中的一种或者多种。非线性操作码可以与操作码结合，用于指示是否需要对该操作码对应的计算进行非线性计算。向量维度用于指示进行近内存计算的数据的向量维度。输入地址 1 为进行近内存计算的第一数据所在的地址。输入地址 2 为进行近内存计算的第二数据所在的地址。输出地址为近内存计算的计算结果的存储地址。常数 1 和常数 2 为进行近内存计算的常数。

近内存计算指令集按照指令执行的功能分可以包括：加法操作、乘法操作、除法操作、查找操作等等。因此，可以用 2 位二进制数的操作码来表示。每种操作对应的子操作数量最多为 4 种，因此，也可以用 2 位二进制数子操作码的来表示。综合来说，采用 4 位二进制数可以表示近内存计算指令集支持的所有操作。非线性操作一般是在计算结果的基础上进行的，因此，加法操作对应的指令、乘法操作对应的指令和除法操作对应的指令之外还需要额外的 2 位二进制数的非线性操作码指示是否需要对计算结果进行非线性变换，以及采用何种非线性变换函数。在内存 BL8 访问模式下，一次访问可返回 64 字节的数据，因此，指令中的向量维度最大值设置为 64，可以采用 6 位二进制数或者 7 位二进制数的向量维度来进行表示。这样一条指令可以多次迭代执行，每次执行按字节索引的地址值增加 64，从而可以通过一条指令访问 4 千字节的物理页空间。对于大于 4 千字节的数据，需要分多条指令完成访问。在双倍速率 SDRAM (double-data-rate fourth generation synchronous dynamic random access memory, DDR4 SDRAM) 的规范中，行地址最多有 18 位二进制数，列地址为 10 位二进制数，库组 (bank group, BG) 和库 (Bank, BA) 索引各 2 位二进制数，再加上 CS0-CS4 共 4 位内存组的片选信号，地址的长度最多为 36 位二进制数。根据不同的指令，36 位二进制数的地址的低 32 位也可用于传输 32 位二进制数的数据。综合上述，近内存计算指令的长度最多有 121 位二进制数：4 位二进制数（操作码+子操作码）+2 位二进制数（非线性操作码）+7 位二进制数（向量维度）+36 位二进制数（输入地址 1）+36 位二进制数（输入地址 2）+36 位二进制数（输出地址）。DDR 接口的双向数据控制引脚 (bi-directional data strobe, DQ) 总线位宽为 64-bit，因此可以通过 2 个 DDR 时钟周期传输 128 位二进制数数据，用 128 位二进制数数据来存储近内存计算指令。

如图 6 所示，近内存计算指令可以采用如下格式：4 位的操作码字段、2 位的非线性操作码字段、7 位的向量维度字段、36 位的第一输入字段、36 位的第二输入字段、36 位的输出字段以及 6 位的保留字段。其中，操作码字段用于存储操作码以及子操作码。非线性操作码字段用于存储非线性操作码。向量维度字段用于存储向量维度。第一输入字段用于存储输入地址 1 或者常数 1。第二输入字段用于存储输入地址 2 或者常数 2。输出字段用于存储输出地址。

可以理解，图 6 所示的近内存计算指令的各个字段的长度，各个字段的内容，总的字段的数量均可以按照需要进行设置，此处不作具体限定。

下面将以表 1 所示的近内存计算指令集为例，对近内存计算指令集中的各个近内存计算指令进行详细的介绍：

表 1 近内存计算指令集

近内存计算指令	操作码	子操作码	非线性操作码	向量维度	输入地址 1/常数 1	输入地址 2/常数 2	输出地址
指令 1	加法	从内存读，写回缓存	非线性操作码	向量维度	输入内存地址 1	输入内存地址 2	输出缓存地址
指令 2		从缓存读，写回缓存	非线性操作码	向量维度	输入内存地址 1	输入缓存地址	输出缓存地址
指令 3		从缓存读，写回内存	非线性操作码	向量维度	输入内存地址 1	输入缓存地址	输出内存地址
指令 4		从内存读，	非线性操	向量维	输入内存	输入内存	输出内存

		写回内存	作码	度	地址 1	地址 2	地址
指令 5	乘法	向量对应元素积	非线性操作码	向量维度	输入内存地址 1	输入内存地址 2	输出内存地址
指令 6		向量数乘	非线性操作码	向量维度	输入内存地址 1	常数	输出内存地址
指令 7		向量内积	非线性操作码	向量维度	输入内存地址 1	输入内存地址 2	输出内存地址
指令 8	除法	向量对应元素相除	非线性操作码	向量维度	输入内存地址 1	输入内存地址 2	输出内存地址
指令 9		向量按元素除以常数	非线性操作码	向量维度	输入内存地址 1	除数	输出内存地址
指令 10		常数按元素除以向量	非线性操作码	向量维度	被除数	输入内存地址 2	输出内存地址
指令 11	查找	归纳	—	向量维度	输入内存地址 1	待归纳数据	输出内存地址
指令 12		匹配	—	向量维度	输入内存地址 1	待匹配数据	输出内存地址

接下来，将对每一条指令进行详细说明：

指令 1 表示从内存（DRAM）的输入地址 1 和内存（DRAM）的输入地址 2 分别读取两个数据，将两个数据按元素进行相加得到计算结果，计算结果保存在近内存计算加速器的缓存单元的输出缓存地址中。

指令 2 表示从内存（DRAM）的输入地址 1 中读取数据 1，从近内存计算加速器的缓存单元的输入缓存地址读取数据 2，将两个数据按元素进行相加得到计算结果，计算结果保存在近内存计算加速器的缓存单元的输出缓存地址中。

指令 3 表示从内存（DRAM）的输入地址 1 中读取数据 1，从近内存计算加速器的缓存单元的输入缓存地址读取数据 2，将两个数据按元素进行相加得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 4 表示从内存（DRAM）的输入地址 1 和内存（DRAM）的输入地址 2 分别读取两个数据，将两个数据进行相加得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 5 表示从内存（DRAM）的输入地址 1 和内存（DRAM）的输入地址 2 分别读取两个数据，将两个数据的对应元素进行相乘得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 6 表示从内存（DRAM）的输入地址 1 中读取数据 1，将数据 1 和常数进行数乘得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 7 表示从内存（DRAM）的输入地址 1 和内存（DRAM）的输入地址 2 分别读取两个数据，将两个数据进行内积计算得到计算结果，计算结果保存在内存（DRAM）的输出内存地址

中。

指令 8 表示从内存（DRAM）的输入地址 1 和内存（DRAM）的输入地址 2 分别读取两个数据，将两个数据的对应元素进行相除得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 9 表示从内存（DRAM）的输入地址 1 中读取数据 1，将数据 1 按元素除以除数得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 10 表示从内存（DRAM）的输入地址 2 中读取数据 2，将被除数按元素除以数据 2 的各个子向量得到计算结果，计算结果保存在内存（DRAM）的输出内存地址中。

指令 11 表示从内存（DRAM）的输入地址 1 中读取数据 1，数据 1 中的一个元素代表一条源顶点到目标顶点的边，数据 1 由连续的边数据组成。将数据 1 按元素与输入的待归纳数据进行比较，按元素与输入的待归纳数据（即目标顶点索引）进行比较，若目标顶点与边数据匹配，则记录相应的源顶点索引，并保存到近内存计算加速器的缓存单元中。在完成对数据 1 存储的边数据比较后，所有匹配的源顶点索引从近内存计算加速器的缓存单元中读出，并写回内存（DRAM）的输出内存地址中。

指令 12 表示从内存（DRAM）的输入地址 1 中读取数据 1，将数据 1 按元素与输入的待匹配数据进行比较，统计与输入数据相匹配的数据 1 中的子向量的个数。在完成对数据 1 中的子向量的个数的比较后，统计数量写回内存（DRAM）的输出内存地址中。

下面将以指令 6 为例详细介绍指令是如何执行的。以图神经网络领域（Graph Convolutional Network, GCN）算法的顶点更新过程为例：

$$h_v^y = \frac{1}{\sqrt{\tilde{D}_v}} \times h_v^y$$

使用指令 6 从内存（DRAM）中读取顶点的特征向量 h_v^l ，与指令输入的常系数 $\frac{1}{\sqrt{\tilde{D}_v}}$ 进行数

乘操作得到的顶点更新特征向量 h_v^l ，并将写回内存（DRAM）的输出内存地址中。

上述近内存计算指令均以简单指令为例进行说明，实际上近内存计算指令还可以是复杂指令，即，近内存计算指令可以是多个简单指令的组合，例如，聚合（gather）指令、嵌入（embedding）指令等等。

下面详细说明如何使用上述表 1 所示的近内存计算指令集完成聚合（gather）指令的操作。

以 GCN 算法的聚合过程为例：

$$h_u^{l+1} = \sum_{v \rightarrow u \cup v=u} h_v^l$$

其中， h_v^l 表示第 l 层顶点特征向量， h_u^{l+1} 表示第 $l+1$ 层顶点特征向量， v 代表源顶点， u 代表目标顶点， $v \rightarrow u$ 表示 v 是目标顶点 u 的相邻顶点。

假设目标顶点 u 的相邻顶点 v 数量较多，首先，使用指令 1 将保存在内存（DRAM）中的两个相邻的顶点特征向量读取到近内存计算加速器中，并按元素进行相加得到第一计算结果，并将第一计算结果保存在近内存计算加速器的缓存单元的输出缓存地址中。

然后，使用多条指令 2，继续从内存中逐条加载新的相邻的顶点特征向量，并与保存在

近内存计算加速器的缓存单元中的计算结果按元素进行相加得到第二计算结果，并将第二计算结果保存在近内存计算加速器的缓存单元的输出缓存地址中。如此，可以重复利用近内存计算加速器的缓存单元的中间计算结果，避免每次累加操作都需要读写计算结果。

当加载最后一个相邻顶点特征向量时，使用指令 3 从内存中加载最后一个相邻的顶点特征向量，并与保存在近内存计算加速器的缓存单元中的计算结果按元素进行相加得到第三计算结果，并将第三计算结果写回内存（DRAM）的输出内存地址中。

假如目标顶点 u 只有一个相邻顶点，则直接使用指令 4，将保存在内存（DRAM）中的相邻的顶点特征向量和目标顶点的特征向量读取到近内存计算加速器中，并按元素做加法操作得到第四计算结果，并将第四结果直接写回内存（DRAM）的输出内存地址中。

在上述计算设备中，无论是数据的向量维度比较多（不论数据量比较少还是数据量比较多）时，或者，数据的量比较大（不论数据的向量维度比较少还是数据的向量维度比较多）时，加速的效果都非常好。

下面将结合图 5 所示的计算设备为例，并以数据的向量维度比较长为例，对图 5 所示的计算设备的工作过程进行详细的介绍。

假设计算设备中 DIMM 的数量为 m 个，每个 DIMM 中包括两个内存组，每个内存组包括 4 个 DRAM，数据 1 的向量维度为 $8m$ ，数据 2 的向量维度为 $8m$ 。其中，

数据 1 的子向量 a_1 被存储在 DIMM₁ 的内存组 1 的 DRAM₁ 中，数据 1 的子向量 a_2 被存储在 DIMM₁ 的内存组 1 的 DRAM₂ 中，数据 1 的子向量 a_3 被存储在 DIMM₁ 的内存组 1 的 DRAM₃ 中，数据 1 的子向量 a_4 被存储在 DIMM₁ 的内存组 1 的 DRAM₄ 中，数据 1 的子向量 a_5 被存储在 DIMM₁ 的内存组 2 的 DRAM₁ 中，数据 1 的子向量 a_6 被存储在 DIMM₁ 的内存组 2 的 DRAM₂ 中，数据 1 的子向量 a_7 被存储在 DIMM₁ 的内存组 2 的 DRAM₃ 中，数据 1 的子向量 a_8 被存储在 DIMM₁ 的内存组 2 的 DRAM₄ 中；

数据 1 的子向量 a_9 被存储在 DIMM₂ 的内存组 1 的 DRAM₁ 中，数据 1 的子向量 a_{10} 被存储在 DIMM₂ 的内存组 1 的 DRAM₂ 中，数据 1 的子向量 a_{11} 被存储在 DIMM₂ 的内存组 1 的 DRAM₃ 中，数据 1 的子向量 a_{12} 被存储在 DIMM₂ 的内存组 1 的 DRAM₄ 中，数据 1 的子向量 a_{13} 被存储在 DIMM₂ 的内存组 2 的 DRAM₁ 中，数据 1 的子向量 a_{14} 被存储在 DIMM₂ 的内存组 2 的 DRAM₂ 中，数据 1 的子向量 a_{15} 被存储在 DIMM₂ 的内存组 2 的 DRAM₃ 中，数据 1 的子向量 a_{16} 被存储在 DIMM₂ 的内存组 2 的 DRAM₄ 中；

...；

数据 1 的子向量 a_{8m-7} 被存储在 DIMM _{m} 的内存组 1 的 DRAM₁ 中，数据 1 的子向量 a_{8m-6} 被存储在 DIMM _{m} 的内存组 1 的 DRAM₂ 中，数据 1 的子向量 a_{8m-5} 被存储在 DIMM _{m} 的内存组 1 的 DRAM₃ 中，数据 1 的子向量 a_{8m-4} 被存储在 DIMM _{m} 的内存组 1 的 DRAM₄ 中，数据 1 的子向量 a_{8m-3} 被存储在 DIMM _{m} 的内存组 2 的 DRAM₁ 中，数据 1 的子向量 a_{8m-2} 被存储在 DIMM _{m} 的内存组 2 的 DRAM₂ 中，数据 1 的子向量 a_{8m-1} 被存储在 DIMM _{m} 的内存组 2 的 DRAM₃ 中，数据 1 的子向量 a_{8m} 被存储在 DIMM _{m} 的内存组 2 的 DRAM₄ 中。

同理，数据 2 的子向量 b_1 被存储在 DIMM₁ 的内存组 1 的 DRAM₁ 中，数据 2 的子向量 b_2 被存储在 DIMM₁ 的内存组 1 的 DRAM₂ 中，数据 2 的子向量 b_3 被存储在 DIMM₁ 的内存组 1 的 DRAM₃ 中，数据 2 的子向量 b_4 被存储在 DIMM₁ 的内存组 1 的 DRAM₄ 中，数据 2 的子向量 b_5 被存储在 DIMM₁ 的内存组 2 的 DRAM₁ 中，数据 2 的子向量 b_6 被存储在 DIMM₁ 的内存组 2 的 DRAM₂ 中，数据 2 的

子向量 b_7 被存储在 DIMM₁ 的内存组 2 的 DRAM₃ 中, 数据 2 的子向量 b_8 被存储在 DIMM₁ 的内存组 2 的 DRAM₄ 中;

数据 2 的子向量 b_9 被存储在 DIMM₂ 的内存组 1 的 DRAM₁ 中, 数据 2 的子向量 b_{10} 被存储在 DIMM₂ 的内存组 1 的 DRAM₂ 中, 数据 2 的子向量 b_{11} 被存储在 DIMM₂ 的内存组 1 的 DRAM₃ 中, 数据 2 的子向量 b_{12} 被存储在 DIMM₂ 的内存组 1 的 DRAM₄ 中, 数据 2 的子向量 b_{13} 被存储在 DIMM₂ 的内存组 2 的 DRAM₁ 中, 数据 2 的子向量 b_{14} 被存储在 DIMM₂ 的内存组 2 的 DRAM₂ 中, 数据 2 的子向量 b_{15} 被存储在 DIMM₂ 的内存组 2 的 DRAM₃ 中, 数据 2 的子向量 b_{16} 被存储在 DIMM₂ 的内存组 2 的 DRAM₄ 中;

...;

数据 2 的子向量 b_{8m-7} 被存储在 DIMM_m 的内存组 1 的 DRAM₁ 中, 数据 2 的子向量 b_{8m-6} 被存储在 DIMM_m 的内存组 1 的 DRAM₂ 中, 数据 2 的子向量 b_{8m-5} 被存储在 DIMM_m 的内存组 1 的 DRAM₃ 中, 数据 2 的子向量 b_{8m-4} 被存储在 DIMM_m 的内存组 1 的 DRAM₄ 中, 数据 2 的子向量 b_{8m-3} 被存储在 DIMM_m 的内存组 2 的 DRAM₁ 中, 数据 2 的子向量 b_{8m-2} 被存储在 DIMM_m 的内存组 2 的 DRAM₂ 中, 数据 2 的子向量 b_{8m-1} 被存储在 DIMM_m 的内存组 2 的 DRAM₃ 中, 数据 2 的子向量 b_{8m} 被存储在 DIMM_m 的内存组 2 的 DRAM₄ 中。

DIMM₁ 中的计算模块通过控制器 1 控制多路选择器 1 从内存组 1 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 的子向量 a_1 至子向量 a_4 , 以及数据 2 的子向量 b_1 至子向量 b_4 , 并保存在输入单元中。DIMM₁ 中的计算模块通过控制器 2 控制多路选择器 2 从内存组 2 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 的子向量 a_5 至子向量 a_8 , 以及数据 2 的子向量 b_5 至子向量 b_8 , 并保存在输入单元中。计算单元阵列将数据 1 的子向量 a_1 至子向量 a_8 , 数据 2 的子向量 b_1 至子向量 b_8 分别进行计算处理, 得到计算结果, 并将计算结果保存在输出单元或者缓存单元中;

DIMM₂ 中的计算模块通过控制器 1 控制多路选择器 1 从内存组 1 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 的子向量 a_9 至子向量 a_{12} , 以及数据 2 的子向量 b_9 至子向量 b_{12} , 并保存在输入单元中。DIMM₂ 中的计算模块通过控制器 2 控制多路选择器 2 从内存组 2 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 的子向量 a_{13} 至子向量 a_{16} , 以及数据 2 的子向量 b_{13} 至子向量 b_{16} , 并保存在输入单元中。计算单元阵列将数据 1 的子向量 a_9 至子向量 a_{16} , 数据 2 的子向量 b_9 至子向量 b_{16} 分别进行计算处理, 得到计算结果, 并将计算结果保存在输出单元或者缓存单元中;

...;

DIMM_m 中的计算模块通过控制器 1 控制多路选择器 1 从内存组 1 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 的子向量 a_{m-7} 至子向量 a_{m-4} , 以及数据 2 的子向量 b_{m-7} 至子向量 b_{m-4} , 并保存在输入单元中。DIMM_m 中的计算模块通过控制器 2 控制多路选择器 2 从内存组 2 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 的子向量 a_{m-3} 至子向量 a_m , 以及数据 2 的子向量 b_{m-3} 至子向量 b_m , 并保存在输入单元中。计算单元阵列将数据 1 的子向量 a_{m-7} 至子向量 a_m , 数据 2 的子向量 b_{m-7} 至子向量 b_m 分别进行计算处理, 得到计算结果, 并将计算结果保存在输出单元或者缓存单元中。

从上述实施例可以看出, 在数据的向量维度比较长的情况下, 可以将数据的子向量分别设置在不同的内存组中, 近内存计算加速器通过多个控制器分别从多个内存组中同步读取数据的子向量并进行计算, 从而起到很好的加速效果。

下面将结合图 5 所示的计算设备为例, 并以数据的数据量比较多为例, 对图 5 所示的计

算设备的工作过程进行详细的介绍。

假设计算设备中 DIMM 的数量为 m 个，每个 DIMM 中包括两个内存组，每个内存组包括 4 个 DRAM，数据的数量为 $16m$ 。其中，

数据 1 被存储在 DIMM₁ 的内存组 1 的 DRAM₁ 中，数据 2 被存储在 DIMM₁ 的内存组 1 的 DRAM₂ 中，数据 3 被存储在 DIMM₁ 的内存组 1 的 DRAM₃ 中，数据 4 被存储在 DIMM₁ 的内存组 1 的 DRAM₄ 中，数据 5 被存储在 DIMM₁ 的内存组 2 的 DRAM₁ 中，数据 6 被存储在 DIMM₁ 的内存组 2 的 DRAM₂ 中，数据 7 被存储在 DIMM₁ 的内存组 2 的 DRAM₃ 中，数据 8 被存储在 DIMM₁ 的内存组 2 的 DRAM₄ 中；

数据 9 被存储在 DIMM₂ 的内存组 1 的 DRAM₁ 中，数据 10 被存储在 DIMM₂ 的内存组 1 的 DRAM₂ 中，数据 11 被存储在 DIMM₂ 的内存组 1 的 DRAM₃ 中，数据 12 被存储在 DIMM₂ 的内存组 1 的 DRAM₄ 中，数据 13 被存储在 DIMM₂ 的内存组 2 的 DRAM₁ 中，数据 14 被存储在 DIMM₂ 的内存组 2 的 DRAM₂ 中，数据 15 被存储在 DIMM₂ 的内存组 2 的 DRAM₃ 中，数据 16 被存储在 DIMM₂ 的内存组 2 的 DRAM₄ 中；

...；

数据 $8m-7$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₁ 中，数据 $8m-6$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₂ 中，数据 $8m-5$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₃ 中，数据 $8m-4$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₄ 中，数据 $8m-3$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₁ 中，数据 $8m-2$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₂ 中，数据 $8m-1$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₃ 中，数据 $8m$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₄ 中；

数据 $8m+1$ 被存储在 DIMM₁ 的内存组 1 的 DRAM₁ 中，数据 $8m+2$ 被存储在 DIMM₁ 的内存组 1 的 DRAM₂ 中，数据 $8m+3$ 被存储在 DIMM₁ 的内存组 1 的 DRAM₃ 中，数据 $8m+4$ 被存储在 DIMM₁ 的内存组 1 的 DRAM₄ 中，数据 $8m+5$ 被存储在 DIMM₁ 的内存组 2 的 DRAM₁ 中，数据 $8m+6$ 被存储在 DIMM₁ 的内存组 2 的 DRAM₂ 中，数据 $8m+7$ 被存储在 DIMM₁ 的内存组 2 的 DRAM₃ 中，数据 $8m+8$ 被存储在 DIMM₁ 的内存组 2 的 DRAM₄ 中；

数据 $8m+9$ 被存储在 DIMM₂ 的内存组 1 的 DRAM₁ 中，数据 $8m+10$ 被存储在 DIMM₂ 的内存组 1 的 DRAM₂ 中，数据 $8m+11$ 被存储在 DIMM₂ 的内存组 1 的 DRAM₃ 中，数据 $8m+12$ 被存储在 DIMM₂ 的内存组 1 的 DRAM₄ 中，数据 $8m+13$ 被存储在 DIMM₂ 的内存组 2 的 DRAM₁ 中，数据 $8m+14$ 被存储在 DIMM₂ 的内存组 2 的 DRAM₂ 中，数据 $8m+15$ 被存储在 DIMM₂ 的内存组 2 的 DRAM₃ 中，数据 $8m+16$ 被存储在 DIMM₂ 的内存组 2 的 DRAM₄ 中；

...；

数据 $16m-7$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₁ 中，数据 $16m-6$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₂ 中，数据 $16m-5$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₃ 中，数据 $16m-4$ 被存储在 DIMM _{m} 的内存组 1 的 DRAM₄ 中，数据 $16m-3$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₁ 中，数据 $16m-2$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₂ 中，数据 $16m-1$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₃ 中，数据 $16m$ 被存储在 DIMM _{m} 的内存组 2 的 DRAM₄ 中。

DIMM₁ 中的计算模块通过控制器 1 控制多路选择器 1 从内存组 1 中的 DRAM₁ 至 DRAM₄ 中读取数据 1 至数据 4，以及数据 $8m+1$ 至数据 $8m+4$ ，并保存在输入单元中。DIMM₁ 中的计算模块通过控制器 2 控制多路选择器 2 从内存组 2 中的 DRAM₁ 至 DRAM₄ 中读取数据 5 至数据 8，以及数据 $8m+5$ 至数据 $8m+8$ ，并保存在输入单元中。计算单元阵列将数据 1 至数据 8，数据 $8m+1$ 至

数据 $8m+8$ 分别进行计算处理, 得到计算结果, 并将计算结果保存在输出单元或者缓存单元中;

DIMM₂ 中的计算模块通过控制器 1 控制多路选择器 1 从内存组 1 中的 DRAM₁ 至 DRAM₄ 中读取数据 9 至数据 12, 以及数据 $8m+9$ 至数据 $8m+12$, 并保存在输入单元中。DIMM₂ 中的计算模块通过控制器 2 控制多路选择器 2 从内存组 2 中的 DRAM₁ 至 DRAM₄ 中读取数据 13 至数据 16, 以及数据 $8m+13$ 至数据 $8m+16$, 并保存在输入单元中。计算单元阵列将数据 9 至数据 16, 数据 $8m+9$ 至数据 $8m+16$ 分别进行计算处理, 得到计算结果, 并将计算结果保存在输出单元或者缓存单元中;

...;

DIMM_n 中的计算模块通过控制器 1 控制多路选择器 1 从内存组 1 中的 DRAM₁ 至 DRAM₄ 中读取数据 $8m-7$ 至数据 $8m-4$, 以及数据 $16m-7$ 至数据 $16m-4$, 并保存在输入单元中。DIMM_n 中的计算模块通过控制器 2 控制多路选择器 2 从内存组 2 中的 DRAM₁ 至 DRAM₄ 中读取数据 $8m-3$ 至数据 $8m$, 以及数据 $16m-3$ 至数据 $16m$, 并保存在输入单元中。计算单元阵列将数据 $8m-7$ 至数据 $8m$, 数据 $16m-7$ 至数据 $16m$ 分别进行计算处理, 得到计算结果, 并将计算结果保存在输出单元或者缓存单元中。

从上述实施例可以看出, 在数据的数量比较多的情况下, 可以将数据分别设置在不同的内存组中, 近内存计算加速器通过多个控制器分别从多个内存组中同步读取数据并进行计算, 从而起到很好的加速效果。

为了便于说明本申请实施例提供的计算设备的加速效果, 下面以图神经网络领域的 GCN 算法为例进行仿真, 并通过仿真结果来观察本申请实施例提供的计算设备的加速效果。

图神经网络领域的 GCN 算法可以表示为如下几个步骤:

$$\text{更新顶点: } h_v^i = W^i \times h_v^l;$$

$$h_v^i = \frac{1}{\sqrt{\tilde{D}_v}} \times h_v^i;$$

$$\text{分散: } edge^l = h_v^i;$$

$$\text{申请边: } edge^{l+1} = edge^l;$$

$$\text{聚合: } h_u^{i+1} = \sum_{v \rightarrow u \cup v=u} h_v^i;$$

$$\text{更新顶点: } h_u^{l+1} = RELU \left(\frac{1}{\sqrt{\tilde{D}_v}} \times h_v^i \right)。$$

其中, 聚合 (Gather) 操作对目标顶点的相邻顶点的特征向量进行累加, 更新顶点 (ApplyVertex) 操作是对顶点的特征向量进行线性变换。若将聚合 (Gather) 操作和更新顶点 (ApplyVertex) 操作卸载到近内存计算加速器, 可以极大地减少 CPU 与内存之间搬运的数据量。

因此, 基于模拟器搭建了计算设备的仿真测试平台, 在仿真测试平台上实现并部署 GCN 算法中的聚合操作和更新顶点操作, 以评估近内存计算加速器对 GNN 算法的关键函数的加速效

果。

在一具体的实施中，可以采用科拉（Cora）数据集以及雷迪特（Reddit）作为测试数据集测试本实施例中的计算设备采用近内存计算加速器对GCN算法中的聚合操作和更新顶点操作进行加速的加速作用的大小。其中，科拉数据集是数据量比较少的数据集，雷迪特数据集是数据量比较多的数据集。此外，通过调整GCN隐藏层的向量维度分别为16、1024来研究向量维度的大小对近内存计算加速器的加速效果的影响。因此，通过不同数据集与向量维度的组合，可以涵盖低数据量低向量维度、低数据量高向量维度、高数据量低向量维度、高数据量高向量维度四种不同的应用场景。以表2为例，科拉数据集以及雷迪特数据集可以是：

表2 测试数据集

测试数据集	科拉数据集	雷迪特数据集
顶点数	2780	232965
边数	10556	114615892
边数/顶点数	3.90	492
测试集顶点数	1000	20
测试集顶点平均度数	3.7	358.7
测试顶点特征向量维度	16或者1024	16或者1024

参见图7，图7是图1所示的近内存计算加速系统和图3所示的计算设备在各个场景下的加速效果的对比示意图。将表2所示的两个不同的科拉数据集以及雷迪特数据集分别采用16维的向量维度和1024维的向量维度对图1所示的近内存计算加速系统和图3所示的计算设备的加速效果进行仿真可以得到的如图7所示的对比示意图。明显可见，在采用科拉数据集，向量维度为16时，图1所示的近内存计算加速系统和图3所示的计算设备的加速效果基本相同；在采用科拉数据集，向量维度为1024时，图3所示的计算设备的加速效果优于图1所示的近内存计算加速系统的加速效果；在采用雷迪特数据集，向量维度为16时，图3所示的计算设备的加速效果优于图1所示的近内存计算加速系统的加速效果；在采用雷迪特数据集，向量维度为1024时，图3所示的计算设备的加速效果优于图1所示的近内存计算加速系统的加速效果。所以，在数据量较多或者数据的向量维度较多时，图3所示的计算设备都能够保证比较好的加速效果。

参见图8，图8是图3所示的计算设备分别在向量维度为16以及向量维度为1024时在不同的内存组数量下的加速效果的示意图。明显地，在向量维度为16以及向量维度为1024时，如果其他的参数都不变，只有DIMM中的内存组的数量不相同，则DIMM中的内存组的数量越多，计算设备的加速效果越好。

下面将结合图3至图5所示的计算设备，对该计算设备内如何进行近内存计算进行详细的说明。参见图9，图9是本申请提供的一种近内存计算方法的流程示意图。如图9所示，本申请实施例中的近内存计算方法包括如下步骤：

S101：通用处理器CPU向DIMM中的近内存计算加速器中发送近内存计算指令。相应地，DIMM中的近内存计算加速器接收CPU发送的近内存计算指令。

在一具体的实施中，通用处理器CPU、DIMM、近内存计算加速器以及指令解码器的结构可以参见图4、图5以及相关描述，此处不在展开描述。

在一具体的实施中，近内存计算指令的详细内容请参见上文，此处不在展开描述。

S102: 近内存计算加速器对近内存计算指令进行解码, 得到解码结果。其中, 所述解码结果指示多个数据的地址以及计算类型。

在一具体的实施例中, 近内存计算加速器可以通过内部的指令解码器对近内存计算指令进行解码, 得到解码结果。

在一具体的实施例中, 多个数据的地址被设置为允许近内存计算加速器进行访问, 禁止CPU进行访问。

在一具体的实施例中, 计算类型为加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。并且, 计算类型需为近内存计算指令集支持的类型。近内存计算指令集的相关内容可以参见上文中的相关介绍。

S103: 近内存计算加速器根据所述多个数据的地址从所述多个内存块中并行获取所述多个数据。

在一具体的实施例中, 内存块可以是CPU以及近内存计算加速器均可以访问的, 也可以是, 近内存计算加速器允许访问, 而, CPU禁止访问的。

S104: 近内存计算加速器根据所述计算类型对所述多个数据执行计算。

在一具体的实施例中, 近内存计算加速器可以先确定计算类型是加法计算、减法计算、乘法计算以及除法计算中的任意一种。当计算类型是加法计算时, 再确定其子计算类型是(从内存读, 写回缓存)、(从缓存读, 写回缓存)、(从缓存读, 写回内存)、(从内存读, 写回内存)中的任意一种。当计算类型是乘法计算时, 再确定其子计算类型是向量对应元素积、向量数乘、向量内积等等子计算类型中的一种或者多种。当计算类型是为除法时, 再确定其子计算类型是向量对应元素相除、向量按元素除以常数、常数按元素除以向量中的任意一种。当计算类型为查找计算时, 再确定其子计算类型是归纳和匹配中的任意一种。如果计算类型是加法计算、减法计算、乘法计算以及除法计算中的任意一种时, 还需要确定是否需要对计算结果进行非线性计算。

为了简便起见, 本实施例中的近内存计算方法并没有进行详细的介绍, 具体请参见图3-图8以及相关描述。

在上述实施例中, 可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时, 可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行所述计算机程序指令时, 全部或部分地产生按照本申请实施例所述的流程或功能。所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中, 或者从一个计算机可读存储介质向另一个计算机可读存储介质传输, 例如, 所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线(例如同轴电缆、光纤、数字用户线)或无线(例如红外、无线、微波等)方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质, (例如, 软盘、存储盘、磁带)、光介质(例如, DVD)、或者半导体介质(例如固态存储盘 Solid State Disk (SSD))等。

权 利 要 求 书

1. 一种近内存计算加速器，其特征在于，包括：

多个控制器，用于并行获取多个内存块中的多个数据，其中，所述多个控制器中的一个控制器用于获取所述多个内存块中的一个内存块的数据；

计算模块，分别连接所述多个控制器，用于对所述多个数据进行计算；

其中，所述近内存计算加速器以及所述多个内存块位于同一块双列直插式内存模块 DIMM 上。

2. 根据权利要求 1 所述的加速器，其特征在于，所述计算包括：加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

3. 根据权利要求 1 或 2 所述的加速器，其特征在于，所述加速器还包括指令解码器，所述指令解码器分别连接所述多个控制器，

所述指令解码器用于对通用处理器 CPU 发送的指令进行解码，得到解码结果，所述解码结果用于指示待进行计算的所述多个数据的地址以及计算类型；

所述多个控制器用于根据所述多个数据的地址并行从所述多个内存块中获取所述多个数据；

所述计算模块用于根据所述计算类型对所述多个数据执行相应的计算。

4. 根据权利要求 1 或 2 或 3 所述的加速器，其特征在于，所述计算模块包括缓存单元，所述缓存单元用于缓存所述多个数据。

5. 一种双列直插式内存模块 DIMM，其特征在于，包括多个内存块以及如权利要求 1-4 任意一项所述的近内存计算加速器，其中：

所述多个内存块，用于存储多个数据，其中，所述多个内存块中的一个内存块用于存储所述多个数据中的一个数据；

所述近内存计算加速器，分别连接所述多个内存块，用于从所述多个内存块中并行获取所述多个数据，对所述多个数据进行计算。

6. 一种计算设备，其特征在于，包括：

通用处理器 CPU，用于发送近内存计算指令；

双列直插式内存模块 DIMM，包括多个内存块以及近内存计算加速器，所述近内存计算加速器用于：

接收所述近内存计算指令，所述近内存计算指令用于指示待执行近内存计算的多个数据的地址以及计算类型；

根据所述多个数据的地址从所述多个内存块中并行获取所述多个数据；

根据所述计算类型对所述多个数据执行计算。

7. 根据权利要求 6 所述的计算设备，其特征在于，

所述多个内存块被配置为允许所述近内存计算加速器访问时，禁止所述 CPU 访问。

8. 根据权利要求 7 所述的计算设备，其特征在于，所述 DIMM 还包括与所述多个内存块均不相同的一个或者多个内存块，所述一个或者多个内存块被配置为允许所述 CPU 访问时，禁止所述近内存计算加速器访问。

9. 根据权利要求 8 所述的计算设备，其特征在于，所述 DIMM 被配置为允许所述近内存计算加速器访问所述多个内存块中的一个内存块时，所述 CPU 同步访问所述一个或者多个内存块。

10. 根据权利要求 9 所述的计算设备，其特征在于，所述近内存计算加速器满足以下条件中的一个或者多个；

所述近内存计算加速器访问所述多个内存块中的一个内存块的能耗低于所述 CPU 访问所述一个或者多个内存块的能耗。

11. 一种近内存计算方法，其特征在于，所述方法包括：

接收近内存计算指令，所述近内存计算指令用于指示待执行近内存计算的多个数据的地址以及计算类型；

根据所述多个数据的地址从多个内存块中并行获取所述多个数据；

根据所述计算类型对所述多个数据执行近内存计算。

12. 根据权利要求 11 所述的近内存计算方法，其特征在于，所述近内存计算包括：加法计算、乘法计算、除法计算、查找计算以及非线性计算中的至少一种。

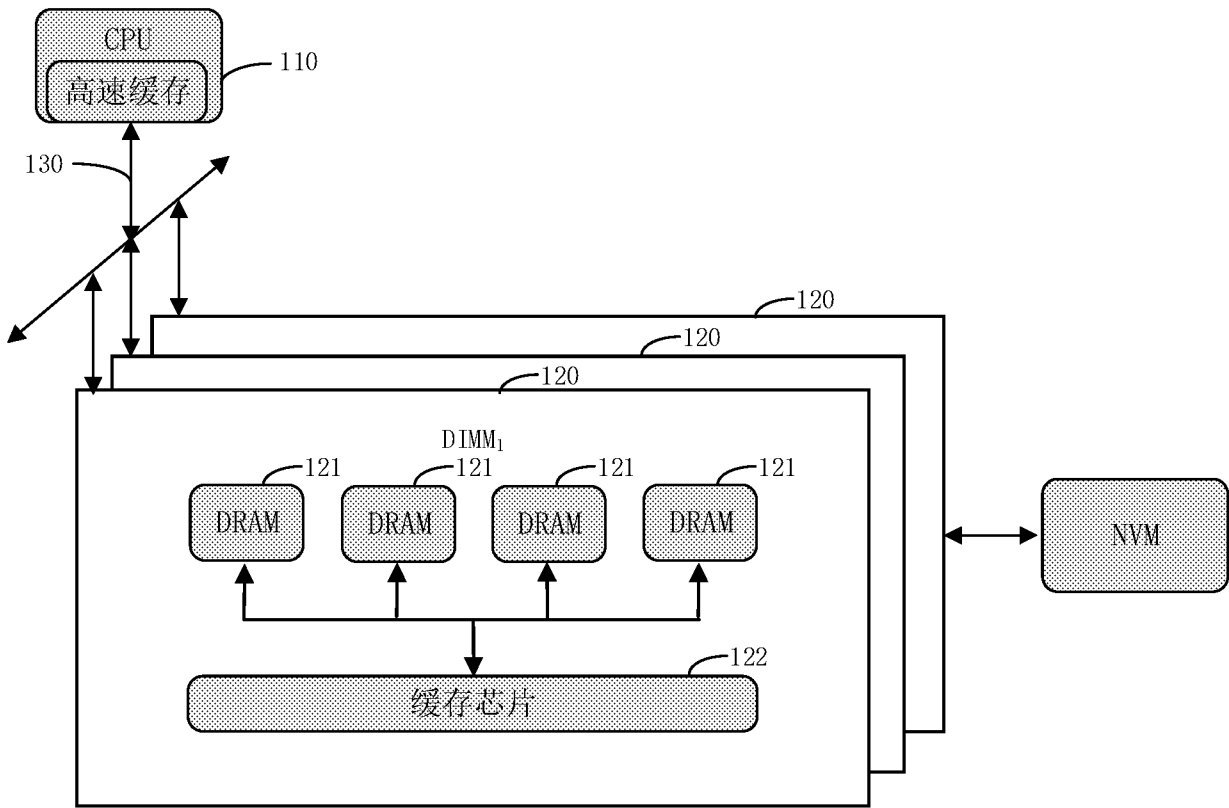


图 1

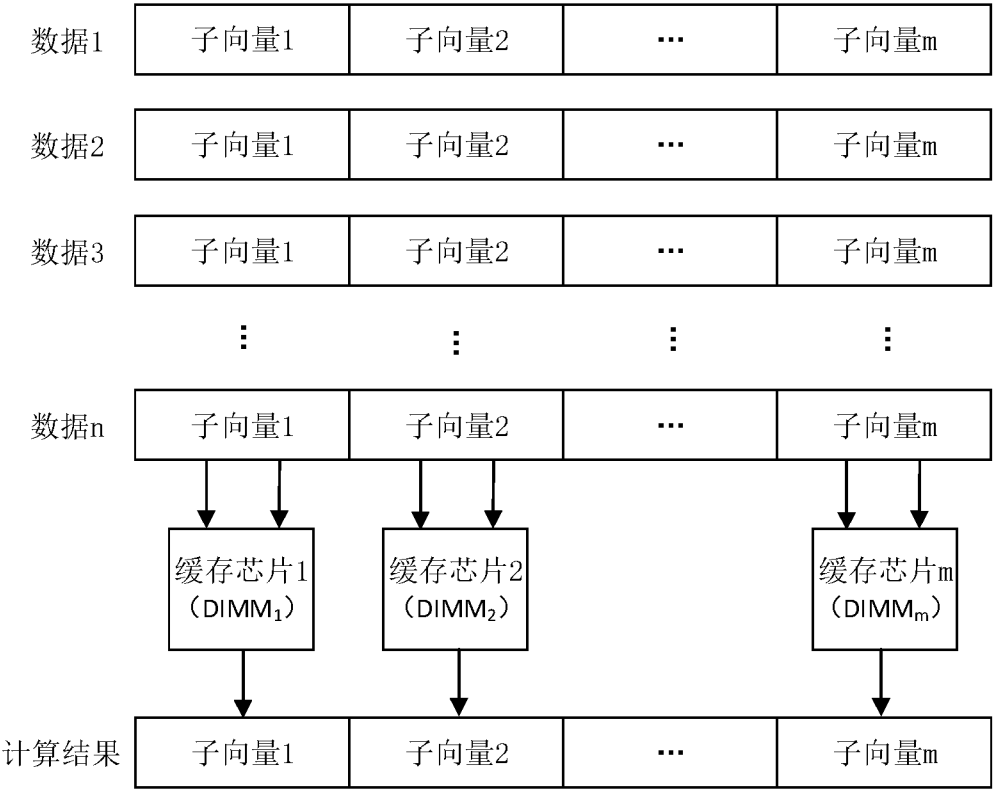


图 2

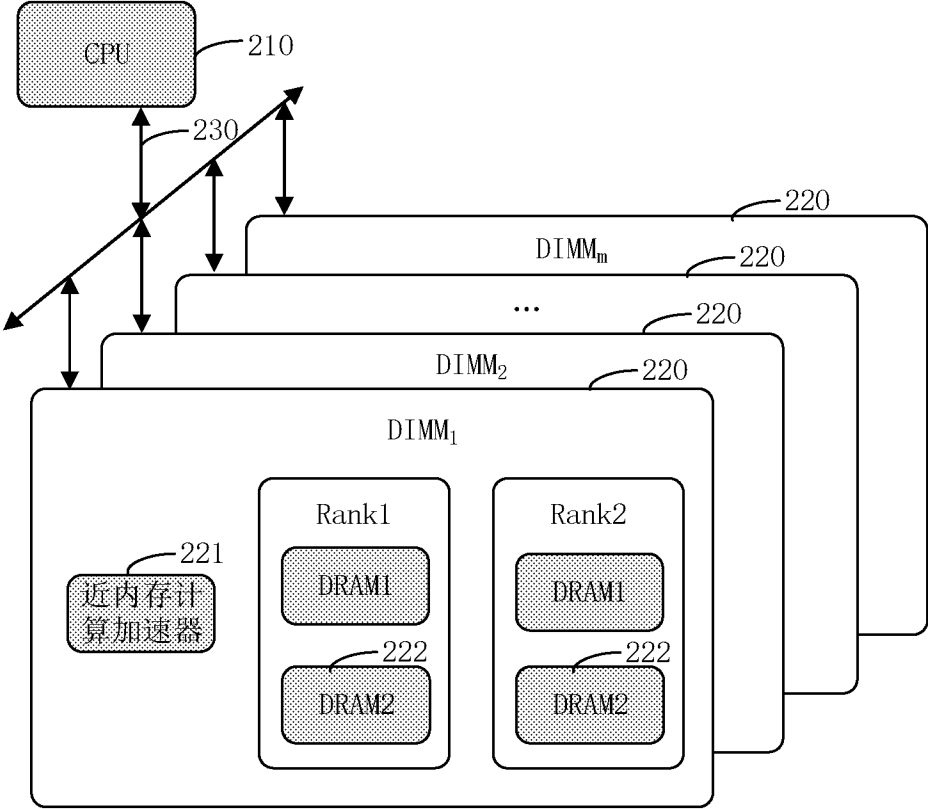


图 3

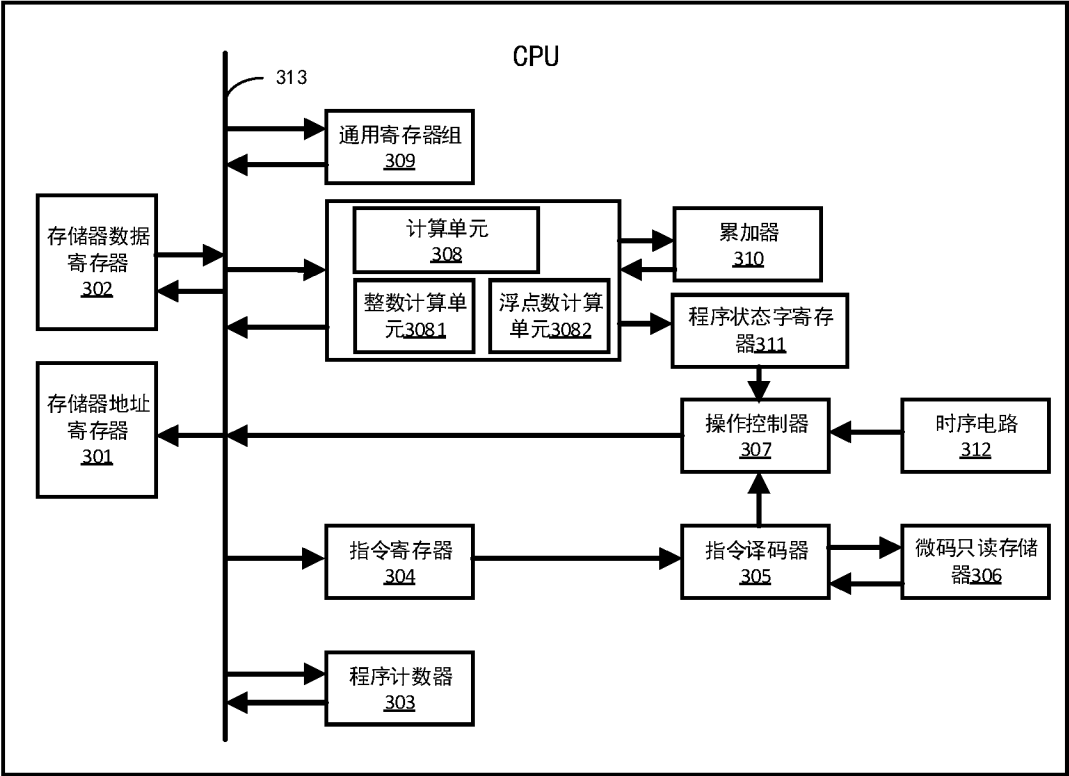


图 4

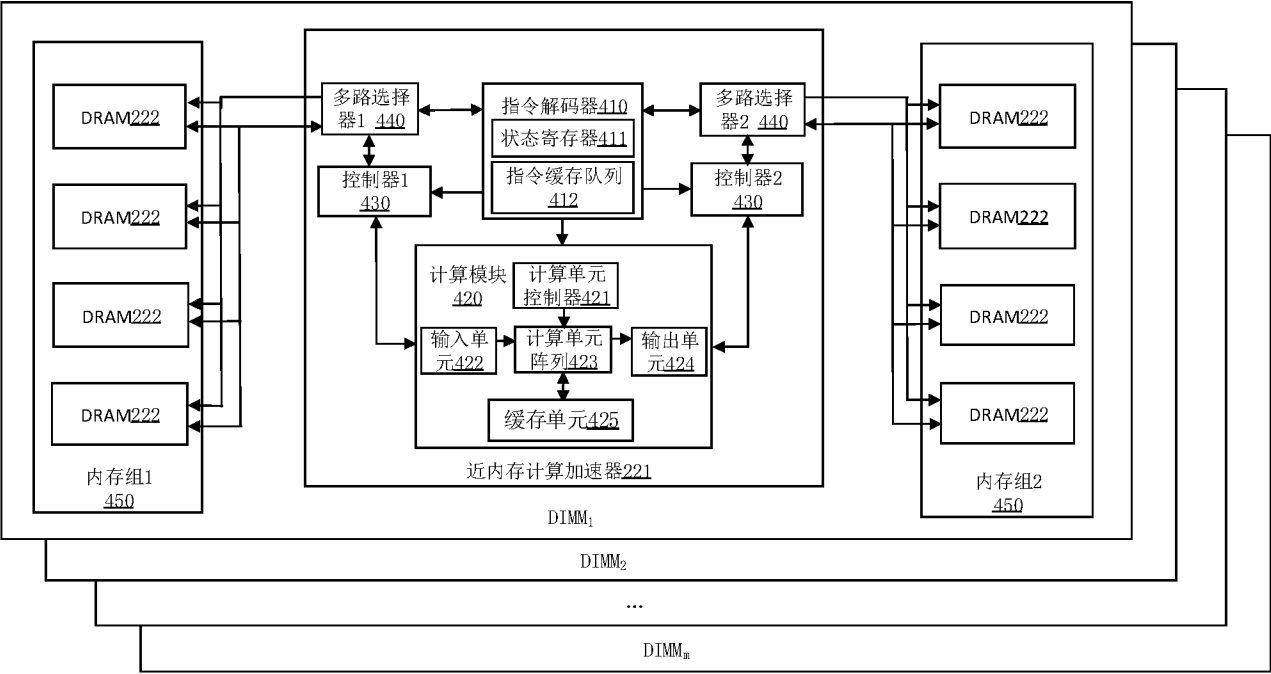


图 5

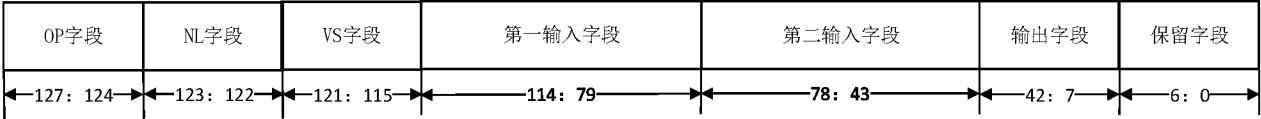


图 6

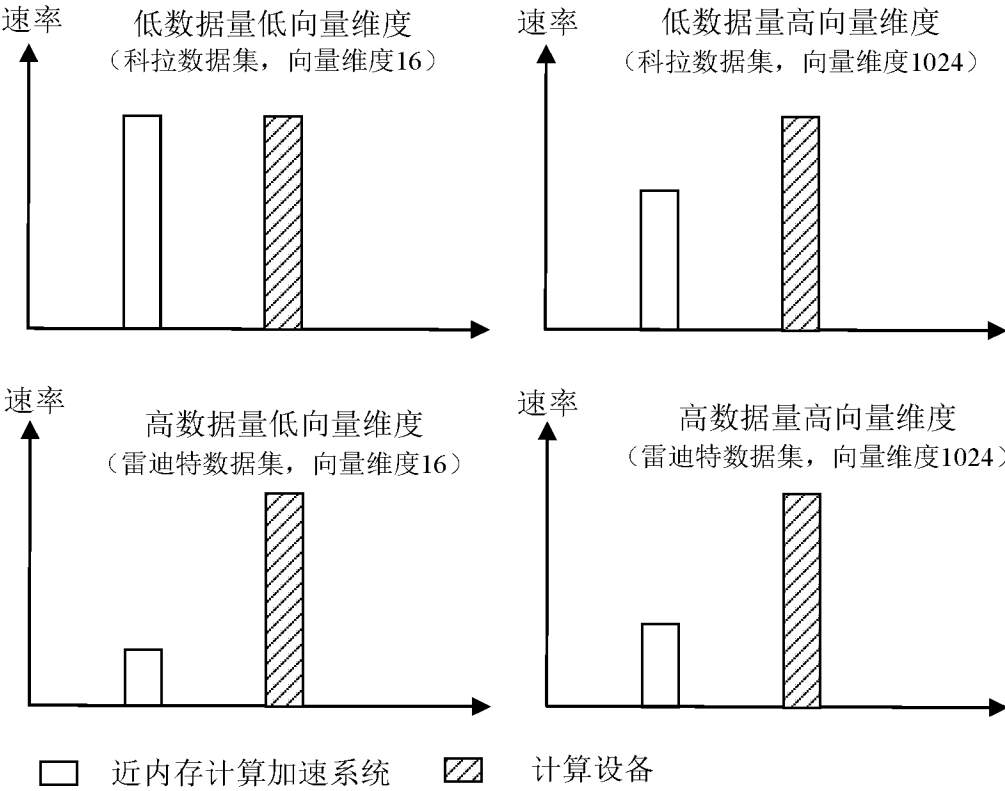


图 7

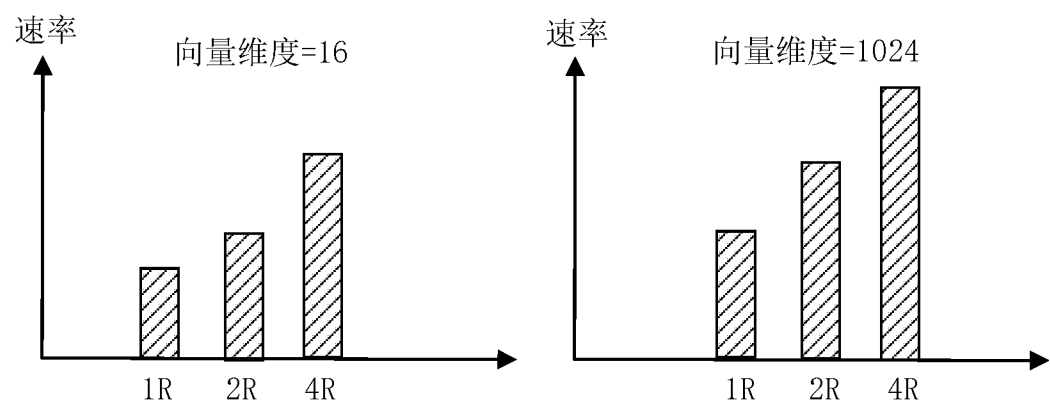


图 8

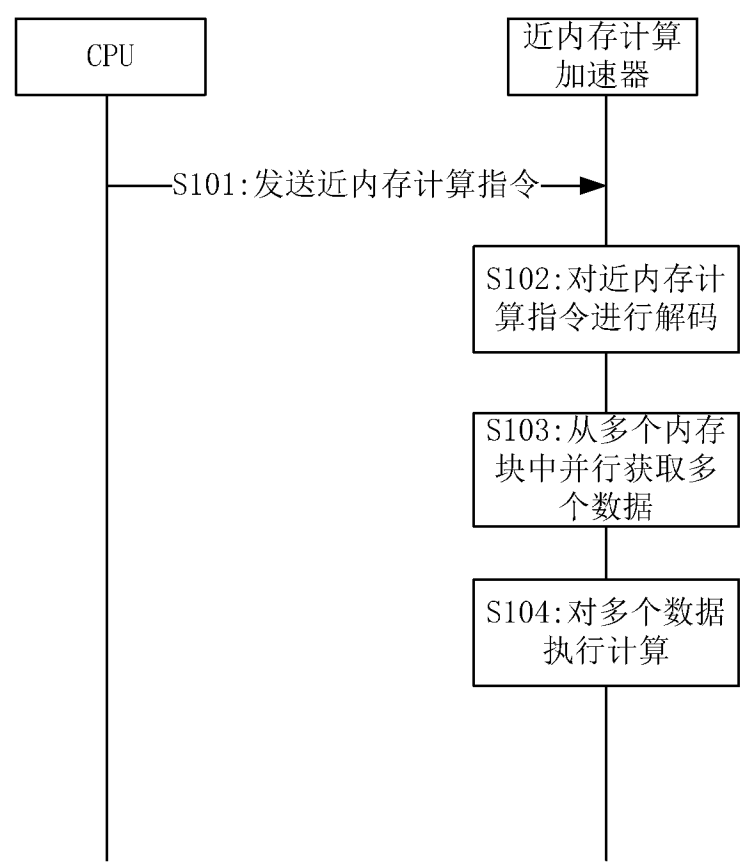


图 9

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2022/119198

A. CLASSIFICATION OF SUBJECT MATTER G06N 3/063(2006.01)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) 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) CNPAT, CNKI, WPI, EPODOC, IEEE: 近内, 加速, 控制, 计算, 内存块, 双列直插, 加法, 乘法, 除法, 查找, 非线性, 解码, 地址, 并行, 缓存, 处理器, 禁止, 能耗, near-memory, accelerat+, control+, comput+, memory w block?, dual w inline, DIMM, add+, multiplicat+, divi+, find+, search+, nonlinear+, decode+, address, parallel, cach+, processor?, CPU, inhibit, energy w consumption																	
C. DOCUMENTS CONSIDERED TO BE RELEVANT <table border="1"> <thead> <tr> <th>Category*</th> <th>Citation of document, with indication, where appropriate, of the relevant passages</th> <th>Relevant to claim No.</th> </tr> </thead> <tbody> <tr> <td>X</td> <td>US 2019108145 A1 (INTEL CORP.) 11 April 2019 (2019-04-11) description, paragraphs 0001-0085, and figures 1-7</td> <td>1-12</td> </tr> <tr> <td>A</td> <td>CN 113222130 A (GUANGDONG UNIVERSITY OF TECHNOLOGY) 06 August 2021 (2021-08-06) entire document</td> <td>1-12</td> </tr> <tr> <td>A</td> <td>CN 111124490 A (FUDAN UNIVERSITY) 08 May 2020 (2020-05-08) entire document</td> <td>1-12</td> </tr> <tr> <td>A</td> <td>CN 111860819 A (NANJING UNIVERSITY) 30 October 2020 (2020-10-30) entire document</td> <td>1-12</td> </tr> </tbody> </table>			Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.	X	US 2019108145 A1 (INTEL CORP.) 11 April 2019 (2019-04-11) description, paragraphs 0001-0085, and figures 1-7	1-12	A	CN 113222130 A (GUANGDONG UNIVERSITY OF TECHNOLOGY) 06 August 2021 (2021-08-06) entire document	1-12	A	CN 111124490 A (FUDAN UNIVERSITY) 08 May 2020 (2020-05-08) entire document	1-12	A	CN 111860819 A (NANJING UNIVERSITY) 30 October 2020 (2020-10-30) entire document	1-12
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.															
X	US 2019108145 A1 (INTEL CORP.) 11 April 2019 (2019-04-11) description, paragraphs 0001-0085, and figures 1-7	1-12															
A	CN 113222130 A (GUANGDONG UNIVERSITY OF TECHNOLOGY) 06 August 2021 (2021-08-06) entire document	1-12															
A	CN 111124490 A (FUDAN UNIVERSITY) 08 May 2020 (2020-05-08) entire document	1-12															
A	CN 111860819 A (NANJING UNIVERSITY) 30 October 2020 (2020-10-30) entire document	1-12															
☐ 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 “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 28 November 2022		Date of mailing of the international search report 13 December 2022															
Name and mailing address of the ISA/CN China National Intellectual Property Administration (ISA/CN) No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088, China Facsimile No. (86-10)62019451		Authorized officer Telephone No.															

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2022/119198

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	2019108145	A1	11 April 2019	CN	110851378	A	28 February 2020
				DE	102019122363	A1	20 February 2020
CN	113222130	A	06 August 2021	None			
CN	111124490	A	08 May 2020	None			
CN	111860819	A	30 October 2020	None			

国际检索报告

国际申请号

PCT/CN2022/119198

A. 主题的分类 G06N 3/063 (2006.01) 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) 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)) CNPAT, CNKI, WPI, EPDOC, IEEE: 近内, 加速, 控制, 计算, 内存块, 双列直插, 加法, 乘法, 除法, 查找, 非线性, 解码, 地址, 并行, 缓存, 处理器, 禁止, 能耗, near-memory, accelerat+, control+, comput+, memory w block?, dual w inline, DIMM, add+, multiplicat+, divi+, find+, search+, nonlinear+, decode+, address, parallel, cach+, processor?, CPU, inhibit, energy w consumption		
C. 相关文件		
类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求
X	US 2019108145 A1 (INTEL CORPORATION) 2019年4月11日 (2019 - 04 - 11) 说明书第0001-0085段, 图1-7	1-12
A	CN 113222130 A (广东工业大学) 2021年8月6日 (2021 - 08 - 06) 全文	1-12
A	CN 111124490 A (复旦大学) 2020年5月8日 (2020 - 05 - 08) 全文	1-12
A	CN 111860819 A (南京大学) 2020年10月30日 (2020 - 10 - 30) 全文	1-12
☐ 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 “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 2022年11月28日		国际检索报告邮寄日期 2022年12月13日
ISA/CN的名称和邮寄地址 中国国家知识产权局 (ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451		受权官员 霍玉明 电话号码 010-53961327

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2022/119198

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)
US	2019108145	A1	2019年4月11日	CN 110851378 A	2020年2月28日
				DE 102019122363 A1	2020年2月20日
CN	113222130	A	2021年8月6日	无	
CN	111124490	A	2020年5月8日	无	
CN	111860819	A	2020年10月30日	无	