

(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
2019 年 3 月 7 日 (07.03.2019)



(10) 国际公布号
WO 2019/042312 A1

(51) 国际专利分类号:
G06F 9/48 (2006.01)

(21) 国际申请号: PCT/CN2018/102919

(22) 国际申请日: 2018 年 8 月 29 日 (29.08.2018)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201710769632.8 2017 年 8 月 31 日 (31.08.2017) 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)。

(72) 发明人: 林健 (LIN, Jian); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 夏命榛 (XIA, Mingzhen); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (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, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, 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,

(54) Title: DISTRIBUTED COMPUTING SYSTEM, DATA TRANSMISSION METHOD AND DEVICE IN DISTRIBUTED COMPUTING SYSTEM

(54) 发明名称: 分布式计算系统, 分布式计算系统中数据传输方法和装置

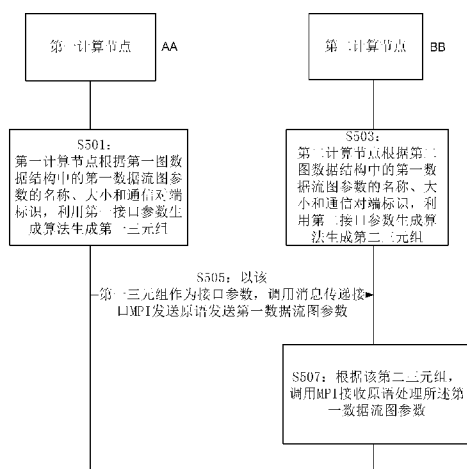


图 5

S501 According to name, size and communication peer identifier of first data flow diagram parameter in first diagram data structure, first computing node utilizes first interface parameter generation algorithm to generate first triplet

S503 According to name, size and communication peer identifier of first data flow diagram parameter in second diagram data structure, second computing node utilizes second interface parameter generation algorithm to generate second triplet

S505 Using first triplet as interface parameter, call messaging interface MPI sends primitive to send first data flow diagram parameter

S507 According to second triplet, call MPI receiving primitive to process first data flow diagram parameter

AA First computing node

BB Second computing node

(57) Abstract: A distributed computing system is disclosed. A first computing node and a second computing node in the distributed computing system both have stored therein information relating to a name, size, and communication peer identifier of a first data flow diagram parameter in a data flow diagram, and the first data flow diagram parameter is stored in the first computing node. The first computing node and the second computing node use the same interface parameter saved in the respective nodes thereof to generate algorithms and the above information of the first data flow diagram parameter, thereby generating respective triplets thereof that are used as interface parameters of a messaging interface MPI primitive for transmitting the first data flow diagram parameter between the first computing node and the second computing node. In this way, the first computing node and the second computing node can transmit the data flow diagram parameter using the MPI primitive without negotiation, thereby improving data communication efficiency between computing nodes and improving computing efficiency of the distributed computing system with respect to the data flow diagram.

(57) 摘要: 一种分布式计算系统, 该分布式计算系统中的第一计算节点和第二计算节点都保存有数据流图中第一数据流图参数的名称、大小和通信对端标识的信息, 第一计算节点中保存有该第一数据流图参数, 第一计算节点和第二计算节点使用各自节点中存储的相同的接口参数生成算法和第一数据流图参数的上述信息, 生成各自的三元组, 该三元组被用作第一计算节点和第二计算节点间用于传输第一数据流图参数的消息传递接口MPI原语的接口参数。这样, 第一计算节点与第二计算节点就可以无需协商地使用MPI原语传输数据流图参数, 能够提高计算节点间数据的通信效率, 从而提升分布式计算系统对数据流图的计算效率。

SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG,
US, UZ, VC, VN, 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:

- 包括国际检索报告(条约第21条(3))。

分布式计算系统，分布式计算系统中数据传输方法和装置

技术领域

本发明涉及计算机领域，尤其涉及分布式计算系统，分布式计算系统中的数据传输方法和装置。

5 背景技术

随着大数据产业和人工智能技术的发展，各类计算平台不断涌现，例如机器学习平台，图计算平台和流计算平台等等。这些计算平台常常被部署在分布式计算系统中以执行大数据计算。例如，机器学习平台通常以数据流图为计算对象，通过将数据流图划分为多个子图或者副本，以及将该多个子图或者副本部署在分布式计算系统中的多个计算节点上，进而可以使用该多个计算节点对数据流图进行协同计算，以提升数据的计算效率。其中，该多个计算节点中的计算节点可以包括多个用于计算的设备，如 CPU（Center Processing Unit，中央处理器）以及在该计算节点的主机中配备的加速硬件，例如加速硬件可以是 GPU（Graphics Processing Unit，图形处理器）。

而数据流图在分布式计算系统的计算过程中，需要进行节点间的数据流图参数的通信，这种通信直接影响机器学习平台对数据流图的计算效率。一种现有的方案中，将通常应用于高性能计算领域的消息传递接口（Message Passing Interface，MPI）库技术作为一个外接的插件引入分布式计算系统，以支持该系统中的数据通信，然而，在 MPI 库技术在进行数据通信前，需要进行数据通信的双方通过信息交互以得到通信对端的信息，从而协商出用于 MPI 原语的通信参数，而计算平台的通信时序的动态性和随机性使得数据通信的双方难以及时确认对端以及进行协商，该协商过程也增加了计算平台中数据通信的负担，从而影响数据的传输效率。

20

发明内容

本发明的实施例提供一种分布式计算系统中的数据传输方法、装置以及一种分布式计算系统，能简化 MPI 技术应用在计算数据流图中的通信过程，无需在数据传输前与通信对端协商对端信息，使得 MPI 技术能够更好地与分布式部署的计算平台适配，从而提高分布式计算系统中数据传输的效率，从而提升对分布式计算系统数据流图的计算效率。

25

为达到上述目的，本发明的实施例采用如下技术方案：

第一方面，本申请提供一种分布式计算系统，该分布式计算系统包括第一计算节点和第二计算节点，其特征在于，所述第一计算节点上的第一图数据结构保存有第一数据流图中的第一数据流图参数的名称、大小和通信对端标识，其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数；所述第二计算节点上的第二图数据结构保存有第二数据流图中的所述第一数据流图参数的名称、大小和通信对端标识；所述第一数据流图中的第一数据流图参数的通信对端标识对应于所述第二计算节点，所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点；所述第一计算节点用于，根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组，所述第一三元组包括消息标记、消息大小和目的进程序列号，其中，所述消息标记对应于所述第一数据流图参数的名称，所述消息大小对应于所述第一数据流图参数的大小，所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程；所述第二计算节点用于，根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第二接口参数生成算法生成第二三

30

35

元组, 所述第二接口参数生成算法与所述第一接口参数生成算法相同, 所述第二三元组包括所述消息标记、所述消息大小和源进程序列号, 其中, 所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程; 所述第一计算节点用于, 以所述第一三元组作为接口参数, 调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数; 所述第二计算节点用于, 根据所述第二三元组, 调用 MPI 接收原语处理所述第一数据流图参数。

这样, 以第一三元组为接口参数的 MPI 发送原语与以第二三元组为接口参数的 MPI 接收原语相对应, 这样, 在第一图数据结构和第二图数据结构中, 包括有通信对端标识, 解决了通信对端的进程在数据流图运行过程中不可知的问题, 以及, 需要传输该第一数据流图参数的通信双方, 使用各自计算节点存储的数据流图中的信息和相同的接口参数生成算法生成三元组, 就无需向对端交互自己的信息, 以及无需协商生成三元组的算法, 该方法能够在数据发送方和接收方内独立运行, 在双方无交互的情况下生成对应的三元组, 简化了使用 MPI 原语通信的过程, 能够提高分布式计算平台中数据传输的效率。

应理解, 根据所述第二三元组, 调用 MPI 接收原语处理所述第一数据流图参数, 可以达到使得第二计算节点中的进程能使用该第一数据流图参数进行数据流图的计算的目的。调用 MPI 接收原语处理所述第一数据流图参数, “处理”在不同的场景下, 可以对应不同的操作, 本申请不做限定。例如可以是以下操作中的一种或多种: 调用 MPI 接收原语将该第一数据流图参数接收到主机内存的数据缓冲区; 调用 MPI 接收原语修改该第一数据流图参数的标记, 以将已在主机内存中的第一数据流图参数提供给计算数据流图的进程使用; 将该第一数据流图参数从数据缓冲区存储到目的地址。

第一数据流图参数的名称(name)用于标识该第一数据流图参数, 可以是该第一图数据结构中的一个字段也可以是分散在该第一图数据结构中的信息。该第一数据流图参数的大小(size)用于表示该第一数据流图参数所占的存储空间, 也就是该数据流图参数的数据量。

该第一数据流图中的第一数据流图参数的通信对端标识可以是该第二计算节点的标识; 或者该第一数据流图参数的目的地址所在的存储设备的标识, 该存储设备位于第二计算节点; 或者该第一数据流图参数对应的目的进程的标识, 该目的进程位于第二计算节点; 或者其他用于指示该第一数据流图参数的接收端的信息。

对于该第二数据流图中的第一数据流图参数的通信对端标识, 类似的, 可以是该第一计算节点的标识; 或者该第一数据流图参数的源地址所在的存储设备的标识, 该存储设备位于第一计算节点; 或者第一计算节点汇总发送该第一数据流图参数的进程的标识; 或者其他用于指示该第一数据流图参数的发送端的信息。

第一计算节点上的第一图数据结构保存有第一数据流图中的第一数据流图参数的名称、大小和通信对端标识, 可以在该第一图数据结构中包括承载该三种信息的字段, 也可以是保存有能够得到该第一数据流图参数的名称、大小或者通信对端标识的信息。即所谓“保存有”可以是能够直接从该第一图数据结构中读取, 也可以是根据该第一图数据结构中的信息分析获得。

第二数据流图保存在第二计算节点上, 第二数据流图可以是第一数据流图的副本, 也可以与第一数据流图都是一个数据流图的两个子图。

其中, 消息标记用于表示该 MPI 发送原语发送的数据。消息大小用于表示该 MPI 发送原语发送的信息的大小。源进程序列号是第一计算节点执行该 MPI 发送原语的进程的序列号, 目的进程序列号是在第二计算节点执行与该 MPI 发送原语相对应的 MPI 接收原语的进程的序列号。三元组

的概念只用于表示其中的三个参数，而不限定这三个参数间的顺序。该三元组中的三个参数的格式满足MPI发送原语所携带的接口函数参数的格式要求。另外，MPI发送原语的接口参数包括但不限于该第一三元组，MPI接收原语的接口参数包括但不限于该第二三元组。

一种实现方式下，以所述第一三元组作为接口参数，调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数的方面第一数据流图参数，所述第一计算节点用于以所述第一三元组作为接口参数，通过消息传递接口 MPI 发送原语从所述第一计算节点的主机内存中读取所述第一数据流图参数，以向所述第二计算节点发送所述第一数据流图参数。

这样，MPI 发送原语直接从主机内存中读取该第一数据流图参数，能提高读取数据的效率。

一种实现方式下，所述第一计算节点还保存有所述第一数据流图参数所在的存储设备的信息，第一数据流图参数所述第一计算节点还用于在所述存储设备的信息指示为其他存储设备的情况下，将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存，所述其他存储设备为所述第一计算节点中除主机内存外的内存。

该存储设备的信息可以是该存储设备的标识，也可以是用于表示该存储设备的编号，可根据标识或编号确定出该存储设备的存储类型，还可以是标识该存储设备的类型的信息等，或者其他形式的能达到上述功能的信息。

这样第一计算节点在使用 MPI 发送原语前，就会将第一数据流图参数准备在第一计算节点的主机内存中，而 MPI 发送原语只从该第一计算节点的主机内存中读取该第一数据流图参数，无需与计算平台争抢读取其他存储设备的资源，提高了 MPI 发送原语的执行效率。

一种实现方式下，所述第一接口参数生成算法包括第一算法、第二算法和第三算法，在根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组第一数据流图参数的方面，所述第一计算节点用于：根据所述第一图数据结构中的第一数据流图参数的名称第一数据流图参数和所述第一算法，确定所述第一三元组中的消息标记，根据所述第一图数据结构中的第一数据流图参数第一数据流图参数的大小和所述第二算法，确定所述第一三元组中的消息大小，以及根据第一图数据结构中的第一数据流图参数的通信对端标识和所述第三算法，确定所述第一三元组中的目的进程序列号；相应的，在根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第二接口参数生成算法生成第二三元组第一数据流图参数的方面，所述第二计算节点用于：根据所述第二图数据结构中的第一数据流图参数的名称和所述第二接口参数生成算法中的第一算法，确定所述第二三元组中的消息标记；根据所述第二图数据结构中的第一数据流图参数的大小和所述第二接口参数生成算法中的第二算法，确定所述第二三元组中的消息大小；以及根据第二图数据结构中的第一数据流图参数的通信对端标识和所述第二接口参数生成算法中的第三算法，确定所述第二三元组中的源进程序列号。

则上文中描述的第一接口参数生成算法与第二接口参数生成算法相同，是指，第一接口参数生成算法包括第一算法、第二算法和第三算法，且，第二接口参数生成算法包括与该第一接口参数生成算法相同或者相应的第一算法、第二算法和第三算法。

第一算法可以是任意二进制长度值转化为固定二进制长度值的算法，例如可以是哈希算法等；或者其他能将第一数据流图参数的名称转化为符合 MPI 原语的接口参数中消息标记的格式的算法。对第二算法，则一种实现方式下，可使该消息大小字段的值等于上述数据流图参数的大小

的参数值,即 size。另一种实现方式下,可以使该消息大小字段的值等于上述的数据流图参数的大的参数值加上一个值。第三算法是进程序列号与通信对端标识之间的映射关系,其中第一计算节点中包括目的进程序列号与通信对端标识之间的映射关系,第二计算节点中包括源进程序列号与通信对端标识之间的映射关系。第三算法可以是函数关系,也可以是计算节点中维护的映射表,本申请不做限制。

一种实现方式下,在根据所述第二三元组,调用 MPI 接收原语处理所述第一数据流图参数的方面第一数据流图参数,所述第二计算节点用于通过 MPI 探测原语检测所述第二计算节点主机内存中的数据缓存区,以获得所述第一数据流图参数的第二三元组,所述数据缓存区专用于存储由 MPI 原语处理的数据;调用 MPI 接收原语,以处理所述第一数据流图参数,所述 MPI 接收原语的接口参数包括所述第二三元组。

这样,就使得数据能更及时地被接收原语处理,也使得第一计算节点其他待执行的发送原语能更快地执行,从而,提升了数据传输的效率。以及通过设置专用的数据缓存区和轮询线程,使得消息传递通信缓冲区能够在消息接收原语尚未调用、消息最终目的地址未知的情况下,使得消息发送原语能够执行数据发送,并在数据发送完成之后立即返回。缓冲区为未来的消息接收原语临时保存数据,从而使得消息发送原语无须与消息接收原语进行同步操作,解决了二者固有的时序约束。发送方无须同步等待,使其节约了执行时间,有助于提升性能。

一种实现方式下,所述第一数据流图参数接收原语携带所述第一数据流图参数的目的地址,在通过第一数据流图参数接收原语处理所述第一数据流图参数的方面,所述第二计算节点用于将所述第二三元组作为所述 MPI 接收原语的接口参数调用所述 MPI 接收原语,以将所述第一数据流图参数从所述数据缓存区存储到所述目的地址。例如,该目的地址位于主机内存中的用户内存空间

第二方面,本发明实施例记载一种分布式计算系统中的数据传输方法,所述分布式计算系统包括第一计算节点和第二计算节点,所述方法包括:从所述第一计算节点中的第一图数据结构中,确定第一数据流图的第一数据流图参数的名称、大小和通信对端标识,其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数,所述通信对端标识对应所述第二计算节点;根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第一接口参数生成算法生成第一三元组,所述第一三元组包括消息标记、消息大小和目的进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程;以所述第一三元组作为接口参数,调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数,以便于所述第二计算节点以与所述第一三元组对应的第二三元组作为接口参数,调用 MPI 接收原语处理所述第一数据流图参数,所述第二三元组是根据所述第二计算节点中的第二图数据结构,利用第二接口参数生成算法生成的,所述第二接口参数生成算法与所述第一接口生成算法相同。

一种实现方式下,在以所述第一三元组作为接口参数,调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数的方面,所述方法包括:以所述第一三元组作为接口参数,通过消息传递接口 MPI 发送原语从所述第一计算节点的主机内存中读取所述第一数据流图参数,以向所述第二计算节点发送所述第一数据流图参数。

一种实现方式下,所述第一计算节点还保存有所述第一数据流图参数所在的存储设备的信

息,所述方法还包括:在所述存储设备的信息指示为其他存储设备的情况下,将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存,所述其他存储设备为所述第一计算节点中除主机内存外的内存。

第三方面,本申请记载一种分布式计算系统中的数据传输装置,所述分布式计算系统包括第一计算节点和第二计算节点,所述数据传输装置位于所述第一计算节点,所述数据传输装置包括:确定模块,所述确定模块用于从所述第一计算节点中的第一图数据结构中,确定第一数据流图的第一数据流图参数的名称、大小和通信对端标识,其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数,所述通信对端标识对应所述第二计算节点;生成模块,所述生成模块用于根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第一接口参数生成算法生成第一三元组,所述第一三元组包括消息标记、消息大小和目的进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程;通信模块,所述通信模块用于以所述第一三元组作为接口参数,调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数,以便于所述第二计算节点以与所述第一三元组对应的第二三元组作为接口参数,调用 MPI 接收原语处理所述第一数据流图参数,所述第二三元组是根据所述第二计算节点中的第二图数据结构,利用第二接口参数生成算法生成的,所述第二接口参数生成算法与所述第一接口生成算法相同。

第四方面,本申请记载一种物理机,所述物理机包括:至少一个处理器和存储可执行代码的非瞬态计算机可读介质以运行分布式计算系统中的第一计算节点,所述分布式计算系统包括所述第一计算节点和第二计算节点;所述可执行代码在被所述至少一个处理器中的处理器执行时被配置为执行上述的系统中第一计算节点执行的任一种方法。

可见,第四方面和第三方面是第二方面的方法对应的装置,第二方面的方法执行于第一计算节点,一些情况下,该第一计算节点为第一方面的系统中的第一计算节点。关于第二方面、第三方面以及第四方面中各步骤的说明,名词的解释、各种实现方式以及有益效果说明,第一方面的系统中有关第一计算节点的论述同样适用,可参见第一方面中的相关内容,此处不再赘述。

第五方面,本申请记载一种分布式计算系统中的数据传输方法,所述分布式计算系统包括第一计算节点和第二计算节点,所述方法包括:从所述第二计算节点的第二图数据结构中,确定第二数据流图中的第一数据流图参数的名称、大小和通信对端标识,所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点;根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第二接口参数生成算法生成第二三元组,所述第二三元组包括消息标记、消息大小和源进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程;根据所述第二三元组,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数,所述第一数据流图参数是所述第一计算节点通过 MPI 发送原语发送来的,所述 MPI 发送原语的接口参数包括与所述第二三元组对应的第一三元组,所述第一三元组是所述第一计算节点根据所述第一计算节点中的第一图数据结构,利用第一接口参数生成算法生成的,所述第二接口参数生成算法与所述第一接口生成算法相同。

一种实现方式下,第二计算节点以所述第二三元组作为接口参数,调用消息传递接口 MPI

接收原语接收该第一数据流图参数,以使得该第二计算节点使用该第一数据流图参数进行数据流图的计算。

一种实现方式下,所述第二计算节点运行有第一线程和第二线程,所述第二计算节点的主机内存中包括数据缓存区,所述数据缓存区专用于存储被 MPI 原语处理的数据,在以所述第二三元组作为接口参数,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数的方面,所述方法包括:所述第一线程通过消息传递接口 MPI 探测原语检测所述主机内存中的数据缓存区,以获得第二三元组;所述第一线程根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语,以处理所述第一数据流图参数,所述数据缓存区中的第二三元组是所述第二计算节点根据所述 MPI 发送原语得到的;所述第二线程在确定所述第一数据流图参数被所述第一 MPI 接收原语处理后,将第二 MPI 接收原语修改为 MPI 等待原语,所述第二 MPI 接收原语为未被所述第二线程执行的与所述第一数据流图参数对应的接收原语,所述第二 MPI 接收原语的接口参数包括所述第二计算节点生成的第二三元组,所述 MPI 等待原语用于等待所述第一 MPI 接收原语执行完毕。

第二三元组可以是根据接收到的 MPI 发送原语的接口参数获得,也可以是根据接口参数以及该 MPI 发送原语传输的数据中分析得到,本申请不做限制。

也就是说第二计算节点可以在主机内存中再专门可具体可以再拉起一个线程(可称为轮询线程)执行 MPI 探测原语,以检测该第二计算节点的主机内存的缓冲区,缓冲区中包括上述的数据缓存区,这样就可以找到未来得及被 MPI 接原语处理的数据。

在一种实现方式下,在所述第一线程根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语,以处理所述第一数据流图参数的方面,所述方法包括:在所述第一数据流图参数的目的地地址对应所述第二计算节点的主机内存中分配给用户使用的内存空间的情况下,所述第一线程以所述数据缓存区中的第二三元组为所述第一 MPI 接收原语的接口参数,调用所述第一 MPI 接收原语,以将所述第一数据流图参数从所述数据缓存区存储到所述第一数据流图参数的目的地地址。

在一种实现方式下,在所述第一数据流图参数的目的地地址对应其他存储设备的情况下,所述第二计算节点将所述主机内存中的第一数据流图参数存储到所述目的地地址,所述其他存储设备为所述第二计算节点中除主机内存之外的内存。

第六方面,本申请还记载一种分布式计算系统中的数据传输装置,所述分布式计算系统包括第一计算节点和第二计算节点,所述数据传输装置位于所述第二计算节点,所述数据传输装置包括:确定模块,所述确定模块用于从所述第二计算节点的第二图数据结构中,确定第二数据流图中的第一数据流图参数的名称、大小和通信对端标识,所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点;生成模块,所述生成模块用于根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第二接口参数生成算法生成第二三元组,所述第二三元组包括消息标记、消息大小和源进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程;通信模块,所述通信模块用于根据所述第二三元组,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数,所述第一数据流图参数是所述第一计算节点通过 MPI 发送原语发送来的,所述 MPI 发送原语的接口参数包括与所述第二三元组对应的第一三元组,所述第一三元组是所述第一计算节点根据所述第一计算节点中的第一图数据结构,利用第一接口参数生成算法生成的,所

述第二接口参数生成算法与所述第一接口生成算法相同。

第七方面，本申请还记载一种物理机，所述物理机包括：至少一个处理器和存储可执行代码的非瞬态计算机可读介质以运行分布式计算系统中的第二计算节点，所述分布式计算系统包括第一计算节点和所述第二计算节点；所述可执行代码在被所述至少一个处理器中的处理器执行时被配置执行前述的任一种由第二计算节点执行的方法。

可见，第六方面和第七方面是第五方面的方法对应的装置，第五方面的方法执行于第二计算节点，一些情况下，该第二计算节点为第一方面的系统中的第二计算节点。关于第五方面、第六方面以及第七方面中各步骤的说明，名词的解释、各种实现方式以及有益效果说明，第一方面的系统中有关第二计算节点的论述同样适用，可参见第一方面中的相关内容，此处不再赘述。

第八方面，本申请记载一种存储有可执行程序的非瞬态计算机可读介质，所述可执行程序用于执行上述的系统中第一计算节点或者第二计算节点执行的任一种方法。关于第八方面中涉及各步骤的说明，名词的解释、各种实现方式以及有益效果说明，前述有关的论述同样适用，可参见前述的相关内容，此处不再赘述。

附图说明

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一些实施例。

图1为本发明实施例提供的一个数据流图的示意图。

图2为本发明实施例提供的一种计算平台示意图；

图3为本发明实施例提供的一种分布式计算平台的网络架构示意图；

图4为本发明实施例提供的一种使用MPI技术进行进程间通信的方法的时序图；

图5为本发明实施例提供的一种数据传输方法的示意图；

图6为本发明实施例提供的一种切分数据流图的示意图；

图7为本发明实施例提供的一种TensorFlow机器学习平台的架构的示意图；

图8为本发明实施例提供的一种数据传输装置的示意图

图9为本发明实施例提供的一种用于执行本申请的方法的物理机的示意图。

具体实施方式

本文中字符“/”，一般表示前后关联对象是一种“或者”的关系。例如，A/B可以理解为A或者B，则和/或可以理解为和或者或。

本发明的说明书和权利要求书中的术语“第一”和“第二”等不是用于描述对象的特定顺序，而是用于区别不同的对象，有特定说明的情况下，“第一”和“第二”可描述相同的对象。例如，无特定说明的情况下，第一进程和第二进程是不同的进程。

在本发明的描述中，除非另有说明，“多个”的含义是指两个或两个以上。例如，多个设备是指两个或两个以上的设备。

此外，本发明的描述中所提到的术语“包括”和“具有”以及它们的任何变形，意图在于覆盖不排他的包含。例如包含了一系列步骤或单元的过程、方法、系统、产品或设备没有局限于已列出的步骤或单元，而是可选地还包括其他没有列出的步骤或单元，或可选地还包括对于这些过程、方法、产品或设备固有的其它步骤或单元。

下面描述本申请涉及的一些名词。

消息传递：计算机系统中，进程间或软件组件间的一类数据通信方法的统称。它将待通信的数据抽象并封装为“消息”，参与通信的双方或多方通过调用消息发送、接收等原语实现消息在进程或组件之间的传递，从而完成数据通信。

5 原语：指由若干条指令组成的，用于完成一个功能或者一个过程的一段代码。原语的执行应当是连续的。

10 计算节点可以是物理机，也可以是运行在物理机上的主机（host），虚拟机（Virture Machine）或者容器。应理解，虚拟机和容器都需要部署在物理机上。也就是说，下文中所描述的第一计算节点和第二计算节点，有可能是相同的物理机或者不同的物理机。例如，计算节点是部署在相同物理机上的虚拟机或者容器。当然，下文实施例中所描述的发送端物理机和接收端物理机，也有可能是相同的物理机或者不同的物理机。

应注意，本申请文件中，计算节点和数据流图中包括的节点是不同性质的名词，具有不同的语义。

15 设备：指物理机中的硬件，例如该硬件能支撑该物理机上的虚拟机或者容器或者进程或者线程的运行。例如可以是计算设备或者存储设备。计算设备指物理机中用于计算的硬件，计算设备可以是CPU，也可以是GPU、FPGA（Field - Programmable Gate Array，现场可编程门阵列）、MIC(Many Intergrated Core，成众核)或者其他具有计算能力的硬件设备。存储设备指物理机中可存储数据或者代码的硬件，如供上述计算设备使用的内存，例如主机内存（又称CPU内存），GPU内存、FPGA内存等各种内存，再例如外存，如硬盘，光盘等。

20 主机：指计算机硬件系统中用于放置主板及其他主要部件的容器（Mainframe）。比如可以包括CPU、内存、硬盘、电源、以及其他输入输出接口。例如该输入输出接口可以是USB 控制器、显卡、网卡、声卡中的任一种。

25 数据流图：一种通过表达数据在计算逻辑中的流动方向和计算关系，从而反映计算逻辑的设计原理及实现流程的图式数据结构。本申请以常见的计算数据流图的机器学习平台为例进行描述。需要说明的是，在机器学习平台，数据流图会在计算前预加载到该平台，该预加载过程包括对数据流图中包括的节点、边以及边上的参数的定义。

在机器学习平台中，算法的计算逻辑通常使用数据流图进行形式化表达。使用这机器学习平台对数据流图进行计算，需要先用代码描述该数据流图，这个过程可以称为数据流图的定义，该数据流图定义好后，将这部分代码进行编译，计算该数据流图时，按照编译后的代码进行读取和执行，而不是按照定义数据流图时的编码顺序执行。

30 数据流图是一种有向无环图（Directed Acyclic Graph），它由若干节点以及节点之间的连接边（简称“边，edge”）组成，也就是说边是从一个节点指向另一个节点的。

35 节点和边，可以从数据流图定义时和运行时两个层面解释。在数据流图定义时，节点代表计算过程中所使用的运算符或变量。其中，运算符就是用于表达运算规则的符号，例如加（+）、减（-）、乘（×）、除（÷）、积分（∫）、微分、指数、对数（log或者ln）以及其他函数形式等等。事实上，变量也可看作是一种特殊的运算符，即有0个输入、1个输出的运算符。边表示运算符和/或变量之间的运算关系。在数据流图运行时，节点代表数据的存储，每个节点对应一个存储位置。例如，节点可以映射到硬盘、内存或CPU寄存器中的某个物理地址或者虚拟地址。其中数据可以是一个变量，或者这个变量所赋的值，或者一个运算结果，该运算结果可以是一变量或者常量等数学表达形式，边则代表数据的传输，也就是一个节点内的数据传输向该边指向的另

一个节点。

通信对端：对通信的双方，在描述一方的通信过程时，这一方为本端，另一方为通信对端。例如发送数据的一端（该数据的发送端）的通信对端为接收数据的一端（该数据的接收端），接收数据的一端（该数据的接收端）的通信对端为发送数据的一端（该数据的发送端）。通信的双方可以从物理机，虚拟机，容器，进程和线程等多个粒度来描述。例如，发送数据的一端通过一进程或者线程执行发送原语以发送数据，接收数据的一端通过一进程或者线程执行接收原语以接收该数据，则也可称执行该接收原语的进程或者线程为该数据的发送端的通信对端，或者可称执行该接收原语的进程或者线程为该发送原语的通信对端；同理可称执行该发送原语的进程或者线程为该数据的接收端的通信对端，或者可称执行该发送原语的进程或者线程为该发送原语的通信对端。下文的描述中涉及上述表达的，不再进行说明。

对端计算节点：对将一个数据从一计算节点传输到另一计算节点的通信过程，使用发送原语发送该数据的计算节点为源计算节点，使用接收原语接收该数据的计算节点为目的计算节点。则对该数据的源计算节点，其对端计算节点为该数据的目的计算节点，对该数据的目的计算节点，其对端计算节点为该数据的源计算节点。

数据流图参数：数据流图中，参数是指图上的边所承载的、用于被计算节点处理或由计算节点反馈的数据。也就是说，数据流图参数就是要从一个节点（即该边的源节点）传输到该边指向的另一个节点（即该边的目的节点）的数据。显然，对一个数据流图，数据流图参数的传输，是计算该数据流图的一部分，并且，一数据流图中的节点指示的存储位置在同一设备（如同一CPU内存，同一GPU内存等）的情况下，该数据流图参数的传输可以是进程内的内存复制过程；另一方面，一数据流图中的节点指示的存储位置跨设备（如同一主机的CPU内存和GPU内存，不同主机上的设备等）的情况下，该数据流图参数的传输可以是进程间的通信过程，而若源节点和目的节点所指示的存储位置跨主机，则需要基于网络进行通信。

数据流图参数的源地址是该数据流图参数在源计算节点中的存储位置，该源地址可以记载在承载该数据流图参数的边的源节点内。

数据流图参数的目的地址是该数据流图参数在目的计算节点中的存储位置，该目的地址可以记载在承载该数据流图参数的边的目的节点内。

节点的地址：对数据流图中的节点，可以是该节点所指示的物理地址或者虚拟地址。节点的地址会在数据流图参数通信中被使用。

大小（size）：例如本申请文件中提及的数据流图参数的大小，或者消息大小。表示一数据或消息所占的存储空间，也就是该数据或消息包括的数据量，一般以字节（Byte）为量纲。例如2KB, 0.5MB等。

下面结合图1，举例说明数据流图、节点、边以及数据流图参数是如何表达计算逻辑的。图1所示的是一个数据流图，该数据流图用于表达“两数相加之后乘以第三个数以得到一个结果”这样的计算逻辑，用算式可表达为 $E = (A + B) \times D$ 。其中，该数据流图具有A、B、C、D、E共5个节点，以及a、b、c、d共4条边。在数据流图定义时，节点A、B、D各表示一个变量，节点C、E分别表示加法与乘法运算。边a、b表示节点C中加法运算的两个加数来自节点A、B，边c、d表示节点E中乘法运算的两个因数来自节点C、D。

在数据流图运行时，节点A、B、D表示输入变量的存储位置，节点C表示加法运算结果的存储位置、E表示乘法运算结果的存储位置。节点所表示的存储位置可以映射到硬盘、内存或CPU

寄存器等物理设备中的用于存储数据的地址。边a、b代表节点A、B映射的存储位置中的数据传输到节点C映射的存储位置的过程，边c、d代表节点C、D映射的存储位置中的数据传输到节点E映射的存储位置的过程。这些边代表的数据传输过程可以是映射到进程内的内存复制过程，例如这些边所连接的节点所指示的存储位置在同一主机。当然，这些边代表的数据传输过程也有可能映射到进程间基于网络的数据通信过程，如这些边所连接的节点分布在分布式系统中。例如图中所示，A中输入1，B中输入3，D中输入5，则a边上传输的值是1，b边上传输的值是3，d边上传输的值是5，则C中得到的值为4，c边上传输的值是4，E中得到的值为20，这个数据流图表示，计算的是 $(1+3) \times 5=20$ 。

应当理解，本申请涉及的计算平台，可以部署在一个或者多个计算节点上，该计算节点可以是物理机或者虚拟机，也就是说，本申请涉及的计算平台，在虚拟机领域同样适用。该物理机可以是服务器或者一些具有计算能力的终端，例如个人电脑或者笔记本电脑等，本申请不做限制。本申请以计算数据流图为例进行说明，上述的每个计算节点上可以运行一个或多个进程，以对数据流图的子图或者副本进行运算，下面举例描述计算平台的软硬件运行环境。需要理解，计算节点与数据流图中的节点是不同的技术含义。用户在计算平台上加载需计算的数据流图，使用计算平台对该数据流图进行计算。

本申请的应用场景是运行在一台或多个计算节点上的、支持分布式计算平台软件。例如，计算平台可以是机器学习平台，再例如机器学习平台可以是用于对多层神经网络进行机器学习的深度学习平台。也就是说，本申请中的计算任务跨设备运行，所谓跨设备，是指将数据流图的程序代码分布在一台或多台服务器的多个计算设备上执行的情况，这里的计算设备可以是CPU，也可以是GPU、FPGA（Field-Programmable Gate Array，现场可编程门阵列）、MIC(Many Intergrated Core，成众核)或者其他具有计算能力的硬件设备。这类平台软件包括但不限于TensorFlow、MXNet、CNTK等。

要理解的是，在上述跨设备运行计算平台软件的场景中，需计算的数据流图可以分成多个子图（各子图可以是该数据流图的一部分）或者多个该数据流图的副本分布于多个设备。

计算平台中，一个数据流图的多个副本或者多个子图是由多个进程执行计算的，例如一个进程计算一个子图或者一个副本，或者一个进程计算一个物理机上多个设备的子图或者副本，或者一个物理机上多个设备的子图或者副本由两个或者两个以上的进程计算。图2是从进程的角度，描述进程对应的计算平台的软硬件资源。以计算平台中的两个用于计算数据流图的进程为例进行说明，图2的例子中，并不限定这两个进程是否在同一计算节点上，以及一个进程可计算一个物理上的多个设备的子图或者副本。

图2所示的进程1，即1011，在计算数据流图的过程中，使用主机内存1012和GPU内存1016。进程2，即1021，在计算数据流图的过程中，使用主机内存1022和GPU内存1026。主机内存1012和1022可以是相同的物理机或不同的物理机的主机内存。GPU内存1016和1026可以是在不同物理机上的GPU内存，或者是在相同物理机上的不同的GPU内存。应理解，主机内存1012和1022是相同的物理机的主机内存的情况下，代表主机内存中分别分配给进程1和进程2的内存地址空间，代表不同的地址段。平台运行时代码1013和1023加载在主机内存1012和1022中运行，平台运行时代码就是计算平台自身系统的代码，用于运行计算平台的软件环境，这部分代码用户是无法编辑的。图2中的核函数代码，则加载在进程对应的主机内存和GPU内存上运行。核函数代码用于实现表达局部计算逻辑的多种核函数，可以理解为一个包括多种核函

数的核函数库，核函数用于表示一些较为复杂的逻辑运算规则，可以被数据流图中的节点调用，例如，核函数可以是矩阵的运算，如点乘，叉乘等，或者卷积计算，这些运算需要较为复杂的一段指令去实现。一种实现方式下，同类别设备的内存中（相同型号的设备，如不同型号的 GPU 不是同类别设备，GPU 与 CPU 也不是同类别设备），部署的核函数是一样的，而对不同类别的设备则部署的核函数会不同。不同类型的设备中，所部署的核函数种类会有交叉，例如某 GPU 内存中部署核函数 A,B,C，另一种 CPU 的内存中部署核函数 B,C,D 等，也就是说一种核函数可部署在多个设备上。至于核函数如何部署，是由计算平台进行分配的，例如可以写在计算平台的库中，本申请文件不做赘述。而在用户使用计算平台的资源进行计算时，会根据不同设备的负载情况以及数据流图的分布情况，动态调度不同的设备上的核函数。

而数据流图的副本或者子图，分别加载在主机和GPU内存上保存,图2中，数据流图中的圆圈代表节点,圆圈之间带有箭头的短线代表连接边，该短线起始的圆圈（短线不带箭头的一端所连接的圆圈）代表该连接边的源节点，该短线的箭头指向的圆圈就是该连接边的目的节点。源节点和目的节点都可以指向任一主机内存或者GPU内存中的物理地址或者虚拟地址。

具体图2中描述为，核函数代码1014运行于主机内存1012，数据流图1015保存于主机内存1012；核函数代码1024运行于主机内存1022，数据流图1025保存于主机内存1022；核函数代码1017运行于GPU内存1016，数据流图1018保存于GPU内存1016；核函数代码1027运行于GPU内存1026。数据流图1028保存于GPU内存1026。数据流图1015、1018、1028以及1025是同一数据流图的副本。例如，连接边1019的源节点记载的地址指向进程1（即1011）使用的主机内存1012中的一个地址，目的节点记载的地址指向进程1（1011）使用的GPU内存1016中的一个地址，因此该边从源节点到目的节点的计算过程需要执行进程内跨设备的数据流图参数通信1010。再例如，连接边1029的源节点记载的地址指向进程1（即1011）使用的GPU内存1016中的一个地址，目的节点记载的地址指向进程2（即1021）使用的GPU内存1026中的一个地址，因此该边从源节点到目的节点的计算过程需要执行进程间跨设备数据流图参数通信1020。若进程1和进程2位于不同的主机，则进程间跨设备数据流图参数通信1020是跨物理机的通信。

下面结合图3，进一步说明本申请涉及到的一种软件运行环境和硬件架构。图3中，以一机器学习平台3011为例，该机器学习平台（即3011）运行在服务器1（即3001）、服务器2（即3002）、服务器3（即3003）和服务器4（即3004）上，服务器1至服务器4这四台服务器之间通过网络交换机3031进行通信。以服务器1为例，详细示意出了服务器中包括的软件和硬件。硬件方面，服务器1到服务器4上都安装有CPU（如服务器1中的3021）、主机内存（如服务器1中的3022）和网卡（如服务器1中的3024）。服务器中可以包括GPU卡（如服务器1中的3023），其中，GPU卡封装有GPU内存（如服务器1中的3025）。软件方面，服务器上1到服务器4部署有机器学习平台软件（如服务器1中的3011），该软件包含编程接口（如服务器1中的3012）、运行时引擎（如服务器1中的3013）、内存管理（如服务器1中的3014）和通信管理（如服务器1中的3015）等模块。其中，内存管理模块（如服务器1中的3014）管理的数据流图（如服务器1中的3016）上的参数存储于主机内存（如服务器1中的3022），还有一部分参数还可以存储于GPU内存（如服务器1中的3025）。内存管理模块3014将主机内存3022或者GPU内存3025中存储的数据流图3016读取出来，以进行计算，计算过程中如果需要与其他服务器进行数据流图参数的通信，则可以通过通信管理模块（如服务器1中的3015）与网卡（如服务器1中的3024），实现数据的接收与发送。需理解的是，机器学习平台3011通过服务器1中的进程运行，而编程接口、运行时引擎、内

存管理等可以认为是具有不同功能的几段代码。

应理解，机器学习平台相当于一个运行于计算机硬件之上的操作系统，也可以类比其他操作系统划分为应用层和核心层，应用层可供用户进行编辑或者输入数据，应用层和核心层之间提供接口，以使用户编辑的指令或者函数调用这些接口使用核心层执行这些指令或者函数。

- 5 应理解，主机内存也就是用于计算数据流图的进程可使用的CPU内存，故即使该进程位于虚拟机或者容器上，其分配到的内存，如虚拟CPU内存，在本申请中也可以被称为主机内存。

本申请涉及的场景要求一幅数据流图的多个子图或多个副本能够跨设备协同执行，包括在同一服务器或多台服务器的多个进程之间协同执行，或者在同一服务器内同一进程管辖的CPU与加速器硬件之间协同执行。然而，跨设备进行数据流图计算必然涉及数据流图参数的交互，目前使用的参数交互方法不能满足机器学习平台对计算效率的需要。

- 10 另一方面，高性能计算（High Performance Computing，HPC）中，采用以消息传递接口（Message Passing Interface，MPI）技术为代表的消息传递（Message Passing，MP）机制，MPI包括协议和语义说明，这种机制传输参数的效率较高。高性能计算是指使用超大型电子计算机，进行大量复杂的计算，比如，生物遗传基因测定，导弹弹道计算，核业的反应堆的模拟，航天领域飞行器轨迹计算，或者天文观测领域的星轨计算等等。这些计算使用的硬件性能远高于一般的民用或者商用场景中的计算机，如超级计算机（super computer）。超级计算机是一种超大型电子计算机，具有很强的计算和处理数据的能力，主要特点表现为高速度和大容量，配有多种外部和外围设备及丰富的、高功能的软件系统。现有的超级计算机运算速度大都可以达到每秒一太(Trillion, 万亿)次以上为了便于理解高性能计算领域中超级计算机与其他领域中使用的服务器的不同，以下来举例说明，一些厂家这样规定：计算机的运算速度平均每秒1000万次以上；存储容量在1000万位以上的属于超级计算机，例如，美国的ILLIAC-IV，日本的NEC，欧洲的尤金，中国的“银河”计算机等。再例如，适用于现代集群体系结构的超级计算机，例如美国的BlueGene、日本的“京”、欧洲的Piz Daint、中国的“神威”、“天河”等。如果超级计算机作为计算节点运用在分布式场景下，那么计算节点间使用高性能通信网络（下文简称高性能网络），例如无限带宽InfiniBand技术，其中使用专用的网络设备（如路由、网卡、网线、交换机等等），其造价约为企业用的同类设备的数倍（如用于InfiniBand的路由是普通路由价格的4倍）。普通的TCP/IP协议或者以太网传输协议运行在这类专用网络设备上，也可以在普通的商用或者民用服务器之间使用InfiniBand等高性能通信网络，但是有效载荷比低，无法充分利用这类专用的网络设备的性能（如带宽，吞吐量），因此，在高性能通信领域，这类专用的网络设备以及超级计算机之间通常使用专用的通信协议进行数据传输。例如，在高性能通信领域，使用MPI技术进行数据传输，在设备中可以表现为，设备中装载有MPI库。

- 25 需要说明的是，MPI库指MPI开发库，包括多种MPI接口函数（也称MPI指令或者MPI原语）和MPI通信协议。从软件架构角度，MPI接口函数可以理解为是应用层和核心层之间接口，若类比使用TCP/IP技术的软件架构，则MPI接口函数在其中相当于Socket接口函数（即套接字）。

一种实现方式下，使用MPI函数库中的接口函数进行一数据的传输，该传输过程中，源进程为发送端物理机上使用发送原语发送数据的进程，目的进程为接收端物理机上使用接收原语接收该数据的进程。MPI函数库中的发送原语和接收原语是成对使用的。对通信的两个进程，源进程使用一发送原语向一目的进程发送数据，该目的进程使用一接收原语处理该数据，这一对原语的

接口参数中,携带相同的消息大小(size),相同的消息标记(tag),以及,该发送原语的接口参数中还携带有该目的进程的标识,该目的进程的信息是用于指示该目的进程,例如,可以是该目的进程的序列号,也可以用其他形式的信息标识该目的进程,本申请不做限定。而该接收原语的接口参数中还携带有该源进程的标识,该源进程的标识是用于标识该源进程,例如,可以是该源进程的序列号,也可以用其他形式的信息标识该源进程,本申请不做限定。

需要理解的是,MPI库中,可对可能进行通信的进程赋予唯一的进程序列号(如下文提及的rank)。而主机中可维护该MPI库中的进程序列号与进程所在物理机的映射关系,或者进程序列号与进程对应的设备的信息之间的映射关系,或者进程序列号,进程所在物理机以及进程对应的设备三者之间的映射关系。这样就可以得知进行通信的进程是否在同一物理机,甚至是否使用相同的设备,从而MPI发送原语可以选择通过网卡,或者共享内存,或者本地回环网络设备(也就是内核虚拟设备)等方式发送。选择何种方式发送MPI原语与源进程和目的进程是否在同一物理主机或同一数据中心(Data Center,DC)有关,也与通信进程间使用的网络通信技术有关(如高性能网络,以太网等),本申请不做赘述。

MPI库可以配套使用RDMA(Remote Direct Memory Access 远程直接数据存储)技术,在目的进程所在主机的CPU未执行接收原语的情况下,源进程就可以通过发送原语将数据以及数据的相关信息(如下文提及的三元组)写入目的进程所在主机的MPI缓冲区。当然,MPI库也可以使用TCP/IP协议,则目的进程所在的主机可以通过I/O(Input/Output)设备接收来自源进程的数据,并将该数据的相关信息(如下文中提及的三元组)写入目的进程所在主机的MPI缓冲区,以在目的进程执行接收原语过程中将数据写入主机的MPI缓冲区。对于MPI技术与何种通信协议搭配使用,属于本申请记载的技术方案所涉及的不同应用场景,本申请不做限制。而在与不同的通信协议搭配使用以进行数据传输的场景中,MPI原语在实现数据的发送或者接收等数据传输的过程中如何工作,属于MPI库的内部运行机制,请参考不同版本的MPI库的相关技术资料和说明,本申请不做赘述。

MPI技术中,使用MPI库的主机内存中有一专用存放通过MPI原语处理的数据的地址空间,被称为MPI缓冲区。该MPI缓冲区通常被定义为固定的大小,如64KB,1MB等。需理解,发送原语要传输的数据可以小于缓冲区大小,也可以大于缓冲区大小,在要传输的数据大于缓冲区大小的情况下,源进程在执行发送原语过程中可以将要传输的数据进行拆分。而在目的进程对应的MPI缓冲区被占满的情况下,发送原语就无法继续将数据写入目的进程的MPI缓冲区了。这种情况下需要目的进程执行接收原语将接收的数据写入该数据的目的地址。例如,该目的地址可以位于GPU芯片,用户使用的内存空间(主机内存中由系统分配给一用户用于存储该用户的数据)以及其他内存设备中。目的进程执行接收原语以接收数据,该接收过程可以包括通过该接收原语探测到目的进程所在的主机的MPI缓冲区中已写入源进程发送的数据,以及将源进程传输来的数据从MPI缓冲区保存至该数据的目的地址中(例如主机内存或者GPU内存等)。这样,该目的进程就可以使用该数据。

表1对常用的MPI接口函数进行说明。

表 1

原语种类	接口函数	语义
发送原语	MPI_Send	同步发送

	MPI_Isend	异步发送
接收原语	MPI_Recv、MPI_Irecv	同步接收 异步接收
探测原语	MPI_Probe、MPI_Iprobe	同步探测 异步探测
等待原语	MPI_Wait	等待

例如，表 1 中的等待原语，用于等待一条原语执行完毕。表 1 只是对几种原语举例说明，例如，发送原语还可以包括其他能够实现发送信息这一功能的其他形式的 MPI 接口函数，接收原语还可以包括其他能够实现接收信息这一功能的其他形式的 MPI 接口函数。

- 为了提高机器学习平台在分布式网络环境下的数据传输性能（数据主要是数据流图参数），
- 5 目前业界已开始尝试将高性能计算领域的 MPI 通信技术使用到计算平台，比如下文中提及的百度（Baidu）开发的 TensorFlow-Allreduce 项目，但是由于计算平台本身的硬件条件与高性能计算领域不同，使得现有的将 MPI 技术使用到计算平台会影响了数据传输性能，同样使得计算平台的计算效率受到较大的限制。

- 以下结合图 4，简单介绍百度（Baidu）开发的 TensorFlow-Allreduce 项目提出的方案，该
- 10 项目的将机器学习平台分布在分布式系统中。其技术方案就是在机器学习平台使用 MPI 库的消息传递原语，构建所有进程共同参与的集合通信，以在集合通信过程中完成数据流图计算中的数据流图参数的归约计算和参数分发，从而能够在 InfiniBand 等高性能网络环境中提升数据流图参数通信的性能。集合通信步骤是多个进程同时参与的包括归约计算的通信步骤，归约计算需要这多个进程进行参数的交互，例如，求最大值，最小值，或者求平均，都是一种归约计算，以求平均
- 15 为例，多个进程各自可读取一部分待求平均的数值，需要参与计算的多个进程将各自的数据发送给执行求平均算法的进程，也就是需要进程间的参数通信。而归约计算显然是机器学习中常用的算法，故而可采用集合通信来进行数据流图参数的传输。

- 图4给出了该技术方案在运行时的时序图。该方案中，所有参与数据流图计算的进程（2011、2021、2031）分别用于运行一子图或者副本。每个进程循环往复地执行两种例程。例程就是一组函数集合，可以执行某种功能，例如一个系统使用例程提供对外接口或者服务。比如，操作系统的API、服务等就是例程；Delphi或C++Builder提供的标准函数和库函数等也是例程。例程A进行数据流图局部参数生成计算（图中的2012、2015、2022、2025、2032、2035都是例程A）；例程B进行数据流图全局参数归约计算（图中2013、2016、2023、2026、2033、2036都是例程B）。其中，例程B的技术实现是基于MPI_Send和MPI_Recv原语的参数通信（2014、2017、2024、2027、2034、2037）。这些参数通信的收发链构成环状，使得每个进程的归约计算结果最终能够到达其他所有进程，从而完成一次全局数据流图参数收集和分发过程。这一集合通信过程构成了一次全局栅障同步（globe barrier synchronization）（2041、2042），即所有进程必须保证能够处在相同的一轮迭代时进行全局参数归约计算，也就是所有进程都完成相同一轮计算后，再进入下一轮迭代，2041和2042表示两轮迭代过程。具体的结合图1，全局栅障同步可以理解为，先
- 20 到达例程B的进程必须等待其他进程也到达例程B，其例程B才能最终完成。例如可以在代码中设置一个标识或者指令，使得各进程在执行到该标识或者指令时，检测其他进程是否也执行到相同的该标识或者指令，在所有进程均执行到相同的标识或者指令的情况下，再继续执行接下来的指
- 25
- 30

令。

需要说明的是,在该方案进行图4所示的在使用MPI原语进行参数通信之前,参与通信的源进程和目的进程需要先进行交互以得到通信对端的信息,否则无法通过MPI原语传输数据。

然而,以 TensorFlow 为代表的计算平台的通信时序是动态、随机的,直到运行时才能得知通信原语所要处理消息的源/目标和大小等信息,且现有的机器学习平台使用的通信技术不要求源进程和目的进程使用包括配对的原语和接口参数,而以 MPI 库为代表的消息传递机制要求在编程开发时就需要为通信原语指定消息源或者目标和消息大小等接口参数,且收发双方配对原语的接口参数紧耦合。TensorFlow-Allreduce 方案没有解决这个矛盾,它新增了一组编程接口,使用该新增的接口编写其方案,开发了数个用于集合通信的指令,在该指令中封装了 MPI 接口,改变 TensorFlow 的编程习惯使之适应 MPI 库的要求。这样一来,用户必须学习使用该方案提供的开发库接口,重新编写或改写应用代码,才能获得消息传递通信所带来的性能优势。因此,该方案的易用性及通用性不足。更重要的是,计算平台的通信时序的动态性和随机性使得数据通信的双方难以及时确认对端以及进行协商,该协商过程也增加了计算平台中数据通信的负担,从而影响数据的传输效率。

另一方面,以MPI库为代表的消息传递机制要求通信是有序、同步的操作为主,在库内缓冲区机制的辅助下允许适度的非配对消息穿插和异步操作。然而,以TensorFlow为代表的机器学习、深度学习平台由于计算过程不要求同步,故其中的通信时序也是乱序和异步的,大量随机的通信操作穿插执行,并不要求每个进程在计算时都处于相同的迭代轮次。TensorFlow-Allreduce方案没有解决这个矛盾,它选择在TensorFlow的计算迭代轮次之间使用全局栅障同步,避免跨轮次的通信穿插,从而满足MPI库的约束,这使得计算速率较快的进程频繁处于等待状态,从而造成计算资源浪费。这种同步等待的时间开销也会有可能降低或抵消消息传递通信的性能优势,使得整个机器学习平台的整体计算速率取决于最慢的进程,从而影响了机器学习平台的计算速率。

百度TensorFlow-Allreduce方案并不是在原本的TensorFlow系统中进行改进,而是一套封装在原本的TensorFlow系统之外的开发库,该开发库是外接的函数库,相对独立,通过TensorFlow系统的外接扩展接口接入系统,其提供的使用集合通信的传输方式(即Allreduce)是与原先TensorFlow系统提供的接口并列的另一组通信接口。作为外接的函数库,并没有修改TensorFlow平台核心层代码。TensorFlow-Allreduce方案是一套独立的代码,这套代码会从TensorFlow平台外部调用TensorFlow平台提供的应用程序编程接口(Application Programming Interface, API)。需理解,对于机器学习平台,也可以分为应用层和核心层,其中应用层用于接收用户输入的数据,待训练或者待学习的数据,运行用户编写的算法或者代码等。而如前文和图3所描述的运行时引擎3013,内存管理3014,通信管理3015等模块都可以认为属于核心层。这套开发库不能分辨待通信的数据流图参数的源地址或目的地址指向的物理位置是主机内存还是GPU内存,TensorFlow系统对该开发库屏蔽了这些信息。这就要求其使用的MPI库有能力感知数据流图参数的源或目的地址的物理位置,从而外部机制能够将数据正确的读取和写入。

TensorFlow-Allreduce方案使用一类CUDA(Compute Unified Device Architecture)感知(CUDA-aware)的MPI库。这类MPI库如果配套使用能够支持CUDA编程接口的GPU,例如NVIDIA(英伟达)的GPU,则该MPI库的发送原语可以确定待被该发送原语处理的信息的源地址位于何种内存,该MPI库的接收原语可以确定待被该接收原语处理的信息的目的地址位于何种内存。例如,如果待发送数据位于GPU内存,该MPI库发送原语就将位于GPU内存中的数据复制

到相应的主机内存再发送。而事实上，不是所有的MPI库都CUDA感知，这限制了将MPI技术运用在机器学习平台时，对MPI库的选择。

另一方面，包括TensorFlow在内的多种机器学习平台，其核心层往往调用非线性安全的CUDA驱动层接口访问GPU内存。这样一来，CUDA感知的MPI库在同TensorFlow等使用CUDA接口的机器学习平台共同使用时，会相互抢占资源，从而存在性能缺陷。可见，MPI库和TensorFlow等平台访问GPU内存的机制相同，而MPI库与TensorFlow平台核心层是使用不同的线程来访问GPU内存的，且多个线程不可以并发访问GPU内存，也就是说，在一个线程占用了CUDA驱动层接口的情况下，其他线程无法使用该接口也就无法访问GPU内存。这就需要采用一些调度方案以使多个线程能访问GPU内存，例如使用互斥锁或流同步机制。由于百度的方案是外接的函数库，故无法感知到一个函数在执行过程中，所涉及的子函数关系和调用过程。例如对于百度方案中使用MPI函数库的发送原语和接收原语，如果传输的数据位于GPU内存，则执行该发送原语或者接收原语的线程在执行该原语的整个过程中都会被加锁，或者计算平台以发送原语或者接收原语作为被流同步机制管理的指令。而实际上，发送原语或者接收原语的执行过程包括多个子过程，并不是所有子过程都需要访问GPU内存，这就会带来额外的等待时间开销，从而影响消息传递的效率。以发送原语为例，执行发送原语的过程包括切割数据，插入指针，内存复制等多个子过程，只有内存复制需要访问GPU内存。

本申请提出一种分布式系统中数据传输的方法，能简化MPI技术应用在计算数据流图中的通信过程。该方法可以通过软件代码来实现，该代码包含在计算平台软件中，部署分布式计算系统中。以下以数据流图为计算平台计算的对象，计算过程中传输数据流图参数为例进行说明，本申请并不限制计算平台的计算对象，也不限制计算过程中传输的数据的种类。部署计算平台的多台物理机中保存有待训练的数据流图的副本或者子图，其中，该分布式计算系统包括第一计算节点和第二计算节点。其中，图5对应的实施例中，第一计算节点和第二计算节点是不同的计算节点。运行时，本发明的程序代码运行于服务器的主机内存，或者，主机内存和GPU内存。以下结合图5进行说明。应理解，如非说明，下述的S501~S508的编号，不代表步骤执行的先后顺序，例如，不规定S501和S502执行的先后顺序。

S501: 第一计算节点根据第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组，所述第一三元组包括消息标记、消息大小和目的进程序列号，其中，所述消息标记对应于所述第一数据流图参数的名称，所述消息大小对应于所述第一数据流图参数的大小，所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程。

其中，第一计算节点上的第一图数据结构保存有第一数据流图中的第一数据流图参数的名称、大小和通信对端标识，其中第一数据流图参数为所述第一数据流图的一连接边所承载的参数。该第一数据流图中的第一数据流图参数的通信对端标识对应于所述第二计算节点。

该第一图数据结构在不同的计算平台中可以是不同的数据结构，本申请不做限制。例如，在TensorFlow平台中可以是Tensor数据结构。可见，在第一数据流图以及第二数据流图中，记载该第一数据流图参数是从该第一计算节点传输到该第二计算节点的数据流图参数。

第一数据流图参数的名称(name)用于标识该第一数据流图参数，可以是该第一图数据结构中的一个字段也可以是分散在该第一图数据结构中的信息，也就是说，第一数据流图参数的名称可以是根据该第一图数据结构中的信息分析得到的，具体实现方式在不同的计算平台中各不相同，

例如在 TensorFlow 中，可参见本申请下文的相关段落。

该第一数据流图参数的大小（size）用于表示该第一数据流图参数所占的存储空间，也就是该数据流图参数的数据量。该第一数据流图参数的大小可以由该第一图数据结构中的一个字段获得，例如单位为 byte，记录数据流图参数的大小这一参数的数值，如 3000，4000 等。也可以是以分散在该第一图数据结构中的信息表示，例如该第一图数据结构参数中，在多个子结构中各标识该第一数据流图参数的一部分数据量，可根据这些信息计算得到该第一数据流图参数的大小。

该第一数据流图中的第一数据流图参数的通信对端标识可以是该第二计算节点的标识；或者该第一数据流图参数的目的地址所在的存储设备的标识，该存储设备位于第二计算节点；或者第二计算节点中接收该第一数据流图参数的进程的标识；或者其他用于指示该第一数据流图参数的接收端的信息，本申请不做限制。

综上，第一计算节点上的第一图数据结构保存有第一数据流图中的第一数据流图参数的名称、大小和通信对端标识，可以是在该第一图数据结构中包括承载该三种信息的字段，也可以是保存有能够得到该第一数据流图参数的名称、大小或者通信对端标识的信息。即所谓“保存有”可以是能够直接从该第一图数据结构中读取，也可以是根据该第一图数据结构中的信息分析获得。

当然，该第一数据流图参数本身也可以存储在第一计算节点的一个或者多个数据结构中。

例如，在 TensorFlow 平台中，该 S501 和 S503 的实现可以是在第一计算节点和第二计算节点的内存管理模块（如图 3 中的 3014）中添加一个元信息管理模块（具体可以是添加一段代码），该元信息管理模块可以将第一计算节点和第二计算节点中的数据流图的边上的数据流图参数的信息存储在数据结构中，该数据结构中包括上述的数据流图参数的名称、大小和通信对端标识。

另外，需要理解的是，一般机器学习平台中，由于通信的乱序和随机，进程执行各种原语对应的操作，即一个进程可以执行发送操作也可以执行接收操作，多数情况下，没有专门执行发送原语或者专门执行接收原语的进程。

S503: 第二计算节点根据第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第二接口参数生成算法生成第二三元组，所述第二接口参数生成算法与所述第一接口参数生成算法相同，所述第二三元组包括所述消息标记、所述消息大小和源进程序列号，其中，所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程；

其中，该第二计算节点上的第二图数据结构保存有第二数据流图中的所述第一数据流图参数的名称、大小和通信对端标识；所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点。

第二数据流图保存在第二计算节点上，第二数据流图可以是第一数据流图的副本，也可以与第一数据流图都是一个数据流图的两个子图。有关该第二数据流图中的第一数据流图参数的名称、大小和通信对端标识的说明，请参见 S501 中的相应的部分，此处不再赘述。

这样，无需与通信对端或者用户进行交互，就可以得到 MPI 发送原语和 MPI 接收原语所需的接口函数参数。

需要说明的是，生成消息标记、消息大小和源（或者目的）进程序列号通常使用不同的算法，即该第一接口参数生成算法包括第一算法、第二算法和第三算法。该第一算法、第二算法和第三算法可以将第一图数据结构和第二图数据结构中的信息转化为符合 MPI 接口参数格式的上述三元组。

而上文中描述的第一接口参数生成算法与第二接口参数生成算法相同，是指，第一接口参数生成算法包括第一算法、第二算法和第三算法，且，第二接口参数生成算法包括与该第一接口参数生成算法相同或者相应的第一算法、第二算法和第三算法。

5 则对 S501，一种实现方式可以是根据所述第一图数据结构中的第一数据流图参数的名称第一数据流图参数和所述第一算法，确定所述第一三元组中的消息标记，根据所述第一图数据结构中的第一数据流图参数第一数据流图参数的大小和所述第二算法，确定所述第一三元组中的消息大小，以及根据第一图数据结构中的第一数据流图参数的通信对端标识和所述第三算法，确定所述

10 相应的，对 S503，一种实现方式是根据所述第二图数据结构中的第一数据流图参数的名称和所述第二接口参数生成算法中的第一算法，确定所述第二三元组中的消息标记；根据所述第二图数据结构中的第一数据流图参数的大小和所述第二接口参数生成算法中的第二算法，确定所述第二三元组中的消息大小；以及根据第二图数据结构中的第一数据流图参数的通信对端标识和所述第二接口参数生成算法中的第三算法，确定所述第二三元组中的源进程序列号。

15 其中，消息标记用于表示该MPI发送原语发送的数据。消息标记可以通过第一算法处理该第一数据流图参数的名称得到，第一算法可以是任意二进制长度值转化为固定二进制长度值的算法，例如可以是哈希算法等；或者其他能将第一数据流图参数的名称转化为符合MPI原语的接口参数中消息标记的格式的算法。

20 消息大小用于表示该 MPI 发送原语发送的信息的大小。对第二算法，则一种实现方式下，可使该消息大小字段的值等于上述数据流图参数的大小的参数值，即 size。另一种实现方式下，可以使该消息大小字段的值等于上述的数据流图参数的大小的参数值加上一个值。这个加上的值是该发送原语要携带的其他信息的大小，如下文提及的该发送原语的包头长度。例如，MPI 发送原语发送的信息包括待发送数据和包头，则该消息的大小的值就是该待发送数据的大小加包头的大小。

25 源进程序列号是第一计算节点执行该MPI发送原语的进程的序列号，目的进程序列号是在第二计算节点执行与该MPI发送原语相对应的MPI接收原语的进程的序列号。需理解的是，由于本申请的第一数据流图和第二数据流图中，保存有该第一数据流图参数的源节点和目的节点，则可知第一数据流图参数的源地址对应的存储设备，以及第一数据流图参数的目的地址对应的存储设备，又由于计算平台中，用于计算数据流图（数据流图参数传输是计算数据流图的一部分）

30 第三算法是进程序列号与通信对端标识之间的映射关系，其中第一计算节点中包括目的进程序列号与通信对端标识之间的映射关系，第二计算节点中包括源进程序列号与通信对端标识之间的映射关系。第三算法可以是函数关系，也可以是计算节点中维护的映射表，本申请不做限制。第一算法、第二算法和第三算法的具体实现可以参见下文TensorFlow 平台的例子，下文中的具体实现方式也可以用在其他计算平台。关于进程序列号的描述可以参见下文TensorFlow 平台的例子，下文中的具体实现方式也可以用在其他计算平台。

35 显然，上文中以第一三元组为接口参数的MPI发送原语与以第二三元组为接口参数的MPI接收原语相对应，这样，在第一图数据结构和第二图数据结构中，包括有通信对端标识，解决了通信对端的进程在数据流图运行过程中不可知的问题，以及，需要传输该第一数据流图参数的通信双方，使用各自计算节点存储的数据流图中的信息和相同的接口参数生成算法生成三元组，就无需向对端交互自己的信息，以及无需协商生成三元组的算法，该方法能够在数据发送方和接收方

内独立运行，在双方无交互的情况下生成对应的三元组，简化了使用MPI原语通信的过程，能够提高分布式计算平台中数据传输的效率。

S505: 第一计算节点以所述第一三元组作为接口参数，调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数；

- 5 其中，本申请中描述的三元组的概念只用于表示其中的三个参数，而不限定这三个参数间的顺序。该三元组中的三个参数的格式满足MPI发送原语所携带的接口函数参数的格式要求。另外，MPI发送原语的接口参数包括但不限于该第一三元组，MPI接收原语的接口参数包括但不限于该第二三元组。

- 10 一种实现方式下，S505 中所述第一计算节点用于以所述第一三元组作为接口参数，通过消息传递接口 MPI 发送原语从所述第一计算节点的主机内存中读取所述第一数据流图参数，以向所述第二计算节点发送所述第一数据流图参数。

- 15 一种实现方式下，第一计算节点还保存有所述第一数据流图参数在所述第一计算节点的存储设备的信息，也就是下文中描述的数据的内存类型。则在 S505 之前，该第一计算节点执行 S504，即在所述存储设备的信息指示为其他存储设备的情况下，将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存，所述其他存储设备为所述第一计算节点中除主机内存外的内存。

该存储设备的信息可以是该存储设备的标识，也可以是用于表示该存储设备的编号，可根据标识或编号确定出该存储设备的存储类型，还可以是标识该存储设备的类型的信息等，或者其他形式的能达到上述功能的信息，本申请不做限制。具体实现可参考下文相关段落。

- 20 例如，该其他存储设备可以是 GPU 内存、或者其他处理单元的内存，如 FPGA 或者 DSP 等处理单元的内存等。该步骤可以参考下文描述的 TensorFlow 平台中具体实现以加强理解。该步骤可理解为是前述的计算平台的通信管理模块实现，使用的是计算平台核心层的访问其他存储设备的机制。例如对 GPU 内存，可采用该平台中的 CUDA 编程接口提供的函数将待发送数据复制到主机内存。这样，第一计算节点在使用 MPI 发送原语前，就会将第一数据流图参数准备在第一计算节点的主机内存中，而 MPI 发送原语只从该第一计算节点的主机内存中读取该第一数据流图参数，无需与计算平台争抢读取其他存储设备的资源，提高了 MPI 发送原语的执行效率。另外对 MPI 库的选择就可以更灵活，不要求 MPI 库支持访问其他存储设备，也不会产生计算平台和 MPI 库对访问其他存储设备的竞争，具体论述可以参考下文的相关段落。当然，如果选用支持访问 GPU 内存的 MPI 库，也可以通过 MPI 库执行该步骤。

- 30 应理解，第一数据流图参数可以存储在主机内存的缓冲区（其具体说明请参考本申请的相关段落），也可以是主机内存中分配给用户的存储空间，本申请不做限制。例如 MPI 库配套使用 RDMA 技术的情况下，可以获取主机内存中任何被注册的地址中的数据，而 MPI 库配套使用 TCP/IP 技术的情况下，则需要将存储在用户的存储空间中的第一数据流图参数复制到 MPI 缓冲区或者数据缓存区（见下文）中使用。

- 35 也就是说，下文中提及的数据缓存区，可以在源计算节点和目的计算节点都设置，这样，该数据缓存区与原有的 MPI 缓冲区（二者可统称为缓冲区）配合使用，在缓冲区未被完全占用的情况下，更能够容忍发送与接收操作的异步进行，更能适应学习平台中需要进行多个数据的复杂、异步和乱序的收发操作。

S507: 第二计算节点根据该第二三元组，调用 MPI 接收原语处理所述第一数据流图参数。

需要理解的是，调用MPI接收原语处理所述第一数据流图参数，“处理”在不同的场景下，可以对应不同的操作，本申请不做限定。例如可以是以下操作中的一种或多种：调用MPI接收原语将该第一数据流图参数接收到主机内存的数据缓冲区；调用MPI接收原语修改该第一数据流图参数的标记，以将已在主机内存中的第一数据流图参数提供给计算数据流图的进程使用；将该第一数据流图参数从数据缓冲区存储到目的地址。关于MPI接收原语如何处理该第一数据流图参数，可进一步参考下文以TensorFlow平台为例的说明中的相关段落。

一种实现方式下，所述第一数据流图参数接收原语携带所述第一数据流图参数的目的地址，则对 S507，具体实现为将所述第二三元组作为所述 MPI 接收原语的接口参数调用所述 MPI 接收原语，以将所述第一数据流图参数从所述数据缓存区存储到所述目的地址。例如，该目的地址位于主机内存中的用户内存空间。

若该目的地址位于其他存储设备，该其他存储设备为该第二计算节点中除主机内存外的存储设备如果使用的 MPI 库支持访问其他存储设备，例如 GPU 内存，该第一 MPI 接收原语也可以将目的地址在 GPU 内存的数据存储到对应的目的地址。而另一种实现方法下，这种情况则可以通过计算平台自身的访问其他存储设备的机制，在 S507 之后进行。即 S508：在所述第一数据流图参数的目的地址对应其他存储设备的情况下，所述第二计算节点将所述主机内存中的第一数据流图参数存储到所述目的地址，所述其他存储设备为所述第二计算节点中除主机内存之外的内存。

S508 与前文所述的第一计算节点的 S504 类似，具体说明和有益效果请参见该步骤的段落以及下文中的相关段落。S507 可以认为是前文提及的 MPI 客户端执行的。

应理解的是，在部署有分布式机器学习平台的多个物理机中，都保存有数据流图，也通过进程执行该机器学习平台的代码以训练该数据流图。那么对于数据流图中第一数据流图参数，第一计算节点是发送端，而对于数据流图中的另一数据，该第一计算节点就可能是接收端。则 S505 和 S507 的具体实现方式可以参考下文中相关的描述。

由于机器学习平台执行指令的乱序性，数据可能无法在写入接收端的主机内存后及时被 MPI 接收语句处理，而 MPI 库自带的 MPI 缓冲区的空间小，无法满足机器学习中动辄几 MB 的数据传输要求，故可以在第二计算节点的主机内存中划分一块数据缓存区，该数据缓存区专用于存储 MPI 原语使用的数据，具体分析请见本申请相关段落。

则一种实现方式下，S507 包括通过 MPI 探测原语检测所述第二计算节点主机内存中的数据缓存区，以获得所述第一数据流图参数的第二三元组，所述数据缓存区专用于存储由 MPI 原语处理的数据；调用 MPI 接收原语，以处理所述第一数据流图参数，所述 MPI 接收原语的接口参数包括所述第二三元组。

一种实现方式下，该第二计算节点运行有第一线程和第二线程，S507 包括：

该第一线程通过消息传递接口 MPI 探测原语检测所述主机内存中的数据缓存区，以获得该第二三元组；该第一线程根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语，以处理所述第一数据流图参数，所述数据缓存区中的第二三元组是所述第二计算节点根据所述 MPI 发送原语得到的；所述第二线程在确定所述第一数据流图参数被所述第一 MPI 接收原语处理后，将第二 MPI 接收原语修改为 MPI 等待原语，所述第二 MPI 接收原语为未被所述第二线程执行的与所述第一数据流图参数对应的接收原语，所述第二 MPI 接收原语的接口参数包括所述第二计算节点生成的第二三元组，所述 MPI 等待原语用于等待所述第一 MPI 接收原语执行完毕。

其中，第二三元组可以是根据接收到的 MPI 发送原语的接口参数获得，也可以是根据接口参数以及该 MPI 发送原语传输的数据中分析得到，本申请不做限制。

也就是说第二计算节点可以再拉起一个线程（可称为轮询线程）执行 MPI 探测原语，以检测该第二计算节点的主机内存的缓冲区，缓冲区中包括上述的数据缓存区，该数据缓存区往往比系统自带的 MPI 缓冲区大，具体说明可参见下文相关段落。这样就可以找到未来得及被 MPI 接原语处理的数据。该线程可以采用轮询的方式执行 MPI 探测原语检测缓冲区，一旦发现这样的数据，则调用该数据对应的 MPI 接收原语（为区分，称为第一 MPI 接收原语）而将原本待执行的 MPI 原语（为区分，称为第二 MPI 接收原语）修改为 MPI 等待原语，该 MPI 等待原语用于等待第一 MPI 接收原语执行完毕，待第一 MPI 接收原语执行完毕，则该线程继续轮询以继续处理待 MPI 接收指令处理的数据。

这样，就使得数据能更及时地被接收原语处理，也使得第一计算节点其他待执行的发送原语能更快地执行，从而，提升了数据传输的效率。

综上描述了从发送端到接收端的机器学习平台的数据传输方法，该方法通过使用本地的图数据结构和接口参数生成算法，获得了使用 MPI 原语所需的接口函数参数，避免了传输数据前发送端与接收端的参数配对，提高了数据通信的效率，进一步的，获得待传输数据在发送端和接收端的存储位置，从而在该存储位置不在主机内存的情况下，在数据发送前和数据接收后，通过机器学习平台的机制将数据进行物理机内的跨存储设备移动，拓宽了 MPI 库的选择范围，也避免了 MPI 库和机器学习平台在跨存储设备移动数据时争抢资源。以及通过设置专用的数据缓存区和轮询线程，使得消息传递通信缓冲区能够在消息接收原语尚未调用、消息最终目的地址未知的情况下，使得消息发送原语能够执行数据发送，并在数据发送完成之后立即返回。缓冲区为未来的消息接收原语临时保存数据，从而使得消息发送原语无须与消息接收原语进行同步操作，解决了二者固有的时序约束。发送方无须同步等待，使其节约了执行时间，有助于提升性能。

上述的这些改进使得 MPI 库能够很好地适应机器学习平台的特点，提高了通信效率，由于 MPI 库是高性能传输领域的技术，这就使得机器学习平台能够充分使用高性能传输网络的资源，大大提高通信效率，从而提升计算平台的计算效率。

关于上述的对应图5的机器学习平台中传输数据的方法的其他技术细节、对其中涉及的名词或者步骤的说明以及各步骤的有益效果的详细描述，可进一步参考本申请文件的其他相关段落。

需要理解的是，上述方法所描述的思想可以用多种计算平台，例如可以是下文详细描述的机器学习平台，也可以是图计算平台，或者流计算平台等，本申请不做限制。

下面介绍本申请涉及的计算平台计算数据流图的一种流程。应理解，该流程是为了解释计算平台在计算数据流图的过程，仅为一种示例，本申请对该过程不做限制。该流程适用于本申请提及的TensorFlow等机器学习平台。总体上，包含数据流图创建（或称为“数据流图定义”）和数据流图运行。其中，在一种实现方式下，数据流图创建可细分为全图构造、子图提取、图切分、图优化等子步骤。数据流图运行可细分为输入数据填充、算法核函数执行、输出数据获取等子步骤。例如，本实施例提出的方法中，步骤S501~S507可以认为属于算法核函数执行子步骤，而在步骤S501前，将数据流图参数的名称、大小等信息写入图数据结构则属于数据流图创建过程。

数据流图创建，该过程将用户使用程序语言编写的算法转换为计算平台可理解的数据流图结构。

具体的，其中包括，全图构造，即将用户编写的所有算法代码全部转换为数据流图结构。之

后，对转换后的数据流图结构进行子图提取，这是由于该数据流图中，往往会包括与得到最终计算结果无关的节点和边。故在一种情况下的子图提取中，计算平台将与最终计算结果所在节点所连接的节点和边从全图中提取出来，作为待运行的子图。其他不被最终计算结果所连接的节点和边将被忽略，不参与后续运行过程。其中，该连接可以是直接与最终计算结果所在节点连接，也可以是通过若干条边与最终计算结果所在节点连接。

接下来，以每个设备上保存一部分子图进行说明。计算平台将提取出来的子图进行图切分，即切分为若干幅局部图，每幅局部图对应到一个设备。例如可以按照用户指定的设备分配策略进行切分。一幅子图上所有节点对应算法逻辑将由它所在的那个设备执行。需理解，图切分过程不会把子图上的一个节点切分到两个设备，但有可能把一条边切断。在这种情况下，计算平台会在切分后的局部图中，自动插入成对的数据发送操作节点（SendOp）和数据接收操作节点（RecvOp）。在通信操作的辅助下，被切分到不同设备上的几幅局部图的整体计算逻辑能够与切分之前的子图保持完全一致。可见，数据流图被切分后，要多进程地完成对这些子图的计算，就需要传输数据流图参数。

也就是说，数据流图中的各种数据和信息，例如数据流图参数，数据流图参数的信息，都可以存储在图数据结构中。

应理解，机器学习平台也可以将一数据流图的副本分发到多个设备，则无需进行子图的切分。这种情况下，同样需要传输数据流图参数，也同样会在数据流图中插入用于表示发送操作和接收操作的节点。由于数据流图中包括边和节点的信息，机器学习平台有多种在数据流图中插入用于表示发送操作和接收操作的节点的方法，本申请不做限制。为方便理解，下面结合图6，以图切分为例进行示意性说明。

如图6所示，图切分之前该数据流图包括节点a,b,c,w,y,x,以及由a指向b的边，由a指向c的边，由w指向y的边和由x指向y的边。图切分过程使得节点a指向b,a指向c,以及x指向y的边被切断，计算平台会在节点a、x所在子图中插入发送操作节点s1、s2，并在节点b、c、y所在的子图中插入接收操作节点r1、r2。这使得s1到r1之间和s2到r2之间分别建立了一对通信关系，保证了两幅局部图的整体计算逻辑同切分之前完全一致。

也就是说，图切分也是一种图分发，那么机器学习平台在进行图切分或者图分发的过程中，必然是已确定了与最终结果所在的节点相关的各个节点的分配方案，即该各个节点被分配在哪个设备上是一定的，因此也能够确定这些节点之间的边所承载的数据的源计算节点与目的计算节点。可见，而基于前文的描述以及图2，可知这些子图被分配给计算平台的多个进程执行，故子图中各个节点对应的进程也是明确的。基于这些信息，本申请提出一些实施例中可以在Tensor数据结构新增字段，以写入通信对端设备的内存类型和通信对端标识。

在一种实现方式下，进行图切分后，还可以进行图优化，即将切分后的子图进行优化处理，以达到在不改变其计算逻辑的前提下，能够使得数据流图在未来运行时的速率有所提升的目的。

以上提及的步骤都属于计算平台创建数据流图的步骤。

接下来计算平台运行数据流图，本步骤中，计算平台调度各个设备执行数据流图，得到算法的最终计算结果。

其中一种实现方式下，包括输入数据填充，即计算平台从存储设备中读取待计算的外部数据集，以将这些外部数据集填充到数据流图中的变量节点，使得计算节点中的运算符具有输入数据。然后，多个计算线程根据各自对应的设备中的子图，通过执行与各自子图相关的核函数对数据流

图进行计算。具体的，计算线程将设备对应的子图上的节点按照一定的调度策略进行排队，依次执行每个节点上的运算符所对应的核函数，以便得到算法的中间运算结果。其中各个节点的执行顺序由调度策略及运行时负载情况动态确定，常见的调度策略有全同步策略（Bulk Synchronous Parallel, BSP）、半同步策略例如 SSP（Stale Synchronous Parallel），以及异步（Asynchronous Parallel, ASP）策略，需要说明的是，在机器学习领域，并不要求多个计算线程计算的同步，因此大多数情况下，这类学习平台的计算都具有异步、随机的特征。

需说明的是，在这些待执行的节点中，有一类是通信操作节点，即此前图切分过程中所插入的数据发送操作节点（SendOp）和数据接收操作节点（RecvOp），如图5中的s1,s2,r1以及r2。也就是说本申请的方法所描述的使用MPI原语通信过程，就是计算平台在执行这些节点所代表数据发送或者数据接收的操作。例如，在现有的TensorFlow平台中，通信操作使用远程过程调用协议（Remote Procedure Call Protocol, RPC）机制实现。

最终，计算平台完成计算，从代表最终计算结果的节点中输出计算结果，返回给用户程序。

下面结合图7，以开源的TensorFlow机器学习平台为例，描述基于本申请描述的方法，实现数据流图参数通信的过程。但是应当理解，下述的数据流图参数过程的实施细节也适用于其他计算平台，并不对本申请造成限制。使用下述的通信过程，使得在TensorFlow这种常见的计算平台中，能简化MPI技术应用在计算数据流图中的通信过程，无需在数据传输前与通信对端协商对端信息，使用MPI接口函数库更加灵活和符合计算平台的计算特点，能更好地发挥高性能网络的通信能力。经测试，在下述例子的硬件环境中使用下述的方法，其通信效率能够提高50%，从而大大缩短了计算平台计算数据流图的时间。

需要说明的是下述过程中涉及的数据的发送方可以认为是上文提及的第一计算节点，接收方可以认为是上文提及的第二计算节点，发送方和接收方可以是相同的计算节点，或者不同的计算节点，可以部署在相同的物理机上，也可以部署在不同的物理机上，下述的数据流图参数通信过程均适用。

在一个例子中，该TensorFlow机器学习平台运行的服务器，配置有英伟达（NVIDIA）的GPU卡和InfiniBand网卡，该服务器使用的消息传递通信库为MPI接口函数。其中，NVIDIA的GPU卡通过CUDA编程接口提供计算加速能力，InfiniBand网卡通过RDMA协议提供高效通信能力。

如图7所示，在TensorFlow（5011）软件框架中，本实施例涉及到的模块包括分布式运行时（Distributed Runtime）模块5012，该模块为TensorFlow中的运行时引擎，具有本申请前文中描述的运行时引擎的功能，可执行相应的方法步骤；公共运行时（Common Runtime）模块5013，该模块实现TensorFlow中的内存管理，具有本申请前文中描述的内存管理模块的功能，可执行相应的方法步骤；以及远程会合点（Remote Rendezvous）模块5014，该模块实现TensorFlow的通信管理，具有本申请前文中描述的通信管理模块的功能，可执行相应的方法步骤。需理解的是，本实施例中所述的模块都是一段段代码，可以认为一个模块的代码是连续地编写在一起的。其中，Common Runtime模块5013中包括数据流图Graph5016。服务器中还包括主机内存5021、GPU内存5022和InfiniBand网卡5023。

图7中虚线框所示部分为本实施例在现有TensorFlow软件框架基础上进行的改进。在Distributed Runtime内部，本实施例添加了MPI调度的功能；在Common Runtime内部，本实施例添加了元信息管理的功能，用于管理数据流图参数的元信息，该元信息包括下文提及的数据流图参数的大小、名称和通信对端标识，在一种实现方式下，元信息还包括数据流图参数的存储位

置。管理可以是增加，删除，修改等操作中的至少一种；在Remote Rendezvous内部，本实施例添加了MPI客户端的功能；以及在TensorFlow平台中集成了MPI库；在TensorFlow使用的主机内存中，还分配了消息传递通信缓冲区（也就是下文描述的缓冲区），以供MPI库中的指令使用。可见，下述的这些改进都在TensorFlow平台的核心层，这就使得下述的改进，例如接口参数生成及接口调用过程隐藏在TensorFlow原有的数据流图创建和运行过程内部，而非暴露给应用程序开发者调用。作为一种集成在TensorFlow平台内部的改进机制，这些改进不改变TensorFlow原有的编程方式，能够实现既有应用程序的加速。

在数据流图的数据结构中，保存消息传递的通信所需的信息，具体可以是上文提及的数据流图参数的名称、大小和通信对端标识。本步骤在TensorFlow的Common Runtime模块5013创建数据流图时执行。TensorFlow现有的Graph5016模块使用Tensor等一系列数据结构保存连接边及其承载的数据流图参数。这些数据结构中已经包含了用于表示数据流图参数的名称（name）和数据流图参数的大小（size）的信息，这些数据结构是MPI原语要传输的数据。现有的Tensor数据结构中包括的信息可以满足现有的数据流图参数传输手段，即使用RPC通信。然而，现有的Tensor数据结构中不包括MPI的发送原语和接收原语需要携带的对端进程的信息，以及待通信参数的内存类型。本发明实施例中，在Tensor数据结构中增加数据流图参数的内存类型（例如表示为Dev type）和通信对端标识字段。需说明的是，对上述提及的信息，可以在Tensor数据结构中定义用于表示该信息的字段，例如定义用于存放内存类型的dev_type 字段以存放一数据流图参数的本端节点对应的内存类型，对发送原语，本端节点为源节点，对接收原语和接收端执行的等待原语，本端节点为目的节点。再例如，也可以定义该内存字段存放对端节点对应的内存类型，对发送原语，对端节点为目的节点，对接收原语和接收端执行的等待原语，对端节点为源节点。一种形式下，也可以定义多个字段以存放以数据流图参数的本端节点和对端节点分别对应内存类型。也可以是携带在Tensor数据结构中多个子数据结构中，需要将其中与上述信息相关的部分进行解析或者拼接或者计算，本发明实施例并不限定，计算平台提供一些辅助的手段去分析Tensor数据结构中的内容，以得到上述的4种信息。

25 其中，数据流图参数的大小就是该数据流图参数的数据量，也就是该数据流图参数所占的存储空间，单位为byte，记录待传输的数据流图参数的大小这一参数的数值，如3000，4000等。

在 **Common Runtime** 模块创建数据流图过程中，至少遍历与最终结果所在的节点有关的连接边（例如也可以遍历该数据流图中的所有连接边），对于连接边上承载的数据流图参数，基于数据流图切分的结果，获知用于表示连接边上承载的数据流图参数的内存类型的信息，将该用于表示连接边上承载的数据流图参数的内存类型的信息其写入 **Tensor** 数据结构，例如可以是填写在定义的内存类型字段，也可以是分布在多个字段中。一种实现方式下，内存类型指一数据流图参数的本端节点对应的内存类型，对发送原语，本端节点为源节点，对接收原语和接收端执行的等待原语，本端节点为目的节点。基于边的名称中包括的对端的设备的标识，获知该边上承载的参数所对应的对端的设备的标识符，将该对端的设备的标识符写入 **Tensor** 数据结构，具体可以是其中的通信对端标识字段。例如，**TensorFlow** 的数据流图中，连接边名称的格式为：

```
[src device];[src incarnation];[dst device];[tensor name];[frame id];[iter id]
```

其中，[dst_device]字段表示的就是该连接边的目的节点（即接收端）的设备标识符。
[src_device]字段表示的是该连接边的源节点（即发送端）的设备标识符。这些设备标识符往往是字符串。[src_device]也可以缩写为Src Dev； [dst_device]也可以缩写为Dst Dev。

以及,对内存类型字段,可以使用不同的枚举值标识不同的内存类型。例如用01标识主机内存,10标识GPU内存,或者用0标识主机内存,1标识GPU内存,或者用001标识主机内存,010标识GPU内存,100标识其他硬件内存等等,本申请不做限制。

在 TensorFlow 的 Remote Rendezvous 模块发起数据流图参数通信之前,使 MPI 接口函数携带上述的信息。这个步骤属于运行数据流图(即计算数据流图)的过程,可以认为是前文提及的 MPI 客户端的功能。需要注意的是,一种实现方式下,上述的信息需要经过处理才能被 MPI 接口函数携带。例如,数据流图参数的名称(name)、大小(size)的信息以及该数据流图参数所对应的通信对端标识,经处理后作为 MPI 原语的接口函数参数携带,而数据流图参数的内存类型作为 MPI 接口函数携带的数据中的一部分进行传输。这样,就可以使用现有的各种通用的 MPI 接口函数,大大提高了通用性。再例如,也可以将待传输的数据流图参数的内存类型也作为 MPI 接口函数的函数参数。当然,这种情况下,需要修改 MPI 接口函数的定义和使用规范,本申请对此方面不详细叙述。

为便于描述,以 MPI 发送原语为例进行说明。对 MPI 接收原语,可参考对 MPI 发送原语的描述,即 MPI 接收原语的接口函数参数中也携带一数据流图参数的消息大小字段,消息标记字段、和目标进程序列号字段。一种实现方式下, MPI 接收原语携带的数据中包括数据流图参数的目的地址所对应的内存类型。此处不再赘述。 MPI 发送原语的接口函数参数包括消息大小字段,该消息大小字段用于表示该发送原语要发送的信息的大小,则一种实现方式下,可使该消息大小字段的值等于上述数据流图参数的大小的参数值,即 size。另一种实现方式下,可以使该消息大小字段的值等于上述的数据流图参数的大小的参数值加上一个值。这个加上的值是发送原语要携带的其他信息的大小,如下文提及的该发送原语的包头长度。一种实现方式下,包头包括该数据流图参数的大小、标识(用于标记该数据流图参数,如其名称)、对应的目标进程的编号以及对应的源进程编号。 MPI 发送原语中携带的消息标记(tag)用于指示该 MPI 发送原语中携带的数据,例如用于表示数据流图参数的名称。 MPI 发送原语中携带的消息标记为一固定长度的二进制数值,故可以将数据流图参数的名称使用一些算法,转换成符合消息标记的格式,将其值作为 MPI 发送原语中消息标记这一参数的值,例如算法可以是哈希函数。应理解,由于 MPI 接口函数的校验机制,可以避免哈希碰撞的影响。

又一方面,执行该 MPI 发送原语的主机从进程映射表中,根据上文中描述的数据流图参数的通信对端标识字段,查找到对端进程序列号,即 rank,对端进程为执行与该 MPI 发送原语对应的 MPI 接收原语的进程。例如,进程序列号可以是一些编号,例如 0,1,2,3,28 等。上述进程映射表,包括该计算平台中设备的标识符与使用该设备的进程序列号间的映射关系。一数据流图参数的通信对端标识字段中保存着的是该数据流图参数的接收端的设备的标识符。需要理解的是,在机器平台运行后,调用一设备的进程一旦拉起,在执行完计算任务前,该进程与该设备的对应关系都不会改变,该进程可以获知其调用的设备的标识符,故,机器学习平台可以生成该进程映射表。例如,可以在进程拉起过程中,通过某些函数将该机器学习平台中的设备的标识符做变换以得到序列号,该变换所得到的序列号就作为设备对应的进程的序列号。也可以在进程拉起后,记录进程的序列号与进程调用的设备的标识符的映射关系。也可以在生成三元组时,也可以对通信对端的标识使用某种函数进行处理,以得到所需的进程的序列号。应理解,这些进程序列号是供 MPI 库中的原语使用的,该机器学习平台还可以保存该 MPI 库中的进程序列号与进程所在物理机的信息之间的映射关系,每个进程的序列号都不同。

当然，一种实现方式下，一种实现方式下，内存类型不在MPI接口参数中携带，而是在MPI传输的数据中携带。例如将内存类型序列化成为字节流，作为Tensor数据结构的一个字段。显然，MPI原语传输到通信对端进程，对端进程解析接收到的数据，即可获知该接收到的数据流图参数对应的目的设备的内存类型。

5 以上，无需传输数据的双方协商，发送方和接收方的程序代码独立运行，在双方无交互的情况下，就得到了MPI接口函数库中，发送原语和接收原语所需的函数参数三元组<rank,size,tag>。其中，rank为通信对端的进程序列号，size为要传输的信息的大小，tag为消息标记。

需说明的是，MPI接收原语的函数参数中，包括待传输的数据（如数据流图参数）的目的地址，也就是说，MPI发送原语中只携带对端设备的信息，但并不包括待传输的数据的目的地址。

10 因此，待传输数据的接收端的可以接收发送端设备传来的数据（例如下文所述存放在接收端的主机内存中），但要在接收侧的进程调用MPI接收原语过程中，甚至调用MPI接收原语后，才能将接收的数据用于训练数据流图。

例如以图6中的切分的数据流图为例，源节点S2与目的节点R2间需进行跨设备通信。其中，节点R2位于设备B，携带有以下信息：Name:Edge_x_y_0,size:4000,Src Dev:Dev A, Dev Type:CUDA_GPU。其中Dev Type:CUDA_GPU表示对端设备即设备A的内存类型。节点S2位于设备A，携带有以下信息：Name:Edge_x_y_0,size:4000,Dst Dev:Dev B, Dev Type:CUDA_GPU。其中Dev Type:CUDA_GPU表示对端设备即设备B的内存类型。则源节点S2的发送原语，可写为MPI_Isend(tag="hash" Edge_x_y_0",size=4000+LEN_HEADER,rank=dev2rank("Dev B")),即该发送原语中携带上文中提及的三元组参数，而Dev

15 Type:CUDA_GPU，则携带在该发送原语的数据部分，该发送原语的数据部分还可以包括该源节点所在的设备的名称，即Dev A。目的节点R2的接收原语携带接口信息，可写为MPI_Irecv(Tag="hash" Edge_x_y_0",size=4000+LEN_HEADER,rank=dev2rank("Dev A")),即该接收原语中携带上文中提及的三元组参数，而Dev Type:CUDA_GPU，则携带在该接收原语的数据部分，该接收原语的数据部分还可以包括目的节点所在的设备的名称，即Dev B。

25 例子中，size中的4000表示该MPI原语携带的数据流图参数的大小，LEN_HEADER，表示前文描述的包头的长度（length of header）。MPI原语传输的数据是序列化（Serialization）之后的Tensor数据结构，即一组字节。Tensor数据结构中除了包含待传输的数据流图参数，也包含一些其他的信息字段，这些信息字段序列化之后称为“包头”。例如该Tensor数据结构的名称。包头的长度是确定的，所以可以加一个常数。

30 这样，在数据流图计算的过程中，发送端与接收端无需进行交互就可以得到对端设备的标识符等生成MPI接口函数的参数，减少了各进程间的通信次数和等待。

下面继续以进程发送数据流图参数的过程进行说明。由上文可知，执行MPI发送原语的进程需先获得该MPI发送原语携带的待发送的数据流图参数。该过程可以这样描述：确定待发送的数据流图参数是否位于主机内存，在该待发送的数据流图参数位于其他存储设备的情况下，将该数据复制到主机内存。例如，该其他存储设备可以是该主机除主机内存外的其他内存，如GPU内存等。这是由于对于一般的MPI接口函数，只能直接使用主机内存中的数据。

一种实现方式下，主机内存中包括MPI缓冲区，该MPI缓冲区是分配给MPI库使用的内存空间。对某些MPI库这是自带的，用以存储通过MPI原语调用的数据，如该内存空间可以是64KB。可选的，MPI缓冲区也可以是分配给某用户的内存空间，即可以复用。显然MPI缓冲区空间较小，虽

然能一定程度上打破通过MPI库的通信的同步性,例如可以接收64KB以下的信息,但很容易用尽,不能满足机器学习场景的需要,例如,几百KB以及几MB的数据流图参数都较为常见。本申请还提出了另一种实现方式,在计算平台所在的主机的内存中,再设置一个数据缓存区,该数据缓存区的地址空间大于MPI缓冲区,该数据缓存区专用于存储通过MPI原语调用的数据。例如,可以在内存中设置几MB或者十几MB的数据缓存区,甚至可以在主机的内存中设置上百MB甚至几个GB的数据缓存区。需理解的是,对部署有机器学习平台的物理机,硬件上配置有较大的内存空间,可以满足前述的内存分配需求。这样,数据缓存区与MPI缓冲区配合使用,扩充了MPI缓冲区的容量,使得处理MPI原语所使用的数据的能力进一步增强。以及,可以配合下文的轮询线程使用,加快接收端对主机内存中的发送来的数据的处理,从而加快MPI发送原语的执行,以加快机器学习平台中的数据交互。为了方便描述,上述的MPI缓冲区和数据缓存区,总称为缓冲区,其共同点为,都专用于存储通过MPI原语调用的数据。

另一方面,在MPI技术使用RDMA通信机制的情况下,可以远程将数据写入一主机的内存而该主机的CPU而不感知,也就是说,在需要接收数据的主机还未执行MPI接收原语时,发送数据的主机就可以通过MPI发送原语将数据发送至需要接收数据的主机。且,MPI发送原语中不携带传输的数据的目的地址,只有MPI接收原语中才携带,则只有对端执行MPI接收原语,才能将接收的数据传送到目的地址,未执行MPI接收原语时,接收的数据会先存放入这些缓冲区。这样一来,如果需要接收数据的主机中,缓冲区中空闲的空间不足以放置将要发送来的数据,携带该将要发送来的数据的MPI发送原语将无法执行。这也就是说,对MPI技术,是存在同步、有序约束的,即发送端为了顺畅地持续执行MPI发送原语,需要接收端在接收到数据后,尽快执行与接收到的数据对应的MPI接收原语。这样一来,增加上文所述的数据缓存区能更大程度的允许异步、乱序的收发操作,无需增加额外的消息同步机制,从而满足TensorFlow这类计算平台的需求。需要同步、有序约束的MPI库可以对接到具有异步、乱序特征的TensorFlow在数据流图通信过程,有助于提升机器学习平台的参数通信的性能。

在TensorFlow平台中,上述的发送端进程获得待传输的数据流图参数的步骤,可以认为是Remote Rendezvous模块执行MPI发送原语(例如MPI_Send或MPI_Isend)之前执行的。例如,Remote Rendezvous模块读取待通信的数据流图参数的内存类型字段,判断其是否位于主机内存地址空间。如果是,则结束本步骤。如果不是,例如该待通信的数据流图参数位于GPU内存,则执行CUDA编程接口提供的cudaMemcpy函数,将该待通信的数据流图参数从GPU内存复制到主机内存中。这样,无论选用的MPI库是否支持访问GPU,都可以在机器学习平台中使用,无需通过MPI接口函数访问GPU内存,从而MPI库的选择范围更大,也大大缓解了前文提及的百度的方案中访问GPU的抢占资源问题。以及,由于执行该步骤的是TensorFlow平台中的Remote Rendezvous模块,属于TensorFlow平台的核心层,则,无需在一线程要执行MPI发送原语或者MPI接收原语的整个过程都对GPU内存加上该进程的锁,而是只需将上文提及的待发送的数据从GPU内存复制到主机内存中这一步加锁即可,减少了其他线程的等待时间大大减小了不同进程在访问GPU内存的锁竞争。

在跨设备部署的数据流图的计算过程中,显然包括数据流图参数的接收与发送,可以认为是上文中提及的MPI客户端的功能。被传输的数据流图参数在发送前和接收(这个接收指的是数据流图参数在接收端被MPI接收原语处理)前,都将被存储在相应的主机内存中的缓冲区,该缓冲区也就是专为MPI库分配的内存中的地址段,可以是前文提及的MPI缓冲区,还可以是数据缓存

区。

下面描述数据流图参数的接收与发送。一种实现方式下，检测数据流图中是否有数据流图参数需要进行跨设备通信。如果有，则确定待执行的一通信操作是数据发送操作还是数据接收操作，如果是数据发送操作，使用MPI_Send或MPI_Isend原语发送数据（即数据流图参数）；如果是数据接收操作，使用MPI_Recv或MPI_Irecv原语接收数据；之后，Remote Rendezvous模块将接收到的数据作为数据流图参数使用。一次发送或接收操作结束后，再次检测是否还有参数需要进行跨设备通信，如此循环执行。该循环过程可以是一个物理机运行的多个线程执行的，这些线程通过一些调度机制控制，从而依调度机制执行不同指令以完成不同的操作，例如发送数据和接收数据是线程可被动态调度执行的操作中的两种。例如该调度机制中定义多种事件发生后执行的原语，也就是发生什么事件触发什么原语的执行。比如上文提及的，如检测到本主机负责计算的数据流图中还有数据流图参数需要进行跨设备通信，则根据操作类型执行MPI发送原语或者MPI接收原语。机器学习平台往往是异步的，乱序的，这种方式在机器学习平台中比较常见。

需要说明的是，MPI接收原语还用于对缓冲区中的数据进行处理以使得机器学习平台中用于计算的进程或者线程使用该缓冲区的数据。例如，可以是对该数据的元信息进行处理（例如确认该数据的状态，确认待接收的数据中，哪个数据被接收了），也可以是对该数据的同步处理，例如告知机器学习平台中用于计算的进程或者线程该数据已准备就绪，还可以是将数据存储到目的地址。不同的MPI库，执行MPI接收原语包括的程序实现可能各不相同。

由于上述的一个线程一次只能执行一条MPI接收原语，而分布式部署的机器学习平台涉及多台物理机的交互，故一部署了机器学习平台的物理机可能会在短时间内接收多条MPI发送原语传输的数据，则上述的线程可能无法及时地通过MPI接收原语处理发送到物理机的主机内存中的数据。另一种实现方式下，部署机器学习平台的物理机可以分配一专用的线程，该线程专用于探测通过MPI发送原语发送来的数据，以及接收探测到的数据。该专用的线程属于一种专用的线程，不需前文所述的调度机制控制。例如，可以是前文描述的Distributed Runtime模块配合Remote Rendezvous模块完成的。这样就可以提高通过MPI接收原语接收信息的实时性，也可以减少发送端程序上，其余的MPI发送原语等待被执行的时间。

下面描述该专用的线程运作的方法：采用轮询的方式循环探测该主机的内存中是否有发送来的数据的三元组信息，以及处理探测到的三元组对应的数据，以便加快对主机内存中的待MPI接收原语处理的数据的处理。为描述方便，这个线程可称为轮询线程。具体的，轮询线程轮询主机内存中的缓冲区，即上文提及的专用于存储通过MPI原语调用的数据的存储空间，如轮询MPI缓冲区；或者在主机内存中包括MPI缓冲区与数据缓存区的情况下，轮询MPI缓冲区与数据缓存区。该过程也可以认为是前文描述的MPI调度的功能，实现在作为数据流图参数的接收端的主机内。下面描述该轮询过程中一轮的流程。轮询线程调用MPI库的MPI_Probe或MPI_Iprobe等探测原语，以探测主机内存中是否有发送来的数据流图参数或者数据流图参数对应的三元组在等待相应的MPI接收原语处理，也就是说该发送来的数据流图参数对应的MPI接收原语还未被执行，如果没有，则继续执行探测原语轮询主机内存中的缓冲区；如果有，则调用与该检测到的数据流图参数对应的MPI_Irecv原语，这样就可以将待接收的数据流图参数接收到本地内存中。其中，该探测线程根据该检测的目的计算节点侧对应的三元组<rank,size,tag>，从而确定出用于处理该数据流图参数的MPI_Irecv原语的接口参数。然后，将原本待其他线程执行的执行策略中与该数据流图参数对应的MPI_Recv或MPI_Irecv原语变更为MPI_Wait原语，以等待轮询线程通过MPI_Irecv

原语完成对该数据流图参数的处理（例如放置到目的地址对应的存储空间）。到轮询线程通过MPI_Irecv原语完成对该数据流图参数的处理，则本轮轮询结束，轮询线程继续轮询主机内存中的缓冲区，以探测主机内存中是否有发送来的数据流图参数在等待相应的MPI接收原语处理。事实上，在该轮询线程的MPI_Irecv原语开始执行时，会向在原本待其他线程执行的执行策略中的接收原语发送返回信息以触发该接收原语变更为MPI_Wait原语，如返回一个MPI请求

（MPI_Request）对象，该对象中可包括在探测到的三元组，这样就可以根据对象调用未被执行的原本用于处理该三元对应的数据的MPI_Wait原语。由此，可以更快地处理已写入到接收端主机内存的缓冲区的数据，这样就可以将接收端主机内存的缓冲区中被已接收的数据占用的空间空闲出来，更快地写入来自发送端的其他数据，也就是说，发送端可以减少执行其他发送原语前的等待时间，从而接收端能在更短的时间内处理更多的缓冲区中的数据，而发送端也能在更短的时间内执行更多的发送原语。

在一种实现方式下，可以以无限循环的方式不断探测。比如该线程运行到整个计算数据流图的进程完成计算。

下面简单描述探测线程是如何探测到主机内存中待MPI接收原语处理的数据的。MPI库中，有一些数据结构做相应的记录，如果一数据尚未被MPI接收原语处理，那么它就会保存在一个数据结构中，MPI探测原语就可以探测到该记录从而确定该记录对应的数据未被MPI接收原语处理。如果一数据已经被MPI接收原语接收或者正在被MPI原语接收，则该数据对应的记录就会从前述的数据结构从清除，从而无法被MPI探测原语探测到。在一种实现方式下，上述的MPI接收原语（例如MPI_Recv）和MPI等待原语（例如MPI_Wait）可以认为是TensorFlow平台中的Remote Rendezvous模块执行的。

通常，这些数据流图参数被通过MPI接收原语提供给接收端用于计算的线程或者进程使用以进行机器学习的计算，例如MPI接收原语将数据流图参数放置在主机内存中属于一用户的存储空间，该用户为进行该机器学习的用户。另一方面，在确定该数据流图参数的目的地址不在主机内存的情况下，将该数据流图参数的副本存储到目的地址对应的设备中。本步骤也可以认为是Remote Rendezvous模块中MPI客户端功能的一部分。以目的地址在GPU内存为例，则执行CUDA编程接口提供的cudaMemcpy函数，将已接收的数据复制到GPU内存中。

由于使用MPI库的原语，数据在写入目的地址前先写入主机内存的缓冲区，而在数据流图计算中，很多数据流图参数的目的地址是GPU等其他设备，这样，接收完数据使用计算平台中的CUDA编程接口写入GPU，无需使用的MPI库支持访问GPU内存，大大扩展了可使用的MPI库的种类，同样，也大大缓解了前文提及的百度的方案中，访问GPU的抢占资源问题。以及，由于执行该步骤的是TensorFlow平台中的Remote Rendezvous模块，属于TensorFlow平台的核心层，则，无需在一进程要执行MPI发送原语或者MPI接收原语的整个过程都对GPU内存加上该进程的锁，而是只需将上文提及的待发送的数据从GPU内存复制到主机内存中这一步加锁即可，减少了其他线程的等待时间大大减小了不同进程在访问GPU内存的锁竞争。

图5到图7对应的方法，可以在图7和图3所示的系统和服务中运行。

综上，本申请提出的数据的传输方法，无需在数据传输前与通信对端协商对端信息，改善了MPI库的同步、有序约束与数据流图通信异步、乱序的矛盾，又改善了MPI库与计算平台对访问GPU的资源抢占问题，使得MPI技术能够更好地与分布式部署的计算平台适配，可以充分利用网络传输资源，提高机器学习平台中数据传输的效率，从而提高机器学习平台的业务处理速度。

另一方面，本发明实施例提供一种如图 8 所示的分布式计算系统中的数据传输装置。该分布式计算系统包括第一计算节点和第二计算节点，其中，该数据传输装置位于所述第一计算节点，该数据传输装置包括：确定模块 801，该确定模块 801 用于从所述第一计算节点中的第一图数据结构中，确定第一数据流图的第一数据流图参数的名称、大小和通信对端标识，其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数，所述通信对端标识对应所述第二计算节点。

生成模块 802，所述生成模块 802 用于根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组，所述第一三元组包括消息标记、消息大小和目的进程序列号，其中，所述消息标记对应于所述第一数据流图参数的名称，所述消息大小对应于所述第一数据流图参数的大小，所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程；

通信模块 803，所述通信模块 803 用于以所述第一三元组作为接口参数，调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数，以便于所述第二计算节点以与所述第一三元组对应的第二三元组作为接口参数，调用 MPI 接收原语处理所述第一数据流图参数，所述第二三元组是根据所述第二计算节点中的第二图数据结构，利用第二接口参数生成算法生成的，所述第二接口参数生成算法与所述第一接口生成算法相同。

一种实现方式下，在以所述第一三元组作为接口参数，调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数的方面，所述通信模块 803 用于以所述第一三元组作为接口参数，通过消息传递接口 MPI 发送原语从所述第一计算节点的主机内存中读取所述第一数据流图参数，以向所述第二计算节点发送所述第一数据流图参数。

一种实现方式下，所述第一计算节点还保存有所述第一数据流图参数所在的存储设备的信息，所述第一计算节点还包括读取模块 804，所述读取模块 804 用于在所述存储设备的信息指示为其他存储设备的情况下，将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存，所述其他存储设备为所述第一计算节点中除主机内存外的内存。

一种实现方式下，所述第一接口参数生成算法包括第一算法、第二算法和第三算法，在根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组的方面，该生成模块 801 用于根据所述第一图数据结构中的第一数据流图参数的名称和所述第一算法，确定所述第一三元组中的消息标记，根据所述第一图数据结构中的第一数据流图参数的大小和所述第二算法，确定所述第一三元组中的消息大小，以及根据第一图数据结构中的第一数据流图参数的通信对端标识和所述第三算法，确定所述第一三元组中的目的进程序列号。

可见，上述的实现方式中，图 8 所示的数据传输装置是作为数据传输中的发送端的。在其他一些实现方式中，图 8 所示的数据传输装置是可以执行与发送端相应的操作，以作为数据传输中的接收端。也就是说，在一些情况下，图 8 所示的数据传输装置可以具有发送端和接收端的功能，或者说，图 8 所示的数据传输装置在一些数据的传输中是发送端，在另一些数据的传输中是接收端。

下面描述图 8 所示的分布式计算系统中的数据传输装置作为数据接收方的实现方式。该分布式计算系统包括第一计算节点和第二计算节点，所述数据传输装置位于所述第二计算节点，其中，该数据传输装置的确定模块 801，该确定模块 801 用于从所述第二计算节点的第二图数据结构中，

确定第二数据流图中的第一数据流图参数的名称、大小和通信对端标识,所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点。

生成模块 802,该生成模块 802 用于根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第二接口参数生成算法生成第二三元组,所述第二三元组包括消息标记、消息大小和源进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程。

通信模块 803,该通信模块 803 用于根据所述第二三元组,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数,所述第一数据流图参数是所述第一计算节点通过 MPI 发送原语发送来的,所述 MPI 发送原语的接口参数包括与所述第二三元组对应的第一三元组,所述第一三元组是所述第一计算节点根据所述第一计算节点中的第一图数据结构,利用第一接口参数生成算法生成的,所述第二接口参数生成算法与所述第一接口生成算法相同。

一种实现方式下,该通信模块 803 包括第一线程和第二线程,所述第二计算节点的主机内存中包括数据缓存区,所述数据缓存区专用于存储被 MPI 原语处理的数据,在以所述第二三元组作为接口参数,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数的方面,所述第一线程用于通过消息传递接口 MPI 探测原语检测所述主机内存中的数据缓存区,以获得第二三元组;所述第一线程用于根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语,以处理所述第一数据流图参数,所述数据缓存区中的第二三元组是所述第二计算节点根据所述 MPI 发送原语得到的;所述第二线程用于在确定所述第一数据流图参数被所述第一 MPI 接收原语处理后,将第二 MPI 接收原语修改为 MPI 等待原语,所述第二 MPI 接收原语为未被所述第二线程执行的与所述第一数据流图参数对应的接收原语,所述第二 MPI 接收原语的接口参数包括所述第二计算节点生成的第二三元组,所述 MPI 等待原语用于等待所述第一 MPI 接收原语执行完毕。

这样,就可以加快处理数据缓存区中三元组所指示的已经准备好的数据流图参数,加快接收端处理接收到的数据的速度,从而加快发送端执行发送原语的速度,另外,数据缓存区也增强了 MPI 原语对乱序、异步收发操作的适应能力,能够更好地适应计算平台中数据传输的特点。

在上述实现方式下,在根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语,以处理所述第一数据流图参数的方面,该通信模块 803 中的第一线程用于,在所述第一数据流图参数的目的地址对应所述第二计算节点的主机内存中分配给用户使用的内存空间的情况下,以所述数据缓存区中的第二三元组为所述第一 MPI 接收原语的接口参数,调用所述第一 MPI 接收原语,以将所述第一数据流图参数从所述数据缓存区存储到所述第一数据流图参数的目的地址。

一种实现方式下,所述数据传输装置还包括存储模块 805,所述存储模块 805 用于在所述第一数据流图参数的目的地址对应其他存储设备的情况下,将所述主机内存中的第一数据流图参数存储到所述目的地址,所述其他存储设备为所述第二计算节点中除主机内存之外的内存。

一种实现方式下,该第二接口参数生成算法包括第一算法、第二算法和第三算法,在根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第二接口参数生成算法生成第二三元组的方面,所述生成模块 802 用于根据所述第二图数据结构中的第一数据流图参数的名称和所述第二接口参数生成算法中的第一算法,确定所述第二三元组中的消息标记;根据所述第二图数据结构中的第一数据流图参数的大小和所述第二接口参数生成算法中的第二算

法，确定所述第二三元组中的消息大小；以及根据第二图数据结构中的第一数据流图参数的通信
对端标识和所述第二接口参数生成算法中的第三算法，确定所述第二三元组中的源进程序列号。

可见，图 8 所对应的数据传输装置在一些情况下对应前文的第一计算节点和第二计算节点，
可以执行前文如图 5 到图 7 所述的方法中发送方或接收方的方法。则，对图 8 所对应的数据传输
5 装置中涉及的各种解释说明以及各模块执行的步骤的有益效果，请参见前文的相应段落，此处不
再赘述。这样，能简化 MPI 技术应用在计算数据流图中的通信过程，无需在数据传输前与通信对
端协商对端信息，使得 MPI 技术能够更好地与分布式部署的计算平台适配，从而提高分布式计算
系统中数据传输的效率，从而提升对分布式计算系统数据流图的计算效率。

还需要说明的是，图 8 中的确定模块 801、生成模块 802 和通信模块 803，可以是不同的代
10 码或者运行代码的进程或者线程，图 8 所示的划分方式仅是一种示例，在一些实现方式中，这些
模块可能采用别的方式命名或者划分，例如某些模块是一个模块。确定模块 801 可对应于上文中的
的内存管理 3014 模块或者 TensorFlow 平台中的 Common Runtime5013 模块；生成模块 802
和通信模块 803 可对应于上文中的通信管理 3015 模块或者 TensorFlow 平台中的 Remote
Rendezvous5014 模块。而在通信模块 803 包括第一线程和第二线程的情况下，可以认为第一
15 线程属于上文中提及的运行引擎 3013 模块或者 TensorFlow 平台中的 Distributed Runtime
5012 模块，而第二线程属于通信管理模块或者 TensorFlow 平台中的 Remote Rendezvous5014
模块。故，图 8 所示的数据传输装置可以实现前文所述的模块的功能，具体的实施方式请参见前
文的描述。

通过以上的实施方式的描述，所属领域的技术人员可以清楚地了解到，为描述的方便和简洁，
20 仅以上述各功能模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同
的功能模块完成，即将装置的内部结构划分成不同的功能模块，以完成以上描述的全部或者部分
功能。上述描述的装置和单元的具体工作过程，可以参考前述方法实施例中的对应过程，在此不
再赘述。

另一方面，如图 9 所示，图 9 示出了本发明实施例提供的一种物理机的示意性框图，本发明
25 实施例提供一种物理机，该物理机包括处理器 40 和存储器 42，该存储器 42 是存储可执行代码
的非瞬态计算机可读介质。该物理机可以是前文所述的第一计算节点或者第二计算节点。分布式
计算平台通过存储器 42 中的程序运行在处理器 40 上。

该物理机可以通过处理器 40 执行存储器 42 中的可执行代码，以执行前文所述的各种方法。
显然，图 9 是比图 3 或者图 7 所示的能够运行本申请中各种方法的服务器更简单的一种表达形式。
30 图 8 所示的数据传输装置可以运行在如图 9 或者图 3 或者图 7 所示的架构中。

在一种实现方式下，该存储可执行代码的非瞬态计算机可读介质为存储器 42，该物理机还包
括接口电路 41 和系统总线 43。

处理器 40 可以通过多个处理器实现。处理器 40 可以为中央处理器（英文：central processing
unit，缩写：CPU）。处理器 40 还可以为其他通用处理器、图形处理器（英文：Graphics Processing
35 Unit，缩写：GPU）数字信号处理器（英文：digital signal processing，缩写：DSP）、专用集
成电路（英文：application specific integrated circuit，缩写：ASIC）、现场可编程门阵列（英
文：field-programmable gate array，缩写：FPGA）或者其他可编程逻辑器件、分立门或者晶体
管逻辑器件、分立硬件组件等。通用处理器可以是微处理器或者该处理器也可以是任何常规的处
理器等。

例如图 5 到图 7 中的物理机以及对应的实施例,以包括 CPU 和 GPU 在内的物理机为例进行了描述。其中,若处理器中包括 GPU,则 GPU 和 GPU 内存通常封装在同一芯片,即处理器 40 中可包括某些处理器的内存。

接口电路 41 具体可以是物理机上硬件的通信接口。该通信接口可以为无线通信接口,还可以包括天线等射频电路。例如,无线通信接口可以是物理机的无线模块等。处理器 40 通过接口电路 41 与其他设备,例如其他物理机之间进行数据的收发。例如图 3 和图 7 中的物理机所示的网卡 3024 或者 InfiniBand 网卡 5023 属于通信接口的一种实现方式。

存储器 42 可以包括易失性存储器(英文:volatile memory),例如随机存取存储器(英文:random-access memory,缩写:RAM),内存等;存储器 42 还可以包括非易失性存储器(英文:non-volatile memory),例如只读存储器(英文:read-only memory,缩写:ROM),快闪存储器(英文:flash memory),硬盘(英文:hard disk drive,缩写:HDD)或固态硬盘(英文:solid-state drive,缩写:SSD);存储器 42 还可以包括上述种类的存储器的组合。存储器 42 可以有多个以供上述的多个处理器 40 使用,例如前述实施例中描述的主机内存,GPU 内存等。

存储器 42 可以包括底层存储介质和内存。其中,内存耦合至底层存储介质,用于作为底层存储介质的缓存。

系统总线 43 可以包括数据总线、电源总线、控制总线和信号状态总线等。系统总线 43 用于将上述的处理器 40,存储器 42 以及接口电路 41 连接起来。本实施例中为了清楚说明,在图 9 中将各种总线都示意为系统总线 43。

本发明实施例提供的物理机,可以用于执行本申请记载的任一种方法,如图 5 至图 7 对应的第一计算节点或者第二计算节点执行的任一方法,则,对图 9 所对应的物理机中涉及的对方法的描述、方法中涉及的名词的解釋说明以及各步骤的有益效果,请参见前文的相应段落,此处不再赘述。这样,能简化 MPI 技术应用在计算数据流图中的通信过程,无需在数据传输前与通信对端协商对端信息,使得 MPI 技术能够更好地与分布式部署的计算平台适配,从而提高分布式计算系统中数据传输的效率,从而提升对分布式计算系统数据流图的计算效率。

可选的,本实施例还提供存储有可执行程序的非瞬态计算机可读介质,可执行程序包括用于执行本申请描述的任一种方法。该非瞬态计算机可读介质可安装于物理机,当物理机运行时,物理机的处理器执行该计算机执行指令,以使物理机执行本申请描述的任一种方法。则,对上述非瞬态计算机可读介质中保存的方法的描述、有关该方法中各种名词的解释说明以及各步骤的有益效果,请参见前文的相应段落,此处不再赘述。

可选的,本实施例中的非瞬态计算机可读介质可以为上述如图 9 所示的存储器 42。

在本申请所提供的几个实施例中,应该理解到,所揭露的系统,装置和方法,可以通过其它的方式实现。例如,以上所描述的装置实施例仅仅是示意性的,例如,所述模块或单元的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,例如多个单元或组件可以结合或者可以集成到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口,装置或单元的间接耦合或通信连接,可以是电性,机械或其它的形式。

所述作为分离部件说明的单元可以是或者也可以不是物理上分开的,作为单元显示的部件可以是或者也可以不是物理单元,即可以位于一个地方,或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

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。

5 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，可以存储在一个计算机可读取存储介质中。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介质中，包括若干指令用以使得一台计算机设备（可以是个人计算机，服务器，或者网络设备等）或处理器执行本发明各个实施例所述方法的全部或部分步骤。而前述的存储介质包括：U 盘、移动硬盘、ROM、RAM、磁碟或者光盘等各种可以存储程序代码的介质。

10 以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应以所述权利要求的保护范围为准。

所属领域的技术人员可以清楚地了解到，为描述的方便和简洁，仅以上述各功能模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同的功能模块完成，即将装置的内部结构划分成不同的功能模块，以完成以上描述的全部或者部分功能。上述描述的系统，装置和单元的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

20 在本申请所提供的几个实施例中，应该理解到，所揭露的系统，装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如，所述模块或单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，装置或单元的间接耦合或通信连接。

所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

25 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。上述集成的单元可以采用软件功能单元的形式实现。

30 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，可以存储在一个计算机可读取存储介质中。基于这样的理解，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介质中，包括若干指令用以使得一台计算机设备（可以是个人计算机，服务器，或者网络设备等）或处理器执行本发明各个实施例所述方法的全部或部分步骤。所述存储介质是非短暂性（英文：non-transitory）介质，包括：快闪存储器、移动硬盘、只读存储器、随机存取存储器、磁碟或者光盘等各种可以存储程序代码的介质。

35 以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应所述以权利要求的保护范围为准。

1、一种分布式计算系统，所述分布式计算系统包括第一计算节点和第二计算节点，其特征在于，所述第一计算节点上的第一图数据结构保存有第一数据流图中的第一数据流图参数的名称、大小和通信对端标识，其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数；所述第二计算节点上的第二图数据结构保存有第二数据流图中的所述第一数据流图参数的名称、大小和通信对端标识；所述第一数据流图中的第一数据流图参数的通信对端标识对应于所述第二计算节点，所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点；

所述第一计算节点用于，根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组，所述第一三元组包括消息标记、消息大小和目的进程序列号，其中，所述消息标记对应于所述第一数据流图参数的名称，所述消息大小对应于所述第一数据流图参数的大小，所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程；

所述第二计算节点用于，根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第二接口参数生成算法生成第二三元组，所述第二接口参数生成算法与所述第一接口参数生成算法相同，所述第二三元组包括所述消息标记、所述消息大小和源进程序列号，其中，所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程；

所述第一计算节点用于，以所述第一三元组作为接口参数，调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数；

所述第二计算节点用于，根据所述第二三元组，调用 MPI 接收原语处理所述第一数据流图参数。

2.根据权利要求 1 所述的系统，其特征在于，在以所述第一三元组作为接口参数，调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数的方面第一数据流图参数，所述第一计算节点用于以所述第一三元组作为接口参数，通过消息传递接口 MPI 发送原语从所述主机内存中读取所述第一数据流图参数，以向所述第二计算节点发送所述第一数据流图参数。

3. 根据权利要求 2 所述的系统，其特征在于，所述第一计算节点还保存有所述第一数据流图参数所在的存储设备的信息，第一数据流图参数所述第一计算节点还用于在所述存储设备的信息指示为其他存储设备的情况下，将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存，所述其他存储设备为所述第一计算节点中除主机内存外的内存。

4.根据权利要求 1 到 3 任一权利要求所述的系统，其特征在于，所述第一接口参数生成算法包括第一算法、第二算法和第三算法，在根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第一接口参数生成算法生成第一三元组第一数据流图参数的方面，所述第一计算节点用于：

根据所述第一图数据结构中的第一数据流图参数的名称第一数据流图参数和所述第一算法，确定所述第一三元组中的消息标记，根据所述第一图数据结构中的第一数据流图参数第一数据流图参数的大小和所述第二算法，确定所述第一三元组中的消息大小，以及根据第一图数据结构中的第一数据流图参数的通信对端标识和所述第三算法，确定所述第一三元组中的目的进程序列号；

相应的, 在根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识, 利用第二接口参数生成算法生成第二三元组第一数据流图参数的方面, 所述第二计算节点用于:

根据所述第二图数据结构中的第一数据流图参数的名称和所述第二接口参数生成算法中的第一算法, 确定所述第二三元组中的消息标记; 根据所述第二图数据结构中的第一数据流图参数的大小和所述第二接口参数生成算法中的第二算法, 确定所述第二三元组中的消息大小; 以及根据第二图数据结构中的第一数据流图参数的通信对端标识和所述第二接口参数生成算法中的第三算法, 确定所述第二三元组中的源进程序列号。

5. 根据权利要求 1 到 4 任一权利要求所述的系统, 其特征在于, 在根据所述第二三元组, 调用 MPI 接收原语处理所述第一数据流图参数的方面第一数据流图参数, 所述第二计算节点用于通过 MPI 探测原语检测所述第二计算节点主机内存中的数据缓存区, 以获得所述第一数据流图参数的第二三元组, 所述数据缓存区专用于存储由 MPI 原语处理的数据; 调用 MPI 接收原语, 以处理所述第一数据流图参数, 所述 MPI 接收原语的接口参数包括所述第二三元组。

6. 根据权利要求 1 到 4 任一权利要求所述的系统, 其特征在于, 所述第一数据流图参数接收原语携带所述第一数据流图参数的目的地址, 在通过第一数据流图参数接收原语处理所述第一数据流图参数的方面, 所述第二计算节点用于将所述第二三元组作为所述 MPI 接收原语的接口参数调用所述 MPI 接收原语, 以将所述第一数据流图参数从所述数据缓存区存储到所述目的地址。

7. 一种分布式计算系统中的数据传输方法, 所述分布式计算系统包括第一计算节点和第二计算节点, 其特征在于, 所述方法包括:

从所述第一计算节点中的第一图数据结构中, 确定第一数据流图的第一数据流图参数的名称、大小和通信对端标识, 其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数, 所述通信对端标识对应所述第二计算节点;

根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识, 利用第一接口参数生成算法生成第一三元组, 所述第一三元组包括消息标记、消息大小和目的进程序列号, 其中, 所述消息标记对应于所述第一数据流图参数的名称, 所述消息大小对应于所述第一数据流图参数的大小, 所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程;

以所述第一三元组作为接口参数, 调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数, 以便于所述第二计算节点以与所述第一三元组对应的第二三元组作为接口参数, 调用 MPI 接收原语处理所述第一数据流图参数, 所述第二三元组是根据所述第二计算节点中的第二图数据结构, 利用第二接口参数生成算法生成的, 所述第二接口参数生成算法与所述第一接口生成算法相同。

8. 根据权利要求 7 所述的方法, 其特征在于, 在以所述第一三元组作为接口参数, 调用消息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数的方面, 所述方法包括:

以所述第一三元组作为接口参数, 通过消息传递接口 MPI 发送原语从所述第一计算节点的主机内存中读取所述第一数据流图参数, 以向所述第二计算节点发送所述第一数据流图参数。

9. 根据权利要求 8 所述的方法, 其特征在于, 所述第一计算节点还保存有所述第一数据流图参数所在的存储设备的信息, 所述方法还包括:

在所述存储设备的信息指示为其他存储设备的情况下, 将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存, 所述其他存储设备为所述第一计算节点中除主机

内存外的内存。

10.根据权利要求 7 到 9 任一所述的方法,其特征在于,所述第一接口参数生成算法包括第一算法、第二算法和第三算法,在根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第一接口参数生成算法生成第一三元组的方面,所述第一计算节点用于:

- 5 根据所述第一图数据结构中的第一数据流图参数的名称和所述第一算法,确定所述第一三元组中的消息标记,根据所述第一图数据结构中的第一数据流图参数的大小和所述第二算法,确定所述第一三元组中的消息大小,以及根据所述第一图数据结构中的第一数据流图参数的通信对端标识和所述第三算法,确定所述第一三元组中的目的进程序列号。

- 11、一种分布式计算系统中的数据传输装置,所述分布式计算系统包括第一计算节点和第二
10 计算节点,其特征在于,所述数据传输装置位于所述第一计算节点,所述数据传输装置包括:

确定模块,所述确定模块用于从所述第一计算节点中的第一图数据结构中,确定第一数据流图的第一数据流图参数的名称、大小和通信对端标识,其中所述第一数据流图参数为所述第一数据流图的一连接边所承载的参数,所述通信对端标识对应所述第二计算节点;

- 生成模块,所述生成模块用于根据所述第一图数据结构中的第一数据流图参数的名称、大小
15 和通信对端标识,利用第一接口参数生成算法生成第一三元组,所述第一三元组包括消息标记、消息大小和目的进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述目的进程序列号对应于所述第二计算节点中接收所述第一数据流图参数的进程;

- 通信模块,所述通信模块用于以所述第一三元组作为接口参数,调用消息传递接口 MPI 发送
20 原语向所述第二计算节点发送所述第一数据流图参数,以便于所述第二计算节点以与所述第一三元组对应的第二三元组作为接口参数,调用 MPI 接收原语处理所述第一数据流图参数,所述第二三元组是根据所述第二计算节点中的第二图数据结构,利用第二接口参数生成算法生成的,所述第二接口参数生成算法与所述第一接口生成算法相同。

12. 根据权利要求 11 所述的装置,其特征在于,在以所述第一三元组作为接口参数,调用消
25 息传递接口 MPI 发送原语向所述第二计算节点发送所述第一数据流图参数的方面,所述通信模块用于以所述第一三元组作为接口参数,通过消息传递接口 MPI 发送原语从所述第一计算节点的主机内存中读取所述第一数据流图参数,以向所述第二计算节点发送所述第一数据流图参数。

- 13.根据权利要求 12 所述的装置,其特征在于,所述第一计算节点还保存有所述第一数据流
30 图参数所在的存储设备的信息,所述第一计算节点还包括读取模块,所述读取模块用于在所述存储设备的信息指示为其他存储设备的情况下,将所述第一数据流图参数从所述其他存储设备复制到所述第一计算节点的主机内存,所述其他存储设备为所述第一计算节点中除主机内存外的内存。

- 14.根据权利要求 11 到 13 任一所述的装置,其特征在于,所述第一接口参数生成算法包括第
35 一算法、第二算法和第三算法,在根据所述第一图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第一接口参数生成算法生成第一三元组的方面,所述生成模块用于根据所述第一图数据结构中的第一数据流图参数的名称和所述第一算法,确定所述第一三元组中的消息标记,根据所述第一图数据结构中的第一数据流图参数的大小和所述第二算法,确定所述第一三元组中的消息大小,以及根据所述第一图数据结构中的第一数据流图参数的通信对端标识和所述第三算法,确定所述第一三元组中的目的进程序列号。

15.一种物理机,所述物理机包括:至少一个处理器和存储可执行代码的非瞬态计算机可读介质以运行分布式计算系统中的第一计算节点,所述分布式计算系统包括所述第一计算节点和第二计算节点;所述可执行代码在被所述至少一个处理器中的处理器执行时被配置为执行权利要求7-10中任一项所述的方法。

5 16.一种存储有可执行程序的非瞬态计算机可读介质,所述可执行程序包括用于执行权利要求7-10中任一项所述的方法的程序。

17.一种分布式计算系统中的数据传输方法,所述分布式计算系统包括第一计算节点和第二计算节点,其特征在于,所述方法包括:

10 从所述第二计算节点的第二图数据结构中,确定第二数据流图中的第一数据流图参数的名称、大小和通信对端标识,所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点;

根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识,利用第二接口参数生成算法生成第二三元组,所述第二三元组包括消息标记、消息大小和源进程序列号,其中,所述消息标记对应于所述第一数据流图参数的名称,所述消息大小对应于所述第一数据流图参数的大小,所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程;

15 根据所述第二三元组,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数,所述第一数据流图参数是所述第一计算节点通过 MPI 发送原语发送来的,所述 MPI 发送原语的接口参数包括与所述第二三元组对应的第一三元组,所述第一三元组是所述第一计算节点根据所述第一计算节点中的第一图数据结构,利用第一接口参数生成算法生成的,所述第二接口参数生成算法与所述第一接口生成算法相同。

20 18.根据权利要求17所述的方法,其特征在于,所述第二计算节点运行有第一线程和第二线程,所述第二计算节点的主机内存中包括数据缓存区,所述数据缓存区专用于存储被 MPI 原语处理的数据,在以所述第二三元组作为接口参数,调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数的方面,所述方法包括:

25 所述第一线程通过消息传递接口 MPI 探测原语检测所述主机内存中的数据缓存区,以获得第二三元组;

所述第一线程根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语,以处理所述第一数据流图参数,所述数据缓存区中的第二三元组是所述第二计算节点根据所述 MPI 发送原语得到的;

30 所述第二线程在确定所述第一数据流图参数被所述第一 MPI 接收原语处理后,将第二 MPI 接收原语修改为 MPI 等待原语,所述第二 MPI 接收原语为未被所述第二线程执行的与所述第一数据流图参数对应的接收原语,所述第二 MPI 接收原语的接口参数包括所述第二计算节点生成的第二三元组,所述 MPI 等待原语用于等待所述第一 MPI 接收原语执行完毕。

35 19.根据权利要求18所述的权利要求,其特征在于,在所述第一线程根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语,以处理所述第一数据流图参数的方面,所述方法包括:

在所述第一数据流图参数的目的地址对应所述第二计算节点的主机内存中分配给用户使用的内存空间的情况下,所述第一线程以所述数据缓存区中的第二三元组为所述第一 MPI 接收原语的接口参数,调用所述第一 MPI 接收原语,以将所述第一数据流图参数从所述数据缓存区存储到所述第一数据流图参数的目的地址。

20. 根据权利要求 17 所述的权利要求, 其特征在于, 所述方法还包括:

在所述第一数据流图参数的目的地址对应其他存储设备的情况下, 所述第二计算节点将所述主机内存中的第一数据流图参数存储到所述目的地址, 所述其他存储设备为所述第二计算节点中除主机内存之外的内存。

5 21. 根据权利要求 17 到 20 任一权利要求, 其特征在于, 所述第二接口参数生成算法包括第一算法、第二算法和第三算法, 在根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识, 利用第二接口参数生成算法生成第二三元组的方面, 所述方法包括:

10 根据所述第二图数据结构中的第一数据流图参数的名称和所述所述第二接口参数生成算法中的第一算法, 确定所述第二三元组中的消息标记; 根据所述第二图数据结构中的第一数据流图参数的大小和所述第二接口参数生成算法中的第二算法, 确定所述第二三元组中的消息大小; 以及根据第二图数据结构中的第一数据流图参数的通信对端标识和所述第二接口参数生成算法中的第三算法, 确定所述第二三元组中的源进程序列号。

22. 一种分布式计算系统中的数据传输装置, 所述分布式计算系统包括第一计算节点和第二计算节点, 所述数据传输装置位于所述第二计算节点, 其特征在于, 所述数据传输装置包括:

15 确定模块, 所述确定模块用于从所述第二计算节点的第二图数据结构中, 确定第二数据流图中的第一数据流图参数的名称、大小和通信对端标识, 所述第二数据流图中的所述第一数据流图参数的通信对端标识对应于所述第一计算节点;

20 生成模块, 所述生成模块用于根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识, 利用第二接口参数生成算法生成第二三元组, 所述第二三元组包括消息标记、消息大小和源进程序列号, 其中, 所述消息标记对应于所述第一数据流图参数的名称, 所述消息大小对应于所述第一数据流图参数的大小, 所述源进程序列号对应于所述第一计算节点中发送所述第一数据流图参数的进程;

25 通信模块, 所述通信模块用于根据所述第二三元组, 调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数, 所述第一数据流图参数是所述第一计算节点通过 MPI 发送原语发送来的, 所述 MPI 发送原语的接口参数包括与所述第二三元组对应的第一三元组, 所述第一三元组是所述第一计算节点根据所述第一计算节点中的第一图数据结构, 利用第一接口参数生成算法生成的, 所述第二接口参数生成算法与所述第一接口生成算法相同。

30 23. 根据权利要求 22 所述的数据传输装置, 其特征在于, 所述通信模块包括第一线程和第二线程, 所述第二计算节点的主机内存中包括数据缓存区, 所述数据缓存区专用于存储被 MPI 原语处理的数据, 在以所述第二三元组作为接口参数, 调用消息传递接口 MPI 接收原语处理来自所述第一计算节点的所述第一数据流图参数的方面, 所述第一线程用于通过消息传递接口 MPI 探测原语检测所述主机内存中的数据缓存区, 以获得第二三元组;

35 所述第一线程用于根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语, 以处理所述第一数据流图参数, 所述数据缓存区中的第二三元组是所述第二计算节点根据所述 MPI 发送原语得到的;

所述第二线程用于在确定所述第一数据流图参数被所述第一 MPI 接收原语处理后, 将第二 MPI 接收原语修改为 MPI 等待原语, 所述第二 MPI 接收原语为未被所述第二线程执行的与所述第一数据流图参数对应的接收原语, 所述第二 MPI 接收原语的接口参数包括所述第二计算节点生成的第二三元组, 所述 MPI 等待原语用于等待所述第一 MPI 接收原语执行完毕。

24.根据权利要求 23 所述的权利要求，其特征在于，在根据所述数据缓存区中的第二三元组调用第一 MPI 接收原语，以处理所述第一数据流图参数的方面，所述第一线程用于在所述第一数据流图参数的目的地址对应所述第二计算节点的主机内存中分配给用户使用的内存空间的情况下，以所述数据缓存区中的第二三元组为所述第一 MPI 接收原语的接口参数，调用所述第一 MPI 接收原语，以将所述第一数据流图参数从所述数据缓存区存储到所述第一数据流图参数的目的地址。

25. 根据权利要求 22 所述的权利要求，其特征在于，所述数据传输装置还包括存储模块，所述存储模块用于在所述第一数据流图参数的目的地址对应其他存储设备的情况下，将所述主机内存中的第一数据流图参数存储到所述目的地址，所述其他存储设备为所述第二计算节点中除主机内存之外的内存。

26.根据权利要求 22 到 25 所述的任一权利要求，其特征在于，所述第二接口参数生成算法包括第一算法、第二算法和第三算法，在根据所述第二图数据结构中的第一数据流图参数的名称、大小和通信对端标识，利用第二接口参数生成算法生成第二三元组的方面，所述生成模块用于：根据所述第二图数据结构中的第一数据流图参数的名称和所述第二接口参数生成算法中的第一算法，确定所述第二三元组中的消息标记；根据所述第二图数据结构中的第一数据流图参数的大小和所述第二接口参数生成算法中的第二算法，确定所述第二三元组中的消息大小；以及根据第二图数据结构中的第一数据流图参数的通信对端标识和所述第二接口参数生成算法中的第三算法，确定所述第二三元组中的源进程序列号。

27.一种物理机，所述物理机包括：至少一个处理器和存储可执行代码的非瞬态计算机可读介质以运行分布式计算系统中的第二计算节点，所述分布式计算系统包括第一计算节点和所述第二计算节点；所述可执行代码在被所述至少一个处理器中的处理器执行时被配置为执行权利要求 17-21 中任一项所述的方法。

28.一种存储有可执行程序的非瞬态计算机可读介质，所述可执行程序包括用于执行权利要求 17-21 中任一项所述的方法的程序。

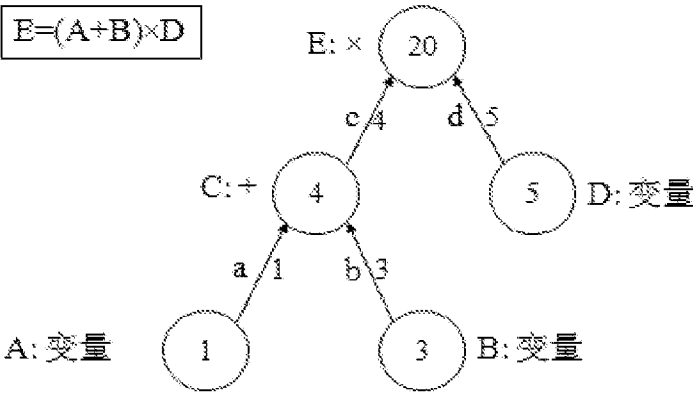


图 1

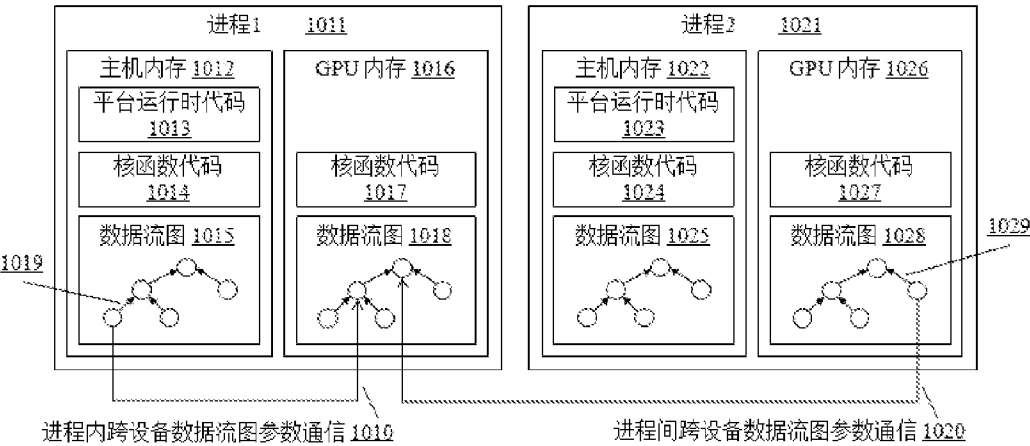


图 2

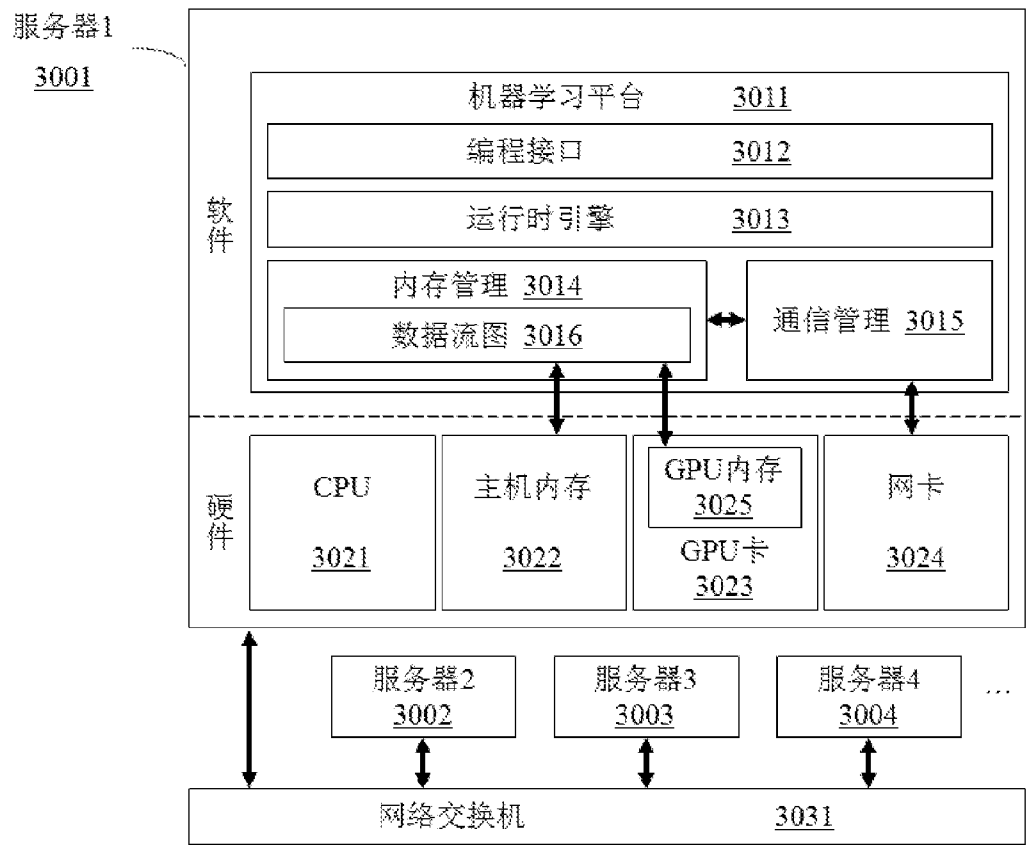


图 3

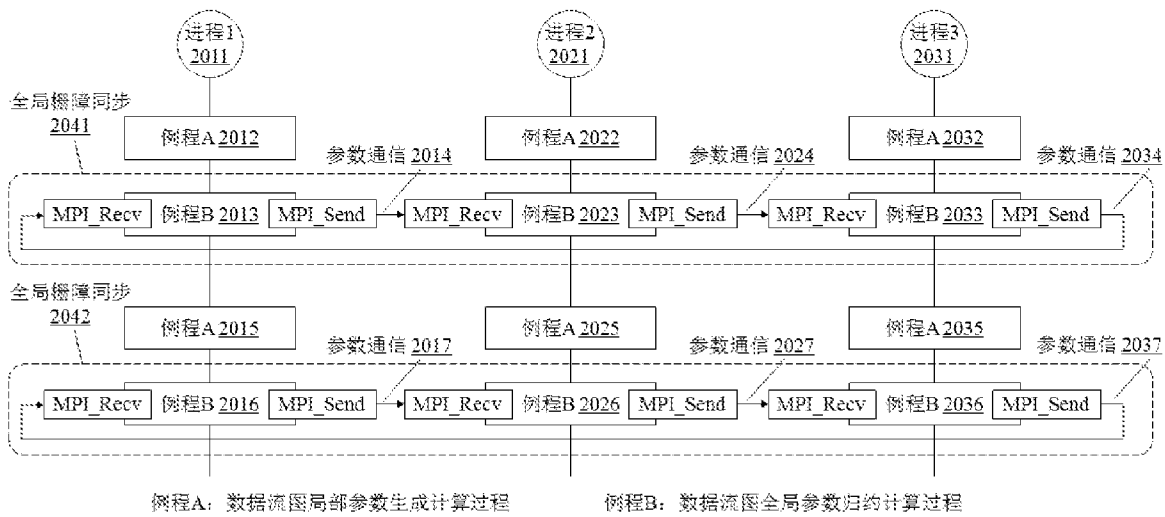


图 4

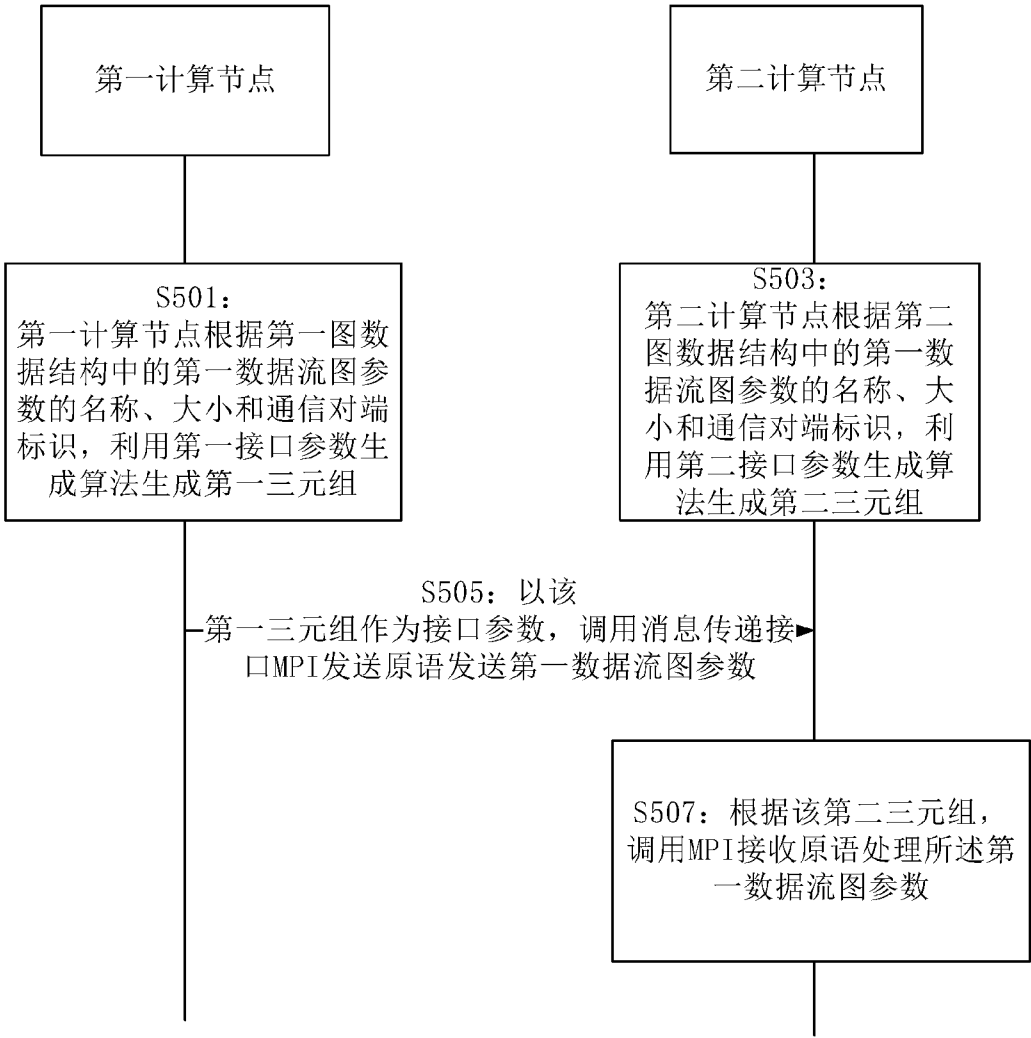


图 5

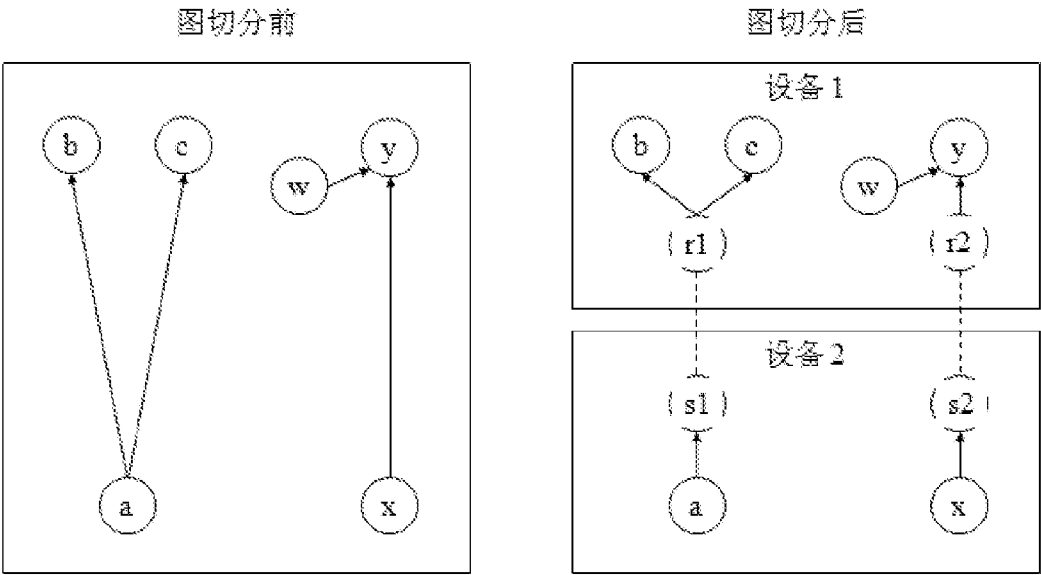


图 6

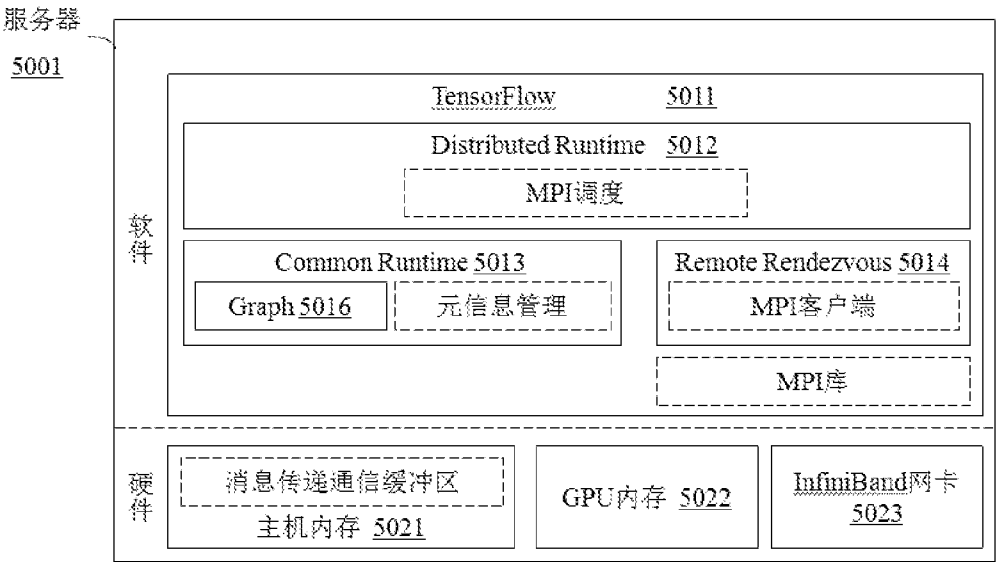


图 7

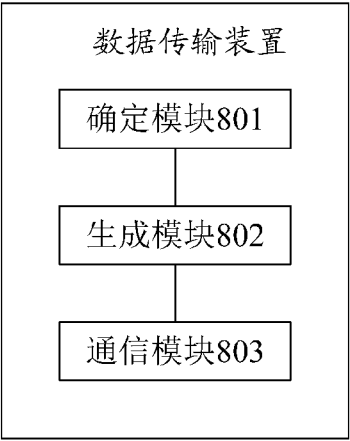


图 8

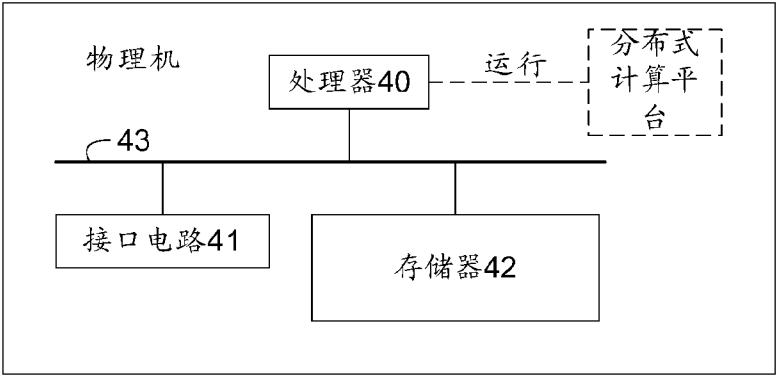


图 9

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2018/102919

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/48(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)

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)

WPI, EPODOC, CNPAT, CNKI, IEEE: 分布式, 节点, 数据, 传输, 通信, 通讯, 数据流图, 消息传递接口, 元组, 参数, 发送, 接收, 原语, 简化, 无需, 协商, 提高, 效率, distribute, node, data, transfer, transmit, communicate, flow, diagram, MPI, tuple, parameter, send, receive, primitive, simplify, negotiate, improve, effectiveness

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 105843706 A (HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY) 10 August 2016 (2016-08-10) claim 1, and description, paragraphs [0027]-[0030]	1-28
A	CN 103970580 A (HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY) 06 August 2014 (2014-08-06) entire document	1-28
A	CN 106547522 A (HUAWEI TECHNOLOGIES CO., LTD.) 29 March 2017 (2017-03-29) entire document	1-28
A	US 2016378560 A1 (PIVOTAL SOFTWARE, INC.) 29 December 2016 (2016-12-29) entire document	1-28

☐ 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

09 November 2018

Date of mailing of the international search report

28 November 2018

Name and mailing address of the ISA/CN

State Intellectual Property Office of the P. R. China
 No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing
 100088
 China

Authorized officer

Facsimile No. (86-10)62019451

Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2018/102919

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	105843706	A	10 August 2016	None			
CN	103970580	A	06 August 2014	None			
CN	106547522	A	29 March 2017	WO	2017045400	A1	23 March 2017
US	2016378560	A1	29 December 2016	WO	2015127668	A1	03 September 2015

国际检索报告

国际申请号

PCT/CN2018/102919

A. 主题的分类

G06F 9/48(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)

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))

WPI, EPODOC, CNPAT, CNKI, IEEE: 分布式, 节点, 数据, 传输, 通信, 通讯, 数据流图, 消息传递接口, 元组, 参数, 发送, 接收, 原语, 简化, 无需, 协商, 提高, 效率, distribute, node, data, transfer, transmit, communicate, flow, diagram, MPI, tuple, parameter, send, receive, primitive, simplify, negotiate, improve, effectiveness

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
A	CN 105843706 A (华中科技大学) 2016年 8月 10日 (2016 - 08 - 10) 权利要求1, 说明书第[0027]-[0030]段	1-28
A	CN 103970580 A (华中科技大学) 2014年 8月 6日 (2014 - 08 - 06) 全文	1-28
A	CN 106547522 A (华为技术有限公司) 2017年 3月 29日 (2017 - 03 - 29) 全文	1-28
A	US 2016378560 A1 (PIVOTAL SOFTWARE, INC.) 2016年 12月 29日 (2016 - 12 - 29) 全文	1-28

☐ 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

2018年 11月 9日

国际检索报告邮寄日期

2018年 11月 28日

ISA/CN的名称和邮寄地址

中华人民共和国国家知识产权局(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

王洋

电话号码 (86-10)53961523

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2018/102919

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)
CN	105843706	A	2016年 8月 10日	无			
CN	103970580	A	2014年 8月 6日	无			
CN	106547522	A	2017年 3月 29日	WO	2017045400	A1	2017年 3月 23日
US	2016378560	A1	2016年 12月 29日	WO	2015127668	A1	2015年 9月 3日

表 PCT/ISA/210 (同族专利附件) (2015年1月)