(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110389826 A
(43)申请公布日 2019.10.29

(21)申请号 201810361441.2

(22)申请日 2018.04.20

(71)申请人 伊姆西IP控股有限责任公司
地址 美国马萨诸塞州

(72)发明人 赵军平 王鲲 刘金鹏

(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华 李峥宇

(51)Int.Cl.
G06F 9/50(2006.01)
G06T 1/20(2006.01)

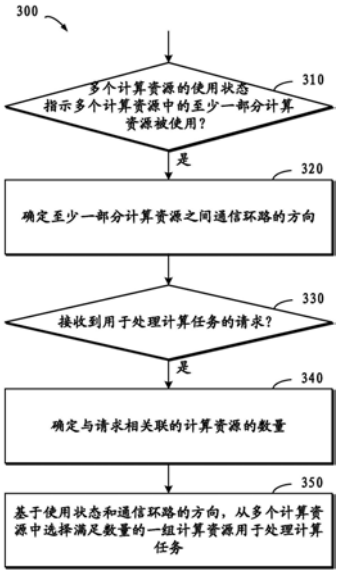
权利要求书3页 说明书12页 附图6页

(54)发明名称

用于处理计算任务的方法、设备和计算程序产品

(57)摘要

本公开的实现涉及用于处理计算任务的方法、设备和计算机程序产品。根据本公开的一个示例性实现，提供了一种用于处理计算任务的方法，包括：响应于多个计算资源的使用状态指示多个计算资源中的至少一部分计算资源被使用，确定至少一部分计算资源之间通信环路的方向；响应于接收到用于处理计算任务的请求，确定与请求相关联的计算资源的数量；以及基于使用状态和通信环路的方向，从多个计算资源中选择满足数量的计算资源的序列用于处理计算任务。根据本公开的一个示例性实现，提供了用于处理计算任务的设备和计算机程序产品。根据本公开的示例性实现，可以充分利用多个计算资源，并且确保多个计算资源之间的负载平衡。



CN 110389826 A

1.一种用于处理计算任务的方法,包括:

响应于多个计算资源的使用状态指示所述多个计算资源中的至少一部分计算资源被使用,确定所述至少一部分计算资源之间通信环路的方向;

响应于接收到用于处理所述计算任务的请求,确定与所述请求相关联的计算资源的数量;以及

基于所述使用状态和所述通信环路的方向,从所述多个计算资源中选择满足所述数量的计算资源的序列用于处理所述计算任务。

2.根据权利要求1所述的方法,其中基于所述使用状态和所述通信环路的方向选择所述计算资源的序列包括:

基于所述使用状态和所述通信环路的所述方向建立拓扑关系,所述拓扑关系中的节点表示所述多个计算资源中的计算资源,以及所述拓扑关系中的有向边表示所述至少一部分计算资源之间的通信环路中的通信路径;

基于序列中的节点构成的环路与所述拓扑关系中的已有环路之间的重叠程度,选择所述序列中的节点所表示的所述计算资源的序列。

3.根据权利要求2所述的方法,进一步包括确定所述重叠程度,包括:

确定所述序列中的节点与所述已有环路中的节点的节点重叠程度以及所述序列中的节点之间的有向边与所述已有环路中的有向边的边重叠程度;以及

基于所述节点重叠程度和所述边重叠程度确定所述重叠程度。

4.根据权利要求3所述的方法,其中确定所述节点重叠程度包括:针对所述序列中的当前节点,

响应于确定所述当前节点被包括在所述已有环路中,提高所述当前节点的节点重叠程度。

5.根据权利要求3所述的方法,其中确定所述边重叠程度包括:针对所述序列中的两个节点之间的当前有向边,

响应于确定所述当前有向边被包括在所述已有环路中,提高所述当前有向边的边重叠程度。

6.根据权利要求3所述的方法,进一步包括:

确定与所述已有环路中的有向边相关联的通信带宽;以及

基于所述通信带宽更新所述有向边的所述边重叠程度。

7.根据权利要求6所述的方法,其中基于所述通信带宽更新所述有向边的所述边重叠程度包括:针对所述序列中的两个节点之间的当前有向边,

响应于确定所述当前有向边所表示的通信路径的通信带宽降低,提高所述边重叠程度。

8.根据权利要求3所述的方法,进一步包括:从所述拓扑关系中选择与所述已有环路之间的重叠程度较低的序列。

9.根据权利要求2所述的方法,进一步包括:响应于所述多个计算资源的使用状态出现变化,更新所述拓扑关系。

10.根据权利要求1所述的方法,其中所述多个计算资源是多个图形处理单元,以及所述方法进一步包括:

在所述一组图形处理单元之间，基于AllReduce规则建立环形通信通道；以及

向所述一组图形处理单元分配待处理数据；以及

由所述一组图形处理单元处理所述待处理数据。

11.一种用于处理计算任务的设备，包括：

至少一个处理器；

易失性存储器；以及

与所述至少一个处理器耦合的存储器，所述存储器具有存储于其中的指令，所述指令在被所述至少一个处理器执行时使得所述设备执行动作，所述动作包括：

响应于多个计算资源的使用状态指示所述多个计算资源中的至少一部分计算资源被使用，确定所述至少一部分计算资源之间通信环路的方向；

响应于接收到用于处理所述计算任务的请求，确定与所述请求相关联的计算资源的数量；以及

基于所述使用状态和所述通信环路的方向，从所述多个计算资源中选择满足所述数量的计算资源的序列用于处理所述计算任务。

12.根据权利要求11所述的设备，其中基于所述使用状态和所述通信环路的方向选择所述计算资源的序列包括：

基于所述使用状态和所述通信环路的所述方向建立拓扑关系，所述拓扑关系中的节点表示所述多个计算资源中的计算资源，以及所述拓扑关系中的有向边表示所述至少一部分计算资源之间的通信环路中的通信路径；

基于序列中的节点构成的环路与所述拓扑关系中的已有环路之间的重叠程度，选择所述序列中的节点所表示的所述计算资源的序列。

13.根据权利要求12所述的设备，所述动作进一步包括确定所述重叠程度，包括：

确定所述序列中的节点与所述已有环路中的节点的节点重叠程度以及所述序列中的节点之间的有向边与所述已有环路中的有向边的边重叠程度；以及

基于所述节点重叠程度和所述边重叠程度确定所述重叠程度。

14.根据权利要求13所述的设备，其中确定所述节点重叠程度包括：针对所述序列中的当前节点，

响应于确定所述当前节点被包括在所述已有环路中，提高所述当前节点的节点重叠程度。

15.根据权利要求13所述的设备，其中确定所述边重叠程度包括：针对所述序列中的两个节点之间的当前有向边，

响应于确定所述当前有向边被包括在所述已有环路中，提高所述当前有向边的边重叠程度。

16.根据权利要求13所述的设备，所述动作进一步包括：

确定与所述已有环路中的有向边相关联的通信带宽；以及

基于所述通信带宽更新所述有向边的所述边重叠程度。

17.根据权利要求16所述的设备，其中基于所述通信带宽更新所述有向边的所述边重叠程度包括：针对所述序列中的两个节点之间的当前有向边，

响应于确定所述当前有向边所表示的通信路径的通信带宽降低，提高所述边重叠程

度。

18.根据权利要求13所述的设备,所述动作进一步包括:从所述拓扑关系中选择与所述已有环路之间的重叠程度较低的序列。

19.根据权利要求12所述的设备,进一步包括:响应于所述多个计算资源的使用状态出现变化,更新所述拓扑关系。

20.根据权利要求11所述的设备,其中所述多个计算资源是多个图形处理单元,以及所述方法进一步包括:

在所述一组图形处理单元之间,基于AllReduce规则建立环形通信通道;以及

向所述一组图形处理单元分配待处理数据;以及

由所述一组图形处理单元处理所述待处理数据。

21.一种计算机程序产品,所述计算机程序产品被有形地存储在计算机可读介质上并且包括机器可执行指令,所述机器可执行指令在被执行时使机器执行根据权利要求1至10中任一项所述的方法。

用于处理计算任务的方法、设备和计算程序产品

技术领域

[0001] 本公开的实现概括地涉及包括专用计算资源的计算系统，并且更具体地，涉及用于处理计算任务的方法、设备以及计算机程序产品。

背景技术

[0002] 客户端上的应用可以被设计用于利用处理和存储资源等计算资源来完成各种处理或分析任务。随着诸如机器学习、深度学习、数据挖掘等任务的需求和复杂度不断增加，需要大量和/或可变的计算资源来满足相应应用的运行。这可以通过具有多个专用计算资源的机器或系统来实现，其中应用可以被调度到该机器或系统的一个或多个专用计算资源上运行。例如，已经开发了基于云的计算系统，该计算系统包括具有一个或多个专用计算资源的机器。不同客户端可以根据需要来租赁该系统的计算资源(例如，专用计算资源)用以运行各自的应用。

[0003] 随着计算机技术的发展，计算资源的种类越来越丰富，并且已经不再局限于传统的诸如中央处理单元的计算资源。例如，目前图形处理单元(Graphic Processing Unit，GPU)的计算能力越来越强。由于GPU特有性质，GPU特别适合于执行有关深度学习(Deep Learning)、高性能计算(High Performance Computing)、以及机器学习(Machine Learning)等方面的计算任务。然而，对于普通的客户端设备以及常规的云计算设备而言，这些设备的图形处理单元的性能通常较为有限，并不具有高性能的处理能力。因而，此时如何利用(例如以远程方式)利用其他设备所具有的图形处理单元的计算能力来处理计算任务成为一个研究焦点。

[0004] 然而，目前的一些技术方案并不能充分且有效地利用远程的计算资源(例如，计算资源池中的计算资源)的处理能力，而是在资源池中将会出现闲置和/或工作负载不平衡的状况。因此，期望能够提供一种以简单并且有效的方式来利用资源池中的多个计算资源处理计算任务的技术方案。

发明内容

[0005] 本公开的实现提供了用于处理计算任务的方法、设备和相应的计算机可读介质。

[0006] 根据本公开的第一方面，提供了一种用于处理计算任务的方法，包括：响应于多个计算资源的使用状态指示多个计算资源中的至少一部分计算资源被使用，确定至少一部分计算资源之间通信环路的方向；响应于接收到用于处理计算任务的请求，确定与请求相关联的计算资源的数量；以及基于使用状态和通信环路的方向，从多个计算资源中选择满足数量的计算资源的序列用于处理计算任务。

[0007] 根据本公开的第二方面，提供了一种用于处理计算任务的设备，包括：至少一个处理器；易失性存储器；以及与至少一个处理器耦合的存储器，存储器具有存储于其中的指令，指令在被至少一个处理器执行时使得设备执行动作。该动作包括：响应于多个计算资源的使用状态指示多个计算资源中的至少一部分计算资源被使用，确定至少一部分计算资源

之间通信环路的方向;响应于接收到用于处理计算任务的请求,确定与请求相关联的计算资源的数量;以及基于使用状态和通信环路的方向,从多个计算资源中选择满足数量的计算资源的序列用于处理计算任务。

[0008] 根据本公开内容的第三方面,提供了一种计算机程序产品。该计算机程序产品被有形地存储在计算机可读介质上并且包括机器可执行指令,机器可执行指令在被执行时使机器执行根据第一方面的方法。

[0009] 提供发明内容部分是为了简化的形式来介绍对概念的选择,它们在下文的具体实施方式中将被进一步描述。发明内容部分无意标识本公开的关键特征或主要特征,也无意限制本公开的范围。

附图说明

[0010] 通过结合附图对本公开示例性实现进行更详细的描述,本公开的上述以及其它目的、特征和优势将变得更加明显,其中,在本公开示例性实现中,相同的参考标号通常代表相同部件。

[0011] 图1示意性示出了适于实现本公开内容实现方式的示例性计算系统的框图;

[0012] 图2A示意性示出了根据一个技术方案的用于处理计算任务的过程的框图,图2B示意性示出了根据本公开的一个示例性实现的用于处理计算任务的过程的框图;

[0013] 图3示意性示出了根据本公开的一个示例性实现的用于处理计算任务的方法的流程图;

[0014] 图4示意性示出了根据本公开的一个示例性实现的拓扑关系的框图;

[0015] 图5示意性示出了根据本公开的一个示例性实现的用于根据拓扑关系来选择一组计算资源的框图;

[0016] 图6示意性示出了根据本公开的一个示例性实现的拓扑关系的框图;

[0017] 图7示意性示出了根据本公开的一个示例性实现的拓扑关系的框图;

[0018] 图8示意性示出了根据本公开的一个示例性实现的用于处理计算任务的设备的框图;以及

[0019] 图9示意性示出了根据本公开的一个示例性实现的用于处理计算任务的设备的框图。

具体实施方式

[0020] 下面将参照附图更详细地描述本公开的优选实现。虽然附图中显示了本公开的优选实现,然而应该理解,可以以各种形式实现本公开而不应被这里阐述的实现所限制。相反,提供这些实现是为了使本公开更加透彻和完整,并且能够将本公开的范围完整地传达给本领域的技术人员。

[0021] 在本文中使用的术语“包括”及其变形表示开放性包括,即“包括但不限于”。除非特别申明,术语“或”表示“和/或”。术语“基于”表示“至少部分地基于”。术语“一个示例实现”和“一个实现”表示“至少一个示例实现”。术语“另一实现”表示“至少一个另外的实现”。术语“第一”、“第二”等等可以指代不同的或相同的对象。下文还可能包括其它明确的和隐含的定义。

[0022] 如上，专用计算资源可以在客户端本地或者可以由远程机器或系统提供。在一些示例中，可以部署基于云的计算系统，其中包括具有一个或多个专用计算资源的多个机器。该计算系统的专用计算资源可以由不同客户端根据需要来使用，以将相应的应用调度到可用的专用计算资源上运行。

[0023] 图1示出了本公开的实现可以在其中被实现的示例计算系统100的示意图。在该计算系统100中部署了用于应用运行的多个服务器包括服务器110-1、服务器110-2、服务器110-3...、服务器110-U(以下统称为或单独称为服务器110，其中U为大于1的自然数)。计算系统100还包括专用计算资源160-1、专用计算资源160-2、专用计算资源160-3...、专用计算资源160-V(以下统称为或单独称为专用计算资源160，其中V为大于1的自然数)。每个服务器110上具有一个或多个专用计算资源160。

[0024] 在图1的示例中，服务器110-1具有专用计算资源160-1，服务器110-2具有专用计算资源160-2，并且服务器110-U具有专用计算资源160-V。将会理解，在此并不限制每个服务器仅具有一个计算资源，而是一个服务器可以具有一个或多个计算资源。因而，在此U和V的数值可以是不相等的。专用计算资源160的示例可以包括但不限于图形专用计算资源(GPU)、现场可编程门阵列(FPGA)等。为便于讨论，某些实现将以GPU作为专用计算资源的示例进行描述。除了专用计算资源160之外，服务器110还可以包括诸如中央处理单元(CPU)的一个或多个通用处理单元(未示出)。

[0025] 图1还示出了多个客户端120-1、120-2……、120-P等(以下统称或单独称为客户端120，其中P为大于1的自然数)，分别具有要运行的应用150-1、150-2、……、150-Q(以下统称为或单独称为应用150，其中Q为大于1的自然数)。应用150可以是机器上可运行的任何应用，该应用可以被设计为执行相应数据处理或分析等任务。作为示例，应用150可以执行与高性能计算(HPC)、机器学习(ML)或深度学习(DL)以及人工智能(AI)等相关的数据处理或分析任务。将会理解，在此并不限制每个客户端仅具有一个应用，而是一个客户端可以具有一个或多个应用。因而，在此P和Q的数值可以是不相等的。

[0026] 为了能够快速高效运行这些应用和/或为了保留本地计算资源，客户端120可以请求服务器110的专用计算资源160来运行这些应用150。在这样的实现中，客户端120可以通过互连网络130连接到一个或多个服务器110，并且将应用150交由服务器110的一个或多个专用计算资源160运行。取决于客户端120、服务器110和/或专用计算资源160所支持的接口，互连网络130可以支持基于诸如远程直接内存访问(RDMA)和传输控制协议(TCP)等各种网络传输技术的不同类型的有线或者无线连接。

[0027] 应当理解，图1示出的设备和/或布置仅是一个示例。在其他示例中，该计算系统100可以包括任意适当数目的服务器110和客户端120。每个服务器110可以安装有任意适当数目的专用计算资源160，并且每个客户端120可以具有待运行的多个应用150。此外，尽管被单独示出，调度器140在实际应用中可以由独立于服务器110的其他设备实现，或者可以被部分或全部实现在一个或多个服务器110上。

[0028] 为了描述清楚和简洁，将主要以GPU内核为例来详细描述本公开的示例实现。如已知的，GPU作为一种专用处理器，其强大的计算能力源自其大量的内核和高带宽的内存。在GPU硬件架构中，一个GPU通常具有大量的GPU内核，例如5120或者接近10000个内核。GPU内核作为一种专用计算资源，是最基本的处理单元，也被称为流处理器(SP)。指令和任务最终

都在GPU内核上被处理。多个GPU内核同时执行指令，从而实现了GPU的并行计算。多个SP加上一些其他资源，例如寄存器、共享内存，可以组成一个流多处理器(SM)。

[0029] 但是，应当理解，GPU仅仅是一种示例性的专用计算资源，并非用于限制本公开的范围。在此描述的精神和原理可以应用于其他专用计算资源，例如诸如现场可编程门阵列(FPGA)之类的加速器中的计算资源，不论是目前已知的还是将来开发的，而并不仅仅限于GPU内核。

[0030] 随着云计算的发展，目前已经提出了基于云架构的用于处理计算任务的技术方案。例如，客户端150中的应用120可以请求服务器110中的计算资源160。应当注意，由于计算任务的复杂性，计算任务通常需要调用多个计算资源。图2A示意性示出了根据一个技术方案的用于处理计算任务的过程的框图200A。如图2A所示，其中示出了计算任务所调用的多个计算资源(例如，160-1、160-2、160-3、160-4、……、160V。基于现有的用于GPU计算的协议，多个计算资源160-1至160-V之间可以按照图2A所示，例如，按照箭头210A所示的方向以环形方式进行通信。

[0031] 将会理解，目前已经提供了多种方式来在各个计算资源之间进行通信。根据一种方式，可以采用网卡之间的通信通路来构建环形通信路径；根据另一方式，对于某些型号的显示卡而言，还可以采取显示卡之间的专用通信通路来构建环形通信路径。将会理解，随着硬件设备的进步，越来越多的硬件已经开始支持双向通信方式。然而，现有的计算资源之间的通信环路的构建仍然仅支持单一方向，这导致计算资源不能被合理地使用，并且存在闲置的状况。

[0032] 基于现有技术中的不足，本公开提出了一种用于处理计算任务的方法。在该方法中，可以监视计算资源池中的多个计算资源的使用状态，并且确定多个计算资源中的被使用的至少一部分计算资源。继而，可以确定至少一部分计算资源之间通信环路的方向，并且基于使用状态和通信环路的方向，从多个计算资源中选择满足数量的一组计算资源用于处理计算任务。

[0033] 图2B示意性示出了根据本公开的一个示例性实现的用于处理计算任务的过程的框图200B。如图2B所示，可以按照如箭头210B所示的两个方向，来建立通信环路。例如，可以在计算资源160-1、160-2、160-3以及160-4之间建立顺时针方向的通信环路。又例如，还可以在计算资源160-1、160-2、160-3以及160-V之间建立逆时针方向的通信环路。在此实现中，由于考虑了被使用的计算资源之间的通信环路的方向，可以更充分地利用计算资源之间的可用通信通路。例如，可以选择计算资源并且尽量按照与现状已经被占用的通信通路相反的方向，来建立通信通路。

[0034] 根据本公开的一个示例性实现，提供了一种用于处理计算任务的方法。参见图3描述该方法的流程，图3示意性示出了根据本公开的一个示例性实现的用于处理计算任务的方法的流程图300。如图3所示，在方框310处，监视计算资源池中的多个计算资源的使用状态。在此的使用状态例如可以包括多个计算资源中的某个计算资源是否被正在被使用。在本公开的一个示例性实现中，使用状态还可以包括例如被使用的起始时间、被分配用于处理哪个计算任务等。如果使用状态指示多个计算资源中的至少一部分计算资源被使用，则在方框320处，确定至少一部分计算资源之间通信环路的方向。在此的方向是指从环路中的一个节点向另一节点发送数据的方向。

[0035] 在方框330处，确定是否接收到用于处理计算任务的请求。如果接收到请求，则在方框340处确定与请求相关联的计算资源的数量。请求可以指定需要多少计算资源来处理计算任务。继而，在方框350处，基于使用状态和通信环路的方向，可以从多个计算资源中选择满足数量的一组计算资源用于处理计算任务。

[0036] 在此示例性实现中，可以考虑已经被使用的计算资源、以及已经被使用的计算资源之间的通信环路的方向，进而可以尽量选择未被使用的计算资源，和/或还可以基于与已经使用的通信环路相反的方向，来选择用于处理计算任务的计算资源的序列。

[0037] 在下文中将参见图4详细描述本公开的实现的更多细节。图4示意性示出了根据本公开的一个示例性实现的拓扑关系400的框图。根据本公开的一个示例性实现，可以建立如图4所示的拓扑关系400，并基于拓扑关系400来选择计算资源的序列。具体地，可以基于使用状态和通信环路的方向建立拓扑关系400。如图4所示，拓扑关系400中的节点410、412、414、416、420、422等表示多个计算资源中的计算资源，以及拓扑关系中的有向边表示已经被使用的计算资源之间的通信环路中的通信路径。例如，节点410、412、414和416所表示的计算资源已经被分配用于处理计算任务，因而在各节点之间存在如箭头所示的有向边，并且这些有向边形成通信环路。节点420和422与其他节点之间不存在有向边，这表示节点420和422所表示的计算资源尚未被分配用于处理计算任务。

[0038] 进一步，可以基于序列中的节点构成的环路与拓扑关系中的已有环路之间的重叠程度，选择序列中的节点所表示的计算资源的序列。在此示例性实现中，重叠程度可以在一定程度上表示将被选择的计算资源的序列与资源池中的已经被使用的计算资源之间的冲突程度(即，资源竞争的程度)。因而，可以尽量选择冲突程度最小的计算资源的序列，来用于处理计算任务。以此方式，可以更加有效地分配资源池中的各个计算资源。

[0039] 在如图4所示的示例中，假设需要4个计算资源来处理新的计算任务，并且已经构建了如下两个序列：

[0040] 序列1：包括节点410、412、414和416所表示的节点；以及

[0041] 序列2：包括节点420、422、414和416所表示的节点。

[0042] 通过比较两个序列与已有拓扑关系400的重叠程度可知，序列1与拓扑关系400的重叠程度较高，而序列2与拓扑关系400的重叠程度较低，因而可以优先选择序列2。

[0043] 根据本公开的一个示例性实现，可以基于多个方面确定重叠程度。例如，可以确定序列中的节点与已有环路中的节点的节点重叠程度。在此节点重叠程度可以在一定程度上指示资源池中的计算资源160的工作负载。节点重叠程度越高，则表示将被选择的计算资源的工作负载越高。例如，可以以“有重叠”和“无重叠”来表示重叠度。继续上文的示例，序列1中包括的节点410、412、414和416已经是拓扑关系400中的环路中的节点，因而可以认为序列1与已有环路“有重叠”。序列2中包括的节点414和416已经是拓扑关系400中的环路中的节点，因而序列2与已有环路“有重叠”。在此实现中，可以尽量选择重叠程度较低的候选序列，以便选择工作负载较低的计算节点来服务于新的计算任务。由于序列2中仅有两个节点与拓扑关系400有重叠，因而可以优先选择序列2。

[0044] 根据本公开的一个示例性实现，对于已经确定的计算节点的序列中的当前节点，可以基于该当前时点是否已经被分配用于处理其他任务来确定节点重叠程度。如果确定当前节点被包括在已有环路(即，已经被分配用于处理其他计算任务)中，则提高当前节点的

节点重叠程度。在此实现中，可以以重叠节点的个数来衡量节点重叠程度，此时序列1与已有环路的节点重叠程度为4，并且序列2与已有环路的节点重叠程度为2。

[0045] 根据本公开的一个示例性实现，还可以确定序列中的节点之间的有向边与已有环路中的有向边的边重叠程度。在此边重叠程度可以在一定程度上指示资源池中的各个计算资源160之间的通信的繁忙程度。边重叠程度越高，则表示将被选择的计算资源之间的通信越频繁。将会理解，在此的边重叠程度考虑了边的方向，即使两个有向边的两个端点完全相同，由于有向边的方向不同，两个有向边之间的边重合度可以为零。以此方式，可以充分利用资源池中的各个计算资源160之间的通信通道。

[0046] 根据本公开的一个示例性实现，可以基于节点重叠程度和边重叠程度确定重叠程度。在此示例性实现中，通过考虑计算资源的工作负载和通信通道的繁忙程度，可以尽量选择资源池中的较为空闲的计算资源来处理新近接收到的新的计算任务。

[0047] 在一个简单实现中，可以针对当前节点正在处理的计算任务的数量来确定节点重合度。例如，当前计算节点已经被分配用于处理两个计算任务(即，处于两个环路中)，则可以将节点重合度设置为2。在另一实现中，可以针对当前节点在处理某个计算任务时被占用的资源量来确定节点重合度。

[0048] 在本公开的一个示例性实现中，还可以基于节点所表示的计算资源的使用率来确定节点重合度。假设当前节点中的10％的计算资源被分配用于处理一个计算任务，并且20％的计算资源被分配用于处理另一计算任务，则此时节点重合度例如可以是1+2＝3。备选地，还可以基于其他指标来计算节点重合度。在此示例性实现中，可以以量化的方式来确定某个计算资源的工作负载，进而在选择用于处理新计算任务时，可以尽量选择工作负载较低的计算节点。

[0049] 根据本公开的一个示例性实现，针对序列中的两个节点之间的当前有向边，可以基于当前有向边所对应的通信通路是否已经被占用，来确定边重合度。例如，如果当前有向边已经被占用(即，被包括在已有环路中)，则可以提高当前有向边的边重叠程度。在一个简单实现中，可以针对当前有向边正在服务于的计算任务的数量来确定边重合度。例如，当前有向边已经被分配用于处理两个计算任务(即，处于两个环路中)，则可以将边重合度设置为2。

[0050] 在另一实现中，可以针对当前节点在处理某个计算任务时被占用的资源量来确定节点重合度。假设当前节点中的10％的计算资源被分配用于处理一个计算任务，并且20％的计算资源被分配用于处理另一计算任务，则此时节点重合度例如可以是10％+20％＝30％。备选地，还可以基于其他指标来计算节点重合度。在此示例性实现中，可以以量化的方式来确定某个计算资源的工作负载，进而在选择用于处理新计算任务时，可以尽量选择工作负载较低的计算节点。

[0051] 图5示意性示出了根据本公开的一个示例性实现的用于根据拓扑关系来选择一组计算资源的框图500。如图5所示，节点410、412、414、416已经为被包括在已有环路中。假设以节点被使用的次数来衡量重叠程度，则与节点410、412、414、416相关联的数值可以是1，而与节点420、422相关联的数值可以是0。此时，可以尽量选择由以阴影示出的节点416、414、422和420，并且可以按照如箭头510、512、514、516所示的方向来构建序列。将会理解，由于新构建的序列中节点416和414之间的通信方向如箭头510所示，并且该方向与已有的

通信环路的方向相反，因而可以更充分地利用资源池中的空闲带宽。

[0052] 根据本公开的一个示例性实现，为了进一步详细描述各个节点之间的通信的频繁程度，还可以检测各个通信路径之间的带宽使用状况。例如，可以确定与已有环路中的有向边相关联的通信带宽，并且基于通信带宽更新有向边的边重叠程度。在此示例性实现中，可以以更为准确的方式记载各个路径的使用状况，进而有利于选择更为空闲的通信路径来建立通信环路。

[0053] 根据本公开的一个示例性实现，可以基于通信带宽更新有向边的边重叠程度包括。例如，针对序列中的两个节点之间的当前有向边，响应于确定当前有向边所表示的通信路径的通信带宽降低，提高边重叠程度。在此示例性实现中，边重叠程度与通信带宽成反比，因而边重叠程度将随着通信带宽的降低而提高。备选地，还可以基于可用通信带宽来更新边重叠程度，此时边重叠程度将随着可用带宽的提高而提高。

[0054] 图6示意性示出了根据本公开的一个示例性实现的拓扑关系的框图600。如图所示，可以以有向边所代表的通信路径被使用的次数来表示边重叠程度。有向边上的数字可以表示与该有向边相关联的边重合度。例如，双向箭头610所示的有向边的边重叠程度为3，这表示沿着双向箭头所示的两个方向所示的方向带宽负载均为3。将会理解，尽管图6示出了沿着两个方向的带宽负载相同的情况，在其他示例中，沿着两个方向的带宽负载可以是不同的。例如，从节点420到422的带宽负载可以是3，而从节点422到420的带宽负载可以是2。

[0055] 如图6所示，由于节点416、414、422和420所构成的两个环路(顺时针方向和逆时针方向)的带宽负载是相同的(均为3)，则此时可以任意选择顺时针方向或逆时针方向中的一个方向来构建环路。在另一示例中，假设顺时针方向的环路的带宽负载为3，而逆时针方向的环路的带宽负载为2，则可以选择沿着逆时针方向，利用节点416、414、422和420构建环路。

[0056] 根据本公开的一个示例性实现，还可以基于上文描述的节点重叠程度和边重叠程度两者来构建环路。例如，可以从拓扑关系中选择与已有环路之间的重叠程度较低的序列。可以首先构建满足请求的计算资源数量的多个候选序列，继而分别基于上文描述的方法来确定各个序列与已有环路之间的重叠程度。进而，可以选择重叠程度较低的序列中的计算资源用于处理新的计算任务。

[0057] 根据本公开的一个示例性实现，可以基于多种方式来构建候选序列。例如，可以首先从资源池中选择工作负载较低的计算资源，继而将这些选择的计算资源之间的通信线路的带宽进行排序，并从根据排序来选择使用率较低的通信线路来构建通信环路。以此方式，可以首先确保由工作负载较低的计算资源来服务于新的计算任务。

[0058] 根据本公开的一个示例性实现，还可以分别针对节点重叠程度和边重叠程度设置权重，以便基于最终的重叠程度来构建通信环路。例如，如果新的计算任务需要较高的计算量，则可以对节点重叠度赋予较高的权重。如果新的计算任务需要较低的计算量，则可以对节点重叠度赋予较低的权重。又例如，如果新的计算任务需要较高的通信带宽，则可以对边重叠度赋予较高的权重；如果新的计算任务需要较低的通信带宽，则可以对边重叠度赋予较低的权重。

[0059] 在本公开中可以基于多种方式来构建候选序列。例如，可以首先从资源池中选择

工作负载较低的计算资源，继而将这些选择的计算资源之间的通信线路的带宽进行排序，并从根据排序来选择使用率较低的通信线路来构建通信环路。以此方式，可以首先确保由工作负载较低的计算资源来服务于新的计算任务。在此基础上，还可以选择使用率较低的通信线路来构建通信环路，以便确保尽量充分地利用资源池中的计算资源的计算能力以及计算资源之间的通信带宽。

[0060] 又例如，可以首先从资源池中选择使用率较低的通信线路，继而将与这些通信线路相关联的计算资源的工作负载进行排序，并从根据排序来选择工作负载较低的计算资源来构建通信环路。以此方式，可以首先确保新构建的通信环路的具有较高的通信效率。在此基础上，还可以通过选择工作负载较低的计算资源，来确保尽量充分地利用资源池中的计算资源的计算能力以及计算资源之间的通信带宽。

[0061] 图7示意性示出了根据本公开的一个示例性实现的拓扑关系的框图700。假设节点410、412、414、416所表示的计算资源的工作负载均为50％，其他节点的工作负载为0。节点410、412、414、416之间沿逆时针方向所示环路的已使用带宽为5GB/s，而其他路径的已使用带宽为0GB/s。可以首先将节点重叠程度和边重叠程度进行归一化，继而基于分别用于节点重叠程度和边重叠程度的权重来计算最终的重叠程度。

[0062] 假设各个节点之间的最大带宽为10GB/s，则节点410、412、414、416之间沿逆时针方向所示环路的边重叠程度可以基于如下方式计算：5/10＝50％。假设用于节点重叠程度和边重叠程度的权重均为0.5，则可以序列的重叠程度可以计算为：

[0063] 重叠程度＝节点权重*(节点重叠程度)+边权重*(边重叠程度)。

[0064] 公式1

[0065] 应当注意，尽管在上文公式1中给出基于乘法和加法来确定重叠程度的具体公式。在其他示例中，还可以基于其他运算来确定重叠程度。例如，可以使用公式2或者其他公式来计算重叠程度。

[0066] 重叠程度＝节点权重*(节点重叠程度)*边权重*(边重叠程度)。

[0067] 公式2

[0068] 继续图7的示例，基于公式1，序列1的重叠程度1可以计算为：

[0069] 重叠程度1

[0070] ＝0.5*(50％+50％+50％+50％)+0.5*(50％+50％+50％+50％)

[0071] ＝1+1＝2

[0072] 基于公式1，序列2的重叠程度2可以计算为：

[0073] 重叠程度2

[0074] ＝0.5*(50％+50％+0+0)+0.5*(0+0+0+0)

[0075] ＝0.5

[0076] 基于上述计算可知，重叠程度1大于重叠程度2，因而可以优先选择序列2相对应的计算资源，并且按照序列2的方向构建通信环路。此时，可以基于节点420、422、414和416所表示的计算资源构建通信环路。

[0077] 根据本公开的一个示例性实现，可以基于计算资源池中的多个计算资源的使用状态出现变化，更新拓扑关系。将会理解，由于资源池中的计算资源的使用状态是不断更新的，因而可以不断基于最新的使用状态来更新拓扑关系。例如，可以在预定时间间隔更新拓

扑关系。根据本公开的一个示例性实现，可以按照每秒1次或者其他频率监视使用带宽。又例如，还可以按照每秒1次或者其他频率监视如下信息：

[0078] 资源池中正在运行的计算任务的ID，例如可以包括名称、来源或者其他信息。

[0079] 资源池中被调用的计算资源的序列，该列表可以记录由被调用的计算资源所建立的通信环路的序列。例如，对于包括如图7所示的计算资源的资源池，序列可以表示为：410-412-414-416。再例如，序列还可以由各个计算资源的地址以及端口表示。又例如，序列还可以由各个计算资源的唯一标识符来表示。

[0080] 资源池中的计算资源被调用的时间，例如，可以包括被分配用于处理某计算任务的开始时间和结束时间。

[0081] 资源池中的计算资源被调用的状态，状态例如可以是以下中的任一项：启动、运行中、成功、失败等等。

[0082] 根据本公开的一个示例性实现，多个计算资源是多个图形处理单元。在一组图形处理单元之间，可以基于AllReduce规则建立环形通信通道，继而向一组图形处理单元分配待处理数据，并且由一组图形处理单元处理待处理数据。AllReduce规则是在GPU之间进行通信的规则，本领域技术人员可以基于该规则的定义来获得更多具体细节。

[0083] 在上文中已经参见图2B至图7详细描述了根据本公开的方法的示例，在下文中将参见图8详细描述相应的设备的实现。图8示意性示出了根据本公开的一个示例性实现的用于处理计算任务的设备800的框图。该设备800包括：方向确定模块810，配置用于响应于多个计算资源的使用状态指示多个计算资源中的至少一部分计算资源被使用，确定至少一部分计算资源之间通信环路的方向；数量确定模块820，配置用于响应于接收到用于处理计算任务的请求，确定与请求相关联的计算资源的数量；以及选择模块830，配置用于基于使用状态和通信环路的方向，从多个计算资源中选择满足数量的计算资源的序列用于处理计算任务。在此的设备800可以配置用于执行上文描述的方法中的各个步骤，在此不再赘述。

[0084] 图9示意性示出了根据本公开的一个示例性实现的用于处理计算任务的设备的框图。如图所示，设备900包括中央处理单元(CPU)901，其可以根据存储在只读存储器(ROM)902中的计算机程序指令或者从存储单元908加载到随机访问存储器(RAM)903中的计算机程序指令，来执行各种适当的动作和处理。在RAM 903中，还可存储设备900操作所需的各种程序和数据。CPU 901、ROM 902以及RAM 903通过总线904彼此相连。输入/输出(I/O)接口905也连接至总线904。

[0085] 设备900中的多个部件连接至I/O接口905，包括：输入单元906，例如键盘、鼠标等；输出单元907，例如各种类型的显示器、扬声器等；存储单元908，例如磁盘、光盘等；以及通信单元909，例如网卡、调制解调器、无线通信收发机等。通信单元909允许设备900通过诸如因特网的计算机网络和/或各种电信网络与其他设备交换信息/数据。

[0086] 上文所描述的各个过程和处理，例如方法300，可由处理单元901执行。例如，在一些实现中，方法300可被实现为计算机软件程序，其被有形地包含于机器可读介质，例如存储单元908。在一些实现中，计算机程序的部分或者全部可以经由ROM 902和/或通信单元909而被载入和/或安装到设备900上。当计算机程序被加载到RAM 903并由CPU 901执行时，可以执行上文描述的方法300的一个或多个步骤。备选地，在其他实现中，CPU 901也可以以其他任何适当的方式被配置以实现上述过程/方法。

[0087] 根据本公开的一个示例性实现，提供了一种用于处理计算任务的设备，包括：至少一个处理器；易失性存储器；以及与所述至少一个处理器耦合的存储器，所述存储器具有存储于其中的指令，所述指令在被所述至少一个处理器执行时使得所述设备执行动作。所述动作包括：响应于多个计算资源的使用状态指示所述多个计算资源中的至少一部分计算资源被使用，确定所述至少一部分计算资源之间通信环路的方向；响应于接收到用于处理所述计算任务的请求，确定与所述请求相关联的计算资源的数量；以及基于所述使用状态和所述通信环路的方向，从所述多个计算资源中选择满足所述数量的计算资源的序列用于处理所述计算任务。

[0088] 根据本公开的一个示例性实现，基于使用状态和通信环路的方向选择计算资源的序列包括：基于使用状态和通信环路的方向建立拓扑关系，拓扑关系中的节点表示多个计算资源中的计算资源，以及拓扑关系中的有向边表示至少一部分计算资源之间的通信环路中的通信路径；基于序列中的节点构成的环路与拓扑关系中的已有环路之间的重叠程度，选择序列中的节点所表示的计算资源的序列。

[0089] 根据本公开的一个示例性实现，该动作进一步包括确定重叠程度。在此确定重叠程度包括：确定序列中的节点与已有环路中的节点的节点重叠程度以及序列中的节点之间的有向边与已有环路中的有向边的边重叠程度；以及基于节点重叠程度和边重叠程度确定重叠程度。

[0090] 根据本公开的一个示例性实现，确定节点重叠程度包括：针对序列中的当前节点，响应于确定当前节点被包括在已有环路中，提高当前节点的节点重叠程度。

[0091] 根据本公开的一个示例性实现，确定边重叠程度包括：针对序列中的两个节点之间的当前有向边，响应于确定当前有向边被包括在已有环路中，提高当前有向边的边重叠程度。

[0092] 根据本公开的一个示例性实现，该动作进一步包括：确定与已有环路中的有向边相关联的通信带宽；以及基于通信带宽更新有向边的边重叠程度。

[0093] 根据本公开的一个示例性实现，基于通信带宽更新有向边的边重叠程度包括：针对序列中的两个节点之间的当前有向边，响应于确定当前有向边所表示的通信路径的通信带宽降低，提高边重叠程度。

[0094] 根据本公开的一个示例性实现，从拓扑关系中选择与已有环路之间的重叠程度较低的序列。

[0095] 根据本公开的一个示例性实现，响应于计算资源池中的多个计算资源的使用状态出现变化，更新拓扑关系。

[0096] 根据本公开的一个示例性实现，多个计算资源是多个图形处理单元，以及方法进一步包括：在一组图形处理单元之间，基于AllReduce规则建立环形通信通道；以及向一组图形处理单元分配待处理数据；以及由一组图形处理单元处理待处理数据。

[0097] 根据本公开的一个示例性实现，提供了一种计算机程序产品。该计算机程序产品被有形地存储在计算机可读介质上并且包括机器可执行指令，机器可执行指令在被执行时使机器执行根据本公开所述方法。

[0098] 根据本公开的一个示例性实现，提供了一种计算机可读介质。计算机可读介质上存储有机器可执行指令，当机器可执行指令在被至少一个处理器执行时，使得至少一个处

理器实现根据本公开所述方法。

[0099] 本公开可以是方法、设备、系统和/或计算机程序产品。计算机程序产品可以包括计算机可读存储介质，其上载有用于执行本公开的各个方面的计算机可读程序指令。

[0100] 计算机可读存储介质可以是可以保持和存储由指令执行设备使用的指令的有形设备。计算机可读存储介质例如可以是——但不限于——电存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备或者上述的任意合适的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括：便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式压缩盘只读存储器(CD-ROM)、数字多功能盘(DVD)、记忆棒、软盘、机械编码设备、例如其上存储有指令的打孔卡或凹槽内凸起结构、以及上述的任意合适的组合。这里所使用的计算机可读存储介质不被解释为瞬时信号本身，诸如无线电波或者其他自由传播的电磁波、通过波导或其他传输媒介传播的电磁波(例如，通过光纤电缆的光脉冲)、或者通过电线传输的电信号。

[0101] 这里所描述的计算机可读程序指令可以从计算机可读存储介质下载到各个计算/处理设备，或者通过网络、例如因特网、局域网、广域网和/或无线网下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光纤传输、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配卡或者网络接口从网络接收计算机可读程序指令，并转发该计算机可读程序指令，以供存储在各个计算/处理设备中的计算机可读存储介质中。

[0102] 用于执行本公开操作的计算机程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或者以一种或多种编程语言的任意组合编写的源代码或目标代码，编程语言包括面向对象的编程语言—诸如Smalltalk、C++等，以及常规的过程式编程语言—诸如“C”语言或类似的编程语言。计算机可读程序指令可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络—包括局域网(LAN)或广域网(WAN)—连接到用户计算机，或者，可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。在一些实现中，通过利用计算机可读程序指令的状态信息来个性化定制电子电路，例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA)，该电子电路可以执行计算机可读程序指令，从而实现本公开的各个方面。

[0103] 这里参照根据本公开实现的方法、装置(系统)和计算机程序产品的流程图和/或框图描述了本公开的各个方面。应当理解，流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合，都可以由计算机可读程序指令实现。

[0104] 这些计算机可读程序指令可以提供给通用计算机、专用计算机或其他可编程数据处理装置的处理单元，从而生产出一种机器，使得这些指令在通过计算机或其他可编程数据处理装置的处理单元执行时，产生了实现流程图和/或框图中的一个或多个方框中规定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储介质中，这些指令使得计算机、可编程数据处理装置和/或其他设备以特定方式工作，从而，存储有指令的计算机可读介质则包括一个制造品，其包括实现流程图和/或框图中的一个或多个方

框中规定的功能/动作的各个方面的指令。

[0105] 也可以把计算机可读程序指令加载到计算机、其他可编程数据处理装置、或其他设备上，使得在计算机、其他可编程数据处理装置或其他设备上执行一系列操作步骤，以产生计算机实现的过程，从而使得在计算机、其他可编程数据处理装置、或其他设备上执行的指令实现流程图和/或框图中的一个或多个方框中规定的功能/动作。

[0106] 附图中的流程图和框图显示了根据本公开的多个实现的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分，模块、程序段或指令的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

[0107] 以上已经描述了本公开的各实现，上述说明是示例性的，并非穷尽性的，并且也不限于所公开的各实现。在不偏离所说明的各实现的范围和精神的情况下，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择，旨在最好地解释各实现的原理、实际应用或对市场中的技术的改进，或者使本技术领域的其他普通技术人员能理解本文公开的各实现。

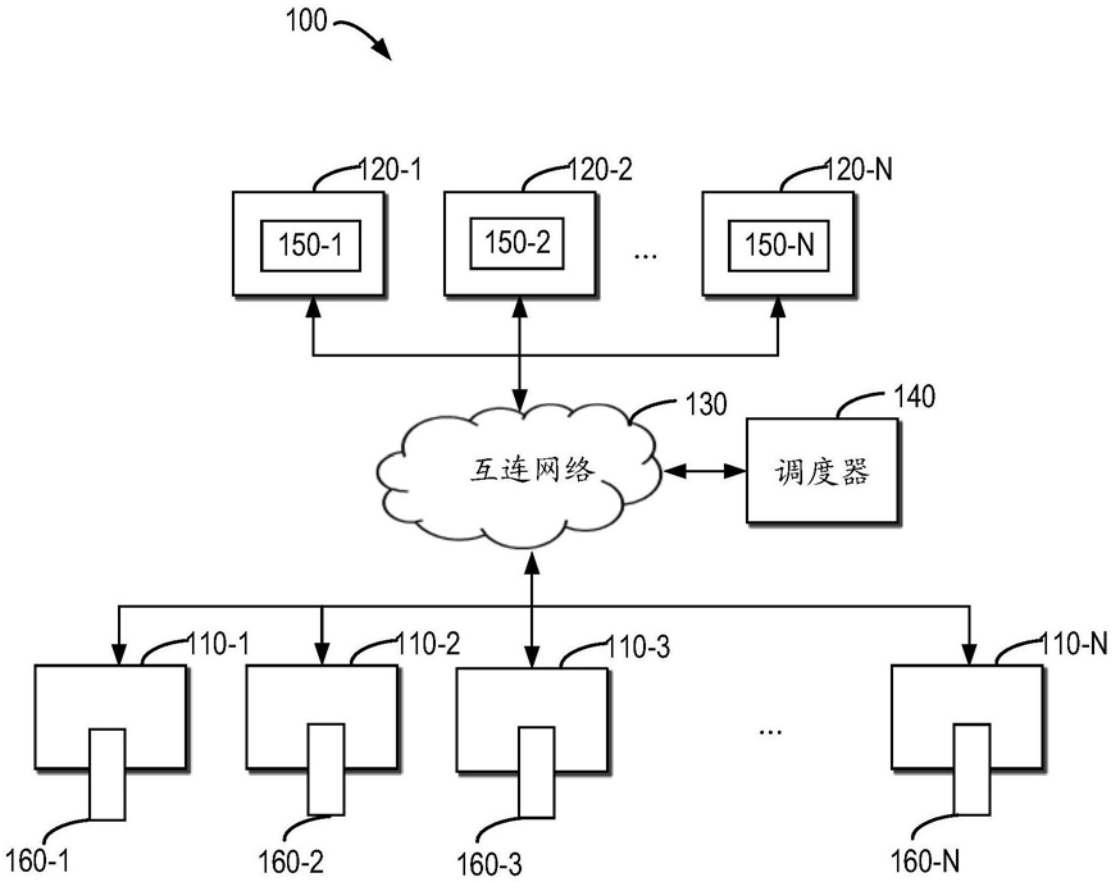

100
120-1
120-2
120-N
150-1
150-2
150-N
130
140
互连网络
调度器
110-1
110-2
110-3
110-N
160-1
160-2
160-3
160-N

图1

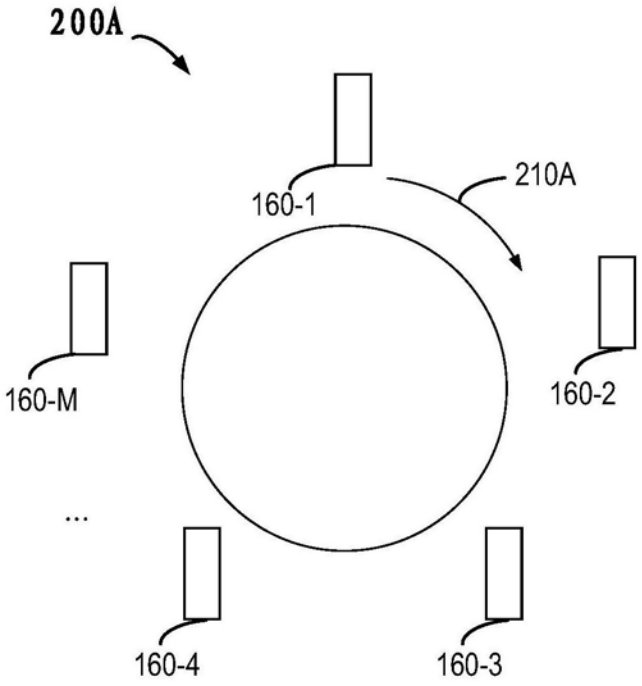

200A
160-1
210A
160-M
160-2
160-4
160-3

图2A

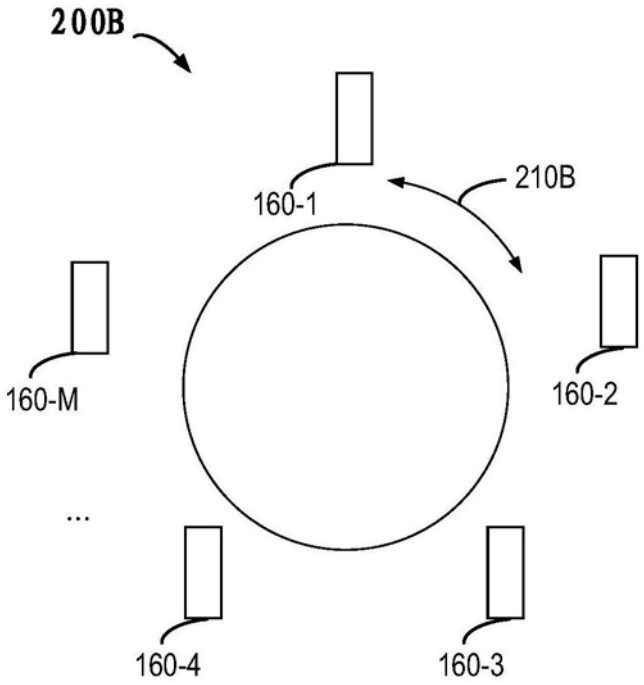


图2B

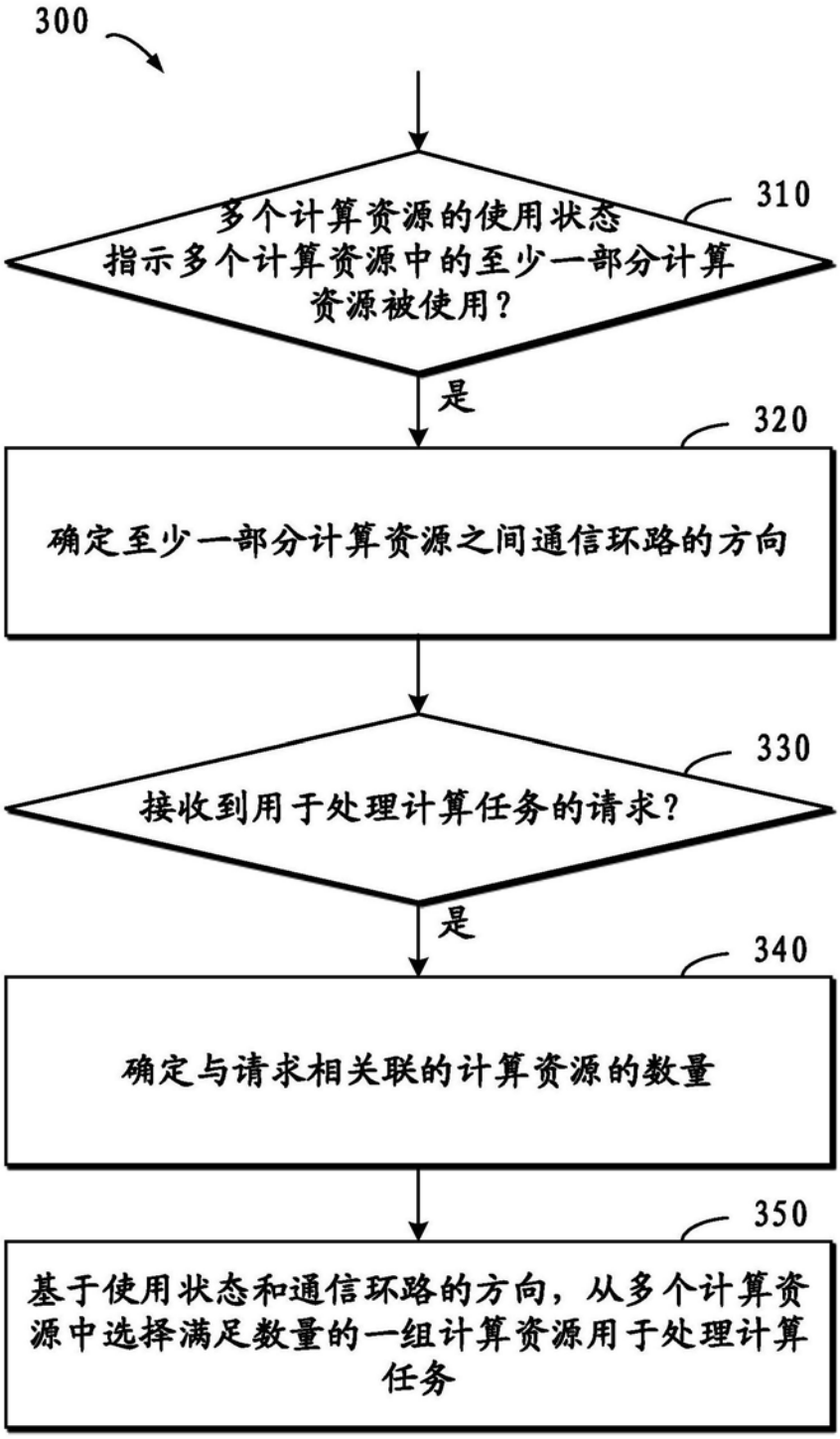

300
多个计算资源的使用状态指示多个计算资源中的至少一部分计算资源被使用？
310
是
320
确定至少一部分计算资源之间通信环路的方向
接收到用于处理计算任务的请求？
330
是
340
确定与请求相关联的计算资源的数量
350
基于使用状态和通信环路的方向，从多个计算资源中选择满足数量的一组计算资源用于处理计算任务

图3

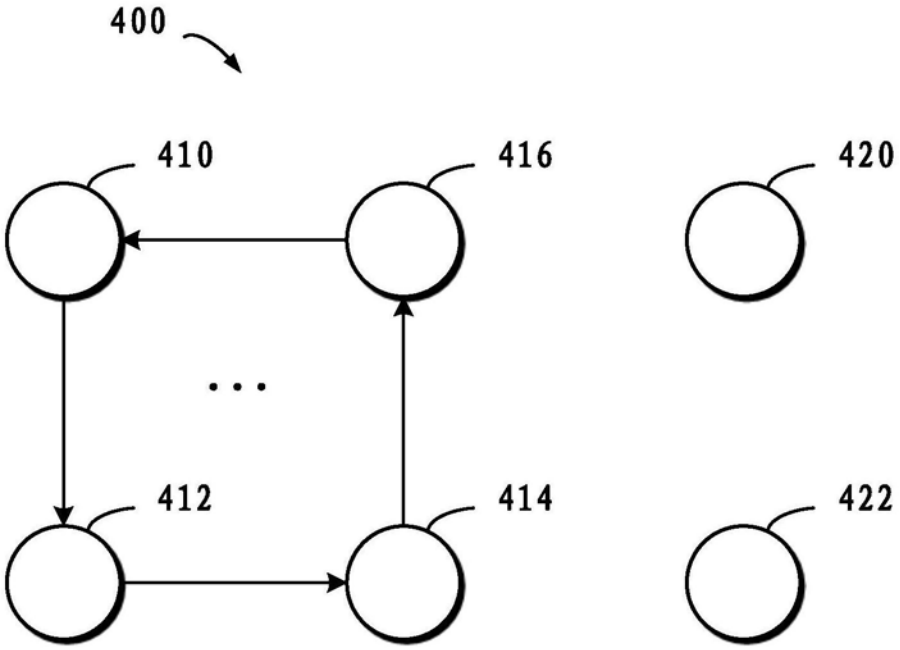

400
410
416
420
412
414
422

图4

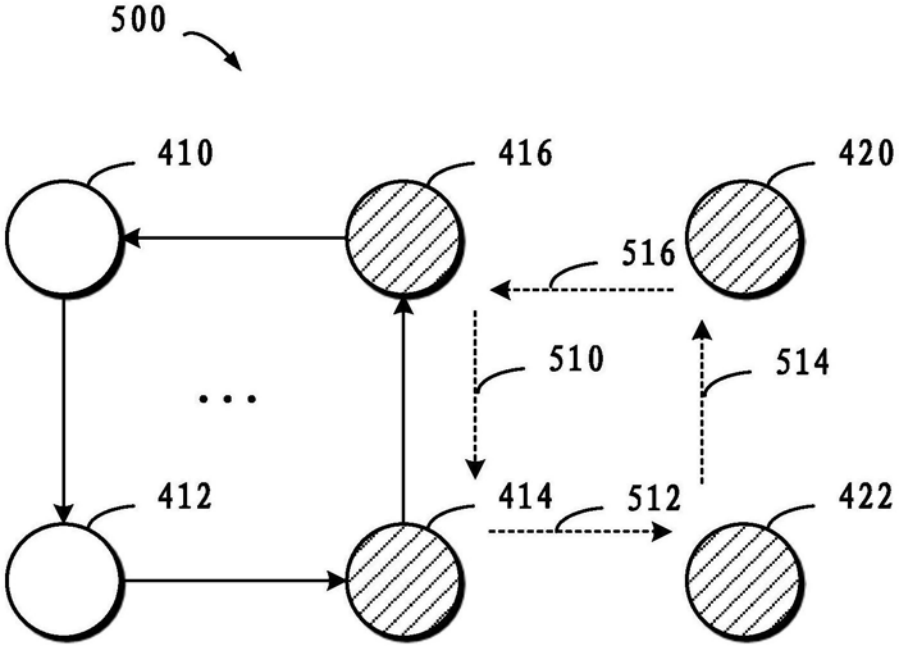

500
410
416
420
516
510
514
412
414
512
422

图5

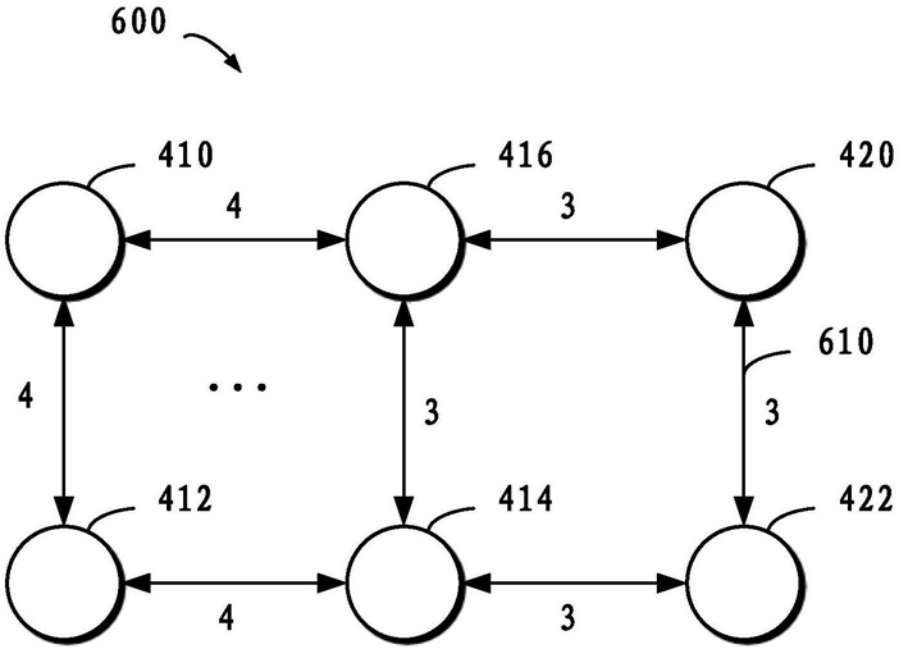

600
410
416
420
4
3
610
4
3
3
412
414
422
4
3

图6

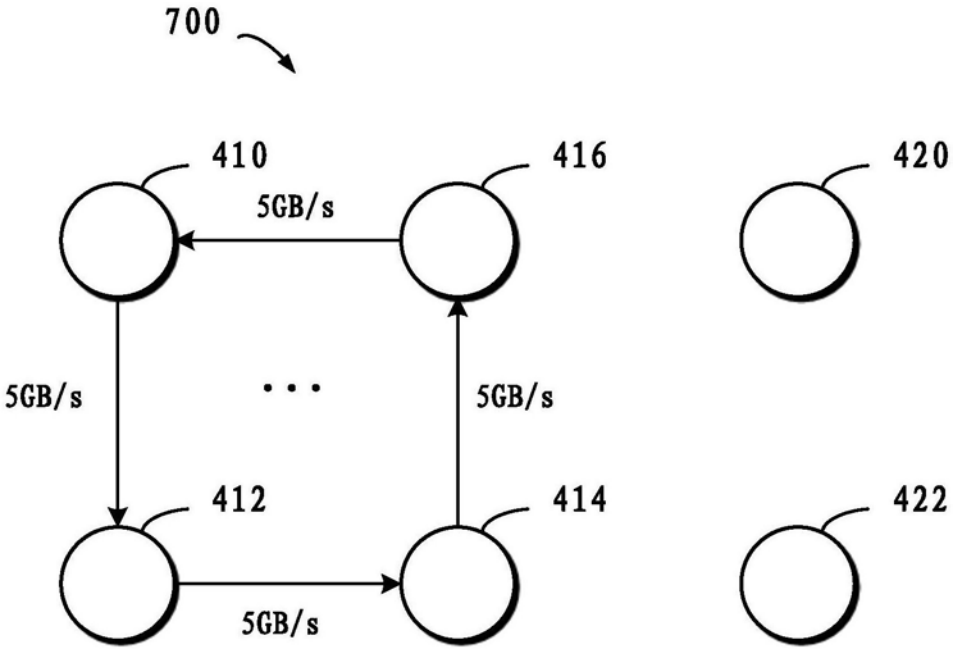

700
410
416
420
5GB/s
5GB/s
5GB/s
412
414
422
5GB/s

图7

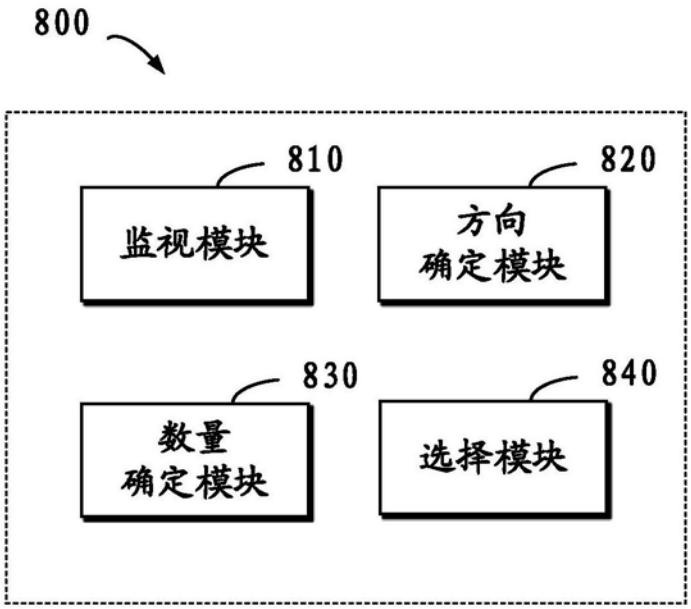

800
810
监视模块
820
方向
确定模块
830
数量
确定模块
840
选择模块

图8

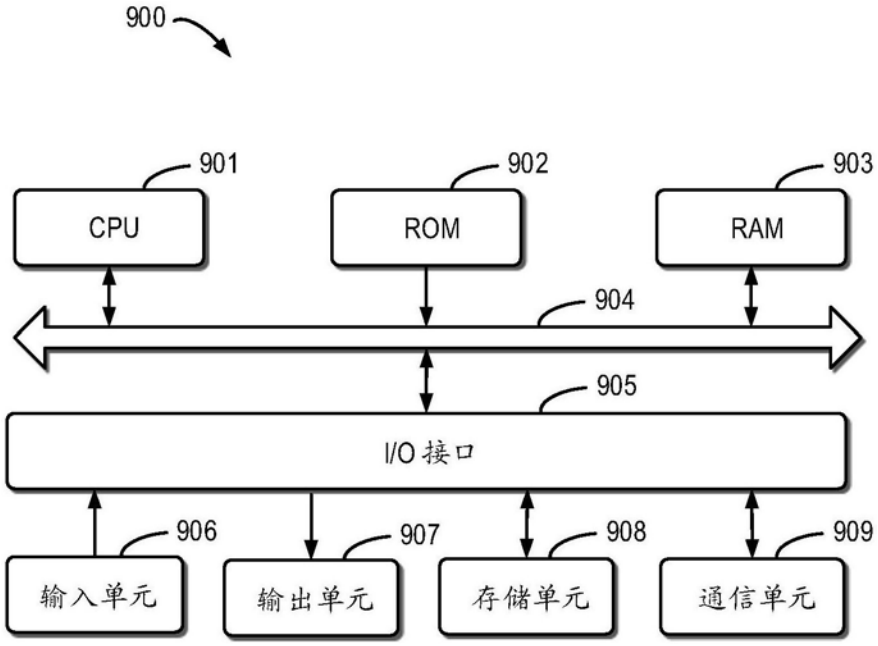

900
901
CPU
902
ROM
903
RAM
904
905
I/O 接口
906
输入单元
907
输出单元
908
存储单元
909
通信单元

图9