(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
2020 年 10 月 29 日 (29.10.2020)

WIPO | PCT

(10) 国际公布号
WO 2020/215752 A1

(51) 国际专利分类号:
G06F 16/901 (2019.01)

(21) 国际申请号: PCT/CN2019/125798

(22) 国际申请日: 2019 年 12 月 17 日 (17.12.2019)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201910335121.4 2019年4月24日 (24.04.2019) 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: 夏 应 龙 (**XIA, Yinglong**); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 张 晨 逸 (**ZHANG, Chenyi**); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 方 扬 (**FANG, Yang**); 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, 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,

(54) **Title:** GRAPH COMPUTING METHOD AND DEVICE

(54) 发明名称: 图计算方法及装置

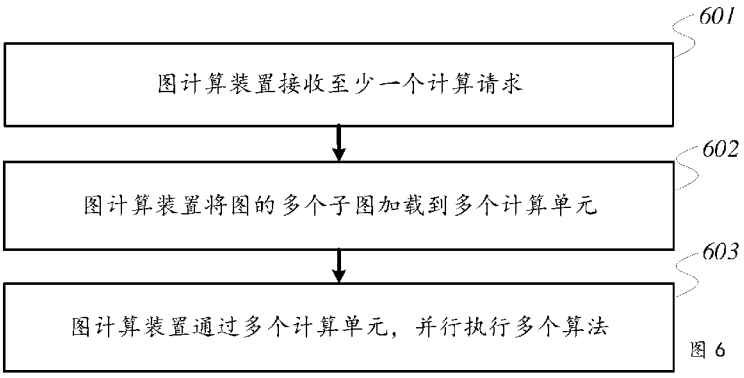

图 6

601 A graph computing device receives at least one computing request
602 The graph computing device loads multiple subgraphs of a graph into multiple computing units
603 The graph computing device concurrently executes multiple algorithms by means of the multiple computing units

(57) **Abstract:** A graph computing method and device, capable of supporting multiple algorithms for concurrently executing graph computing. Multiple subgraphs of a graph are loaded into multiple computing units, multiple algorithms are concurrently executed by means of the multiple computing units; a same graph can be shared among the multiple algorithms, and the multiple algorithms are concurrently executed on the same graph, so that the time delay caused by waiting for execution of other algorithms to finish when executing a certain algorithm is saved, and thus the overall efficiency of graph computing by multiple algorithms is improved and the overall time for graph computing by multiple algorithms is shortened.

WO 2020/215752 A1

[见续页]

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))。

(57) 摘要: 一种图计算方法及装置，能够支持多算法并发执行图计算。通过将图的多个子图加载到多个计算单元，通过多个计算单元，并行执行多个算法，可以在多个算法之间共享同一份图，从而在同一份图上并行执行多个算法，节省了执行某个算法时需要等待其他算法执行结束所造成的时延，从而提高了多算法进行图计算的整体效率，缩短了多算法进行图计算的整体时间。

图计算方法及装置

技术领域

本申请涉及计算机技术领域，尤其涉及一种图计算方法及装置。

背景技术

图是一种基础的数据结构，它包含一系列的节点以及连接节点的边。生活中的许多实体以及实体之间的关系可以通过图上的点和边来直观表达，于是，基于图的分析技术—图计算技术应运而生，应用图计算技术，可以将社交网络、通话网络、用户与产品之间的二分图、论文中作者之间的合作关系网、文章之间的索引关系、金融交易网络、一个地区的无线基站与云端各个服务器之间的交互、手游用户之间的数据交流关系等各种各样的实际应用场景建模为图，通过对图进行计算，来挖掘出图表示的实体所蕴含的规律。

随着图计算技术的推广，经常面临大量用户同时要求对图进行计算的情况，不同用户需要采用相同或不同的算法对图进行计算，因此计算设备会接收到批量化的计算请求，每个计算请求用于请求采用某一种或多种算法对图进行计算。计算设备根据计算请求，会以串行的方式，依次采用每个算法对图进行计算。具体来说，假设计算请求要请求采用N个算法对某个图进行计算，则计算设备会首先将这个图从外存加载到内存，采用算法1对图进行计算，当算法1执行结束，得到算法1的计算结果后，再将图从内存中释放；同理地，再次将图从外存加载到内存，采用算法2对图进行计算，当算法2执行结束，得到算法2的计算结果后，再将图从内存中释放，依次类推，当每个算法依次执行结束后，可以得到N个算法的计算结果。其中，N为正整数。

通过上述示例可以看出，采用串行的方法对图进行计算时，采用一个算法计算结束后，才能采用下一个算法进行计算，导致计算效率低下。

发明内容

本申请提供了一种图计算方法及装置，能够提升图计算的计算效率。

本申请的第一方面提供了一种图计算方法，执行该方法的可以而不限于是图计算装置，例如，图计算装置可以是分布云环境中的多个计算单元。该方法包括：

接收至少一个计算请求，所述至少一个计算请求用于请求采用多个算法对图进行计算；

将所述图的多个子图加载到多个计算单元；

通过所述多个计算单元，并行执行所述多个算法。

以上提供了一种能够支持多算法并发执行图计算的方法，通过将图的多个子图加载到多个计算单元，通过多个计算单元，并行执行多个算法，可以在多个算法之间共享同一份图，从而在同一份图上并行执行多个算法，节省了执行某个算法时需要等待其他算法执行结束所造成的时延，从而提高了多算法进行图计算的整体效率，缩短了

多算法进行图计算的整体时间。

并且，通过在多算法之间共享同一份图，多算法执行图计算时可以复用内存中已载入的图，而无需为每个算法分别执行将图加载到内存以及将图从内存中释放的步骤，也就避免了将同一个图反复地载入内存以及从内存中释放所造成的开销，节省了访问内存的时间，打破了输入/输出(英文全称：input/output，英文简称：IO)瓶颈。

在一种可能的实现方式中，所述通过所述多个计算单元，并行执行所述多个算法，包括：

获取每个算法中的至少一个任务；

通过所述多个计算单元，并行执行所述多个算法中的任务。

在一种可能的实现方式中，所述获取每个算法中的至少一个任务，包括下述至少一项：

根据所述算法中每个步骤对应的函数名称，将所述算法中同一函数名称对应的至少一个步骤划分为一个任务；

基于收集应用分发GAS模型，将所述算法划分为收集任务、应用任务以及分发任务；

基于整体通信同步BSP模型，将所述算法划分为本地任务、通信任务以及同步任务；

根据所述算法中每个步骤的执行主体，将所述算法中执行主体相同的步骤划分为一个任务；

根据所述算法中每个步骤对图中顶点或边的访问顺序，将所述算法中访问顺序相同的步骤划分为一个任务；

根据所述算法中每个步骤在图中访问的顶点或边，将所述算法中访问的顶点或边相同的步骤划分为一个任务；

将所述算法中的每一个步骤划分为一个任务；

根据所述算法中每个步骤执行的动作，将所述算法中动作相同的步骤划分为一个任务；

根据所述算法中每个步骤所属的迭代过程，将所述算法中属于同一次迭代过程的步骤划分为一个任务；

根据所述算法中每个步骤所属的判断分支，将所述算法中属于同一判断分支的步骤划分为一个任务。

以上提供的方法，通过将算法分解为任务，对于处理同一个图的不同算法，能够将具有相同数据访问方式的任务聚集在一起，打破了不同算法之间的隔阂，能够有效地暴露各个任务的数据访问的模式的相似性，从而利用而同一类型的任务对图的存取具有相似的规律性，对任务进行合理的调度和分配，从而合理地利用和调度系统资源，提供更高总体性能的服务。尤其是，应用在多用户多算法并发执行，通过对多种算法分解为任务，有助于对多种算法进行统一化管理。

在一种可能的实现方式中，所述将所述图的多个子图加载到多个计算单元，包括：

将至少一种模态的所述多个子图加载到所述多个计算单元；

所述通过所述多个计算单元，并行执行所述多个算法中的任务，包括：

对于所述多个算法的任一任务，通过所述多个计算单元，根据目标模态的子图执行所述任务，所述目标模态为所述至少一种模态中与所述任务匹配的模态。

在一种可能的实现方式中，所述任务包括从所述图中第一顶点搜索至所述图中第二顶点的步骤，所述目标模态的子图中所述第二顶点排在所述第一顶点之前。

以上提供的方法，考虑了不同类型的任务与不同模式的子图之间的亲近关系，实现了图的多模态管理，能够将任务的分配将与子图的模态关联起来，从而将任务分配给装载了合适的子图的计算单元。可以为任务提供相对优化的数据结构，改进在图计算领域中普遍存在的数据局部性有限的问题，从而提高计算任务的执行速度。具体地，由于收集任务要访问每个顶点的前序信息，而入边模态的子图中，正好将每个顶点的前序信息聚合在一起，因此可以提高数据局部性；同理地，对于分发任务来说，分发任务要向所有后序结点提供信息，那么将分发任务分配给装载了出边模态的计算单元进行处理，能提高数据局部性，从而提高计算效率。

在一种可能的实现方式中，所述通过所述多个计算单元，并行执行所述多个算法中的任务，包括：

根据所述多个算法的迭代次数，获取所述多个算法的优先级；

根据每个算法的优先级，获取调度方案，所述调度方案用于指示至少一个目标任务以及至少一个目标子图之间的对应关系，所述目标任务为所述多个算法的任务中本次调度的任务，所述目标子图为所述多个子图中的子图本次调度的子图；

通过加载了所述至少一个目标子图的多个计算单元，并行执行所述至少一个目标任务。

在一种可能的实现方式中，所述通过所述多个计算单元，并行执行所述多个算法中的任务，包括：

根据配置指令，获取所述多个算法的优先级，所述配置指令用于指示所述多个算法的优先级；

根据每个算法的优先级，获取调度方案，所述调度方案用于指示至少一个目标任务以及至少一个目标子图之间的对应关系，所述目标任务为所述多个算法的任务中本次调度的任务，所述目标子图为所述多个子图中的子图本次调度的子图；

通过加载了所述至少一个目标子图的多个计算单元，并行执行所述至少一个目标任务。

以上提供的方法，可以保证算法的优先级的定义方式可以由用户自定义设置，满足用户需求。也即是，有利于系统根据不同的调度目标，实施调度，使得系统具备良好的可配置能力。

在一种可能的实现方式中，所述通过所述多个计算单元，并行执行所述多个算法，包括下述至少一项：

对于所述多个算法中的任一算法，执行所述算法中的加载步骤之外的部分，所述加载步骤为将所述图加载到计算单元的内存的步骤；

对于所述多个算法中的任一算法，执行所述算法中的释放步骤之外的部分，所述释放步骤为将所述图从计算单元的内存中释放的步骤。

以上提供的方法，对于同一个子图来说，在该子图上执行多个算法的过程中，通

过删除了一个或多个算法中的释放步骤，可以避免执行该一个或多个算法时，要将子图从计算单元的内存释放的过程，也就节省了反复释放同一份子图造成的时间开销以及性能开销。

在一种可能的实现方式中，所述接收至少一个计算请求之前，所述方法还包括：

对所述图进行划分，得到所述多个子图；

将所述多个子图存入图存储装置；

所述将所述图的多个子图加载到至少一个多个计算单元，包括：

将所述图的多个子图从所述图存储装置加载到所述多个计算单元。

以上提供的方法，通过预先将图划分为多个子图并存储多个子图，如果接收到计算请求，可以直接将多个子图加载到该多个计算单元，而无需临时对图划分为子图，也就节省了临时对图划分为子图会造成的时延，可以加快执行多个算法的速度，提升执行多个算法的效率，有助于快速处理完成计算请求。

在一种可能的实现方式中，所述对所述图进行划分，包括：

根据所述多个计算单元的数量，对所述图进行划分。

以上提供的方法，可以支持向多个计算单元平均加载子图，即，将图的多个子图加载到计算单元时，不同计算单元载入的子图的数量相同或近似相同，从而保证每个计算单元根据载入的子图执行算法时，不同计算单元之间负载均衡。

在一种可能的实现方式中，所述方法还包括：

发送扩容请求，所述扩容请求用于请求对所述多个计算单元进行扩容；

接收扩容指令，所述扩容指令用于指示对所述多个计算单元进行扩容；

创建至少一个计算单元；

对所述图的至少一个子图进行复制，得到至少一个子图的实例；

将所述至少一个子图的实例加载到所述创建的至少一个计算单元；

通过所述创建的至少一个计算单元，并行执行所述多个算法。

在一种可能的实现方式中，扩容请求包括要新增的计算单元的数量以及扩容费用中的至少一项。

以上提供的方法，可以在图上并行执行多算法的过程中，自动地进行动态扩容，通过创建更多的计算单元来执行算法，可以提高系统整体的计算能力，从而缩短执行多算法的时延。另外，通过将执行算法所需负载的任务量分流到新增的计算单元上，能够实现系统的负载均衡。尤其是，应用在用户租用的多个计算单元不足以支持并行执行多算法的场景下，通过自动触发扩容请求，来提示用户租用更多的计算单元以执行算法，可以提高系统的自适应能力。

在一种可能的实现方式中，所述对所述图的至少一个子图进行复制，包括：

统计所述图中每个子图被所述多个算法请求的次数；

对所述图中请求次数达到阈值的子图进行复制。

以上提供的方法，考虑到单个图能够支持的访问数量具有物理上限的，如果图的访问数量超过物理上限，很可能在图上无法支持并行执行多个算法，也就对图算法的执行过程造成瓶颈。例如，如果某个子图是热点子图，比如说，该子图是包含了名人信息的子图，同一时刻很可能有大量用户访问该子图以查询名人信息，则该子图可能

无法支持被多个任务同时调度，也就无法并行执行多个算法，导致对图计算的整体速度造成限制。而上述实现方式中，图计算装置能够感知每个子图被请求的数量，如果某个子图的请求次数超过阈值，表明该子图的需求量很大，很可能是热点子图，那么通过触发该子图被复制为多份，分别部署在不同的计算单元上，由多个计算单元对该子图进行处理，能够提高这个子图的处理效率。也即是，通过让并发的计算请求能够分流至不同的实例上来进行计算，从而线性扩展了并发性。

本申请的第二方面提供了一种图计算装置，所述装置用于执行第一方面或第一方面的任意可能的实现方式提供的方法。具体地，该图计算装置包括用于执行第一方面或第一方面的任意可能的实现方式提供的方法的各个单元。

本申请的第三方面提供了一种图计算装置，包括多个计算单元，每个计算单元包括处理器和存储器，所述存储器中存储有至少一条指令，所述指令由所处理器加载并执行以实现第一方面或第一方面的任意可能的实现方式提供的方法。

本申请的第四方面提供了一种图计算系统，包括图计算装置和图存储装置；

所述图计算装置包括多个计算单元；

所述图存储装置用于存储图的多个子图；

所述图计算装置用于将所述多个子图从所述图存储装置加载至所述多个计算单元，以执行第一方面或第一方面的任意可能的实现方式提供的方法。

本申请的第五方面提供了一种非瞬态的可读存储介质，所述非瞬态的可读存储介质被图计算装置执行时，所述图计算装置执行前述第一方面或第一方面的任意可能的实现方式中提供的方法。该存储介质中存储了程序。该存储介质的类型包括但不限于易失性存储器，例如随机访问存储器，非易失性存储器，例如快闪存储器、硬盘（hard disk drive，HDD）、固态硬盘（solid state drive，SSD）。

本申请的第六方面提供了一种计算机程序产品，所述计算机程序产品被图计算装置执行时，所述图计算装置执行前述第一方面或第一方面的任意可能的实现方式中提供的方法。该计算机程序产品可以为一个软件安装包，在需要使用前述第一方面或第一方面的任意可能的实现方式中提供的方法的情况下，可以下载该计算机程序产品并在图计算装置上执行该计算机程序产品。

附图说明

为了更清楚地说明本申请实施例的技术方法，下面将对实施例中所需要使用的附图作以简单地介绍。

图 1 为本申请实施例提供的一种图计算系统的架构图；

图 2 为本申请实施例提供的一种图计算装置的功能架构图；

图 3 为本申请实施例提供的图计算装置的工作流程图；

图 4 为本申请实施例提供的图计算装置的另一种工作流程图；

图 5 为本申请实施例提供的图计算装置的另一种工作流程图；

图 6 为本申请提供的方法流程示意图；

图 7 为本申请提供的入边模态的子图以及出边模态的子图的示意图；

图 8 为本申请提供的算法分解以及多算法统一管理的示意图；

图 9 为本申请提供的一种多算法的任务调度的示意图；

图 10 为本申请提供的一种基于子图模态的任务调度的示意图；

图 11 为本申请提供的一种第一二分图以及第二二分图的示意图；

图 12 为本申请提供的一种执行迭代算法的流程图；

图 13 为本申请提供的图计算装置的一种结构示意图；

图 14 为本申请提供的图计算装置的另一种结构示意图；

图 15 为本申请提供的图计算装置的另一种结构示意图；

图 16 为本申请提供的图计算系统的一种结构示意图。

具体实施方式

下面结合本申请实施例中的附图，对本申请实施例中的技术方法进行描述。

本申请中的术语“单元”可以通过硬件实现，也可以通过硬件执行相应的软件实现，该硬件或软件具有执行下述方法中相应步骤的功能。例如，接收单元可以由收发器替代，以执行下述方法中的接收步骤，计算单元可以由处理器以及存储器替代，以执行下述方法中的执行步骤。

本申请中的术语“模块”可以为软件模块。

本申请中术语“第一”“第二”等字样用于对作用和功能基本相同的相同项或相似项进行区分，应理解，“第一”、“第二”、“第 n”之间不具有逻辑或时序上的依赖关系，也不对数量和执行顺序进行限定。

以下，介绍本申请使用的概念。

图（Graph）：为包括至少一个顶点以及至少一条边的数据结构。在一些场景中，图中的顶点可以映射为实体，图中的边可以映射为实体与实体之间的关系。图可以是有向图或无向图。当然，图还可以包括顶点以及边以外的其他数据，例如顶点的标签以及边的标签等。在一个示例性场景中，应用于好友推荐的场景中，图中的每个顶点可以表示一个用户，图中的每条边可以表示不同用户之间的社交关系，图中每个顶点的数据为用户的画像数据以及用户的行为数据，例如用户的年龄、职业、爱好、学历等。又如，应用于在商品推荐的场景中，图中的每个顶点可以表示一个用户或一个商品，图中的每条边可以表示用户与商品之间的交互关系，例如购买关系、收藏关系等。又如，应用于金融风控的场景中，图中的每个顶点可以表示账号、交易或资金。图中的边可以表示资金的流动关系，例如图中的环路可以表示循环转账。再如，应用于企业网络优化的场景中，图中的每个顶点可以表示一个网元，例如路由器、交换机、终端等，图中的每条边可以表示不同网元之间的连接关系。

算法：包括对图的顶点和/或边的一系列操作，以对图的顶点和/或边进行统计、排序、路径选择等运算。作为示例，算法可以是网页排名算法（也称 PageRank 算法），Pixie 随机游走算法、广度优先搜索（英文全称：Breadth-First Search，英文简称：BFS）算法、深度优先搜索（英文全称：Depth-First-Search）算法、个性化网页排名算法（也称 PersonalRank 算法）、k 核（英文：k-core）算法、k 跳（英文：k-hop）算法、最短路径（英文：shortest paths）算法，全最短路径（英文：all shortest paths）算法、关联路径算法、紧密中心度算法、标签传播算法、基于模块度的社区发现算法

（英文：Louvain 算法）、关联预测算法、node2vec 算法（一种把网络中的节点映射到欧式空间的算法）、实时推荐算法、共同邻居算法、单源最短路算法、联通分量算法、三角计数算法、聚类系数算法等。

子图（英文：subgraph）：为图的一部分，包括图中的部分顶点以及部分边。子图也可以称为图中的分区（英文：partition）。一个图可以包括多个子图。

收集应用分发（英文全称：gather apply scatter，英文简称：GAS）模型：一种算法的编程规范，按照 GAS 模型，可以将算法划分为三种任务，分别是收集(英文：gather)任务、应用(英文：apply)任务以及分发(英文：scatter)任务。

收集任务可以包括从图中顶点的相邻顶点获取数据的步骤以及获取图中边的数据的步骤，收集任务可以视为工作顶点从相邻顶点和自身收集数据的过程，例如，收集任务可以为对图中各条边的数据进行求和，又如，收集任务可以是计算某个顶点相邻的顶点的数量。在执行收集任务的过程中，点以及边可以是只读状态。

应用任务可以包括更新图中顶点的数据的步骤以及将图中顶点的数据同步至顶点的镜像顶点的步骤。例如，应用任务可以是各计算顶点计算出来的同一顶点的相邻顶点的数量。应用任务可以是镜像顶点将收集任务的计算结果发送给主顶点，主顶点汇总多个镜像顶点的计算结果，利用汇总结果和上一步的顶点数据，按照业务需求进行进一步的计算,然后更新主顶点的数据并同步到镜像顶点。在执行应用任务的过程中，工作顶点可以是可修改的状态，边可以为不可修改的状态。

分发任务可以包括向图中顶点的相邻顶点发送数据的步骤以及更新图中边的数据的步骤。例如，分发任务可以是工作顶点对数据更新完成之后，更新边上的数据，并通知与工作顶点存在依赖关系的相邻顶点更新状态的过程。在执行分发任务的过程中，工作顶点可以为只读状态，边可以为可写状态。

整体通信同步（英文全称：bulk synchronous parallel computing model，英文简称：BSP）模型：一种算法的编程规范，按照 BSP 模型，可以将算法划分为三种任务，分别是本地任务、通信任务以及同步任务。

本地任务：也称本地计算（英文：local computation）任务可以包括算法中根据本端的数据进行计算的步骤，例如，本地任务可以包括单个处理单元需要完成的计算过程。

通信（英文：communication）任务可以包括算法中需要不同计算单元进行交互的步骤。例如，通信任务可以包括算法中处理单元 A 将子图的计算结果发送至处理单元 B 的步骤。

同步任务可以包括算法中等待该通信任务结束的步骤，例如，同步任务可以包括栅栏同步（barrier synchronization）的过程。

模态（英文：modality）：是指数据的形式，或者说数据的表达方式。对于同一数据来说，该数据的不同模态在形式上有所区别，而包含的实质信息相同。在本申请实施例中，子图可以具有至少一种模态。举例来说，对于某一个子图来说，模态 1 的该子图包含的顶点和模态 2 的该子图包含的顶点相同，模态 1 的该子图包含的边和模态 2 的该子图包含的边相同，而模态 1 的该子图包含的顶点的排列顺序和模态 2 的该子图包含的顶点的排列顺序可以不同。比如说，如果子图包括顶点 1、顶点 2……顶点

100，模态 1 的该子图以及模态 2 的该子图均包含顶点 1、顶点 2……顶点 100，而模态 1 的该子图中顶点 1 排在顶点 2 的前面，而模态 2 的该子图中顶点 1 排在顶点 2 的后面。

入边模态：是指基于入边构建的子图的模态，入边模态用于指示子图中不同顶点之间的入边关系。具体来说，子图中每条边的终点可以排在前，每条边的起点可以排在后。例如，如果子图中包括顶点 A、顶点 B 以及顶点 C，这 3 个顶点之间的关系是：顶点 A 与顶点 B 之间存在一条边，且边的方向是顶点 A 指向顶点 B，顶点 A 与顶点 C 之间存在另一条边，且边的方向是顶点 A 指向顶点 C，则入边模态的子图中顶点 A 可以排在前，顶点 B 和顶点 C 可以排在后。

出边模态：是指基于出边构建的子图的模态，出边模态的子图可以包括至少一条出边，具体来说，子图中每条边的起点可以排在前，每条边的终点可以排在后。例如，如果子图中包括顶点 A、顶点 B 以及顶点 C，这 3 个顶点之间的关系是：顶点 A 与顶点 B 之间存在一条边，且边的方向是顶点 A 指向顶点 B，顶点 A 与顶点 C 之间存在另一条边，且边的方向是顶点 A 指向顶点 C，则出边模态的子图中顶点 B 和顶点 C 可以排在前，顶点 A 可以排在后。

以下，示例性介绍本申请的系统架构。

图 1 为本申请实施例提供的一种图计算系统的架构图，该图计算系统包括图计算装置 101 以及终端 102，终端 102 用于向图计算装置 101 发送计算请求，图计算装置 101 用于接收终端 102 的计算请求，执行下述方法实施例提供的图计算方法，将计算结果发送至终端 102。

如图 1 所示，图计算装置 101 可以包括一个或多个计算单元，不同计算单元可以执行下述方法实施例中相同或不同的步骤。在一些可能的实施例中，计算单元可以为物理设备；例如，计算单元可以是物理服务器、中央处理器（英文全称：central processing unit、英文简称：CPU）等。在另一些可能的实施例中，计算单元可以为虚拟化设备；例如，计算单元可以是虚拟机、容器、pod（pod 是指 Kubernetes 中运行、管理、编排容器化应用时容器运行的基本单位）、处理器的核心等等。在另一些可能的实施例中，计算单元可以为软件中的基本单位，例如，计算单元可以为应用、服务、微服务、模块、子模块、类或函数等。其中，图计算装置 101 中的一个或多个计算单元的规格可以相同，例如一个或多个计算单元可以包括相同数量的处理器以及相同容量的内存，当然一个或多个计算单元的规格也可以存在差异，本实施例对此不做限定。

多个计算单元可以通过单机实现，也可以组成分布式系统。在一些可能的实施例中，多个计算单元可以运行在同一物理设备上，不同计算单元可以通过物理设备内部的通讯网络进行通信。例如，多个计算单元可以是一台物理服务器中的多个虚拟机或多个容器；再如，多个计算单元可以是一台物理服务器中通过总线进行通信的多个处理器或者同一个处理器中的多个内核。在另一些可能的实施例中，图计算装置 101 中的不同计算单元可以运行在不同物理设备上，比如可以运行在不同的地点、不同的计算中心、不同的机房、不同的机架等，图计算装置 101 中的不同物理设备可以通过网

络交互。

在一些可能的实施例中，图计算装置 101 可以作为云计算服务向用户提供，例如，可以作为图引擎服务（英文全称：Graph Engine Service，英文简称：GES）向用户提供。图计算装置 101 可以运行在云环境中，例如可以运行在公有云、私有云或混合云上。作为示例，图计算装置 101 可以为弹性云服务器（英文全称：elastic cloud server，英文简称：ECS）集群，图计算装置 101 中的每个计算单元为一个 ECS；又如，图计算装置 101 可以为虚拟机集群，图计算装置 101 中的每个计算单元为一个在云环境中运行的虚拟机；又如，图计算装置 101 可以提供为云容器引擎（英文全称：cloud container engine，英文简称：CCE），图计算装置 101 中的每个计算单元为一个在云环境中运行的容器；又如，图计算装置 101 可以提供为云服务平台（英文：cloud service stage），图计算装置 101 中的每个计算单元为一个在云环境中运行的应用、服务或微服务。

应理解，图计算装置 101 运行在云环境中仅是示例，图计算装置 101 也可以运行在边缘环境中，图计算装置 101 中的每个计算单元可以为边缘环境中的边缘计算设备；图计算装置 101 也可以运行在边缘环境中，图计算装置 101 中的每个计算单元可以为终端环境中的终端设备；本实施例对图计算装置 101 的运行环境不做限定。另外，图计算装置 101 的各个计算单元也可以分别运行在不同环境中。例如，图计算装置 101 可以在云环境、边缘环境、终端环境中的三个，或在其中任意两个环境上运行图计算装置 101 的部分计算单元。

终端 102 可以为手机、笔记本、服务器、台式电脑等。终端 102 可以和图计算装置 101 通过网络进行交互。

在一些可能的实施例中，该图计算系统还可以包括图存储装置 103，该图存储装置 103 用于为图计算装置 101 提供存储图的服务。其中，该图存储装置 103 可以通过单机实现，也可以组成分布式系统。例如，该图存储装置 103 可以为分布式存储器。其中，图存储装置 103 可以通过云存储服务实现，该图存储装置 103 可以运行在云环境中，例如可以运行在公有云、私有云或混合云上。作为示例，图存储装置 103 可以为对象存储服务（英文全称：object storage service，英文简称：OBS）、云硬盘、云数据库等。

图 1 所示的图计算系统可以提供为大数据引擎，该图计算装置 101 以及该图存储装置 103 可以提供为大数据引擎按照逻辑功能划分的不同层次。例如，该图计算装置 101 可以提供为计算层，该图存储装置 103 可以提供为存储层。

需要说明的一点是，上文仅是以图计算装置 101 以及图存储装置 103 为两个分离的装置为例进行说明，在一些可能的实施例中，图计算装置 101 以及图存储装置 103 可以集成在一起，也即是，图计算装置 101 可以同时具有对图进行计算以及对图进行存储的功能。

以下，示例性介绍本申请的应用场景。

本申请实施例提供的图计算方法，可以应用于各种与图计算、图分析、图查询等各种相关的线上和/或线下平台，也可以封装为各种不同的应用。其中，下述方法实施

例提供的处理逻辑可以作为平台或应用内置的图计算功能，使得平台和/或应用可以在公共安全、金融风控、反欺诈、社交媒体、根因分析、数字资产管理、数据溯源等很多领域发挥作用。

尤其是，应用在云平台中，由于随着图分析的日益推广，对图进行分析的算法越来越丰富；同时，图的数据规模也不断增加；而云平台上通常会同时有多个，甚至多组用户/租户在使用平台，各组用户的图不尽相同，经常面临大量的用户需要同时使用多种分析算法在大规模的图上进行计算，例如，对于同一租户下的多个用户来说，这些用户的终端可以并发地请求采用不同算法对同一张图进行计算；又如，不同的租户可以通过终端请求使用相同的算法对不同的图进行计算。而通过实施本申请提供的方法，云平台可以支持在同一份图上并发执行多个算法的功能，一方面，不同算法通过并发执行，可以提升云平台的计算效率以及计算速度，从而提升云平台的性能。另一方面，不同算法可以共享同一份图，使得图的装载次数不会随着算法数量的增加而增加，打破了输入/输出(英文全称：input/output，英文简称：I/O)瓶颈。

在一个示例性场景中，可以用于在社交媒体类的企业中进行数据分析和洞察。在此场景中，企业的终端可以向图计算装置发送社交网络图，图计算装置将社交网络图存入图存储装置。终端可以向图计算装置发送计算请求，计算请求用于请求采用网页排名算法（以下简称算法P）和宽度优先遍历算法（以下简称算法B）对社交网络图进行计算，图计算装置接收到计算请求后，可以将社交网络图从图存储装置加载到该多个计算单元，多个计算单元可以在社交网络图上，并行执行算法P和算法B，得到算法P的结果以及算法B的结果，将算法P的结果以及算法B的结果返回给企业的终端。其中算法P的结果为社交网络图中的每个顶点的得分，该得分为0到1之间的浮点数，该得分表示根据社交网络图的结构计算出的顶点的重要性；而算法B的结果则是从给定的根节点R出发，在社交网络图上为每个顶点找到一个前序顶点，从而形成一棵从根节点R出发的树，将所有可达的顶点关联起来。

以下，示例性介绍本申请提供的图计算装置的逻辑功能架构。

参见图2，按照图2中从上至下的顺序来说，如果终端1、终端2以及终端3同时向图计算装置发送了计算请求1、计算请求2以及计算请求3，计算请求1、计算请求2以及计算请求3分别指示采用算法1、算法2以及算法3对同一图进行计算。图计算装置接收到计算请求1、计算请求2以及计算请求3以后，可以调用算法1、算法2以及算法3，将图的多个子图加载到处理单元1、处理单元2以及处理单元3的内存中，生成算法与图之间的二分图，根据该二分图生成子图与任务之间的二分图，根据该二分图生成子图与任务之间的调度方案，按照该调度方案，通过处理单元1、处理单元2以及处理单元3执行算法1、算法2以及算法3的任务，以执行调度方案。

以下，示例性介绍本申请的图计算装置的工作流程。

参见图3，其示出了本申请提供的图计算装置的一种工作流程图。图3所示的图计算装置包括请求缓存模块、调度模块、分布式任务执行模块、第一二分图构建模块、第二二分图构建模块、任务生成模块、子图模态管理模块，数据注入模块、模式编辑模块、分区模块、子图模态生成模块。

其中，图 3 中的模块可以用于执行图 2 涉及的步骤。例如，第一二分图构建模块可以用于执行图 2 中生成算法与图之间的二分图的步骤，第二二分图构建模块可以用于执行图 2 中生成子图与任务之间的二分图的步骤，调度模块可以用于执行图 2 中生成调度方案的步骤，分布式任务执行模块可以为图 2 中的处理单元 1、处理单元 2 以及处理单元 3，分布式任务执行模块可以用于执行图 2 中执行调度方案的步骤。

从图 3 左侧可见，图计算装置可以具有两个输入和一个输出。其中一个输入为原始数据，该原始数据用于构建图，原始数据经由数据注入模块注入图计算装置中；在数据注入的过程中，模式编辑模块从数据注入模块获取原始数据，根据终端输入的模式（英文：schema）信息，将原始数据构建为图。例如，模式编辑模块可以根据大量的用户数据，构建出表示社交网络的图。分区模块用于对构建的图进行分区，每个分区形成一个子图。之后，每个子图通过子图模式生成模块产生一种或多种模态的子图，子图模态管理模块可以对每个模态的子图进行管理，另外每种模态的每种子图可以被持久化存储至该图存储装置，以供后续对图进行计算的时候调用。

其中，模式信息用于指示构建出图的语义的方式，即定义图中顶点和边的数据项的方法。其中，该模式信息可以包括顶点声明、边声明、格式属性、模式声明等。

而图计算装置的另一个输入则是单个用户或用户群触发的计算请求，该计算请求也称一个查询，该计算请求指示了图和要执行的算法，例如包含算法的名称和图的名称。计算请求会输入请求缓存模块，请求缓存模块会对计算请求进行缓存，以便批量化处理。根据计算请求指示的算法以及图，可以通过第一二分图构建模块，生成第一二分图，即算法与任务之间的二分图，该第一二分图会通过任务生成模块，将算法根据其实现依据的编程模型，分解成一系列的任务，如基于 GAS 模型生成收集任务、应用任务以及分发任务。生成的任务通过子图模态管理模块，可以与匹配的模态的子图关联，生成第二二分图，即任务和子图之间的二分图。调度模块可以根据第二二分图，调度该分布式任务执行模块执行任务。其中，考虑到一些算法是迭代算法，会随着不断的迭代而动态产生新的任务，调度模块有一个输出箭头指向请求缓存模块，以便动态向请求缓存模块添加新的迭代的任务。分布式任务执行模块用于执行多个算法。另外，分布式任务执行模块可以监控迭代算法的任务，将这些任务的中间结果进行保存和累计，当迭代算法迭代终止后，向用户输出最终的结果。

图计算装置的输出为多个算法的计算结果，可以反馈给用户。

在图 3 的基础上，参见图 4，图 4 是图 3 的一个具体实例，图 4 示出了图计算装置采用网页排名算法、BFS 算法以及随机游走算法，对图进行计算的过程。与图 3 相区别的是，图 4 提供的图计算装置不仅包含图 3 中的各个模块，还包含数据清洗模块、读取模块、写入模块、可视化模块、读取模块、调度模块。其中，图 4 与图 3 共有的模块以及工作流程还请参见上述对图 3 的描述，以下描述图 4 相对于图 3 来说特有的模块和工作流程。

数据清洗模块用于对数据注入模块中的原始数据进行清洗，例如过滤掉噪声数据等，数据清洗模块输出的数据可以通过模式编辑模块构建为图。 读取模块用于在图计算过程中，从图存储装置读取图的多个子图。写入模块用于将图的多个子图写入图存储装置。可视化模块用于对计算单元计算出的结果进行可视化，例如可以生成用户界

面（英文全称：User Interface，英文简称：UI）显示指令，UI 显示指令用于指示终端显示包括计算结果的将用户界面，可以将UI 显示指令发送至终端。调度模块用于根据多个算法中的任务以及多个子图进行调度，从而通过计算单元执行任务。

在图 3 以及图 4 的基础上，参见图 5，其示出了本申请提供的图计算装置的另一种工作流程图。与图 4 区别的是，图 4 所示的是采用多个算法对单个图进行并行计算的流程，而图 5 示出了采用多个算法对多个图并行进行计算的流程。并且，与图 3 和图 4 相区别是，图 5 提供的图计算装置不仅包含图 3 以及图 4 中的各个模块，图 5 还包含负载均衡模块。其中，图 5 与图 4 以及图 3 共有的模块以及工作流程，还请参见上述对图 3 以及图 4 的描述，以下描述图 5 相对于图 3 以及图 4 来说特有的模块和工作流程。

负载均衡模块用于对多个计算单元进行负载均衡。具体来说，如图 5 所示，租户 1 对应的用户 1、用户 2……用户 n，以及租户 2 对应的用户 1、用户 2……用户 n 可以通过各自的终端同时触发计算请求，图计算装置可以同时接收到两个租户对应的两组用户的终端的请求，其中 n 为正整数。在此场景下，图 5 所示的负载均衡模块可以根据多个计算单元的 CPU 使用率、空闲内存或者其他信息，确定多个计算单元的计算能力可能不足以支持处理大量的计算请求产生的任务，为此，负载均衡模块可以向终端发送扩容请求，终端接收到扩容请求后，返回扩容指令，负载均衡模块可以对计算单元进行扩容后，可以将子图复制为多个实例，将这些实例加载到新增的计算单元，以使新增的计算单元分担任务量。

如图 6 所示，介绍图计算方法的流程，该方法包括由图计算装置执行的步骤 601 至 603。

步骤 601、图计算装置接收至少一个计算请求。

至少一个计算请求用于请求采用多个算法对图进行计算。具体来说，每个计算请求可以用于请求采用一个或多个算法对图进行计算。在一些可能的实施例中，计算请求可以包括算法的标识以及图的标识。算法的标识用于指示对应的算法，例如可以为算法的名称、身份标识符（英文全称：identification，英文简称：ID）、编号等。图的标识用于指示对应的图，例如可以为算法的 ID、名称、编号等。另外，计算请求还可以包括用户标识，用户标识用于指示对应的用户，例如可以为用户的 ID、名称等。

在一些可能的实施例中，一个或多个终端可以生成一个或多个计算请求，向图计算装置发送一个或多个计算请求，图计算装置可以接收一个或多个终端的至少一个计算请求。例如，终端可以运行在终端环境中，图计算装置可以运行在云环境中，终端和图计算装置可以通过彼此的公网互联网协议地址（英文全称：internet protocol address，英文简称：IP）地址交互计算请求。

其中，不同终端的计算请求所请求的图可以相同或不同，不同终端的计算请求所请求的算法可以相同或不同。例如，应用在图 4 所示的单图多算法的场景中，在同一时刻可以有多个用户要求使用不同的算法在同一个图上进行计算，则多个用户的终端可以并发向图计算装置发送计算请求，图计算装置可以同时接收到多个终端的计算请求，通过执行下述步骤 602 至步骤 603，来在同一个图上并行执行多个算法，以处理

多个终端的计算请求。又如，应用在图 5 所示的多图多算法的场景中，在同一时刻可以有多个租户的多个用户要使用不同的算法在相同或不同图上进行计算，则多个租户的多个用户的终端可以并发向图计算装置发送计算请求，图计算装置可以同时接收到多个终端的计算请求，通过执行下述步骤 602 至步骤 603，来在多个图中的每个图上并行执行对应的多个算法，以处理多个终端的计算请求。

需要说明的一点是，图计算装置可以为计算单元集群，步骤 601 的执行主体可以为计算单元集群中的某一个或多个计算单元。在一些可能的实施例中，计算单元集群中可以包括一个或多个控制面的计算单元，这一个或多个控制面的计算单元用于接收至少一个计算请求。例如，该控制面的计算单元可以为计算单元集群中的主节点、客户端节点等。在另一些可能的实施例中，计算单元集群中的每个计算单元均可以接收至少一个计算请求。

步骤 602、图计算装置将图的多个子图加载到多个计算单元。

图计算装置可以从计算请求中获取图的标识，根据图的标识对应的图，确定图的多个子图，将多个子图加载到该多个计算单元的内存，则加载完成后，每个计算单元的内存会缓存有多个子图。

图计算装置可以将图的全部的子图或部分子图加载到计算单元。具体地，如果图中总共包括 N 个子图，对于每个计算单元，图计算装置可以将 N 个子图均加载到该计算单元；另外，图计算装置也可以将 M 个子图加载到该计算单元，其中 N 和 M 为正整数，且 M 小于 N。

在一些可能的实施例中，图计算装置可以确定计算单元的存储容量，如果计算单元的存储容量足够，例如存储容量大于预设阈值，则图计算装置可以将全量的子图加载到计算单元；如果计算单元的存储容量的存储容量不足，例如存储容量小于预设阈值，则图计算装置可以将部分子图加载到计算单元。

作为示例，图计算装置可以采用平均加载的方式，可以获取子图的总数量与计算单元的总数量之间的比值，作为每个计算单元要加载的子图的数量，将该数量个子图加载到每个计算单元，如此不同计算单元加载的子图数量可以相等或近似相等。例如，如果图中包括 N 个子图，计算单元包括 M 个，则对于每个计算单元，可以将 N/M 个子图加载到该计算单元，其中/表示相除。

在一些可能的实施例中，图的多个子图可以预存在图存储装置中，图计算装置可以将多个子图从图存储装置加载到多个计算单元。例如，图计算装置可以预先将图划分为多个子图，将该多个子图存入图存储装置中，根据多个子图在图存储装置中的存储位置，生成图的标识以及多个子图的存储位置之间的对应关系，存储该对应关系。当接收至少一个计算请求，可以根据图的标识，从对应关系查询到该多个子图的存储位置，从图存储装置中的该存储位置，读取多个子图，将多个子图从图存储装置加载到多个计算单元。

其中，图计算装置可以预先将图划分为多个子图，将划分得到的多个子图存入图存储装置中。作为示例，图存储装置可以存储有数据库，该数据库用于存储图的子图，当图计算装置划分得到子图后，可以将子图存入该数据库中，如果接收到计算请求，则从该数据库读取图的多个子图。

通过预先将图划分为多个子图并存储多个子图，如果接收到计算请求，可以直接将多个子图加载到该多个计算单元，而无需临时对图划分为子图，也就节省了临时对图划分为子图会造成的时延，可以加快执行多个算法的速度，提升执行多个算法的效率，有助于快速处理完成终端的计算请求。

当然，预先将图划分为子图仅是示例，在另一些可能的实施例中，如果图计算装置接收到计算请求，可以确定计算请求指示的图，对图进行划分，得到多个子图。

可选地，如果图计算装置接收到计算请求，图计算装置可以查询图存储装置是否存储有图的子图，如果图存储装置存储有图的多个子图，则将多个子图加载到计算单元;如果图存储装置未存储有图的多个子图,则将计算请求指示的图划分为多个子图，将多个子图加载到计算单元，另外，可以将多个子图存入图存储装置，以便下次接收到对该图的计算请求时，从图存储装置加载子图。

通过这种实现方式，图计算装置可以随着计算的图的数量的增长，多次向图存储装置存入新的图的子图，从而扩大图存储装置中子图的数据量，可以动态地维护图的子图。

在另一些可能的实施例中，图计算装置可以在接收到计算请求后，再将图划分为多个子图，本实施例对将图划分为子图的时机不做限定。

关于对图划分为子图的过程，图计算装置可以对图进行分区（partition），每个分区形成一个子图。在一些可能的实施例中，图计算装置可以根据多个计算单元的数量，对图进行划分。作为示例，图计算装置可以将图划分为计算单元的数量的整数倍的子图。比如说，如果多个计算单元的总数量为K个，图计算装置可以将图划分为K倍个子图，K为正整数。

通过这种实现方式，可以支持向多个计算单元平均加载子图，保证将子图加载到计算单元时，不同计算单元载入的子图的数量相同或近似相同。作为示例，比如用户租用了5个虚拟机，可以将图划分为10个子图，如果接收到计算请求，可以向每个虚拟机加载2个子图。

作为示例，划分得到的子图的总数量可以根据图的大小以及多个计算单元的能力中的至少一项确定。划分得到的子图的总数量可以与图的大小正相关。图的大小越大，则子图的总数量可以越大。划分得到的子图的总数量可以与多个计算单元的能力负相关。计算单元的能力越强，则子图的总数量可以越小，则单个子图的数据量越大。其中，计算单元的能力可以包括计算单元的处理能力以及存储能力，该处理能力可以通过计算单元的主频、核数等指标表示，该存储能力可以通过计算单元的存储容量等指标表示。

在一些可能的实施例中，子图的总数量可以作为能够在图计算装置配置的参数，该参数可通过系统的配置（英文：profiler）工具调整，或通过对图计算装置的历史数据进行分析来调整，该参数可以配合各类图划分（英文：graph partitioning）算法使用。

在另一些可能的实施例中，图计算装置也可以按照对图备份存储的方式，将图划分为多个子图。例如，如果备份存储时，将图划分为M个数据块，可以将M个数据块中的每个数据块作为一个子图，其中M为正整数。

在一些可能的实施例中，图的每个子图可以包括至少一种模态，图计算装置可以将图的至少一种模态的多个子图加载到多个计算单元。其中，图计算装置可以将图的全量的模态的子图加载到多个计算单元。例如，如果子图包括W种模态，对于每个计算单元，图计算装置可以将图的W种模态的多个子图分别加载到计算单元，其中W为正整数。

以子图包括入边模态和出边模态这2种模态为例，如果图中总共包括n个子图，则图总共包括（2*n）个不同模态的子图，该（2*n）个不同模态的子图分别是入边模态的子图1、出边模态的子图1、入边模态的子图2、出边模态的子图2……入边模态的子图n、出边模态的子图n。对于任一计算单元来说，如果图计算装置要将图的全量的子图加载到计算单元，则图计算装置会将(2*n)个不同模态的子图加载到计算单元；如果图计算装置要采用平均加载的方式，将图的子图加载到计算单元，并假设计算单元共计m个，则图计算装置会将（2*n/m）个不同模态的子图加载到计算单元；其中，n和m为正整数，且m小于n，*表示相乘，/表示相除。

在一些可能的实施例中，图计算装置可以预先将图划分为多个子图；对于每个子图，生成至少一种模态的该子图；将至少一种模态的多个子图预存在图存储装置中；如果接收到计算请求，图计算装置可以从图存储装置加载到少一种模态的多个子图。例如，参见图7，对于图中的每个子图，图计算装置可以生成入边模态的子图以及出边模态的子图，将两种模态的该子图均存入图存储装置中。其中，图7中的实心顶点与空心顶点表示子图中不同顶点之间的从属关系（英文：ownership），以区分子图本身包含的顶点以及顶点的镜像。

在一些可能的实施例中，图计算装置可以从计算请求中获取多个算法的标识，根据算法的标识对应的算法，确定多个算法，调用该多个算法。其中，调用多个算法的过程与步骤602可以顺序执行。作为示例，可以先调用多个算法，再执行步骤602；也可以先执行步骤602，再调用多个算法。当然，调用多个算法的过程与步骤602也可以并行执行，本实施例对调用多个算法的过程与步骤602的顺序不做限定。

在一个示例性场景中，应用于云计算领域，图计算装置可以将多个子图加载到用户租用的全部计算单元；另外，图计算装置也可以从用户租用的全部计算单元中选择部分计算单元，将多个子图加载到该部分计算单元。

其中，图计算装置可以在接收到计算请求后，在集群中临时创建至少一个计算单元，将多个子图加载到创建的至少一个计算单元；另外，图计算装置也可以在接收到计算请求时，上电集群中处于休眠状态的多个计算单元，将多个子图加载到上电的多个计算单元；另外，图计算装置也可以在接收到计算请求时，确定集群中处于运行态的多个计算单元，将多个子图加载到多个计算单元，本实施例对实施中采用哪种实现方式不做限定。

需要说明的一点是，步骤602的执行主体可以就是要加载子图的多个计算单元，即，步骤602可以包括：多个计算单元加载图的多个子图。步骤602的执行主体也可以图计算装置中要加载子图的计算单元以外的计算单元。例如，图计算装置可以为计算单元集群，计算单元集群包括控制面的计算单元以及数据面的计算单元，控制面的计算单元可以将多个子图加载到数据面的计算单元。例如，控制面的计算单元可以生

成控制指令，向数据面的计算单元发送控制指令，控制指令用于指示加载多个子图，数据面的计算单元可以接收控制指令，加载图的多个子图。其中，该数据面的计算单元可以为计算单元集群中的从节点。作为示例，图计算装置中控制面的计算单元可以从计算请求中获取用户标识，查询该用户标识租用的数据面的计算单元，将多个子图加载到该数据面的计算单元。

步骤 603、图计算装置通过多个计算单元，并行执行多个算法。

多个计算单元可以根据已加载的至少一个子图，并行执行多个算法，也即是，可以在至少一个子图上同时执行多个算法。其中，多个计算单元在执行多个算法时可以复用同一个子图。具体来说，如果任两个或两个以上的算法的输入是同一个子图，则计算单元可以根据该子图，同时执行该两个或两个以上的算法。例如，如果算法 1、算法 2 以及算法 3 均需要对子图 1 进行计算，则多个计算单元可以根据子图 1，并行执行算法 1、算法 2 以及算法 3。

通过在至少一个子图上并行执行多个算法，达到的效果包括而不限于下述两个方面：

一方面，对于任一个子图来说，当根据该子图执行某个算法时，可以同时执行其他算法，从而避免了该算法需要等待其他算法执行结束所造成的时延，缩短了多算法进行图计算的整体时间，提高了多算法进行图计算的整体效率。例如，在根据子图 1 执行算法 1 时，可以同时执行算法 2 以及算法 2，避免了执行算法 2 时等待算法 1 执行结束造成的时延，也避免了执行算法 3 时等待算法 2 执行结束造成的时延。

另一方面，通过在多算法之间共享同一份图，多算法执行图计算时可以复用内存中已载入的图，而无需为每个算法分别执行将图加载到内存以及将图从内存中释放的步骤，也就避免了将同一个图反复地载入内存以及从内存中释放所造成的开销，节省了访问内存的时间，打破了 IO 瓶颈。例如，在根据子图 1 执行算法 1 以及算法 2 以及算法 3 时，可以为算法 1、算法 2 以及算法 3 执行一次加载子图 1 的步骤，避免在已经为算法 1 执行一遍加载子图 1 的过程的情况下，再次为算法 2 执行一遍加载子图 1 的过程，也避免了再次为算法 3 执行一遍加载子图 1 的过程，从而避免反复加载同一个子图 1 造成的时延。另外，可以当算法 1、算法 2 以及算法 3 均执行结束后，再释放子图 1，避免在已经为算法 1 执行一遍释放子图 1 的情况下，再次为算法 2 执行一遍释放子图 1 的过程，也避免了再次为算法 3 执行一遍释放子图 1 的过程，从而避免反复释放同一个子图 1 造成的时延。

在一些可能的实施例中，步骤 603 可以包括下述步骤一至步骤二：

步骤一、对于多个算法中的每个算法，图计算装置获取算法中的至少一个任务。

任务也称算子，任务为算法中的部分步骤，每个算法可以包括一个或多个任务。例如，如果算法包括 Q 个步骤，任务可以是算法中的 P 个步骤，P 大于或等于 1 且小于或等于 Q，P 和 Q 为正整数。同一算法可以按照不同的功能，划分为不同任务。例如，算法可以包括收集任务、应用任务以及分发任务；又如，算法可以包括本地任务、通信任务以及同步任务。

图计算装置通过获取算法中的至少一个任务，可以将算法分解为粒度更细的任务，以便通过对任务进行管理和调度，来对相应的算法进行管理和调度。

在一些可能的实施例中，步骤一可以包括下述方式一至方式九中的任一项或多项的组合。可选地，图计算装置可以判断算法是否符合编程模型，如果算法符合编程模型，则执行下述方式一，如果算法不符合编程模型，则执行下述方式二至至方式九。其中，编程模型是指编写算法的规范，例如，编程模型可以为 Pregel 模型（一种以顶点为中心的编程模型）、GAS 模型、BSP 模型等。算法符合编程模型，是指算法包含的函数名称和编程模型中规定的函数名称一致，例如，如果算法符合 GAS 算法，则算法中会包含函数名称为“gather”的函数、函数名称为“ apply”的函数，函数名称为“ scatter”的函数；算法不符合编程模型，是指算法中的函数名称和编程模型中规定的函数名称不一致，例如算法包含的函数名称是用户自定义的函数名称。

方式一、图计算装置根据算法中每个步骤对应的函数名称，将算法中同一函数名称对应的至少一个步骤划分为一个任务。

作为示例，图计算装置可以预存至少一个函数名称，或者从算法中读取至少一个函数名称；图计算装置可以根据至少一个函数名称，将同一函数名称对应的至少一个步骤划分为一个任务。例如，可以获取算法中函数名称“gather”对应的步骤，作为收集任务，获取算法中函数名称“apply”对应的步骤，作为应用任务，获取算法中函数名称“scatter”对应的步骤，作为分发任务。

方式二、图计算装置根据算法的编程模型，将算法划分为多个任务。

方式二可以包括下述方式（2.1）方式（2.2）中的任一项或多项的组合。

方式（2.1）图计算装置基于 GAS 模型，将算法划分为收集任务、应用任务以及分发任务。

作为示例，图计算装置可以根据算法中每个步骤的处理逻辑，将处理逻辑为从图中顶点的相邻顶点获取数据的步骤以及获取图中边的数据的步骤划分为收集任务，将处理逻辑为更新图中顶点的数据的步骤以及将图中顶点的数据同步至顶点的镜像顶点的步骤划分为应用任务，将处理逻辑为更新图中顶点的数据的步骤以及将图中顶点的数据同步至顶点的镜像顶点的步骤划分为分发任务。

方式（2.2）图计算装置可以基于 BSP 模型，将算法划分为本地任务、通信任务以及同步任务。

作为示例，图计算装置可以根据算法中每个步骤的处理逻辑，将处理逻辑为根据本端的数据进行计算的步骤划分为本地任务；将处理逻辑为与本端以外的其他设备进行交互的步骤划分为通信任务；将处理逻辑为等待该通信任务结束的步骤划分为同步任务。

在一些可能的实施例中，对于采用同一编程模型的多个算法来说，图计算装置可以根据该编程模型，将多个算法划分为多个任务，则不同算法的多个任务的类型相同。例如，如果算法 1 以及算法 2 均符合 GAS 模型，则图计算装置可以将算法 1 划分为收集任务、应用任务以及分发任务，将算法 2 也划分为收集任务、应用任务以及分发任务。

需要说明的一点是，基于 GAS 模型或 BSP 模型将算法划分为任务仅是示例，在一些可能的实施例中，算法可以基于 GAS 模型或 BSP 模型以外的其他编程模型实现，相应地，可以基于其他编程模型将算法划分为任务，本实施例对划分算法所采用的编程

模型不做限定。

示例性地，图 8 示出了算法分解以及多算法统一管理的示意图，图计算装置执行的任一算法通常为算法的一个实例，即基于一个具体的算法（如最短路径算法或子图匹配算法）与一个给定的编程模型的软件实现。如图 8 所示，实线方框、虚线方框以及加粗方框分别代表不同的算法，每个算法通过一个编程模型实现成为一个图分析应用的实例。以 GAS 模型为例，根据这种编程模型，各类算法都能够被分解成三种任务的组合，即收集任务、应用任务以及分发任务，每个任务按一定规律作用在图上。这三个算子的执行过程，形成图计算过程的一次迭代；其中，分发任务指向收集任务的箭头表示算法是由一系列迭代组成的。由此可见，可以把在一个图计算平台上各式各样的算法都拆解成一系列的任务，这些任务将会被分配到云上或者线下的计算单元上去。由于每个算法被分解成更小的任务，多个算法的执行可以混在一起进行。对于不同算法，同一类型的任务的数据访问方式是一致的，例如算法 1、算法 2 以及算法 3 的收集任务的数据访问方式是一致的，算法 1、算法 2 以及算法 3 的分发任务的数据访问方式是一致的。

通过将算法分解为任务，对于处理同一个图的不同算法，能够将具有相同数据访问方式的任务聚集在一起，打破了不同算法之间的隔阂，能够有效地暴露各个任务的数据访问的模式的相似性，从而利用而同一类型的任务对图的存取具有相似的规律性，对任务进行合理的调度和分配，从而合理地利用和调度系统资源，提供更高总体性能的服务。尤其是，应用在多用户多算法并发执行，通过对多种算法分解为任务，有助于对多种算法进行统一化管理。

方式三、图计算装置可以根据算法中每个步骤的执行主体，将算法中执行主体相同的步骤划分为一个任务。

步骤的执行主体是指执行步骤的硬件或软件，在本实施例中，执行主体可以指计算单元，也可以指计算单元包含的具有计算能力的计算单位。举例来说，计算单元可以是虚拟机，执行主体可以是虚拟机中的线程，如果算法中的多个步骤要通过虚拟机中的 T 个线程分别执行，可以将同一个线程执行的步骤划分为一个任务。

方式四、图计算装置可以根据算法中每个步骤对图中顶点或边的访问顺序，将算法中访问顺序相同的步骤划分为一个任务。

例如，可以将算法中从顶点访问至前续顶点的步骤划分为一个任务，将算法中从顶点访问至后继顶点的步骤划分为另一个任务。

方式五、图计算装置可以根据算法中每个步骤在图中访问的顶点或边，将算法中访问的顶点或边相同的步骤划分为一个任务。

例如，如果算法中包含 5 个步骤，步骤 1 以及步骤 2 要访问顶点 1、顶点 2 至顶点 1000，步骤 3 以及步骤 6 要访问顶点 1001、顶点 1002 至顶点 6000，步骤 5 要访问顶点 6001、顶点 6002 至顶点 7000，则可以将步骤 1 以及步骤 2 划分为任务 1，将步骤 3 以及步骤 6 划分为任务 2，将步骤 5 划分为任务 3。

方式六、图计算装置可以将算法中的每一个步骤划分为一个任务。

例如，如果算法中包含 H 个步骤，可以将算法划分为 H 个任务，每个任务包括 1 个步骤，其中 H 为正整数。

方式七、图计算装置可以根据算法中每个步骤执行的动作，将算法中动作相同的步骤划分为一个任务。

例如，如果算法中包含3个步骤，步骤1执行获取图中数据1的动作，步骤2要执行获取图中数据2的动作，步骤3要执行数据1和数据2求取平均值的动作，则可以将步骤1以及步骤2划分为任务1，将步骤3划分为任务2。

方式八、图计算装置可以根据算法中每个步骤所属的迭代过程，将算法中属于同一次迭代过程的步骤划分为一个任务。

例如，如果算法中包含R次迭代过程，可以将第1次迭代过程的所有步骤划分为任务1，将第2次迭代过程的所有步骤划分为任务2，将第3次迭代过程的所有步骤划分为任务3，依次类推。其中R为正整数。

方式九、图计算装置可以根据算法中每个步骤所属的判断分支，将算法中属于同一判断分支的步骤划分为一个任务。

例如，如果算法中包括：如果数据大于阈值，则对数据执行处理步骤1、处理步骤2以及处理步骤3,如果数据不大于阈值,则对数据执行处理步骤6以及处理步骤5，则可以将处理步骤1、处理步骤2以及处理步骤3划分为任务1，将处理步骤6以及处理步骤5划分为任务2。

在一些可能的实施例中，对于多个算法中的任一算法，图计算装置可以执行该算法中的加载步骤之外的部分，该加载步骤为将该图加载到计算单元的内存的步骤。例如，可以将加载步骤之外的部分划分为至少一个任务，通过该多个计算单元，并行执行划分出的至少一个任务。

通过这种实现方式，对于同一个子图来说，在该子图上执行多个算法的过程中，通过避免执行该一个或多个算法时，要将子图加载到计算单元的内存的过程，也就节省了反复加载同一份子图造成的时间开销以及性能开销。

在一些可能的实施例中，对于多个算法中的任一算法，图计算装置可以执行该算法中的释放步骤之外的部分。其中,释放步骤为将图从计算单元的内存中释放的步骤。例如，可以将释放步骤之外的部分划分为至少一个任务，通过该多个计算单元，并行执行划分出的至少一个任务。

通过这种实现方式，对于同一个子图来说，在该子图上执行多个算法的过程中，通过避免执行该一个或多个算法时，要将子图从计算单元的内存释放的过程，也就节省了反复释放同一份子图造成的时间开销以及性能开销。

步骤二、图计算装置通过多个计算单元，并行执行多个算法中的任务。

作为示例，对于任一计算单元来说，该计算单元可以在同一个子图上，并行执行多个算法中不同算法中的任务。其中，图计算装置可以将多个算法中的任务分配给该多个计算单元，每个计算单元可以并行执行图计算装置分配的任务。

在一些可能的实施例中，图计算装置可以创建任务池，当获取到多个算法的任务后，图计算装置可以将多个算法的任务缓存在该任务池中；图计算装置可以从任务池中选取一个或多个任务，将一个或多个任务分配给该多个计算单元；每个计算单元可以并行执行图计算装置分配的任务。当多个计算单元对分配的任务执行结束后，图计算装置可以从任务池中选取剩余的一个或多个任务，将剩余的一个或多个任务分配给

该多个计算单元，每个计算单元可以再次并行执行图计算装置分配的任务，依次类推，直至任务池为空，即多个算法的任务均已分配为止。其中，在分配任务的过程中，每个任务可以被分配且仅被分配到一个模态的一个子图上，而每个模态的每个子图可以被分配一个或多个任务。

示例性地，参见图 9，图 9 上方的实线框包括算法 1、算法 2……算法 k，以及图 1、图 2……图 n，它表示算法和图之间的对应关系，反映了并发执行环境中一组待执行的算法和每个算法上需要在哪些图上进行处理；图 9 下方的实线框中的虚线框表示任务和子图之间的对应关系，是对图 9 上方的实线框的细粒度的表达，它反映了并发执行环境中一组待执行的任务和每个任务上需要在哪个模态的哪个子图上进行处理；在分配任务的过程中，可以视为从虚线框中选择出一组边的过程。其中，k 和 n 为正整数。

通过上述步骤一以及步骤二，将不同的算法分解为了一系列的任务，让多个算法可以充分地混在一起执行，从而更好地支持了多个算法的并发执行。同时，由于任务的粒度比算法的粒度更细，提供了更多的调度空间，能够更好的利用已经加载的子图，使其服务更多并发分析的算法，提升批量处理时的系统总体性能，

在一些可能的实施例中，图计算装置可以将子图的模态与任务的分配过程关联起来。具体来说，步骤二可以包括下述步骤(2.1)至步骤(2.2)：

步骤(2.1)图计算装置可以将图的至少一种模态的多个子图加载到该多个计算单元。例如，对于多个计算单元中的每个计算单元，图计算装置可以将入边模态的子图和出边模态的子图加载到该计算单元。

步骤(2.2)对于多个算法中的任一任务，根据目标模态的子图执行该任务。

目标模态为至少一种模态中与任务匹配的模态。作为示例，目标模态的子图中顶点的排列顺序可以任务中对顶点的搜索顺序一致。在一些可能的实施例中，以图中的两个顶点为例，不妨称这两个顶点分别为第一顶点以及第二顶点，任务可以包括从图中第一顶点搜索至图中第二顶点的步骤，即，第一顶点是任务中先访问的顶点，第二顶点是任务中后访问的顶点。目标模态的子图中第二顶点排在第一顶点之前，即，在子图中第二顶点排在前，第一顶点排在后。

在一些可能的实施例中，考虑到图上信息的更新，通常来自沿着边流动的数据以及在顶点上进行的计算。而沿着边流动的数据包括两个层面：1）各个顶点沿着入边方向从其前续顶点中获取最新的数据，用于更新该顶点的本地信息；2）每个顶点将其最新的本地信息沿着出边的方向传播到后继结点。因此，可以将子图表达为两个模态，入边模态表示子图的入边关系，出边模态表示子图的出边关系。

相应地，步骤（2.2）可以包括：根据入边模态的子图，执行收集任务；根据出边模态的子图，执行分发任务。其中，可以根据入边模态的同一子图，并行执行多个算法的收集任务；根据出边模态的同一子图，并行执行多个算法的分发任务。例如，参见图 10，可以根据入边模态的同一子图，并行执行算法 1、算法 2 以及算法 3 的收集任务。

通过上述步骤(2.1)至步骤(2.2)，考虑了不同类型的任务与不同模式的子图之间的亲近关系（英文：affinity），实现了图的多模态管理，能够将任务的分配将与子

图的模态关联起来，从而将任务分配给装载了合适的子图的计算单元。可以为任务提供相对优化的数据结构，改进在图计算领域中普遍存在的数据局部性(英文：data locality)有限的问题，从而提高计算任务的执行速度。例如，基于GAS模型构建的算法中，算法分解出的任务中，收集任务要访问每个顶点的前序信息，而入边模态的子图中，正好将每个顶点的前序信息聚合在一起，因此可以提高数据局部性；同理地，对于分发任务来说，分发任务要向所有后序结点提供信息，那么将分发任务分配给装载了出边模态的计算单元进行处理，能提高数据局部性，从而提高计算效率。

另外，在入边模态的子图和出边模态的子图之间，可通过小批量(英文:mini-batch)同步机制，为镜像节点提供信息更新。

在一些可能的实施例中，图计算装置可以获取任务与子图之间的调度方案，按照调度方案来并行执行多个算法的任务。具体来说，步骤603可以包括下述步骤A至步骤C：

步骤A、图计算装置获取多个算法的优先级。

在一些可能的实施例中，优先级的获取方式包括而不限于下述方式一至方式二中任一项或多项的结合：

方式一、图计算装置根据配置指令，获取多个算法的优先级，配置指令用于指示该多个算法的优先级。

在一些可能的实施例中，配置指令可以携带每个算法的重要性系数，可以根据配置指令，图计算装置可以获取每个算法的重要性系数，根据每个算法的重要性系数，获取每个算法的优先级。其中，算法的优先级可以与算法的重要性系数正相关。重要性系数表示算法的重要性，重要性系数越大，表示算法越重要，越需要被及时调度。重要性系数可以通过归一化，将取值范围控制在0至1的区间中。

方式二、图计算装置根据多个算法的迭代次数，获取多个算法的优先级。

图计算装置可以获取多个算法的迭代次数，对多个算法的迭代次数进行运算，得到多个算法的优先级。其中，算法的迭代次数为已采用算法对图进行计算的次数，即算法当前已执行的次数。算法的迭代次数可以小于或等于算法总共需要计算的次数。在一些可能的实施例中，如果期望新任务可以尽快得到结果，可以获取与迭代次数负相关的数值，作为算法的优先级，那么由于新任务的迭代次数较少，能够得到优先调度，从而尽快被计算单元执行。在另一些可能的实施例中，如果期望已经执行了较长时间的老任务可以尽快得到结果，可以获取与迭代次数正相关的数值，作为算法的优先级，那么由于老任务的迭代次数较多，能够得到优先调度，从而尽快执行。

作为示例，图计算装置可以按照下述公式（1）、公式（2）以及公式（3）中的任一项或多项的组合，对多个算法的重要性系数以及多个算法的迭代次数进行运算，得到多个算法的优先级。

$$a_{iA,jG} = \delta_{iA,jG} * \rho_{\langle A,G\rangle} * e^{-t\langle A,G\rangle}; \quad (1)$$

$$a_{iA,jG} = \delta_{iA,jG} * \rho_{\langle A,G\rangle} * (1 - e^{-t\langle A,G\rangle}); \quad (2)$$

$$a_{iA,jG} = 1; \quad (3)$$

其中，$a_{iA,jG}$表示算法A对图G进行计算的优先级，ρ<A,G>为采用算法A对图G进行计算的重要性系数，ρ<A,G>进行了归一化，ρ<A,G>大于0且小于或等于1,t<A,G>为算法A当前对图G进行计算的迭代次数；δ<iA,jG>为狄拉克（Dirac Delta）函数，如果计算请求指示采用算法A对图G进行计算，则δ<iA,jG>取1，如果计算请求未指示采用算法A对图G进行计算，则δ<iA,jG>取0，*表示相乘。

在一些可能的实施例中，用户可以通过配置指令，控制新任务优先调度还是老任务优先调度。具体来说，如果用户期望新任务优先调度，可以触发用于指示图计算装置按照公式（1）运算的配置指令，图计算装置接收到配置指令后，会按照公式（1）来获取算法的优先级，使得算法的优先级与算法的迭代次数负相关。如果用户期望老任务能够得以优先调度，可以触发用于指示图计算装置按照公式(2)运算的配置指令，图计算装置接收到配置指令后，会按照公式（2）来获取算法的优先级，使得算法的优先级与算法的迭代次数正相关。如此，充分考虑了图被迭代算法执行时的场景，可以保证算法的优先级的定义方式可以由用户自定义设置，满足用户需求。也即是，有利于系统根据不同的调度目标，实施调度，使得系统具备良好的可配置能力。

步骤B、图计算装置根据多个算法中每个算法的优先级，获取调度方案。

调度方案用于指示至少一个目标任务以及至少一个目标子图之间的对应关系。例如，调度方案可以包括至少一个目标任务的标识以及至少一个目标子图的标识。其中，每个目标任务为多个算法中的任务，目标任务为多个算法的任务中本次调度的任务，目标子图为多个子图中的子图本次调度的子图。

算法的优先级用于指示算法在多个算法中的优先程度，每当要调度多个算法的任务时，可以先调度优先级高的多个算法的任务，后调度优先级低的多个算法的任务，从而控制不同算法的执行结束时间，让优先级高的算法能够尽快执行结束。

在一些可能的实施例中，调度方案的生成过程可以包括下述步骤（B.1）至步骤（B.3）：

步骤（B.1）图计算装置生成第一二分图。

第一二分图用于指示图和算法之间的调度关系，第一二分图可以视为粗粒度的二分图。第一二分图包括第一顶点、第二顶点以及第一边，第一顶点表示图，第二顶点表示算法，第一边连接第一顶点以及第二顶点，第一边表示当前接收到的计算请求中包括用于请求采用第二顶点对应的算法对第一顶点对应的图进行计算的计算请求。

示例性的，参见图11，第一二分图可以为图11中顶层的二分图，第一顶点为顶层的二分图中内侧的一排空心顶点，第二顶点为顶层的二分图中外侧的一排实心顶点，第一边为顶层的二分图中的实现。在图11的顶层的二分图中，存在与同一第一顶点连接的多个第二顶点，代表在图计算装置接收到的一批计算请求中，包括对同一图执行多个算法的请求，在此场景中，可以通过多算法并发执行，来提高计算速度。

步骤（B.2）图计算装置根据第一二分图，生成第二二分图。

第二二分图用于指示子图和任务之间的调度关系，第二二分图可以视为细粒度的二分图。包括第三顶点、第四顶点以及第二边，第三顶点表示任务，第四顶点表示子图，第二边连接第三顶点以及第四顶点。其中，结合子图的模态，第二二分图可以用于指示具有任一模态的子图和任务之间的调度关系，第四顶点表示某个模态的子图。

另外，第二边可以根据第一二分图的边来确定，例如，如果当前接收到的一批计算请求中包括采用算法 i 对图 j 进行计算的计算请求，则第一二分图中算法 i 对应的第一顶点与图 j 对应的第二顶点之间的第一边会得以连接，相应地，第二二分图中算法 i 的每个任务对应的第三顶点与图 j 的每个子图对应的第四顶点之间的第二边会连接。

示例性的，参见图 11，第二二分图可以为图 11 中底层的二分图，第三顶点为底层的二分图中外侧的一排实心顶点，第四顶点为底层的二分图中内侧的一排空心顶点，第二边为底层的二分图中的实线。

在一些可能的实施例中，可以根据图与子图之间的对应关系，将第一顶点映射为第四顶点；根据算法与任务之间的对应关系，将第二顶点映射为第三顶点；将第一边映射为第二边，得到第二二分图。

示例性地，参见图 11，图与子图之间的对应关系以及算法与任务之间的对应关系可以通过顶层的二分图与底层的二分图之间的层次关系来表示。具体地，图与子图之间的一条对应关系可以通过顶层的二分图与底层的二分图之间的一条内侧的虚线表示。例如，图 A 与子图 B 之间的对应关系可以通过顶层的二分图中内侧的顶点 a 与底层的二分图中内侧的顶点 b 之间的虚线表示。算法与任务之间的一条对应关系可以通过顶层的二分图与底层的二分图之间的一条外侧的虚线表示，例如，算法 C 与任务 D 之间的对应关系可以通过顶层的二分图中外侧的顶点 c 与底层的二分图中外侧的顶点 d 之间的虚线表示。

步骤（B.3）图计算装置从第二二分图中选择目标边，将目标边表示的任务与子图之间的对应关系，作为调度方案。

获取调度方案的过程可以建模为从底层的二分图中选择一组边的过程，在此不妨将选择出的边称为目标边。目标边连接的第三顶点为本次要调度的任务，目标边连接的第四顶点为本次要调度的子图。在一些可能的实施例中，考虑到一个任务最多分配到一个子图上，则一组目标边中同一个第三顶点最多连接一个第四顶点；另外，一个子图可以分配到一个或多个任务，则一组目标边中同一个第四顶点可以连接一个或多个第三顶点。

作为示例，选择目标边的过程可以包括下述步骤（B.3.1）至步骤（B.3.2）。

步骤（B.3.1）对于第一二分图中的每条边，图计算装置根据边对应的子图以及任务，获取边的权重。

步骤（B.3.1）可以包括下述方式一至方式二中任一项或多项的结合。

方式一、获取边对应的重要性系数，根据重要性系数，获取边的权重。

例如，如果边连接的第一顶点表示图 G，边连接的第二顶点表示算法 A，可以获取采用算法 A 对图 G 进行计算的重要性系数，对重要性系数进行运算，得到权重，其中权重与重要性系数正相关。

方式二、获取边对应的迭代次数，根据迭代次数，获取边的权重。

例如，如果边连接的第一顶点表示图 G，边连接的第二顶点表示算法 A，可以获取已采用算法 A 对图 G 进行计算的次数，对迭代次数进行运算，得到权重。

需要说明的一点是，方式一和方式二可以择一执行也可以结合执行。例如，如果方式一和方式二结合，可以获取边对应的重要性系数以及迭代次数，对重要性系数以

及迭代次数进行运算，得到边的权重。例如，可以采用上述公式（1）以及公式（2）中的任一项对重要性系数以及迭代次数进行运算。

步骤（B.3.2）图计算装置根据第一二分图中每条边的权重，从第一二分图中选择目标边，目标边的权重符合预设条件。

该预设条件可以包括权重之和在第二二分图中的多个边的权重之和中最大、权重之和在第二二分图中的多个边的权重之和中排在前预设位数、权重之和超过阈值等。

在一些可能的实施例中，图计算装置可以利用约束条件，采用线性规划的方法来选择目标边。具体来说，图计算装置可以根据第一二分图，获取目标函数；采用线性规划算法，根据约束条件对目标函数进行运算，得到目标函数的解，作为目标边。

其中，目标函数用于从第一二分图中选择目标边，例如，目标函数可以指示求解权重之和最大的一组边。示例性地，目标函数可以如下式（4）所示。

$\max(TrB^T X)$；（4）

其中，max 表示求最大值，Tr 表示对矩阵求迹（trace），B 是重要性矩阵 A 的转写，$B = c + \log_2|A|$。c 是常数，通过 B 和 c 可以用于矩阵的预处理（precodnition）技术，能获取数值计算上的好处。A 是重要性系数矩阵，A 中的每一行表示一个算法分解后得到的任务，A 中的每一列表示一个模态的一个子图，A 中的每一个非零元素表示元素对应的任务以及元素对应的子图时能够分配到一起，且该非零元素的数值为对子图执行任务的重要性系数。X 是全排列矩阵（premutation matric），X 的作用是对 B^T 的列进行交换，使得交换后的矩阵的对角线上的元素之和最大，即找到一组最优解。对角线上元素表示的任务与子图之间的对应关系，即为选择出来的调度方案。

约束条件用于保证目标边中的不同边之间不交连（disjoint），例如，约束条件可以用于约束上述公式（4）求解的 X 中每行有且只有一个 1，每列有且只有一个 1。作为示例，约束条件可以包括下式（5）、下式（6）以及下式（7），其中，T 表示转置运算。

$$X\vec{1} = \vec{1}^T \quad ; \quad (5)$$

$$X^T\vec{1}^T = \vec{1} \quad ; \quad (6)$$

$$X^T\vec{1}^T = \vec{1} \quad ; \quad (7)$$

线性规划算法可以是二分图的子图匹配算法，例如可以是拍卖算法（英文：auction algorithm）、匈牙利路径增强算法（英文：Hungarian augmented path algorithm）等。

通过运行上述线性规划算法，可以得到上式（4）中的 X，X 即为第二二分图中的一组不交连的边，X 正好给出一组调度方案：把第二二分图中的某一条边中一端的顶点（表示任务）与另一端的顶点（表示某个模态的某个子图）调度在一起执行。

通过上述实现方式，对于在同一张图上执行的多个算法，其关联关系会在粗粒度的算法与图之间的第一二分图上体现，进而体现到细粒度的第二二分图中，因此在按照上述基于线性规划的方法实施调度的时候，一个模态的一个子图能够被多个任务并

行处理，而不会被反复加载和退出（evict）。

步骤C、图计算装置通过加载了至少一个目标子图的多个计算单元，并行执行至少一个目标任务。

图计算装置可以根据调度方案，确定本次调度的目标任务以及目标子图，确定加载了目标子图的计算单元，通过这些加载了目标子图的计算单元，来并行执行目标任务。在一些可能的实施例中，当计算单元并行执行目标任务结束后，图计算装置可以重复执行上述步骤A至步骤C，即重新获取调度方案，以得到剩余的任务与子图之间的对应关系，从而分配剩余的任务，直至多个算法的任务均已执行结束。

参见图12，其示出了图计算方法的示例性的流程，图12中左侧的分支可以在离线时预先执行，包括下述步骤（1）至步骤（4）：

（1）用户编辑图的模式信息；

（2）根据模式信息导入原始数据，根据原始数据构建图；

（3）将输入的图划分为多个子图；

（4）对每个模态的每个子图进行持久化存储。

图12右侧的分支可以在在线时实时执行，包括下述步骤（5）至步骤（10）：

（5）接收批量的计算请求；

（6）生成第一二分图；

（7）将算法分解为任务，加载每个模态的每个子图；

（8）基于任务以及子图，生成第二二分图；

（9）求解二分图，得到调度方案；

（10）根据调度方案，加载子图，执行任务。

考虑到对图执行的算法大多为迭代算法，参见图12，采用迭代算法对图进行计算的过程还可以包括下述步骤一至步骤三：

步骤一、计算单元根据多个子图，执行迭代算法的第一次执行过程的任务，得到第一次执行过程的计算结果。

步骤二、计算单元判断算法是否收敛。

作为示例，计算单元可以根据第一次执行过程的计算结果以及迭代次数中的至少一项，判断迭代算法尚未收敛。例如，计算单元可以判断迭代次数是否达到预设次数，如果迭代次数未达到预设次数，则计算单元确定迭代算法尚未收敛，如果迭代次数已达到预设次数，则计算单元确定迭代算法收敛。又如，计算单元可以判断第一次执行过程的计算结果是否符合预设条件，如果第一次执行过程的计算结果未符合预设条件，则计算单元确定迭代算法尚未收敛。

步骤三、如果迭代算法未收敛，计算单元根据多个子图以及第一次执行过程的计算结果，执行迭代算法的第二次执行过程的任务，得到第二次执行过程的计算结果。

之后，计算单元可以重新判断迭代算法是否收敛，如果迭代算法尚未收敛，则根据多个子图以及第二次执行过程的计算结果，执行迭代算法的第三次执行过程的任务，依次类推，直至迭代算法收敛，可以输出本次执行过程的计算结果，例如对本次执行过程的计算结果进行可视化，呈现本次执行过程的计算结果。

在一些可能的实施例中，参见图12，对于多个算法中尚未收敛的算法来说，可以

获取尚未收敛的算法的任务，作为剩余任务，从而动态生成新的任务。如果接收到下一次计算请求，可以获取下一次计算请求中请求的算法的任务，将这些新的任务与剩余任务一起并发执行，直至没有新的计算请求需要处理。

在一些可能的实施例中，在多个计算单元并行执行多个算法的过程中，图计算装置可以对多个计算单元进行扩容，以便通过更多的计算单元来执行算法，具体来说，扩容的过程详见下述步骤一至步骤八。

步骤一、图计算装置向终端发送扩容请求。

扩容请求用于请求对多个计算单元进行扩容。在一些可能的实施例中，扩容请求可以包括要新增的计算单元的数量以及扩容费用中的至少一项，扩容费用为用户要为新增的计算单元支付的费用。例如，如果单位时长内租用一个计算单元需要支付 n，而当前要新增 m 个计算单元，则扩容请求可以包括要新增的计算单元的数量 m 以及扩容费用（m*n），其中 m 为正整数，n 为正数。

在一些可能的实施例中，图计算装置可以在多个计算单元的计算能力不满足需求时，生成扩容请求，向终端发送扩容请求。例如，终端向图计算装置发送的计算请求可以包括期望时长，该期望时长为用户期望多个计算单元完成对图计算的时长，图计算装置可以判断多个计算单元是否能够在期望时长内完成对图进行计算，如果多个计算单元不能够在期望时长内完成对图进行计算，图计算装置可以生成扩容请求。又如，图计算装置可以监控每个计算单元的负载情况，实时判断每个计算装置的负载情况是否超过阈值，当一个或多个计算单元的负载情况超过阈值时，图计算装置可以生成扩容请求。又如，多个计算单元可以监控自身的负载情况是否超过阈值，如果自身的负载情况超过阈值，则多个计算单元向图计算装置发送通知消息，图计算装置接收到一个或多个计算单元的通知消息后，图计算装置可以生成扩容请求。又如，图计算装置可以根据图的数据量以及每个计算单元的规格，判断多个计算单元能够计算的总数据量是否大于图的数据量，如果多个计算单元能够计算的总数据量小于或等于图的数据量，图计算装置可以生成扩容请求。

步骤二、终端接收图计算装置的扩容请求，显示提示信息。

提示信息用于提示用户是否对多个计算单元进行扩容。在一些可能的实施例中，该提示信息还可以包括要新增的计算单元的数量以及扩容费用中的至少一项。

步骤三、终端接收确认指令，生成扩容指令，向图计算装置发送扩容指令，扩容指令用于指示对多个计算单元进行扩容。

确认指令用于指示确认对多个计算单元进行扩容，确认指令可以通过用户在终端上执行的确认操作触发。作为示例，终端显示的提示信息可以包括确认选项，如果用户对确认选项触发点击操作，则终端接收到确认指令。

步骤四、图计算装置接收终端的扩容指令。

在一些可能的实施例中，终端接收确认指令后，可以向金融服务器发送支付请求，支付请求包括扩容费用、用户账户的标识以及图计算装置对应的商家账户的标识，金融服务器接收到支付请求后，会从支付请求中获取用户账户的标识以及商家账户的标识，从用户账户中扣除扩容费用，向商家账户中增加扩容费用；扣款成功后，金融服务器可以向终端发送支付成功消息，终端接收到支付成功消息后，向图计算装置发送

扩容指令，图计算装置接收扩容指令，执行下述步骤五；或者，金融服务器可以向图计算装置发送支付成功消息，图计算装置接收到支付成功消息且接收到扩容指令后，执行下述步骤五。

需要说明的一点是，上述仅是以图计算装置和金融服务器为两个分离的装置为例进行描述，在另一些可能的实施例中，图计算装置和金融服务器可以集成在一起，该集成的装置同时具有图计算的功能以及在线交易的功能。在这种实现方式中，该集成的装置可以预存有商家账户的标识，终端接收确认指令后，可以向该集成的装置发送支付请求，该集成的装置接收到支付请求后，会从用户账户中扣除扩容费用，向商家账户中增加扩容费用；扣款成功后，该集成的装置可以执行下述步骤五。

步骤五、图计算装置创建至少一个计算单元。

步骤六、图计算装置对图的至少一个子图进行复制，得到至少一个子图的实例。

其中，子图的实例是子图的拷贝，子图的实例可以和子图相同。

步骤七、图计算装置将至少一个子图的实例加载到创建的至少一个计算单元。

步骤七与上述步骤 602 同理，在此不做赘述。

步骤八、图计算装置通过该创建的至少一个计算单元，并行执行多个算法。

步骤八与上述步骤 603 同理，在此不做赘述。

在一些可能的实施例中，图计算装置可以统计图中每个子图被多个算法请求的次数，判断每个子图的请求次数是否达到阈值，对图中请求次数达到阈值的子图进行复制，得到请求次数达到阈值的子图的实例，将这些子图的实例加载到扩容出的目标计算单元，由目标计算单元在这些子图的实例上并行执行多个算法。其中，请求次数为多个算法请求子图的次数之和。

通过这种实现方式，考虑到单个图能够支持的访问数量具有物理上限的，如果图的访问数量超过物理上限，很可能在图上无法支持并行执行多个算法，也就对图算法的执行过程造成瓶颈。例如，如果某个子图是热点子图，比如说，该子图是包含了名人信息的子图，同一时刻很可能有大量用户访问该子图以查询名人信息，则该子图可能无法支持被多个任务同时调度，也就无法并行执行多个算法，导致对图计算的整体速度造成限制。而上述实现方式中，图计算装置能够感知每个子图被请求的数量，如果某个子图的请求次数超过阈值，表明该子图的需求量很大，很可能是热点子图，那么通过触发该子图被复制为多份，分别部署在不同的计算单元上，由多个计算单元对该子图进行处理，能够提高这个子图的处理效率。也即是，通过让并发的计算请求能够分流至不同的实例上来进行计算，从而线性扩展了并发性。

在一些可能的实施例中，在并行执行多个算法的过程中，可以对多个子图进行写保护。作为示例，写保护的方式包括而不限于下述方式一至方式三中的任一项或多项的结合：

方式一（加锁）、当任一算法的任一任务写入任一子图时，计算单元可以对子图加锁，在执行该任务的期间，除了该任务以外的其他任务可以无法写入该子图。当算法的任务对子图写入结束后，计算单元可以释放锁。通过执行方式一，一方面，可以保证对子图执行的写入步骤得以串行执行，另一方面，这种方式在实现和管理起来较为简单。

方式二（多版本并发控制（英文全称：multi-version concurrency control ，英文简称：MVCC））。计算单元可以缓存多个子图中每个子图的全态；例如，可以存储子图的当前态、历史态以及过渡态。通过执行方式二，有利于同时进行写的操作。

方式三、对于多个子图中的任一子图，可以从子图的多个实例中选择目标实例，该目标实例可以视为多个实例中的首选实例。当接收到对子图的任一实例的写指令时，可以根据写指令，先向目标实例写入数据，当写入结束时，对目标实例与多个实例中目标实例以外的其他实例进行数据同步，从而实现子图的所有实例的数据一致性。可选地，当接收到对任一实例的写指令时，可以终止分发任务，待所有实例完成写操作后方可恢复，从而实现更严格的一致。

通过上述实现方式，可以降低保证多个任务共享同一子图时，发生数据冲突的概率，保证分布式计算单元并行执行多任务时系统的稳健性。

在一些可能的实施例中，图计算装置可以为每个算法的每个任务创建缓存空间，当得到该任务的任一中间结果时，图计算装置可以将中间结果缓存在该缓存空间中，当该任务执行结束后，可以释放该缓存空间。

本实施例提供的方法，提供了一种能够支持多算法并发执行图计算的方法，通过将图的多个子图加载到多个计算单元，通过多个计算单元，并行执行多个算法，可以在多个算法之间共享同一份图，从而在同一份图上并行执行多个算法，节省了执行某个算法时需要等待其他算法执行结束所造成的时延，从而提高了多算法进行图计算的整体效率，缩短了多算法进行图计算的整体时间。

本申请还提供了一种图计算装置1300。如图13所示，图计算装置1300包括接收单元1301，加载单元1302以及多个计算单元1303。

接收单元1301用于执行步骤601。加载单元1302用于执行步骤602。多个计算单元1303用于执行步骤603。

在一种可能的实现中，多个计算单元1303，包括：获取模块，用于执行步骤603中的步骤一，例如可以用于执行步骤603中的步骤一中的方式一至方式九中的任一项或多项的组合，以将算法划分为多个任务；执行模块，用于执行步骤603中的步骤二。

在一种可能的实现中，加载单元1302，用于将至少一种模态的多个子图加载到多个计算单元；执行模块，用于根据目标模态的子图执行任务；

在一种可能的实现中，执行模块，用于执行步骤603中的步骤A至步骤C。

在一种可能的实现中，多个计算单元1303还用于执行算法中的加载步骤以外的部分。

在一种可能的实现中，多个计算单元1303还用于执行算法中的释放步骤以外的部分。

在一种可能的实现中，该装置还包括：划分单元，用于对图进行划分；存储单元，用于将多个子图存入图存储装置；加载单元1302，用于将图的多个子图从图存储装置加载到多个计算单元。

在一种可能的实现中，划分单元，用于根据多个计算单元1303的数量，对图进行划分。

在一种可能的实现中，装置还包括发送单元，用于发送扩容请求；接收单元，还

用于接收扩容指令；创建单元，用于创建至少一个计算单元；复制单元，用于对图的至少一个子图进行复制；加载单元1302，还用于将至少一个子图的实例加载到创建的至少一个计算单元；创建的至少一个计算单元，用于并行执行多个算法。

图计算装置1300可以作为云计算服务向用户提供，例如作为图引擎服务向用户提供。例如图1所示，图计算装置1300（或其部分）部署在云环境上，用户通过终端触发计算请求后，启动图计算装置1300对图执行多个算法，计算结果可以被提供给用户。

需要说明的一点是，图13实施例提供的图计算装置在进行图计算时，仅以上述各单元以及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单元以及模块完成，即将图计算装置的内部结构划分成不同的单元以及模块，以完成以上描述的全部或者部分功能。另外，上述实施例提供的图计算装置与图计算方法实施例属于同一构思，其具体实现过程详见方法实施例，这里不再赘述。

本申请还提供了一种图计算装置1400。如图14所示，图计算装置1400包括多个计算单元1410，每个计算单元1410包括处理器1411、存储器1412、收发器1413以及总线1414，处理器1411、存储器1412、收发器1413之间通过总线1414通信。

其中，处理器可以为中央处理器(英文：central processing unit，缩写：CPU)。存储器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)。存储器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，HDD或SSD。存储器中存储有可执行代码，处理器执行该可执行代码以执行前述图计算方法。存储器中还可以包括操作系统等其他运行进程所需的软件模块。操作系统可以为LINUX™,UNIX™,WINDOWS™等。

图计算装置1400的每个计算单元1410的存储器1412中存储了图计算装置1300的各个单元对应的代码，处理器1411执行这些代码实现了图计算装置1300的各个单元的功能，即执行了图6实施例提供的图计算方法。

图计算装置1400可以部署在云环境中，图计算装置1400中的多个计算单元1410可以组成分布式系统，不同计算单元1410可以通过有线或无线网络进行通信。

本申请还提供了一种图计算装置1500。如图15所示，图计算装置1500包括多个计算单元1510、收发器1511以及总线1512；

每个计算单元1510包括处理器1521以及存储器1522；其中，处理器1521可以与处理器1411同理，存储器1522可以与存储器1412同理，在此不做赘述。

图计算装置1500中的多个计算单元1510可以部署在单机中，不同计算单元1510可以通过总线1512进行通信。

本申请还提出了一种图计算系统，如图16所示，该图计算系统在包括图14实施例所示的图计算装置1400的基础上，还包括图存储装置1600，图存储装置1600用于存储图的多个子图。在一些可能的实施例中，图存储装置1600可以作为云存储服务向用户提供，用户可以在该云存储服务中申请一定容量的存储空间，在该存储空间中存储图的多个子图。图计算装置1400运行时，通过通信网络从图存储装置1600中加载所需的多个子图。

作为一种可能的产品形态，本申请实施例的图计算装置可以由通用处理器来实现。

通用处理器包括处理电路和与处理电路内部连接通信的输入接口，输入接口用于接收至少一个计算请求；处理电路用于执行上述图计算方法。可选地，该通用处理器还可以包括存储介质，存储介质用于存储处理电路执行的指令。

作为一种可能的产品形态，本申请实施例的图计算装置，还可以使用下述来实现：一个或多个现场可编程门阵列（英文全称：field—programmable gate array，英文简称：FPGA）、可编程逻辑器件(英文全称：programmable logic device，英文简称：PLD)、控制器、状态机、门逻辑、分立硬件部件、任何其它适合的电路、或者能够执行本申请通篇所描述的各种功能的电路的任意组合。

在一些可能的实施例中，本申请还提供了一种计算机程序产品，该计算机程序产品被图计算装置执行时，该图计算装置执行上述图计算方法。该计算机程序产品可以为一个软件安装包，在需要使用前述图计算方法的情况下，可以下载该计算机程序产品并在图计算装置上执行该计算机程序产品。

上述各个附图对应的流程的描述各有侧重，某个流程中没有详述的部分，可以参见其他流程的相关描述。

本领域普通技术人员可以意识到，结合本文中所公开的实施例中描述的各方法步骤和单元，能够以电子硬件、计算机软件或者二者的结合来实现，为了清楚地说明硬件和软件的可互换性，在上述说明中已经按照功能一般性地描述了各实施例的步骤及组成。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。本领域普通技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

所属领域的技术人员可以清楚地了解到，为了描述的方便和简洁，上述描述的系统、装置和单元的具体工作过程，可以参见前述方法实施例中的对应过程，在此不再赘述。

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，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。另外，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，也可以是电的，机械的或其它的形式连接。

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申请实施例方案的目的。

另外，在本申请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，也可以是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。

集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本申请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）执行本申请各个实施例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、只读存储器（read-only memory， ROM）、随机存取存储器（random access memory，RAM）、磁碟或者光盘等各种可以存储程序代码的介质。

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

应理解，上述各种产品形态的图计算装置，分别具有上述图6方法实施例中图计算方法的任意功能，此处不再赘述。

以上该，仅为本申请的具体实施方式，但本申请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申请揭露的技术范围内，可轻易想到各种等效的修改或替换，这些修改或替换都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应以权利要求的保护范围为准。

权 利 要 求 书

1、一种图计算方法，其特征在于，所述方法包括：

接收至少一个计算请求，所述至少一个计算请求用于请求采用多个算法对图进行计算；

将所述图的多个子图加载到多个计算单元；

通过所述多个计算单元，并行执行所述多个算法。

2、根据权利要求 1 所述的方法，其特征在于，所述通过所述多个计算单元，并行执行所述多个算法，包括：

获取每个算法中的至少一个任务；

通过所述多个计算单元，并行执行所述多个算法中的任务。

3、根据权利要求 2 所述的方法，其特征在于，所述获取每个算法中的至少一个任务，包括下述至少一项：

根据所述算法中每个步骤对应的函数名称，将所述算法中同一函数名称对应的至少一个步骤划分为一个任务；

根据所述算法中每个步骤的执行主体，将所述算法中执行主体相同的步骤划分为一个任务；

根据所述算法中每个步骤对图中顶点或边的访问顺序，将所述算法中访问顺序相同的步骤划分为一个任务；

根据所述算法中每个步骤在图中访问的顶点或边，将所述算法中访问的顶点或边相同的步骤划分为一个任务；

根据所述算法中每个步骤执行的动作，将所述算法中动作相同的步骤划分为一个任务；

根据所述算法中每个步骤所属的迭代过程，将所述算法中属于同一次迭代过程的步骤划分为一个任务；

根据所述算法中每个步骤所属的判断分支，将所述算法中属于同一判断分支的步骤划分为一个任务。

4、根据权利要求 2 所述的方法，其特征在于，

所述将所述图的多个子图加载到多个计算单元，包括：

将至少一种模态的所述多个子图加载到所述多个计算单元；

所述通过所述多个计算单元，并行执行所述多个算法中的任务，包括：

对于所述多个算法的任一任务，通过所述多个计算单元，根据目标模态的子图执行所述任务，所述目标模态为所述至少一种模态中与所述任务匹配的模态。

5、根据权利要求 4 所述的方法，其特征在于，所述任务包括从所述图中第一顶点搜索至所述图中第二顶点的步骤，所述目标模态的子图中所述第二顶点排在所述第一顶点之前。

6、根据权利要求 2 所述的方法，其特征在于，所述通过所述多个计算单元，并行执行所述多个算法中的任务，包括：

根据所述多个算法的迭代次数，获取所述多个算法的优先级；

根据每个算法的优先级，获取调度方案，所述调度方案用于指示至少一个目标任

务以及至少一个目标子图之间的对应关系，所述目标任务为所述多个算法的任务中本次调度的任务，所述目标子图为所述多个子图中的子图本次调度的子图；

通过加载了所述至少一个目标子图的多个计算单元，并行执行所述至少一个目标任务。

7、根据权利要求 1 至 6 中任意一项所述的方法，其特征在于，所述通过所述多个计算单元，并行执行所述多个算法，包括下述至少一项：

对于所述多个算法中的任一算法，执行所述算法中的加载步骤之外的部分，所述加载步骤为将所述图加载到计算单元的内存的步骤；

对于所述多个算法中的任一算法，执行所述算法中的释放步骤之外的部分，所述释放步骤为将所述图从计算单元的内存中释放的步骤。

8、根据权利要求 1 所述的方法，其特征在于，所述方法还包括：

发送扩容请求，所述扩容请求用于请求对所述多个计算单元进行扩容；

接收扩容指令，所述扩容指令用于指示对所述多个计算单元进行扩容；

创建至少一个计算单元；

对所述图的至少一个子图进行复制，得到至少一个子图的实例；

将所述至少一个子图的实例加载到所述创建的至少一个计算单元；

通过所述创建的至少一个计算单元，并行执行所述多个算法。

9、根据权利要求 8 所述的方法，其特征在于，所述对所述图的至少一个子图进行复制，包括：

统计所述图中每个子图被所述多个算法请求的次数；

对所述图中请求次数达到阈值的子图进行复制。

10、一种图计算装置，其特征在于，包括：

接收单元，用于接收至少一个计算请求，所述至少一个计算请求用于请求采用多个算法对图进行计算；

加载单元，用于将所述图的多个子图加载到多个计算单元；

所述多个计算单元，用于通过所述多个计算单元，并行执行所述多个算法。

11、根据权利要求 10 所述的装置，其特征在于，所述多个计算单元，包括：

获取模块，用于获取每个算法中的至少一个任务；

执行模块，用于通过所述多个计算单元，并行执行所述多个算法中的任务。

12、根据权利要求 11 所述的装置，其特征在于，所述获取模块，用于执行下述至少一项：

根据所述算法中每个步骤对应的函数名称，将所述算法中同一函数名称对应的至少一个步骤划分为一个任务；

根据所述算法中每个步骤的执行主体，将所述算法中执行主体相同的步骤划分为一个任务；

根据所述算法中每个步骤对图中顶点或边的访问顺序，将所述算法中访问顺序相同的步骤划分为一个任务；

根据所述算法中每个步骤在图中访问的顶点或边，将所述算法中访问的顶点或边相同的步骤划分为一个任务；

根据所述算法中每个步骤执行的动作，将所述算法中动作相同的步骤划分为一个任务；

根据所述算法中每个步骤所属的迭代过程，将所述算法中属于同一次迭代过程的步骤划分为一个任务；

根据所述算法中每个步骤所属的判断分支，将所述算法中属于同一判断分支的步骤划分为一个任务。

13、根据权利要求 11 所述的装置，其特征在于，

所述加载单元，用于将至少一种模态的所述多个子图加载到所述多个计算单元；

所述执行模块，用于对于所述多个算法的任一任务，通过所述多个计算单元，根据目标模态的子图执行所述任务，所述目标模态为所述至少一种模态中与所述任务匹配的模态。

14、根据权利要求 13 所述的装置，其特征在于，所述任务包括从所述图中第一顶点搜索至所述图中第二顶点的步骤，所述目标模态的子图中所述第二顶点排在所述第一顶点之前。

15、根据权利要求 11 所述的装置，其特征在于，所述执行模块，用于根据所述多个算法的迭代次数，获取所述多个算法的优先级；根据每个算法的优先级，获取调度方案，所述调度方案用于指示至少一个目标任务以及至少一个目标子图之间的对应关系，所述目标任务为所述多个算法的任务中本次调度的任务，所述目标子图为所述多个子图中的子图本次调度的子图；通过加载了所述至少一个目标子图的多个计算单元，并行执行所述至少一个目标任务。

16、根据权利要求 10 至 15 中任意一项所述的装置，其特征在于，所述多个计算单元，用于执行下述至少一项：对于所述多个算法中的任一算法，执行所述算法中的加载步骤之外的部分，所述加载步骤为将所述图加载到计算单元的内存的步骤；对于所述多个算法中的任一算法，执行所述算法中的释放步骤之外的部分，所述释放步骤为将所述图从计算单元的内存中释放的步骤。

17、根据权利要求 10 所述的装置，其特征在于，所述装置还包括：

发送单元，用于发送扩容请求，所述扩容请求用于请求对所述多个计算单元进行扩容；

所述接收单元，还用于接收扩容指令，所述扩容指令用于指示对所述多个计算单元进行扩容；

创建单元，用于创建至少一个计算单元；

复制单元，用于对所述图的至少一个子图进行复制，得到至少一个子图的实例；

所述加载单元，还用于将所述至少一个子图的实例加载到所述创建的至少一个计算单元；

所述创建的至少一个计算单元，用于并行执行所述多个算法。

18、根据权利要求 17 所述的装置，其特征在于，所述复制单元，用于统计所述图中每个子图被所述多个算法请求的次数；对所述图中请求次数达到阈值的子图进行复制。

19、一种图计算装置，其特征在于，包括多个计算单元，每个计算单元包括处理

器和存储器，所述存储器中存储有至少一条指令，所述指令由所处理器加载并执行以实现如权利要求 1 至 9 中任一所述的方法。

20、一种图计算系统，其特征在于，包括图计算装置和图存储装置；

所述图计算装置包括多个计算单元；

所述图存储装置用于存储图的多个子图；

所述图计算装置用于将所述多个子图从所述图存储装置加载至所述多个计算单元，以执行权利要求 1 至 9 中任一所述的方法。

21、一种非瞬态的可读存储介质，其特征在于，所述非瞬态的可读存储介质被图计算装置执行时，所述图计算装置执行权利要求 1 至 9 中任一所述的方法。

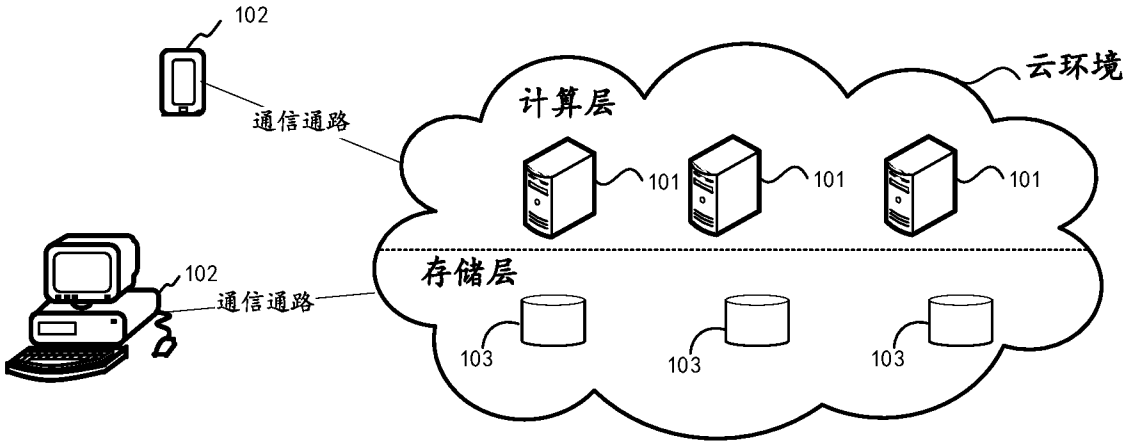

图 1

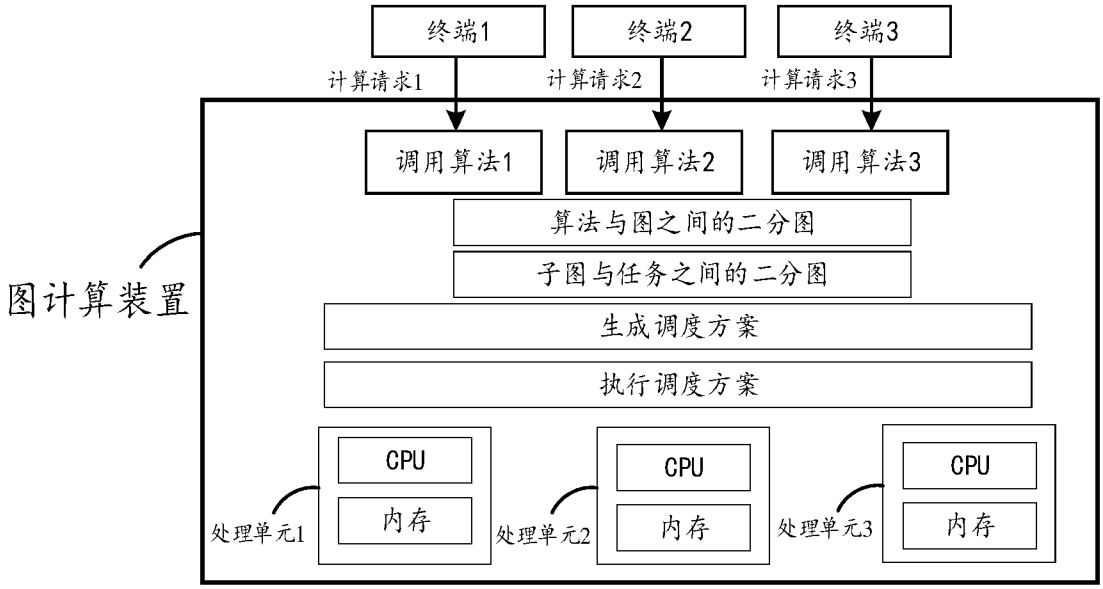
终端1
终端2
终端3
计算请求1
计算请求2
计算请求3
调用算法1
调用算法2
调用算法3
算法与图之间的二分图
子图与任务之间的二分图
生成调度方案
执行调度方案
图计算装置
CPU
内存
处理单元1
处理单元2
处理单元3

图 2

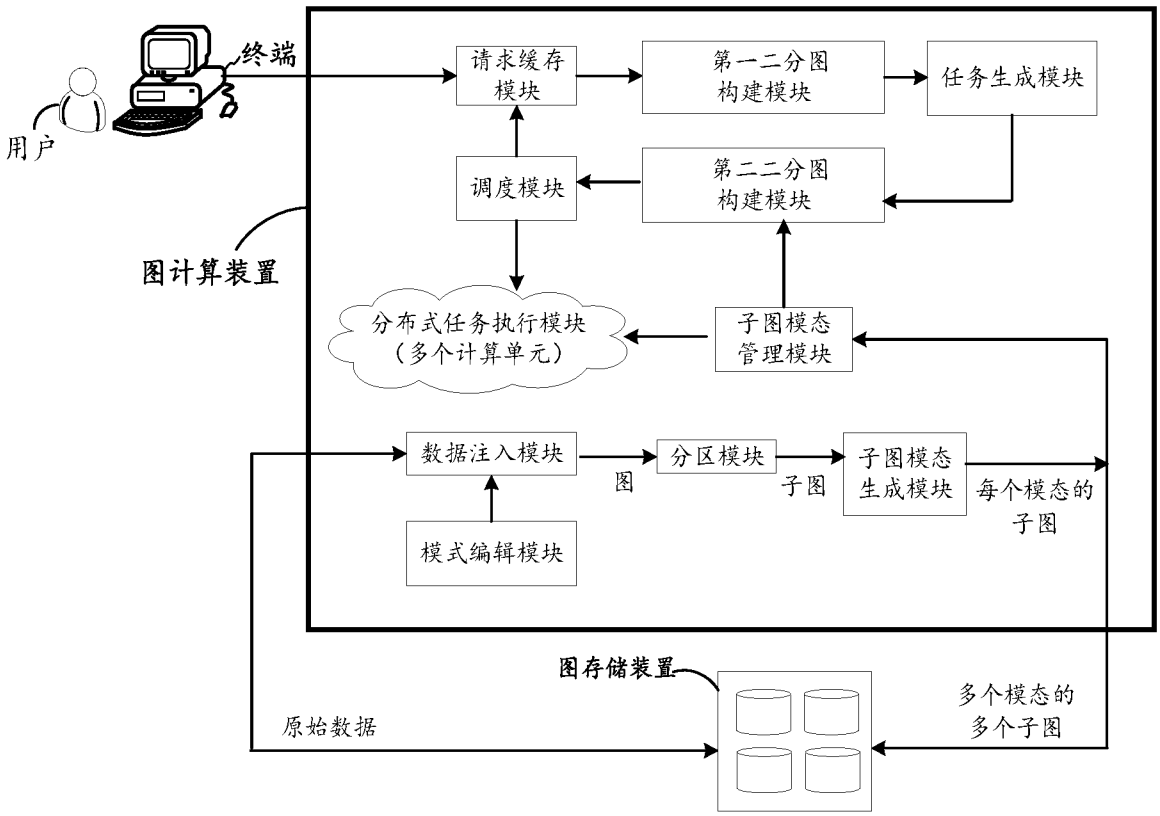
终端
用户
图计算装置
请求缓存模块
第一二分图构建模块
任务生成模块
调度模块
第二二分图构建模块
分布式任务执行模块（多个计算单元）
子图模态管理模块
数据注入模块
图
分区模块
子图
子图模态生成模块
每个模态的子图
模式编辑模块
图存储装置
原始数据
多个模态的多个子图

图 3

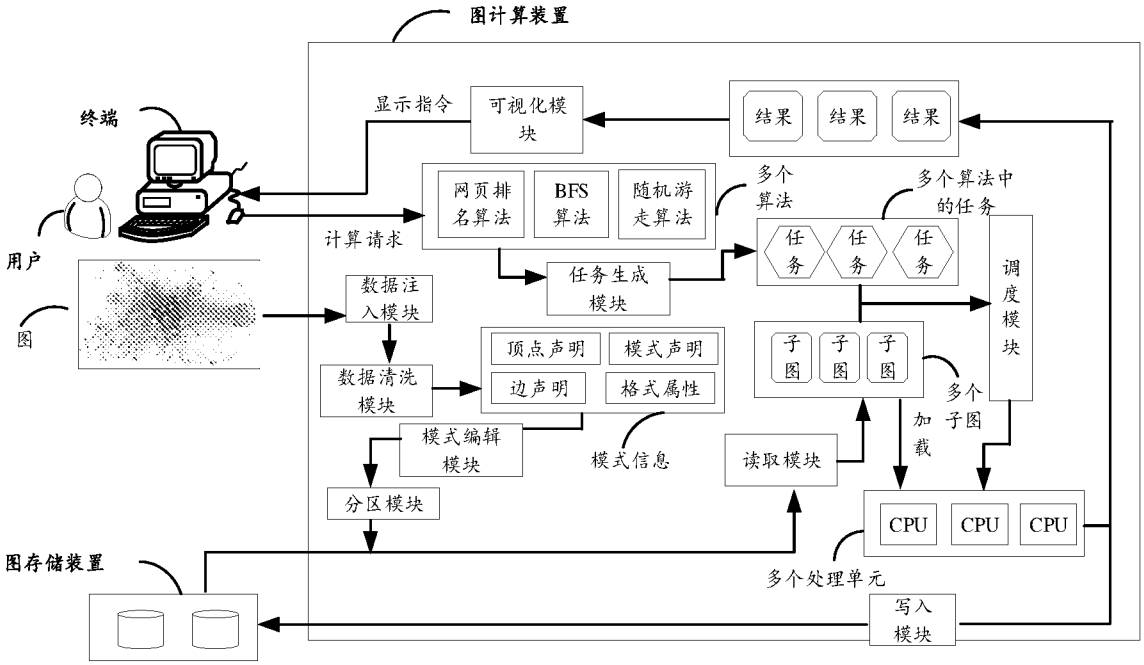
图计算装置
终端
显示指令
可视化模块
结果
结果
结果
网页排名算法
BFS算法
随机游走算法
多个算法
多个算法中的任务
用户
计算请求
任务
任务
任务
调度模块
图
数据注入模块
任务生成模块
顶点声明
模式声明
边声明
格式属性
子图
子图
子图
数据清洗模块
多个子图加载
模式编辑模块
模式信息
读取模块
分区模块
CPU
CPU
CPU
图存储装置
多个处理单元
写入模块

图 4

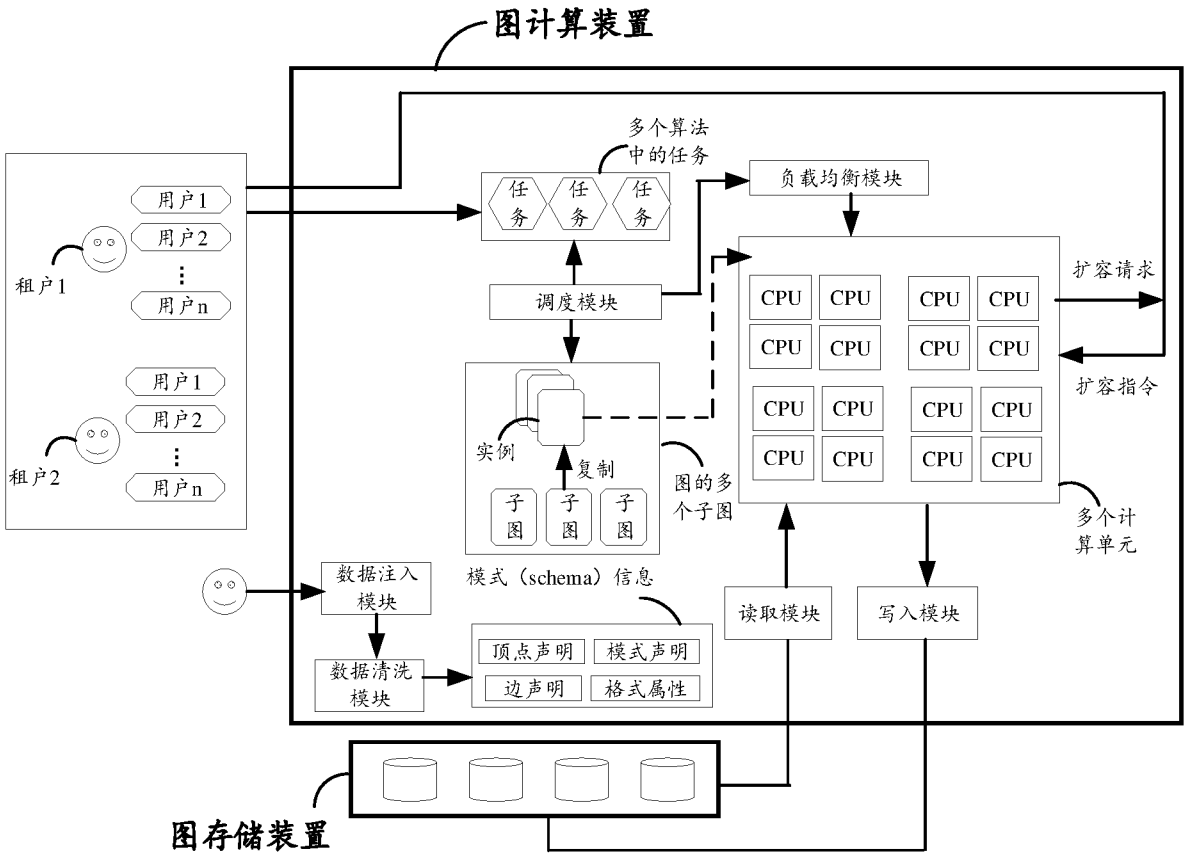
图计算装置
多个算法中的任务
任务
任务
任务
负载均衡模块
用户1
用户2
用户n
租户1
调度模块
扩容请求
CPU
扩容指令
实例
复制
子图
子图
子图
图的多个子图
多个计算单元
用户1
用户2
用户n
租户2
数据注入模块
模式（schema）信息
读取模块
写入模块
顶点声明
模式声明
数据清洗模块
边声明
格式属性
图存储装置

图 5

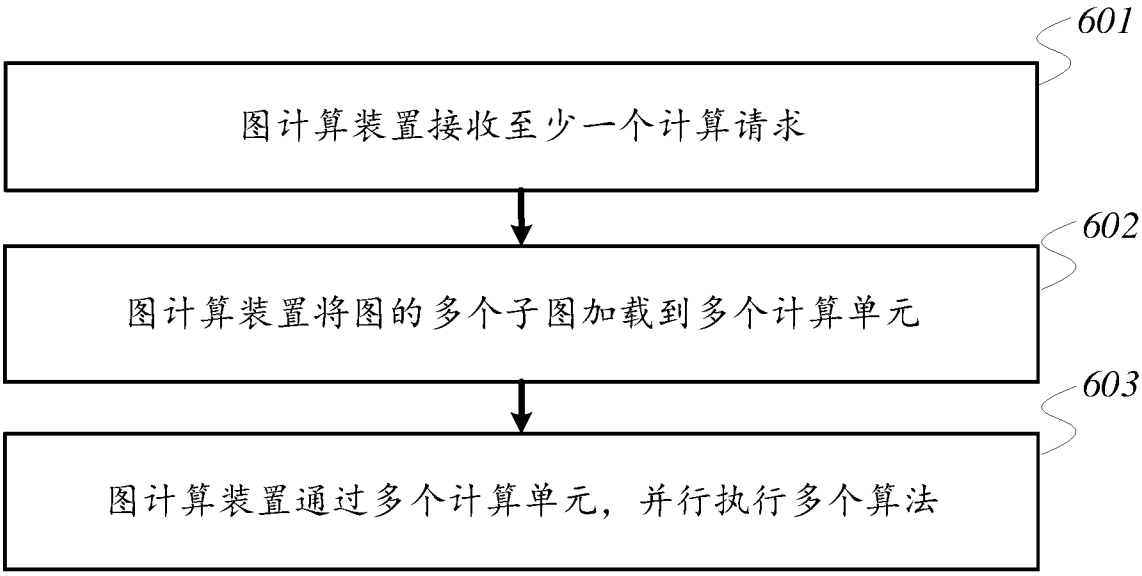
601
图计算装置接收至少一个计算请求
602
图计算装置将图的多个子图加载到多个计算单元
603
图计算装置通过多个计算单元，并行执行多个算法

图 6

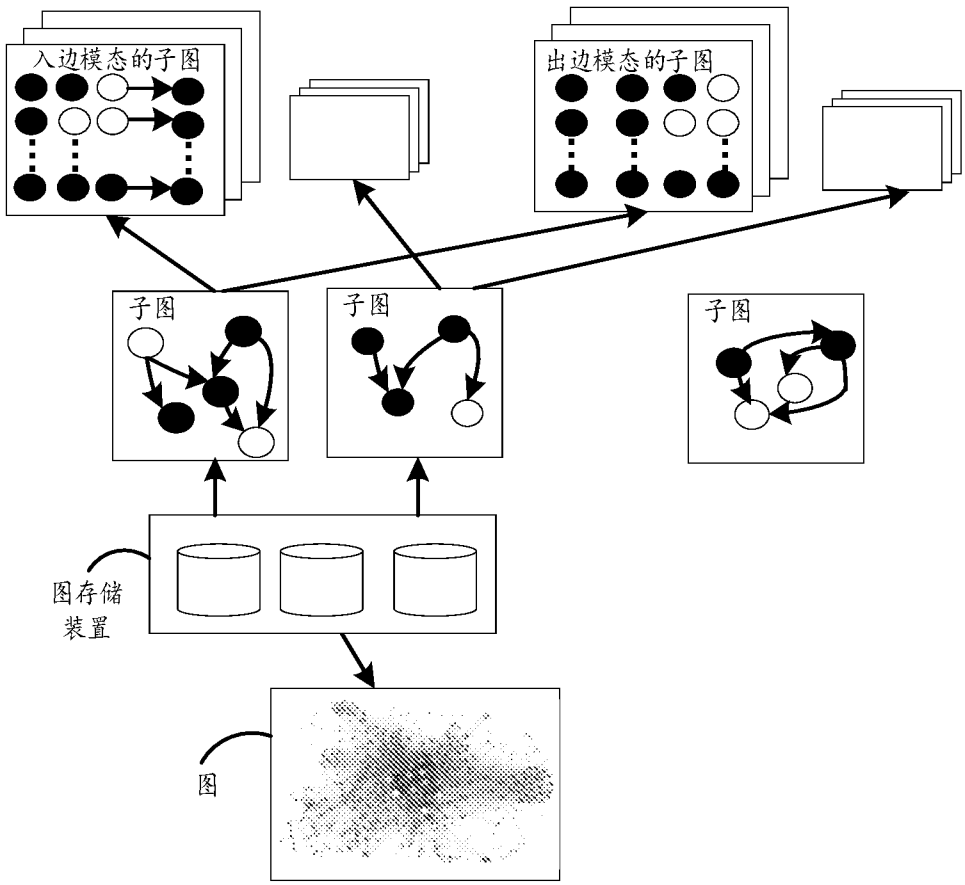

图 7

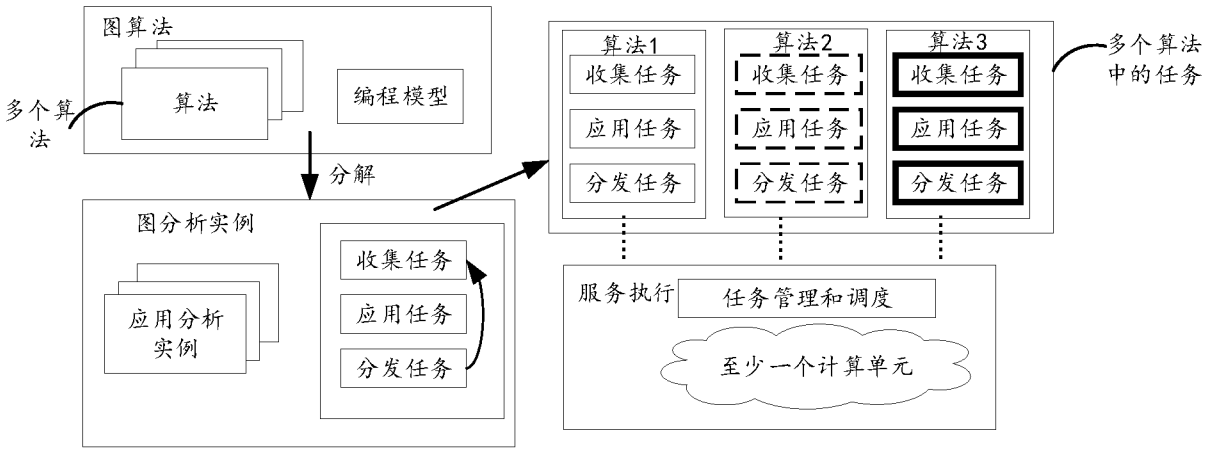
图算法
多个算法
算法
编程模型
分解
图分析实例
应用分析实例
收集任务
应用任务
分发任务
算法1
收集任务
应用任务
分发任务
算法2
收集任务
应用任务
分发任务
算法3
收集任务
应用任务
分发任务
多个算法中的任务
服务执行
任务管理和调度
至少一个计算单元

图 8

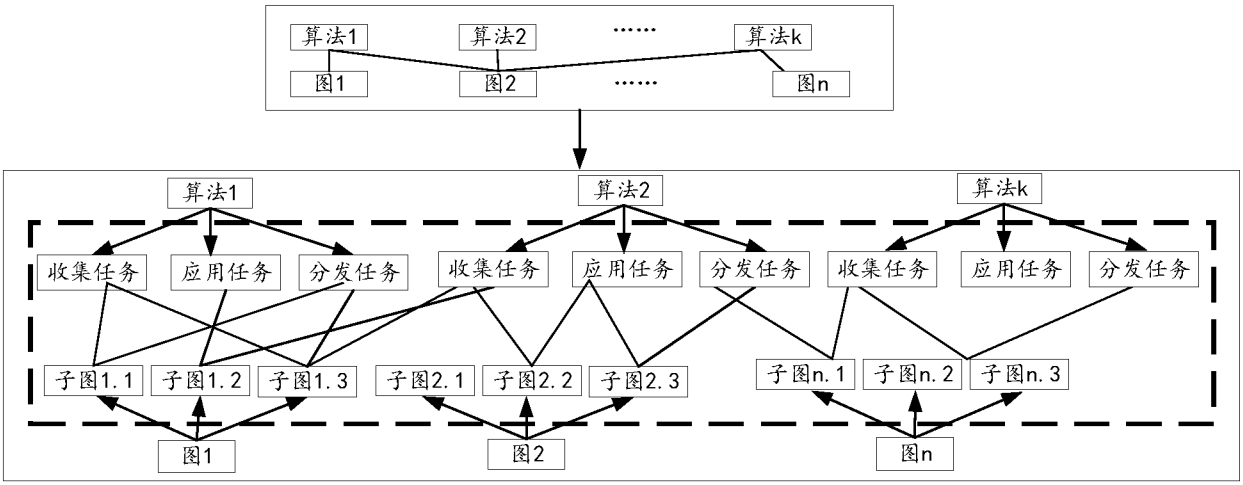
算法1
算法2
……
算法k
图1
图2
……
图n
算法1
算法2
算法k
收集任务
应用任务
分发任务
收集任务
应用任务
分发任务
收集任务
应用任务
分发任务
子图1.1
子图1.2
子图1.3
子图2.1
子图2.2
子图2.3
子图n.1
子图n.2
子图n.3
图1
图2
图n

图 9

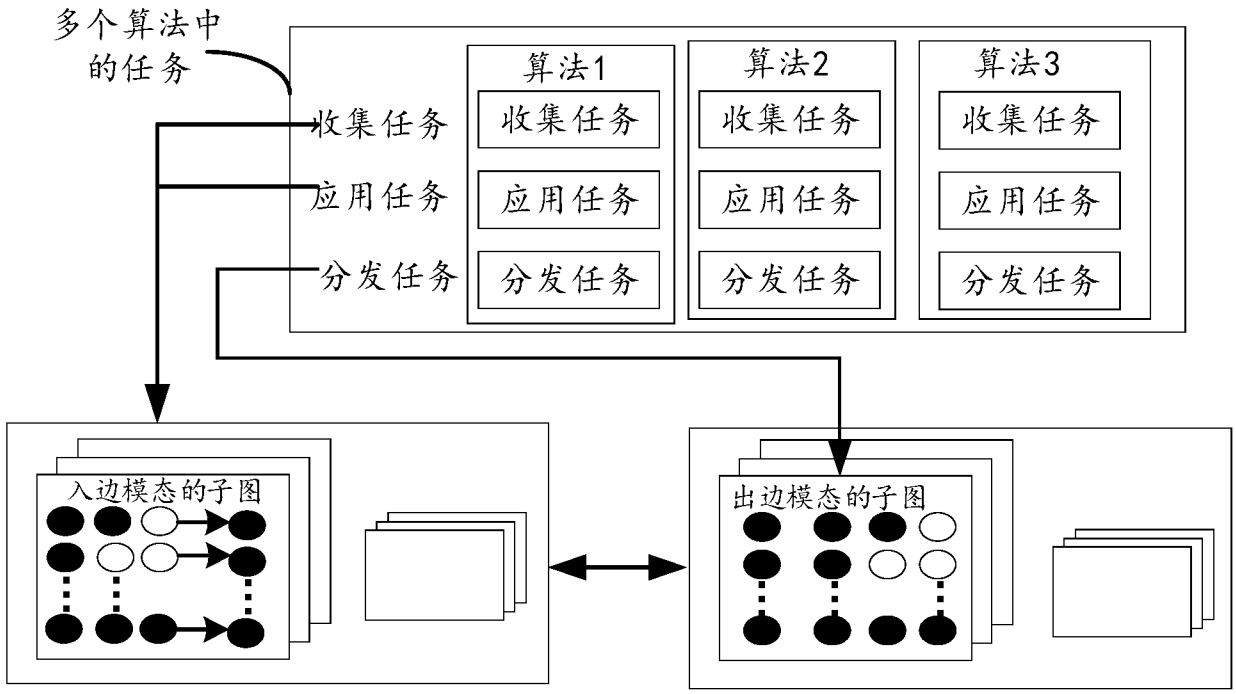
多个算法中
的任务
收集任务
应用任务
分发任务
算法1
收集任务
应用任务
分发任务
算法2
收集任务
应用任务
分发任务
算法3
收集任务
应用任务
分发任务
入边模态的子图
出边模态的子图

图 10

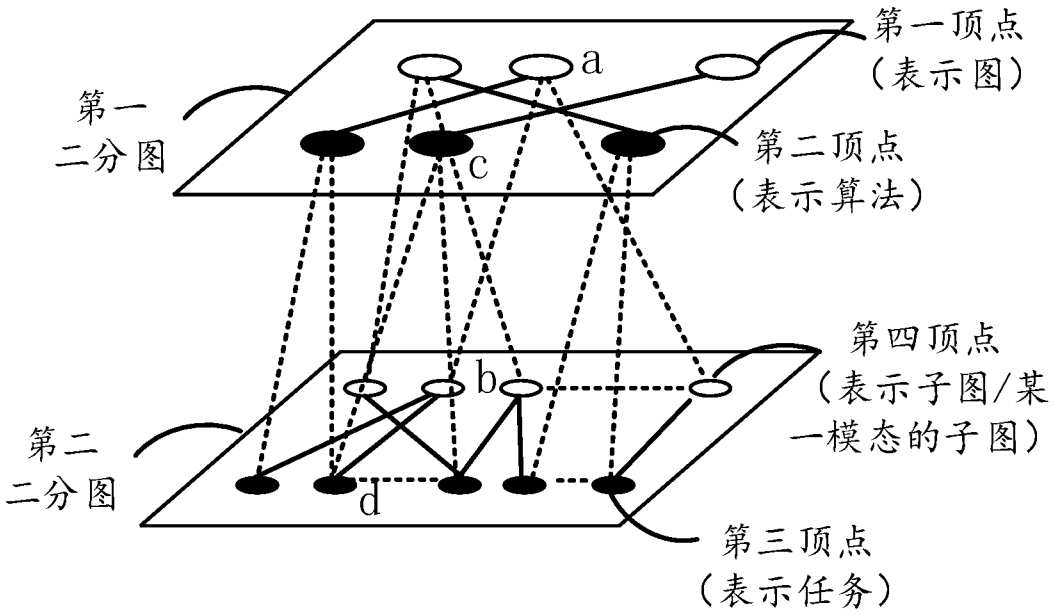
第一顶点
（表示图）
a
第一
二分图
第二顶点
（表示算法）
c
第四顶点
（表示子图/某
一模态的子图）
b
第二
二分图
d
第三顶点
（表示任务）

图 11

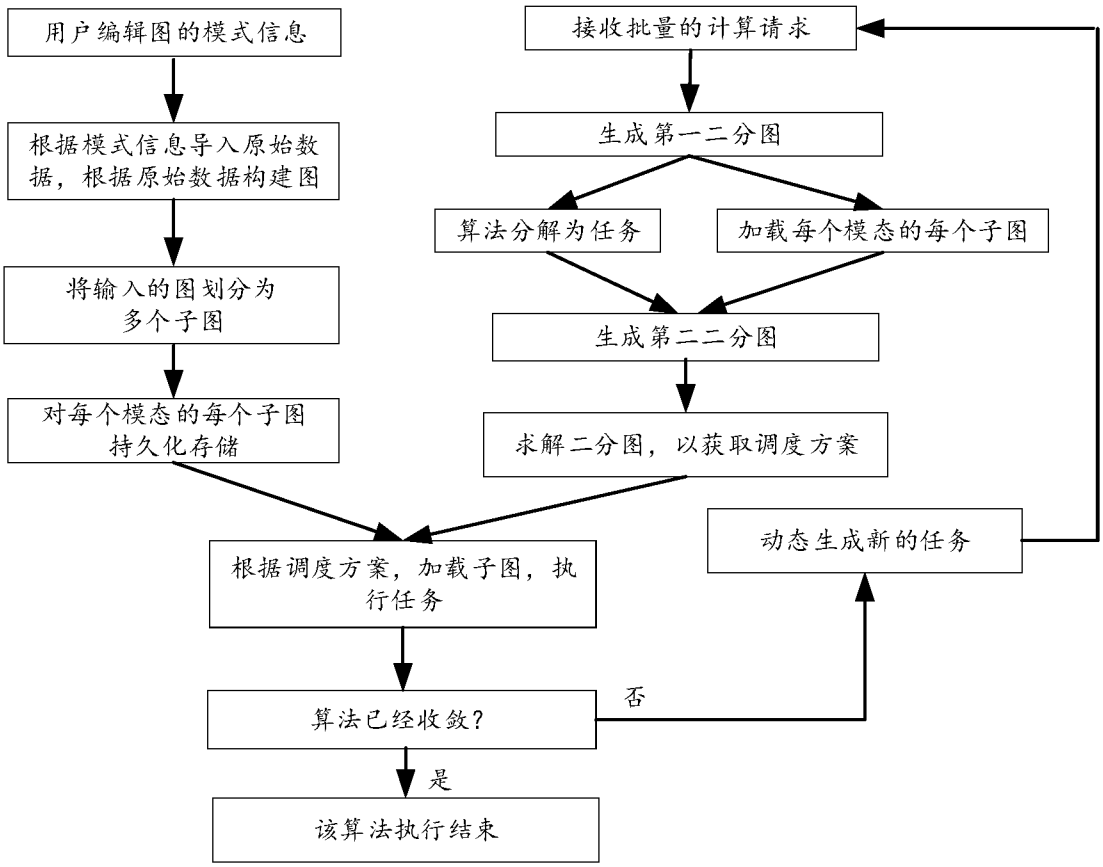
用户编辑图的模式信息
根据模式信息导入原始数据，根据原始数据构建图
将输入的图划分为多个子图
对每个模态的每个子图持久化存储
接收批量的计算请求
生成第一二分图
算法分解为任务
加载每个模态的每个子图
生成第二二分图
求解二分图，以获取调度方案
根据调度方案，加载子图，执行任务
动态生成新的任务
算法已经收敛？
否
是
该算法执行结束

图 12

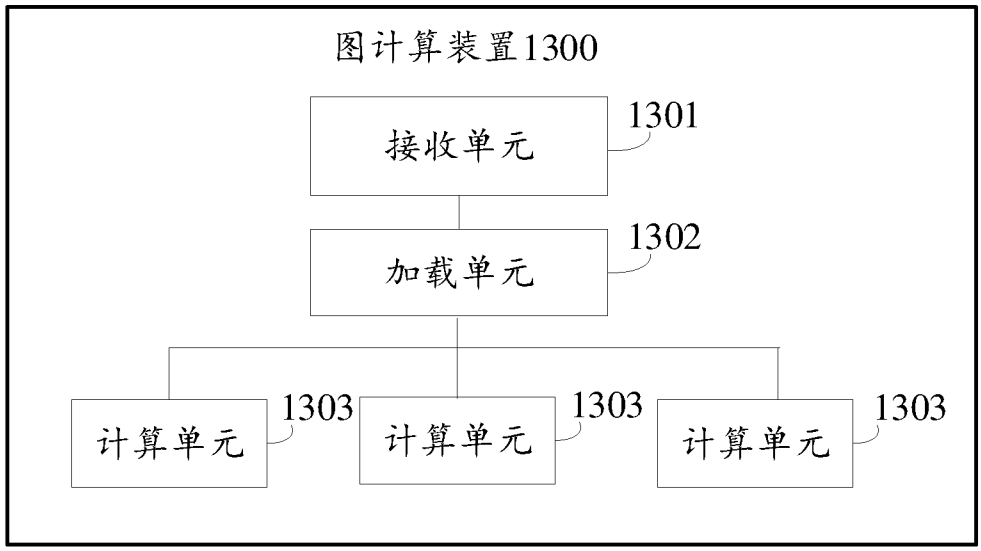

图 13

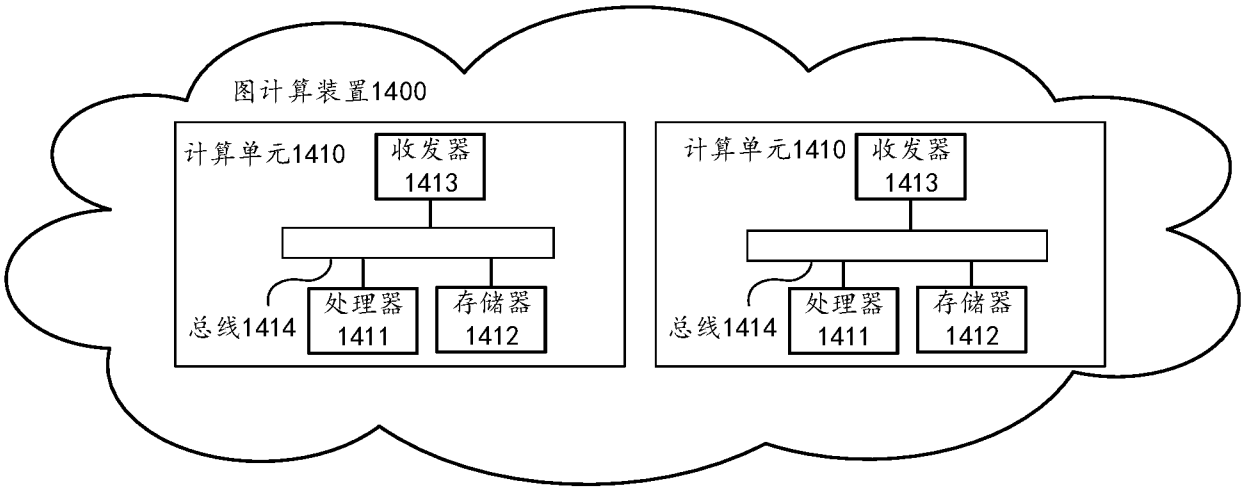
图计算装置1400
计算单元1410
收发器
1413
总线1414
处理器
1411
存储器
1412
计算单元1410
收发器
1413
总线1414
处理器
1411
存储器
1412

图 14

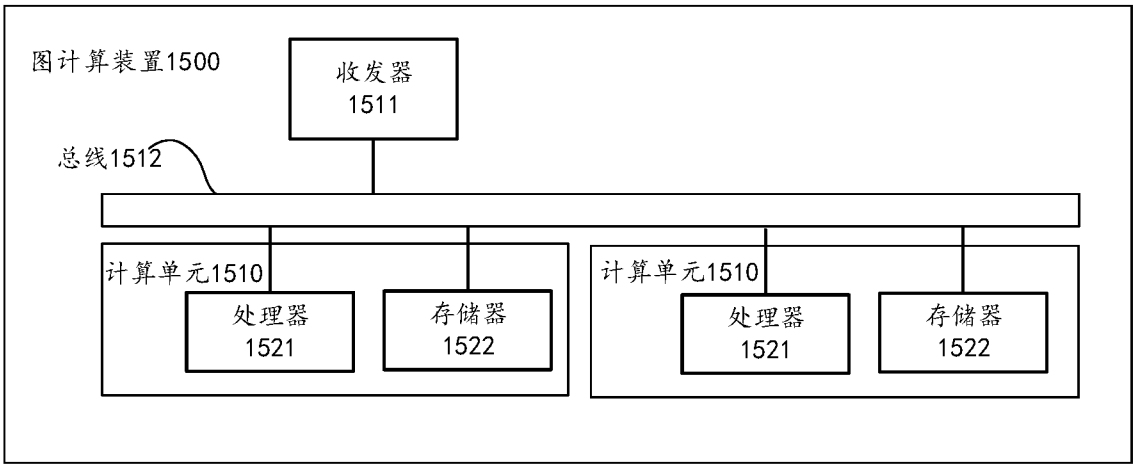
图计算装置1500
收发器
1511
总线1512
计算单元1510
处理器
1521
存储器
1522
计算单元1510
处理器
1521
存储器
1522

图 15

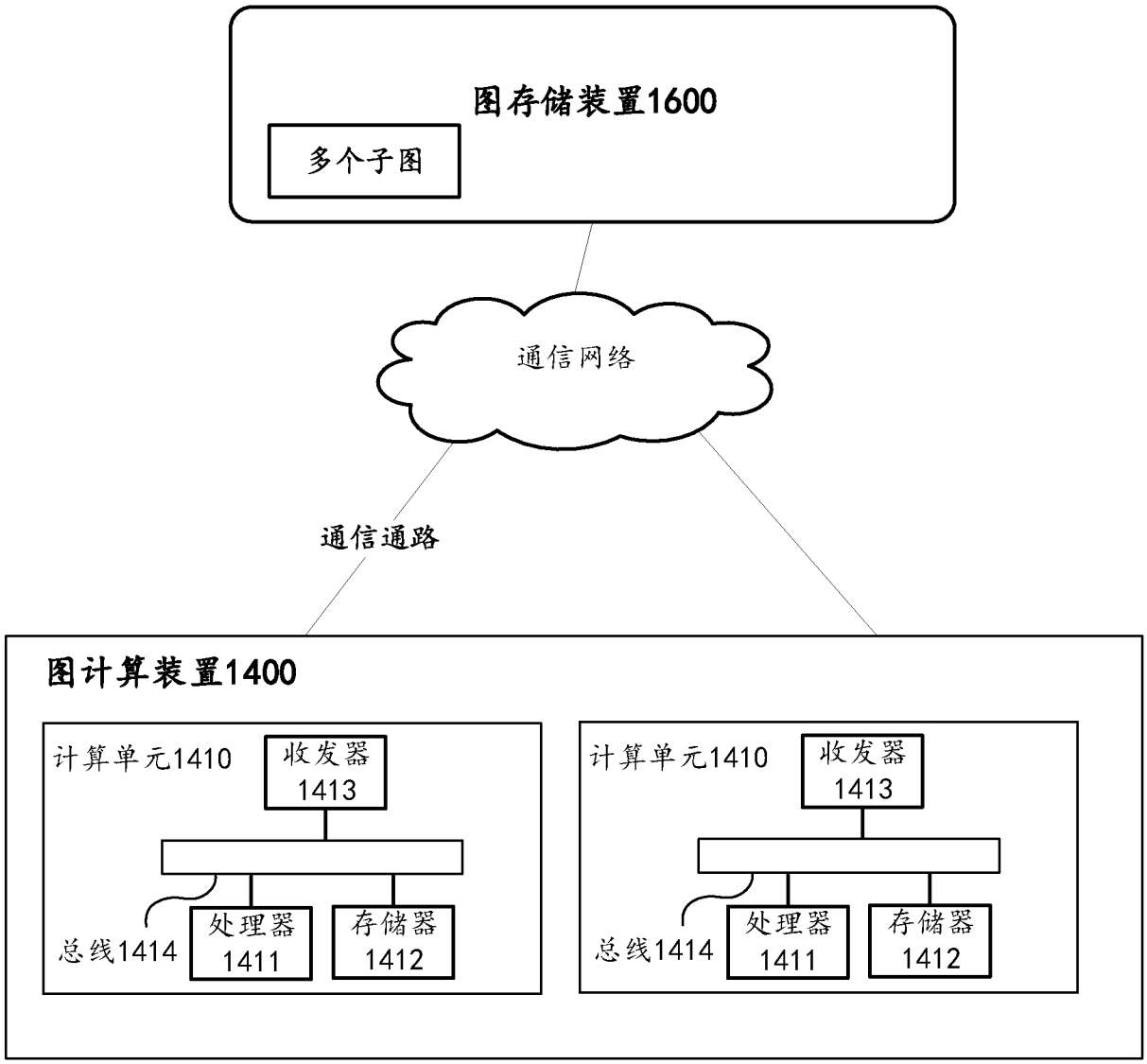
图存储装置1600
多个子图
通信网络
通信通路
图计算装置1400
计算单元1410
收发器
1413
总线1414
处理器
1411
存储器
1412
计算单元1410
收发器
1413
总线1414
处理器
1411
存储器
1412

图 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2019/125798

A. CLASSIFICATION OF SUBJECT MATTER

G06F 16/901(2019.01)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, CNKI, CNPAT, IEEE: 图, 子图, 块, 任务, 多个, 计算中心, 算法, 并行, 模态, 优先级, 加载, 释放, 复制, graph, subgraph, block, task, multi-, compute center, algorithm, parallel, modal, priority, load, release, copy

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	CN 103532710 A (RESEARCH AND EDUCATION CENTER OF THE CHINESE ACADEMY OF SCIENCES, DATA AND COMMUNICATIONS PROTECTION) 22 January 2014 (2014-01-22) description, paragraphs [0120]-[0149]	1-21
Y	CN 106445688 A (UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA) 22 February 2017 (2017-02-22) description, paragraphs [0085]-[0091]	1-21
A	CN 103793442 A (SUPERMAP SOFTWARE CO., LTD.) 14 May 2014 (2014-05-14) entire document	1-21
A	CN 102841816 A (BEIJING REMOTE SENSING INSTITUTE et al.) 26 December 2012 (2012-12-26) entire document	1-21

☐ 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	Date of mailing of the international search report
28 February 2020	**24 March 2020**

Name and mailing address of the ISA/CN	Authorized officer
China National Intellectual Property Administration (ISA/CN) **No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088** **China**	
Facsimile No. **(86-10)62019451**	Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/125798

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 103532710 A	22 January 2014	None	
CN 106445688 A	22 February 2017	None	
CN 103793442 A	14 May 2014	None	
CN 102841816 A	26 December 2012	None	

国际检索报告

国际申请号
PCT/CN2019/125798

A. 主题的分类

G06F 16/901(2019.01)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, CNKI, CNPAT, IEEE: 图，子图，块，任务，多个，计算中心，算法，并行，模态，优先级，加载，释放，复制，graph, subgraph, block, task, multi-, compute center, algorithm, parallel, modal, priority, load, release, copy

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
Y	CN 103532710 A (中国科学院数据与通信保护研究教育中心) 2014年 1月 22日 (2014-01-22) 说明书第[0120]-[0149]段	1-21
Y	CN 106445688 A (电子科技大学) 2017年 2月 22日 (2017-02-22) 说明书第[0085]-[0091]段	1-21
A	CN 103793442 A (北京超图软件股份有限公司) 2014年 5月 14日 (2014-05-14) 全文	1-21
A	CN 102841816 A (北京市遥感信息研究所 等) 2012年 12月 26日 (2012-12-26) 全文	1-21

☐ 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	国际检索报告邮寄日期
2020年 2月 28日	2020年 3月 24日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451	受权官员 马春黎 电话号码 86-(10)-53961336

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2019/125798

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 103532710 A	2014年 1月 22日	无	
CN 106445688 A	2017年 2月 22日	无	
CN 103793442 A	2014年 5月 14日	无	
CN 102841816 A	2012年 12月 26日	无	