计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220778
61(1): 20−42, 2024

图计算体系结构和系统软件关键技术综述

张 宇[1,2,3,4] 姜新宇[1,2,3,4] 余 辉[1,2,3,4] 赵 进[1,2,3,4] 齐 豪[1,2,3,4] 廖小飞[1,2,3,4] 金 海[1,2,3,4]
王 彪[5] 余 婷[5]

[1](大数据技术与系统国家地方联合工程研究中心(华中科技大学) 武汉 430074)
[2](服务计算技术与系统教育部重点实验室(华中科技大学) 武汉 430074)
[3](集群与网格计算湖北省重点实验室(华中科技大学) 武汉 430074)
[4](华中科技大学计算机科学与技术学院 武汉 430074)
[5](之江实验室 杭州 311121)
(zhyu@hust. edu. cn)

Review of Key Technologies in Graph Processing Architectures and Systems Software

Zhang Yu[1,2,3,4], Jiang Xinyu[1,2,3,4], Yu Hui[1,2,3,4], Zhao Jin[1,2,3,4], Qi Hao[1,2,3,4], Liao Xiaofei[1,2,3,4], Jin Hai[1,2,3,4], Wang Biao[5], and Yu Ting[5]

[1](*National Engineering Research Center for Big Data Technology and System* (*Huazhong University of Science and Technology*), *Wuhan* 430074)
[2](*Services Computing Technology and System Lab* (*Huazhong University of Science and Technology*), *Ministry of Education*, *Wuhan* 430074)
[3](*Cluster and Grid Computing Lab* (*Huazhong University of Science and Technology*), *Wuhan* 430074)
[4](*School of Computer Science and Technology*, *Huazhong University of Science and Technology*, *Wuhan* 430074)
[5](*Zhejiang Lab*, *Hangzhou* 311121)

Abstract In recent years, some progress has been made in the key technologies of the architecture and systems software for graph processing. Large-scale graph processing has also been widely used in many fields, including scientific computing, machine learning, social networks, intelligent transportation, bioinformatics, etc. However, most real-world graph computations have characteristics such as dynamic changes and complex and diverse application requirements. This poses new demands and challenges for graph processing in terms of basic theory, architecture, and key technologies of systems software. To address these challenges, researchers have proposed a series of graph processing systems and accelerators, which optimize the graph processing process through technologies such as high-performance computing and parallel computing. Furthermore, in order to meet the demands of practical application scenarios, various graph processing frameworks and algorithms are constantly being innovated and optimized, thus enhancing the practical value of graph processing in terms of processing large-scale graph data and improving computational efficiency. We review the research and development status of key technologies in graph processing architecture and systems software, and summarize, compare, analyze the latest progress of research at home and abroad, and select fields closely related to national economy and people's livelihood in combination with national development strategies and major application requirements. The industry progress of graph processing-related

收稿日期: 2022-09-01; **修回日期**: 2023-06-13
基金项目: 国家重点研发计划项目(2022YFB2404202); 国家自然科学基金项目(62072193); 之江实验室开放课题(2021KD0AB01); 之江实验室重大科研项目(2022PI0AC03)
This work was supported by the National Key Research and Development Program of China (2022YFB2404202), the National Natural Science Foundation of China (62072193), the Open Research Projects of Zhejiang Lab (2021KD0AB01), and the Major Scientific Research Project of Zhejiang Lab (2022PI0AC03).
通信作者: 赵进(zjin@hust. edu. cn)

technologies is analyzed and summarized from typical applications. Finally, the future technical challenges and research directions are prospected.

Key words graph processing; architecture; systems software; graph traversal; graph mining; graph neural network; single machine system; distributed system; accelerator; industry applications

摘 要 图计算作为分析事物之间关联关系的重要工具，近年来已成为各国政府及公司争夺的关键技术. 学术界和工业界在图计算体系结构和系统软件关键技术方面取得了一定进展. 然而，现实场景图计算大多具有动态变化、应用需求复杂多样等特征. 这给图计算在基础理论、体系架构和系统软件关键技术方面提出了新的需求，同时也带来了新的挑战. 为应对这些挑战，科研人员提出了一系列图计算系统或图计算加速器，通过高性能计算、并行计算等技术来优化图计算过程. 综述国内外图计算体系结构和系统软件关键技术的研究发展现状，对国内外研究的最新进展进行归纳、比较和分析，并结合国家发展战略和重大应用需求，选取与我国国计民生密切相关的领域，从典型应用分析总结图计算相关技术的行业进展. 最后，就未来的技术挑战和研究方向进行展望.

关键词 图计算；体系结构；系统软件；图遍历；图挖掘；图神经网络；单机系统；分布式系统；加速器；行业应用

中图法分类号 TP301

万物皆关联. 图计算作为重要的大数据分析手段，已在多个行业领域得到应用，接下来我们将详细介绍图计算的基本概念和理论，以及一些关键技术.

1）图计算——数据处理的关键使能技术和赋能手段

随着大数据时代的来临，人类生活进入了数据爆炸时代. 大数据有“4V”的特征，分别是数据体量大、数据类型多、增长速度快以及价值密度低. 图作为最经典、最常用的数据结构之一，现实生活中很多数据经常被抽象为多种多样的图结构数据. 图中的顶点可以代表不同的实体，图中的边可以代表不同实体之间的关系. 由于图是表达事物之间复杂关联关系的数据结构，因此现实生活中的诸多应用场景都需要用到图，例如，淘宝用户好友关系图、道路图、电路图、病毒传播网、国家电网、文献网、社交网和知识图谱. 图计算技术因此快速发展，它们通过对大型图数据的迭代处理，获得图数据中隐藏的重要信息. 在实际的场景应用中，图计算应用逐渐朝属性多类型、结构多变和属性多维的方向发展. 除了常规的图算法外，如动态图处理、图挖掘和图学习等复杂的图算法不断涌现，这些图算法在真实场景中发挥着举足轻重的作用.

常规图算法，例如重要度排名的 PageRank 算法、视频推荐的 adsorption 算法、道路选择的单源最短路算法和聚类的连通分量算法，通过对图数据进行反复遍历和迭代处理，从而挖掘图数据中隐藏的重要信息，例如重要度排名、最短路径和连通分量. 这类图算法的许多操作都建立在遍历操作的基础之上，并且通常关注于在图上执行线性代数类的计算操作. 相比于传统计算模式，迭代图算法对关联关系型数据具有丰富、高效和敏捷的分析能力，并在实际生活中应用广泛. 例如，Google 需要定期对 Web 中数以亿计的网页进行影响力排序，Facebook 需要对其社交网络图进行迭代分析以掌控社交网络的结构状态和提高广告的推送准确率. 为了从动态图中提取有用信息，大量动态图算法被提出，并在许多领域有着广泛的应用，比如社交网络分析[1-2]、实时金融欺诈检测[3-4]、异常检测[5] 和推荐系统[6].

图挖掘（graph mining）作为数据挖掘的重要组成部分已经引起了学术和工业界的广泛关注. 图挖掘是指利用图模型从海量数据中发现和提取有用知识和信息的过程，旨在发现图中特定的结构或模式. 图挖掘技术除了具有传统数据挖掘技术所具有的性质外，还具有数据对象关系复杂、数据表现形式丰富等特点，是处理复杂数据结构的理想工具. 通过图挖掘来获取知识和信息已广泛应用于各种领域，如社会科学[7-10]、生物信息学[11-14]、化学信息学[15-18]. 具体地，图挖掘可用于发现社交媒体数据中的结构-内容关系，挖掘社区密集子图，提取蛋白质-蛋白质或基因相互作用网络中的网络基序或重要子图，发现蛋白质结构或化学化合物中的 3D 基序等.

图作为非欧几里得空间数据的典型代表可以表征万物之间的关联关系. 但是由于图数据的不规则性，现有深度学习模型[19-20]，如处理欧几里得空间（Eucli-

dean space)[21]数据,其规则化数据的性质设计无法直接应用在图结构数据上.为此,图神经网络(graph neural network, GNN)[22]应运而生.图神经网络对非欧氏空间数据建立了深度学习框架,相比传统网络表示学习模型,它对图结构能够实施更加深层的信息聚合操作.目前,图神经网络能够解决诸多深度学习任务,例如节点分类[23]、链路预测[24]、图聚类[25]、图分类[26]和推荐系统[27].

鉴于图计算重要的社会经济和学术研究价值,目前世界各国均对图计算展开了重要规划和科研布局.例如美国国防高级研究计划局启动的HIVE项目研发图计算加速器,我国也相应启动了国家重点研发计划"面向图计算的通用计算机技术与系统".未来,作为大数据处理的关键使能技术和赋能手段,图计算将担负起服务国家新基建和"数字中国"战略的重要使命,促进国家经济与社会的数字转型、智能升级和融合创新.

2) 基础理论与关键技术

图是一种通用的数据结构,由顶点和顶点之间的边构成,通常可以表示为$G=(V,E)$,其中V表示顶点集合,E表示边的集合.当顶点i和顶点j之间存在关系,则可以用$e=(v_i;v_j)$来表示.在图中,每个顶点/边都可以拥有自己的属性值,例如在图计算和图挖掘中,表示为顶点/边的状态值;在图神经网络中,表示为顶点/边的特征向量.而且,同一个顶点/边可以拥有多种属性值,这样的网络统称为异质图.例如,在社交网络中,每个人都可以被视为图中的一个顶点,而边则表示为人与人之间存在联系,顶点的属性值可以表示为个人的热度值,而边的属性值则表示为有关联的2个人之间的紧密程度.

图数据结构很好地表达了数据之间的关联性,关联性计算是大数据计算的核心,通过获得数据的关联性,可以从海量数据中得到潜在的信息.因此,每天有大量图算法,或称作并发图分析任务(提交各种图算法所生成的任务或者同一图算法在用户给定不同参数时所生成的任务),并发地运行在同一图数据分析平台上对其上的图数据进行迭代分析,以对同一个图的同一个快照或不同快照进行全方位的分析,获得多样的信息,为不同应用产品提供各类信息服务.例如BC(betweenness centrality)算法就是执行多个独立的单源最短路算法.

在现实世界中,图通常是动态变化的,称为动态图.动态图通常可以由一段时间内多个静态图来表示,每一个时间段内的静态图称为快照.因此随着时间的不断推移,生成的图快照的数量也随之增加.例如,在社交网络中,用户的好友关系是不断发生变化的,每隔一段时间结交到新的好友或者与好友失去联系,都会使得图结构的顶点和边的数量都发生变化.然而,现实的动态图更新(改变图中的顶点或边)非常频繁,并且具有随机性.因此如何使用或者设计新的存储格式来高效支持动态图更新成为工业界和学术界研究的热点.

图计算具有的数据规模大但局部性低、计算-访存比小、关联迭代过程复杂等特点,对以控制流体系结构为主的现代计算机系统带来了并行流水执行效率低、访存局部性低和内外存通道有限、锁同步的扩展性差等系列挑战.因此,图计算近年成为学术界和工业界的研究热点.为了解决通用架构面对大规模图计算的诸多问题,近年围绕体系结构、运行时系统和编程环境,国内外科研人员均展开了广泛的基础研究和关键技术攻关,人们提出了一系列图计算加速器、硬件加速技术、图计算系统和优化技术,取得了显著性能提升.以图计算为载体的关系型数据分析典型应用,包括推荐系统、金融风控、链接预测等,也逐渐走进各个行业领域.然而,现实场景图计算大多具有动态变化、应用需求(如动态图分析处理、图模式挖掘、并发图处理、图神经网络训练和推理)复杂多样特征,这给图计算在基础理论、体系架构、系统软件关键技术方面提出了新需求,带来新挑战.

1 国际研究现状

1.1 图存储与动态图更新

在静态图中,存在很多常用的数据存储格式,比如邻接矩阵(adjacency matrix, AM)、边数组(edge array, EA)、邻接表(adjacent list, AL)和压缩稀疏行(compressed sparse row, CSR).邻接矩阵支持快速的插入、删除以及修改,时间复杂度都为$O(1)$,但是需要大量的存储空间,空间复杂度为$O(V^2)$.邻接表的空间复杂度为$O(V+E)$,并且允许快速更新,但是由于内存不连续,导致低效的图遍历.CSR能同时提供空间效率和快速遍历,因此在静态图中,CSR数据存储格式被工业界和学术界广泛应用.

最早开始研究动态图存储时,许多科研工作者考虑直接使用常用的静态图存储结构来支持动态图存储.然而,在静态图中被广泛应用的CSR数据存储格式并不适用动态图.对于CSR而言,更新的开销是非常大的.因为每次更新都需要在整个数组中移动

图数据以匹配压缩格式. 这是因为动态图的数据是动态变化的, 需要频繁的更新操作, 导致大量的访存操作. 并且动态图的数据更新是随机的, 所以动态图更新的数据局部性差. 随着现实中的图规模不断地增大, 动态图更新的规模也随之增加. 为了提高动态图更新的效率, 需要大规模的并行处理. 此外, 动态图的数据存储格式不仅用于动态图更新, 还用于动态图计算, 所以动态图的数据存储格式还需要兼顾图计算的访存局部性. 因此, 动态图的数据存储格式要支持高效的访存操作、并行操作以及考虑图计算的访存局部性, 这给动态图数据存储格式的设计带来了巨大的挑战. 目前许多工作对动态图的数据存储格式进行了深入的研究, 这些工作可以根据基本数据存储格式分为 4 类, 分别是基于邻接表的数据存储格式、基于 CSR 的数据存储格式、基于树的数据存储格式以及混合数据结构的数据存储格式.

1.1.1 基于邻接表的数据存储格式

在基于邻接表的动态图数据存储格式中, 每个顶点都有一个与该顶点相邻的边集的集合. 使用邻接表的数据存储格式可以高效地处理更新, 因为顶点之间的数据互不影响, 可以高度并行.

STINGER[28] 是一种针对动态图设计的高性能、可移植、可扩展的数据结构. STINGER 数据结构基于邻接表. 图顶点数据存储在 1 个连续数组中, 每个顶点数据存储 1 个指向边的指针. 每个顶点的边被划分为多个块, 这些块通过指针的方式连接在一起, 存储在 1 个基于块的链表中(链表中每个节点存储 1 块数据). STINGER 的顶点和边都有类型, 1 个顶点可以有不同类型的相邻边, 1 个块只包含 1 种类型的边. 此外, STINGER 还提供了边类型数组(edge type array, ETA)来快速索引不同类型的边. STINGER 数据结构的并行体现在 2 个方面: 一个是顶点之间的并行; 另一个是每个顶点的边的并行. 其中边的并行是指块链表中的一块可以并行处理. 为了进一步提高并行度, STINGER 支持动态图的批量更新. 对于非无尺度图, STINGER 会先对更新进行排序, 以便于将发生在同一顶点上的所有边更新分组在一起. 对于无尺度图, 没有排序过程. 因为少数顶点将面临很多更新, 大多数顶点只会有 1 个更新甚至没有更新, 从而导致工作负载不均衡.

aimGraph[29] 是一种新的动态图存储数据结构. 该结构旨在通过直接在 GPU 上使用自主内存管理来提高更新效率, 同时保持低的内存占用. aimGraph 通过将内存管理转移到 GPU 上, 可以实现图结构的高效更新和快速初始化, 达到较好的性能, 因为无需额外的内存分配调用或重新分配过程.

为了提高 GPU 平台上动态图的边查找和更新速度, Awad 等人[30] 设计了一种使用哈希表的快速动态图存储数据结构. 该数据结构基于邻接表, 使用动态数组存储顶点数据. 由于哈希表数据结构的更新和搜索效率都高, 所以每个顶点的边可以使用一个哈希表存储. Awad 等人[30] 使用 SlabHash[31] 作为基本的哈希表数据结构. SlabHash 是一种针对 GPU 的动态哈希表数据结构, 它由多个桶组成, 每个桶中存储一个指针, 指向一个基于块的链表. 此外, Awad 等人[30] 还在 SlabHash 的基础上进行扩展, 提供了 2 种动态图数据结构的变体: 第 1 种变体是使用 SlabHash 的并发映射, 适用于每条边需要存储一个值的情况; 第 2 种变体是使用 SlabHash 的新并发集合, 适用于每条边不需要存储一个值的情况.

1.1.2 基于 CSR 的数据存储格式

CSR 数据格式被广泛应用于静态图中, 并使用 3 个数组表示稀疏图, 分别是顶点数组、边数组和顶点属性数组. 目前, 基于 CSR 的数据存储格式多数与 PMA[32](packed memory array)数据结构相结合, 以支持快速的查询和更新, 比如 PCSR[33], PPCSR[34], GPMA[35]. 此外, 基于 CSR 的动态图数据结构还有 LLAMA[36] 等.

PMA[32] 存储数据在一个有序的数组中, 并且数据之间存在空隙, 即无效数据. PMA 搜索的时间复杂度是 $O(\mathrm{lb}n)$, 插入和删除的时间复杂度为 $O(\mathrm{lb}n)$. PMA 由一棵隐式完全二叉树组成. 二叉树的叶子节点大小为 $\mathrm{lb}n$(又称为段大小), 并且叶子之间数据存储是连续的. 二叉树的每个节点都有一个上限和下限的密度(节点中的数据除以节点中的空间). 当一个节点的密度低于下限或者高于上限时, 可以与邻居节点一起重新分配数据以达到合理的密度. 由于 PMA 存储在数组中的数据之间存在空隙, 在插入或删除时只需要移动少量元素以避免在每次插入或删除后更改整个数据结构. 由于数据在内存或磁盘中按排序顺序物理存储, 因此 PMA 与 CSR 类似, 可用于支持极其高效的范围查询.

目前, 很多基于 CSR 的数据存储结构与 PMA 相结合, 并且在多种平台上都有实现. 其中在 CPU 平台上实现的工作有 PCSR[33] 和 PPCSR[34]; 在 GPU 平台上实现的工作有 GPMA[35].

PCSR[33] 与 CSR 类似, 使用顶点列表和边列表. 但是不同的是 PCSR 使用 PMA 存储边列表, 而不是使用一个数组. 顶点列表中的元素存储顶点的边在

边列表中的起始位置和结束位置. 在边列表中, 每个顶点的边的前面都存在一个对应的哨兵, 它指向顶点列表中对应的源顶点, 用于更新源顶点的数据.

GPMA[35] 是一种基于 PMA[32] 的 GPU 动态图数据存储格式. GPMA 设计了 2 种面向 GPU 的算法: GPMA 和 GPMA+, 以支持高效的并行批量更新. 其中 GPMA 探索了一种基于锁的方法来支持 GPU 上的动态图更新. 虽然 GPMA 在并发更新冲突较少的情况下可以有效工作, 但也有发生大规模冲突的情况. 因此, GPMA 提出了一种无锁算法, 即 GPMA+. GPMA+是一种自下而上的方法, 它优先考虑发生在相邻位置的更新. GPMA+能够最大化合并内存访问并实现与 GPU 上计算单元数量相关的线性性能扩展.

文献 [33−35] 的工作都是 CSR 和 PMA 相结合的数据存储格式, 此外还有一些基于 CSR 的数据存储格式, 例如 LLAMA[36]. LLAMA 提出了一种支持图更新的多版本 CSR 数据存储格式. 当一个图首次加载到 LLAMA 中时, 它驻留在单个基本快照中, 随后每个增量的加载都会创建一个新的“增量”快照, 其中包含新加载的顶点和边. 因此, 一个顶点的邻接表可以分布在多个快照中, 新的快照会有指针指向旧的快照, 快照之间共享数据, 从而实现轻量级更新. 因为 PMA 使用连续的数组存储边并且对数据进行排序, 所以基于 CSR 的数据结构遍历图和查询的效率都比较高. 但是由于所有的边共享一个数组, 不同更新之间可能会频繁冲突, 所以难以支持高并发的快速更新.

1.1.3 基于树的数据存储格式

基于树的数据存储格式使用树作为基本数据结构. 这类数据结构通常使用一个搜索树以支持快速查询, 并且在树的非叶子节点或叶子节点上做优化. 基于树的动态图数据存储格式的主要工作有 Aspen[37], TEGRA[38], Teseo[39] 等.

Aspen[37] 提出一种压缩的纯功能(purely-functional)搜索树数据结构, 称为 C-trees. 纯功能数据结构在修改时保留其以前的版本, 并产生更新的新结构. C-trees 提供轻量级的快照, 并且可以支持并发运行低延迟的查询和更新. C-trees 扩展了纯功能搜索树, 克服了局部性低和空间利用率低的问题. C-trees 的主要思想是在树上应用块技术, 每个树节点上使用连续数组存储多个元素, 从而提高了局部性. 在元素是整数的情况下, C-trees 数据结构可以利用元素在块中按排序顺序存储的事实, 使用差异编码来压缩数据. Aspen 将顶点集表示为一棵树, 称为顶点树. 顶点树中的每个顶点通过存储其相邻边的树来表示其邻接信息, 称为边树. 附加信息存储在顶点树中, 以便于查询基本图结构属性, 例如边和顶点的总数.

Aspen[37] 只允许存储动态图的几个最新版本, 并且不支持存储中间状态或更新先前查询的结果. TEGRA[38] 利用持久数据结构来构建分布式、版本化的图状态存储. 持久数据结构的关键思想是在修改时保持数据的先前版本, 从而允许访问早期版本. TEGRA 使用自适应基数树(adaptive radix tree, ART)[40] 的持久版本作为其数据结构. ART 提供了一些对图存储有用的属性, 例如高效更新和范围遍历. 持久自适应基数树(persistent adaptive radix tree, PART)[41] 通过简单的路径复制为 ART 添加了持久性. TEGRA 将图存储在顶点树和边树, 并通过生成包含差异的新根节点来创建轻量级快照.

Teseo[39] 提出了一种支持事务的基于 B+树的动态图存储数据结构, 称为胖树(fat tree). 胖树将树的叶子的大小扩展到 MB 的数量级, 以减少遍历叶子节点的随机访存开销. 树的叶子节点数据存储结构是稀疏数组, 即有空隙的数组. 数组中的数据是有序的, 在更新之后可能需要重新分布数据. 胖树使用 ART[42] 作为主索引以高效支持查询. 此外, 还提供一个哈希索引, 映射每个顶点到树中的地址. 基于树的数据存储格式虽然支持快速的更新, 但是树数据结构的数据存储是不连续的, 因此在查找和更新时存在大量的随机访存.

1.1.4 混合数据结构的数据存储格式

与静态图不同, 动态图不仅要考虑图的表示, 而且还要考虑更新的边的表示. 此外, 不同度数的顶点使用不同的数据结构的遍历和搜索性能会不同. 因此, 使用单一的数据存储格式可能难以高效支持动态图. 现有许多工作使用混合数据存储格式来存储动态图数据. 混合数据存储格式通常使用多个数据存储格式进行存储, 通常的划分方法分为 2 种, 分别是以图表示和更新的边使用不同的数据存储格式、顶点的度数大小来划分. 其中前者工作有 GraphIn[43], GraphOne[44] 等, 后者工作有 DegAwareRHH[45], Terrace[46] 等.

GraphIn[43] 使用一种混合的数据结构, 包括边列表和压缩矩阵格式. 其中边列表存储增量更新, 压缩矩阵格式存储图的静态版本. 边列表支持快速的更新, 而不会降低增量计算的性能; 压缩矩阵格式允许对图的整个静态版本进行更快的并行计算. GraphIn 在需要时会合并更新列表和静态图.

与 GraphIn 类似, GraphOne[44] 也使用了 2 种图数据存储格式. 不同的是 GraphOne 使用邻接表存储图,

即使用边列表和邻接表. 边列表格式保留了数据到达顺序, 并为快速更新提供了良好的支持, 因为每次更新都是将数据简单地添加到列表的末尾. 邻接表保留了由源顶点索引的顶点的所有邻居, 这为图分析提供了有效的数据访问. 图分析和查询在执行期间需要最新数据的不可变快照. 边列表格式为细粒度快照创建提供了自然支持. 同时, 邻接表格式通过其粗粒度快照功能补充边列表. 此外, GraphOne 提出了一种新的数据抽象 GraphView, 可以以 2 个不同的粒度访问数据, 并且只需要少量的数据拷贝. GraphIn 和 GraphOne 将图表示和更新的边分别使用不同的数据结构存储, 一定程度上弥补了各自的缺陷. 但是一个理想的数据结构应取决于图算法的访问模式和更新的成本. 如果顶点的度数较低, 那么简单的数据结构(例如数组)会产生最少的间接访存并支持高效的遍历和更新. 但是, 如果顶点的度数较高, 则可能更适合复杂的数据结构, 例如具有更好搜索和更新性能的哈希表和树. 因此, 基于顶点度数的划分策略将会是一个很好的方案. 目前, 基于顶点度数的划分策略的主要工作有 DegAwareRHH[45], Terrace[46] 等.

DegAwareRHH[45] 是一种高性能动态图数据存储格式, 通过使用具有高数据局部性的紧凑哈希表以存储大型无尺度图. DegAwareRHH 支持多进程和分布式内存, 并且可以在大型无尺度图上执行动态图构建. DegAwareRHH 根据顶点度数的大小将顶点分为 2 类: 中高度数顶点和低度数顶点, 不同的类别使用的数据结构不同. 对于中高度数顶点使用基于邻接表的数据结构, 每个顶点的边列表使用哈希表存储, 以便于快速搜索一条边. 对于低度数顶点, 所有的顶点的边数据使用一个哈希表存储, 以减少低度数顶点的相对开销高的操作.

与 DegAwareRHH 类似, Terrace[46] 也使用分层数据结构设计, 根据顶点的度数将顶点的邻居存储在不同的数据结构中. 与 DegAwareRHH 不同的是 Terrace 按照顶点的度数划分为 3 类顶点, 分别是低度数顶点、中度数顶点和高度数顶点. 这 3 类顶点对应的数据存储格式分别是排序的数组、基于 PMA 的数据存储格式和 B 树. 低度数顶点存储在一个排序的数组中, 并且为每个顶点分配缓存行大小倍数的空间, 减少甚至避免了缓存命中率低的问题. 中度数顶点的数据存储格式基于 PMA, PMA 支持缓存高效的遍历, 因为顶点的所有邻居都存储在连续的内存位置中, 类似于 CSR 的边列表. 然而, 在 PMA 中执行更新或查询操作的开销随着 PMA 的数据大小增大而逐渐高于 B 树, 因此 Terrace 限制存储在 PMA 的顶点度数. 高度数顶点使用 B 树存储数据, 以支持顶点的快速更新. 混合数据存储格式使用了多种数据存储格式, 一定程度上可以互相弥补各自的缺陷, 但是多种数据存储格式的合并或者之间的转换存在开销.

1.2 面向高性能图计算应用的体系结构

加速器因具备丰富的带宽资源、超高的并行能力及极低的数据传输延迟等技术优势, 是实现高效图计算的重要技术手段之一. 按照加速器物理器件的性质来看, 目前的图计算加速器可以被分为面向单图处理加速器和面向动态图处理加速器.

1.2.1 面向单图任务处理加速器

面向单图任务分析是目前主流的图分析模式, 为了解决面向单图任务分析带来的挑战, 为图计算定制专用的加速架构是一种高效的解决方案.

GPU 拥有众多的计算单元和丰富的带宽资源, 可提供比 CPU 更强的并行计算能力, 能够高效地支撑大规模图顶点遍历与更新. 现有大量的工作提出基于 GPU 的图计算加速器[47-52].

为解决 GPU 上图计算负载不均衡的问题, Back 40Computing(B40C)[47] 实现了基于 GPU 的高性能遍历算法. B40C 采用 Scan+Warp+CTA 任务划分策略, 能够较好地解决以顶点为中心的编程模型下 GPU 图处理的负载均衡问题. 具体而言, B40C 根据顶点度数执行混合调度策略, 解决同一个 Warp 中不同线程以及同一个 CTA(compute thread array) 中不同 Warp 间的负载不均衡问题. 此外, B40C 通过基于位掩码的缓存状态记录来减少图计算过程中的全局访存量, 从而提高系统性能.

Gunrock[48] 设计了基于 GPU 的图计算通用加速库, 提出了一个以数据为中心的编程抽象, 将高性能 GPU 计算原语与高级编程模型的优化策略相结合, 能够在 GPU 上快速实现高性能图计算原语. Gunrock 使用线程细粒度调度与基于 warp/CTA 的混合调度策略, 能够实现不同粒度间的计算任务负载均衡. 同时 Gunrock 采用了 CSR 和 Edge list 的混合数据结构来提高聚合访存的效率, 减少乱序访存带来的额外开销. 针对不同的图应用特征, Gunrock 还提供了特定的优化接口, 例如 SSSP 算法的优先调度、BFS 算法的 Push/Pull 切换调度.

为提升 CPU 上大规模图数据处理性能, Graphie[49] 提出了异步边数据流运行时, 通过将边分区异步流式传输到 GPU 来隐藏数据传输开销, 实现跨迭代高效重用图数据. 此外, Graphie 还提出了一种重命名技

术，通过重命名顶点和插入虚拟顶点来有效利用共享内存并激活边分区数据，从而提高图遍历的性能.

Groute[50]提出了一种多GPU异步编程模型与运行时环境，通过异步执行与通信来促进多GPU平台上图计算应用的负载均衡，从而提高多GPU节点的利用率. Tigr[51]提出了一种图数据转换框架，该框架通过分割转换将高度数顶点拆分为低度数顶点集，从而规则化真实世界中的图数据，并保证各种图数据分析算法的正确性.

为减少在GPU上处理大规模图数据时CPU与GPU间的高额数据传输开销，Subway[52]提出了一种快速子图生成算法，每轮迭代前通过GPU加速将待处理的图数据快速生成子图，再将子图加载入GPU进行处理，从而有效减少CPU与GPU间的数据传输开销. 此外，Subway通过延迟GPU显存中的子图数据与CPU内存中的图数据之间的同步来减少数据传输次数，进一步提高图处理性能.

现有研究表明，内存访问延迟过高、并行度不足等问题导致传统CPU架构在处理图应用时往往面临着严重的性能与能源损耗. FPGA作为一种介于通用芯片(如CPU，GPU)与定制化芯片(ASIC)之间的计算平台，一方面，提供大量计算资源以保证较高的并行度，另一方面，提供较好的可重构性以保证较低的能源损耗. 有大量的工作提出基于FPGA的图计算加速器[53-55].

GraphStep[53]提出了一种面向图处理的FPGA并发系统架构，该架构通过轻量级网络将专用图处理节点互连，并将图顶点存储于相对应的小型分布式内存中，从而提供了一种可扩展方法来映射图计算应用，以便利用FPGA嵌入式内存的高带宽与低延时来加速图处理.

CyGraph[54]提出了一种基于FPGA的图遍历实现方法，该方法优化了CSR存储格式，对BFS算法进行了重构与优化，能够有效减少共享内存访问.

HitGraph[55]是一个用于加速以边为中心的图算法的FPGA框架，该框架通过划分图分区来实现顶点数据的有效缓存，并提高并行性，同时HitGraph对数据布局进行优化，以减少非顺序的外部存储访问与数据通信. 此外，HitGraph还包括一个设计自动化工具，能够根据用户的自定义参数生成高度优化的FPGA加速器的RTL(register transfer level)设计.

传统CPU架构在处理图应用时面临访存延迟较高、并行度较低等问题，尽管现有的图计算系统能在一定程度上缓解这个问题，但是其性能与性能功耗比依旧受限于顶层的硬件架构. 因此，许多研究人员与机构开始尝试为图计算设计专用加速芯片. 例如Graphicionado[56]和Ozdal等人[57]分别针对图计算设计了专用的流水线架构和可配置硬件加速器体系结构. 具体来说，这些架构能够对具有不规则访问模式和非对称收敛的以顶点为中心的迭代图应用进行加速；从细分角度对传统Cache架构进行改进，将传统的Uniform Cache细分为多个子Cache以保存不同类型的图数据；还能够支持异步执行模式.

图挖掘应用拓展了图计算应用的基本范式，利用图模型从海量数据中发现和提取有用知识和信息的过程，去挖掘图中特定的结构或模式. 相较于传统图计算应用所具有的特征外，还有数据关系复杂、数据表现形式复杂等特点，因此围绕着图挖掘的硬件加速器也相继被提出[58-62]. 从专门用于图模式匹配问题的片上加速器TrieJax[58]到以集合为中心的模式，专门为图挖掘设计新型架构扩展的NDMiner[62]，都是需求硬件方法的帮助来加速图挖掘问题，特别是在现实场景中，图挖掘问题的规模大，内存访问极其不规则，需要大量的资源才能完成.

不管是现有的图分析加速器还是图挖掘加速器，本质上还是挖掘数据中潜在的信息或者特定的子图结构模式，但无法对图数据进行学习. 为了解决这一问题，图神经网络专有加速器相继被提出[63-69].

之前的工作提出了一些预处理方法，在开始推理阶段之前在线或离线调整图的邻接矩阵以获得更好的局部性并减少冗余计算，而GCoD[67]创造性地专注于GCN训练阶段. 在训练期间，首先在分区图上对GCoD进行预训练；然后，通过图稀疏化和极化技术继续调整图邻接矩阵；最后，GCoD修剪具有小于指定阈值的非零元素，以平衡稀疏性和准确性. GCoD在修改图结构后，有一个重新训练过程来恢复测试精度. 为了避免显著的训练开销，GCoD在训练的早期阶段识别出稀疏子图，并且只在这些稀疏子图上重新训练. 该图在训练阶段结束时可以被分为2部分：一部分呈现更局部稀疏；另一部分呈现更局部密集. 这2种不同的工作负载在硬件方面的处理方式不同，稀疏分支处理不规则但轻量级的稀疏工作负载(主要在芯片上)，密集分支处理常规密集子图并结合基于块的微架构. 稀疏分支处理和密集分支处理双管齐下的加速器可以进一步提高整体硬件利用率和执行效率.

GCN具有混合执行的模式，即聚合操作和更新操作，其表现出截然不同的数据流特征. 现有的GCN

加速器通过将聚合以及更新的2个阶段转换成一系列稀疏密集矩阵乘法操作，因此可以通过设计一个统一的稀疏矩阵乘法加速器来加快这个操作. 然而现有的工作经常受到低效的数据移动的影响，留下了显著的优化空间. 因此专属GCN数据流加速器GROW[70]被提出，其基于Gustavson算法并用于构建基于行乘积的稀疏密集GEMM加速器. 针对2个不同量级的稀疏密集乘，分别设计了专属的图划分方式和缓存策略，进一步提高了GCN的数据局部性和并行性.

1.2.2 面向动态图任务处理加速器

由于动态图的图结构频繁发生变化，其相邻图顶点之间的状态更新存在复杂的依赖关系，这使得现有的软硬件方法在处理单调图算法时依然面临数据访问成本高和收敛速度慢的问题. 为了加快动态图处理的收敛速度，第1个国外动态图处理加速器JetStream[71]被提出.

JetStream通过将图更新转化为增量来加快动态图处理，并且扩展了最近提出的基于事件的加速器GraphPulse静态图增量加速器，以支持流式更新. JetStream通过事件驱动的计算模型处理累积和单调图算法，该模型限制对图顶点的较小子集的访问，有效地重用先前的查询结果以消除冗余，并优化内存访问模式以提高内存带宽利用率.

1.3 面向高性能图计算应用的系统软件

1.3.1 面向单图任务处理的系统软件

单机图计算系统能充分发挥单台机器处理图计算任务的能力，避免分布式系统下代价高昂的网络通信开销，但是此类系统受限于固定的硬件资源，无法实现很好的扩展性，处理的时间通常和图数据大小成比例. 单机图计算系统可以分为2类，即面向高端多核、大内存服务器的内存(in-memory)图计算系统，以及面向商用PC核外(out-of-core)图计算系统. 前者在处理过程中将图数据完全放入内存；后者通常利用磁盘存放图数据，采取一定的划分策略分块处理.

单机内存图计算系统往往拥有多核，并且支持超过1TB的超大内存，可以处理数百亿甚至数千亿条边的图数据. 因此许多基于单机图计算系统被提出[72-81]，例如Ligra[57]，Galois[74]，GraphMat[72].

与单机核外图计算系统相比，单机内存图计算系统将图数据存储在内存中，能够有效地减少磁盘I/O开销，但单机共享内存系统只能通过增加CPU数量或者扩展内存大小来实现可伸缩性. GraphMat[72]旨在建立用户友好的图计算框架与原生的、手动优化的代码间的连接，它采用顶点程序，并将其映射到后端的高性能稀疏矩阵操作，从而在不牺牲性能的情况下获得顶点编程框架的生产力优势，并实现了良好的多核可扩展性.

分布式图计算系统由多个计算节点组成，每个节点都拥有自己的内存与外存. 因此，与单机图计算系统相比，分布式图计算系统在可扩展性方面受硬件限制较少. 然而在分布式图计算系统中，图数据被分配到多个节点进行处理，因此数据划分策略对分布式图计算系统的性能影响较大，如何设计合适的数据划分策略是一个挑战，同时计算过程中需要进行通信，因此计算节点之间的通信成为性能瓶颈，系统的整体性能以及处理数据的规模都受到网络带宽的限制. 因此随着图数据规模的不断增大，分布式图计算系统[82-85]也逐渐成为研究热点.

Pregel[82]提出了一个适用于这项任务的计算模型. 程序被表示为一系列迭代，其中每个顶点都可以接收上一次迭代中发送的消息，向其他顶点发送消息，并修改其自身状态及其输出边的状态，或修改图形拓扑. 这种以顶点为中心的方法足够灵活，可以表达一系列广泛的算法. 该模型设计用于在数千台商用计算机集群上高效、可扩展和容错地实现，其隐含的同步性使程序推理更容易. 与分发相关的详细信息隐藏在抽象API后面，其结果是一个用于处理大型图的框架，该框架具有表达性且易于编程.

许多图算法显示不规则的访问模式. 因此，大多数图形处理系统要求图完全适合于内存，这就需要超大的集群. 像GraphChi[77]和X-Stream[78]这样的系统已经证明，通过依赖2级存储，可以在单个机器上处理具有数十亿个边缘的图，这种方法大大降低了处理大型图的门槛. 然而可以附加到单个机器上的存储容量是有限的，而兴趣图却在继续增长. 此外，基于附加到单个机器上的2级存储器的图处理系统的性能受到其到2级存储器带宽的限制.

Chaos[85]主要研究将基于2级存储的图计算系统扩展到基于多个机器节点的分布式图计算系统，通过并行访问每个机器节点上的局部内存，以支持在1万亿边的超大规模图上高效执行图算法的性能. Chaos采用了一种完全不同的方法来扩展2级存储上的图处理，没有执行复杂的分区步骤来实现负载平衡和局部性，而是执行非常简单的初始分区来实现顺序存储访问；通过使用X-Stream引入的流分区的变体来实现这一点. Chaos并不是在一台机器上为

每个分区定位数据,而是将所有图数据(顶点、边和中间数据,称为更新)均匀随机地分布在所有 2 级存储设备上. 数据存储在足够大的块中,以维持顺序存储访问. 由于不同的流分区可以有非常不同的边缘和更新数量,因此需要非常不同的工作量,Chaos 允许多台机器在同一个流分区上工作,使用工作偷取方法来平衡负载.

现有的分布式外核系统通常关注于优化 I/O 效率,而对通信成本关注较少. 对于远超出内存容量的大量图,由于用于组合的内存有限,这种减少的效果会小得多. 因此,如果存在一个能够有效地降低通信成本并设计一个可扩展的系统,那么就可以放松对网络的假设(只需要与磁盘吞吐量相当的网络带宽),从而在更多类型的硬件配置上实现分布式处理. DFOGraph[86] 填补了传统分布式内存图处理系统和核外图处理系统之间的性能差距. DFOGraph 以配备高速 NVMe ssd 和网络的集群环境为目标,以实现容量和性能的可伸缩性,并提高全核外图处理的整体效率. DFOGraph 应用的技术重点是优化 I/O 和通信效率,尽量避免不必要的磁盘和网络操作,自适应选择策略,以充分利用 CPU、磁盘或网络的瓶颈. DFOGraph 的关键选择是将以顶点为中心的 push 抽象和 2 级(节点间和节点内)面向列的分区结合起来. push 操作使各种优化成为可能,而分区缩小了随机访问的范围,并使推操作在分布式的全外核场景中实现,而不涉及过多的磁盘上的页面. 推送和 2 层面向列的分区支持有效的 I/O 和通信优化. 自适应选择压缩稀疏行(CSR)或双压缩稀疏行(DCSR)作为每个分区的边缘表示,降低了空间消耗和 I/O 成本. 消息被有效地过滤,只在网络上发送需要的消息,以减少网络流量. 因此,DFOGraph 只需要网络带宽相当于每个节点的磁盘吞吐量就可以向外扩展. 此外,DFOGraph 开发多种自适应通信策略,与磁盘和网络相关的操作被仔细地分解和流水线化,这在很大程度上隐藏了额外的优化延迟,并有助于更好地权衡 CPU、网络和 I/O 之间的关系.

近年来,许多图挖掘软件系统被提出,它们在输入图 G 中搜索满足算法条件的子图. 寻找子图的过程可以用搜索树来模拟,其中每个节点都是一个子图,k+1 层的子图是由 k 层的子图扩展而来. 基于搜索树模型,这些系统按编程模型分为两大类:模式不感知(pattern-oblivious)和模式感知(pattern-aware),又称以嵌入为中心和以集合为中心. 此外,图挖掘软件系统采用了多种技术以提高图挖掘问题的性能,比如图的遍历策略和多模式优化. 大多数图挖掘系统[87-94] 采用模式不感知模型来解决图挖掘问题,对于中间嵌入,采用了剪枝技术来防止重复和不必要的探索. 对于最终的嵌入,使用昂贵的同构测试来确定它们是否与模式同构. 而对于采用模式感知编程模型的系统[95-98],主要是解决图挖掘过程中子图同构测试和修剪搜索空间开销大的问题,通过生成匹配顺序[97] 和对称顺序[96] 以消除同构测试和重复枚举.

典型的 GNN 系统分为数据模块和计算模块. 其中,数据模块主要负责 I/O 数据以及预处理数据,计算模块主要负责算法模型的训练和推理. 在早期 GNN 加速系统的开发中,单机 GPU 的 GNN 系统是最先被关注的. 这是因为在图结构以及特征向量数据比较小的情况下,这些数据都可以直接存储在 GPU 内存中. 在这方面,国外涌现出大量的工作[99-108].

Marius++[109] 主要解决的是超大规模 GNN 训练和推理,其专注于 GNN 的扩展性训练和资源利用率. Marius++ 采用单机核外流水线小批量训练方式来训练数亿顶点大规模图,创新性地提出了基于磁盘优化的大规模 GNN 训练方法,使得基于磁盘小批量训练出来的模型与全图训练出来的 GNN 表征能力相近,并且最大限度地减少训练过程中的内存占用和端到端时间. 与文献 [99−108] 所述的国外涌现出的框架不同的是,Marius++ 并不要求所有的图结构数据和特征向量数据都保存在 GPU 内存中,极大地增加了 GNN 的扩展性,并与分布式 GNN 框架和单机多 GPU 的 GNN 系统不同的是,其不需要进行多节点或多 GPU 之间昂贵的特征向量通信,提高了 GPU 计算核利用率.

主流单机 GPU 的 GNN 系统都是将全部数据保存在 GPU 内存中进行 GNN 训练和推理,然而在实际应用场景中,GNN 训练和推理的图都是数亿、数十亿甚至是数百亿顶点规模的图,这时候单节点内存根本不可能全部存下这些图,因此现有分布式 GNN 系统[110-113] 不断被提出来加速大规模 GNN 效率.

现有的分布式 GNN 系统主要从 2 个方面进行优化:1)基于传统的 GNN 训练和推理调度方法,提高数据局部性以减少多个计算节点之间的特征向量通信,这部分工作以 Euler[110] 为代表;2)打破传统的 GNN 调度方法,设计全新的混合并行调度策略,从而减少机器节点的通信,这部分工作主要以 P3[113] 为代表. 然而,对于分布式 GNN 系统中的自动微分技术,目前还没有被实现. 现有的方式还是将每个机器节点本地的梯度进行自动微分然后同步.

1.3.2 面向动态图任务处理的系统软件

为了高效执行和处理动态图，现有大量基于动态图任务处理的系统软件被提出来[1, 114-117].

Kineograph[1]是基于增量的动态图处理系统，其主要采用 epoch 的更新方式来生成动态图的最新快照，并且设计一个增量计算引擎来处理最新图快照中发生改变的图结构数据以保证最终结果的正确性. 然而，Kineograph 是通过同步迭代模型 BSP 来执行增量计算，图顶点状态值传播缓慢.

为了能够同时处理动态图顶点/边增加或者删除的情况，KickStarter[117]采用了一种剪枝的方法来标记所有受影响的图顶点和边，通过重新计算受影响顶点来保证动态图处理结果的正确性. Tripoline[116]指出现有的增量图计算系统在查询方面往往需要依赖先验知识，通过在一个图查询的评估和另一个图查询的结果之间建立严格的约束，从而能够重复使用结果来加速评估，这将不依赖于任何先验知识. 然而，以上这些软件系统[1, 114-117]要么采用同步迭代模型（BSP），由于同步屏障的存在导致图顶点状态传递缓慢，要么采用异步执行模型，存在高额的运行开销和通信开销，均不能高效满足动态有向图中单调图算法的处理性能需求.

2 国内研究现状

2.1 图存储与动态图更新

动态图处理具有访存密集、局部性差、实时计算难、数据存储难和数据依赖高等特点. 动态图更新决定了动态图处理是否高效，图更新是对图顶点和边的插入、删除等操作，这些都是对数据进行访存操作. 图在更新的过程中，数据的大小和存放位置可能随之变化，导致现有的静态图的存储数据结构难以高效地存储动态图数据，这对动态图存储数据结构带来了巨大挑战. 国内也出现了对动态图更新优化的一些工作，主要有 RisGraph[118]，LiveGraph[119]，GraSU[120].

哈希表是动态图更新存储格式中最常用的一种基础结构，它支持快速的索引操作，但是哈希表的创建、哈希冲突等存在大量的开销. 为了减少这些开销，RisGraph[118]提出了一种基于邻接表的混合数据存储格式，称为索引邻接表（indexed adjacency lists）. 在索引邻接表中，每个顶点的边使用一个连续的动态数组存储. 当 1 个顶点的边的个数超过一定阈值时，RisGraph 会为顶点的边建立索引，以提高边查找的效率. RisGraph 使用哈希表作为默认索引，因为哈希表数据结构的更新时间复杂度平均为 $O(1)$.

RisGraph 关注如何设计数据存储格式以提高动态图中边查找和更新的效率，没有事务和多版本的支持. LiveGraph[119]是一种支持事务的多版本图存储系统，它基于邻接表数据结构，并且能够确保邻接表扫描是纯顺序的，没有随机访存. 这种纯顺序操作是通过将新颖的图感知数据结构事务边日志（transactional edge log, TEL）与利用 TEL 数据布局的并发控制机制相结合来实现的. TEL 不仅支持顺序邻接表扫描，而且同时支持快速边插入.

GraSU[120]提出了一种 FPGA 上的基于 PMA 的动态图数据存储格式，它利用空间相似性进行快速的动态图更新. 为了方便组织数据，GraSU 强制 PMA 的每个段只包含来自一个顶点的边. 此外，由于 FPGA 不能高效支持动态内存分配，GraSU 通过片外内存预分配的方式进行 PMA 的空间扩容.

在基于邻接表的数据存储格式中，基于块的链表虽然提高了空间局部性，但是遍历链表仍然需要随机访存. 数组避免了遍历 1 个顶点的邻居的随机访存，但是搜索一条边最坏的情况需要遍历 1 个顶点的所有邻居. 哈希的方法能快速定位 1 条边，但是哈希表的创建、哈希冲突以及哈希的计算都存在开销. 在混合数据结构存储 RisGraph 中，高度数顶点使用哈希索引，低度数顶点使用数组，在一定程度上减少了哈希的开销. 基于邻接表的数据存储格式虽然支持快速更新，但是由于数据存储不连续，图遍历的效率低.

2.2 面向高性能图计算应用的体系结构

2.2.1 面向单图任务处理的体系结构

近年来，国内研究人员在图计算硬件加速技术的研究领域中也有所突破，结合图处理算法的特征提出了多种基于 GPU/FPGA/ASIC 的图计算[121-127]加速方案.

近些年来，国内对于 GPU 图计算系统[121-124]做出不少的努力，DiGraph[121]提出了一种基于路径的多 GPU 加速方案，该方案将有向图表示为一组不相交的有向路径，并将路径作为基本的并行处理单元，使得顶点状态在 GPU 加速下沿路径进行高效传播，从而加快收敛速度. DiGraph 还包括一种依赖感知的路径调度策略，该策略根据路径依赖图的拓扑顺序对路径进行处理，能够有效减少图数据的冗余处理，有效加速图算法收敛.

为了在 GPU 上更快地进行迭代图形处理，Asyn-

Graph[122]提出了图结构感知异步处理方法和前后向路径处理机制，以此来最大限度地提高GPU上图处理的数据并行性. 图结构感知异步处理方法能够在GPU上有效地进行大多数顶点的并行状态传播，通过有效地处理重要图顶点之间的路径，获得更高的GPU利用率；前后向路径处理机制能够有效地异步处理每条路径上的图顶点，进而进一步提高沿路径进行的状态传播速度，同时确保较小的数据访问成本.

为了有效地支持out-of-GPU-memory下的图处理，LargeGraph[123]提出了一种依赖感知的数据驱动执行方法. 具体来说，该方法能够分析顶点之间的依赖关系，只对与活跃顶点的依赖链相关联的图数据进行加载和处理，以此来减少访存开销. 同时，通过在GPU上对路径子集进行动态识别、维护和处理，LargeGraph能够加速大多数活跃顶点的状态传播以更快达成收敛状态，其他的图数据则在CPU上进行处理.

为了利用GPU高效支持动态图中不同快照的处理，Zhang等人[124]提出了基于GPU的动态图处理系统EGraph，它能够集成到现有的GPU核外静态图处理系统中，使其能够高效利用GPU资源来有效地支持不同时序迭代图计算任务对动态图不同快照的并发处理. 与现有方法不同，EGraph提出了一种有效的加载—处理—切换执行模型. 其通过充分利用时序迭代图计算任务之间的数据访问相似性来有效降低CPU和GPU之间的数据传输开销，并保证更高的GPU利用率，从而实现时序迭代图计算任务的高效执行.

GraVF[125]是一种基于FPGA的分布式图计算框架，该框架采用了以顶点为中心的同步编程模型，能够在FPGA平台上高效灵活地实现各种分布式图算法. 此外，GraVF在图处理中引入了一种分离式apply-scatter内核，能够有效提高流水线性能并减少通信开销.

为支持FPGA上大规模图数据处理，ForeGraph[126]提出一种基于多FPGA架构的大规模图计算框架. 在ForeGraph中，每个FPGA板仅在片外内存中存储大规模图的一个分区，FPGA板间的通信开销被最小化，因此在处理大规模图数据上具有良好的可扩展性. ForeGraph提供的数据划分与通信方案能够有效减少数据冲突并保证数据局部性.

DepGraph[127]是一种依赖驱动的可编程加速器，通过与多核处理器的核心架构相结合，能够有效地加速图处理. DepGraph包含了一种有效的依赖驱动异步执行方法，通过沿依赖链预取图顶点来提高图顶点状态的传播速度，并获得更好的数据局部性. 此外，DepGraph在运行时将路径依赖关系转换为直接依赖关系，能够有效提高并行度，进一步加速图顶点状态传播. 为了解决流图处理中由于受图更新影响的图顶点的最新状态值沿着图拓扑不规则地传递而导致的严重冗余图计算和不规则内存访问问题.

图挖掘应用程序面临对顶点和边的大量随机内存访问. 硬件加速器提供适用于加速图挖掘操作的快速片上存储器，例如BRAM. 现有的工作[128-130]主要集中在缓存层次设计，以提高加速器上图挖掘的执行效率. 这些图挖掘硬件加速器主要是采用多级流水和定制的缓存架构来加速图挖掘的访存效率.

首个GNN加速器是由国内学者提出的，这也代表着国内学者在GNN加速器这个无人区领域的研究成果得到了业界认可. 正是由于第1个GNN加速器工作的提出，促使国内高校和工业界设计和开发更多更高效、更低能耗的GNN加速器[131-133].

DeepBurning-GL[133]是基于PFGA的GNN硬件加速器自动化生成框架，它与最先进的图形学习框架(例如Deep Graph Library)兼容，以便开发人员可以轻松地从软件描述的模型中生成特定于应用程序的GNN加速器. 首先，DeepBurning-GL使用GNN性能分析器来定位特定GNN应用的性能瓶颈，并确定满足用户指定约束的主要设计架构和参数. 其次，DeepBurning-GL提供一系列预构建的计算模板、内存模板等设计模板，可以对其进行参数化和融合，生成最终的加速器设计. DeepBurning-GL还包括1个优化器，通过调整加速器架构参数进行自动优化. 在评估中，DeepBurning-GL可以在3个不同的FPGA平台上为各种GNN模型和工作负载生成定制加速器.

2.2.2 面向流图和多图处理的加速器

现实世界的图通常随着时间动态改变，该类型的图被称为流图. 多图处理(并发图)指的是大量图计算任务(相同/不同图算法)同时处理同一个图的同一个或不同快照，来挖掘各自需要的信息. 现有的流图和多图处理加速器的相关研究主要是由国内团队主导研发的.

为有效支持流图处理，大量流图处理系统已经被提出. 然而，在对流图的每个图快照进行处理时，受图更新影响的图顶点的最新状态值会沿着图拓扑不规则地传递，从而导致现有方法面临严重的冗余图计算和不规则内存访问的问题. 为此，华中科技大

学提出了拓扑驱动的流图处理加速器 TDGraph[134]，通过拓扑驱动的流图处理模式从本质上高效地解决此问题. TDGraph 能够有效规则化流图处理中受影响图顶点的状态传递，并提高数据访问局部性，通过有效减少冗余计算和数据访问开销，以及提高缓存和内存带宽的利用率，提高流图计算在加速器上的计算效率.

在并发图执行过程中，并发图计算任务往往沿着不同路径独立地遍历和处理相同图数据，以此来传播各任务的图顶点状态，因此现有方法在支持并发图计算任务的执行时面临数据访问行为不规则、冗余访存开销大以及通信效率低等问题，导致现有图计算系统和现有体系结构在支持并发图计算任务执行时吞吐率低. 为此，Zhao 等人[135] 提出面向并发图任务处理加速器 LCCG，即一种以位置为中心的加速器来增强现有多核处理器，该加速器由遍历规则化单元以及预取索引单元构成，实现了一种拓扑感知执行方法，能够通过规则化图遍历以及共享图数据的存储与访问来降低并发图处理作业的数据访问成本，获得更高的核心利用率.

2.3 面向高性能图计算的系统软件

2.3.1 面向单图任务处理的系统软件

为充分发挥单台机器处理图计算任务的能力，国内学者从编程模型、执行模型、任务调度和图划分策略等方面采取了相应的优化手段，提出了许多单机内存图计算系统[136-138] 和单机核外图计算系统[139-143].

由于现实世界中图的幂律特性，图的绝大多数顶点连接相对较少的边，而小部分顶点（称作核心图顶点）连接大量的边. 处于这些核心图顶点之间的路径成为绝大多数图顶点之间进行状态推送的必经之路，这使得核心图顶点对整个图算法的收敛起到更大的作用并且其收敛需要经过更多次数更新. 由于缺乏对核心图顶点特征的感知，传统图计算系统常将核心图顶点及其之间的路径切分到多个图划分块中，这使得大量图顶点之间的状态推送都需要经过多个图划分块，导致大量图划分块被加载入内存，并且这些图划分块中大部分图顶点可能已经收敛. 这让系统在执行图分析任务时面临严重的数据访问开销，并且浪费了大量云平台资源来加载已经收敛的图顶点. 为解决这个问题，HotGraph[136] 提出了一种关联性感知的图数据处理顺序以及任务执行调度策略. 该策略首先基于核心图顶点之间的结构特征来构建核心图和进行图划分，然后通过核心图数据的优先访问调度来有效利用核心图顶点之间的路径，从而减少绝大部分图顶点之间状态推送所需跨越的图划分块数量，同时提高每次加载的图划分块的利用率，使得各图分析任务对于核心图顶点数据的访问开销更少. 此外，该策略通过构建各图分析任务在每轮迭代所需处理的图划分块之间的依赖关系图来调度数据访问顺序，以充分挖掘各图分析任务之间的数据访问空间关联性和时间相似性，优化多个图分析任务的数据访问.

大规模单图任务处理面临局部性差和存储效率低 2 个挑战. 现有的图计算系统主要基于以顶点为中心或以边为中心的计算模型，以及基于随机哈希的图的顶点或边划分，导致访问局部性差和计算负载不平衡. PathGraph[139] 通过使用一组基于树的分区来建模一个大型图，从而改进迭代计算算法在大图上的内存和磁盘访问局部性. 这使我们能够使用路径为中心的计算，而不是顶点为中心或边为中心的计算. 对于每个树分区，PathGraph 使用 DFS 重新标记顶点，以保持顶点 ID 顺序和路径中的顶点顺序的一致性. 并且，PathGraph 实现了一种优化了大规模迭代图并行计算的压缩存储. 具体来说，PathGraph 使用增量压缩和以 DFS 顺序存储基于树的分区. 通过将高度相关的路径聚类为基于树的分区，最大化存储介质的顺序访问和最小化随机访问. PathGraph 在分区树级并行迭代计算，并对每个分区树中的顶点进行连续局部更新，以提高收敛速度. 为了在树分区级别上在并行线程之间提供良好的工作负载平衡，引入了基于任务队列的多窃取点的概念，以允许从任务队列中的多个点窃取工作.

为提高分布式平台上的图计算性能，清华大学、北京大学、华中科技大学、上海交通大学等国内研究团队在分布式图计算领域开展了大量研发工作，提出了多种高性能分布式内存图计算系统[144-148].

Powerlyra[145] 是基于 NUMA aware 的多核图分析系统，根据拓扑数据、应用程序定义的数据和图系统的可变运行时状态，它们的访问模式进行不同的分配和放置，以减少远程通信访问. 而对于剩余的一些随机访问，Polymer 通过在 NUMA 节点上使用轻量级的顶点复制，小心地将随机远程访问转换为顺序远程访问. 为了提高负载平衡和顶点收敛性，进一步构建了聚合物，采用分层障碍增强并行性和局部性，面向边的倾斜图平衡划分，并根据活动顶点的比例自适应数据结构.

传统的分布式单图任务处理系统主要关注优化节点间通信和负载平衡的可伸缩性. 然而，与共享内

存图形计算框架相比，它们的总体处理效率往往不令人满意. 其中，实现高可扩展性的开销成为了分布式效率的主要限制因素，特别是在现代多核处理器和高速互连网络中. 为解决这个问题，Gemini[146] 采用以计算为中心，针对计算性能进行了多种优化以保证高效率的同时拥有高可扩展性. 其主要采用了基于块的分区方案，可以实现低开销的扩展设计和位置保持的顶点访问. Gemini 还采用了基于稀疏密集计算的数据抽象，将混合计算从共享内存拓展至分布式场景中. 针对机器节点的工作负载不均衡问题，采取位置感知的分块和细粒度的工作窃取来改善节点间和节点间的负载均衡.

GNN 训练和推理加速是一个极具挑战性的问题，因此国内也有诸多研究学者开发单机 GNN 加速系统[149-153]，主流的工作主要有 NeuGraph[149]，Seastar[150]，DA-SpMM[153] 等.

北京大学和微软研究院开发了第 1 个在 GPU 中并行处理 GNN 的专门框架，即 NeuGraph[149]. NeuGraph 构建在 Tensorflow 深度学习框架之上，目前还未开源. 该框架实现了一个编程模型 SAGA-NN，基于函数 Scatter 用于边聚合，ApplyEdge 用于边组合，Gather 用于顶点聚合，ApplyVertex 用于节点组合. 在同名函数中使用了分散-聚集内核，而在矩阵组合函数中使用了乘法原语. NeuGraph 还具有许多优化功能以加速 GNN 计算. 首先，通过 Kernighan-Lin 算法对大型图进行分区，以使分区更密集并最小化分区之间的传输，这会造成性能损失. 其次，通过将可以一起计算的小稀疏分区批处理在一起，以及对第 1 层 GNN 中的传输和计算时间进行分析，以完美的流水线处理不同的块，从而优化了对 GPU 的分区调度. 再次，NeuGraph 还通过将多条边融合在一起来消除冗余计算. 最后，NeuGraph 允许通过分布计算将 GNN 扩展到多个 GPU，并通过使用基于环的数据流来优化信息传输，从而最大限度地减少互连处的争用. NeuGraph 的全新编程模型虽然可以很好地优化 GNN 执行过程中的图操作，但它并不能支持大规模图 GNN 训练和推理.

Seastar[150] 与现有 GNN 训练框架的区别主要在于：Seastar 是以顶点为中心的编程模型（为了更好的可用性）和执行计划生成（为了更高的性能）. Seastar 的编程模型允许用户使用以顶点为中心的用户定义函数（UDF）轻松地编程 GNN 模型，并在 GNN 训练的计算图中识别出丰富的算子融合机会. Seastar 采用自适应特征组、以位置为中心的执行和动态负载平衡等新颖设计，为 GNN 训练过程中的前向传播和反向传播生成高性能的融合内核.

DA-SpMM[153] 是由清华大学电路与系统研究所汪玉团队所提出，相比单机 GNN 框架，他们从新的自动调整的角度考虑输入动态，而 3 点问题仍有待解决：1）考虑稀疏性的正交设计原则. 应提取此类稀疏问题的正交设计原则以形成不同的算法，并进一步用于性能调整. 2）算法空间中的非平凡实现. 结合正交设计原则来创建新算法需要应对线程竞争处理等新挑战. 3）对输入动态的启发式适应性. 需要启发式适应性来动态优化输入动态的代码. 为解决这 3 个挑战，汪玉团队提出了一种用于 SpMM 的数据感知启发式 GPU 内核 DA-SpMM，DA-SpMM 不仅涵盖了以前的 SpMM 设计，而且还提出了以前研究中没有的新设计，并采用条件归约等技术来实现先前研究中缺少的算法，DA-SpMM 还能在考虑输入动态的情况下自适应地优化代码.

国内分布式图神经网络系统[154-156] 主要是由工业界互联网公司开发，主要负责开发内部图神经网络相关业务，例如推荐系统、社交网络分析和金融分析. 为了优化多 GPU 之间的数据通信，分布式图通信库 DGCL[155] 由中国香港中文大学团队提出，主要用于在多个 GPU 上进行高效的 GNN 训练. DGCL 的核心是为 GNN 训练量身定制的通信规划算法，DGCL 共同考虑了充分利用快速链路、融合通信、避免竞争和平衡不同链路上的负载，可以很容易地采用 DGCL 将现有的单 GPU 的 GNN 系统扩展到分布式训练.

2.3.2 面向流图和多图任务处理的系统软件

尽管现有图计算系统为了实现高性能，主要通过加速状态传播和改善数据局部性，以确保负载平衡和减少内存消耗等方法来支持分布式环境上的大规模图分析，但在分布式平台上有效执行多图任务处理仍面临重大挑战. 当现有分布式系统在同一底层图上运行大量多图任务分析时，每个图任务分析会分别沿不同的图路径访问共享图，从而导致不同作业在不同时间从本地节点或远程节点的主存重复加载相同的数据到缓存中. 由于缓存干扰、存储器/网络带宽未充分利用等因素，造成了昂贵的数据访问开销. 数据访问成本与图算法计算开销的高比率使得当前的分布式图处理系统存在低吞吐量问题. 为解决这一问题，CGraph[137-138] 提出了一种关联性感知的并发图处理机制来提高分布式平台上多图任务处理的吞吐量，主要通过充分利用数据访问的相关性来有效支持分布式环境上的多图任务分析，以获

得系统的高吞吐量.

首先, CGraph 使用以数据为中心的模型将图结构数据从与每个作业相关的顶点状态中分离, 然后在每次迭代中将多个多图任务分析可共享的图结构分区加载进缓存, 使相关的多图任务分析以一种新颖的同步方式进行处理, 这样多图任务分析能够在高速缓存/主存中共享图结构数据的单个副本, 使得数据访问和存储开销由多个多图任务分析共同分摊. 其次, 采用一种新颖的通信方案来降低多图任务分析的通信成本, 根据多图任务分析的通信分布特征, 使作业通信以更规则的方式批量化进行. 然后, 基于连续迭代的分区之间的高负载分布相似性, 对分布式平台上的多图任务分析采用有效的增量负载平衡策略. 同时还采用了一种图重分区方案, 以确保在演化图上执行多图任务分析时数据的局部性.

随着实际应用中对图分析的需求快速增加, 同一底层图上往往并发地运行着大量的图计算任务. 然而, 现有图计算框架的存储系统主要为有效地服务单个任务而设计, 忽视了并发任务之间冗余的数据存储和访问开销, 从而导致现有图计算框架在执行并发任务时往往效率低下. GraphM[142] 能够方便地嵌入到现有的图计算系统中并充分利用并发图计算任务的数据访问相似性, 使得图结构数据能够规则地流入到内存/缓存中并被并发图计算任务共享, 通过有效地降低数据访问和存储开销来提高并发任务的吞吐量. 并且, GraphM 可以轻松地嵌入到与现有的图计算框架中并充分利用并发图计算任务之间的数据访问相似性, 通过有效地降低数据访问和存储开销来提高现有图计算框架在处理并发任务时的吞吐量, 同时减少用户的编程负担. GraphM 通过 Share-Synchronize 机制来挖掘并发运行任务之间的数据访问相似性.

针对面向大规模动态图的增量计算, Ingress[148] 能够将以顶点为中心的图算法自动实现增量化, 并且配备了 4 类不同的计算状态记忆策略, 给出了每类策略适用图算法的充分条件, 系统能够根据用户指定的算法逻辑自动选择最优记忆策略.

3 国内外研究进展比较

3.1 面向图计算类应用的体系结构和系统

国外研究团队对于迭代图处理方向的理论研究与系统研发早于国内, 但随着国内互联网行业与技术的迅猛发展, 国内图计算市场需求日益高涨, 国内研究人员也在迭代图处理方向进行了大量科研工作并获得了许多突破性成果, 国内迭代图处理的理论研究与系统研发已达到世界前列水平. 在单机图计算系统方向, 国外学者提出了首个基于内存的单机图计算系统与首个基于磁盘的单机图计算系统, 并提出了以顶点为中心和以边为中心的图计算编程模型, 国内研究团队则在编程模型、执行模型、任务调度和图数据划分策略等方面进行了优化与创新, 提出了多种高性能单机图计算系统. 在分布式图计算系统方向, 国外研究人员提出了批量同步并行模型、异步执行模型、Gather-Apply-Scatter 计算模型等多种分布式图计算模型, 并设计了多种分布式内存图计算系统, 同时还将单机核外图计算系统扩展成为分布式核外图计算系统, 国内学者则主要在分布式内存图计算系统上投入了大量研发工作, 提出了多种分布式内存图计算模型, 例如同步/异步混合计算模型、以计算为中心的处理模型、有界异步迭代处理模型. 对于图计算加速器领域的研究, 国内外学者在基于 GPU/FPGA/ASIC 的图计算硬件加速技术上都有所突破, 提出了多种 GPU 上的高性能异构图计算框架以及具有高扩展性的多 FPGA 图计算架构, 同时还设计了多款图计算专用的可编程加速器, 有效地提高了图计算性能.

3.2 面向图挖掘类应用的体系结构和系统

对于图匹配软件系统的研究, 国外起步早于国内. 但是国内学者及时地跟进, 在很多方面有新的研究成果并有所突破. 在编程模型方面, 虽然模式不感知和模式感知编程模型均由国外学者提出, 但是国内学者对这 2 个编程模型做了进一步的优化和创新. 例如, 国内研究团队减少了之前模式不感知系统中存在的同步问题, 选取模式感知模型中的最优匹配顺序和对称顺序. 在图遍历策略方面, 国外学者提出了多种 BFS-DFS 混合遍历策略, 不仅减少内存使用而且保持了并行性. 在多模式匹配方面, 国内外很多机构都对此方面进行研究并有所突破. 在此方面, 国内学者已经走到了国际前沿, 华中科技大学提出的模式抽象方法可以完全消除多模式匹配中的冗余问题, 极大地提升了多模式图匹配的效率. 对图匹配硬件加速器的研究, 国内已经走在世界学术前沿. 国外学者主要关注基于 ASIC 和 PIM 加速器的研究, 国内学者对基于 FPGA, ASIC, PIM 加速器均有研究. 在基于 FPGA 加速器方面, 国内学者主要关注于 FPGA 片上缓存的利用和流水线并行性. 在基于 ASIC 加速器方面, 国外研究团队主要关注于硬件定制计算单元

和通用性设计,而国内研究团队更关注于高并行性的设计,探索了多种细粒度并行.在基于 PIM 加速器方面,国内外研究团队使用 PIM 架构减少图挖掘处理应用中的大量访存开销.

3.3 面向图学习类应用的体系结构和系统

国外主流图神经网络加速系统,都是将现有的深度学习框架作为 GNN 训练的后端,并在其基础加上 GNN 训练专属图操作模块,以实现 GNN 高效训练和推理,例如 DGL, PyG, RoC, Euler.这些系统同样也提供了高效灵活的 API 接口,方便用户自定义构建和训练 GNN 模型.但是这样做会存在一些局限性,主要体现在 GNN 执行过程中的图操作.而国内北京大学开发的 NeuGraph 提出了一种全新的 GNN 训练架构,将图操作和数据流模型结合起来以支持 GNN 高效训练,这种方式既弥补了传统深度学习的数据流架构不能有效支持 GNN 执行过程中的图操作,又能够很好地解决了图计算过程中不能有效支持数据流编程模型等特点.对于分布式 GNN 系统,国内和国外的研究重点也是大相径庭.国外分布式 GNN 系统都是打破现有的 GNN 训练模式,从而寻找一种对稀疏数据运算更加友好的分布式训练策略.比如 OSDI 的 P3 工作,相对于传统的数据向数据进行移动,采用计算向数据移动的全新训练方式来减少分布式 GNN 训练过程中的数据通信量和时间.而基于 CPU 的分布式 GNN 系统 Dorylus 也是如此,采用全异步 GNN 训练方式,不仅是将每一层梯度更新和同步进行异步流水线并行,而且对于不同层之间的梯度更新都是全异步进行,大大减少了梯度同步时间和网络通信量.而国内的分布式 GNN 系统则是遵循传统的 GNN 训练模式优化影响其训练时间的主要瓶颈,比如优化数据局部性、缓存高度顶点、采用流水线方式尽可能覆盖网络通信时间.同时, NeutronStar 通过混合依赖管理实现了一种新的并行执行模型.在 GNN 加速器方面,虽然国内中国科学院高性能团队提出了 GNN 加速器,但是后续研究力度明显不足,并且还是仅局限于如何在现有的 GNN 推理模式下进行优化.而国外 GNN 加速器研究明显具有更高的开放性,例如 I-GCN 提出将图结构划分为多个子社区,每个子社区之间通过高度顶点链接,以保证每个子社区的局部性良好,从而加快 GNN 推理过程.像 GCoD 提出新型 GNN 训练方式,通过分而治之的方式将图拓扑结构进行删减以保证局部性良好,提高缓存命中率.通过这种方式,在既不丢失 GNN 模型精度的前提下,还能将图拓扑结构划分为稀疏部分和密集部分,并分别计算处理.相比之下,国内在 GNN 加速器领域还需要更深和多维研究视角才能做出更创新的工作.

3.4 面向多图任务和流图类应用的体系结构和系统

在目前图计算系统中,各图分析任务在访问共享数据时相互独立、互不感知,面临严重的内存墙和性能干扰问题.因此很有必要研究多图任务处理优化系统和加速器.目前面向多图任务分析的体系结构和系统都是由国内主导研究的,像华中科技大学提出的面向多图任务处理的系统 GraphM 和首个面向多图任务处理的加速器 LCCG,这也说明了国内对于多图任务处理的工作是处于国际领先水平.而对于流图,国外的主要研究集中于流图系统,像 KickStarter, GraphBolt, DZiG 等,主要是考虑使用增量计算来加速流图处理的同时保证结果的正确性.而国外对于流图加速器的研究较少,即 JetStream,其在静态图增量加速器 GraphPluse 的基础上拓展使其高效支持动态图增量处理.国内的主要研究为东北大学的 Ingress 和华中科技大学的 TDGraph,与国外研究的区别在于, TDGraph 采用拓扑驱动的动态图增量机制规则化动态图处理中的顶点状态传递,从而降低冗余计算和数据访问成本.这些工作不局限于设计更高效的增量计算技术,而是从拓扑结构中发现可优化空间.但无论是国内还是国外,对于流图的研究都处于初级阶段.如何设计更高效的动态图更新存储结构、动态图划分策略和处理模式依然是一个具有高挑战性的难题.

4 发展趋势与展望

图计算是大数据时代的重要数据处理工具,与我国民生密切相关,可普遍应用于生命健康、芯片设计、经济建设、国防安全、社会治理、科学发现等重要领域.因此,需要构筑面向图计算的软/硬协同一体化的计算机技术生态链,促进国家信息化行动的深入推进,增强各行业各领域大型复杂科学研究的创新能力,为社会经济和科技教育注入新动力.现如今,各大高校、科研机构和商业互联网络公司都已经意识到图计算的重要战略意义,纷纷投入精力加速对图计算体系结构和系统软件关键技术的研究与应用.同时,图计算技术发展至今虽然已经有 10 余年,但随着新型图计算应用不断涌现、图属性逐渐丰富等特征的出现,仍有大量问题需要被研究.可以看到,在未来的一段时间内,对图挖掘、图学习类图算法的

高效支持是图计算领域内的研究热点. 在此，我们对图挖掘、图学习等图计算应用的未来发展趋势进行简要介绍.

1)动态图计算加速机制. 随着数据规模的不断扩大和图结构的不断变化，动态图计算已经成为了一个重要的研究方向. 与静态图计算相比，动态图计算更具有挑战性，因为它需要处理图的实时更新，同时还要确保计算的准确性和效率. 在动态图算法方面，我们需要设计新的动态图算法，以支持图的动态变化和快速更新. 这可能需要我们引入时间因素，使图算法能够捕捉到图的历史变化信息. 同时，我们也需要考虑如何有效地存储和管理动态图，以减少数据更新的开销. 在计算模型方面，我们需要研发新的图计算执行模型，以适应动态图的特性，特别是能够利用之前快照的计算来避免重算. 在系统软件方面，我们需要考虑如何优化动态图计算的性能. 这可能涉及到动态任务调度、动态数据分布和动态负载均衡等技术. 这些技术可以根据图的实时变化和系统的运行状态，自动调整计算的执行策略，从而提高系统的运行效率. 此外，为动态图计算设计专门的加速器，也是一种全新的解决方案. 例如，基于 ReRAM 的动态图挖掘加速器等，基于 FPGA 的动态图神经网络加速器，通过定制化来解决动态图计算中的一些困难问题. 总的来说，高效的动态图计算技术对于处理大规模、实时的图数据具有重要的意义，但也面临着很多挑战，需要进行深入的研究和探索.

2)数据访问模式感知的图数据存储和访问机制. 在现有图计算平台上执行的多图分析任务，往往独立地加载图结构数据和它们的私有图顶点状态数据进行处理. 这使得它们共享的图结构数据被反复加载到内存和 cache 中，并在内存和 cache 中维护多个副本，从而产生大量冗余的数据访问开销和存储开销，致使内存墙等问题. 因此，很有必要提供一种数据访问模式感知的图数据存储和访问机制，能够透明地使现有图计算系统上运行的大量图分析任务有效感知它们之间存在的这些数据访问行为相似性，以减少冗余图数据的存储开销和为共享图数据访问提供条件.

3)多模态的图计算技术. 随着现代社会对大数据的依赖越来越深，数据的来源也变得多元化，包括文本、图像、社交网络等多种类型，这些数据在各自的维度上都有独特的语义和信息. 因此，有效地利用这些多模态数据，挖掘其中的深层次关系，成为了图计算的一个重要任务. 在这个背景下，多模态的图计算技术逐渐引起了研究者的关注，这需要我们在图计算算法、计算算法和系统软件等方面进行全新的设计和优化. 在图模型方面，我们需要考虑如何将多模态数据的特性和语义信息嵌入到图模型中. 一个可能的方向是发展新的图嵌入技术，以支持多模态数据的表征和交互. 例如，我们可以设计多层的图计算算法，每一层对应一种模态的数据，通过特定的连接和操作，实现不同模态数据间的互动和融合. 在计算模型方面，我们需要考虑如何有效地处理多模态数据的异质性和复杂性. 一方面，我们需要设计新的多模态图计算计算模型，以适应多模态数据的特性；另一方面，我们需要利用深度学习等机器学习技术，自动学习和理解多模态数据间的复杂关系. 在系统软件方面，我们需要考虑如何优化多模态图计算的性能和效率. 这需要我们开发新的数据存储和管理技术，以支持多模态数据的高效存储和快速访问；同时，我们需要设计新的任务调度和负载均衡策略，以提高多模态图计算的并行度和效率. 另外，研发适用于多模态图计算的硬件加速器也是未来的研究热点. 通过定制化的多模态图计算加速器来挖掘多模态数据之间的复杂关联关系，从而获得多模态数据间更加丰富的潜在价值.

5 结束语

万物皆关联. 图计算作为分析事物之间关联关系的重要工具，是人机物三元融合的万物智能互联时代的支撑技术，目前已广泛应用于社会治理、医疗健康、电网分析、金融安全、网络安全、电路检测、电子商务、军事信息化以及包括天文、制药、材料、育种、基因在内的科学发现等领域，被国际权威 IT 研究与顾问咨询公司高德纳列为 2022 年最具影响力的 5 项新兴技术之一. 图计算已成为学术界竞相发展的前沿热点，是各国政府和企业争夺的关键技术.

本文围绕图计算体系结构和系统软件关键技术的研究进展与趋势展开系统性论述，内容包括：图存储与动态图更新. 具体为：面向高性能图计算的体系结构和系统软件，并给出了国内与国际的当前研究现状，还对国内外研究进展进行了比较，最后对这些关键技术的发展趋势进行了展望. 需要注意的是，随

着大数据和人工智能的蓬勃发展,数据之间的关联关系变化速度日益加快,数据自身及其关联关系的附属信息也日益丰富.为了从这些数据中获取有用信息,新型图计算任务(例如新型图神经网络、流图计算、超图计算)不断涌现,图计算需求日益复杂多样.复杂多样化的图计算需求给现有图计算体系结构带来了巨大挑战,如何设计新型高能效图计算体系结构满足复杂多样的图计算需求是一个亟待解决的难题.

作者贡献声明:张宇和金海确定了综述论文的主体框架;张宇和廖小飞确定了论文框架的具体内容;张宇撰写了图计算基础定义、基本理论和关键技术,并对图计算体系结构与系统软件的发展进行了总结与展望;姜新宇调研了大量图计算相关的论文,并撰写了国内外发展研究现状;余辉与齐豪负责撰写图计算的数据存储格式和系统软件发展脉络,并详细阐述一些典型的图计算系统的设计与优缺点;赵进负责撰写图计算体系结构的发展脉络,并对一些经典的图计算加速器进行详细介绍;王彪和余婷负责论文参考文献的整理和论文美观排版.

参 考 文 献

[1] Cheng R, Hong Ji, Kyrola A, et al. Kineograph: Taking the pulse of a fast-changing and connected world[C]//Proc of the 7th ACM European Conf on Computer Systems. New York: ACM, 2012: 85−98

[2] Zhao Jin, Jiang Xinyu, Zhang Yu, et al. An efficient storage system for high concurrency graph analysis tasks[J]. SCIENTIA SINCIA Informationis, 2022, 52(1): 111−128 (in Chinese)
(赵进, 姜新宇, 张宇, 等. 一种高效的面向高并发图分析任务的存储系统[J]. 中国科学: 信息科学, 2022, 52(1): 111−128)

[3] Qiu Xiafei, Cen Wubin, Qian Zhengping, et al. Real-time constrained cycle detection in large dynamic graphs[J]. Proceedings of the VLDB Endowment, 2018, 11(12): 1876−1888

[4] Liao Xiaofei, Chen Yicheng, Zhang Yu, et al. An efficient incremental strongly connected component algorithm for dynamic directed graphs[J]. SCIENTIA SINCIA Informationis, 2019, 49(8): 988−1004 (in Chinese)
(廖小飞, 陈意诚, 张宇, 等. 一种高效的面向动态有向图的增量强连通分量算法[J]. 中国科学: 信息科学, 2019, 49(8): 988−1004)

[5] Eswaran D, Faloutsos C, Guha S, et al. Spotlight: Detecting anomalies in streaming graphs[C]//Proc of the 24th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining. New York: ACM, 2018: 1378−1386

[6] Eksombatchai C, Jindal P, Liu J, et al. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time[C]//Proc of the 17th World Wide Web Conf. New York: ACM, 2018: 1775−1784

[7] Faust K. A puzzle concerning triads in social networks: Graph constraints and the triad census[J]. Social Networks, 2010, 32(3): 221−233

[8] Granovetter M. The strength of weak ties: A network theory revisited[J]. Sociological Theory, 1983, 2(1): 201−233

[9] Zhao Jin, Zhang Yu, He Ligang, et al. GraphTune: An efficient dependency-aware substrate to alleviate irregularity in concurrent graph processing[J]. ACM Transactions on Architecture and Code Optimization, 2023, 20(3): 37:1−37:24

[10] Milo R, Shen-Orr S, Itzkovitz S, et al. Network motifs: Simple building blocks of complex networks[J]. Science, 2002, 298(5594): 824−827

[11] Alon N, Dao P, Hajirasouliha I, et al. Biomolecular network motif counting and discovery by color coding[J]. Bioinformatics, 2008, 24(13): i241−i249

[12] Cho Y R, Zhang Aidong. Predicting protein function by frequent functional association pattern mining in protein interaction networks[J]. IEEE Transactions on Information Technology in Biomedicine, 2009, 14(1): 30−36

[13] Milenković T, Ng W L, Hayes W, et al. Optimal network alignment with graphlet degree vectors[J]. Cancer Informatics, 2010, 9(9): 121−137

[14] Milenković T, Pržulj N. Uncovering biological network function via graphlet degree signatures[J]. Cancer Informatics, 2008, 6: 257−273

[15] Deshpande M, Kuramochi M, Wale N, et al. Frequent substructure-based approaches for classifying chemical compounds[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(8): 1036−1050

[16] Gaüzere B, Brun L, Villemin D. Two new graphs kernels in chemoinformatics[J]. Pattern Recognition Letters, 2012, 33(15): 2038−2047

[17] Kashima H, Saigo H, Hattori M, et al. Graph kernels for chemoinformatics[M/OL]//Chemoinformatics and Advanced Machine Learning Perspectives: Complex Computational Methods and Collaborative techniques. 2011[2022-06-14].https://doi.org/10.48550/arXiv.2208.04929

[18] Chen Hanhua, Jin Hhai, Cui Xiaolong. Hybrid followee recommendation in microblogging systems[J]. Science China Information Sciences, 2017, 60(1): 1−14

[19] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series[G]//LNCS 3361: The Handbook of Brain Theory and Neural Networks. Berlin: Springer, 1998: 255−258

[20] Grnarova P, Levy K Y, Lucchi A, et al. A domain agnostic measure for monitoring and evaluating GANs[J]. Advances in Neural Information Processing Systems, 2019, 3(24): 32−44

[21] Wu Zonghan, Pan Shirui, Chen Fengwen, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(1): 4−24

[22] Sperduti A, Starita A. Supervised neural networks for the classification of structures[J]. IEEE Transactions on Neural Networks, 1997, 8(3): 714−735

[23] Welling M, Kipf T N. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint, arXiv: 1609. 02907, 2016

[24] Fout A, Byrd J, Shariat B, et al. Protein interface prediction using graph convolutional networks[J]. Advances in Neural Information Processing Systems, 2017, 2(8): 12−30

[25] Ying R, He Ruining, Chen Kaifeng, et al. Graph convolutional neural networks for web-scale recommender systems[C]//Proc of the 24th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2018: 974−983

[26] Ying Z, You Jiaxuan, Morris C, et al. Hierarchical graph representation learning with differentiable pooling[J/OL]. Advances in Neural Information Processing Systems, 2018, 19−31.[2022-10-03].https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html

[27] Dai Hanjun, Kozareva Z, Dai Bo, et al. Learning steady-states of iterative algorithms over graphs[C/OL]//Proc of the 35th Int Conf on Machine Learning. 2018: 1106−1114. [2022-07-03].https://proceedings.mlr.press/v80/dai18a.html

[28] Ediger D, McColl R, Riedy J, et al. STINGER: High performance data structure for streaming graphs[C/OL]// Proc of the 9th IEEE Conf on High Performance Extreme Computing. Piscataway, NJ: IEEE, 2012[2022-10-14]. https://doi.org/10.1109/HPEC.2012.6408680

[29] Winter M, Zayer R, Steinberger M. Autonomous, independent management of dynamic graphs on GPUs[C/OL]// Proc of the 14th IEEE High Performance Extreme Computing Conf. Piscataway, NJ: IEEE, 2017[2022-10-14].https://doi.org/10.1109/HPEC.2017.8091058

[30] Awad M A, Ashkiani S, Porumbescu S D, et al. Dynamic graphs on the GPU[C]// Proc of the 17th IEEE Int Parallel and Distributed Processing Symp. Piscataway, NJ: IEEE, 2020: 739−748

[31] Ashkiani S, Farach-Colton M, Owens J D. A dynamic Hash table for the GPU[C]// Proc of the 15th IEEE Int Parallel and Distributed Processing Symp. Piscataway, NJ: IEEE, 2018: 419−429

[32] Bender M A, Hu Haodong. An adaptive packed-memory array[J]. ACM Transactions on Database Systems, 2007, 32(4): 26−37

[33] Wheatman B, Xu H. Packed compressed sparse row: A dynamic graph representation[C/OL]// Proc of the 2018 IEEE High Performance Extreme Computing Conf. Piscataway, NJ: IEEE, 2018[2022-07-14].https://doi.org/10.1109/HPEC.2018.8547566

[34] Wheatman B, Xu H. A parallel packed memory array to store dynamic graphs[C/OL]// Proc of the 2021 Workshop on Algorithm Engineering and Experiments. 2021: 31−45. [2022-08-07].https://people.csail.mit.edu/hjxu/papers/ppcsr-alenex.pdf

[35] Sha Mo, Li Yuchen, He Bingsheng, et al. Technical report: Accelerating dynamic graph analytics on GPUs[J]. arXiv preprint, arXiv: 1709.05061, 2017

[36] Macko P, Marathe V J, Margo D W, et al. LLAMA: Efficient graph analytics using large multiversioned arrays[C]// Proc of the 31st IEEE Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2015: 363−374

[37] Dhulipala L, Blelloch G E, Shun J. Low-latency graph streaming using compressed purely-functional trees[C]//Proc of the 40th ACM SIGPLAN Conf on Programming Language Design and Implementation. New York: ACM, 2019: 918−934

[38] Iyer A P, Pu Qifan, Patel K, et al. TEGRA: Efficient ad-hoc analytics on evolving graphs[C]// Proc of the 18th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2021: 337−355

[39] De Leo D, Boncz P. Teseo and the analysis of structural dynamic graphs[J]. Proceedings of the VLDB Endowment, 2021, 14(6): 1053−1066

[40] Iyer A P, Pu Qifan, Patel K, et al. TEGRA: Efficient ad-hoc analytics on evolving graphs [C]//Proc of the 18th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2021: 337−355

[41] Dave A, Gonzalez J E, Franklin M J, et al. Persistent adaptive radix trees: Efficient fine-grained updates to immutable data[J/OL]. 2013, 12: 1−15. [2022-04-25].https://ankurdave.com/dl/part-tr.pdf

[42] Leis V, Kemper A, Neumann T. The adaptive radix tree: ARTful indexing for main-memory databases[C]// Proc of the 29th IEEE Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2013: 38−49

[43] Sengupta D, Sundaram N, Zhu Xia, et al. GraphIn: An online high performance incremental graph processing framework[C]// Proc of the European Conf on Parallel Processing. New York: ACM, 2016: 319−333

[44] Kumar P, Howie H. GraphOne: A data store for real-time analytics on evolving graphs[J]. ACM Transactions on Storage, 2020, 15(4): 1−40

[45] Iwabuchi K, Sallinen S, Pearce R, et al. Towards a distributed large-scale dynamic graph data store[C]// Proc of the 17th IEEE Int Parallel and Distributed Processing Symp Workshops. Piscataway, NJ: IEEE, 2016: 892−901

[46] Pandey P, Wheatman B, Xu H, et al. Terrace: A hierarchical graph container for skewed dynamic graphs[C]//Proc of the 2021 Int Conf on Management of Data. New York: ACM, 2021: 1372−1385

[47] Merrill D, Garland M, Grimshaw A. Scalable GPU graph traversal[J]. ACM SIGPLAN Notices, 2012, 47(8): 117−128

[48] Wang Yangzihao, Davidson A, Pan Yuechao, et al. Gunrock: A high-performance graph processing library on the GPU[C/OL]//Proc of the 21st ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2016[2022-02-17].https://doi.org/10.1145/3016078.2851145

[49] Han Wei, Mawhirter D, Wu Bo, et al. Graphie: Large-scale asynchronous graph traversals on just a GPU[C]// Proc of The 26th Int Conf on Parallel Architectures and Compilation Techniques. Piscataway, NJ: IEEE, 2017: 233−245

[50] Ben-Nun T, Sutton M, Pai S, et al. Groute: An asynchronous multi-GPU programming model for irregular computations[J]. ACM SIGPLAN Notices, 2017, 52(8): 235−248

[51] Sabet A H N, Qiu Junqiao, Zhao Zhijia. Tigr: Transforming irregular graphs for GPU-friendly graph processing[J]. ACM SIGPLAN Notices, 2018, 53(2): 622−636

[52] Sabet A H N, Zhao Zhijia, Gupta R. Subway: Minimizing data transfer during out-of-GPU-memory graph processing[C/OL]//Proc of the 15th European Conf on Computer Systems. New York: ACM, 2020[2022-07-13].https://doi.org/10.1145/3342195.3387537

[53] DeLorimier M, Kapre N, Mehta N, et al. GraphStep: A system architecture for sparse-graph algorithms[C]//Proc of the 14th Annual IEEE Symp on Field-Programmable Custom Computing Machines. Piscataway, NJ: IEEE, 2006: 143−151

[54] Attia O G, Johnson T , Townsend K , et al. CyGraph: A reconfigurable architecture for parallel breadth-First search[C/OL]//Proc of the 14th Parallel & Distributed Processing Symp Workshops. Piscataway, NJ: IEEE, 2014[2022-06-14].https://doi.org/10.1109/IPDPSW.2014.30

[55] Zhou Shijie, Kannan R, Prasanna V K, et al. HitGraph: High-throughput graph processing framework on FPGA[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(10): 2249−2264

[56] Ham T J, Wu Lisa, Sundaram N, et al. Graphicionado: A high-performance and energy-efficient accelerator for graph analytics[C/OL]//Proc of the 49th Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2016[2022-07-18].https://doi.org/10.1109/MICRO.2016.7783759

[57] Ozdal M M, Yesil S, Kim T, et al. Energy efficient architecture for graph analytics accelerators[J]. ACM SIGARCH Computer Architecture News, 2016, 44(3): 166−177

[58] Kalinsky O, Kimelfeld B, Etsion Y. The TrieJax architecture: Accelerating graph operations through relational joins[C]//Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 1217−1231

[59] Chen Xuhao, Huang Tianhao, Xu Shuotao, et al. FlexMiner: A pattern-aware accelerator for graph pattern mining[C]// Proc of the 48th Annual Int Symp on Computer Architecture. New York: ACM, 2021: 581−594

[60] Rao Gengyu, Chen Jingji, Yik J, et al. SparseCore: Stream ISA and processor specialization for sparse computation[C]//Proc of the 27th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2022: 186−199

[61] Besta M, Kanakagiri R, Kwasniewski G, et al. Sisa: Set-centric instruction set architecture for graph mining on processing-in-memory systems[C]// Proc of the 54th Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2021: 282−297

[62] Talati N, Ye Haojie, Yang Yichen, et al. NDMiner: Accelerating graph pattern mining using near data processing[C]//Proc of the 49th Annual Int Symp on Computer Architecture. New York: ACM, 2022: 146−159

[63] Zeng Hanqing, Prasanna V. GraphACT: Accelerating GCN training on CPU-FPGA heterogeneous platforms[C]//Proc of the 28th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2020: 255–265

[64] Geng Tong, Li Ang, Shi Runbin, et al. AWB-GCN: A graph convolutional network accelerator with runtime workload rebalancing[C]//Proc of the 53rd Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2020: 922−936

[65] Li Jiajun, Louri A, Karanth A, et al. GCNAX: A flexible and energy-efficient accelerator for graph convolutional neural networks[C]//Proc of the 27th IEEE Int Symp on High-Performance Computer Architecture. Piscataway, NJ: IEEE, 2021: 775−788

[66] Geng Tong, Wu Chunshu, Zhang Yongan, et al. I-GCN: A graph convolutional network accelerator with runtime locality enhancement through islandization[C]// Proc of the 54th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2021: 1051−1063

[67] You Haoran, Geng Tong, Zhang Yongan, et al. GCoD: Graph convolutional network acceleration via dedicated algorithm and accelerator co-design[C]// Proc of the 28th IEEE Int Symp on High-Performance Computer Architecture. Piscataway, NJ: IEEE, 2021, 460−474

[68] Auten A, Tomei M, Kumar R. Hardware acceleration of graph neural networks[C/OL]//Proc of the 57th ACM/IEEE Design Automation Conf. Piscataway, NJ: IEEE, 2020[2022-05-13].https://doi.org/10.48550/arXiv.2209.02916

[69] Zhang Bingyi, Kannan R, Prasanna V. BoostGCN: A framework for optimizing GCN inference on FPGA[C]// Proc of the 29th IEEE Annual Int Symp on Field-Programmable Custom Computing Machines. Piscataway, NJ: IEEE, 2021: 29−39

[70] Kang M, Hwang R, Lee J, et al. GROW: A row-stationary sparse-dense GEMM accelerator for memory-efficient graph convolutional neural networks[J]. arXiv preprint, arXiv: 2203.00158, 2022

[71] Rahman S, Afarin M, Abu-Ghazaleh N, et al. JetStream: Graph analytics on streaming data with event-driven hardware accelerator[C]// Proc of the 54th Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2021: 1091−1105

[72] Sundaram N, Satish N R, Patwary M M A, et al. GraphMat: High performance graph analytics made productive[J]. arXiv preprint, arXiv: 1503.07241, 2015

[73] Shun Julian, Blelloch G E. Ligra: A lightweight graph processing framework for shared memory[C]//Proc of the 18th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2013: 135−146

[74] Nguyen D, Lenharth A, Pingali K. A lightweight infrastructure for graph analytics[C]//Proc of the 24th ACM Symp on Operating Systems Principles. New York: ACM, 2013: 456−471

[75] Wang Guozhang, Xie Wenlei, Demers A, et al. Asynchronous large-Scale graph processing made easy[C/OL]//Proc of the 6th Biennial Conf on Innovative Data Systems Research. 2013[2022-08-12].https://doi.org/10.1002/spe.2979

[76] Perez Y, Sosič R, Banerjee A, et al. Ringo: Interactive graph analytics on big-memory machines[C]//Proc of the 2015 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2015: 1105−1110

[77] Kyrola A, Blelloch G, Guestrin C. GraphChi: Large-scale graph computation on just a PC[C]//Proc of the 10th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: 31−46

[78] Roy A, Mihailovic I, Zwaenepoel W. X-Stream: Edge-centric graph processing using streaming partitions[C]//Proc of the 24th ACM Symp on Operating Systems Principles. New York: ACM, 2013: 472−488

[79] Maass S, Min C, Kashyap S, et al. Mosaic: Processing a trillion-edge graph on a single machine[C]//Proc of the 12th European Conf on Computer Systems. New York: ACM, 2017: 527−543

[80] Jun S W, Wright A, Zhang S, et al. GraFBoost: Using accelerated flash storage for external graph analytics[C]//Proc of the 45th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2018: 411−424

[81] Vora K. LUMOS: Dependency-driven disk-based graph

processing[C]// Proc of the 24th USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2019: 429−442

[82] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing[C]//Proc of the 2010 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 135−146

[83] Low Y, Gonzalez J E, Kyrola A, et al. Graphlab: A new framework for parallel machine learning[J]. arXiv preprint, arXiv: 1408.2041, 2014

[84] Gonzalez J E, Low Y, Gu Haijie, et al. PowerGraph: Distributed graph-parallel computation on natural graphs[C]//Proc of the 10th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: 17−30

[85] Roy A, Bindschaedler L, Malicevic J, et al. Chaos: Scale-out graph processing from secondary storage[C]//Proc of the 25th Symp on Operating Systems Principles. New York: ACM, 2015: 410−424

[86] Yu Jiping, Qin Wei, Zhu Xiaowei, et al. DFOGraph: An I/O-and communication-efficient system for distributed fully-out-of-core graph processing[C]//Proc of the 26th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2021: 474−476

[87] Teixeira C H C, Fonseca A J, Serafini M, et al. Arabesque: A system for distributed graph mining[C]//Proc of the 25th Symp on Operating Systems Principles. New York: ACM, 2015: 425−440

[88] Abdelhamid E, Abdelaziz I, Kalnis P, et al. Scalemine: Scalable parallel frequent subgraph mining in a single large graph[C]//Proc of the 28th Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2016: 716−727

[89] Chen Hongzhi, Liu Miao, Zhao Yunjian, et al. G-miner: An efficient task-oriented graph mining system[C/OL]//Proc of the 13th ACM European Conf on Computer Systems. New York: ACM, 2018[2022-07-20].https://dl.acm.org/doi/10.1145/3190508.3190545

[90] Wang Kai, Zuo Zhiqiang, Thorpe J, et al. RStream: Marrying relational algebra with streaming for efficient graph mining on a single machine[C]// Proc of the 13th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2018: 763−782

[91] Dias V, Teixeira C H C, Guedes D, et al. Fractal: A general-purpose graph pattern mining system[C/OL]//Proc of the 2019 Int Conf on Management of Data. 2019[2022-09-17].https://dl.acm.org/doi/10.1145/3299869.3319875

[92] Yan Da, Guo Guimu, Chowdhury M M R, et al. G-thinker: A distributed framework for mining subgraphs in a big graph[C]//Proc of the 36th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2020: 1369−1380

[93] Chen Xuhao, Dathathri R, Gill G, et al. Pangolin: An efficient and flexible graph mining system on CPU and GPU[J]. Proceedings of the VLDB Endowment, 2020, 13(8): 1190−1205

[94] Trigonakis V, Lozi J P, Faltín T, et al. aDFS: An almost depth-first-search distributed graph-querying system[C]//Proc of the 26th USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2021: 209−224

[95] Mawhirter D, Wu Bo. Automine: Harmonizing high-level abstraction and high performance for graph mining[C]//Proc of the 27th ACM Symp on Operating Systems Principles. New York: ACM, 2019: 509−523

[96] Mawhirter D, Reinehr S, Holmes C, et al. Graphzero: Breaking symmetry for efficient graph mining[J]. arXiv preprint, arXiv: 1911.12877, 2019

[97] Jamshidi K, Mahadasa R, Vora K. Peregrine: A pattern-aware graph mining system[C/OL]//Proc of the 15th European Conf on Computer Systems. New York: ACM, 2020[2022-05-13].https://doi.org/10.1145/3342195.3387548

[98] Shi Tianhui, Zhai Mingshu, Xu Yi, et al. Graphpi: High performance graph pattern matching through effective redundancy elimination[C]. arXiv preprint, arXiv: 2009.10955, 2020

[99] Wang Minjie, Yu Lingfan. Deep graph library: Towards efficient and scalable deep learning on graphs[C]. arXiv preprint, arXiv: 1909.01315, 2019

[100] Fey M, Lenssen J E. Fast graph representation learning with PyTorch Geometric[J]. arXiv preprint, arXiv: 1903.02428, 2019

[101] Kaler T, Stathas N, Ouyang A, et al. Accelerating training and inference of graph neural networks with fast sampling and pipelining[J]. Proceedings of Machine Learning and Systems, 2022, 4: 172−189

[102] Min S W, Wu Kun, Huang Sitao, et al. Pytorch-direct: Enabling GPU centric data access for very large graph neural network training with irregular accesses[J]. arXiv preprint, arXiv: 2101.07956, 2021

[103] Liu Husong, Lu Shengliang, Chen Xinyu, et al. G^3: When graph neural networks meet parallel graph processing systems on GPUs[J]. Proceedings of the VLDB Endowment, 2020, 13(12): 2813−2816

[104] Gemawat A. GraphGem: Optimized scalable system for graph convolutional networks[C]//Proc of the 2021 Int Conf on Management of Data. Berkeley, CA: USENIX Association, 2021: 2920−2922

[105] Chen Zhaodong, Yan Mingyu, Zhu Maohua, et al. FuseGNN: Accelerating graph convolutional neural network training on GPGPU[C/OL]//Proc of the 41st IEEE/ACM Int Conf on Computer Aided Design. New York: ACM, 2020[2022-11-03].https://doi.org/10.1145/3400302.3415610

[106] Hamilton W, Ying Zhitao, Leskovec J. Inductive representation learning on large graphs[J]. Advances in Neural Information Processing Systems, 2017, 30: 12−30

[107] Fey M, Lenssen J E, Weichert F, et al. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings[C]// Proc of the 38th Int Conf on Machine Learning. New York: ACM, 2021: 3294−3304

[108] Tailor S A, Opolka F L, Lio P, et al. Adaptive filters and aggregator fusion for efficient graph convolutions[J]. arXiv preprint, arXiv: 2104.01481, 2021

[109] Waleffe R, Mohoney J, Rekatsinas T, et al. Marius++: Large-scale training of graph neural networks on a single machine[J]. arXiv preprint, arXiv: 2202.02365, 2022

[110] Khan N A, Khalaf O I, Romero C A T, et al. Application of Euler neural networks with soft computing paradigm to solve nonlinear problems arising in heat transfer[J]. Entropy, 2021, 23(8): 1053

[111] Hoang L, Chen Xuhao, Lee H, et al. Efficient distribution for deep learning on large graphs[J]. Update, 2021, 4(5): 1−10

[112] Jia Zhihao, Lin Sina, Gao Mingyu, et al. Improving the accuracy,

scalability, and performance of graph neural networks with roc[J]. Proceedings of Machine Learning and Systems, 2020, 2: 187−198

[113] Gandhi S, Iyer A P. P3: Distributed deep graph learning at scale[C]// Proc of the 15th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2021: 551−568

[114] Mariappan M, Vora K. Graphbolt: Dependency-driven synchronous processing of streaming graphs[C]//Proc of the 14th European Conf on Computer Systems. New York: ACM, 2019[2022-02-09].https://doi.org/10.1145/3302424.3303974

[115] Mariappan M, Che J, Vora K. DZiG: Sparsity-aware incremental processing of streaming graphs[C]//Proc of the 16th European Conf on Computer Systems. New York: ACM, 2021: 83−98

[116] Jiang Xiaolin, Xu Chengshuo, Yin Xizhe, et al. Tripoline: Generalized incremental graph processing via graph triangle inequality[C]//Proc of the 16th European Conf on Computer Systems. New York: ACM, 2021: 17−32

[117] Vora K, Gupta R, Xu Guoqing. KickStarter: Fast and accurate computations on streaming graPhs via trimmed approximations[C]//Proc of the 22nd Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM , 2017: 237−251

[118] Feng Guanyu, Ma Zixuan, Li Daixuan, et al. RisGraph: A real-time streaming system for evolving graphs to support sub-millisecond per-update analysis at millions ops/s[C]//Proc of the 2021 Int Conf on Management of Data. New York: ACM, 2021: 513−527

[119] Zhu Xiaowei, Feng Guanyu, Serafini M, et al. LiveGraph: A transactional graph storage system with purely sequential adjacency list scans[J]. arXiv preprint, arXiv: 1910.05773, 2019

[120] Wang Qinggang, Zheng Long, Huang Yu, et al. GraSU: A fast graph update library for FPGA-based dynamic graph processing[C]// Proc of the 30th Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2021: 149−159

[121] Zhang Yu, Liao Xiaofei, Jin Hai, et al. DiGraph: An efficient path-based iterative directed graph processing system on multiple GPUs[C/OL]//Proc of the 24th Int Conf on Architectural Support for Programming Languages and Operating Systems. 2019[2022-03-12].https://dl.acm.org/doi/10.1145/3297858.3304029

[122] Zhang Yu, Liao Xiaofei, Gu Lin, et al. AsynGraph: Maximizing data parallelism for efficient iterative graph processing on GPUS[J]. ACM Transactions on Architecture and Code Optimization, 2020, 17(4): 1−21

[123] Zhang Yu, Peng Da, Liao Xiaofei, et al. LargeGraph: An efficient dependency-aware GPU-accelerated large-scale graph processing[J]. ACM Transactions on Architecture and Code Optimization, 2021, 18(4): 1−24

[124] Zhang Yu, Liang Yuxuan, Zhao Jin, et al. EGraph: Efficient concurrent GPU-based dynamic graph processing[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 35(6): 1−13

[125] Engelhardt N, So H K H. GraVF: A vertex-centric distributed graph processing framework on FPGAs[C/OL]// Proc of the 26th Int Conf on Field Programmable Logic and Applications. Piscataway, NJ: IEEE, 2016[2022-11-14].https://doi.org/10.1109/FPL.2016.7577360

[126] Dai Guohao, Huang Tianhao, Chi Yuze, et al. ForeGraph: Exploring large-scale graph processing on multi-FPGA architecture[C]//Proc of the 26th ACM/SIGDA Int Symp on Field-Programmable Gate Arrays. New York: ACM, 2017: 217−226

[127] Zhang Yu, Liao Xiaofei, Jin Hai, et al. DepGraph: A dependency-driven accelerator for efficient iterative graph processing[C]// Proc of the 48th IEEE Int Symp on High-Performance Computer Architecture. Piscataway, NJ: IEEE, 2021: 371−384

[128] Jin Xin, Yang Zhengyi, Lin Xuemin, et al. Fast: FPGA-based subgraph matching on massive graphs[C]//Proc of the 37th IEEE Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2021: 1452−1463

[129] Chen Qihang, Tian Boyu, Gao Mingyu. FINGERS: Exploiting fine-grained parallelism in graph mining accelerators[C]//Proc of the 27th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2022: 43−55

[130] Dai Guohao, Zhu Zhenhua, Fu Tianyu, et al. DIMMining: Pruning-efficient and parallel graph mining on near-memory-computing[C]//Proc of the 49th Annual Int Symp on Computer Architecture. New York: ISCA, 2022: 130−145

[131] Yan Mingyu, Deng Lei, Hu Xing, et al. Hygcn: A GCN accelerator with hybrid architecture[C]// Proc of the 26th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2020: 15−29

[132] Liang Shengwen, Wang Ying, Liu Cheng, et al. Engn: A high-throughput and energy-efficient accelerator for large graph neural networks[J]. IEEE Transactions on Computers. Piscataway, NJ: IEEE, 2020, 70(9): 1511−1525

[133] Liang Shengwen, Liu Cheng, Wang Ying, et al. DeepBurning-GL: An automated framework for generating graph neural network accelerators[C/OL]//Proc of the 41st IEEE/ACM Int Conf on Computer Aided Design. Piscataway, NJ: IEEE, 2020[2022-11-12].https://doi.org/10.1145/3400302.3415645

[134] Zhao Jin, Yang Yun, Zhang Yu, et al. TDGraph: A topology-driven accelerator for high-performance streaming graph processing [C]//Proc of the 49th Annual Int Symp on Computer Architecture. New York: ISCA, 2022: 116−129

[135] Zhao Jin, Zhang Yu, Liao Xiaofei, et al. LCCG: A locality-centric hardware accelerator for high throughput of concurrent graph processing[C/OL]//Proc of the 33rd Int Conf for High Performance Computing, Networking, Storage and Analysis. 2021[2022-11-08].https://doi.org/10.1145/3458817.3480854

[136] Zhang Yu, Liao Xiaofei, Jin Hai, et al. HotGraph: Efficient asynchronous processing for real-world graphs[J]. IEEE Transactions on Computers, 2016, 66(5): 799−809

[137] Zhang Yu, Liao Xiaofei, Jin Hai, et al. CGraph: A correlations-aware approach for efficient concurrent iterative graph processing[C]// Proc of the 16th USENIX Annual Technical Conf (USENIX ATC 18). Berkeley, CA: USENIX Association, 2018: 441−452

[138] Zhang Yu, Zhao Jin, Liao Xiaofei, et al. CGraph: A distributed storage and processing system for concurrent iterative graph analysis jobs[J]. ACM Transactions on Storage, 2019, 15(2): 1−26

[139] Yuan Pingpeng, Xie Changfeng, Liu Ling, et al. PathGraph: A path centric graph processing system[J]. IEEE Transactions on Parallel

and Distributed Systems, 2016, 27(10): 2998−3012

[140] Zhang Yu, Liao Xiaofei, Shi Xiang, et al. Efficient disk-based directed graph processing: A strongly connected component approach[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 29(4): 830−842

[141] Zhang Mingxing, Wu Yongwei, Zhuo Youwei, et al. Wonderland: A novel abstraction-based out-of-core graph processing system[J]. ACM SIGPLAN Notices, 2018, 53(2): 608−621

[142] Zhao Jin, Zhang Yu, Liao Xiaofei, et al. GraphM: An efficient storage system for high throughput of concurrent graph processing[C/OL]//Proc of the 31st Int Conf for High Performance Computing, Networking, Storage and Analysis. 2019[2022-10-25].https://doi.org/10.1145/3295500.3356143

[143] Liao Xiaofei, Zhao Jin, Zhang Yu, et al. A structure-aware storage optimization for out-of-core concurrent graph processing[J]. IEEE Transactions on Computers, 2021, 71(7): 1612−1625

[144] Xie Chennning, Chen Rong, Guan Haibing, et al. Sync or async: Time to fuse for distributed graph-parallel computation[J]. ACM SIGPLAN Notices, 2015, 50(8): 194−204

[145] Chen Rong, Shi Jiaxin, Chen Yanzhe, et al. Powerlyra: Differentiated graph computation and partitioning on skewed graphs[J]. ACM Transactions on Parallel Computing, 2019, 5(3): 1−39

[146] Zhu Xiaowei, Chen Wenguang, Zheng Weimin, et al. Gemini: A computation-centric distributed graph processing system[C]//Proc of the 12th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2016: 301−316

[147] Zhang Yanfeng, Gao Qixin, Gao Lixin, et al. Maiter: An asynchronous graph processing framework for delta-based accumulative iterative computation[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 25(8): 2091−2100

[148] Wang Qiange, Zhang Yanfeng, Wang Hao, et al. Automating incremental and asynchronous evaluation for recursive aggregate data processing[C]//Proc of the 2020 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2020: 2439−2454

[149] Ma Lingxiao, Yang Zhi, Miao Youshan, et al. NeuGraph: Parallel deep neural network computation on large graphs[C]// Proc of the 15th USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2019: 443−458

[150] Wu Yidi, Ma Kaihao, Cai Zhenkun, et al. Seastar: Vertex-centric programming for graph neural networks[C]//Proc of the 16th European Conf on Computer Systems. New York: ACM, 2021: 359−375

[151] Wang Yuke, Feng Boyuan, Li Gushu, et al. GNNAdvisor: An adaptive and efficient runtime system for GNN acceleration on GPUs[C]// Proc of the 15th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2021: 515−531

[152] Yang Jianbang, Tang Dahai, Song Xiaoniu, et al. GNNLab: A factored system for sample-based GNN training over GPUs[C]//Proc of the 17th European Conf on Computer Systems. New York: ACM, 2022: 417−434

[153] Dai Guohao, Huang Guyue, Yang Shang, et al. Heuristic adaptability to input dynamics for SpMM on GPUs[J]. arXiv preprint, arXiv: 2202.08556, 2022

[154] Zhu Rong, Zhao Kun, Yang Hongxia, et al. Aligraph: A comprehensive graph neural network platform[J]. arXiv preprint, arXiv: 1902.08730, 2019

[155] Cai Zhenkun, Yan Xiao, Wu Yidi, et al. DGCL: An efficient communication library for distributed GNN training[C]//Proc of the 16th European Conf on Computer Systems. New York: ACM, 2021: 130−144

[156] Wang Qiange, Zhang Yanfeng, Wang Hao, et al. NeutronStar: Distributed GNN training with hybrid dependency management[C]//Proc of the 2022 Int Conf on Management of Data. New York: ACM, 2022: 1301−1315

Zhang Yu, born in 1987. PhD, associate professor, PhD supervisor. Senior member of CCF. His main research interests include computer architecture, system software, and high-performance graph processing.

张　宇, 1987 年生. 博士, 副教授, 博士生导师. CCF 高级会员. 主要研究方向为计算机体系结构、系统软件、高性能图计算.

Jiang Xinyu, born in 1998. Master. His main research interests include graph processing and graph neural network.

姜新宇, 1998 年生. 硕士. 主要研究方向为图处理、图神经网络.

Yu Hui, born in 1993. PhD candidate. His main research interests include graph processing and graph neural network.

余　辉, 1993 年生. 博士研究生. 主要研究方向为图处理和图神经网络.

Zhao Jin, born in 1996. PhD. His main research interests include graph processing system and accelerator design.

赵　进, 1996 年生. 博士. 主要研究方向为图处理系统与加速器设计.

Qi Hao, born in 1996. PhD. His main research interests include graph processing, graph neural network system design.

齐　豪, 1996 年生. 博士. 主要研究方向为图处理和图神经网络系统设计.

Liao Xiaofei, born in 1978. PhD, professor, PhD supervisor. Distinguished member of CCF, member of ACM and IEEE. His main research interests include system software, P2P system, cluster computing, and streaming services.

廖小飞, 1978 年生. 博士，教授，博士生导师. CCF 杰出会员，ACM 和 IEEE 会员. 主要研究方向为系统软件、P2P 系统、集群计算、流媒体服务.

Jin Hai, born in 1966. PhD, professor, PhD supervisor. CCF fellow, IEEE fellow, life member of ACM. His main research interests include computer architecture and virtualization.

金　海, 1966 年生. 博士，教授，博士生导师. CCF 会士，IEEE 会士，ACM 终生会员. 主要研究方向为计算机体系结构、虚拟化.

Wang Biao, born in 1990. PhD. His main research interests include large-scale graph data mining and anomaly detection.

王　彪, 1990 年生. 博士. 主要研究方向为大规模图数据挖掘、异常检测.

Yu Ting, born in 1989. PhD. Her main research interests include graph mining algorithms and parallel graph computing.

余　婷, 1989 年生. 博士. 主要研究方向为图挖掘算法、并发图计算.