图神经网络加速综述

文苏洋 M202173728

摘要 图神经网络（GNN）由于具有对图结构数据进行建模和学习的能力，近年来在机器学习领域得到了快速发展。这种能力在数据具有内在关系的大量领域中具有重要意义，但是传统的神经网络在这方面表现不佳。事实上，GNN 领域的研究已经迅速增长，并衍生出了各种 GNN 算法的变体，在化学、神经学、电子或通信网络等领域也产生了一些突破性的探索。然而，在目前的研究阶段，GNN 的高效处理仍然是一个公开的挑战。除了研究较为新颖之外，GNN 还很难计算，因为它们严重依赖输入的图数据，它们结合了密集和稀疏的操作，并且在某些应用程序中需要扩展出巨大的图。在此背景下，本文一方面从计算的角度对 GNN 进行了综述，包括了 GNN 基本原理的介绍、该领域在过去十年的发展概况，以及在不同 GNN 算法变体的多个阶段中进行的操作总结。另一方面，本文对当前的软件和硬件加速方案进行了分析，从中提炼出关于 GNN 加速器的软硬件、图形感知以及通信中心的愿景。

Abstract Graph Neural Networks (GNNs) have exploded onto the machine learning scene in recent years owing to their capability to model and learn from graph-structured data. Such an ability has strong implications in a wide variety of fields whose data is inherently relational, for which conventional neural networks do not perform well. Indeed, as recent reviews can attest, research in the area of GNNs has grown rapidly and has lead to the development of a variety of GNN algorithm variants as well as to the exploration of groundbreaking applications in chemistry, neurology, electronics, or communication networks, among others. At the current stage of research, however, the efficient processing of GNNs is still an open challenge for several reasons. Besides of their novelty, GNNs are hard to compute due to their dependence on the input graph, their combination of dense and very sparse operations, or the need to scale to huge graphs in some applications. In this context, this paper aims to make two main contributions. On the one hand, a review of the field of GNNs is presented from the perspective of computing. This includes a brief tutorial on the GNN fundamentals, an overview of the evolution of the field in the last decade, and a summary of operations carried out in the multiple phases of different GNN algorithm variants. On the other hand, an in-depth analysis of current software and hardware acceleration schemes is provided, from which a hardware-software, graph-aware, and communication-centric vision for GNN accelerators is distilled.

关键词 **GNN、机器学习、加速器**

1 引言

机器学习 (Machine Learning) 已经席卷了全世界，并成为了工程学的一个基本支柱，因为它能够解决极其复杂的问题，在大量数据中检测出复杂的特征，并能够自动生成比精心设计、精心优化的解决方案更好的替代

方案。在过去的十年里，我们目睹了神经网络的爆炸式发展，并亲眼见证了机器学习给我们生活带来方方面面的进步。

然而，尽管神经网络具有普遍的适用性和巨大的潜力，但并不是所有的神经网络结构都适用于所有的问题。考虑到 DNN 连接结构产生的归纳偏差，DNN 接受输入数据并尝试从中提取出需要的知识。从本质上说，这意味着 DNN 的层数及其预设的连接决定了它对特定任务的适用性。相比之下，卷积神经网络 (Convolutional Neural Networks, CNN) 或递归神经网络 (Recursive Neural Networks, RNN) 等技术则倾向于从数据的位置和时间顺序中提取知识。这使得它们更适合于特定的任务，如图像识别或处理时间信号，但不能有效地处理任意结构的数据。

最近，人们对能够对图结构进行数据建模的深度学习技术产生了兴趣。这种图结构广泛存在于大量复杂的系统中，并适用于的特定领域，如通信网络中的拓扑和路由决定其性能、合成化学中的分子结构决定化合物的性质、社会网络中的人际关系、或者神经科学中神经元类型和大脑区域之间的特定连接决定了大脑功能等等。

图神经网络 (GNN) 是一组连接驱动的模型，自 2000 年以来，一直用于解决几何深度学习中的问题。本质上，GNN 通过调整其结构以适应输入图的结构，并通过跨顶点信息聚合的迭代过程，捕获底层系统的复杂依赖关系。这允许预测特定节点、连接或整体图的属性，并将其推广到不可见的图。由于这些特点，近年来 GNN 领域的研究活动迅速增加。具体而言，人们正致力于提高算法的效率，特别是对于大型图形，并证明它们在上述应用领域的有效性。

然而，正如本文所述，很少有人关注这种新型神经网络的高效处理。虽然这个问题已经在 CNN 与 RNN 中进行了相当深入的研究，但 GNN 处理在很大程度上仍未被探索。这是因为 GNN 相对新颖，构成了独特的计算挑战，包括需要（1）同时支持密度和稀疏的操作，（2）计算适用于特定 GNN 算法变体的图的结构，（3）处理规模非常大的图形并实现特定的应用程序。

为了应对 GNN 计算面临的挑战，一些工作从软件优化的角度提高 GNN 性能和效率，例如调整计算操作以更好地匹配 CPU 或 GPU 的能力，或者从硬件的角度，为 GNN 的需求设计专门的处理器。此外，稀疏/不规则张量的加速研究可能在 GNN 中被证明是有用的。然而，从最近的调查和综述中进行全面的分析，可以推断出 GNN 处理领域仍处于起步阶段。

为了弥补这一空白，本文首次从计算的角度对 GNN 的研究进行了综述：首先，在原理与优势部分对 GNN 的基本原理提供了一个全面的描述，并从多个角度介绍了研究领域的演变。在研究进展部分，本文使用知识图（KG）的方法，绘制了该领域从诞生到撰写本文的时间的演化图，深入研究了 GNN 算法 (将其视为学习系统) 及其相关计算 (将其视为矩阵乘法和非线性运算的集合) 之间的对偶性；最后将重点放在计算方面，并对当前的软件和硬件加速方案进行了深入的分析，从中揭示了 GNN 加速器的新兴领域，总结了最近的工作，并阐述了现有的挑战和机遇，并在总结与展望部分对全文进行总结。

2 原理与优势

从根本上说，GNN 是一种算法，它利用图的连接性来学习和建模节点之间的关系。通过依赖于图结构的迭代过程，GNN 获取输入边、顶点和图特征向量 (表示它们的已知属性)，并将它们转换为输出特征向量 (表示目标预测)。一般情况下，GNN 操作如图 1 所示，包括以下步骤：

1) 预处理：这是一个初始的、可选的步骤，通常是离线完成的，可以通过预编码过程对输入的特征向量和图结构表示进行转换。这可以用来对图进行采样，降低算法复杂度，或者对特征向量进行编码。

2) 迭代更新：预处理完成后，通过聚合函数迭代更新每条边和顶点的特征向量。为了更新每条边，将边本身的属性、连接的顶

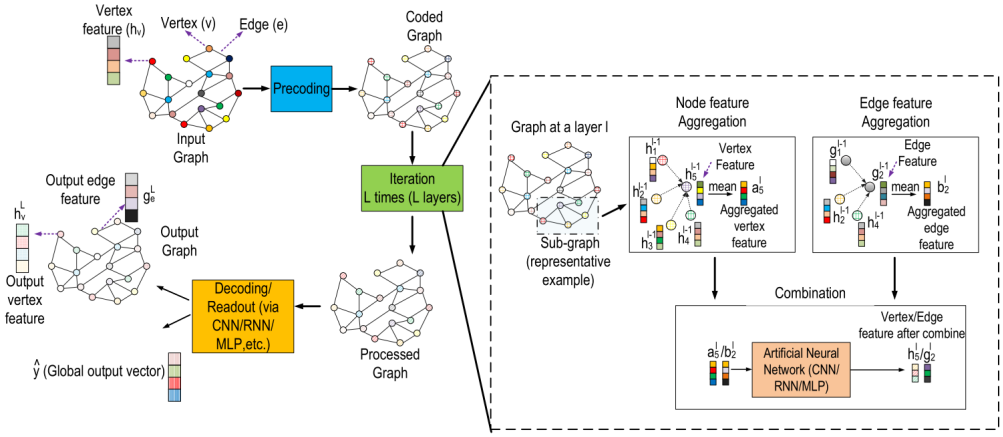

图 1: GNN 的执行阶段：预处理、迭代更新、解码或读出

点和图的属性聚合成一个集合，并组合生成一个新的边特征向量。类似地，更新顶点意味着将相邻顶点 N(v) 的特征向量进行聚合，并将其组合得到一个新的特征向量。需要注意的是，每个步骤更新每个边和顶点的信息都来自于单跳的邻居。因此，迭代过程允许逐渐考虑更加远的节点和边之间的关系。

3) 解码或读出：如果图有全局特征向量，则在边和节点更新完成后对全局特征向量更新一次。最终的输出要么是边/节点的嵌入，这是代表特定边或节点信息的低维特征向量；要么是概括整个输出图信息的图嵌入。

与其他任何神经网络一样，GNN 的处理取决于它的体系结构。GNN 通常只有几层，每一层对应于上面描述的更新过程中的一次迭代。层的数目越大，来自一个给定节点的信息可以传播到图中更远的另一个节点。因此，所需层的数量取决于应用程序以及遥远节点之间关系的相关性。通常，过多的层会导致模型的过度拟合，因此，这个数字通常保持在较低的水平。在每一层中，顶点之间的信息流使用聚合函数、聚合后的组合函数来更新特征向量。边和顶点的聚合和组合函数由特定的 GNN 算法根据需要学习的特定关系来确定。这类函数种类繁多，从聚合的简单平均或求和到不同类型的神经网络，组合具有各自的加权和和非线性激活函数。操作可能因层而异，也可能因边、顶点或全局更新而异，然而，结构通常通过（1）跨层共享相同的操作和（2）删除或考虑更细粒度的组合函数来简化边或节点的更新操作。总之，我们可以把 GNN 理解为在图的连接性上工作的神经网络的集合。在每一层的范围内，我们有两个具有可学习权值的神经网络，分别确定边和顶点的组合，而在整个 GNN 的范围内，我们有一个具有可学习权值的神经网络来决定全局参数的更新。

聚合、组合和读出函数通常由神经网络来实现，因此，在应用推理之前需要对神经网络进行训练。训练是通过修改传统的反向传播算法来完成的，该算法考虑到 GNN 的独特架构。由于 GNN 与 RNN 类似可以展开为 L 层，因此大多数 GNN 采用 BPTT(Back-Propagation-Through-Time) 方案或其变体。具体来说，在 BPTT 中，首先对具有 L 层的 GNN 的展开版本进行一次前向传递，然后计算损失函数，并将得到的梯度反向传播到各个层。由于权值在所有 L 层中共享，因此它们会相应地进行更新。这一过程对多个数据来说是反复进行的，直到达到用户定义的目标精度。

损失函数的计算取决于学习的类型，而学习的类型又取决于应用程序的需求。以图为中心的方法往往使用监督学习来训练，而以节点为中心的方法往往使用半监督学习来训练，其中节点特征的信息来自图的一部分，而不是整个图。前一种方法的一个例子是，使用已知分子属性作为 ground truth 的 GNN

来学习一个特定的新分子是否具有特定的属性。对于后一种方法，可以以推荐系统为例：在这样一个系统中，图表示一个商店，节点是购物商品及其特征，边是商品之间的关系。输出特征向量可以描述特定用户对特定商品满意的可能性。在这种情况下，先验的完整信息往往是不可用的，并且使用半监督学习方法学习到的该用户和其他用户过去购买的商品信息来代替。

3 研究进展

在学术界和工业界揭示 DNN 算法的突出潜力及其普遍适用性后不久，机器学习算法的优化以及高性能、高能效的专用硬件的构建经历了前所未有的增长。正如前几节所描述的那样，GNN 领域的发展已经到达了一个类似的转折点。目前，GNN 算法的研究已经很广泛，并不断在具有较高影响力的新应用场景中得到应用。因此，如何高效地计算 GNN 以充分发挥其潜力将是未来几年的一个关键研究方向。

GNN 算法带来了一些独特的挑战，使得现有的库和硬件平台计算效率低下，包括：(1) 存在多种 GNN 变体，主要包括归纳型和转导型，并具有多种聚合和组合功能。(2) GNN 计算对输入图的特征在大小、稀疏性、聚类或相关特征向量的长度方面具有很强的依赖性，图的连通性可以服从幂律分布、均匀分布或二项分布。(3) 深度学习和图处理计算特性的独特组合，导致了不同的执行模式。前者是计算受限的，涉及密集矩阵乘法，而后者是存储受限的，涉及稀疏代数。(4) 在某些应用中，需要扩展出具有数十亿条边的非常大的图结构。

鉴于这些挑战，GNN 要求在软件和硬件方面都有新的解决方案。在软件方面，业内已经提出了几个专用库来改进对 GNN 的支持并处理它的多种变体，比如流行库 PyTorch 或 Tensorflow 的扩展。在硬件方面，最近出现了新的加速器架构，试图解决 GNN 的灵活性和可伸缩性的挑战。

3.1 软件框架和加速器

GNN 处理的挑战使得传统的 DNN 库效率低下。为了弥补这一差距，最近的工作已经开始研究如何调整库使得：(1) 提供易于编程的接口来实现多种 GNN 变体，(2) 在广泛的 GPU 硬件中有效地处理 GNN 操作的稀疏性，(3) 将计算扩展到大规模图结构和多个 GPU。

接下来，本文回顾了软件框架和加速器的综合选择。软件框架已经对各种各样的 GNN 算法和相关数据集进行了测试，已经评估了大约 20 种不同的 GNN 变体，其中 GCN、GS 和 GIN 是最常见的。尽管 Amazon、Reddit、Protein、Cora 或 CiteSeer 数据集在社区中很流行，但由于缺乏广泛采用的基准测试套件，这些数据集之间的差异会变得非常大。然而，值得注意的是，图形可以从化学应用中的数百条边到大规模推荐系统中的数十亿条边不等。

PyTorch Geometric (PyG)。PyG 是一个广泛使用的库，它建立在 PyTorch 上，提供了对关系学习的支持。关键方面是消息传递接口的定义，定义了消息和更新函数，分别用于邻居聚合和组合。以及多个池操作。为了加速 GNN 处理，PyG 通过专用的 GPU 优化算法分散处理图数据的稀疏性，并收集并行运行在所有边和节点上的核，而不是使用稀疏矩阵的乘法核。与之相关的是，Facebook 发布了 Pytorch-BigGraph，它通过引入分区和分布式处理，允许处理任意大的图结构，并可以为 PyG 库进行补充。

Deep Graph Library (DGL)。DGL 是一个可以在 TF、PyTorch 或 MXNet 上工作的库，并为多个 GNN 变体提供了大量的示例和代码。DGL 库定义了三个函数：用于边缘聚合的 message 和用于节点聚合和更新的 update 与 reduce。为了提高性能，DGL 采用矩阵乘法的方式，并利用 GPU 或 TPU 的专用内核。特别地，将采样密集矩阵乘法和稀疏矩阵乘法与节点、边缘或特征并行化结合起来考虑。DGL 使用启发式方法在不同的选项中进行选择，因为最优并行化方案取决

于包括输入图在内的多个因素。由于采用了这种方法，DGL 声称其训练速度比 PyG 快一个数量级。

NeuGraph。微软研究院领导了第一个在 GPU 上并行处理 GNN 的专用框架 NeuGraph。尽管 NeuGraph 构建在 TF 之上，但它还不是开源的。该框架基于散点 (Scatter) 函数、ApplyEdge 函数、Gather 函数和 ApplyVertex 函数实现了 SAGA-NN 编程模型。散集核用于同名函数，而矩阵乘法用于组合函数。NeuGraph 还提供了一些优化来加速 GNN 计算。首先，通过 Kernighan-Lin 算法对大型图进行分区，使分区更加密集，并最小化分区之间的传输，从而降低性能。其次，通过将可以一起计算的稀疏小分区进行批处理，优化了分区对 GPU 的调度，并将 GNN 层的传输和计算时间完美地映射到后续的不同管道块中。此外，NeuGraph 还通过将多条边融合在一起，消除了冗余计算。最后，它允许通过分配计算将 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，从而最小化带宽上的争用。

AliGraph/Euler。AliGraph 由阿里巴巴集团开发，又名 Euler，是一个建立在 TF 之上的 GNN 框架。该框架能够用于处理大型计算系统中非常大的动态图，目前在阿里巴巴的推荐服务中广泛使用。它实现了三层，即：(1) storage 层，用四种不同的算法实现分区，在这种情况下，以分布式的方式存储图；(2) sampling 层，与其他框架不同，它允许定义与算法 (如 GraphSAGE) 相关的节点邻域的自定义抽样；(3) operator 层，实现了聚合和组合函数。在总体上，AliGraph 的独特性主要是由于其分布式算法和许多优化存储层减少了数据的移动，如根据图像的特点使用四个不同的分区算法，或对重要的顶点进行缓存降低机器故障带来的影响。

ROC。ROC 是另一个针对多 GPU 系统的 GNN 框架。与 AliGraph 类似，ROC 能够将大型图结构分布到多台机器上。但是，这个框架与其他框架的不同之处在于，分区和内存管理是通过提供额外加速的动态方法来执行的。首先，ROC 使用线性回归模型最优地逼近分区，该模型使用训练迭代来学习特定图的最佳策略，显著优于静态方法。其次，内存管理被视为是成本最小化问题，可以通过在线算法找到最佳存储每个分区的位置来解决。ROC 的作者证明，这种加速方法在单个 GPU 上比 DGL 和 PyG 提供了更好的可伸缩性，而在多个 GPU 上比 NeuGraph 提供了更好的可伸缩性。

GNNAdvisor。GNNAdvisor 提出了一个运行时系统，旨在系统地在 GPU 上加速 GNN。GNNAdvisor 没有像 ROC 中那样通过抽象模型来处理这个问题，而是对输入图和 GNN 操作进行在线分析，以指导 GPU 中的内存和工作负载管理。具体来说，该算法利用节点的度对 GPU 基于组的工作负载管理进行微调，利用节点嵌入的大小优化工作负载共享，利用图内社区性来指导划分和调度。虽然前两个特性很容易获得，但是社区检测通常比较困难。在这种情况下，作者使用节点重编号和反向 CuthillâSMcKee 算法的组合，以一种可用密集分区的方式重新排序邻接矩阵。由于这些技术，GNNAdvisor 的作者声称在高端 GPU 中可以比 DGL、PyG 和 NeuGraph 多加速 3 -4 次。

3.2 硬件加速

我们已经看到，上述软件方法简化了大多数计算系统中通用计算平台的 GNN 执行步骤，使得 GNN 计算速度得到显著提升。部分论文分析了 TPU 中 GNN 训练的性能，TPU 通常用于密集型深度学习，结果依然相似。然而，一个相关的问题是，专用硬件加速器是否能够解决 GNN 计算的独特挑战，并实现 GNN 计算速度数量级进步。为了实现这一目标，出现了几种不同的硬件加速器，它们试图处理 GNN 的极端密度和交替计算需求。本文接下来讨论所有设计均使用如图 2 所示的结构示意图作为参考。该图还试图根据它们的紧密耦合程度 (从统一到平排) 以及将它们适应多个 GNN 变体的容易程度对它们进行分类。

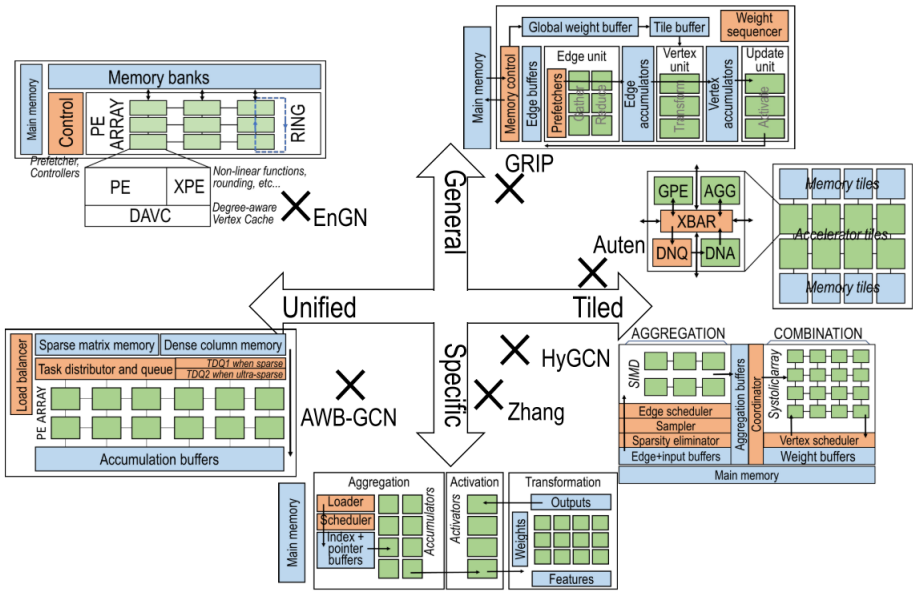

图 2: 用于 GNN 推理的硬件加速器的分类和原理图表示。绿色、蓝色和红色方块分别代表处理器、内存和控制单元。

目前，大多数的工作都围绕着 GCN 算法进行，GCN 算法很流行，那些评估多种算法的人通常会试图证明它们的通用性。硬件加速使用的数据集通常比软件加速使用的数据集要小，这主要是因为在推理中硬件加速器的内存限制和模拟硬件架构的成本，Cora、CiteSeer 和 Redit 是几种最常见的数据集。由于涉及到许多变量，很难进行性能比较，大多数工作都使用 CPU 和 GPU 作为基线，在某些情况下，甚至使用一个硬件加速器。但是，在基线中使用哪个软件框架还没有达成共识。

EnGN。在最早出现的加速器中，EnGN 采用了一个统一的架构，深受 CNN accelerators 的启发。GNN 基本上被视为特征向量、邻接矩阵和权值的串联矩阵乘法——所有这些都安排在一个数据流中。为了实现聚合，PE 的每一列通过一个环相互连接，结果根据邻接矩阵进行传递和添加，这个过程被称为 Ring-Edge Reduce (RER)。在 EnGN 中，稀疏性是通过多次优化来处理的。首先，对于稀疏连接的节点，RER 聚合可能会导致多个无效计算，为了避免这种情况，EnGN 在 RER 的每一步中动态地重新排序每条边。其次，PE 集群被连接到一个度感知的顶点缓存，该缓存保存着关于高度的顶点数据。其原因是，连接良好的顶点将在计算过程中出现多次，缓存它们将以适中的成本提供高收益。其他优化设计决策涉及到聚合函数为 sum 时矩阵乘法的顺序，这会影响总操作数，或者平铺策略，同时也会影响数据重用和 I/O 的成本。

HyGCN。HyGCN 的作者在观测到 GNN 存在两个主要交替阶段的计算需求的基础上，引入了 GCNs 的混合体系结构。HyGCN 由独立的用于聚合和组合阶段的专用引擎组成，外加一个协调这两个函数流水线执行的控制机制，由密集合并期通过常规的收缩阵列方法进行计算。聚合阶段还有一个更复杂的体系结构，具有采样器、边缘调度程序和提供一组 SIMD 核心的稀疏消除器。在 HyGCN 中，由于高效的调度和稀疏消除器，稀疏性由聚合引擎处理，后者采用基于窗口的滑动和收缩方法来动态适应不同程度的稀疏乘法。为了进一步适应工作负载，HyGCN 允许根据特征向量的大小以不同的方式对 SIMD 核进行聚合分组，并对 PEs 进行组合。最后，需要特别关注引擎间协调器的设计，HyGCN 可以优化内存访问，并允许细粒度的流水线执行，从而动态地最大化并行性。

AWB-GCN。自动调优、工作负载平衡的 GCN 加速器主要主张对 GNN 的结构稀疏性进行积极的适应。AWB-GCN 的作者通过分析大多数图的幂律分布来优化他们的设计，他们认为计算的某些部分将是密集的，而其他部分则是非常稀疏的，从而造成了计算资源分配的不平衡。为了解决这种不平衡，该架构开发了一个定制的矩阵乘法引擎，它有效地支持对零值的跳过。为此，内存中的数据通过任务分发器和队列 (TDQ) 提供一组 PE 和累加器，TDQ 采用两种适合于稀疏度中等的设计。AWB-GCN 的关键在于实现了三种负载均衡功能。第一种是本地的，尝试在相邻的 PE 之间均衡负载；第二个是远程的，尝试将溢出计算从繁忙的 PE 倒向单个未充分利用的远程 PE；第三种方法承担处理非常密集节点集群的非常繁忙的 PE 的负载，并跨多个空闲 PE 进行划分。为了支持这一点，AWB-GCN 在 TDQ 提供硬件到 PE 的连接，以允许节点重新映射到远程 PE，并将它们带回来进行相干聚合。此外，所有关于平衡的决策都是基于从队列的简单计数中提取的信息。

3.3 软硬件协同设计

前几节讨论了如何将 GNN 理解为一组在图结构数据上工作的经典神经网络。我们已经看到，为了从图中提取特定的知识，可以使用不同的 NN 层，从而产生各种各样的 GNN 变体。这一点，再加上 GNN 对输入图 (可能非常大) 的基本依赖，极大地简化了其执行的任务。因此，已经出现的试图加速 GNN 的研究隐含地做出了选择：要么为特定的 GNN 变体提供一个极其有效的加速方案，要么为多种类型的 GNN 提供一个不够有效但通用且足够灵活的加速方案。因此，GNN 加速的关键挑战是如何实现一个操作和架构框架，能够最大化性能和效率，同时保持一定程度的灵活性，以适应不同图结构的大小、特征和 GNN 算法。尽管这是一项艰巨的任务，但在本节中，我们的目标是利用对现有加速工作的分析来假设未来 GNN 加速器应该具有的主要特征。

对之前工作的分析表明，软件和硬件方法都可以提供显著的加速。在某些情况下，人们可能会认为这两种策略处理问题的方式相似，例如软件中的节点重新排序和硬件中的工作负载平衡。然而，一些论文也开始意识到这两种方法并不是相互排斥的，它们的优点可以叠加，或者一种可以简化另一种。例如，可以通过软件预处理消除冗余操作，然后通过专门的聚合和组合模块优化执行。软件方面能够避免使用专门的硬件结构来消除冗余操作。基于这一观察，若将软硬件协同设计作为一种策略，可以有效地处理不同的 GNN 和图形，同时保持一些硬件的简单性。本文提倡一种控制数据平面模型，在这种模型中，控制平面将完全在软件中实现，提供了灵活性，而数据平面将在定制硬件中实现，提供了效率。虽然概念分离（如图 3 所示），但两个平面的操作将紧密耦合。

一方面，控制平面通过拥有完整 GNN 结构和输入图的全局视图来管理加速器的动作。控制平面负责控制在数据平面中运行的数据流，方法是（1）将 GNN 计算划分为可管理的计算段，（2）将不同的顶点和边映射到数据平面的硬件资源，以及（3）调度不同的执行以平衡工作负载，最后，还需要考虑部分控制平面驱动预处理 (可能离线) 步骤，如删除冗余操作或某些图方面 (如社区检测) 的计算。通过在软件中实现，这些功能可以提供所需的灵活性，以加速任何 GNN 工作负载。

另一方面，数据平面由按照执行 GNN 的控制平面指令工作的处理和内存元素组成，可以采用许多策略来构建数据平面。然而，与 MAERI 体系结构类似，在该体系结构中，一个同构的处理元素数组 (PEs) 和一个专门的内存层次结构通过一个轻量级可配置的互连结构组合在一起。这种结构根据控制平面的命令调整数据流，从而允许为一个算法或不同算法的多个执行阶段提供服务。

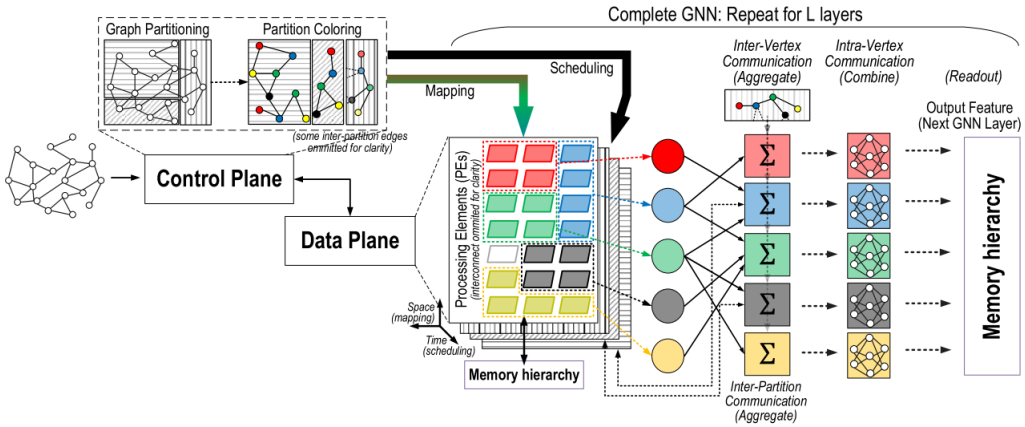

图 3: GNN 加速器的架构愿景，包括软硬件协同设计 (即控制和数据平面)、图形感知 (即引导映射和调度) 和以通信为中心的设计 (即可重构互连)。

4 总结与展望

最近对几何深度学习的兴趣，以及能够建模和预测图结构数据的方法，导致了围绕 GNN 的研究的爆炸式增长。正如本文对当前技术状态的分析中所提及的那样，大多数工作都集中在算法及其应用上，这使得 GNN 计算领域变得鲜为人知。但是，可以预见，GNN 的软硬件支持领域将快速增长，延续从 2018 年到今天所观察到的上升趋势。

对 GNN 更高效计算手段的研究可能增加的原因有几个。首先，该领域正在成熟，更多的理论算法驱动的研究让位于面向应用的发展，这一趋势的一个最明显的例子是统一标准等方法的出现。其次，GNN 是多个领域中许多颠覆性应用的关键，因而推动对更高效处理的需求。最后，GNN 提出了多种独特的挑战，如各种各样的算法变体以及它们对图特征的依赖，或者它们在某些应用中的大规模部署，这使得 GNN 处理领域在可预见的未来不太可能饱和。

最后，本文强调了软件框架的日益流行，以及最近出现的用于 GNN 的硬件加速器。在软件方面，DGL 或 NeuGraph 等库的目标是加快计算速度，并为 TensorFlow 或 PyTorch 等广泛使用的框架添加功能。此外，通过图分析或预编码加速 GNN，以及计算在大型系统中的分布，这对于大型推荐系统来说至关重要。在硬件方面，本文没有观察到明确的体系结构趋势，现有学术界正在争论是使用统一的体系结构还是使用更分层、平摊的方式进行组织。基于这一观察结果，本文设想未来的加速器将采用一种软硬件协同设计的方法来最大化性能，将图形感知作为一个可能的优化方向，并通过可重构互连技术来解决工作负载的变化。