发布日期 xxxx 年 00 月 00 日，当前版本日期 xxxx 年 00 月 00 日。数字对象标识符*10.1109/ACCESS.2017.DOI*

MQ-GNN：面向可扩展高效图神经网络训练的多队列流水线架构

IRFAN ULLAH[1], 和 YOUNG-KOO LEE[2]（IEEE 会员）[1]庆熙大学计算机科学与工程系（国际校区），大韩民国（电子邮箱：irfanr@khu.ac.kr）[2]庆熙大学软件学院（国际校区），大韩民国（电子邮箱：yklee@khu.ac.kr）

C通信作者：Young-Koo Lee（电子邮箱：yklee@khu.ac.kr）。

arXiv:2601.04707v1 [cs.LG] 8 Jan 2026

摘要 图神经网络（GNN）是学习图结构数据的强大工具，但其可扩展性受到低效小批次生成、数据传输瓶颈以及高昂的 GPU 间同步开销的限制。现有训练框架无法使这些阶段重叠执行，导致资源利用率不佳。本文提出 MQ-GNN，这是一种多队列流水线框架，通过交错执行图神经网络训练阶段并优化资源利用率，最大限度地提高训练效率。MQ-GNN 引入了就绪更新异步一致模型（RaCoM），能够在确保全局一致性的同时，通过自适应周期性同步实现异步梯度共享和模型更新。此外，该框架采用结合缓存的全局邻居采样，以降低数据传输开销，并采用自适应队列大小策略，在计算效率和内存效率之间取得平衡。在四个大规模数据集和十个基线模型上的实验表明，MQ-GNN 实现了最高 **4.6** × 的训练时间提升，并实现了 **30%** 的 GPU 利用率提升，同时保持了具有竞争力的准确率。这些结果证明，MQ-GNN 是一种面向多 GPU 图神经网络训练的可扩展高效解决方案。代码可在 MQ-GNN 获取。

索引词 图神经网络，多 GPU，流水线，优化，CPU-GPU混合训练，图神经网络训练，陈旧性，GPU 间通信

I. 引言

图 GRAPHS 能够自然地表示许多现实应用中的复杂关系，例如社交网络，b生物系统和知识图谱 [1]。通过学习l图节点的低维嵌入，图神经网络能够高效地表c示图信息，从而支持广泛的应用o用于下游应用，例如节点分类 [2]–[3] 和链接预测 [1], [4], [5]。多种模型[2], [3] 在各类领域均展现出最先进的性能，包括蛋白质结构 [6]、社交网络 [7], 和知识图谱 [8]。

尽管取得了成功，在分布式图形处理器环境中对大规模图进行图神经网络训练，仍面临着诸多挑战，其原因在于内存限制、计算成本以及图结构固有的相互依赖关系。现实世界中的图通常包含数百万个节点和边，以及高维的节点和边特征 [9]–[10]。这些因素会带来巨大的存储和计算开销。某些数据集可能需要数 GB 的存储空间 [11], ，因此，对每个训练节点而言，将 L 跳内的所有邻居作为单个批次来处理，效率很低，甚至不可行。在使用 L 层[12]训练图神经网络模型时，此外，大多数图都呈现高度偏斜的度分布 [13], ： 少数连接紧密的节点仅需经过几跳，就会汇集来自图中很大一部分区域的特征。这些挑战使得小批量训练成为必要；该方法通过迭代处理较小的样本，降低计算和内存开销 [14]。

对于小批量训练，已经开发了逐节点采样 [2], [15] 和逐层采样[4], [5], [16] 等基于采样的方法。逐节点采样和逐层采样是训练图神经网络最常用、最实用的方法之一，各自应对着独特的可扩展性和计算效率挑战。逐节点采样为每个节点选取固定数量的邻居。然而，它也带来了额外的计算挑战，尤其是在小批次大小增加时。例如，在 L 层图神经网络中计算嵌入，需要聚合来自 L 跳邻居的信息，从而导致众所周知的“邻居爆炸”问题 [15]–[17]。为了解决这一问题，人们开发了逐层采样。该方法在每一层选取节点子集以生成样本。

然而，逐节点采样和逐层采样都需要在中央处理器与图形处理器内存之间频繁传输数据，从而降低训练效率。为减少频繁且冗余的数据传输，全局邻居采样（GNS）[18]会定期为所有小批次采样少量节点，并将其特征缓存在 GPU 内存中。GNS 优先选择缓存中的邻居，从而减少每个批次中的不同节点数量、提高节点间的重叠度，并最大限度地减少 CPU-GPU数据传输，以加快训练速度。然而，样本生成、数据传输和计算未能高效重叠。缺乏高效的队列和流水线，以及图神经网络训练任务（例如小批次准备、数据传输、计算和更新）交错执行不佳，导致延迟显著增加，计算资源利用不足。对于大规模图，这一问题尤其严重，因为采样和数据传输占据了大部分运行时间，导致图形处理器因缺少数据而空等，训练过程也因此停滞。

近年来，多图形处理器已被广泛用于加速图神经网络训练 [19], [20]。在典型的单机多 GPU 配置中，大规模图（包括拓扑结构和特征数据）超出了有限的 GPU 内存容量（例如，NVIDIA 3060 的 12 GB），因此大多数现有图神经网络系统会将拓扑数据和特征数据存储在主机内存中。中央处理器反复对输入图进行采样，提取采样节点的特征，并将其传输到图形处理器进行训练。遗憾的是，这一过程会产生高昂的数据传输成本，成为训练的主要耗时，并导致图形处理器利用率严重不足。这是因为图神经网络模型通常使用深度神经网络，其计算复杂度相对于庞大的数据量而言很低。因此，图形处理器上的图神经网络计算速度远快于数据加载。例如， [12]报告称，数据加载占训练时间的 74%。现有图神经网络训练框架，例如深度图学习库（DGL）[21] 和 PyTorch Geometric（PyG） [22], , 对小批量生成、数据传输和计算等过程进行了部分交错执行。然而，这些框架仍存在明显的低效问题，因为这种交错执行无法让这些任务充分重叠。图形处理器经常因等待中央处理器完成采样和数据传输而处于空闲状态，导致计算资源利用率低下（最高仅为 50%）。在更大的图上，这些瓶颈会进一步加剧，造成节点的冗余访问、通信成本增加以及训练时间延长。在多 GPU 系统中，数据样本需求会线性增长，进而增加通信和同步开销，进一步限制可扩展性。随着 GPU 数量增加，系统必须生成并分发更多小批次，才能让所有图形处理器保持活跃。这加重了系统的负担，包括生成样本、获取中央处理器上的特征数据并将其传输到图形处理器。随着图形处理器数量增加，通信量也会随之增长，从而带来更大的开销和延迟。每个图形处理器都需要与其他图形处理器交换梯度或中间结果，以更新模型。协调各图形处理器上的模型更新和梯度需要频繁同步。随着图形处理器数量增加，同步成本也会随之上升，导致出现瓶颈：图形处理器只能空闲等待其他图形处理器赶上进度。

为克服这些局限性，我们提出了 MQ-GNN，一种多队列流水线架构。这种灵活的可扩展流水线架构通过交错执行小批量生成、数据传输、图神经网络计算、梯度共享和模型更新，优化资源利用率。MQ-GNN 通过使用多队列来管理中央处理器和图形处理器内存中的小批次，以及图形处理器内存中的梯度，实现了这一目标。借助主存中的小批量队列（每个图形处理器对应一个），MQ-GNN 确保数据传输与计算能够并发进行，从而最大限度地减少图形处理器空闲时间。该架构能够高效适配任意节点级或逐层采样方法，无论采用何种采样方法，都能提供稳定的训练性能。此外，MQ-GNN 采用全局邻居采样，定期缓存小批次之间的共享节点，减少冗余访问和 CPU-GPU数据传输，从而加速训练。

此外，MQ-GNN 引入了就绪更新异步一致模型（RaCoM），利用梯度队列实现图形处理器之间的异步梯度共享。这使得梯度能够在模型与下一小批次计算并发进行更新的同时累积。MQ-GNN 根据数据集稀疏性或密度调整周期性同步的时间间隔，以确保模型一致性并减轻发散。该方法平衡了异步梯度共享与更新，降低了陈旧性并提高了训练效率。

在多队列流水线中确定最佳队列大小，对于平衡计算效率和内存使用量至关重要。队列过大会增加内存使用量，并可能导致溢出；队列过小则会增加延迟，无法充分利用资源。在使用内存和计算需求各异的大型数据集进行训练时，这种权衡尤为关键。我们提出了一种确定最佳队列大小的方法，具体包括：(1) 将采样和数据传输时间与 GPU 计算时间相结合；(2) 将队列大小限制在峰值内存使用量允许的范围内。这样既能确保内存密集型数据集的训练可行，又能在内存限制内最大限度地提高 GPU 利用率。

最后，我们研究了节点级采样和逐层采样两种方法，以展示 MQ-GNN 在图神经网络训练中对不同采样方法的有效性和适应性。此外，我们对这两种采样方法进行了全面的多 GPU 分析，考察了 GPU 数量、数据集差异以及陈旧性对性能的影响。据我们所知，这是首项在流水线式多队列框架下评估多 GPU 训练中逐层采样的工作，展示了 MQ-GNN 在各种配置下的可扩展性和效率。

总之，本文的主要贡献如下：1）提出 MQ-GNN，这是一种多队列流水线架构，可优化中央处理器、内存和图形处理器的利用率
n
通过交错执行小批量生成、数据传输、计算和梯度共享。这种方法

降低了数据传输延迟，并在图神经网络训练期间最大限度地提高了图形处理器的利用效率。

- 为提高 GPU 利用率，我们提出一种根据可用内存和处理时间确定最佳队列大小的方法。
- 采用全局邻域采样缓存频繁访问的节点，从而减少冗余访问并最大限度地降低 CPU-GPU数据传输。

2）提出 RaCoM，这是一种用于异步梯度共享的框架，通过使用梯度队列，使图形处理器能够独立进行本地模型更新，从而减少同步延迟。

- 引入一种周期性同步机制，根据图的稀疏度或密集度，在通信开销与梯度陈旧度之间取得平衡，以提高模型一致性，增强可扩展性和鲁棒性。

3）对多 GPU 环境下的节点级采样和逐层采样进行全面的实证分析，首次评估多 GPU 环境下逐层采样的性能和陈旧性，并为可扩展性和效率提供见解。

II. 相关工作

图数据集规模不断扩大，推动了高效且可扩展的图神经网络（GNN）训练范式的发展。本节旨在探讨并回顾现有研究。

A. 可扩展图神经网络训练

已有多个系统旨在促进基于图形处理器的图神经网络训练 [18], [23], [24]。深度图学习库（DGL） [21]和 PyTorch Geometric（PyG）[22] 等框架利用 CPU 内存存储图，并支持在多个图形处理器上进行分布式小批次训练。然而，这些框架往往存在效率低下的问题，包括图形处理器利用率不足、多 GPU 配置下加速比低于线性水平[25], 以及较高的数据传输开销。为应对这些挑战，PaGraph [12]通过计算感知缓存降低数据传输开销，而 DSP [26]则通过动态图划分和同步小批量改进多图形处理器训练。MSPipe [27] 利用流水线执行和内存高效缓存优化时序图神经网络，以缓解陈旧性。

MQ-GNN 的不同之处在于，它将小批次生成、数据传输、计算和梯度共享整合到一个采用多队列的流水线架构中。

B. 图采样方法

图采样方法对于应对图神经网络训练中的邻居爆炸和内存限制等挑战至关重要。GraphSAGE 中的固定邻居采样等逐节点采样（NS） [2],方法会针对每个节点独立选择邻居，从而减少计算，但也会导致嵌入计算中出现冗余 [16]。其他研究提出了层级采样（LS）方法，例如 FastGCN [5]。FastGCN 根据每个节点度数导出的概率，为每层采样指定数量的节点。由于连续层中采样的节点可能并不相连，且其采样概率是独立计算的，因此图神经网络泛化和训练性能会受到影响。为确保快速、平稳地收敛，理想情况下应计算采样概率，以降低 FastGCN 中的估计方差 [5]。因此，邻接矩阵可能变得非常稀疏，或包含全零行，导致节点非连通、计算图不精确，最终降低 FastGCN 的训练和泛化性能。Huang 等人 [16]提出了一种自适应可训练采样方法，该方法根据前一层执行 LS，以捕捉层间相关性并降低估计方差。该方法取得了比 FastGCN 更高的准确率，但代价是采样方法复杂得多。Zou 等人 [28] 提出了层依赖重要性采样（LADIES），这是一种高效的采样算法，兼具以往方法的优势并弥补其不足。LADIES 进一步降低了训练方差，并旨在缓解 FastGCN 中的稀疏连接问题。为提高基于采样的 GCN 训练的可扩展性，Chen 等人 [29]关注不依赖历史数据的 LS 方法（例如 FastGCN 和 LADIES），此类方法在构建采样概率时无需借助历史数据。他们从矩阵近似的角度重新审视这种方法，并解决两个关键问题：次优采样概率，以及现有 LS 方法中无放回采样所导致的估计偏差。

MQ-GNN 将这些先进的采样技术与全局邻居采样相结合，并定期缓存各小批次中频繁访问的节点。这减少了冗余数据访问，缓解了内存瓶颈，并在保持高准确率的同时加快了训练。它支持节点级采样和层级采样策略，无论采用何种采样方法，都能确保高效训练。

C. 流水线架构

流水线架构通过重叠数据传输与计算，有效提高了图形处理器利用率。Marius [30]和 MSPipe [27]分别展示了流水线化在图嵌入和时序 GNN 中的优势。然而，这些系统主要针对小型图或特定应用领域，因此可扩展性有限。DSP[26] 针对采用同步训练的多 GPU 配置优化了流水线执行，但其性能高度依赖图划分。Marius [30] 采用队列处理小批次，但不支持图神经网络训练，因为它专为非 GNN 图嵌入而设计。

通信开销是多 GPU 图神经网络训练中的一个重大瓶颈。GNNPipe [31] 和 PipeGCN [32]针对全批量训练场景解决了这一问题。GNNPipe 通过引入层级模型并行，将 GNN 层划分到多个图形处理器上，使通信量与层数成比例地减少。

每个图形处理器都会处理其负责层对应的整个图，并利用历史嵌入和专门的训练技术确保收敛。尽管 GNNPipe 降低了通信和训练开销，但它依赖于全图处理，因此限制了其在大型图上的可扩展性。它对层划分的依赖也降低了整合基于采样的方法的灵活性。PipeGCN 通过重叠通信与计算，降低了通信开销。相比之下，MQ-GNN 通过基于采样的框架消除了全图依赖。其流水线架构无需处理全图，便能最大限度地降低通信开销。MQ-GNN 支持节点级和逐层采样，以适应不同的数据集和计算配置。

MQ-GNN 提出了一种无需对图进行分区的基于多队列的流水线，用于大规模图神经网络训练。该方法交错执行小批量生成、数据传输和梯度共享，从而提高效率。为小批次和梯度设置的专用队列可减轻陈旧性，而周期性同步则确保一致性。该设计能够在稠密图和稀疏图上实现高效的图神经网络训练，解决数据传输、计算和 GPU 间通信中的瓶颈。

III. 预备知识与符号说明

本节介绍各种采样方法用于图神经网络训练时所需的符号和背景知识。

设 $\mathbf{G} = (V, E)$ 为一个图。节点集定义为 $V = \{v_i \mid i \in [n]\}$，其中 $[n] = \{0, 1, 2, \ldots, n-1\}$ 表示节点的索引集。边集表示为 $E = \{(v_i, v_j) \mid i, j \in [n]\}$。边 $(v_i, v_j) \in E$ 表示节点 v_i 与 v_j 之间的连接。每个节点 v_i 都有一个特征向量$f_i \in \mathbb{R}^d$，其中 d 为特征维度。所有节点特征集记为 $F = \{f_i \mid v_i \in V\}$。每个节点 v_i 都与一个标签 $y_i \in \mathcal{Y}$ 相关联，其中 $\mathcal{Y}$ 为可能的标签集。所有节点的标签集为 $\mathcal{Z} = \{y_i \mid v_i \in V\}$，其中$y_i$ 表示节点 v_i的标签。

图神经网络训练过程中可以采用多种采样方法。采样对于构建小批次至关重要，可提高训练吞吐量并加快模型收敛。这些采样方法基于图神经网络层和节点邻域，具体如下：

A. 逐节点采样（*NS*）

逐节点采样（NS）为每个节点选取一部分邻居，以构建更小且便于计算的子图用于训练。其主要目标是在保留图的局部邻域结构的同时，降低内存和计算开销。

图卷积网络 对于给定的图**G**，图卷积网络中的第 l个卷积层可定义为：

$$Z^l = PH^{l-1}W^{l-1}, H^{l-1} = \sigma(Z^{l-1}), \tag{1}$$

其中，L 表示层的总数，$l \in \{1, \ldots, L\}$ 表示层索引，σ 表示激活函数，H^{l-1} 表示第 $l-1$层的嵌入，Z^{l-1} 表示中间嵌入在第 $l-1$层，W^{l-1} 表示权重矩阵。P是归一化拉普拉斯矩阵，定义如下：

$$P = \widehat{D}^{-1/2}\widehat{A}\widehat{D}^{-1/2}, \widehat{D}_{i,j} = \sum_j \widehat{A}_{i,j}, \tag{2}$$

其中 $\widehat{D}$ 是一个对角矩阵，$\widehat{A}$ 表示对邻接矩阵 A进行归一化，即 $\widehat{A} = A + I$。

GraphSAGEGraphSAGE [2] 通过邻居采样控制图神经网络中的感受野大小。对于第 l层的每个节点，都会随机、均匀地采样固定数量的邻居，记为s_{node}。该方法构造无偏估计量 $\widehat{P}^{l-1}H^{l-1}$，以近似PH^{l-1} 图卷积层中的值，并优化计算和内存效率 [29]。

$$\widehat{P}^{l-1}_{i,j} = \begin{cases} \frac{|\mathcal{N}(v_i)|}{s_{node}} P_{i,j}, & \text{if } v_j \in \widehat{\mathcal{N}}^{l-1}(v_i) \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

其中，$\mathcal{N}(v_i)$ 和 $\widehat{\mathcal{N}}^{l-1}(v_i)$ 分别表示节点 v_i 在第 $(l-1)-th$ 层的完整邻居集合和采样邻居集合。

B. 层级采样（*LS*）

LS 不会为每个节点采样固定的邻居集合，而是在每一层选择一个全局定义的节点集合。与 NS 相比，LS 覆盖更广的感受野，并减少节点表示的变化。通过采用面向整层的视角，LS 在计算成本和信息覆盖之间取得平衡，因此非常适用于大型图以及需要稳健层级特征聚合的任务。不过，对于依赖保留细粒度局部邻域结构的任务，LS 的效果可能较差。

FastGCN FastGCN 不会为每个节点采样邻居，而是 [33]在层级选择节点，将图的每一层建模为作用于节点的嵌入函数，并将这些节点视为概率测度下的随机变量P。重要性采样概率决定训练期间节点的优先选择顺序，从而降低训练结果的方差。每个节点 i 被分配一个概率 $p_i \propto \|RP_i\|^2$ 用于 $i \in [n]$，其中 R 是行选择矩阵，并假设各项相互独立。对于给定层 l，r_l 抽取 i.i.d. 样本 $\{u_1^l, \ldots, u_{r_l}^l\} \sim P$，以近似该层所有节点的嵌入，具体形式如下 [5], [33]:

$$\begin{aligned} \widehat{z}^{l+1}_{r_{l+1}}(v) &= \frac{1}{r_l}\sum_{i=1}^{r_l} \widehat{A}\left(v, u_i^l\right) z^l_{r_l}\left(u_i^l\right) W^l, \\ h^{l+1}_{r_l+1}(v) &= \sigma\left(\widehat{z}^{l+1}_{r_{l+1}}(v)\right), \quad l = 0, 1, ..., L-1, \end{aligned} \tag{4}$$

其中，$h^{l+1}_{r_l+1}$ 表示第 $l+1$ 层的嵌入，$z^0_{r_0} = z^0$ 表示节点特征（f），$\widehat{A}(v, u_i^l)$ 对应于 $\widehat{A}$ 的 (v, u) 元素。带有函数 g 的损失函数可估计为：

$$\text{loss}_{r_0, r_2, \ldots, r_L} = \frac{1}{r_L}\sum_{i=1}^{r_L} g\left(z^L_{r_L}(u_i^{r_L})\right). \tag{5}$$

LADIES在不同层独立执行逐层采样可能效率低下，因为生成的二部图可能是稀疏的，甚至包含全零行。LADIES[28] 修改了独立的逐层采样方式，以便在 FastGCN 训练中构建计算图，从而解决这一问题。在其方法中，每一层都从先前采样节点的邻居并集中采样节点，如下所示

$$\mathcal{V}^{l-1} = \bigcup_{v_i \in \mathcal{S}_l} \mathcal{N}(v_i), \tag{6}$$

其中，$\mathcal{V}^{l-1}$ 表示第 $l-1$层的采样节点，$\mathcal{S}_l$ 是第 l 层采样的节点集合，$\mathcal{N}(v_i)$表示节点 v_i的邻居集合。因此，在采样过程中，仅为 $\mathcal{V}^{l-1}$中的节点分配概率，具体如下：$\{p_i^{l-1}\}_{v_i \in \mathcal{V}^{l-1}}$。具体而言，节点的选择概率定义如下：

$$p_i^{l-1} = \frac{\|R^l P_{*,i}\|_2^2}{\|R^l P\|_F^2}, \tag{7}$$

其中，R^l 是行选择矩阵。假设第 l 层和第 $l-2$层的采样节点集合已确定，且每个节点 v_i 被赋予概率 $p_1^{l-1} \ldots p_{|V|}^{l-1}$，则对角矩阵$S_{s\,s}^{l-1}$ 定义如下：

$$S_{s,s}^{l-1} = \begin{cases} \frac{1}{s_{l-1} p_{i_k^{l-1}}^{l-1}}, & s = i_k^{l-1}, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

这里，s_{l-1} 是第 $l-1$层的采样节点数量。由于在表征第 $1-th$ 层的样本时，缺乏有关中间嵌入或激活矩阵的信息，因此采用一种仅依赖行选择矩阵R^l 和归一化拉普拉斯矩阵 L的基本采样方案。行选择矩阵 R^l 可定义为：

$$R_{c,r}^l = \begin{cases} 1, & (c,r) = (c, i_c^l), \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

其中，i_c^l 表示 S_l处的节点集合。同时，L 定义为 $L = \widehat{D}^{-1/2}\widehat{A}\widehat{D}^{-1/2}$。最后，采样得到的训练小批次根据拉普拉斯邻接矩阵及相应的嵌入（或第 0 层的特征）定义为$\frac{1}{p_k^{l-1}} \widehat{L}_{*,\,i}$和$\widehat{Z}_{k\,*}^{l-1}$，其中 $\widehat{L} \leftarrow D_{\widehat{p}^l}^{l-1} \widehat{p}^l$ 为归一化拉普拉斯矩阵，可避免梯度消失和梯度爆炸。

校准并消除层级采样的偏差层级采样的效率在很大程度上取决于采样过程的重要性：该过程在保留准确率的同时，使用显著更少的节点来估计节点聚合。采样概率的选择至关重要，因为它会直接影响图卷积网络的准确率[29]。为提高基于采样的 GCN 训练的可扩展性，Chen 等人 [29] 研究了不依赖历史数据的逐层采样技术，例如 FastGCN 和 LADIES，这些方法在构建采样概率时无需依赖历史数据。通过从矩阵近似的角度分析这些方法，他们发现并解决了两个关键问题：次优采样概率，以及无放回采样导致的估计偏差。

$\|HW_i\|$ 与 $\|P^i\|$ 之间较弱或负相关（其中W_i 为权重矩阵），凸显了 FastGCN 和 LADIES 采样概率中比例假设的局限性。为解决这一局限，作者认识到在历史无关设置中，关于 HW 的先验知识受到限制。受最大熵原理指导，他们假设$\|HW_i\|$服从均匀分布。基于这一假设，他们推导出以下采样概率，称为“扁平”采样 [29]:

$$p_i \propto \|HW_i\|, \quad \forall i \in [n]. \tag{10}$$

通过将目标矩阵乘积重新表述为 $RPI(HW)$（其中 RPI 为近似值），可以调整采样概率，以平衡方差和准确率。假设 HW 中各行的范数服从均匀分布，可以同时提高矩阵近似的稳定性和图卷积网络的预测准确率，从而提供更加稳健可靠的采样框架。采用这种方法的 LADIES 和 FastGCN 高级变体分别称为 LADIES+ flat（或 LADIES+f）和 FastGCN+flat（或 Fast-GCN+ f） [29]。

尽管（10）中的概率能够提升性能，但它们假设邻居节点采用有放回采样，以保持 GCN 嵌入无偏。然而，在实际实现中，采样通常采用无放回采样，由于估计量是针对有放回采样设计的，因此会引入偏差。为提高准确率并减轻这种偏差，研究人员提出了一种去偏方法。使用 s 个采样索引后，FastGCN/LADIES 利用以下重要性采样估计量来近似目标矩阵之和：

$$\frac{1}{|s_{node}|} \sum_{i \in [n]} \frac{X_{s_{node_i}}}{p_{s_{node_i}}}, \quad s_{node} = \{1, 2, \ldots, s\}, \tag{11}$$

$$X = RP(HW). \tag{12}$$

该估计量表示$RP(HW)$的加权平均，而在无放回采样时会引入偏差。为解决这一问题，作者旨在去偏过程中保留线性形式 $Y_s = \sum_{i=1}^{s} \beta_i X_{s_{\text{node}}}$，并为每个 $X_{s_{\text{node}}}$ 设计新的系数 β_i，以确保 Y_s 保持无偏。具体而言，去偏通过递归加权平均实现，如下所示：

$$Y_0 := 0, \quad Y_{i+1} = (1 - \alpha_{i+1}) Y_i + \alpha_{i+1} \Pi_{i+1},$$
$$\text{where } \Pi_{i+1} = \sum_{j \in S_i} X_i + \frac{X_{s_{node_{i+1}}}}{p^i_{s_{node_{i+1}}}}, \quad \forall i = 0, 1, \ldots, |s_{node}| - 1. \tag{13}$$

这里，S_i 表示通过加权随机采样 [34], 得到的索引集合，其中 $0 \le i \le |s_{node}| - 1$且 $S_0 = \emptyset$。$\alpha_1 = 1$，而 α_{k+1} 是依赖于 k的常数。具体而言，选择 $\alpha_{k+1} = \sqrt{(n-k)(k+1)}$，以确保当所有 $p_i = 1/n$ 时，输出系数与简单随机采样情形下的系数一致。原论文将此方法称为“去偏”。采用该方法的 LADIES 和 FastGCN 变体分别为 LADIES+去偏 (LADIES+d) 和 FastGCN+去偏 (FastGCN+d)。此外，同时采用平坦采样 (10) 和去偏 (13) 的采样方法分别称为 LADIES+ 平坦 + 去偏 (LADIES+f+d) 和 FastGCN+平坦+去偏 (FastGCN+f+d)。

C. 特征缓存

为了加速训练，全局邻居采样（GNS）[18] 通过定期构建节点缓存（记为 $\mathcal{C}$，其中 $\mathcal{C} = \{c_i \mid i \in \mathcal{V}_c \subset V\}$。其中的节点$\mathcal{C}$ 采用有偏采样策略进行选择，以确保它们有较高概率能够从训练集中的节点到达。在训练期间，$\mathcal{C}$ 中的节点特征会预加载到图形处理器上。为了满足 GPU 内存限制，只缓存训练集中的节点有较高概率能够到达的节点中的 1%。这一策略通过缓存少量高阶节点，有效减少数据移动开销并加快训练。GNS 定义了两种确定缓存采样概率的方法。如果图中的大多数节点都属于训练集，则根据节点度确定采样概率。对于节点 i，其被采样到缓存中的概率为：

$$p_{v_i} = \frac{\deg^-(v_i)}{\sum_{v_j \in V} \deg^-(v_j)}, \tag{14}$$

其中，$\deg^-(v_i)$ 表示节点 v_i 的入度。在幂律图中，由于度分布高度偏斜，只需缓存少量节点即可覆盖大多数节点。当训练集仅包含图中一小部分节点时，GNS 会执行短随机游走来计算采样概率。对于采样得到的节点 $v \in V$ 的邻居 $\mathcal{N}_v$，采样率 $\mathbf{d}$ 定义如下：

$$\mathbf{d} = \left\{ \frac{\mathcal{N}(v_1)}{\deg^-(v_1)}, \frac{\mathcal{N}(v_2)}{\deg^-(v_2)}, \ldots, \frac{\mathcal{N}(v_{|V|})}{\deg^-(v_{|V|})} \right\}. \tag{15}$$

$P^{(l)} \in \mathbb{R}^{|V|}$ $l-th$层的节点采样概率计算如下：

$$P^{(l)} = (\mathbf{D}A + I)\, P^{(l-1)}, \quad \mathbf{D} = \operatorname{diag}(\mathbf{d}).$$
$$p_{v_i}^{(0)} = \begin{cases} \frac{1}{|V_t|}, & \text{if } v_i \in V_t, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

其中，V_t 表示训练集。缓存的最终采样概率为 $P^{(L)}$。

总之，节点级采样和层级采样等技术及其变体（如扁平采样和去偏 LS）有助于缓解大规模图神经网络训练中的计算和内存限制。这些方法在可扩展性和准确率之间进行权衡，同时也带来了采样偏差和计算开销增加等挑战。节点缓存策略则通过全局邻居采样，将频繁访问的节点预加载到 GPU 内存中，进一步提升效率。

IV. 多队列流水线式 GNN 训练架构

本节介绍所提出的 MQ-GNN 架构，用于训练图神经网络模型。首先，我们概述系统设计，并详细分析各个训练阶段。

总体设计 MQ-GNN 引入了一种流水线架构，以优化图神经网络训练期间中央处理器、内存和图形处理器的利用率。该架构采用小批次队列、梯度队列和周期性同步，以应对多 GPU 环境中的可扩展性和效率挑战。如图1所示，MQ-GNN 的流水线架构促进了小批量生成、数据传输和计算之间的平稳数据流。该架构由七个阶段组成：三个专用于计算，三个专用于数据传输，另一个则将计算和数据传输结合在一起。该架构在各阶段使用多个进程和线程，以最大限度地提高计算重叠程度并支持异步处理。为有效管理数据流，我们引入了三个专用队列：两个用于小批次，一个用于梯度。每个队列都由并行进程和线程填充。

下面将详细介绍 MQ-GNN 流水线中的七个不同阶段：

数据加载 在初始阶段，系统将图的拓扑结构和特征加载到主存中，为后续的图神经网络训练处理做好准备。

模型传输首先，初始化模型并将其传输至图形处理器进行训练。这确保所有图形处理器都具有相同的模型参数，从而支持设备间的异步处理。

小批量生成与入队在此阶段，通过采样生成训练批次，并将其加入中央处理器上的专用小批量队列，每个图形处理器各分配一个队列。这种并行化设计减少了瓶颈，并实现了向图形处理器高效传输数据。

高效的小批量生成与管理对于提高训练吞吐量至关重要，正如第III 节所述。为此，MQ-GNN采用节点级采样方法，例如图卷积网络 [35] 和GraphSAGE [2], , 以及层级采样方法，包括FastGCN [5], LADIES [4], 及其先进变体 [29], , 相关内容均在第III节讨论。

小批量传输至 GPU 生成的小批次会在传输阶段传输至图形处理器上的小批次队列。图1 展示了

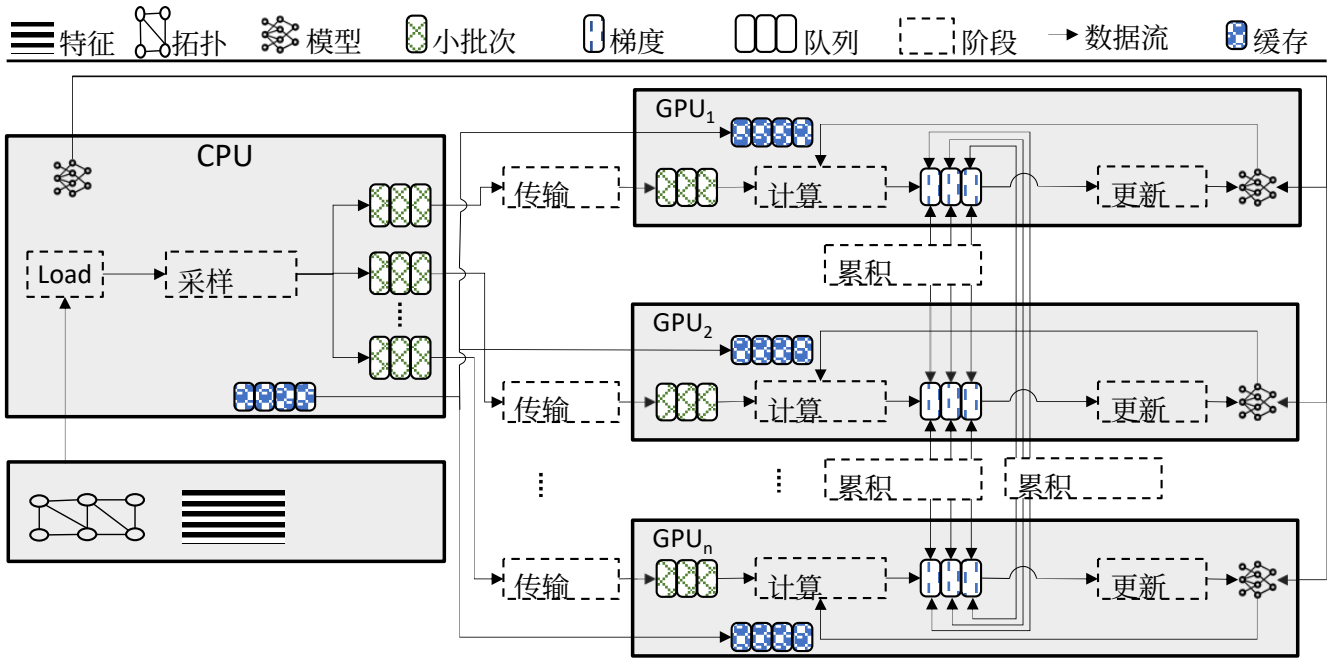


图1。MQ-GNN 架构采用多队列流水线，使小批量生成、数据传输、计算、异步梯度共享以及模型更新能够在中央处理器和图形处理器之间重叠进行。该设计最大限度地减少延迟、提高 GPU 利用率，并加速大规模图神经网络训练。

小批次生成与传输的流水线，突出展示了计算与数据传输如何重叠进行。

中央处理器会提取 GPU 缓存中不存在的节点的特征。缺失的特征从中央处理器的主存中收集，并通过 PCIe 传输到图形处理器，这可能会消耗大量资源。此阶段的开销取决于 GPU 缓存大小、缓存策略以及度分布等图属性。高效或完整的 GPU 缓存可以最大限度地减少或消除这项成本，从而优化数据传输和内存使用量。此外，由于图形处理器上的每个小批次都包含从中央处理器传输而来的特征和拓扑组件，因此在图形处理器上收集目标节点的邻居特征时，无需在多个图形处理器之间进行同步。

计算图形处理器上运行计算 阶段，处理从传输阶段加入队列的小批次。此阶段专注于处理小批次并执行前向（Fwd）和反向（Bwd）传播，以与前两个阶段并行训练图神经网络。计算 阶段专注于最大限度地提高图形处理器的利用率，并优化计算操作：执行前向和反向操作时，无需等待其他图形处理器的模型更新。工作线程通过执行前向传播来处理小批次，以如下方式训练图神经网络：

$$Z^{l+1}(v) = GNN^l\left(\left[Z^l_{agg}(v), Z^l(v)\right], W^l\right), \quad Z^l_{agg}(v) = Mean\left(H^l(v)\right), \tag{17}$$

其中， H^l 表示第l层的中间嵌入， Z^l 表示第 l 层的嵌入， W^l 表示权重矩阵，而 $\left[Z^l_{agg}(v)\ Z^l(v)\right]$ 表示 GraphSAGE 的聚合邻域表示与节点 v 在第 l 层的先前嵌入的拼接。

对于图卷积网络， $Z^{l+1}(v)$经过修改，将拼接替换为直接邻域聚合，如下所示：

$$Z^{l+1}(v) = GNN^l\left(Z^l_{agg}(v), W^l\right) \tag{18}$$

由于前向传播包含两个主要步骤：收集和聚合邻居特征（收集，GA），以及应用神经网络（应用神经网络，AN） [36], [37]。从矩阵角度来看，GA 对应于 AH 矩阵乘法，而 AN 对应于$(AH)W$。

批次 s、 $\mathcal{L}^s$ 的损失计算如下：

$$\mathcal{L}^s = -\sum_{v \in V^s}\left[y_v \log(pr_v) + (1-y_v)\log(1-pr_v)\right], \tag{19}$$

其中， V^s 表示 s中的节点集合， y_v 表示真实标签， pr_v 表示节点 v的预测概率。

在反向传播过程中，GA 步骤的梯度计算（记为 ∇ GA）涉及 AG运算，其中 G 表示梯度矩阵。GA 和 ∇ GA 均不需要图形处理器之间的同步。

梯度计算完成后，梯度会被加入相应的梯度队列，而不是立即在各图形处理器之间同步。如图2所示，图形处理器 $\mathcal{G}_0$ 在时间 τ_1 将梯度加入队列，并继续处理另一个小批次；与此同时，其余图形处理器（$\mathcal{G}_i$， $i \neq 0$）仍在完成第一次迭代。这种流水线执行有助于实现最佳资源利用率并

实现操作重叠。在 MQ-GNN 中，梯度队列用于存放等待同步的梯度，而小批次队列则保存供图形处理器处理的训练数据。这一区分确保了流水线各阶段之间界限分明。

跨图形处理器的梯度共享与累积MQ-GNN采用异步梯度共享机制，在多个图形处理器之间累积梯度，以平衡更新。此过程可确保本地计算不间断地进行，从而降低通信延迟的影响。

梯度入队后，会在所有图形处理器之间异步共享，确保及时分发，同时不中断正在进行的计算。收到延迟的梯度后，图形处理器会使用以下公式更新其队列：

$$\Delta^{\text{New}} = \Delta^{\text{Current}} + \frac{\left(\Delta^{\mathcal{G}_i} - \Delta^{\text{Current}}\right)}{\text{Average Count}}, \tag{20}$$

其中，$\Delta^{\mathcal{G}_i}$ 是来自图形处理器 $\mathcal{G}_i$的输入梯度，Δ^{Current}是现有梯度，而“Average Count”用于跟踪收到的梯度数量（即“Average Count” $\leq |\mathcal{G}|$）。这确保了即使存在通信延迟，更新仍能保持平衡。

图2 展示了梯度共享与累积过程。图形处理器$\mathcal{G}_1$ 在图形处理器 $\mathcal{G}_0$ 期间 τ_2入队并共享梯度，同时处理下一次迭代的数据。这种设计确保了高效执行，避免了同步或资源利用不足造成的延迟。

每个图形处理器上的模型更新收到所有图形处理器的梯度后（即“Average Count” $= |\mathcal{G}|$），按如下方式计算累积梯度：

$$\Delta_l^{acc} = \frac{\sum_{i=1}^{|\mathcal{G}|} \Delta^{\mathcal{G}i}}{|\mathcal{G}|}, \tag{21}$$

其中，Δ_l^{acc}是第 l层的累积梯度，$\Delta_l^{\mathcal{G}i}$ 是图形处理器 $\mathcal{G}_i$上第 l 层计算出的批次特定梯度。该聚合操作汇总了处理不同小批次的所有图形处理器所作出的贡献。更新阶段将累积梯度应用于图形处理器上的模型参数更新。此阶段确保在多图形处理器环境中所有图形处理器上的模型参数均得到更新，并整合累积梯度的集体贡献。

完成当前迭代后，每个图形处理器都会在进入下一次迭代之前，将累积梯度纳入其模型更新中。第 l 层的模型更新规则定义如下：

$$W^l = W^l - \eta \Delta_l^{acc}, \tag{22}$$

其中，W^l 表示权重矩阵，η 表示学习率。尽管梯度共享有助于实现图形处理器之间的高效通信，但异步更新可能导致模型陈旧性。为缓解这一问题，我们引入了就绪更新异步一致性模型（RaCoM），这是一种动态同步机制，可确保图形处理器之间的模型一致性。

RaCoM RaCoM 通过将异步本地更新与周期性同步相结合，解决了异步图神经网络训练中的陈旧性问题，同时确保全局一致性和计算效率。周期性同步机制会根据数据集的稀疏性或稠密性进行调整，使更新保持一致，从而有效平衡陈旧性与通信成本。该设计确保图神经网络模型在不同工作负载强度下仍能实现可扩展性，同时保持准确率。同步周期 P_{ms} 为：

$$P_{ms} = \left\lceil \frac{\sqrt{|V|}}{\sqrt{|\mathcal{G}|\,|E|}} \right\rceil, \tag{23}$$

其中，$|V|$ 和 $|E|$ 分别表示图中节点和边的数量，而 $|\mathcal{G}|$ 表示图形处理器的数量。同步周期 P_{ms} 在确保及时更新的同时，最大限度地降低通信开销。这种平衡使 RaCoM 能够高效扩展，以处理大型图和大量图形处理器，同时仅造成模型准确率的轻微下降。在 MQ-GNN 中，由（23）管理的周期性同步会动态平衡同步成本与陈旧性。MQ-GNN 根据图的结构调整同步周期P_{ms} ，从而降低异步更新对训练效率的影响。这确保了在维持模型一致性的同时，充分利用各图形处理器上的计算资源。周期性同步机制及其推导的详细证明见附录A。这些推导说明了P_{ms} 如何动态适应图属性，以最大限度地降低同步成本。

图 2 展示了 MQ-GNN 提出的梯度共享机制和周期性同步机制。图中描绘了梯度共享过程，重点说明工作线程如何重叠执行梯度入队和计算任务。梯度队列用于简化图形处理器之间的梯度通信，从而支持异步更新和计算重叠。图中还展示了周期性同步如何对齐各图形处理器上的梯度，以保持一致性。图形处理器$\mathcal{G}_i$ 在迭代 t 生成的每个梯度$\Delta_t^{\mathcal{G}_i}$都会先进入其本地梯度队列，然后再传播到所有其他图形处理器（$\mathcal{G}_j$、 $j \neq i$）。梯度$\Delta_t^{\mathcal{G}_i}$在到达 $\mathcal{G}_j$之前会经历延迟$r_{\mathcal{G}_i,\,\mathcal{G}_j}^t$。例如，拖尾图形处理器（$\mathcal{G}_n$）在梯度$\Delta_t^{\mathcal{G}_n}$入队后立即将其共享出去，但该梯度经过延迟$r_{\mathcal{G}_j,\,\mathcal{G}_n}^t$后才到达其他图形处理器（$\mathcal{G}_j$、 $j \neq n$）。当来自拖尾图形处理器的梯度到达接收图形处理器时，会使用公式（20）对其进行累积。一旦梯度（即公式（21））准备好用于更新模型，接收图形处理器便完成当前迭代，将更新后的梯度纳入模型更新（22），并进入下一次迭代。最后，经过 P_{ms} 次迭代后，模型同步确保所有图形处理器保持一致，从而维持模型准确率。

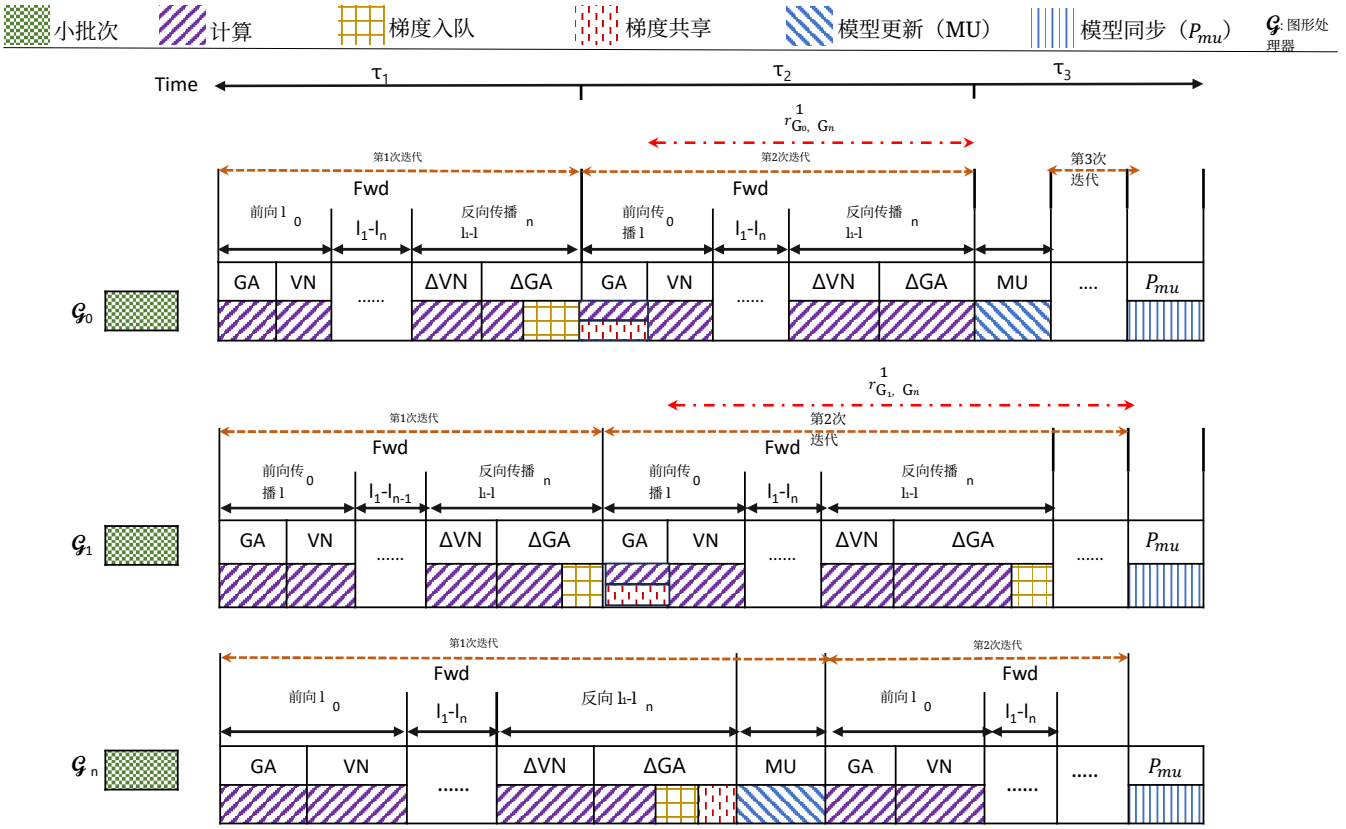


图 2。 MQ-GNN 中的 RaCoM 流水线支持在图形处理器之间进行异步梯度共享、模型更新和周期性同步。它能够高效地重叠小批次生成、数据传输、前向传播与反向传播以及模型同步 P_{ms}，从而最大限度地提高资源利用率并最小化训练延迟。

A. 确定最佳队列大小

在大规模数据集上训练图神经网络时，确定最优队列大小对于充分利用图形处理器至关重要。实际应用中，需要在计算效率与内存限制之间取得平衡，以确保高吞吐量并避免溢出。队列大小根据批处理时间和图形处理器可用内存计算得出。这种混合方法会动态调整队列大小，在维持 GPU 利用率的同时遵守内存限制。

队列大小由以下公式给出：

$$\mathcal{Q} = \min\left(C, \max\left(2, \left\lceil \frac{\max(T_{\text{sampling}} + T_{\text{transfer}})}{\text{mean}(T_{\text{compute}})} \right\rceil\right)\right) \tag{24}$$

其中，$\max(T_{\text{sampling}} + T_{\text{transfer}})$ 表示采样和数据传输所需的最大时间。相比之下，$\text{mean}(T_{\text{compute}})$ 表示图形处理器上的平均计算时间。上限大小 C 为队列大小设置了上限，以防超过图形处理器的内存容量。峰值内存使用量考虑了 CUDA 计算期间产生的临时内存开销。C 定义如下：

$$C = \left\lfloor \frac{\mathcal{M}_{\mathcal{Q}}}{\mathcal{M}_{\text{mini-batch}}} \right\rfloor \tag{25}$$

其中，$\mathcal{M}_{\mathcal{Q}}$ 是队列的可用内存，$\mathcal{M}_{\text{mini-batch}}$ 是每个小批次的内存。$\mathcal{M}_{\mathcal{Q}}$ 的计算方式是从图形处理器的总内存中减去测得的峰值内存使用量（包括 5–10% 的安全余量）。每个批次的内存取决于批量大小、特征维度以及所使用的数据类型（例如，float32 占用 4 字节）。这可确保队列大小不超过图形处理器的内存容量，同时保持效率。

峰值内存使用量可以通过使用 PyTorch Profiler 或 NVIDIA Nsight Systems 等性能分析工具运行单个训练轮次来估算。这些工具能够提供有关内存分配以及 CUDA 计算临时开销的有用信息。根据该峰值内存值，可以计算峰值容量，并将其纳入队列大小公式中。

V. 评估

本节通过将 MQ-GNN 架构与基线模型进行比较，对其进行了全面评估。

A. 实验设置

本节介绍实验所用的硬件配置。

硬件配置 实验在由 SLURM 工作负载管理器管理的 Linux 集群上进行。

表 1。 数据集概览。每条无向边计数一次。“Deg.”表示图的平均度。训练、验证和测试数据的比例称为“划分比例”。

数据集	节点	边	平均度	特征维度	类别	划分比例	度量指标
ogbn-proteins	132,534	39,561,252	597	8	2	65/16/19	ROC-AUC
ogbn-arxiv	160,343	1,166,243	13.7	128	40	54/18/28	准确率
Reddit	232965	11606919	50	602	41	66/10/24	F1 分数
ogbn-products	2,449,029	61,859,140	50.5	100	47	1990年 8月2日	准确率

本工作的硬件配置由 SLURM 集群账户允许的最大资源分配决定。每个作业最多分配 4 个图形处理器、32 个中央处理器和 126 GB 的随机存取存储器，最长运行时间为 4 天。

实验在一台运行 Ubuntu 20.04.6 LTS（内核 5.4.0-192-generic）的机器上进行，该机器配备四块 NVIDIA GeForce RTX 3090 图形处理器（每块 24 GB）和 32 个 Intel Xeon E-2234 CPU（3.60 GHz，每个插槽四个核心，每个核心两个线程）。为充分利用这些资源，我们针对单块、两块、三块和四块图形处理器配置，优化了中央处理器、图形处理器和内存的使用。

- **使用单块图形处理器进行实验**每项实验使用 32 个 CPU 和 128 GB 内存。
- **使用两块图形处理器进行实验** 每项实验为每块图形处理器配备 16 个 CPU 和 64GB 内存。
- **使用三块或四块图形处理器进行实验** 每项实验为每块图形处理器配备 8 个 CPU 和 32 GB 内存。

通过在不同图形处理器配置下进行扩展，可以评估 MQ-GNN 在不同并行性和资源竞争水平下的性能与可扩展性。

数据集与评估指标为确保比较公平且结果具有代表性，我们在四个广泛使用的基准数据集上进行了实验：Reddit [2] 以及开放图基准（OGB）的三个数据集——ogbn-arxiv、ogbn-proteins 和 ogbn-products [38]。表 1 总结了这些数据集。这些数据集具有不同的特性。Reddit 数据集在以往研究中被广泛使用[5], [15], , 其中包含高阶节点和复杂的社区结构，因此是评估可扩展性的标准基准。该数据集的平均节点度较高、特征维度较大且包含稠密子图，因而需要高效的采样策略。OGB 数据集则带来了其他挑战，包括 ogbn-proteins 中的类别不平衡，以及 ogbn-products 中的大量节点，这些特性可用于检验可扩展性和准确率 [38]。这些数据集共同构成了一个全面的测试平台，可用于评估 MQ-GNN 在不同图结构下的可扩展性、效率和泛化能力。

如表1所示，我们针对每个数据集采用了不同的评估指标，以与近期文献保持一致，并遵循 [38] 的建议，确保与基线方法兼容。

模型和超参数 为确保比较公平，所有基线模型均使用 PyTorch 和 DGL 实现，并在所有方法中保持训练参数一致。对于逐节点采样，每个节点选取五个邻居；对于逐层采样，每层选取 512 个节点。所有模型均使用 ADAM优化器，以学习率 $\eta = 0.001$ 训练两层图神经网络。我们采用缓存策略，每个训练周期定期存储 1% 的节点，以提高效率。如果验证指标连续 200 个批次未至少提升 0.01，则提前停止训练。选择验证指标最高的模型作为最终模型。为确保统计稳健性，每种方法和每个数据集上的模型均训练十次，并报告评估的均值和方差。为降低内核初始化和同步开销导致的图形处理器计时波动，我们排除了前后各 20 个小批次的计时结果。这样可确保报告的计时反映稳态性能，减少预热和冷却阶段带来的噪声。

B. 与现有系统的比较

为证明 MQ-GNN 能比现有系统更有效地优化资源利用率，从而加快训练速度，我们将其与一系列基线模型进行比较。这些模型包括节点级采样方法，例如小批量 GCN 和 GraphSAGE [21], , 以及基于层级采样的方法，包括 FastGCN [5], 、LADIES 及其改进变体 [29]。选择这些基线方法，是因为它们在以往研究中具有重要地位，且能够有效代表当前最先进的基于节点级采样和层级采样的图神经网络训练方法。

评估结果汇总于表2 至13，报告了每个数据集的批次时间和训练时间的均值（± 标准差）（单位为毫秒，ms），以及评估指标百分比。表1 总结了数据集的特征。

在详细讨论之前，我们先重点介绍结果中的关键见解。

主要亮点 MQ-GNN 架构通过优化资源利用并降低图神经网络训练开销，显著提升了性能。在多个数据集上，MQ-GNN 的训练速度最高可比基线模型快 $4.6\times$ 倍，同时保持相当的评估指标。这一加速主要得益于 MQ-GNN 高效的多队列设计，该设计能够在多个图形处理器上实现计算、数据传输和梯度共享的并发执行，以及定期进行模型同步。此外，MQ-GNN 能够有效适配各种数据集和配置，在不同的 GPU 配置、采样策略和节点缓存大小下均能持续提升性能。这些结果凸显了 MQ-GNN 在大规模图神经网络训练中的可扩展性、效率和有效性。

表 2。 在单 GPU 配置下，GCN、GraphSAGE、MQ-GCN 和 MQ-GraphSAGE 的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）结果。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
GCN	批次	10.29 ± 0.20	11.53 ± 0.15	24.70 ± 0.18	14.11 ± 0.24
	训练	351669.12 ± 3616.96	505270.1 ± 4952.69	**1471507.1 ± 15604.08**	1081781.22 ± 21635.62
	指标	66.31 ± 0.50	67.69 ± 0.55	88.81 ± 0.80	76.10 ± 0.48
MQ-GCN	批次	**6.43 ± 0.15**	**6.50 ± 0.20**	11.89 ± 0.18	6.83 ± 0.22
	训练	**197371.73 ± 1953.72**	**280705.61 ± 2807.06**	**731104.99 ± 8175.05**	559839.94 ± 6011.34
	指标	66.87 ± 0.35	67.25 ± 0.40	89.71 ± 0.55	76.95 ± 0.40
GraphSAGE	批次	23.03 ± 0.25	28.11 ± 0.30	38.01 ± 0.35	21.23 ± 0.40
	训练	786946.56 ± 6993.38	1050783.26 ± 12418.31	2326703.17 ± 22372.31	**1725299.42 ± 25712.47**
	指标	66.01 ± 0.45	68.51 ± 0.55	94.36 ± 0.25	76.87 ± 0.65
MQ-GraphSAGE	批次	15.99 ± 0.20	17.99 ± 0.25	23.36 ± 0.30	12.27 ± 0.25
	训练	536981.42 ± 4360.81	639797.48 ± 5837.68	1351612.87 ± 12926.13	**958494.12 ± 9584.94**
	指标	66.11 ± 0.35	68.41 ± 0.40	94.41 ± 0.30	77.01 ± 0.40

C. 结果与讨论

本节评估 MQ-GNN 在各种图神经网络模型和标准数据集上的性能。具体而言，我们利用 NS 和 LS 采样方法，分析其在单 GPU 和多 GPU 配置下的性能。

1) 单 GPU 性能

MQ-GNN 在单 GPU 配置下展现出显著的性能提升，训练和批处理时间明显短于基线方法。MQ-GNN 在四个数据集上将训练时间最多缩短了 2.06 × ，同时保持了相近的评估指标。

NS 性能 MQ-GraphSAGE 等 NS 方法展现出显著的性能提升。对于 Reddit 这种图结构稠密且包含许多高阶节点的数据集，MQ-GCN 实现了 2.01 × 的加速，如表 2所示，这证明了 MQ-GNN 的排队机制能够高效管理大特征尺寸和稠密邻域下的数据传输与计算。同样，在 ogbn-products 数据集上，MQ-GraphSAGE 将训练时间缩短了 1.81 × ，并将批处理时间缩短了 1.98 ×。MQ-GNN 在 Reddit 数据集上取得了显著的性能提升，这得益于其稠密结构和较大的特征维度。较大的特征维度和稠密连接使排队机制能够交错执行数据传输与计算，从而有效提高 GPU 利用率。相比之下，

ogbn-products 的稠密结构有利于采用排队机制，但其较大的图规模增加了数据传输需求。尽管较小的特征尺寸带来了一定的缓存收益，但其影响不如 Reddit 显著，因为 Reddit 的特征维度更大。

即使面对 ogbn-proteins 这类较小的图，以及 ogbn-arxiv 这类稀疏数据集，MQ-GCN 仍能保持高效。表 2 显示，MQ-GCN 通过利用 ogbn-arxiv 适中的图密度和较大的特征维度（128），实现了 1.5 × 的加速。由于特征维度更大，MQ-GNN 的缓存方法通过减少冗余数据传输，并有效地将数据传输与 GNN计算交错进行，从而提高了图形处理器的效率。相比之下，在 ogbn-proteins 数据集上，MQ-GCN 将训练时间缩短了 1.4 × 。较小的特征维度（8）限-

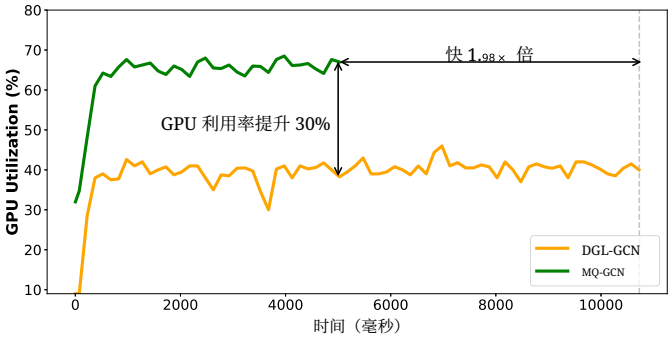


图3。 在 Reddit 上进行单个训练轮次期间，MQ-GCN 和 DGL-GCN 的图形处理器利用率。利用率按 150 毫秒进行平滑处理。

制了缓存的效果，也降低了数据传输与 GNN计算交错进行的效率，因此，与 ogbn-arxiv 相比，带来的收益相对较小。

层级采样性能 层级采样方法得益于 MQ-GNN 的流水线化机制。对于 ogbn-products，MQ- FastGCN+f+d 实现了 2.06 × 训练时间的缩减（表 3），展现了其利用多队列和缓存机制处理该数据集的大规模与高特征维度的能力。对于 ogbn-proteins 等较小的图，MQ-FastGCN+f+d 将训练时间缩短了1.6 ×，而 MQ-LADIES+f+d 将其缩短了 1.7 × （表 3和 4），展现了 MQ-GNN 在不同数据集规模和图结构下的通用性。MQ-GNN 的队列方法通过增加计算与数据传输之间的重叠，减少了图形处理器空闲时间并提高了利用率。得益于缓存机制能够减少由于图连接稠密而产生的冗余节点访问，小批次准备得以加速。

比较分析在所有数据集以及层级采样和节点级采样策略下，MQ-GNN 始终优于基线方法，在不影响指标的前提下实现了更快的训练时间和批处理速度。将采样与训练分离，消除了传统系统中的瓶颈，例如 LADIES 中的采样分布更新（第III-B节）会拖慢小批次准备。MQ-GNN 的排队机制使数据传输、采样和计算阶段相互重叠，从而确保图形处理器持续得到利用——

表3. 单 GPU 配置下 FastGCN、MQ-FastGCN 及其增强变体的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）结果。

模型	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
FastGCN	批次	8.99 ± 0.20	8.84 ± 0.25	9.75 ± 0.30	10.09 ± 0.50
	训练	307300.5 ± 3073.01	316598.2 ± 26256.19	588347.7 ± 4975.32	400561.65 ± 7831.35
	指标	53.2 ± 0.50	26.11 ± 0.40	45.7 ± 0.55	27.3 ± 0.55
MQ-FastGCN	批次	5.42 ± 0.18	5.11 ± 0.20	5.39 ± 0.22	5.01 ± 0.25
	训练	180722.50 ± 1707.23	173888.99 ± 1758.89	298911.13 ± 3268.60	200101.12 ± 2225.34
	指标	53.1 ± 0.45	26.21 ± 0.40	45.9 ± 0.50	27.1 ± 0.50
FastGCN+f	批次	8.57 ± 0.15	8.85 ± 0.18	9.18 ± 0.20	10.07 ± 0.45
	训练	292837.5 ± 2839.29	316811.5 ± 3057.12	553561.12 ± 5535.61	390916.17 ± 6929.41
	指标	61.1 ± 0.45	23.61 ± 0.35	53.61 ± 0.49	33.22 ± 0.51
MQ-FastGCN+f	批次	5.36 ± 0.15	5.23 ± 0.18	5.74 ± 0.20	6.29 ± 0.22
	训练	162687.50 ± 1626.88	176006.39 ± 1760.06	307534.00 ± 3075.34	217175.65 ± 2171.76
	指标	61.12 ± 0.30	23.13 ± 0.25	54.11 ± 0.40	33.21 ± 0.35
FastGCN+d	批次	8.52 ± 0.15	8.65 ± 0.18	9.31 ± 0.25	9.97 ± 0.50
	训练	291145.5 ± 2844.56	309764.5 ± 3284.55	561751.15 ± 4727.52	386924.19 ± 6736.37
	指标	53.69 ± 0.51	27.32 ± 0.65	44.25 ± 0.70	27.97 ± 0.65
MQ-FastGCN+d	批次	5.23 ± 0.18	5.01 ± 0.20	5.12 ± 0.22	5.23 ± 0.25
	训练	181747.50 ± 1617.48	171091.39 ± 1720.91	310028.42 ± 3110.28	202957.88 ± 2149.58
	指标	53.75 ± 0.51	27.25 ± 0.65	44.66 ± 0.70	27.84 ± 0.65
FastGCN+f+d	批次	8.65 ± 0.20	8.42 ± 0.18	9.47 ± 0.22	20.22 ± 0.60
	训练	**295582.1 ± 3023.79**	301607.1 ± 3161.71	571206.23 ± 4821.12	**784496.26 ± 15689.93**
	指标	61.1 ± 0.75	24.79 ± 0.70	53.97 ± 0.85	30.11 ± 1.50
MQ-FastGCN+f+d	批次	5.39 ± 0.20	5.27 ± 0.22	5.94 ± 0.25	6.12 ± 0.30
	训练	**163400.35 ± 1634.00**	170154.30 ± 1701.54	297258.84 ± 2972.59	**378999.76 ± 3790.00**
	指标	61.13 ± 0.85	24.85 ± 0.70	53.99 ± 0.85	29.97 ± 1.50

此外，MQ-GNN 有效地将去偏（13）和 LADIES 中的平坦采样（10）+f+d 以及 FastGCN+f+d 等计算开销较高的流程整合到其流水线中，从而最大限度地降低它们对运行时间的影响。

如图所示，我们还比较了使用 MQ-GCN 和 GCN 在 Reddit 数据集的单个训练轮次中 GPU 的利用率。3，其中 MQ-GNN 实现了 $1.98\times$ 的加速比，并使 GPU 利用率提高了 30% 。MQ-GNN 的 GPU 利用率显著更高，平均利用率约为 64.2%，峰值利用率达到 73%；相比之下，DGL 的利用率稳定在约 39.42%。影响 GPU 利用率的一个关键因素是确定最佳队列大小，并交错进行数据传输和 GNN 计算，以减少 GPU 因等待数据而产生的饥饿。

在 MQ-GNN 中，队列大小直接影响批次准备与 GPU 计算之间的重叠。我们的分析表明，采样和数据传输耗时较长的数据集（如 Reddit）需要更大的队列大小，才能充分利用 GPU。相反，批次准备时间变化较小、规模较小的数据集（如 ogbn-arxiv），使用较小的队列大小可获得更好的性能。

MQ-GNN 通过分析工作负载特征并应用一个考虑最大准备时间、平均计算时间和内存限制的公式（24），动态调整队列大小，以在避免内存溢出的同时最大化吞吐量。例如，在 Reddit 数据集中，峰值内存使用量为 730.95 MB，最大采样与数据传输总时间为 51 ms，此时队列大小设为 5，可确保 GPU 利用率稳定且不超过内存限制。同样，在 ogbn- products 数据集中，峰值内存使用量为 240.62 MB，准备时间较短，因此较小的队列大小 3 便足以维持优化的 GPU 利用率。

这些结果凸显了 MQ-GNN 应对数据集特定挑战的适应能力，并能确保高效的内存使用和稳定的 GPU 利用率。与 DGL 相比，MQ-GNN 展现出更高的峰值 GPU 利用率和更出色的稳定性；DGL 的利用率则频繁下滑，这可能是由顺序处理数据和传输操作引入延迟所致。这种稳定性源于整合排队机制和基于内存的约束，以优化数据流水线。

此外，对于大型数据集，MQ-GNN 会根据可用 GPU 内存和批量大小要求限制队列大小，从而防止 GPU 内存饱和。MQ-GNN 根据性能分析结果调整队列大小，在更好地提升利用率的同时，平衡资源约束与计算效率。该方法会分析每个数据集的峰值内存使用量，并在 GPU 内存限制下预测队列大小，同时考虑执行 CUDA 操作时使用的临时数据。这种方法能够提升 GPU 利用率。这些结果表明，MQ-GNN 的排队机制和异步数据处理具有优势，能够维持 GPU 活跃度并提高整体吞吐量。

MQ-GNN 的轻量化设计在单 GPU 配置下取消了梯度队列，简化了训练流水线，并缩短了训练时间。其队列机制和流水线机制通过将采样与计算解耦，增加数据传输与训练之间的重叠，从而实现了这些改进。通过避免顺序处理（在这种处理方式下，GPU 往往会在数据准备期间处于空闲状态），MQ-GNN 消除了这些瓶颈，实现了稳定的 GPU 利用率并提高了吞吐量。这些研究结果确立了 MQ-GNN 作为一种稳健、可扩展且高效的解决方案

表 4. 单 GPU 配置下 LADIES、MQ-LADIES 及其增强变体的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）。

模型	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
LADIES	批次	10.18 ± 0.20	11.39 ± 0.15	13.26 ± 0.18	9.11 ± 0.40
	训练	348144.5 ± 4021.12	**604201.54 ± 5898.82**	807788.34 ± 7965.98	476977.05 ± 9539.54
	指标	68.27 ± 0.12	61.0 ± 0.20	74.9 ± 0.18	52.84 ± 0.40
MQ-LADIES	批次	6.51 ± 0.18	7.11 ± 0.20	7.64 ± 0.18	4.99 ± 0.25
	训练	211709.12 ± 2427.09	**348106.70 ± 2851.07**	442648.37 ± 5726.48	248904.21 ± 2359.04
	指标	68.37 ± 0.10	61.4 ± 0.15	75.0 ± 0.20	52.71 ± 0.25
LADIES+f	批次	10.24 ± 0.22	11.27 ± 0.18	13.29 ± 0.20	9.13 ± 0.50
	训练	349983.5 ± 2799.98	493595.2 ± 5365.12	805818.11 ± 7933.98	528674.09 ± 10573.48
	指标	68.18 ± 0.10	62.8 ± 0.18	90.12 ± 0.20	62.14 ± 0.30
MQ-LADIES+f	批次	6.35 ± 0.18	6.61 ± 0.22	7.36 ± 0.18	4.79 ± 0.25
	训练	201255.32 ± 2432.55	269853.61 ± 2858.54	416927.39 ± 5769.27	247821.93 ± 2378.22
	指标	68.38 ± 0.15	62.82 ± 0.20	90.21 ± 0.25	62.11 ± 0.40
LADIES+d	批次	11.62 ± 0.25	13.0 ± 0.22	14.28 ± 0.25	9.18 ± 0.60
	训练	397178.5 ± 2989.76	571631.9 ± 4979.89	1063068.25 ± 11520.23	432755.37 ± 8655.11
	指标	68.76 ± 0.15	61.48 ± 0.22	86.81 ± 0.25	54.82 ± 0.50
MQ-LADIES+d	批次	7.31 ± 0.25	7.98 ± 0.22	8.40 ± 0.25	5.09 ± 0.30
	训练	248188.67 ± 2521.89	334029.90 ± 2990.30	590965.54 ± 5929.66	239021.27 ± 2510.21
	指标	68.9 ± 0.20	61.45 ± 0.25	86.89 ± 0.30	54.98 ± 0.35
LADIES+f+d	批次	11.58 ± 0.22	**14.96 ± 0.25**	15.35 ± 0.30	9.11 ± 0.70
	训练	395726.1 ± 2989.67	697795.4 ± 5969.87	1381972.57 ± 14728.65	404781.39 ± 15895.63
	指标	67.55 ± 0.20	61.90 ± 0.25	88.43 ± 0.30	62.20 ± 0.60
MQ-LADIES+f+d	批次	7.09 ± 0.22	7.55 ± 0.25	8.45 ± 0.30	5.01 ± 0.30
	训练	231872.88 ± 2518.73	383751.93 ± 2997.52	694872.38 ± 5948.72	201580.35 ± 2515.80
	指标	67.49 ± 0.25	62.01 ± 0.30	88.29 ± 0.35	62.30 ± 0.40

适用于图神经网络训练，即使是在单GPU配置下也是如此。

2) 多GPU性能

与基线方法相比，MQ-GNN 在多 GPU 配置下实现了显著的性能提升（最高可达 4.6 ×），如表 5-13所示。借助 RaCoM 框架（第 IV 节）、基于队列的梯度共享以及高效的排队机制，MQ-GNN 在多种数据集上实现了更快的训练和批处理时间，同时在评估性能相当的情况下，评估指标仅下降约 2.0% 。这些结果凸显了该系统通过周期性同步降低通信开销、减少 GPU 空闲时间并处理陈旧性问题的能力。

NS性能 在双 GPU 配置下，MQ-GNN 在保持指标竞争力的同时，大幅缩短了训练时间和批处理时间，如表5-7所示。例如，在 Reddit 数据集上，MQ-GCN 相较于 GCN 显著缩短了训练时间，降幅为3.10 ×。MQ-GraphSAGE 也实现了 3.15 × 倍加速（表5）。这些结果凸显了 MQ-GNN 高效的队列机制，该机制减少了 GPU 空闲时间，并实现了流畅的小批次转换。尽管取得了这些改进，指标仍略有下降：MQ-GCN 下降了 0.05% ，MQ-GraphSAGE 下降了 0.2%。梯度共享和模型更新期间的异步更新会导致陈旧性，从而造成这种下降。RaCoM 无法消除陈旧性的影响，尤其是在更容易受到异步更新影响的模型中。不过，通过定期同步和恰当的梯度管理，它能显著降低这种影响。

对于 ogbn-proteins 这类较小的数据集，MQ-GraphSAGE 提供了 2.61 × 改进。批处理时间按比例减少，突出了 MQ-GNN 适应不同图规模的能力。由于这些图的稀疏性以及不频繁的模型同步，MQ-GNN 在 ogbn-proteins 和 ogbn-arxiv 等较小数据集上的指标下降更为明显。在 ogbn-proteins 上，MQ-GraphSAGE 的评估指标下降了 0。3%，而在 ogbn-arxiv 上下降了0。28%（表5）。这些数据集的稀疏连接削弱了缓存和排队带来的收益，而不频繁的同步则加剧了陈旧性对异步更新的影响，从而导致整体模型准确率下降。

扩展到三卡和四卡GPU时，MQ-GNN在训练时间和批处理时间方面仍持续优于基线方法，这一点已由附录 8 至 13 中的表 B所证实。对于 Reddit 数据集，MQ-GCN在三卡GPU上取得了显著的 3.95 × 改进，如表 8所示。类似地，对于 ogbn-products 数据集，MQ-GraphSAGE实现了 3.36 ×加速。对于 ogbn-arxiv 数据集，MQ-GraphSAGE在训练时间方面实现了 2.89 × 加速。批处理时间显著下降，从31.59 毫秒降至10.45 毫秒（表 8）。使用四卡GPU时，MQ-GNN仍保持了高效率，如 ogbn-arxiv 数据集上的结果所示，其中MQ-GCN在训练时间方面取得了3.45 × 改进（表 11）。在 Reddit 和 ogbn-products 等稠密数据集上，MQ-GNN表现出色，因为它利用缓存和重叠来实现最佳效率。MQ-GNN架构能够有效降低同步和通信开销，尤其是在稠密图中。这些结果表明，MQ-GNN能够高效管理稠密数据集，尽管随着图形处理器数量增加，同步会变得更加困难。

层级采样性能MQ-GNN实现了显著的速度提升

表 5. 双 GPU 配置下 GCN、GraphSAGE、MQ-GCN 和 MQ-GraphSAGE 的批处理时间（ms）、训练时间（ms）和评估指标（%）。

模型	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
GCN	批次	12.15 ± 0.58	13.84 ± 0.47	**23.54 ± 0.42**	13.32 ± 0.49
	训练	207644.09 ± 11074.32	298242.34 ± 3345.21	**716904.12 ± 7458.76**	224901.25 ± 5289.03
	指标	66.28 ± 0.49	67.71 ± 0.52	**88.35 ± 0.79**	75.36 ± 0.51
MQ-GCN	批次	4.81 ± 8.25	5.20 ± 9.12	**7.47 ± 22.50**	4.61 ± 10.34
	训练	83392.37 ± 0.49	114318.65 ± 0.52	**230592.43 ± 0.79**	80000.44 ± 0.51
	指标	66.08 ± 0.40	67.61 ± 0.45	**88.3 ± 0.70**	75.26 ± 0.40
GraphSAGE	批次	23.71 ± 0.40	30.85 ± 0.61	36.13 ± 0.55	23.09 ± 0.37
	训练	**405154.41 ± 4358.29**	637116.36 ± 6728.41	**1132782.14 ± 11834.12**	504154.02 ± 5310.89
	指标	**66.03 ± 0.42**	**68.58 ± 0.57**	94.89 ± 0.23	**78.39 ± 0.62**
MQ-GraphSAGE	批次	8.91 ± 12.51	11.41 ± 15.34	11.21 ± 27.56	7.62 ± 13.28
	训练	**154708.64 ± 0.50**	241777.02 ± 0.52	**359521.25 ± 0.80**	171465.83 ± 0.62
	指标	**65.73 ± 0.35**	**68.38 ± 0.40**	94.84 ± 0.18	**78.19 ± 0.50**

表 6. 双 GPU 配置下 FastGCN、MQ-FastGCN 及其增强变体的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）结果。

模型	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
FastGCN	批次	8.04 ± 0.37	7.46 ± 0.28	9.08 ± 0.41	10.36 ± 0.45
	训练	137406.91 ± 1421.17	133517.13 ± 1402.34	**273813.32 ± 2845.98**	151858.83 ± 1559.87
	指标	53.21 ± 0.48	26.19 ± 0.39	45.81 ± 0.55	28.88 ± 0.52
MQ-FastGCN	批次	2.90 ± 8.21	2.61 ± 7.89	2.72 ± 10.45	3.15 ± 9.34
	训练	51189.87 ± 0.55	48299.73 ± 0.40	**83994.41 ± 0.60**	47605.60 ± 0.49
	指标	52.71 ± 0.40	25.89 ± 0.35	45.71 ± 0.48	28.58 ± 0.45
FastGCN+f	批次	8.04 ± 0.42	7.87 ± 0.39	8.93 ± 0.48	8.78 ± 0.43
	训练	137441.27 ± 1457.12	140884.09 ± 1482.94	269335.06 ± 2798.42	128694.32 ± 1351.89
	指标	61.12 ± 0.48	26.20 ± 0.65	53.64 ± 0.69	33.24 ± 0.63
MQ-FastGCN+f	批次	3.04 ± 8.32	2.91 ± 8.45	2.75 ± 10.75	2.98 ± 9.15
	训练	52834.47 ± 0.52	53127.50 ± 0.45	84174.25 ± 0.63	44628.97 ± 0.58
	指标	60.72 ± 0.40	26.0 ± 0.50	53.54 ± 0.60	33.04 ± 0.50
FastGCN+d	批次	8.05 ± 0.39	8.06 ± 0.35	8.79 ± 0.47	9.59 ± 0.52
	训练	137557.27 ± 1435.89	144348.22 ± 1487.32	265117.67 ± 2759.83	140487.23 ± 1453.76
	指标	53.70 ± 0.72	27.37 ± 0.68	44.28 ± 0.83	28.01 ± 1.43
MQ-FastGCN+d	批次	3.12 ± 8.57	3.07 ± 8.94	2.75 ± 11.10	3.35 ± 9.78
	训练	54547.94 ± 0.62	55924.65 ± 0.65	84382.63 ± 0.73	50125.19 ± 0.85
	指标	53.20 ± 0.60	27.07 ± 0.60	44.18 ± 0.70	27.71 ± 1.20
FastGCN+f+d	批次	7.67 ± 0.43	7.66 ± 0.35	8.88 ± 0.50	9.59 ± 0.34
	训练	131084.62 ± 1348.91	137053.24 ± 1421.38	267829.32 ± 2823.94	**140487.28 ± 1459.75**
	指标	61.15 ± 0.12	24.32 ± 0.18	53.90 ± 0.19	**30.14 ± 0.37**
MQ-FastGCN+f+d	批次	2.82 ± 8.11	2.77 ± 8.45	2.69 ± 10.98	3.22 ± 9.05
	训练	49439.57 ± 0.50	50670.86 ± 0.45	82618.40 ± 0.58	**48081.08 ± 0.55**
	指标	60.75 ± 0.10	24.12 ± 0.15	53.80 ± 0.12	**29.94 ± 0.35**

与逐层方法相比，在使用两块图形处理器时性能有所提升，如表 6 和 7所示。在 Reddit 数据集上使用 MQ-FastGCN 时，训练时间缩短了 3.26 × (表 6)。同样，MQ-LADIES 将训练时间缩短了 3.23 × (表 7)。在 ogbn-products 数据集上，MQ-FastGCN+f+d 实现了 2.92 × 的性能提升，批处理时间缩短了 2.97 × ，且度量指标仅略微下降了 0.2% 。对于 Reddit 这类稠密图，MQ-LADIES+f 的评估指标降幅较小，仅下降了 0.05% ，同时相较于对应方法实现了3.27 × 加速。这些结果凸显了 MQ-GNN 在 LADIES 和 FastGCN 的高级变体中整合额外计算开销（例如平坦采样（10）和去偏（13））的能力，同时仅带来极小的运行时开销。

随着图形处理器数量增加至三块和四块，MQ-GNN 仍展现出显著的效率优势，如附录8-13 中的表格所示 B。在三块图形处理器上，MQ-FastGCN 实现了 3.38 × 的提升，如表格 9所示。同样，在 ogbn-arxiv 数据集上，MQ-LADIES 将训练时间缩短了 2.95 ×。批处理时间也呈现类似趋势，从 10.92 ms 降至 3.52 ms（表 10）。尽管训练加速，评估指标仍因陈旧性而略有下降，其中 MQ-FastGCN+f+d 的度量指标下降了 0.8%（表9）。MQ-LADIES+f+d 在四个图形处理器上展现出出色的可扩展性，使 ogbn-products 数据集的训练时间缩短了 3.6 × 。批处理时间也按比例改善，从 8.36 ms 降至 2.23 ms，代价是度量指标下降 1.0%。这展现了 MQ-GNN 动态平衡陈旧性与同步成本的能力，如表13所示。在 Reddit 数据集上，MQ-LADIES+f+d 和 MQ-FastGCN+f+d 的训练加速比为 4.35 × ，度量指标仅略降 0.6%（表13和表 12 ）。即使使用三卡和四卡 GPU，MQ-GNN 的评估指标仍接近基线方法；借助 RaCoM 进行周期性同步以降低陈旧性后，只有轻微偏差。此外，诸如平坦采样（10）等额外计算开销的整合

表 7. LADIES、MQ-LADIES 及其增强变体在双 GPU 配置下的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）结果。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
LADIES	批次	6.85 ± 0.19	10.92 ± 0.41	8.20 ± 0.33	7.59 ± 0.28
	训练	117160.16 ± 1198.45	235377.73 ± 2430.52	**191979.81 ± 1998.45**	366542.86 ± 3721.24
	指标	68.34 ± 0.16	60.62 ± 0.23	75.47 ± 0.22	53.47 ± 0.42
MQ-LADIES	批次	2.49 ± 5.25	3.90 ± 8.90	2.49 ± 11.25	2.41 ± 9.95
	训练	43320.33 ± 0.08	85432.90 ± 0.11	**59420.67 ± 0.13**	118779.52 ± 0.15
	指标	67.94 ± 0.10	60.42 ± 0.20	75.42 ± 0.18	53.27 ± 0.40
LADIES+f	批次	6.72 ± 0.24	10.60 ± 0.43	8.43 ± 0.48	8.23 ± 0.36
	训练	114955.31 ± 1203.42	222753.14 ± 2299.01	**198578.13 ± 2011.68**	{438722.13} ± 4487.24
	指标	68.66 ± 0.07	63.45 ± 0.14	**90.55 ± 0.16**	**63.21 ± 0.22**
MQ-LADIES+f	批次	2.44 ± 5.15	3.75 ± 8.65	2.53 ± 10.95	2.53 ± 10.25
	训练	42378.54 ± 0.09	80344.90 ± 0.13	**60673.42 ± 0.15**	**137522.97 ± 0.22**
	指标	68.36 ± 0.08	63.25 ± 0.15	**90.50 ± 0.12**	**63.01 ± 0.20**
LADIES+d	批次	11.05 ± 0.53	13.68 ± 0.39	9.84 ± 0.47	7.85 ± 0.34
	训练	188865.71 ± 1924.32	327562.53 ± 3321.40	443242.73 ± 4520.24	451953.42 ± 4615.22
	指标	68.77 ± 0.11	61.43 ± 0.17	87.71 ± 0.18	55.58 ± 0.37
MQ-LADIES+d	批次	4.03 ± 8.35	5.01 ± 10.78	3.08 ± 12.45	2.53 ± 11.35
	训练	70227.20 ± 0.12	122060.70 ± 0.14	141574.51 ± 0.18	147443.91 ± 0.23
	指标	68.37 ± 0.12	61.13 ± 0.18	87.61 ± 0.18	55.28 ± 0.35
LADIES+f+d	批次	11.78 ± 0.41	12.80 ± 0.34	10.69 ± 0.50	7.80 ± 0.42
	训练	201339.29 ± 2034.67	280377.04 ± 2863.15	481194.71 ± 4895.74	452177.30 ± 4610.84
	指标	67.56 ± 0.16	62.99 ± 0.18	88.94 ± 0.24	63.53 ± 0.48
LADIES+f+d	批次	4.52 ± 8.95	4.77 ± 10.95	3.23 ± 13.10	2.67 ± 12.15
	训练	78953.76 ± 0.11	106808.64 ± 0.15	147969.36 ± 0.20	158074.05 ± 0.28
	指标	67.16 ± 0.15	62.79 ± 0.18	88.84 ± 0.20	63.33 ± 0.45

以及去偏（13）在 LADIES+f+d 和 FastGCN+f+d 中，都能在 MQ-GNN 的流水线中得到高效管理，而不会显著增加运行时间。这是通过将这些任务与其他操作高效重叠来实现的。某些模型（如 LADIES）对异步更新引起的陈旧性更加敏感，这会影响评估指标，尤其是在同步频率较低的情况下。

比较分析 通过定期对齐更新，RaCoM 确保模型保持同步，即使在通信需求较高的情况下也是如此。这种方法使 MQ-GNN 在使用两块或三块图形处理器的配置下，能够提供稳定的评估指标；扩展到四块图形处理器时，指标略有下降，尤其是在较小或稀疏的数据集（如 ogbn-proteins）上。

图稀疏性会加剧陈旧性的影响，因为 RaCoM 中的周期性同步间隔无法完全满足实现一致特征聚合所需的异步更新。例如，在 (1) 中，由于梯度更新延迟，陈旧性可能会在使用过时的节点嵌入 (Z^{l-1}) 时扰乱聚合过程。这会导致特征传播不一致，在稀疏数据集中尤为棘手，因为每条边和每个特征对整体聚合的影响都更加显著。同样，在 GraphSAGE 中，若邻居嵌入 (3) 未同步，陈旧性会扰乱采样过程。由于邻居数量有限，稀疏图会加剧这一问题，使每个邻居的嵌入都至关重要。因此，RaCoM 的同步周期与更新无法很好地对齐，导致聚合不一致，性能下降。

此外，RaCoM模型中的陈旧性也会影响LS方法。例如，FastGCN（定义见（4））在每一层采样节点，并假设采样节点之间相互独立。因此，同步延迟会影响FastGCN，因为它难以应对过时的采样嵌入；这些嵌入无法反映图的当前状态，从而导致估计误差，并使聚合过程的方差增大。稠密数据集中的冗余在一定程度上缓解了这一问题，但在稀疏图中，由于采样节点较少，问题会更加严重。与FastGCN相比，LADIES受到的影响更大，因为每一层都依赖前一层的采样节点（定义见（6））。层与层之间的层级依赖进一步放大了陈旧性的影响。较早层的嵌入更新若未同步完成，后续层的近似质量就会下降，尤其是在稀疏图中。MQ-GNN通过RaCoM进行的周期性同步，无法有效适应LADIES依赖层的采样方式，导致估计误差增大、性能下降。FastGCN和LADIES的平坦采样（10）与去偏（13）技术旨在通过降低方差和校正估计偏差来提高采样鲁棒性；但在多 GPU环境中，由于梯度同步延迟会导致采样分布不均匀，这些技术的效果会减弱。这又增加了一层复杂性，而MQ-GNN当前的同步策略难以彻底解决这一问题。

在多 GPU 系统中，MQ-GNN 的多队列流水线化方法对于最大限度地提高资源效率至关重要。该方法将预定数量的小批次加入队列，以减少 GPU 空闲时间并确保任务无缝转换。这种排队策略能够提升训练效率，并在各种数据集和配置下表现出色。该方法适用于大型数据集，

在这类数据集中，可扩展性取决于能否在多个图形处理器上维持高吞吐量。

将 RaCoM、多队列和 GNS 这些技术结合起来，使 MQ-GNN 成为适用于多 GPU 环境的稳健框架。RaCoM 提供的周期性同步减轻了异步处理和通信延迟的影响。多队列方法通过减少空闲时间并确保训练顺畅进行，提高了 GPU 利用率。GNS 通过减少向 GPU 传输数据时的通信开销，进一步加快了训练速度。综合运用这些技术，MQ-GNN 得以实现可扩展且高效的性能，尽管在大型和小型数据集上的指标略有下降。这些改进表明，MQ-GNN 是现有多 GPU 环境基线方法的一种有效且可扩展的替代方案。尽管 MQ-GNN 显著提升了计算速度和可扩展性，但在多图形处理器训练中，其同步和排队机制在稳定逐层采样模型的性能方面效果较差。

VI. 结论

本文提出了 MQ-GNN，这是一种可扩展且高效的多 GPU 图神经网络训练框架。MQ-GNN 采用多队列流水线，高效地重叠小批次生成、数据传输和计算，从而减少空闲时间并最大限度地提高资源利用率。它采用自适应队列大小调整策略，在平衡内存限制与计算效率的同时，利用带缓存的全局邻居采样来最大限度地减少数据传输开销。此外，RaCoM 周期性同步在考虑图特征和 GPU 数量的情况下，最大限度地减少了通信瓶颈，同时保持模型准确率。

在四个大规模数据集和十种基线模型上的实验结果表明，MQ-GNN 的训练速度最高可提升$4.6\times$，GPU 利用率提高 30%，而准确率仅降低 2%。这些结果验证了 MQ-GNN 在不同图结构上优化大规模图神经网络训练的有效性。

MQ-GNN 为推荐引擎、社交网络和生物系统等应用提供了高性能且可扩展的图学习解决方案。未来的研究将把 MQ-GNN 扩展至动态图和更复杂的图神经网络结构，以进一步提升其对不断变化的大规模数据集的适用性。

.

附录 A
周期性同步

图神经网络训练涉及三个主要因素：逐节点计算、逐边计算和图形处理器工作负载。逐节点计算随 $|V|$扩展，并通过节点嵌入和局部特征变换增加工作负载。较大的 $|V|$ 需要更长的同步间隔，以在保持效率的同时平衡计算成本。逐边计算随$|E|$扩展，对邻居聚合和通信开销产生影响。$|E|$ 的平方根用于描述额外边对陈旧性的递减收益，确保更稠密的图更频繁地进行同步，以防止异步更新导致发散。图形处理器工作负载的分配也会进一步影响同步；随着 $|\mathcal{G}|$ 增加，工作负载会分散到更多设备上，从而缩短同步间隔，增强分布式系统中的一致性。

命题1. 在计算效率和模型一致性之间取得平衡的最优周期性同步间隔 P_{ms} 为：

$$P_{ms} = \left\lceil \frac{\sqrt{|V|}}{\sqrt{|\mathcal{G}|\,|E|}} \right\rceil \tag{26}$$

此间隔可确保：

- 同步频率会根据图密度动态调整。
- 陈旧性（由于异步更新导致）与同步开销（由频繁通信造成）之间的权衡得以最小化。

证明。 设 $C(P)$ 为平衡陈旧性成本与同步成本的总成本函数：

$$C(P) = \alpha P|E| + \beta \frac{1}{P}\frac{|V|}{|\mathcal{G}|},$$

其中， $P|E|$ 表示陈旧性成本，该成本随 P 线性增加，并与边的数量$|E|$成正比。项 $\frac{1}{P}\frac{|V|}{|\mathcal{G}|}$ 表示同步成本，该成本随 P 反向降低，并与节点数量 $|V|$成正比，再除以图形处理器的数量 $|\mathcal{G}|$。常数 α 和 β 是加权系数，用于控制每一项的相对贡献。

为了最小化 $C(P)$，我们对 $C(P)$ 关于 P求导，并令 $\frac{dC(P)}{dP} = 0$：

$$\frac{dC(P)}{dP} = \alpha|E| - \beta\frac{1}{P^2}\frac{|V|}{|\mathcal{G}|}$$

重新整理各项可得：

$$\alpha|E| = \beta\frac{1}{P^2}\frac{|V|}{|\mathcal{G}|}$$

两边同时乘以 P^2，得到：

$$P^2 = \frac{\beta}{\alpha}\frac{|V|}{|\mathcal{G}|\,|E|}$$

两边开平方，得到：

$$P = \sqrt{\frac{\beta}{\alpha}}\sqrt{\frac{|V|}{|\mathcal{G}|\,|E|}}$$

此处，$\sqrt{\frac{\beta}{\alpha}}$ 被视为常数缩放因子 k。为简便起见，对 $k = 1$进行归一化，得到：

$$P = \frac{\sqrt{|V|}}{\sqrt{|\mathcal{G}|\,|E|}}$$

为确保 P 是适合实现的离散值，应用向上取整函数：

$$P_{ms} = \left\lceil \frac{\sqrt{|V|}}{\sqrt{|\mathcal{G}|\,|E|}} \right\rceil$$

这一推导表明，P_{ms} 会根据图 G（$|V|$ 和 $|E|$）的结构以及图形处理器的数量（$|\mathcal{G}|$）进行动态适配，从而确保陈旧性与同步成本之间的平衡。

□

附录 B 三个和四个图形处理器的结果与最先进水平对比

表 8 至13 展示了三种和四种图形处理器配置下的对比结果，体现了 MQ-GNN 的可扩展性和效率。结果突显了训练加速比、批处理时间的改进，以及陈旧性对评估指标的影响，说明 MQ-GNN 如何通过定期同步平衡性能与准确率。

表8. 三GPU配置下，GCN、GraphSAGE、MQ-GCN 和 MQ-GraphSAGE 的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）结果。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
GCN	批次	13.58 ± 0.12	13.85 ± 0.10	24.25 ± 0.18	15.19 ± 0.11
	训练	155635.53 ± 1525.12	249346.67 ± 2050.21	**506732.17 ± 3810.48**	296226.39 ± 1820.34
	指标	66.79 ± 0.70	68.45 ± 0.50	90.82 ± 0.35	71.04 ± 0.42
MQ-GCN	批次	4.52 ± 0.18	4.39 ± 0.25	5.91 ± 0.30	4.22 ± 0.20
	训练	54487.44 ± 2400.35	**83083.30 ± 3260.28**	**128234.18 ± 5210.45**	**85834.73 ± 2100.48**
	指标	66.29 ± 0.35	68.15 ± 0.42	90.62 ± 0.60	70.74 ± 0.50
GraphSAGE	批次	25.01 ± 0.64	**31.59 ± 0.43**	38.29 ± 0.26	23.65 ± 0.59
	训练	277099.73 ± 0.16	**432195.54 ± 0.28**	777316.19 ± 0.94	**378204.24 ± 0.05**
	指标	66.62 ± 0.98	69.07 ± 0.47	94.53 ± 0.95	63.72± 0.26
MQ-GraphSAGE	批次	8.50 ± 4.12	**10.45 ± 5.85**	9.45 ± 7.15	6.76 ± 5.05
	训练	98610.54 ± 110.25	**149490.10 ± 150.35**	199308.66 ± 220.18	**112520.80 ± 175.45**
	指标	65.92 ± 0.32	67.66 ± 0.38	90.83 ± 0.15	70.74 ± 0.45

表9。 三图形处理器下的批处理时间（毫秒 ）、训练时间（毫秒）和评估指标（%）：FastGCN、MQ-FastGCN 及其增强变体配置。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
FastGCN	批次	7.82 ± 0.35	8.09 ± 0.30	9.21 ± 0.40	9.62 ± 0.35
	训练	89664.64 ± 725	97615.04 ± 810	185078.34 ± 1290	**94393.48 ± 520**
	指标	53.69 ± 0.35	26.9 ± 0.42	43.61 ± 0.50	28.81 ± 0.45
MQ-FastGCN	批次	2.74 ± 1.50	2.70 ± 1.80	2.30 ± 3.25	2.73 ± 2.40
	训练	33072.93 ± 125.35	34224.56 ± 135.45	48058.41 ± 155.25	**27917.82 ± 145.40**
	指标	52.69 ± 0.20	26.1 ± 0.25	43.31 ± 0.25	28.11 ± 0.40
FastGCN+f	批次	7.69 ± 0.20	7.81 ± 0.15	8.54 ± 0.25	9.02 ± 0.18
	训练	88148.54 ± 680	70788.87 ± 590	173517.37 ± 1200	89131.48 ± 610
	指标	62.1 ± 0.40	22.46 ± 0.35	52.99 ± 0.40	32.29 ± 0.32
MQ-FastGCN+f	批次	2.66 ± 1.70	2.56 ± 2.00	2.08 ± 3.18	2.52 ± 2.20
	训练	32371.82 ± 130.25	24400.47 ± 110.25	43924.31 ± 145.30	26060.72 ± 125.45
	指标	61.3 ± 0.22	21.86 ± 0.25	52.69 ± 0.22	31.69 ± 0.42
FastGCN+d	批次	7.62 ± 0.06	7.70 ± 0.37	8.75 ± 0.87	9.54 ± 1.05
	训练	87327.23 ± 0.96	69783.84 ± 0.62	175894.47 ± 0.92	93658.32 ± 0.05
	指标	54.22 ± 1.04	27.0 ± 0.76	44.93 ± 0.29	30.09 ± 0.38
MQ-FastGCN+d	批次	2.71 ± 1.75	2.64 ± 1.90	2.24 ± 3.45	2.80 ± 2.75
	训练	32623.50 ± 125.45	25098.34 ± 130.25	46716.02 ± 145.80	28718.33 ± 135.45
	指标	53.32 ± 0.25	26.30 ± 0.30	44.53 ± 0.20	29.29 ± 0.40
FastGCN+f+d	批次	7.85 ± 0.22	8.01 ± 0.74	8.91± 0.7	9.54± 0.95
	训练	**89942.67 ± 1.01**	96594.43 ± 0.01	179902.72 ± 0.38	93632.14 ± 0.64
	指标	**61.76 ± 0.07**	24.25 ± 0.2	54.53 ± 0.96	32.23 ± 0.88
MQ-FastGCN+f+d	批次	2.77 ± 1.80	2.67 ± 2.15	2.20 ± 3.30	2.72 ± 2.50
	训练	**33275.04 ± 135.40**	33642.75 ± 140.20	45964.57 ± 150.30	27860.07 ± 130.35
	指标	**60.96 ± 0.15**	23.75 ± 0.20	54.23 ± 0.20	31.73 ± 0.35

表 10. LADIES、MQ-LADIES 及其增强变体在三GPU配置下的批处理时间（ms）、训练时间（ms）和评估指标（%）。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
LADIES	批次	6.85 ± 0.20	10.92 ± 0.35	8.20 ± 0.30	7.59 ± 0.25
	训练	117160.16 ± 910.25	**235377.73 ± 1850.52**	191979.81 ± 1510.34	366542.86 ± 2760.28
	指标	68.34 ± 0.18	60.62 ± 0.28	75.47 ± 0.16	53.47 ± 0.30
MQ-LADIES	批次	2.38 ± 0.54	3.52 ± 0.58	1.97 ± 0.16	2.08 ± 0.47
	训练	42568.80 ± 980.25	**79730.09 ± 1785.50**	47994.18 ± 1687.45	104707.20 ± 1520.37
	指标	67.44 ± 0.15	60.02 ± 0.33	75.17 ± 0.37	52.97 ± 0.74
LADIES+f	批次	6.02 ± 0.20	10.30 ± 0.30	8.51 ± 0.35	8.12 ± 0.15
	训练	68952.23 ± 520.10	134775.56 ± 950.45	209233.72 ± 1320.75	150464.29 ± 920.30
	指标	68.67 ± 0.12	63.30 ± 0.20	90.87 ± 0.25	62.41 ± 0.18
MQ-LADIES+f	批次	2.06 ± 0.42	3.26 ± 0.38	2.02 ± 0.18	2.19 ± 0.22
	训练	24675.93 ± 920.45	44787.82 ± 1187.35	51639.00 ± 1520.62	42384.17 ± 1387.44
	指标	67.87 ± 0.13	62.80 ± 0.23	90.67 ± 0.15	62.01 ± 0.30
LADIES+d	批次	10.87 ± 0.35	12.34 ± 0.40	9.97 ± 0.28	8.02 ± 0.20
	训练	124582.76 ± 980.15	222182.65 ± 1450.52	286160.53 ± 1980.85	137120.38 ± 1150.30
	指标	69.30 ± 0.15	62.29 ± 0.18	88.43 ± 0.14	54.91 ± 0.24
MQ-LADIES+d	批次	3.80 ± 0.38	4.18 ± 0.28	2.49 ± 0.45	2.30 ± 0.22
	训练	45733.19 ± 1250.45	78674.80 ± 1487.34	74325.03 ± 2287.64	41063.36 ± 1350.52
	指标	68.40 ± 0.15	61.69 ± 0.28	88.13 ± 0.26	54.31 ± 0.42
LADIES+f+d	批次	11.82 ± 0.40	12.88 ± 0.35	9.97 ± 0.28	8.01 ± 0.20
	训练	135476.01 ± 1080.34	231789.48 ± 1630.25	229440.32 ± 1970.56	131150.32 ± 1140.78
	指标	68.06 ± 0.12	60.01 ± 0.18	88.78 ± 0.15	62.11 ± 0.22
MQ-LADIES+f+d	批次	4.22 ± 0.28	4.43 ± 0.38	2.52 ± 0.12	2.35 ± 0.28
	训练	50542.12 ± 1025.35	83561.08 ± 1820.44	60232.61 ± 2350.54	40353.69 ± 1425.87
	指标	67.26 ± 0.23	59.51 ± 0.72	88.48 ± 0.25	61.61 ± 0.35

表11. 四GPU配置下，GCN、GraphSAGE、MQ-GCN 和 MQ-GraphSAGE 的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
GCN	批次	13.17 ± 0.15	14.48 ± 0.09	22.42 ± 0.85	15.70 ± 0.38
	训练	113834.39 ± 845.32	**173942.63 ± 1378.41**	351391.48 ± 2825.67	226895.51 ± 1545.27
	指标	66.81 ± 0.80	68.54 ± 0.40	90.97 ± 0.70	71.40 ± 0.45
MQ-GCN	批次	3.87 ± 0.25	4.02 ± 0.30	4.87 ± 0.50	3.92 ± 0.45
	训练	34954.27 ± 265.15	**50400.06 ± 290.27**	78948.03 ± 400.67	58925.16 ± 335.28
	指标	65.61 ± 0.40	67.74 ± 0.45	90.47 ± 0.55	70.80 ± 0.50
GraphSAGE	批次	20.54 ± 0.32	32.08 ± 0.25	39.36 ± 0.60	24.38 ± 0.50
	训练	177562.71 ± 1260.32	353692.29 ± 2058.45	**602149.33 ± 3450.87**	298715.49 ± 1885.40
	指标	66.62 ± 0.65	69.33 ± 0.50	**94.79 ± 0.30**	63.85 ± 0.40
MQ-GraphSAGE	批次	6.14 ± 0.50	9.14 ± 0.60	8.65 ± 0.70	6.25 ± 0.60
	训练	55442.75 ± 385.45	105458.24 ± 500.62	**136716.51 ± 670.54**	79655.60 ± 450.27
	指标	65.12 ± 0.45	68.33 ± 0.50	**94.19 ± 0.35**	62.95 ± 0.42

表12。 FastGCN、MQ-FastGCN 及其增强变体在四GPU配置下的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
FastGCN	批次	7.85 ± 0.97	7.84 ± 0.85	9.24 ± 0.75	12.75 ± 0.58
	训练	67839.25 ± 525.45	53289.46 ± 395.25	139282.58 ± 865.35	125574.71 ± 710.15
	指标	53.69 ± 0.50	26.90 ± 0.55	43.67 ± 0.35	27.01 ± 0.25
MQ-FastGCN	批次	2.41 ± 0.25	2.30 ± 0.30	2.05 ± 0.38	3.35 ± 0.42
	训练	21868.93 ± 165.25	16377.15 ± 150.35	31989.47 ± 195.50	34383.46 ± 215.40
	指标	51.69 ± 0.32	25.40 ± 0.38	42.97 ± 0.30	25.81 ± 0.48
FastGCN+f	批次	7.57 ± 0.45	7.90 ± 0.55	8.61 ± 0.52	9.36 ± 0.65
	训练	65406.74 ± 495.30	53655.51 ± 425.10	129790.42 ± 875.30	92650.35 ± 640.45
	指标	62.10 ± 0.65	24.46 ± 0.45	53.12 ± 0.50	32.32 ± 0.40
MQ-FastGCN+f	批次	2.30 ± 0.28	2.29 ± 0.32	1.87 ± 0.45	2.43 ± 0.55
	训练	20741.16 ± 170.20	16211.68 ± 155.50	29163.13 ± 205.75	25026.60 ± 185.40
	指标	60.40 ± 0.32	23.16 ± 0.35	52.52 ± 0.45	31.22 ± 0.50
FastGCN+d	批次	7.45 ± 0.45	7.66 ± 0.40	8.49 ± 0.58	9.38 ± 0.55
	训练	64367.37 ± 470.50	52058.51 ± 400.50	128041.19 ± 860.25	92877.09 ± 645.35
	指标	61.78 ± 0.60	24.14 ± 0.55	54.74 ± 0.60	32.27 ± 0.45
MQ-FastGCN+d	批次	2.33 ± 0.28	2.32 ± 0.32	1.93 ± 0.42	2.53 ± 0.50
	训练	20934.90 ± 180.20	16419.90 ± 165.50	30123.73 ± 215.00	26154.32 ± 150.00
	指标	59.98 ± 0.35	22.74 ± 0.40	54.04 ± 0.50	30.97 ± 0.55
FastGCN+f+d	批次	7.45 ± 0.40	7.66 ± 0.35	8.49 ± 0.52	9.38 ± 0.50
	训练	64367.37 ± 455.35	52058.51 ± 385.25	**128041.19 ± 825.50**	92877.09 ± 620.15
	指标	61.78 ± 0.55	24.14 ± 0.50	**54.74 ± 0.50**	32.27 ± 0.40
MQ-FastGCN+f+d	批次	2.31 ± 0.25	2.29 ± 0.28	1.88 ± 0.35	2.50 ± 0.42
	训练	20793.48 ± 165.25	16162.30 ± 150.35	**29399.34 ± 210.50**	25782.21 ± 185.25
	指标	60.18 ± 0.30	22.94 ± 0.35	**54.14 ± 0.40**	31.27 ± 0.50

表13。 四GPU配置下，LADIES、MQ-LADIES及其增强变体的批处理时间（毫秒）、训练时间（毫秒）和评估指标（%）结果。

	基准测试	ogbn-proteins	ogbn-arxiv	Reddit	ogbn-products
LADIES	批次	6.46 ± 0.28	10.18 ± 0.38	8.85 ± 0.45	8.37 ± 0.42
	训练	55798.51 ± 485.20	126832.24 ± 1025.75	139342.54 ± 905.80	123057.34 ± 925.35
	指标	69.30 ± 0.55	62.34 ± 0.65	75.70 ± 0.50	54.00 ± 0.60
MQ-LADIES	批次	1.97 ± 1.45	2.91 ± 1.20	1.88 ± 0.50	2.11 ± 0.65
	训练	17756.09 ± 110.25	37821.10 ± 165.45	30600.62 ± 125.80	32376.30 ± 165.60
	指标	67.50 ± 0.25	61.04 ± 0.65	75.10 ± 0.37	53.00 ± 0.95
LADIES+f	批次	6.81 ± 0.30	10.44 ± 0.40	11.00 ± 0.50	8.18 ± 0.35
	训练	58849.62 ± 520.35	102393.24 ± 850.40	173280.67 ± 1020.25	120272.43 ± 900.20
	指标	68.47 ± 0.45	64.10 ± 0.55	91.08 ± 0.50	62.31 ± 0.55
MQ-LADIES+f	批次	2.05 ± 1.25	2.94 ± 1.00	2.31 ± 0.75	2.04 ± 0.50
	训练	18479.93 ± 125.72	30093.14 ± 160.65	37667.94 ± 145.50	31208.25 ± 180.40
	指标	66.97 ± 0.95	63.00 ± 0.45	90.58 ± 0.50	61.51 ± 0.35
LADIES+d	批次	11.90 ± 0.50	12.11 ± 0.40	9.47 ± 0.60	8.31 ± 0.45
	训练	102820.34 ± 810.35	163524.31 ± 1325.40	213050.34 ± 1650.50	122178.21 ± 1050.35
	指标	69.29 ± 0.55	62.30 ± 0.45	88.44 ± 0.50	54.89 ± 0.60
MQ-LADIES+d	批次	3.66 ± 1.00	3.56 ± 0.90	2.06 ± 0.50	2.16 ± 0.30
	训练	33037.28 ± 150.50	50276.67 ± 190.85	47833.83 ± 175.65	32997.83 ± 180.90
	指标	67.59 ± 1.00	60.90 ± 0.85	87.74 ± 0.50	53.79 ± 0.45
LADIES+f+d	批次	11.70 ± 0.50	13.03 ± 0.65	11.00 ± 0.55	**8.36 ± 0.45**
	训练	101075.36 ± 810.45	129564.43 ± 1250.35	**247618.18 ± 1620.65**	**122857.42 ± 1035.25**
	指标	68.07 ± 0.45	60.00 ± 0.55	**89.43 ± 0.55**	62.60 ± 0.45
MQ-LADIES+f+d	批次	3.63 ± 1.20	3.88 ± 1.35	2.44 ± 0.40	**2.23 ± 0.85**
	训练	32675.67 ± 145.85	40188.03 ± 175.90	**56834.54 ± 150.55**	**34087.17 ± 125.50**
	指标	66.47 ± 0.25	58.80 ± 0.85	**88.83 ± 0.60**	61.60 ± 0.20

致谢

本研究获得韩国政府（MSIT）资助的信息通信技术规划与评估院（IITP）项目支持（编号：2021- 0-00859，面向智能处理大图的分布式图数据库管理系统开发）。

本研究获得韩国政府（MSIT）资助的信息通信技术规划与评估院（IITP）项目支持（编号：RS-2022-00155911，人工智能融合创新人才培养项目（庆熙大学））。

参考文献

[1] I. Chami、S. Abu-El-Haija、B. Perozzi、C. Ré 和 K. Murphy，“图上的机器学习：模型与综合分类法”，《机器学习研究杂志》，第 23 卷，第 1 期，第 3840–3903 页，2022年。[2] W. Hamilton、Z. Ying 和 J. Leskovec，“大型图上的归纳式表示学习”，《神经信息处理系统进展》，第 30 卷，2017年。[3] P. Velickovic、G. Cucurull、A. Casanova、A. Romero、P. Lio、Y. Bengio 等，“图注意力网络”，stat，第 1050 卷，48 550，2017 20，10no. pp. – 。[4] D. Zou、Z. Hu、Y. Wang、S. Jiang、Y. Sun 和 Q. Gu，“用于训练深层大型图卷积网络的层依赖重要性采样”，《神经信息处理系统进展》，第 32 卷，2019年。[5] J. Chen、T. Ma 和 C. Xiao，“FastGCN：通过重要性采样利用图卷积网络进行快速学习”，载于《国际学习表征会议》。国际学习表征会议，ICLR，2018年。[6] A. Strokach、D. Becerra、C. Corbi-Verge、A. Perez-Riba 和 P. M. Kim，“使用深度图神经网络进行快速灵活的蛋白质设计”，《细胞系统》，第 11 卷，第 4 期，第 402–411 页，2020年。[7] W. Fan、Y. Ma、Q. Li、Y. He、E. Zhao、J. Tang 和 D. Yin，“用于社交推荐的图神经网络”，载于《万维网大会》，2019年，第 417–426 页。[8] N. Park、A. Kan、X. L. Dong、T. Zhao 和 C. Faloutsos，“使用图神经网络估计知识图谱中的节点重要性”，载于《第 25 届 ACM SIGKDD国际知识发现与数据挖掘大会论文集》，2019年，第 596–606 页。[9] S. Gandhi 和 A. P. Iyer，“P3：大规模分布式深度图学习”，载于第 15 届 {USENIX} 操作系统设计与实现研讨会（{OSDI} 21），2021年，第 551–568 页。[10] J. Thorpe、Y. Qiao、J. Eyolfson、S. Teng、G. Hu、Z. Jia、J. Wei、K. Vora、R. Netravali、M. Kim 等，“Dorylus：使用分布式 {中央处理器} 服务器和无服务器线程实现经济、可扩展且准确的{图神经网络} 训练”，载于第 15 届 USENIX 操作系统设计与实现研讨会（OSDI 21），2021年，第 495–514 页。[11] I. F. Ilyas、T. Rekatsinas、V. Konda、J. Pound、X. Qi 和 M. Soliman，“Saga：用于持续构建并大规模提供知识服务的平台”，载于《2022年数据管理国际会议论文集》，2022年，第 2259–2272 页。[12] Z. Lin、C. Li、Y. Miao、Y. Liu 和 Y. Xu，“PaGraph：通过计算感知缓存扩展大型图上的图神经网络训练”，载于《第 11 届 ACM云计算研讨会论文集》，2020年，第 401–415 页。[13] R. Chen、J. Shi、Y. Chen、B. Zang、H. Guan 和 H. Chen，“PowerLyra：偏斜图上的差异化图计算与分区”，《ACM并行计算汇刊》（TOPC），第 5 卷，第 3 期，第 1–39 页，2019年。[14] Z. Zhang、P. Cui 和 W. Zhu，“图上的深度学习：综述”，《IEEE知识与数据工程汇刊》，第 34 卷，第 1 期，第 249–270 页，2020年。[15] J. Chen、J. Zhu 和 L. Song，“采用方差缩减的图卷积网络随机训练”，载于《国际机器学习会议》。PMLR，2018年，第 942–950 页。[16] W. Huang、T. Zhang、Y. Rong 和 J. Huang，“面向快速图表示学习的自适应采样”，《神经信息处理系统进展》，第 31 卷，2018年。

[17]H. Zeng、H. Zhou、A. Srivastava、R. Kannan 和 V. Prasanna，《GraphSAINT：基于图采样的归纳学习方法》，载于国际学习表征会议，2020年。 [在线]。获取地址：https://openreview.net/forum?id=BJe8pkHFwS [18] J. Dong、D. Zheng、L. F. Yang 和 G. Karypis，《面向超大规模图的 CPU-GPU 混合训练的全局邻居采样》，载于第 27 届 ACM SIGKDD 知识发现与数据挖掘会议论文集，KDD ’21 系列。美国纽约州纽约市：计算机协会，2021年，第 289–299 页。 [在线]。获取地址：https://doi.org/ 10 1145 3447548 3467437 . / . K. Huang、[19] J. Zhai、Z. Zheng、Y. Yi 和 X. Shen，《理解并弥合当前图神经网络性能优化中的差距》，载于第 26 届 ACM SIGPLAN 并行编程原理与实践研讨会论文集，2021年，第 119–132 页。[20] Z. Jia、S. Lin、M. Gao、M. Zaharia 和 A. Aiken，《借助 ROC 提升图神经网络的准确率、可扩展性和性能》，《机器学习与系统会议论文集》，第 2 卷，第 187–198 页，2020年。[21] D. Zheng、M. Wang、Q. Gan、X. Song、Z. Zhang 和 G. Karypis，《借助深度图学习库实现可扩展的图神经网络》，第 1141 1142, – 页。在 2021 [ ] 10 1145 3437963 线。获取地址：https://doi.org/ . / . 3441663 [22] S. W. Min、K. Wu、S. Huang、M. Hidayetoglu、J. Xiong、E. Ebrahimi、D. Chen 和 W.-m. Hwu，《PyTorch-direct：通过不规则访问，为超大规模图神经网络训练实现以图形处理器为中心的数据访问》，arXiv 电子印本，第 arXiv–2101 页，2021年。[23] T. Kaler、N. Stathas、A. Ouyang、A.-S. Iliopoulos、T. Schardl、C. E. Leiserson 和 J. Chen，《利用快速采样和流水线化加速图神经网络的训练与推理》，《机器学习与系统会议论文集》，第 4 卷，第 172–189 页，2022年。[24] Z. Wu、S. Pan、F. Chen、G. Long、C. Zhang 和 S. Y. Philip，《图神经网络综合综述》，《IEEE 神经网络与学习系统汇刊》，第 32 卷，第 1 期，第 4–24 页，2020年。[25] D. Zheng、X. Song、C. Yang、D. LaSalle 和 G. Karypis，《面向十亿规模异构图的分布式混合 CPU 和 GPU 训练》，载于第 28 届 ACM SIGKDD 知识发现与数据挖掘会议论文集，2022年，第 4582–4591 页。[26] L. Van der Perre、L. Liu 和 E. G. Larsson，《大规模多输入多输出的高效 DSP 与电路架构：现状与未来方向》，《IEEE 信号处理汇刊》，第 66 卷，4736，2018 18，第 4717 页，期号。[27] G. Sheng、J. Su、C. Huang 和 C. Wu，《MSPipe：通过感知陈旧度的流水线实现高效时序图神经网络训练》，载于第 30 届 ACM SIGKDD 知识发现与数据挖掘会议论文集，2024年，第 2651–2662 页。[28] D. Zou、Z. Hu、Y. Wang、S. Jiang、Y. Sun 和 Q. Gu，《用于训练深层大型图卷积网络的层依赖重要性采样》，载于《神经信息处理系统进展》第 32 卷：神经信息处理系统年度会议，NeurIPS，2019年。[29] Y. Chen、T. Xu、D. Hakkani-Tur、D. Jin、Y. Yang 和 R. Zhu，《校准并去偏图卷积网络的逐层采样》，《机器学习研究汇刊》，2023年。 [在线]。获取地址：https://openreview.net/forum?id=JyKNuoZGux [30] J. Mohoney、R. Waleffe、H. Xu、T. Rekatsinas 和 S. Venkataraman，《Marius：在单台机器上学习大规模图嵌入》，载于第 15 届 {USENIX} 操作系统设计与实现研讨会（{OSDI} 21），2021年，第 533–549 页。[31] J. Chen、Z. Chen 和 X. Qian，《GNNPipe：通过流水线模型并行扩展深层图神经网络训练》，arXiv 预印本 arXiv:2308.10087，2023年。[32] C. Wan、Y. Li、C. R. Wolfe、A. Kyrillidis、N. S. Kim 和 Y. Lin，《PipeGCN：通过流水线特征通信高效进行图卷积网络全图训练》，载于第 10 届国际学习表征会议（ICLR 2022），2022年。[33] M. Fey 和 J. E. Lenssen，《使用 PyTorch Geometric 快速进行图表示学习》，2019年。[34] P. S. Efraimidis 和 P. G. Spirakis，《使用蓄水池的加权随机采样》，《信息处理快报》，第 97 卷，185，2006 5，第 181 页，期号。[35] T. N. Kipf 和 M. Welling，《利用图卷积网络进行半监督分类》，载于国际学习表征会议，2016年。[36] L. Ma、Z. Yang、Y. Miao、J. Xue、M. Wu、L. Zhou 和 Y. Dai，《{NeuGraph}：大型图上的并行深度神经网络计算》，载于 2019 年 USENIX 年度技术会议（USENIX ATC 19），2019年，第 443–458 页。

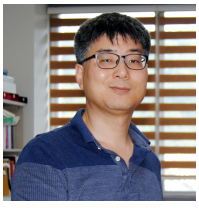
[37] Y. Shao、H. Li、X. Gu、H. Yin、Y. Li、X. Miao、W. Zhang、B. Cui 和 L. Chen，《分布式图神经网络训练：综述》，《ACM Computing Surveys》，第 56 卷，第 8 期，第 1–39 页，2024年。[38] W. Hu、M. Fey、M. Zitnik、Y. Dong、H. Ren、B. Liu、M. Catasta 和 J. Leskovec，《开放图基准：用于图上机器学习的数据集》，载于第 34 届神经信息处理系统国际会议论文集，NIPS '20 系列。美国纽约州雷德胡克：Curran Associates Inc.，2020年。



IRFAN ULLAH 获得了巴基斯坦 NUST 计算机科学硕士学位。目前，他正在韩国庆熙大学攻读计算机科学博士学位，并在该校计算机科学与工程系担任研究助理。

他的研究兴趣包括图神经网络、机器学习、深度学习、大数据分析、分布式计算、社会计算、自然语言处理、操作系统设计和存储系统优化。他参与了多个研究项目，涉及可扩展图神经网络训练、存储感知计算、分布式图学习和社交媒体数据分析。他的研究成果发表于高影响力期刊和国际会议，并拥有多项与自然语言处理和图神经网络相关的专利。

此前，他曾在 NUST、联邦乌尔都艺术、科学与技术大学（FUUAST）以及巴基斯坦跨国公司 Knowledge Platform 担任研究和教学职务。他还积极参与开源项目，并发表过关于编程与开发、自然语言处理、分布式计算、机器学习以及基于图的机器学习的演讲并提供相关教程。



Young-Koo Lee（IEEE会员）于1992年、1994年和2002年分别获得韩国大田韩国科学技术院（KAIST）计算机科学学士、硕士和博士学位。

2002年至2004年，他先后在伊利诺伊大学厄巴纳-香槟分校担任博士后研究员，并在KAIST先进信息技术研究中心（AITrc）担任博士后研究员。2004年，他加入庆熙大学韩国国际校区，并在此担任过多个学术职务。2004年至2010年，他担任助理教授；2010年至2015年，担任副教授；自2015年起，他一直在软件学院担任教授。他的研究方向包括图神经网络（GNN）、数据挖掘、在线分析处理、查询优化和大数据处理。

Lee教授在顶级期刊和会议上发表了大量研究论文。他积极推动数据科学和基于GNN的学习方法的发展。他是IEEE会员，并因在大规模图处理和深度学习应用方面的贡献而受到认可。