



一种面向深度图神经网络推理的高能效三维堆叠架构加速器

隆中天^{1,2,3,4}, 张宇^{1,2,3,4*}, 付宇涛^{1,2,3,4}, 牛齐元^{1,2,3,4}, 吕顺森^{1,2,3,4}, 翟嘉琪^{1,2,3,4},
赵文举^{1,2,3,4}, 赵英淇^{1,2,3,4}, 赵进^{1,2,3,4}, 金海^{1,2,3,4}

1. 华中科技大学大数据技术与系统国家地方联合工程研究中心, 武汉 430074

2. 华中科技大学服务计算技术与系统教育部重点实验室, 武汉 430074

3. 华中科技大学集群与网格计算湖北省重点实验室, 武汉 430074

4. 华中科技大学计算机科学与技术学院, 武汉 430074

* 通信作者. E-mail: zhyu@hust.edu.cn

收稿日期: 2025-09-03; 修回日期: 2025-11-12; 接受日期: 2026-01-04; 网络出版日期: 2026-05-20

国家重点研发计划 (批准号: 2024YFB4504200) 和国家自然科学基金 (批准号: 62472183) 资助项目

摘要 三维存内计算 (3D compute-in-memory, 3D-CIM) 架构凭借高带宽与强并行性, 为图神经网络 (graph neural network, GNN) 推理加速提供了重要机遇. 然而, 现有 3D-CIM 加速器在处理深度图神经网络 (deep graph neural network, DeGNN) 时仍面临两方面挑战: (1) 对逐层稀疏化缺乏高效支持, 零值导致的冗余访存与无效计算难以规避, 造成带宽利用率下降并增加延迟; (2) 对跨层残差依赖缺乏原生支持, 频繁数据搬移带来额外存储与通信开销, 进一步加剧推理延迟. 为此, 本文提出高能效的 3D-CIM 加速器 G3DMA, 通过软硬件协同实现对 DeGNN 的高效支持. 在稀疏编码方面, 采用差异化压缩: 邻接矩阵使用双位图稀疏表示 (DBSR), 节点特征采用 bitmap-value 分离的块稀疏表示 (BSR), 在提高压缩率与索引效率的同时显著降低 DRAM 访问开销. 在执行模型方面, 设计“组合-中间聚合-残差累加”三阶段机制, 结合稀疏感知调度与零值跳过, 以避免中间结果全量写回并减少无效计算. 在硬件方面, G3DMA 于近存逻辑层集成轻量计算阵列与专用编解码单元, 高效支持 DBSR/BSR 处理与按块访存; 同时实现三阶段数据通路流水控制, 并提供残差友好的累加路径与跨-vault 低开销路由. 实验结果表明, G3DMA 在多种 GNN 模型与标准数据集上相较 CPU 与 GPU 分别实现 6007.41 $\times$ 和 106.28 $\times$ 的加速. 相较最新的 SOTA 加速器 HyGCN, GCIM, GCNim 与 SGCN 分别实现 26.27 $\times$, 11.93 $\times$, 1.46 $\times$ 和 2.42 $\times$ 的性能提升, 并在能效上显著超越现有方案.

关键词 深度图神经网络, 硬件加速器, 推理加速, 稀疏压缩, 三维存内计算架构

1 引言

卷积神经网络 (convolutional neural network, CNN) 主要处理欧几里得数据, 例如一维文本序列^[1] 和二维图像^[2,3], 但在推广至非欧几里得数据方面存在局限. 由边和顶点组成的图是一种常见的非欧几里得数据结构, 具

引用格式: 隆中天, 张宇, 付宇涛, 等. 一种面向深度图神经网络推理的高能效三维堆叠架构加速器. 中国科学: 信息科学, 2026, 56: 1330–1347, doi: 10.1360/SSI-2025-0381

Long Z T, Zhang Y, Fu Y T, et al. A high-efficiency 3D-stacked accelerator for deep graph neural network inference. Sci Sin Inform, 2026, 56: 1330–1347, doi: 10.1360/SSI-2025-0381

有更强的表达能力. 为突破这一瓶颈, 提出了图神经网络 (graph neural network, GNN) 作为 CNN^[4] 的变体, 利用节点间的邻接关系学习嵌入表示. 凭借在节点分类^[5,6]、链接预测^[7] 以及推荐系统^[8~10] 等任务中的优异性能, GNN 已成为图表示学习的主流方法.

GNN 的每一层计算通常包括聚合和组合两个阶段. 在聚合阶段, 节点收集邻居节点的特征向量, 这涉及邻接矩阵或邻接表的遍历; 在组合阶段, 通常通过多层感知机 (multilayer perceptron, MLP) 或其他神经网络结构对特征进行更新, 可抽象为矩阵-向量乘. 这两个阶段展现出截然不同的访存模式: 聚合阶段受图结构稀疏性与不规则性影响, 访存模式分散且难以预测; 而组合阶段的矩阵计算较为规则, 访存模式相对有序. 这种不一致性使传统 CPU 与 GPU 架构难以兼顾两类需求, 尤其是聚合阶段的不规则访存易导致计算延迟上升与能耗增加, 这也推动了专用 GNN 加速器的研究.

随着应用场景^[9,11,12] 日益多样, 对 GNN 的表达能力、泛化性与计算性能提出了更高要求. 研究者提出了多种 GNN 变体^[13~15], 并逐渐加深网络层数以提升建模能力. 然而, 当层数过深时, 节点表示趋于一致, 出现过平滑 (over-smoothing)^[16] 现象, 使得细粒度信息丢失并削弱模型判别力. 为缓解过平滑, 已有方法主要分为两类: 一类通过引入残差连接 (residual connection)^[17,18] 等结构性改进以增强信息流动; 另一类通过设计度量指标以约束训练. 其中, 残差连接在深层结构中效果显著, 成为深度图神经网络^[18,19] 的重要设计思路.

然而, DeGNN 在推理过程中对计算资源、存储容量与带宽的需求更为苛刻, 硬件加速难度显著增加. 3D 堆叠内存 (3D-stacked memory)^[20] 通过在 DRAM 堆叠中集成逻辑层, 提供了极高的带宽与并行度, 为满足 DeGNN 推理需求提供了新的机遇. 尽管已有多种三维存内计算 (3D compute-in-memory, 3D-CIM)^[20] 加速器与 GNN 专用硬件^[21,22], 但在深层网络场景下仍存在多方面不足: 一方面, 它们未能充分利用深层结构中逐渐增强的中间稀疏性, 导致大量零值乘加操作, 带来冗余计算与能耗浪费; 另一方面, 部分方案虽利用了稀疏性, 却依赖额外的不规则访存加载索引与特征, 增加访存开销并降低带宽利用率; 此外, 在支持残差连接时往往依赖中间结果的全量写回与重读, 扩大了片上存储需求并加剧 DRAM 带宽压力. 因此, 现有方案缺乏一种能够同时兼顾稀疏感知、访存友好与残差高效支持的统一优化框架.

针对上述 DeGNN 在深层结构下产生的冗余计算、稀疏访存低效及残差支持不足等问题, 本文提出了一种基于 3D-CIM 架构的高能效加速器 G3DMA. 通过软硬件协同设计, G3DMA 充分发挥了三维堆叠存储 (3D stacked memory) 在带宽和并行度方面的优势. 在软件层面, 我们针对异质稀疏性提出差异化表示机制: 邻接矩阵采用双位图稀疏表示 (dual-bitmap sparse representation, DBSR), 以双重索引兼顾压缩率与解码效率; 节点特征则采用 bitmap-value 分离的块稀疏表示 (block sparse representation, BSR), 以消除填充冗余并提升带宽利用率. 在此基础上, 我们设计了三阶段执行模型 (组合-中间聚合-残差累加), 并结合稀疏感知调度与零值跳过机制, 以避免中间结果的全量物化、减少片上/片外写回, 并有效跳过无效访存与乘加操作. 值得注意的是, 该执行模式不仅提升了稀疏计算效率, 还能够在硬件中自然支持残差连接, 从而增强 DeGNN 在深层结构下的可扩展性. 在硬件层面, G3DMA 集成轻量级计算与编解码单元, 高效支持 DBSR 与 BSR 的并行处理, 并在此之上于硬件层面承载三阶段执行模型, 实现了对软件机制的高效支撑与软硬件协同优化. 我们在多个 GNN 模型和主流图数据集上对 G3DMA 进行了评估, 实验结果表明, 与现有方法相比, G3DMA 在推理延迟和能耗方面均取得了显著降低.

本文主要贡献如下.

- 提出差异化稀疏表示与高效存取机制: 邻接矩阵采用双位图稀疏表示 (DBSR), 节点特征采用 bitmap-value 分离的 BSR, 在提升压缩率与索引/解码效率的同时, 显著降低 DRAM 访问与数据搬移开销, 从而有效提升带宽利用率.
- 设计三阶段执行模型及稀疏感知优化机制: 我们提出的“组合-中间聚合-残差累加”执行模型, 避免中间结果的全量物化, 通过按需传输与行级切分减少片上/片外写回. 同时, 结合稀疏感知调度与零值跳过机制, 有效规避无效访存与乘加操作. 此外, 硬件中对残差连接的自然支持进一步提升了深层 DeGNN 的可扩展性与能效.
- 提出 G3DMA 架构: 一种基于 3D-CIM 的高能效加速器, 在 DRAM 近侧集成轻量级计算与编解码单元, 高效支持 DBSR 与 BSR 稀疏格式, 并在硬件层面承载“组合-中间聚合-残差累加”执行模型. 该设计在提升

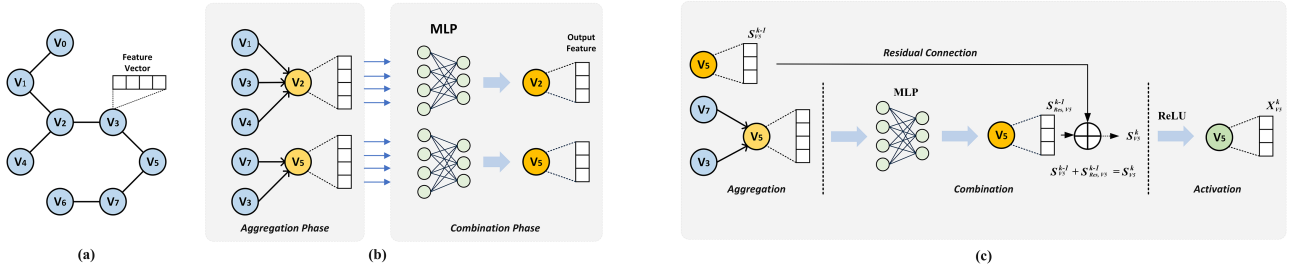


图 1 (网络版彩图) DeGNN 推理示例. (a) 图结构; (b) GCN 推理的两个阶段; (c) 节点 V5 的 DeGNN 推理示例.

Figure 1 (Color online) DeGNN inference example. (a) Graph structure; (b) the two phases of GCN inference; (c) the DeGNN example on node V5.

稀疏计算效率的同时, 自然支持残差连接, 从而增强深层 DeGNN 的可扩展性与能效.

● 实现并系统评估 G3DMA: 在多种 GNN 模型与基准数据集上, G3DMA 相比 CPU 与 GPU 实现分别获得 $6007.41\times$ 与 $106.28\times$ 的加速; 相比 HyGCN, GCIM, GCNim 与 SGCN 等代表性加速器, 分别实现 $26.27\times$, $11.93\times$, $1.46\times$ 与 $2.42\times$ 的性能提升, 并显著降低能耗.

本文的其余部分组织如下. 第 2 节回顾超图神经网络的研究背景并分析现有方法的局限性; 第 3 节阐述所提出的核心方法; 第 4 节描述硬件的整体设计与具体实现; 第 5 节呈现实验设置与结果分析; 第 6 节讨论相关工作; 最后, 第 7 节对实验结论进行了总结.

2 背景和动机

2.1 深度图神经网络概念

图神经网络是一类专门用于处理图结构数据的学习模型, 是传统神经网络 (neural network, NN) 的变体. 通过学习节点的嵌入表示并利用节点之间的关系, GNN 能够有效捕获局部与全局信息, 适用于节点分类、图分类等多种图结构数据分析任务. 目前, GNN 已在社交网络 [5, 6]、推荐系统 [8~10]、生物信息 [23, 24] 等领域展现出优异性能.

GNN 通过聚合与组合两个过程学习节点表示, 以捕获图的结构特征. GNN 通常采用多层结构, 每层包括聚合与组合两个关键步骤, 如图 1 所示. 在聚合步骤中, 节点从其邻居收集特征信息, 需要遍历邻接矩阵或邻接表以访问邻居节点的特征向量, 节点通过汇总邻居特征生成新的表示; 在组合步骤中, 通常采用多层感知机或其他神经网络结构进一步更新节点特征. 图卷积网络 (graph convolutional network, GCN) [13]、图同构网络 (graph isomorphic network, GIN) [14] 和 GraphSAGE [15] 是常见的 GNN 模型, 其计算可统一表示为

$$\mathbf{X}^k = \sigma(\tilde{\mathbf{A}}\mathbf{X}^{k-1}\mathbf{W}^{k-1}), \quad (1)$$

其中, $\tilde{\mathbf{A}}$ 为归一化后的邻接矩阵, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$, $\mathbf{I}$ 为单位矩阵, $\mathbf{D}$ 为度矩阵. $\mathbf{X}^{(k-1)}$ 与 $\mathbf{W}^{(k-1)}$ 分别表示第 $(k-1)$ 层的特征矩阵与权重矩阵, σ 为激活函数 (如 ReLU). 经过第 k 层计算后, 节点特征包含其 k -跳邻居的结构信息, 从而更全面地表达图结构, 逐层抽象以学习高层次表示.

为增强全局信息感知, 近年来 GNN 采用更深的网络结构. 引入残差连接 (residual connection) [17, 18] 可有效缓解深度图神经网络 (deep GNN, DeGNN) [18, 19] 中的过平滑问题, 残差结构 GNN 的计算表达式如下:

$$\begin{aligned} \mathbf{S}^k &= \mathbf{S}_{res}^{k-1} + \mathbf{S}^{k-1}, \\ \mathbf{S}_{res}^{k-1} &= \tilde{\mathbf{A}}\mathbf{X}^{k-1}\mathbf{W}^{k-1}, \\ \mathbf{X}^k &= \sigma(\mathbf{S}^k). \end{aligned} \quad (2)$$

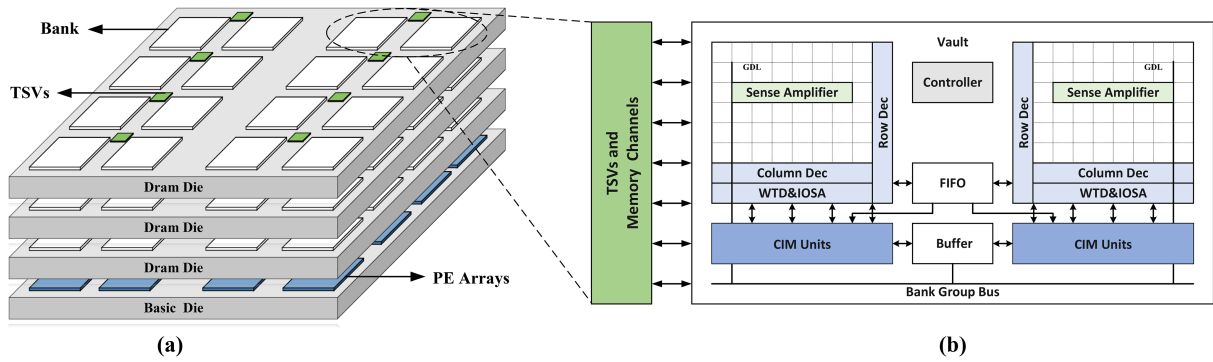


图 2 (网络版彩图) 典型的 3D 堆叠内存架构与垂直 vault 设计. (a) 典型 3D 堆叠内存架构; (b) DRAM 芯片中的垂直 vault 设计.

Figure 2 (Color online) A typical 3D-stacked memory architecture and vertical vault design. (a) A typical 3D-stacked memory architecture; (b) the design of vertical vaults in DRAM dies.

2.2 三维存内计算架构

三维堆叠存储^[20]是一种提升存储密度与性能的创新技术. 与传统二维平面存储结构相比, 3D 堆叠存储通过垂直堆叠增加存储单元数量, 实现更高容量与带宽. 随着图数据集规模持续扩大, 数据传输速率与存储带宽已成为 GNN 加速的瓶颈, 而基于 3D 堆叠内存架构的 GNN 加速器可提供更高存储密度与数据传输速率, 其增加的带宽支持大规模并行计算, 满足先进计算任务需求.

如图 2(a) 所示, 3D 堆叠存储架构通常包含基底逻辑层与多个堆叠的 DRAM 层, 通过硅通孔 (through-silicon vias, TSV) 互连形成整体结构, 组织为多个垂直分区, 每个分区包含 DRAM 层的若干部分, 并通过各层的内存控制器访问, 实现高速并行访存.

图 2(b) 展示了 DRAM die 的内部结构, 其通道内包含多个 vault (即由多个 HBM 通道在垂直方向上对齐堆叠形成的并行访问单元, 用于抽象表示贯穿多层 DRAM 的垂直访问路径与数据通道). 每个 vault 由若干 bank group 与 TSV I/O 电路组成, bank group 内含多个 bank 以提升数据访问效率 (其 bank-level 带宽约为 1024 GB/s, 显著高于 channel-level 带宽约 256 GB/s, 体现出垂直并行访问优势). 每个 bank 包含 DRAM 单元阵列、感应放大器以及行列译码器, 部分 bank 配备存内计算单元, 集成 SRAM 缓存与处理单元以支持近存计算. DRAM 层提供快速访存, 逻辑层 (basic die) 增强可编程性与计算能力.

2.3 现有方案存在的不足

现有 GCN 加速器系统 (如 SGCN^[25] 与 GCIM^[26]) 在 DeGNN 推理过程中, 普遍面临计算操作数冗余、内存空间浪费以及跨层数据访问效率低下等问题. 其中, SGCN 固定采用 Aggregation-first 的计算顺序, 缺乏顺序切换机制, 导致在特征维度不变的情况下无法通过先组合后聚合减少计算操作数; 同时, 其所采用的 BEICSR 压缩格式虽提升了访存效率, 却因 in-place 设计无法释放未使用内存空间, 限制了存储利用率提升. 作为代表性的 3D-CIM 架构加速器, GCIM 将组合与聚合操作分别映射至基底逻辑层与 DRAM 层以支持并行执行, 然而受限于远程随机访存开销与引擎间负载不均, 其整体性能与能效仍未达最优, 成为进一步提升系统效率的关键瓶颈.

计算操作数冗余问题. SGCN 与 GCIM 均采用 Aggregation-first 的计算顺序, 即先执行聚合操作, 再进行组合计算. 该设计虽然有利于模型结构简化, 但在现代深层 GCN 中已暴露出显著的冗余计算问题. 由于各层节点特征维度通常保持一致, Aggregation-first 无法通过“先组合再聚合”来压缩中间表示. 相比之下, 若采用 Combination-first 策略, 在输出维度小于输入维度的场景下, 可先通过组合操作降低特征维度, 进而在聚合阶段显著减少乘法操作数量, 降低整体计算复杂度. 然而, 现有架构缺乏对计算顺序的灵活支持, 尤其 GCIM 将聚合与组合物理分布于 DRAM 层与基底层, 更难以实现计算流程切换, 导致潜在的操作数优化空间难以发挥.

内存空间浪费问题. 尽管 SGCN 提出了 Bitmap-index Embedded In-place CSR (BEICSR)^[25] 格式, 通过

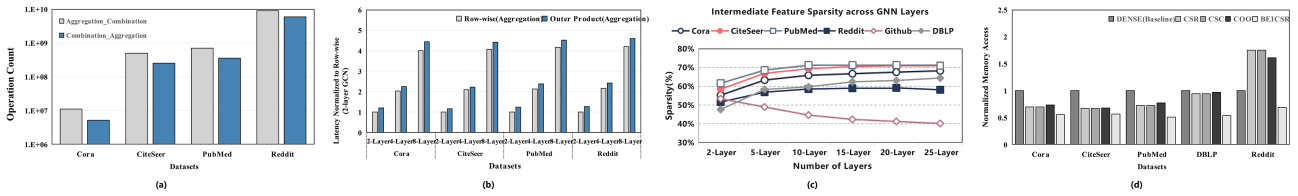


图 3 (网络版彩图) (a) 先组合后聚合与先聚合后组合两种执行模式下的操作数比较; (b) row-wise aggregation 与 outer product aggregation 在不同数据集和 GNN 层数下的延迟表现; (c) 不同数据集的 GNN 各层中间特征稀疏度; (d) 不同数据集下各压缩格式对内存访问的影响。

Figure 3 (Color online) (a) Operation counts for combination-then-aggregation versus aggregation-then-combination; (b) latency of row-wise versus outer product aggregation across datasets and GNN layers; (c) intermediate feature sparsity across GNN layers for different datasets; (d) impact of different compression formats on memory access across datasets.

bitmap 索引压缩特征矩阵以减少内存访问次数,但其采用的 in-place compression 设计要求每行预留与未压缩相同的内存空间。具体来说, BEICSR 虽可有效减少特征索引存储开销,但为了确保 cacheline 对齐和并行写入,其行内存布局需保持固定长度,导致大量未使用空间无法释放,未能真正降低总内存容量占用。这种空间浪费在多层 GCN 推理时愈加显著,成为限制存储利用率提升的主要瓶颈。

稀疏适配性不足。此外, BEICSR 对不同层级、不同顶点及不同数据集的稀疏度变化缺乏动态适配性。SGCN 采用固定切片大小 (如 $C = 96$) 优化缓存局部性,虽然可保证内存访问对齐,但实际稀疏分布常常具有显著波动,固定切片配置无法保证在所有场景下均实现最佳访存效率,虽然内存访问可以对齐,但为了保证对齐又会浪费大量存储空间,未能充分发挥压缩带来的容量节省优势。

远程访存与负载不均问题。GCIM 利用 3D 堆叠架构分别在 DRAM 层和基底层执行聚合与组合操作,以实现并行计算与延迟隐藏。然而,由于图结构与特征分布的非结构化特性,邻居节点的数据往往分布在不同 vault 或 bank 中,导致聚合过程中频繁触发远程随机访存,相较本地访问产生显著延迟。此外, GCIM 的模块划分方式在面对动态计算负载变化时,不同计算单元之间易出现处理时间差异,快速模块需等待慢速模块完成计算,叠加跨模块通信开销,限制了整体并行执行效率的提升。

2.4 设计动机与创新

两阶段执行模式。如图 3(a) 所示,主流 GNN 加速器普遍采用“先组合、后聚合”的两阶段执行顺序,通过在 Combination 阶段对节点特征进行线性变换以降低维度,从而在 Aggregation 阶段以较小的特征宽度执行邻居累加操作,以减少整体计算量与内存带宽压力。该策略在浅层模型中表现出良好的效率,尤其适用于特征维度较高、邻居数量有限的场景。从计算模式来看,现有设计主要分为两类:一类在 Combination 与 Aggregation 阶段均采用行乘法 (row-wise multiplication, 见图 4(b))^[25,26],另一类则在 Combination 阶段采用行乘法,在 Aggregation 阶段采用外积乘法 (outer-product multiplication, 见图 4(a))^[27]。然而,在面向 3D 堆叠近内存计算 (3D-CIM) 架构的深层 GNN 推理中,这两类范式均面临关键挑战。

具体而言,纯行乘法模式虽然可逐行生成最终输出、避免对中间矩阵的全局累加,但其执行过程高度依赖计算顺序与数据路径,难以灵活插入残差融合逻辑,限制了跨层数据重用与调度自由度。同时,该模式在 vault 间缺乏高效的数据重组能力,易引发片上缓存压力与远程访存冲突,进而制约吞吐率。

另一方面,纯外积范式具有较好的输入特征重用性,但需要完整构建中间矩阵以支持后续累加,导致片上资源占用显著增加,并引入大量中间数据的搬移与缓存访问,难以满足 3D-CIM 架构中对数据局部性与带宽利用的高要求。随着模型深度的提升,残差连接机制在性能与精度中的作用日益突出,进一步加剧了传统两阶段执行模式在计算解耦性、资源高效利用与异构结构适配性上的局限。因此,亟需设计一种更具架构亲和性与残差融合灵活性的执行模型,以支持高效的 DeGNN 推理。

稀疏存储适配。如图 3(c) 所示,随着 GNN 层数的加深,DeGNN 在 Cora^[28], CiteSeer^[29], PubMed^[30], Reddit^[31], Github^[32] 与 DBLP^[33] 等典型图数据集上呈现出中间特征稀疏性逐层增强的趋势,即非零元素

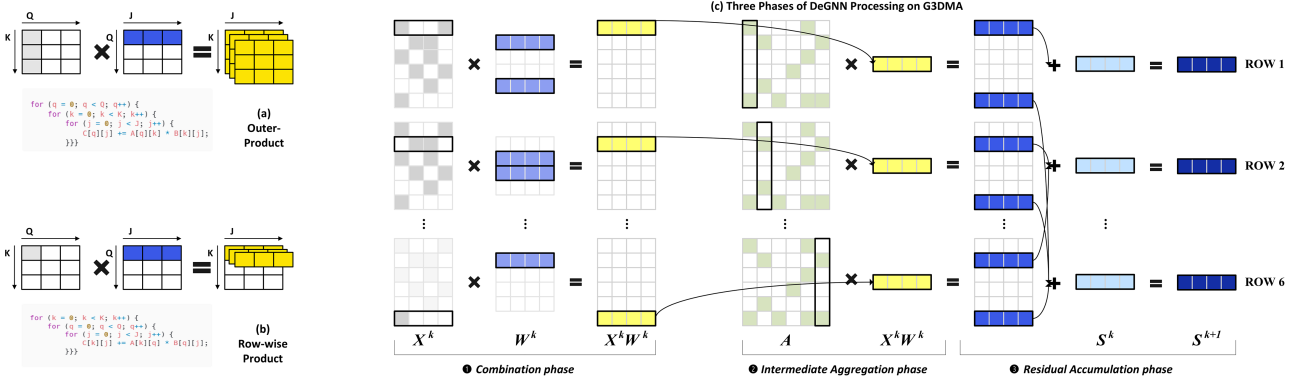


图 4 (网络版彩图) 两种矩阵乘法方式与 G3DMA 中 DeGNN 推理的三阶段流程示意. (a) 外积乘法 (outer product); (b) 行乘法 (row-wise product); (c) 三阶段执行模型.

Figure 4 (Color online) Illustration of two matrix multiplication modes and the three-phase DeGNN execution flow on G3DMA. (a) Outer-product multiplication; (b) row-wise multiplication; (c) three-phase execution model.

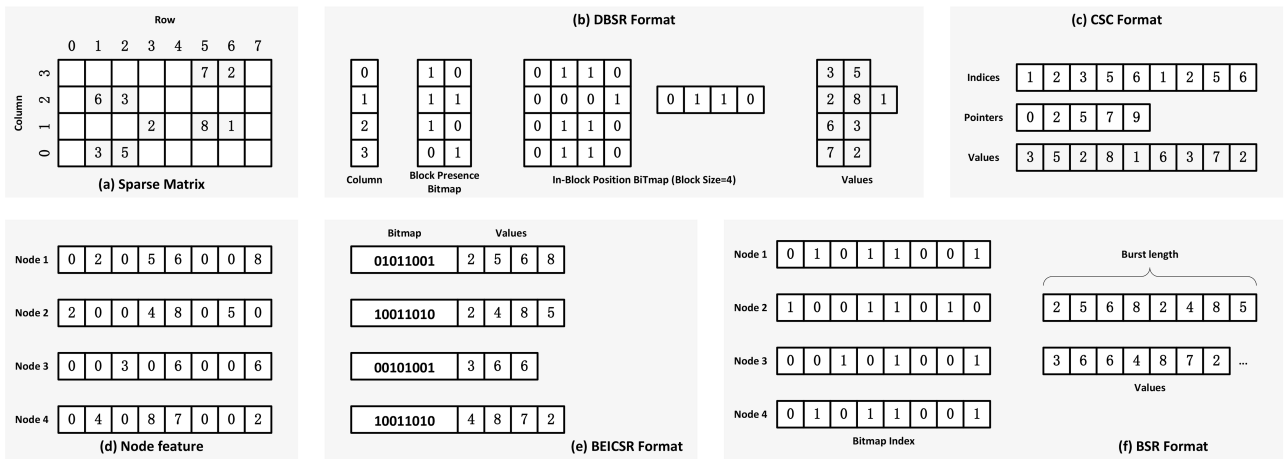


图 5 不同稀疏存储格式对比示意图. (a) 原始稀疏矩阵; (b) 双位图稀疏表示 (DBSR) 格式; (c) 压缩列存储 (CSC) 格式; (d) 节点特征矩阵; (e) 嵌位索引增强 CSR (BEICSR) 格式; (f) 块稀疏表示 (BSR) 格式.

Figure 5 Comparison of different sparse matrix storage formats. (a) Original sparse matrix; (b) dual-bitmap sparse representation (DBSR) format; (c) compressed sparse column (CSC) format; (d) node feature matrix; (e) bitmap-index embedded in-place CSR (BEICSR) format; (f) block sparse representation (BSR) format.

比例持续下降. 此外, 不同数据集间的稀疏性分布差异显著, 且在层间具有动态变化特性. 传统压缩格式 (如 CSC^[34], CSR^[35] 与 COO^[36]) 结构固定、索引开销大, 难以高效适配这一动态稀疏性, 导致在低稀疏度层中出现冗余访存, 在高稀疏度层中则资源利用率偏低. 如图 3(d) 所示, 该问题将显著增加内存访问开销, 限制 DeGNN 的推理效率. 为提升稀疏存储性能, BEICSR 格式^[25] (图 5(e)) 在行内引入 bitmap 索引, 降低了传统索引成本, 并在 DeGNN 中展现出良好的压缩率与访存效率. 然而, 该格式采用固定切片大小的 in-place 存储设计, 尽管有利于访问对齐与并行写入, 但在面对不同层间稀疏分布不均时, 需引入大量 padding, 造成空间浪费, 削弱了整体的存储利用率. 上述问题表明, 现有稀疏格式在动态适应性与结构灵活性方面仍存在局限, 亟需设计一种支持自适应切片与高效解码的新型稀疏存储方案, 以更有效支撑 DeGNN 的高效推理需求.

硬件支持需求. 硬件层面必须为新型执行模型与稀疏存储方案提供支持. 一方面, 架构需要具备灵活的数据路径与调度机制, 以适配更具架构亲和性的执行模型, 从而高效融合残差计算与跨层特征重用. 另一方面, 还需针对新型稀疏格式设计高效的编码与解码机制, 使其能够在层间稀疏度动态变化时仍保持紧凑存储与低开销访问. 只有在执行模型与存储格式两个层面实现协同优化, 硬件才能充分释放 DeGNN 中稀疏性与残差机制的

潜力,并满足 3D-CIM 架构下的高效推理需求.

3 设计方法概述

3.1 压缩格式选择与设计方法

为有效利用深度 GNN 推理过程中节点特征与邻接矩阵的不同稀疏特性,本文提出了差异化压缩方案.

具体而言,首先针对邻接矩阵 A ,采用双位图稀疏表示 (dual-bitmap sparse representation, DBSR) 结构.如图 5(b) 所示, DBSR 首先将稀疏矩阵按 block size 划分,由两类 bitmap 组成: block presence bitmap 用于标记每行中哪些 block 含有非零值, in-block position bitmap 用于记录每个非零 block 内部非零元素所在列的位置.在该结构中,仅对 block presence bitmap 标记为“1”的有效 block,存储其对应的 in-block position bitmap 及非零 value array.以图 5(a) 中的 4×8 稀疏矩阵为例,其中第 1 行被划分为两个 block, block presence bitmap 标记第一个 block 含有非零值.对应的 in-block position bitmap 记录该 block 内非零元素的位置,例如若该 block 的第 2 和第 3 列为非零,则 bitmap 相应位置标“1”.非零值按照块顺序依次存储在 values 中,保证解码时可按块有序恢复.此外, block presence bitmap 作为 block-level valid mask,支持多行并行解码,提升聚合阶段带宽利用率,并减少冗余访存开销.

相比之下,传统 CSC 格式^[34]如图 5(c) 所示,需存储 column_ptr, row_idx 和 value 三部分.在极度稀疏场景下(稀疏度 0.01%~5%),例如在 4×8 邻接矩阵仅包含 1 个非零元素时, CSR 需要存储 5 个 column_ptr (每个占用 2 bits,即 $\log_2 4$), 1 个 col_idx (占用 3 bits,即 $\log_2 8$), 以及对应的 value.总计, CSC 用于存储图结构的开销为 13 bits ($5 \times 2 + 3$).相比之下, DBSR 仅需存储 1 个 2 bits 的 row index, 1 个 2 bits 的 block presence bitmap, 1 个 4 bits 的 in-block position bitmap, 以及对应的 value, 图结构部分总开销为 8 bits ($2 + 2 + 4$).因此, DBSR 的图结构存储开销仅为 CSC 的 61.54%,显著降低了索引存储成本.这表明, DBSR 通过双层 bitmap 结构避免了对 row_idx 的显式存储,在极端稀疏矩阵场景下展现出更高的存储效率与硬件解码友好性.值得一提的是, DBSR 结构亦可应用于行压缩,相较于传统 CSR 格式^[35],同样具备良好的存储效率与解码性能.

此外,针对节点特征 X ,本文借鉴 MEGA 中将 BEICSR 嵌入式 bitmap 与 value 分离存储的设计思想,提出了 bitmap-value 分离的稀疏存储方案.如图 5(e) 所示,传统 BEICSR 将 bitmap 与非零值按固定切片大小 in-place 存储,虽可保证内存对齐与并行写入,但在稀疏度动态变化时,为满足对齐要求需引入大量 padding,导致存储空间浪费.

为解决上述问题,本方案采用如图 5(f) 所示的 BSR (block sparse representation) 分离存储结构,将 bitmap 与 value 分离存放: bitmap 记录非零特征的位置索引, value array 顺序存储非零值.这种设计带来了三方面好处.首先,避免了 BEICSR 固定包长结构带来的 padding 空间浪费,特别适用于深度 GNN 中间特征稀疏度动态变化场景.其次,分离存储结构简化解码逻辑,不需要复杂的对齐与 zero-skipping 控制,易于在硬件中实现高效 pipeline 解码.第三,非零值按 burst length 顺序连续排列存储,提升了 DRAM 访存带宽利用率与整体存储效率.结合邻接矩阵 A 的 DBSR 压缩与节点特征 X 的 BSR 分离存储方案,本设计在 3D-CIM 架构下实现了对深度 GNN 推理中聚合与组合阶段稀疏性的高效支持.

此外,表 1 对比了现有典型稀疏格式 (CSR/CSC, BEICSR) 与本文提出的 DBSR, BSR 的适用对象与性能特征,展示了所提方案在索引效率、内存利用率与稀疏适配性方面的显著优势.

3.2 新型执行模型

虽然 3D 堆叠近内存计算架构通过将处理单元 (PE) 紧邻部署于 DRAM bank,在本地数据访问延迟和带宽利用率方面展现出显著优势,但在深度图神经网络推理中,其性能潜力仍受到图数据本身的稀疏性与不规则性所限制.具体而言,节点特征与图结构往往跨越多个 vault 存储,特别是在聚合阶段频繁触发远程随机访存,严重削弱了近数据处理架构的本地性优势,成为制约系统整体加速效率的关键瓶颈.

表 1 稀疏存储格式在效率与适用性方面的比较.

Table 1 Comparison of sparse storage formats in terms of efficiency and applicability.

Format	Compressed target	Effective sparsity range	Indexing efficiency	Padding overhead elimination	Adaptability to sparsity
CSR/CSC	Adjacency	Ultra-sparse (<5%)	✗	✓	✗
BEICSR [25]	Feature	Moderate (20%~50%)	✓	✗	✗
BSR (this work)	Feature	Mid-to-dense (10%~60%)	✓	✓	✓
DBSR (this work)	Adjacency	Ultra- to moderate-sparse (<50%)	✓✓	✓	✓✓

为应对 DeGNN 推理过程中组合、聚合与残差传播等操作在计算模式与数据依赖特性上的差异, 我们提出了一种面向 3D-CIM 架构的三阶段执行模型, 如图 4(c) 所示. 该模型将推理流程划分为组合 (combination)、中间聚合 (intermediate aggregation) 和残差累加 (residual accumulation) 3 个阶段, 并通过解耦化的数据通路设计、稀疏感知调度机制与跨 -vault 中间结果协调策略, 实现对数据复用性、访存局部性与并行计算效率的系统级优化.

在该模型中, 系统充分考虑特征与结构的稀疏特征, 采用差异化的压缩格式与计算策略以提升整体的数据处理效率与资源利用率. 具体而言, 节点特征矩阵 $\mathbf{X}$ 采用块稀疏行 (block sparse row, BSR) 格式压缩, 显式记录非零特征块的行索引信息, 配合按行乘法 (row-wise product) 模式, 能够跳过零值对应的冗余访存与计算, 从源头上规避无效操作, 形成调零优化 (zero-skipping) 机制; 邻接矩阵 $\mathbf{A}$ 则采用双位图稀疏表示 (dual-bitmap sparse representation, DBSR), 通过两级位图分别标识活跃列和列内非零元素, 兼顾高压缩比与高效索引解码能力, 适配于结构稀疏且访问路径不规则的聚合场景.

在组合阶段, 系统依据 BSR 格式中记录的非零块索引, 逐行解析压缩后的节点特征, 生成对应的权重索引并调度至目标计算路径. 随后采用按行乘法方式, 仅对包含非零块的特征行发起权重访存与乘加运算, 从而逐行生成节点嵌入向量 $\mathbf{XW}$. 该阶段通过稀疏索引实现调零优化机制 (zero-skipping), 有效避免整矩阵解压与零值乘加操作, 在降低冗余计算与访存带宽开销的同时, 提升片上数据路径的利用率与整体能效.

中间聚合阶段延续了组合阶段的稀疏感知执行策略. 系统以组合阶段逐行生成的嵌入向量 $\mathbf{XW}$ 为输入, 结合稀疏邻接矩阵 $\mathbf{A}$ 执行跨邻居的特征聚合操作. 依托 DBSR 格式提供的列索引与列内活跃行信息, 系统可在数据加载与计算过程中高效规避无效路径, 进一步强调零优化效果. 在该阶段, 系统采用外积 (outer-product) 方式构建聚合路径, 对 $\mathbf{A}$ 的每一列与 $\mathbf{XW}$ 中对应行向量执行乘法运算, 生成多个部分和 (partial sum) 中间向量. 为适配 3D-CIM 中 vault 本地缓存容量有限的约束, 系统不保留完整的中间矩阵结果, 而是将其按行切分为可调度的中间向量, 并在后续阶段灵活传输至目标 vault, 缓解片上压力并提升缓存利用率与调度灵活性.

进入残差累加阶段后, 系统依据目标节点的分布信息, 将上述中间向量经由跨 -vault 数据路径传输至相应计算单元, 并与前一层的原始节点特征同步执行逐元素加法操作, 完成聚合结果与残差路径的融合更新. 该阶段融合了远程聚合与残差连接两个关键过程, 不仅提高了数据复用效率, 同时提升了聚合与残差计算的并行性, 显著缓解了访存瓶颈对系统性能的制约. 最终, 融合后的节点特征向量通过压缩编码模块重新编码为 BSR 格式并写回本地 bank, 作为下一层推理计算的高效输入.

需要说明的是, 三阶段模型作为一种通用的执行抽象, 并不依赖于特定的残差结构. 在浅层 GNN 中, Residual Accumulation 阶段退化为仅对聚合结果进行特征累加与写回; 而在采用 Concatenation^[31] 或 Jumping Knowledge^[37] 等跨层特征融合机制的模型中, 该阶段亦可扩展为可配置的特征融合单元 (如拼接或加权聚合), 以保持统一的数据流与执行管线.

3.3 预处理和负载分配策略

DeGNN 在三阶段执行模型下需要同时处理动态稀疏特征和跨层残差连接, 其计算与访存的负载呈现出高度不均衡和跨 -vault 通信频繁的问题. 如果不加以优化, 最慢的 PE 会成为性能瓶颈, 而频繁的跨 -vault 数据路

由则会造成显著的带宽浪费. 因此, 我们在推理前引入轻量级的预处理与负载分配策略, 目标是在不引入复杂离线重排序的前提下, 通过结构感知与压缩感知的分析, 在顶点到 PE 的映射中实现负载均衡、局部性提升与跨层残差友好性. 在预处理阶段, 首先需要为顶点之间建立结构亲和度, 以减少残差阶段的跨 -vault 汇聚通信. 常见的度量方式包括基于邻居集合的 Jaccard 相似度^[38]:

$$\text{Aff}_{\text{Jacc}}(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \quad (3)$$

以及结合 DBSR 压缩结构的加权 Jaccard 度量. 设 $\mathcal{B}$ 为所有块索引, $p_{u,b}$ 表示顶点 u 在块 b 中的有效位置数, 则有

$$\text{Aff}_{\text{DBSR}}(u, v) = \frac{\sum_{b \in \mathcal{B}} \min(p_{u,b}, p_{v,b})}{\sum_{b \in \mathcal{B}} \max(p_{u,b}, p_{v,b})}. \quad (4)$$

在实际实现中, 亲和度计算仅针对共享块的顶点, 以避免全图两两比较的高复杂度. 基于该稀疏计算方式, 我们为每个顶点 i 构建锚点集合 S_i , 通过累积其在共享块上的亲和度, 最终保留 top- m 个最相关的邻居, 以辅助后续映射决策.

随后, 我们利用压缩感知统计量建模各阶段的有效工作量. 具体而言, 定义 $\text{nnzb_X}[i]$ 表示 BSR 格式下顶点 i 特征向量的非零块数, $\text{POP}[i]$ 表示与 i 相关的 DBSR 块的 in-block popcount 总和, $\text{Brow}[i]$ 表示 DBSR 中行 i 的活跃块数. 我们据此定义关键度:

$$\text{Key}[i] = w_1 \cdot \text{nnzb_X}[i] + w_2 \cdot \text{POP}[i] + w_3 \cdot \text{Brow}[i], \quad (5)$$

并以其作为顶点的重要性排序指标.

在映射阶段, 顶点首先按关键度降序排列, 并通过一个维护各 PE 当前负载的优先队列进行调度. 对于顶点 i , 我们优先选择负载最轻的 PE; 若多个候选 PE 的工作量接近, 则进一步利用锚点集 S_i 中的亲和度作为局部性判据, 将其分配到与其高亲和邻居相同的 vault, 从而降低跨 -vault 通信. 在完成分配后, 每个 PE 内部还需对顶点顺序进行重排, 以提升压缩块的连续性和突发访问效率, 常用方法包括基于块的 Hilbert 曲线或贪心式的连续块链路优化.

上述过程的核心步骤如算法 1 所示. 从计算复杂度角度分析, 预处理与映射过程的总体开销近似线性. 主要步骤包括: (1) 对稀疏结构的遍历与统计, 复杂度为 $O(E)$ (其中 E 为非零边或非零块数量); (2) 顶点关键度计算与排序, 复杂度为 $O(V \log V)$ (V 为顶点数); (3) 基于优先队列的负载调度与局部重排, 复杂度约为 $O(V \log P)$ (P 为处理单元数量). 综合可得总体复杂度为 $O(E + V \log V)$. 在典型图神经网络数据集中, E 与 V 通常呈线性关系 (平均度保持常数级别), 因此该预处理流程的总体复杂度近似随图规模线性增长.

算法 1 Compression-aware pre-processing with affinity-guided mapping.

Input: DBSR-form adjacency matrix $\mathbf{A}$, BSR-form feature matrix $\mathbf{X}$, number of PEs P ;

Procedure: Pre-processing and mapping steps;

Output: (1) Vertex-to-vault/PE mapping table, (2) partitioned adjacency $\mathbf{A}$ and features $\mathbf{X}$, (3) reordering tables for block-contiguous access;

- 1: Compute compression-aware statistics for each vertex: $\text{nnzb_X}[i]$, $\text{POP}[i]$, $\text{Brow}[i]$;
 - 2: Build sparse affinity anchors S_i using block co-occurrence;
 - 3: Compute vertex key $\text{Key}[i]$ and sort vertices in descending order;
 - 4: **for** each vertex i in order **do**
 - 5: Select candidate PEs with minimal projected workload;
 - 6: Choose p^* maximizing locality $\sum_{v \in S_i(p)} \text{Aff}(i, v)$;
 - 7: Assign i to p^* and update workload $L(p^*)$;
 - 8: **end for**
 - 9: **for** each PE p **do**
 - 10: Reorder assigned vertices to maximize block contiguity;
 - 11: **end for**
-

此外,该预处理步骤仅在训练或推理前执行一次,且结果可在各层间复用,因此其开销可在多层网络中均摊.在两层 GCN 模型中,预处理阶段仅占端到端执行时间的约 2.56%~3.62%,且在更深的 DeGNN 中将进一步下降(详见第 5.2 小节整体性能评估).由此可见,所提出的策略在保持轻量代价的同时,显著提升了负载分配与内存访问效率,从而在浅层 GNN 中实现性能提升,并在深层模型中获得更高的加速收益.

4 G3DMA 的硬件设计与实现

4.1 DRAM 层架构设计

整个 DRAM 层的存储空间被划分为多个 vault,以增强内存访问并行性并降低数据传输延迟.如图 6(a)所示,每个 vault 包括一个由两个 bank 组成的 bank 组、一个 vault 控制器、一个内存寻址器(content addressable memory, CAM),以及两个计算引擎(computation engine, CE).bank 用于存储计算所需数据;vault 控制器负责跨 -vault 的数据访问调度;CAM 用于特征稀疏场景下权重参数的快速匹配与重用,以提升数据访问效率;计算引擎则执行组合与聚合操作.

计算引擎.如图 6(b)所示,每个计算引擎由控制器、四个缓冲区(包括 DBSR 缓冲区,BSR 缓冲区,密集缓冲区和输出缓冲区),获取器与分发器(fetcher and dispatcher),解码器与编码器(decoder and encoder),以及激活函数模块(activator)组成.

在组合阶段中,控制器首先利用多路复用器从 BSR 缓冲区选通压缩特征数据,并将其加载至解码器,以支持后续的解码处理(步骤 1).节点特征 $\mathbf{X}$ 被加载至 PE 的寄存器的同时,解码器根据 BSR 中的 bitmap 生成与特征对应的权重索引,并将该索引传递至获取器与分发器模块(步骤 2).随后,获取器与分发器根据输入的权重索引从存储中调取与 $\mathbf{X}$ 对应的权重矩阵 $\mathbf{W}$,并将其送入 PE;PE 通过行积的方式执行计算,得到节点的嵌入表示 $\mathbf{XW}$ (步骤 3).最终,节点嵌入表示 $\mathbf{XW}$ 经由输出缓冲区写回至本地 bank,或通过 TSVs 传输至其他 vault 以支持跨分区计算(步骤 4).

在中间聚合阶段,与组合阶段的主要区别体现在步骤 1 与 3.在步骤 1 中,控制器通过多路复用器从 DBSR 缓冲区加载压缩后的邻接矩阵 $\mathbf{A}$,用于后续的解码与聚合操作.在步骤 3 中,邻接矩阵 $\mathbf{A}$ 与节点嵌入表示 $\mathbf{XW}$ 在 PE 中执行外积运算.

在残差累加阶段,邻接矩阵 $\mathbf{A}$ 与节点嵌入表示 $\mathbf{XW}$ 外积生成的中间矩阵会按行拆解为向量,并分发至对应的计算引擎中.随后,这些向量将与上一层的节点特征一同进行累加,节点状态更新过程中融合残差连接机制,从而有效缓解 DeGNN 中的过平滑问题.累加结果经由激活函数模块进行非线性变换,得到下一层的节点特征向量(步骤 5).最终,这些特征向量将被送入编码器,压缩为 BSR 格式(步骤 6),并通过输出缓冲区写回至本地 bank,或经由 TSVs 传输至其他 vault,支持下一层跨分区的后续计算.

解码器.图 6(c)展示了我们所设计的解码器结构.该解码器可同时支持 DBSR 与 BSR 两种稀疏格式,实现对邻接矩阵与节点特征的统一解码处理.

在对 DBSR 格式数据进行解码时,解码器首先通过块读取器提取 Block Presence Bitmap,以确定目标列中各个有效块的位置;同时,将 In-Block Position Bitmap 加载至 Bitmap 寄存器,为后续块内索引的生成提供位置信息(步骤 1).随后,解码器将 In-Block Position Bitmap 输入至 BitIndex 计数模块,对当前块内的非零位置进行前缀累加编码,以明确每个非零元素在块内的偏移位置.例如,若输入的 Bitmap 为“1011”,则 BitIndex 模块将其转换为“1123”(步骤 2).接着,系统将原始 Bitmap 与该累加结果执行逐位与操作,从而生成对应节点嵌入表示 $\mathbf{XW}$ 的有效索引.例如,“1011”与“1123”进行与操作后得到“1023”(步骤 3).最终,获取器与分发器根据解码生成的块索引(步骤 4)和块内索引(步骤 5),提取相应的节点嵌入表示 $\mathbf{XW}$,并将其分发至对应的 PE 中完成计算.

对于 BSR 格式的解码过程,与 DBSR 类似,唯一区别在于无需执行 Block Presence Bitmap 的读取操作.

处理单元.处理单元的设计如图 6(d)所示,其中集成了 3 个寄存器、2 个解复用器(demultiplexers)、1 个乘

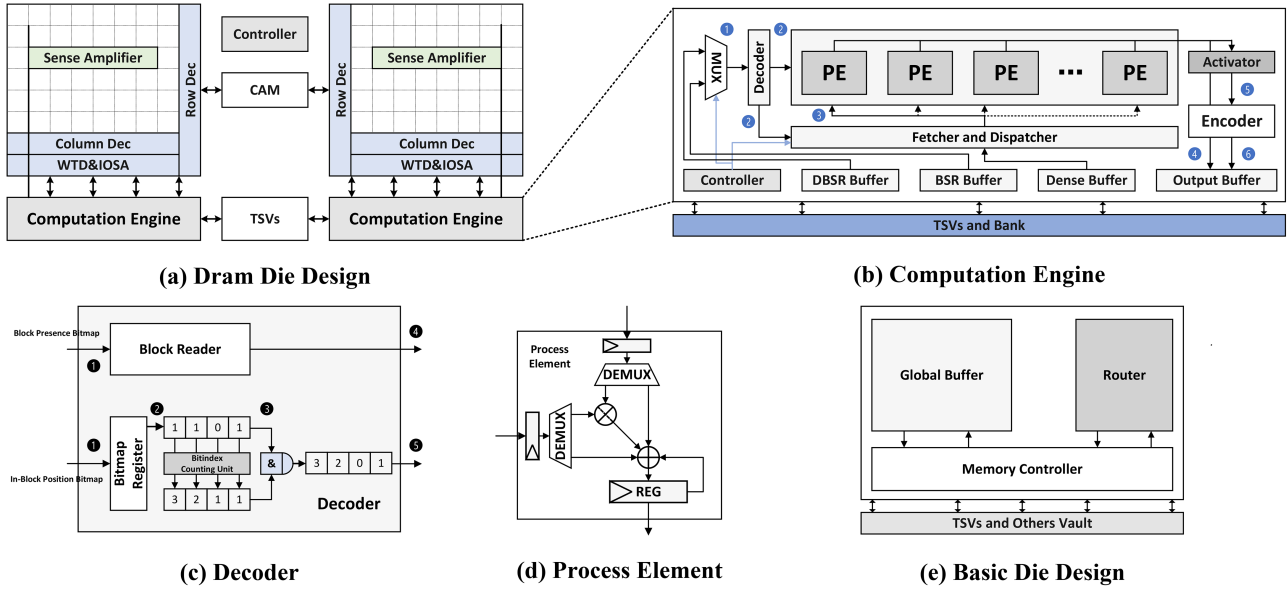


图 6 (网络版彩图) G3DMA 的架构与模块设计示意图. (a) DRAM 层的结构总览; (b) 计算引擎设计; (c) 压缩数据解码器; (d) 处理单元 (PE); (e) Basic 层的结构总览.

Figure 6 (Color online) G3DMA architecture and module design. (a) Overview of DRAM die design; (b) design of the computation engine; (c) decoder for compressed data; (d) processing element (PE); (e) overview of basic die design.

法器与 1 个加法器, 用于支持中间结果的存储、路由调度与乘加 (multiply-accumulate, MAC) 操作. 为适应我们提出的 GNN 计算模型在组合阶段、中间聚合阶段与残差累加阶段中的异构计算需求, 硬件结构中引入了解复用器, 以实现乘法与加法操作的解耦, 从而灵活支持不同计算路径下的运算调度.

4.2 Basic 层架构设计

如图 6(e) 所示, 每个 vault 的逻辑层均由路由器 (router)、内存控制器与 vault 全局缓冲区组成, 结构统一, 既可独立完成本地请求的解析与调度, 也支持与其他 vault 间的数据交互与协同通信, 以支撑组合阶段、中间聚合阶段与残差累加阶段的跨分区计算任务.

在组合阶段, 逻辑层需根据解码器生成的索引访问对应的权重向量. 若权重数据位于本地缓冲区, 则内存控制器直接调取并经由路由器送入计算引擎; 若所需权重不在本地, 系统则会发起远程请求, 由目标 vault 返回结果以完成跨分区加载. 在此过程中, 控制器需维护一张静态的参数映射表, 用于标识权重与 vault 的对应关系, 以确保索引访问的正确性与高效性. 在执行中间聚合阶段时, 若节点嵌入表示数据位于本地缓冲区, 则内存控制器直接调取并经由路由器送入计算引擎; 若不在本地, 系统则会发起远程请求, 由目标 vault 返回结果以完成跨分区加载.

中间聚合阶段的访存逻辑与组合阶段类似, 但操作对象从权重向量切换为节点嵌入表示 $\mathbf{XW}$. 若 $\mathbf{XW}$ 已驻留于本地缓冲区, 内存控制器即可就地调度至计算引擎; 否则, 需协调其他 vault 提供相应数据, 实现节点特征在多个 vault 间的对齐与整合, 以支撑聚合运算.

在残差累加阶段, 此时生成的部分嵌入向量需跨 -vault 汇总. 系统一方面需将本地中间结果传送至其他 vault, 另一方面也要接收来自其他 vault 的中间向量, 并将其路由至 DRAM 层的目标计算单元中. 为此, 控制器维护了特征向量与 bank 的静态映射关系, 同时记录非本地特征的分布信息, 从而保障在聚合与残差累加阶段的通信有序进行.

表 2 各种图数据集.

Table 2 Various graph datasets.

	Cora (CR)	Citeseer (CS)	PubMed (PB)	NELL (NL)	Reddit (RD)
#Vertices	2708	3327	19717	65755	232965
#Edges	10556	9104	88648	266144	114615892
#Features	1433	3703	500	61278	602
#Classes	7	6	3	186	41
#Average degree	3.893	2.736	4.496	4.047	491.988

表 3 G3DMA 中 DRAM 层在每个 bank group 的硬件配置.

Table 3 Hardware configurations of each bank group in G3DMA's DRAM die.

Component	Params	Specs	Area (mm ²)	Power (mW)
On-chip_Buffer	Size	6 kB	0.0143	2.83
Process_Element	Num	16	0.1268	36.52
Others	—	—	0.0186	3.64

5 实验与结果

5.1 实验配置

模型与数据集设置. 为验证所提出框架的通用性与有效性, 本文选取 3 种具有代表性的图神经网络模型进行评估, 包括 GCN^[12], GIN^[13] 和 GraphSAGE^[14], 并在 5 个广泛使用的基准数据集上开展实验, 具体包括 Cora (CR)^[28], Citeseer (CS)^[29], PubMed (PB)^[30], Reddit (RD)^[31] 和 NELL (NL)^[39] (如表 2 所示). 上述数据集涵盖引文网络、知识图谱与社交网络等不同类型, 具有广泛的结构多样性与规模差异性, 为全面评估系统在不同图结构下的推理效率与适应能力提供了基础.

具体而言, Cora, Citeseer 与 PubMed 均为标准引文网络, 节点表示学术论文, 边表示引用关系, 整体图结构较为稀疏, 类别数量适中, 常用于节点分类任务的基础性能验证; NELL 来源于网页抽取构建的知识图谱, 节点和边类型丰富, 结构异质性强, 适合评估模型在异构图环境下的表达能力; Reddit 为大规模社交网络, 节点表示用户, 边代表评论交互, 图密度较高, 适用于考察模型在大图场景下的可扩展性与系统负载能力. 这些数据集在结构属性、规模层级与应用场景方面各具特征, 为深入评估所提框架在 DeGNN 推理中的性能表现与鲁棒性提供了充分支撑.

硬件设置. 与当前主流采用 3D-stacked memory architecture 的加速器设计保持一致, G3DMA 采用由 1 个逻辑层 (basic die) 与 8 个垂直堆叠的存储层 (DRAM dies) 组成的 memory cube 架构. 为提升并行访问效率与内部带宽利用率, 整个存储立方体被划分为 16 个垂直分区 (vault), 每个 vault 包含来自不同 DRAM 层的存储资源, 并通过 TSVs 与底部 basic die 建立高速连接通道, 从而实现跨层级的数据传输.

在存储组织方面, 每个 DRAM die 被进一步划分为多个 banks, 每层包含两个 banks, 每个 bank 的容量为 16 MB. 因此, 整个 memory cube 共包含 8 (DRAM dies) $\times$ 2 (banks per die) $\times$ 16 (vaults) = 512 个 banks, 总存储容量达到 8 GB. 这种高度并行的存储结构在提供大带宽访问能力的同时, 也支持灵活的数据调度与高效的随机访问, 非常适合图神经网络中大规模邻接特征拉取和稀疏索引访问等计算场景.

在计算单元部署上, 我们在每个 bank 旁均配置一个计算引擎, 每个引擎内集成 16 个 PE, 且每个 PE 实现一个支持 FP32 精度的乘加单元 (MAC). 此外, 每个计算引擎配备的 DBSR_Buffer 与 BSR_Buffer 容量均为 1 kB, 而 Dense_Buffer 与 Output_Buffer 的容量均为 2 kB, 以兼顾不同稀疏度与数据密度下的高效特征存取需求. 在 basic die 中, Global_Buffer 的容量为 128 kB. 存储层 (DRAM die) 与逻辑层 (basic die) 的具体硬件配置详见表 3 和 4.

表 4 G3DMA 中 Basic 层的硬件配置.
Table 4 Hardware configurations in G3DMA's basic die.

Component	Params	Specs	Area (mm ²)	Power (mW)
On-chip_Buffer	Size	128 kB	0.3682	48.86
Others	—	—	0.0226	3.82

基线设置. 为全面评估 G3DMA 的性能与能效, 本文选取了通用图神经网络框架及多款高性能专用加速器作为对比对象, 以确保评测结果具有代表性与多维度覆盖. 首先, 在通用框架方面, 选用了基于 PyTorch 的图神经网络库 PyTorch Geometric (PyG) [40], 该框架广泛应用于图神经网络的训练与推理任务. 实验在运行 Linux 系统的工作站上进行, 平台配置包括双路 Intel Xeon E5-2680 v4 处理器与 NVIDIA Tesla V100 GPU, 并分别测试 PyG 在 CPU (PyG CPU) 与 GPU (PyG GPU) 上的实现性能, 作为与 G3DMA 对比的通用基准.

其次, 在专用硬件方面, 本文选取了当前具有代表性的图神经网络加速器 HyGCN [41], GCIM [26], GCNim [27] 与 SGCN [25] 作为对比对象. 这些加速器在体系结构、数据流优化及存储管理等方面采用了多种先进技术, 覆盖了不同设计思路与优化策略. 通过与上述框架及加速器的性能与能效对比, 可以全面验证 G3DMA 相较于现有解决方案的优势与适用性.

环境设置. 为了准确评估 G3DMA 的性能与功耗, 本文采用 Verilog HDL 进行设计, 并利用 Synopsys Design Compiler 在 TSMC 28 nm 标准单元库下完成综合, 工作频率设定为 1 GHz. 存储与互连部分则基于 CACTI-3DD [42] 建模, 涵盖 3D 堆叠 DRAM、片上存储单元、互连通路以及逻辑层, 从而获得精确的面积、功耗与时延估计. 此外, 本文进行了周期精确的体系结构级仿真, 在统一的 DRAM 带宽约束下, 对 G3DMA 与多种对比加速器的微体系结构行为进行了分析, 以获得执行周期和性能能效的全面评估.

为确保不同加速器间对比的公平性, 本文对关键硬件与计算资源进行了标准化处理. 首先, 所有加速器均配置相同的 HBM 存储、片上缓冲区大小与缓存机制. 其次, 对比基线的计算资源规模与处理单元面积与 G3DMA 保持一致, 包括 3D 堆叠架构 (如 GCIM [26] 和 GCNim [27]) 以及异构加速器 (如 HyGCN [41] 和 SGCN [25]). 最后, 所有算术运算均归一化为 BitOPs, 以保证不同加速器间的性能评估具有可比性. 通过上述标准化设置, 可以准确反映 G3DMA 在混合精度量化下所带来的性能提升与能效优势.

5.2 整体性能评估

延迟与加速比. 图 7(a) 展示了 G3DMA 与 PyG 在 CPU 与 GPU 平台上的执行时间对比. 实验结果表明, 在 GNN 任务中, G3DMA 相较于 PyG-CPU 与 PyG-GPU 分别实现了平均 $6007.41\times$ 与 $106.28\times$ 的加速, 主要得益于其近内存计算架构, 而图数据的稀疏性限制了 GPU 在推理过程中的并行效率. 图 7(b) 给出了 G3DMA 与多款基于 3D 堆叠存储架构的专用加速器的比较结果. 相较于 HyGCN [41], GCIM [26], GCNim [27] 与 SGCN [25], G3DMA 分别实现了 $26.27\times$, $11.93\times$, $1.46\times$ 与 $2.42\times$ 的加速. 这一优势主要来源于所提出的压缩格式与三阶段执行模型: 邻接矩阵采用 DBSR 表示, 节点特征采用 BSR 表示, 并结合零值跳过与稀疏感知调度机制, 从而提升了访存局部性与带宽利用率, 并减少冗余计算与数据搬移.

能耗. 图 7(c) 和 (d) 给出了不同系统与加速器在各数据集上的归一化能耗对比. 在统一的 DRAM 带宽与片上存储配置下, 实验结果显示 G3DMA 的能耗显著低于对比方案, 表明其在能效方面具有明显优势. 这一改善主要得益于以下机制: (i) 邻接矩阵采用 DBSR 表示、节点特征采用 BSR 表示, 提升压缩率与索引/解码效率, 从而降低 DRAM 访问频次与数据搬移开销; (ii) 三阶段执行模型避免中间结果的全量物化, 通过按行切分的中间向量在 vault 间按需传输, 减少片上/片外写回; (iii) 稀疏感知调度与零值跳过机制在组合与聚合阶段跳过无效访存与乘加, 提升带宽利用率并降低动态能耗; (iv) 基于 bitmap-value 分离的 BSR 设计减少稀疏度变化场景下的 padding 开销, 使非零值能够以连续突发方式读写, 进一步降低访问能耗.

时间分解. 图 7(e) 展示了 G3DMA 的执行时间构成, 结果显示预处理时间仅占 2 层 GCN 模型总执行时间的 $2.56\%\sim 3.62\%$, 并随网络层数增加进一步下降, 表明预处理开销可忽略. 聚合与组合阶段构成主要运行时间,

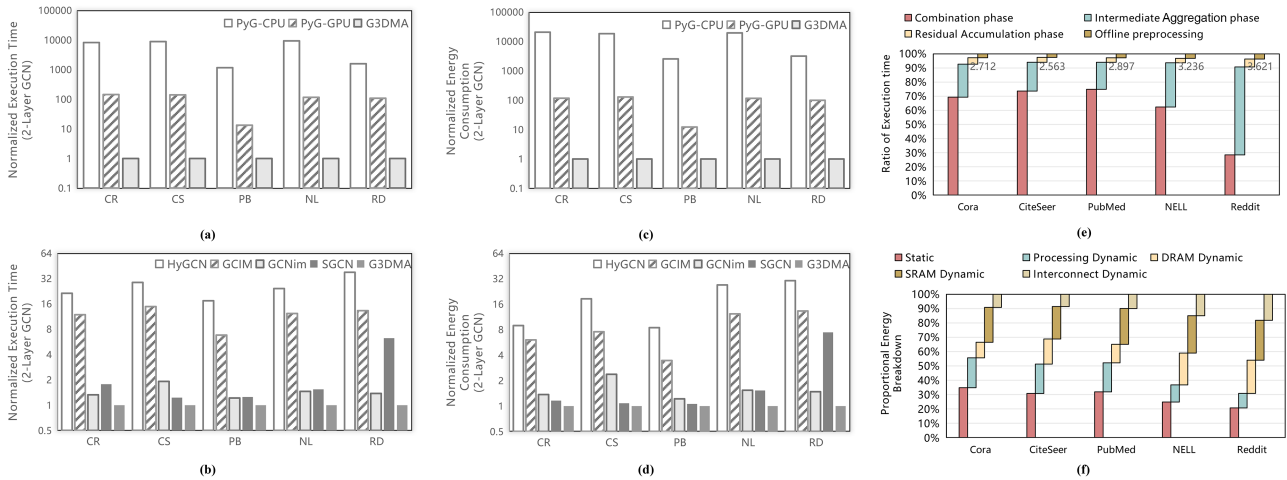


图 7 (网络版彩图) (a) G3DMA 与 CPU, GPU 的执行时间对比; (b) G3DMA 与其他加速器的执行时间对比; (c) G3DMA 与 CPU, GPU 的能耗对比; (d) G3DMA 与其他加速器的能耗对比; (e) G3DMA 的执行时间分解; (f) G3DMA 的能耗分解。

Figure 7 (Color online) (a) Execution time comparison of G3DMA with CPU and GPU; (b) execution time comparison of G3DMA with other accelerators; (c) energy consumption comparison of G3DMA with CPU and GPU; (d) energy consumption comparison of G3DMA with other accelerators; (e) execution time breakdown of G3DMA; (f) energy breakdown of G3DMA.

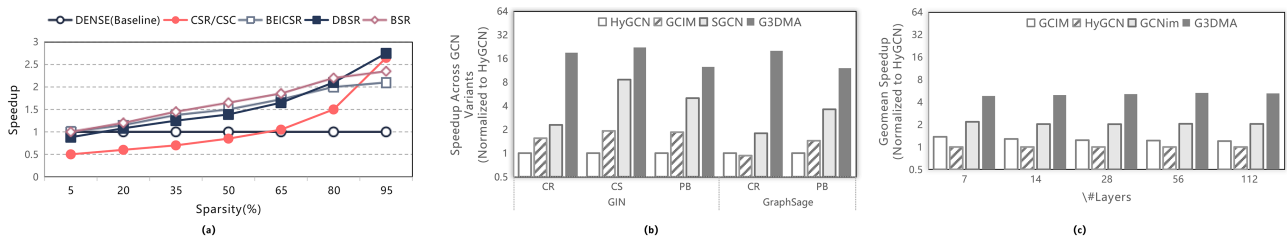


图 8 (网络版彩图) (a) 各类压缩格式在不同稀疏度下的压缩效果对比; (b) 不同加速器在 GIN 与 GraphSAGE 模型上的加速性能对比; (c) 模型层数对性能的敏感性分析。

Figure 8 (Color online) (a) Compression of different formats under varying sparsity; (b) speedup of accelerators on GIN and GraphSAGE; (c) sensitivity to the number of layers.

其中聚合阶段在大规模图上占比更高, 原因在于其以邻居特征加载与累加为主, 受限于访存带宽; 而组合阶段执行稠密运算, 数据局部性与复用率更高, 时间占比因此相对较低。

能耗分解. 图 7(f) 进一步给出了能耗构成。分析表明, 小规模图中静态功耗和 SRAM 访问占比较高; 随着图规模增大, 互连开销逐渐增加, 导致处理单元部分在总能耗中的比例下降。

面积. 表 3 与 4 给出了 DRAM 层与 Basic 层的面积分布情况。实验结果表明, 每个 DRAM die 的处理组件面积仅为 0.1597 mm^2 , 占 bank group 面积的 16%, 开销极小。所有逻辑层组件的总面积为 15.1704 mm^2 , 占基础层面积的 12.04%。由此可见, G3DMA 在聚合逻辑与基础层扩展方面实现了显著的片上面积优化, 并将额外开销有效控制在 10%~30% 的设计预算范围内, 从而保证了整体架构的可扩展性。

5.3 敏感性研究

压缩格式对性能的影响. 中间特征的稀疏性主要源于激活函数 ReLU 的作用, 其输出分布通常导致特征稀疏度维持在 40%~70%。从图 8(a) 可以观察到, 在这一稀疏度区间内, BSR 格式展现出最佳的压缩效果, 有效平衡了存储开销与索引开销。相比之下, 图的邻接矩阵通常呈现更高的稀疏性, 稀疏度普遍低于 5%。在该场景下, DBSR 格式相较于传统的 CSR 格式表现出更优的压缩效率, 更加适配结构稀疏性突出的邻接矩阵存储需求。

GNN 变体对性能的影响. 图 8(b) 对比了 G3DMA 在两种 GCN 变体 GIN 与 GraphSAGE 上的性能表现, 二

者采用了不同的聚合策略. GIN 去除了边权, 在降低邻接矩阵存储开销的同时, 增加了对特征计算的依赖. 由于 G3DMA 能够有效减少特征矩阵的访存开销, 因此在 GIN 上表现出更显著的加速优势. 相比之下, GraphSAGE 通过边采样降低有效边数, 从而减少计算开销与数据搬运. 然而, 该方法导致计算资源利用率不足, 整体加速效果相对有限.

层数对性能的影响. 图 8(c) 展示了 HyGCN, GCIM, GCNim 与 G3DMA 在 28~112 层 GCN 上, 使用 CR, CS 与 PB 数据集时的几何平均加速比. 实验结果表明, G3DMA 在所有深度下均保持最高的加速性能, 且随着模型深度的增加, 其优势愈加显著. 值得注意的是, 即便与专门针对 DeGNN 优化的 SGCN 相比, G3DMA 仍表现出更优的加速效果.

6 相关工作

6.1 图神经网络加速器

近年来, 随着图神经网络在社交网络分析、推荐系统、生物信息学等领域的广泛应用, 其在计算资源、存储带宽和执行效率方面的挑战日益突出. 为应对这些挑战, 研究者们纷纷提出了多种专用硬件加速器 [25~27, 41, 43~48], 以实现 GNN 推理过程的高效加速.

HyGCN [41] 是较早提出的一种异构图神经网络加速器, 其设计理念在于将 GNN 推理流程中的聚合 (aggregation) 与组合 (combination) 两个阶段分离并独立优化. 具体而言, HyGCN 在硬件层面分别为这两个阶段构建专用计算单元, 从而提升每个阶段的执行效率. 然而, 由于聚合阶段与组合阶段在计算密度与访存模式上存在显著差异, 该策略在执行过程中容易导致工作负载不均衡, 进而影响整体系统的吞吐率与资源利用率. 为解决上述问题, SGCN [25] 提出了一种面向深度 GNN 的异构加速框架, 通过在硬件中融合特征压缩 (如稀疏编码) 与稀疏矩阵乘操作, 有效减少了无效计算与冗余访存. 在该架构中, 稀疏性不仅被用于压缩特征与结构数据, 还被用于引导计算与数据流的优化, 从而显著提升执行效率与能效比, 尤其适用于层数较深的图神经网络. MEGA [48] 则引入了一种基于节点度数的混合精度量化机制, 通过动态调整不同节点的特征精度, 实现了对低度节点采用低比特宽表示、而对高度节点保持高精度表示的策略. 这种“度感知”的量化方法既充分利用了图结构的统计特性, 又有效降低了存储需求和计算负载, 在保持模型精度的同时显著提升了推理速度与硬件资源利用效率.

此外, GCIM [26], GCNim [27] 与 Lift [43] 等一系列加速器进一步探索了基于 3D 堆叠存储的 GNN 计算方案. 该类架构通过将计算单元集成于高带宽堆叠存储之中, 打破了传统冯·诺依曼 (Neumann) 架构下存储与计算分离所导致的瓶颈, 能够实现高并发、低延迟的数据访问与处理. 其中, GCIM 提出了感知图结构的映射机制以优化访存局部性, GCNim 针对聚合路径进行剪枝以减少数据移动开销, 而 Lift 则结合了负载均衡与映射策略, 通过在逻辑层与存储层之间协同优化, 进一步释放了 3D-CIM 架构的潜力.

6.2 近数据处理架构

近年来, 随着图神经网络、深度学习和大规模图处理任务对数据访问效率提出更高要求, 近数据处理 (near-data processing, NDP) 架构因其在性能与能效方面的优势, 已成为高性能计算系统设计的重要方向. 其提出背景源于长期存在的“内存墙”问题, 即处理器计算能力增长速度远快于内存带宽的提升, 导致数据传输延迟逐渐成为系统瓶颈. 为缓解该矛盾, 数据中心计算 (data-centric computing, DCC) 范式被提出, 其核心思想是在靠近存储的位置部署计算逻辑, 以减少数据搬运开销. DCC 范式主要由内存中计算 (processing-in-memory, PIM) 与近内存处理 (near-memory processing, NMP) 两类架构支撑, 已广泛应用于图计算 [44, 49, 50]、深度学习等高带宽场景.

现有 NDP 架构主要可按计算逻辑的集成位置划分为三类. 第一类为逻辑层 NDP (logic-die NDP) [26, 27, 43], 该类架构依托 3D 堆叠 DRAM (如 HMC 与 HBM), 将计算单元集成于底部逻辑层, 并通过硅通孔 (TSV) 与上

层存储单元实现高带宽、低延迟的数据交互。该方式能够在保持高并发访问能力的同时,显著减少传统处理器与内存间的数据传输负担。已有多项图神经网络加速器采用此类结构,以提升邻接关系处理与特征聚合的效率。

第二类为 DIMM-buffer NDP^[51],适用于商用 DDR DRAM,计算逻辑集成于 DIMM 模组的缓冲芯片中。该类架构能够在数据进入主机内存通道前,完成初步的数据聚合与预处理,从而缓解总线带宽压力,并提升整体系统的数据传输效率。

第三类为 DRAM-bank 级 NDP^[52],采用更细粒度的集成方式,将处理逻辑部署于 DRAM 的 Bank 级甚至子阵列内部,部分设计进一步将计算单元靠近或嵌入感应放大器(sense amplifier)。该架构实现了更紧耦合的存算协同,并具备良好的可扩展性与接口兼容性。近年来,一些激进方案提出基于 In-DRAM Computation 的逻辑运算机制,可在 DRAM 内部直接执行位运算,进一步降低冗余数据搬移带来的性能和能效损耗。

7 结论

本文提出了 G3DMA,一种基于 3D-CIM 架构的高性能 GNN 加速器,旨在高效支持深度图神经网络的推理计算。本文的创新主要体现在两个方面:一是提出了差异化的稀疏表示方法,针对邻接矩阵与节点特征采用高效压缩格式,显著提升了存储与访存效率,有效缓解了 DRAM 带宽压力与数据搬移开销;二是设计了新型的三阶段执行模型,结合稀疏感知调度与零值跳过机制,在避免冗余计算和写回的同时,提升了访存局部性与整体能效。实验结果表明,与现有先进系统和专用加速器相比,G3DMA 在性能与能效方面均取得了显著提升。这些结果验证了 G3DMA 在 DeGNN 推理中的可扩展性与高效性。

参考文献

- Aggarwal C C. Mining text and social streams. SIGKDD Explor Newsl, 2014, 15: 9–19
- Guillemot C, Le Meur O. Image inpainting: overview and recent advances. IEEE Signal Process Mag, 2013, 31: 127–144
- Zhao Z Q, Zheng P, Xu S T, et al. Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst, 2019, 30: 3212–3232
- Alzubaidi L, Zhang J, Humaidi A J, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data, 2021, 8: 1–74
- Xiao S, Wang S, Dai Y, et al. Graph neural networks in node classification: survey and evaluation. Machine Vision Appl, 2022, 33: 4
- Sun H, Li X, Wu Z, et al. Breaking the entanglement of homophily and heterophily in semi-supervised node classification. In: Proceedings of IEEE Int Conf Data Eng (ICDE), 2024. 2379–2392
- Qin M, Yeung D Y. Temporal link prediction: a unified framework, taxonomy, and review. ACM Comput Surv, 2023, 56: 1–40
- Liao H, Zhang X, Long Z, et al. Diffusion-based location-aware recommender systems. J Stat Mech, 2020, 2020: 043401
- Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey. ACM Comput Surv, 2022, 55: 1–37
- Wu L, He X, Wang X, et al. A survey on accuracy-oriented neural recommendation: from collaborative filtering to information-rich recommendation. IEEE Trans Knowl Data Eng, 2022, 35: 4425–4445
- Bipin B, Nair J J. Image convolution optimization using sparse matrix vector multiplication technique. In: Proceedings of Int Conf Advances in Computing, Communications and Informatics (ICACCI), 2016. 1453–1457
- Huang Y, Zheng L, Liu H, et al. MEG2: In-memory acceleration for genome graphs analysis. In: Proceedings of the 60th ACM/IEEE Design Autom Conf (DAC), 2023. 1–6
- Chiang W L, Liu X, Si S, et al. Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of ACM SIGKDD Int Conf Knowl Discov Data Min, 2019. 257–266
- Liu J, Ong G P, Chen X. GraphSAGE-based traffic speed forecasting for segment network with sparse data. IEEE Trans Intell Transp Syst, 2020, 23: 1755–1766
- Bouritsas G, Frasca F, Zafeiriou S, et al. Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Trans Pattern Anal Mach Intell, 2022, 45: 657–668
- Chen D, Lin Y, Li W, et al. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of AAAI Conf Artif Intell, 2020. 3438–3445
- Li G, Muller M, Ghanem B, et al. Training graph neural networks with 1000 layers. In: Proceedings of Int Conf Mach Learn (ICML), 2021. 6437–6449

- 18 Li G, Muller M, Thabet A, et al. DeepGCNs: can GCNs go as deep as CNNs? In: Proceedings of IEEE/CVF Int Conf Comput Vis (ICCV), 2019. 9267–9276
- 19 Li G, Xiong C, Thabet A, et al. DeeperGCN: all you need to train deeper GCNs. 2020. ArXiv:2006.07739
- 20 Hsieh K, Khan S, Vijaykumar N, et al. Accelerating pointer chasing in 3D-stacked memory: challenges, mechanisms, evaluation. In: Proceedings of IEEE Int Conf Comput Design (ICCD), 2016. 25–32
- 21 Monmasson E, Cirstea M N. FPGA design methodology for industrial control systems—a review. IEEE Trans Ind Electron, 2007, 54: 1824–1842
- 22 Chinnery D, Keutzer K W. Closing the Gap Between ASIC& Custom: Tools and Techniques for High-performance ASIC Design. Berlin: Springer, 2002
- 23 Liu T, Wang Y, Ying R, et al. MUSE-GNN: Learning unified gene representation from multimodal biological graph data. In: Proceedings of Adv Neural Inf Process Syst, 2024. 36
- 24 Burkhart J G, Wu G, Song X, et al. Biology-inspired graph neural network encodes Reactome and reveals biochemical reactions of disease. Patterns, 2023, 4: 100758
- 25 Yoo M, Song J, Lee J, et al. SGCN: Exploiting compressed-sparse features in deep graph convolutional network accelerators. In: Proceedings of IEEE Int Symp High-Perform Comput Archit (HPCA), 2023. 1–14
- 26 Chen J, Lin Y, Sun K, et al. GCIM: toward efficient processing of graph convolutional networks in 3D-stacked memory. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2022, 41: 3579–3590
- 27 Wang R, Hu A, Zheng L, et al. An efficient GCNs accelerator using 3D-stacked processing-in-memory architectures. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2024, 43: 1360–1373
- 28 McCallum A K, Nigam K, Rennie J, et al. Automating the construction of internet portals with machine learning. Inf Retrieval, 2000, 3: 127–163
- 29 Sen P, Namata G, Bilgic M, et al. Collective classification in network data. AI Mag, 2008, 29: 93–106
- 30 Namata G, London B, Getoor L, et al. Query-driven active surveying for collective classification. In: Proceedings of the 10th International Workshop on Mining and Learning with Graphs (MLG), 2012. 1–8
- 31 Hamilton W L, Ying R, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2017. 1024–1034
- 32 Rozemberczki B, Allen C, Sarkar R, et al. Multi-scale attributed node embedding. J Complex Netws, 2021, 9: cnab014
- 33 Tang J, Zhang J, Yao L, et al. Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2008. 990–998
- 34 Saad Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia: SIAM, 2003. 42–47
- 35 Barrett R, Berry M, Chan T F, et al. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. Philadelphia: SIAM, 1994. 17–20
- 36 Dongarra J J, Duff I S, Sorensen D C, et al. Solving Linear Systems on Vector and Shared Memory Computers. Philadelphia: SIAM, 1991. 55–60
- 37 Xu K, Li J, Zhang M, et al. Representation learning on graphs with jumping knowledge networks. In: Proceedings of Int Conf Mach Learn (ICML), 2018. 5453–5462
- 38 Jaccard P. étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin de la Société Vaudoise des Sciences Naturelles, 1901, 37: 547–579
- 39 Carlson A, Betteridge J, Kisiel B, et al. Toward an architecture for never-ending language learning. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), 2010. 1306–1313
- 40 Fey M, Lenssen J E. Fast graph representation learning with PyTorch Geometric. 2019. ArXiv:1903.02428
- 41 Yan M, Deng L, Hu X, et al. HyGCN: a GCN accelerator with hybrid architecture. In: Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020. 15–29
- 42 Chen K, Li S, Muralimanohar N, et al. CACTI-3DD: architecture-level modeling for 3D die-stacked DRAM main memory. In: Proceedings of Design, Autom Test Europe Conf Exhib (DATE), 2012. 33–38
- 43 Chen J, Zhong Z, Sun K, et al. LIFT: exploiting hybrid stacked memory for energy-efficient processing of graph convolutional networks. In: Proceedings of the 60th ACM/IEEE Design Autom Conf (DAC), 2023. 1–6
- 44 Yu H, Zhang Y, Li X T, et al. An efficient high throughput system of hypergraph neural network under CPU-GPU heterogeneous environments. Sci Sin Inform, 2025, 55: 841–859 [余辉, 张宇, 李鑫滔, 等. 面向 CPU-GPU 异构环境下的高性能超图神经网络加速系统. 中国科学: 信息科学, 2025, 55: 841–859]
- 45 Zhang Y, Jiang X Y, Yu H, et al. Review of key technologies in graph processing architectures and systems software. J Comput Res Dev, 2024, 61: 20–42
- 46 Zhao J, Jiang X Y, Zhang Y, et al. An efficient storage system for highly concurrent graph analysis tasks. Sci Sin Inform, 2022, 52: 111–128 [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. 中国科学: 信息科学, 2022, 52: 111–128]

- 47 Liao X F, Chen Y C, Zhang Y, et al. An efficient incremental strongly connected components algorithm for dynamic directed graphs. *Sci Sin Inform*, 2019, 49: 988–1004 [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. *中国科学: 信息科学*, 2019, 49: 988–1004]
- 48 Zhu Z, Li F, Li G, et al. MEGA: a memory-efficient GNN accelerator exploiting degree-aware mixed-precision quantization. In: *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024. 124–138
- 49 Liu D, Ren Z L, Long Z T, et al. Mining design pattern use scenarios and related design pattern pairs: a case study on online posts. *J Comput Sci Technol*, 2020, 35: 963–978
- 50 Fu Y, Long Z, Zhang Y, et al. PairGraph: an efficient search-space-aware accelerator for high-performance concurrent pairwise queries. In: *Proceedings of ACM/IEEE Design Autom Conf (DAC)*, 2025. 1–7
- 51 Zhou Z, Li C, Yang F, et al. DIMM-Link: enabling efficient inter-DIMM communication for near-memory processing. In: *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023. 302–316
- 52 Tian B, Li Y, Jiang L, et al. NDPBridge: enabling cross-bank coordination in near-DRAM-bank processing architectures. In: *Proceedings of ACM/IEEE 51st Annu Int Symp Comput Archit (ISCA)*, 2024. 628–643

A high-efficiency 3D-stacked accelerator for deep graph neural network inference

Zhongtian LONG^{1,2,3,4}, Yu ZHANG^{1,2,3,4*}, Yutao FU^{1,2,3,4}, Qiyuan NIU^{1,2,3,4}, Shunsen LV^{1,2,3,4}, Jiaqi ZHAI^{1,2,3,4}, Wenju ZHAO^{1,2,3,4}, Yingqi ZHAO^{1,2,3,4}, Jin ZHAO^{1,2,3,4} & Hai JIN^{1,2,3,4}

1. *National Engineering Research Center for Big Data Technology and System, Huazhong University of Science and Technology, Wuhan 430074, China*

2. *Service Computing Technology and System Laboratory, Huazhong University of Science and Technology, Wuhan 430074, China*

3. *Cluster and Grid Computing Laboratory, Huazhong University of Science and Technology, Wuhan 430074, China*

4. *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

* Corresponding author. E-mail: zhyu@hust.edu.cn

Abstract Three-dimensional compute-in-memory (3D-CIM) architectures, with their high bandwidth and strong parallelism, provide significant opportunities for accelerating graph neural network (GNN) inference. However, existing 3D-CIM accelerators still face two major challenges when handling deep graph neural networks (DeGNNs): (1) insufficient support for layer-wise sparsity, where zero values lead to redundant memory accesses and ineffective computations, resulting in reduced bandwidth utilization and increased latency; and (2) lack of native support for cross-layer residual dependencies, where frequent data movement incurs additional storage and communication overhead, further exacerbating inference latency. To address these issues, we propose G3DMA, a high-efficiency 3D-CIM accelerator designed through hardware-software co-optimization for DeGNN inference. For sparse encoding, G3DMA employs differentiated compression: adjacency matrices are stored using Dual-Bitmap Sparse Representation (DBSR), while node features adopt a bitmap-value separated Block Sparse Representation (BSR), significantly reducing DRAM access overhead while improving compression ratio and indexing efficiency. At the execution level, we design a three-stage model—“combination-intermediate aggregation-residual accumulation”—augmented with sparsity-aware scheduling and zero-skipping, thereby avoiding full materialization of intermediate results and reducing ineffective computations. At the hardware level, G3DMA integrates lightweight compute arrays and dedicated codec units in the near-memory logic layer, efficiently supporting DBSR/BSR processing and block-wise memory accesses; it further implements a three-stage dataflow with pipelined control, residual-friendly accumulation paths, and low-overhead cross-vault routing. Experimental results demonstrate that G3DMA achieves speedups of $6007.41\times$ and $106.28\times$ over advanced CPU and GPU platforms, respectively. Compared with the latest state-of-the-art (SOTA) accelerators—HyGCN, GCIM, GCNim, and SGCN—G3DMA delivers $26.27\times$, $11.93\times$, $1.46\times$, and $2.42\times$ performance improvements, respectively, and consistently outperforms SOTA designs in both performance and energy efficiency.

Keywords deep graph neural network, hardware accelerator, inference acceleration, sparse compression, 3D compute-in-memory architecture