基于 Tensor Core 的高效深度神经网络加速技术研究现状

吕诗杰[1)]

[1)](华中科技大学 计算机科学与技术学院, 武汉市 430000)

摘 要 在深度学习模型飞速发展的同时，模型越来越复杂化，消耗的计算和存储资源越来越大，如何高效的训练和推理模型已经成为当前挑战。本文研究了基于 NVIDIA GPU Tensor Core 硬件的高效深度神经网络加速技术。通过设计高效内核算法 SpMM 和 SDDMM，相较于 CUDA 官方 API cuDNN 和 cuSPARSE，取得了更高的计算效率。同时，详细介绍了图神经网络的研究背景，并提出了 TC-GNN 框架用于加速图神经网络。该框架利用稀疏图变换和计算任务分解方法，显著提升了训练和推理速度，比当前的 PyG 和 Deep Graph Library 框架更快。在此基础上，对利用异构硬件加速大模型等相关研究方向进行了分析，最后对整篇文章进行了总结。

关键词 张量核心；图形处理器；稀疏矩阵；图神经网络；剪枝；异构计算；

Research Status of Efficient Deep Neural Network Acceleration Techniques Based on Tensor-Core.

Shijie Lv[1)]

[1)](Huazhong University of Science and Technology, Wuhan 430000)

Abstract With the rapid development of deep learning models, the complexity of models keeps increasing, leading to a growing demand for computational and storage resources. As a result, efficient model training and inference have become crucial challenges in the present scenario.This article studies efficient deep neural network acceleration technology based on NVIDIA GPU Tensor Core hardware. By designing efficient kernel algorithms SpMM and SDDMM, higher computing efficiency is achieved compared to CUDA official API cuDNN and cuSPARSE. At the same time, the research background of graph neural networks is introduced in detail, and the TC-GNN framework is proposed to accelerate graph neural networks. The framework utilizes sparse graph transformation and computational task decomposition methods to significantly improve training and inference speeds, making it faster than the current PyG and Deep Graph Library frameworks. On this basis, related research directions such as using heterogeneous hardware to accelerate large models are analyzed. Finally, the entire article is summarized.

Key words Tensor Core; Graphics Processing Unit; Sparse Matrix; Graph Neural Network; pruning; heterogeneous computing;

1 引言

随着深度学习和相关技术的发展，深度神经网络（DNN）的成功往往伴随着高昂的计算成本和内存需求。为了解决这一问题，减少精度和利用稀疏性成为了一种常见的解决方案。然而，现有的稀疏 GPU 内核在实际中并不能有效加速半精度计算，无法满足要求。

近年来，基于图的学习在电子商务、金融服务等广泛领域的任务计算中变得越来越普遍，图神经网络（GNN）在其中占据主导地位。相对于传统的

最终修改稿日期：2023年12月27日.吕诗杰，男，2000年生，硕士，.E-mail: 344501734@qq.com.

图分析方法, GNN 以更高的准确度和更好的通用性脱颖而出。然而，由于高度神经网络的复杂性，现有的 GNN 性能往往无法令人满意，特别是在处理稀疏和不规则图时。

随着先进深度学习模型的不断发展，计算、数据和模型规模的扩大推动着其最新进展，并且这种扩展趋势预计将持续下去。然而，这些大规模深度学习模型需要大量的训练资源，并且使用有限的计算和内存资源进行端到端推理具有挑战性。因此，优化内存占用和推理延迟成为优化深度神经网络模型等关键技术的重要方向。

2 研究背景

深度神经网络在图像分类[1]、机器翻译和文本转语音等领域取得了超越人工的性能。然而，实现和部署大型神经网络模型所需的资源常常令人望而却步，尽管模型质量已被证明可以根据模型和数据集的大小进行缩放[2]。目前的最先进模型，如用于图像分类和机器翻译的模型，通常具有数千万个参数，并需要数十亿次浮点运算才能对单个输入样本进行预测。为了解决这些问题，稀疏性已经成为一种主要的方法挑战。稀疏性指的是模型参数的子集属性恰好为零。当权重为零时，可以跳过任何涉及这些权重的矩阵乘法运算，从而使模型能够使用紧凑的稀疏矩阵进行存储和传输。深度神经网络可以容忍高度稀疏性，并且利用这个特点可以显著降低部署相关的成本。

GPU（Graphics Processing Unit）是一种专门用于处理图形和并行计算的硬件设备。相比传统的 CPU，GPU 拥有更多的并行处理单元，能同时执行大量计算任务。它被广泛应用于图形渲染、科学计算、深度学习和游戏开发等领域，具有高效的并行计算能力。开发者可以使用 CUDA、OpenCL 等编程模型和库来编写和执行在 GPU 上的计算任务。总之, GPU 在加速计算和处理并行任务方面具有重要作用。

图神经网络（GNN）是处理图结构数据的深度学习模型。近年来，研究者们提出了多种 GNN 架构，如 GCN、GraphSAGE 和 GAT 等。这些模型在节点分类、图分类和链接预测等任务上表现出色。为了改进性能，研究者们提出了图注意力机制和改进的图卷积操作。针对大规模图数据，他们还提出了高效的模型和算法，如采样方法和图压缩算法。GNN 在社交网络分析、化学和生物信息学以及推荐系统等领域得到广泛应用。

2.1 深度学习中的稀疏化

许多早期的研究工作致力于通过稀疏化来提高模型的泛化能力。这些工作主要关注参数数量在数十到数百个的模型，并描述了稀疏模型在可解释性方面的优势。然而，当今的模型参数数量已经达到数百万甚至数十亿级别，因此稀疏性对于提高可解释性和可解释性的影响尚待观察。

将剪枝视为类似于 dropout 或数据增强中的噪声的方法[3]被提出来解释模型的泛化能力。有许多研究工作致力于提高模型的计算效率，同时保持模型的准确性。现代网络的计算成本非常昂贵，Inception-V3 模型，需要进行 57 亿次算术运算和 2700 万个参数的计算；而 GPT-3 自然语言处理网络，需要进行 1750 亿个参数的计算。此外，训练这些深度神经网络模型的成本越来越高，最大的语言模型已经需要超级计算机进行训练，每次训练可能花费数百万美元。因此，在训练过程中研究稀疏性以控制训练成本变得非常重要。稀疏性在提高泛化能力、可解释性和抵抗对抗性攻击方面具有潜力。同时，研究计算效率的方法可以降低模型评估和训练的成本，使得大规模深度神经网络的应用更加可行。

2.2 图形处理器

图形处理单元(GPU)提供比 CPU 高得多的指令吞吐量和内存带宽。许多应用程序利用这些更高的功能在 GPU 上运行得比在 CPU 上运行得更快。GPU 专门用于高度并行计算，因此经过设计，更多晶体管专用于数据处理，而不是数据缓存和流量控制。将更多晶体管用于数据处理，例如浮点计算，有利于高度并行计算。GPU 可以通过计算来隐藏内存访问延迟，而不是依赖大数据缓存和复杂的流程控制来避免较长的内存访问延迟。

2.2.1 GPU 内存层次结构

GPU 的内存层次结构包括全局内存、共享内存和寄存器。全局内存是 GPU 中最大的内存池，用于存储大量的数据。共享内存是一种高速缓存，位于 GPU 的多个流处理器之间共享，用于加速数据的共享和通信。寄存器是每个流处理器私有的存储器，用于存储局部变量和计算中间结果。NVIDIA GPU 由一系列流式多处理器(SMs)。在 Volta 中，所有 SM 共享 6MiBL2 缓存和 16GiBDRAM。每个

SM 都有一组子核心，其中每个子核都有自己的寄存器文件、调度程序和执行单位。同一 SM 内的所有子核心共享私有 128KiBL1 缓存，其中一部分可以配置为共享内存。为了最大限度地提高内存带宽利用率，向量内存使用 LDG.128 的操作可用于增加每扇区数向 L1 缓存请求。128 表示每个线程读取 128 位来自全局内存(例如半精度下的 8 个操作数)。此外，内存访问模式预计为 128B 合并利用 L1 和 L2 缓存之间的 128B 事务。

CUDA 线程在执行期间可以访问多个内存空间中的数据。每个线程都有私有本地内存。每个线程块都有对该块的所有线程可见的共享内存，并且与该块具有相同的生命周期。线程块簇中的线程块可以对彼此的共享内存执行读、写和原子操作。所有线程都可以访问相同的全局内存。还有两个可供所有线程访问的附加只读内存空间：常量内存空间和纹理内存空间。全局、常量和纹理内存空间针对不同的内存使用进行了优化。纹理内存还为某些特定的数据格式提供不同的寻址模式以及数据过滤。全局、常量和纹理内存空间在同一应用程序的内核启动过程中是持久的。

2.2.1　Tensor Core

在最新的 GPU 架构中，NVIDIA 引入了 Tensor Core 单元，用于加速密集的深度学习操作，例如密集矩阵乘法 (GEMM)。图 1 展示了带有 Tensor Core 单元的 GPU 流多处理器。需要注意的是，FP64、FP32、INT 和 SFU 分别代表双精度、单精度、整数和特殊功能单元。与 CUDA 核心上的标量计算不同，Tensor Core 单元在寄存器片段上提供基于分片的矩阵计算原语，可以提供超过 10 倍的吞吐量提升。Tensor Core 单元支持计算原语 D=A·B+C，其中矩阵 A 和 B 可以是某种精度类型（如半精度、TF-32），而矩阵 C 和 D 存储在 FP32 中。对于不同的数据精度和 GPU 架构版本，矩阵大小 (M × K、K × N 和 M × N) 需要遵循一些规则。

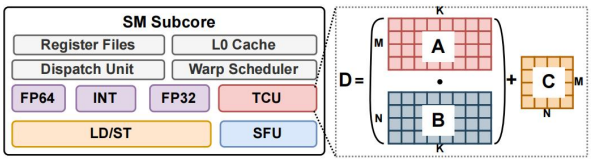

图 1　流式多处理器中的子核架构

Volta 架构首次引入了 Tensor Core 单元，这是一种专门用于加速密集矩阵乘法（GEMM）操作的计算单元。在引入 Tensor Core 计算硬件后，GPU 的峰值算力如表 1 所示。与传统的浮点计算单元（FPU）相比，Tensor Core 单元在峰值浮点运算性能方面提供了高达 8 倍的提升。此外，Volta 的 Tensor Core 单元不仅仅适用于 GEMM 操作，也可以用于其他非 GEMM 应用，如压缩算法等。

表 1 GPU 峰值算力

GPU	FP16	INT8	INT4
V100	126 TFLOPS		
A100	390 TFLOPS	702 TOPS	1248 TOPS
H100	1120 TFLOPS	1696 TOPS	

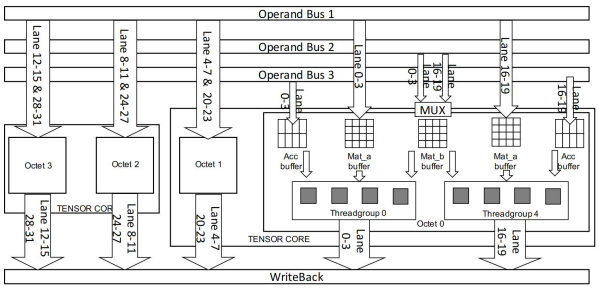

图 2 Volta 架构中 Tensor Core

Volta 架构中的 Tensor Core 单元专注于高计算吞吐量的加速，它的架构如图 2 所示。每个 warp（线程束）同时使用两个 Tensor Core 单元。每个 Tensor Core 单元由两个八位位组（byte lane）控制。在每个八位位组内，线程组具有自己的缓冲区，用于存储 LHS 矩阵（Mat_a 缓冲区）和累加结果（Acc 缓冲区），并且每个线程可以访问一个四乘四的内积单元。RHS 矩阵缓冲区（Mat_b 缓冲区）在每个八位位组内的线程之间共享，其选择由多路复用器控制。

2.3 GNN

图神经网络 GNN（Graph Neural Networks）是一种用于处理图结构数据的机器学习模型。与传统的神经网络模型专注于处理向量和矩阵数据不同，GNN 能够有效地捕捉和学习图数据中的结构和关系。

图由节点和边组成，用于表示实体之间的关系。在图中，节点可以表示对象或实体，边表示它们之间的连接或关联。图结构数据广泛应用于社交网络、推荐系统、生物信息学、化学分子等领域。GNN 通过多个图神经网络层来逐步学习和更新节点的表示。每个图神经网络层将节点的特征信息传递给它的邻居节点，并结合邻居节点的特征来更新节点

的表示。这样，每个节点可以逐渐聚合和整合它周围节点的信息。

GNN 的目标是学习每个节点的低维表示，以便用于下游任务，如节点分类、图分类、链接预测等。通过多个图神经网络层的迭代计算，GNN 能够将节点的局部信息与全局图结构相结合，生成具有丰富语义的节点表示。

图卷积是 GNN 中的核心操作，用于在图结构中传播和聚合节点特征。它通过考虑节点的邻居特征来更新节点的表示。图卷积操作可以根据不同的定义和形式进行扩展，如 GCN（Graph Convolutional Network）、GAT（Graph Attention Network）等。

GNN 利用局部邻居信息和全局图结构来学习节点表示。局部信息通过聚合节点的邻居特征来捕捉节点的局部上下文信息。全局图结构则允许节点之间的信息传播和交互，从而考虑整个图的全局结构和关系。GNN 模型可以使用反向传播算法进行训练，以最小化预测输出与真实标签之间的差距。通过定义适当的损失函数和优化算法，可以进行端到端的模型训练和参数优化。

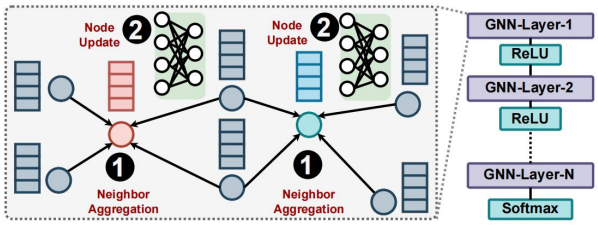

图 3 图神经网络

图神经网络（GNN）是基于图的机器学习的有效工具。GNN 的详细计算流程如图 3 所示。GNN 基本上根据第 k 层 ($k \geq 0$) 的嵌入信息计算第 k+1 层节点 v 的节点特征向量[4]，如公式 1 所示。

$$\begin{aligned} a_v^{(k+1)} &= Aggregate^{(k+1)}(h_u^{(k)} | u \in \mathbf{N}(v) \cup h_v^{(k)}) \\ h_v^{(k+1)} &= Update^{(k+1)}(a_v^{(k+1)}) \end{aligned} \tag{1}$$

从计算的角度来看，GNN 具有聚合阶段的图操作和更新阶段的神经网络操作的交错执行阶段，其主要涉及的是 SpMM 和 SDDMM 两种操作。涉及不规则输入图高度稀疏计算的聚合阶段通常占用 GNN 训练和推理运行时间的 80%以上。

Deep Graph Libary 和 PyTorch Gemetric 等框架大多建立在流行的神经网络框架之上，这些框架最初针对稠密矩阵运算进行了优化，例如通用矩阵-矩阵乘法（GEMM）。为了支持 GNN 中的稀疏计算，它们的常见策略是在后端实现中加入稀疏基元。然而，cuSPARSE 利用稀疏线性代数算法，该算法涉及对稀疏矩阵的非零元素进行大量高成本的间接内存访问。此外，cuSPARSE 设计为仅利用 CUDA 核心。因此，它无法从 GPU 硬件功能的进步中受益，最近的 NVIDIA Ampere 和 Hopper GPU 上的张量核心单元（TCU），这种单独硬件单元设计也是许多其他人工智能特定加速器（例如 AMD GPU 上的 Google TPU 和 Matrix Core）的趋势，并且可以显着提高密集运算。

3 研究现状

近年来，计算机视觉和自然语言处理等领域在强大的深度神经网络推动下取得了快速发展。然而，这些成就是以巨大的内存占用和计算消耗为代价的。为了缓解这个问题，常常采用降低精度和剪枝等方法。在深度神经网络中，稀疏操作是常用的技术，其中稀疏矩阵-矩阵乘法（SpMM）和采样密集-密集矩阵乘法（SDDMM）[5]占据了大部分计算资源。为了加速这两种操作，已经开发了针对高性能 GPU 的内核。

3.1 稀疏化

泛化性能是深度学习模型最关键的方面之一。它衡量模型在未见过的数据上的表现，这些数据具有与训练数据相同的分布。稀疏化方法通常遵循奥卡姆剃刀原则（Rasmussen and Ghahramani），如图 4 所示。当开始进行稀疏化时，准确性通常会提高，这是因为学习噪声减少的影响。直观地说，较小的模型具有更强的正则化效果，迫使学习算法更加关注模型的重要和一般性方面。

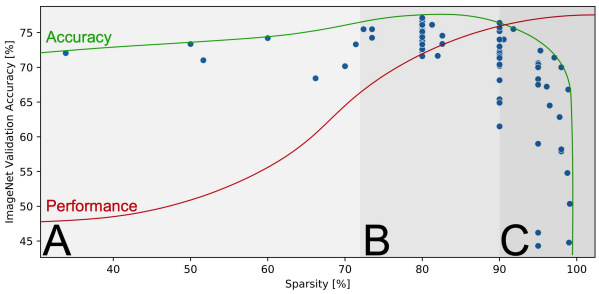

图 4 稀疏度和准确率曲线

随后模型逐渐达到一定的稀疏度范围，性能保持稳定或略有下降。在这个阶段，模型在权衡稀疏性和准确性之间。最后，随着稀疏度的进一步增加，模型的质量会迅速下降。

稀疏性并不总是能够显著提高模型的可解释性

和泛化能力，特别是对于当今具有数百万或数十亿个参数的大型模型。在这些情况下，稀疏化的影响尚待进一步观察和研究。因此，对于大规模深度学习模型，除了稀疏性之外，还需要探索其他方法和技术来提高泛化性能和可解释性。

将模型和短暂稀疏化区分开来[6]，如图 5 所示。模型稀疏化改变了模型本身，可以被看作是神经架构搜索（NAS）方法的一种推广。NAS 是一类自动寻找更好深度学习模型的方法。

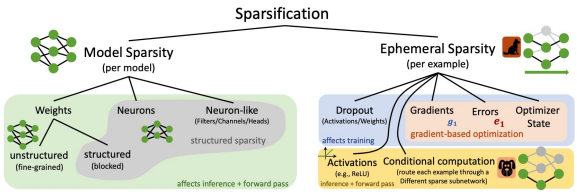

图 5 神经网络模型稀疏化

模型稀疏化会改变模型本身，但不会改变多个推理或前向传递中的稀疏模式。权重和神经元这两个主要元素可以进行稀疏化。在特定层中的元素，例如卷积层中的滤波器或注意力层中的头部，类似于剪枝上下文中的神经元，也可以被删除。神经元、滤波器和头部的稀疏化减少了模型的简单参数，可以显著减小模型的大小，并得到本质上密集的新模型，即可以在与原始模型相同的硬件上有效执行。如果对任意权重进行稀疏化，得到的模型可能是非结构化的，这时可能需要记住前面描述的索引。这增加了索引结构的开销，并导致在针对密集计算进行优化的硬件上执行效率较低。然而，权重稀疏化的粒度非常细，因此已经开发出了结构化权重稀疏化的方法，以减少索引开销并提高执行效率。这些方法通常存储连续的权重块而不是单个元素。

另一类稀疏化方法是短暂稀疏化，它是针对每个样本在计算过程中单独应用的，并且与该样本相关。最常见的结构稀疏化适用于激活函数，事实上，广为人知的 ReLU 和 Softmax 操作会自然地导致稀疏化。这两种操作都会根据固定的阈值将某些值设置为零，在 Softmax 的情况下进行舍入。此外，还可以考虑随机激活稀疏化，例如 dropout，或者使用 top-k 稀疏化。另一组短暂稀疏化的元素与基于梯度的训练值相关。SGD 的反向传播阶段使用误差和梯度来更新权重。这两者都可以进行稀疏化，即仅更新部分权重。这可以与前向传播中的短暂稀疏化产生类似的效果，并导致显著的性能改进，特别是在分布式设置中。在这方面，一种选择是延迟局部梯度贡献的更新，直到它们变得显著为止。另一类重要的临时技术是条件计算，其中模型动态地决定每个样本的。

3.2 基于TCU的高性能内核

Chen 等人提出了一种平铺设计[7]，旨在选择最大化内核和计算效率，他们还提出了设计原则。

(1) 减小程序大小以避免溢出指令缓存。

(2) 增加网格大小以通过线程级并行性（TLP）隐藏延迟。

(3) 通过循环展开、在编译时计算偏移和常数，以及将浮点操作合并到 HMMA 与 TCU 中，减少固定延迟操作。

(4) 在不使用共享内存的情况下，直接将数据加载到寄存器中，减少重用机会。

(5) 使用 128B 合并事务和长向量内存操作（LDG.128）来提高带宽利用率。

根据上面这些高性能内核设计准则，将 1-D 平铺分配给 CTA（Cooperative Thread Array），而不是子线程束（subwarp），这样可以将网格大小的上限设置为 M。此外，在准则 1 和 3 的指导下，利用 Tensor Core 单元(TCU)，因为它能够将多个 HMUL（Half Precision Multiply）和 FADD（Floating Point Addition）指令合并为单个 HMMA（Half Precision Matrix Multiply and Accumulate）指令。此外，TCU 具有固定的计算模式，可以减少索引计算的开销。基于这些特点，在图 6 中展示了基于 TCU 的 1-D 线程束平铺设计。

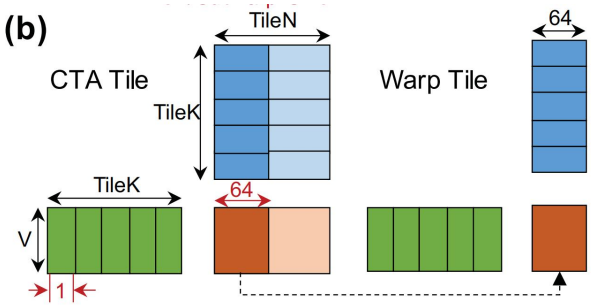

图 6 1-D 线程束平铺设计图

一维 CTA 切片进一步分解为(V × tile_K) × (Tile_K × 64)的 warp tile。尽管这种平铺设计具有较高的内核和计算效率，但其内存访问模式并不是最优的。为了解决这个问题，图 10 说明了如何经典地将 warp tile 进一步分解为每个线程。由于的内核目标是 V 为 2，4 或 8。使用 CUDA C++中的 wmma.m8n32k16 来减少计算浪费。通过这种优化，可以提高内存访问效率并进一步提升性能。

SDDMM（Structured Dense Dense Matrix Multiplication）是基于 TCU（Tensor Core Unit）的一维线程束平铺方法。在该方法中，每个 CTA（Cooperative Thread Array）图块仅包含一个一维图块，而该图块进一步分解为尺寸为(V × 64) × (64 × Tile_N)的 warp tile。图 7 展示了 warp tile 的处理步骤，每个步骤都使用 wmma.m8n32k16 指令。

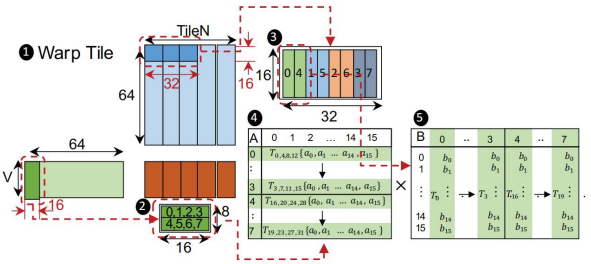

图 7 SDDMM 数据布局和计算步骤

与 SpMM（Sparse Matrix Multiplication）内核类似，在准则 1、2 和 3 下，SDDMM 方法也有较高的计算和访存效率而具有较高的内核性能。相比之下，SDDMM 方法使用更少的寄存器来存储部分和。然而，不利之处在于其内存访问模式并非最佳。图 7 展示了每个 wmma.m8n32k16 指令的操作数布局。在图 7 中，每个块表示一个 4 × 16 的数据块，其中的数字表示线程组所保存的备份。图中展示了具体的数据布局，其中每个块代表一个 16 × 4 的数据块，数字表示持有该块的线程组。

以图 7 中线程组 0 和 4 的数据布局为例进行说明。如果遵循准则 4，直接将 LHS（Left-Hand Side）片段和 RHS（Right-Hand Side）片段加载到寄存器中，只能进行 16B 的合并访存。具体而言，图 7 中行优先排列的每一行和列优先排列的每一列中的 16 个操作数是连续的，并且它们被映射到线程的 16 个寄存器中。然而，每个 128 位全局内存加载指令 LDG.128 只能加载其中的 8 个操作数，因此无法实现完全的 16B 合并访存。另一方面，如果引入 V 并将全局内存访问与共享内存合并，将违反准则 4。此外，LHS 片段被复制了 4 次，这会消耗额外的寄存器资源，从而降低了可用的寄存器数量。此外，Tile_N 必须是 32 的倍数，这在处理残差时会引入额外的开销。最后，当 V 小于 8 时，可能会出现冗余计算的情况。

为了优化 SDDMM 内核，需要改进内存访问模式以提高性能。同时，还可以考虑引入其他优化策略，如数据重排、共享内存的使用等，以进一步提升计算效率和减少资源消耗。

3.3 稀疏图变换

Wang 等人提出的 TC-GNN 图神经网络加速框架[8]中，作为 TC-GNN 的主要组成部分，引入了一种新颖的稀疏图变换（SGT）技术来促进 GNN 的 TCU 加速。其设计的核心思想是，通过有效的图结构操作，可以在保证输出正确性的同时，很好地调整图稀疏性的模式，以用于 TCU 计算，变换过程如图 8 所示。 基于主要观察结果是，邻居共享在现实世界的图中很常见，并且已被用于链接预测等各种任务[11]。

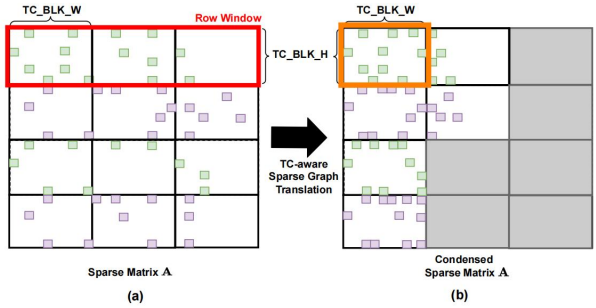

图 8 SGT 变换过程

与使用规则矩阵 tile 覆盖不规则分散的非零元素的稀疏矩阵格式相比，稀疏图变换 SGT 减少了非零元素布局的不规则性，以将它们容纳到更少数量的 TCU 块，从而减少不必要的计算和内存开销。SGT 适用于 GNN 稀疏操作中的 SpMM 和 SDDMM，并且可以轻松并行化，因为各个行窗口的处理是独立的。 在大多数情况下，SGT 只需要执行一次，其结果可以在 GNN 训练和推理的多个 epoch 轮次中重复使用。

除了压缩稀疏图块的有效方法之外，下一个主要挑战是如何定制 GNN 算法的计算，以便于可以利用压缩稀疏图和强大的 TCU 的性能。关注 GNN 中的两种主要计算类型。GNN 稀疏计算的主要部分是邻居聚合，通常可以通过许多最先进的框架将其形式化为 SpMM 操作。在 CUDA 核心上采用 cuSPARSE 作为支持稀疏 GNN 计算的技术。相比之下，TC-GNN 设计针对 TCU 进行主要邻居聚合计算，这需要专门的算法设计。 TC-GNN 专注于通过将原本高度不规则的 SpMM 优雅地批处理为密集 GEMM 计算并在 TCU 上有效求解来最大化净性能增益。节点聚合处理来自每个行窗口的所有 TC 块。将边的特征信息纳入图神经网络模型中对于算法性能是十分重要的。生成边的特征的

基本构建块是采样的稠密矩阵乘法（Sampled Dense-Dense Matrix Multiplication，SDDMM）类似操作。在 TC-GNN 中，通过以上稀疏图转换和针对 TCU 定制的算法设计来支持 SDDMM。整体算法结构和输入与上述邻居聚合类似。主要区别在于输出。对于邻居聚合，输出是更新后的稠密节点嵌入矩阵，而边的特征计算将生成与图边列表形状相同的稀疏输出。

在与针对 TCU 优化的算法设计进行协作时，将算法设计有效映射到底层 GPU 原语是实现高性能传递不可或缺的。与之前关注仅限于 CUDA 核心的工作不同，TC-GNN 通过有效的两级工作负载映射，突出了 CUDA 核心和 TCU 的协作。这个想法基于 CUDA 核心以 SIMT 方式工作，并由单独的线程操作，而专门用于 GEMM 计算的 TCU 需要来自一个线程束 32 个线程的协作。关键的设计原则是将这两种类型的计算单元混合为一个单一的 GPU 内核，可以有效地协调不同级别执行粒度的内核执行，其解构执行过程如图 9 所示。

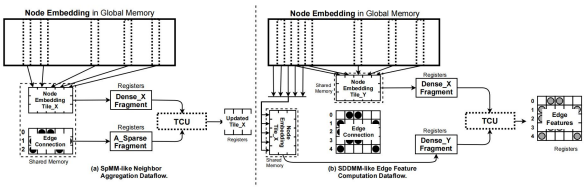

图 9 TC-GNN 解构执行过程

在 TC-GNN 中，通过线程块操作 CUDA 核心，并通过线程束管理 TCU。来自同一线程块的 CUDA 核心线程将从全局内存加载数据到共享内存。需要注意的是，在设计中，将每个行窗口分配给一个线程块。每个块中的线程数应该能够被每个线程束 32 个线程的线程数整除，以获得更好的性能。一旦运行在 CUDA 核心上的线程完成数据加载，每个 TCU 线程束将操作 TCU 进行 GEMM 计算，包括将数据从共享内存加载到线程本地寄存器，在寄存器中进行 GEMM 计算，将结果在寄存器上累积，将最终结果存储回全局内存。

CUDA 核心线程和 TCU 线程存在较大的重叠，它们都是同一块中的线程，但在不同的时间帧上运行。一般来说，使用更多的 CUDA 核心线程而不是 TCU 线程，考虑到全局内存访问需要更多的并行化。这种两级工作负载分解有两个主要优点。首先，来自同一块的线程可以共同工作，以改善内存访问并行化，更好地利用内存带宽。其次，来自同一块的线程束可以重用加载的数据，包括翻译图的信息和稠密节点嵌入矩阵的块。因此，可以避免冗余的高成本全局内存操作。

3.4 高效的量化操作

量化操作主要是针对 LHS 和 RHS 两个矩阵的操作。将矩阵的数据块送到线程束级矩阵乘法（Warp Matrix Multiply and Accumulate，WMMA）的右手边（RHS）矩阵中。假设矩阵 B 是以行优先的方式进行存储。然而，MMA 的 RHS 矩阵必须以列优先的方式进行布局，两者的布局不匹配。

事先将矩阵 B 进行转置以匹配 RHS 矩阵的列优先布局并不起作用，因为矩阵 B 的列索引是不连续的。因此，Li 等人[9]提出了一种针对矩阵 B 的行主块的高效在线转置策略，其中包括三个主要步骤。首先，线程协作地将矩阵 B 的行从全局内存加载到共享内存中的填充缓冲区。对于 BSn=64 的情况，每行的 64 个 int8 值被合并为一个 64B 的内存事务，从而提高了全局内存访问的效率。值得注意的是，对于更大的 N，可以设置 BSn=128，以合并为 128B 的内存事务，以提高效率。

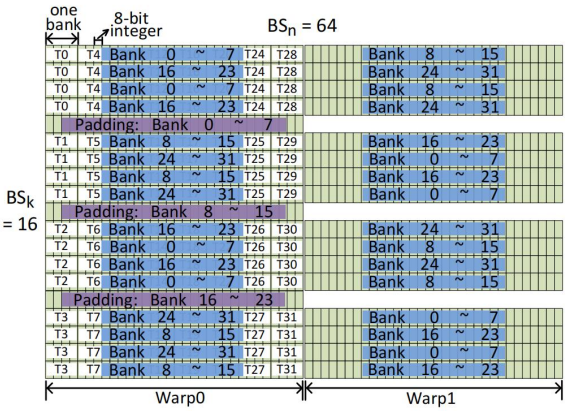

图 10 线程映射图

其次，每个线程将从共享内存中加载 4 个 int32 项到本地寄存器中。如图 10 所示，warp0 中的每个线程访问的项是如何排列的。在 NVIDIA GPU 上，共享内存被划分为多个存储体，每个存储体的宽度为 32 位[10]。线程束中访问不同存储体的线程可以在一个周期内完成服务。否则线程束中访问不同存储体将导致存储体冲突降低性能。通过在每 64 个 int32 项后填充 8 个 int32 项，可以避免在 warp 内出现存储体冲突。最后每个线程对本地寄存器进行按 int8 粒度的转置操作，如图 11 所示。转置后寄存器块的每一行包含矩阵 B 中某一列的 4 个连续 int8 值，满足数据布局的要求。每个线程在转置后的寄存器块中有 4 行，分别输入到 4 个 MMA 操作的 RHS

矩阵中。因此对于一个线程块内的两个 warp，每一步需要执行 8 个 MMA 操作。

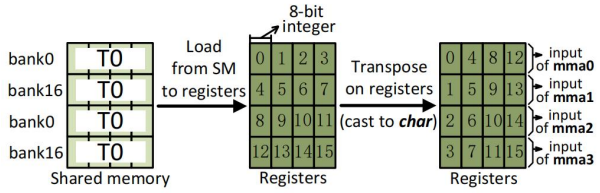

图 11 解决 Bank 冲突

通过上述优化策略，能够高效地将矩阵 B 的数据块转换并馈送到 MMA 的 RHS 矩阵中，以实现更好的性能。矩阵 B 的在线转置策略具有合并全局内存访问、共享内存上的无冲突访问以及本地寄存器上的高性能转置，非常高效，

5 未来研究方向

随着数据规模的增加和计算硬件的升级，深度学习模型变得越来越复杂。然而，大模型的训练和端到端的推理速度往往受到计算和存储资源的限制。为了克服这一挑战，剪枝技术被广泛应用于异构计算领域，以在保持模型精度不受太大影响的前提下加速模型的训练和推理速度。在现代加速硬件如 Tensor Core 的支持下，设计高性能的内核成为当前的主要研究方向。

通过剪枝技术，可以减少模型中冗余和不必要的参数，从而减小模型的规模。这不仅有助于减少存储需求，还可以降低计算负载，提高模型的训练和推理效率。通过剪枝，可以保留模型中最关键的参数和连接，以最大程度地保持模型的性能。
未来，随着深度学习模型的不断发展和数据规模的进一步增加，大模型加速仍然是一个重要的研究方向。研究人员将继续探索更高效的剪枝方法和策略，以实现更大规模模型的快速训练和推理。此外，结合剪枝技术和其他优化方法（如量化、蒸馏等）也是一个有前景的方向，可以进一步提高大模型的性能和效率。

同时，随着计算硬件的不断创新和进步，异构计算平台如 GPU、TPU 等的性能将继续提升。这将为大模型加速提供更强大的计算能力和更高的并行性，进一步推动模型训练和推理的速度提升。因此，未来可以预期在更强大的硬件支持下，大模型加速技术将取得更大的突破和进展，并在各个领域的深度学习应用中发挥重要作用。

4 总结

本文通过对当前主流加速硬件 GPU 和 Tensor Core 的分析，对大模型加速技术进行了深入探讨。首先介绍了 GPU 的相关内容，并详细分析了深度神经网络中的关键计算操作 SpMM 和 SDDMM。在这些背景知识的基础上，对当前的研究现状进行了总结。

研究表明，SpMM 和 SDDMM 占据了深度神经网络计算资源的绝大部分[10]，因此加速这两种计算已成为当前的主要研究方向。为此，基于 Tensor Core 硬件，设计了高性能的内核（kernel），用于加速 SpMM 和 SDDMM, 从而提高模型的训练和推理速度。此外，还对基于高性能内核实现的稀疏图网络深度学习框架 TC-GNN 进行了分析, 并证明其加速能力优于当前主流的图神经网络框架 DGL 和 PyG。

本文对大模型加速技术进行了全面研究。通过优化 SpMM 和 SDDMM 这两个关键计算操作，并结合 Tensor Core 硬件的优势，成功设计了高性能的内核以加速模型训练和推理。此外，TC-GNN 等稀疏图网络框架的出现也为加速大模型提供了新的解决方案。随着深度神经网络的发展，稀疏性将成为一个重要趋势。因此，利用各种异构计算硬件来加速模型将成为未来的主要研究方向。继续优化关键计算操作、设计更高效的内核，以及探索新的硬件架构和算法优化方法，将进一步推动大模型加速技术的发展。这将有助于提高深度学习模型的训练速度、推理效率和实际应用的可行性。

致 谢 非常感谢施展老师、童薇老师和胡燏翀老师在教授数据中心课程过程中的辛勤付出。他们的教导让我深入了解了数据中心作为当前计算机系统发展的前沿技术。在课程中，我获得了关于云技术、ZNS 固态硬盘和纠删码等方面的宝贵知识。在课程讨论环节，我在汇报过程中对于一些知识细节没有查阅充足的文献，施展老师给我指出了汇报过程中的问题，让我能在以后汇报过程中更加注意查阅相关文献。这门课程不仅拓宽了我的知识面，还让我学会汇报过程中的一些知识。

参 考 文 献

[1] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[2] Hestness J, Narang S, Ardalani N, et al. Deep learning scaling is predictable, empirically[J]. arXiv preprint arXiv:1712.00409, 2017.

[3] Gale T, Elsen E, Hooker S. The state of sparsity in deep neural networks[J]. arXiv preprint arXiv:1902.09574, 2019.Hoefler T, Alistarh D, Ben-Nun T, et al.

[4] Garcia Duran A, Niepert M. Learning graph representations with embedding propagation[J]. Advances in neural information processing systems, 2017, 30.

[5] Gale T, Zaharia M, Young C, et al. Sparse gpu kernels for deep learning[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-14.

[6] Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks[J]. The Journal of Machine Learning Research, 2021, 22(1): 10882-11005.

[7] Wang Y, Feng B, Wang Z, et al. {TC-GNN}: Bridging Sparse {GNN} Computation and Dense Tensor Cores on {GPUs}[C]//2023 USENIX Annual Technical Conference (USENIX ATC 23). 2023: 149-164.

[8] Chen Z, Qu Z, Liu L, et al. Efficient tensor core-based gpu kernels for structured sparsity under reduced precision[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021: 1-14.

[9] Li S, Osawa K, Hoefler T. Efficient quantized sparse matrix operations on tensor cores[C]//SC22: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2022: 1-15.。

[10] Kurt S E, Sukumaran-Rajam A, Rastello F, et al. Efficient tiled sparse matrix multiplication through matrix signatures[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-14.

[11] Zhang M, Chen Y. Link prediction based on graph neural networks[J]. Advances in neural information processing systems, 2018, 31.

附录 1.

问题 1:根据汇报PPT第一页，你为什么认为现在图大部分都是稀疏图?

回答 1:根据现在社交网络图结构来看, 每个用户代表一个节点，节点之间有相互交集联系则会有边。用户不会于其他太多节点之间有联系，因此整个图结构是稀疏的。以后面稀疏矩阵为例，来分析了现在的图是稀疏的（在老师的提醒下，需要结合文中的文献来作为严格证据）。

问题 2:文章作者以什么样的动机提出了TC-GNN这个框架？为什么之前没有，他们是第一个使用Tensor Core提出这个框的?

回答 2:当前NVIDIA GPU内部加速硬件正在不断发展。在Volta架构中，引入了Tensor Core这一加速硬件，相较于传统的CUDA Core，Tensor Core在计算SpMM和SDDMM等任务时具有更高的计算效率。然而，要充分利用Tensor Core，需要对矩阵数据的布局进行调整。调用Tensor Core的指令包括m8n8k4、m8n8k16 等，因此需要对矩阵进行相应的调整以适应这些指令的要求。

问题 3:Tensor Core在使用时和传统的CUDA框架有什么不一样?

回答 3:Tensor Core是在CUDA框架下进行使用的，只是在使用的过程中，按照CUDA官方文档的要求，需要严格按照wmma指令或者mma指令对数据进行重排。其中包括half，int等不同的精度，不同的精度其使用tile排列不一样。