ConvStencil

摘要

Tensor Core Unit (TCU) 已广泛集成于现代高性能处理器中，以提升矩阵乘法性能。然而，由于其规格的过度设计，TCU 在提升其他科学计算任务，如 Stencil 计算的潜力，尚未得到充分利用。作者提出了 ConvStencil，一种创新的 Stencil 计算系统，旨在高效地将 Stencil 计算转化为矩阵乘法操作，并充分利用 Tensor Cores。通过引入性能模型，提出了三项关键技术：1) 使用 stencil2row 方法进行内存高效的布局转换；2) 采用双重镶嵌和内核融合的计算适应策略；3) 通过查找表和脏位填充技术进行性能提升。实验结果表明，ConvStencil 显著超越了 AMOS、cuDNN、Brick、DRStencil 和 TCStencil 等现有优化框架，为各类科学与工程应用的计算性能提供了显著加速。

关键词：ConvStencil；Stencil 计算

1 引言

1.1 背景知识

1) Stencil 问题

Stencil 问题是一种重要的计算模式，在科学计算、高性能计算（HPC）以及工程领域有着广泛的应用 [1]。它涉及对多维网格数据的重复更新，通过网格点及其邻域数据的计算来推进系统状态。具体来说，Stencil 问题是一种模式化计算，根据某个点及其固定形状邻域（如十字形或矩形）的数据来更新该点的值，这种操作在一系列时间步中重复进行，最终在整个网格上传播计算结果。其典型公式为

$$u(t+1,x) = f(u(t,x), u(t,x+\ x), u(t,x-\ x), ...) \tag{1}$$

u(t,x) 表示时间 t 时刻位置 x 的值，f 是一个依赖于当前点及其邻域点值的更新函数。意味着某一点的值将由附近的值更新而来。

Stencil 问题广泛出现在流体动力学、气象学和气候模拟、图像处理以及有限差分法等领域。它具有局部性、计算密集和内存访问模式复杂等特点。由于 Stencil 问题常涉及大量的数据访问，内存带宽往往成为性能瓶颈，同时数据重用困难和并行化复杂性也是主要挑战。为了高效解决 Stencil 问题，研究者提出了多种优化策略，如空间分块（Tiling）以提高缓存数据的重用率，时间分块（Time Tiling）以减少内存带宽需求，向量化以利用现代处理器的 SIMD 能力加速计算，以及使用特殊硬件支持如 Tensor Core、FPGA 或专用加速器。

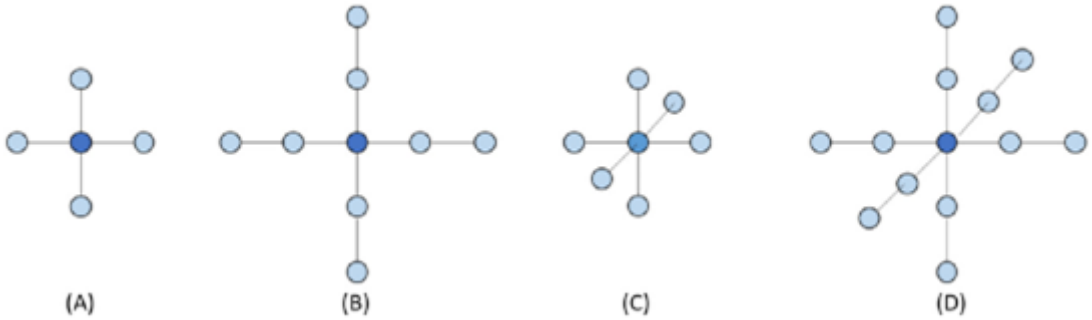

图 1. stencil 计算模型

2) Tensor core

Tensor Core 是 NVIDIA 在 Volta 架构中引入的专用硬件单元，旨在通过加速混合精度矩阵乘法（MMA）操作来显著提升深度学习工作负载（特别是神经网络训练和推理）的性能，并可能为其他计算密集型任务提供优化机会。它采用较低精度的输入矩阵（如 FP16 或 INT8）以减少内存带宽需求，并在输出矩阵累加时使用高精度（如 FP32）保持计算精度。Tensor Core 专注于 4×4 矩阵乘法与累加操作，具有远高于传统 CUDA 核心的吞吐量。尽管其设计高度依赖矩阵乘法模式，灵活性不如通用计算单元，但其在并行计算任务（如 Stencil 计算）中仍展现出巨大潜力，可通过将网格点邻域操作转换为矩阵乘法形式来提高硬件利用率和内存效率，同时处理多个矩阵块以实现并行化。然而，Stencil 计算中的不规则内存访问模式、数据布局变换复杂性、线程分歧与冲突以及对非矩阵任务的适配性挑战，增加了实现难度和性能开销。

图 2. Volta 架构 GPU

1.2 研究意义

Tensor Core 作为现代 GPU 的核心硬件单元，原本主要用于加速深度学习中的矩阵乘法操作，尤其是卷积神经网络。然而，通过深入研究发现，Stencil 计算与卷积在计算模式上存在相似性，这一发现促使将 Stencil 计算映射到 Tensor Core 上，从而不仅极大地扩展了 Tensor Core 的适用范围，还为其在非矩阵乘法任务中的应用探索出了一条新路径。这种创新性的适配不仅实现了硬件资源的高效利用，还极大地推动了 Tensor Core 在科学计算领域的广泛应用。对流体动力学、地球建模和天气模拟等领域产生了些许影响。这一结合是硬件与算法协

同优化的典范，为未来设计更高效的硬件架构提供了宝贵经验，进一步促进了高性能计算的发展，并为科学计算的效率提升提供了新思路与方法。

2 相关工作

将 Stencil 计算映射到 GPU 的 Tensor Core 上，这需要将 Stencil 计算转换为类似矩阵乘法（GEMM）的问题，以进一步提升计算效率。在该过程中将会产生以下问题。

2.1 内存膨胀和低利用率

Stencil 计算，作为一种对多维网格中每个点及其邻域进行操作的计算模式，在尝试适配 Tensor Core 的矩阵乘法操作时，常采用 im2row 方法将网格数据转换为矩阵形式。然而，这一过程却带来了显著的内存膨胀问题。具体而言，由于每个点的邻域数据在展开过程中需要重复存储，导致内存需求急剧增加，尤其是在处理大规模网格时，内存膨胀问题尤为突出 [3]。此外，展开后的矩阵规模远大于原始数据集，这不仅增加了内存占用，还带来了低计算密度的问题。Stencil 计算通常形成小规模或稀疏矩阵，这使得展开后的矩阵更适合进行矩阵-向量乘法而非矩阵-矩阵乘法，从而无法充分利用 Tensor Core 的计算能力。同时，稀疏矩阵中存在大量零元素，这些无效计算不仅增加了计算量，还浪费了宝贵的硬件资源。因此，内存膨胀和低计算密度成为 Stencil 计算在移植到 Tensor Core 时面临的核心挑战，它们直接导致了内存带宽压力和硬件资源浪费等性能瓶颈，使得 Stencil 计算难以充分发挥 Tensor Core 的高性能特点，严重限制了计算效率和硬件利用率。

表 1. 原始方法以及改进方法的内存膨胀倍数

shapes	im2row	stencil2row	Memory Saving
Heat-2D	5	1.5	70.00 %
Box-2D9P	9	1.5	83.33 %
Star-2D9P	9	1.67	81.49 %
Box-2D25P	25	1.67	93.33 %
Star-2D13P	13	1.75	86.54 %
Box-2D49P	49	1.75	96.43 %

2.2 精度和操作限制

Stencil 计算对计算精度有严格要求，特别是需要使用双精度浮点数（FP64）来确保结果的准确性。然而，Tensor Core，这一原本设计用于加速深度学习矩阵乘法操作的硬件单元，在支持 FP64 运算时存在显著限制。具体来说，Tensor Core 在 FP64 模式下仅支持一种非对称的小规模矩阵乘法形式，缺乏灵活性，且不能直接支持大规模矩阵或复杂操作。此外，Stencil 计算的矩阵转换通常会产生不对称矩阵，而 Tensor Core 在 FP64 下对支持的矩阵形状有严格限制，如小规模限制和对称性要求，这进一步增加了算法适配的复杂性。同时，FP64 模式下的 Tensor Core 性能表现也会受到影响，如吞吐量下降和硬件利用率低，这限制了硬件性

能的发挥。因此，Stencil 计算的高精度需求（FP64）与 Tensor Core 在 FP64 模式下的运算和形状支持限制之间的冲突，成为 Stencil 计算在 Tensor Core 上实现的主要挑战之一，对算法适配和性能优化提出了更高要求。

2.3 硬件冲突

将 Stencil 计算移植到 Tensor Core 时，硬件冲突问题成为主要挑战之一。这些问题包括全局内存访问不合并，导致非合并内存访问增加内存访问延迟和带宽利用率低；共享内存银行冲突，由于多线程访问同一银行的不同地址而引发，导致共享内存并行效率显著降低；线程分歧，由于处理不同网格点邻域操作的条件判断或分支语句导致同一 warp 中的线程执行路径不同，破坏了 GPU 的 SIMD 并行模型，增加了计算开销；以及 Tensor Core 计算中的对齐冲突和硬件资源竞争，由于不规则数据布局和访存延迟导致 Tensor Core 等待数据加载，从而限制了其性能潜力。这些硬件冲突问题限制了 Stencil 计算在 GPU 上的性能，成为进一步优化和适配的关键挑战。

3 本文方法

本文采取 Stencil2Row 布局转换、Dual Tessellation 技术和冲突消除策略，来解决以上问题。并同时显著减少了内存开销，提高了计算密度，并优化了冲突处理，从而大幅提升了 Stencil 计算的性能。这一突破不仅使得 Stencil 计算在科学与工程领域的应用更加高效，还扩展了高性能计（HPC）的边界。此外与传统方法相比，该方法在性能提升、适用范围和超越现有技术方面展现出显著优势。

3.1 Stencil2row

针对内存膨胀问题，作者提出了一种专为优化 Stencil 计算设计的技术 Stencil2Row，它通过重新组织内存布局，使 Stencil 计算能够更高效地适配 Tensor Core 的矩阵乘法（MM）操作，同时显著减轻内存膨胀问题。在传统 Stencil 计算中，每个点的计算依赖于其自身及其邻域的数据，这种数据访问模式直接映射为矩阵乘法时会导致数据冗余和内存膨胀。Stencil2Row 的核心思想在于对 Stencil 计算的邻域数据进行压缩式展开，仅保留计算所需的关键数据点，从而大幅减少内存使用。它通过最小化冗余和采用行优先布局，将数据以更高效的方式重新组织，形成新的矩阵表示，直接对齐 Stencil 计算需求，避免了不必要的数据冗余。相较于传统的 im2row 方法，Stencil2Row 能减少 70.0%-96.4% 的内存占用，提高访存效率，减少内存带宽需求，加快数据加载速度，并提升硬件利用率，使转换后的矩阵更符合 Tensor Core 的矩阵计算需求，提高计算密度。

3.2 双重镶嵌和内核融合

1）双重镶嵌（Dual Tessellation）技术

双重镶嵌技术是针对 Stencil 计算中大规模网格数据与 Tensor Core 计算能力限制之间矛盾的一种优化策略。该技术通过将大规模 Stencil 计算分解为适配 Tensor Core 的小规模计算块，实现了数据布局和硬件利用率的双重优化。具体地，它首先在一级镶嵌阶段将输入

网格划分为多个较小的子块，以满足 Tensor Core 的输入限制；然后在二级镶嵌阶段，进一步将子块细分为更小的计算单元，以最大化每个 Tensor Core 计算块的利用率。这种多层次分解不仅提升了硬件利用率，使 Tensor Core 的计算单元更充分地参与计算，还优化了内存访问，使数据更加集中，提高了访存效率。因此，双重镶嵌技术有效解决了大规模 Stencil 计算在 Tensor Core 上的执行难题，显著提升了计算性能。

2) 内核融合（Kernel Fusion）技术

内核融合技术则是针对 Stencil 计算中内核启动开销和内存传输频率过高问题的一种优化手段。该技术通过将多个操作（或时间步）的计算合并到一个内核中，减少了内核调用的开销和内存传输的频率。具体而言，它实现了时间融合和空间融合两种策略：时间融合将多个时间步的计算合并到单个内核中，空间融合则将多个操作或维度的计算合并处理。这种操作合并不仅降低了 GPU 内核启动的频率和开销，还通过减少矩阵中的零元素提升了计算密度，并通过单次访存多次计算减少了冗余访存操作。因此，内核融合技术有效降低了 Stencil 计算的内核开销和内存访问频率，提高了计算密度和内存带宽利用率，从而显著优化了 Stencil 计算的整体性能。

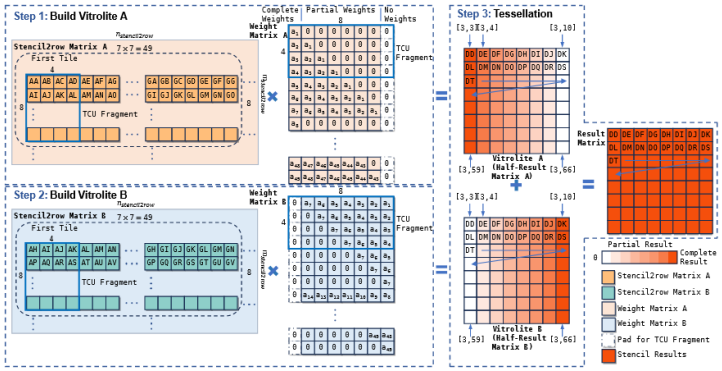

图 3. 双重镶嵌和内核融合示意图

3.3 冲突消除和查找表

1) 冲突消除（Conflict Removal）在 Stencil 计算中的应用 Stencil 计算频繁访问邻域数据，这往往导致共享内存银行冲突和线程分歧两大问题。共享内存银行冲突是因为多个线程可能同时访问共享内存的同一个银行，导致访问被串行化，降低了并行计算效率。而线程分歧则是由于条件分支，使得 warp 中不同线程的执行路径不一致，同样影响了计算效率。为解决这些问题，冲突消除技术通过优化数据布局和访问模式，旨在避免共享内存冲突和条件分支执行，从而最大化硬件资源的利用效率。具体来说，冲突消除技术采用了脏位填充（Dirty Bits Padding）和无条件操作两种方法。脏位填充是在共享内存中插入额外的填充区域，使得不同线程的内存访问对齐，从而消除银行冲突。这些填充区域不仅用于存储冗余数据，还能避免线程间条件分支导致的线程分歧。无条件操作则是通过消除条件判断，将分支逻辑通过预计算替代，统一 warp 中所有线程的执行路径，进一步提升并行计算效率。通过这两种方法，冲突消除技术显著减少了银行冲突，优化了线程执行，提高了 Stencil 计算的并行性和整体性能。

2) 查找表（Lookup Table）在 Stencil 计算中的优化作用在 Stencil 计算中，计算邻域数据的索引和地址通常需要执行复杂的整数除法和取模操作，这些操作开销较大，会显著影响性能。为降低这些计算开销，查找表技术应运而生。查找表技术的核心思想是预计算所有可能的邻域索引并存储在查找表中，从而在运行时直接访问，避免了实时的除法和模运算。在程序初始化阶段，查找表技术会计算出每个网格点邻域的全局内存偏移量，并将这些偏移量存储在查找表中。在 Stencil 计算过程中，直接查表获取邻域地址，无需执行繁重的整数运算。这种方法不仅消除了复杂的运算开销，还提升了访存效率。因为查表操作速度快，简化了邻域数据的加载过程，使得 Stencil 计算的性能得到显著提升。因此，查找表技术是优化 Stencil 计算性能的一种有效手段。

整体流程如下图

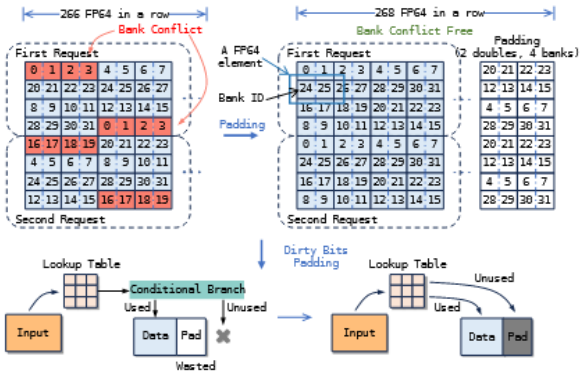

图 4. 冲突消除和查找表

4 复现细节

参照论文方法如 fig5, 完成复现任务。

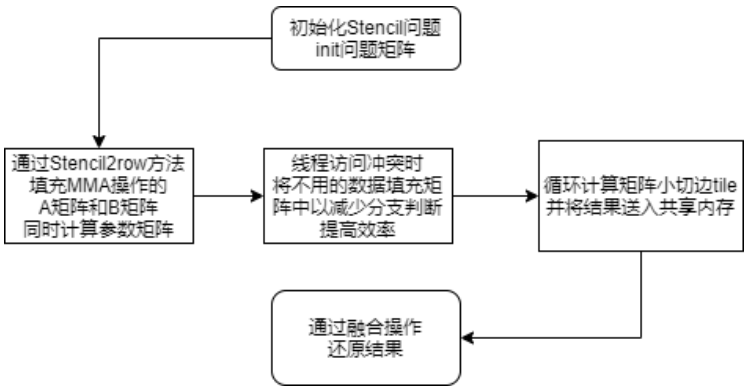

图 5. 算法结构流程图

4.1 与已有开源代码对比

参考 ConvStencil 论文中开源代码 [2]，复现其效果，并实现将其扩展到常见商用卡。

4.2 实验环境搭建

ConvStencil 论文中涉及的数据精度为 FP64 双精度，故其实验运行硬件环境为 GPUA100。软件环境为 CUDA12.6，cmake3.22.1。

5 实验结果分析

在此做了 ConvStencil 和 cuDNN 两种方法在 A100 平台处理不同的问题尺寸下的效率对比效果其结果如下：

问题尺寸		1d3p	1d5p	2d9p	2d49p	3d27p
问题规模		W = 10240000; T = 100000;		H = 10000; W = 10000; T = 10000;	H = 10000; W = 10000; T = 100;	H = 512; W = 512; L = 512; T = 512;
ConvStencil	Time	12566[ms]	12429[ms]	16303[ms]	205[ms]	8680[ms]
	GStencil/s	244.464028	164.775715	184.012977	48.750524	23.749502
cuDNN	Time	13168[ms]	13798[ms]	16285[ms]	693[ms]	24320[ms]
	GStencil/s	77.758549	74.208658	61.404701	14.418883	2.825636

图 6. 实验结果示意

对比于论文上的运行结果图如下：

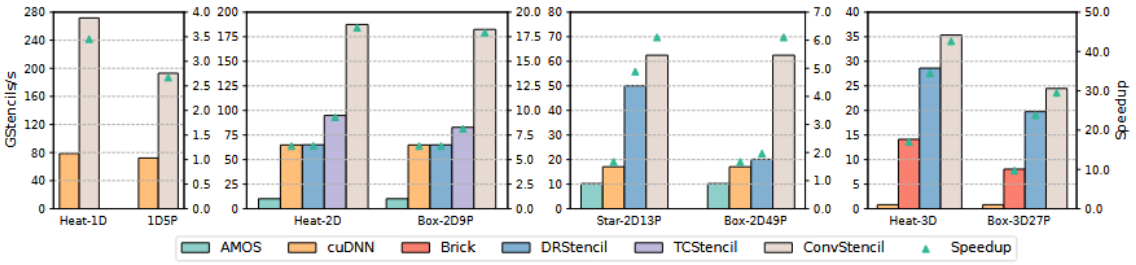

图 7. 论文中实验结果图

6 总结与展望

论文中介绍的技术在 FP64（双精度）和 NVIDIA A100 平台上实现了高效表现，但将其移植到其他精度（如 FP32 或 FP16）或硬件平台（如旧 GPU 架构或非 NVIDIA 硬件）面临多重挑战。精度适配方面，低精度可能导致结果准确性下降，且硬件可能引入额外误差累积。硬件适配方面，其他平台可能缺乏 A100 Tensor Core 的高效 FP64 支持，硬件资源与架构差异可能导致性能瓶颈。此外，算法通用性问题也不容忽视，特定优化方法在其他平台上效果可能大打折扣。展望未来，研究与优化需聚焦于算法适配，设计适用于不同精度的数值稳定性增强技术和混合精度算法；跨平台优化，针对不同硬件平台特点重新设计优化策略；硬件特性敏感性分析，调整优化策略以适应不同硬件；可扩展性研究，支持未来硬件的多样化精度和并行架构；以及引入自适应优化机制，根据硬件特性动态调整策略，以实现算法在不同平台上的高效移植和接近最佳的性能表现。

参考文献

[1] Yulong Ao, Chao Yang, Xinliang Wang, Wei Xue, Haohuan Fu, Fangfang Liu, Lin Gan, Ping Xu, and Wenjing Ma. 26 pflops stencil computations for atmospheric modeling on sunway taihulight. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017.

[2] Yuetao Chen, Kun Li, Yuhao Wang, Donglin Bai, Lei Wang, Lingxiao Ma, Liang Yuan, Yunquan Zhang, Ting Cao, and Mao Yang. Convstencil: Transform stencil computation to matrix multiplication on tensor cores. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, 2024.

[3] Kun Li, Liang Yuan, Yunquan Zhang, and Yue Yue. Reducing redundancy in data organization and arithmetic calculation for stencil computations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.