多方合作项目：大规模计算流体动力学模拟的多GPU性能可移植性分析

Panagiotis-Eleftherios Eleftherakis*、George Anagnostopoulos*、Anastassis Kapetanakis*、Mohammad U mair†、Jean-Yves Vet‡、Konstantinos Iliakis*、Jonathan Vincent†、Jing Gong†、Akshay Patil§、Clara Garc´ıa-S´ anchez§、Gerardo Zampino†、Ricardo Vinuesa¶、Sotirios Xydis*

*雅典国立技术大学，希腊　　†瑞典皇家理工学院，瑞典

‡慧与科技（*HPE*），法国 §代尔夫特理工大学，荷兰，¶密歇根大学，美国　　*A*

arXiv:2601.14159v1 [cs.DC] 20 Jan 2026

摘要——**随着采用GPU的异构超级计算架构日益成为高性能计算（HPC）的核心，作为事实上的HPC工作负载，计算流体力学（CFD）模拟必须能够高效利用此类硬件，这一点至关重要。HPC代码面临的关键挑战之一是性能可移植性，即在不同加速器上保持近乎最优性能的能力。在旨在实现可扩展、GPU加速、多保真计算流体力学并用于城市气流预测的REFMAP项目背景下，本文分析了先进的SOD2D谱元模拟框架在AMD和NVIDIA GPU架构上的性能可移植性。我们首先介绍SOD2D所依据的物理和数值模型，并重点阐述其计算热点。随后，我们从多个层面考察其性能和可扩展性，即界定并刻画一个广泛的全栈设计空间，其中涵盖与应用、软件和硬件基础设施相关的参数。对服务器级NVIDIA和AMD GPU架构以及厂商专用编译器栈进行的单GPU性能表征，展示了内存访问优化的潜力及其不同影响，即加速比偏差为0.69×–3.91× 。此外，我们还在LUMI多GPU集群上进一步考察了SOD2D的大规模性能；性能分析显示吞吐量变化相似，凸显了性能预测的局限性，以及开展多层次、有依据调优的必要性。**

索引词—**性能可移植性、计算流体力学、谱有限元法（有限元法）、高保真模拟、多GPU加速、设计空间探索、可扩展性分析**

I. 引言

计算流体力学（计算流体力学）是现代工程不可或缺的工具，无需昂贵的物理原型，即可对航空航天、汽车工程、海事和环境系统中的流体行为进行详细分析。近年来，计算流体力学生成的数据集也成为训练物理信息神经模型的核心，这些模型将物理定律嵌入机器学习模型中，以提高预测准确性 [1],[2]。计算流体力学的核心是求解纳维–斯托克斯方程，用以表达流体流动中的质量守恒、动量和能量。解析解十分罕见，因此数值方法等这里采用有限差分法（FDM）和有限元法（FEM）。有限差分法适用于结构化网格，而有限元法更擅长处理复杂几何形状，但计算成本较高。

高性能计算极大推动了计算流体力学的发展，使复杂现象的高保真模拟成为可能。异构架构，尤其是利用图形处理器（GPU）的架构，在OpenFOAM、NekRS等先进求解器的支持下，推动了百亿亿次级计算的发展 [3], [4]。然而，由于不同GPU架构、编译器生态系统和编程模型之间存在差异，针对多样化硬件优化计算流体力学求解器仍然具有挑战性。针对某一平台优化的求解器在另一平台上往往性能不佳，从而阻碍了性能可移植性，即在不同架构上保持持续高效的能力。

在此背景下，欧盟资助的**REFMAP** 项目（通过变革性多尺度航空规划减少环境足迹）致力于实现低延迟、多保真度的城市气流预测，以支持城市环境中的可持续创新型空中交通。其高保真度计算流体力学组件将GPU加速模拟与机器学习代理模型相结合。高分辨率直接数值模拟（DNS）数据用于训练可部署于边缘端或云端的紧凑型预测器。由于此类工作负载的计算成本高昂，REFMAP整合了GPU加速计算流体力学与自动调优，以突破数据生成瓶颈。谱元求解器**SOD2D** [5] 通过OpenACC [6] 和MPI移植至GPU，使其能够在NVIDIA和AMD系统上开展可扩展的湍流解析模拟。实现异构高性能计算平台间的性能可移植性，对REFMAP的数据驱动工作流至关重要，可确保代理模型训练与部署具备可扩展性和成本效益。

采用高阶数值格式并支持多架构的计算流体力学求解器——例如 URANOS [7],STREAmS [8],和 PyFR [9]——已实现显著的 GPU加速效果，但跨厂商分析仍然有限。Reguly 和 Mudalige [10] 全面概述了性能可移植性策略，并指出只有少数研究（例如 Clover- Leaf [11],TeaLeaf [12],Nekbone [13]）使用 OpenACC 在多个平台上运行，且直接跨厂商比较的研究有限——

本研究部分经费由欧盟 Horizon 2022 计划根据资助协议 No 101096698 REFMAP 提供（网站：https://www.refmap.eu）。我们感谢欧洲高性能计算联合 undertaking 通过 EuroHPC常规访问征集，授予本项目使用 EuroHPC 超级计算机 LUMI 的权限。该计算机由 CSC（芬兰）和 LUMI联盟托管。

比较。近期的研究[14]同样未对不同厂商的 GPU 进行直接评估。据我们所知，本研究是首批使用 OpenACC 对 GPU加速计算流体力学开展详细、全栈跨厂商研究的工作之一，涵盖从求解器配置到硬件级内存优化的各个环节。

II. REFMAP 项目概述

REFMAP旨在实现低延迟、多保真度的城市气流预测，以支持可持续的创新型空中交通。如图1所示，低保真度分支由OpenFOAM计算流体力学模拟器驱动（并由NVIDIA AmgX求解器加速），执行用于评估街区尺度风相关风险的模型 [15]。在高保真度框架中，则采用SOD2D模拟器 [5] （一种基于Fortran的谱元法代码，使用OpenACC卸载和MPI分区，其中关键内核——即对流、扩散——在单元和节点间并行化），从而实现GPU加速计算流体力学与数据驱动代理模型的耦合。高分辨率大涡模拟/直接数值模拟会生成湍流解析数据，用于训练紧凑模型 [16]；这些模型经过压缩和剪枝等广泛优化，以便部署在边缘端或云端，实现无人机轨迹优化和高保真度传感器布置，进而为高效、安全的无人机导航提供坚实基础。在仅配备CPU的系统上，此类案例需要数周才能完成计算，这限制了代理模型训练所用的数据集。REFMAP通过对SOD2D进行广泛的自动调优和性能探索，克服了这一瓶颈。初步研究表明，使用OpenTuner的AUC-Bandit搜索技术自由调整线程组数量和向量长度时，性能会受到影响，并获得了1.2–1.5× 的加速比 [17], [18]。随着REFMAP迈向成熟阶段，研究采用了多种基础设施，因此需要开展多层次性能探索 [19] ——即对硬件、软件和应用特定参数进行全面探索。在代理模型方面，REFMAP采用全卷积网络（FCN），将壁面剪切应力和压力作为输入，映射为选定壁面法向位置处的速度脉动场 $\hat{u}$，$\hat{v}$，$\hat{w}$ 。模型使用DNS数据（Re_τ=180）进行训练，每个 $y^+ = 15$，30，50，100均包含6,000个快照，并采用均方误差损失和归一化目标值来平衡各分量误差。虽然也在探索其他生成对抗网络和Transformer模型，但全卷积网络仍是与加速计算流体力学相结合、用于基于壁面的流场重建的主要代理模型。借助结构化剪枝、量化和编译器级优化，最终的湍流预测模型在精度损失可忽略不计的情况下，压缩率最高可达 50× ，显著降低延迟并提升吞吐量。这些进展实现了快速、可靠的流场预测，并为将高性能模拟和数据驱动代理模型集成至实时无人机导航和城市风险评估工作流程奠定了基础。本文其余部分将重点分析性能可移植性，以支持在REFMAP工作流程中高效生成代理模型训练数据集。

III. SOD2D：用于可压缩尺度解析模拟的谱有限元法

A. 实现与控制方程

所考虑的物理模型由纳维–斯托克斯方程构成，该方程描述可压缩流体中的质量、动量和能量守恒 [5]。其守恒形式为：

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{1}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) - \nabla \boldsymbol{\tau} = -\nabla p + \boldsymbol{f} \tag{2}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + p)\boldsymbol{u}) - \nabla \cdot (\boldsymbol{\tau} \boldsymbol{u}) - \nabla (\kappa \nabla T) = S \tag{3}$$

其中 ρ、$\boldsymbol{u}$、p、T 和 E 分别表示密度、速度、压力、温度和总能量。系统通过理想气体定律闭合。着色项分别表示**时间演化**、**对流**和**扩散**。方程通过分离时间离散与空间离散求解。空间项采用有限元法（FEM），时间积分则采用四阶龙格–库塔（RK4）格式。

SOD2D [5]是一款基于 OpenACC 的计算流体力学软件，采用 MPI 进程划分技术，因此可面向多个平台。它专注于有限元法（FEM）的一种变体——谱元法（SEM），并可处理非结构网格，从而模拟复杂几何形状。该方法在每个单元内使用非均匀分布节点，同时降低高网格分辨率下的计算成本。此外，软件采用 Gauss–Lobatto–Legendre 求积规则，使求积点与离散网格中的单元节点对齐。

SOD2D上述设计选择体现了其首要目标：创建一种经过硬件优化的计算流体力学模拟器，能够最大限度地提升多核CPU以及单GPU和多GPU架构上的并行性 [5], [17]。在谱元法框架内对偏微分方程进行数学处理后，可得到计算效率较高的对角线性系统。该结构消除了对复杂线性求解器的依赖，而复杂线性求解器往往是传统计算流体动力学应用中的主要瓶颈 [10]。

B. *SOD2D*执行流程

图2展示了SOD2D各个计算内核的执行流程，这些内核以蓝色矩形表示。该算法围绕四个龙格–库塔步构成循环，如 [5]所述。在每一步中，都会计算评估龙格–库塔（RK）函数 $\boldsymbol{f}$ 所需的数值。随后，在每个网格单元 e内的每个点 n 处，计算扩散项和对流项残差。接着，将这两项残差在每个节点处累积，并更新当前RK步中的各个量值。最后，计算熵黏度；这是一种用于稳定流体流动模拟的数值技术，尤其适用于激波解析场景。

重点在于扩散项和对流项的计算，该过程分为三个子部分：预加载、

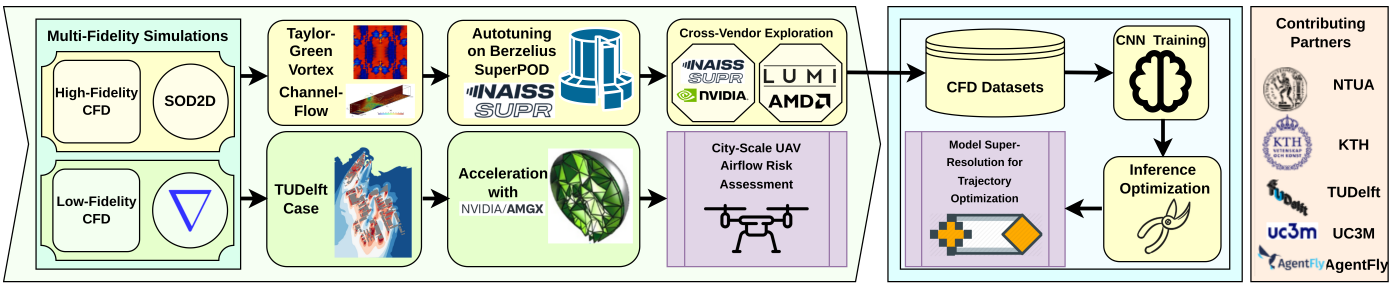

图1. REFMAP 从高性能计算到机器学习的工作流程概览，重点介绍计算流体力学加速。

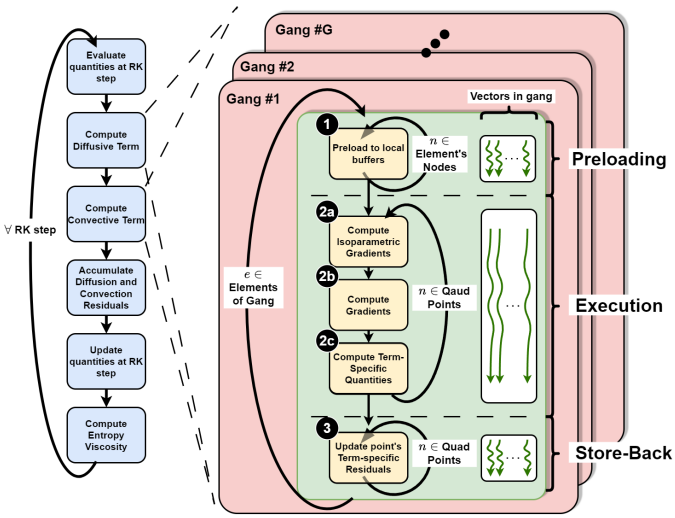

图2. SOD2D 的高层算法概览

执行与回写。这些子部分彼此独立执行，同步屏障以虚线表示。每个项的计算都会遍历网格中的所有元素。该元素循环分布在完全并行的组及组内的向量之间，其定义见 [6]，每个组的大小由内核的执行参数定义。在步骤 中，与当前元素节点对应的全局量被提取到局部数组中，随后主要执行部分开始。第一项操作是计算等参梯度 ，即自然坐标系中的梯度。然后，通过将其与逆雅可比矩阵相乘，计算全局坐标系中的梯度 在下一步 中，计算其他特定于项的量。对于所有求积点都会重复此过程；在谱元法框架中，这些求积点与需要计算其值的节 重合。计算的最后一步是计算当前项中每个点的残差贡献 由于此步骤会更新全局残差数组，因此需要执行多次原子更新，以确保正确的内存同步；这会使内存访问和更新串行化。

我们针对湍流模拟中的两个案例进行分析：TGV（Taylor-Green涡）和 CF（通道流）案例配置。性能分析显示，在这两个模拟案例中，热点行为均表现出一致性，其中完整的_对流（Convection）内核占据了主要运行时间——分别占 NVIDIA 和 AMD 总运行时间的最高达 50% 和 70 %——其次是完整的_扩散（Diffusion）内核，分别占 22% 和 10%。鉴于热点分布具有高度相似性，后续分析聚焦于通道流（CF）案例，因为它更能代表城市环境中的气流；该案例描述了受限于两个平行表面/壁面之间的流体流动，通常形成一个通道。对流内核仍是主要的计算瓶颈，并且对浮点精度、GPU厂商和网格大小表现出显著敏感性，其中网格大小表示非结构化网格中的点数。

IV. 多层次设计空间探索

如图3所示，我们考察应用在所有参数空间层中的行为。我们从应用输入参数的配置开始，逐步扩展到硬件基础设施参数，同时探索代码内存优化。我们的探索框架中的运行时参数层 考虑单精度（FP32）和双精度（FP64）模拟。内存访问优化部分在层次结构上介于运行时参数与软件基础设施空间之间，因为它涉及核心手动代码修改，详见第 IV-A 小节。软件基础设施层 包括所考察的编译器框架，具体而言，我们针对本地NVIDIA GPU使用 NVHPC v24.5，针对 LUMI集群使用 Cray Clang v17.0.1，并采用 AMD GPU。对于硬件基础设施层，我们区分单 GPU 和多 GPU 运行，以及不同的加速器厂商。

有趣的是，初步性能分析结果表明，在处理800万个节点、使用单精度算术且未启用任何代码优化时，单个 AMD MI250X 图形计算芯片（GCD）的性能比 NVIDIA Tesla V100 慢了 15.83×。我们将这些差异归因于四个原因：i）LUMI 上 GPU 的功耗上限仅为400W，而其额定值为500W；ii）使用了 MI250X 的单个 GCD；iii）Cray 编译器工具链可能未针对底层硬件进行优化；iv）与双精度相比，MI250X 的单精度吞吐量相较于 V100 差距更大。

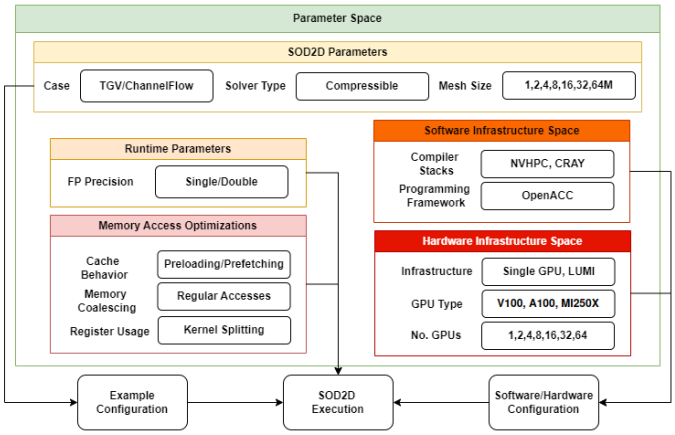

图3. 所研究的多层次设计空间的组织结构。

A. 内存访问优化

使用 Rocprofv3 和 Omnitrace 对多GPU运行中的应用进行性能分析后发现，本地内存使用率已达到上限，同时还存在寄存器使用率过高，以及共享内存和全局内存中的内存访问合并效率低下等问题。为此，我们针对目标内核参数化了内存访问优化策略，并通过手工编写的模板将其应用于源代码，以缓解上述各项问题。下面将介绍内存访问优化的详细信息：

内核拆分： `full_convec`是 SOD2D 的主要热点，并发执行图2中质量、动量和[redacted]方程的步骤 - ，将全局子数组提取到逐元素私有缓冲区中。上述步骤彼此独立，涉及所计算的部分结果和最终输出内存位置。由于 MI250X 上每个线程的寄存器压力增加，即平均 *VGPR* 等于最大值 *128*，平均本地数据共享内存分配等于最大值 *4608*，我们选择将该内核拆分为分别对应每个方程的三个独立内核。这使 5×full_convec立即获得了 1 倍的加速比。

预加载： 预加载策略通常用于 GPU优化代码，以：(1) 改善数据空间局部性；(2) 充分利用线程块内的共享内存；(3) 通过合并内存请求，同时提取多个元素，最大限度地提高内存带宽。

SOD2D 的默认版本采用了这一技术 [5], , 如图2中的[redacted]阶段 所示。该技术应用于需要通过单元索引访问、且最初以不规则方式存储的全局矩阵。由于这些矩阵位于图形处理器的全局内存空间中，并且在图2的执行阶段（ - ）中会被频繁访问，[redacted]会先将它们提取到逐元素私有局部变量中，从而去除单元节点索引，使内存访问更加规则，并更贴近计算。

移除这些预加载变量并直接访问全局矩阵，可在多种运行类型中带来适度的性能提升。我们认为，这一效果归因于局部矩阵的庞大规模，这会导致内存层次结构中最低层的利用效率低下。

预取： 预取是指在特定位置使用 OpenACC 缓存指令，应用于 full_convec 子程序中。根据 OpenACC标准2.7，这些指令与位于循环上方或循环内部的循环相关联，并指定应预取到该循环体最高级缓存中的子数组元素或范围。

直观而言，将预取机制应用于采用间接访问模式的大型全局数组时，无法发挥作用。事实证明，基于范围的全局数组预取效果不佳。因此，我们将该指令与 IV-A 小节中所述的预加载局部数组结合使用。此外，在求积点循环中为每个量预取局部数组同样效果不佳。我们得出的结论是，该指令的最佳用法是针对参与求积循环主体中最深层嵌套循环的量。对于采用 FP64、拆分内核和 V100 GPU 的 CF算例，相较于仅使用预加载，这可带来 1.24× 的加速比。

规则访问： 提高图形处理器内存带宽的一种常见方法是采用内存合并访问。由于次优的合并访问（每个请求 *6.5* 个扇区，最多 *16* 个）会限制性能，我们在 SOD2D 的“无预加载”变体中对全局内存访问进行了规则化。在求积循环（ - ）中，访问遵循`rho(connec(ielem,invAtoIJK()))`，其中`connec` 是网格连通性矩阵，而`invAtoIJK`是索引线性化函数。这类宏会造成严重的性能损失 [8]。由于`connec`和`invAtoIJK` 仅取决于静态网格拓扑，我们预先计算`connec(ielem,invAtoIJK())`，并将其存储在一个按单元节点、求积点和多项式阶数索引的全局矩阵中。后续的内核启动会复用该矩阵，从而实现完美的内存合并访问。尽管`rho` 的访问仍不规则，但耗时的线性化操作已被移除。为提高效率，这些矩阵必须能放入较低层级的内存中。

五、评估

A. 实验设置

本节将介绍评估结果，并深入分析 SOD2D 在参数空间的不同配置下的行为。我们使用可压缩求解器，对网格规模为 100 万、200 万、400 万、800 万和 1600 万个节点的 CF算例进行研究。所有输入算例均在 SOD2D 的 FP32 和 FP64 模式下运行。就 SOD2D 的版本而言，该应用总共编译了 8 次。具体来说，编译的版本包括不使用预加载（**不使用预加载**）、使用预加载（**基础版**）、使用预加载和预取（**预加载 预取**）以及使用常规内存访问（**常规**），其中每种版本均分别采用（**拆分**）和不采用（**统一**）内核拆分。

所有算例和 SOD2D 版本均在 NVIDIA 和 AMD 系统上运行。具体而言，使用了本地服务器中的一块 NVIDIA Tesla V100- PCIE-32GB 图形处理器、来自 Berzelius SuperPOD 的一块 NVIDIA A100 图形处理器，以及由 LUMI 高性能计算集群提供的一块 AMD CDNA™ 2 MI250X-128GB 图形计算芯粒（GCD）。可扩展性实验仅在 LUMI 上进行。

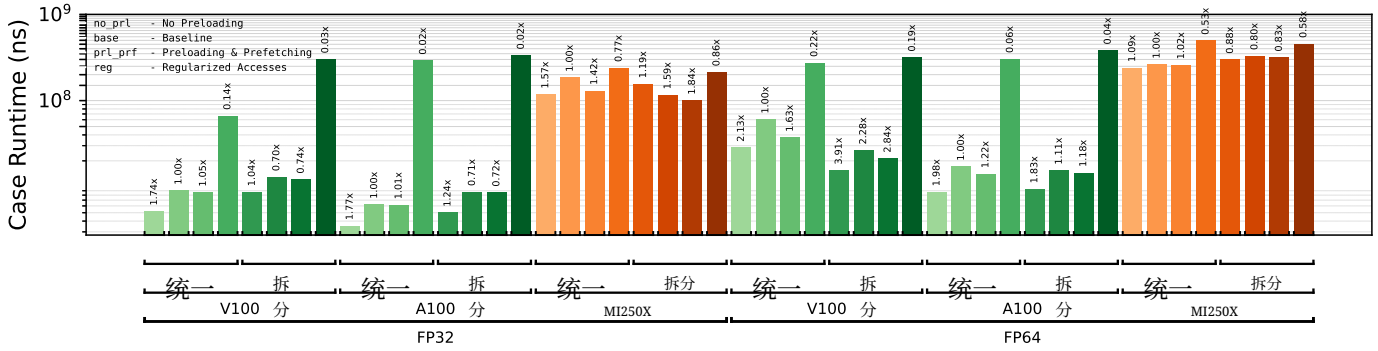

图4. 通道流所有内核优化方案的单GPU运行时间。

B. 单*GPU*探索

由于本节中的 TGV 和通道流（CF）算例结果几乎完全相同，因此我们将重点讨论更贴近实际的 CF算例。多层次参数空间探索和跨厂商单GPU对比分析表明，在扫描不同参数时，主要热点（full_convec）的表现各不相同，这凸显了选择局部最优修改方案的潜力。

如前所述，在 800 万个节点和 FP32 条件下，MI250X 单个 GCD 的运行速度比 V100 慢 15.83× （A100 慢 23.38× ）。SOD2D 的 **prl prf** 版本中，拆分内核的相应减速值为 9.87× 和 8.66×。在双精度条件下，后一个数值为 5.23×。至于 NVIDIA GPU 之间的性能差异，A100 在 FP32 和 FP64 条件下相较 V100 的加速比分别为 1.47× 和 3.05×。

由于 AMD 的性能始终较差，图4中未标注其相较 NVIDIA 的相对减速情况。相反，运行时间是在同一厂商和 FP 精度下进行评估的。每个柱形代表 5 次运行的平均值，运行规模为 100 万至 1600 万个节点，因此该图总计对应 SOD2D 的 240 次运行。

我们注意到，基准版本 SOD2D 已采用第 IV-A 节详细介绍的预加载技术进行了优化。该过程的运行时间开销较大，因为程序流程必须经过与 - 相同的循环边界，才能预加载全局变量。由于 **no prl**版本对缓存的固有预取同样可能很高效，因此尚不明确预加载带来的收益能否抵消 块 的开销。如图4a所示，**no prl**版本始终优于**base(**预加载)版本，但 MI250X 上采用FP64 的拆分版本除外。与 FP32（单精度）相比，这种影响在 FP64（双精度）下更大；对于 FP32，AMD MI250X 上的影响比 NVIDIA GPU 上更明显，而 FP64 下则相反。

预加载与预取结合（**预加载 预取**）始终能提升性能（**基准**）。其影响大多较小，但在 AMD MI250X 的 FP32 上，以及 NVIDIA V100 的 FP64 拆分内核上除外。前者的统一内核和拆分内核分别提升了 42% 和 16%，而后者则分别提升了 63% 和 25%，因此，这分别是 AMD 在 FP32 上和 NVIDIA 在 FP64 上的最佳代码修改。我们再次注意到，优化效果在 NVIDIA 与 AMD 之间、以及 FP32 与 FP64 之间交替出现。

由于规则矩阵的大小为连接矩阵的 18× ，超过了内存层次结构最低层的容量，因此规则访问（**reg**）在不同平台上的表现较差。因此，相较于对较大矩阵进行规则访问，对较小矩阵进行不规则访问更有优势。然而，**reg**是唯一一个在不同运行案例中表现出显著差异的参数。具体而言，在 FP32 下，它在统一内核上的表现更好。在 AMD 平台上，这种表现再次反转。FP64 下也存在类似的细微差异。

内核拆分（**拆分**）的影响在不同厂商和浮点精度之间差异很大，但在不同测试用例中大体一致。对于 FP32，它能加快 AMD 上的执行速度，但会导致 NVIDIA 上的执行速度变慢。有趣的是，对于 FP64，情况恰好相反。

除访问规则化外，上述行为在所有优化中都一致。我们认为，这些差异源于不同架构中 FP32 和 FP64 所使用的寄存器文件大小不同。

C. 可扩展性分析

图 5 展示了对一个弱扩展问题进行 160 次运行后得到的数据。对于每种图形处理器总数配置，问题规模（以节点数衡量）均线性增加，从而使每个图形处理器承担相同的工作量，约为 100 万个节点。

在 SOD2D 可扩展性分析中，我们以吞吐量作为效率指标。因此，我们引入每秒十亿节点操作数（*GNOPS*），定义如下：

$$\frac{\#\text{总网格节点} \times \#\text{龙格–库塔步数} \times \#\text{总时间步数}}{\textit{SOD2D}\text{运行时间（纳秒）}}$$

GNOPS 这一指标源于各节点在各步骤中以一致的方式更新 - 如图2所示。

如图 5 所示，在 FP32（RP4）下，各项优化的扩展表现总体上基本一致（总体而言，FP32 的 GNOPS 比 FP64 高 1.22× 。）。然而，我们观察到，CF 在不同

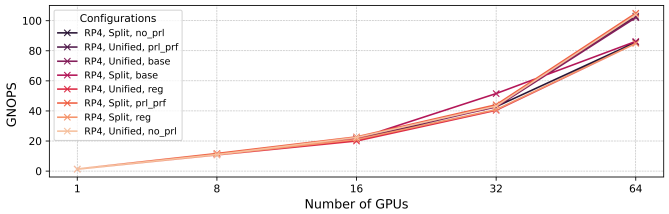

图 5. 通道流的吞吐量扩展性。

内存优化方案的扩展表现方面存在较大差异，不同规模下的最优配置交替变化（在特定节点数下，由于分区效率低下，会出现超线性性能表现 [17]）。我们报告称，对于统一内核，不进行预加载（**无预加载**）是所有 GPU 配置下的最优选择。我们认为这是冗余预加载操作所致。对于 CF，从最差配置到最优配置的性能提升分别为 23.8%（FP32）和 14.6%（FP64）。因此，正如图 5 所示，若根据 32 个 GPU 上的最优方案为 64 个 GPU 上的通道流选择优化方案，将导致吞吐量分别下降 23.8%（FP32）和 8.3%（FP64）。TGV 不存在此类差异，其变化幅度为 12.3%，且最优配置保持一致。以上分析为大规模加速可压缩流体动力学提供了重要启示：不针对具体算例进行区分，仅根据小规模下最优的内存访问决策进行简单外推，来预测不同规模下的性能，是不可靠的；也就是说，需要更复杂的规模化建模。

VI. 结论

本文通过对 SOD2D 框架进行多层次分析，探讨了 GPU 加速的计算流体力学模拟中的性能可移植性挑战。我们重点展示了不同 GPU 厂商之间的架构差异如何显著影响求解器性能，揭示了可扩展性和性能可移植性方面的显著差异，强调了基于参数空间进行优化的必要性。本研究凸显了多层次探索对于开发具备性能可移植性的 CFD 框架的重要性，从而实现向异构高性能计算环境的高效适配。未来的工作将把这一方法扩展到新的编译器和硬件基础设施，以及其他类型的模拟和输入网格，并着重进一步实现参数优化的自动化。

参考文献

[1]R. Vinuesa 和 S. L. Brunton，《利用机器学习增强计算流体力学》，《自然·计算科学》，第 2 卷，第 6 期，第 358–366 页，2022 年 6 月。 [在线]。网址：http: //dx.doi.org/ 10 1038 43588 022 00264 7 。/ s - - - H. [2] Eivazi、M. Tahani、P. Schlatter 和 R. Vinuesa，《利用物理信息神经网络求解雷诺平均纳维–斯托克斯方程》，《流体物理学》，第 34 卷，第 7 期，2022 年 7 月。 [在线]。网址：http://dx.doi.org/ 10 1063 5 0095270 。/ . P. Fischer、S. Kerkemeier、M. [3] Min、Y.-H. Lan、M. Phillips、T. Rathnayake、E. Merzari、A. Tomboulides、A. Karakus、N. Chalmers 等，《Nekrs：GPU加速谱元纳维–斯托克斯求解器》，《并行计算》，第 114 卷，第 102982 页，2022 年。[4] H. Jasak，《Openfoam：研究与工业中的开源 CFD》，《国际船舶结构与海洋工程期刊》，第 1 卷，第 2 期，第 89–94 页，2009 年。

[5] L. Gasparino、F. Spiga 和 O. Lehmkuhl，《SOD2D：用于可压缩尺度解析模拟的 GPU加速谱有限元方法》，《计算机物理通信》，第 297， 109067，页。在线。网址：2024 [ ] 0010465523004125 https://www.sciencedirect.com/science/article/ pii/S[6] O. Organization，《OpenACC应用程序编程接口》2.7 版，2018 年 11 月，https://www.openacc.org/sites/default/files/ inline-files/OpenACC.2.7.pdf。 [在线]。网址：https://www.openacc. org/sites/default/files/inline-files/OpenACC.2.7.pdf [7] F. De Vanna 和 G. Baldan，《Uranos-2.0：提升性能与可移植性，并扩展模型以实现高速工程流动的百亿亿次级计算》，《计算机物理通信》，第 303 卷，第 109285 页，2024 年。 [在线]。网址：https://www.sciencedirect. com/science/article/pii/S001046552400208X [8] S. Sathyanarayana、M. Bernardini、D. Modesti、S. Pirozzoli 和 F. Salvadore，《迈向百亿亿次级计算的高速湍流：STREAmS-2 移植与性能》，《并行与分布式计算期刊》，第 196 卷，第 104993 页，2025 年。 [在线]。网址：https: // www.sciencedirect.com/science/article/pii/S 0743731524001576 [9] F. Witherden、A. Farrington 和 P. Vincent，《PyFR：一种开源框架，利用通量重构方法在流式架构上求解平流–扩散类问题》，《计算机物理通信》，第 185， 11， 3028 3040，卷，第 期，第 – 2014 [ ] 页。在线。网址：https://www.sciencedirect.com/science/article/ pii/S 0010465514002549 [10] I. Z. Reguly 和 G. R. Mudalige，《计算流体力学应用的生产力、性能与可移植性》，《计算机与流体》，第 199 卷，第 104425 页，2020 年。 [在线]。网址：https: //www.sciencedirect.com/science/article/pii/S 0045793020300013 [11] S. McIntosh-Smith，《多样化计算机架构上的性能可移植性》，载于《P3MA：第 4 届众核或加速器性能可移植编程模型国际研讨会》，2019 年。[12] R. O. Kirk、G. R. Mudalige、I. Z. Reguly、S. A. Wright、M. J. Martineau 和 S. A. Jarvis，《在现代多核系统上实现热传导求解器迷你应用程序的性能可移植性》，载于《2017 IEEE国际集群计算会议（CLUSTER）》。IEEE，2017 年，第 834–841 页。[13] J. Gong、S. Markidis、E. Laure、M. Otten、P. Fischer 和 M. Min，《采用 OpenACC 和 CUDA Fortran 实现的 Nekbone GPU 性能》，《超级计算期刊》，第 72 卷，第 11 期，第 4160–4180 页，2016 年 11 月。 2016 [ ] 10 1007 11227 016 1744 5 。在线。网址：[14] https://doi.org/ 。/s - - - Z. Dai、L. Deng、Y. Che、M. Li、J. Zhang 和 Y. Wang，《使用高阶非结构化CFD求解器评估五种共享内存编程模型的性能可移植性》，《并行与分布式计算期刊》，第 187 卷，第 104831 页，2024 年。 [在线]。网址：https://www.sciencedirect.com/science/article/pii/S 0743731523002010 [15] A. Patil 和 C. Garcıa-Sanchez，《量化城市几何细节对城市空中交通风险预测的影响》，《可持续城市与社会》，第 132 卷，第 106750 页，2025 年。 [在线]。网址：https://www.sciencedirect.com/science/article/pii/S 2210670725006249 [16] L. Guastoni、A. Guemes Jimenez、A. Ianiro、S. Discetti、P. Schlatter、H. Azizpour 和 R. Vinuesa，《利用卷积网络模型从壁面量预测壁面湍流》，2020 年 6 月。[17] K. Koliogeorgi、G. Anagnostopoulos、G. Zampino、M. Sanchis、R. Vinuesa 和 S. Xydis，《面向城市空中交通的高保真数值模拟多GPU自动调优》，载于《欧洲设计、自动化与测试会议暨展览会，DATE 2024》，西班牙瓦伦西亚，2024 年 3 月 25–27 日。IEEE，2024 年，第 1–6 页。 [在线]。网址：https://doi.org/ 10 23919 58400 2024 10546549 。/DATE . . G. [18] Anagnostopoulos、P. Eleftherakis、A. C. Maroudis、A. Kapetanakis、K. Iliakis、L. Guastoni、M. Umair、J. Vet、J. Vincent、J. Gong、R. Vinuesa 和 S. Xydis，《REFMAP：从高性能计算到高保真CFD模拟，再到面向新一代城市空域机动的优化流动预测模型》，载于《嵌入式计算机系统：架构、建模与仿真——第 24 届国际会议，SAMOS 2025》，希腊萨摩斯，2024 年 6 月 28 日至 7 月 3 日，《计算机科学讲义》系列，L. Carro、F. Regazzoni 和 C. Pilato 编。Springer，2025 年，第 1–16 页。[19] P.-E. Eleftherakis、G. Anagnostopoulos、A. Kapetanakis、M. Umair、J.-Y. Vet、K. Iliakis、J. Vincent、J. Gong、R. Vinuesa 和 S. Xydis，《海报：GPU加速谱有限元流体模拟的性能可移植性：一种跨层探索方法》，载于《第 22 届 ACM国际计算前沿会议论文集》，CF '25 系列。美国纽约：计算机协会，2025 年，第 228–229 页。 [在线]。网址：https://doi.org/10.1145/3719276.3729458