《国际高性能计算应用期刊》XX(X):1–21
©作者 2024 重印与许可:
w
w
w.
s
agepub.co.uk/journalsPer
missions.nav DOI: 10.1177/待定
www.sagepub.com/SAGE

异构 CPU-GPU 架构上全解析流体-粒子模拟的效率与可扩展性

Samuel Kemmler1,2，Christoph Rettinger1，Ulrich R¨ude1,3，Pablo Cu´ellar2 和 Harald K¨ostler1

arXiv:2303.11811v2 [cs.CE] 9 Dec 2024

摘要

当前的超级计算机通常采用同时配备中央处理器和图形处理器的异构架构。与此同时，数值模拟任务往往涉及多物理场情景，其中各组成部分因多种原因（例如架构要求、实用性等）而运行在不同硬件上。这自然促成了一种软件设计：将不同的模拟模块映射到异构架构的不同子系统。本文对全解析含颗粒流的混合四向耦合模拟进行了详细的性能分析。流动的欧拉描述利用图形处理器，而粒子的拉格朗日模型则在中央处理器上运行。首先，采用屋顶线模型预测节点级性能，并表明基于格子玻尔兹曼方法的流体模拟在单个图形处理器上可实现极佳性能。此外，得益于 CUDA 感知型 MPI 通信的高效性以及通信隐藏技术，大规模流动模拟中的 GPU-GPU 通信仅造成适度的性能下降。在 1024 个 A100 GPU 上，并行效率最高可达 71%。尽管流动模拟具有良好的性能特征，但刚性拉格朗日粒子系统的集成需要频繁进行 CPU-CPU 通信，这可能成为瓶颈。此外，本文特别关注 CPU-GPU 通信开销，因为这对于将粒子与流动模拟耦合至关重要。然而，得益于问题感知协同分区，CPU-GPU 通信开销可以忽略不计。基于这一开发过程中的经验，我们提出了混合实现必须满足的四项准则，以便高效利用异构超级计算机。此外，本文还提出了一种对混合实现加速比进行先验估算的方法。

关键词

混合实现、高性能计算、颗粒流、格子玻尔兹曼方法、离散元法

1 引言

Top500 榜单公布了全球性能最强的超级计算机。近年来，Top500 榜单中的异构超级计算机数量（即配备图形处理器（GPU）等额外加速器的系统）稳步增长，截至 2024* 年 6 月约占 39%。此类异构超级计算机的每个节点通常由一个或多个中央处理器（CPU）和图形处理器组成（Kim 等人，2012）。数值多物理场模拟是一种强大的技术，可提供详细数据，从而深入调查复杂的物理现象；然而，获取这些数据具有挑战性，甚至

[1]系统模拟教席，埃尔朗根-纽伦堡弗里德里希-亚历山大大学，埃尔朗根，德国 [2]建筑与结构第7.2处，德国联邦材料研究与测试研究所（BAM），柏林，德国 [3]CERFACS，图卢兹，法国

通讯作者： Samuel Kemmler，系统模拟教席，埃尔朗根-纽伦堡弗里德里希-亚历山大大学，考尔街11号，91058 埃尔朗根，德国 电子邮箱：samuel.kemmler@fau.de

*www.top500.org/statistics/list/，Top500 发布：2024年6月，类别：加速器/协处理器，访问日期：2024年10月18日

不可能通过实验收集这些数据。这类模拟面临的一项重大挑战是计算成本高昂，因此通常在超级计算机上运行。近年来，特别是配备图形处理器的超级计算机，已越来越广泛地用于数值模拟（Shimokawabe 等人，2017；Rohr 等人，2014；Oyarzun 等人，2017），因为它们能提供前所未有的计算能力。文献中利用此类异构超级计算机进行多物理场模拟的一种常见方法是混合实现（Kotsalos 等人（2021）；Feichtinger，2012；Xu 等人，2012），即让不同的模拟模块运行在不同的硬件上。在多物理场模拟中，有若干原因使混合实现成为理想选择，有时甚至是必然选择。首先，此类多物理场模拟中结合的各种方法可能呈现截然不同的计算特性，例如问题规模、并行与串行部分、条件判断和分支。因此，不同模拟模块最适合的硬件架构可能各不相同。其次，如果某个模拟模块占据了整体运行时间的主要部分，仅使用图形处理器加速这一部分，是一种直接、务实且能节省开发时间的方案，优于将整个代码库移植到图形处理器上。第三，实际限制也可能给多物理场模拟带来挑战：并非所有耦合的软件框架和模块都支持相同的硬件，例如模拟的某些部分使用商业软件时。虽然出于上述原因，混合实现已在实践中得到广泛应用，但它们也带来了性能和可扩展性方面固有的挑战。

多物理场模拟的一个典型例子是采用完全解析颗粒的流体-颗粒耦合模拟（即每个颗粒直径对应多个流体网格单元，见 Rettinger2023）。文献中已使用此类模拟来研究河床中沙丘的形成与演变（Rettinger 等人（ 2017）；Schwarzmeier 等人（2023）；Kidanemariam 和 Uhlmann（2014））、分析颗粒介质中的烟囱式流化（Ngoma 等人（2018））、研究冲击射流作用下土壤的侵蚀动力学（Benseghier 等人（2020）），以及分析可移动沉积物床（Vowinckel 等人（2014）；Rettinger等（2022））。在采用完全解析颗粒的流体-颗粒耦合模拟中，也会遇到前文提到的一些需要混合实现的原因。首先，采用完全解析方法将粒子模拟与流体模拟耦合时，粒子数量通常至少比流体网格单元数量少三个数量级，导致工作负载不均衡，流体模拟可能会占据总模拟运行时间的绝大部分。因此，对于这一应用，仅使用图形处理器加速流体模拟是一种务实的方案。其次，本文采用的流体模拟使用结构化笛卡尔网格方法论，因此非常适合 GPU 并行化（Kuznik 等人（2010）；Holzer 等人（2021））；而粒子模拟的情况则更为复杂。一种极端情况是分子动力学模拟，其中包含数百万个粒子，但粒子间相互作用相对简单且均匀，因此非常适合使用图形处理器（Machado 等人（2021））。另一种极端情况是粒子模拟，其中粒子数量少但体积大、形状复杂，且粒子间相互作用多样而复杂（Iglberger 和 R¨ude 2010），这类模拟可能更适合使用中央处理器。本文采用的粒子方法介于两者之间。虽然使用了球形颗粒，但粒子数量相对较少（因为采用了完全解析方法），且颗粒间相互作用比文献中典型的、在图形处理器上运行的分子动力学或离散元法（DEM）模拟更复杂（例如润滑修正）。未来若引入更复杂的颗粒形状或额外的颗粒间相互作用（如内聚力），这种影响还会加剧。

文献中有大量流体-颗粒模拟以某种方式采用了混合并行化，下面我们将介绍其中具有代表性的一些案例。一个例子是在流体中模拟可变形体（采用有限元法），即细胞血流中的血细胞（Kotsalos 等人（2021））。在这种情况下，研究重点更多在结构力学上，因为有限元法占据了总运行时间的大部分，因此它是代码中被加速的部分。文献中也采用过混合方法，将运行在中央处理器上的商业计算流体动力学（CFD）求解器（例如 Ansys（He 等人，2020；Sousani 等人，2019）或 AVL fire（Jajcevic 等人，2013））与运行在图形处理器上的离散元法（DEM）耦合。此外，研究人员还通过混合并行化加速模拟，以深入了解流化床（Xu 等人，2012；Norouzi 等人，2017），或研究孔隙率和曲折度对土壤渗透性的影响（Sheikh 和 Pak，2015）。另一种方案是使用图形处理器同时处理流体和粒子，而中央处理器则通过碰撞检测支持粒子动力学（Junior 等人，2010）。此外，我们还请读者关注一项采用异构混合方法的相关研究：粒子由中央处理器处理，而流体模拟则分布在中央处理器和图形处理器上（ Feichtinger，2012）。我们选择部分饱和单元法（ PSM）来耦合流体和固相，因为该方法已成功应用于图形处理器环境（Benseghier 等人，2020；

Fukumoto 等人（2021）），这是由于其单指令多数据（SIMD）的特性。

在本研究中，我们提出了若干性能指标；依我们之见，混合实现必须满足这些指标，才能在不浪费资源的情况下高效利用异构超级计算机。首先，混合实现引入的开销（即 CPU-GPU通信）相对于总运行时间必须可以忽略不计。其次，性能关键部分（在本研究中为流体模拟）应在图形处理器上展现出良好的性能。第三，图形处理器运行时间占总运行时间的比例必须足够高，才能证明使用异构集群是合理的。第四，该实现必须展现出令人满意的弱扩展性能，以便高效利用多个超级计算机节点。本研究的主要贡献之一，是提出一种用于完全解析流体-颗粒模拟的多CPU多GPU实现，并满足上述四项指标。为评估所提出的实现，本文在一台先进的异构超级计算机上进行了深入的性能分析，并对流化床模拟的两个对比鲜明的案例进行了比较，分别对应固体颗粒相的密集状态和稀疏状态。本文重点介绍所提出的混合实现在超级计算机上的性能与可扩展性方面的经验，而非阐述物理结果。文献中采用可比粒子方法的流体-颗粒耦合实现仍常局限于中央处理器（Rettinger 等人（2017）；Kidanemariam 和 Uhlmann（2014）；Vo winckel 等人（2014））。在本研究中，我们报告并探讨了所提出的混合实现在超级计算机上的性能与可扩展性经验，以便其他研究人员能够据此作出明智判断，确定这种“部分”加速是否适用于他们的代码。

本文首先在第 2 节介绍数值方法，即格子玻尔兹曼方法（LBM）、离散元法（DEM）及其相应的耦合。随后，第 3 节介绍实现细节。第 4 节对性能进行详细分析，包括 CPU-GPU 通信开销、不同模拟模块的运行时间、屋顶线模型、弱扩展、强扩展性，以及混合加速比的先验估算。第 5 节进一步阐述研究心得，以及这些结果在本研究之外的适用性。最后，本文在第 6 节作总结。

2 数值方法

通常，完全解析的流体-颗粒耦合模拟由三个模块组成：流体动力学、颗粒物理，

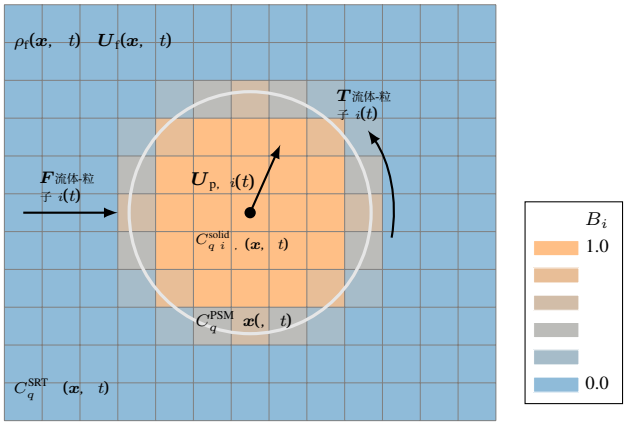


图 1. 使用 PSM 的流体-颗粒耦合模拟二维示意图

以及流固耦合。本节将介绍基于 Rettinger 和 R¨ude（2017，2022）工作的相关方法背景。图 1 展示了采用 PSM 的流体-颗粒耦合模拟，后续章节将对此进行说明。

2.1 格子玻尔兹曼方法

我们使用基于 D3Q19 格子模型的格子玻尔兹曼方法进行流体动力学模拟，作为传统纳维–斯托克斯求解器的一种替代方案（Kr¨uger 等，2017）。对于三维笛卡尔格子的每个单元，我们演化 19 个粒子分布函数（概率分布函数） f_q with $q \in \{0, \ldots, 18\}$。每个 f_q 都与一个格子速度 $\boldsymbol{c}_q$相关联。其基本更新规则基于玻尔兹曼方程，通常分为迁移和碰撞两个步骤。单元局部碰撞使概率分布函数向热力学平衡态弛豫。迁移将碰撞后的 ˜概率分布函数 f_q 传播到相邻单元。格子单元 $\boldsymbol{x}$ 在时间步 t 的碰撞步骤定义为

$$\widetilde{f}_q(\boldsymbol{x}, t) = f_q(\boldsymbol{x}, t) + C_q(\boldsymbol{x}, t) + F_q(\boldsymbol{x}, t), \tag{1}$$

其中 C_q 为碰撞算子， F_q 为外力算子。迁移步骤

$$f_q(\boldsymbol{x} + \boldsymbol{c}_q \Delta t, t + \Delta t) = \widetilde{f}_q(\boldsymbol{x}, t) \tag{2}$$

将概率分布函数分配到相邻单元。 Δt 是时间步长，在格子玻尔兹曼方法中通常为 1。尽管近期研究在 PSM 的背景下采用了更复杂的碰撞算子（Wang 等，2018），但单松弛时间（SRT）模型（也称为 BGK模型（ Qian 等（1992））） 在 PSM 中仍被广泛使用，下面将对此进行介绍。它使概率分布函数向其平衡态弛豫

使用单个弛豫时间 τ

$$C_q^{\mathrm{SRT}}(\boldsymbol{x},t)=\frac{\Delta t}{\tau}(f_q^{\mathrm{eq}}(\rho_{\mathrm{f}},\boldsymbol{U}_{\mathrm{f}})-f_q(\boldsymbol{x},t)). \tag{3}$$

弛豫时间 τ 与流体运动黏度 ν 的关系为

$$\nu=(\tau-\Delta t/2)c_{\mathrm{s}}^2. \tag{4}$$

平衡态定义为

$$f_q^{\mathrm{eq}}(\rho_{\mathrm{f}},\boldsymbol{U}_{\mathrm{f}})=w_q\left(\rho_{\mathrm{f}}+\rho_0\left(\frac{\boldsymbol{c}_q\cdot\boldsymbol{U}_{\mathrm{f}}}{c_{\mathrm{s}}^2}+\frac{(\boldsymbol{c}_q\cdot\boldsymbol{U}_{\mathrm{f}})^2}{2c_{\mathrm{s}}^4}-\frac{\boldsymbol{U}_{\mathrm{f}}\cdot\boldsymbol{U}_{\mathrm{f}}}{2c_{\mathrm{s}}^2}\right)\right) \tag{5}$$

对于不可压缩流（He 和 Luo（1997））， 其中 $\rho_0=1$ 为格子权重 w_q （Qian 等（1992）），格子声速为 $c_{\mathrm{s}}=1/\sqrt{3}$。单元局部量

$$\rho_{\mathrm{f}}(\boldsymbol{x},t)=\sum_q f_q(\boldsymbol{x},t), \tag{6}$$

$$\boldsymbol{U}_{\mathrm{f}}(\boldsymbol{x},t)=\frac{1}{\rho_0}\sum_q f_q(\boldsymbol{x},t)\boldsymbol{c}_q+\frac{\Delta t}{2\rho_0}\boldsymbol{f}^{\mathrm{ext}} \tag{7}$$

根据概率分布函数的矩计算。外力算子

$$F_q(\boldsymbol{x},t)=\Delta t w_q\left[\frac{\boldsymbol{c}_q-\boldsymbol{U}_{\mathrm{f}}}{c_{\mathrm{s}}^2}+\frac{\boldsymbol{c}_q\cdot\boldsymbol{U}_{\mathrm{f}}}{c_{\mathrm{s}}^4}\cdot\boldsymbol{c}_q\right]\cdot\boldsymbol{f}^{\mathrm{ext}} \tag{8}$$

可以使用恒定力密度 $\boldsymbol{f}^{\mathrm{ext}}$ （Ladd 和 Verberg（2001））来纳入外力。

2.2 粒子动力学

采用离散元法（Cundall 和 Strack（1979））对粒子的行为进行建模。作用于粒子的总力 $\boldsymbol{F}_{\mathrm{p},i}$ 由以下模块组成

$$\boldsymbol{F}_{\mathrm{p},i}=\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{col}}+\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{hyd}}+\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{ext}}. \tag{9}$$

除流体动力 $\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{hyd}}$ 和外力 $\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{ext}}$ （例如重力）外，粒子之间还会因碰撞而产生颗粒相互作用力 $\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{col}}$。为模拟粒子的运动，必须对运动方程进行积分。

2.2.1 使用离散元法计算颗粒相互作用 粒子 i 与 j 之间的碰撞采用线性弹簧阻尼模型进行建模。粒子 i 上的碰撞力 $\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{col}}$和力矩 $\boldsymbol{T}_{\mathrm{p},i}^{\mathrm{col}}$ 计算如下:

$$\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{col}}=\sum_{j,j\neq i}(\boldsymbol{F}_{ij,\mathrm{n}}^{\mathrm{col}}+\boldsymbol{F}_{ij,\mathrm{t}}^{\mathrm{col}}), \tag{10}$$

$$\boldsymbol{T}_{\mathrm{p},i}^{\mathrm{col}}=\sum_{j,j\neq i}(\boldsymbol{x}_{ij}^{\mathrm{cp}}-\boldsymbol{x}_{\mathrm{p},i})\times\boldsymbol{F}_{ij,\mathrm{t}}^{\mathrm{col}}, \tag{11}$$

其中，作用于位置为$\boldsymbol{x}_{\mathrm{p},i}$ 的粒子 i 上的碰撞力法向分量 $\boldsymbol{F}_{ij,\mathrm{n}}^{\mathrm{col}}$ 计算如下:

$$\boldsymbol{F}_{ij,\mathrm{n}}^{\mathrm{col}}=-k_{\mathrm{n}}\delta_{ij,\mathrm{n}}\boldsymbol{n}_{ij}-d_{\mathrm{n}}\boldsymbol{U}_{ij,\mathrm{n}}^{\mathrm{cp}}. \tag{12}$$

其中， k_{n} 和 d_{n} 分别为法向刚度和阻尼系数， $\boldsymbol{n}_{ij}$ 为法向量，$\delta_{ij,\mathrm{n}}$ 为穿透深度，$\boldsymbol{U}_{ij,\mathrm{n}}^{\mathrm{cp}}$ 为粒子表面在接触点 $\boldsymbol{x}_{ij}^{\mathrm{cp}}$ 处相对速度的法向分量。碰撞力的切向部分

$$\boldsymbol{F}_{ij,\mathrm{t}}^{\mathrm{col}}=-k_{\mathrm{t}}\boldsymbol{\delta}_{ij,\mathrm{t}}-d_{\mathrm{t}}\boldsymbol{U}_{ij,\mathrm{t}}^{\mathrm{cp}} \tag{13}$$

采用切向刚度和阻尼系数 k_{t} 和d_{t} ， 而 $\boldsymbol{U}_{ij,\mathrm{t}}^{\mathrm{cp}}$ 为粒子表面在接触点处相对速度的切向分量。

$$\boldsymbol{\delta}_{ij,\mathrm{t}}=\int_{t_i}^{t}\boldsymbol{U}_{ij,\mathrm{t}}^{\mathrm{cp}}(t')\mathrm{d}t' \tag{14}$$

表示两个粒子之间累积的相对切向运动，其中 t_i 为碰撞的时间步。更多详情请参见 Rettinger 和 R¨ude（2022）。

2.2.2 粒子属性的积分我们使用速度Verlet积分器求解牛顿-欧拉运动方程，以更新粒子的位置和速度:

$$\boldsymbol{x}_{\mathrm{p},i}(t+\Delta t_{\mathrm{p}})=\boldsymbol{x}_{\mathrm{p},i}(t)+\Delta t_{\mathrm{p}}\boldsymbol{U}_{\mathrm{p},i}(t)+\frac{\Delta t_{\mathrm{p}}^2}{2m_{\mathrm{p},i}}\boldsymbol{F}_{\mathrm{p},i}(t), \tag{15}$$

$$\boldsymbol{U}_{\mathrm{p},i}(t+\Delta t_{\mathrm{p}})=\boldsymbol{U}_{\mathrm{p},i}(t)+\frac{\Delta t_{\mathrm{p}}}{2m_{\mathrm{p},i}}(\boldsymbol{F}_{\mathrm{p},i}(t)+\boldsymbol{F}_{\mathrm{p},i}(t+\Delta t_{\mathrm{p}})), \tag{16}$$

其中 $m_{\mathrm{p},i}$ 是粒子的质量 i。$\boldsymbol{x}_{\mathrm{p},i}(t+\Delta t_{\mathrm{p}})$ 在每个粒子时间步开始时使用旧力计算。随后，使用更新后的位置计算新力 $\boldsymbol{F}_{\mathrm{p},i}(t+\Delta t_{\mathrm{p}})$。在时间步结束时，使用更新后的力计算颗粒速度 $\boldsymbol{U}_{\mathrm{p},i}(t+\Delta t_{\mathrm{p}})$。角速度的更新方式类似。

2.3 全解析流体-粒子耦合方法

耦合的任务是在流体与固相之间进行动量交换。我们采用 PSM 实现完全解析的流固耦合（Noble 和 Torczynski，1998年）。该方法通过引入固体体积分数，对式（1）中的 LBM 碰撞步骤进行修正，得到

引入固体体积分数 $B(\boldsymbol{x},\ t)$，从而得到

$$\widetilde{f}_q(\boldsymbol{x},t) = f_q(\boldsymbol{x},t) + C_q^{\mathrm{PSM}}(\boldsymbol{x},t) + (1 - B(\boldsymbol{x},t))F_q(\boldsymbol{x},t), \tag{17}$$

其中，$B(\boldsymbol{x},\ t)$ 是流体网格单元 $\boldsymbol{x}$ 被一个或多个粒子部分或完全覆盖的比例。第 2.3.2 节详细说明了这种固体体积分数计算。式（17）中使用的修正碰撞算子 C_q^{PSM}定义如下：

$$\begin{aligned} C_q^{\mathrm{PSM}}(\boldsymbol{x},t) = (1 - B(\boldsymbol{x},t))C_q^{\mathrm{SRT}}(\boldsymbol{x},t) \\ + \sum_I B_i(\boldsymbol{x},t)C_{q,i}^{\mathrm{solid}}(\boldsymbol{x},t). \end{aligned} \tag{18}$$

C_q^{SRT}是式（3）中描述的格子玻尔兹曼方法碰撞算子。$B(\boldsymbol{x},\ t)$ 是所有粒子 I各自重叠分数$B_i(\boldsymbol{x},\ t)$之和。如果 $B(\boldsymbol{x},\ t) > 1$，则将其归一化为 1。当碰撞粒子在接触过程中允许相互重叠时，就可能出现这种情况。此时，单个流体网格单元甚至可能被两个粒子完全覆盖，即 $B(\boldsymbol{x},\ t) = 2$。固体碰撞算子

$$\begin{aligned} C_{q,i}^{\mathrm{solid}}(\boldsymbol{x},t) = [f_{\bar{q}}(\boldsymbol{x},t) - f_{\bar{q}}^{\mathrm{eq}}(\rho_{\mathrm{f}},\boldsymbol{U}_{\mathrm{f}})] - [f_q(\boldsymbol{x},t) \\ - f_q^{\mathrm{eq}}(\rho_{\mathrm{f}},\boldsymbol{U}_{\mathrm{p},i}(\boldsymbol{x},t))] \end{aligned} \tag{19}$$

在粒子与单元相交时起作用。固体碰撞算子有多种不同形式。$f_{\bar{q}}$ 对应于 f_q的反向格子速度。

$\boldsymbol{U}_{\mathrm{p},\ i}(\boldsymbol{x},\ t)$ 是在单元中心 $\boldsymbol{x}$ 处计算得到的粒子 i 速度，其计算公式为：

$$\boldsymbol{U}_{\mathrm{p},i}(\boldsymbol{x}_i,t) = \boldsymbol{U}_{\mathrm{p},i}(t) + \boldsymbol{\Omega}_{\mathrm{p},i}(t) \times (\boldsymbol{x}_i - \boldsymbol{x}_{\mathrm{p},i}(t)), \tag{20}$$

其中，颗粒平动速度为$\boldsymbol{U}_{\mathrm{p},\ i}(t)$，颗粒转动速度为$\boldsymbol{\Omega}_{\mathrm{p},\ i}(t)$，颗粒质心为$\boldsymbol{x}_{\mathrm{p},\ i}(t)$。$\boldsymbol{x}_i$ 是所有与粒子相交的单元的网格单元中心。

到目前为止，我们只考虑了粒子对流体的影响。然而，流体也会通过流体动力作用于粒子。我们计算流体作用于粒子 i 的力 $\boldsymbol{F}^{\mathrm{fp}}{}_{\mathrm{p}\,i}(t)$ 和力矩 $\boldsymbol{T}^{\mathrm{fp}}{}_{\mathrm{p}\,i}(t)$，计算式如下：

$$\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{fp}}(t) = \frac{(\Delta x)^3}{\Delta t}\sum_{\boldsymbol{x_i}}[B_i(\boldsymbol{x}_i,t)\sum_q(C_{q,i}^{\mathrm{solid}}(\boldsymbol{x}_i,t)\boldsymbol{c}_{\bar{q}})], \tag{21}$$

$$\begin{aligned} \boldsymbol{T}_{\mathrm{p},i}^{\mathrm{fp}}(t) = \frac{(\Delta x)^3}{\Delta t}\sum_{\boldsymbol{x_i}}[B_i(\boldsymbol{x}_i,t)(\boldsymbol{x}_i - \boldsymbol{x}_{\mathrm{p},i}) \\ \times \sum_q(C_{q,i}^{\mathrm{solid}}(\boldsymbol{x}_i,t)\boldsymbol{c}_{\bar{q}})]. \end{aligned} \tag{22}$$

2.3.1 润滑修正润滑力和力矩作用于相互靠近的两个粒子上。这

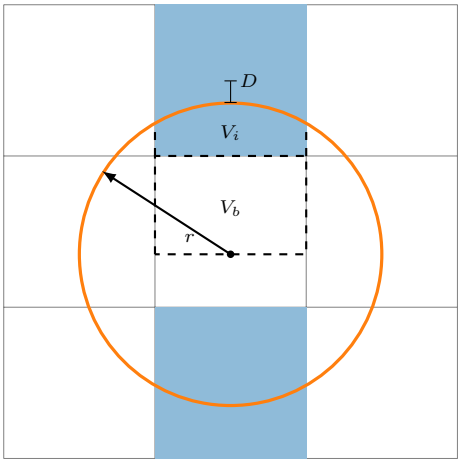


图 2. 线性近似可得蓝色单元的解析解。粒子以橙色圆圈表示。请注意，为了清晰起见，网格经过了粗化处理。

两个粒子将间隙中的流体挤出，流体会沿相对运动的反方向施加力。然而，只有采用极精细的网格分辨率时，流固耦合才能正确涵盖这一效应，但这在计算上代价过高。由于润滑力影响显著，我们计算润滑修正力项，以弥补耦合方法无法正确表示这些力的不足。我们计算由法向平移、切向平移和转动所引起的润滑修正项。因此，总流体动力 $\boldsymbol{F}^{\mathrm{hyd}}{}_{\mathrm{p}\,i}$和力矩 $\boldsymbol{T}_{\mathrm{hyd\,p}\,i}$是完全解析的流体-颗粒耦合方法（第 2.3 节）所产生的力与润滑修正

$$\boldsymbol{F}_{\mathrm{p},i}^{\mathrm{hyd}} = \boldsymbol{F}_{\mathrm{p},i}^{\mathrm{fp}} + \boldsymbol{F}_{\mathrm{p},i}^{\mathrm{lub,cor}}, \tag{23}$$

$$\boldsymbol{T}_{\mathrm{p},i}^{\mathrm{hyd}} = \boldsymbol{T}_{\mathrm{p},i}^{\mathrm{fp}} + \boldsymbol{T}_{\mathrm{p},i}^{\mathrm{lub,cor}}. \tag{24}$$

有关如何计算润滑修正的详细信息，请参见 Rettinger 和 R¨ude（2022）。

2.3.2 粒子映射 采用 PSM 的流体-粒子耦合模拟需要计算固体体积分数 $B_i(\boldsymbol{x},\ t)$（第 2.3 节），即流体网格单元 $\boldsymbol{x}$ 被粒子 i部分覆盖的比例。本文仅考虑球形颗粒。Jones 和 Williams（2017）指出，通常不存在用于计算球体与单元/立方体重叠比例的唯一解析解。他们根据特定网格单元取向下球体与单元表面的解析解，提出了一种线性近似。下文假设网格单元的无量纲边长为 1（格子玻尔兹曼方法中通常如此）。在图 2 中，上方蓝色格子单元与橙色

粒子的重叠比例计算如下

$$\epsilon = V_i = V_a - V_b = V_a - (D + r - 1/2), \qquad (25)$$

其中， V_a 是 V_i 和 V_b的并集。 D 是单元中心到球面的距离（如果单元中心位于球体内部，则为负值）。当 $\epsilon \in]0,\ 1]$时，单元与粒子发生重叠。我们可以将其重新表述为

$$\epsilon = V_a - (D + r - 1/2) = -D + f(r), \qquad (26)$$

其中， $f(r) = V_a - r + 1/2$ 仅取决于粒子半径 r ，因此对每个粒子而言都是常数。 V_a 的计算方式为

$$\begin{aligned} V_a &= \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} \sqrt{r^2 - x^2 - y^2}\mathrm{d}x\mathrm{d}y \\ &= (1/12 - r^2)\tan^{-1}\left(\frac{\frac{1}{2}\sqrt{r^2 - 1/2}}{1/2 - r^2}\right) + \frac{1}{3}\sqrt{r^2 - 1/2} \\ &+ (r^2 - 1/12)\tan^{-1}\left(\frac{1/2}{\sqrt{r^2 - 1/2}}\right) \\ &- \frac{4}{3}r^3\tan^{-1}\left(\frac{1/4}{r\sqrt{r^2 - 1/2}}\right). \end{aligned} \qquad (27)$$

这种近似方法即使对于任意网格单元取向也能得到准确结果，且与细分采样等当前技术相比，计算效率更高（Jones 和 Williams，2017）。

3 实现

我们在大规模并行多物理场框架 WALBERLA（Bauer 等，2021a）（https://www.walberla.net/）中实现了混合耦合流体-粒子模拟。WALBERLA 支持在中央处理器和图形处理器上高效、可扩展地运行格子玻尔兹曼方法模拟（Holzer 等，2021）。MesaPD 模块（Eibl 和 R¨ude，2019）使 WALBERLA 能够使用离散元法在中央处理器上进行粒子模拟。大规模模拟需要使用超级计算机的众多节点。异构超级计算机的每个节点通常由一个或多个中央处理器和图形处理器组成。多个 CPU核心隶属于一块图形处理器，即它们与该图形处理器直接相连。我们将模拟域划分为多个块，并将每个块专门分配给一块图形处理器。图3展示了域划分及相应的硬件分配。我们想强调的是，粒子模拟并非一个独立框架；对于在物理上彼此接近（即处于同一块中）的粒子和流体，由一个 MPI进程同时负责粒子和

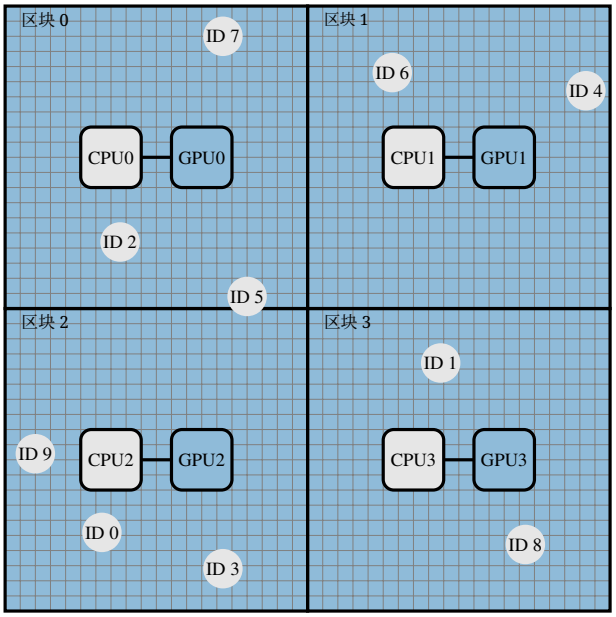


图3. 将二维模拟域划分为四个区块。标记为 ID 0 至 ID 9 的圆圈表示粒子，蓝色单元表示流体。每个区块 x 都分配一台图形处理器 GPUx，用于更新流体单元；相应的中央处理器 CPUx 核心负责粒子动力学。CPUx 表示与 GPUx 直接连接或具有亲和性的 CPU核心。MPI进程 x 分配给 CPUx，使用 OpenMP 在 CPUx 核心之间分配粒子计算，并使用 GPUx 进行流体动力学计算。

流体动力学与中央处理器和图形处理器之间存在直接联系/亲和性。与各图形处理器对应的中央处理器核心负责更新质心位于该块内的粒子（局部粒子）。在图3中，CPU0负责处理ID为2、5和7的粒子。此外，粒子可能与某个块重叠，但其质心位于另一个块内（幽灵粒子）。在图3中，ID为5的粒子对于块0而言是局部粒子，对于块2而言则是幽灵粒子。这种重叠使中央处理器之间必须进行通信。块内的粒子计算通过OpenMP在中央处理器核心之间并行执行。相邻块之间的通信通过CUDA感知消息传递接口实现。在多个图形处理器共享同一节点且图形处理器之间通过NVLink连接的集群上，使用 NVIDIA GPUDirect实现节点上图形处理器之间直接的 GPU-GPU MPI通信。图4展示了模拟中的不同模块、它们运行的硬件、工作流程以及所需的通信步骤。我们将在后续章节中详细解释该图。一般而言，图形处理器负责流体单元上的所有操作（即格子玻尔兹曼方法和耦合），而中央处理器负责执行粒子上的所有计算。因此，相关数据结构分别存储在对应的

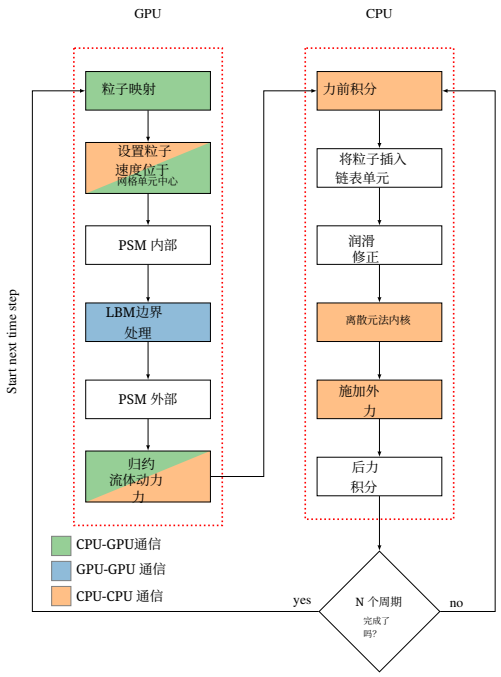


图4. 从负责同一块的中央处理器和图形处理器的角度，展示我们混合中央处理器-图形处理器实现的流程图。颜色编码表示每个步骤中所需的通信类型。

内存中（流体单元位于图形处理器内存中，粒子位于中央处理器内存中）。

3.1 图形处理器上的流体动力学与耦合

执行一次格子玻尔兹曼方法更新步骤，需要在相邻图形处理器之间通信边界单元。然而，前三个内核不需要相邻单元的信息。因此，在粒子映射之前启动非阻塞发送，即可将通信隐藏在这些内核的执行过程中。一个时间步从粒子向流体的耦合开始。

3.1.1 从粒子到流体的耦合 进行粒子映射时，图形处理器必须检查所有单元-粒子组合是否重叠，尽管大多数单元-粒子组合并不重叠。这项检查很快就会变得计算开销极大。因此，我们将每个块在每个维度上划分为 k 个子块，以减少计算量。中央处理器将每个粒子插入与该粒子重叠的所有子块中，这与链式单元法类似。采用子块后，图形处理器在计算特定网格单元的重叠情况时，只需考虑一小部分粒子，即与该单元所在子块重叠的粒子。我们的流体-粒子耦合模拟需要传输各种数据。图 5 从负责同一数据块的中央处理器和图形处理器的视角，概述了不同类型的通信。我们将在下文说明通信步骤。对于每个粒子 i，粒子的位置 $\boldsymbol{x}_{\mathrm{p},\,i}$、半径 r_i 以及 $f(r_i)$（第 2.3.2 节）均通过 PCIe 从中央处理器传输到图形处理器。此外，中央处理器还会将每个子块中重叠粒子的数量及相应的粒子标识符传输到图形处理器。随后，图形处理器执行粒子映射。详情请参见第 2.3.2 节中关于固体体积分数计算（即粒子映射）的说明。在我们的模拟中，由于采用了几何分辨的球形颗粒，并设置了适当的离散元法参数以仅允许较小的粒子间穿透，因此一个给定网格单元最多只能与两个粒子重叠。因此，用于存储 $B_i(\boldsymbol{x},\ t)$ 的网格可以在每个网格单元中存储两个分数值。接下来，必须在中央处理器之间同步粒子的线速度和角速度 $\boldsymbol{U}_{\mathrm{p},\,i}$ 和 $\boldsymbol{\Omega}_{\mathrm{p},\,i}$，以确保每个中央处理器不仅拥有其局部粒子的正确速度，也拥有幽灵粒子的正确速度。随后，这些速度会从中央处理器传输到图形处理器，以便图形处理器计算每个单元中心处重叠粒子的速度（公式 (20)）。

3.1.2 流体模拟接下来执行 PSM 内核（内部）。这里的“内部”表示该内核会更新除最外层单元以外的所有单元。跳过最外层单元，可确保无需等待先前启动的 GPU-GPU 通信完成，即可调用此例程。PSM 内核的工作负载在整个模拟中最高，因此对性能至关重要。我们使用代码生成框架 Lbmpy（Bauer等，2021b；Hennig等，2023），生成高效且可扩展的 LBM CUDA 内核。Lbmpy 支持将任意 LBM 方法（例如 PSM）表示为符号形式，并生成经过优化的并行计算内核。我们将生成的计算内核集成到 WALBERLA 的模拟中。接下来，我们等待在时间步开始时启动的非阻塞 GPU-GPU 通信完成。根据可用硬件的不同，如果先前的计算完全掩盖了通信耗时，此时可能会立即返回。下一步是 LBM边界处理，它通过正确更新计算域边界处的流体单元，为流体模拟施加边界条件。由于此时已可获取相邻值，我们随后在 PSM 外部内核中更新最外层单元。图形处理器上的最后一步是从流体到颗粒的耦合。

3.1.3 从流体到颗粒的耦合 最后，图形处理器对流体施加在颗粒上的力和力矩进行归约

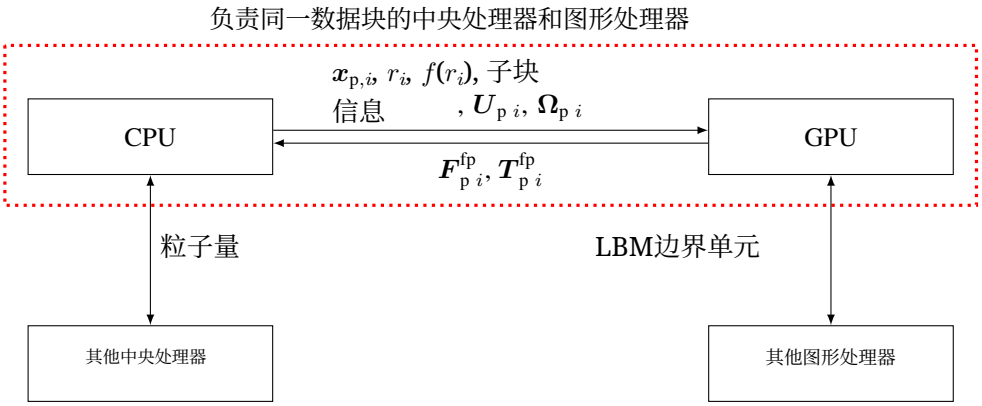


图5. 从负责同一数据块的中央处理器和图形处理器的角度，对不同通信步骤进行概述

到颗粒上 $F^{fp}{}_{p\,i}$ 和 $T_{fp\,p\,i}$（公式（21）和（22））。随后，$F^{fp}{}_{p\,i}$ 和 $T_{fp\,p\,i}$ 从图形处理器传输到中央处理器，以供接下来的中央处理器上的离散元模拟使用。单个粒子可能与位于多个区块上的单元重叠。因此，多个图形处理器可能会为同一粒子计算出 $F^{fp}{}_{p\,i}$ 和 $T^{fp}{}_{p\,i}$。因此，相应的中央处理器必须将耦合对颗粒施加的这些力和力矩贡献归约为单个变量（CPU-CPU 通信）。时间循环随后继续执行中央处理器上的粒子动力学。

3.2 中央处理器上的粒子动力学

在中央处理器上进行颗粒动力学（PD）模拟的第一步，是对速度进行力前积分，以更新粒子位置（式 (15)）。后续的粒子移动需要在中央处理器之间进行同步，以处理位置更新；位置更新可能会使粒子从一个块移至另一个块，从而由其他中央处理器负责这些粒子。通过遍历所有粒子对来计算颗粒间相互作用，很快就会因其 $O(n^2)$ 复杂度而变得非常耗费计算资源。因此，我们将粒子插入链表单元，使粒子对的遍历仅限于相邻的链表单元。链表单元的尺寸限制了确保颗粒间相互作用正确计算（碰撞、润滑修正）的最大距离。链表单元的尺寸为颗粒直径的 1.01 倍，接近仍能正确进行碰撞检测的最小尺寸。接下来，对彼此接近但尚未接触的所有粒子对应用润滑修正例程（第 2.3.1 节）。颗粒间相互作用采用离散元碰撞核（线性弹簧阻尼器）建模，该模型会对相互重叠的粒子施加力和力矩。碰撞核需要上一时间步的历史信息，即两个发生碰撞的粒子之间累积的切向运动（式 (14)）。由于不同进程可能处理过先前的碰撞，因此有必要在中央处理器之间归并碰撞历史信息，即在一个进程中收集给定粒子的所有碰撞历史信息。随后，将流体施加于粒子的流体动力和力矩，以及重力，加到总力上。由于不同进程可能已对给定粒子施加力和力矩，因此必须在一个进程中汇总这些贡献（另一次 CPU-CPU 通信）。然后，通过后积分更新颗粒速度（式 (16)）。此处可以省略通信，因为在即将到来的子循环进行后续通信之前不会用到这些速度。通常每个时间步会执行 j 个粒子子循环，因为为了准确表示接触，离散元法所需的时间分辨率高于格子玻尔兹曼方法（Rettinger 和 R¨ude，2022）。完成 j 个子循环后，下一个时间步从图形处理器上的流体动力学计算开始。

首先在图形处理器上运行流体动力学和耦合，然后在中央处理器上运行粒子动力学，这似乎是通过重叠中央处理器和图形处理器的计算来提升性能的一种可行方案。因此，本节余下部分将详细探讨这一可能性。下文将采用图4中对不同模拟模块的命名。

在数值误差不得增大的前提下，由于双向耦合所带来的依赖关系，中央处理器和图形处理器的计算只有部分能够重叠，其他部分则无法重叠。中央处理器和图形处理器部分的重叠有两种方式：在一个时间步内重叠，或在相邻时间步之间重叠。只有时间步 n 中中央处理器上的粒子模拟后力积分步骤依赖于时间步 n 中图形处理器上对流体动力的归约，因此，从力前积分到施加外力的步骤原则上可以与图形处理器部分重叠。图形处理器上的粒子映射

在时间步 $n+1$ 开始时，需要力前积分在时间步n更新后的粒子位置。因此，从将粒子插入链表单元到时间步 n的后力积分这些步骤，原则上可以与时间步 $n+1$ 中图形处理器上的粒子映射重叠。然而，粒子模拟通常包含多个子循环（即，图4中的中央处理器部分在每个时间步内执行多次），且只有第一个和最后一个子循环中相应的部分可以重叠。因此，重叠所能达到的最大加速比取决于子循环的数量。根据相关文献（Rettinger 和 R¨ude 2022），我们采用十个子循环；如果第一个和最后一个子循环能够完全重叠，理论上可使中央处理器上粒子模拟的运行时间减少 20%。然而，这一假设较为乐观，因为重叠打断连续的粒子子循环，可能会对相邻子循环之间缓存粒子数据的能力产生负面影响。

如果不考虑数值误差不得增大的要求，就可以使用先前时间步的结果，使图形处理器上的流体模拟与中央处理器上的粒子模拟完全并行运行。对于粒子移动可忽略不计的系统，这或许能带来可以接受的小误差，但这种做法并不普遍适用，而且需要进行详细的误差分析。

4 性能分析

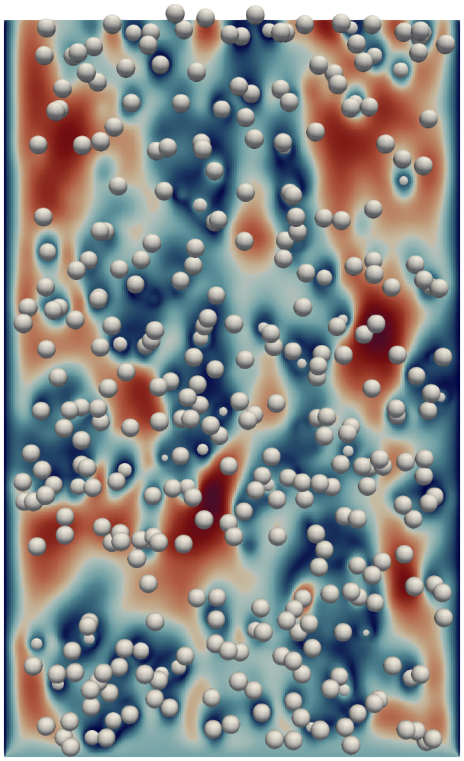
我们使用 Juwels Booster 集群进行性能评估。每个 GPGPU 节点由四块 Nvidia A100 40 GB GPU、两颗 AMD EPYC 7402 CPU（每颗芯片 24 个核心）和八个 NUMA 域组成。因此，每颗 CPU 由两块 GPU 共享，并划分为四个 NUMA 域，每个 NUMA 域包含六个核心。Juwels Booster 上的八个 NUMA 域中，有四个各自通过独立的 PCIe 通道连接到四块 GPU，从而确保 GPU 之间的 CPU-GPU 通信不会相互干扰。下文将由一块 GPU 及与之关联的（即通过 PCIe 通道直接连接的）包含六个核心的 NUMA 域构成的组合称为 CPU-GPU 对。节点内的所有 GPU 均通过 NVLink 互连，可直接进行 GPU-GPU 通信。对于不在同一节点上的 GPU 之间的通信，每块 GPU 都通过 PCIe 通道连接到专属的 Mellanox HDR200 InfiniBand ConnectX 6 适配器。SCALE 内核（即读写比为 1:1）在 A100 40 GB 上可实现约 1400 GB/s 的内存带宽（Ernst 等，2023）。我们采用每个直径 20 个网格单元来几何解析粒子（ Rettinger 等，2022；Rettinger 和 R¨ude，2022；Biegert 等，2017；Costa 等，2015）。接下来的章节将首先介绍模拟案例的计算特性，随后给出其性能结果。

4.1 模拟设置

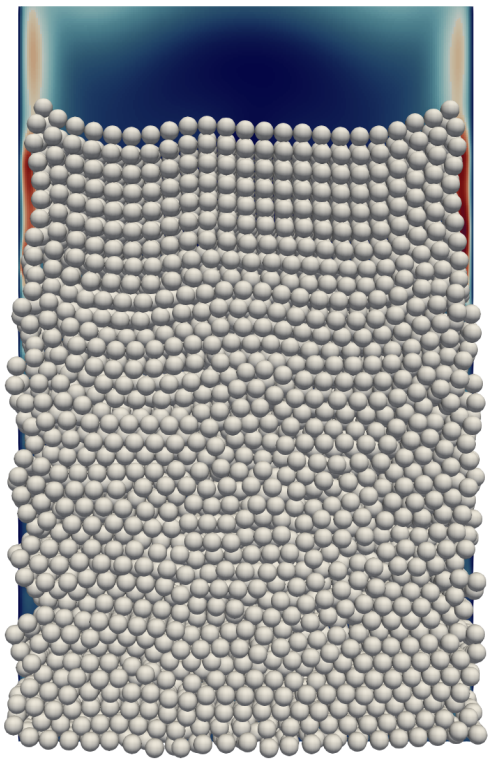
为研究本文介绍的混合耦合流体-颗粒实现的性能，我们采用流化床模拟。我们比较两种情形：稀疏情形和密集情形。两者在单位体积内的粒子数量以及颗粒间相互作用的数量方面各有不同。我们选择这两种情形，以考察中央处理器上的不同粒子工作负载如何影响混合 CPU-GPU 实现的整体性能。每个时间步使用十个粒子子循环（第 3 节）。我们使用 $500 \times 200 \times 800 = 80 \times 10^6$ 个流体网格单元对计算域进行离散。计算域底部的速度（入口）边界条件（BC）和顶部的压力（出口）边界条件控制流体动力学。其余四个边界为无滑移条件，确保流体速度为零；也就是说，计算域不是周期性的。颗粒雷诺数为 1.0，伽利略数约为 8.9，重力加速度为 9.81 $\mathrm{m/s^2}$, ，颗粒与流体密度比为 1.1。计算域周围的平面构成箱形壁面，防止粒子离开计算域。稀疏情形包含 627 个粒子。图 6a 展示了稀疏情形的设置。由于粒子浓度较低，稀疏情形中计算颗粒间相互作用（碰撞和润滑修正）所需的工作量较小。密集情形的设置总体上与稀疏情形相同，但粒子浓度显著提高，从而限制了流化（图 6b），粒子数量达到 8073 个，几乎是稀疏情形的 13 倍。

4.2 性能结果

本节将展示并分析性能结果。我们首先考察不同模拟模块各自的运行时间，以了解其中的瓶颈。此外，我们针对两种情况展示最多使用 1024 个 CPU-GPU 对的弱扩展基准测试结果。我们还展示了三种不同问题规模下，最多使用 1024 个 CPU-GPU 对的强扩展结果。最后，我们将混合实现与文献中的大规模纯中央处理器模拟进行比较，以展示混合实现的加速潜力。所有结果均取 500 个时间步的平均值。下文将参考第 1 节中提出的性能指标。



(a)稀疏情形



(b) 密集情形

图 6. 在一个 CPU-GPU 对上运行的整合式流化床设置的可视化。对于流体场，仅显示了一个二维切片。

4.2.1 不同模拟模块的运行时间我们考察不同模拟模块的运行时间，以分析混合实现引入的开销，即 CPU-GPU 通信，并评估

图形处理器的性能，并检测整体瓶颈。为了评估 PS M 内核在图形处理器上的性能，我们针对 LBM 内核采用屋顶线模型（Hager和Wellein 2010）。在利用最大内存带宽时，我们确定给定内核和硬件所能达到的最大性能（即性能“光速”估算）。PSM 内核由 LBM 内核以及额外的内存传输组成，传输量取决于重叠粒子的数量。由于这一数量因单元而异，PSM屋顶线模型并不容易直接建立。因此，我们仅分析 LBM 模型，同时牢记这是一种过于乐观的性能估算。由于我们使用 D3Q19格子模型，每个时间步都会读取和写入每个单元的 19 个概率分布函数。这相当于 19 次读取和 19 次写入（双精度），即更新一个格子单元需要 304 字节（Feichtinger等人 2015）。计算域包含 8e7 个流体单元。根据屋顶线模型，每个时间步的最短运行时间如下：

$$\begin{aligned} T_{\min} &= \frac{304\ \text{B/cell} \cdot 8\text{e}7\ \text{cells/time step}}{1400\ \text{GB/s}} \\ &= 17.4\ \text{ms/time step}. \end{aligned} \tag{28}$$

我们将总运行时间分为以下模块：PSM 内核（PSM）、CPU-GPU通信（comm）、粒子映射（mapping）、设置颗粒速度（setU）、对粒子施加的流体动力和力矩求和（redF），计算颗粒动力学，最后是其余任务（other），例如 LBM边界处理。图 7 给出了使用单个 CPU-GPU 对时，这些模块每个时间步的运行时间。此外，图中的水平虚线表示计算域中没有粒子时 PSM 内核的运行时间。虽然在物理行为上它与 LBM 内核完全相同，但它可以量化 PSM 与 LBM 内核不同的代码结构所带来的开销，同时无需承担计算域中存在粒子所增加的额外开销。在稀疏情形下，PSM 内核每个时间步所需时间比 LBM 的光速估算多约 42%；在密集情形下则多 83%。与不含粒子的 PSM 内核相比，稀疏情形每个时间步所需时间多约 4%，密集情形则多 34%。两种情形下，CPU-GPU通信都可以忽略不计。密集情形下所有模块的耗时都比稀疏情形更长。在稀疏情形下，PSM 内核占据了大部分运行时间；而在密集情形下，颗粒动力学模拟耗时超过了 PSM 内核。不过，密集情形下大部分运行时间仍花在 GPU例程上。

混合实现引入的开销

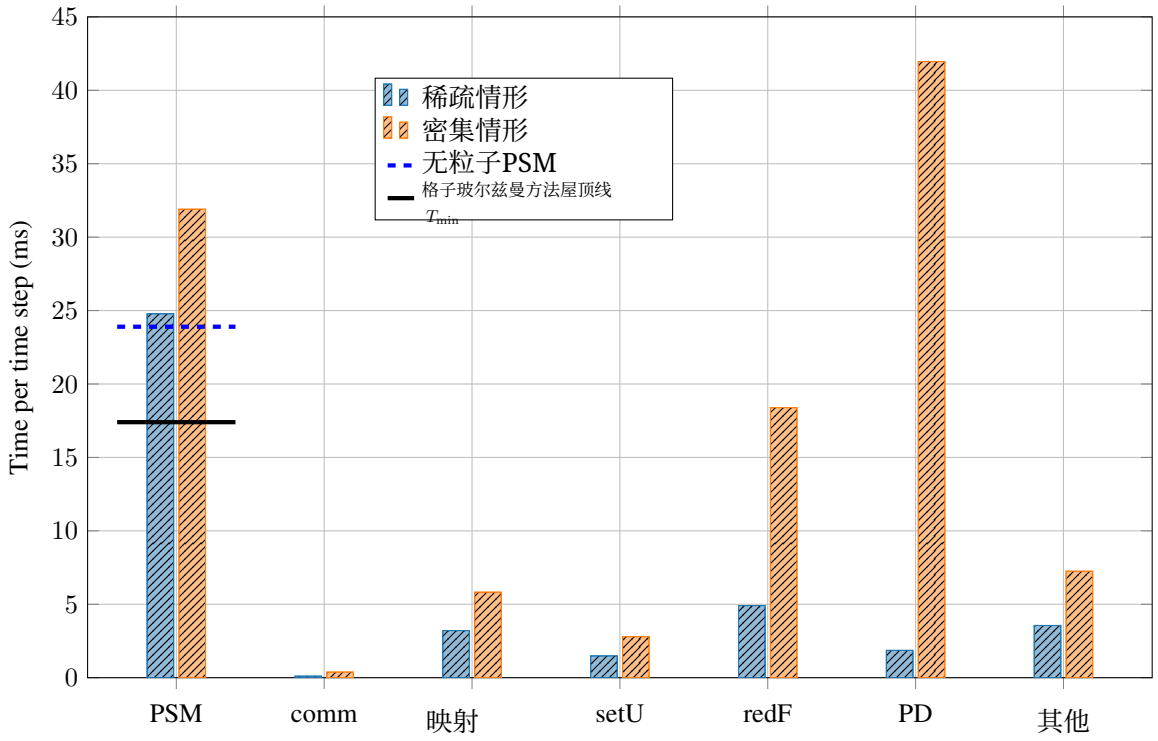


图7. CPU-GPU 对上各模拟模块的单独运行时间

可以忽略不计，因为我们每个粒子只传输少量双精度值，而不传输流体单元，这也得益于问题感知协同分区。因此，这种开销表明，采用所述技术的混合并行化是一种可行的方法，第一个标准已满足。PSM 内核的性能接近于充分利用 A100 的总内存带宽，尤其是考虑到给出的屋顶线模型并未计入 PSM 内核固体部分所产生的内存流量。因此，第二个标准已满足。与仅 CPU 实现相比，PSM 内核在图形处理器上的性能提升显著，因为该内核几乎充分利用了内存带宽，而中央处理器的内存带宽要低得多。即使在密集情形下，图形处理器运行时间也占总时间的大部分，因为流体和耦合工作负载远高于粒子工作负载，尽管我们使用了十次粒子子循环。因此，第三个标准已满足。PSM 内核呈现出两种值得注意的现象，我们将在下文使用 NVIDIA Nsight Compute[†]进行详细分析。

首先，图 7 中格子玻尔兹曼方法主存屋顶线（水平线）与无粒子 PSM 内核（虚线水平线）之间的显著差异表明，PSM 内核无法充分利用主存带宽，因为在没有粒子的情况下，PSM 内核的内存传输与格子玻尔兹曼方法主存屋顶线中所假设的传输非常相似。这一点值得注意，因为 PSM 是格子玻尔兹曼方法的扩展，而且众所周知

格子玻尔兹曼方法在 A100 架构上几乎可以充分利用主存带宽（Lehmann 等，2022；Holzer 等，2024）。PSM 内核每个线程使用 196 个寄存器。每个线程使用如此多的寄存器，严重限制了每个流式多处理器（SM）上的线程束数量，使 SM 的最大占用率仅为 12.50%。如此低的占用率不足以发出足够多的加载/存储指令来充分利用主存带宽，因此无粒子 PSM 与格子玻尔兹曼方法内存屋顶线之间存在差异。

其次，虽然在稀疏情形下，PSM 内核的运行速度仅略慢于不含粒子的情况，但在密集情形下，这种差异更加显著。稀疏情形与密集情形之间有几种不同的影响因素。首先，粒子数量增加会加重 GPU 工作负载中的不利因素，即线程束分歧和合并访问。与不含粒子的 PSM 内核相比，分支指令在稀疏情形下增加了 2.06%，在密集情形下增加了 18.18%。由于非合并全局访问而产生的额外扇区数量，从稀疏情形到密集情形增加了 78.49%，使其在密集情形下占总扇区数的 21%。此外，发射的指令数量在稀疏情形下增加了 4.32%，在密集情形下增加了 39.51%，其增幅与运行时间的增幅相近，这与

[†]https://developer.nvidia.com/nsight-compute

首个影响因素中提到的代码的指令受限性相对应（即流式多处理器的低占用率限制了发射的指令数量）。由于主存流量仅增加了 0.98%（稀疏情形）和 12.15%（密集情形），低于运行时间的增幅，主存利用率便从 65.85%（稀疏情形）降至 52.56%（密集情形）。

4.2.2 弱扩展为了进一步扩大模拟域，以模拟具有实际物理意义的情形，使用单个 CPU-GPU 对通常并不足够。因此，必须使用多个 CPU-GPU 对，甚至超级计算机的多个节点。因此，理想的弱扩展性能是我们所期望的。对于弱扩展，随着 CPU-GPU 对数量增加，问题规模也随之扩大，同时保持每个 CPU-GPU 对的工作负载不变。在实现完美弱扩展时，每个 CPU-GPU 对的性能保持不变，不受所用 CPU-GPU 对数量的影响。在格子玻尔兹曼方法中，每秒百万格点更新数是衡量弱扩展性能的标准指标（Holzer 等人，2021），表示硬件每秒执行的格子单元更新次数。我们使用计算每秒百万格点更新数所需的总运行时间，其中包括中央处理器（粒子）和图形处理器（流体、耦合）所耗费的时间。对于弱扩展图，我们至少进行了三次基准测试运行，并在下文中采用最佳结果。我们从单个 CPU-GPU 对和单个域块开始，如第4.1节所述。随后，我们逐步将 CPU-GPU 对的数量加倍，直至达到 1024。同时，我们在各个方向交替将域块数量/大小加倍： $2\times1\times1$ 块（两块图形处理器）、$2\times2\times1$ 块（四块图形处理器）、$2\times2\times2$（八块图形处理器）等。图 8 和图 9 展示了两种情形下的弱扩展结果。虚线表示理想曲线。据我们所知，这是文献中已发表的混合流体-粒子实现中规模最大的弱扩展测试。我们观察到，稀疏情形的性能约为密集情形的三倍。两种情形的并行效率都出现了下降，且初期尤为显著。使用 1024 个 CPU-GPU 对时，稀疏情形的并行效率为 71%，密集情形为 53%，对应的计算域大小为 $8000\times1600\times6400=8.192\times10^{10}$ 个流体单元。文献中针对其他混合实现（Kotsalos 等人，2021）以及仅使用中央处理器的流体-粒子实现（Rettinger 等人，2017）也观察到了类似的扩展行为。

要解读整体弱扩展行为，需要更详细地分析不同模拟模块的扩展性。使用单个 CPU-GPU 对时，占主导地位的例程是 PD、PSM 内核和耦合（即粒子映射、设置粒子速度，以及计算粒子所受流体动力和力矩的总和）。此外，我们还必须考虑使用多个 CPU-GPU 对所产生的通信开销。一方面是 PD 通信（CPU-CPU 通信），另一方面是 PSM 通信（GPU-GPU 通信）。图 10 和图 11 展示了两种情形下占主导地位的模拟模块的弱扩展行为。通信数据不仅涵盖通信步骤本身，也涵盖两个通信步骤之间的负载不平衡。比较两种情形时，不同模块呈现出相似的定性扩展行为。两种情形下，PSM 内核的扩展性都相当好。相应的 GPU-GPU 通信（PSM 通信）可以忽略不计。PD 的运行时间起初增加，随后趋于饱和。CPU-CPU 通信（PD 通信）大幅增加，在密集情形下超过 PSM 内核和耦合的运行时间。在密集情形下，PD 及其相应的通信对整体扩展性的影响比在稀疏情形下更大。耦合的扩展性与 PD 类似。

弱扩展过程中，每个图形处理器上的 PSM 工作负载保持不变，这解释了为何扩展性能近乎完美。由于我们隐藏了 PSM 通信（第 3 节），因此预计其影响可以忽略不计。由于相邻块数量增加，我们预计颗粒动力学和耦合的运行时间最初会有所增加。相邻块越多，每个块对应的幽灵粒子就越多，从而导致工作负载增加。当块在所有方向上都有相邻块时，这种影响会减弱，此后扩展性能几乎呈线性增长。文献中也曾报道过这一现象（Rettinger 等人（2017））。要实现物理上准确的模拟，该方法论要求每个时间步执行十个粒子子循环，每个子循环执行三个通信步骤。此外，除子循环外，每个时间步还需要执行两个 CPU-CPU 通信步骤（图4）。我们每个时间步需要执行 32 个 CPU-CPU 通信步骤，这些步骤无法隐藏在其他例程之后。因此，在这两种情况下，它都成为整体弱扩展性能下降的主导因素。

为了更深入地了解弱扩展中 PD 通信时间大幅增加的原因，并估算超过 1024 个 CPU-GPU 对时的扩展行为，我们针对密集情形对 PD 通信的数据传输进行了更详细的研究。PD 通信由两部分组成：进程间的粒子量同步，以及粒子物理量归约（第 3.2 节）。下文重点讨论占主导地位的部分——同步。图 12 绘出了

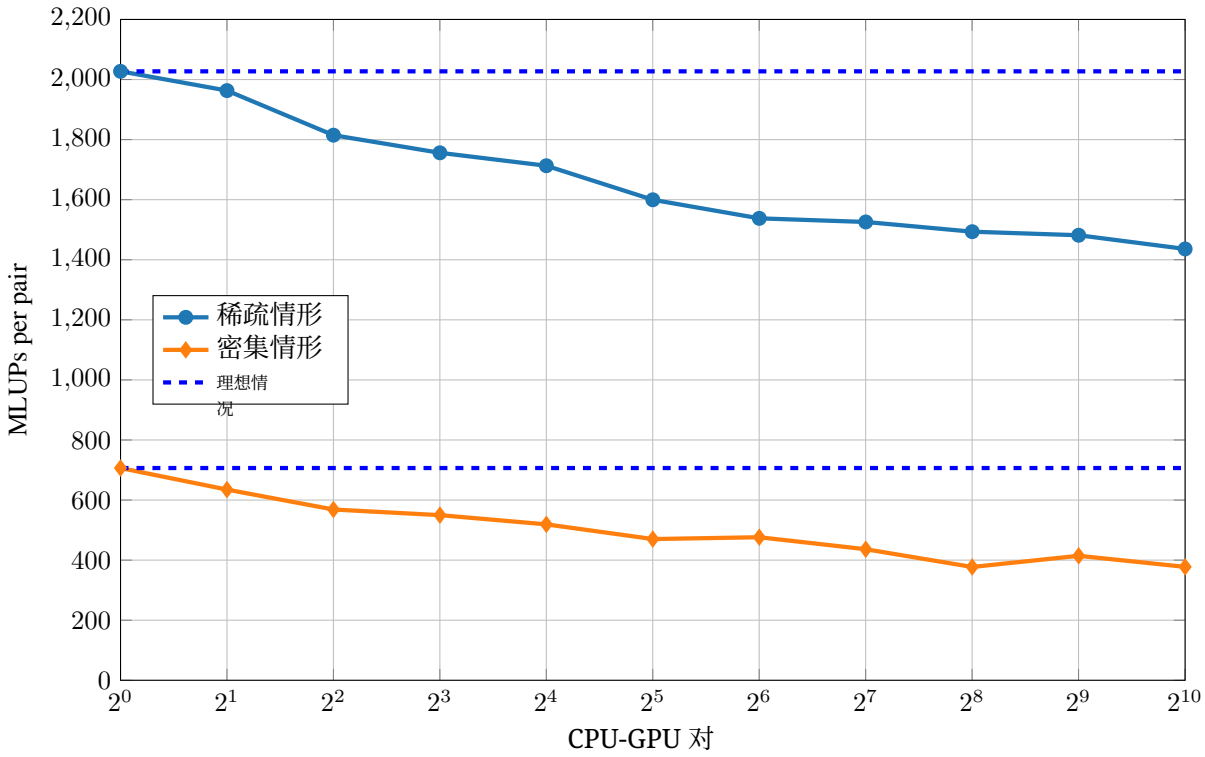


图 8. 两种情形下最多 1024 个 CPU-GPU 对的弱扩展性能

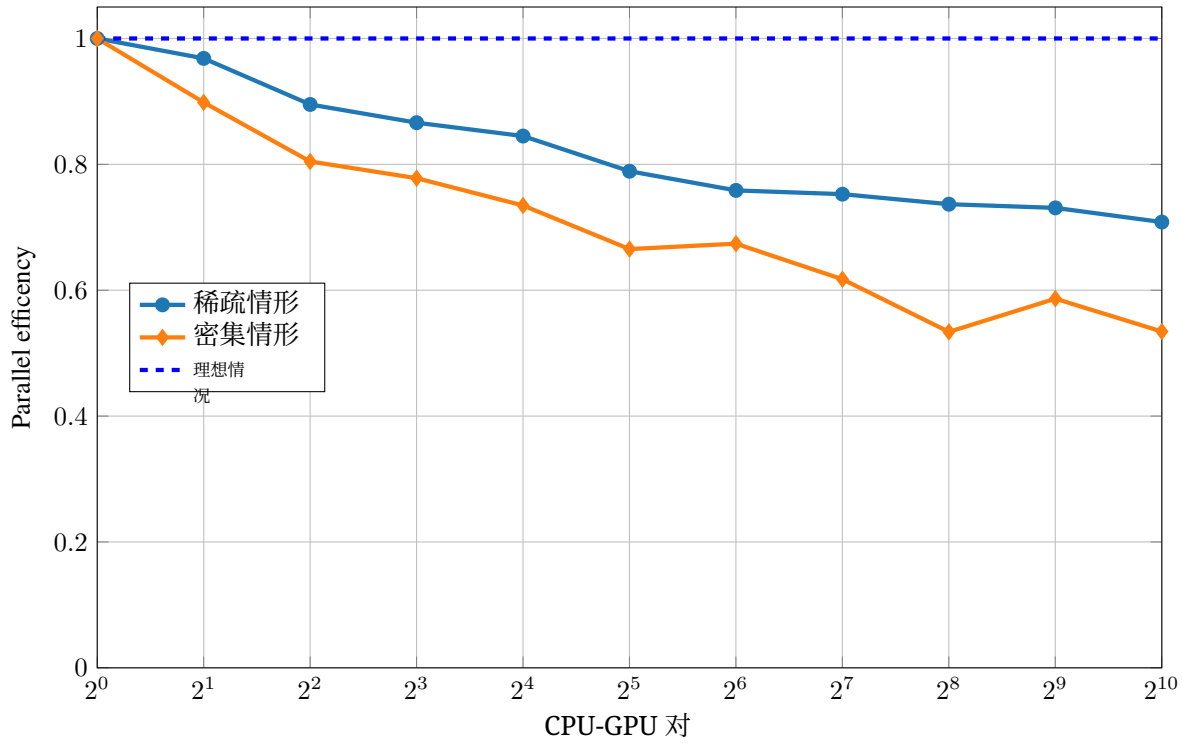


图 9. 两种情形下最多扩展至 1024 个 CPU-GPU 对的弱扩展并行效率

每次同步调用中，一个进程向相邻进程发送或从相邻进程接收的数据量的最大值和平均值。两个箭头标示了弱扩展将计算域沿 y 方向（即垂直于横截面的面外方向）加倍的步骤

图 6a）前两次。此外，该图还展示了每个进程的同步（即通信）伙伴数量的最大值。通信伙伴数量的最大值最初有所增加，从 2^6 个进程开始达到 26，随后保持不变。

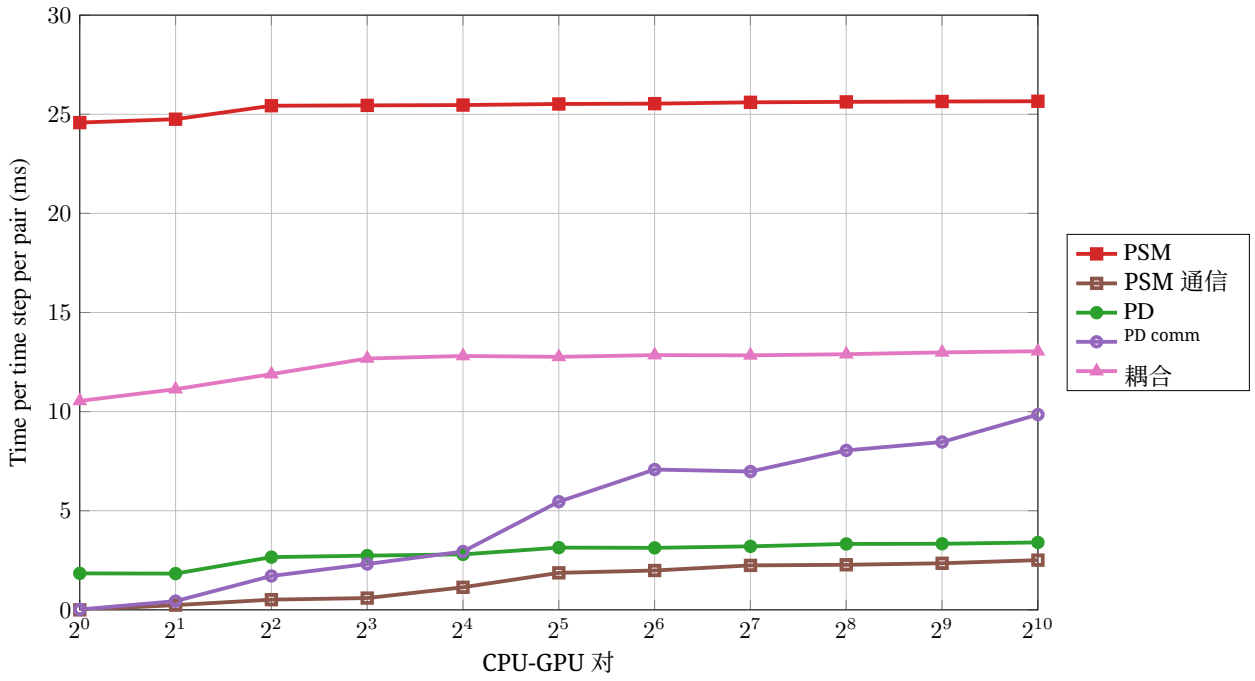


图 10. 稀疏情形下主导模块的弱扩展性能，最多可扩展至 1024 个 CPU-GPU 对

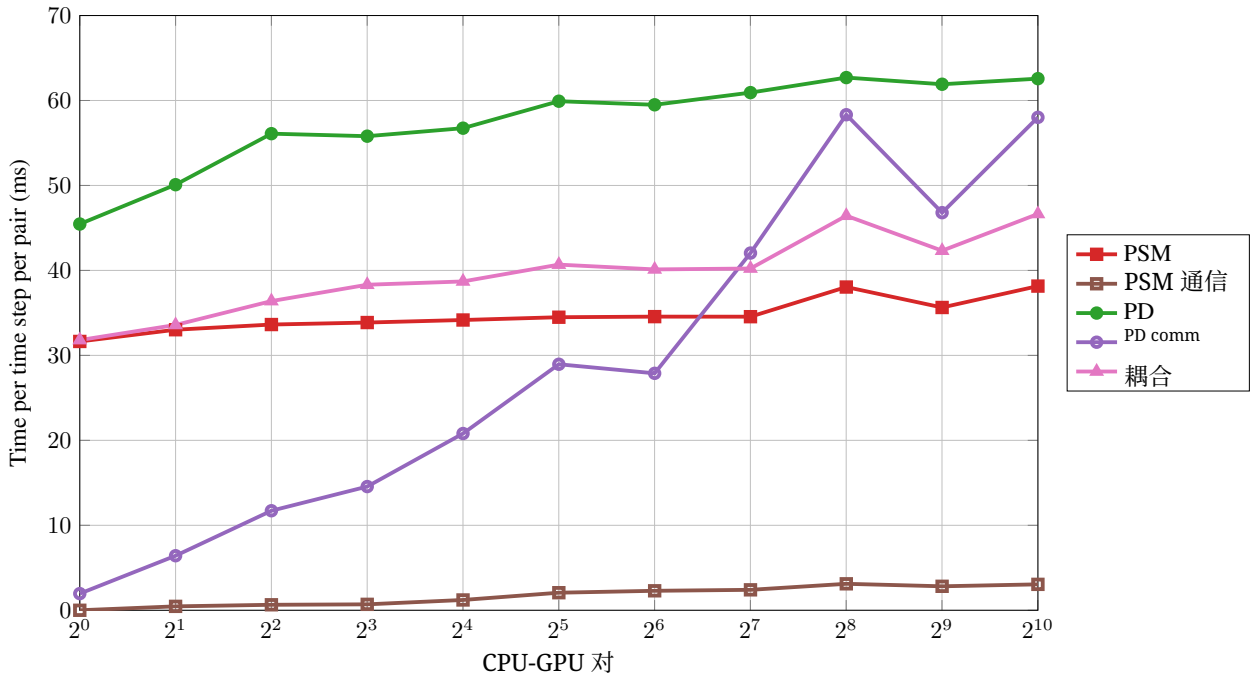


图 11. 密集情形下主导模块的弱扩展性能，最多扩展至 1024 个 CPU-GPU 对

每个进程向相邻进程发送或从相邻进程接收的数据量最大值增加至 2^5 个进程后趋于饱和。每个进程的最大数据量比通信伙伴数量最大值早一个进程就已达到饱和。每个进程的平均数据量则持续增加，直至 2^{10} 个进程。不过，从 2^0 到 2^5 个进程的增幅大于从 2^5 到 2^{10}个进程的增幅。对于最大值和平均值两条数据曲线，最显著的峰值出现在从 2^1 到 2^2 以及从 2^4 到 2^5 个进程，即弱扩展在 y 方向上将计算域大小翻倍的前两次。图 11 中，最大通信数据量和平均通信数据量彼此相关，也与 PD 通信时间相关。通信伙伴的最大数量上限为 26，因为在三维中，相邻块的最大数量为 26（包括仅共享角点的块）；这是因为一个包含 $3 \times 3 \times 3$ 个块的布局由一个中心块和 26 个相邻块组成。

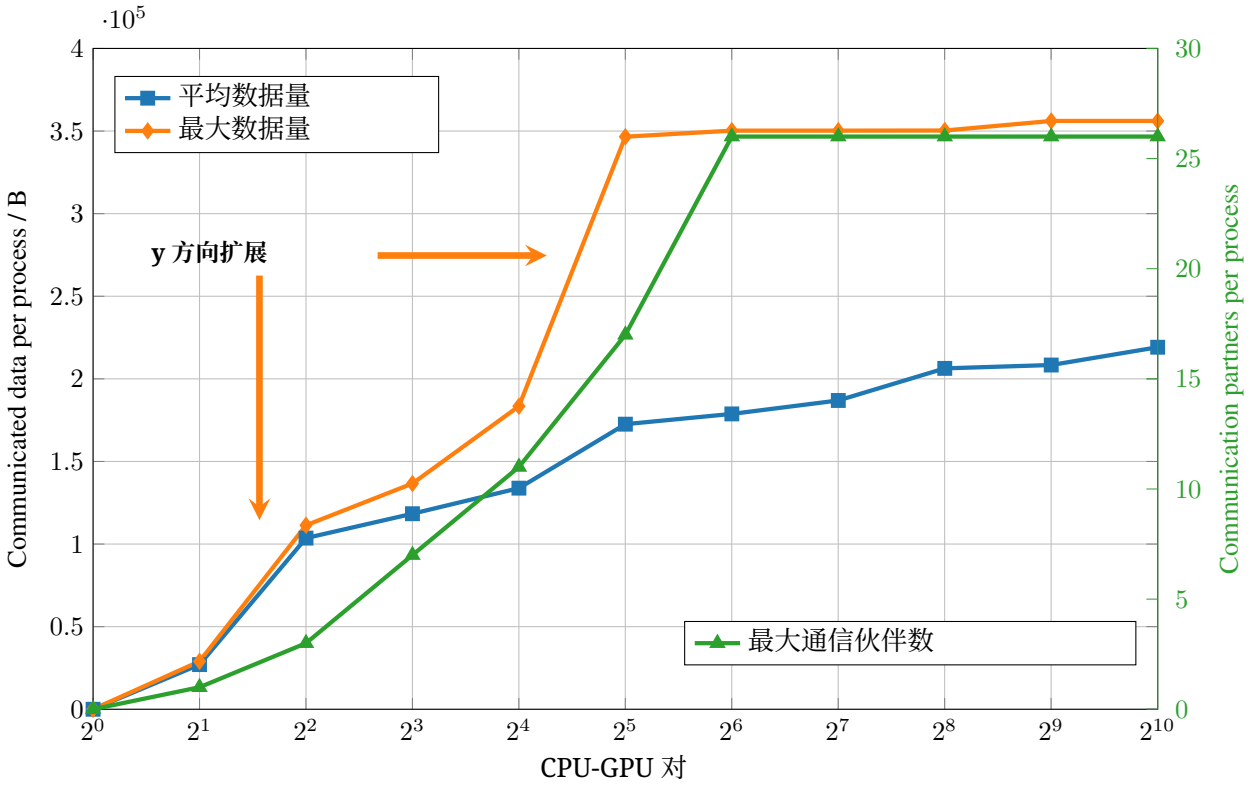


图 12。 密集情形下粒子同步的弱扩展性能，最多扩展至 1024 个 CPU-GPU 对。图中展示了每次同步调用期间，一个进程向相邻进程发送或从相邻进程接收的数据量（以字节为单位）的最大值和平均值（左侧 y 轴），以及同步伙伴的最大数量（右侧 y 轴）。

弱扩展首次在 2^6 个进程时出现这种情况，因为这对应于 $4 \times 4 \times 4$ 个块，因此至少有一个块完全被相邻块包围。弱扩展中最大数据传输量最初增加是意料之中的，因为最大邻居数量在增加。因此，每次同步都需要传输更多数据。较大的峰值对应于计算域沿 y 方向扩展的情况，这会使两个块之间的边界层增加最多，因此通信数据量也增加最多。从 2^4 到 2^5 的变化使通信伙伴的最大数量增加得比从 2^1 到 2^2的变化更显著。因此，最大数据量也相应地增加得更多（见图 12 中的两个箭头）。与最近邻通信的预期一致，当其他进程完全包围至少一个进程时，最大通信量会趋于饱和。然而，每个进程的最大数据量早在通信伙伴数量达到最大值之前，就已提前一个进程数达到饱和。这是因为从 2^5 到 2^6 个进程时，计算域第二次沿 z 方向扩展。不过，由于粒子的排列方式，同步只发生在 z 正方向。因此，沿 z 方向将计算域大小翻倍不需要额外通信，因为仍然，没有进程需要在两个 z 方向上进行通信。最大数据量达到饱和后，平均数据量仍会继续增加，这是因为边界块（通信伙伴少于 26 个）与区块总数的比例在下降，也就是说，相对而言，越来越多的区块成为具有最大数据量的区块。最大通信数据量为平均通信数据收敛提供了上界。由于采用纯最近邻通信，且每个进程的平均数据量趋于收敛，我们预计在超过 1024 个 CPU-GPU 对的更大型系统上仍将具有良好的可扩展性。然而，PD comm 的扩展性很大程度上取决于 PD 方法、其实现、可用硬件以及区块到硬件的分配。减少碰撞历史属于 PD comm 的一部分，其中包括交换旧的和新的接触信息。由于即使不使用多个 CPU-GPU 对也必须进行这种交换，因此即便只使用一个 CPU-GPU 对，PD comm 也大于零。总体而言，我们观察到了足以证明使用多个超级计算机节点合理的弱扩展性能。因此，第四项标准得以满足。

4.2.3 强扩展性 对于强扩展性，问题规模保持不变，同时增加 CPU-GPU 对的数量，从而减少每个 CPU-GPU 对的工作负载。在理想的强扩展情况下，随着 CPU-GPU 对数量增加，运行时间会以相同的

比例缩短。我们展示了三种问题规模（小、中、大）的强扩展结果。小规模问题包含 $500 \times 200 \times 800$ 个流体单元，如第4.1节所述，可在单个 CPU-GPU 对上运行。对于中等规模问题，小规模问题的尺寸在三个方向上均扩大一倍，得到 $1000 \times 400 \times 1600$ 个流体单元，可在八个 CPU-GPU 对上运行。对于大规模问题，中等规模问题的尺寸在三个方向上再次沿三个方向均扩大一倍，得到 $2000 \times 800 \times 3200$ 个流体单元，可在 64 个 CPU-GPU 对上运行。对于每种问题规模，在保持问题规模不变的同时，我们依次将 CPU-GPU 对的数量翻倍四次，方向依次为 x、y、z，再次为 x。图13和图14展示了三种不同问题规模下，稀疏情形和密集情形的强扩展结果。虚线表示理想曲线。对于这三种问题规模，稀疏情形和密集情形表现出相似的强扩展行为。与弱扩展不同，密集情形在强扩展中并未表现出整体并行效率较低的情况。所有情形下，CPU-GPU 对数量翻倍时，每个时间步的耗时都会缩短。不过，缩短幅度不到一半，因此偏离理想曲线。所有情形下，CPU-GPU 对数量连续翻倍两次时，相应的并行效率约为 60%；翻倍四次时，则降至约 40%。强扩展中的并行效率下降是意料之中的，因为最终问题规模会变得过小，无法有效利用计算资源，而通信开销却不断增加。与弱扩展分析（第4.2.2节）类似，强扩展中并行效率下降的主要原因是粒子属性频繁同步（PD comm）。文献中也报告了其他超级计算机上运行的其他 CFD 代码存在类似的强扩展行为，尤其是并行效率下降（Min等（2024）；Karp等（2023））。

4.2.4 混合实现的潜在加速比 我们预计，与仅使用中央处理器的代码 S_{hyb} 相比，混合实现的加速比可以根据阿姆达尔定律估算为

$$S_{\text{hyb}} \approx \frac{1}{1 + frac_{\text{acc}} \cdot (\frac{BW_{\text{CPU}}}{BW_{\text{GPU}}} - 1)}, \tag{29}$$

其中， BW_{CPU} 和 BW_{GPU} 分别是中央处理器和图形处理器内存的带宽。$frac_{\text{acc}}$ 是由混合实现加速的模块的纯CPU运行时间占比。在我们的案例中，该模块为PSM和耦合。我们假设$frac_{\text{acc}}$ 为

受内存带宽限制。我们将本文的混合实现性能结果与规模最大的多分散沉积床纯CPU模拟之一（Rettinger等（2022））进行比较。作者使用320个英特尔至强铂金8174中央处理器完成了后一项模拟。总计，他们在48小时内计算了 $2.25 \cdot 10^{15}$个格子单元更新，对应的性能约为每个中央处理器41每秒百万格点更新数；而在密集情形下使用1024个图形处理器时，每个中央处理器-图形处理器组合的性能为377每秒百万格点更新数。后一组数据表明，实测加速比约为9.2。我们测得英特尔至强铂金8174的内存带宽 BW_{CPU} 为70 GB/s。根据公式（29）估算的加速比约为10.3（假设$frac_{\text{acc}} = 0.95$ 且 $BW_{\text{GPU}} = 1400$ GB/s），与实测加速比相近。后一项计算仅是粗略估算，因为它忽略了中央处理器、网络、物理设置、数值参数等方面的差异所带来的影响。

5 启示与经验教训

我们得到的一条经验是，对于耦合模块之间的数据交换与模块内部计算相比明显偏向计算的耦合应用而言，CPU与图形处理器之间相对较少的数据传输可以变得微不足道，因此混合耦合是一种可行的方法。这一点对于本研究采用的应用尤为成立，因为只需交换数量少了好几个数量级的粒子数据，而无需交换图形处理器处理的整个流体场。因此，在密集情形下，CPU-GPU通信仅占总运行时间的0.35%。这些发现也适用于其他具有这种理想不平衡的应用，即计算量与耦合模块间数据传输量之间存在明显差异。其他耦合应用也可能呈现这种不平衡，例如复杂可变形几何体周围的大型流场，像细胞血流中的血细胞变形（第1节中有所提及）。由于表面数据是变形模拟的边界条件，因此只需在流体相和固体相之间交换这些表面数据。文献中发现，在以往的架构上，数据交换与计算之比很高的方法（即每个时间步都在中央处理器和图形处理器之间交换整个场）表现不佳；但随着硬件发展，这种情况未来可能会改变。中央处理器与图形处理器主存之间的高带宽内存传输是 NVIDIA GH200 Grace Hopper 等架构的发展趋势，这使混合实现有望适用于更多应用（甚至可在每个时间步传输全部图形处理器数据），值得在未来进一步研究。

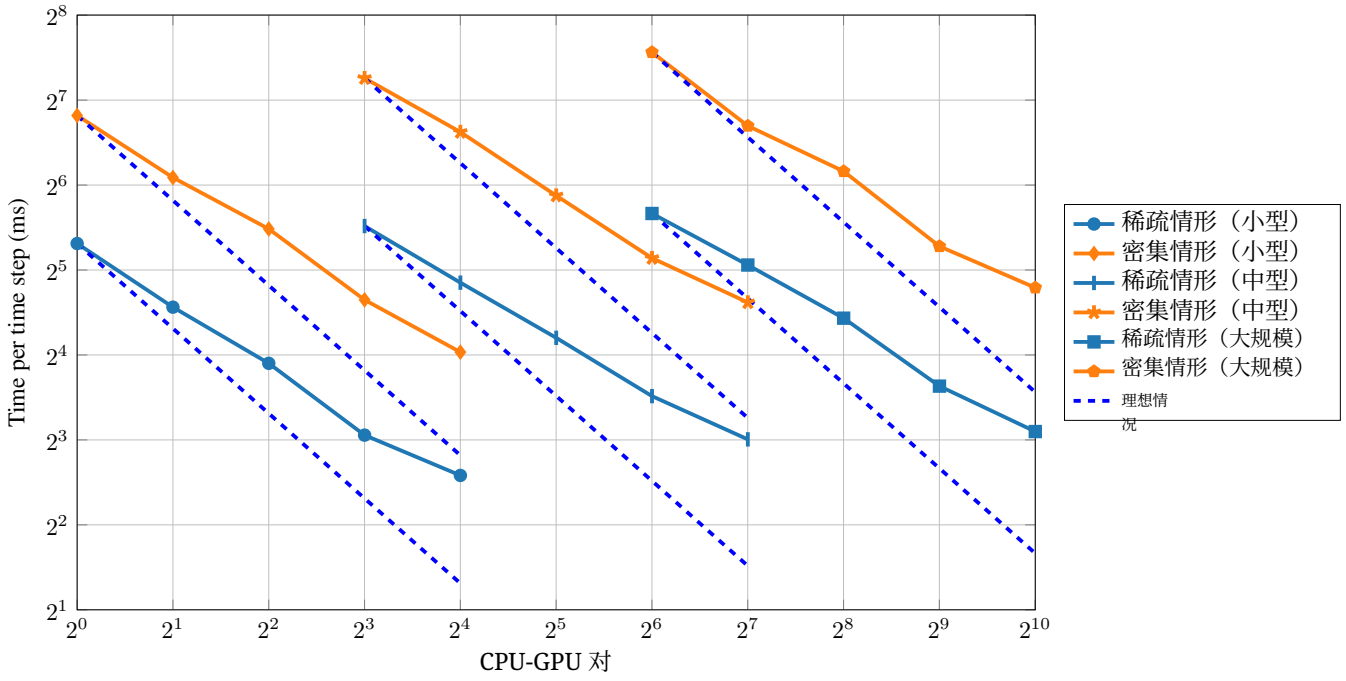


图 13. 两种情形下三种不同问题规模（最多使用 1024 个 CPU-GPU 对）的强扩展性能 irs

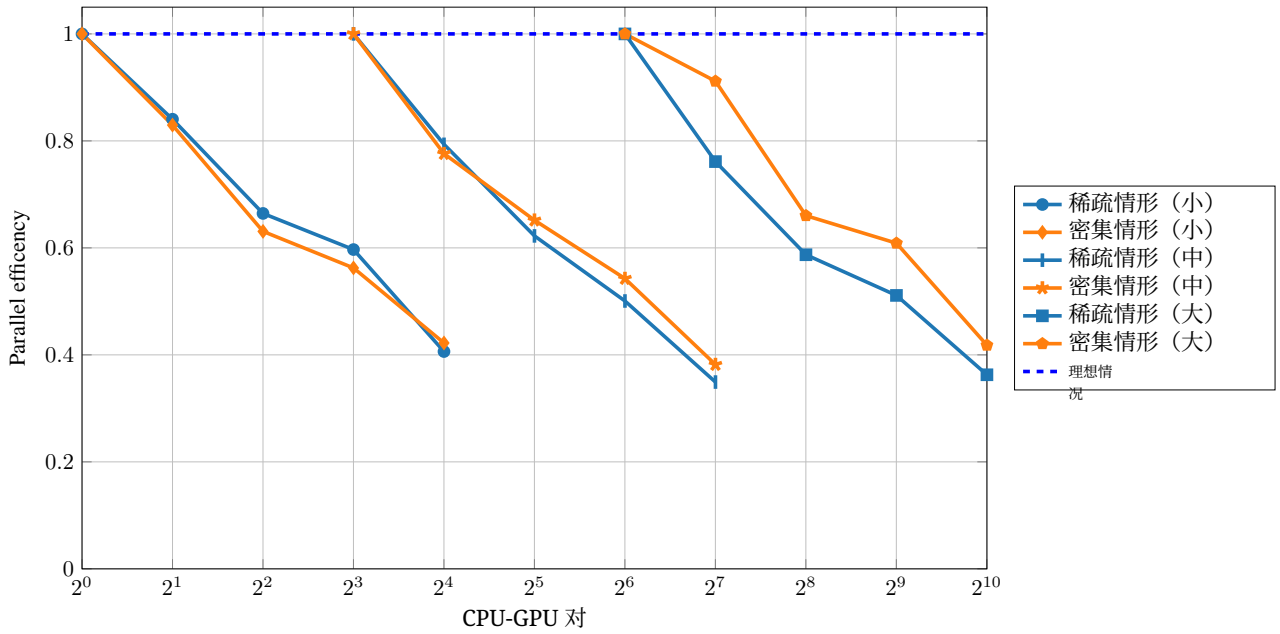


图14。 两种情况在三种不同问题规模下，最多扩展至 1024 个 CPU-GPU 对时的强扩展并行效率

代码生成（第3.1.2节）的优势是另一条经验。我们发现，代码生成会在实现阶段带来额外开销；只有在长期支持的框架中，当人们依赖高度优化的相似代码、对其他架构的可移植性，以及某种可扩展、可持续的方法时，它才有益。不过，由于采用了代码生成，代码能够完美运行

采用不同格子玻尔兹曼方法变体，在其他现代 GPU架构（如使用 HIP API 的 AMD MI250X）以及消费级图形处理器上进行研究。后续工作将开展系统的性能比较。

6 结论

在异构系统上，采用混合并行化是务实且因此颇具吸引力的做法，也就是说，不同的

模拟模块运行在不同的硬件上。然而，混合实现增加了实现良好性能和可扩展性的复杂性，尤其是在大规模系统上。本文研究了一种采用几何分辨颗粒的流体-颗粒混合耦合模拟。我们使用图形处理器进行流体动力学计算，而粒子模拟则在中央处理器上运行。我们报告并研究了该方法在两种流化床模拟情形下的性能，这两种情形的单位体积粒子数不同。混合实现所引入的开销（即 CPU-GPU 通信）可以忽略不计，因为我们只传输少量的粒子数据，而不传输流体单元。流体模拟的性能接近充分利用 A100 的全部内存带宽，这表明使用图形处理器进行流体模拟是不错的选择。在这两种情形下，GPU例程都占据了大部分运行时间。在一项弱扩展基准测试中，使用 1024 个 CPU-GPU 对时，混合流体-颗粒实现的并行效率在稀疏情形下达到 71%，在密集情形下达到 53%。当前的颗粒动力学方法要求每个时间步执行 32 个 CPU-CPU 通信步骤，这是整体并行效率下降的主要原因。我们的结果存在一定局限，因为粒子子循环次数、每个直径的流体网格单元数等因素不同，都会导致不同的性能结果。我们提出了混合实现必须满足的四项标准，以确保异构超级计算机得到负责任的使用。性能结果表明，我们的混合实现满足所有标准，因此适用于异构超级计算机上的大规模模拟。未来，我们计划更详细地研究颗粒通信步骤的瓶颈及优化可能性。我们采用子循环来提高刚性系统的稳定性。使用其他积分器或许能够采用更长的时间步，从而减少子循环带来的通信。我们展示了混合实现的加速潜力。因此，我们计划运行规模更大的流体-颗粒耦合模拟，以便更好地分析沉积床侵蚀等物理现象。

致谢

作者衷心感谢 Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) 通过提供 J¨ulich 超级计算中心 (JSC) 的 GCS超级计算机JUWELS 计算时间，为本项目提供资助。

作者衷心感谢埃尔朗根国家高性能计算中心（NHR@FAU）提供的科学支持和高性能计算资源。该中心隶属于弗里德里希-亚历山大大学埃尔朗根-纽伦堡分校 (FAU)。硬件由德国研究基金会 (DFG) 资助。

利益冲突声明

作者声明不存在利益冲突。

资金支持

作者披露，他们在本研究、本文撰写和/或发表过程中获得了以下资金支持：作者感谢德国研究联合会（DFG）为项目 433735254 提供资助。DFG 未直接参与本文工作；此外，本研究获得了欧洲高性能计算联合事业以及瑞典、德国、西班牙、希腊和丹麦根据资助协议第 101093393 号提供的资金。

数据可用性

数据可在 Zenodo 获取：DOI：10.5281/zenodo.13951599。

作者 ORCID

S. Kemmler，https://orcid.org/0000-0002-9631-7349；C. Rettinger，https://orcid.org/0000-0002-0605-3731；U. R¨ude，https://orcid.org/0000- 0001-8796-8599；P. Cu´ellar，https: //orcid.org/0000-0003-2446-8065；H. K¨ostler，https://orcid.org/0000-0002-6992-2690；

作者贡献

S. Kemmler：概念化、方法论、软件、验证、正式分析、调查、数据整理、撰写——初稿、可视化、项目管理；C. Rettinger：概念化、撰写——审阅与编辑；U. R¨ude：撰写——审阅与编辑；P. Cu´ellar：撰写——审阅与编辑；H. K¨ostler：资源、撰写——审阅与编辑、监督、经费获取；

参考文献

Bauer M, Eibl S, Godenschwager C, Kohl N, Kuron M, Rettinger C, Schornbaum F,Schwarzmeier C, Th¨onnes D,

K¨ostler H and R¨ude U（2021a）waLBerla：用于多物理场模拟的块结构高性能框架。《计算机与应用数学》81：478–501。DOI:10.1016/j.camwa.2020.01.007。Bauer M, K¨ostler H and R¨ude U（2021b）Lbmpy：用于高效并行格子玻尔兹曼方法的自动代码生成。《计算科学杂志》49：101269。DOI:10.1 016/j.jocs.2020.101269。Benseghier Z, Cu´ellar P, Luu LH, Bonelli S and Philippe P（2020）用于岩土工程和侵蚀问题微观力学分析的并行基于 GPU 的计算框架。《计算机与岩土工程》120：103404。DOI:10.1016/j.compgeo.2019.103404。Biegert E, Vowinckel B and Meiburg E（2017）用于致密、可移动多分散颗粒沉积床上方流动的颗粒分辨模拟的碰撞模型。《计算物理学杂志》340：105–127。DOI:10.1016/j. jcp.2017.03.035。Costa P, Boersma BJ, Westerweel J and Breugem WP（2015）含有限尺寸颗粒流动的完全解析模拟碰撞模型。《物理评论*E*》92(5)：053012。DOI:10.1103/PhysRevE.92.053012。Cundall PA and Strack ODL（1979）颗粒集合体的离散数值模型。*G´*岩土技术29(1)：47–65。DOI: 10.1680/geot.1979.29.1.47。Eibl S and R¨ude U（2019）颗粒动力学的模块化可扩展软件架构。Ernst D, Holzer M, Hager G, Knorr M and Wellein G（2023）现代 GPU 上代码生成过程中的分析性能估算。《并行与分布式计算杂志》173：152–167。DOI:10.1016/j.jpdc.2022.11.003。Feichtinger C（2012）异构超级计算机上多物理场模拟软件框架的设计与性能评估。博士论文。Feichtinger C, Habich J, K¨ostler H, R¨ude U and Aoki T（2015）CPU–GPU 集群上异构格子玻尔兹曼模拟的性能建模与分析。《并行计算》46：1–13。DOI:10.1016/j.parco.2014 .12.003。Fukumoto Y, Yang H, Hosoyamada T and Ohtsuka S（2021）无宏观假设条件下，嵌入式板桩周围饱和颗粒土渗透破坏的二维流固耦合数值分析。《计算机与岩土工程》136：104234。DOI:10.1016/j.compgeo.2021.104234。Hager G and Wellein G（2010）《科学家与工程师高性能计算导论》。第0版。CRC出版社。ISBN 978-1-4398-1193-1。DOI:10.1201/EBK1439811 924。

He X and Luo LS (1997) 不可压缩纳维–斯托克斯方程的格子玻尔兹曼模型。《统计物理学杂志》88(3/4): 927–944. DOI:10.1023/B:JOSS.000 0015179.12689.e4. He Y, Muller F, Hassanpour A and Bayly AE (2020) 一种适用于复杂颗粒-流体流动的 CPU-GPU 跨平台耦合 CFD-DEM 方法。化学工程科学 223: 115712. DOI:10.1016/j.ces.2020.115712. Hennig F, Holzer M and R¨ude U (2023) 多松弛时间格子玻尔兹曼方法的高级自动代码生成。*SIAM* 科学计算杂志 45(4): C233– C254. DOI:10.1137/22M1531348. Holzer M, Bauer M, K¨ostler H and R¨ude U (2021) 通过代码生成，在中央处理器和图形处理器上高效进行高密度比不混溶流体的格子玻尔兹曼多相模拟。《国际高性能计算应用期刊》35(4): 413–427. DOI:10.1177/10 943420211016525. Holzer M, Mitchell TR, Leonardi CR and R¨ude U (2024) 热毛细流动中心矩相场格子玻尔兹曼模型的开发：液滴捕获与计算性能。《计算物理学杂志》518: 113337. DOI:10.1016/j.jcp.2024.113337. Iglberger K and R¨ude U (2010) 非球形颗粒的大规模并行颗粒流模拟。计算机科学——研究与发展 25(1-2): 105–113. DOI:10.100 7/s00450-010-0114-4. Jajcevic D, Siegmann E, Radeke C and Khinast JG (2013) 流化颗粒系统的大规模 CFD–DEM 模拟。化学工程科学 98: 298–310. DOI:10.101 6/j.ces.2013.05.014. Jones BD and Williams JR (2017) 球体与立方体相交体积的快速精确计算。工程计算34(4): 1204–1216. DOI:10.1108/EC-02-2016-0052. Junior JRdS, Clua EW, Montenegro A and Pagliosa PA (2010) 在异构 GPU 和 CPU 环境中使用双向交互刚体进行流体模拟。载于：*2010* 年巴西游戏与数字娱乐研讨会。Florianpolis，圣卡塔琳娜，待定，巴西：IEEE。ISBN 978-1-61284-391-9，第 156–164 页。DOI:10.1109/SBGAMES.2010.25. Karp M, Massaro D, Jansson N, Hart A, Wahlgren J, Schlatter P and Markidis S (2023) 使用图形处理器和现代 Fortran 进行大规模湍流直接数值模拟。《国际高性能计算应用期刊》37(5): 487–502. DOI:10.1177/10943420231158616. Kidanemariam AG and Uhlmann M (2014) 水下沉积物中图案形成的直接数值模拟。《流体力学杂志》750: R2. DOI:10.1017/jf

m.2014.284. Kim J, Seo S, Lee J, Nah J, Jo G and Lee J (2012) SnuCL：一种面向异构 CPU/GPU 集群的 OpenCL 框架。载于：第*26*届*ACM*国际超级计算大会论文集。圣塞沃洛岛，威尼斯，意大利：ACM。ISBN 978-1-4503-1316-2，第341–352 页。DOI:10.1145/230457 6.2304623. Kotsalos C, Latt J, Beny J and Chopard B (2021) 用于研究血小板输运的大规模并行 CPU/GPU 系统中的数字血液。《界面聚焦》11(1): 20190116. DOI:10.1098/ rsfs.2019.0116. Kr¨uger T, Kusumaatmaja H, Kuzmin A, Shardt O, Silva G and Viggen EM (2017)格子玻尔兹曼方法：原理与实践。物理学研究生教材。查姆：施普林格国际出版。ISBN 978-3-319-44647-9 978-3-319- 44649-3。DOI:10.1007/978-3-319-44649-3. Kuznik F, Obrecht C, Rusaouen G and Roux JJ (2010) 使用 GPU 计算处理器进行基于LBM的流动模拟。《计算机与应用数学》59(7): 2380–2392. DOI: 10.1016/j.camwa.2009.08.052. Ladd AJC and Verberg R (2001) 颗粒-流体悬浮液的格子玻尔兹曼模拟。《统计物理学杂志》104(5): 1191–1251. DOI:10.1023/A:1010414013942. Lehmann M, Krause MJ, Amati G, Sega M, Harting J and Gekle S (2022) 采用64位、32位和定制的16位数值格式的格子玻尔兹曼方法的精度与性能。《物理评论*E*》106(1): 015308. DOI:10.110 3/PhysRevE.106.015308. Machado RRL, Schmitt J, Eibl S, Eitzinger J, Leißa R, Hack S, P´erard-Gayot A, Membarth R and K¨ostler H (2021) tinyMD：通过部分求值将分子动力学模拟映射到异构硬件。《计算科学杂志》54: 101425. DOI:10.1016/j.jocs.2021.10 1425. Min M, Brazell M, Tomboulides A, Churchfield M, Fischer P and Sprague M (2024) 迈向风能模拟的百亿亿次级计算。《国际高性能计算应用期刊》38(4): 337–355. DOI:10.1177/10 943420241252511. Ngoma J, Philippe P, Bonelli S, Radja¨ı F and Delenne JY (2018) 结合离散元法和格子玻尔兹曼方法，对颗粒介质中的烟囱式流化进行二维数值模拟。《物理评论*E*》97(5): 052902. DOI:10.1103/PhysRevE.97.052902. Noble DR and Torczynski JR (1998) 一种适用于部分饱和计算单元的格子玻尔兹曼方法。《国际现代物理学杂志*C*辑》09(08): 1189–1201. DOI: 10.1142/S0129183198001084. Norouzi H、Zarghami R 和 Mostoufi N（2017）用于流化床 CFD-DEM 模拟的新型混合 CPU-GPU 求解器。《粉末技术》316：233–244。DOI:10.1016/j.powt ec.2016.11.061。Oyarzun G、Borrell R、Gorobets A 和 Oliva A（2017）用于 CFD 模拟的可移植实现模型：在混合 CPU/GPU 超级计算机上的应用。《国际计算流体动力学杂志》31(9)：396–411。DOI:10.1 080/10618562.2017.1390084。Qian YH、D’Humi`eres D 和 Lallemand P（1992）纳维–斯托克斯方程的格子BGK模型。《欧洲物理快报》*(EPL)*17(6)：479–484。DOI:10.1209/0295-5075/17/6/001。Rettinger C（2023）采用格子玻尔兹曼法与离散元法并行耦合的颗粒流完全解析模拟。博士论文，弗里德里希-亚历山大埃尔朗根-纽伦堡大学。Rettinger C、Eibl S、R¨ude U 和 Vowinckel B（2022）层流剪切流中可移动沉积物床的流变特性：蠕变和多分散性的影响。《流体力学杂志》932：A1。DOI:10.1017/jfm.2021.870。Rettinger C、Godenschwager C、Eibl S、Preclik T、Schruff T、Frings R 和 R¨ude U（2017）河床中沙丘形成的完全解析模拟。载于：Kunkel JM、Yokota R、Balaji P 和 Keyes D（编）高性能计算，第10266卷。查姆：施普林格国际出版。ISBN 978-3-319- 58666-3 978-3-319-58667-0，第3–21页。DOI:10.1007/97 8-3-319-58667-0 1。_Rettinger C 和 R¨ude U（2017）完全解析格子玻尔兹曼模拟中流体-粒子耦合方法的比较研究。《计算机与流体》154：74–89。DOI:10.101 6/j.compfluid.2017.05.033。Rettinger C 和 R¨ude U（2022）一种用于含颗粒流完全解析模拟的高效四向耦合格子玻尔兹曼–离散元法。《计算物理学杂志》453：110942。DOI:10.1016/j.jcp.2022.110 942。Rohr D、Kalcher S、Bach M、Alaqeeliy AA、Alzaidy HM、Eschweiler D、Lindenstruth V、Alkhereyfy SB、Alharthiy A、Almubaraky A、Alqwaizy I 和 Bin Suliman R（2014）一台节能型多GPU超级计算机。载于：*2014 IEEE*国际高性能计算与通信会议、*2014 IEEE*第*6*届网络空间安全国际研讨会、*2014 IEEE*第*11*届嵌入式软件与系统国际会议（*HPCC,CSS,ICESS*）。巴黎：IEEE。ISBN 978-1-4799-6123-8，第42–45页。DOI:10.1109/HPCC.2014.14。Schwarzmeier C、Rettinger C、Kemmler S、Plewinski J、N´u˜nez- Gonz´alez F、K¨ostler H、R¨ude U 和 Vowinckel B（2023）

自由表面流中逆沙丘的粒子解析模拟。《流体力学杂志》961：R1。DOI:10.1017/jfm. 2023.262。

Sheikh B 和 Pak A（2015）采用耦合的三维离散元法和格子玻尔兹曼方法，数值研究孔隙率和曲折度对土壤渗透率的影响。《物理评论*E*》91(5): 053301. DOI: 10.1103/PhysRevE.91.053301.

Shimokawabe T、Endo T、Onodera N 和 Aoki T（2017）一种可在 GPU 超级计算机上实现超越设备内存容量的大规模计算的模板框架。载于：*2017 IEEE*国际集群计算会议（*CLUSTER*）。美国夏威夷州檀香山：IEEE。ISBN 978-1-5386- 2326-8，第 525–529 页。DOI:10.1109/CLUSTER.2017.9 7.

Sousani M、Hobbs AM、Anderson A 和 Wood R（2019）采用耦合的离散元法和计算流体力学进行加速传热模拟。《粉末技术》357: 367–376. DOI:10.1016/j. powtec.2019.08.095.

Vowinckel B、Kempe T 和 Fr¨ohlich J（2014）完全解析移动床湍流明渠流中的流体–颗粒相互作用。《水资源进展》72：32–44。DOI：10.1016/j.advwatres.2014.04.019。

Wang D、Leonardi CR 和 Aminossadati SM（2018）采用格子玻尔兹曼法和离散元法，改进颗粒悬浮液中时间积分与流体动力学相互作用的耦合。《计算机与应用数学》75(7): 2593–2606。DOI：10.1016/j.camw a.2018.01.002。

Xu M、Chen F、Liu X、Ge W 和 Li J（2012）采用多尺度中央处理器–图形处理器混合计算的气固两相流离散粒子模拟。《化学工程杂志》207–208：746–757。DOI: 10.1016/j.cej.2012 .07.049。

作者简介

Samuel Kemmler 是弗里德里希-亚历山大埃尔朗根-纽伦堡大学（FAU）系统仿真教席的博士生，同时也是柏林联邦材料研究与测试研究所（BAM）的研究助理。他拥有计算工程理学硕士学位，也是 WALBERLA 高性能计算框架的核心开发者之一。他的研究兴趣包括高性能计算和颗粒分辨沉积物输运模拟。

ChristophRettinger 曾在 FAU 学习计算工程。2023 年，他在系统仿真教席完成了关于颗粒流全分辨模拟的哲学博士研究。

Ulrich R¨ude 担任 FAU 系统仿真教席主任。他曾就读于慕尼黑工业大学（TUM）和佛罗里达州立大学，学习数学与计算机科学。他在 TUM 获得哲学博士和教授资格。他的研究重点是数值模拟和高端计算，尤其是计算流体动力学、多层方法，以及高性能计算软件工程。他是工业与应用数学学会会士。

Pablo Cu´ellar 曾在马德里理工大学学习土木工程，并于2011年在柏林工业大学获得土木工程哲学博士学位。他是 BAM 的客座科学家。

HaraldK¨ostler 于2008年获得计算机科学哲学博士学位，研究方向为医学图像处理中的变分模型和并行多重网格方法。2014年，他完成了关于高性能计算高效数值算法与软件工程的教授资格。目前，他在 FAU 系统仿真教席工作。他的研究兴趣包括软件工程理念，尤其是利用代码生成技术为高性能计算集群开发仿真软件、多重网格方法，以及并行硬件编程技术，特别是图形处理器。应用领域包括计算流体动力学、刚体动力学和医学成像。