pMSz：用于在有损压缩中校正极值与莫尔斯-斯梅尔分割的分布式并行算法

Yuxiao Li俄亥俄州立大学美国俄亥俄州哥伦布 li.14025@osu.edu

Mingze Xia俄勒冈州立大学科瓦利斯，俄勒冈州，美国 xiaml@oregonstate.edu

Xin Liang俄勒冈州立大学美国俄勒冈州科瓦利斯 li nuxin@oregonstate.edu

Bei Wang 犹他大学 美国犹他州盐湖城 beiwang@sci.utah.edu

Sheng Di阿贡国家实验室莱蒙特，美国伊利诺伊州 sdi1@anl.gov

Hemant Sharma阿贡国家实验室莱蒙特，伊利诺伊州，美国 hsharma@anl.gov

Dishant Beniwal阿贡国家实验室美国伊利诺伊州勒蒙特 dbeniwal@anl.gov

Franck Cappello 阿贡国家实验室 莱蒙特，IL，USA cappello@mcs.anl.gov

Robert Underwood阿贡国家实验室美国伊利诺伊州勒蒙特 runderwood @anl.gov

郭汉奇俄亥俄州立大学美国俄亥俄州哥伦布 guo.2154@osu.edu

arXiv:2601.01787v1 [cs.DC] 5 Jan 2026

摘要—**有损压缩被科学家广泛用于缩减来自模拟、实验和观测的数据，但即使误差有界，也可能扭曲感兴趣的特征。这种失真可能损害下游分析，并导致燃烧和宇宙学等应用得出错误的科学结论。本文提出一种用于校正拓扑特征的分布式并行算法，具体来说是分段线性 Morse–Smale 分割（PLMSS）。该分割将域划分为单调区域，并以相应的局部极小值和极大值标记这些区域。虽然已有一种单 GPU 算法（MSz）可在压缩后校正 PLMSS，但目前尚无能够扩展至单个 GPU 以外、适用于超大规模数据的方法。我们发现，PLMSS 校正扩展性的主要瓶颈在于积分路径的并行计算；这类计算通信密集，且众所周知难以扩展。我们的算法不显式计算和校正积分路径，而是通过在所有位置保留最陡上升和下降方向来简化 MSz，从而最大限度地减少进程间通信，同时仅引入可忽略的额外存储开销。借助这一简化算法和宽松同步，我们的方法在 Perlmutter 超级计算机上使用 128 个 GPU 处理真实世界数据集时，实现了超过 90% 的并行效率。**

索引词—**高性能计算，有损压缩**

一、引言

高性能计算 (HPC) 和科学仪器的快速发展，使宇宙学、燃烧和气候建模等领域的科学数据量呈爆炸式增长，给存储、输入/输出和分析流程带来了重大挑战 [1]。例如，使用 HACC 框架进行的万亿粒子宇宙学模拟，单次运行即可生成超过 20 PB 的原始数据 [2]。为此，压缩技术，尤其是能显著减小科学数据大小的有损压缩，已广泛应用于气候建模和宇宙学模拟等科学领域 [3],，这些领域通常会生成大规模数据集，因而需要高效的存储和传输。与通用压缩不同，科学应用通常要求在严格的误差界限内保持数据保真度，以确保下游分析的有效性。因此，误差有界有损压缩技术已得到广泛采用，能够在严格控制解压数据误差的同时实现高压缩比 [3]–[9]。为满足大规模模拟日益增长的数据需求，许多此类压缩器还进一步扩展了并行实现，以高效处理不断增长的数据规模 [5], [8], [10], [11]。

尽管许多误差有界有损压缩器已扩展出并行实现，以高效处理不断增长的数据规模 [5], [8], [10], [11], 但目前缺乏能够保留科学特征或拓扑结构的分布式并行算法，而这些特征和结构对于确保下游分析的正确性往往至关重要。在化学 [12], [13], 宇宙学 [14], 和燃烧 [15]–[16], 等应用中，分析的正确性不仅取决于数值精度，还取决于拓扑特征（例如极值点和莫尔斯–斯梅尔分割 [17]），这些特征能够帮助抽象和洞察数据的复杂结构。原始数据与解压缩数据之间拓扑特征的不一致（如图1所示）可能会影响下游分析，并导致错误的科学结论。例如，在高能衍射显微术（HEDM）分析中，

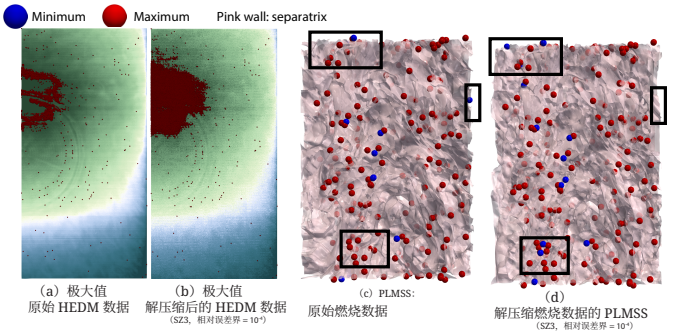


图 1：有损压缩（SZ3）对拓扑特征的影响。（a）和（b）：原始及解压缩后的高能衍射显微术（HEDM）数据集中局部极大值的位置（SZ3，相对误差界 10^{-4}）。解压缩后的数据中可以观察到极大值明显失真，这可能导致后续峰值分析不准确。（c）和（d）：原始及解压缩后的燃烧数据集的分段线性 Morse–Smale 分割（PLMSS）。黑框标示的分割失真可能会影响后续的流动和燃烧结构分析。

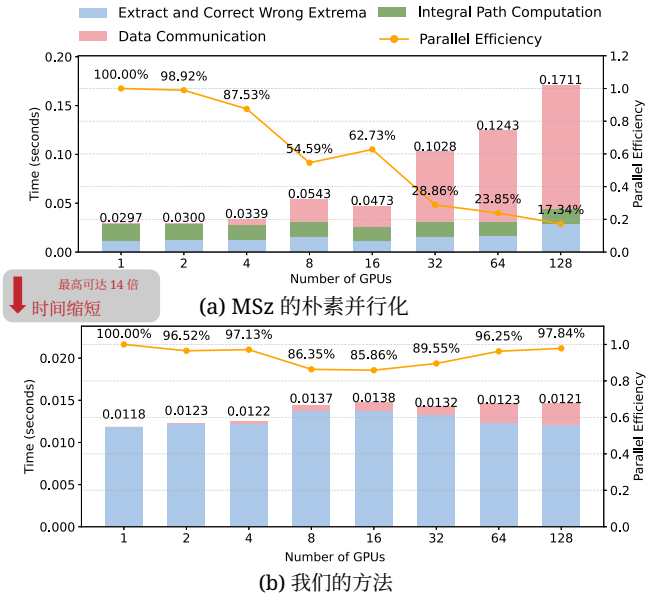


图 2：在 Perlin 噪声数据集上进行弱扩展时，MSz 的直接分布式并行化与我们提出的方法在并行效率方面的比较（每个 GPU 处理固定的 512^3 问题规模）。(a) 由于每次迭代中的积分路径计算引入了全局依赖，MSz 的直接并行化可扩展性较差，在使用 16 个 GPU 时效率低于 20%。(b) 我们的方法通过放宽充分条件并采用局部操作，消除了全局积分路径追踪，在使用 128 个 GPU 时实现了超过 90% 的并行效率。

数据中对应晶体材料晶粒取向和应变状态的峰值发生失真，可能导致晶粒识别错误，并误解内部应变分布，如图 1(a)和(b)所示；在燃烧分析中，失真的分割可能无法准确呈现火焰边界（如图1(c)和(d)所示），进而导致对燃烧区域的解读有误，以及对底层物理过程的评估不准确 [16]。

尽管已有若干特征保持方法被提出 [18]–[23]，现有方法由于依赖共享内存并行设计，可扩展性仍然有限。例如，MSz [18] 支持单 GPU 并行，以保持分段线性 Morse–Smale 分割（PLMSS），但由于依赖积分路径计算，无法在分布式内存系统上实现高效扩展。积分路径计算是指从每个数据点出发，沿最陡上升或下降方向追踪轨迹，直至到达对应的局部极值。在我们的实验中，如图 2(a) 所示，将 MSz 朴素并行化到分布式进程后，在弱扩展条件下使用 128 个 GPU 时，其并行效率降至 20% 以下。MSz 并行效率下降主要归因于并行积分线计算，这是一个众所周知的难以高效扩展的问题 [17]。具体而言，在每次迭代中，MSz 都会为每个数据点执行积分路径追踪，沿梯度方向上升或下降，直至收敛到局部极大值或极小值，从而重新计算 PLMSS；积分路径计算本身需要频繁通信和同步，难以实现高效并行化 [17]

并且最多占总计算时间的 60%（图 2(a) 中的绿色条）。每次迭代中积分

路径的计算还会导致穿越块边界的路径需要在相邻进程之间进行边界数据交换，以确保正确性，从而产生大量通信开销，这体现在图 2(a) 中不断增加的通信时间上。

本文显著提升了 MSz 的并行执行性能和可扩展性，尤其是在分布式内存环境中。为解决积分路径追踪中的可扩展性瓶颈，我们放宽了 PLMSS 保持的要求，以避免全局积分线追踪并降低数据通信频率。具体而言，我们将数据集划分为若干块，并在分布式内存环境中处理，每个处理器负责其分配到的块。在每次迭代中，算法检测并校正极值点处的失真，同时确保每个数据点保持一致的最陡上升或下降方向，这称为局部排序。此过程仅涉及各数据点及其直接相邻的邻居，不会引入全局数据依赖，从而能够高效并行执行，如图 2(b) 所示。此外，放宽 PLMSS 保持的要求使每个块能够独立进行校正，直至局部收敛。总之，本文的主要贡献如下：

- 我们提出了一种可扩展分布式迭代算法，用于

通过引入一种宽松的 PLMSS 保持充分条件来校正误差有界有损压缩中的 PLMSS，从而避免全局积分线追踪。

- 我们放宽了算法各次迭代之间的同步要求，进一步降低了通信开销。
- 我们展示了在弱扩展条件下使用 128 个 GPU 时，并行效率超过 90%；此外，我们还对多个科学数据集进行了全面评估，纠正了两种最先进压缩器（SZ3 和 ZFP）产生的拓扑错误。

II. 相关工作

我们回顾了科学数据压缩、拓扑保持有损压缩以及并行积分路径计算方面的相关工作。

A. 科学数据压缩

误差有界有损压缩 [3]–[11]， [24]–[27]在保证误差不超过用户指定界限的同时，实现了高压缩比。然而，大多数现有压缩器的一个关键局限是，它们只能保证逐点精度，却无法保留对科学分析至关重要的特征或拓扑结构。关于科学数据集误差有界有损压缩的全面综述，请参阅 Di 等人 [1]。

根据在量化和编码前对数据去相关的方式，误差有界有损压缩器可分为基于预测的方法和基于变换的方法。SZ 系列 [3]–[7], [10], [11], [24], [25] 是基于预测的压缩器中的代表性系列，它利用邻近信息预测数据值，对预测残差进行量化，并使用霍夫曼编码等技术编码残差，随后使用 ZSTD [28] 或 GZIP [29]等工具进行无损压缩。近期研究还探索了使用神经网络提高预测准确性，包括 AE-SZ [30] 和 SRNN-SZ [31]。与 SZ 系列不同，FPZIP [9] 通过忽略指定数量的位平面来减小数据大小，同时支持按需控制数据失真。MGARD [26] 是基于变换的有损压缩器的另一实例，它将小波理论应用于均匀/非均匀结构化网格和非结构化网格。ZFP [8] 是基于变换方法的代表性实例，它通过对块内数据应用定制的正交块变换来消除数据相关性，将数据变换为稀疏系数，然后对其进行高效压缩。

B. 拓扑保持有损压缩

拓扑保持有损压缩旨在压缩过程中保留数据的拓扑特征，而非仅仅确保逐点精度。临界点和 PLMSS 等拓扑描述子能够捕捉标量场的结构组织，并为复杂数据提供高层次的抽象，构成特征提取、分割和定量分析的基础，应用于燃烧、宇宙学和气候建模等多个科学领域 [32]。

尽管已有研究针对拓扑特征 [18]–[23], [33], 探索了保持拓扑的压缩，但现有方法通常仅适用于共享内存或单节点环境，因而限制了其在大规模数据集中的应用。例如，MSz [18] 仅支持单 GPU 执行，而 TopoSZ [19] 缺乏并行设计，因此限制了其在大规模数据集中的应用。

一种广泛采用的拓扑保持压缩范式是基于编辑的策略，该策略在解压缩后、原始数据和解压缩数据均在内存中可用时进行拓扑校正。基于编辑的策略会识别出一个数据点子集，在规定的误差界内对其进行修改，以恢复拓扑一致性。例如，MSz [18]通过基于编辑的策略，以迭代工作流校正 PLMSS 中的失真。我们的工作以基于编辑的策略为基础，并将其扩展到分布式内存环境，使每个数据块都能独立执行局部校正，同时保持全局拓扑一致性，具体内容将在后文介绍。

此外，还有其他特征保持策略，包括在压缩前修改输入数据 [34] ，以及将特征约束直接嵌入压缩算法中 [19]。尽管这些方法有效，但它们往往与特定压缩器紧密耦合，或需要全局数据依赖，从而限制了其可扩展性。

C. 并行积分路径计算

积分路径计算是科学可视化中的一项基本操作，用于追踪流线和梯度积分线等轨迹 [35]。积分路径是一条连续曲线，其切线方向遵循底层向量场或梯度场，从而揭示数据中的流动结构或特征连通性。在 PLMSS 中，积分路径通过沿梯度场从每个数据点追踪至对应的极值点，来定义分割边界 [17]。

然而，积分路径计算难以实现高效并行化，因为每条路径都具有很强的顺序依赖性和非局部依赖性：路径轨迹可能穿越多个数据分区，因而在分布式环境中需要进行不规则内存访问和频繁的进程间通信。我们的算法并未试图优化这项本质上难以处理的操作，而是通过放宽对 PLMSS 的保持要求，消除了积分路径计算的必要性。

现有的积分路径计算并行算法大致可分为两类：共享内存方法和分布式内存方法。共享内存方法利用单个节点内的线程级并行来加速路径追踪。Nouanesengsy 等人 [36],[37]以及 M¨uller 等人 [38]开发了动态任务调度和工作请求方案，以平衡粒子平流和有限时间李雅普诺夫指数计算过程中的不规则工作负载。Kendall 等人 [39] 以及 Peterka 等人 [35] 改进了大规模集合流可视化中的遍历和任务迁移策略。

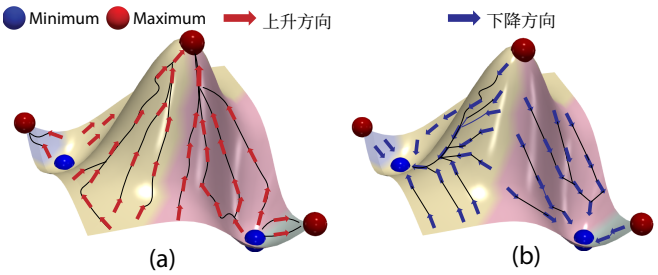


图 3：MS 分割示意图。红色和蓝色球体分别表示局部极大值和极小值。黑色路径表示上升和下降积分路径。

分布式内存方法将积分路径计算扩展到大规模环境，在这些环境中，数据被分配到各个进程中。Li 和 Fujishiro [40]以及 Garth 等人 [41] 通过空间分解和相干结构检测减少了进程间通信。Lu 等人 [42], 、Zhang 等人 [43], 和 Guo 等人 [44] 通过优化数据访问和重新分配提高了可扩展性。Morozov 和 Peterka [45]引入了用于块并行执行的 DIY2，而 Willetal. [46]则将积分线计算扩展到分布式系统上的大规模莫斯–斯梅尔分割。

III. 背景

我们回顾分段线性标量场中的莫斯–斯梅尔分割的背景，以及此前保留 PLMSS 的方法（MSz [18]）；该方法仅限于单个图形处理器。

A. 分段线性标量场中的莫斯–斯梅尔分割

我们简要回顾分段线性 Morse-Smale 分割（PLMSS） [17]。如图3所示，对于定义在数据点 i 上、取值互不相同 f_i 的分段线性函数（由规则的三角形／四面体网格表示），PLMSS 会将数据划分为梯度行为一致的不同区域：如以下所述，沿最陡上升和下降方向追踪时，同一区域内的任意点都会收敛到相同的极小值 m_j 和极大值 M_k 。

极值点是指其标量值高于或低于所有邻点的数据点，如图3中红色（极大值）和蓝色（极小值）球体所示。**积分线**是沿网格边的单调路径，从每个数据点出发，沿最陡上升或下降方向（图3中的红色或蓝色箭头）前进，直至到达极值点；图3(b)中的红色和蓝色路径展示了这一过程。**上升**分割和**下降分割**分别将收敛到同一极小值或极大值的数据点归为一组。**分段线性莫斯–斯梅尔分割（PLMSS）**结合上升分割和下降分割，对域进行划分。每个区域由一对极值点$\langle m_j$、$M_k\rangle$ 定义，并由积分线与其他区域隔开，如图3中不同颜色的区域所示。

B. PLMSS 单 *GPU* 校正的相关工作

此前的工作 MSz [18] 是一种单 GPU 算法，通过迭代修正解压缩数据中的拓扑失真，在误差有界有损压缩中保留 PLMSS。然而，MSz 依赖显式的积分路径计算，这会引入强数据依赖，并限制其在分布式内存环境中的可扩展性。

我们的算法与 MSz 的一个关键区别在于，我们的方法省去了显式的积分路径计算，而这正是 MSz 的主要可扩展性瓶颈，因此提高了效率，尤其是在分布式内存环境中。为更好地说明这一差异，我们先简要回顾 MSz 的设计。

MSz 的输入是一个二维或三维标量场 f 及其经任意具有全局误差界 ξ 的有损压缩器压缩后得到的解压版本 f 。该算法输出一组编辑 $\delta = \{\delta_i\}$ ，将其应用于 f ，以获得校正后的场 $g = f + \delta$ ，使其保留 PLMSS，并且所有数据点均严格满足全局误差界 $|f_i - g_i| \le \xi$ 。

MSz 的核心思想是基于编辑的策略：识别解压缩数据中的一部分数据点，在用户指定的误差界内修改其值，以纠正拓扑失真。在有误差界的有损压缩中，拓扑失真主要是由于压缩误差扰乱了相邻数据点之间的标量排序而产生的，而标量排序是底层标量场拓扑的基础。基于编辑的策略通过在误差界内调整选定的数据值，恢复必要的排序，从而纠正失真的拓扑。

具体而言，基于编辑的策略会迭代检测拓扑失真，识别导致失真的数据点 i ，并应用编辑 δ_i 以恢复拓扑一致性。设 f_i 表示 i 的原始值，$g_i^{(k)}$ 表示第 k 次迭代时的修改值，ξ 表示用户指定的误差界。每次编辑都会根据以下方程更新 i 的标量值：

$$g_i^{(k+1)} = \begin{cases} g_i^{(k)} - \delta_i, & g_i^{(k)} - \delta_i \ge f_i - \xi, \\ f_i - \xi, & \text{otherwise.} \end{cases} \tag{1}$$

MSz 包含两个基于编辑策略构建的、用于在有误差界的有损压缩数据中保留 PLMSS 的替代性迭代子循环：C 循环和 R 循环，具体如下所述。

C 循环致力于保留解压数据中的局部极值，因为局部极值是 PLMSS 的基本组成部分。MSz 会校正 C 循环中的四种失真：假阳性极大值（FPmax）、假阴性极大值（FNmax）、假阳性极小值（FPmin）和假阴性极小值（FNmin）。我们以 FPmax 为例，说明用于识别要修改的数据点 i 的规则。

具体而言，如果存在至少一个相邻点 j ，其标量值高于 f_i，则数据点 i 会被视为 FPmax；在原始数据 f 中存在这样的相邻点（如图4 (a)中标量值为0.35的球体），但

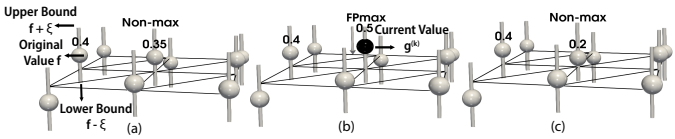


图 4：假阳性极大值校正过程示意图。每个球体的垂直位置表示其在当前迭代中的标量值。黑色球体表示失真，白色球体表示非极值点，圆柱体表示每个数据点的值域 $[f-\xi, f+\xi]$。

ˆ 在解压缩数据 f中不存在这样的相邻点，导致 i 被错误地视为极大值，如图4 (b)中的黑色球体所示。在这种情况下，我们将 g_i 减小 δ_{edit}，以恢复正确的顺序，如图4所示。对于其余情况，类似校正策略的详情请参阅 MSz [18]。

R 循环。 在保留所有极值点后，MSz 进入 R 循环，该循环旨在通过反复追踪积分线来定位并编辑问题点 （积分线上首次出现差异的位置），从而校正标记错误的数据点。R 循环的计算开销极大，而且难以高效并行化，因为每次迭代都需要重新计算积分路径，因此成为 MSz 的主要瓶颈。在分布式内存环境中，例如我们对 MSz 的朴素并行化方案，由于每条积分路径都可能穿越多个数据块，需要通过进程间通信访问路径追踪所需的数据，因此成本会进一步增加。相比之下，我们的算法完全消除了对 R 循环的需求，无需执行任何积分路径计算即可确保 PLMSS 一致性。

迭代策略的收敛性。 请注意，迭代过程中每次修改都可能引入新的失真，因此需要在后续迭代中进一步校正。尽管如此，仍可保证收敛。其关键原因在于，对数据点 i 应用的每次编辑都会使其值单调地向下界 $f_i-\xi \quad f_i = g_i^{(0)} \geq g_i^{(1)} \geq$ 减小，即 $g_i^{(2)} \geq \cdots \geq g_i^{(k)} > f_i - \xi$。因此，没有数据点会被无限次更新。在最坏情况下，每个数据点i 都会收敛到其下界 $f_i - \xi$，此时，保留的极值点和局部排序与原始场保持一致，从而确保校正后的场 g 的 PLMSS 与 f的 PLMSS 相同，进而保证迭代过程收敛。关于收敛性的详细证明，请参阅 MSz [18]。

MSz 的可扩展性局限。 虽然 MSz 能够在单个图形处理器上有效保留 PLMSS，但其设计在扩展到分布式内存环境时会带来显著的可扩展性挑战，主要有两个原因。

首先，R 循环需要在每次迭代中重新计算积分路径，从而引入强烈的全局数据依赖。每条积分路径都可能遍历域中的大片区域并跨越多个进程边界，导致路径追踪期间频繁进行进程间通信。其次，每次迭代还需要额外的数据通信，以便在相邻块之间同步更新后的积分路径结果，进一步增加总体开销。我们的方法通过移除 R 循环并重新设计 C 循环来解决这些局限性，具体内容将在下一节讨论。

IV. 我们的方法

为了解决 MSz 的可扩展性局限， [18],我们的算法移除了计算开销巨大的 R 循环，并重新设计 C 循环，以确保完全局部的校正能够在数据块之间高效同步。具体而言，我们（1）通过消除积分路径追踪引入的全局依赖、移除 R 循环，并修改 C 循环以通过纯局部操作保留 PLMSS，简化了 MSz；（2）通过分块处理、幽灵层同步以及降低进程间通信频率，实现了可扩展的分布式并行化。

本节其余部分将从两个方面介绍我们用于在误差界约束下的有损压缩中保留 PLMSS 的分布式算法：（1） MSz 的简化（第IV-A节）；（2）确保可扩展性和高效通信的分布式并行设计（第IV-B节）；以及（3）分布式内存环境中迭代校正过程的收敛分析（第IV-C节）。

本文中使用的符号如下： ξ表示用户定义的误差界， i 表示第 i个数据点。原始数据和解压缩数据中 i处的标量值分别表示为 f_i 和 f_i，而 g_i 表示 i处编辑后的标量值。

A. MSz 的简化：移除计算密集型的 *R* 循环

我们的算法完全消除了 R 循环，转而为 PLMSS 保持施加一个宽松的充分条件。具体而言，我们不再显式追踪积分路径以确定正确的分割，而是确保：（1）所有极值点都得到保持；（2）每个数据点与其最小邻点和最大邻点之间的局部排序与原始场保持一致。PLMSS 保持的这些宽松条件足以保证每个数据点沿着相同的梯度方向朝向其对应的极值点（即图3中的箭头），从而无需计算积分路径即可保持 PLMSS。

虽然保持所有数据点的局部排序足以维持相同的积分路径，但与 MSz 相比，这一宽松条件可能会引入更多编辑。然而，正如我们的评估所示，额外编辑的数量非常少（如图9所示）；与此同时，这种宽松的表述通过消除 R 循环并支持完全局部的校正，带来了显著的性能提升。

具体而言，对于极值点的保留，我们遵循 MSz 的 C 循环中的相同规则 [18], ，确保所有局部极小值和极大值都与原始数据保持一致。对于局部排序的保留，我们重新设计了 MSz 的 C 循环规则，以确保每个数据点与其邻域之间的局部排序一致：每个数据点

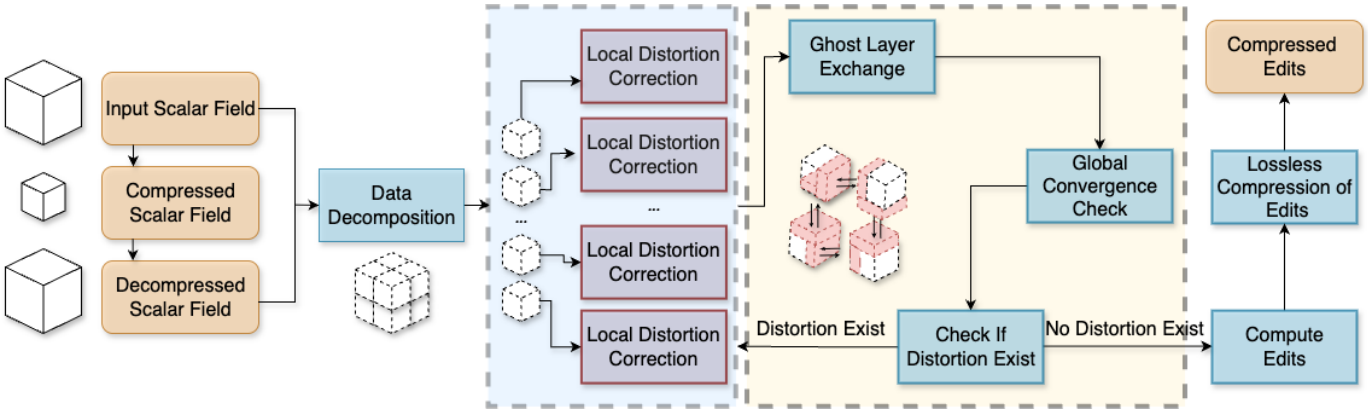


图 5：迭代工作流概览。输入是误差有界有损压缩器生成的解压缩数据，输出是一组编辑。将场划分为带幽灵层的块，每次迭代都执行局部失真校正、幽灵层同步和收敛检查。

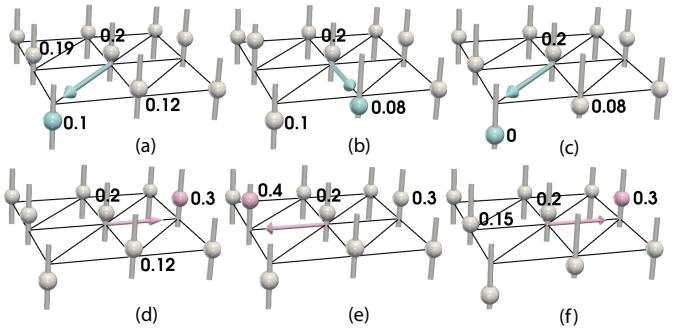


图 6：局部排序校正过程示意图。(a–c) 展示下降方向的校正步骤，(d–f) 展示上升方向的校正步骤。蓝色和粉色箭头分别表示下降方向和上升方向，蓝色／粉色球体表示每次迭代中中心球体的最小／最大邻点。

i，我们要求每个数据点最快的上升方向和下降方向都与原始数据 f 保持一致，具体讨论如下。

对于每个数据点 i，令 $n_{\min}(i)$ 和 $n_{\max}(i)$ 分别表示其在原始数据 f 中的最小邻点和最大邻点，即 $n_{\min}(i) = \arg\min_{j\in\mathcal{N}(i)} f_j$ 和 $n_{\max}(i) = \arg\max_{j\in\mathcal{N}(i)} f_j$。如图 6 所示，蓝色和粉色球体分别表示 $n_{\min}(i)$ 和 $n_{\max}(i)$。在解压缩数据 f 中，相应的邻点记为 $\hat{n}_{\min}(i)$ 和 $\hat{n}_{\max}(i)$。

下降方向。在 f（图 6(a)）中，i 的最小邻点是值为 0.1 的蓝色球体，即 $n_{\min}(i)$ 有 $f_{n_{\min}(i)} = 0.1$。压缩后（图 6(b)），局部排序遭到扭曲：g 中另一个邻点变成了最小邻点，其值为 0.08，因此 $\hat{n}_{\min}(i) \neq n_{\min}(i)$。由于我们的编辑是非递增的，无法增大值为 0.08 的数据点；相反，我们降低原始极小值处的值，使其再次成为所有邻点中的最小值，同时满足下界。我们将 $g_{n_{\min}(i)}$ 从 0.1 降至 0（图 6(c)），从而恢复 i 周围正确的下降关系（沿下降方向的邻点再次是原来的 $n_{\min}(i)$）。**上升方向**。在 f（图 6(d)）中，值为 0.3 的粉色球体是 i 的最大邻点（即 $n_{\max}(i)$）。压缩后（图 6(e)），另一个邻点伪增大至 0.4，因此 $\hat{n}_{\max}(i) \neq n_{\max}(i)$。此处我们可以直接降低伪最大值：将 $g_{\hat{n}_{\max}(i)}$ 从 0.4 降至 0.3（或在容差范围内略低一些），得到图 6(f) 所示的排序。例如，沿上升方向，数值依次为 $0.15 \to 0.2 \to 0.3$，与原始 f 一致。

在这两种情况下，编辑都是单调的（非递增的），且受局部容差约束，从而确保不会引入新的极值点，同时 i 周围的局部排序与原始场一致。

B. 分布式并行算法

为将我们的方法扩展到大规模数据集，我们设计了一种分布式并行算法，在每个数据块内独立执行局部校正，同时保持块边界之间的一致性。其核心思想是在每个数据块内独立执行局部校正，并通过块边界之间的轻量级通信交换幽灵层值，从而确保全局一致性。我们的分布式算法由三个主要部分组成：域分解与局部校正，以及幽灵层同步。**域分解与局部校正**。我们将数据域划分为互不重叠的区域，并为每个区域扩展一个单层幽灵区域，以交换边界信息；每个区域分配给一个图形处理器，如图 7 所示，其中红色边框的点表示幽灵区域。每个块按照前文介绍的宽松保持条件独立执行局部校正，确保块内的所有极值点和局部排序得到保持。

幽灵层同步。每个块完成局部校正后，相邻块会交换幽灵层

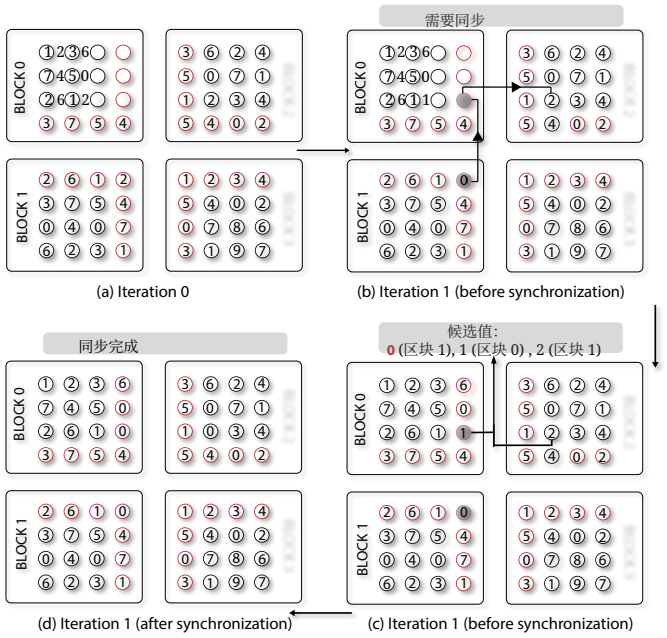


图 7：每个数据块都与一个带红色边框的幽灵层相关联，该幽灵层提供跨块边界的邻域信息。灰色点表示在局部校正过程中已被修改的数据值。

以确保共享边界上的标量值一致。当多个块以不同值更新同一个边界点 i时，通过从所有候选值中确定性地选取最小值来解决冲突，从而保证更新单调不增并趋向误差下界 $f_i - \xi$。如图 7 所示，块 0 和块 1 都修改了同一个边界点（图 7 (b) 中的灰色点），分别将其值改为 1 和 2，而块 2 保留原始值 0。在同步过程中，如图 7 (d) 所示，通过选取最小值 0 来协调这三个候选值（0、1、2），从而确保相邻块之间的更新一致且单调。

宽松同步。 尽管幽灵层同步能够确保块边界处的正确性，但其频率会直接影响整体性能。有两种同步策略可供选择：（1）逐次迭代同步：在每次局部迭代后立即更新幽灵数据点，确保数据即时一致，但会导致频繁通信，因为每次迭代都需要进行同步阶段，如图 8 (b) 所示；（2）仅在块内的局部校正收敛后执行同步，如图 8 (c) 所示。我们采用第二种策略，因为核心数据点的校正仅限于局部块内，只有边界点依赖相邻数据。允许每个块在同步前达到局部收敛，可以避免不必要的通信，同时确保块接口处的正确性。我们的宽松同步带来了一种权衡：由于边界更新有所延迟，可能需要更多次局部校正迭代才能达到全局收敛。然而在实践中，幽灵层同步的频率大幅降低，如图 8 所示，从而显著提升了性能。

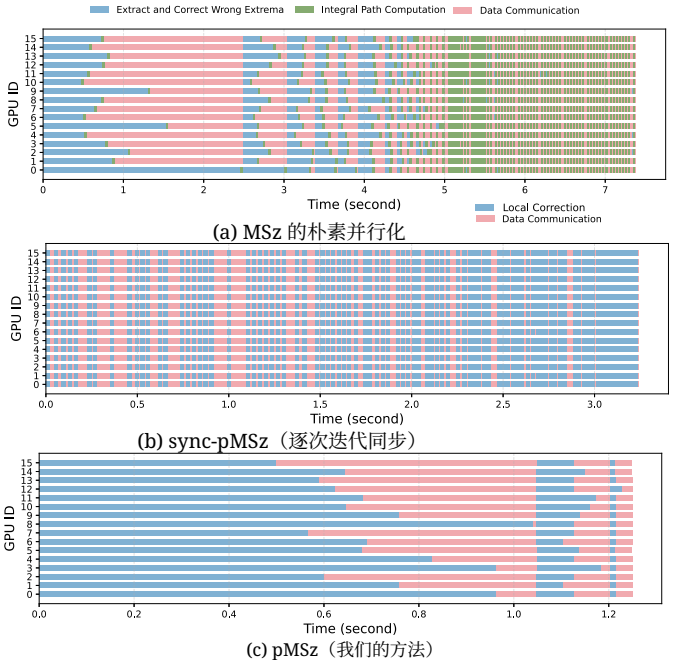


图 8：比较（a）MSz 的朴素并行化、（b）Sync-pMSz（采用我们的方法进行逐次迭代同步的基线方法）和（c）pMSz（我们的方法）在各次迭代中的计算（蓝色和绿色）以及通信/等待（粉色）阶段。

C. 收敛性分析

我们的算法在局部校正和幽灵层同步之间交替进行，直到所有数据块都满足 PLMSS 保持的放宽条件。正如第 III-B 节所述，每个块内局部校正的收敛性已经得到证明；由于在全局误差界 $|f_i - f_i| \leq \xi$ 下局部应用相同的编辑规则，我们的方法继承了这一性质。然而，在局部校正收敛后，相邻块在共享边界点上仍可能具有不同的标量值，如图 7 所示。边界点上的不一致可能会在块边界附近引入新的局部失真，但借助我们的递减编辑策略，这些失真将在有限次迭代后消失。在幽灵层同步步骤中，每个共享的幽灵数据点取其相邻块提供的所有候选值中的最小值，从而确保跨块更新保持非递增，并使所有值都处于全局误差界之内。由于同步只会降低值且绝不会违反误差界，因此我们的算法保证收敛。

因此，局部校正和同步所产生的更新均单调不增，且受 $f_i - \xi$限制。从理论上讲，在最极端的情况下，所有数据点

（包括幽灵层中的数据点）都达到各自的下界，此时局部排序便完全恢复。因此，分布式迭代过程会收敛到一个稳定场g，该场同时满足全局误差界和 PLMSS 保持的宽松充分条件。

五、评估

本节从三个方面评估我们的方法：（1）消除 MSz 中 R 循环的算法简化的有效性及其权衡；（2）我们所采用的分布式并行设计的可扩展性和性能；（3）我们的方法保持 PLMSS 的能力。

实验凸显了三个主要发现 gs:

- **可扩展性**。我们的方法在弱扩展条件下可扩展至 128 个 GPU，并行效率超过 90%；相比之下，MSz 的朴素并行化仅达到 8.66 %。
- **单 GPU 性能**。与 MSz 相比，我们的方法通过消除 R 循环并重新表述保持条件以实现完全局部校正，在单个 GPU 上最高可实现 $14\times$ 的加速比。
- **压缩有效性**。在经过测试的数据集上，我们的方法在保持 100% PLMSS 的同时，压缩率最高可达 $88\times$；而基础压缩器只能在无损模式下或压缩率极低时保持 PLMSS。

A. 实验设置

所有实验均在 NERSC 的 Perlmutter 系统上进行。每个计算节点配备 4 个 NVIDIA A100 GPU（每个 GPU 的内存为 40 GB）。我们的实现采用 CUDA 和 CUDA 感知 MPI 开发，可直接使用 GPU 内存进行数据通信。在可扩展性研究中，我们采用弱扩展，最多使用 128 个 GPU，每个进程处理固定的 512^3 子域。我们先概述实验中使用的基线方法，再介绍我们的方法和评估的细节。**基线方法 1：单 GPU MSz**。该基线方法使用在单个 GPU 上运行的原始 MSz [18]，作为评估我们进行算法简化（特别是移除 R 循环）的参照。**基线方法 2：naive-MSz**。为评估可扩展性的改进，我们通过将域划分到多个 GPU 上，实现了 MSz 的分布式内存版本。每个 GPU 都独立执行原始 MSz 算法，并在每次 R 循环迭代后交换幽灵区域。**基线方法 3：sync-pMSz**。该基线方法采用我们的方法，但在相邻块之间进行同步，且同步发生在每次迭代后

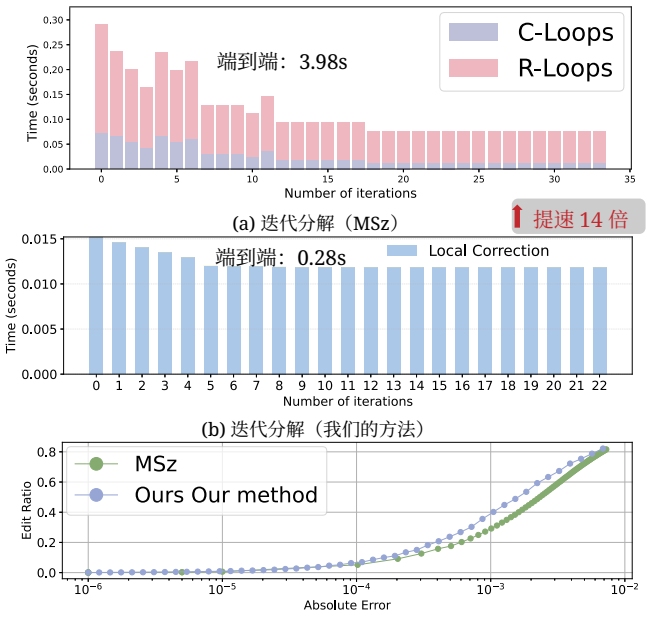
并用于分析我们采用的宽松通信策略中同步频率的影响。

对于单 GPU 评估，我们使用 NYX 数据集，将简化后的公式与原始 MSz 进行比较。对于可扩展性研究，我们使用全局规模逐渐增大的合成 Perlin 噪声数据集，以确保不同尺度下的数据特征一致。为了评估 PLMSS 保持性，我们使用来自宇宙学、燃烧、分子动力学和气候模拟的所有真实世界数据集，以测试我们的方法在压缩

有效性、开销和拓扑正确性方面的表现。

B. 单图形处理器评估：简化的有效性

本节评估我们所采用的算法简化的有效性：在 NYX 数据集上使用 SZ3 进行误差有界有损压缩时，移除 MSz 中的 R 循环，同时保留 PLMSS，其中相对误差界为 10^{-6}。单图形处理器实验将 pMSz（我们的方法）与原始 MSz 进行比较，以展示移除全局积分路径计算带来的益处。



(a) 迭代分解（MSz）

(b) 迭代分解（我们的方法）

(c) 编辑比例对比

图9：单个图形处理器上 (a) MSz 和 (b) 我们的方法各次迭代的耗时，以及 (c) MSz 与我们的方法的编辑比例（修改的数据点数占总数据点数的比例）对比。

如图9(b)所示，我们的方法相较于 MSz 的加速比最高可达 $14\times$，但需要修改的数据点数量略有增加，如图9(c)所示。性能提升主要源于移除了 R 循环；由于 MSz 需要反复进行全局积分路径计算来纠正错误标记的数据点，R 循环占用了 MSz 运行时间的主要部分。如图9(a)所示，在 MSz 后期的迭代中，R 循环占总运行时间的 80% 以上，而 C 循环的开销则微不足道。

C. 多 *GPU* 评估：可扩展性与通信效率

多 GPU 评估着重考察我们的方法在分布式环境下的可扩展性和通信效率。我们将自己的分布式算法与两种多 GPU 基线方法进行比较：（1） MSz 的朴素并行化；（2）Sync-pMSz。

1) 与 naive-MSz 的比较：为展示 MSz 面临的可扩展性挑战，我们实现了一个将数据划分到多个 GPU 上的朴素分布式版本。每个 GPU

在本地执行原始 MSz，并在每个 R 循环后交换幽灵区域。如图 8 (a) 所示，R 循环占据了总运行时间的主要部分，并涉及进程之间密集的数据通信，最多占计算时间的 60 %。即使与每次迭代后执行同步的基线方法相比，朴素并行 MSz 由于反复进行全局积分路径计算，仍然明显更慢。相比之下，我们的方法避免了 R 循环，并允许每个块在同步前迭代纠正失真，直至实现局部收敛，从而降低通信频率。

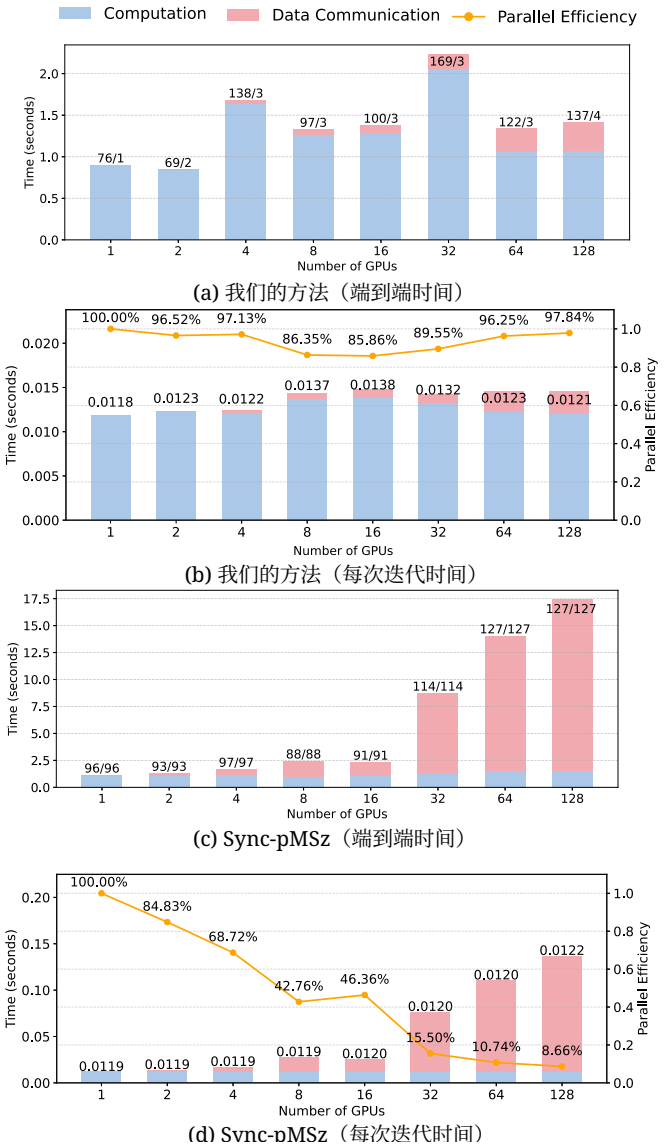
与 sync-pMSz 相比，pMSz 会引入轻微的工作负载不均衡，因为不同块的局部收敛所需迭代次数可能不同（图 7(b)）。虽然这种不均衡影响较小，但可能导致较早完成的块出现空闲时间；探索异步通信策略将作为未来工作。

2) 可扩展性研究：接下来，我们在弱扩展和强扩展两种情况下评估分布式算法的可扩展性，以考察其在大规模 GPU 系统上的并行效率。

弱扩展。我们使用通过 Perlin 噪声生成的合成数据集评估算法的弱可扩展性。每个图形处理器处理一个固定的 512^3 子域，图形处理器数量逐步增加至 128 个，形成大小为 $2048 \times 2048 \times 4096$ 的全局域。尽管每个图形处理器处理的数据大小相同，但由于不同数据集之间以及同一数据集内不同空间块的失真程度各异，整体问题复杂度并不相同，如图 10 (a) 和 (c) 所示。包含失真更严重区域的块需要更多局部校正，因此需要更多迭代才能收敛，如图 8(c) 所示，其中不同块的局部校正时间各不相同。为消除这种差异，我们使用每次迭代的平均计算时间评估弱扩展，从而将每次迭代的可扩展性与数据集相关的收敛行为区分开来，并公平衡量不同问题规模下的并行效率。如图 10 (b) 所示，算法在弱扩展下保持稳定性能，在 128 个 GPU 时并行效率超过 90%；并行效率的计算方式为 $E_p = T_1/T_p$，其中 T_1 和 T_p 分别表示使用一个 GPU 和 p 个 GPU 时每次迭代的运行时间。

强扩展。我们进一步使用固定的 1024^3 Perlin 噪声数据集评估强扩展，将 GPU 数量从 1 增加到 16。这里，我们评估端到端运行时间，而非每次迭代时间，因为所有情况下处理的都是同一个全局数据集，且整体问题规模（PLMSS 失真）在不同 GPU 数量下保持不变。如图 11 所示，我们的方法在所有测试配置下均保持较高的并行效率，始终高于 87%，并在使用 16 个 GPU 时达到 99.3%；效率的计算方式为 $E_p = T_1/(p \cdot T_p)$。

3) 宽松同步策略的影响：我们通过比较 pMSz（我们的方法）与 sync-pMSz，分析同步频率的影响。如图 10(c,d) 所示，我们的方法在 128 个 GPU 上仍能保持高达 97% 的并行效率，



(a) 我们的方法（端到端时间）

(b) 我们的方法（每次迭代时间）

(c) Sync-pMSz（端到端时间）

(d) Sync-pMSz（每次迭代时间）

图 10：我们的方法 (a,b) 与每次迭代都进行通信的基线方法 (c,d) 的弱扩展对比。(a,c) 计算和通信的运行时间分解。(b,d) 平均每次迭代计算时间（柱形）和并行效率（折线）。每个柱形上方的数字表示：(a,c) 迭代次数/同步次数，以及 (b,d) 平均每次迭代计算时间。

而 sync-pMSz 的并行效率则降至仅 8%。通过允许每个块在通信前达到局部收敛，我们的方法将同步频率降低了超过 30×（全局交换最多三次，而基线方法中超过 127 次），从而使通信开销保持轻量，并具备良好的可扩展性。

D. PLMSS 保持性评估

为了评估我们的方法保持 PLMSS 的能力，我们开展实验分析：(1) 在保持 PLMSS 的前提下可实现的最大压缩比；(2) 额外编辑带来的计算与存储开销；

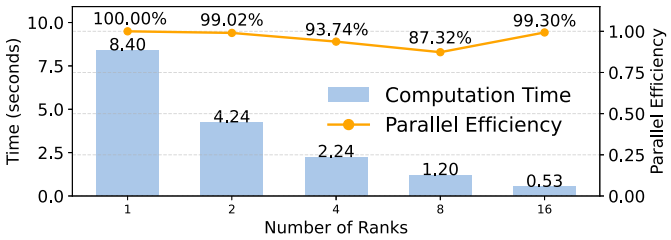


图 11：我们的方法在固定大小的1024^3 Perlin噪声数据集上的强扩展结果。柱状图表示计算时间，折线表示相应的并行效率。

以及 (3) 在不同数据集上保持 PLMSS 的效果。

PLMSS 保持下的最大压缩比如表 I 所示，我们评估了在不同数据集和基础压缩器（SZ3 和 ZFP）下实现 PLMSS 保持时的最大压缩比（压缩率）。基础压缩器通常只能在无损或极低压缩比下保持 PLMSS。相比之下，我们的方法在保持 PLMSS 的情况下最高可实现 88× 压缩，相较于所测试数据集中的基础压缩器，提升幅度最高达 a 75× 。

此外，由于拓扑复杂度存在差异，不同数据集的最大压缩率也有所不同。PLMSS 更复杂或对扰动更敏感的数据集，往往会限制可实现的压缩效果。例如，湍流射流数据集包含大量填充值，这会放大 PLMSS 失真；在考虑 PLMSS 保持时，与其他数据集相比，其压缩率相对较低。另一个总体趋势是，在 PLMSS 保持方面，ZFP（我们的方法）始终比 SZ3（我们的方法）实现更低的压缩率，因为 ZFP 的原始压缩比低于 SZ3。

表 I：不同压缩器在保留 PLMSS 时达到的最大压缩比（CR）。标有 * 的值表示只有无损压缩才能保留 PLMSS。

数据集	Size	SZ3	SZ3（含编辑）	ZFP	ZFP（含编辑）
NYX（暗密度）	8GB	1.20×*	**8.24**×	1.01×*	**7.35**×
NYX	1GB	1.06×	**8.49**×	1.16×*	**7.78**×
湍流射流	317MB	1.96×	**3.90**×	1.70×	**4.42**×
CSEM	50MB	1.06×*	**10.75**×	1.07×*	**2.45**×
涡旋	16MB	1.93×	**25.80**×	2.80×	**11.48**×
AT	6.2MB	1.16×	**88.00**×	1.32×*	**29.29**×

计算与存储开销。尽管我们的方法引入了额外的存储和计算开销，但图12中在单个节点上测得的结果表明，在 PLMSS 保持的情况下，我们始终能够实现高达 88× 的压缩比。就计算开销而言，我们的方法的校正时间不会超过压缩时间，这表明我们的方法不会成为压缩流程中的瓶颈。

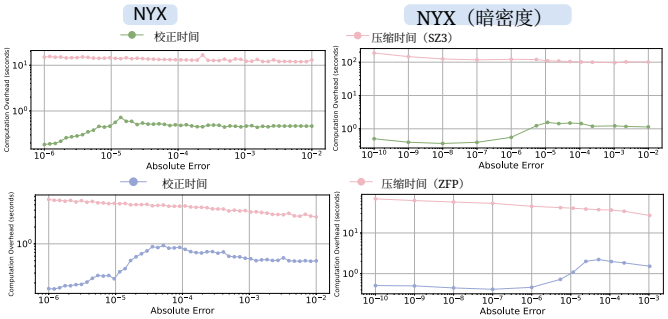


图 12：不同数据集上的计算开销与绝对误差。上：SZ3；下：ZFP。粉色线表示基础压缩器的压缩时间。

对于存储开销，额外成本来自存储编辑。不过，即使计入存储成本，所实现的压缩比仍显著高于基础压缩器。例如，在腺嘌呤-胸腺嘧啶（AT）数据集上，使用 SZ3 时，我们的方法可达到 88× 的压缩率，而如表 I 所示，基础压缩器只能在无损模式下保持 PLMSS。

PLMSS 保持。我们使用涡旋数据集，在相对误差界为 10^{-4} 的条件下评估 PLMSS 保持情况。尽管 SZ3 和 ZFP 的解压体数据在视觉上与原始数据相似，但二者都会引入 PLMSS 错误：ZFP 产生了 3,082 个错误标记点、8 个伪极小值和 0 个伪极大值；SZ3 产生了 176,162 个错误标记点、56 个伪极大值和 153 个伪极小值。相比之下，我们的方法使 SZ3 和 ZFP 均实现了 100% 的 PLMSS 保持，修正了所有 PLMSS 错误（图 13）。

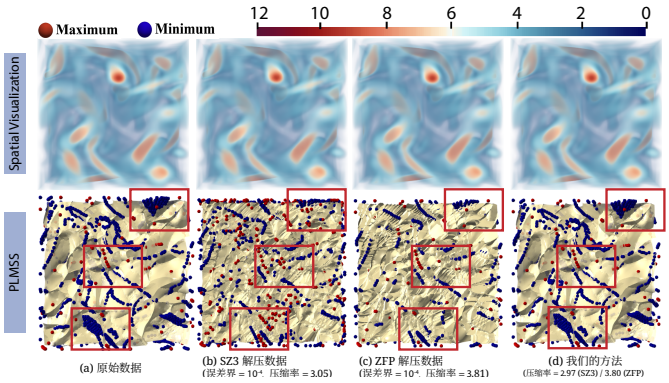


图 13：(a) 原始数据的 PLMSS。(b、c) 在误差界 10^{-4} 下，SZ3 和 ZFP 解压缩数据及其在 PLMSS 中引入的失真（红框）。(d) 我们的方法应用于 SZ3 和 ZFP 后，不会产生 PLMSS 畸变。

VI. 结论与未来工作

本文通过移除依赖重复积分路径计算、计算开销高昂的 R 循环，解决了 MSz 的主要可扩展性瓶颈。我们提出了一种用于误差有界有损压缩的 PLMSS 分布式内存算法。通过重新表述保持条件，使其能够进行完全局部的校正，我们的方法

在单图形处理器执行上相较于 MSz 实现了最高达 14× 的加速比。此外，我们设计了一种宽松的进程间通信策略，使每个数据块在与相邻进程排名通信之前达到局部收敛，从而大幅降低通信频率和成本。借助 CUDA 感知 MPI，我们的分布式实现可高效扩展至 128 个 GPU，在弱扩展下效率超过 90%，表明我们的方法在保持 100% PLMSS 保持率的同时，有效解决了 MSz 的计算和通信瓶颈。

我们的工作存在一些局限性。首先，由于我们的方法严格保留每个数据点的局部排序，因此相比 MSz 会引入更多编辑。采用自适应策略，有选择地放宽顺序保留要求，可进一步减少编辑数量和存储开销。其次，尽管我们目前通过允许每个块在幽灵交换前达到局部收敛，放宽了幽灵层同步要求，但在某些情况下，边界一致性可能会延迟。未来的工作可以探索自适应通信方案，以更好地平衡准确性和效率。第三，允许在同步前实现局部收敛，会使不同块可能在不同时间收敛，因为各块的失真可能不同；较早收敛的块将处于空闲状态，而相邻块仍在进行局部校正。采用异步数据通信可以缓解这一问题，并进一步提升可扩展性。

致谢

本研究得到美国能源部科学办公室先进科学计算研究项目的支持，合同编号为 DE-AC02-06CH11357；同时也得到美国国家科学基金会项目 OAC-2313122、OIA-2327266 和 OAC-2313124 的资助。本研究使用了美国能源部科学办公室用户设施——国家能源研究科学计算中心（NERSC）的资源。

参考文献

[1] S. Di, J. Liu, K. Zhao, X. Liang, R. Underwood, Z. Zhang, M. Shah, Y. Huang, J. Huang, X. Yu, C. Ren, H. Guo, G. Wilkins, D. Tao, J. Tian, S. Jin, Z. Jian, D. Wang, M. H. Rahman, B. Zhang, S. Song, J. Calhoun, G. Li, K. Yoshii, K. Alharthi 和 F. Cappello, 《科学数据集误差有界有损压缩综述》, *ACM*计算调查, 第57卷, 第11期, 2025年6月。 [在线]。可访问：https://doi.org/ 10 1145 3733104 。 / R.

[2] Kettimuthua、Z. Liua、D. Wheelerd、I. Fostera、K. Heitmanna 和 F. Cappello, 《一天内传输一个拍字节》, 载于*IEEE/ACM SC17* 数据密集型科学网络创新研讨会 *(INDIS 2017)*, 2017年。[3] D. Tao、S. Di、Z. Chen 和 F. Cappello, 《基于多维预测和误差控制量化显著改善科学数据集的有损压缩》, 载于*2017*年*IEEE*国际并行与分布式处理研讨会论文集, 2017年, 第1129–1139页。[4] X. Liang、S. Di、D. Tao、S. Li、S. Li、H. Guo、Z. Chen 和 F. Cappello, 《针对科学数据集高压缩比优化的误差控制有损压缩》, 载于*2018*年*IEEE*国际大数据会议论文集, 2018年, 第438–447页。

[5] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, Z. Chen, and F. Cappello, 《SZ3：用于组合基于预测的误差有界有损压缩器的模块化框架》, IEEE大数据汇刊, 第9卷, 第2期, 第485–498页, 2023年。[6] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cap- pello, 《通过动态样条插值优化科学数据的误差有界有损压缩》, 载于《2021年IEEE第37届数据工程国际会议论文集》, 2021年, 第1643–1654页。[7] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, and F. Cappello, 《通过二阶预测和参数优化显著提升高性能计算数据集的有损压缩》, 载于《第29届高性能并行与分布式计算国际研讨会论文集》, 2020年, 第89–100页。 [在线]。获取地址：https://doi.org/ 10 1145 3369583 3392688 。/。P.

[8] Lindstrom, 《定码率压缩浮点数组》, IEEE可视化与计算机图形学汇刊, 第20卷, 第12期, 第2674–2683页, 2014年。[9] P. Lindstrom and M. Isenburg, 《浮点数据的快速高效压缩》, IEEE可视化与计算机图形学汇刊, 第12卷, 第5期, 第1245–1250页, 2006年。[10] Y. Huang, S. Di, X. Yu, G. Li, and F. Cappello, 《cuSZp：具备优化端到端性能的超高速图形处理器误差有界有损压缩框架》, 载于《高性能计算、网络、存储与分析国际会议论文集》, 2023年, 第1–13页。[11] Y. Huang, S. Di, G. Li, and F. Cappello, 《cuszp2：吞吐量极高且压缩率经过优化的图形处理器有损压缩器》, 载于《高性能计算、网络、存储与分析国际会议论文集》, 2024年, 第1–18页。[12] H. Bhatia, A. G. Gyulassy, V. Lordi, J. E. Pask, V. Pascucci, and P.-T. Bremer, 《TMS：面向分子和凝聚态系统的全面拓扑探索》, 《计算化学杂志》, 第39卷, 第16期, 第936–952页, 2018年。 [在线]。获取地址：https://onlinelibrary.wiley.com/doi/abs/ 10 1002 25181 。/jcc。D.

[13] Gunther, R. A. Boto, J. Contreras-Garcia, J.-P. Piquemal, and J. Tierny, 《表征化学系统中的分子相互作用》, IEEE可视化与计算机图形学汇刊, 第20卷, 第12期, 第2476–2485页, 2014年。[14] N. Shivashankar, P. Pranav, V. Natarajan, R. v. d. Weygaert, E. P. Bos, and S. Rieder, 《Felix：用于可视化探索宇宙丝状结构的拓扑框架》, IEEE可视化与计算机图形学汇刊, 第22卷, 第6期, 第1745–1759页, 2016年。[15] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah, 《用于识别和追踪云系统移动的探索框架》, IEEE可视化与计算机图形学汇刊, 第19卷, 第12期, 第2896–2905页, 2013年。[16] J. H. Chen, 《使用s3d进行湍流燃烧的太尺度直接数值模拟》, 《计算科学与发现》, 第2卷, 第1期, 第015001页, 2009年。[17] R. G. C. Maack, J. Lukasczyk, J. Tierny, H. Hagen, R. Maciejewski, and C. Garth, 《分段线性莫尔斯-斯梅尔分割的并行计算》, IEEE可视化与计算机图形学汇刊, 第1–14页, 2023年。[18] Y. Li, X. Liang, B. Wang, Y. Qiu, L. Yan, and H. Guo, 《用于校正误差有界有损压缩器中莫尔斯-斯梅尔分割的高效并行算法MSz》, IEEE可视化与计算机图形学汇刊, 第31卷, 第1期, 第130–140页, 2025年。[19] L. Yan, X. Liang, H. Guo, and B. Wang, 《TopoSZ：在误差有界有损压缩中保持拓扑》, IEEE可视化与计算机图形学汇刊, 第30卷, 第1期, 第1302–1312页, 2024年。[20] N. Gorski, X. Liang, H. Guo, L. Yan, and B. Wang, 《为有损压缩器增添拓扑保证的通用框架》, IEEE可视化与计算机图形学汇刊, 2025年, 预印本。[在线]。获取地址：https://arxiv.org/abs/ 2502 14022 。M. Xia, S. Di, F. Cappello, P. Jiao,

[21] K. Zhao, J. Liu, X. Wu, X. Liang, and H. Guo, 《在有损压缩中使用行列式符号谓词保持拓扑特征：向量场临界点案例研究》, 载于《2024年IEEE第40届数据工程国际会议（ICDE）》, 2024年, 第4979–4992页。[22] M. Xia, B. Wang, Y. Li, P. Jiao, X. Liang, and H. Guo, 《TspSZ：用于拓扑骨架保持的高效并行误差有界有损压缩器》, 载于《2025年IEEE第41届国际会议

数据工程（ICDE）》。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会，2025年5月，第3682–3695页。 [在线]。获取地址：https://doi.ieeecomputersociety.org/ 10 1109 65448 2025 00275 。/ICDE。。[23] X. Liang, H. Guo, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, and T. Peterka, 《迈向特征保持的二维和三维矢量场压缩》，载于《2020年IEEE太平洋可视化研讨会论文集》，2020年，第81–90页。[24] S. Di and F. Cappello, 《难压缩高性能计算数据的误差有界有损压缩优化》，IEEE并行与分布式系统汇刊，第29卷，第1期，第129–143页，2018年。[25] ——, 《使用SZ实现快速误差有界高性能计算数据有损压缩》，载于《IPDPS'16：2016年IEEE国际并行与分布式处理研讨会论文集》，2016年，第730–739页。[26] Q. Gong, J. Chen, B. Whitney, X. Liang, V. Reshniak, T. Banerjee, J. Lee, A. Rangarajan, L. Wan, N. Vidal, Q. Liu, A. Gainaru, N. Podhorszki, R. Archibald, S. Ranka, and S. Klasky, 《MGARD：用于高性能、误差可控数据压缩与重构的多重网格框架》，SoftwareX，第24卷，第101590页，2023年。 [ ] 10 1016 2023 101590 在线。获取地[27] 址：http://dx.doi.org/。/j.softx。。K. Zhao, S. Di, D. Perez, X. Liang, Z. Chen, and F. Cappello, 《Mdz：面向分子动力学的高效误差有界有损压缩器》，载于《2022年IEEE第38届数据工程国际会议（ICDE）》，2022年，第27–40页。[28] "ZSTD"，http://www.zstd.net，访问日期：2024年9月20日。[29] "GZIP"，https://www.gzip.org/，访问日期：2024年9月20日。[30] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen, and F. Cappello, 《探索基于自编码器的科学数据误差有界压缩》，载于《2021年IEEE国际集群计算会议（CLUSTER）论文集》，2021年，第294–306页。[31] J. Liu, S. Di, S. Jin, K. Zhao, X. Liang, Z. Chen, and F. Cappello, 《SRN-SZ：基于深度学习和超分辨率神经网络的科学数据误差有界有损压缩》，2023年。[32] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang, 《利用拓扑描述子进行标量场比较：科学可视化中的性质与应用》，Computer Graphics Forum，第40卷，第3期，第599–633页，2021年6月。 [ ] 10 1111 14331 在线。获取地址：[33] http://dx.doi.org/。/cgf。X. Liang, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, T. Peterka, and H. Guo, 《迈向特征保持的矢量场压缩》，IEEE可视化与计算机图形学汇刊，第29卷，第12期，第5434–5450页，2023年。[34] M. Soler, M. Plainchault, B. Conche, and J. Tierny, 《拓扑控制有损压缩》，载于《2018年IEEE太平洋可视化研讨会论文集》，2018年，第46–55页。 [在线]。获取地址：https://doi.ieeessociety.org/ 10 1109 2018 00015 。/PacificVis。。T.[35] Peterka, R. Ross, B. Nouanesengsy, T.-Y. Lee, H.-W. Shen, W. Kendall, and J. Huang, 《稳态和时变流场的并行粒子追踪研究》，载于《2011年IEEE国际并行与分布式处理研讨会》，2011年，第580–591页。[36] B. Nouanesengsy, T.-Y. Lee, and H.-W. Shen, 《大规模向量场上的负载均衡并行流线生成》，IEEE可视化与计算机图形学汇刊，第17卷，1794，2011年12月，第1785–页。[37] B. Nouanesengsy, T.-Y. Lee, K. Lu, H.-W. Shen, and T. Peterka, 《时变流场的并行粒子平流和FTLE计算》，载于《SC '12：高性能计算、网络、存储与分析国际会议论文集》，2012年，第1–11页。[38] C. M¨uller, D. Camp, B. Hentschel, and C. Garth, 《使用工作请求进行分布式并行粒子平流》，载于《2013年IEEE大规模数据分析与可视化研讨会（LDAV）》，2013年，第1–6页。[39] W. Kendall, J. Wang, M. Allen, T. Peterka, J. Huang, and D. Erickson, 《简化的并行域遍历》，载于《SC '11：2011年高性能计算、网络、存储与分析国际会议论文集》，2011年，第1–11页。[40] L. Chen and I. Fujishiro, 《优化大规模分布式流数据集的流线可视化并行性能》，载于《2008年IEEE太平洋可视化研讨会》，2008年，第87–94页。[41] C. Garth, F. Gerhardt, X. Tricoche, and H. Hans, 《流体流动应用中相干结构的高效计算与可视化》，IEEE可视化与计算机图形学汇刊，第13卷，第6期，第1464–1471页，2007年。[42] K. Lu, H.-W. Shen, and T. Peterka, 《大规模向量场上流面的可扩展计算》，载于《SC '14：论文集 高性能计算、网络、存储与分析国际会议，2014年，pp. 1008–1019。[43] J. Zhang、H. Guo 和 X. Yuan，"基于高阶访问依赖的高效非定常流可视化"，载于*2016 IEEE Pacific Visualiza- tion Symposium (PacificVis)*，2016年，pp. 80–87。[44] H. Guo、X. Yuan、J. Huang 和 X. Zhu，"耦合集合流线平流与分析"，*IEEE*可视化与计算机图形学汇刊，vol. 19，no. 12，pp. 2733–2742，2013年。[45] D. Morozov 和 T. Peterka，"使用 DIY2 进行块并行数据分析"，载于*2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV)*，2016年，pp. 29–36。[46] M. Will、J. Lukasczyk、J. Tierny 和 C. Garth，"用于分段线性 Morse-Smale 分割与连通分量的分布式路径压缩"，2024年。 [在线]。获取地址：https://arxiv.org/abs/2409.03771