# 大规模辐射流体力学：使用 FleCSI 比较消息传递接口与异步多任务运行时

1st Alexander Strack并行与分布式系统研究所 斯图加特大学斯图加特，德国
alexander.strack
@ipvs.uni-stuttgart.de

2nd HartmutKaiser计算与技术中心 路易斯安那州立大学美国路易斯安那州巴吞鲁日
hkaiser@cct.lsu.edu

3th DirkPfl¨uger并行与分布式系统研究所 斯图加特大学斯图加特，德国
dirk.pflueger
@ipvs.uni-stuttgart.de

arXiv:2603.05366v1 [cs.DC] 5 Mar 2026

**摘要—编写高效的分布式代码仍是一项耗费人力且复杂的工作。为简化应用程序开发，灵活计算科学基础设施（FleCSI）框架提供了一种面向用户的高级编程接口，该接口基于基于任务的运行时模型构建。在内部，FleCSI 集成了最先进的并行化后端，包括消息传递接口以及异步多任务运行时（AMTR）Legion 和 HPX，使应用程序能够充分利用异步并行性。**

**在本研究中，我们使用 FleCSI 的三个后端对两种应用进行基准测试，最多扩展至 1024 个节点，旨在量化 AMTR 后端带来的优势与开销。我们选取简单的泊松求解器和多维辐射流体力学代码 HARD 作为代表性应用。在以通信为主的泊松求解器基准测试中，采用消息传递接口后端的 FleCSI 在最多 131072 个核心的弱扩展测试中实现了超过 97% 的并行效率，表明其抽象层仅引入了极小的开销。Legion 后端则表现出显著的开销和扩展性限制，而 HPX 后端相较于 MPI+Kokkos 仅引入了微不足道的开销。不过，由于使用了未经优化的 HPX 集合操作，HPX 后端目前的可扩展性仍受到限制。**

**在以计算为主的辐射流体力学基准测试中，消息传递接口后端与 HPX 后端之间的性能差距缩小了。在少于 64 个节点时，HPX 后端的性能优于 MPI+Kokkos：弱扩展下平均加速比为 1.31，强扩展下最高可达 1.27。对于仅测试流体动力学的 HARD 基准测试，HPX 后端在少于 32 个节点时表现出更优的性能；相较于 MPI，加速比最高可达 1.20，相较于 MPI+Kokkos 则最高可达 1.64。**

**这些结果凸显了两个关键见解：尽管引入了抽象层，FleCSI仍保持出色的可扩展性和效率；AMTR后端虽然带来了额外开销，但在复杂的异步工作负载中展现出显著的性能提升潜力。**

索引词—**辐射流体动力学、异步任务、消息传递接口、HPX、Legion、FleCSI。**

## I. 引言

科学计算正朝着一种方向发展，使我们能够通过更复杂、更精细的模型，在模拟中捕捉更多真实世界的现象。这类模拟通常在大型超级计算机上运行。鉴于近期计算正转向众核计算节点（其中也可能包括加速器），程序员需要熟练掌握先进的并行化工具，例如消息传递接口 [1], CUDA [2], 和 Kokkos [3],，以充分利用现代计算架构。灵活计算科学基础设施（FleCSI） [4] 通过现代 C++ 语法提供简洁的高级接口。此外，代码具有很强的可移植性，并可通过 Kokkos [3] 后端适配所有现代硬件。分布式通信完全由系统抽象，而高级的基于任务的编程模型则支持使用异步多任务运行时（AMTR），例如 HPX [5] 和 Legion [6]，进行并行化。FleCSI 也支持通过消息传递接口进行同步执行。FleCSI 本身引入的抽象以及 AMTR 后端都会带来开销，进而可能影响性能。在涉及 AMTR 的基准测试中，通常会选择预期能借助 AMTR 提升性能的工作负载。因此，实际的抽象开销无法得到准确量化。为此，我们在本研究中特意选择了一个更适合采用基于 MPI+X 的经典并行化方法的应用。这样便能直观呈现，在 FleCSI 中使用 AMTR 相较于优化后的 MPI 所产生的纯粹成本。此外，我们还选择了一个更适合异步并行化的复杂应用。基于这一方法，我们提出以下研究问题：

- FleCSI引入的通用抽象会如何影响基于基本消息传递接口的并行化的可扩展性?
- 与同步MPI以及MPI+Kokkos后端相比，两个AMTR后端分别会带来多大开销?
- 在更复杂的应用程序中，AMTR任务管理开销能否有效隐藏？AMTR能否进一步提升性能?

因此，我们重点评估 FleCSI 三种后端在仅含CPU的节点上的可扩展性和性能。本研究不考虑 GPU加速。尽管如此，

FleCSI 框架支持通过 Kokkos 配合所有后端使用加速器。然而，异构计算会增加 CPU 与 GPU 之间的通信开销；为单独考察异步多任务运行时带来的开销，我们不考虑这部分开销。

我们选择了计算流体力学（CFD）领域的两个科学计算应用程序作为基准测试。第一个应用程序是用于泊松方程的红黑高斯-赛德尔压力求解器，它提供了一个以通信为重点的基准测试问题。尽管不包含复杂的物理过程或更高精度的数值方法，但它能够暴露异步多任务运行时引入的任何开销。第二个应用程序是辐射流体力学代码 HARD [7], ，它类似于计算密集型的真实工作负载。HARD 支持一维、二维和三维问题的模拟。为了用这两个应用程序评估 FleCSI，我们在 Chicoma 超级计算机上最多 1024 个节点上进行了强扩展和弱扩展测试。

我们在这项工作中的主要贡献包括：

- 首次在大规模上、跨所有可用 FleCSI 后端、使用多个应用程序对 FleCSI 框架进行的评估，
- 对消息传递接口、Legion 和 HPX 后端的对比分析，这些后端均执行使用 FleCSI 实现的同一 C++ 代码，以及
- 为 FleCSI 泊松示例开发了一种新的基准测试模式，从而支持异步计时，并简化对各种 FleCSI 后端的基准测试。

本文的其余部分安排如下：第二节回顾 FleCSI 和异步多任务运行时相关的工作。第三节介绍 FleCSI 框架，并概述 FleCSI 中可用的不同并行化后端。第四节介绍本文用于对 FleCSI 进行基准测试的应用程序。第五节讨论我们在 Chicoma 超级计算机上进行基准测试的结果。最后，第六节总结本文并展望未来工作。

## II. 相关工作

FleCSI 的开发早在 2016 [8] 就已开始。FleCSI 2.0 于 [4] 推出，相较于最初的实现进行了重大改动。FleCSI 框架为共享内存编程和分布式内存编程提供了统一接口。Uintah框架 [9] 采用了类似的方法，提供统一接口和基于任务的运行时。然而，尽管任务依赖关系在 FleCSI 中是隐式定义的，Uintah 的用户必须显式允许并发执行。此外，Uintah 采用静态资源映射，而 FleCSI 支持对计算任务进行动态资源映射。多处理器计算运行时（MPC） [10] 也采用了另一种类似的方法。虽然它同样支持 MPI+X 编程模型，但资源映射是静态的，映射到 MPI进程。

FleCSI 支持用户使用多种后端进行分布式计算，包括异步消息传递接口后端，以及另外两种分别利用 Legion [6]和 HPX [5]的异步后端。有关不同后端的详细介绍，请参阅第三节。Kokkos 后端确保性能可移植性 [3]，并可与 FleCSI 的分布式后端无缝集成。此外，FleCSI 内置支持 Caliper工具箱 [11]。Caliper 可轻松对基于 FleCSI 构建的应用程序进行性能测量和性能分析。

FleCSI 的应用包括 FleCSPH [12], Moya [13],和 HARD [7]。FleCSPH 实现了一个利用基于树的数据结构的光滑粒子流体动力学求解器。相比之下，HARD 使用规则网格，但采用通量限制扩散方法实现辐射流体力学 [14], [15]。

现有多种异步任务运行时，尽管它们有一些相似之处，但各自也有独特的特性。关于通用分类体系，请参阅 Thoman 等人 [16] ；关于异步多任务运行时的比较综述，请参阅 Schuchart 等人 [17]。在性能方面，Task Bench 框架可用于比较不同异步多任务运行时在多种算法上的表现 [18], [19]。然而，在 Task Bench 中，每种异步多任务运行时对算法的实现方式各不相同。使用 FleCSI，我们可以通过不同后端运行相同的代码。

## III. 软件框架

FleCSI 提供一套编程工具 [4], ，降低了并行编程的入门门槛。具体而言，FleCSI 核心框架提供一系列分布式数据拓扑：topo::narray 拓扑实现任意 n 维数组，适用于欧拉流体力学等结构化网格应用；topo::unstructured 拓扑是一种图数据结构，专为有限元或有限体积法等非结构化网格应用设计；对于光滑粒子流体动力学等采用基于树的算法的应用，FleCSI 则提供 topo::ntree 拓扑。FleCSI 运行时负责管理这些分布式拓扑中相互依赖部分之间的通信。这样，开发者无需显式通信，即可直接访问幽灵层或边界层中的数据。除了这些底层数据结构，FleCSI 特化还支持实现一系列更高层次的数据结构，例如多维网格。

FleCSI 中的控制流通过控制模型来定义，其中包含控制点、动作、任务和内核的层次结构。这种结构使用户能够编写可移植、可扩展的应用程序代码，使其适用于不同的系统架构，并能够识别和利用并发。有关 FleCSI 控制模型、数据模型和执行模型的详细说明，请参阅 [4]。

在这项工作中，我们重点关注不同的 FleCSI 后端。图 1 展示了 FleCSI 后端栈。FleCSI 控制模型中定义的任务可通过三种不同的后端执行。后端的选择在编译时确定。如果使用消息传递接口后端构建，FleCSI 会按顺序遍历任务图执行。相比之下，Legion

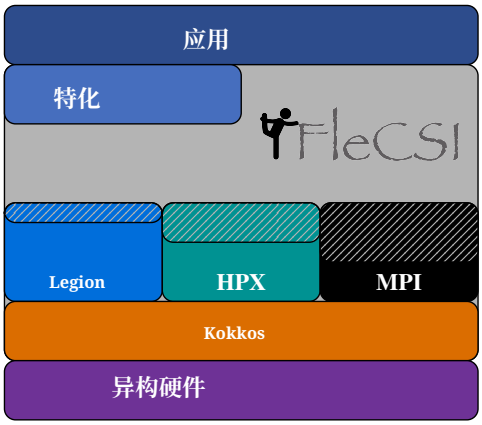


图 1. FleCSI 后端栈

和 HPX 后端能够分析任务之间的依赖关系，并发运行多个互不依赖的任务。在 FleCSI 中，任务间的依赖关系由传入任务的字段的数据访问权限（在 FleCSI 的术语中称为 特权 ）隐式定义。这与 Legion 的依赖管理方法十分相似（见第 III-B 小节）。HPX 后端会自动将这些隐式依赖映射为基于 future 的依赖表示 [13]。

对于计算内核，FleCSI 在 Kokkos::parallel_for 之上提供了一个轻量的 for_all 抽象，该抽象保留了 FleCSI 范围模型，并为集合操作强制执行内存一致性。因此，FleCSI 能够获得 Kokkos 的全部可移植性优势，同时提供分布式并行化与共享内存并行化之间的交汇。在中央处理器上，FleCSI 的 for_all 可通过 Kokkos 使用 OpenMP 或 HPX。在图形处理器上，它可通过 Kokkos 使用 CUDA、ROCm 或 SYCL。因此，FleCSI 能够面向所有主要厂商的硬件，并充分适用于现代异构计算环境。

由于分布式后端是本文的主要关注点，我们先简要介绍消息传递接口，随后在后续小节中介绍 Legion 和 HPX。

### A. 消息传递接口

消息传递接口（MPI） [1] 于1994年问世，现已成为分布式内存实体之间通信的事实标准。顾名思义，MPI通过显式发送和接收消息来实现数据交换。它支持MPI进程之间阻塞和非阻塞的点对点通信。然而，对于复杂的并行算法，仅依赖点对点通信可能会变得难以管理，导致代码杂乱且出错风险增加。为了解决这些问题，MPI提供了一系列与预定义通信模式相匹配的集合操作。这些集合通信在通信器中组织的进程组内进行。需要注意的是，MPI是一种定义语法和功能的规范，而非具体实现。广泛可用的常见实现包括 OpenMPI [20]和MPICH [21]。一些硬件供应商会提供自有或专门定制的MPI版本，例如HPECray [22] 提供的Cray MPICH。

FleCSI 中的 MPI 后端最初是作为一种风险缓解策略而开发的，旨在提供 FleCSI 编程模型的串行实现。尽管如此，FleCSI 的数据权限使得 MPI实现非常简洁，能够在没有显式通信语义的情况下维持内存一致性，即依赖关系由字段访问权限推断得出。FleCSI 的集合通信接口直接使用相应的 MPI 集合通信，并使用点对点通信进行幽灵层和边界层交换。由于任务在 MPI 后端中按顺序执行，运行时依赖分析、任务创建和调度所产生的开销非常小。相比之下，HPX 和 Legion 需要动态分析以实现乱序任务执行。

### *B. Legion*

Legion 是由斯坦福大学开发的并行编程系统 [6]。Legion 引入了一种抽象，将计算与执行细节分离。数据被组织成逻辑区域。局部区域是用户定义的集合，可映照复杂的数据结构。程序员以这些区域为基础描述任务，并声明每个任务如何访问数据。Legion 提供默认的 映射器，用于确定任务和数据在可用资源上的放置位置与方式。也可以针对特定应用或系统架构编写自定义映射器。

Legion 基于 Realm [23]构建，将其用作底层资源管理器，负责管理内存分配和通信等事项。Realm 目前支持多种网络传输协议，包括消息传递接口 [1],GASNet [24]，和 UCX [25]。可以通过命令行参数设置各种运行时参数。Regent [26]领域专用语言构建于 Legion 之上。它简化了 Legion 模型，使用户能够编写高级代码，并以更少的代码行表达相同行为。PENNANT [27]是一款使用 Regent 编写的非结构化网格应用示例。文中使用单粒子成像代码 SpiniFEL [29]，对比了采用 Legion 运行时开发软件与使用消息传递接口 [28]，开发软件的情况。

FleCSI 为 Legion 后端实现了部分定制的映射器，并支持在特化层和应用层进行更多定制。由于 Legion 后端支持将数据和任务动态映射到系统计算资源上，因此 Realm 的运行时配置还有更多可选项。例如，Legion 支持为每个进程设置多个数据分区（颜色）。虽然这一功能在逻辑上更为复杂，但其优点是：为实现相同程度的分布式内存并行，需要由运行时管理的进程更少。这样的配置消除了 Legion 后端中的冗余工作，因为每个地址空间只需一个运行时进程来进行依赖分析和调度。Legion 的另一项性能优化是提供了跟踪接口，可通过识别

代码中的交互式片段，在这些片段中，每次迭代都可以使用相同的运行时分析。这可以显著提升性能，不过对于主要采用批量同步模式的应用程序而言，通常很难克服 Legion 的运行时开销。

### C. HPX

HPX [5] 运行时于 2008 年首次发布。HPX 是一个符合 C++ 标准的库，专注于实现高性能并发与并行性。其 API 遵循 ISO C++11/14/17/20 标准，并符合 Boost C++ 库的编程指南。HPX 的一项显著特点是其完全异步的执行模型，这通过轻量级 HPX 期值实现。这种方法消除了传统消息传递或 fork-join 方法中常见的隐式同步屏障。借助主动全局地址空间（AGAS），HPX 将其能力扩展至分布式计算 [30]。通信通过称为包的主动消息进行抽象。HPX 可以使用不同的包传输端口来传输这些主动消息。可选方案包括 TCP [31], 、消息传递接口（MPI） [1], 以及轻量级通信接口（LCI） [32]。与 MPI 和 TCP 相比，LCI parcelport 展现出显著的性能优势 [33]。HPX 旨在广泛适配各种处理器架构，包括 x86、ARM [34],和 RISC-V [35], [36]。HPX 还直接支持 CUDA、ROCm、Kokkos 和 SYCL，以面向加速器架构 [37], [38]。HPX 的旗舰应用是天体物理学代码 Octo-Tiger [39]。HPX 的其他应用包括高斯过程回归库 GPRat [40]和快速傅里叶变换工具 HPX-FFT [41]。

HPX 后端通过检查用户在任务参数中提供的特定注解，来识别 FleCSI 任务之间的执行依赖关系。随后，它会自动构建基于期值的依赖树，将这些执行依赖关系提供给调度器。每个 FleCSI 任务都会被调度为一个 HPX 任务，以确保它仅在所依赖的所有相关 FleCSI 数据更新后才执行，并遵循为该任务对应参数指定的访问权限注解。这样可确保所有任务尽早运行，并尽可能提高并发度。具体而言，如果有足够的计算资源，彼此独立的 FleCSI 任务可以并发运行。

在基准测试期间尚不可用的一项功能，是 [13], 中添加到 HPX 后端的通信器回收机制，该机制可提升更复杂应用程序的扩展性。此外，我们还遇到了一些 FleCSI 中 HPX 与 Kokkos 之间的轻微兼容性问题。为使用 for all_ 抽象，我们将 Kokkos::parallel_for 替换为 hpx::for_each 并行循环，作为面向 CPU 的临时解决方法。

## IV. 应用

与通用基准测试相比，评估实际使用 FleCSI 的科学应用，能够更具代表性地反映现实世界中的性能。在科学计算中，计算流体力学（CFD）是各种旨在求解流体或气体物理方程的方法的统称。在本工作中，我们选择了两个复杂程度不同的计算流体力学应用，并将在以下小节中介绍。

*1)* 泊松求解器：我们从一个计算流体力学和数值方法课程中常见的标准问题入手，即求解泊松方程。泊松方程表示为

$$\Delta p = f, \tag{1}$$

这是一个椭圆型偏微分方程，其中 $f$ 通常已知，$\Delta$ 表示拉普拉斯算子，$p$ 则是未知量。在二维情况下，方程（1）可写为

$$\left(\frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2}\right) p(x,y) = f(x,y). \tag{2}$$

在本应用中，我们采用规则结构化网格上的简单有限差分离散化。因此，$p$ 的拉普拉斯算子可近似为

$$\Delta p = \left(\frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2}\right) p \tag{3}$$

$$\approx \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\partial x^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\partial y^2}. \tag{4}$$

将方程（3）代入方程（2），可得到线性方程组

$$A \cdot p = f\,, \tag{5}$$

其中 $A$ 是一个稀疏矩阵。因此，迭代求解器可以加速方程（5）的求解。在本例中，我们选择高斯-赛德尔方法（GSM） [42]。由于 GSM 中的迭代依赖于先前计算的值，因此该方法无法直接并行化。不过，可以将网格划分为棋盘格模式。在文献中，这种方法称为红黑高斯-赛德尔方法 [42]。红色网格仅依赖于黑色网格中的值，反之亦然。因此，可以并行计算各自的子网格。

使用 FleCSI 框架的基于任务的实现，我们获得了一个执行一个时间步并重复运行直至收敛的求解任务。在此任务中，执行 50 次红黑高斯-赛德尔方法迭代。因此，生成的任务图是由相互依赖的求解任务组成的流水线。通过 FleCSI 提供的基于分布式 topo::narray 拓扑的网格数据结构，数据被均匀分布到所有进程中。因此，工作负载均衡且计算开销较低。这些特性的结合使泊松应用程序最适合使用同步MPI后端。这使我们能够突出 FleCSI 中 AMTR后端的开销。

FleCSI 中提供了一个基础的泊松应用程序，可作为独立示例，便于进一步开发。在本工作中，我们扩展了该示例，加入基准测试模式以支持异步计时。原始实现仅采用高层级的同步计时，这会引入隐式同步

屏障。这类同步屏障会引入人为同步，从而扭曲 AMTR后端的运行时间测量结果。新的基准测试模式将计时插桩移至较低层级的 FleCSI 任务中，这些任务以异步方式执行，从而确保性能测量不受同步屏障影响。此外，基准测试模式还会禁用输入/输出操作，并生成包含所记录计时数据的输出文件。

*2) HARD*：作为一个更复杂、能够代表实际 FleCSI 应用场景的应用，我们纳入了 HARD 库 [7]。HARD 可模拟与辐射扩散耦合的可压缩流体动力学，并采用通量限制扩散近似求解辐射流体动力学方程 [14]。HARD 在规则网格上运行，并使用算子分裂将辐射传输的对流部分与非对流部分分离。对流部分采用有限体积法求解，通量通过近似黎曼求解器计算。为达到更高的空间精度，HARD 采用五阶 WENO5-Z重构 [43]。流体动力学部分使用 HLL通量 [44], , 而辐射部分则采用 Lax-Friedrichs通量 [45]。时间精度达到二阶，使用的是 Heun方法。辐射扩散通过带有 Jacobi平滑的几何多重网格求解器求解 [46]。为提高稳健性和可扩展性，本文仅使用 Jacobi求解器。

多种高阶数值方法的结合，以及多维流体动力学与辐射之间的紧密耦合，形成了复杂且计算密集的任务图。FleCSI 的任务式编程模型能够自然地表达这些依赖关系，同时自动管理所有所需的通信。

作为基准测试场景，我们采用三维 Rankine–Hugoniot 问题 [47], [48], 。由于存在需要高阶数值方法处理的尖锐间断，该问题尤其难以计算。辐射部分可以完全禁用，使 HARD 成为纯流体动力学求解器。此外，HARD 提供内置的基准测试模式，可禁用输出写入并启用异步计时，这与我们对泊松应用程序所做的扩展类似。

## V. 结果

本文呈现的结果分为两部分。首先，我们使用轻量级泊松应用程序，展示 Legion 和 HPX 后端相较于消息传递接口后端的开销。在单个节点上，我们进行问题规模基准测试，以展示这些运行时间的节点内任务管理开销。此外，我们还进行强扩展和弱扩展基准测试，着重考察节点间通信，并研究两个 AMTR 后端的可扩展性。其次，我们使用 HARD 应用程序进行强扩展和弱扩展基准测试，以考察复杂的真实应用能否受益于 HPX 后端引入的异步并行。

Chicoma 超级计算机的硬件规格见表 I。所有后端均采用各自最快的进程与每个进程线程数组合，每个节点记为 (#$processes$/#$threads$)。显然，对于纯消息传递接口，这一组合为 (128/1)。其他后端在 Chicoma 上的组合则经过手动调优。AMD Zen 2 CPU 的小芯片设计将芯片划分为四个独立象限，每个象限包含两个核心复合芯片。因此，双路节点的最佳配置为 (8/16)。Legion 后端属于特殊情况，因为每个进程会预留一个线程来管理运行时。借助 Kokkos，FleCSI 可在消息传递接口和 Legion 后端中使用 OpenMP 实现共享内存并行。FleCSI 的 HPX 后端依赖 HPX 并行循环来实现共享内存并行。我们将线程绑定到物理核心，以确保 OpenMP 和 HPX 的线程分布达到最佳状态。

为了更好地比较消息传递接口后端与HPX后端，我们使用HPX的MPI包裹端口，从而使两个后端使用相同的MPI实现进行通信。尽管这会降低HPX后端的性能，但它确保相对于MPI后端的潜在性能提升仅与HPX运行时引入的异步并行化有关。对于Legion后端，我们使用GASNet进行通信；据称，它是相应Legion Spack软件包中最为成熟的方案。

| Name | Chicoma |
|---|---|
| 理论峰值性能 | 9.54PFLOPS |
| 节点 | 1792 |
| 连接 | Slingshot-11 |
| 速度 | 200Gb/s |
| 插槽 | 2 |
| CPU | AMD EPYC 7H12 |
| 核心 | 64 |
| 基准时钟频率 | 2.6GHz |
| 三级缓存 | 256MB |
| RAM | 512GB |

表 I Chicoma 超级计算机的硬件规范

*1)* 泊松求解器：从泊松应用程序开始，我们在 Chicoma 超级计算机上对最多 1024 个节点开展扩展性测试。泊松求解器的所有结果均基于十次运行、每次运行五次迭代计算得到的平均运行时间。误差线表示 95%置信区间。为了正确看待这些扩展性测试，我们首先给出在一个 Chicoma节点上进行问题规模基准测试的结果（见图2）。对于 Legion，与轨迹优化相关的前三次迭代不予考虑。当问题规模大于 $2^{25}$,消息传递接口时，消息传递接口+Kokkos 与 HPX 的表现相近。然而，当问题规模为 $2^{25}$ 及更小时，消息传递接口具有明显的性能优势。消息传递接口+Kokkos 与 HPX 后端在所有问题规模上的扩展情况相似，这表明相对于纯消息传递接口的开销是由计算内核的共享内存分叉-汇合方法造成的。对于较小的问题规模，Legion 后端更大的开销变得明显。此外，Legion 需要更多内存，从而降低了单个节点上的最大问题规模。请注意，该扩展呈非线性，这可以

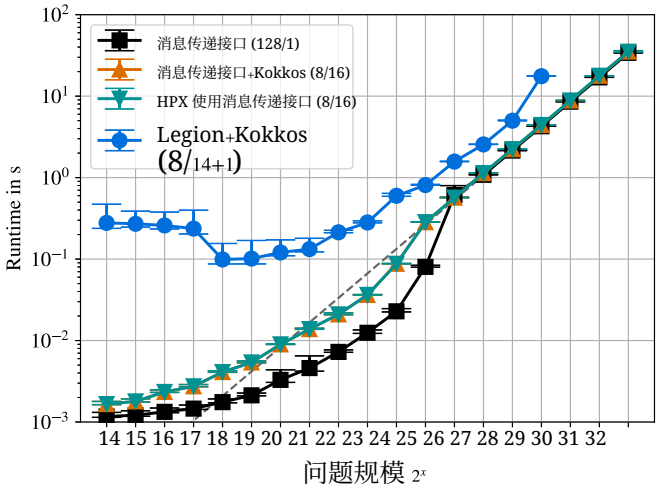


图2. 单个节点上一次泊松求解器迭代的问题规模扩展。对于 Legion，运行时间不包含跟踪优化开销。虚线表示线性扩展。运行配置用 (#*process*/#*threads*) 表示。

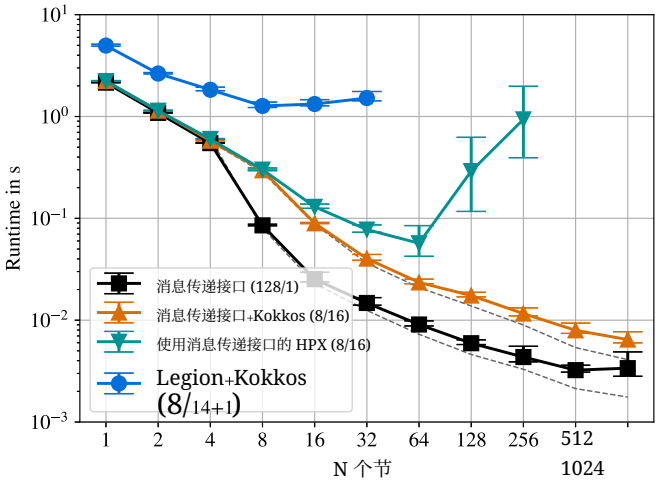


图 3. 最多 1024 个节点上进行一次泊松求解器迭代的强扩展运行时间。问题规模固定为 $2^{28}$。虚线表示理想的单节点性能。每个节点的运行配置以 (#*process*/#*threads*) 表示。

由较小问题规模下更高效的缓存使用来解释。这为后续的强扩展基准测试引入了非线性。

全规模强扩展基准测试采用的问题规模为 $2^{28}$。图3展示了最多使用 1024 个节点时的运行时间。虚线表示单节点的最佳性能。如图2中的规模扩展基准测试所示，纯MPI后端性能最佳。MPI+Kokkos 后端的表现符合预期：与 MPI 后端相比，每节点问题规模减半后，扩展便开始偏离线性。使用 1024 个节点时，MPI 和 MPI+Kokkos 后端的并行效率分别为 55% 和 64%。MPI+Kokkos 后端的效率更高，这与其参与通信的 MPI进程数量减少了 16 倍有关。

相比之下，HPX 后端仅在最多 8 个节点时保持理想的强扩展。超过这一规模后，其扩展性一直下降到 64 个节点，而在更大规模的配置下，性能会显著恶化。我们认为，这种现象是由 FleCSI 依赖集合操作所致。在 HPX 后端中，这些集合操作映射到相应的 HPX集体操作实现。与 MPI 集合通信不同，目前这些实现尚未采用高效的基于树的算法。在当前的 HPX版本中，集合操作的实现方式是由一个主机进程充当通信处理程序。这一瓶颈会在进程数量较多时带来显著开销。HPX开发团队目前正致力于提高 HPX 集合操作实现的效率，并计划在未来的出版物中展示初步结果。

在弱扩展方面，我们在最多 1024 个节点上进行了大规模基准测试，每个节点的问题规模为 $2^{28}$ （见图 4）。基于消息传递接口的后端展现出近乎最优的效率，在 1024 个节点上实现了超过 97% 的并行效率。纯消息传递接口的性能优势略高于线程后端，约为 4%。HPX 后端的波动较大，扩展性也略逊于基于消息传递接口的后端。对于超过 64 个节点的情况，HPX 遇到了与强扩展基准测试中相同的限制，因此我们不列出更高节点数下的性能数据。由于在分布式环境中进行 Legion 跟踪非常耗时，我们无法提供此次测试中的 Legion 性能数据。

在我们的泊松基准测试中，所有异步多任务运行时后端按设计都会产生相较于纯MPI后端可测量的开销。不过，我们的分析还揭示了意料之外的扩展性限制。对于 HPX 后端，主要瓶颈源于 HPX 中的集体通信算法。除这一内部问题外，HPX 的性能与消息传递接口+Kokkos 后端相当，表明其运行时开销较小。相比之下，对于 Legion 后端，我们未能准确确定其运行时开销较大、内存占用较高且扩展效率降低的原因。

*2）HARD*：对于 HARD 应用，本文展示的所有结果均基于五次运行的迭代运行时间中位数。误差线表示这五次运行中的最小值和最大值。我们直接从强扩展基准测试开始，针对规模为 $2^{9+9+9}$ 的三维问题测试至 1024 个节点（见图 5）。由于工作负载的计算密集度更高、对通信的依赖更低，HPX 后端的扩展性限制得以缓解。纯MPI后端再次表现出最佳性能，并在最多 64 个核心范围内实现最优扩展；而 HPX 后端在最多 32 个节点时的性能优于消息传递接口+Kokkos 后端。在最多 16 个节点的范围内，我们观察到 HPX 相较于消息传递接口+Kokkos 的加速比为 1.14 至 1.45。

如果禁用辐射，只计算流体动力学

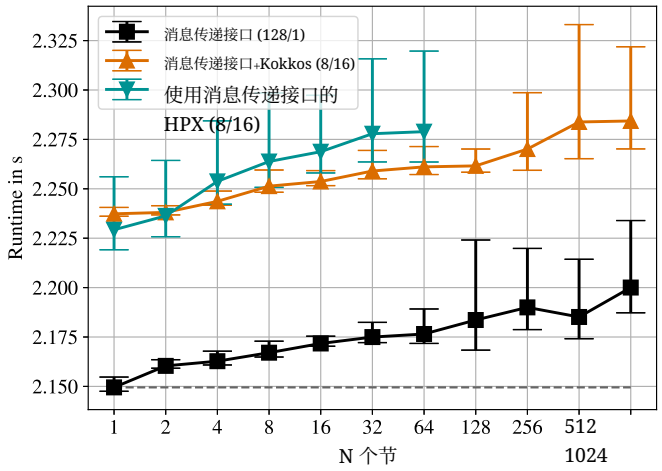


图 4. 在最多 1024 个节点上，泊松求解器单次迭代的弱扩展运行时间。每个节点的问题规模设为 $2^{28}$。虚线表示理想扩展。每个节点的运行配置以（$\#process/\#threads$）表示。

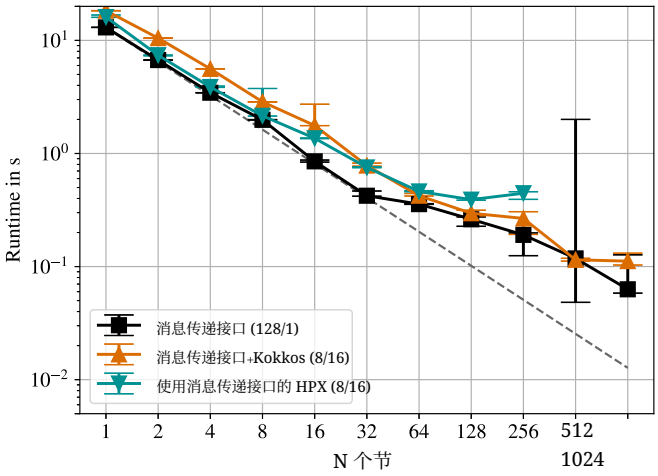


图 5. 使用 HARD 在最多 1024 个 Chicoma节点上运行三维辐射流体力学基准测试时，单次迭代的强扩展运行时间（见图 I）。问题规模固定为 $2^{9+9+9}$。虚线表示理想扩展。每个节点的运行配置以 ($\#process/\#threads$) 表示。

我们使用 HARD 进行强扩展基准测试时，可以在图 6 中观察到 HPX 后端在单节点上的性能优势。HPX 后端的单节点性能最佳，相较于 MPI 后端，速度提升了 1.14 倍；相较于 MPI+Kokkos，速度提升了 1.27 倍。此外，HPX 后端的性能优势一直保持到 64 个节点，之后其扩展性限制开始显现。对于 32 个或更少的节点，HPX 相较于 MPI+Kokkos 的速度提升最高可达 1.64 倍，相较于 MPI 则最高可达 1.20 倍。

图 7 展示了三维辐射问题在最多 512 个节点上的 HARD 弱扩展基准测试结果。每个节点的问题规模固定为 $2^{8+8+8}$。尽管纯 MPI 在最多 64 个节点时能达到理想效率，但其性能开始迅速下降。相比之下，MPI+Kokkos 使用的 MPI 进程数量减少了 16 倍，扩展性更好。在单个节点至 64 个节点的范围内，HPX 的平均速度比 MPI+Kokkos 方法快 1.31 倍。两个后端都依靠异步 MPI 进行通信，并对大型计算内核采用分叉-汇合方法。不过，HPX 后端能够利用 HPX 运行时异步调度计算内核的能力。

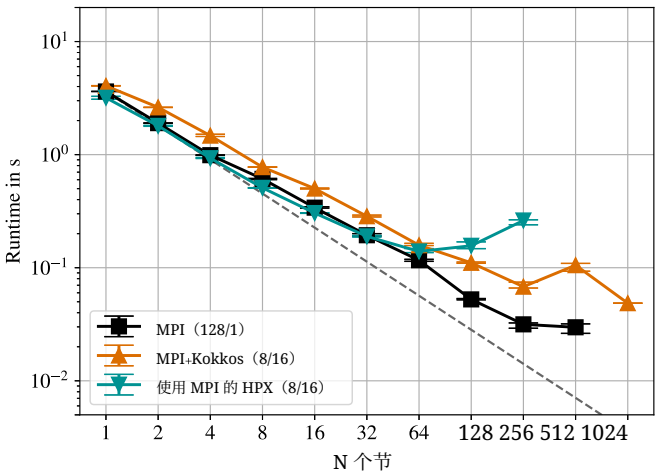


图 6. 使用 HARD 在最多 1024 个 Chicoma节点上运行三维纯流体动力学基准测试时，单次迭代的强扩展运行时间（见图 I）。问题规模固定为 $2^{9+9+9}$。虚线表示理想扩展。每个节点的运行配置以 ($\#process/\#threads$) 表示。

## VI. 结论与展望

在本研究中，我们使用迭代泊松求解器和辐射流体力学代码 HARD，在 Chicoma 超级计算机上最多使用 1024 个节点，大规模评估了 FleCSI 框架及其后端的性能与可扩展性。我们为 FleCSI 泊松示例开发了基准测试模式，使其能够进行异步计时，并便于准确比较不同后端的性能。借助 FleCSI 的后端架构，我们分析了两种常用异步多任务运行时相对于传统消息传递接口执行方式的运行时开销。

在泊松应用程序中，在我们的弱扩展基准测试中，FleCSI 在最多 131072 个核心上实现了超过 97% 的并行效率。将消息传递接口与 Kokkos 结合使用，由于分叉-汇合内核开销，性能略低；然而，FleCSI 凭借较低的通信成本展现出更强的可扩展性。在使用 HARD 的复杂辐射流体力学基准测试中，HPX 后端展现了其异步执行模型的优势。在最多 64 个节点上，HPX 后端在强扩展方面的性能最高可达 MPI+Kokkos 后端的 1.27 倍，在弱扩展方面平均可达 1.31 倍。对于

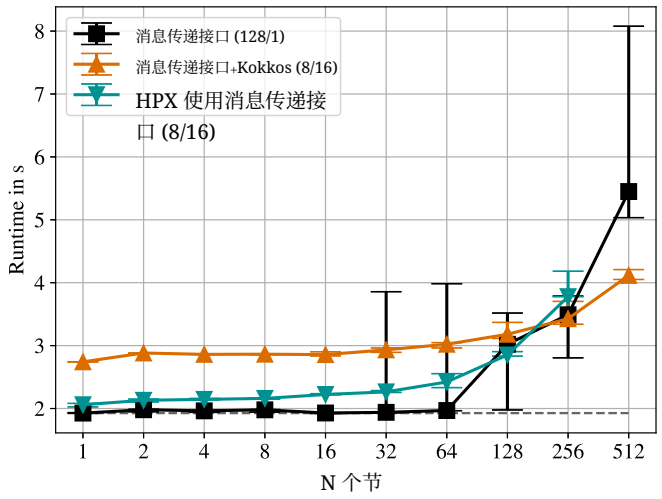


图 7. 在最多 512 个Chicoma节点上，使用 HARD 对三维辐射流体动力学基准测试进行一次迭代的弱扩展运行时间（见图 I）。每个节点的问题规模固定为 $2^{8+8+8}$。理想扩展以虚线表示。每个节点的运行配置以 (#*process*/#*threads*) 表示。

我们的纯流体动力学强扩展基准测试，HPX 后端相较于 MPI+Kokkos 的加速比最高达 1.64 倍，相较于纯 MPI 的加速比最高达 1.20 倍。

针对我们提出的研究问题，泊松基准测试结果表明，FleCSI 仅在传统的基于 MPI 的并行化之上引入了一个轻量级抽象层。与基于 MPI 的实现相比，两种异步多任务运行时都会产生可测量的运行时开销，并表现出扩展性限制。不过，HPX 运行时的开销低于 Legion 运行时。相比之下，HARD 基准测试结果凸显了异步多任务运行时在复杂科学模拟中的优势：计算与通信的异步交错可以提高资源利用率和整体性能。

未来仍有若干问题有待研究。尤其是，Legion 后端的性能还需要进一步分析。我们计划使用泊松应用程序对 Legion 后端进行性能剖析，以找出并量化运行时开销和内存使用量增加的原因。对于 HPX 后端的扩展性限制，我们的分析表明，主要瓶颈在于 HPX 集合通信的使用。因此，待优化的 HPX 集合通信可用后，我们打算更新扩展性基准测试。最后，我们计划将基准测试扩展到包含 GPU 加速器的异构架构。在这类环境中，AMTR后端通过重叠计算与 CPU-GPU 数据传输，可能比同步MPI后端带来更大的性能优势。

## 致谢

本研究中的基准测试是洛斯阿拉莫斯国家实验室（LANL）主办的协同设计暑期学校项目的一部分。作者特别想要感谢暑期学校主办方 J. Loiseau 和 H. Lim，以及 E. Maestas、C. Junghans、K. Thompson 和我们的导师 B. Bergen、R. Berger、P. Edelmann、B. Krueger、S. Lakshmiranganatha、A. Y. L´opez、M. Moraru、N. Prajapati、A. Reisner 和 T. Vogel。计算通过洛斯阿拉莫斯国家实验室的高级模拟与计算（ASC）项目，在 Chicoma 超级计算机上完成。该数据已获准公开发布，编号为 LA-UR-24-28836。

## 补充材料

FleCSI 代码可在 GitHub[1] 上公开获取。FleCSI 2.3 版支持使用消息传递接口、Legion 和 Kokkos 进行编译。本研究使用的 HPX 后端的确切实现并未纳入正式发布版本。不过，FleCSI 2.4 版提供了改进后的 HPX 后端。我们的辐射流体力学基准测试使用了 HARD 1.0 版，该版本可在 GitHub[2] 上公开获取。

我们的基准测试使用了修改版的泊松示例。该基准测试版本禁用了输入/输出，并允许对 Legion 和 HPX 后端进行异步性能计时。FleCSI 示例对应的 Git 补丁、包含所有已用软件版本的 Spack 环境、场景配置文件，以及用于强扩展和弱扩展的基准测试脚本均可在 DaRUS 获取[3]。

## 人工智能使用情况披露

我们使用了 Grammarly [49]、Writefull [50], 和 ChatGPT [51], 等生成式人工智能（AI）工具，以提升稿件的清晰度、语法和整体连贯性。所有技术内容、数据分析和研究发现均由作者独立构思并完成。作者仔细审阅、核实并编辑了 AI 辅助生成的内容，以确保事实准确、诠释严谨且符合学术诚信。最终稿件体现了作者原创的智力贡献和分析工作。

## 参考文献

[1] 消息传递接口论坛，*MPI*：消息传递接口标准 *3.0* 版，2021。[2] 英伟达公司，*CUDA C++* 编程指南，英伟达，2025年，版本 12.9。 [在线]。网址：https://docs.nvidia.com/cuda/ cuda-c-programming-guide/ [3] C. R. Trott、D. Lebrun-Grandi´e、D. Arndt、J. Ciesko、V. Dang、N. Elling- wood、R. Gayatri、E. Harvey、D. S. Hollman、D. Ibanez、N. Liber、J. Mad- sen、J. Miles、D. Poliakoff、A. Powell、S. Rajamanickam、M. Simberg、D. Sunderland、B. Turcksin 和 J. Wilke，“Kokkos 3：面向百亿亿次时代的编程模型扩展”，《*IEEE* 并行与分布式系统汇刊》，第 33 卷，第 4 期，第 805–817 页，2022。[4] B. Bergen、I. Demeshko、C. Ferenbaugh、D. Herring、L.-T. Lo、J. Loiseau、N. Ray 和 A. Reisner，“Flecsi 2.0：灵活计算科学基础设施项目”，载于 *Euro-Par 2021*：并行处理研讨会。Cham：施普林格国际出版公司，2022，第 480–495 页。

[1]https://github.com/flecsi访问日期：2026 10 2 01- -
https://github.com/lanl/HARD 访问日期：2026 10 3 01- -
https://doi.org/10.18419/DARUS-5193 访问日期：2026-03-03

[5] H. Kaiser、P. Diehl、A. S. Lemoine、B. A. Lelbach、P. Amini、A. Berge、J. Biddiscombe、S. R. Brandt、N. Gupta、T. Heller、K. Huck、Z. Khatami、A. Kheirkhahan、A. Reverdell、S. Shirzad、M. Simberg、B. Wagle、W. Wei 和 T. Zhang，“HPX——用于并行性与并发的 C++标准库”，开源软件期刊，第 5 卷，第 53 期，第 2352 页，2020。 [在线]。网址：https://doi.org/ 10 21105 02352 . /joss。M. Bauer、S. Treichler、E. Slaughter 和 A. Aiken，“Legion：利用逻辑区域表达局部性与独立性”，载于 SC ’12：高性能计算、网络、存储与分析国际会议论文集，2012，第 1–11 页。[7] J. Loiseau、H. Lim、A. Yag¨ue L´opez、M. Baghirzade、S. S. Khan、Y. Kim、S. Neopane、A. Strack、F. Taiyebah 和 B. Bergen，“HARD：基于 FleCSI 框架的可移植高性能辐射流体力学代码”，SoftwareX，第 32 卷，第 102441 页，2025年。 [在线]。网址：https://www.sciencedirect.com/science/article/pii/S 2352711025004078 B. Bergen、N. Moss 和 M. R. J. Charest，“灵活计算机科学基础设施（FleCSI）”，2016 年 4 月。 [在线]。网址：https: //www.osti.gov/biblio/ 1311634 J. Holmen、D. Sahasrabudhe 和 M. Berzins，“面向异步多任务运行时系统的异构 MPI+PPL 任务调度方法”，载于《2021 年先进研究计算实践与体验：可持续性、成功与影响》（PEARC21）会议论文集。ACM，2021。[10] M. P´erache、P. Carribault 和 H. Jourdren，“MPC-MPI：一种降低总体内存消耗的 MPI 实现”，载于 EuroPVM/MPI 2009，丛书《计算机科学讲义》，M. Ropo、J. Westerholm 和 J. Dongarra 编，第 5759 卷。海德堡：施普林格，2009，第 94–103 页。[11] D. Boehme、T. Gamblin、D. Beckingsale、P.-T. Bremer、A. Gimenez、M. LeGendre、O. Pearce 和 M. Schulz，“Caliper：高性能计算软件栈的性能内省工具”，载于高性能计算、网络、存储与分析国际会议（SC16）论文集，2016，第 550–560 页。[12] J. Loiseau、H. Lim、M. A. Kaltenborn、O. Korobkin、C. M. Mauney、I. Sagert、W. P. Even 和 B. K. Bergen，“Flecsph：面向光滑粒子流体动力学的新一代灵活并行计算基础设施”，SoftwareX，第 12 卷，第 100602 页，2020。[在线]。网址：https://www.sciencedirect.com/science/article/pii/ S2352711020303150[13] S. D. Herring、M. Moraru、S. Pakin、J. Loiseau、R. Berger、P. V. F. Edelmann 和 B. Bergen，“通过 FleCSI 增强 HPX：自动检测隐式任务依赖”，载于高性能计算、网络、存储与分析国际会议 SC ’25 研讨会论文集，丛书 SC Workshops ’25。美国纽约州纽约：美国计算机协会，2025，第 1330–1340 页。[在线]。网址：https://doi.org/ 10 1145 3731599 3767507 . / 。[14] C. D. Levermore 和 G. C. Pomraning，“通量限制扩散理论”，天体物理学报，第 248 卷，第 321–334 页，1981 年 8 月。[15] N. J. Turner 和 J. M. Stone，“使用通量限制扩散的 ZEUS-2D 辐射流体力学计算模块”，《天体物理学报增刊》，第 135 卷，第 95–108 页，2001。[16] P. Thoman、K. Dichev、T. Heller、R. Iakymchuk、X. Aguilar、K. Hasanov、P. Gschwandtner、P. Lemarinier、S. Markidis、H. Jordan、T. Fahringer、K. Katrinis、E. Laure 和 D. S. Nikolopoulos，“高性能计算任务型并行编程技术分类体系”，《超级计算期刊》，第 74 卷，第 4 期，第 1422–1434 页，2018 年 4 月。[17] J. Schuchart、P. Diehl、M. Bauer、A. Bouteiller、G. Daiss、E. Kayrak- lioglu、S. Khandekar、T. Herault、J. Holmen、R. Rao、A. Strack、E. Schlaughter、J. Spinti、J. Thornock、A. Aiken、O. Aumage、M. Berzins、G. Bosilca、B. L. Chamberlain 和 L. Kale，“分布式异步多任务模型及其应用综述”，Authorea 预印本，2025 年 12 月，预印本。[18] E. Slaughter、W. Wu、Y. Fu、L. Brandenburg、N. Garcia、W. Kautz、E. Marx、K. S. Morris、Q. Cao、G. Bosilca、S. Mirchandaney、W. Leek、S. Treichlerk、P. McCormick 和 A. Aiken，“Task Bench：用于评估并行运行时性能的参数化基准测试”，载于 SC20：高性能计算、网络、存储与分析国际会议，2020，第 1–15 页。[19] N. Wu、I. Gonidelis、S. Liu、Z. Fink、N. Gupta、K. Mohammadipor- shokooh、P. Diehl、H. Kaiser 和 L. V. Kale，“量化 Charm++ 和 HPX 使用 Task Bench 时的开销”，载于 Euro-Par 2022：并行处理研讨会。施普林格·自然瑞士，2023，第 5–16 页。

[20] E. Gabriel、G. E. Fagg、G. Bosilca、T. Angskun、J. J. Dongarra、J. M. Squyres、V. Sahay、P. Kambadur、B. Barrett、A. Lumsdaine、R. H. Castain、D. J. Daniel、R. L. Graham与T. S. Woodall，“Open MPI：下一代消息传递接口实现的目标、概念与设计”，载于第*11*届欧洲*PVM/MPI*用户组会议论文集，D. Kranzlm¨uller、P. Kacsuk与J. Dongarra编，匈牙利布达佩斯，2004年9月，第97–104页。[21] W. Gropp、E. Lusk与A. Skjellum，《使用*MPI*：消息传递接口的可移植并行编程》，第2版，麻省理工学院出版社，1999年。[22] Cray公司，“Cray MPICH”，https://cpe.ext.hpe.com/docs/24.03/mpt/ mpich/index.html#mpich，2025年，8.1.26版。访问日期：2025年7月14日。[23] S. Treichler、M. Bauer与A. Aiken，“Realm：面向分布式内存架构的基于事件的底层运行时”，载于*2014*年第*23*届并行架构与编译技术国际会议（*PACT*），2014年，第263–275页。[24] D. Bonachea与P. H. Hargrove，“GASNet规范，v1.8.1”，劳伦斯伯克利国家实验室，技术报告LBNL-2001064，2017年8月，https://doi.org/10.2172/1398512。[25] P. Shamis、M. G. Venkata、M. G. Lopez、M. B. Baker、O. Hernandez、Y. Itigin、M. Dubman、G. Shainer、R. L. Graham、L. Liss、Y. Shahar、S. Potluri、D. Rossetti、D. Becker、D. Poole、C. Lamb、S. Kumar、C. Stunkel、G. Bosilca与A. Bouteiller，“Ucx：面向高性能计算网络API及其他的开源框架”，载于*2015*年*IEEE*第*23*届高性能互连年度研讨会，2015年，第40–43页。[26] E. Slaughter、W. Lee、S. Treichler、M. Bauer与A. Aiken，“Regent：一种采用逻辑区域、面向高性能计算的高生产力编程语言”，载于高性能计算、网络、存储与分析国际会议论文集，丛书SC ’15。美国纽约州纽约：美国计算机协会，2015年。 [在线]。可获取：https://doi.org/ 10 1145 2807591 2807629 。 / 。C. Ferenbaugh，“Pennant：面向先进架构研究的非结构网格迷你应用”，《并发与计算：实践与经验》，第27卷，2014年10月。[28] S. Mirchandaney、A. Aiken与E. Slaughter，“《谈Pygion》：编写百亿亿级单粒子成像代码的经验”，载于《异步多任务系统与应用》，P. Diehl、J. Schuchart、P. Valero-Lara与G. Bosilca编。Cham：施普林格·自然瑞士，2024年，第1–8页。[29] H.-Y. Chang、E. Slaughter、S. Mirchandaney、J. Donatelli与C. H. Yoon，“利用SpiniFEL扩展并加速单粒子成像实验的三维结构测定”，2021 [ ] 2109 05339 。在线。可获取：https://arxiv.org/abs/ 。T. Heller，“利用HPX运行时系统将C异步编程模型扩展至分布式内存计算”，博士学位论文，FAU Erlangen-N¨urnberg，2019年。[31] V. G. Cerf与Y. K. Dalal，“互联网传输控制程序规范”，RFC 675，1974年。[32] J. Yan、H. Kaiser与M. Snir，“异步多任务系统网络软件栈的设计与分析——HPX的LCI parcelport”，载于*SC ’23*研讨会论文集。美国纽约州纽约：ACM，2023年，第1151–1161页。[33] A. Strack与D. Pfl¨uger，“HPX通信基准测试：使用集合通信的分布式快速傅里叶变换”，载于*Euro-Par 2024*：并行处理研讨会：*Euro-Par 2024*国际研讨会，西班牙马德里，*2024*年*8*月*26*日至*30*日，论文集，第二部分。柏林，海德堡：施普林格出版社，2025年，第271–274页。 [在线]。可获取：https://doi.org/ 10 1007 978 3 031 90203 1 25 。 / - - - - P. Diehl、G. Daiß、K. Huck、D. Marcello、S. Shiber、H. Kaiser与D. Pfl¨uger，“在富岳超级计算机的A64FX上使用HPX/Kokkos模拟恒星并合”，2023年3月。[35] P. Diehl、G. Daiss、S. Brandt、A. Kheirkhahan、H. Kaiser、C. Taylor与J. Leidel，“使用天体物理学应用Octo-Tiger在RISC-V上评估HPX与Kokkos”，载于*SC-W 2023*。美国纽约州纽约：美国计算机协会，2023年，第1533–1542页。[36] A. Strack、C. Taylor与D. Pfl¨uger，“RISC-V上的并行FFTW：涵盖OpenMP、消息传递接口与HPX的比较研究”，载于高性能计算，S. Neuwirth、A. K. Paul、T. Weinzierl与E. C. Carson编。Cham：施普林格·自然瑞士，2026年，第586–597页。[37] G. Daiß、S. Y. Singanaboina、P. Diehl、H. Kaiser与D. Pfluger，“从框架合并到恒星并合：使用HPX、Kokkos与SIMD类型的经验”，载于*2022*年*IEEE/ACM*第七届国际研讨会：

极端规模编程模型与中间件（*ESPM2*）。IEEE，2022年11月。[38] G. Daiß、P. Diehl、H. Kaiser与D. Pfl¨uger，“使用HPX-Kokkos与SYCL的恒星并合：将异步多任务运行时系统与SYCL结合使用的方法”，载于*2023*年*OpenCL*国际研讨会论文集，丛书IWOCL ’23。美国纽约州纽约：美国计算机协会，2023年。[39] D. C. Marcello、S. Shiber、O. D. Marco、J. Frank、G. C. Clayton、P. M. Motl、P. Diehl与H. Kaiser，“Octo-Tiger：一种采用HPX并行化、用于恒星并合的新型三维流体动力学代码”，皇家天文学会月报，第504卷，5382，2021年4月4，第5345期，页–。[40] M. Helmann、A. Strack与D. Pfl¨uger，“Gprat：基于异步任务的高斯过程回归”，载于《异步多任务系统与应用》，P. Diehl、Q. Cao、T. Herault与G. Bosilca编。Cham：施普林格·自然瑞士，2026年，第83–94页。[41] A. Strack、C. Taylor、P. Diehl与D. Pfl¨uger，将共享与分布式应用移植到异步任务的经验：多维*FFT*案例研究。施普林格·自然瑞士，2024年，第111–122页。[42] Y. Saad，《稀疏线性系统的迭代方法》，第2版。美国：工业与应用数学学会，2003年。[43] R. Borges、M. Carmona、B. Costa与W. S. Don，“一种改进的用于双曲型守恒律的加权本质无振荡格式”，计算物理杂志，第227卷，第6期，第3191–3211页，2008年。[44] A. Harten、P. D. Lax与B. v. Leer，“论双曲型守恒律的迎风差分与Godunov型格式”，*SIAM*评论，第25卷，第1期，第35–61页，1983年。[45] V. Rusanov，“非定常激波与障碍物相互作用的计算”，《苏联计算数学与数学物理》，第1卷，第2期，第304–320页，1962年。[46] W. L. Briggs、V. E. Henson与S. F. McCormick，《多重网格教程（第二版）》，第2版。工业与应用数学学会，2000年。[47] W. J. M. Rankine，“论有限纵向扰动波的热力学理论”，伦敦皇家学会哲学汇刊，第160卷，第277–288页，1870年。[48] H. Hugoniot，“M´关于运动在无界流体ind´中传播的论文（premi`第一部分）”，数学期刊´纯粹与应用数学´，第*4*系´列，第3卷，第477–492页，1887年。[49] Grammarly, Inc.，“Grammarly”，https://www.grammarly.com/，2026年，访问日期：2026年1月14日。[50] Digital Science，“Writefull”，https://www.writefull.com，2025年，访问日期：2026年1月14日。[51]OpenAI，“ChatGPT 5”，https://openai.com/chatgpt，2025年，访问日期：2026年1月14日。