面向持续交互对象的通信感知扩散负载均衡

Maya Taylor，Kavitha Chandrasekar，Laxmikant V. Kale伊利诺伊大学厄巴纳-香槟分校计算机科学系美国伊利诺伊州香槟市{mayat4、kchndrs2、kale}@illinois.edu

arXiv:2603.23329v1 [cs.DC] 24 Mar 2026

摘要——**具有不规则且随时间变化的工作负载的并行应用常常会遭遇负载不均衡。动态负载均衡技术通过在执行过程中重新分配工作来应对这一挑战。我们提出一种新型的分布式扩散式负载均衡，专为具有持续通信对象的通信密集型应用而设计。利用应用的通信图，我们的策略在有效分配负载的同时减少跨节点通信。对于无法轻易获取通信模式的情况，我们还提出了一种算法变体。我们通过模拟以及在 Perlmutter（NERSC）上最多使用 8 个节点运行粒子-网格基准测试，探讨了该算法的优化方案，并将其与其他相关负载均衡策略进行了比较。**

索引词—**动态负载均衡、扩散方法、并行处理、通信感知算法、不规则应用**

I. 引言

不规则并行性是许多科学工作负载的典型特征，涵盖粒子模拟、自适应网格细化和图分析等应用。这些应用的负载不均衡和通信模式会随时间变化，二者都带来性能挑战。尤其是在强扩展极限下，每个处理器承担的工作量较少，通信成本占据主导地位，此时即使是轻微的负载不均衡或通信局部性遭到破坏，也会显著降低性能。

为应对这些挑战，许多系统依赖运行时负载均衡。我们尤其关注采用基于块或“过度分解”执行模型的运行时系统；在这种模型中，计算被划分为更多的工作与数据单元，而非对象，处理器。这些对象可以动态迁移，使系统能够随着应用程序的发展重新分配工作。

本材料所述工作得到美国能源部科学办公室先进科学计算研究办公室能源部计算科学研究生奖学金的支持，资助编号为 DE-SC0025528。

本报告是对美国政府某机构资助的工作所作说明。美国政府及其任何机构，或其任何雇员，均不对所披露的任何信息、设备、产品或过程的准确性、完整性或实用性作出任何明示或暗示的保证，也不承担任何法律责任或义务，亦不声明使用这些信息、设备、产品或过程不会侵犯私人所有权利。本文提及任何特定商业产品、过程或服务时，无论使用其商标名称、商标、制造商名称或其他方式，均不一定构成或暗示美国政府或其任何机构对其予以认可、推荐或偏好。本文作者所表达的观点和意见不一定代表或反映美国政府或其任何机构的立场。

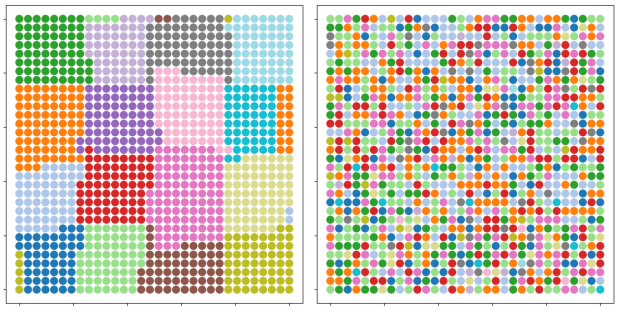


图 1：合成二维模板应用的负载可视化。圆圈表示可迁移对象，颜色表示所属处理器。这些图由我们的负载均衡模拟基础设施生成，展示了扩散方法（左）和贪心细化策略（右）。

在这一模型下，我们提出了一种受基于扩散的技术启发的新型分布式负载均衡算法。我们的方法面向 Charm++ 运行时系统的过度分解模型 [1]，，但也广泛适用于任何支持运行时迁移的基于块的编程框架，包括自适应网格细化代码中使用的框架，以及 DARMA/vt 运行时[2]等系统。该算法通过在已有通信的节点之间迁移或扩散工作单元来平衡负载。通过使负载均衡决策与现有通信模式相协调，该方法在实现负载分布均衡的同时保留了通信局部性。

为了直观地理解我们的扩散方案，我们使用一个示例性二维模板基准测试及其相应的可视化。考虑这样一个应用：一个由对象构成的二维网格（我们将使用 Charm++ 术语称其为 *chares*）参与迭代五点模板算法。每个二维网格点都根据自身状态及其四个紧邻邻居（北、南、东、西）的状态更新其值。暂且假设每个网格点都分配给各自的 chare。虽然我们将使用这个模板框架作为理解和模拟负载均衡的简单工具，

模板也是科学应用中非常常见的计算模式，包括有限差分求解器和波传播模型。

图 1 展示了模板应用的两种可视化结果，以帮助直观理解通信局部性。在左图中，单元及其邻居经常位于同一处理器上，因此形成连续的色块。这代表良好的通信局部性，因为大多数邻居之间的通信都是本地的。右图则呈现了相反的情况：对象随机分散，通信局部性完全遭到破坏。

我们提出的负载均衡方案旨在通过在整个负载均衡过程中维持通信局部性，防止这种破坏。我们介绍了算法的两种变体，在模拟环境和粒子网格（PIC）基准测试中对其进行评估，并指出了该方法的优势及进一步研究的机会。

II. 相关工作与问题定义

在并行计算中，针对动态不规则并行，存在若干不同类别的负载均衡问题。两种常见的研究表述是：（1）简单的工作列表表述，其中每个处理器持有一组独立任务，每个任务都关联一个计算负载 [3,4],；（2）分治计算，其中任务生命周期很短，但在执行期间会不断生成 [5,6]。这两类问题都已得到广泛研究，其中包括许多受扩散过程启发的方法 [7]–[9]。

然而，这些公式通常假设工作项之间不会相互通信（分治场景中有限的父子交互除外）。相比之下，我们关注这样一类情况：一组长生命周期对象在整个执行过程中持续存在，并反复与少数其他对象通信。

问题定义： 给定一组相互交互的对象，每个对象都有当前的处理器分配和相应的计算负载，以及一个由加权通信边构成的稀疏图，用于表示对象之间的交互。目标是计算对象到处理器的新映射。我们通过若干成本指标评估该映射：（1）负载不均衡，以处理器最大负载与平均负载之比衡量（其中，负载以对象内耗费的运行时间近似表示）；（2）通信成本，以节点间通信量与节点内通信量（字节数）之比衡量；（3）迁移成本，即将对象迁移到新处理器所产生的开销；（4）计算映射本身的成本。

针对这些具体约束，我们的工作与解决前文所述类别问题的研究有很大不同，其中包括大多数现有的基于扩散的方法。主要的例外是近期将通信感知融入分布式负载均衡的研究，它解决了相关约束，但没有采用扩散 [10,11]。有趣的是，METIS 和 ParMETIS 等传统图划分工具也可用于此场景中的负载均衡，尤其是在存在显式通信图的情况下 [12,13]。尤其是 ParMETIS 支持一种“重新划分”算法，可在运行时迭代划分对象通信图，同时将迁移降至最低，并保持通信局部性。Lieber 等人 [14] 全面概述了多种扩散策略，并将其与图划分策略进行比较，得出的结论是，扩散能够带来显著益处，尤其是在减少迁移和通信，以及缩短负载均衡策略耗时方面。

我们在大量既有工作的基础上，运用扩散的基本原理，为过度分解生态系统中的通信密集型应用开发一种负载均衡策略。我们在实现节点间负载均衡的同时，尽量减少迁移并保持通信局部性。

III. 三阶段通信感知扩散

我们的算法包括：（1）根据应用的通信模式构建节点邻居图；（2）在不考虑对象粒度的情况下计算理想的负载再分配方案；（3）选择要迁移的具体对象。下面将详细讨论这些阶段。

关键在于，我们注意到，像我们这样采用增量式优化的策略，假设初始对象分布具有良好的通信局部性。如果无法保证这一点，可以先执行一次初始图划分负载均衡（例如使用 METIS），以形成这种局部性。此外，像我们这样的自适应运行时策略，假设负载均衡迭代期间负载和通信模式具有一定的持续性。

A. 邻居选择

我们的算法与以往工作的不同之处在于，初始图是根据通信模式而非网络拓扑构建的 [14]。邻居选择阶段会隐式构建节点通信图，其中的边决定了哪些节点之间可以发生迁移。

图中的每个顶点代表一个计算节点，邻居则根据节点间的通信情况进行选择。所需的邻居数量，即顶点度数，是一个可调参数，可根据应用场景进行设置，后文将对此进行说明。假设 K 是所需的顶点度数；随后通过以下迭代流程选择邻居。

1) 每个节点计算 l，即其达到所需 K还需要的邻居数量。
2) 每个节点按通信量从高到低对邻居排序，并向排名最靠前的 $l/2$发送请求。此处使用 $l/2$ ，以避免过多的邻居请求在系统中传递。3) 节点收到邻居请求后进行检查：a) 已确认的邻居数量，以及该数量是否达到 K；b) 已确认的邻居数量加上额外的保持 数量是否已达到 K

如果满足其中任一条件，节点就会拒绝收到的请求。否则，它会回复一条

接受消息，并递增预留量，为潜在邻居预留空间。4) 收到接受消息后，原请求节点必须确认其邻居数量和预留量在此期间均未超过 K。如果没有超过，该节点便接受这个传入的邻居，并发送最终确认消息。由此便完成一组已确认的邻居配对。5) 重复上述过程，直到所有节点各自拥有 K 个已确认的邻居，或达到迭代次数上限。

上述过程保证每个节点最多有 K 个邻居，前提是它至少与 K 个节点通信。K 参数可在运行时调整，并由用户提供。第 V 节探讨了 K 不同取值之间的权衡。在当前实现中，每个负载均衡阶段都会重建邻居图，但未来工作应探索邻居图的复用，因为大规模节点间通信模式很可能会在多次负载均衡迭代中持续存在。

B. 虚拟负载均衡

负载均衡的第二阶段是“虚拟”负载均衡，在此阶段只交换负载量。该阶段使用迭代不动点算法，根据上一阶段施加的邻域约束，计算理想的负载再分配。各节点反复与邻居交换负载量，直到收敛到一种负载分布，使每个邻域内的负载方差都低于预设阈值。在此阶段结束时，每个节点都会获得一份邻居列表，以及其应向每个邻居发送或从每个邻居接收的负载量。

此阶段使用的迭代方法，是对先前工作 [3,15], 中所述算法的一种简单一阶收敛实现的变体，但有一项重要限制：只允许“单跳迁移”，即负载只能从其起始处理器移动一跳。未来工作可以探索多跳迁移，但这会增加算法其余部分的复杂度。

C. 对象选择

虚拟负载均衡阶段计算出理想的负载转移后，必须选择具体对象进行迁移。我们的对象选择策略优先考虑通信局部性，并遵循以下约束：

- 对于给定的邻居 n，按照上一负载均衡阶段中与 n 通信的字节数，从多到少依次迁移对象。
- 选定对象 o 进行迁移后，所有与 o 通信的对象都必须更新其通信模式，以纳入与对象 o的新驻留处理器之间的通信。

这种约束组合会尽可能迁移那些已经与现有邻居通信的对象，从而保持通信局部性。第二项要求尤为重要，因为处理器迁移的对象数量可能超过其最初与特定邻居通信的对象数量。

D. 层次化负载均衡

为简便起见，本文其余部分将上述三个阶段称为以进程为粒度的“节点”操作。我们的实现还支持进程内负载均衡：对先前计算出的迁移模式进一步细化，以便在进程内的所有线程之间均衡负载。直到这一步（含这一步），对象迁移都通过代理令牌进行；只有该阶段完成后，所有对象才会实际迁移到新的目的地。与跨进程负载均衡相比，进程内负载均衡的算法简单得多，只考虑负载，而不考虑通信模式。为了能在无需分配数百个节点的情况下研究大规模场景中的策略，我们在研究中采用每个核心运行一个进程的配置。

IV. 坐标变体

如果无法直接获取通信图，我们提出一种基于对象坐标近似通信模式的方法。这种近似要求应用开发者能够将对象映射到逻辑位置，使距离倒数与通信相关。这在具有物理表示和通信局部性的应用中尤其可行，例如粒子模拟。

二维模板应用是理解坐标变体的一个有用示例，正是因为通信模式与 chare 坐标密切相关。在应用层，我们为每个 chare 分配二维坐标，对应其所负责网格部分的位置，然后算法假设彼此相邻的 chares 也会相互通信。

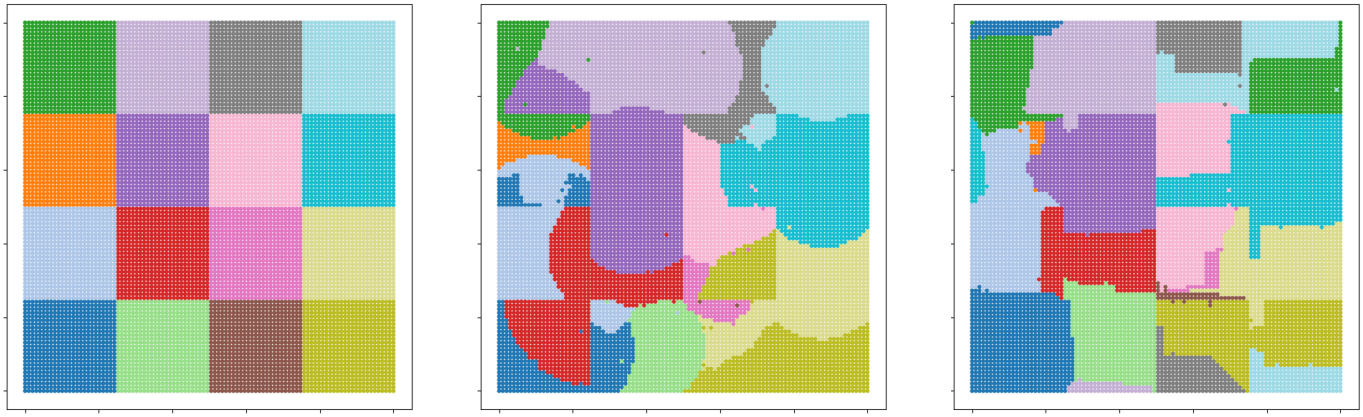
为了在缺少通信图的情况下有效使用这些坐标，我们对感知通信的扩散算法进行以下修改：

- 在初始化期间，每个处理器都会计算其**质心**，即其所有 chares 位置的平均值。
- 在选择邻居时，会分发质心并加以利用，使所有处理器都按质心距离倒数对其他处理器排序。请注意，这种方法的可扩展性远不如基于通信的方法，因为节点必须对所有处理器排序，而不只是与其通信的处理器。未来工作可以解决这一局限，详情见第 VII 节。
- 对象选择依据对象与相邻处理器质心之间的距离，而非通信模式。与基于通信的变体类似，对象迁移时会更新处理器质心。

需要指出的是，坐标感知变体策略只是一种近似。评估部分将讨论这些近似对性能的影响。

V. 模拟评估

为了在受控、可复现且高效的环境中评估通信感知和坐标感知的扩散策略，我们使用 Charm++ 实现了一个负载均衡模拟基础设施。该基础设施需要输入以下信息的描述：



(a) 原始分布：最大负载与平均负载比为 1.74，外部与内部通信比为 .043。

(b) 基于坐标的扩散：最大负载与平均负载比为 1.02，外部与内部通信比为 .072。

（c）基于通信的扩散：最大负载与平均负载比为 1.04，外部与内部通信比为 0.06。

图 2：使用 16 个处理器和初始分块分解的二维模板基准测试中的对象迁移。通过合成方式引入负载不均衡：将每个对象的负载随机增加或减少 40%。两种策略都使用 4 个邻居。

对象负载、坐标和通信边；在负载均衡步骤中，可以轻松为任何 Charm++ 应用生成这些信息。获得这些信息后，模拟器便可独立于应用对各种负载均衡策略进行建模，并模拟任意数量的进程，而无需进行真正的大规模执行。模拟器利用 Charm++ 基于对象的分解机制，在单个进程上模拟大规模负载均衡问题，其中每个 chare 都代表一个进程，从而能够探索不同的不平衡模式和迁移决策。我们还使用该模拟器生成图 1 和图 2 中的可视化结果，以便提供直观的理解。

A. 基于坐标的扩散与基于通信的扩散

图 2 直观比较了我们的基于通信和基于坐标的扩散算法，在具有合成负载不平衡的二维模板应用中，经过一次负载均衡迭代后的表现。通常，基于坐标的再平衡会形成更圆润的边界，这是因为迁移度量由对象到处理器质心的距离决定。基于通信的再平衡能更好地保留原有的处理器域形状，从而形成更锐利的直线边缘。

在我们当前的实现中，只有基于通信的方法能够捕捉模板应用中的周期性边界。未来工作可以探索将周期性边界支持整合到坐标方法中，但目前该方案更多受限于模拟边界。这会导致邻居选择及后续迁移不够理想，例如图 2b 中深蓝色左下角处理器的划分。当这个深蓝色处理器搜索距离最近的四个邻居时，它会选择橙色处理器（位于模拟的左边界），而不是跨越周期性边界的真正通信邻居。

由于坐标方案偶尔会出现此类近似，这种方法对通信局部性的保持不如通信方法。为了总体量化通信局部性，我们考察外部 与内部通信 指标，该指标衡量处理器之间与处理器内部通信的字节数。以图 2 为例，坐标方法产生的比率为 0.072，高于原始扩散方法的外部通信量，其比率为 0.06。

B. 邻居数量选择

要有效使用扩散算法，用户必须选择合适的邻居数量 K。直观而言，邻居数量较少会限制迁移选项，因为节点只能与其直接相邻的邻居交换负载，这可能会降低负载均衡效果。邻居数量较多会增加迁移选项，但计算成本可能更高；此外，如果邻居相距过远，还会破坏通信局部性。

为直观展示这种权衡，考虑一个合成的模板应用实例：处理器构成一维环，其中一个处理器的负载过重，达到其他处理器的 10 倍。初始模板分布存在负载不均衡，最大负载与平均负载比约为五。采用不同邻居数量运行扩散负载均衡后，表 I 展示了所得的负载分布。邻居数量为 1 时，过载处理器无法充分扩散其负载；随着邻居数量增加，负载均衡有所改善。然而，增加邻居数量也会提高外部与内部通信比，因为更多邻居意味着可以进行距离更远的迁移。例如，某个节点可能会选择将对象迁移到一个与其没有通信的邻居处，以尝试分散负载。

邻居数量	1	2	4	8
最大/平均负载	4.9	1.7	1.3	1.1
外部/内部通信（MB）	.142	.151	0.25	0.26

表 I：邻居数量对单个过载处理器的合成模板应用中负载均衡质量的影响。

C. 模拟中与其他策略的比较

我们将扩散策略与模拟基础设施中实现的其他几种负载均衡算法进行比较，包括第 II 节讨论的 GreedyRefine、METIS 和 ParMETIS。Charm++ 已支持 GreedyRefine 和 METIS，因此很容易将它们接入我们的模拟基础设施。我们还评估了基于 MPI 的 ParMETIS 库。该库是 METIS 的一种分布式替代方案，支持自适应重新分区；但出于第 VI 节所述的原因，我们仅在模拟中对其进行评估。

表 II 总结了在三个规模递增的合成基准测试中进行的比较结果，这些测试均基于前文所述的二维模板应用。GreedyRefine 始终能取得最佳的最大/平均负载比，但对通信局部性的影响最差。METIS 能够实现良好的通信局部性，但会导致数量极大的迁移。通信感知扩散策略旨在这些极端情况之间寻求折中。

在此背景下评估 ParMETIS颇具挑战，原因在于其参数化设置，尤其是需要调整迁移量与保留原始分区程度之间的权衡。在这些实验中，我们选择的参数值使迁移数量与扩散策略处于相同范围内，但在一般应用场景中，如此程度的参数探索并不切实际。这些权衡的复杂性在规模最大的基准测试中尤为明显：要实现更好的负载均衡，就必须付出迁移数量大幅增加的代价。尽管 ParMETIS 是多个对比基线之一，这些观察结果仍凸显了动态负载均衡场景中采用自适应图分区策略所面临的实际挑战。

VI. 性能

为评估通信感知扩散负载均衡的质量，我们使用 Charm++ 实现了并行研究内核（PRK） [16,17] 中介绍的、受 PIC 启发的基准测试。

此 PIC PRK 基准测试旨在评估并行运行时系统的负载均衡能力。它模拟具有周期性边界的二维网格，并在每个网格点固定电磁荷。粒子通过多种不同的初始化模式分布在整个网格中；在每个时间步，粒子都会与附近的网格电荷相互作用，计算作用于粒子的库仑力，并据此更新其状态。该模拟经过构造，使得任意粒子在一个时间步后的水平位移均确定为 $(2k+1)$ 个网格单元，其中 k 是用户定义的参数。此特性确保负载不均衡模式随时间以可预测的方式演变，因此该基准测试适用于负载均衡评估。PIC PRK 并非完整的 PIC 模拟，而是侧重于与负载均衡最相关的算法组件。

我们实现了 PIC PRK 基准测试的 Charm++ 版本：该版本将网格划分为单元，并通过基本的二维分解将这些单元分配给各个 chare。它支持现有 PIC PRK 实现的所有相关使用模式（包括可变网格大小、粒子数量和粒子分布模式），并额外提供用于指定 chare 数量和负载均衡频率的参数。我们评估了 Charm++ PIC PRK 在无负载均衡、GreedyRefine Charm++ 策略以及扩散策略下的性能。这里不考察并行图划分方案 ParMETIS，因为 Charm++ 不支持该策略。ParMETIS 是一个 MPI 库，而 Charm++ 使用 UCX 等自身的底层通信层，因此两者不兼容。使用基于 MPI 的 Charm++ 版本可以解决这一问题，但通信成本会不合理地增加。

A. PIC PRK 负载不均衡模式

PIC PRK 基准测试支持多个可调参数，这些参数会影响负载不均衡模式，包括初始粒子分布的偏斜度、粒子的水平速度以及粒子的垂直速度。本节将探讨我们对这些参数的选择，以更好地了解应用中的负载不均衡行为，以及负载均衡策略的预期性能。

初始粒子分布： PIC PRK 支持多种初始粒子分布，详见 [17]。我们使用“GEOMETRIC”分布模式，该模式根据指数分布将粒子放置在网格上。在一个 $c \times c$网格中，第 i 列网格单元中的粒子数量由 $A\rho^i$ 给出，其中 A 是归一化常数， ρ 是用户定义的偏斜参数。在每一列中，粒子均匀随机地放置在各行中。 ρ 的值越高，分布越偏斜，越多粒子集中在最左侧的列中。了解初始分布以及前文所述的依赖于 k 的水平位移特性后，我们便可以完整刻画随时间推移将出现的负载不均衡模式。

粒子速度： 水平速度参数 k 决定粒子在每个时间步内水平移动的距离，具体为（$2k+1$）个网格单元。较高的 k 会使粒子在网格中移动得更快，因此确定负载均衡频率时必须谨慎考虑 k 。我们将垂直速度设为每个时间步移动 1 个网格单元，并忽略垂直维度上的负载不均衡，因为粒子沿该轴均匀分布。

在我们的评估中，我们使用 100，000 个粒子，在 1000 × 1000 的网格上运行 100 个时间步，使用 $k = 2$ 和 $\rho = .9$，以及由 12 × 12 个 chare 组成的网格。每个 chare 覆盖一个子网格，其大小为

指标	初始	GreedyRefine	METIS	ParMETIS	通信差异	坐标差异
基准测试 1：8 个处理单元						
最大/平均负载	1.32	1.00	1.00	1.01	1.06	1.05
外部/内部通信（MB）	.5	.887	.343	.637	.58	.648
迁移百分比	-	19.9%	87.1%	14.6%	18.9%	17%
基准测试 2：32 个处理单元						
最大/平均负载	1.37	1.00	1.00	1.04	1.02	1.02
外部/内部通信（MB）	.143	.422	.158	.194	.173	.227
% 迁移	-	18.7%	98.9%	6.6%	15.4%	17.6%
基准测试 3：128 个处理单元						
最大/平均负载	1.37	1.00	1.00	1.20	1.09	1.14
外部/内部通信 (MB)	.6	.999	.217	.602	.625	.616
迁移百分比	-	18.8%	99%	10.8%	18.2%	17.8%

表 II：比较 5 种算法和采用三维模板通信模式的合成基准测试中的负载平衡指标。所有基准测试均采用相同的合成负载不平衡注入方式：每 7 个处理单元中，第 1 个和第 2 个过载，第 3 个负载不足。

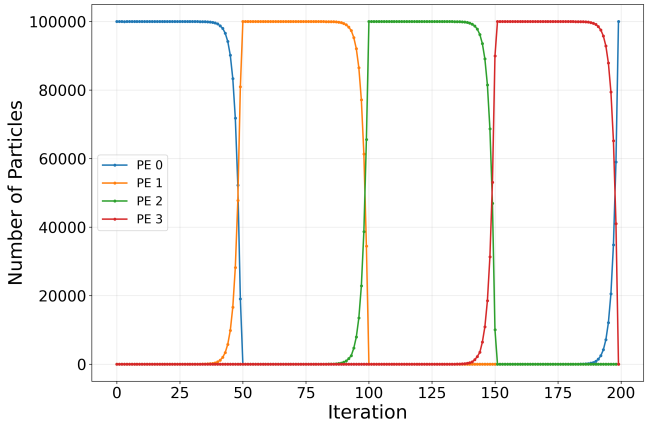


图 3：在 PIC PRK 基准测试中，使用 $k = 2$ 和 $\rho = .9$时，粒子分布随时间在处理器间的演变。

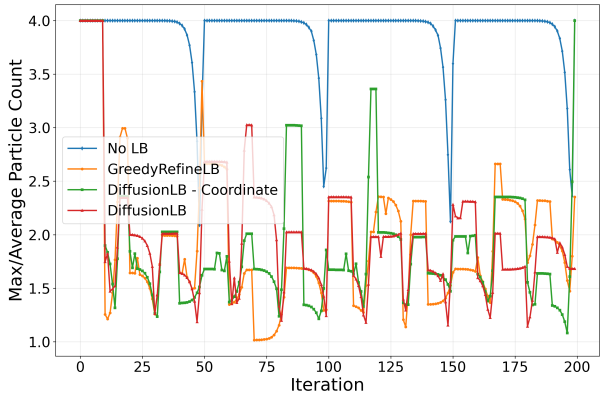


图 4：在 PIC PRK 基准测试中，使用 $k = 2$和 $\rho = .9$ 以及 4 个处理器时，每个处理器上的最大粒子数量与平均粒子数量之比随时间的变化。负载均衡每 10 次迭代执行一次，扩散策略使用 4 个邻居

约 80 × 80 个网格单元。粒子最初呈指数分布，集中在网格左侧；随着时间推移，它们以每个时间步移动 5 个网格单元的速度向右移动，并定期从一个 chare 迁移到下一个 chare。

处理器分解：Charm++ PIC PRK 基准测试支持两种初始 chare 到处理器的映射方式：(1) 条带映射，即按列主序将 chare 分配给处理器；(2) 四边形映射，即将 chare 分配到连续的二维瓦片中。条带映射会导致更多处理器间通信，因为粒子经常在处理器之间移动；而四边形映射能更好地保持通信局部性。此处我们采用条带映射，以便更好地理解应用程序按列负载不均衡模式。

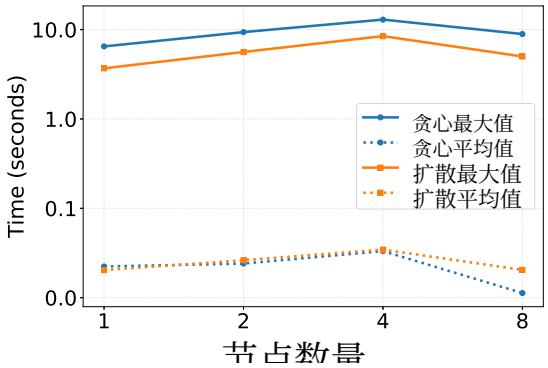
图 3 展示了在 4 个处理器上经过 200 次迭代期间，每个处理器的粒子数量随时间的变化，以观察粒子在整个网格中的迁移情况。负载不均衡的模式十分明显：粒子移入处理器所在列时，该处理器会变得过载；粒子移出时，则会变得负载不足。

B. 负载均衡下的粒子数

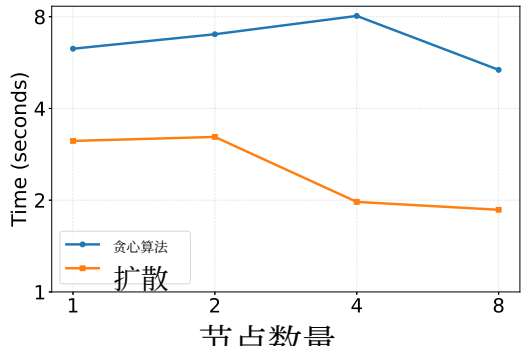
虽然在 PIC PRK 基准测试中，粒子数量只是负载的一种近似指标，并非我们的负载均衡器所使用的指标，但了解负载均衡策略如何影响粒子分布随时间的变化仍然很有帮助。图 4 展示了每个处理器的粒子数量最大值与平均值之比随时间的变化，并比较了无负载均衡、GreedyRefine 和扩散负载均衡策略。与无负载均衡相比，所有负载均衡策略都能将每个处理器的粒子数量最大值与平均值之比维持在低得多的水平。在这个特定用例中，GreedyRefine 和基于坐标的扩散平均提升了 50%，基于通信的扩散平均提升了 48%。

C. 多节点性能

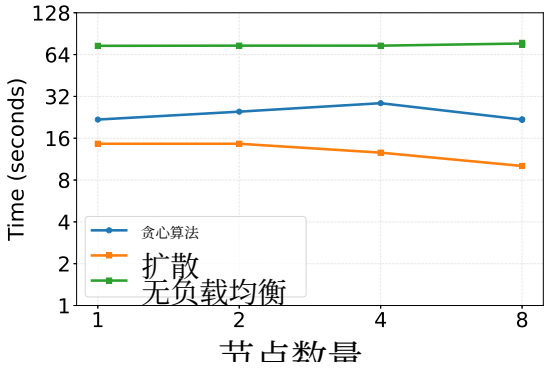
我们在 NERSC 的 Perlmutter 超级计算机上使用最多 8 个节点评估 PIC PRK 基准测试在扩散和 GreedyRefine 负载均衡下的性能，每个节点使用 16 个进程，每个进程使用 8 个物理核心。图 5 展示了 PIC PRK



（a）通信时间



（b）负载均衡时间

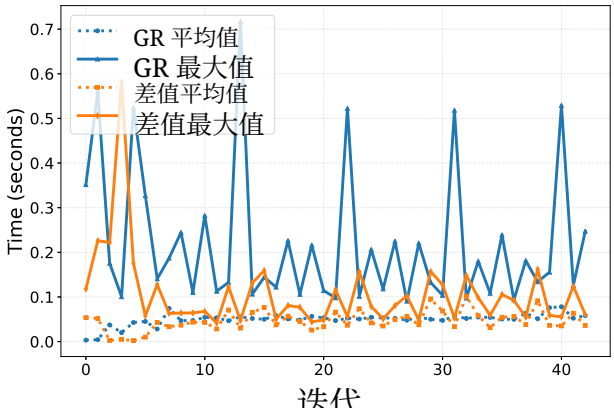


(c) 总运行时间

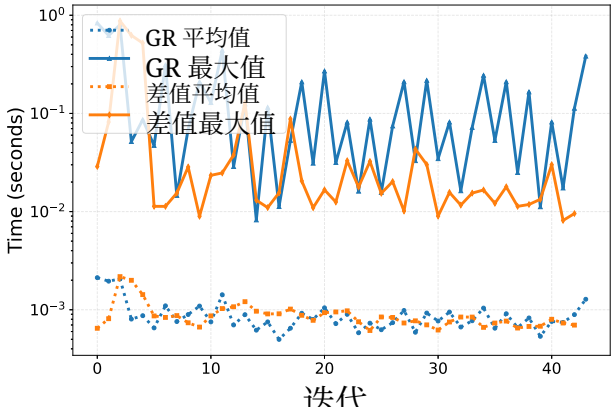
图 5：在 Perlmutter 上对 PIC PRK 基准测试进行强扩展性评估，比较基于通信的扩散与 GreedyRefine 负载均衡策略；使用 6000x6000 网格上的 1000 万个粒子，参数为 $k = 4$ 和 $\rho = .9$，并使 chare 数量随节点数量缩放。

基准测试的强扩展性结果，并细分了通信、负载均衡和总运行时间所耗费的时间。PIC PRK 基准测试存在极端的负载不均衡，因此如图 5c 所示，如果不进行负载均衡，应用程序将完全无法扩展。扩散能够稍微恢复扩展能力，但仍受应用程序的负载不均衡和通信开销所限。扩散在所有规模下均优于 GreedyRefine，且规模越大，性能差距越明显。在 8 个节点时，扩散相较于 GreedyRefine 实现了 2 倍加速比，相较于原始运行时间实现了 7 倍加速比。

为了评估通信局部性的影响，我们进一步分析 8 节点的结果。图 6 显示了在 100 次迭代期间，各处理单元的通信和计算时间平均值；每 5 次迭代进行一次负载均衡。尽管 GreedyRefine 偶尔能够降低最大通信时间，但其通信时间会出现峰值，且通信成本远高于扩散。平均而言，与 GreedyRefine 相比，扩散使每个处理单元的最大通信时间加速 2 倍。



(a) 通信时间



(b) 计算时间

图 6：在 Perlmutter 的 8 个节点上，对 PIC PRK 基准测试进行 100 个负载均衡阶段期间每个进程的通信和计算时间。使用 6000x6000 网格上的 1000 万个粒子，参数为 $k = 4$ 和 $\rho = .9$，以及 200x100 个 chare。每 5 次迭代执行一次负载均衡，以比较基于通信的扩散与 GreedyRefine。

正如预期，两种算法的平均计算时间大致相同。不过，采用扩散后，最大计算时间有所改善，平均而言，相较于 GreedyRefine 可实现 2.5 倍加速比，因为扩散能够更好地适应模拟动态，并在各次迭代中实现更一致的负载均衡。

VII. 未来工作

最值得关注的是，未来工作应探索通信感知扩散负载均衡在 PIC PRK 基准测试之外的完整规模应用中的性能。具有复杂且动态的通信模式的应用

将为所提出的策略提供更严格的测试。具体而言，ChaNGa [18], a Charm++ 宇宙学模拟将受益于改进的负载均衡策略，以保持通信局部性。未来工作将评估扩散与 ChaNGa 的自定义负载均衡策略，并重点考察在多次负载均衡迭代中保持通信局部性的能力。

此外，将我们提出的策略与更广泛的 Charm++ 策略进行比较也很有价值，而不仅仅是本文探讨的 GreedyRefine。具体而言，ParMETIS 重分区在我们的模拟研究中展现出良好前景，但由于它是一个不兼容的、基于 MPI 的库，目前无法在 Charm++中使用。将其集成到 Charm++ 中需要投入更多工程工作，因此留待未来研究。

当前的基于坐标的变体要求每个处理器在进行邻居选择时，按照质心距离对所有其他处理器排序；与基于通信的变体相比，这限制了其可扩展性。未来工作可以探索更高效的空间索引技术，例如空间填充曲线（SFC）或基于八叉树的 k 近邻算法，以更高效地识别相邻处理器。

尽管理论上我们提出的算法适用于许多基于任务的场景，但由于负载均衡策略与运行时特定功能和应用程序接口紧密集成，因此难以将其移植到不同的运行时系统并进行评估。未来工作将探索一种通用框架，将核心扩散策略与运行时特定细节抽象分离，从而支持与其他基于任务的运行时系统集成，并促进更广泛的采用和比较研究。

VIII. 结论

我们提出了一种通信感知扩散负载均衡算法，专为过度分解运行时系统中的不规则通信密集型并行应用而设计。此外，我们还提出了一种基于坐标的变体，适用于无法获取显式通信模式的场景；该变体以空间邻近性作为通信关系的代理，并对结果进行了比较。

通过基于模拟的评估，我们证明，与 GreedyRefine 和图划分等成熟方法相比，我们的扩散策略能够实现具有竞争力的负载均衡，同时显著改善通信局部性并减少对象迁移。PIC PRK 基准测试的性能评估表明，通信感知扩散相比 GreedyRefine 实现了整体加速比，并降低了每个 chare 的平均通信时间，体现了在动态负载均衡中保持局部性的实际优势。

我们的结果表明，在此场景中，将通信感知融入扩散式算法，能够在限制通信开销的同时改善负载均衡。尽管我们的评估规模有限，但这些结果表明，基于扩散的方法是通信密集型应用中一种颇具前景的负载均衡方向。

参考文献

[1] L. V. Kale 和 S. Krishnan，“Charm++：一种基于 c++ 的可移植并发面向对象系统”，SIGPLAN Not.，第 28 卷，第 10 期，第 91–108 页，1993年10月。 [在线]。可访问：https://doi.org/10 1145 167962 165874 。/。N. Morales、R. Jones、J. Lifflander、P. P. P´eba¨y、S. McGovern、C. Skrzy´nski 和 C. Schilly，“基于 DARMA/VT 的分布式异步接触力学”，载于《异步多任务系统与应用：第二届国际研讨会，WAMTA 2024》。柏林，海德堡：施普林格出版社，2024年，第 34–45 页。[3] G. Cybenko，“分布式内存多处理器的动态负载均衡”，《并行与分布式计算杂志》，第 7 卷，第 2 期，第 279–301 页，1989年10月。 [在线]。可访问：https://linkinghub.elsevier.com/retrieve/pii/074373158990021X [4] H. Menon 和 L. Kal´e，“面向迭代应用的分布式动态负载均衡器”，载于《高性能计算、网络、存储与分析国际会议论文集》。丹佛，科罗拉多州：ACM，2013年11月，第 1–11 页。 [在线]。可访问：https://dl.acm.org/doi/10 1145 2503210 2503284 。/。F. Lin 和 R. Keller，“梯度模型负载均衡方法”，《IEEE软件工程汇刊》，第 13 卷，第 01 期，第 32–38 页，1987年1月。 [在线]。可访问：https://doi.ieeecomputersociety.org/10 1109 1987 232563 。/TSE.。V. Saletore，“面向中粒度任务并行处理的分布式自适应动态负载均衡方案”，载于《第五届分布式内存计算会议论文集》，1990年，第 2 卷，1990年，第 994–999 页。[7] R. Els¨asser、B. Monien 和 R. Preis，“异构网络负载均衡扩散方案”，《计算系统理论》，第 35 卷，第 3 期，第 305–320 页，2002年6月。 [在线]。可访问：http://link.springer.com/10 1007 00224 002 1056 4 。/s - - - G. Canright 、 A. Deutsch 和 T. Urnes，“ 受趋化性启发的负载均衡 ”，Complexus，第 3 卷，第 1-3 期，第 8–23 页，2006年。 [在线]。可访问：https://karger.com/CPU/article/doi/10 1159 000094184 。/ Y. Deng 和 R. W. H. Lau，“面向分布式虚拟环境的基于热扩散的动态负载均衡”，载于《第17届ACM虚拟现实软件与技术研讨会论文集》。香港：ACM，2010年11月，第 203–210 页。 [在线]。可访问：https://dl.acm.org/doi/10 1145 1889863 1889910 。/。J. Lifflander、N. L. Slattengren、P. P. Pebay、P. L. P´eba¨y、C. W. Schilly、R. A. Pfeiffer 和 J. D. Kotulski，“通过受内存约束的任务式负载均衡加速电磁模拟”，圣迭戈，2025年。[11] J. Lifflander、P. P. Pebay、N. L. Slattengren、P. L. Pebay、R. A. Pfeiffer、J. D. Kotulski 和 S. T. McGovern，“面向任务负载均衡的通信与内存感知模型”，4月。2024， 2404 16793 arXiv：. cs . 在线。[ ] [ ] 2404 可访问：http://arxiv.org/abs/ 。16793 [12] G. Karypis 和 V. Kumar，“不规则图的并行多层 k 路划分方案”，载于《Supercomputing ’96：1996年ACM/IEEE超级计算会议论文集》，1996年，第 35–35 页。[13] K. Schloegel、G. Karypis 和 V. Kumar，“用于自适应科学模拟负载均衡的统一算法”，载于《SC ’00：2000年ACM/IEEE超级计算会议论文集》，2000年，第 59–59 页。[14] M. Lieber、K. G¨oßner 和 W. E. Nagel，“大规模扩散式负载均衡的潜力”，载于《第23届欧洲MPI用户组会议论文集》。英国爱丁堡：ACM，2016年9月，第 154–157 页。 [在线]。可访问：https://dl.acm.org/doi/10 1145 2966884 2966887 。/。Y. F. Hu 和 R. J. Blake，“用于动态负载均衡的改进扩散算法”，《并行计算》，1999年。[16] R. F. Van der Wijngaart 和 T. G. Mattson，“并行研究内核”，载于《2014年IEEE高性能极限计算会议（HPEC）》，2014年，第 1–6 页。[17] E. Georganas、R. F. Van Der Wijngaart 和 T. G. Mattson，“用于评估动态负载均衡能力的并行研究内核的设计与实现”，载于《2016年IEEE国际并行与分布式处理研讨会（IPDPS）》，2016年，第 73–82 页。[18] P. Jetley、F. Gioachin、C. Mendes、L. V. Kale 和 T. Quinn，“使用ChaNGa进行大规模并行宇宙学模拟”，载于《2008年IEEE国际并行与分布式处理研讨会》，2008年，第 1–12 页。