将机器学习力场表征为图形处理器上新兴的分子动力学工作负载

Udari De Alwis*‡、Benjamin E. Mayer*†、Tom J. Ashby‡、Maria Barrera、Timon Evenblij、Joyjit Kundu IMEC，鲁汶，比利时 电子邮箱：{udari.dealwis, tom.ashby, mariacecilia.barreramoreno,timon.evenblij, joyjit.kundu}@imec.be † 电子邮箱：benjamin.e.mayer@gmail.com·

arXiv:2603.04092v1 [cs.PF] 4 Mar 2026

摘要——**分子动力学（MD）模拟由原子间作用力支配的原子系统的时间演化，这些模拟的保真度关键取决于底层力模型。经典力场（CFFs）依赖拟合实验或理论数据的固定函数形式，具有计算效率高、适用范围广的特点，但在化学多样性或反应性环境中精度有限。相比之下，机器学习力场（MLFFs）通过直接从高层次电子结构数据学习原子间相互作用，以分子力学的计算成本实现接近量子化学精度。**

机器学习力场虽然能以量子方法一小部分的成本提供更高的精度，但也带来了显著的计算开销，尤其是在描述符评估和神经网络推理方面。由于存在不规则内存访问、数据重用率低以及内核执行效率低下，这些操作给并行硬件带来了挑战。

本研究采用聚丙氨酸链来考察此类模型的硬件性能。聚丙氨酸链是一种新颖的基准分子系统，输入规模可控，可用作性能评估测试用例，揭示机器学习力场模拟规模扩展时图形处理器的计算瓶颈。分析找出了描述符计算、力计算和内存处理方面的主要瓶颈，并指出了在药物发现中新兴的基于机器学习力场的分子动力学领域中改进性能的机会；从计算机体系结构的角度来看，该领域此前受到的关注有限。

索引词—**机器学习力场、分子动力学、性能评估**

I. 引言

分子动力学（MD）是一种广泛使用的计算技术 [1], [6], [12] ，通过数值求解牛顿运动方程，模拟原子和分子系统随时间的演化（图 1）。分子动力学模拟的精度和计算效率，根本上取决于用于模拟原子间相互作用的力场（FF）。

经典力场（CFFs），例如 AMBER [5] 和 CHARMM [4],，依赖经验函数形式，其参数根据实验和量子力学数据拟合而得。尽管经典力场具有很高的计算效率和广泛的适用性，但其预测准确性有限，尤其是在具有化学反应环境的系统中，固定参数可能无法泛化。因此，在计算药物发现等对准确性要求很高的应用领域，需要更准确的一类力场。

机器学习力场（MLFF），例如 ANIxx [7][18], [27], , 代表着新一代原子间势，旨在克服这些局限。这些模型利用描述符对局部原子邻域进行编码，并直接从高精度量子力学（QM）计算中学习势能面。随后使用神经网络预测原子对总能量的贡献。与经典力场相比，这种方法使机器学习力场能够达到更高的精度。然而，机器学习力场带来的计算开销远高于经典力场。

尽管经典力场的分子动力学可扩展性已得到充分研究 [13], [16], [20], [24],，但机器学习力场的计算开销仍未得到深入了解。在药物设计中，由于机器学习力场的计算开销很高，通常只用于系统中对精度要求最高的部分，因此形成了一种混合工作负载，将特征已充分掌握的经典分子动力学工作负载与了解较少的机器学习工作负载结合起来。然而，随着计算基础设施效率不断提升，机器学习力场有望应用于更大的原子系统，从而为药物发现应用提供更准确的预测。确定扩展机器学习力场模拟中的计算瓶颈，对于设计未来硬件至关重要。本研究旨在填补这一知识空白，通过分析在图形处理器架构上扩展分子工作负载时机器学习力场的性能，并根据观察到的瓶颈提出优化建议。据作者所知，这是最早从计算机体系结构角度研究机器学习力场工作负载扩展的研究之一。

本文的贡献包括：(a) 设计在化学上与机器学习力场兼容的可变大小分子结构；(b) 分析多种机器学习力场网络架构以及图形处理器上的原子邻域嵌入性能；(c) 通过基于性能分析的建议识别工作流瓶颈。第 II–VI 节涵盖背景、分子

本研究是在 Benjamin Mayer 担任 imec 博士后研究员期间完成的。
*作者贡献相同

系统设置与性能分析、性能分析、瓶颈评估以及结论。

II. 背景

A. 药物发现中的分子动力学与机器学习力场（*MLFF*）

标准分子动力学通过数值方案对分子力学进行积分（图 1）。力分为短程（键合）项和长程（非键合）项。朴素的全原子对计算需要 $\mathcal{O}(n^2)$ 次运算，但基于截断的方法可将其降至 $\mathcal{O}(n\log n)$ 或 $\mathcal{O}(n)$ [10]。生物分子系统包含 10^5–10^8 个原子，模拟性能以纳秒/天衡量。近期研究报告显示，对于分别含有 82k、53M 和 204M 个原子的系统，模拟速度分别为 687、116 和 35 纳秒/天 [15]。由于积分采用 fs 级时间步长 [12], ，要达到具有生物学意义的毫秒至秒级时间尺度，需要数月的图形处理器计算时间 [25]。长程静电作用中的内核启动和通信开销限制了强扩展性。专用加速器（Cerebras WSE [25], 、ANTON [26], 、MDGRAPE [19]）可缓解这些问题，但都无法处理机器学习力场工作负载。

机器学习力场近似势能面（PES）和原子力，能够达到接近量子精度的水平，相较于经典力场和 从头算 方法具有更高的精度 [3]。一种重要的神经网络架构是 Behler–Parrinello 神经网络（BPNN），它将系统能量表示为原子贡献之和。每个原子能量都由一个物种特异性神经网络预测，该网络使用原子中心对称函数（径向 + 角向）来确保平移不变性和旋转不变性 [2], [3]。ANI2x 在 BPNN 框架的基础上进行了扩展，采用改进的对称函数，并在大型数据集（ANI-1x、ANI-1ccx）上进行训练；这些数据集包含数百万个在 ωB97X/6-31G(d) 理论水平下获得的密度泛函理论（DFT）构象 [28]。这使其具备广泛的化学泛化能力，包括构象采样以及蛋白质-配体建模。在 ANI 之外，机器学习力场也从系统特异性架构（例如 sGDML）发展为 TorchMD-Net [23],SchNet 和 DeepMD 等通用架构。这些框架支持生物分子模拟，能够捕捉蛋白质折叠、结合和药物发现中的复杂相互作用与热力学性质 [32]。

III. 用于机器学习力场模拟与图形处理器性能分析的分子系统

A. 模拟工作负载的设置

一个化学相容、可扩展且尺寸可控的分子系统，是评估机器学习力场在 GPU 上所产生计算开销的理想候选对象。然而，由于训练数据的性质，机器学习力场通常并未在大型生物分子系统上训练。相比之下，经典力场已经成熟且可扩展。由于机器学习力场是基于小分子的 DFT 数据训练的，其假想用途在于改进较大系统中小分子的动力学。一种合理的未来计算工作负载是，不仅将机器学习力场应用于小分子，也应用于周围的反应区域（结合口袋）和局部环境，从而捕捉其全部相互作用。目前，机器学习力场的训练集仅包含极小的类肽系统，例如丙氨酸二肽；尽管它比单个丙氨酸残基略为复杂，但仍缺乏真实蛋白质折叠成三维结构所具有的结构复杂性。尽管这一用例超出了当前训练数据的范围，但对于性能研究仍然具有相关性。为了在贴近训练域的同时实现可扩展性，我们选择了聚丙氨酸（丙氨酸残基的均聚物）作为模型系统。为获得足够大的系统，使用 AlphaFold 2 [17] [14]对一条含 1000 个残基的聚丙氨酸（即由 1000 个丙氨酸组成的序列）进行折叠，并据此生成尺寸（按对数）递增的聚丙氨酸片段。这在图 2 中进行了可视化。如图 2 所示，从该 1000 残基聚丙氨酸蛋白质中生成一系列片段，以构建基准数据集。片段尺寸按对数递增，例如含 $[10, 20, \ldots, 100, 200, \ldots, 1000]$ 个氨基酸的分子。

为量化机器学习力场系统在分子动力学工作流中的性能，我们使用聚丙氨酸片段作为输入结构，在 OpenMM/Gromacs 中对三种配置 CFFsys、MLFFsys 和 CMLsys 进行了基准测试。CFFsys 指的是聚丙氨酸和溶剂均采用经典力场的分子系统（基线）。MLFFsys 指的是聚丙氨酸处于真空中且仅应用机器学习力场的分子系统。CMLsys 指的是聚丙氨酸在水中进行模拟的分子系统，其中聚丙氨酸采用机器学习力场，系统其余部分采用经典力场，包括溶剂与聚丙氨酸分子之间的相互作用。将溶质加入扩展后的聚丙氨酸残基时，由于折叠蛋白质的结构，溶质直径在 200 至 900 个残基之间几乎保持不变。然而，随着最后 100 个残基的加入，系统中的原子数量约增加了 2.5 倍，因为蛋白质的形状发生了显著变化。

B. *GPU*分析器设置与数据采集

分子动力学模拟通过许多连续的时间步进行（图 1）。在生产运行中，对 1000 个时间步进行了计时，分子动力学模拟的分辨率为 0.5 fs。为分析图形处理器资源利用率，使用 PyTorch 中的 CUDA分析器API，结合 OpenMM 和 Gromacs 中的检查点，隔离分子动力学模拟生产运行中的单个步骤（即轨迹计算）进行性能分析。生产步骤包括势能计算和力计算。使用 Nsight Systems (NSys) [22]和 Nsight Compute (NCU) [21] 对应用程序的整体硬件资源利用率以及各个内核的资源利用率进行性能分析。排除模拟开始时的几个步骤，以消除 OpenMM/Gromacs 中的应用程序执行和 PyTorch编译所产生的设置开销。性能分析前会执行能量最小化步骤，以确保分子系统的稳定性。除在分子动力学主流程中运行机器学习力场外，还使用 PyTorch 性能分析器 [30] 以及 NSys 和 NCU 对机器学习力场（ANI2x、TorchMD-Net）单独进行性能分析，以量化机器学习前向传播的独立影响

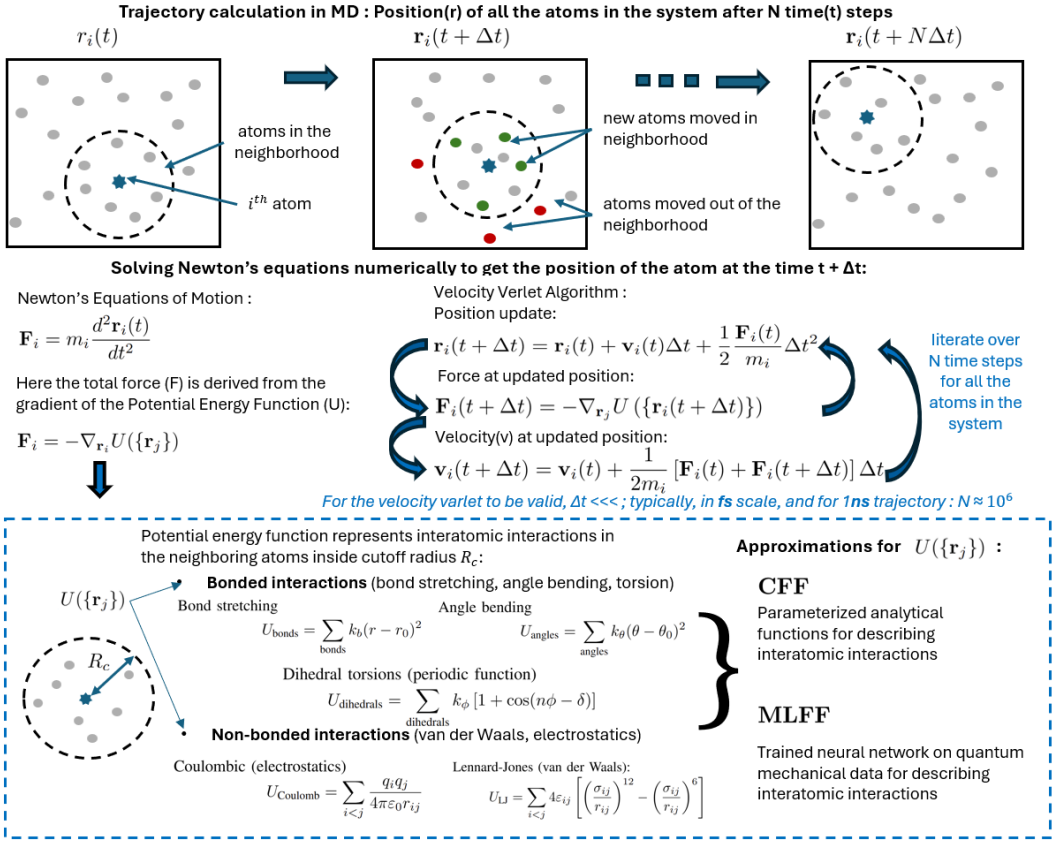

图 1. 分子动力学中的轨迹计算，以及用于轨迹中力计算步骤的势能函数近似（考虑固定截断系统，不包括长程相互作用）。

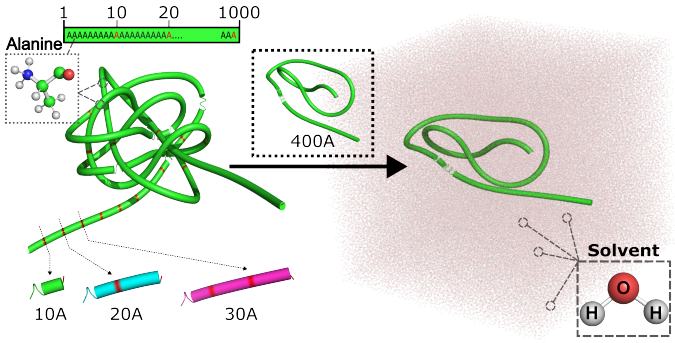

图 2. 通过从一条含 1000 个残基的聚丙氨酸蛋白质中逐步提取片段来构建基准系统的示意图。图中展示了全长蛋白质的序列和结构，红色标记表示用于生成不同尺寸系统的切割点。每个生成的系统都在水盒中进行平衡。

以及使用不同系统规模下随机生成的原子坐标评估反向传播路径的独立影响。

GPU 实验在一台配备 NVIDIA A100 80GB PCIe 卡的计算节点上进行基础测试；其他 GPU 实验则在同一集群中搭载 NVIDIA A100 40GB SXM4、NVIDIA H100 80GBHBM3 和 Tesla V100 PCIe 32GB GPU 的设备上进行。

IV. 机器学习力场架构分析

ANI-2x 和 TorchMD-Net 具有共同的架构基础，根植于局域性、对称性保持和原子能量分解。两者都将分子能量表示为原子中心贡献之和，这些贡献源自局部邻域，并以尊重物理系统平移、旋转和置换对称性的方式进行编码。深度神经网络架构对这些局部描述符进行细化，以捕捉复杂的多体效应；而力则通过对学习得到的势能进行自动微分而获得。这些共同原则使两种模型都能以适用于大规模分子动力学模拟的计算成本，提供量子精度预测。

A. ANI2x

ANI2x [7]是一种基于集成的神经网络势，旨在预测分子能量和力，同时保持物理对称性。其框架包含三个主要阶段：(i) 通过原子环境向量（AEV）进行原子环境嵌入，(ii) 通过元素特定的多层感知机进行能量预测，以及 (iii) 通过自动微分进行力评估。

1) 原子环境向量（*AEV*）：每个原子由固定长度的描述符描述，该描述符编码其局部环境

在截断半径$R_c \approx 5.1$ 内。原子环境向量由径向分量和角向分量组成：

$$G_m^R(i) = \sum_{j \neq i} e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}), \tag{1}$$

$$f_c(R_{ij}) = \begin{cases} \frac{1}{2}\left[\cos\left(\frac{\pi R_{ij}}{R_c}\right)+1\right], & R_{ij} \leq R_c, \\ 0, & R_{ij} > R_c, \end{cases} \tag{2}$$

$$\begin{aligned} G_m^A(i) = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} \left[1+\cos(\theta_{ijk}-\theta_s)\right]^\zeta \\ \cdot \exp\left[-\eta\left(\frac{R_{ij}+R_{ik}}{2}-R_s\right)^2\right] \cdot f_c(R_{ij}) \cdot f_c \end{aligned} \tag{3}$$

径向对称函数G_{Rm} 使用高斯基函数对两两距离进行编码，并通过平滑截断进行调制；而角向函数G_{Am} 则对三元组夹角进行编码 θ_{ijk} ，以捕捉局部几何构型。AEV 的大小随着元素类型数量以及径向和角向函数数量的增加而增大。对于 ANI2x，所得 AEV 包含 1008 个分量。

2）神经网络势（*NNP*）：ANI2x 采用特定于元素的神经网络势，高效预测分子能量和力。分子总能量表示为原子贡献之和：

$$E_T = \sum_{i=1}^{N} \mathcal{N}_{X_i}\left(\mathbf{G}_i^X\right), \tag{4}$$

其中N 为原子数量，X_i 为原子i的原子序数，$\mathcal{N}_{X_i}$ 为特定于元素的多层感知机，而$\mathbf{G}_{X_i}$ 为表示局部化学环境的原子环境向量（AEV）。每种元素使用多个多层感知机组成的集成，相较于单个大型网络可提高预测精度。

3）能量与力的计算：力通过自动微分计算：

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{r}_i}. \tag{5}$$

在前向传播过程中，原子坐标经过集成多层感知机映射，以计算总能量E_T。通过对原子位置的能量求导，并将梯度通过网络反向传播，可以获得力：

$$\frac{\partial E}{\partial \mathbf{r}_i} = \frac{\partial E}{\partial h_n} \cdot \frac{\partial h_n}{\partial h_{n-1}} \cdots \frac{\partial h_1}{\partial \mathbf{r}_i}, \tag{6}$$

或者等价地，使用原子环境向量表示：

$$\mathbf{F}_j = -\frac{\partial E}{\partial \mathbf{r}_j} = -\sum_i \left(\frac{\partial E_i}{\partial \mathrm{AEV}_i} \cdot \frac{\partial \mathrm{AEV}_i}{\partial \mathbf{r}_j}\right). \tag{7}$$

这确保了力与能量面保持一致，并保留了分子系统的物理对称性。

B. ANI2x 性能分析

通过测量三个主要阶段的计算时间，分析了 ANI2x 的性能随系统规模的变化：原子环境向量（AEV）构建、神经网络能量计算（MLP-energy），以及通过反向传播进行的力计算。AEV 计算表现出与原子数量大致线性的扩展关系，这反映了其逐原子、基于邻域的特性；在系统规模较小时，由于 GPU 利用不足，会出现轻微波动（图 3）。在 AEV 计算步骤中，径向环境计算需要对截断区域内的每一对 i,j 原子计算该函数（不使用邻域列表）。相比之下，角向环境计算需要对每个 i,j,k 原子三元组计算该函数。对于具有 N 个原子且平均每个原子有 M 个邻居的分子系统，径向函数的计算复杂度为 $\mathcal{O}(N \times M)$。同样，角向函数的计算复杂度为 $\mathcal{O}(N \times M \times M)$。

相比之下，MLP 能量评估在大型系统中显著增加，尤其是原子数量超过 6,000 时；这可归因于稠密线性运算以及大型张量操作带来的图形处理器内存开销。力计算随原子数量适度扩展，因为它需要通过原子环境向量和多层感知机层传播梯度，而每个原子的邻居数量大致保持不变。总体而言，原子环境向量计算的开销仍然很低，而对于大型分子，MLP 能量评估占据了大部分模拟时间；力计算则占总成本的比例适中，但不可忽略。这些观察结果表明，ANI2x 对中小型系统而言效率较高，但在超大型分子系统中会遇到内存和计算瓶颈。

此外，随着 M 的增加，内存需求迅速增长（图 4）。与多层感知机中通常使用的简单得多的通用矩阵乘法不同，此处的计算需要存储中间结果以供下一步使用。此外，由邻域变化引起的不规则内存访问所导致的 L1 缓存未命中，以及未利用缓存复用的直接内存访问，都会给 L2 缓存带来沉重负担。虽然前向传播相对轻量，但用于力计算的反向传播需要显著更多的内存和计算资源，尤其是在原子数量众多的大型系统中。

1) PyTorch 函数分析：在 ANI2x 神经网络势中，计算热点分布于 AEV、能量和力阶段，并呈现出差异，这反映了模型底层的算法结构（图 5）。在 AEV 计算期间，最耗时的组件与邻居列表生成和角三元组枚举有关，它们高度依赖 aten::index、aten::index_select 和 aten::sort 等受内存带宽限制的张量索引操作。这些内核会执行分散的内存访问和掩码操作，导致较低的算术强度和较低的 GPU 利用率。相比之下，能量评估阶段主要由稠密神经网络

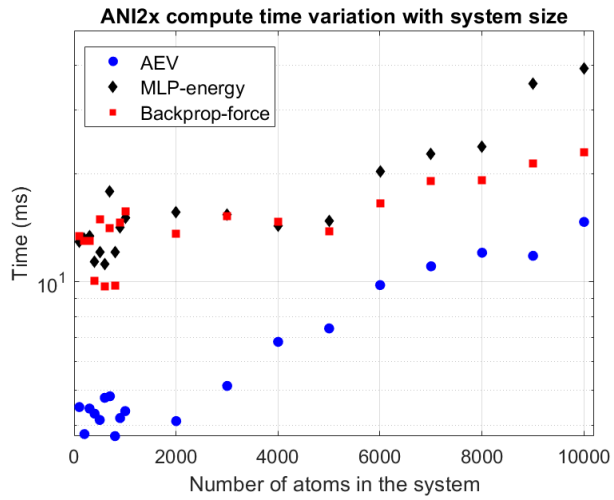

图 3. ANI2x 势能/力计算时间与原子环境向量计算时间随系统规模变化的比较。

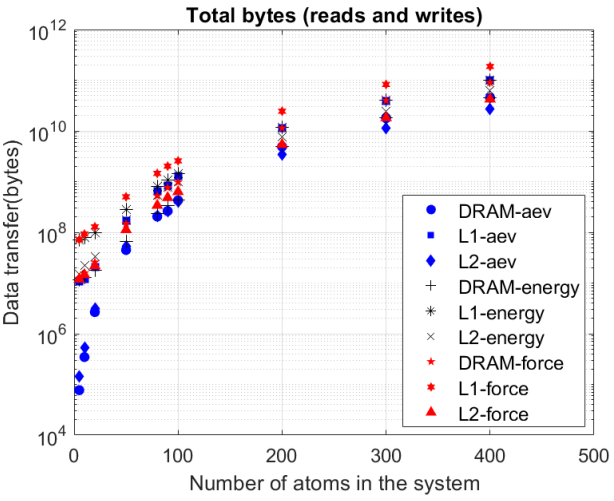

图 4. ANI2x 原子环境向量/势能/力内存传输随邻域大小变化的情况。

推理构成，其中通过 aten::linear 和 aten::addmm 实现的全连接层占据了大部分计算时间。这些操作属于受计算能力限制的矩阵乘法，其计算量会直接随原子数量和 ANI 子模型集成的深度扩展。最后，由于自动微分，力评估阶段产生了最高的总体计算成本。maskedScatterBackward、addmmBackward 和 aten::maskedSelect 等反向传播内核占据了主要运行时间，因为它们在执行不规则的聚集与散射操作时，既需要进行梯度传播，又需要重建中间值。这些梯度内核表现出较差的内存局部性和较高的同步开销，导致力计算耗时显著增加。

2) CUDA 内核分析：图 6 展示了若干内核类型的内存吞吐量和计算吞吐量，这些内核

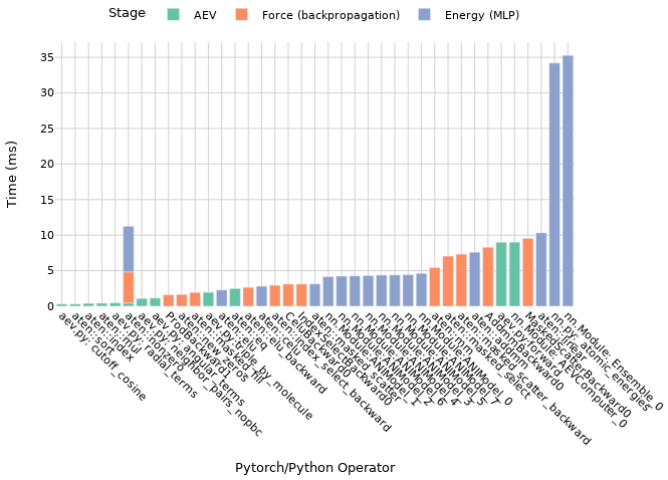

图 5. 组合算子计时：原子环境向量 vs 能量 vs 力

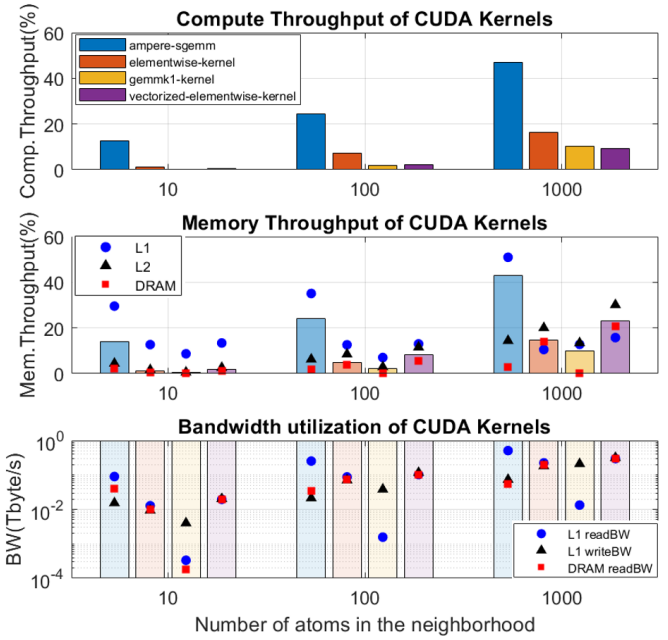

图 6. CUDA 内核的相对计算（SM）、内存吞吐量与带宽利用率。

广泛应用于 ANI2x 机器学习力场的 AEV、势能（前向传播）和力（反向传播）计算中（这些计算占 GPU 总运行时间的 44%–58%）。通过 AEV 计算和多个神经网络层进行反向传播，会触发比前向传播更多的 CUDA 内核启动。根据 NCU 内核性能分析，TorchANI 实现的 ANI2x 能量计算在邻域大小为 10 至 100 个原子时，会启动 77–81 个 CUDA 内核。相比之下，对于相同的邻域大小，力计算最多需要启动 96 个 CUDA 内核。这些吞吐量矩阵反映了给定内核对图形处理器硬件资源的有效利用率。含有数百个原子的分子无法充分利用图形处理器资源，因为内核网格

未能充分占用设备的可用资源，导致所有流式多处理器（SM）上的完整波次数量减少。此外，对于所研究的 CUDA 内核，整体内存带宽利用率低于 1 Tbytes/s，表明硬件内存资源未得到充分利用。随着工作负载规模增大，内核的计算和内存利用率随之提高，从而更有效地利用硬件资源，并摊销内核启动开销。在 AEV 计算中，每次迭代都要根据原子间距离计算原子邻域（式 2）。当原子在空间中的排列不规则时，邻域大小也会不一致。因此，图形处理器线程会访问分散的内存地址，造成非合并内存访问。这会损害缓存效率并增加内存延迟，进而降低图形处理器性能。

C. TorchMD-Net：等变 *Transformer*（*ET*）架构

TorchMD-Net [31] 是一种等变神经网络模型，旨在借助强物理先验学习分子势能面和力场。其架构由三个主要阶段组成：(i) 原子嵌入与环境编码，(ii) 能量预测，以及 (iii) 力评估。该模型确保了分子物理学中至关重要的旋转、平移和置换对称性。

1) 嵌入与环境编码：每个原子由其原子序数 Z_i标识，首先映射为一个学习得到的嵌入：

$$\mathbf{h}_i^{(0)} = \mathrm{Embed}(Z_i), \tag{8}$$

从而提供初始的标量特征表示。接下来，通过构建邻居列表 $\mathcal{N}(i)$，纳入截断距离 r_c内的所有原子 j，以引入局部几何信息。原子 i 和 j 之间的成对距离计算如下：

$$r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|. \tag{9}$$

使用径向基函数（RBF）对该距离进行展开：

$$\phi_{ij} = \mathrm{RBF}(r_{ij}), \quad r_{ij} < r_c. \tag{10}$$

TorchMD-Net 使用两种特征表示每个原子：(i) 标量特征 $\mathbf{h}_i \in \mathbb{R}^d$（旋转不变）和 (ii) 向量特征 $\mathbf{v}_i \in \mathbb{R}^{d\times 3}$（旋转等变）。在旋转 $R \in SO(3)$ 下，这些特征按如下方式变换：

$$\mathbf{h}_i \to \mathbf{h}_i, \quad \mathbf{v}_i \to R\mathbf{v}_i, \tag{11}$$

确保在旋转下具有正确的变换行为。

这些特征通过一系列 ET 层进行更新，每一层都利用几何项执行基于注意力的消息传递。标量特征更新如下：

$$\mathbf{h}_i' = \sum_{j\in\mathcal{N}(i)} \alpha_{ij} W_h \mathbf{h}_j, \tag{12}$$

向量特征则更新为：

$$\mathbf{v}_i' = \sum_{j\in\mathcal{N}(i)} \alpha_{ij} W_v\left(\mathbf{v}_j + \mathbf{r}_{ij} \otimes \mathbf{h}_j\right), \tag{13}$$

其中 α_{ij} 表示距离调制的注意力权重。

2）能量预测：经过L 个等变层后，每个原子都具有经过精炼的标量和向量特征（$\mathbf{h}_{(L)\,i}$，$\mathbf{v}_{(L)\,i}$）。由于势能具有旋转不变性，向量特征通过其模长发挥作用：

$$\tilde{v}_i = \|\mathbf{v}_i^{(L)}\|. \tag{14}$$

随后，一个轻量级神经网络将这些特征映射为原子能量贡献：

$$E_i = f_\theta\left(\mathbf{h}_i^{(L)}, \tilde{v}_i\right), \tag{15}$$

总能量则是原子贡献之和：

$$E = \sum_{i=1}^{N} E_i. \tag{16}$$

3) 力评估：作用于原子的力计算为总能量相对于原子坐标的负梯度：

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{r}_i}. \tag{17}$$

D. Torchmd-NET 性能分析

在 TorchMD-Net 中，主要计算开销来自 ET 每一层所采用的等变多头注意力机制。对于由 N 个原子组成、每个原子平均有k 个邻居的分子系统，该注意力机制会在约 Nk 条边上执行消息传递操作，每条边均需经过H 个注意力头和C 个隐藏通道的处理。这些计算涉及标量特征和向量特征；后者需要进行向量投影、方向归一化以及基于外积的更新。因此，每层的计算复杂度为 ($\mathcal{O}(NkHC)$)，使得向量等变注意力内核成为 GPU执行期间的主要瓶颈。性能分析表明，总耗时的 45–70% 用于这些注意力和消息传递操作（图 7）。对于较小的分子系统（约 100–1000 个原子），ET、能量读出和力计算阶段的执行时间相对稳定，仅有轻微波动。这表明，在此规模下，内核启动、图构建和邻居列表初始化等固定开销占据了主要计算成本。当原子数增至约 2000 以上时，所有三个阶段的运行时间都会随系统规模近乎线性增长，这反映了基于邻域的消息传递、等变注意力操作以及散射归约内核的开销随之增加。在这三个阶段中，对于大型系统，力计算的计算开销最大，因为它需要对 ET 表示和能量读出层进行反向传播。

TorchMD-Net 的内存访问统计呈现出随系统规模增大而显著扩展的趋势（图 8）。随着原子数量增加，DRAM、L2缓存和 L1缓存流量迅速上升，表明其高度依赖邻域大小和消息传递操作。

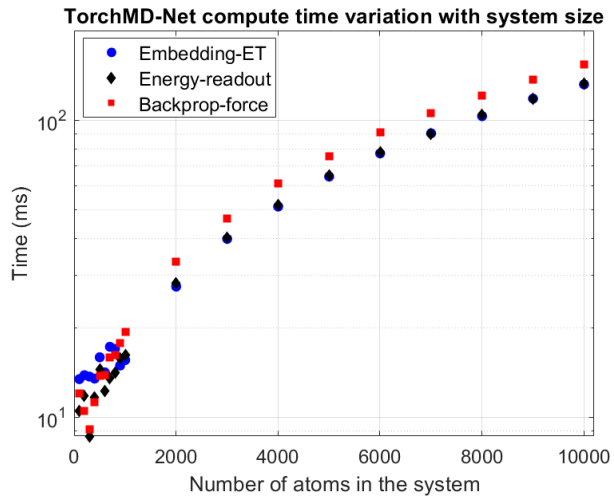

图 7. 不同系统规模下，TorchMD-Net 势能/力计算时间与邻域嵌入计算时间的对比。

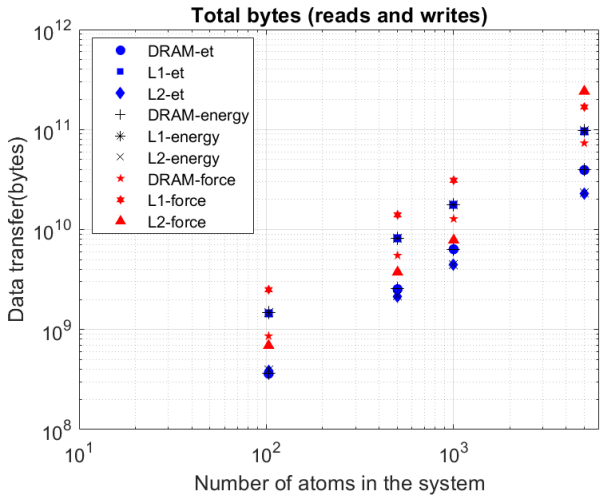

图 8. 不同邻域大小下，Torchmd-NET Embedded-ET/势能/力的数据传输。

嵌入式 ET 和能量阶段表现出几乎相同的内存访问模式，反映出它们共同依赖等变注意力、散射归约操作和特征聚合。相比之下，力计算阶段在所有缓存层级上都会产生显著更高的内存流量，尤其是在 DRAM 和 L2缓存中；这是由于需要通过完整的计算图执行反向传播，并反复访问中间激活。

1) PyTorch 函数分析：在 TorchMD-Net 中，主要计算开销来自构成模型架构基础的消息传递和等变注意力机制（图 9）。在 embedded-ET 阶段，大部分运行时间都花在 TorchMD_ET 模块中，其中消息传递、传播和聚合构成了主要瓶颈。这些操作高度依赖不规则的聚集/散射模式，这一点从

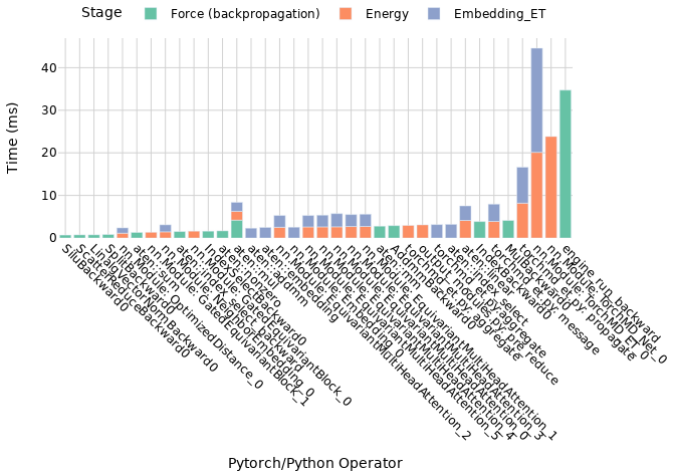

图 9. CombinedOperator 计时：Embedded-ET 与能量、力的对比。

耗费在 aten::index_select、散射和聚合例程上的大量时间可见。多头等变注意力块对特征向量应用学习得到的旋转和张量投影，因此增加了额外的线性层和张量重塑开销。消息函数本身会产生稠密线性变换（aten::linear）的开销，但与受内存带宽限制的索引和邻域聚合开销相比，这部分仍是次要的。在能量预测阶段，同样的注意力和消息传递内核再次出现；由于读出前需要进行更深层的变换，稠密线性代数（aten::linear；aten::addmm）的占比更加显著。力阶段的主要开销来自这些内存密集型内核的反向传播，其中自动微分必须重新执行所有基于邻域的索引操作和线性变换。因此，MulBackward0、AddmmBackward0 和 IndexBackward0 等梯度内核占据了大部分运行时间。

2) CUDA 内核分析：对于小型分子系统，ampere_sgemm、elementwise_kernel 和 vectorized_elementwise_kernel 等 CUDA 内核最多占运行时间的 57%，它们处理的张量相对较小，因此线程级并行性有限，流式多处理器的利用率也较低（图 10）。在这种情况下，内核启动开销和内存访问延迟主导执行过程，导致计算资源和内存资源的利用率都不高。随着原子数量增加，与等变消息传递、注意力投影和线性变换相关的中间张量规模大幅增长，从而提高线程束占用率，并更有效地利用图形处理器执行单元。这使 ampere_sgemm 内核的计算吞吐量显著提高。同时，较大工作负载的空间局部性和缓存复用增加，改善了内存吞吐量，使 elementwise 和 vectorized_elementwise 内核能够更好地分摊全局内存访问开销。尽管如此，吞吐量仍远低于硬件峰值能力。

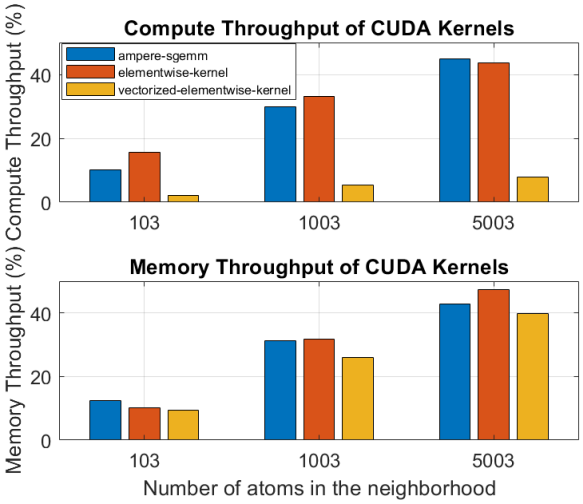

图 10. CUDA 内核的相对计算（SM）与内存吞吐量。

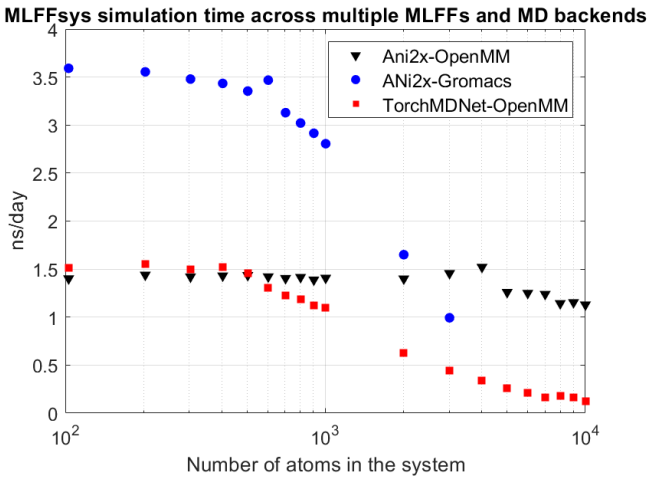

图 11. 在不同工作负载、机器学习力场和分子动力学后端下，MLsys 系统执行分子动力学生产步骤的模拟时间（在单个 A100 80GB 节点上模拟）。

V. 机器学习力场在分子动力学轨迹计算中的性能分析

A. CFF 与机器学习力场在 *GPU* 上的性能比较

基于 NNPot [11] 的 Gromacs 集成 ANI2x，在性能上略优于 OpenMM 中集成的 openmm–ml [9] ANI2x（图 11）。然而，随着工作负载规模扩大，Gromacs 的效率稳步下降。Torchmd–Net 通过 TorchForce [8] 对象集成到 OpenMM 中。尽管 Torchmd–Net 在原子数量较少时效率略高，但原子数量超过 700 后，性能便持续下降。图 12 汇总了 CMLsys 在不同工作负载下的实验结果，其中 ANI2x（机器学习力场）作用于分子，而 AMBER（经典力场）作用于溶剂。这里以纳秒/天衡量计算效率。对于包含 2000 至 35000 个原子的系统，CMLsys 所需的计算时间是另一系统的 230 至 100 倍。当系统规模扩大到 145000 至 600000 个原子时，时间差缩小至 75 至 15 倍。比较 ANI2x MLsys（图 11）和 CMLsys 可见，在相应工作负载规模下，两者耗时几乎相同，这表明计算时间主要由机器学习力场计算所占据。从原子相互作用类型来看，非键相互作用（范德华相互作用和库仑相互作用（图 1））带来了很高的计算开销，因为它们涉及系统中的所有原子。将解析经典力场的运算次数相加（方程列于图 1），每对原子的浮点运算次数约为 25–30 次（即，对于每个原子有 M 个邻居的 N 个原子，总短程非键相互作用的浮点运算次数 $\approx 25(NM/2)$。对于 Ani2x 机器学习力场计算，邻域中有 M 对原子时，原子环境向量层最多需要 $\approx 16M + 32(M/2)$ 次运算；若对 H 使用单个多层感知机，则单个 H 原子的乘加运算总计为 $\approx 650k$ 次运算。对比两个系统中的同一个丙氨酸二肽分子，经典力场的运算次数总计为 8k，而机器学习力场则总计为 10M，这导致了

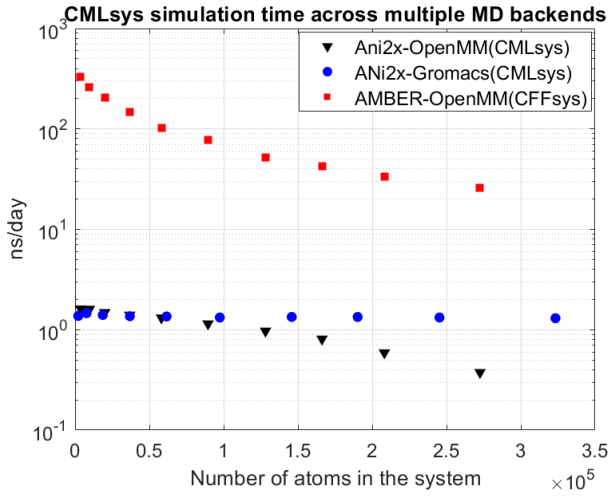

图 12. 在不同工作负载下，CMLsys 系统执行 1000 个分子动力学生产步骤的模拟时间（在单个 A100 80GB 节点上模拟）。

即使多层感知机采用高度并行执行，仍观察到 200 倍的开销。

B. 多个平台上增加工作负载时的 *GPU* 性能

将 ANI2x 集成到 OpenMM/Gromacs 中进行分子动力学模拟时，由于 OpenMM/Gromacs 框架中的机器学习力场缺乏域分解支持，与 CFF 情况不同，其在多个 GPU 上的性能扩展受到限制。这种缺乏域分解的情况使得无法测试机器学习力场性能在多个 GPU 及其相应分子动力学后端上的可扩展性。这是未来分子动力学后端版本开发中必须实现的功能，以便在更大系统规模中高效使用机器学习力场。

在不同 GPU 架构上对 ANI2x 力计算进行基准测试时（图 13），观察到 V100

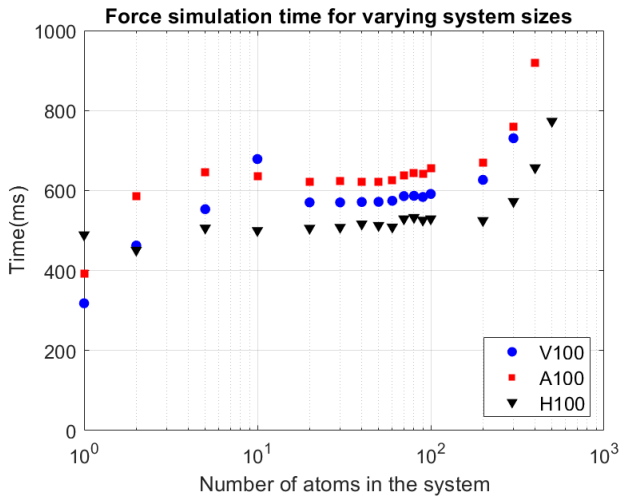

图 13. TorchANI 在不同 GPU 硬件平台上对不同系统规模进行 ANI2x 力计算。

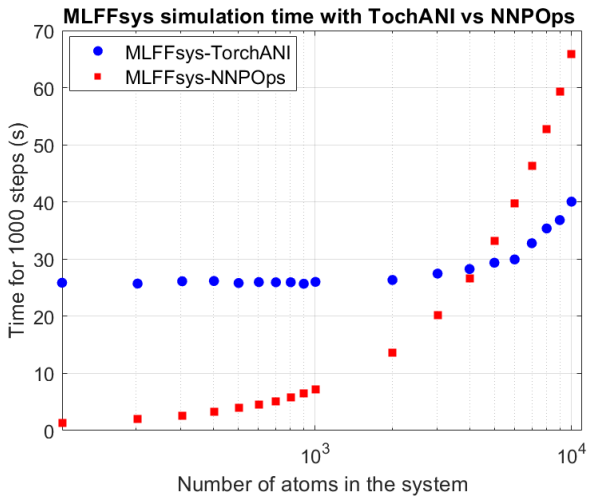

图 14. TorchANI 与 NNPOps 原子环境向量加速下，MLFFsys 运行 1000 步的模拟时间。

在几乎所有系统规模下的表现都优于 A100。同时，H100 在小型和大型系统上始终表现出更优的性能。对于机器学习力场工作负载而言，调度模型优化程度较低的 V100 比 A100 性能更好，因为机器学习力场主要由大量生命周期短、算术强度低的 CUDA 内核构成，而非融合式、受吞吐量限制的内核；A100 更复杂的调度机制正是为后者而设计的。A100 增加的资源（例如更多的流式多处理器和更大的缓存）在低负载下未能得到充分利用，导致性能欠佳。相比之下，H100 在架构上进行了改进，包括降低内核启动延迟并增强隐藏延迟的能力，从而能够高效处理小型和大型工作负载。

C. 图形处理器中的原子环境向量加速

NNPOps [29] 通过以融合CUDA内核取代 PyTorch 原生张量操作，加速 ANI 中的 AEV 计算。TorchANI 使用标准 PyTorch 操作计算 AEV，每个 AEV 会启动 $60-65$ 个 CUDA 内核。虽然这种方式灵活，但会因大量小型内核而产生开销、导致内存使用效率低下和图形处理器利用率不佳，尤其是在分子动力学中常见的 $50-100$ 个原子邻域大小下。对于较小的邻域，Tor- chANI 的内核启动开销可能超过实际计算开销；随着邻域大小增加，性能才逐渐改善。NNPOps 将多个步骤整合为四个融合内核，并将中间结果保存在寄存器/共享内存中，从而降低启动开销。NSys 统计数据显示，$20\%-60\%$ 的 CUDA API 时间耗费在 cudaMallocManaged 上，表明存在统一内存开销。NNPOps 还采用空间分箱和优化的邻居列表，使 AEV 计算速度提升近 100 倍；若计入能量/力计算，提速为 $17-20$ 倍。NCU 性能分析显示，对于 $10-100$ 的邻域大小，径向/角向内核在 A100 上受计算能力限制，但较小的邻域会导致流式多处理器利用不足。结合 OpenMM（图 14）中，NNPOps 凭借针对延迟优化的内核，在小型系统上表现出色，但在较大规模下扩展性较差。

D. 可能的硬件与软件优化

目前，ANI2x 和采用 OpenMM/Gromacs 后端的 TorchMD-Net 都不支持多图形处理器，因为它们缺乏针对机器学习势的域分解。空间划分或按原子划分可以实现跨图形处理器的数据并行推理，从而提升性能。在 ANI2x 中，每个原子环境向量包含 1008 个浮点数，此外还有反向传播路径所需的中间激活和坐标梯度。对于大型系统，这会增加图形处理器的内存负担，并限制批次大小。混合精度训练/推理、原子环境向量压缩（例如，使用更少的对称函数或基于注意力的描述符），或稀疏表示，可能有助于减轻这些机器学习力场的内存负担。定制 CUDA 实现（类似于 NNPOps）和融合内核可以降低启动延迟，并减少内存与计算停顿的相互影响。TorchMD-Net 尤其在其等变 Transformer 的嵌入和消息传递阶段，会产生大量较小的 CUDA 内核启动和非合并内存访问，导致图形处理器利用率不佳、延迟较高。可以尝试多种策略来缓解这些瓶颈。内核与算子融合可以将多个小型操作合并为单个 CUDA 内核，从而减少启动开销并提高算术强度。通过重新排列原子索引、使用连续邻居列表或采用压缩稀疏表示，可以优化内存访问模式，改善内存合并访问和吞吐量。此外，注意力层和消息传递层可以采用低秩或块稀疏近似、融合投影及优化的通用矩阵乘法内核，以提高算术强度。通过算法修改，例如减少 ET 层数量、进行截断平滑或采用轻量级几何编码器，或许能够在保持精度的同时改善扩展性，适用于

更大的分子系统。展望未来，面向机器学习力场工作负载的图形处理器应支持容量更大的可配置 L1/共享缓存、针对非连续内存访问的硬件预取、针对小尺寸通用矩阵乘法优化的张量核心、低延迟内核启动与融合，以及针对异质邻域的自适应线程束调度。计算单元之间的高带宽互连可以最大限度地降低数据传输成本。

VI. 结论

由于机器学习力场能够提高精度，因此它们正逐渐成为分子动力学模拟中的新兴工作负载。然而，与经典力场相比，这也带来了额外的计算开销。本文采用聚丙氨酸切片作为一种系统规模可控的分子系统，以分析机器学习力场在图形处理器上随工作负载规模扩展时的影响。分析涵盖两种流行的机器学习力场网络架构（ANI2x 和 TorchMD-Net），以及它们在工作负载规模不断增大时的性能表现。

引入该工作负载并制定详细的性能分析策略，揭示了机器学习力场中的性能瓶颈。每个时间步都要进行前向与反向路径计算，这使机器学习力场计算与传统神经网络推理显著不同，因此带来了独特的内存和计算挑战。此外，与机器学习中使用的传统嵌入技术相比，原子环境中的动态邻域变化也给原子环境向量/电子转移计算带来了内存访问挑战。这使得分子动力学相关工作负载需要采用新的内存架构设计，而标准机器学习工作负载并不需要这些设计。此外，使用融合内核并引入域分解等并行化技术，有助于有效利用现有的 GPU基础设施。

致谢

本研究由 imec.prospect项目IMPACT 资助。作者感谢 Peter Kourzanov 提供宝贵的反馈和故障排查协助，感谢 GeertVanmeerbeeck、Tom Vander Aa 提供技术支持，并感谢 Peter Vrancx 和 Eleni Litsa 进行富有见地的讨论。

参考文献

[1] M. Ahmed、A. M. Maldonado 和 J. D. Durrant，《从字节到实验台再到病床：分子动力学模拟与药物发现》，*BMC Biology*，第 21 卷，第 1 期，2023 年 12 月。[2] J. Behler，《用于构建高维神经网络势的原子中心对称函数》，*The Journal of Chemical Physics*，第 134 卷，第 7 期，页码 074106，2011年。[3] J. Behler 和 M. Parrinello，《高维势能面的广义神经网络表示》，*Physical Review Letters*，第 98 卷，第 14 期，页码 146401，2007年。[4] B. R. Brooks、C. L. Brooks III 等，《CHARMM：生物分子模拟程序》，*Journal of Computational Chemistry*，第 30 卷，10，2009 年第 .
[5] D. A. Case、K. Belfon、I. Y. Ben-Shalom 等，《Amber 2020》，2020年。[6] A. K. Chew、M. A. F. Afzal、Z. Kaplan、E. M. Collins、S. Gattani、M. Misra、A. Chandrasekaran、K. Leswing 和 M. D. Halls，《利用高通量分子模拟和机器学习设计化学混合物》，*npj ComputationalMaterials*，第 11 卷，第 1 期，2025 年 3 月。
[7] C. Devereux、J. S. Smith、K. K. Huddleston、K. Barros、R. Zubatyuk、O. Isayev 和 A. E. Roitberg，“将 ANI 深度学习分子势的适用范围扩展至硫和卤素”，《化学理论与计算杂志》，第16卷，第7期，第4192–4202页，2020年，pMID：32543858 [ ] 10 1021 0 00121 。在线获取：[8] https://doi.org/ 。 /acs.jctc. c S. Doerr、M. Majewsk、A. Pérez、A. Krämer、C. Clementi、F. Noe、T. Giorgino 和 G. D. Fabritiis，“Torchmd：用于分子模拟的深度学习框架”，2020年。[9] Eastman 等，“Openmm 8：使用机器学习势进行分子动力学模拟”，*arXiv*，第2310.03121卷，2023年，OpenMM 8：使用机器学习势进行分子动力学模拟。[10] D. Frenkel 和 B. Smit，《理解分子模拟》，第2版，计算科学丛书。美国加利福尼亚州圣迭戈：Academic Press，2001年10月。[11] GROMACS开发团队，神经网络势（nnpot），GROMACS，2025年，参考手册，神经网络势接口。[在线]。获取地址：https://manual.gromacs.org/nightly/reference-manual/special/nnpot.html [12] S. A. Hollingsworth 和 R. O. Dror，“人人可用的分子动力学模拟”，《神经元》，第99卷，第6期，第1129–1143页，2018年9月。[13] D. Jones、J. E. Allen、Y. Yang、W. F. Drew Bennett、M. Gokhale、N. Moshiri 和 T. S. Rosing，“用于生物分子经典分子动力学模拟的加速器”，《化学理论与计算杂志》，第18卷，第7期，第4047–4069页，2022年，pMID：35710099。[ ] 10 1021 1 01214 在线获取：[14] https://doi.org/ 。 /acs.jctc. c J. Jumper、R. Evans、A. Pritzel、T. Green、M. Figurnov、O. Ronneberger、K. Tunyasuvunakool、R. Bates、A. Žídek、A. Potapenko、A. Bridgland、C. Meyer、S. A. A. Kohl、A. J. Ballard、A. Cowie、B. Romera-Paredes、S. Nikolov、R. Jain、J. Adler、T. Back、S. Petersen、D. Reiman、E. Clancy、M. Zielinski、M. Steinegger、M. Pacholska、T. Berghammer、S. Bodenstein、D. Silver、O. Vinyals、A. W. Senior、K. Kavukcuoglu、P. Kohli 和 D. Hassabis，“利用 AlphaFold 实现高精度蛋白质结构预测”，《自然》，第596卷，第7873期，第583–589页，2021年7月。[15] C. Kutzner、V. Miletić、K. Palacio Rodríguez、M. Rampp、G. Hummer、B. L. de Groot 和 H. Grubmüller，“GROMACS分子动力学代码在高性能计算集群上扩展至65,000个CPU核心”，*Journal of Computational Chemistry*，第46卷，第5期，2025年2月。[16] R. R. L. Machado、J. Eitzinger、H. Köstler 和 G. Wellein，“MD-Bench：面向先进分子动力学算法的通用代理应用工具箱”，2022 [ ] 2207 13094 。在线获取：[17] https://arxiv.org/abs/ 。 M. Mirdita、K. Schütze、Y. Moriwaki、L. Heo、S. Ovchinnikov 和 M. Steinegger，“ColabFold：让所有人都能进行蛋白质折叠”，《自然·方法》，第19卷，第6期，第679–682页，2022年5月。[18] J. Morado、P. N. Mortenson、J. W. M. Nissink、J. W. Essex 和 C.-K. Skylaris，“机器学习势是否优于经过最优调参的传统力场？——以氟代醇为例的案例研究”，《化学信息与建模杂志》，第63卷，第9期，第2810–2827页，2023年，pMID：37071825。 [在线]。获取地址：https://doi.org/ 10 1021 2 01510 。 /acs.jcim. c [19] G. Morimoto、Y. M. Koyama、H. Zhang、T. S. Komatsu、Y. Ohno、K. Nishida、I. Ohmura、H. Koyama 和 M. Taijit，“在MDGRAPE-4A（一种用于分子动力学模拟的专用计算机系统）上对张量结构多层次埃瓦尔德求和法进行硬件加速”，载于*SC21*：国际高性能计算、网络、存储与分析会议，2021年，第1–15页。[20] M. Naderan-Tahan 和 L. Eeckhout，“Cactus：使用实际应用进行自顶向下GPU计算基准测试”，载于*2021*年*IEEE*国际工作负载特征分析研讨会（*IISWC*），2021年，第176–188页。[21] NVIDIA公司，*NVIDIA Nsight Compute*，https://developer.nvidia.com/nsight-compute，NVIDIA，2023年。[22] ——，*NVIDIA Nsight Systems*，https://developer.nvidia.com/nsight- systems，NVIDIA，2023年。[23] R. P. Pelaez、G. Simeon、R. Galvelis、A. Mirarchi、P. Eastman、S. Doerr、P. Thölke、T. E. Markland 和 G. D. Fabritiis，“TorchMD-Net 2.0：用于分子模拟的快速神经网络势”，2024年。[24] F. Peverelli、D. Confidconi、D. B. Bartolini、A. Scolari 和 M. D. Santambrogio，“商用平台上的分子动力学模拟特征分析”，第65–78页，2022年。[25] K. Santos、S. Moore、T. Oppelstrup、A. Sharifian、I. Sharapov、A. Thomp-son、D. Z. Kalchev、D. Perez、R. Schreiber、S. Pakin、E. A. Leon、J. H. Laros、M. James 和 S. Rajamanickam，“利用晶圆级系统突破分子动力学时间尺度瓶颈”，载于*SC24*：国际高性能计算、网络、存储与分析会议，2024年，第1–13页。

[26] D. E. Shaw、P. J. Adams、A. Azaria 等，“Anton 3：午餐前完成20微秒的分子动力学模拟”，载于*SC21*：国际高性能计算、网络、存储与分析会议，2021年，第1–11页。[27] J. S. Smith、O. Isayev 和 A. E. Roitberg，“ANI-1：一种可扩展的神经网络势，以力场计算成本实现密度泛函理论精度”，《化学科学》，第8卷，第3192–3203页，2017年。 [在线]。获取地址：http://dx.doi.org/10.1039/C6SC05720A [28] ——，“ANI-1：一种可扩展的神经网络势，以力场计算成本实现密度泛函理论精度”，《化学科学》，第8卷，第4期，第3192–3203页，2017年。[29] O. D. Team，“NNPOps：用于OpenMM的神经网络势运算”，https://github.com/openmm/NNPOps，2023年，访问日期：2025年5月20日。[30] P. Team，“PyTorch 性能分析器：PyTorch模型的性能调试工具”，2023年，https://docs.pytorch.org/docs/stable/profiler.html。[31] P. Thölke 和 G. D. Fabritiis，“用于基于神经网络的分子势的等变变换器”，载于国际学习表征会议，2022年。 [在线]。获取地址：https://openreview.net/forum?id=zNHzqZ9wrRB [32] O. T. Unke、S. Chmiela、M. Gastegger、K. T. Schütt、H. E. Sauceda 和 K.-R. Müller，“机器学习力场”，《化学评论》，第121卷，第16期，第10 142–10 186页，2021年。