arXiv:2303.06150v1 [cs.CE] 9 Mar 2023

# 利用输入和架构特征提升虚拟筛选应用的计算效率

Accordi Gianmarco
米兰理工大学电子、信息与生物工程系，意大利米兰
gi
anmarco.accordi@polimi.it

Vitali Emanuele
CSC – IT 科学中心，芬兰埃斯波 emanuele.vitali@csc.fi

Gadioli Davide
电子、信息与生物工程系 米兰理工大学 米兰，意大利
davide.gadioli@polimi.it

Crisci Luigi
计算机系 萨莱诺大学 萨莱诺，意大利
lcrisci@unisa.it

Cosenza Biagio
计算机系 萨莱诺大学 萨莱诺，意大利
bcosenza@unisa.it

Bisson Mauro 英伟达公司 圣克拉拉，加利福尼亚州，美国
maurob@nvidia.com

Massimiliano Fatica
英伟达公司 圣克拉拉，加利福尼亚州，美国
mfatica@nvidia.com

Beccari Andrea
Dompé farmaceutici SpA 那不勒斯，意大利
andrea.beccari@dompe.com

Gianluca Palermo
电子、信息与生物工程系 米兰理工大学 米兰，意大利
gianluca.palermo@polimi.it

## CCS 概念

• **计算方法** → **并行编程语言**； • **计算机系统组织** → **单指令、多数据**。

## 关键词

SYCL、CUDA、并行编程、虚拟筛选

**ACM 参考格式：**

Accordi Gianmarco、Vitali Emanuele、Gadioli Davide、Crisci Luigi、Cosenza Biagio、Bisson Mauro、Massimiliano Fatica、Beccari Andrea 和 Gianluca Palermo。2023年。利用输入和架构特征提高虚拟筛选应用的计算效率。载于 论文集。 ACM，美国纽约州纽约，5 页。
https://doi.org/XXXXXXX.XXXXXXX

## 1 引言

药物发现旨在找到一种能够对抗目标疾病并产生有益效果的小分子。它包括体外和体内阶段，这些阶段会增加成本和耗时，从而限制可评估候选药物的数量。近期研究表明，通过引入一个计算机模拟阶段，筛选出要进行体外测试的分子，可以提高找到药物的概率，[1, 6]。在这一阶段，我们使用虚拟筛选软件来估算一种小分子（称为配体）与靶蛋白之间的相互作用强度。我们可以利用这一数值为包含大量配体的化学库排序，并将最有希望的配体送入后续药物发现阶段。

通过模拟已知的化学反应，可以构建化学库。借助这种方法，我们能够探索庞大的化学空间。因此，化学库的规模仅受虚拟筛选可用计算资源的限制。正因如此，超级计算机成为虚拟筛选活动的目标系统 [3, 4]。当我们根据 TOP500[1] 榜单考察速度最快的超级计算机的节点架构时，会发现它们高度依赖加速器来提高吞吐量。尽管加速器通常是 GPU，但它们来自不同的供应商。因此，无法通过单一实现来使用加速器的原生语言。在这种情况下，紧急计算尤为重要，因为我们希望降低大流行病带来的社会和经济影响，因此虚拟筛选软件必须充分利用所有可用的计算资源。本文以 LiGen 虚拟筛选应用为现实案例进行研究[2, 9]。具体而言，我们引入了一个数据准备阶段，利用架构特征和输入特征来组织 CUDA 内核的输入。该阶段旨在提高图形处理器上的计算效率。此外，我们还将此功能移植到 SYCL 内核实现中。实验结果表明，这种内核外优化提升了两种实现的性能。虽然这些结果针对 LiGen，但也可能适用于其他设计相似的应用。

在第 2 节中，我们介绍虚拟筛选问题并描述 LiGen 的实现，以便更清楚地说明第 3 节中报告的内核外优化，以及第 4 节中描述的 SYCL 2020 移植。第5 节报告所提出优化的实验评估。最后，第 6节总结全文。

, ,©2023年计算机协会。本文为作者版本，仅供个人使用。请勿再分发。最终的记录版本发表于 *Proceedingsof* ,https://doi.org/XXXXXXX.XXXXXXX。

[1] https://www.top500.org/

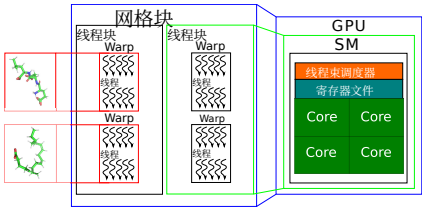



**图 1：配体计算在图形处理器逻辑架构和物理架构上的映射。**

## 2 背景与相关工作

对配体进行虚拟筛选需要完成三项主要任务。由于配体分子比蛋白质小得多，第一项任务旨在确定蛋白质中一个或多个希望与配体进行对接的区域。我们将这些区域分别称为对接位点。第二项任务旨在估算配体与目标对接位点相互作用时其原子的三维位移：也就是将配体对接到蛋白质上。最后，第三项任务评估对接后的配体与蛋白质之间的相互作用强度。因此，对于每个对接位点，我们都可以对输入的化学库进行排序，以帮助领域专家找出最适合进行体外测试的候选物。

虚拟筛选是文献中一个众所周知的问题，人们已提出、实现并评估了许多方法 [7, 10]。在紧急计算背景下，人们正致力于重新设计现有软件，以便在现代异构高性能计算节点上实现扩展并利用加速器 [3, 4],，从而显著提升性能。例如，AutoDock- Vina [8] 和 PLANTS [5]移植到 CUDA 后，观察到超过 50 倍的加速比。此前案例中可用的数据并行性与图形处理器架构十分契合。

### 2.1 LiGen 的实现

我们可以沿着两条主要路径利用图形处理器的并行性。最常见的方法是将计算分散到图形处理器上，尽可能缩短执行时间。AutoDock-GPU 和 LiGen 的开发者为针对 SARS-CoV-2 开展的超大规模虚拟筛选活动采用了这一路径 [4] [3] 。由于单个配体的计算与其他配体相互独立，第二种方法则将其计算限制在少量 GPU 线程上，从而使同一设备能够并行计算多个配体，如图1所示。与 CPU 相比，这种方法会延长单个配体的计算时间，但可以在一个批次中并行虚拟筛选更多配体。文献已分析过这种方法 [9], , 由于无需进行同步，其加速比最高可达5倍（相较于第一种方法）。

从实现角度来看，LiGen 使用多个 CPU线程（称为工作线程）启动计算内核，并采用双缓冲技术隐藏与设备之间的数据传输。具体来说，当一个工作线程收到一批配体时，它会开始将数据复制到图形处理器上第一个可用的缓冲区。随后，它会等待图形处理器空闲，并以互斥方式执行代码。最后，它将结果复制到 CPU内存，并释放缓冲区供其他工作线程使用。系统中的每个可用图形处理器都可以采用这种方式。

## 3 内核外优化

图形处理器内核的硬件需求（例如寄存器和共享内存）很大程度上取决于配体的原子数。问题在于，内核必须按最坏情况预留内存，这会严重影响性能，因为原子数可能从不到 20 个到几百个不等。我们为每一“类”配体使用不同的内核来解决这一问题：将内核可处理的最大原子数设为非类型模板参数，并在运行时选择正确的内核版本。这样，我们就能为原子数相近的一批配体实例化专用内核，从而尽量减少资源浪费。此外，为了进一步提高效率，我们希望一批中的所有配体具有相近的执行时间，也就是说，负载均衡。我们可以从算法看出，其复杂度与原子数和旋转异构体 [3]密切相关。后者是分子原子中的一个子集，这些原子可以旋转并改变几何形状，同时不改变其物理和化学性质。

因此，我们需要将原子数和旋转异构体数量相近的配体分成若干桶。为此，我们引入一个运行时输入准备阶段，在进行虚拟筛选之前将配体收集到各个桶中。这种方法涉及两个设计选择：*i)* 每个桶中放多少个配体，以及*ii)* 需要多少个簇。

为了解决第一个问题，LiGen 使用 cudaOccupancyMaxActiveBlocksPerMultiprocessor 函数查询 CUDA 运行时，以计算有多少个线程块 $b$ 可以在同一流式多处理器 (SM) 上运行。然后，我们按如下方式即时计算每个桶中的配体数量 $l$ :

$$l = b \times SM \times \frac{t}{ws} \tag{1}$$

其中， $SM$ 表示图形处理器上可用的流式多处理器数量， $t$ 表示一个 CUDA 线程块中的 CUDA 线程数量， $ws$ 表示线程束大小。由于我们使用一个线程束处理一个配体， $t$ 是一个可调参数，用于指定我们希望在一个 CUDA 线程块中计算多少个配体。在当前版本中，我们在每个 CUDA 线程块中对一个配体进行虚拟筛选。我们将在第5.2节评估改变每个桶中配体数量的影响。

为解决第二个设计选择问题，我们认为这两种聚类标准彼此正交，因为它们之间的关联较弱。因此，从原子和旋转异构体数量上限出发，我们需要定义一种启发式方法来生成中间值。生成簇时会利用原子数量和线程束大小，在 CUDA 线程与硬件资源之间平衡计算负载。例如，如果希望将原子划分为三个簇，就会把最多含有32 和 64 个原子的配体分别归入同一批次，并将其余配体归入最后一个批次。对于旋转异构体，我们采用一种启发式方法，将数值范围不均匀地划分。具体而言，我们更倾向于在较低数值范围内创建簇，因为这些范围会导致最大的相对负载不均衡。我们在第5.3节评估这一设计参数的影响。

## 4 SYCL 移植

上一节所述的内核外优化会对输入进行预处理，以提高图形处理器上的计算效率。

虽然该方法不依赖于内核实现，但我们会利用 CUDA 运行时的信息来计算每个批次中需要包含多少个配体。此功能在 SYCL2020[2] 中的移植使用了标准中的内核捆绑包信息。因此，我们需要按如下方式计算每个批次中的配体数量 $l$：

$$l = \frac{wgs}{t} \times CU \times \frac{t}{sgs} \tag{2}$$

其中 $wgs$ 是我们可以在单个计算单元（CU），即流式多处理器上执行的最大工作项数，也就是 CUDA 线程数。$CU$ 是 CU 的数量，而 $sgs$ 是子组的最大大小，也就是线程束大小。最后，$t$ 是工作组大小，也就是一个 CUDA 线程块中的 CUDA 线程数。尽管后者可以简化，我们仍将公式 2 写成与公式 1中的术语相匹配的形式。我们可以使用 SYCL 2020 API 查询 $wgs$、$sgs$和 $CU$。具体而言，$wgs$ 是 内核_设备_specific::work_group_size属性。参数 $t$ 与 CUDA 实现中的参数取值相同。遗憾的是，如果 SYCL 编译器不符合 SYCL 2020 标准，用户就需要手动进行调优。

## 5 实验结果

本节报告内核外优化效果实验的结果。

### 5.1 实验设置

所有实验均使用 IT4I 的 Karolina[3] 超级计算机上的单个节点。该节点配备两颗 AMD EPYC 7763 @ 2.53GHz 处理器和八块 NVIDIA A100 GPU。我们使用支持两种内核实现的 LiGen 5.0。编译 C++ 和 SYCL 源代码时，我们使用 Intel(R) oneAPI DPC++/C++ Compiler 2023.0.0；编译 CUDA 源代码时，则使用 NVCC 11.7.99。

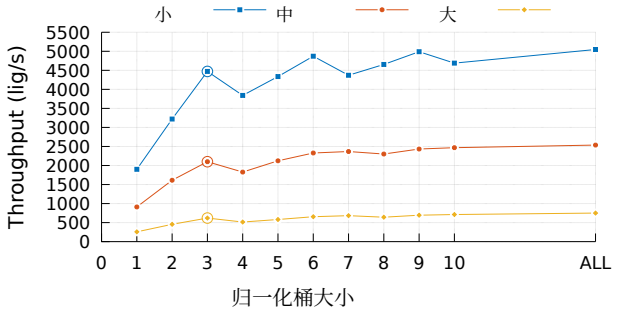
我们使用应用程序的平均吞吐量来衡量其性能，因为这是一种典型的高性能计算批处理作业。测量吞吐量时，我们用处理的分子数量除以实际运行时间。

### 5.2 优化桶大小

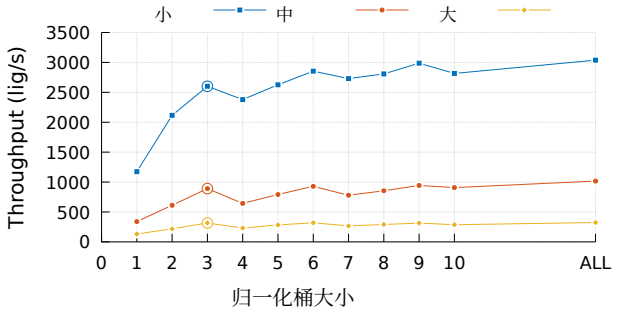
在此实验中，我们根据批次中的配体数量来衡量计算效率。为降低不同特征（例如旋转异构体数量）的影响，我们重点考察三种配体：“小”“中”和“大”。它们分别含有 17、53 和 99 个原子。我们将每种配体复制一千万次，并在靶蛋白的十二个对接位点上对这些配体进行对接，以测得稳定的吞吐量。

图2 展示了使用 CUDA（图 2a）和 SYCL（图 2b）实现的实验结果。x 轴表示
表示每个批次中的配体数量。我们将一个批次中的配体数量归一化为方程1 或方程 2建议值的三分之一，如表 1所示。我们希望每隔三个点最大限度地提高图形处理器利用率。此外，我们还评估了能够装入图形处理器内存的最大配体数量下的应用程序性能，并将其标记为“全部”。

从实验结果中可以观察到两个阶段：公式所建议的数值之前和之后。在达到这个数值之前，




**(a) CUDA 实现**




**(b) SYCL 实现**

**图 2：对于三类配体，改变批次中配体数量时 LiGen 的平均吞吐量。**

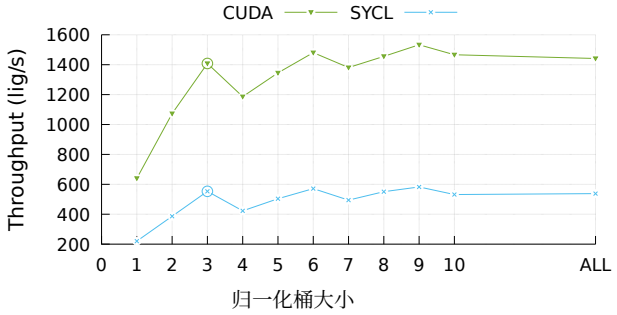



**图 3：使用配体混合数据集时，改变簇数量得到的 LiGen 平均吞吐量。**

应用吞吐量会随着配体数量的增加而提高。这是因为活动线程束的数量不断增加，SM调度器可以利用这些线程束来隐藏计算停顿。超过这个数值后，性能会有所下降，因为我们处理的是相同的配体，其计算所需的时间也大致相同。随后，随着创建的活动线程束增多，吞吐量又会提高，周而复始。对于小配体，这种模式在 SYCL 和 CUDA 实现中更为明显；而对于中、大配体，吞吐量的变化则更加平缓。

吞吐量趋势表明，应使用一个配体数量的倍数，该倍数等于根据公式计算出的值 1 或公式 2。理想情况下，我们希望使用 GPU

[2] https://registry.khronos.org/SYCL/specs/sycl-2020/html/sycl-2020.html [3] https://www.it4i.cz/en/infrastructure/karolina

在 GPU 内存允许的范围内。然而，配体的内存占用会随着虚拟筛选活动中发现的对接位点数量线性增长。因此，每个批次中的配体数量过多，将会根据目标 GPU 的容量限制我们能够考虑的对接位点数量。

为了在更真实的场景中衡量性能，我们使用第 5.3 节所报告的异构数据集。图 3 展示了应用在 CUDA 和 SYCL 实现下的性能。为实现这一目标，我们针对原子数量使用 6 个簇，针对旋转异构体使用 23 个簇。实验结果证实了该执行模式，表明使用不同输入特征、通过公式 1 和公式 2 计算配体数量是可靠的。

在比较 CUDA 和 SYCL 的性能时，我们分析了计算密集程度最高的内核，该内核占总执行时间的 90% 以上，即对接内核，因为我们注意到其性能下降幅度大于预期。事实上，SYCL 中主内核的执行时间比 CUDA 慢 12%，而应用吞吐量的差距更大。这是由于 SYCL 实现具有更高的寄存器压力，如表 1 所示，从而降低了内核之间可实现的并行性。因此，SYCL 能够并行处理的配体数量更少。在该节点上可用的八个 GPU 的使用效率在两种实现之间相同

## 5.3 测量计算效率

本实验旨在测量聚类过程对性能的影响。我们通过在一个包含异质配体的数据集上开展虚拟筛选活动来实现这一目标。输入数据集包含 1000 万个不同的配体，其原子数介于20 至 120之间，每个配体含有 0 至 20 个旋转异构体。我们使用公式 1 和公式 2 来设置每个批次的大小。图 4 展示了应用吞吐量的加速比随旋转异构体的簇数（x 轴）和原子数的簇数（y 轴）变化的情况。我们以两项特征均仅使用一个簇作为基线。

实验结果表明，采用所分析的内核外优化，可以使应用吞吐量大致翻倍。此外，x 轴和 y 轴上的变化趋势表明，当批次中的配体具有相似特征时，计算效率会有所提升。而且，这种优化带来的收益在两种内核实现中都一致。

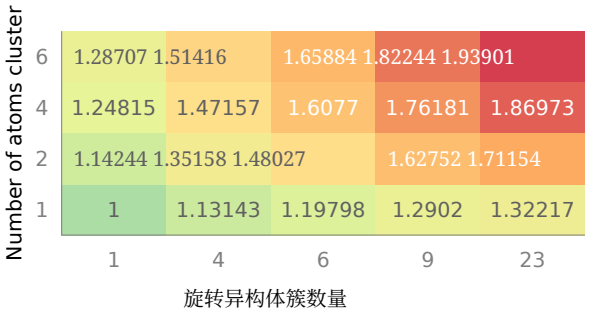
然而，增加簇的数量时，输入数据集会被分散到更多批次中。因此，执行结束时半空批次的数量会增加，从而降低计算效率。因此，要从细粒度聚类中获益，我们需要大量配体。这一限制并不

**表 1：通过改变原子数量上限，对 CUDA 和 SYCL 内核实现进行比较。**

| 原子数 | CUDA | | SYCL | |
|---|---|---|---|---|
| | 寄存器 | 活动块 | 寄存器 | 活动块 |
| 32 | 104 | 1728 | 160 | 1296 |
| 64 | 91 | 2160 | 174 | 864 |
| 96 | 124 | 1728 | 178 | 864 |




**(a) CUDA 实现**




**（b）SYCL 实现**

**图 4：通过改变簇的数量来提高应用吞吐量的加速比，同时针对配体中的原子数和旋转异构体数进行调整。**

会影响超大规模虚拟筛选活动，但对于日常使用而言则是相关因素。

## 6 结论

本文探讨了输入数据准备阶段的影响，该阶段利用输入和架构特征来提高图形处理器上的计算效率。我们以一款实际应用中的虚拟筛选应用为案例研究，评估了这些特征的影响及其局限性。不过，对于设计相似的应用，也可以采用相同的方法。

从实验结果中可以看出，计算效率能够得到提升，应用吞吐量几乎翻倍。无论是 SYCL 还是 CUDA 实现，都能获得这一性能提升。不过，要达到这一效率水平，需要规模较大或同质化程度较高的输入集。这并非问题，因为只有在极大规模的虚拟筛选活动中，我们才需要实现最高效率。

## 致谢

本节因双盲评审而省略。

## 参考文献

[1] Marcello Allegretti, Maria Candida Cesta, Mara Zippoli, Andrea Beccari, Carmine Talarico, Flavio Mantelli, Enrico M Bucci, Laura Scorzolini 和 Emanuele Nicastri。2022 年。将雌激素受体调节剂雷洛昔芬重新用于治疗 SARS-CoV-2 感染。《细胞死亡与分化》29, 1（2022），156–166。

[2] Luigi Crisci、Majid Salimi Beni、Biagio Cosenza、Nicolò Scipione、Davide Gadioli、Emanuele Vitali、Gianluca Palermo 和 Andrea Beccari。2022 年。借助 SYCL 2020 构建便携式药物发现流水线。收录于 *International Workshop on OpenCL*。[3] Davide Gadioli、Emanuele Vitali、Federico Ficarelli、Chiara Latini、Candida Manelfi、Carmine Talarico、Cristina Silvano、Carlo Cavazzoni、Gianluca Palermo 和 Andrea Rosario Beccari。2022 年。EXSCALATE：面向多靶点药理学、用于对抗 SARS-CoV-2 的超大规模药物发现虚拟筛选平台。*IEEE Transactions on EmergingTopics in Computing*（2022），1–12。[4] Jens Glaser、Josh V Vermaas、David M Rogers、Jeff Larkin、Scott LeGrand、Swen Boehm、Matthew B Baker、Aaron Scheinberg、Andreas F Tillack、Mathialakan Thavappiragasam、Ada Sedova 和 Oscar Hernandez。2021 年。利用海量数据集实现药物发现的高通量虚拟实验室。*The International Journal of High Performance Computing Applications* 35, 5（2021），452–468。[5] Oliver Korb、Thomas Stützle 和 Thomas E. Exner。2011 年。利用图形处理器加速分子对接计算。*Journal of Chemical Infor- mation and Modeling* 51, 4（2011），865–876。https://doi.org/10.1021/ci100459b

[6] Hans Matter 和 Christoph Sotriffer。2011。《虚拟筛选的应用与成功案例》。John Wiley & Sons, Ltd，第12章，319–358。[7] Nataraj S. Pagadala、Khajamohiddin Syed 和 Jack Tuszynski。2017。分子对接软件：综述。*Biophysical Reviews* 9, 2 (2017), 91–102。[8] Shidi Tang、Ruiqi Chen、Mengru Lin、Qingde Lin、Yanxiang Zhu、Ji Ding、Haifeng Hu、MingLing 和 Jiansheng Wu。2022。利用 GPU 加速 AutoDock Vina。*Molecules* 27, 9 (2022), 3041。[9] Emanuele Vitali、Federico Ficarelli、Mauro Bisson、Davide Gadioli、Massimil- iano Fatica、Andrea R. Beccari 和 Gianluca Palermo。2022。药物发现中基于分子对接的虚拟筛选 GPU优化方法：比较分析。[10] Elizabeth Yuriev、Jessica Holien 和 Paul A. Ramsland。2015。分子对接的改进、趋势与新思路：2012—2013年回顾。*Journal of Molecular Recognition* 28, 10 (2015), 581–604。