

ПОИСК ВЗАИМОДЕЙСТВУЮЩИХ АТОМОВ В ОЦЕНОЧНОЙ ФУНКЦИИ С ИСПОЛЬЗОВАНИЕМ GPU-ВЫЧИСЛЕНИЙ

С. В. Полуян^{1,*}, *Н. М. Ершов*²

¹ Государственный университет «Дубна», Дубна, Россия

² Московский государственный университет им. М. В. Ломоносова, Москва, Россия

В данной работе рассматривается функция для оценки энергии взаимодействия белок-белковых комплексов. Приведены результаты сравнения различных подходов к начальному этапу вычисления энергии взаимодействия, включающему выбор всех пар атомов, находящихся в пределах заданного порогового расстояния. Проанализированы два подхода с использованием CUDA-ядер и тензорных ядер с учетом специфики задачи.

In this study, we consider a scoring function for evaluating the interaction energy of protein–protein complexes composed of two all-atom proteins. We present the results of a comparison of different approaches for the initial stage of interaction energy evaluation, which includes selecting all atom pairs located within a predefined cutoff. We analyze two approaches based on the use of CUDA and Tensor cores, taking into account the specifics of the task.

PACS: 02.70.–с; 07.05.Tr

ВВЕДЕНИЕ

Первым этапом вычисления оценочной функции при оценке энергии взаимодействия белок-белкового комплекса с использованием потенциалов [1] является вычисление межатомных расстояний для всех пар атомов, находящихся в пределах заданного порога. Формально данная задача может быть поставлена следующим образом.

Пусть A и B — множества в трехмерном пространстве, содержащие n и m точек соответственно. Поиск для одной точки $q \in B$ состоит в нахождении всех точек $a_i \in A$, таких, что $d(q, a_i) < \varepsilon$, где $d(\cdot, \cdot)$ — евклидово расстояние, а ε — заданный порог. Поиск всех пар точек из двух множеств, расстояние между которыми меньше заданного порога, соответствует операции distance similarity join (DSJ) [2]. Найденные пары сохраняются в результирующем множестве.

* E-mail: poluyan@uni-dubna.ru

КОНТЕКСТ ЗАДАЧИ

Оценочная функция разработана для использования в методе выборки переходных поверхностей [3]. Процесс связывания комплекса рассматривается в рамках модели ключ–замок: белки рассматриваются в виде твердых тел, без каких-либо структурных изменений; белки совершают только поступательные и вращательные движения. Для оценочной функции [1] при выполнении операции DSJ необходимо сохранять межатомные расстояния между всеми парами атомов в пределах заданного порога для последующей оценки энергии на основе физических потенциалов.

Таким образом, одно множество точек остается фиксированным без смещения и вращения, а для вычисления потенциалов требуются только квадраты расстояний, которые хранятся в матрице расстояний для дальнейших вычислений парных потенциалов с использованием GPU.

СРАВНЕНИЕ ВРЕМЕНИ ВЫПОЛНЕНИЯ

В простейшем случае задача требует полного перебора всех пар точек с временной сложностью $O(n \times m)$. Для построения точного множества взаимодействующих пар могут применяться различные подходы, включая древовидные структуры данных, такие как деревья k-d и AABB, а также специальные алгоритмы, например Super-EGO [4]. В подходах ключевым критерием является время выполнения.

В качестве тестового примера рассматривается траектория связывания из 10^3 позиций при образовании комплекса с пороговым значением 14 Å для определения взаимодействующих атомов. Множества A и B содержат 7030 и 2790 точек соответственно. В первой позиции нет ни одной пары точек, расстояние между которыми меньше порога, а в последней таких пар около 150 000. В исследовании показана реализация и проведено сравнение метода полного перебора (версии CPU и GPU), древовидных подходов с однократным построением дерева для множества A , а также модифицированного алгоритма Super-EGO.

Первая задача заключалась в подсчете общего числа взаимодействующих пар для всей траектории. На рис. 1 видно, что время выполнения при полном переборе (Brute Force) остается постоянным.

Вторая задача является расширением первой, в которой требуется сохранять попарные расстояния на каждом шаге траектории. Поскольку потенциалы вычисляются на GPU, в исследовании рассматривается метод полного перебора с использованием тензорных и CUDA-ядер.

Метод полного перебора был реализован двумя способами. В первом случае использовались только CUDA-ядра. Оба множества задавались в виде многомерных матриц, размеры которых кратны 16. Это позволило вычислять матрицу расстояний блочно, с попарными расчетами расстоя-

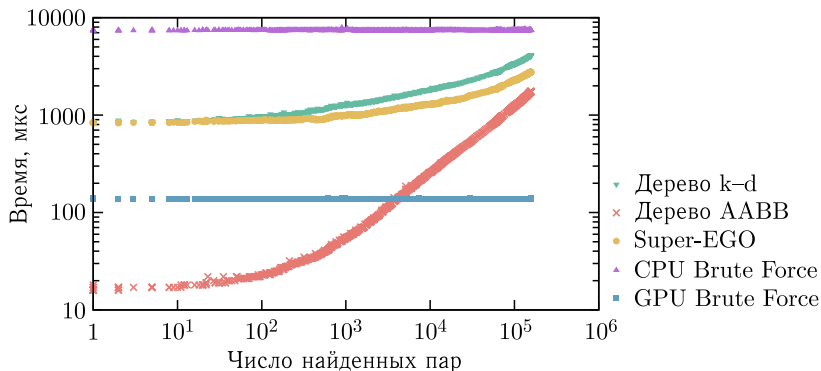


Рис. 1. Время поиска в зависимости от числа найденных пар

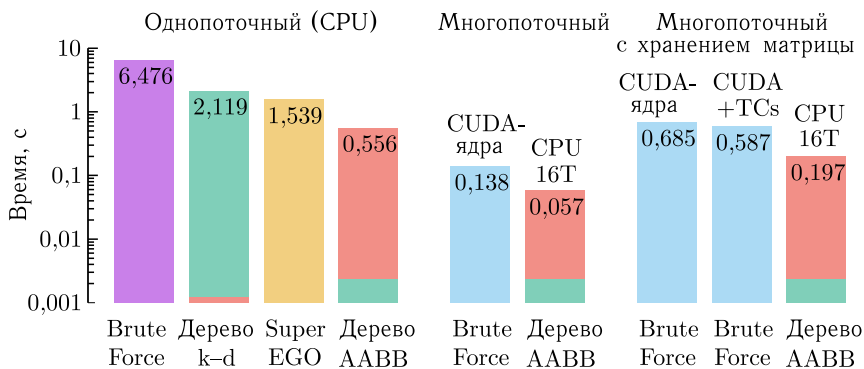


Рис. 2. Сравнение времени выполнения для траектории из 1000 позиций

ний и параллелизмом на уровне блоков потоков в двумерном индексном пространстве. Во втором случае помимо CUDA-ядер использовались тензорные аналогично [2].

При использовании тензорных ядер в матричном виде вычисление расстояний между двумя множествами A и B выполняется в два шага [2]. Первый шаг с использованием тензорных ядер: $T = A' \times B + C$, где $A' = -2 \times A$ и $C = \|B_j\|^2$. Второй шаг с использованием CUDA-ядер: $D = T + \|A_i\|^2$, где D — матрица попарных квадратов расстояний.

Матричные вычисления включают два этапа. На первом этапе выполняется GEMM (GEneral Matrix Multiply) с использованием тензорных ядер. На втором этапе применяются CUDA-ядра для вычисления суммы двух матриц, в результате чего формируется матрица расстояний — в данном случае матрица квадратов расстояний. Так как одно из множеств фиксировано, на первом этапе обновляются только матрицы B

и C , а на втором — только матрица T . Расстояние между точками p и q в d -мерном пространстве может быть записано следующим образом:

$$d(p, q) = \sqrt{\sum_{i=1}^d (p_i - q_i)^2} = \sqrt{\underbrace{p_1^2 + (-2p_1q_1 + q_1^2)}_{\text{CUDA cores}} + \dots + \underbrace{p_d^2 + (-2p_dq_d + q_d^2)}_{\text{CUDA cores}}}. \quad \text{Tensor cores}$$

На рис. 2 показано время для всех реализованных подходов. Для AABV-дерева использовалась параллельная реализация.

Все тесты проводились на системе с процессором AMD Ryzen 7 5700X (8 ядер, 16 потоков) и видеокартой NVIDIA GeForce RTX 3070 (CUDA 12.9, Compute Capability 8.6, 5888 CUDA и 184 тензорных ядер). Вычисления выполнялись с одинарной точностью.

ЗАКЛЮЧЕНИЕ

Использование тензорных ядер позволило сократить время выполнения на 14 %. Однако по сравнению с использованием только CUDA-ядер выявленные недостатки перевешивают преимущества.

Во-первых, данные не хранятся в оптимальном представлении для операций вращения и смещения. Это препятствует использованию параллелизма на уровне блоков потоков. Во-вторых, при использовании тензорных ядер две матрицы на первом этапе представляются в 16-битной точности, что приводит к потере точности. В-третьих, такая точность может приводить к потере дальнедействующих взаимодействий.

Так как задача относится к классу задач, ограниченных пропускной способностью памяти (memory-bound), для применения тензорных ядер существуют теоретические оценки предела ускорения [5]. Полученные результаты ускорения соответствуют диапазону этих оценок.

Следует отметить, что наилучшая производительность была достигнута при использовании AABV-дерева. Эта структура данных имеет только один параметр — максимальное число дочерних узлов, которое в экспериментах было зафиксировано значением 32. Реализации методов, тестовая задача и дополнительные материалы доступны по адресу [6].

СПИСОК ЛИТЕРАТУРЫ

1. Poluyan S., Nikulin D., Ershov N. Development and Verification of a Score Function for Estimation of Intermolecular Interactions in Protein-Protein Complexes // Proc. Intern. Conf. "Information and Telecommun. Technol. and Math. Modeling" (ITTMM). 2023. P. 231–235; <https://events.rudn.ru/event/198/attachments/550/1474/ittmm-2023.pdf> (accessed 15.09.2025).
2. Gallet B., Gowanlock M. Leveraging GPU Tensor Cores for Double Precision Euclidean Distance Calculations // 29th IEEE Intern. Conf. on High Performance Computing, Data, and Analytics. arXiv:2209.11287. 2022.

3. *Poluyan S., Ershov N.* Using Parallel Computing in the Transition Interface Sampling Method Implementation // Numerical Meth. Programming. 2024. V. 25, No. 4.
4. *Kalashnikov D. V.* Super-EGO: Fast Multi-Dimensional Similarity Join // Intern. J. Very Large Data Bases. 2013. V. 22. P. 561–585.
5. *Zhang L., Huang J., Di S., Matsuoka S., Wahib M.* Can Tensor Cores Benefit Memory-Bound Kernels? (NO!) // Proc. 17th Workshop on General Purpose Processing Using GPU. 2025. P. 28–34.
6. Repository with Supplemental Materials. <https://vcs.uni-dubna.ru/dsj/test> (accessed 15.09.2025).