МОДЕЛИРОВАНИЕ РОСТА НАНОСТРУКТУР МЕТОДОМ МОЛЕКУЛЯРНОЙ ДИНАМИКИ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ CUDA

А.М. Сатанин, Д.В. Федосеев, Д.В. Ахматнуров

Нижегородский госуниверситет им. Н.И. Лобачевского

Представлена реализация метода молекулярной динамики многокомпонентных систем с короткодействующим потенциалом, выполняемая полностью на CUDA. Алгоритм адаптирован для задач, связанных с физикой твёрдого тела.

Введение

Для изучения процессов образования наноструктур часто требуется моделировать динамику громадного числа взаимодействующих частиц. Типичный размер задачи делает необходимым применение высокопроизводительных вычислений. В данной работе рассмотрена реализация метода молекулярной динамики с использованием CUDA, разработанная для изучения процесса роста нанокристаллов кремния в оксидной матрице. Предложенный алгоритм адаптирован, прежде всего, для исследования процессов в твёрдом теле.

Поскольку взаимодействие атомов в твёрдых телах в основном описывается короткодействующими потенциалами, использовались оптимизации, ограничивающие число взаимодействующих атомов.

1. Описание задачи

Схема алгоритма представлена на рис. 1. Начальные положения атомов либо читаются из файла, либо генерируются. Для задания начальных скоростей использовано распределение Максвелла при заданной температуре. Граничные условия могут быть либо жёсткими, либо периодическими.

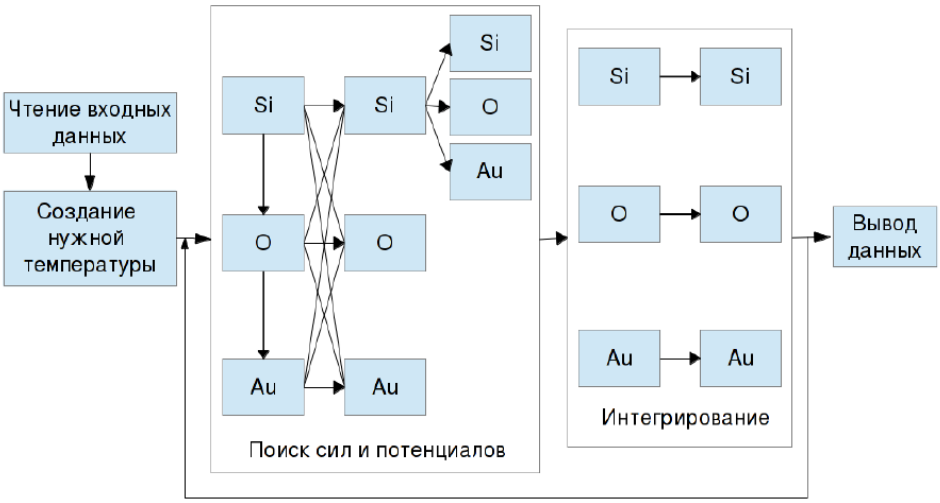

Рис. 1. Общая схема алгоритма

Итерацию моделирования молекулярной динамики можно разбить на два основных этапа:

1) Вычисление сил, действующих на каждый атом со стороны всех остальных атомов. Для моделирования многокомпонентных систем предусмотрена возможность задания разных потенциалов для описания взаимодействия разных пар атомов.
2) Интегрирование уравнений движения каждого атома.

Вычислительная сложность этапа нахождения сил зависит от типа используемого потенциала. В случае простейшего двухчастичного потенциала вида $\varphi_{ij} = f(r_i, r_j)$ она пропорциональна n^2. Вычислительная сложность этапа интегрирования – O(n), где n – число атомов.

Для описания взаимодействия атомов кремния использовался эмпирический потенциал [1], имеющий вид

$$\varphi_{ij} = \left(Ae^{-\alpha r_{ij}} - B\left(\sum_k \psi(\vec{r}_i, \vec{r}_j, \vec{r}_k) \right) e^{-\beta r_{ij}} \right) f_{cut}(r_{ij}), \qquad (1)$$

где $f_{\text{cut}}(r)$, $B(x)$ и $\psi(x)$ – некоторые функции. Известно, что $f_{\text{cut}}(r)$ обращается в нуль при $r > R_{\text{cut}}$. Потенциал (1) является разновидностью потенциала Терсова [2], учитывающего влияние соседних атомов на характеристики связи. Наличие суммы по всем атомам в (1) приводит к увеличению вычислительной сложности этапа суммирования до O(n^3). Таким образом, наиболее ресурсоёмким этапом является нахождение сил.

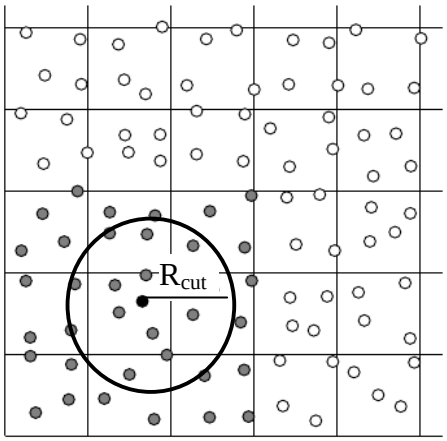

Рис. 2. Отсечение невзаимодействующих атомов

При оптимизации вычисления сил учтено, что потенциал обращается в нуль при $r > R_{\text{cut}}$. Наличие такого радиуса отсечения позволяет не рассчитывать взаимодействие с атомами, находящимися дальше R_{cut} от рассматриваемой. Применён подход spatial indexing [3], в котором всё пространство разбивается на кубические ячейки размером $d \geq R_{\text{cut}}$. Суммирование ведётся только по атомам, находящимся в ближайших ячейках (рис. 2). Это позволяет уменьшить сложность алгоритма до O(n) при изменении размеров системы при постоянной плотности.

Рассмотрим этапы вычисления сил:

1) Для каждого атома вычисляется индекс ячейки, в которой он находится (пространственный хэш). Сложность этого этапа – O(n). На данном этапе присутствует параллелизм по данным, поэтому он идеален для распараллеливания.
2) Сортировка массива атомов по их пространственному хэшу. Данная операция выполняется на GPU функцией из библиотеки thrust. Сложность O($n \log(n)$).
3) Поиск минимальных и максимальных индексов атомов, принадлежащих каждой ячейке. Сложность O(n). Выполняется на GPU, поддерживающем операции atomicMin и atomicMax в глобальной памяти, либо на CPU.

4) Нахождение сил с учётом отсечения. Сложность O(n) (при фиксированной плотности атомов), выполняется на GPU. В нашей реализации это самая длительная операция, занимающая до 80% от общего времени.

Поскольку положение атомов не может сильно измениться за один шаг, результаты этапов 1–3 могут быть использованы на нескольких шагах интегрирования.

2. Описание программы

Программа реализована на языках CUDA C и C++. Использовались контейнеры и алгоритмы из библиотеки thrust, в вспомогательных целях использовались библиотеки STL и boost.

Реализовано две версии алгоритма вычисления сил: с использованием spatial indexing и без. Для предотвращения “комбинаторного взрыва”, когда число необходимых подпрограмм вычисления сил равно произведению числа потенциалов на число версий алгоритма, использовалась эмуляция функций высшего порядка с помощью шаблонов C++.

Литература

1. Umeno Y. at el. Optimization of interatomic potential for Si/SiO 2 system based on force matching // Computational materials. 2002. V. 25. P. 447–456.
2. Tersoff J. New Empirical Model for the Structural Properties of Silicon // Physical Review Letters. 1986. V. 56. P. 632–635.
3. Goswami P., Schlegel P., Solenthaler B. Interactive SPH Simulation and Rendering on the GPU // Proceedings of the 2010. P. 1–10.