УДК 004.021+519.683+519.684

УСКОРЕНИЕ МОЛЕКУЛЯРНО-ДИНАМИЧЕСКИХ РАСЧЕТОВ С ПОМОЩЬЮ БЫСТРОГО МЕТОДА МУЛЬТИПОЛЕЙ И ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Д. Ф. Марьин[1], В. Л. Малышев[1], Е. Ф. Моисеева[1], Н. А. Гумеров[2], И. Ш. Ахатов[3], К. И. Михайленко[1]

Работа посвящена ускорению моделирования методами молекулярной динамики электростатических потенциалов на примере молекул воды. Высокопроизводительные вычисления достигаются путем комбинированного подхода с применением быстрого метода мультиполей (FMM) для вычисления сил и применением гетерогенных архитектур, состоящих из центральных и графических процессорных устройств (CPU и GPU соответственно). Метод FMM позволяет ускорить вычисления дальнодействующих взаимодействий за счет уменьшения вычислительной сложности до линейной. Использование GPU позволяет достичь значительных ускорений в вычислениях. Реализация метода FMM на GPU предоставляет возможность проводить численные эксперименты над большими системами. Показано, что предложенная методика имеет хорошую масштабируемость и может быть использована для моделирования динамики воды в области с размерами 60 нм или числом молекул воды порядка 10 миллионов на персональных рабочих станциях, оборудованных одним GPU.

Ключевые слова: молекулярная динамика, электростатическое взаимодействие, быстрый метод мультиполей, FMM, гетерогенные вычисления, GPU.

1. Введение. Детальное изучение макроскопических явлений на молекулярном уровне вызывает интерес различных научных сообществ. Для такого типа исследований одним из наиболее подходящих инструментов стали методы молекулярной динамики (МД). Эти методы могут применяться для исследования динамических свойств газов, жидкостей, твердых тел и взаимодействий между ними [1, 2]. Задачи тепломассопереноса, кавитации и зародышеобразования вблизи твердой стенки исследовались при помощи методов МД [3, 4].

Несмотря на то что методы МД успешно применяются для исследования подобных задач, остается открытым вопрос, касающийся моделирования реальных физических процессов. Применение методов МД для исследования макроскопических явлений требует рассмотрения большого количества молекул или наблюдения за поведением системы в течение долгого времени, что приводит к значительным вычислительным затратам. Так, например, область размером $100 \times 100 \times 100$ нм содержит более 30 миллионов молекул воды. Большинство существующих вычислительных рабочих станций не способны рассчитать все парные взаимодействия для таких больших систем. Ускорение расчетов в молекулярной динамике может быть достигнуто как благодаря использованию современного высокопроизводительного аппаратного обеспечения, так и благодаря использованию высокоэффективных численных алгоритмов.

Первый способ ускорить расчет сил взаимодействий — использование графических процессоров (Graphics Processing Unit, GPU). Одна из первых реализаций МД-моделирования леннард–джонсовских систем с использованием GPU была представлена Андерсоном в 2008 г. [5]. Позднее Сунарсо и др. представили реализацию МД-моделирования на GPU по исследованию жидких кристаллов, обладающих анизотропными свойствами [6].

Использование гетерогенных вычислительных систем, состоящих из CPU и GPU, позволяет достигать значительного сокращения времени вычислений, особенно для задач типа N-тел [7]. Графические процессоры состоят из множества арифметических модулей, работающих параллельно, что позволяет

[1] Башкирский государственный университет, Центр микро- и наномасштабной динамики дисперсных систем (ЦМНДДС, БашГУ); ул. Заки Валиди, 32, 450076, г. Уфа; Д. Ф. Марьин, стажер-исследователь, e-mail: marcom.dimar@gmail.com ; В. Л. Малышев, стажер-исследователь, e-mail: victor.l.malyshev@gmail.com; Е. Ф. Моисеева, стажер-исследователь, e-mail: elena.f.moiseeva@gmail.com; К. И. Михайленко, науч. сотр., e-mail: const.mkh@gmail.com

[2] Университет штата Мэриленд, США, Институт передовых компьютерных исследований (UMIACS), Room 3305 A.V. Williams Building, College Park, MD 20742; профессор, e-mail: gumerov@umiacs.umd.edu

[3] Университет штата Северная Дакота, США (NDSU), NDSU Dept 2490, 210 Dolve Hall, P.O. Box 6050, Fargo, ND 58108; профессор, e-mail: Iskander.Akhatov@ndsu.edu

значительно увеличить вычислительную производительность при меньших затратах по сравнению с CPU (соотношение производительности к стоимости и потребляемой энергии). Однако использование GPU не позволяет избавиться от квадратичной сложности алгоритма.

Наиболее интенсивная часть МД-моделирования — это расчет сил кулоновского взаимодействия между частицами. При использовании алгоритма прямого суммирования (расчет всех парных взаимодействий) вычислительная сложность алгоритма оценивается как $O(N^2)$, где N — число частиц. Пожалуй, самым распространенным методом быстрого расчета кулоновского взаимодействия в бесконечной периодической системе частиц является метод суммирования Эвальда [8] и модификации этого метода, основанные на быстром преобразовании Фурье (см., например, [9]). Вычислительная сложность подобных методов оценивается как $O(N \log N)$, где N — число частиц в периодической ячейке. Однако моделирование непериодических систем также имеет смысл (см., например, [10, 11]). Примером может служить моделирование систем, находящихся в вакууме. Несмотря на то что существуют методы со сложностью $O(N)$ или $O(N \log N)$ для непериодических систем, такие как Barnes–Hut treecode [12], Particle–Mesh методы [8] и быстрый метод мультиполей (Fast Multipole Method, FMM) [13], их применение в молекулярной динамике существенно ограничено, что связано со сложностью реализации алгоритмов и относительно низкой скоростью по сравнению с быстрым преобразованием Фурье. Таким образом, алгоритмическое и аппаратное ускорения этих методов, в частности с использованием высокопроизводительных гетерогенных вычислительных систем, представляет существенный интерес.

В отличие от других вышеупомянутых методов, метод FMM имеет линейную вычислительную сложность с контролируемой вплоть до машинной точностью. Кроме того, в связи с меньшим числом коммуникаций этот метод гораздо лучше подходит для распараллеливания, чем быстрое преобразование Фурье, что особенно ценно для реализации на GPU. Впервые метод FMM, распараллеленный на GPU, был описан в [14]. Эта работа была развита в последующих публикациях, а метод FMM был реализован уже на кластере из GPU [15]. Комбинация вычислительной сложности этого метода и возможность его параллельной реализации делают его привлекательным алгоритмом в эру пета- и экзавычислений.

Настоящая статья посвящена вопросам совместного использования FMM и GPU. Впервые показывается применение FMM для моделирования методами МД на гетерогенных вычислительных системах, состоящих из CPU и GPU. В то же время алгоритм построения структуры данных может быть применим и в других иерархических методах или же в вычислениях взаимодействия частиц. Кроме того, описанные алгоритмы могут быть использованы не только в применении к GPU, но и к другим many-core системам, например Intel Xeon Phi.

2. Математическая модель. Молекулярная динамика — это метод, используемый для определения макроскопических свойств системы из N тел, в котором движение частиц подчиняется второму закону Ньютона. Движение твердого тела можно представить как сумму двух движений: поступательного движения со скоростью центра масс тела и вращательного движения относительно оси, проходящей через центр масс. В классической молекулярной динамике положения атомов вычисляются из начальных условий $(\boldsymbol{r}_0, \boldsymbol{v}_0)$ посредством решения уравнений движения

$$\frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{v}_i, \quad \frac{d\boldsymbol{v}_i}{dt} = \frac{\boldsymbol{F}(\boldsymbol{r}_i)}{m_i}, \quad \boldsymbol{F}(\boldsymbol{r}_i) = -\frac{\partial}{\partial \boldsymbol{r}_i}\boldsymbol{U}_i(\boldsymbol{r}^N), \quad \boldsymbol{I}_i\frac{d\boldsymbol{\omega}_i}{dt} = \boldsymbol{M}_i, \tag{1}$$

где $\boldsymbol{r}_i$ — положение центра масс i-й молекулы; m_i — ее масса; $\boldsymbol{v}_i$ — ее скорость; $\boldsymbol{r}^N = \{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\}$ — множество координат всех молекул; $\boldsymbol{U}_i(\boldsymbol{r}^N)$ — суммарный потенциал взаимодействия i-й молекулы со всеми остальными; $\boldsymbol{I}_i$ — момент инерции; $\boldsymbol{\omega}_i$ — вектор угловой скорости; $\boldsymbol{M}_i$ — момент сил. За исключением простейших случаев, уравнения (1) решаются численно согласно выбранному алгоритму (метод скоростей Верле, leapfrog или др.). Для интегрирования этих уравнений необходимо вычислить силу $\boldsymbol{F}(\boldsymbol{r}_i)$, действующую на центр масс i-й молекулы, которая отвечает за поступательное движение, и момент сил $\boldsymbol{M}_i$, соответствующий вращательному движению. Детальное описание такого подхода можно найти, например, в [16].

В качестве граничных условий рассматриваются отражающие стенки. Молекула, достигшая границы области моделирования, отражается внутрь области, при этом скорость частицы сохраняется, а угол отражения частицы равен углу падения. Для моделирования статистических свойств системы рассматривается NVT-ансамбль (Number Volume Temperature ensemble), в котором фиксирована кинетическая энергия молекул. Поддержание постоянной температуры осуществляется при помощи термостата Нозе–Хувера [17].

2.1. Модель воды. Компьютерное моделирование воды впервые рассмотрено в [18, 19]. Основной задачей при моделировании воды является выбор модели потенциала, описывающей взаимодействие молекул. Нами используется модель воды TIP4P. Подобная геометрия была предложена Йоргенсеном [20],

который определил параметры потенциала с целью наиболее точного воспроизведения энтальпии парообразования и плотности жидкой воды при комнатной температуре. Особенность этой модели заключается в том, что расположение частицы, имеющей отрицательный заряд (чаще всего обозначается M), не совпадает с положением кислорода, а лежит на биссектрисе угла H–O–H. Эта модель воды, изображенная на рис. 1, имеет плоскую конфигурацию и содержит 4 жестко связанных частицы, две из которых, обозначенные M и O, ассоциированы с ядром кислорода, и две — обозначенные H — с водородом. Величины, определяющие структуру молекулы воды, имеют следующие численные значения: $r_{OH} = 0.957$ Å; $r_{OM} = 0.15$ Å; $\angle_{HOH} = 104.5°$.

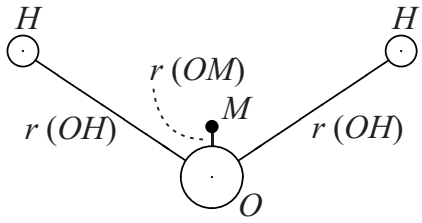
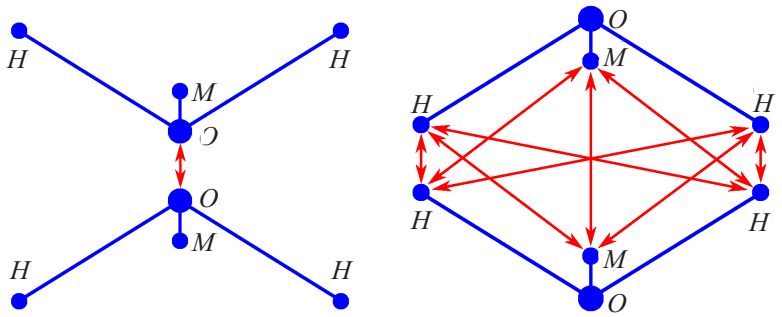


Рис. 1. Модель воды TIP4P



а) б)

Рис. 2. Взаимодействие атомов двух молекул TIP4P:
а) ближнее взаимодействие (потенциал Леннарда–Джонса),
б) дальнее взаимодействие (потенциал Кулона)

Таким образом, заряд кислорода смещается в другую точку; в результате кислород распадается на две воображаемые частицы, одна из которых (M) заряжена, но не имеет массы, а другая (O) — не заряжена и имеет массу кислорода. Энергия взаимодействия между двумя молекулами i и j представляет собой сумму энергий взаимодействия всех атомов, часть из которых взаимодействует согласно потенциалу Леннарда–Джонса (первое слагаемое в уравнении (2); рис. 2а), а часть — согласно закону Кулона (второе слагаемое в уравнении (2); рис. 2б):

$$u_{ij} = 4\varepsilon_w \left[\left(\frac{\sigma_w}{r_{OO}} \right)^{12} - \left(\frac{\sigma_w}{r_{OO}} \right)^{6} \right] + \sum_{k \in \{i \backslash O\}} \sum_{l \in \{j \backslash O\}} \frac{q_k q_l}{r_{kl}}. \tag{2}$$

Соответственно сила взаимодействия имеет вид

$$f_{ij} = 48\varepsilon_w \left[\left(\frac{\sigma_w^{12}}{r_{OO}^{14}} \right) - \frac{1}{2} \left(\frac{\sigma_w^{6}}{r_{OO}^{8}} \right) \right] \boldsymbol{r}_{OO} + \sum_{k \in \{i \backslash O\}} \sum_{l \in \{j \backslash O\}} \frac{q_k q_l}{r_{kl}^3} r_{ij}, \tag{3}$$

где q_l — заряд атома l; q_k — заряд атома k; r_{kl} — расстояние между атомами l и k двух различных молекул. Параметры потенциала Леннарда–Джонса в этой модели имеют значения $\sigma_w = 3.154$ Å и $\varepsilon_w = 1.07 \times 10^{-21}$ Дж, а заряды $q_O = -1.04|e|$ и $q_H = -q_O/2$. Ввиду быстро убывающей природы потенциала Леннарда–Джонса, при расчетах он обрезается по радиусу r_{cutoff}. За его значение можно брать разные величины [21]. В настоящей работе мы принимаем $r_{\text{cutoff}} = 5\sigma$.

3. Валидация модели воды. Термин “структура жидкости” весьма распространен. В отличие от кристаллической структуры твердого тела под структурой жидкости следует понимать статистическую закономерность межмолекулярных расстояний и ориентации, характерные для любой плотно упакованной системы. Благодаря конечному размеру молекул и силам межмолекулярного взаимодействия, любой жидкости свойственен ближний порядок в расположении частиц и отсутствие дальнего порядка. Отсутствие дальнего порядка означает, что порядок в одном месте никак не действует на порядок в другом, отдаленном от него месте. Удобным методом описания структуры простой жидкости является функция радиального распределения (ФРР), которая может быть измерена экспериментально на основании данных по рассеянию рентгеновских лучей и нейтронов.

Исследование структуры жидкости посредством ФРР было предложено в 1920 г. Дебаем и Менке [22]. Основной способ экспериментального определения ФРР — анализ картин, полученных с помощью дифракции рентгеновских лучей на изучаемом образце.

Процесс численного расчета ФРР следующий. Выбирается произвольный атом в системе. Строится серия концентрических сфер с центром в выбранном атоме. Расстояние между соседними сферами фиксировано и равно Δr (рис. 3). Через равные промежутки времени вычисляется и хранится количество атомов, находящихся внутри каждой оболочки. Вычисленное значение делится на объем соответствующей оболочки и среднюю плотность системы. Математически формула имеет вид: $g(r) = \dfrac{n(r)}{4\pi\rho r^2 \Delta r}$, где $g(r)$ — ФРР, $n(r)$ — число атомов, находящихся в оболочке толщиной Δr на расстоянии r, и ρ — плотность системы.

Численные результаты получены авторами настоящей статьи в процессе валидации кода. При моделировании были рассмотрены 8000 молекул воды структуры TIP4P (32 000 атомов), распределенных в начальный момент равномерно по области размером $63.45 \times 63.45 \times 63.45$ Å. В системе используется NVT-ансамбль с термостатом Нозе–Хувера, который поддерживает постоянную температуру, равную 298 K. Во всех направлениях используются периодические граничные условия. Временной шаг моделирования составлет 0.8 фемтосекунд. Согласно приведенной формуле ФРР вычисляются с 10 000 по 50 000 шаг алгоритма каждые 50 шагов с осреднением по всему набору атомов. Полученные результаты осредняются по всем временны́м шагам, на которых рассчитывались ФРР. На рис. 4 приведены результаты экспериментального и численного исследований ФРР для молекулы воды. Можно заметить, что результаты численного эксперимента хорошо согласуются с известными экспериментальными данными [23], что показывает справедливость рассматриваемой модели и ее реализации.

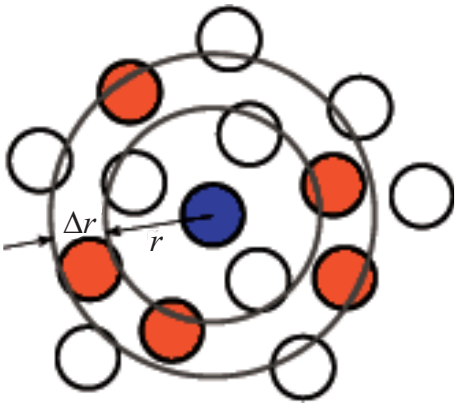
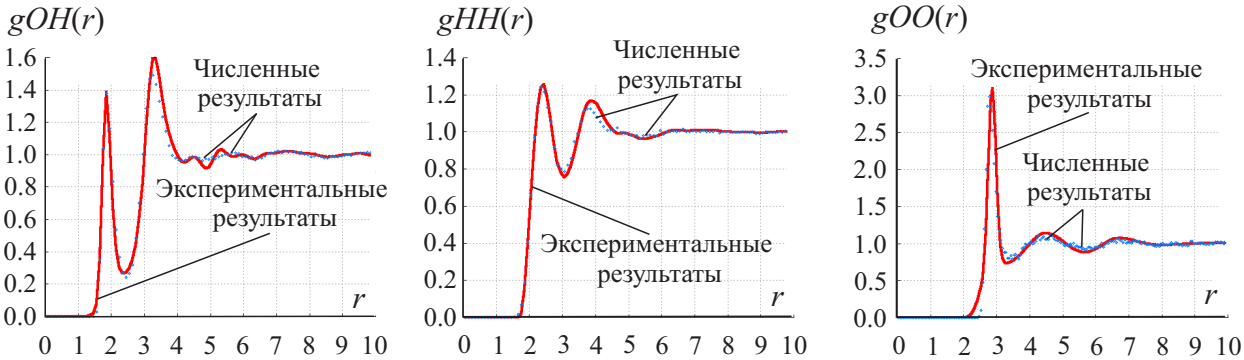


Рис. 3. Пространственная дискретизация для нахождения ФРР



а) б) в)

Рис. 4. Сравнение численного исследования функции радиального распределения (ФРР) для модели TIP4P (показано точками) с экспериментальными результатами (показаны сплошными линиями): а) ФРР атомов водорода относительно атома кислорода, б) ФРР атомов водорода относительно атома водорода, в) ФРР атомов кислорода относительно атома кислорода

4. Ускорение при помощи FMM/GPU.

4.1. Быстрый метод мультиполей. Быстрый метод мультиполей (Fast Multipole Method, FMM) впервые представлен в работе [13] и направлен на уменьшение вычислительной сложности расчета электростатического потенциала. Метод FMM был признан одним из 10 наиболее важных алгоритмов XX века [24]. В данной работе рассматривается многоуровневый FMM (Multilevel FMM, MLFMM).

Метод FMM — это высокоэффективный иерархический метод, основанный на идее о том, что влияние группы частиц $G = \{p_i : p_i \in \Gamma\}$ на рассматриваемую частицу p^* можно учесть путем аппроксимации совместного влияния G вместо прямого вычисления взаимодействия каждой частицы $p_i \in G$ с частицей p^* при условии, что область Γ и частица p^* находятся на отдалении друг от друга.

Метод FMM может быть использован для ускорения расчетов дальнего взаимодействия (электромагнитного и гравитационного). Этот метод имеет линейную вычислительную сложность при гарантированной точности (вплоть до машинной). Рассмотрим его применение для задачи расчета дальнего взаимодействия, описываемого потенциалом Кулона. В дальнейшем в этом разделе будет говориться только

о частицах, имеющих заряд. Ввиду слабого затухания электростатического взаимодействия, необходимо рассчитывать воздействие всех частиц друг на друга. Пусть необходимо оценить потенциал в точке $\boldsymbol{y}_j$, создаваемый N частицами с зарядами q_i и координатами $\boldsymbol{x}_i$:

$$\phi(\boldsymbol{y}_j) = \sum_{i=1,\, i\neq j}^{N} q_i \Phi(\boldsymbol{y}_j - \boldsymbol{x}_i). \tag{4}$$

Здесь $\Phi(r) = 1/|\boldsymbol{r}|$ — ядро Лапласа и q_i — заряд i-й частицы. Сумму (4) можно разбить на две суммы

$$\phi(\boldsymbol{y}_j) = \sum_{\boldsymbol{x}_i \in \Omega(\boldsymbol{y}_j),\, \boldsymbol{x}_i \neq \boldsymbol{y}_j} q_i \Phi(\boldsymbol{y}_j - \boldsymbol{x}_i) + \sum_{\boldsymbol{x}_i \notin \Omega(\boldsymbol{y}_j)} q_i \Phi(\boldsymbol{y}_j - \boldsymbol{x}_i) = \phi_{\text{near}}(\boldsymbol{y}_j) + \phi_{\text{far}}(\boldsymbol{y}_j), \tag{5}$$

где $\Omega(\boldsymbol{y}_j)$ — область, окружающая частицу $\boldsymbol{y}_j$. Метод FMM напрямую рассчитывает ближнее взаимодействие (ϕ_{near}) и аппроксимирует дальнее (ϕ_{far}) с заданной точностью. Таким образом, этот метод разделяет матрично-векторное произведение на разреженную (ϕ_{near}) и плотную (ϕ_{far}) части. В уравнении (5) член ϕ_{near} для всех частиц в системе может быть вычислен за MN операций, где M — среднее число частиц по всем областям $\Omega(\boldsymbol{y}_j)$, $j = 1, \ldots, N$. Для большой системы при оптимальном выборе глубины разбиения иерархической структуры данных получается $M \ll N$, поэтому вычислительная сложность расчета всех ближних взаимодействий будет $O(N)$. Для аппроксимации члена ϕ_{far} в уравнении (5) ядро аппроксимируется бесконечной суммой, в которой удерживается только p^2 первых членов разложения (p называется числом усечения). Для большинства задач число $p \leqslant 20$ является достаточным. Аппроксимация производится при помощи мультипольных (сингулярных) сферических базисных функций S_l и локальных (регулярных) сферических базисных функций R_l. В результате появляется возможность разбиения расчета ядра для наборов $\{\boldsymbol{x}_i\}$ и $\{\boldsymbol{y}_j\}$ и, тем самым, объединения операций для множества частиц следующим образом:

$$\begin{aligned} \phi_{\text{far}}(\boldsymbol{y}_j) &= \sum_{\boldsymbol{x}_i \notin \Omega(\boldsymbol{y}_j)} q_i \Phi(\boldsymbol{y}_j - \boldsymbol{x}_i) = \sum_{\boldsymbol{x}_i \notin \Omega(\boldsymbol{y}_j)} q_i \sum_{l=1}^{p^2} S_l(\boldsymbol{y}_j - \boldsymbol{x}_*) R_l(\boldsymbol{x}_i - \boldsymbol{x}_*) = \\ &= \sum_{l=1}^{p^2} S_l(\boldsymbol{y}_j - \boldsymbol{x}_*) \sum_{\boldsymbol{x}_i \notin \Omega(\boldsymbol{y}_j)} q_i R_l(\boldsymbol{x}_i - \boldsymbol{x}_*) = \sum_{l=1}^{p^2} C_l S_l(\boldsymbol{y}_j - \boldsymbol{x}_*). \end{aligned} \tag{6}$$

Коэффициенты C_l строятся за pN операций и могут быть использованы для вычисления взаимодействия всех частиц $\boldsymbol{y}_j$, на что потребуется еще pN операций. Такой подход позволяет уменьшить вычислительную сложность расчета дальнего взаимодействия до $O(2pN)$. Так как для большой системы $p \ll N$, то в итоге вычислительная сложность будет порядка $O(N)$. Таким образом, вычислительная сложность расчета всего электростатического взаимодействия также составляет $O(N)$.

Для ядра Лапласа $\Phi(r)$ используются методы разложения и трансляций, описанные в [14, 25]. Для уменьшения вычислительной сложности каждой трансляции с $O(p^4)$ до $O(p^3)$ используется RCR-декомпозиция (поворот–коаксиальная трансляция–обратный поворот, Rotation–Coaxial translation–Rotation) матриц трансляционных операторов [25].

Опишем основные шаги метода FMM.

Генерация структуры данных. Изначально область приводится к единичному кубу и рекурсивно делится при помощи восьмеричного дерева до уровня $L_{\max}$. На уровне l будет 2^{ld} боксов, где d — размерность пространства (в рассматриваемой задаче $d = 3$). На рис. 5 показан пример разбиения до уровня $L_{\max} = 4$ для двумерной задачи. На каждом временно́м шаге строится структура данных: исключаются пустые боксы (не содержащие частиц), строится список взаимодействий и др. Алгоритм построения структуры данных описан в разделе 4.2.

Разреженная часть суммы $\phi_{\text{near}}(\boldsymbol{y}_j)$ вычисляется путем прохода по всем боксам на нижнем уровне $L_{\max}$ и прямого расчета для каждой частицы в боксе влияния остальных частиц в этом боксе и частиц из соседних боксов (P2P, рис. 5).

Плотная часть вычисляется на каждом временно́м шаге по следующей схеме.

а) *Предварительные вычисления.* Рассчитываются параметры, общие для нескольких боксов (например, элементы матриц трансляций).

б) *Восхождение (upward pass).* Генерируются коэффициенты разложения по мультипольному базису (C_l из формулы (6)) для каждого непустого бокса на уровне $L_{\max}$ (P2M, рис. 5). Это производится путем суммирования коэффициентов разложения по мультипольному базису всех частиц в боксе относительно

центра бокса. Далее производится трансляция этих коэффициентов вверх по уровням с $l = L_{\max}$ до $l = 2$ от дочерних к родительским боксам по восьмеричному дереву (M2M, рис. 5). Коэффициенты в родительском боксе получаются путем суммирования коэффициентов дочерних боксов, транслированных из центра дочернего бокса в центр родительского. На этом этапе все коэффициенты разложены по мультипольному базису.

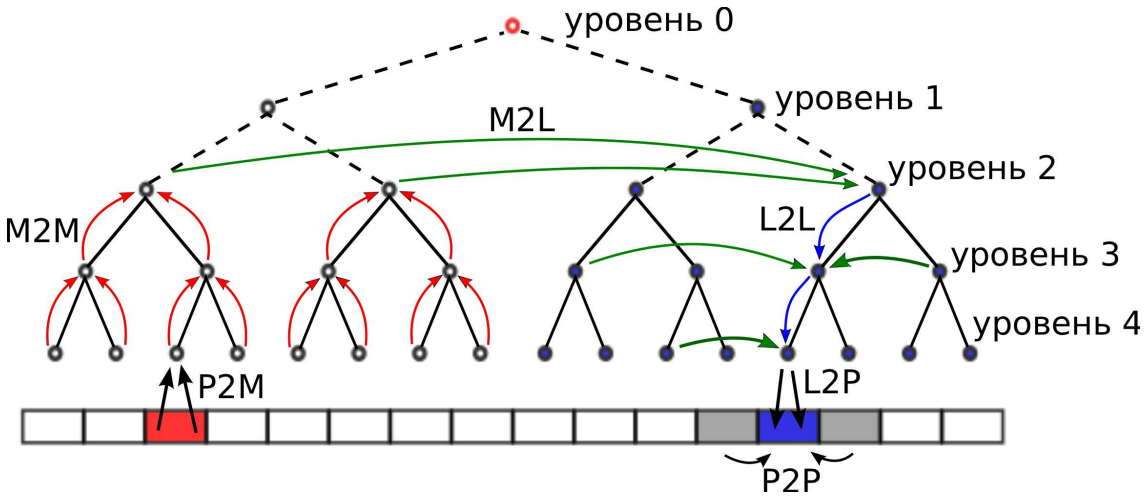


Рис. 5. Схема многоуровневого FMM, $L_{\max} = 4$, $d = 2$

в) *Нисхождение (downward pass)*. Рассчитываются локальные коэффициенты для каждого непустого бокса от уровня $l = 2$ до $l = L_{\max}$ при помощи двухшаговой процедуры на каждом из уровней. Первый шаг: производится трансляция коэффициентов из мультипольного в локальный базис (M2L, рис. 5) по специальному шаблону с последующим суммированием коэффициентов. Второй шаг: трансляция локальных коэффициентов вниз по восьмеричному дереву (L2L, рис. 5) от родительского бокса к дочерним с объединением с локальными коэффициентами, полученными на первом шаге.

г) *Окончательное суммирование*. Оцениваются локальные разложения (L2P, рис. 5) на уровне $L_{\max}$ для всех частиц всех непустых боксов, что дает $\phi_{\text{far}}(\boldsymbol{y}_j)$. Эти значения суммируются с результатом разреженного матрично-векторного произведения $\phi_{\text{near}}(\boldsymbol{y}_j)$.

Разреженная и плотная части могут быть вычислены независимо и параллельно. Для оптимальной производительности декомпозиция должна быть проведена так, чтобы сбалансировать времена вычисления соответствующих частей алгоритма. Декомпозиция определяется глубиной иерархического дерева.

4.2. Иерархическая структура данных. Метод FMM имеет сложную иерархическую структуру, которая сложна в реализации на SIMD (Single Instruction, Multiple Data) процессорах. Одним из стандартных путей создания структуры данных является алгоритм, использующий сортировку, вычислительная сложность которого равна $O(N \log N)$ [14]. Данный алгоритм обычно применяется на CPU, и его применение оправдано, если нет необходимости в перестроении структуры данных на каждом временно́м шаге (структура данных строится один раз или же один раз за несколько десятков шагов). В больших динамических системах, которые возникают в МД-моделировании, позиции частиц меняются на каждом временно́м шаге; соответственно, необходимо постоянно перестраивать структуру данных. С ростом размеров таких систем время генерации структуры данных становится значительным и начинает превалировать над временем выполнения FMM, так как сложность генерации структуры данных $O(N \log N)$ выше сложности $O(N)$ метода FMM. Кроме того, реализация на GPU алгоритма, применяемого для CPU, не позволяет достичь того уровня ускорения, который необходим. Таким образом, важной задачей является разработка эффективного алгоритма генерации структуры данных на GPU, вычислительная сложность которого не превышает вычислительной сложности FMM. Причем организация данных должна быть эффективной для расчетов, поиска и операций с этими данными как на GPU, так и на CPU. Такой алгоритм был предложен в работе [15]. Он основан на использовании гистограмм размещения (число частиц в каждом боксе), бинарной сортировке и параллельном сканировании [26].

Потенциальным недостатком такого подхода является ограничение по памяти, а именно тот факт, что для гистограммы необходимо выделение массива с числом элементов $2^{dL_{\max}}$ в памяти GPU. Для задач в трехмерном пространстве ($d = 3$) это может быть существенным ограничением. Тем не менее, данный алгоритм для GPU с размером глобальной памяти 3 GB позволяет размещать структуры данных

до максимального уровня $L_{\max} = 8$ (с учетом необходимости размещения других данных на GPU тоже), что является достаточным для большинства задач.

Алгоритм генерации структуры данных, основанный на [15], был модифицирован в применении к моделированию методами МД. Этот алгоритм состоит из следующих шагов.

1. Построение упорядоченного списка боксов. Существует ряд методов, преобразующих многомерный список боксов в одномерный. Для нашей задачи используется метод заполнения вдоль фрактальной кривой (space-filling curve), который позволяет заполнить многомерный куб одной непрерывной кривой. Наиболее подходящей для используемой структуры данных и параллельной реализации на CPU и на GPU является кривая Мортона (Z-order curve, Morton curve; рис. 6). Благодаря ее связи с восьмеричным деревом, эта кривая является очень эффективным инструментом для построения иерархических структур данных, основанных на таких деревьях, и для использования в вычислениях за счет простоты получения путем чередования бинарного представления положения частиц/боксов [13, 27].

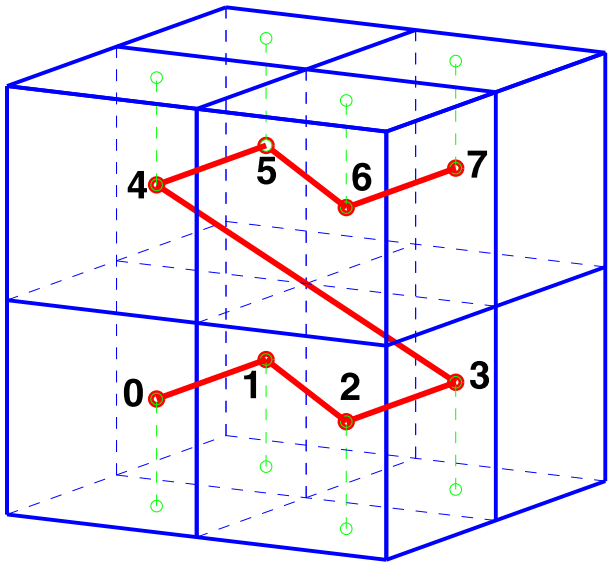


Рис. 6. Мортоновская кривая в трехмерном случае для группы боксов $2 \times 2 \times 2$

2. Построение гистограммы заполненности боксов частицами и получение локального индекса частиц внутри бокса.

3. Сканирование гистограммы (параллельный скан) и получение глобальных индексов частиц.

4. Пересортировка частиц согласно их глобальным индексам.

5. Определение непустых боксов исходя из гистограммы и получение списка непустых соседей для каждого бокса.

Каждый шаг описанного алгоритма имеет сложность $O(N)$ и достаточно легко распараллеливаем при помощи GPU.

Выбор глубины восьмеричного дерева $L_{\max}$ основан на балансе между временем вычисления потенциалов (их плотной и разреженной частей) и временем генерации структуры данных с учетом гетерогенности алгоритма, о которой будет говориться в следующем разделе. Кроме того, структура данных может быть эффективно применена для уменьшения времени расчета ближнего взаимодействия. Это применение подобно расчету разреженного матрично-векторного произведения для потенциала дальнего взаимодействия — рассчитывается взаимодействие только с частицами, находящимися в соседних боксах и в самом боксе, которому принадлежит рассматриваемая частица, и попадающими в сферу с радиусом обрезания для ближнего взаимодействия (центр сферы в рассматриваемой частице). Тем самым уменьшается вычислительная сложность за счет того, что отпадает необходимость в проверке взаимодействия всех частиц в области моделирования. Поэтому выбор уровня $L_{\max}$ зависит и от величины радиуса обрезания r_{cutoff} для потенциала ближнего взаимодействия.

Размер бокса на нижнем уровне ($L_{\max}$) выбирается бóльшим по сравнению с r_{cutoff}, для того чтобы сфера взаимодействия для рассматриваемой частицы вмещалась в область, ограниченную самим боксом, где частица находится, и ближайшими соседями этого бокса. Увеличение глубины $L_{\max}$, ввиду увеличения общего числа боксов, ведет к повышению затрат на генерацию структуры данных, затрат на трансляции и затрат на копирование информации о структуре данных и о коэффициентах разложения между GPU и CPU. Однако оно уменьшает время, необходимое для расчета разреженного матрично-векторного произведения, в том числе ближнего взаимодействия. Поэтому выбор $L_{\max}$ является нетривиальной задачей и зависит от параметров моделирования (например, плотности распределения частиц).

4.3. Метод FMM для гетерогенных архитектур. Все вычисления, параллельные по данным (частицам, боксам), в которых мало ветвлений и сложных обращений к глобальной памяти, в большинстве своем очень эффективно реализуемы на GPU. Однако операции трансляций коэффициентов из-за их сложной структуры, в том числе структуры обращения к памяти, реализуются на GPU недостаточно эффективно в сравнении с CPU. Это также связано с относительно малым размером кэша GPU. Использование многоядерных CPU для операций трансляции гораздо более эффективно. Кроме того, вычисле-

ния разреженной и плотной частей матрицы независимы друг от друга, поэтому возможно реализовать параллельность по задачам и использовать подход к вычислениям, описанный в [15, 28], что дает дополнительное ускорение. Основной его смысл в том, чтобы оптимально разделить вычисления на GPU и на CPU и производить их параллельно друг другу. За счет этого такой подход оптимальнее в сравнении с реализацией всего метода FMM на GPU [29].

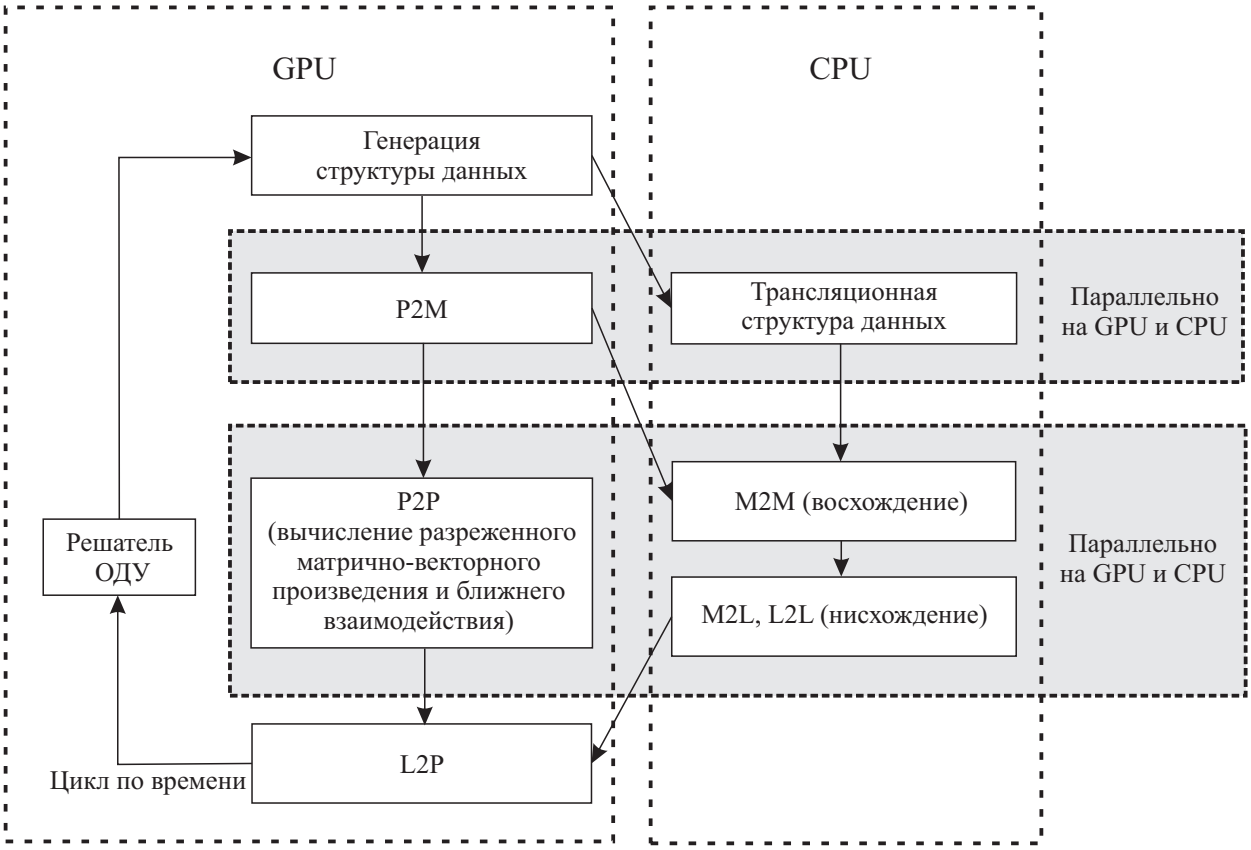


Рис. 7. Блок-схема FMM для гетерогенной вычислительной станции. Шаги алгоритма в светло-серых прямоугольниках исполняются параллельно на CPU и GPU

На рис. 7 представлена блок-схема рассматриваемой реализации метода FMM для гетерогенной вычислительной системы. Большие наборы данных по частицам (положения, скорости, ускорения) хранятся в глобальной памяти GPU, а все операции с данными, характеризующими частицы, производятся только на GPU. Таким образом, изменения в положениях частиц могут эффективно осуществляться на GPU, минимизируя обмен данными между CPU и GPU. Копирование данных по частицам на CPU для вывода осуществляется не часто — раз в несколько тысяч шагов.

На устройстве GPU выполняется работа, которая может выполняться наиболее эффективно: генерация структуры данных, мультипольное разложение (P2M) на уровне $L_{\max}$, вычисления разреженной части (P2P) и ближнего взаимодействия, локального разложения (L2P) и конечного результата. На CPU, в свою очередь, выполняется вся работа, связанная с иерархическим деревом и сложными трансляциями: восходящие по дереву трансляции (M2M) и нисходящие по дереву (M2L и L2L).

Ниже представлено пошаговое описание алгоритма.

1. На GPU генерируется иерархическая структура данных.

2. Генерация на GPU коэффициентов разложения по мультипольному базису (P2M) для каждого из непустых боксов и, параллельно этому, копирование необходимых для трансляций данных о структуре на CPU при помощи асинхронной функции копирования.

3. Копирование коэффициентов разложения по мультипольному базису с GPU на CPU.

4. На GPU производится вычисление разреженного матрично-векторного произведения для дальнего (P2P) и ближнего (LJ) взаимодействий; параллельно на CPU производятся трансляции коэффициентов разложения из мультипольного базиса в локальный (M2M, M2L и L2L) по непустым боксам.

5. Копирование коэффициентов разложения по локальному базису для непустых боксов с CPU на

GPU.

6. Вычисление на GPU конечного разложения (L2P) и суммирование результатов полного и разреженного матрично-векторного произведения.

Описанный алгоритм имеет ряд преимуществ:

1) CPU не простаивает, пока на GPU производятся вычисления, и загрузки CPU и GPU сбалансированы;

2) CPU и GPU загружены работой, которую они могут выполнять наиболее эффективно;

3) обмен данными между CPU и GPU минимизирован, так как необходимо лишь передавать p^2 коэффициентов разложения для каждого непустого бокса, что на несколько порядков меньше, чем информация о координатах и положении всех частиц;

4) код для CPU может быть распараллелен с использованием технологии OpenMP или других технологий.

5. Результаты. Вычисления производились на гетерогенной рабочей станции с двумя 6-ядерными CPU Intel Xeon 5660 2.8 GHz (всего 12 физических ядер и 12 виртуальных ядер с использованием технологии Hyper-Threading), 12 GB RAM и одной GPU NVIDIA Tesla C2075 с 6 GB RAM.

Для всех вычислений, если не оговорено иное, используются числа с плавающей точкой двойной точности, так как они дают бóльшую точность расчетов. Для случая чисел с плавающей точкой одинарной точности, как показали исследования, времена расчетов при использовании карты NVIDIA Tesla C2075 следует уменьшить примерно в два раза.

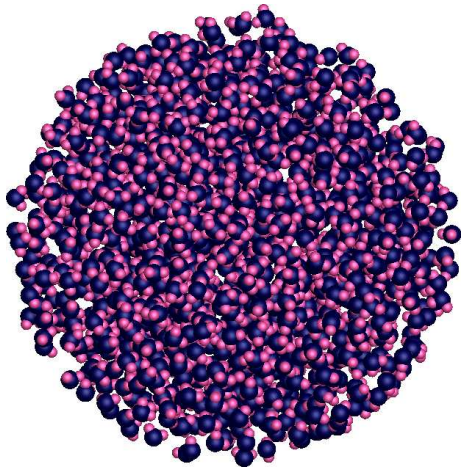


Рис. 8. Часть расчетной области, заполненная молекулами воды

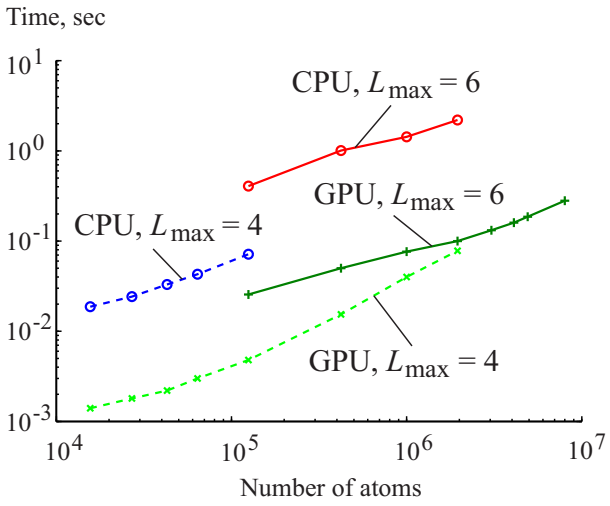


Рис. 9. Время генерации структуры данных в зависимости от числа частиц для двух выбранных максимальных глубин $L_{\max}$ восьмеричных деревьев

В тестах на производительность используется равномерное распределение частиц по области, что позволяет равномерно загрузить мультипроцессоры на GPU. Радиус обрезания r_{cutoff} для расчета ближнего взаимодействия берется равным 5σ. На рис. 8 для примера показана небольшая часть расчетной области, заполненная молекулами воды, образующими каплю.

Код для CPU был распараллелен при помощи OpenMP на 24 потока. Код для GPU был написан при помощи CUDA (Compute Unified Device Architecture) [30]. Оптимальное число потоков на блок в ядрах CUDA, как показало исследование, равняется 64 для рассматриваемой задачи начиная с некоторого числа частиц. Оптимальное число потоков на блок зависит от плотности рассматриваемых веществ и радиуса обрезания для ближнего взаимодействия. Тем не менее, для реализации алгоритмов могут быть использованы другие технологии программирования на GPU, например OpenCL.

5.1. Результаты по ускорению. Рис. 9 иллюстрирует время генерации иерархической структуры данных в зависимости от числа всех частиц (заряженных и не заряженных) для двух выбранных максимальных глубин ($L_{\max}$) восьмеричных деревьев. Прерывистые линии показывают время генерации на CPU (верхняя линия) и GPU (нижняя линия) для $L_{\max} = 4$. Сплошные линии показывают время генерации на CPU (верхняя линия) и GPU (нижняя линия) для $L_{\max} = 6$. Видно, что для каждого из $L_{\max}$ линии имеют одинаковые наклоны. Плавный рост, который виден в конце линии, описывающей время генерации на GPU для $L_{\max} = 4$, объясняется тем, что с ростом общего числа частиц растет и среднее

число частиц на бокс (так как $L_{\max}$ не меняется); на нижнем уровне структуры данных при достижении примерно 256 частиц на бокс число потоков на блок, равное 64, перестает быть оптимальным. В данном случае необходимо увеличивать максимальную глубину $L_{\max}$ восьмеричного дерева.

Как видно из табл. 1, использование GPU позволяет ускорить генерацию структуры данных примерно в 40 раз для чисел с плавающей точкой одинарной точности. Для чисел с плавающей точкой двойной точности ускорение составляет порядка 20 раз. Таким образом, время генерации структуры данных для 50 миллионов частиц и чисел одинарной точности на GPU составляет примерно 1 секунду.

Таблица 1

Время генерации структуры данных для числа атомов $N = 4 \times 10^6$ в зависимости от максимальной глубины $L_{\max}$ восьмеричного дерева

$L_{\max}$	CPU, сек	GPU, сек	Ускорение
6	3.3	0.08	41
7	9.1	0.24	38

Рис. 10 показывает время вычисления потенциала Леннарда–Джонса на CPU и на GPU в зависимости от числа частиц методом прямого суммирования и с использованием структуры данных. Асимптотика, показанная на рисунке, подтверждает, что использование структуры данных уменьшает вычислительную сложность алгоритма расчета ближнего взаимодействия с $O(N^2)$ до $O(N)$. Последнее является очень важным результатом для моделирования больших систем. Как видно из рисунка, расчет ближнего взаимодействия и примененные для его ускорения методы имеют хорошую масштабируемость.

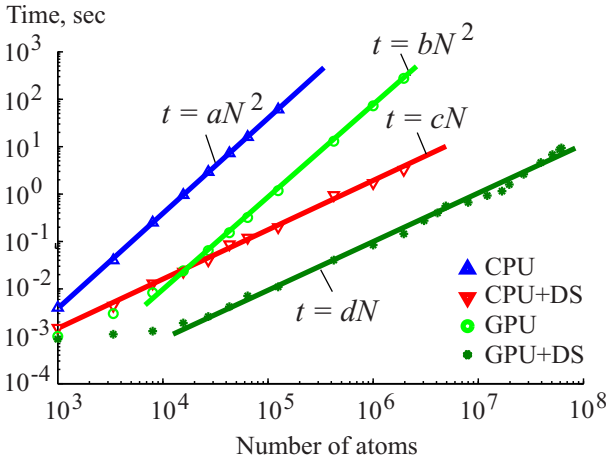


Рис. 10. Время вычисления ближнего взаимодействия методом прямого суммирования и с использованием структуры данных (DS) в зависимости от числа частиц для одного временно́го шага

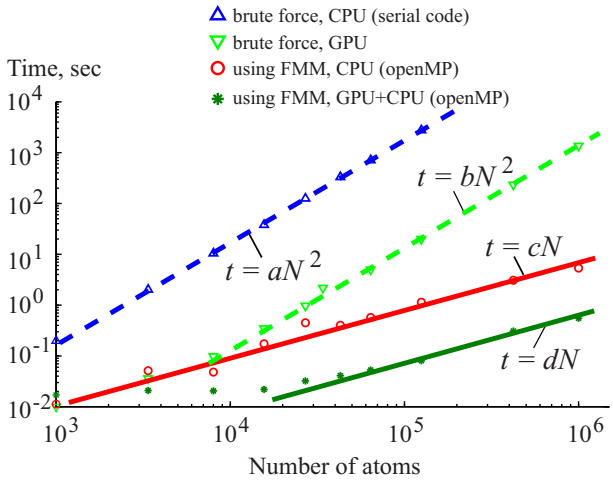


Рис. 11. Полное время вычислений одного временно́го шага в зависимости от числа частиц на CPU и на GPU с использованием метода прямого суммирования (brute force) и с использованием FMM, число усечения для FMM $p = 8$

Таблица 2

Время расчета ближнего взаимодействия методом прямого суммирования и с использованием структуры данных (вместе со временем ее генерации) на CPU и на GPU для одного временно́го шага. Аббревиатуры SP и DP означают одинарную и двойную точность соответственно

N	CPU, сек		GPU, сек		CPU+DS, сек		GPU+DS, сек	
	SP	DP	SP	DP	SP	DP	SP	DP
1.25×10^5	609	715	1.18	2.55	0.23	0.27	0.011	0.023
4×10^6	—	—	—	—	15.06	17.2	0.4	0.85

Применение GPU позволяет ускорить вычисление ближнего взаимодействия методом прямого суммирования до 520 раз при использовании чисел с плавающей точкой одинарной точности и до 280 раз при использовании двойной точности в сравнении с однопоточным кодом на CPU (табл. 2). Применение же GPU наряду с использованием иерархической структуры данных ускоряет расчет до 40 раз для чисел одинарной точности и до 20 раз для чисел двойной точности. Такое резкое падение в ускорении связано с ограничением в ускорении генерации структуры данных на GPU.

На рис. 11 показано общее время расчета одного временно́го шага на CPU и на GPU методом прямого суммирования и с использованием FMM. Оно включает в себя время генерации структуры данных, расчета ближнего и дальнего взаимодействия и время обмена данными в случае использования GPU. Как видно из асимптотики, время расчета методом прямого суммирования (прерывистые линии) растет пропорционально квадрату числа частиц $O\big(N^2\big)$ как для CPU (верхняя прерывистая линия), так и для GPU (нижняя прерывистая линия). Время расчета с использованием продвинутых методов (FMM, структура данных; показано сплошными линиями) растет линейно с ростом числа частиц ($O(N)$) как для CPU (верхняя сплошная линия), так и для GPU (нижняя сплошная линия).

Как видно из рис. 11, применение FMM для вычисления дальнего взаимодействия и применение иерархической структуры данных для расчета ближнего взаимодействия позволяют получить линейную масштабируемость алгоритма в зависимости от числа частиц.

В табл. 3 показано сравнение времени расчета одного временно́го шага с использованием FMM для случаев, когда код выполняется только на архитектуре CPU, когда код выполняется только на архитектуре GPU и когда гибридный код выполняется параллельно на CPU и на GPU. Поскольку не все функции реализованы на GPU, а именно не реализованы функции, отвечающие за трансляции по иерархической структуре данных, ввиду того, что их оптимальнее вычислять на CPU, то для оценок мы использовали ожидаемые времена для этих процедур на GPU, основываясь на времени их выполнения на CPU и времени выполнения подобных процедур на GPU, в том числе с учетом статей [14, 28]. Как видно из табл. 3, использование гетерогенного подхода CPU+GPU является наиболее оптимальным. Этого позволяет добиться то, что код исполняется параллельно на CPU и на GPU, т.е. осуществляется параллельность по задачам, что описывается подробно в разделе 4.3. Затраты на копирование данных по трансляциям между CPU и GPU скрываются за счет их асинхронности — они происходят параллельно выполнению кода.

Таблица 3

Сравнение времени расчета одного временно́го шага с использованием FMM на CPU и на GPU в зависимости от числа атомов N

N	CPU, сек	GPU, сек	CPU+GPU, сек
1.25×10^5	1.14	0.14	0.08
1×10^6	5.3	0.92	0.55

Таблица 4

Относительная ошибка FMM в зависимости от числа усечения p для числа частиц $N = 1 \times 10^6$

p	4	8	12	16	20
Отн. ошибка	5.5×10^{-4}	9.9×10^{-6}	5.6×10^{-7}	3.2×10^{-8}	5.4×10^{-9}

Для уменьшения ошибки FMM необходимо увеличение числа усечения p. Однако даже для небольшого $p = 8$ точность метода достаточна для вычислений (табл. 4). Увеличение p ведет к увеличению времени FMM, поэтому выбор этого числа является компромиссом между временем расчетов и точностью метода.

В табл. 5 показано сравнение эффективности моделирования с использованием предложенной методики с симулятором LAMMPS [31]. Этот симулятор был выбран исходя из того, что в нем реализован метод PPPM (Particle-Particle Particle-Mesh), позволяющий снизить вычислительную сложность всего алгоритма, и реализована возможность использования GPU. Сравнение было проведено на примере молекул воды (модель молекулы воды описана выше) при одинаковых физических параметрах и равномерном распределении молекул по области; рассчитывались ближнее и дальнее взаимодействия на основе потенциалов Леннарда–Джонса и Кулона соответственно. Радиус обрезания для LAMMPS был выбран наиболее оптимальным ($r_{\text{cutoff}} \approx 20$ Å). Для сравнения использовались расчеты, проведенные с числами двойной точности. Для расчетов на GPU в LAMMPS использовался пакет USER-CUDA. Сравнение проводилось на идентичных вычислительных станциях, описанных выше.

Следует отметить, что LAMMPS вычисляет периодическую систему, в то время как представленная методика на данный момент не может рассчитывать периодические системы в полной мере без существенных доработок. С другой стороны, LAMMPS не может рассчитывать непериодические системы с использованием PPPM и CUDA. Как видно из табл. 5, для систем с одинаковым количеством атомов в ячейке производительность предложенного метода в три с лишним раза выше, чем производительность LAMMPS. Разница в соотношении времен вызвана дискретностью выбора уровня иерархической структуры данных. Одним из существенных недостатков LAMMPS, выявленных во время тестирования, является ограничение на число моделируемых атомов из-за размеров памяти GPU. Так, на используемой GPU NVIDIA Tesla C2075 с 6 GB RAM симулятор LAMMPS не позволяет смоделировать с используемыми

физическими параметрами более 1 миллиона молекул, что является недостаточным для многих задач.

Таблица 5

Сравнение времени расчета одного временно́го шага при помощи описываемой методики с симулятором LAMMPS в зависимости от числа атомов N

N	Описываемый метод, сек	LAMMPS, сек	Соотношение
4.3×10^4	0.041	0.124	3.02
6.4×10^4	0.053	0.165	3.11
1.25×10^5	0.08	0.29	3.62
4.22×10^5	0.3	0.9	3.00
9.86×10^5	0.55	2.19	3.98

При помощи предлагаемого метода высокопроизводительная графическая карта GPU NVIDIA Tesla C2075 позволяет моделировать системы, содержащие вплоть до 25 миллионов атомов для чисел с двойной точностью и до 50 миллионов для чисел с одинарной точностью. Из-за ограничений на память на GPU моделирование бóльших систем требует использования нескольких GPU на одной вычислительной машине или же использования CPU/GPU кластеров. Развитие алгоритма для таких кластеров представляет предмет будущих исследований.

6. Заключение. Комбинация алгоритмического и аппаратного ускорения позволяет значительно ускорить моделирование методами МД. Алгоритмическое ускорение, которое включает в себя применение FMM для расчета дальнего взаимодействия и применение иерархической структуры данных для расчета ближнего и дальнего взаимодействий позволяют добиться понижения вычислительной сложности до $O(N)$ в противопоставлении к вычислительной сложности $O\left(N^2\right)$ метода прямого вычисления всех парных взаимодействий. Дополнительного ускорения до двух порядков можно добиться путем использования гетерогенных вычислительных систем, состоящих из CPU и GPU.

Объединение двух вышеописанных подходов позволяет рассчитывать один временной шаг для непериодической системы, состоящей из 1 миллиона атомов, менее чем за 1 секунду на персональном компьютере, оборудованном CPU и GPU уровня Tesla C2075. Кроме того, эти методы имеют хорошую масштабируемость, что показывает перспективность их применения на отдельных вычислительных узлах и кластерах с множеством GPU.

Работа выполнена при поддержке Министерства образования и науки Российской Федерации (грант № 11.G34.31.0040) и РФФИ (код проекта № 12–01–31083–мол_а). Часть программного обеспечения, использованного при разработке основного кода, описанного в настоящей статье, любезно предоставлена компанией Fantalgo, LLC (USA).

СПИСОК ЛИТЕРАТУРЫ

1. *Haile J.M.* Molecular dynamics simulation: elementary methods. New York: Wiley publication, 1992.
2. *Allen M.P.* Introduction to molecular dynamics simulation // Computational Soft Matter: From Synthetic Polymers to Proteins. Lecture Notes. Vol. 23. Jülich: Neumann Inst. for Computing, 2004. 1–28.
3. *Maruyama S., Kimura T.* A molecular dynamics simulation of a bubble nucleation on solid surface // Int. J. of Heat & Technology. 2000. **18**. 69–74.
4. *Maruyama S.* Molecular dynamics method for microscale heat transfer // Adv. Numer. Heat Transfer. 2000. **2**. 189–226.
5. *Anderson J.A., Lorenz C.D., Travesset A.* General purpose molecular dynamics simulations fully implemented on graphics processing units // J. of Computational Physics. 2008. **227**, N 10. 5342–5359.
6. *Sunarso A., Tsuji T., Chono S.* GPU-accelerated molecular dynamics simulation for study of liquid crystalline flows // J. of Computational Physics. 2010. **229**, N 15. 5486–5497.
7. *Nyland L., Harris M., Prins J.* Fast N-body simulation with CUDA // GPU Gems 3. Boston: Addison Wesley, 2007. 677–695.
8. *Darden T., York D., Pedersen L.* Particle mesh Ewald: an $N\log N$ method for Ewald sums in large systems // J. of Computational Physics. 1993. **98**, N 12. 10089–10092.
9. *Lindbo D., Tornberg A.-K.* Fast and spectrally accurate Ewald summation for 2-periodic electrostatic systems // J. Chem. Phys. 2012. **136**. 164111.
10. *Luty B.A., van Gunsteren W.F.* Calculating electrostatic interactions using the Particle-Particle Particle-Mesh method with nonperiodic long-range interactions // J. Phys. Chem. 1996. **100**, N 7. 2581–2587.
11. *Bossis G.* Molecular dynamics calculation of the dielectric constant without periodic boundary conditions // Molecular Physics: An International Journal at the Interface Between Chemistry and Physics. 1979. **38**. 2023–2035.

12. *Barnes J., Hut P.* A hierarchical *O(N*log*N)* force-calculation algorithm // Nature. 1986. **324**. 446–449.
13. *Greengard L., Rokhlin V.* A fast algorithm for particle simulations // J. of Computational Physics. 1987. **73**, N 2. 325–348.
14. *Gumerov N.A., Duraiswami R.* Fast multipole methods on graphics processors // J. of Computational Physics. 2008. **227**. 8290–8313.
15. *Hu Q., Gumerov N.A., Duraiswami R.* Scalable fast multipole methods on distributed heterogeneous architectures // Proc. of 2011 Int. Conf. for High Performance Computing, Networking, Storage and Analysis. New York: ACM Press, 2011. Article N 36.
16. *Rapaport D.C.* The art of molecular dynamics simulation. Cambridge: Cambridge Univ. Press, 2004.
17. *Frenkel D., Smit B.* Understanding molecular simulation. New York: Academic, 2002.
18. *Barker J., Watts R.* Structure of water: a Monte Carlo calculation // Chemical Physics Letters. 1969. N 3. 144–145.
19. *Rahman A., Stillinger F.H.* Molecular Dynamics Study of Liquid Water // J. of Chemical Physics. 1971. **55**. 3336–3360.
20. *Jorgensen W.L., Chandrasekhar J., Madura J.D., Impey R.W., Klein M.L.* Comparison of simple potential functions for simulating liquid water // J. of Chemical Physics. 1983. **79**. 926–935.
21. *Martin J.F.* A practical introduction to the simulation of molecular systems. Cambridge: Cambridge Univ. Press, 2007.
22. *Зацепина Г.* Свойства и структура воды. М.: Изд-во Моск. ун-та, 1974.
23. *Soper A., Phillips M.G.* A new determination of the structure of water at 25°C // J. of Chemical Physics. 1986. **107**. 47–60.
24. *Dongarra J., Sullivan F.* Guest editor's introduction: the top 10 algorithms // Computing in Science and Engineering. 2000. **2**, N 1. 22–23.
25. *Gumerov N.A., Duraiswami R., Borovikov E.A.* Data structures, optimal choice of parameters, and complexity results for generalized multilevel fast multipole methods in d dimensions // Technical Report CS–TR–4458. College Park: Univ. of Maryland, 2003.
26. *Harris M., Sengupta S., Owens J.D.* Parallel prefix sum (scan) with CUDA // GPU Gems 3. Boston: Addison Wesley, 2007. 851–876.
27. *Morton G.M., Phillips M.G.* A computer oriented geodetic database and a new technique in file sequencing // IBM Technical Report. Ottawa, 1966.
28. *Hu Q., Gumerov N.A., Duraiswami R.* Scalable distributed fast multipole methods // Proc. of 14th IEEE International Conference on High Performance and Communications (HPCC-2012). New York: IEEE Press, 2012. 270–279.
29. *Yokota R., Barba L.* Treecode and fast multipole method for N-body simulation with CUDA // GPU Computing Gems. Emerald Edition. Boston: Elsevier and Morgan Kaufmann, 2011. 113–132.
30. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Version 5.0. 2012.
31. http://lammps.sandia.gov

Поступила в редакцию
31.07.2013