Ускорение расчета процессов молекулярной динамики при помощи GPU *

Д.Ф. Марьин[1,2]

Центр микро- и наномасштабной динамики дисперсных систем, Башкирский государственный университет[1],
Институт механики УНЦ РАН[2]

В работе представлены результаты по производительности и эффективности использования GPU при моделировании процессов молекулярной динамики. Моделировался обрезанный потенциал Леннарда–Джонса при помощи вычислительной схемы leapfrog.

1. Введение

Задачи динамики дисперсных систем в микро- и наномасштабе возникают во многих отраслях науки и промышленности: механической, химической, нефтяной, экологической и др.. Проведение физических экспериментов над процессами, происходящими на микро- и наноуровнях сильно затруднено тем фактом, что размеры исследуемых элементов и структур зачастую оказываются на порядок меньше размеров длины волны видимого света. Это означает, что фиксация происходящих процессов либо достаточно сложна и требует сложного дорогостоящего оборудования, либо невозможна, так как коротковолновое рентгеновское и гамма излучение характеризуются высокой энергией квантов излучения, то есть их использование может в значительной степени искажать реальную картину наблюдаемых процессов.

Для решения вышеописанных сложностей используется вычислительный эксперимент, который позволяет описывать и измерять мельчайшие детали.

Однако и при проведении вычислительного эксперимента имеется ряд проблем, которые связаны с тем, что при достаточно подробном математическом описании проблемы, учитывающем многомерность и многопараметричность, а также с использование при моделировании большого числа частиц, серьезно возрастают требования к производительности как используемого программного кода, так и вычислительной системы в целом.

Для проведения вычислительного эксперимента в разумные сроки требуется удовлетворить ряд важных условий. Первое условие заключается в снижении вычислительной сложности используемых алгоритмов. Так сложность прямого алгоритма пропорциональна числу всех парных взаимодействий в системе размера N, то есть равна $O(N^2)$. И использование алгоритмов, которые имеют меньшую вычислительную сложность, является задачей чрезвычайно важной и актуальной.

Наряду с использованием алгоритмов с низкой вычислительной сложностью существует ещё одно направление повышения производительности вычислений — использование высокопроизводительных вычислительных систем. В настоящее время наиболее эффективными для задач динамики многих тел являются гетерогенные системы, представляющие собой вычислительные кластеры, узлы которых содержат как CPU (центральный процессор), так и GPU (графический процессор). Однако эффективное использование описанных вычислительных систем требует как значительной модификации существующих алгоритмов, так и разработки новых.

Следует отметить ещё один факт. Он заключается в том, что на протяжении масштабной шкалы от микро- до наноуровней располагается условная точка, после которой исполь-

*Работа выполнена при финансовой поддержке Министерства образования и науки Российской Федерации, грант 11.G34.31.0040.

зование классических континуальных моделей многофазных систем оказывается недопустимым, и актуальность приобретают кинетические модели, используемые в методах молекулярной динамики.

Таким образом, проведение исследований процессов динамики дисперсных систем, происходящих на микро- и наноуровнях, требует реализации молекулярно-динамических моделей при помощи численных методов, ориентированных на использование современных высокопроизводительных алгоритмов и терафлопсных гетерогенных вычислительных систем, содержащих как CPU, так и GPU.

2. Теоретическая часть

2.1. Математическая модель

Математическая модель среды, описываемой в терминах молекулярной динамики для случая неполярных молекул, основана на предположении о том, что среда состоит из сферических частиц, которые взаимодействуют друг с другом по определённому закону.

Функция (потенциальная функция, потенциал), описывающая такое взаимодействие, может принимать различные формы и зависит от поставленной задачи. Самой известной такой функцией является потенциал Леннарда—Джонса (эта модель достаточно реалистично передаёт свойства реального взаимодействия сферических неполярных молекул и поэтому широко используется в расчётах и при компьютерном моделировании) [1]

$$U_{LJ}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right],$$

где r — расстояние между частицами; ε — глубина потенциальной ямы; σ — расстояние, на котором энергия взаимодействия становится равной нулю. Параметры ε и σ являются характеристиками молекул соответствующего вещества.

Для ускорения расчётов потенциал Леннарда—Джонса обрывается на расстоянии $r_c = 2,5\sigma$. И, чтобы избежать нефизичной ситуации, такой, что при пересечении сферы радиуса r_c какой-то молекулой энергия системы меняется скачкообразно, потенциал сдвигается, чтобы выполнялось условие $U(r_c) = 0$. Таким образом, обрезанный потенциал Леннарда—Джонса принимает следующий вид

$$U_{LJ_{trunc}}(r) = \begin{cases} U_{LJ}(r) - U_{LJ}(r_c), & r \leqslant r_c\,, \\ 0, & r > r_c\,. \end{cases}$$

Кинетические уравнения движения атомов следуют из второго закона Ньютона:

$$m\ddot{\mathbf{r}}_\mathbf{i} = \mathbf{f}_\mathbf{i} = \sum_{\substack{j=1 \\ (j \neq i)}}^{N} \mathbf{f}_\mathbf{ij},$$

где $\mathbf{f_{ij}} = -\nabla U_{LJ_{trunc}}(r_{ij})$. Макроскопические параметры (температура, давление, плотность среды и др.) могут быть получены, исходя из положений молекулярно-кинетической теории.

2.2. Численный метод

Для интегрирования уравнений движения используется простая численная схема — метод чехарды (leapfrog) [1], который имеет вид

$$\mathbf{v}_\mathbf{i}(t + h/2) = \mathbf{v}_\mathbf{i}(t - h/2) + h\mathbf{a}_\mathbf{i}(t),$$

$$\mathbf{r}_\mathbf{i}(t + h) = \mathbf{r}_\mathbf{i}(t) + h\mathbf{v}_\mathbf{i}(t + h/2),$$

где $\mathbf{r_i}$ — координаты i–ой частицы; $\mathbf{v_i}$ — её скорость; $\mathbf{a_i}$ — её ускорение; t — текущий временной шаг; h — шаг по времени.

Если для оценки необходимо значение скорости на шаге по времени соответствующем шагу, на котором вычислены координаты, то можно использовать следующую формулу

$$\mathbf{v_i}(t) = \mathbf{v_i}(t - h/2) + (h/2)\mathbf{a_i}(t).$$

3. Реализационная часть

В работе представлены результаты для прямого расчёта, когда для каждой частицы рассматривается взаимодействие со всеми другими частицами с проверкой по радиусу обрезания, на CPU и на GPU. Версия на GPU реализована при помощи программно-аппаратной архитектуры CUDA.

Так же в работе представлены результаты расчета на CPU с использованием структуры данных и списка соседей. Использование структуры данных позволяет снизить вычислительную сложность всего алгоритма с $O(N^2)$ до $O(NM)$, где N — общее число частиц; M — среднее число частиц в соседних боксах. Построение структуры данных основано на использовании гистограммы распределения частиц по боксам и bucket–сортировки частиц [2]. Вычислительная сложность алгоритма построения структуры данных равна $O(N)$. Таким образом общая вычислительная сложность снижается при оптимальном подборе числа боксов в зависимости от N.

4. Экспериментальная часть

Тестовые расчёты проводились на вычислительной системе с CPU Intel Xeon 5660, 2.8GHz, GPU NVIDIA Tesla C2050, операционной системой Linux 64bit, компиляторами GCC v.4.4, CUDA v.4.0. Размер блока при проведении расчетов выбирался исходя из оптимальности и, начиная с некоторого числа частиц, равнялся 256 потокам на блок.

4.1. Результаты прямого расчёта

На рис. 1 показано время расчёта в зависимости от числа частиц в рассматриваемой системе для чисел с плавающей точкой одинарной точности. Как видно из рисунка графики выходят на постоянный тренд и имеют одинаковый наклон. Аналогичную картину можно наблюдать и для чисел с плавающей точкой двойной точности (см. рис. 2).

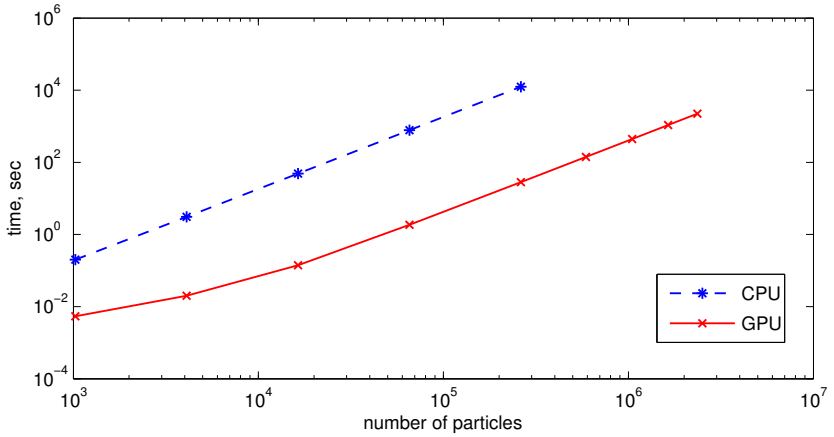

Рис. 1. Время выполнения в зависимости от числа частиц для чисел с плавающей запятой одинарной точности

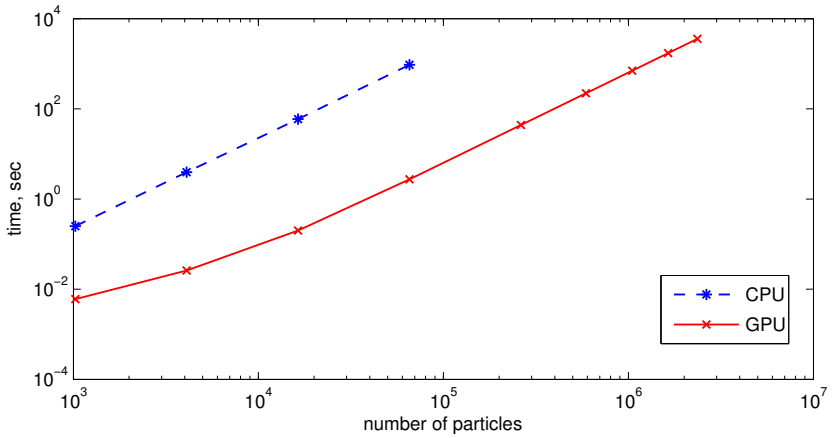

Рис. 2. Время выполнения в зависимости от числа частиц для чисел с плавающей запятой двойной точности точности

На рис. 3 показано ускорение полученное на GPU в сравнении с запуском на одном ядре CPU для чисел с плавающей точкой одинарной и двойной точности. Оно достигает 490 и 380 раз для чисел с одинарной и двойной точностью соответственно. Ускорение получено по времени вычислений без учета времени коммуникаций. Важно отметить, что сравнение ведется с оптимизированным, но не распараллеленным кодом на CPU (была включена автоматическая оптимизация компилятором как для CPU–кода, так и для GPU–кода). Целью является не соревнование между CPU и GPU, а использование кода на CPU в качестве основы для разработки кода на GPU.

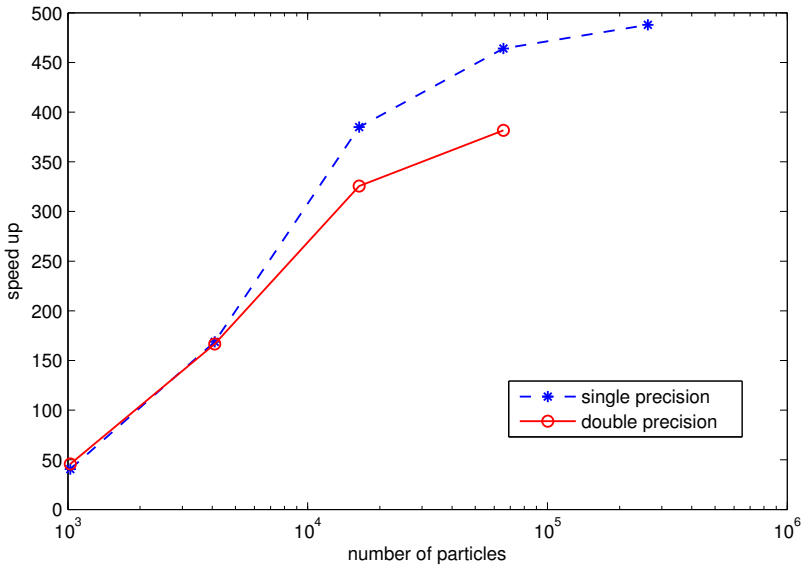

Рис. 3. Ускорение в зависимости от числа частиц

Об эффективности использования GPU можно судить по получаемой производительности, графики которой для чисел с плавающей точкой одинарной и двойной точности показаны на рис. 4. Производительность достигает 525 и 330 GFLOPS для чисел с одинарной и двойной точностью соответственно, что превышает половину пиковой производительности GPU C2050[1]. При расчёте производительности арифметические операции и стандартные

[1] пиковая производительность для чисел с плавающей точкой одинарной точности — 1030 GFLOPS, двой-

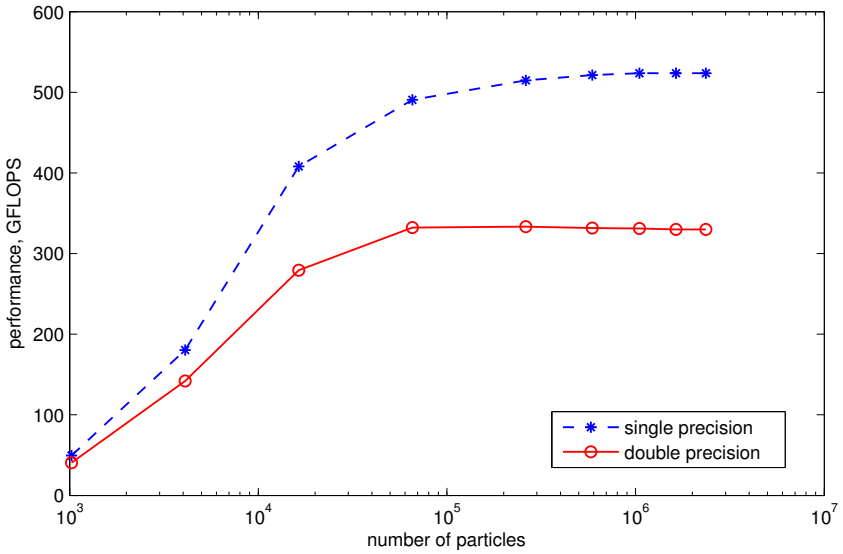

Рис. 4. Производительность в зависимости от числа частиц

функции (например, модуль числа) по занимаемому числу тактов приравнивались к операции сложения. Так же стоит отметить, что в связи со спецификой реализации на GPU, при расчёте сил взаимодействия на GPU не учитывался третий закон Ньютона, в отличии от расчёта на CPU, то есть GPU произвело примерно в два раза больше вычислений, чем CPU.

4.2. Результаты расчёта с использованием структуры данных на CPU

На рис. 5 показано время расчёта в зависимости от числа частиц в рассматриваемой системе для чисел с плавающей точкой одинарной точности.

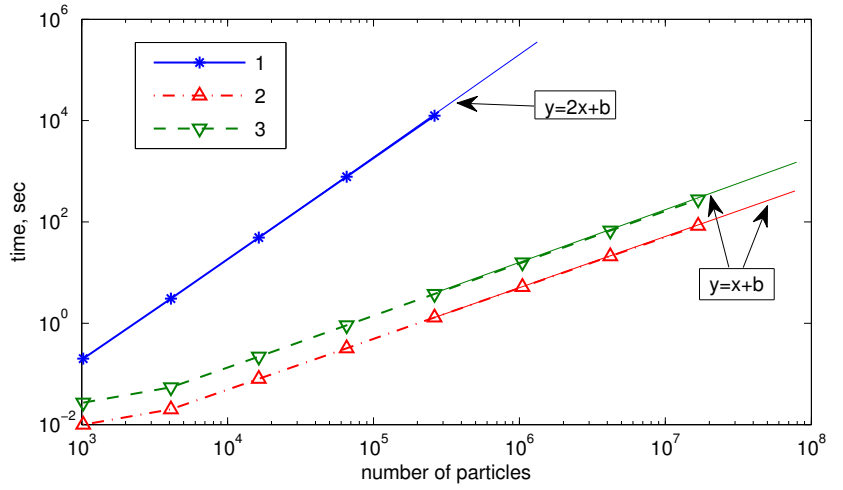

Рис. 5. Время выполнения на CPU, одинарная точность: (1) — время прямого расчёта, (2) — время расчёта с использованием структуры данных, (3) — (2) вместе со временем генерации структуры данных для каждой итерации

На графике так же показаны аппроксимирующие линии и их уравнения в логарифми-

ной точности — 515 GFLOPS

ческой системе координат. Как видно из рисунка, графики выходят на постоянный тренд, и угол наклона графиков (2, 3) в логарифмической системе координат, соответствующий вычислительной сложности $O(N)$, где N — число частиц, меньше угла наклона графика (1) из предыдущего пункта, соответствующего вычислительной сложности $O(N^2)$. Для чисел с плавающей точкой двойной точности картина аналогична.

Для динамичных систем, когда положение частиц меняется на каждой итерации, время генерации структуры данных превышает время одной итерации и, как видно из рис. 6, растёт с ростом числа частиц.

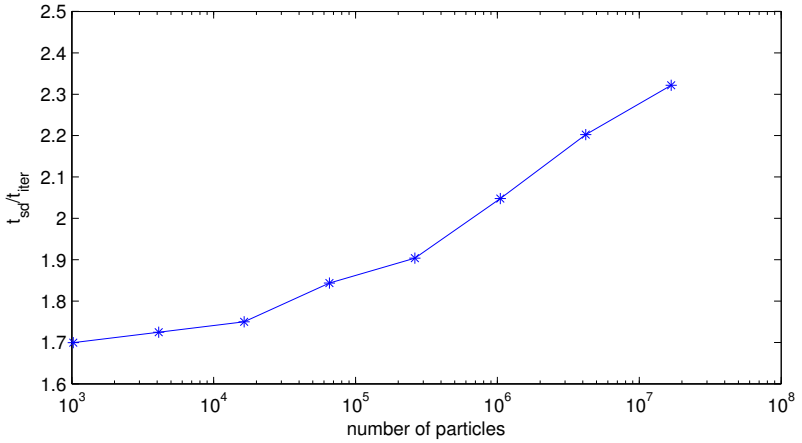

Рис. 6. Отношение времени генерации структуры данных ко времени одной итерации в зависимости от числа частиц

Но в зависимости от характерных параметров рассматриваемой среды, числа частиц и согласно условию Куранта можно подобрать число боксов так, чтобы проводить генерацию структуры данных один раз на 5–10 итераций.

5. Заключение

Таким образом можно сделать вывод, что достигнута очень хорошая производительность на GPU, благодаря чему удалось достичь ускорения проведения расчётов по сравнению с CPU в 490 и 380 раз для чисел с плавающей точкой одинарной и двойной точности соответственно.

Использование структуры данных позволяет достичь хорошего ускорения, путём снижения общей вычислительной сложности алгоритма. И реализация структуры данных на GPU является перспективным направлением для ускорения проведения расчётов рассматриваемой задачи.

Литература

1. Rapaport D.C. The art of molecular dynamics simulation. 2004. p. 400.

2. Q. Hu, N.A. Gumerov, and R. Duraiswami Scalable fast multipole methods on distributed heterogeneous architectures // in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC'11, (New York, NY, USA), ACM, 2011.

3. NVIDIA Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Version 3.2. 2010.