УДК 533;519.6

Использование графических процессоров при решении задач молекулярной динамики

Марьин Д. Ф.

Институт механики УНЦ РАН, Уфа

Центр «Микро- и наномасштабная динамика дисперсных систем»,

БашГУ, Уфа

Аннотация. В работе представлены результаты по производительности и эффективности использования GPU при моделировании процессов молекулярной динамики. Проводилось моделирование потенциала Леннарда–Джонса с помощью вычислительной схемы leapfrog.

1. Введение

Задачи динамики дисперсных систем в микро- и наномасштабах 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 дорогостоящего оборудования, либо невозможна, так как коротковолновое рентгеновское и гамма излучения 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 из-

мерять мельчайшие детали.

Однако и при проведении вычислительного эксперимента имеется ряд проблем, связанных с тем, что при достаточно подробном математическом описании проблемы, учитывающем многомерность и многопараметричность, а также при 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 необходимо использовать высокопроизводительные вычислительные системы. 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 их эффективное использование 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 использование 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 с использование гетерогенных вычислительных систем.

2. Математическая модель

Математическая модель среды, описываемой в терминах молекулярной динамики для случая неполярных молекул, основана на предположении, 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):

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

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

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

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

3. Численный метод

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

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

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

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

4. Результаты

Для тестирования программы использовалась равномерная генерация частиц по пространству. В работе не использовались структуры данных и списки соседей.

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.

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).

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

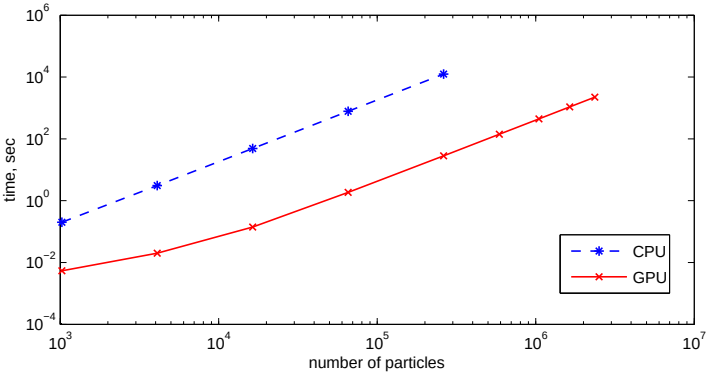

Рис. 1. Время выполнения, одинарная точность

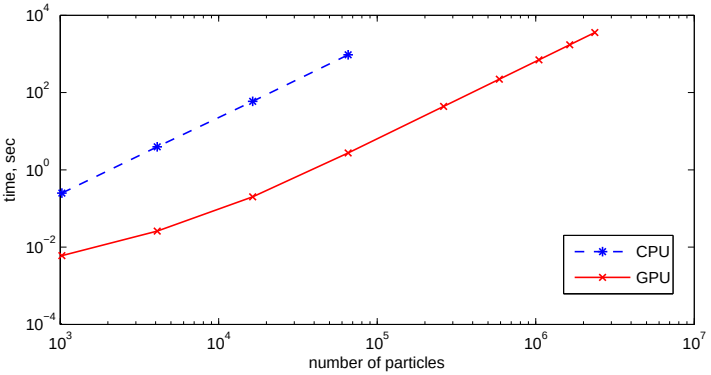

Рис. 2. Время выполнения, двойная точность

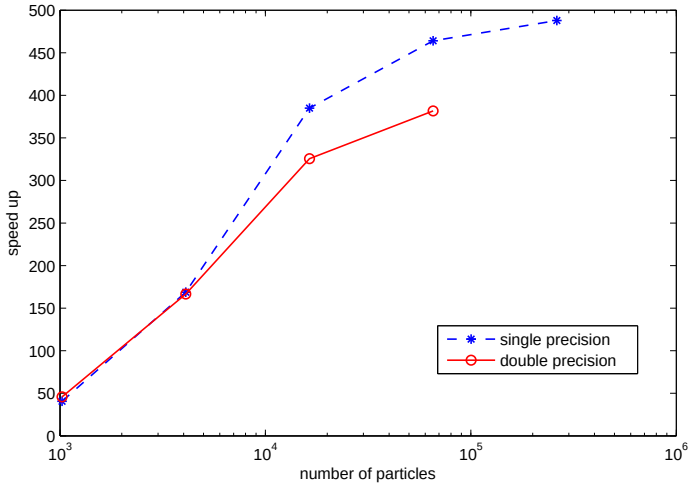

Рис. 3. Ускорение в зависимости от числа частиц

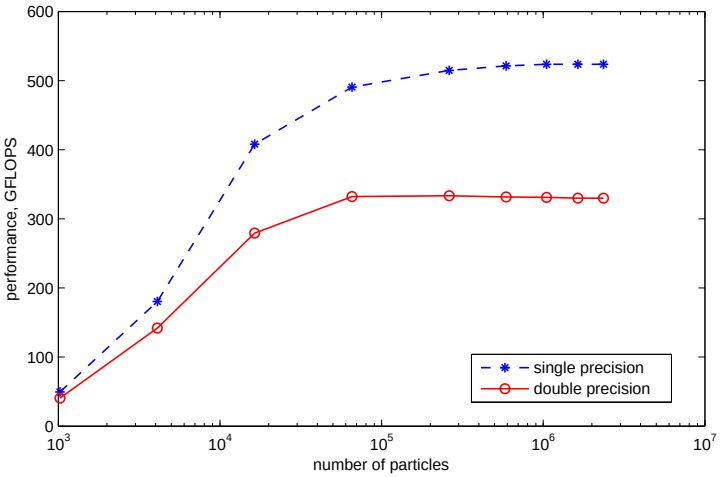

Рис. 4. Производительность в зависимости от числа частиц

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.

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 пиковая производительность для чисел с плавающей точкой одинарной точности — 1030 GFLOPS, двойной точности — 515 GFLOPS. При расчёте производительности арифметические операции и стандартные функции (например, модуль числа, корень) по занимаемому числу тактов приравнивались к операции сложения. Также стоит отметить, что в связи со спецификой реализации на GPU, при расчёте сил взаимодействия на GPU не учитывался третий закон Ньютона в отличие от расчёта на CPU, то есть GPU произвело примерно в два раза больше вычислений, чем CPU.

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.

Список литературы

[1] Rapaport D.C. The art of molecular dynamics simulation. 2004. P. 400.

[2] Gumerov N.A., Duraiswami R. Fast multipole methods on graphics processors // Journal of computational physics. 2008. Vol. 227. Pp. 8290–8313.

[3] Nyland L., Harris M., Prins J. Fast N-Body Simulation with CUDA NVIDIA Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Version 3.2. 2010.