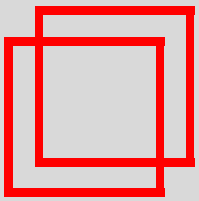

Math-Net.Ru
All Russian mathematical portal

D. S. Tarasov, E. D. Izotova, D. A. Alisheva, N. I. Akberova, GPAMM – software package for molecular dynamics on graphical processing units,
Mat. Model., 2009, Volume 21, Number 3, 31–40

https://www.mathnet.ru/eng/mm2745

Use of the all-Russian mathematical portal Math-Net.Ru implies that you have read and agreed to these terms of use
https://www.mathnet.ru/eng/agreement

Download details:
IP: 18.97.14.84
April 23, 2025, 18:21:44

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ 2009 год, том 21, номер 3, стр. 31-40

GPAMM – ПРОГРАММНЫЙ ПАКЕТ ДЛЯ РАСЧЕТОВ МОЛЕКУЛЯРНОЙ ДИНАМИКИ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

© *2009 г.* ***Д.С. Тарасов*, Е.Д. Изотова, Д.А. Алишева, Н.И. Акберова***

* Суперкомпьютерный центр коллективного пользования КНЦ РАН
Казанский государственный университет

Представлен программный пакет и рассмотрены соответствующие алгоритмы для расчетов молекулярной динамики на графических процессорах (GPU) NVIDIA G80,G84,G92 с использованием технологии NVIDIA CUDA. Реализованы потенциалы MM2 и AMBER с возможностью комбинации функций для создания других потенциалов. Поддерживается расчет молекулярной динамики в условиях NVT (канонический ансамбль) и NVE (микроканонический ансамбль). Сравнивается производительность кода для различных графических процессоров с производительностью МД-алгоритмов для процессоров Intel Celeron, Intel Pentium 4 и Intel Core2 Duo. Показано, что прирост производительности составляет до 30 раз. Программный пакет доступен по адресу www.gpamm.mntech.ru

GPAMM – SOFTWARE PACKAGE FOR MOLECULAR DYNAMICS ON GRAPHICAL PROCESSING UNITS

D.S. Tarasov, E.D. Izotova, D.A. Alisheva, N.I. Akberova*

* Supercomputer center of Kazan Scientific Center, Russian Academy of Sciences
Kazan State University

This work describes software package and algorithms for molecular dynamics using NVIDIA GPU G8x, and G92. All potentials needed for MM2 and AMBER force fields are implemented and combination of different potentials is allowed. Performance comparison of different MD algorithms on GPU and CPU is presented. All software is available from www.gpamm.mntech.ru

Введение

Моделирование молекулярной динамики является важным инструментом физической химии, молекулярной биологии и нанотехнологии [1]. Молекулярная динамика используется для предсказания результатов опытов, проверки адекватности теоретических построений и анализа поведения систем, изучение которых экспериментальными методами в настоящее время затруднительно или невозможно.

Классическая молекулярная динамика основана на расчете сил, действующих на все атомы, исходя из некоторой потенциальной функции и численного интегрирования уравнений движения. Однако для многих практических задач вычислительной мощности одного процессора недостаточно для получения результата в разумные сроки. Поэтому для задач молекулярной динамики используются параллельные вычисления. Алгоритмы и программные пакеты существуют для векторных суперкомпьютеров [1–3] и высокопроизводительных вычислительных кластеров [4–7]. Разработаны также и специализированные аппаратные архитектуры для ускорения вычислений молекулярной динамики [8], среди которых наиболее современной разработкой является MDGRAPE [9]. Такие модули позволяют на несколько порядков увеличить скорость расчетов, но одновременно имеют очень высокую стоимость и медленно совершенствуются.

С 2003 г. начинает развиваться новое направление в проблеме ускорения расчетов молекулярной динамики – использование графических процессоров (GPU). Появляется возможность

использовать графические акселераторы, применяемые в персональных компьютерах для воспроизведения трехмерных изображений как процессоры общего назначения, программируемые с помощью С-подобных языков, таких как Cg [9] или NVIDIA CUDA [10]. Графические процессоры обладают исключительно высокой производительностью по сравнению с центральными процессорами персональных компьютеров или серверов. Так видеокарта GeForce 8800 GTX имеет пиковую производительность в 300 Гигафлоп, для сравнения производительность процессора Pentium 4 составляет 4 Гигафлоп.

Рядом авторов разработаны алгоритмы для расчета молекулярной динамики на GPU. В [11] GPU используется для расчета тепловой проводимости в твердом аргоне, но используемый алгоритм поддерживает только парный потенциал Леннарда-Джонса и не имеет средств обновления списка соседей, что ограничивает его практическую применимость. Аналогичные алгоритмы, разработанные в [12, 13], переносят весь процесс расчета молекулярной динамики на GPU и используют обновляемый список соседей (список Верле) или метод связанных ячеек. В [14] разработан код молекулярной динамики, специфичный для расчета биологических макромолекул, который интегрирован в пакет NAMD. Сравнение производительности GPU и CPU версий в этих работах показывает выигрыш от 16 до 50 раз, при этом сравнение производительности различных реализаций часто затруднительно ввиду использования различных тестовых задач и вычислительных систем. Недостаточно изучены вопросы точности результатов, получаемых на GPU (графические процессоры NVIDIA поддерживают только операции с одинарной точностью и не полностью совместимы со стандартом IEEE 854). Не изучена производительность других видеокарт в задачах молекулярной динамики, возможность применения видеокарт нижнего ценового диапазона.

В настоящей работе представлен готовый к использованию программный пакет, содержащий основной набор функций для расчета молекулярной динамики на GPU. В пакете реализованы потенциалы MM2 [15] и AMBER [16] с возможностью комбинирования различных потенциальных функций для создания потенциалов, необходимых пользователю. Рассматриваются алгоритмы, использованные для расчетов на GPU, сравнивается производительность различных алгоритмов и видеокарт. Изучена точность получаемых на GPU результатов.

Реализация алгоритмов

Архитектура GPU. В соответствии с [11], GPU серии G8x и G9x представляют собой набор из нескольких SIMD мультипроцессоров. Каждый мультипроцессор содержит 8 SIMD-конвейеров. На каждом мультипроцессоре одновременно исполняется несколько потоков, содержащих одинаковые инструкции, но использующих различные данные. Потоки на одном мультипроцессоре организованы в SIMD-группы (warps), в каждый момент времени исполняется только одна SIMD-группа. Переключение между SIMD-группами осуществляется менеджером потоков. Потоки, выполняемые на одном мультипроцессоре (в одном блоке), могут синхронизироваться друг с другом, в то время как синхронизация между блоками невозможна.

Память видеокарты организована в несколько различных уровней.

1) Каждый мультипроцессор имеет 8192 32-х-разрядных регистра, в которых хранятся локальные переменные потоков.

2) Потоки одного блока могут обмениваться информацией посредством разделяемой памяти (shared memory) объемом 16 килобайт.

3) Основные данные хранятся в глобальной памяти (global memory). Доступ к глобальной памяти является узким местом, ограничивающим производительность, поскольку получение одного значения из глобальной памяти занимает 300-400 циклов процессора.

Работа с GPU осуществляется путем запуска ядра – функции (последовательности команд GPU), выполняемой одновременно над различными наборами данных. Необходимые для работы

данные должны быть предварительно скопированы в память видеокарты. После окончания работы ядра результаты копируются из памяти видеокарты в память компьютера.

Метод молекулярной динамики. Детальное описание методов молекулярной динамики имеется в многочисленной литературе, например, [1]. Здесь кратко рассмотрены только методы и детали их реализации в предлагаемом программном пакете.

Обычно при расчете молекулярной динамики перемещение атомов происходит в соответствии с классическими уравнениями движения. Силы, действующие на атомы, определяются из градиента потенциальной энергии как функции расстояния между взаимодействующими атомами. Наиболее распространенные потенциальные функции для молекулярной динамики имеют следующую форму:

$$V = V_b + V_a + V_d + V_{Vdw} + V_{elec}, \tag{1}$$

где V_b – силы, возникающие из-за растяжения ковалентных связей, V_a – силы, возникающие вследствие изменения валентных углов, V_d – двугранных углов, V_{vdw} – ван-дер-ваальсовы взаимодействия и V_{elec} – кулоновские взаимодействия. В пакете GPAMM реализованы наборы потенциальных функций, необходимых для проведения вычислений методами MM2 и AMBER. Реализованные потенциальные функции могут комбинироваться произвольным образом в соответствии с потребностями пользователя. При расчете градиента энергии наибольшие вычислительные трудности представляют нековалентные взаимодействия V_{vdw} и $V_{elec.}$

При расчете ван-дер-ваальсовых взаимодействий используется потенциал Леннарда-Джонса (1) и потенциал Букингема (3)

$$E_{Vdw} = \sum_{ij} \left[A_{ij} / R_{ij}^{12} - B_{ij} / R_{ij}^{6} \right], \tag{2}$$

$$E_{Vdw} = \sum_{ij} e_{ij} (2.9 \cdot 10^5 e^{-12.5 R_{ij} / r_{ij}} - 2.25 (R_{ij} / r_{ij})^{-6}), \qquad r_{ij} = r_i + r_j, \qquad e_{ij} = \sqrt{e_i e_j}, \tag{3}$$

где i и j – индексы атомов, R_{ij} – расстояние между атомами, r_i, r_j – стандартные радиусы атомов i и j. A_{ij}, B_{ij} – параметры, вычисляемые исходя из r_i, r_j, e_i, e_j. Электростатические взаимодействия вычисляются исходя из закона Кулона.

Организация программного комплекса GPAMM. Программный пакет GPAMM реализован на языке C++ с использованием компилятора Microsoft C++ Compiler 8.0; части, выполняющиеся на видеопроцессоре, написаны на языке NVIDIA CUDA, представляющем собой диалект C. Решение о создании нового пакета вместо интеграции GPU-функциональности в уже существующий пакет объясняется тем, что оптимальные структуры данных и последовательности операций для GPU существенно отличаются и требование интеграции с существующим ПО существенно скажется на производительности. Общая структура пакета GPAMM изображена на рис.1.

В целях повышения производительности вместо реализации одной общей GPU-функции в пакете GPAMM реализовано несколько специализированных GPU-функций, отличающихся по типу используемого GPU и решаемой задачи.

Нековалентные взаимодействия без обрезания потенциала. Поскольку ван-дер-ваальсовы силы быстро уменьшаются с расстоянием, при практических расчетах достаточно больших систем обычно игнорируются все взаимодействия частиц, находящихся на расстоянии, большем, чем заранее заданный радиус обрезания R_c. Поэтому в разработанном пакете производительности кода, выполняющего расчет без использования радиуса обрезания, не уделялось специального внимания. Эффективные алгоритмы реализации расчета молекулярной динамики на GPU для этого случая можно найти в [13], где использовался потенциал Леннарда-Джонса. Пакет GPAMM

реализует как потенциал Леннарда-Джонса, так и потенциал Букингема, который используется в силовых полях ММ2, ММ3 и ММ4. Отдельные функции реализуют вычисление электростатических взаимодействий.

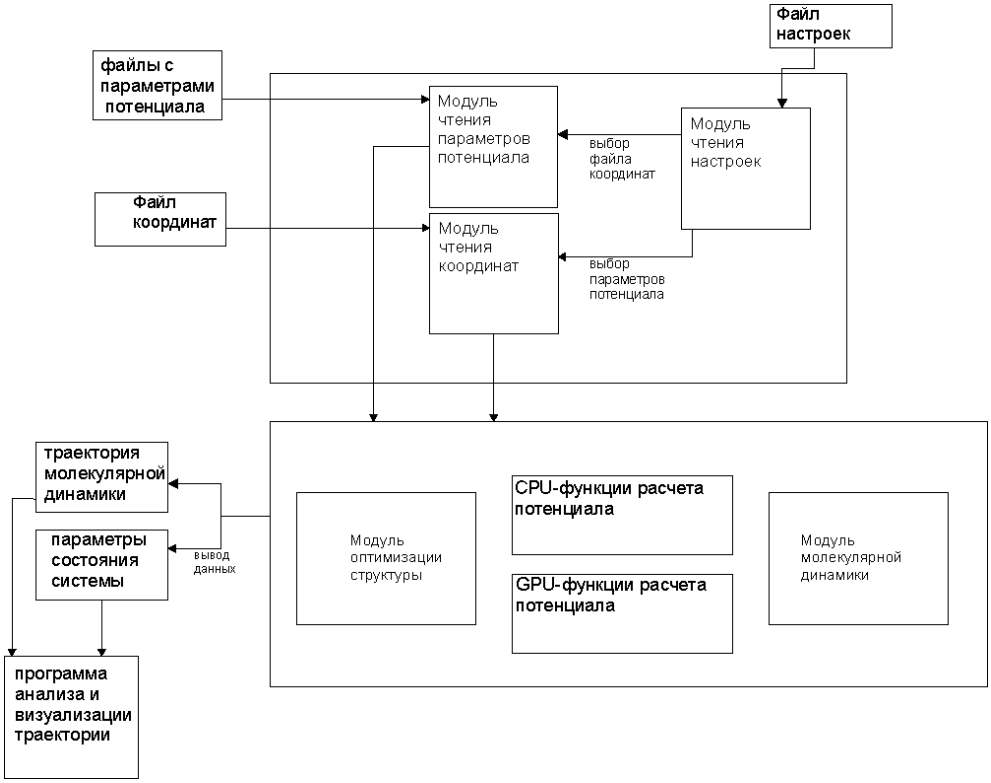


Рис.1. Структурная организация пакета GPAMM.

Нековалентные взаимодействия с обрезанием потенциала. Алгоритм реализован в виде двух ядер – функции составления списка соседей и функции расчета взаимодействий.

Составление списка соседей является методом, обычно используемым для ускорения расчетов парных нековалентных потенциалов в молекулярной динамике. Для каждого атома составляется список атомов, которые находятся к нему ближе, чем R_c+L. Данный список обновляется каждые n шагов молекулярной динамики, величина n зависит от характера исследуемой системы.

Алгоритм 1

tid – номер потока в блоке;

bid – номер блока;

M – число потоков в блоке;

indata – массив координат частиц, находится в глобальной памяти;

neigbors_data[i,j] – массив списков соседей;

i – номер списка соседей,

j – номер атома в списке соседей, находится в глобальной памяти, нулевой элемент содержит число найденных соседей i-го атома

p – счетчик числа соседей;

acount – общее число атомов;

```
int i = (bid*M)+tid;
p=1
if (i>count-1) return;
загрузка координаты i-й частицы из глобальной памяти
p_i = indata[i]
for (int j=0;j<i;j++)
{
    vij = p_i - indata[j]
      if (|vij|<Rc)
      {
            neigbors_data[i,p] =j;
            p=p+1;
       }
}
for (int j=i+1;j<count;j++)
{
    vij = p_i - indata[j]
    im_ij =  БлижайшийОбраз(vij)
    if (|vij|<Rc)
     {
          neigbors_data[i,p] =j;
          p=p+1;
      }
}
сохранить число найденных соседей
neigbors_data[i,0]=p-1;
```

Список соседей i-го атома не должен включать самого i-го атома. Чтобы избежать включения в функцию дополнительного условия, вся процедура разбита на два цикла – до i-го атома и после. Такое решение заметно повышает производительность.

Для GPU G92 возможно использование атомарных функций, что позволяет применять для формирования списка соседей более эффективный алгоритм:

***Алгоритм* 2**

Все обозначения переменных такие же, как в алгоритме 1.

```
int i = (bid*M)+tid;
p=1
if (i>count-1) return;
загрузка координаты i-й частицы из глобальной памяти
p_i = indata[i]
for (int j=0;j<i;j++)
{
      vij = p_i - indata[j]
        im_ij = БлижайшийОбраз(vij)
        if ((|vij|<Rc) или (|im_ij|< Rc))
        {
            p= atomicInc(адрес(neigbors_data[i,0]))
            neigbors_data[i,p] =j;
            p= atomicInc(адрес(neigbors_data[i,0]))
            neigbors_data[j,p] =i;
        }
}
… Второй цикл от i до acount, аналогично первому…
```

В алгоритме 2 используется встроенная функция CUDA atomicInc, выполняющая атомарное увеличение на единицу числа по заданному адресу и возвращающая предыдущее значение числа.

Расчет взаимодействий

Расчет ван-дер-ваальсовых и электростатических взаимодействий объединен в одно ядро с целью увеличения отношения числа арифметических операций к числу операций доступа к памяти.

Алгоритм 3

```
indata – массив структур с координатами и параметрами атомов;
outdata – массив структур с векторами градиентов на атомах и вкладом атома в общую энергию системы, находится в глобальной памяти;
acount – общее число атомов;
forces[] - массив структур с векторами градиентов на атомах и вкладом атома i в общую энергию системы, находится в разделяемой памяти;
tid
bid
atom1 – структура с данными об атоме i, находится в разделяемой памяти;
neigbors_data – списки соседей;
i=bid
if (tid ==0)
{
        Загрузить данные i-го атома в atom1
   }
__syncthreads();  //синхронизация потоков в блоке
    num = neigbors_data[i,0] (число соседей атома i);
    инициализировать значения forces[tid] в нуль
for (int th=0;th<8;th++)
{
        ind = tid*8+th;
        if (ind>num-1) break;
        j = neigbors_data[i,ind]
        int p = 1;
        проверить, не входят ли атомы в список 1-2 или 1-3 взаимодействий,
        если нет, продолжить расчет
        {
                Загрузить j-й атом в atom2
Вычислить значение энергии и вектора градиента, записать данные в forces[tid]
          }
}
Сложить все элементы forces[] до N, методом параллельной редукции (сначала все потоки складывают элементы попарно, затем половина всех потоков складывает пары и т.д.).
Записать полученные значения в outdata[bid]
```

Нековалентные взаимодействия с обрезанием потенциала в ячейке периодичности. Расчет молекулярной динамики для ограниченного количества атомов, помещенных в вакуум, приводит к тому, что поверхностные эффекты окажутся в системе доминирующими. Это является нежелательным, если исследуется поведение молекул в жидкости или твердые тела. Чтобы избежать этого используются периодические граничные условия [1], когда моделируемая система рассматривается как состоящая из фрагментов, повторяющихся во всех направлениях бесконечное количество раз. Форма повторяющегося фрагмента (ячейки периодичности) может быть кубическая, прямоугольная и др.

Если радиус R_c меньше половины длины прямоугольной ячейки периодичности во всех измерениях, то каждый атом будет взаимодействовать максимум с одним ближайшим образом другого атома (этот факт носит название соглашения «минимального образа» или «ближайшего образа»).

Для расчета с использованием ячейки периодичности пакет GPAMM содержит модифицированные соответствующим образом ядра алгоритма поиска соседей и расчета взаимодействий.

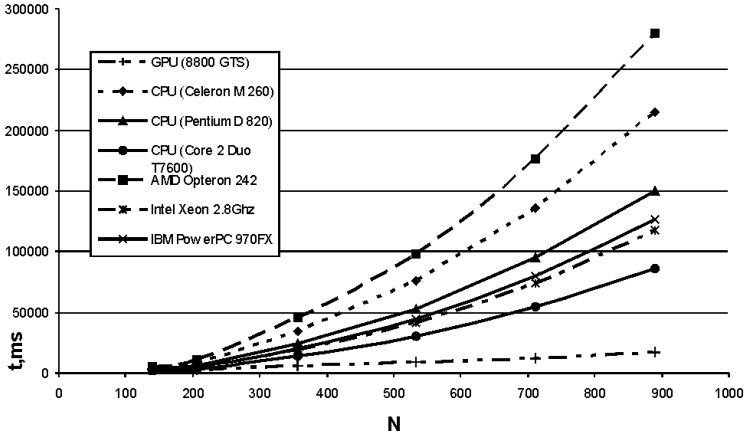
Производительность

Аппаратная конфигурация и использованные программы. Использовалась видеокарта GeForce 8800 GTS 320 MB, содержащая 12 мультипроцессоров, работающих на частоте 650 Mhz, с разрядностью шины памяти 320 бит. Видеокарта использовалась в системе, содержащей процессор Intel Pentium D 805 и 256 Mb ОЗУ под управлением операционной системы Windows XP. Для сравнения использовались системы на базе распространенных процессоров для персональных компьютеров и кластерных систем. Хотя часть из этих процессоров снята с производства, сравнение производительности для них представляет интерес, т.к. они все еще используются в составе ПК и, особенно, в составе кластеров, где замена процессоров происходит более медленными темпами.

1. Intel Celeron M 340 (1.5 Ghz, 400Mhz FSB, 512Kb Cache, WinXP)
2. Intel Pentium D 820 (2.8 Ghz, 800 Mhz FSB 1Mb Cache, WinXP).
3. Intel Core2 Duo T7500 (2.2 Ghz 800 Mhz FSB 4Mb Cache, Windows Vista).
4. AMD Opteron 242 (1.6Ghz, Linux)
5. Intel Xeon 2.8 Ghz, Linux
6. IBM PowerPC 970FX (2.2 Ghz, Linux).

Работа реализации молекулярной динамики для графического процессора сравнивалась с результатами и производительностью пакета TINKER [18], написанного на FORTRAN 77. Для Intel-совместимых процессоров пакет TINKER компилировался с помощью Intel Fortran Compiler 8.0, для IBM PowerPC использовался компилятор IBM XLF.

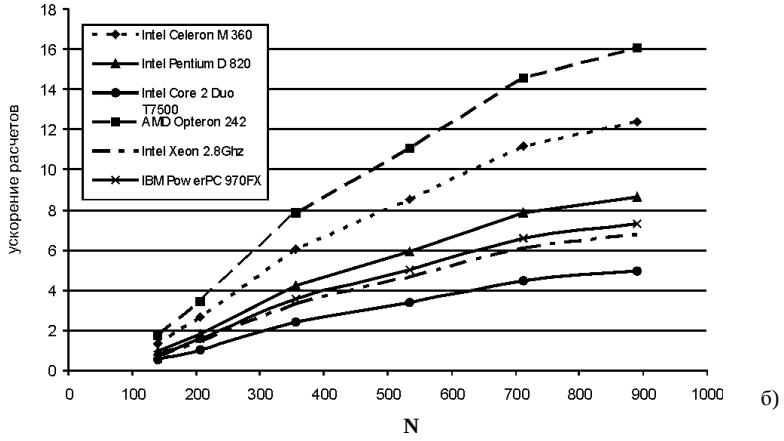
Ван-дер-ваальсово взаимодействие без обрезания потенциала. Результаты тестов приведены на рис.2а,б. Наблюдается 4-кратный прирост производительности по сравнению с Intel Xeon 2.8 и почти 16-кратный по сравнению с AMD Opteron 242.



а)

Рис.2а. Сравнение производительности при расчете VDW-взаимодействий без обрезания потенциала. График времени расчета 1000 шагов молекулярной динамики в зависимости от числа атомов для различных процессоров.

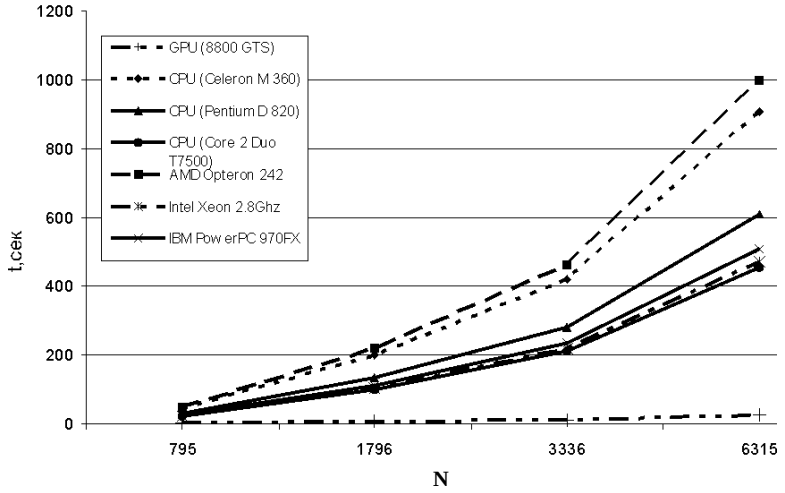
Ван-дер-ваальсово взаимодействие с обрезанием потенциала. Наблюдается 15-кратный прирост производительности по сравнению с Intel Xeon 2.8 и более чем 30-кратный по сравнению с Opteron 242. Данный результат сопоставим с результатами, полученными в [13], с учетом меньшей производительности карты 8800GTS.



б)

Рис.2б. Сравнение производительности при расчете VDW-взаимодействий без обрезания потенциала. График ускорения расчетов на GPU по сравнению с различными процессорами.

Ван-дер-ваальсово взаимодействие и электростатическое взаимодействие в ячейке периодичности. Результаты тестов приведены на рис.3а,б. Наблюдается 17-кратный прирост производительности по сравнению с Intel Xeon 2.8 и 34-кратный по сравнению с Opteron 242.



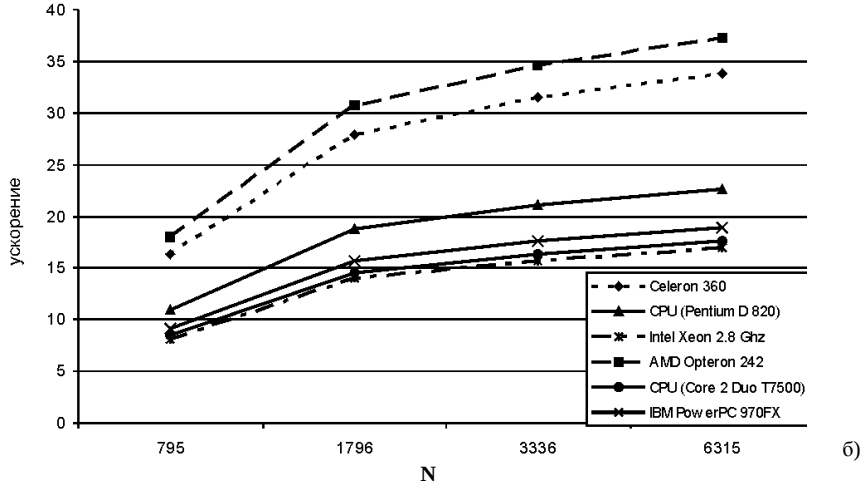
а)

Рис.3а. Сравнение производительности при расчете VDW-взаимодействий и электростатического взаимодействия в ячейке периодичности. График времени расчета 1000 шагов молекулярной динамики в зависимости от числа атомов для различных процессоров.

Тестирование точности. Был выполнен расчет молекулярной динамики для 1000 атомов аргона при использовании потенциала Букингема, применением потенциала ММ2. Мы сравнили изменение и среднее значение температуры с аналогичными расчетами, выполненными в прог-

рамме HyperChem 7.1 [19]. Начальная температура была установлена равной 100K. Результаты представлены на рис.4.

Среднее значение температуры на интервале 5-10 пс при расчете на GPU составило 59.8K, а при использовании программы HyperChem – 59.4K.



б)

Рис.3б. Сравнение производительности при расчете VDW-взаимодействий и электростатического взаимодействия в ячейке периодичности. График ускорения расчетов на GPU по сравнению с различными процессорами.

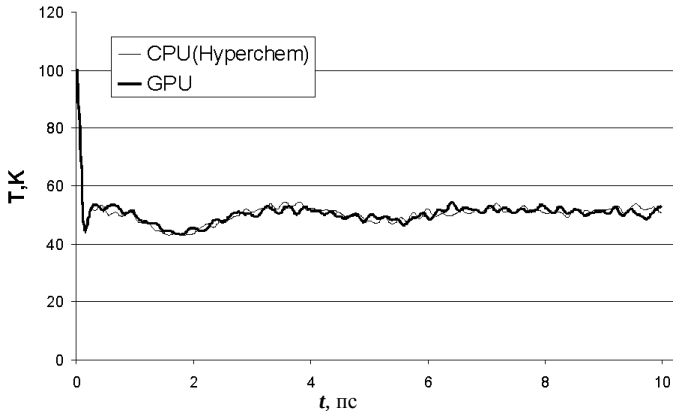


Рис.4. Изменение температуры в кластере из 1000 атомов аргона; результаты при расчете на GPU и CPU.

Заключение

Разработан новый программный пакет для расчета молекулярной динамики, особенностью которого является возможность проведения расчетов как на обычных, так и на графических процессорах. Использование графических процессоров позволяет получить существенный выигрыш в производительности.

В отличие от аналогичных программ [13,14], в GPAMM реализован расчет всех потенциалов, необходимых для использования популярных семейств силовых полей MM2 и AMBER (в

т.ч. отсутствующие в других решениях потенциалы для углов и двугранных углов, электростатические взаимодействия). Возможно комбинирование GPU и CPU-потенциалов, при этом CPU-потенциалы могут вычисляться с двойной точностью. Кроме того, впервые разработаны и реализованы алгоритмы, использующие атомарные операции, поддерживаемые новыми GPU семейства G92.

СПИСОК ЛИТЕРАТУРЫ

1. *D. Frankel, B. Smith.* Understanding molecular simulations. Second edition. NY, Academic press, 2002, 658 p.
2. *K. Esselink, B. Smit, P.A. Hilbers.* Efficient parallel implementation of molecular dynamics on a toroidal network: I. Parallelizing strategy // J. Comp. Phys., 1993, v.106, p.101-107.
3. *G.S. Grest, B. Dunweg, K. Kremer.* Vectorized link cell Fortran code for molecular dynamics simulations for large number of particles // Comp. Phys. Comm., 1989, v.55, p.269-285.
4. *S. Plimpton.* Fast parallel Algorithms for Short-Range Molecular Dynamics // J. of computational physics, 1995, v.117, p.1-19.
5. *H.J.C. Berendsen, D. van der Spoel, R. van Drunen.* GROMACS: A message-passing parallel molecular dynamics implementation // Comp. Phys. Comm., 1995, v.91, p.43-56.
6. *P. Alexander Lyubartsev, Aatto Laaksonen.* M.DynaMix – a scalable portable parallel MD simulation package for arbitrary molecular mixtures // Computer Physics Communications, 2000, v.128, p.565–589.
7. *J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kal_e, K. Schulten.* Scalable molecular dynamics with NAMD // J. Comp. Chem., 2005, v.26, p.1781-1802.
8. *A.F. Bakker, G.H. Glimer, M.H. Grabow, K. Thompson.* A special purpose computer for molecular dynamics calculations // J. Comp. Phys., 1990, v.90, p.313-335.
9. *Ryutaro Susukita, Toshikazu Ebisuzaki, Bruce G. Elmegreen, Hideaki Furusawa, Kenya Kato, Atsushi Kawai, Yoshinao Kobayashi, Takahiro Koishi, Geoffrey D. McNiven, Tetsu Narumi, Kenji Yasuoka.* Hardware accelerator for molecular dynamics: MDGRAPE- 2 // Computer Physics Communications, 2003, v.155, p.115–131.
10. *Randima Fernando, Mark J. Kilgard.* The Cg Tutorial. The Definitive Guide to Programmable Real-Time Graphics. Addison Wesley Professional, 2003.
11. NVIDIA. CUDA Programming Guide Version 1.0, 2006.
12. *Juekuan Yang, Yujuan Wang, Yunfei Chen.* GPU accelerated molecular dynamics simulation of thermal conductivities // J. of Computational Physics, 2007, v.221, p.799–804.
13. *J.A. van Meel, A. Arnold, D. Frenkel, S.F.P. Zwart, R.G. Belleman*. Harvesting graphics power for MD simulations, arXiv:0709.3225v1 [cond-mat.other].
14. *Joshua A. Anderson, Chris D. Lorenz, A. Travesset*. General Purpose Molecular Dynamics Simulations Fully Implemented on Graphics Processing Units // J. of Computational Physics, DOI:10.1016/j.jcp. 2008.01.047.
15. *J.E. Stone, J.C. Phillips, P.L. Freddolino, D.J. Hardy, L.G. Trabuco, K. Schulten.* Accelerating molecular modeling applications with graphics processors // J. Comp. Chem., 2007, v.28, p.2618-2640.
16. *U. Burker, N.L. Allinger*. Molecular Mechanics. ACS Monograph, 1982, v.177.
17. *D.W. Cornell, P. Cieplak, I.B. Christopher, I.R. Gould, M.M. Kenneth, D.M. Ferguson, D.C. Spellmeyer, T. Fox, J.W. Caldwell, P.A. Kollman.* A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules // J. Am. Chem. SOC., 1995, v.117, p.5179-5197.
18. TINKER - Software Tools for Molecular Design // http://dasher.wustl.edu/tinker/
19. HyperCube, Inc., HyperChem Computational Chemistry, Publication HC70-00-04-00, 2002.

Поступила в редакцию 16.04.08