Март, № 2 **Физико-математические науки** **2013**

УДК 004.021:004.94:519.683:519.684:544.223

ДМИТРИЙ СЕРГЕЕВИЧ КРУПЯНСКИЙ
аспирант кафедры физики твердого тела физико-технического факультета, Петрозаводский государственный университет (Петрозаводск, Российская Федерация)
krupjanski@rambler.ru

ДЕНИС ВЛАДИМИРОВИЧ ЛОБОВ
кандидат физико-математических наук, доцент кафедры физики твердого тела физико-технического факультета, Петрозаводский государственный университет (Петрозаводск, Российская Федерация)
ldenis@petrsu.karelia.ru

РОМАН НИКОЛАЕВИЧ ОСАУЛЕНКО
кандидат физико-математических наук, доцент кафедры физики твердого тела физико-технического факультета, Петрозаводский государственный университет (Петрозаводск, Российская Федерация)
oroman@psu.karelia.ru

РЕАЛИЗАЦИЯ МЕТОДА МОЛЕКУЛЯРНОЙ ДИНАМИКИ ПОСРЕДСТВОМ ТЕХНОЛОГИИ NVIDIA CUDA*

В последнее время значительно возрос интерес к моделированию систем с большим количеством частиц. Также растет и время проведения расчета, вследствие чего время моделирования на центральном процессоре становится недопустимо большим. Нами проведен поиск возможностей снижения времени проведения расчета методом молекулярной динамики (МД). Представлены результаты МД-моделирования с использованием графического процессора (ГП) NVIDIA GF 114. Расчет межмолекулярных взаимодействий реализован с помощью технологии NVIDIA CUDA, позволяющей значительно ускорить вычисления. Для определения прироста производительности был произведен ряд расчетов различных систем, содержащих от 1000 до 18 000 атомов. Обнаружено существенное влияние способа параллельной организации расчета на время его проведения. Максимальный прирост производительности дает оптимизация доступа к латентной глобальной памяти ГП. Показано, что применение недорогих ГП для проведения МД-эксперимента позволяет ускорить вычисления с двойной точностью в 60 раз. Разработанная схема параллельного расчета универсальна и может быть использована для решения схожих задач.

Ключевые слова: графические процессоры, технология NVIDIA CUDA, параллельные вычисления, молекулярная динамика

ВВЕДЕНИЕ

Современный ГП является мощным вычислительным устройством для расчетов общего назначения. Технология NVIDIA CUDA [7] предоставляет простую и удобную модель реализации таких расчетов, в ней ГП рассматривается как специализированное устройство, которое является сопроцессором к центральному процессору (ЦП), обладает собственной памятью и возможностью параллельного выполнения огромного количества потоков (нитей). Доступность этой технологии способствует широкому применению ГП в различных расчетах, в том числе и методом МД. В некоторых работах [1] мера производительности определялась в Gflops. В других работах [2], [3] использовалась более объективная оценка – отношение времени расчета на одном процессоре ко времени расчета на нескольких вычислительных устройствах. Цель настоящей работы – поиск возможностей снижения времени проведения МД-расчета.

РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Распараллеливание алгоритма является фактически единственным способом увеличить скорость расчета, не меняя самой модели. Поэтому основная задача данной работы – разработка реализации МД для ГП на основе существующей программы. Для этого было решено использовать наиболее доступную и удобную технологию – NVIDIA CUDA. Нами рассматривалась динамика системы частиц, в которой каждая частица взаимодействует со всеми остальными согласно определенным законам, а результирующие силы – это суперпозиция $N(N-1)$ независимых парных взаимодействий. Наибольший вклад в прирост производительности вносит распараллеливание блока исходной программы, отвечающего за расчет парных взаимодействий, который представляет собой треугольный цикл:

```
for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
                \\ учет периодических граничных условий,
                   определение параметров
                \\ потенциала взаимодействия, расчет
                   потенциала и сил
                \\ взаимодействия i-й и j-й частиц
        }
}
```

где n – общее количество частиц.

Время выполнения этого блока пропорционально квадрату n и при больших значениях n становится недопустимо большим. Таким образом, параллельное исполнение этой части алгоритма может дать наиболее весомый вклад в прирост производительности. Тестирование проводилось на следующей системе:

ЦП:	AMD Phenom II x4 B65 @3.6ГГц
Видеокарта:	Nvidia GeForce GTX 560 Ti (GF114)
Компилятор:	++ 4.4.3, CUDA Toolkit 4.2 g

Наша реализация метода МД позволяет проводить моделирование систем произвольного количества частиц разных типов с двойной точностью вычислений. Параллельный расчет с использованием CUDA предполагает определенную организацию нитей, детально описанную в [7]. Нити группируются в линейные, плоские или трехмерные блоки, которые, в свою очередь, группируются в линейную или плоскую сетку. Для данной задачи оптимальной оказалась линейная структура блоков и сетки, в которой каждая нить соответствует расчету парных взаимодействий одной частицы со всеми остальными. Каждый блок включает в себя BS нитей, взаимодействующих между собой через механизм разделяемой памяти. Оптимальный размер блока нитей BS (64) определен с помощью инструмента «CUDA GPU Occupancy Calculator» из NVIDIA CUDA SDK. Конфигурация и запуск ядра производятся следующим образом:

```
threads = dim3(64, 1);
blocks = dim3(ceil(n/64),1);          // n – количество частиц
compute_forces<<<blocks, threads>>>(...)
                                      // расчет взаимодействий
```

Если n не кратно размеру блока, то количество нитей в сетке превосходит необходимое значение. Неиспользуемые нити отбрасываются.

АЛГОРИТМ ДЛЯ GPU

1. Вычисление индекса нити: $i = blockIdx.x * blockDim.x + threadIdx.x$, где *blockIdx.x* – номер блока; *blockDim.x* – размер блока; *threadIdx.x* – индекс нити в блоке. Этот индекс необходим для доступа к соответствующим элементам массивов данных.

2. Описание локальных переменных и массивов данных в разделяемой памяти. В ядре (функции для GPU) для расчета парных взаимодействий используются 5 массивов в разделяемой памяти. Размер массивов соответствует количеству нитей в блоке. Один массив используется для хранения потенциала взаимодействия, другой – для типов частиц, а три остальных – для их координат. Использование разделяемой памяти критически влияет на производительность программы.

```
__shared__ double upot[BS];      // Потенциал
__shared__ int type[BS];         // Тип частицы
__shared__ double pos_x[BS];     // Координаты: x
__shared__ double pos_y[BS];     // y
__shared__ double pos_z[BS];     // z
```

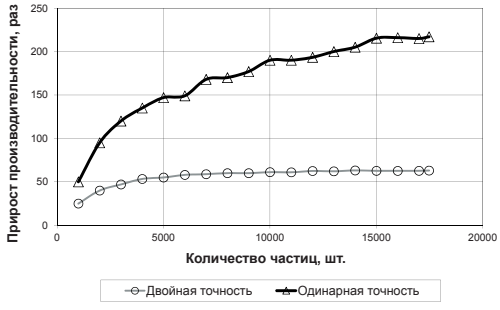
где BS – размер блока нитей.

3. Расчет взаимодействия i-й частицы со всеми остальными частицами системы. Цикл по всем частицам разбивается на два вложенных цикла:

```
for(int k=0;k<n;k=k+BS){
    pos_x[threadIdx.x]=g_posx[k+threadIdx.x];
    pos_y[threadIdx.x]=g_posy[k+threadIdx.x];
    pos_z[threadIdx.x]=g_posz[k+threadIdx.x];
    type[threadIdx.x]=g_type[k+threadIdx.x];
    __syncthreads();
    for(int j=k;j<k+BS;j++)
        if(i!=j){
        \\расчет взаимодействия i-й и j-й частиц
        \\(вклад j-й частицы в потенциальную
          энергию
        \\и ускорение i-й частицы)
        }
}
```

В цикле по *k* происходит параллельное чтение данных всеми нитями блока из медленной глобальной памяти в быструю разделяемую. В цикле по *j* – учет периодических граничных условий, расчет взаимодействия *i*-й и *j*-й частиц, причем координаты и тип *j*-й частицы берутся из переменных *pos_x[j-k]*, *pos_y[j-k]*, *pos_z[j-k]* и *type[j-k]*, объявленных в разделяемой памяти. Таким образом можно уменьшить необходимое количество чтений из медленной памяти в *BS* раз, а объем необходимой разделяемой памяти регулировать путем изменения размера блока нитей.

4. Запись результатов в глобальную память.



Прирост производительности, одинарная и двойная точность

ЗАКЛЮЧЕНИЕ

Для определения прироста производительности был произведен ряд тестовых запусков программы с различными конфигурациями. Усредненные результаты приведены на рисунке. Прирост производительности определяется как отношение времени моделирования 100 временных шагов на ЦП ко времени выполнения той же задачи на ГП. Анализируя график, можно сделать следующие выводы:

- Показана высокая эффективность применения видеоадаптеров в качестве устройства для расчетов общего назначения.
- Найдена достаточно эффективная организация параллельного расчета для задач, содержащих вложенные циклы, несмотря на высокую связанность обрабатываемых данных.

- Данная организация с минимальными изменениями может быть эффективно использована для задач, содержащих вложенные циклы.
- На всех современных видеоадаптерах есть возможность расчетов с двойной точностью, но с потерей производительности.

* Работа выполнена при поддержке Программы стратегического развития ПетрГУ в рамках реализации комплекса мероприятий по развитию научно-исследовательской деятельности на 2012–2016 гг.

СПИСОК ЛИТЕРАТУРЫ

1. Боярченков А. С., Поташников С. И. Использование графических процессоров и технологии CUDA для задач молекулярной динамики // Вычислительные методы и программирование. 2009. Т. 10. С. 9–23.
2. Янилкин А. В., Жиляев П. А., Куксин А. Ю., Норман Г. Э., Писарев В. В., Стегайлов В. В. Применение суперкомпьютеров для молекулярно-динамического моделирования процессов в конденсированных средах // Вычислительные методы и программирование. 2010. Т. 11. С. 111–116.
3. Anderson J. A., Lorenz C. D., Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units // Journal of Computational Physics 227. 2008. P. 5342–5359.
4. Belleman R. G., Bedorf J., Zwart S. P. NewAstron.12: High performance direct gravitational N-body simulations on graphics processing units-II: An implementation in CUDA. 2007. P. 641–650.
5. Davis J. E., Ozsoy A., Patel S., Taufer M. Towards Large-Scale Molecular Dynamics Simulations on Graphics Processors // Proceedings of the 1st international conference on bioinformatics and computational biology. Heidelberg, 2009. P. 176–186.
6. Liu W., Schmidt B., Voss G., Muller-Wittig W. Molecular Dynamics Simulations on Commodity GPUs with CUDA // Berlin, Lecture Notes in Computer Science 4873, 2007. P. 185–196.
7. NVIDIA Corporation. NVIDIA CUDA C Programming Guide. Santa Clara, CA, 2012.
8. Walters J. P., Balu V., Chaudhary V., Kofke D., Shultz A. Accelerating Molecular Dynamics Simulations with GPUs // ISCA PDCS. 2008. P. 44–49.

Krupyanskiy D. S., Petrozavodsk State University (Petrozavodsk, Russian Federation)
Lobov D. V., Petrozavodsk State University (Petrozavodsk, Russian Federation)
Osaulenko R. N., Petrozavodsk State University (Petrozavodsk, Russian Federation)

IMPLEMENTATION OF MOLECULAR DYNAMICS METHOD USING NVIDIA CUDA TECHNOLOGY

Simulation of systems represented by a large number of atoms poses an exceptional interest. However, the simulation time grows proportionally to the square of the atoms' number. Accordingly, the time needed for the calculation of a large system on a Central Processing Unit(CPU) becomes unacceptably high. We studied the possibilities that can provide acceleration of molecular dynamics simulations. Since the use of the parallel program model is the best way to increase computing performance we used the most convenient and accessible technology – NVIDIA CUDA. This technology allows speeding up calculation with the use of NVIDIA Graphics Processing Unit (GPU). We present the results of molecular dynamics simulations using NVIDIA GF114 GPU. The study tested GPU and CPU implementations of the molecular dynamics method. We ran series simulations with a different number of atoms (up to 18000 atoms) to determine the performance gains. The study showed that there is a significant fluency of parallel organization of calculations on its execution time. Moreover, performance reaches its maximum in case of the well-optimized data access to high latency global memory of a GPU. Due to the use of a relatively cheap graphics' card for a molecular dynamics simulation the computing performance can be improved up to 60 times for double precision and more than 200 times for single precision. The designed scheme of parallel computation is quite efficient and can be easily applied for similar problems.

Key words: GPU, NVIDIA CUDA, parallel computing, molecular dynamics

REFERENCES

1. Boyarchenkov A. S., Potashnikov S. I. The use of GPUs and CUDA technology for molecular dynamics applications [Ispol'zovanie graficheskikh protsessorov i tekhnologii CUDA dlya zadach molekulyarnoy dinamiki]. *Vychislitel'nye metody i programmirovanie*. 2009. Vol. 10. P. 9–23.
2. Yanilkin A. V., Zhilyaev P. A., Kuksin A. Yu., Norman G. E., Pisarev V. V., Stegailov V. V. The use of supercomputers for molecular dynamics simulation of condensed matter processes [Primenenie superkomp'yuterov dlya molekulayrno-dinamicheskogo modelirovaniya protsessov v kondensirovannykh sredakh]. *Vychislitelnye metody i programmirovanie*. 2010. Vol. 11. P. 111–116.
3. Anderson J. A., Lorenz C. D., Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units // Journal of Computational Physics 227. 2008. P. 5342–5359.
4. Belleman R. G., Bedorf J., Zwart S. P. NewAstron.12: High performance direct gravitational N-body simulations on graphics processing units-II: An implementation in CUDA. 2007. P. 641–650.
5. Davis J. E., Ozsoy A., Patel S., Taufer M. Towards Large-Scale Molecular Dynamics Simulations on Graphics Processors // Proceedings of the 1st international conference on bioinformatics and computational biology. Heidelberg, 2009. P. 176–186.
6. Liu W., Schmidt B., Voss G., Muller-Wittig W. Molecular Dynamics Simulations on Commodity GPUs with CUDA // Berlin, Lecture Notes in Computer Science 4873, 2007. P. 185–196.
7. NVIDIA Corporation. NVIDIA CUDA C Programming Guide. Santa Clara, CA, 2012.
8. Walters J. P., Balu V., Chaudhary V., Kofke D., Shultz A. Accelerating Molecular Dynamics Simulations with GPUs // ISCA PDCS. 2008. P. 44–49.

Поступила в редакцию 29.12.2012