МОЛЕКУЛЯРНО-ДИНАМИЧЕСКОЕ МОДЕЛИРОВАНИЕ КОНДЕНСИРОВАННЫХ ВЕЩЕСТВ НА ГИБРИДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ, ВКЛЮЧАЮЩИХ ВИДЕОАДАПТЕРЫ

Р.Г. Быстрый, А.М. Казённов, И.В. Морозов, В.В. Писарев

Введение.

Метод молекулярной динамики (МД) широко применяется для моделирования статических и динамических свойств твердых тел и жидкостей. Особое значение этот метод имеет для описания метастабильных состояний и фазовых переходов [1,2], возникающих при ударно-волновом нагружении, лазерной абляции и других интенсивных воздействиях. Суть метода МД заключается в представлении исследуемой системы в виде набора частиц, взаимодействующих друг с другом по определенному закону (потенциалу). Поведение системы во времени определяется простым решением уравнения движения каждой частицы. Таким образом, можно исследовать макроскопическое и микроскопическое поведении системы во времени.

Для некоторых задач (МД) требуется проведение моделирование на системах с большим числом частиц, или же с большим временным промежутком. Примером такой задачи является молекулярно-динамическое моделирование флуктуаций в конденсированной фазе при приближении к границам устойчивости. Задача является очень тяжелой с вычислительной точки зрения, поэтому эффективно решать её можно только методами параллельной обработки данных на высокопроизводительных вычислительных системах.

Тенденция развития вычислительных систем (ВЧ) показывает то, что производительность ВЧ повышается последнее время не за счет увеличения производительности отдельного процессора, а за счет увеличения числа вычислительных ядер в процессоре или вычислительных блоков в ядре. Наибольшая плотность вычислительных блоков, на данный момент, достигнута на видеокартах (GPU) (максимум 480 вычислительных конвейеров на 1 плату), что делает гибридные системы, включающие GPU, очень удобным инструментом для параллельных вычислений. В 2006 компания NVIDIA выпустила программный пакет CUDA, предназначенный для разработки ПО общего назначения под GPU, что значительно упростило разработку новых программ, не связанных с обработкой видеоизображений.

В данной работе обсуждается производительность программ МД моделирования на GPU по сравнению с CPU для модельной системы с парным потенциалом взаимодействия Леннарда-Джонса, а также для реальной задачи, в которой используется эмпирический многочастичный потенциал «погруженного атома». Рассматриваются особенности реализации многочастичных потенциалов на GPU.

Обзор пакетов МД моделирования для гибридных систем

В настоящее время все большее число коммерческих и свободно распространяемых пакетов МД моделирования поддерживают вычисления на GPU. Однако, пока реализованы лишь простейшие потенциалы взаимодействия, такие как потенциал Леннарда-Джонса, потенциал гармонической связи между атомами в макромолекуле и т.п. В то же время для задач физики твердого тела [1,2] требуются более сложные потенциалы, основанные, как правило, на экспериментальных данных. Таким потенциалом является потенциал погруженного атома (Embedded Atom Method - EAM) [3].

Таким образом, целью данной работы было разработать процедуру расчета взаимодействий атомов с использованием потенциала EAM, и провести тестовые расчеты. В качестве базового пакета МД моделирования был выбран пакет Highly Optimized Object-oriented Many-particle Dynamics (HOOMD) [4]. В отличие от других широко известных пакетов LAMMPS и NAMD, которые также имеют поддержку расчетов на видеоадаптерах, HOOMD с самого начала создавался под гибридную архитектуру.

На конец 2008 года пакет HOOMD имел следующие возможности:

Поддерживаемые операционные системы: Windows, Linux, and Mac OS X

Парные потенциалы: Lennard-Jones и простые потенциалы для связей атомов в макромолекулах;

Интеграторы уравнений движения: Brownian dynamics NVT, NPT, NVE, NVT;

Форматы файлов ввода/вывода: HOOMD's XML input format, MOL2, DCD, PDB;

Особенности реализации потенциала погруженного атома на GPU

Потенциал погруженного атома (EAM) обладает рядом особенностей по сравнению с потенциалом Леннарда-Джонса, поэтому его адаптация для GPU потребовала существенной переработки функции расчета взаимодействий частиц, имевшейся в HOOMD. Полная потенциальная энергия системы складывается из «энергии погружения» и дополнительного парного потенциала:

$$U = U^{em} + U^{pair} = \sum_i F_i\left(\rho_i^{\Sigma}\right) + \sum_i \sum_{i<j} \Phi_{ij}\left(r_{ij}\right),\ \rho_i^{\Sigma} = \sum_{j \neq i} \rho_j(r_{ij})$$

Здесь ρ – эффективная электронная плотность, которая создается вокруг каждого ядра *j*. При расчете энергии погружения для каждого атома *i* определяется суммарная электронная плотность ρ^{Σ}, которая наводится всеми соседними атомами *j* в точке $r = r_i$, после чего вычисляется энергия $F(\rho^{\Sigma})$. Таким образом, для вычисления потенциала требуется задание трех функций: электронной плотности, энергии погружения и парного потенциала. Вид этих функций находится, как правило, из сравнения результатов расчета с экспериментом, а сами функции задаются в виде интерполяционных таблиц.

При расчете сил, действующих на частицы, необходимо знать значения производных указанных функций, которые целесообразно посчитать один раз, и сохранить также в виде интерполяционных таблиц. Сила, соответствующая парному взаимодействию имеет достаточно простой вид, а сила, связанная с энергией погружения, выражается по формуле:

$$\overline{f}_k^{em} = -\frac{\partial U^{em}}{\partial \overline{r}_k} = -\sum_{j \neq k} \frac{\vec{r}_{kj}}{r_{kj}} \left[F_k{}'(\rho_k^{\Sigma}) \rho_j'(r_{kj}) + F_j{}'(\rho_j^{\Sigma}) \rho_k'(r_{kj}) \right]$$

Основной особенностью здесь является то, что расчет приходится проводить в два этапа. На первом этапе необходимо произвести суммирование электронных плотностей и сохранить полученный массив значений ρ^{Σ} (или $F'(\rho^{\Sigma})$) для каждого атома. На втором этапе производится суммирование по *j*, в результате чего определяется сила, действующая на частицу *k*.

При выполнении программы на GPU выполняется декомпозиция по частицам, т.е. каждый поток вычисляет энергию и силу для одной частицы. Таким образом распараллеливается неявный цикл по *k*. При этом между двумя этапами расчета сил необходимо осуществлять промежуточную синхронизацию всех потоков. Фактически это означает разделение процедуры на два вычислительных ядра (kernels), последовательно запускаемых на GPU. Для текущей реализации CUDA это единственный способ синхронизации всех выполняющихся потоков.

Кроме того, возникает необходимость хранения в глобальной памяти промежуточного массива с результатами первого этапа расчета. Логичным решением здесь оказалось, использовать специальный контейнер хранения данных cudaArray и связывать с ним текстуру, так как текстура, заданная таким образом, позволяет не только использовать кэширование на видеокарте, но и производить аппаратную линейную интерполяцию, что уменьшает количество запросов к памяти.

Изменения быстродействия реализация для GPU и CPU

Полученная программа применялась для расчета тестовых и реальных задач с различными веществами (медь, алюминий), находящихся в разных агрегатных состояниях (кристалл, переохлажденный расплав). Варьировалось также число частиц в МД ячейке ($N = 10^3$-10^6).

Расчеты проводились на гибридной машине «X8 NVidia»:

- MB: TYAN S7025;
- 2 x CPU - Intel Xeon 5520 (Four-core 2.3GHz) (Nehalem), 8Mb L3
- RAM - 32Гб DDR3Dimm 10660 ECC Registered Kingston;
- **8 x GPU - GeForce 480GTX.**

Наличие нескольких видеокарт в машине позволяло проверять «чистую» производительность видеокарты на задаче (ничего кроме самой задачи на видеокарте не выполнялось).

Результаты отражены на рисунках 1-3.

Как видно из рис. 1, видеокарта показывает прирост производительности в **10-58** раз по сравнению с одним ядром и в **3-9** раз по сравнению с 8 ядрами двух центральных процессоров. Кроме того, если сравнивать скорость расчета системы с потенциалом EAM со скоростью расчета системы с потенциалом Ленарда-Джонса то обнаруживается, что при большом числе частиц скорость расчета на видеокарте для EAM оказывается больше чем для Леннарда-Джонса, в то время как на центральном процессоре EAM в среднем в 2 раза медленнее. Также видно, что, в принципе, не достигнут ещё предел по эффективности распараллеливания, то есть с ростом числа частиц отношение производительностей продолжает расти. Однако, существует иное ограничение на количество частиц в системе. Для ускорения работы программы используется список ближайших соседей, который занимает около 1 Гб оперативной памяти видеокарты для 1 млн. частиц в системе. На данный момент это ограничение может быть преодолимо только заменой видеокарт специализированными вычислительными модулями Tesla, имеющими больший объем внутренней памяти.

Кроме ускорения на одном вычислительном узле, нас интересовало, сколько ядер традиционного вычислительного кластера «заменяет» видеокарта. То есть, сколько ядер вычислительного кластера необходимо

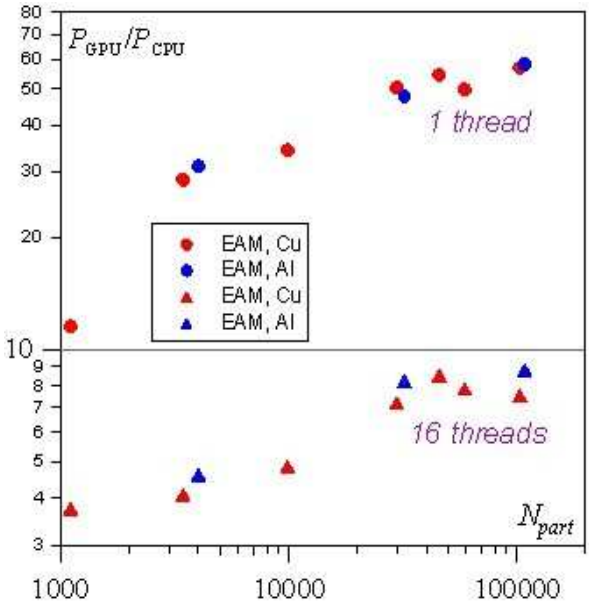

Рис 1. Зависимость отношения производительности видеокарты к центральному процессору от числа частиц. Для верхнего ряда точек использовалось одно ядро CPU, для нижнего – 8 ядер (16 потоков с применением Hyper Threading).

для того, что бы система считалась с такой же скоростью как на одной видеокарте.

Для сравнения был выбран вычислительный кластер МФТИ-60:

Число вычислительных узлов / процессоров:	132/264 (528 ядер)
Тип процессоров:	Dual-Core Intel Xeon 5160 3ГГц
Оперативная память полная/на узел, Гб:	528 Гб/ 4 Гб (1Гб на ядро)
Коммуникационная сеть:	Myrinet 2000

Для расчета на кластере использовался пакет LAMMPS, имеющий хорошие показатели по эффективности распараллеливания. Как видно из рис. 2, одна видеокарта GeForce 480GTX «заменяет» около

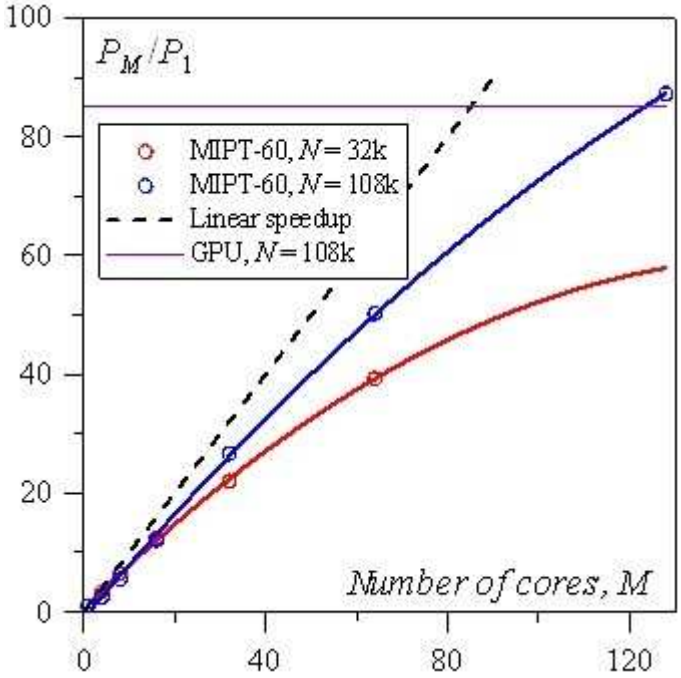

Рис 2. Отношение производительности M ядер к одному ядру для кластера МФТИ-60 (кружки). Быстродействие видеокарты показано сплошной горизонтальной линией. Для тестов использовались две системы с расплавом алюминия, содержащие 32000 (красные кружки) и 108000 (синие кружки) частиц.

120 ядер МФТИ-60, конечно при условии, что объема памяти видеокарты достаточно для полного размещении в ней задачи.

Следующим вопросом после ускорения, становится вопрос о совпадении результатом между CPU и GPU. Для исследования данного вопроса мы рассматривали задачу кристаллизации алюминия при помощи пакетов HOOMD и LAMMPS. Причем, на машине «X8 NVidia» одновременно запускался расчет 8-ми различных траекторий по одной траектории на видеокарте (что давало идеальную распараллеливаемость).

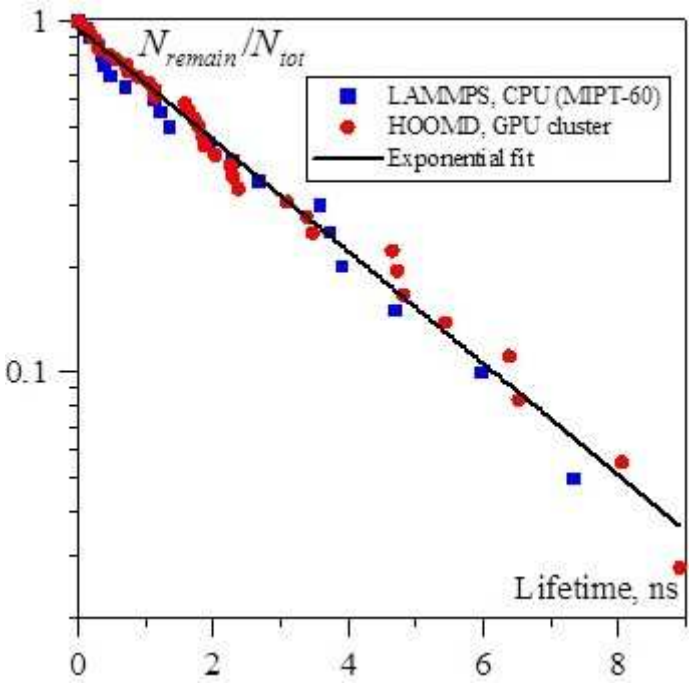

Рис 3. Распределение метастабильных состояний по временам жизни, полученное на CPU (квадраты) и GPU (кружки).

На рис. 3 показан результат расчета: распределение метастабильных состояний по временам жизни. Физических смысл здесь имеет наклон кривой, который определяет скорость кристаллизации переохлажденного алюминия. Как видно из рис. 3, точки полученные на GPU и CPU хорошо совпадают, несмотря на тот факт, что на видеокарте задача считалась с одинарной точностью, а на центральном процессоре с двойной. Кроме того на видеокарте применялась линейная интерполяция, а на процессоре сплайн 4-го порядка, что также не оказало влияния на результаты в пределах статистической погрешности.

Заключение

В работе предложена реализация многочастичного потенциала EAM для NVIDIA GPU на основе свободно распространяемого пакета HOOMD. Проведено тестирование быстродействия программ на CPU и GPU для различного типа моделируемых систем и различного числа частиц в МД ячейке. Для потенциала EAM на видеокарте GeForce 480GTX получено ускорении более чем в 8 раз по сравнению с двумя 4-х ядерными процессорами Intel Xeon E5520.

В целом следует отметить, что метод молекулярной динамики хорошо подходит для распараллеливания на GPU. Моделирование метастабильных состояний и релаксационных процессов, требующее одновременного расчета нескольких независимых траекторий, позволяют эффективно использовать кластеры из нескольких GPU для набора статистики. Для рассмотренных задач использование расчетов на GPU с одинарной точностью оказалось приемлемым, и полученные результаты хорошо согласуются с аналогичными расчетами на CPU с двойной точностью.

ЛИТЕРАТУРА:

1. 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.
2. Kuksin A.Yu., Morozov I.V., Norman G.E., Stegailov V.V., Valuev I.A. Standards for Molecular Dynamics «Modeling and Simulation of Relaxation» // Molecular Simulation. 2005. V. 31. P. 1005.
3. M. S. Daw, S. M. Foiles, and M. I. Baskes, «The Embedded Atom Method: A Review of Theory and Applications» // Mater. Sci. Rep. 9, 251-310 (1993).
4. J. A. Anderson, C. D. Lorenz, A. Travesset «General purpose molecular dynamics simulations fully implemented on graphics processing units» // Journal of Computational Physics, 2008, N. 227, P. 5342-5359.