Алгоритм расчёта ближних невалентных взаимодействий на процессоре PowerXCell8i. SPE-центричная модель

Н.А. Алемасов, Э.С. Фомин

Институт цитологии и генетики СО РАН, Новосибирск

Введение

Алгоритм расчёта ближних невалентных (электростатических и Ван-дер-ваальсовых) взаимодействий является наиболее вычислительно затратным в программах, реализующих метод молекулярной динамики (МД). Скорость работы данного алгоритма не может быть кардинально увеличена с помощью процессоров традиционной архитектуры x86, имеющих производительность не более 50 GFlop/s. В этой ситуации является естественным перенос вычислений на специализированные процессоры, такие как IBM Cell [1-3] и NVIDIA GPU [4, 5], так как эти процессоры обладают гораздо большей, чем традиционные, производительностью – от 150 GFlop/s до 1 TFlop/s.

Ранее [6, 7] нами сообщались результаты по увеличению эффективности выполнения алгоритма расчёта ближних невалентных взаимодействий в реализации для процессора Cell. В указанных работах была реализована PPE-центричная модель вычислений, в которой для подготовки данных для расчётов использовалось ядро PPE, в то время как SPE выполняли основные вычисления. Основным недостатком такой модели является ограничение производительности всей программы скоростью подготовки данных, так как PPE работает существенно медленнее, чем ядра SPE. В данной работе мы реализовали SPE-центричную модель вычислений, в которой SPE самостоятельно загружают необходимые для расчётов данные, тем самым, избегая зависимости скорости расчётов от скорости подготовки данных и существенно улучшая масштабируемость программы.

Исходный алгоритм

Данная работа была выполнена в рамках программного комплекса MOLKERN [8]. Электростатические взаимодействия в данном комплексе реализуются через разделение кулоновского потенциала на ближнюю $\mathrm{erfc}(\alpha \times r)/r$ и дальнюю части $\mathrm{erf}(\alpha \times r)/r$ с последующим прямым суммированием ближних кулоновских взаимодействий и расчётом дальних кулоновских взаимодействий с помощью P^3M метода [9]. Ван-дер-ваальсовые взаимодействия представляются 6-12 потенциалом Леннарда-Джонса. Подход, основанный на разделении взаимодействия на ближнюю и дальнюю части уменьшает вычислительную сложность расчётов ближних невалентных взаимодействий с $O(N^2)$ до $O(N \times \log N)$.

Детали исходного последовательного алгоритма и его модификации для параллельного исполнения описаны нами ранее в [6, 7].

Процессор PowerXCell8i

Архитектура Cell Broadband Engine (CBEA) была разработана при сотрудничестве компаний Sony, Toshiba и IBM. Процессор на её основе начал применяться для обработки мультимедиа и векторных данных, но впоследствии на него стали переноситься и научные расчёты. Процессор PowerXCell8i [10] (далее – просто Cell) представляет собой многоядерный процессор, состоящий из 9 ядер. Одно ядро – PPE (PowerPC Processor Element) – архитектуры PowerPC, предназначенное для взаимодействия с операционной системой. Остальные восемь ядер, SPE (Synergistic Processor Element), имеют отличную от PPE архитектуру и используются, как основные вычислительные устройства. SPE не имеют прямого доступа в основную память и могут напрямую работать только с локальной памятью, LS. Память LS имеет размер 256 Кб и предназначена, как для хранения кода, так и для хранения данных. Для обеспечения большей пропускной

способности при доступе к данным, в процессоре Cell присутствует DMA-контроллер, позволяющий осуществлять DMA-передачи между любыми двумя ядрами параллельно с вычислениями на них. Каждый SPE имеет 128 128-битных регистров, позволяющих компилятору перегруппировывать команды, а также широкий набор SIMD-инструкций, позволяющий эффективно работать с этими регистрами. В качестве типов данных с плавающей точкой могут быть использованы типы одинарной и двойной точности.

Тестирование производительности параллельной реализации рассматриваемого в данной работе алгоритма происходило на сервере PeakCell S [11]. Сервер содержит два процессора PowerXCell8i с частотой 3.2 ГГц, 16 Гб оперативной памяти и работает под управлением ОС Fedora 7. Оба процессора системы соединены шиной FlexIO, имеющей пропускную способность в 20 Гб/сек. Память вычислительных узлов организована по схеме NUMA (Non-Uniform Memory Architecture).

Программные инструменты

В качестве инструментов для параллельной реализации была выбрана runtime-библиотека libspe2 из IBM SDK for Multicore acceleration for Linux [12] и библиотека Pthreads для создания SPE-потоков. Все SPE-потоки создаются при запуске основной программы, а уничтожаются, соответственно, при её завершении. После запуска каждый SPE-поток в цикле ожидает команду от PPE, выполняет её и после отправляет статус выполнения на PPE.

При переносе функции расчёта парных невалентных взаимодействий dU__dX() на SPE, весь вычислительный код был переписан с учётом поддержки SIMD-инструкций. Использовались инструкции с элементами векторов одинарной точности, поскольку для молекулярной динамики потеря в точности не является существенной. И, поскольку для нашей реализации характерно большое количество векторов ограниченной длины, при выборе библиотеки векторных операций мы решили остановиться на inline-варианте библиотеки MASS.

SPE-центричная модель

SPE-центричная модель распределения задач в отличие от PPE-центричной, реализованной нами ранее, не предполагает подготовки данных на стороне PPE. В этой модели SPE самостоятельно обращаются в основную память за данными, необходимыми для расчётов. Таким образом, скорость расчётов на SPE не зависит от производительности внешних потоков и может масштабироваться на любое количество ядер SPE. Этот факт играет существенную роль в ситуации, когда скорость обработки данных превышает скорость их подготовки.

В SPE-центричной модели программа организована сходным с PPE-центричной моделью образом. Она имеет функцию подготовки блоков «данных» – put(),– функцию выполнения основных расчётов – calculate(),– а также функцию сбора и сохранения результатов – get() (см. рис. 1). Однако, в отличие от алгоритма PPE-центричной модели, функция put() на PPE занимается подготовкой блоков, содержащих не сами атомы, а только указатели на них. Реальная же загрузка атомов происходит на стороне SPE по полученным от PPE указателям. Основные вычисления происходят на SPE, а их результат отправляется обратно на PPE, где, как и в предыдущей модели [6], полученные от SPE силы и энергии, суммируются в функции get().

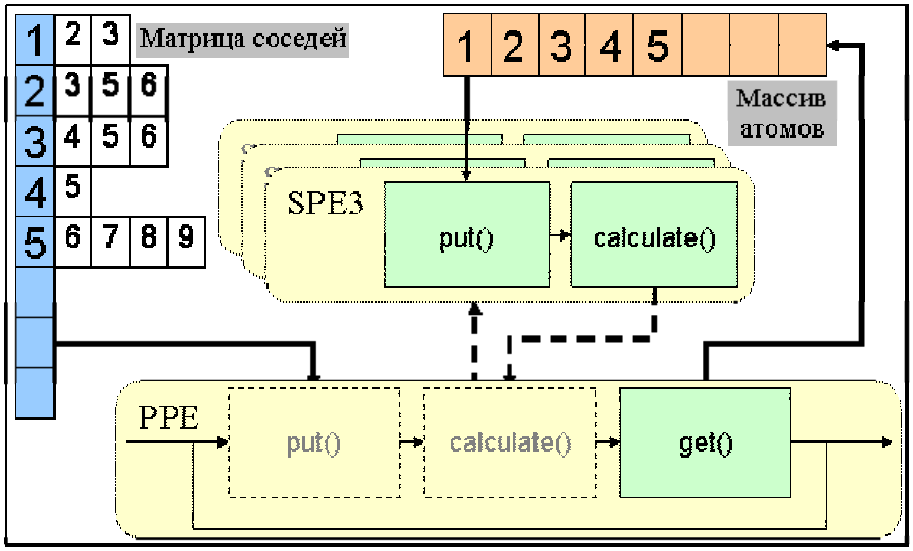

Рисунок 1. Схема алгоритма SPE-центричного расчёта энергии и сил.

При реализации SPE-центричной модели, была также изменена последовательность работы функций put() и calculate(). Ранее использовалась подобная конвейеру схема put() – calculate() – get(). Она последовательно работала с большими (в десятки килобайт) блоками данных и позволяла разделить по времени операции работы с памятью и вычислительные операции. Это увеличивало производительность программы на обычных процессорах, которые обладают кэш-памятью. В реализациях для Cell такая схема не позволяла в полной мере использовать одну из основных особенностей процессора Cell – принимать и передавать данные в локальную или основную память параллельно с выполнением вычислений. Поэтому перенос функции put() на SPE и уменьшение размера блока до небольшой порции данных дали нам возможность совместить DMA-операции обращения в память и расчёты. Каждая порция данных содержала параметры для 4 атомов, что позволило разместить их в элементах SIMD-векторов, и вычисления проводить с помощью SIMD-инструкций.

Результаты и обсуждение

На рисунке 2 представлена в логарифмическом масштабе диаграмма ускорения исходной функции в зависимости от количества используемых SPE. Для тестов был использован комплекс 1GC1 в водном окружении, с числом атомов 124 723 и числом пар атомов более 25.0 млн.

Для PPE-центричной модели максимальное ускорение достигалось при использовании 8 SPE и относительно одного PPE составляло приблизительно 32 раза; относительно процессора AMD Athlon X2, 2.6 Ггц в однопоточном режиме – 6 раз. SPE-центричная модель даёт существенно лучшие результаты. Максимальное ускорение достигается при использовании 16 SPE и составляет 149 и 30 раз относительно одного PPE и процессора AMD соответственно. Кроме того, в случае использования модели с «активными» SPE-потоками наблюдается практически линейное ускорение с увеличением числа SPE, занятых в расчётах. Только при переходе от загрузки одного процессора Cell к загрузке двух, происходит заметное ухудшение масштабируемости, что может быть связано с организацией памяти NUMA сервера PeakCell S.

Заключение

Была реализована SPE-центричная модель вычислений для алгоритма расчёта ближних невалентных взаимодействий. Данная схема показала практически линейную масштабируемость в зависимости от числа используемых SPE (вплоть до 8 SPE), что

говорит о высокой оптимизации параллельной реализации. В данной модели было достигнуто 149-кратное ускорение относительно программы, запущенной на одном ядре PPE и 30-кратное ускорение вычислений относительно системы на базе процессора AMD Athlon X2.

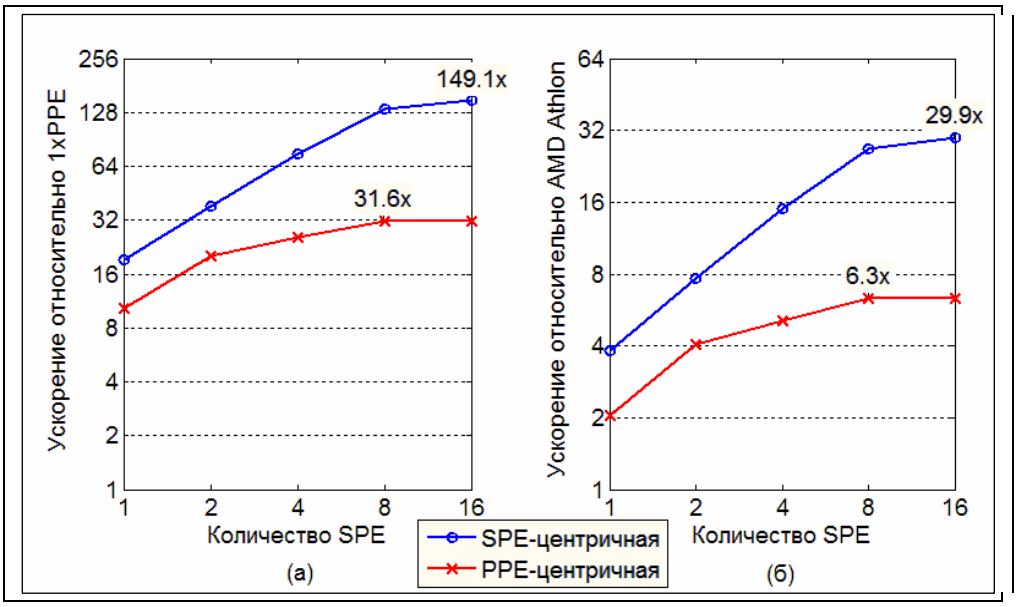

Рисунок 2. Диаграмма зависимости ускорения функции dU__dX() от числа используемых SPE. (а) – ускорение относительно одного PPE, (б) – относительно однопоточной программы на процессоре AMD Athlon X2, 2.6 ГГц.

Работа выполнена при поддержке грантов: Междисциплинарный интеграционный проект фундаментальных исследований СО РАН № 26 «Математические модели, численные методы и параллельные алгоритмы для решения больших задач СО РАН и их реализация на многопроцессорных суперЭВМ» и Междисциплинарный интеграционный проект фундаментальных исследований СО РАН № 113 «Разработка вычислительных методов, алгоритмов и аппаратурно-программного инструментария параллельного моделирования природных процессов». Авторы благодарят компанию T-platforms (http://www.t-platforms.ru) за предоставление доступа к серверу PeakCell S с двумя процессорами PowerXCell8i.

Литература

1. Olivier S., Prins J., Derby J. Porting the GROMACS Molecular Dynamics Code to the Cell Processor // 21st International Parallel and Distributed Processing Symposium (IPDPS 2007). Proceedings. Long Beach, California, USA, 26-30 March 2007. P. 1–8.
2. Shi G., Kindratenko V. Implementation of NAMD molecular dynamics non-bonded force-field on the Cell Broadband Engine processor // 9th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC 2008). Proceedings. 2008.
3. Using Cell Broadband Engine Technology to Improve Molecular Modeling Applications // SimBioSys: White Papers. 2009. [Electronic resource]. URL: http://www.simbiosys.ca/science/white_papers/IBM_eHiTS_BLW03019USEN_1.1.pdf (date of access 09.29.2009).

4. Anderson J. A., Lorenz C. D., Travesset A. General Purpose Molecular Dynamics Simulations Fully Implemented on Graphics Processing Units // J. Comput. Science. 2008. Vol. 227. P. 5342.
5. Liua W., Schmidt B., Vossa G., Müller-Wittig W. Accelerating molecular dynamics simulations using Graphics Processing Units with CUDA // Computer Physics Communications. 2008. Vol. 179, № 9. P. 634–641.
6. Фомин Э.С., Алемасов Н.А. Оптимизация вычислительного ядра библиотеки молекулярного моделирования MOLKERN под архитектуру Cell // Параллельные вычислительные технологии (ПаВТ'2009): Труды международной научной конференции. Нижний Новгород, 30 марта – 3 апреля 2009 г. Челябинск: Изд. ЮУрГУ, 2009. С. 772–777.
7. Fomin E., Alemasov N. Implementation of Non-bonded Interaction Algorithm for the Cell Architecture. // Parallel computing technologies 2009 / Malyshkin V; Springer. LNCS, Vol. 5698. 2009. P. 399–405.
8. Фомин Э. С., Алемасов Н. А.,Чирцов А. С., Фомин А. Э. Библиотека программных компонент MOLKERN для построения программ молекулярного моделирования // Биофизика. 2006. Т. 51, Вып. 7. С. 110–113.
9. Hockney, R., and Eastwood, J. Computer simulation using particles. New York: McGraw-Hill, 1981.
10. IBM PowerXCell 8i processor datasheet // IBM: Resources. 2009. [Electronic resource]. URL: http://www-03.ibm.com/technology/resources/technology_cell_pdf_PowerXCell_PB_7May2008_pub.pdf (date of access 09.29.2009).
11. Сервер PeakCell S // T-Платформы. 2009. [Электронный ресурс]. URL: http://www.t-platforms.ru/ru/tcell/peakcellsserver.html (дата обращения 29.09.2009).
12. Programmer's Guide to the IBM SDK for Multicore Acceleration v3.0 // IBM. 2009. [Electronic resource]. URL: https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/1DAAA0A3B6404763002573530066008C/$file/CBE_Programmers_Guide_v3.0.pdf (date of access 09.29.2009).