Оптимизация вычислительного ядра библиотеки молекулярного моделирования MOLKERN под архитектуру Cell*

Э.С. Фомин, Н.А. Алемасов

Общим «узким» местом программ молекулярной динамики является расчет сил парных невалентных взаимодействий. Процессор Cell, разработанный STI, имеет большой потенциал для того, чтобы заметно ускорить подобные расчеты. В работе демонстрируются результаты переноса на архитектуру Cell ядра программного комплекса MOLKERN, предназначенного для моделирования структуры и динамики комплексов биомакромолекул. Показано, что оптимизация функции расчета ближних парных невалентных взаимодействий дает 5-6 кратный рост ее производительности, что увеличивает общую производительность комплекса до 60% для ряда задач.

1. Введение

Общим «узким местом» программ молекулярной динамики является функция расчета сил, обусловленных невалентными парными (электростатическими и Ван-дер-ваальсовыми) взаимодействиями. Полное время выполнения данной функции превышает 4/5 от полного времени выполнения программ. Тесты производительности программы GROMACS по моделированию небольшого из 35 аминокислотных остатков белка villin headpiece в ячейке, содержащей 3000 молекул воды, показали значение в 83% [1] для данной функции. Причем, в данном примере подавляющая часть взаимодействующих пар связана с молекулами воды, для которых программа GROMACS использует эффективные алгоритмы оптимизации [2].

При расчете парных взаимодействий используются разные методы для суммирования парных взаимодействия в ближней и дальней области, условно разделяемых параметром r_{cut}. Расстояние r_{cut} подбирается таким образом, чтобы большая часть взаимодействия, резко меняющаяся при малых расстояниях, оказалась в области $r < r_{cut}$, оставив за пределами r_{cut} плавно спадающую асимптотику. Это позволяет делать суммирование парных взаимодействий при малых расстояниях в координатном пространстве, а для больших расстояний в обратном k-пространстве. Стандартным значением r_{cut} является величина в диапазоне от 0.8 до 1.4 nm. Данный подход, основанный на разделении взаимодействия на части, позволяет уменьшить вычислительную сложность алгоритмов с $O(N^2)$ до $O(N \log N)$.

Для ближней области $r < r_{cut}$, функция расчета взаимодействий стандартно разбивается на два шага. В первом, вычислительно более трудоемком, шаге находятся все пары атомов, расположенные на расстоянии друг от друга не более заданного r_{cut} и создаются, так называемые, списки соседей. На втором шаге производятся расчеты энергии и сил для всех пар атомов, попавших в списки соседей. Разбиение на два шага оптимизирует выполнение, позволяя избежать обновления списка соседей на каждой итерации.

Тем не менее, расчеты энергии и сил для всех пар атомов, попавших в списки соседей, по прежнему занимают большую часть времени в молекулярной динамике. В последнее время, для ускорения подобных алгоритмов используется их распараллеливание как для вычислительных кластеров, построенных на x86 процессорах [3], так и на спецпроцессорах, типа GPU[4-5] и Cell [6-8].

* 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 поддержку данной работы и предоставление доступа к серверу с двумя PowerXCell8i для запуска приложений для Cell/B.E.

2. Постановка задачи

Архитектура Cell Broadband Engine (CBEA) была разработана при сотрудничестве компаний Sony, Toshiba и IBM. Процессор на её основе начал применяться для обработки мультимедиа данных, но впоследствии на него стали переноситься и научные расчёты [6-8].

Процессор Cell/B.E. [9] представляет собой многоядерный процессор, содержащий 9 ядер. Одно ядро — PPE (PowerPC Processor Element) — процессор архитектуры PowerPC, предназначенный для взаимодействия с операционной системой. Оставшиеся 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, поэтому приходится учитывать затраты на пересылки из основной оперативной памяти, доступной PPE в локальную память SPE — LS (Local Storage). Память LS имеет размер 256 Кб и предназначена, как для хранения кода, так и для хранения данных. Передачи данных происходят по шине EIB (Element Interconnect Bus), соединяющей все вычислительные элементы процессора. Для обеспечения большей пропускной способности шины, в процессоре Cell присутствует DMA-контроллер – он позволяет без участия PPE запрашивать данные из основной памяти и помещать их в LS, а также производить операции в обратном направлении, от SPE к PPE и операции передачи данных между SPE.

SPE имеет 128 128-битных регистров и широкий набор SIMD-инструкций для работы с ними. В качестве типов данных с плавающей точкой могут быть использованы типы одинарной и двойной точности. Помимо возможности векторной обработки, присутствует возможность выполнения до двух команд за один такт; один слот выдачи команды поддерживает операции с плавающей и фиксированной запятой, а другой обеспечивает загрузку (сохранение), операции перестановки байтов и перехода. Все эти особенности Cell-процессора позволяют получить существенно больший прирост скорости выполнения программ по сравнению с x86 архитектурой.

Программный комплекс MOLKERN [10] предназначен для решения задач, связанных с моделированием структуры и динамики комплексов биомакромолекул, состоящих из десятков и сотен тысяч атомов. В силу высоких требований на скорость вычислений в подавляющем большинстве задач молекулярного моделирования (докинг, виртуальный скрининг, оптимизация структуры, молекулярная динамика и т. д.), в реализации комплекса MOLKERN используются оптимизированные алгоритмы с вычислительной сложностью не выше O(N log(N)). Библиотека MOLKERN написана на языке C++ с использованием библиотек STL, BOOST, BLAS, MPI и интерфейса OpenMP.

Программа MOLKERN следует стандартному подходу по расчету парных взаимодействий. Она вводит параметр r_{cut} для разделения ближней и дальней части взаимодействий, составляет списки соседей для области $r < r_{cut}$, и выполняет для них прямое суммирование Ван-дер-ваальсовых и ближних кулоновских взаимодействий, выполняет расчет дальней кулоновской части потенциала методом PPPM для $r > r_{cut}$. Результаты профилирования программы для задач не требующих расчета дальних кулоновских взаимодействий (оптимизация геометрии, локальная молекулярная динамика) показывают, что большая часть расчетного времени 80-86% при r_{cut} = 1 nm тратится на прямое суммирование Ван-дер-ваальсовых и ближних кулоновских взаимодействий для пар, попавших в списки соседей. Увеличение производительности выполнения функции, выполняющей данное суммирование, дает прямой путь увеличения производительности программы.

Исходная версия программного комплекса MOLKERN реализована для процессоров архитектуры x86. Целью данной работы являлся перенос библиотеки молекулярного моделирования MOLKERN на архитектуру процессора Cell и оптимизация наиболее затратной функции вычислительного ядра библиотеки dU__dX, связанной с вычислением энергии и сил, образованных всеми парными взаимодействиями в области $r < r_{cut}$.

3. Описание программой реализации алгоритма

3.1 Предварительная модификация кода

Для стандартной реализации функции расчета парных сил характерно множество операций «чтение-модификация-запись» по ячейкам памяти в диапазоне адресов, определяемыми структурами данных, содержащим всю необходимую информацию для расчетов (атомные параметры, координаты, силы). Размер диапазона адресов при расчетах больших систем значителен и не зависит от того, каким образом организовано хранение этих данных. Эти данные не могут быть помещены целиком в кэш-память компьютера, имеющей максимальную скорость доступа (кэш L1 на архитектуре x86, LS на архитектуре Cell и т.п.), потому порядок обращения к памяти существенно влияет на скорость выполнения.

Кроме того, для алгоритмов расчета парных сил характерно хаотичное распределение данных, поскольку номера атомов, попадающих в пары, могут иметь достаточно большую разницу в значениях. Загрузка в кэш-память даже большого блока данных, с адресов близких к запрашиваемым, при расчете взаимодействий для некоторой пары атомов, не гарантирует, что для следующей пары атомов уже загруженный блок данных будет включать нужную информацию и не потребуется перезагрузка кэша. Это является причиной многочисленных промахов при работе с кэшем и соответствующего падения производительности приложения.

Для уменьшения числа промахов с кэшем необходимо обеспечить упорядочение данных таким образом, чтобы максимизировать вероятность того, что атомы соседних пар имеют близкие номера. Данное условие обычно выполняется в начале любого процесса моделирования благодаря способу хранения данных в pdb формате, который обеспечивает близость номеров для близких в пространстве атомов. Данная упорядоченность сохраняется при оптимизации геометрии, поскольку в этом процессе маловероятны дальние смещения атомов, и исчезает только в длительной молекулярной динамике. Для восстановления упорядоченности в этом случае используются алгоритмы переупорядочивания частиц, построенные на использовании кривой Гильберта-Пеано [11], и затраты на использование которых малы в силу относительно редкого их использования.

Другим стандартным способом минимизации промахов кэша является уменьшение размеров структур данных, необходимых для расчетов. Удаление из структур данных лишней информации увеличивает диапазон загружаемых элементов массивов, что, в свою очередь, увеличивает вероятность нахождения нужных данных в кэше.

Поскольку для программы MOLKERN существует упорядоченность данных, обусловленная начальной загрузкой, то пути повышения производительности могут быть связаны с минимизацией структур данных и числом обращений в память.

То, что число обращений к памяти необходимо не просто минимизировать, а даже полностью исключить (и именно на этапе непосредственного расчета), показал наш опыт первого неудачного портирования MOLKERN на Cell, в котором был использован интерфейс OpenMP для Cell/B.E. В отличии от платформы x86, где использование OpenMP оказалось оправданным и давало 45% на дополнительное ядро процессора [12], на платформе Cell не удалось получить какого-либо выигрыша. Даже при использовании 17 ядер (сервера с двумя PowerXCell8i), доступных программе в виде нитей OpenMP, программа работала существенно (в десятки раз) медленнее, чем программа, использующая только PPE для расчётов. Данный факт был исследован — программа требовала слишком интенсивных передач данных между основной памятью и LS SPE. В то же время с помощью OpenMP невозможно контролировать подобные передачи.

Таким образом, для минимизации обращений к памяти и удаления их с этапа непосредственного расчета функция расчета парных взаимодействий dU__dX была разбита на три функции, которые выполняли следующие задачи для любой заданной совокупности пар:

- get() - считывание всех данных из памяти в блок данных;
- calculate() - расчет энергий и сил для пар и сохранение их в этом же блоке;
- put() - рассылка рассчитанных данных из блока в память.

Выигрыш данной схемы расчета, относительно привычного для программиста создания цикла, внутри которого происходит цикл «чтение-модификация-запись» для каждой пары отдельно, заключается в том, что запросы в память не прерываются расчетным кодом, тем самым

не провоцируются дополнительные перезагрузки кэша. Другим и существенным преимуществом схемы является возможность выполнения функции calculate() в другом потоке выполнения, в то время как основной поток может заниматься выборкой из/в память для другого блока данных, что особенное важно в случае, когда calculate() занимает достаточно много времени. Этот подход характеризуется естественным распараллеливанием задачи и отсутствием какой-либо необходимости на синхронизацию подзадач, выполняющихся в различных потоках, в силу их независимости друг от друга. Все, что должен делать основной поток — это загружать данные в блоки и после получения сигнала о завершении от какого-либо блока пересылать его данные в память. Поскольку пересылкой из/в память занимается только один поток, также нет необходимости блокировки памяти, которая является затратной операцией, поскольку требует системных вызовов.

Таким образом, была сформирована следующая архитектура программного комплекса MOLKERN в адаптации под Cell процессор. MOLKERN полностью выполняется на PPE за исключением одной, вычислительно затратной функции dU__dX, вычисляющей энергию и силы ближних парных невалентных взаимодействий. Данная функция dU__dX разбивается на три подфункции, две из которых также выполняются на PPE {get(), put()} и пересылают данные из/в память, а третья calculate() выполняется на каждом SPE и рассчитывает энергии и силы для всех пар в переданном SPE блоке данных.

3.1 Портирование функций dU__dX::{get, calculate, put} на Cell SPE.

Набор из трех функций dU__dX::{get, calculate, put} был реализован для выполнения на Cell SPE в двух вариантах, отличающихся друг от друга распределением выполняемой работы между PPE и SPE.

В варианте (1) на SPE рассчитывались энергии и силы для каждой пары отдельно, а работа по суммированию всех сил предоставлялась PPE. С алгоритмической точки зрения этот вариант максимально прост, поскольку это суммирование выполняется во время пересылки в память путем добавления значения к уже накопленному значению. Недостатком варианта (1) является то, что при записи результатов к одной и той же ячейке в памяти приходится обращаться многократно, из-за того, что один и тот же атом входит во множество пар.

В варианте (2) на SPE не только рассчитывались энергии и силы для каждой пары, но и происходило суммирование значений для всех пар, попавших в блок данных. Такой подход увеличивает сложность программирования существенно (отношение времен разработки для двух вариантов приблизительно равно 1 : 20 и более), однако он компенсируется большей эффективностью выполнения кода.

Для обоих вариантов был написан одинаковый код инициализации потоков и пересылки данных между PPE и SPE, по размеру не превышающим 300 строк. В реализации использовалась библиотека libspe 2 IBM SDK [13].

4. Результаты вычислительных экспериментов

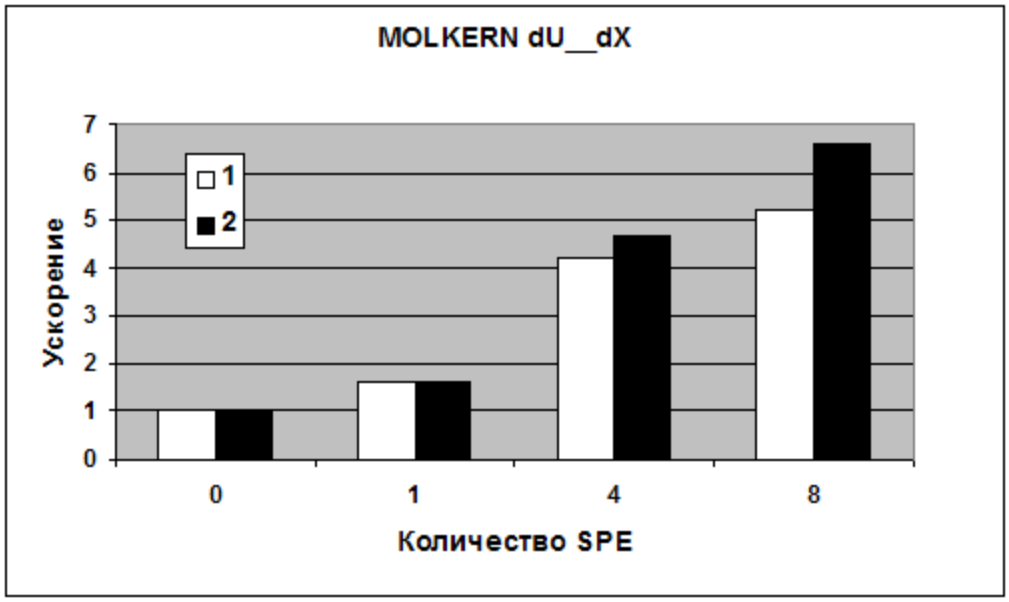

Рис. 1. Диаграмма зависимости ускорения функции dU__dX от числа используемых SPE.

На рис.1 приведена диаграмма, демонстрирующая зависимость ускорения счета функции dU__dX из ядра MOLKERN от числа используемых SPE для двух вышеупомянутых вариантов реализации. Белыми столбцами отмечены значения ускорения для варианта (1), в котором суммирование значений сил для одинаковых атомов, попавших в разные пары, осуществлялось только PPE. Черными столбцами отмечены значения ускорения для варианта (2), в котором SPE выполняли частичное суммирование сил для таких атомов внутри переданного им блока данных. Значение числа SPE равное 0 означало, что расчет делался только на PPE. В этом случае на PPE выполнялся исходный не модифицированный код программы MOLKERN. Приведенные на диаграмме результаты получены усреднением времени выполнения функции dU__dX по первым 5-ти итерациям процесса оптимизации геометрии для белка 1GC1, состоящего из 14104 атомов. В расчетах использовались стандартный 6-12 потенциал для описания Ван-дер-ваальсовых взаимодействий и потенциал (1 - erf(alpha * r)) / r, эффективно подавляющий кулоновское взаимодействие на расстояниях более чем $\sqrt{\pi}$ / alpha и называемый ближним кулоновским взаимодействием.

Как видно из диаграммы, с увеличением числа используемых SPE наблюдается рост ускорения выполнения функции dX__dU ядра MOLKERN. Максимальное ускорение, полученное для варианта (1) равно 5.25, а для варианта (2) - 6.57. При дальнейшем увеличении числа задействованных в расчетах SPE, дополнительного повышения производительности не наблюдалось, что, предположительно, обусловлено рассогласованием загрузки PPE и SPE, то есть PPE не успевал обработать данные, поступающие на него от многих SPE, вынуждая последние частично простаивать. Косвенным подтверждением возникновения рассогласования нагрузки PPE и SPE служит также тот факт, что при снижении нагрузки на SPE, сделанное заменой в расчетах ближнего кулоновского потенциала на более простой полный кулоновский потенциал 1 / r, максимум ускорения наблюдался уже при 4 SPE и не превышал значения в 2.26 раза.

5. Выводы и направления дальнейших исследований

Перенос dU__dX функции ядра программного комплекса MOLKERN на архитектуру процессора Cell уменьшил время ее выполнения приблизительно в 5-6 раз, что существенно увеличило общую производительность выполнения ядра (приблизительно до 60% для ряда задач, не требующих расчета дальних кулоновских взаимодействий). Мы полагаем, что возможно получить сходное по порядку величины увеличение производительности MOLKERN также для за-

дач, где подобный расчет необходим (например, для длительной молекулярной динамики). Современные методы расчета дальнего кулоновского взаимодействия используют быстрое Фурье преобразование на дискретной пространственной сетке и при уменьшении шага пространственной сетки могут занимать больше времени, чем расчет парных потенциалов в области r < 0.8-1.2 nm. Задача оптимизации для Cell соответствующей функции расчета ядра MOLKERN запланирована. Мы полагаем, что благодаря появлению стабильной версии пакета FFTW 3.2 с поддержкой Cell [14], возможно решить эту задачу с минимальными усилиями.

Литература

1. GROMACS. Benchmarks: [http://www.gromacs.org/content/view/24/38/], 09 September 2005.
2. Berendsen H.J.C., Postma J.P.M., van Gunsteren W.F., Hermans J. Interaction models for water in relation to protein hydration. // In: Intermolecular Forces. Pullman B. D. Reidel Publishing Company Dordrecht. –1981. –P.331–342.
3. Khodade P., Prabhu R., Chandra N., Raha S., Govindarajan R. Parallel implementation of AutoDock. // Applied Crystallography. –2007. –Vol.40. –P. 598-599.
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. Fabritiis G. D. Performance of the Cell processor for biomolecular simulations. // Computer Physics Communication. –2007. –Vol. 176, №11/12. –P.660-664.
7. Olivier S., Prins J., Derby J. Porting the GROMACS Molecular Dynamics Code to the Cell Processor. // 21th International Parallel and Distributed Processing Symposium (IPDPS 2007). Proceedings. –26-30 March 2007. –Long Beach, California, USA.
8. Shi G., Kindratenko V. Implementation of NAMD molecular dynamics non-bonded force-field on the Cell Broadband Engine processor //In Proc. 9th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing. –PDSEC, –2008.
9. Kahle J. A., Day M.N., Hofstee H.P., Johns C.R., Maeurer T.R., Shippy D. Introduction to the Cell multiprocessor. // IBM Journal of Research and Development. –2005. –Vol. 49, № 4/5. –P.589-604
10. 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.
11. Moon B., Jagadish H., Faloutsos C., Saltz J.H. Analysis of the clustering properties of the hilbert space-filling curve. // IEEE Transactions on Knowledge and Data Engineering –2001. –Vol.13, № 1. –P.124-141.
12. Alemasov N. A., Fomin E. S. OPENMP+MPI parallel implementation of the MOLKERN molecular modeling software package // Proceedings of the 6th Internation Conference on Bioinformatics of Genome Regulation and Structure. –2008. –P.24.
13. IBM SDK for Multicore acceleration v3.0 [http://www.ibm.com/developerworks/power/cell/]
14. Frigo M., Johnson S.G. FFTW on the Cell Processor –2007. [http://www.fftw.org/cell/index.html]