[1*]Э.С. Фомин, [1]Н.А. Алемасов, [2]С.А.Матвиенко
[1] Институт цитологии и генетики СО РАН
[2] Новосибирский государственный университет

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ПРОЦЕССОРОВ POWERXCELL8I И INTEL X7560 НА ПРИМЕРЕ ЗАДАЧ МОЛЕКУЛЯРНОЙ ДИНАМИКИ

Метод молекулярной динамики (МД) является методом компьютерного моделирования, позволяющим в течение заданного периода времени проследить эволюцию системы взаимодействующих частиц с помощью численного интегрирования уравнений движения Ньютона [1]. В молекулярной биологии метод МД широко используется для решения задач исследования термостабильности белков, конформационных переходов, транспорта молекул, белкового фолдинга и проч. Программы МД могут служить также тестами производительности различных вычислительных платформ благодаря следующим особенностям метода:

– алгоритмическая простота (используются общеизвестные уравнения Ньютона);

– естественный параллелизм высокой степени (силы, действующие на каждую частицу могут быть рассчитаны независимо, и возможная степень параллелизма по порядку величины сравнима с числом частиц в исследуемой системе);

– большой объем вычислений (запросы реальных задач).

Программы МД работают на всех существующих в мире вычислительных платформах, использующих как стандартные Intel, AMD, Alpha, так и специализированные Cell [2, 3] и GP GPU [4, 5] процессоры. Данная работа посвящена сравнительному анализу эффективности выполнения задач молекулярной динамики на сравнимых по архитектуре серверах, построенных на процессорах Cell8i и X7560. Доступ к серверу, вычислительные узлы которого состоят из двух PowerXCell8i-процессоров, каждый из которых включает 8 SPE ядер, был предоставлен компанией T-platforms [6]. Доступ к серверу, вычислительные узлы которого построены на четырех восьмиядерных Intel X7560-процессорах был предоставлен компанией Intel в рамках

Manycore Testing Lab [7]. Память вычислительных узлов для обоих кластеров организована по схеме NUMA.

Инструменты и методы

Исходный алгоритм и тесты

Данная работа выполнена с использованием программы молекулярного моделирования MOLKERN [8], использующей силовое поле AMBER [9]. Для снижения вычислительной сложности с $O(N^2)$ до $O(N)$ при расчете парных взаимодействий N частиц область действия потенциалов ограничивается некоторым радиусом r_{cut}. Все пары атомов, для которых расстояния между атомами $r < r_{cut}$, помещаются в списки соседей. Для создания списка соседей используется улучшенный метод Верлет-таблицы [10], разделяющей пространственную область системы на ячейки с длиной ребра r_{cut} и с поиском соседей для каждого атома только в ячейке самого атома и в 26 соседних ячейках. Для избежания излишних расчетов расстояний между атомами всевозможных пар соседних ячеек атомы в ячейках упорядочиваются согласно проекции их координат на вектор, соединяющий данные ячейки, как предложено в [11]. В расчетах используется третий закон Ньютона, и никакие взаимодействия не вычисляются дважды, поэтому Верлет-таблица хранит индексы (i, j) только тех пар атомов, для которых выполняется условие $i<j$. Для минимизации числа промахов кэша все данные в памяти компьютера упорядочиваются согласно LCR-алгоритму [12]. Цель упорядочения – обеспечить выполнение принципа локальности, то есть данные, связанные с близкими в пространстве атомами, должны располагаться в памяти компьютера так же недалеко друг от друга.

В расчетах использовалась область 120 x 120 x 120 A^3, содержащая 59 тысяч молекул воды, которые помещались в нее случайным образом с плотностью 0,03 молекулы A^{-3}. Использовалась SPC-модель воды. Для тестирования производительности использовались кулоновский $1 / r$ и 6-12 потенциал Леннарда – Джонса $(\varepsilon/4) [(\sigma/r)^{12} – (\sigma/r)^6]$. Все потенциалы обрезались при 10 А. Дополнительно использовался потенциал $\mathrm{erfc}(\sqrt{\pi} * r / r_{cut}) / r$, называемый ближним кулоновским потенциалом, совпадающий с кулоновским потенциалом при $r \rightarrow 0$. В начале любого расчета

выполнялось 10 итераций оптимизации геометрии системы. Далее, согласно Больцмановскому распределению все молекулы получали случайные значения скоростей, и выполнялась МД в диапазоне 0,1 ps. Использовался Verlet leap-frog-алгоритм [13] для интегрирования уравнений движения с шагом в 1 *fs*. Во тестах использовался NVT ансамбль при температуре $T = 300$ К.

Процессор PowerXCell8i

Процессор PowerXCell8i [14] содержит 9 ядер. Одно ядро – PPE (PowerPC Processor Element) – процессор архитектуры PowerPC, предназначенный для взаимодействия с операционной системой. Остальные восемь ядер, SPE (Synergistic Processor Element) используются как основные вычислительные устройства. SPE могут напрямую работать только со своей локальной памятью (LS) размером в 256 Кб. Передачи данных происходят по кольцевой шине EIB (Element Interconnect Bus), соединяющей все вычислительные элементы процессора и имеющей пропускную способность более 200 Гб/с. В процессоре есть DMA-контроллер, позволяющий без участия PPE и SPE передавать данные между основной памятью и LS. Каждый SPE имеет 128 128-битных регистров с широким набором SIMD-инструкций.

Сервер PeakCell S содержит два процессора PowerXCell8i с частотой 3.2 ГГц и 16 Гб DDR2 оперативной памяти. Оба процессора системы соединены шиной FlexIO с пропускной способностью в 20 Гб/с.

Особенности реализации кода для процессора Cell

Для минимизации программистских затрат для Cell процессора была выбрана модель реализации, в которой большая часть кода выполнялась на PPE, а на SPE выполнялся только один, наиболее затратный по времени, алгоритм расчета ближних невалентных взаимодействий.

Реализация кода для SPE следовала всем рекомендациям из IBM SDK [15]. Загрузки данных и счет выполнялись на SPE, а PPE выступал в роли диспетчера ресурсов (SPE-центричная модель программирования). Для совмещения операций чтения данных и оперирования с ними использовалась техника двойной буферизации, для ускорения загрузки данных – выравнивание данных на гра-

ницу 128 байт как для приемника, так и для передатчика данных. Реализация доступа ядер SPE в основную память выполнена с помощью библиотеки libspe2, а векторизация и многопоточность – с помощью MASS и Pthreads, соответственно.

Процессор Xeon

Процессор Intel Xeon X7560 на данное время является последней разработкой в серии Intel Xeon-процессоров и основан на микроархитектуре Nehalem. Процессор включает: 8 ядер, работающих на частоте 2,26GHz, L2 кэш с 256KB памяти на ядро, 24MB на L3 кэше (разделяемом между ядрами), поддержку набора инструкций x86-64 и технологии Hyper Threading. Процессоры системы соединены высокоскоростной (до 25.6 GB/s) шиной Intel® Quickpath Interconnect, позволяющей объединять по схеме «каждый с каждым» до 8 процессоров и поддерживающей до 16 слотов 16GB DDR3 памяти на каждый процессор.

Особенности реализации кода для процессора Intel Xeon

Благодаря тому что разработка кода для данного процессора имеет меньшую трудоемкость, чем для Cell-процессора, векторизация и параллелизация делалась для большей части программы, что включало: расчет навалентных взаимодействий, построение списков атомов для ячеек, сортировку атомов в ячейках, фильтрацию пар атомов по расстояниям и по наличию/отсутствию химической связи между ними. Векторизация сделана с помощью SSE и SSE2 intricics.

Параллелизация пакета была сделана с помощью библиотеки BOOST threads. При чтении данных использовалась техника декомпозиции по индексам, то есть каждый процесс выполнял расчеты только с назначенным ему диапазоном рядов Верлет-таблицы. При записи промежуточных результатов расчета сил и избегания конфликтов памяти использовалась техника дублирования данных. В этом случае каждый поток записывал результаты в свой массив данных и в конце шага молекулярной динамики все эти массивы объединялись. В условиях небольшого числа потоков накладные расходы для данной техники находились в пределах 2 %.

Результаты и обсуждение. На рисунке представлена в логарифмическом масштабе диаграмма эффективности выполнения программы MOLKERN в зависимости от числа используемых потоков для разных процессоров. Эффективность измеряется в миллиардах пар (10^9), обрабатываемых за одну секунду. Результаты представлены для процессоров PowerXCell8i (Cell8i), Intel Quad (Q8400) и Intel Xeon (X7560). На всех процессорах выполнялся алгоритмически единый код, но оптимизированный под ту или иную платформу.

Ранее, в [16], мы сообщали результаты по оптимизации данного кода под Cell-платформу. В [16] была реализована PPE-центричная модель вычислений и максимальное ускорение оптимизированного PPE + SPE кода относительно выполнения неоптимизированного кода на одном PPE составляло 26 раз при расчетах с потенциалом $\mathrm{erfc}(\sqrt{\pi} * r / r_{\mathrm{cut}}) / r$. Данное ускорение достигалось при использовании 8 SPE для счета и 2 PPE для загрузки данных. Масштабируемость приложения была низкой, и производительность не увеличивалась даже при использовании всех 16 SPE. Основная причина была в том, что PPE были перегружены и не успевали подготавливать данные для всех SPE. В [17] была реализована SPE-центричная модель. Максимальное ускорение для расчетов с тем же потенциалом $\mathrm{erfc}(\sqrt{\pi} * r / r_{\mathrm{cut}}) / r$ было достигнуто при использовании всех 16 SPE и составляло 218 раз. При использовании стандартного кулона $1/r$ максимальное ускорение равнялось 63, что весьма близко к результатам других авторов. Например, в реализации CHARMM27 для Cell [2] максимальное достигнутое ускорение расчётов при использовании 8 SPE относительно одного PPE составило 35 раз, что соответствует до 70 раз при использовании сервера с 16 SPE. Для программы NAMD в версии для Cell [3] получено ускорение в 61 раз.

Из диаграммы видно, что масштабируемость кода для Cell-реализации, вплоть до 8 используемых SPE, является идеальной. И только при использовании 16 SPE наблюдаются потери в пределах ~15 % от идеального значения, что обусловлено перегрузкой процессорной шины EIB.

Для Intel-процессоров получается существенно большая производительность при выполнении кода при малом числе потоков. Так, разница в производительности при выполнении кода в одном потоке равна 2,75 раза в пользу процессора Intel. Однако рост производительности при увеличении числа потоков заметно меньше, чем для Cell-процессора, и у кривой существует эффект насыщения. Так, при выполнении кода на двух SPE вместо одного для Cell-процессора рост производительности приложения был равен 1.997, а для Intel процессора при переходе от использования одного ядра к двум ядрам рост составляет всего 1,82 раза. На диаграмме это приводит к разному наклону кривых, прекращению роста производительности при большом числе потоков и сближению эффективности процессоров Cell и Xeon при увеличении числа ядер. Так, разница в производительности счета при использовании 16 потоков равна всего 28 %. Можно предположить, что если бы существовал Cell-процессор с 32 ядрами, то он мог обойти Intel-процессор по производительности. В чем могут быть причины различной масштабируемости выполнения задачи для разных процессоров?

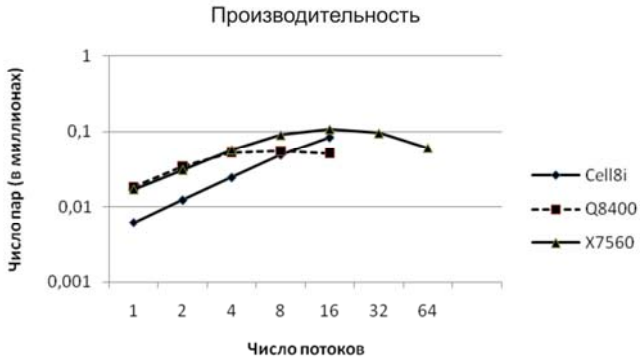

Рис. Диаграмма зависимости эффективности счета от числа потоков для разных процессоров

Возможной причиной низкой масштабируемости приложения при выполнении его на Intel-процессоре может являться специфика алгоритма. В алгоритме для хранения пар соседей

используется Верлет-таблица, объем которой составляет $(4\pi\rho/6)$ R_{cut}^3 N элементов, где ρ – плотность частиц в системе, R_{cut} – радиус обрезания потенциала, N – число атомов в системе. Используя параметры теста, указанные выше, можно оценить объем памяти Верлет-таблицы, который составляет величину ~47 Мб. Таким образом, данная таблица не может целиком поместиться в L3 кэш, объем которого вдвое меньше. Поскольку каждое ядро процессора в многопоточном выполнении работает со своей частью Верлет-таблицы и кроме идентификаторов атомов, хранимых в ней, для расчетов необходимо подгружать множество других данных, то ядра начинают «конфликтовать» друг с другом за то, чьи данные должны быть помещены в L3 кэш. Это неизбежно должно приводить к многочисленным промахам кэша и низкой масштабируемости приложения.

Заметим, что в Cell-процессоре ситуация иная и управление загрузкой LS (аналог кэша) целиком возлагается на программиста. В SPE-центричной модели каждое SPE извлекает данные из оперативной памяти независимо друг от друга. И, как показывает наш опыт, это позволяет добиться практически идеальной масштабируемости.

Работа выполнена при поддержке грантов Междисциплинарных интеграционных проектов фундаментальных исследований СО РАН № 26, № 113 и № 119, а также частично Министерства образования и науки РФ (Госконтракт № П857).

Список литературы

1. Alder B. J., Wainwright T. E. J. Chem. Phys. 1957, 27, 1208.

2. Fabritiis G. D. Comput. Phys. Comm. 2007, 176, No. 11/12. P. 660–664.

3. Shi G., Kindratenko V. 9th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC 2008).

4. Anderson J. A., Lorenz C. D., Travesset A. J. Comput. Science. 2008, 227, 5342.

5. Liua W., Schmidt B., Vossa G., Müller-Wittig W. Comp. Phys. Comm. 2008, 179, No. 9, 634–641.

6. http://www.t-platforms.ru/

7. http://software.intel.com/en-us/articles/intel-many-core-testing-lab/

8. E. S. Fomin, N. A. Alemasov, A. S. Chirtsov and A. E. Fomin. Biophysics. 2006, 51, No. 7, 110–113.

9. J.W. Ponder and D.A. Case. Adv. Prot. Chem. 2003, 66, 27–85.

10. Yao, Z.; Wang, J.-S.; Liu, G.-R.; Cheng, M. Comp. Phys. Comm. 2004, 161, 27–35.

11. Gonnet, P.J. Comp. Chem. 2007, 28(2), 570–573.

12. Meloni S. and Rosati M. J. Chem. Phys., 2007, 126, 121102.

13. L. Verlet, Phys. Rev. 159, 98 (1967); Phys. Rev. 165, 201 (1967).

14. Kahle J.A., Day M.N., Hofstee H.P., Johns C.R., Maeurer T.R., Shippy D. // IBM Journal of Research and Development. – 2005. – Vol. 49, No. 4/5. – P. 589–604.

15. Programmer's Guide to the IBM SDK for Multicore Acceleration. Vol. 3,0. IBM. 2009.

16. Fomin E., Alemasov N. Parallel computing technologies 2009 / Malyshkin V; Springer. LNCS, Vol. 5698. 2009. P. 399–405.

Д.А. Чарнцев, А.Н. Ефремов

ОАО «НПО “Искра”», г. Пермь

ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА КЛАСТЕРНЫХ СИСТЕМАХ ПРИ ИССЛЕДОВАНИИ ГАЗОДИНАМИЧЕСКИХ ХАРАКТЕРИСТИК ШУМОТЕПЛОЗАЩИТНОГО КОЖУХА ГАЗОТУРБИННОЙ УСТАНОВКИ

В последнее время большую актуальность приобрели численные трехмерные газодинамические исследования проточных трактов газоперекачивающих агрегатов. Одним из наиболее сложных с точки зрения течения воздуха и распределения газодинамических характеристик является проточный тракт системы охлаждения газотурбинной установки (ГТУ), а именно про-