Параллельная реализация разностных схем решения уравнения переноса на процессорах Cell

Д.Н. Микушин

Уравнение переноса имеет большое число приложений в численном моделировании изменений климата, прогнозе погоды и оценке влияния антропогенных загрязнителей на окружающую среду. Метод потоковой параллельной обработки включает явное управление асинхронной загрузкой данных с автоматической настройкой для шаблона конкретной разностной схемы. Измерения на тестовых задачах различных размерностей показывают увеличение быстродействия до 2.5 раз для 6-8 SPE процессора Cell по сравнению с OpenMP-версией, использующей 2 четырёхядерных процессора Intel Xeon E5472. Результаты данного исследования могут быть использованы при разработке приложений для Cell-узлов кластера "Ломоносов"Московского Государственного Универсситета.

1. Введение

Моделирование переноса примеси под действием конвективных процессов в атмосфере и гидрологической неоднородности подстилающей поверхности (бризовые циркуляции) на пространственном масштабе 100-1000 км производится с помощью региональных моделей. Негидростатическая региональная модель NH3D [2], развиваемая в НИВЦ МГУ, масштабируется линейно вплоть до нескольких сотен ядер (сетка $193 \times 193 \times 21$) [1], региональная модель WRF версии 3.0 — до нескольких тысяч ядер ($425 \times 300 \times 35$) [3]. Тем не менее, эти результаты показывают, что модели всё ещё недостаточно пригодны для петафлопных вычислений. С вычислительной точки зрения, производительность многих разностных схем ограничена пропускной способностью оперативной памяти и обменной сети многопроцессорной системы. Как правило, это не является препятствием для хорошей масштабируемости явных схем, однако эффективность вычислений остаётся низкой из-за большого потока данных и кеш-промахов. К примеру, 512 узлами кластера Endeavor с Intel Xeon X5560 (Nehalem) (пиковая производительность 4 ядер — 85,8 GFLOPS на тесте LINPACK) для эксперимента 12KM CONUS модели WRF можно получить 700 эффективных GFLOPS, что составляет 6-12% пика в зависимости от точности операций. Методы повышения эффективности сочетают изменения структуры данных в памяти с использованием особенностей конкретной вычислительной системы [4]. В данном исследовании предпочтение отдано структурно-независимым оптимизациям, что делает возможным использование существующего программного кода без значительных изменений. В качестве целевой архитектуры выбрана Cell Broadband Engine.

2. Архитектура Cell Broadband Engine

Cell Broadband Engine (Cell B.E.) — это архитектура однокристальных мультипроцессоров, разработанная консорциумом STI (Sony Computer Entertainment, Toshba Corporation, IBM) [5]. В процессоре Cell высокоскоростная коммуникационная сеть объединяет Power-совместимое ядро общего назначения (PPE, Power Processing Element) и несколько потоковых сопроцессоров (SPE, Synergetic Processing Elements) с поддержкой векторных вычислений (рис. 1). Совместимость с архитектурой Power позволяет устанавливать на некоторые модели устройств с процессорами Cell полноценные операционные системы. Основная масса устройств на базе архитектуры Cell – это высокопроизводительные сервера IBM и игровые приставки Sony Playstation 3, которые оснащаются вариантами процессора с 1 PPE и 7-8 SPE. Пиковая производительность использованного в расчётах сервера IBM QS22 составляет 204,8 GFLOPS в режиме двойной точности.

Ряд конструктивных особенностей отличает архитектуру Cell B.E. от GPU и много-

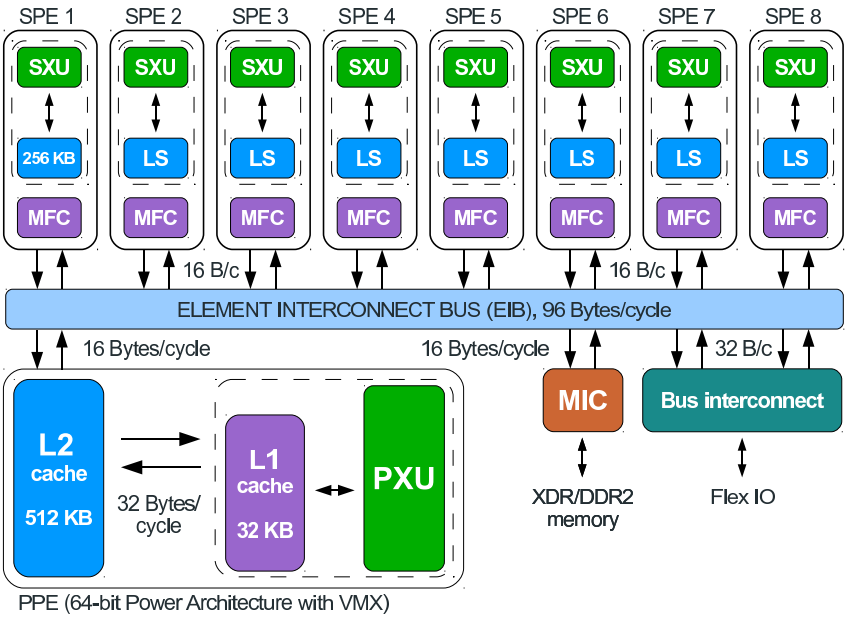

Рис. 1. Архитектура процессора Cell

ядерных процессоров. Каждый SPE обладает собственными 256 Кбайт локальной памяти для команд и данных. По характеристикам локальная память SPE похожа на кэш первого уровня, с той разницей, что обмен данными с основной памятью происходит явно, при помощи управляющих команд Memory Flow Controller (MFC). Малый объём локальной памяти компенсируется быстрой обменной сетью: PPE, SPE и контроллер памяти объединены четырёхканальной шиной, каждый из каналов может передавать 16 байт за 2 такта. В Sony Playstation 3 и серверах QS20/QS21 используется оперативная память XDR (eXtreme Data Rate), пропускная способность составляет 11-13 Гбайт/сек между SPE и памятью на одной шине и 6-8 Гбайт/сек — через северный мост [6]. В сервере QS22 используется память DDR2 (Double Data Rate 2), обеспечивающая на 10-15% меньшую скорость передачи данных. Наилучшие результаты достигаются при заполнении всех DIMM-слотов и эффективном использовании виртуальной страничной памяти. Пиковая скорость передачи данных напрямую между SPE (без использования оперативной памяти) составляет 204.8 Гб/сек [5].

Архитектура Cell B.E. реализует параллелизм по данным, командам, потокам, а также асинхронность вычислений и операций с памятью [7]. Возможность одновременно производить одинаковые вычисления над несколькими элементами компактной последовательности данных (Single Instruction Multiple Data, SIMD) обеспечивается специальным набором векторных команд, для PPE — VMX/Altivec, для SPE — SPU Assembly Language. Размер векторного регистра равен 128 бит. На PPE доступно два потока, по одному потоку может исполняться на каждом SPE. Под управлением специальным образом модифицированной операционной системы множеством SPE-потоков может быть представлена система Cell-устройств с неоднородным доступом к памяти. К примеру, на сервере QS22, состоящем из двух отдельных Cell плат по 8 SPE доступно 16 SPE-потоков. Ещё одной важной особенностью SPE является распределение команд обмена данными и вычислений между двумя конвейерами исполнения (pipelines). На каждом такте SPE может одновременно выполнять операцию над вещественными или целочисленными данными и любую другую независимую команду [5]. Основной метод эффективного программирования потоковых вычислений на Cell, учитывающий данное свойство называется двойной (множественной) буферизацией (double buffering) или циклической очередью (dual circular queue). По сути он похож на операцию $prefetch$: опережающая загрузка в локальную память SPE очередной порции данных (или нескольких порций) происходит асинхронно во время обработки текущей порции.

2.1. Ограничения

- Латентность операций обмена данными для SPE в десятки раз выше, чем у процессоров общего назначения. На аппаратном уровне сообщение любой длины передаётся порциями по 128 байт. Эти особенности не позволяют вести эффективный обмен короткими сообщениями.
- В SPE отсутствует аппаратная реализация деления (программное деление $divd2$ требует около 72 операций), операций над 8-байтовыми целыми и предсказателя ветвлений.
- Некоторые модели процессора оптимизированны только для вещественных операций с одинарной точностью.

3. Прикладная задача

Одной из простейших схем численного решения уравнения переноса является "чехарда"по времени и аппроксимация производных по пространственным переменным центральными разностями со вторым порядком точности 1. В вычислительном отношении схема приводит к 32 операциям умножения и сложения и одному делению (которое нельзя заменить умножением на обратное) для расчёта значения в одном узле сетки. При этом используется 18 значений из 10 массивов.

$$\begin{aligned} -\frac{1}{2\Delta t}([qp]^{n+1}_{i,j,k} - [qp]^{n-1}_{i,j,k}) =& L_x([uqp]^n_{i,j,k}) + L_y([vqp]^n_{i,j,k}) + L_\sigma([\sigma qp]^n_{i,j,k}), \\ L_x(f_{i,j,k}) =& \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2\Delta x}, \\ L_y(f_{i,j,k}) =& \frac{f_{i,j+1,k} - f_{i,j-1,k}}{2\Delta y}, \\ L_\sigma(f_{i,j,k}) =& \frac{f_{i,j,k+1} - f_{i,j,k-1}}{\sigma_{k+1} - \sigma_{k-1}}\ . \end{aligned} \tag{1}$$

4. Реализация

С появлением ускорителей начинает формироваться методология адаптации существующего программного кода к новым архитектурным решениям. Минимизировать влияние инерции и отсутствия специальных знаний у пользователей и разработчиков позволяет подход, при котором необходимые преобразования ведутся на уровне препроцессора и скриптов трансляции, сохраняя оригинальный вид исходного кода. В этом смысле реализация разностных схем для Cell Broadband Engine имеет много общего с версией блока микрофизики WSM5 модели WRF для GPU [8].

При использовании двойной буферизации производительность SPE существенно зависит от баланса скорости вычислений и обмена данными. Для достижения большей эффективности оптимизировать необходимо более медленную составляющую. В задачах с большим потоком данных производительность как правило ограничена пропускной способностью общей памяти, если только требуемые данные не удастся получить из локальной памяти другого SPE, что происходит на порядок быстрее.

4.1. Декомпозиция области

Декомпозиция области в реализациях для многопроцессорных систем с распределённой памятью как правило основана на выделении каждому узлу двумерной или трёхмерной подобласти расчётой сетки. Каждый узел использует собственную локальную память и синхронизирует граничные значения с другими узлами. Данный подход предполагает, что

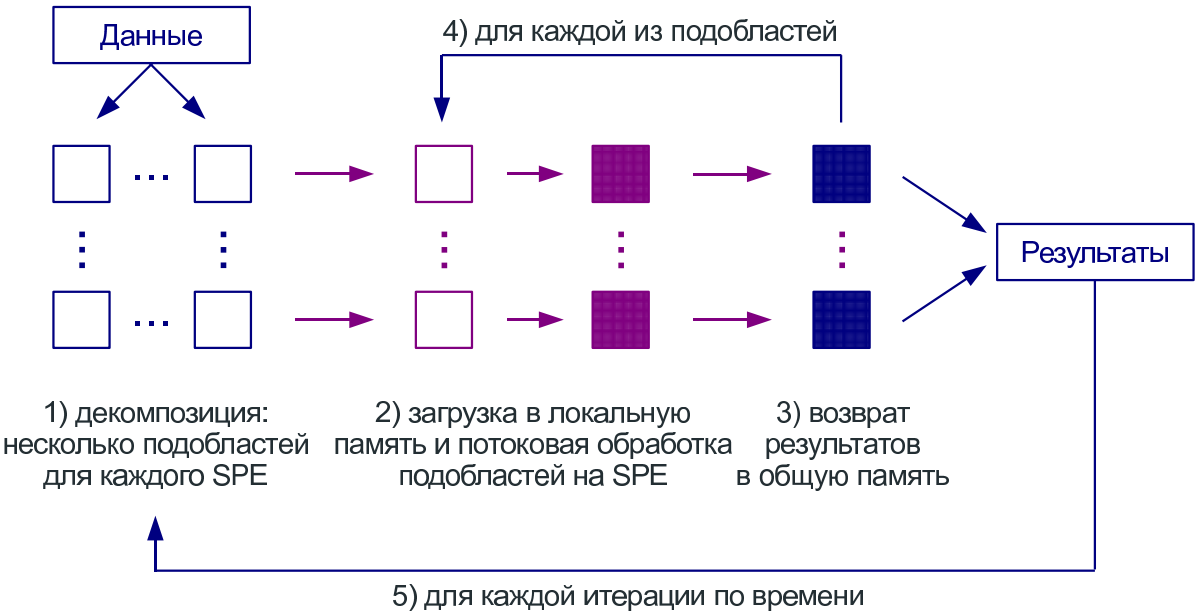

Рис. 2. Схема организации вычислений на SPE

локальной памяти узла всегда достаточно для хранения данных подобласти. Но если собственной памяти узлов не хватает для размещения данных задачи, как в случае Cell, данный алгоритм декомпозиции неприменим. С каждым вычислительным элементом приходится ассоциировать несколько подобластей, так чтобы одна подобласть целиком помещалась в локальную память (Рис. 2). Необходимость последовательной обработки нескольких подобластей, извлекаемых из внешней памяти приводит к тому, что синхронизация граничных значений на самих вычислительных узлах значительно усложняется, и по сути становится избыточной, поскольку ни одна подобласть не хранится в памяти узла постоянно.

Одна из моделей параллельного исполнения Cell Broadband Engine предполагает, что ядро общего назначения (PPE) управляет распределением нагрузки на вычислительные ядра (SPE). Следуя этой модели, единственную синхронизацию для каждого временного шага достаточно организовать на стороне PPE, статически распределив нагрузку по потоковой обработке подобластей между доступными SPE.

4.2. Векторизация

В параллельной реализации численной схемы решения уравнения переноса для архитектуры Cell за основу был взят скалярный код на языке C и проведён анализ того, какое минимальное множество изменений должно быть внесено для получения векторной версии для SPE. Оказалось, что весь вычислительный цикл может быть использован в оригинальном виде с добавлением ключевого слова *vector* в объявления переменных. В явном использовании векторных команд, функций библиотек simdmath или MASS нет необходимости, все операции для векторных переменных работают точно так же, как для скаляров, за исключением деления, вместо которого был вставлен вызов *divd*2.

Векторные операции доступны только для адресов, кратных 16 (выровненных), поэтому наличие в шаблоне левого и правого элемента вносит небольшое ограничение. Поскольку они имеют нечётный сдвиг, то если предположить, что адрес центрального элемента выровнен, то адреса соседних элементов по оси X невыровнены, когда как все остальные элементы шаблона будут иметь выровненные адреса при чётной размерности сетки по оси X (Рис. 3). По этой причине перед тем как использовать векторные операции с участием невыровненных элементов, требуется их выровнить, скопировав в промежуточную переменную, неявно

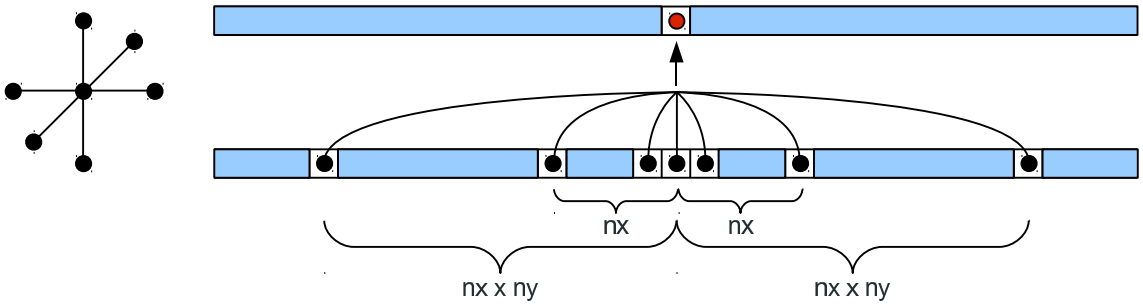

Рис. 3. Операции в памяти для 7-точечного шаблона разностной схемы

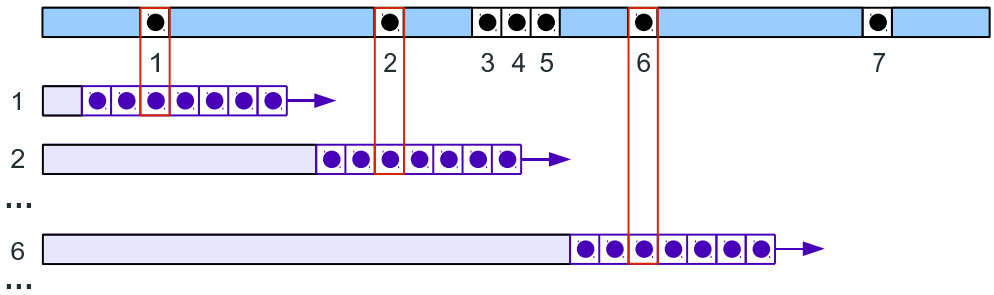

Рис. 4. Одномерная потоковая загрузка данных на SPE

выполняя дополнительные операции сдвига. Для этого достаточно использовать препроцессор, изменений в код вычислений не требуется.

4.3. Обмен данными

Многомерная декомпозиция области требует соответствующей блочной структуры хранения данных, в противном случае, загрузки непрерывных сегментов памяти, соответствующих данным одной подобласти, становятся сильно фрагментированными, что приводит к дополнительным накладным расходам. Поскольку данная реализация, по условию, не должна менять структуру данных, от многомерной декомпозиции было решено отказаться в пользу представления в виде одномерного вектора. Относительно центрального элемента на SPE задаются одномерные смещения для вычисления адресов остальных элементов шаблона (Рис. 3). Использование других позиций в шаблоне требует комбинации нескольких смещений, поддержка которых так же предусмотренна в программном коде. Для данного абсолютного адреса центрального элемента в общей памяти смещениями определяются абсолютные адреса других элементов шаблона, которые выравниваются с отступом влево и используются для загрузки данных в локальную память. За одну операцию последовательно загружается сегмент (порция) данных (Рис. 4), пересекающиеся загрузки заменяются одной.

Описанный алгоритм потоковой загрузки не привязан к конкретной разностной схеме и не требует перенастройки при её изменении. На каждом шаге по времени проводится одна фиктивная итерация вычислительного цикла, по которой определяются смещения и порядок использования данных.

Управление буферизацией данных отделено от вычислений. Для данной схемы загрузка новой порции и выгрузка результатов должна производиться раз в 512 итераций — такое ограничение вносит малый размер локальной памяти. По этой причине диапазоны индексов вычислительного цикла соответствующим образом уменьшены, а сам цикл погружен во внешний, контролирующий буферизацию данных.

Таблица 1. Производительность реализаций для 1 и 8 ядер (4 + 4) Intel Xeon E5472 и IBM QS22 на тестовых задачах различных размерностей: $N = D^2 \times 32$ — число узлов, m — среднее значение скорости обработки данных (Гбайт/сек), t — время расчёта (сек), f – количество эффективных операций в секунду (GFlops).

D	$m_{xeon(1)}$	$m_{xeon(8)}$	m_{cell}	$t_{xeon(1)}$	$t_{xeon(8)}$	t_{cell}	$f_{xeon(1)}$	$f_{xeon(8)}$	f_{cell}	$\frac{t_{xeon(8)}}{t_{cell}}$
256	2.75	5.76	9.49	0.102	0.048	0.029	0.63	1.32	2.18	1.65
512	2.38	5.62	11.52	0.472	0.200	0.097	0.55	1.29	2.64	2.05
1024	2.39	6.26	16.05	1.882	0.718	0.280	0.55	1.44	3.68	2.56

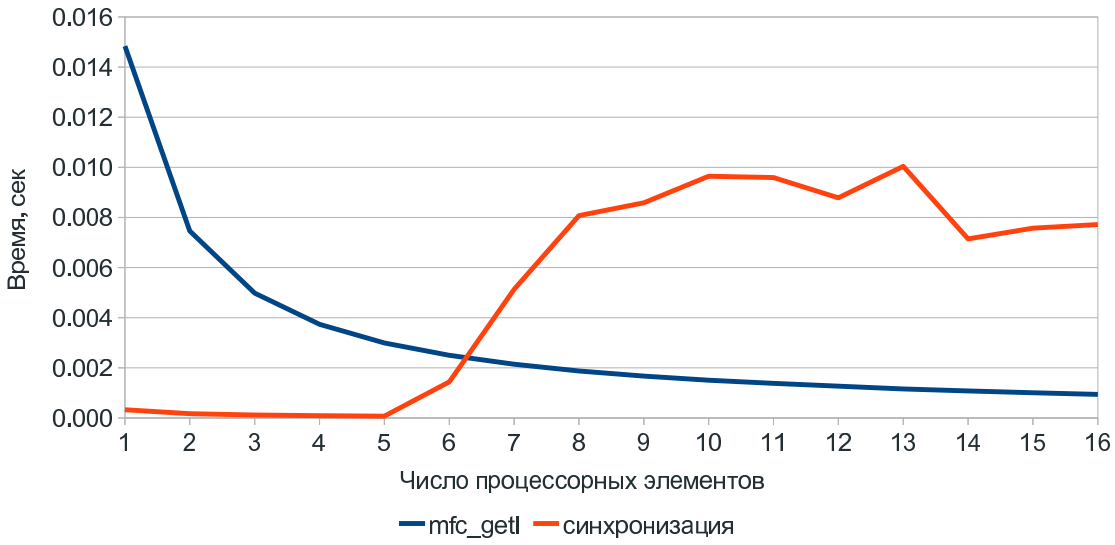

Рис. 5. Задержки для тестового расчёта $256 \times 256 \times 32$ на сервере IBM QS22

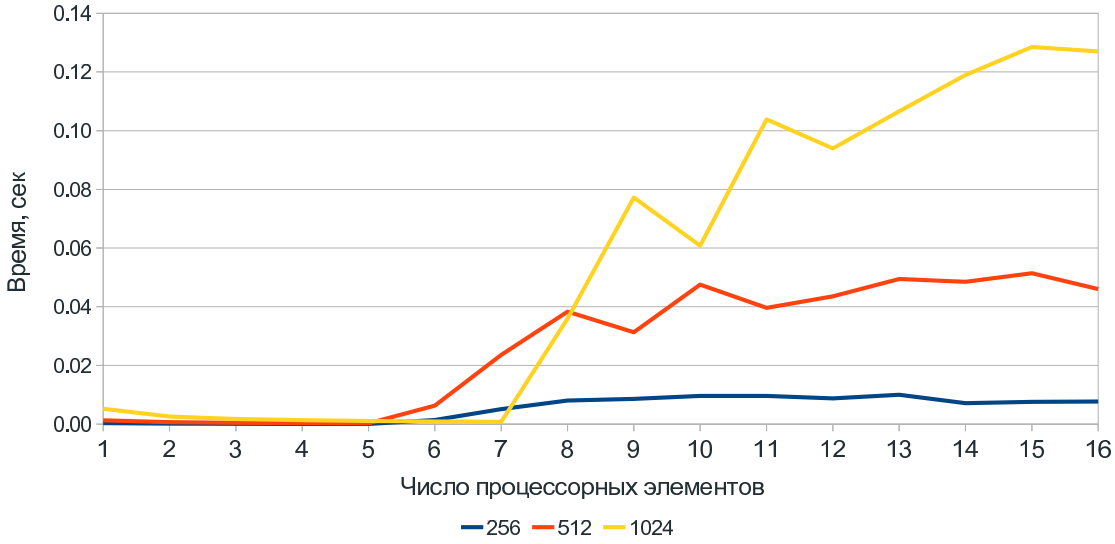

Рис. 6. Времена синхронизаци для тестовых расчётов $256 \times 256 \times 32$, $512 \times 512 \times 32$, $1024 \times 1024 \times 32$ на сервере IBM QS22

Таблица 2. Минимальные и максимальне времена расчётов на отдельных SPE сервера QS22 при использовании различного числа SPE. Общее время работы теста соответствует максимальному времени SPE.

N_{SPE}	t_{min}	t_{max}	N_{SPE}	t_{min}	t_{max}
1	1.86	1.86	9	0.27	0.31
2	0.93	0.93	10	0.21	0.27
3	0.62	0.62	11	0.26	0.30
4	0.46	0.46	12	0.23	0.27
5	0.37	0.38	13	0.22	0.27
6	0.31	0.31	14	0.23	0.28
7	0.27	0.28	15	0.25	0.28
8	0.27	0.29	16	0.25	0.27

4.4. Производительность

Полученная реализация была протестированна на сервере IBM QS22 с процессорами PowerXCell8i, 4 Гб памяти DDR2 667 МГц и компилятором IBM XL C/C++ 10.1. В Табл. 1 приведены основные показатели для QS22 и двухпроцессорного узла с четырёхядерными Intel Xeon E5472. На последнем измерялось время работы эквивалентного кода, использующего 1 и 8 ядер (OpenMP). Поскольку производительность ограничена пропускной способностью памяти, количество операций мало.

При больших размерностях задачи дополнительную выгоду даёт использование больших страниц памяти (huge memory pages) — производительность увеличивается на 15-20%. Увеличение количества уровней буферизации не даёт прироста. Длина очереди запросов MFC составляет 16, при превышении этого лимита, исполнение останавливается до тех пор, пока не завершится одна из передач. DMA-списки (Direct memory access lists) позволяют минимизировать этот эффект. Тем не менее, измерения показывают, что время подготовки DMA-запросов (вызов mfc_getl) является значительной долей общего времени при использовании малого количества SPE (Рис. 5). В то же время рост задержек, связанных с синхронизацией (ожиданием завершения текущей передачи данных) увеличивается при росте числа SPE (Рис. 6). Это показывает эффект насыщения: вычисления производятся быстрее, чем асинхронные операции с памятью.

Тесты производительности показывают, что более чем двукратный прирост по сравнению с 8 ядрами двух Intel Xeon E5472 получен при использовании 6-8 ядер QS22 (Рис. 2), т.е. достаточно одного 8-ядерного процессора PowerXCell8i (остальные 8 ядер сервера физически расположены на втором процессоре). Даже если попытаться уменьшить разбалансировку, возрастающую при использовании большего числа SPE, дальнейшее увеличение производительности мало. По этой причине на уровне отдельных процессоров PowerXCell8i большую эффективность может обеспечить MPI-подобная организация вычислений.

5. Заключение

Преимущества текущей Cell-реализации перед версиями для многоядерных процессоров незначительны. Судя по всему, представление расчётной области в виде вектора и простое разбиение на фрагменты является чрезмерным упрощением, которое приводит к неоднократной загрузке на SPE одних и тех же данных. В ходе дальнейшей работы представляется целесообразным реализовать схемы обхода узлов расчётной области, минимизирующие зависимость производительности от пропускной способности памяти за счёт обменов между SPE и повторного использования загруженных данных.

Данная реализация перспективна для решения разностных задач на кластерах, узлами которых являются Cell-сервера. Развитие в этом направлении ограничено отсутствием доступа к действующим системам подобного рода. Практика использования процессоров Cell для численного решения уравнения переноса имеет важное значение при анализе применимости различных процессорных архитектур в задачах атмосферного моделирования.

Автор выражает благодарность компании Т-Платформы за предоставленный доступ к оборудованию и техническую поддержку. Плодотворные обсуждения с А.В. Адинцом способствовали значительному улучшению результатов. Подробные разъяснения по некоторым особенностям архитектуры Cell дали M. Kistler и J. Adamczewski.

По ссылке ftp://geophyslab.srcc.msu.ru/Research/forge_100115031228.tar.gz доступен программный код реализации, описанной в настоящей работе.

Литература

1. В.М. Степаненко, Д.Н. Микушин, С.В. Ткачук. Реализация мезомасштабной атмосферной модели на вычислительных системах с распределённой памятью и её приложения. Труды Всероссийской научной конференции "Научный сервис в сети Интернет с. 121-125, 2009.

2. P.M.A. Miranda and I.N. James. Non-linear three-dimensional effects on gravity wave drag: splitting flow and breaking waves. *Quart. J.R. Met. Soc.*, 118:1057–1082, 1992.

3. WRF V3 Parallel Benchmark Page. http://www.mmm.ucar.edu/wrf/WG2/bench/

4. Kaushik Datta, Mark Murphy, Vasily Volkov, Samuel Williams, Jonathan Carter, Leonid Oliker, David Patterson, John Shalf and Katherine Yelick. Stencil computation optimization and auto-tuning on state-of-art multicore architectures. *International Conference for High Performance Computing, Networking, Storage and Analysis* (SC2008), November 2008, Austin, Texas.

5. Michael Kistler and Michael Perrone and Fabrizio Petrini. Cell Multiprocessor Communication Network: Built for Speed. IEEE Micro, Vol. 26, pp. 10–23, 2006.

6. P. Altevogt and H. Boettiger and T. Kiss and Z. Krnjajic. Evaluating IBM BladeCenter QS21 hardware performance. IBM Multicore Acceleration Technical Library, 2008, http://www.ibm.com/developerworks/library/pa-qs21perf/index.html

7. Michael Gschwind. The Cell Broadband Engine: Exploiting Multiple Levels of Parallelism in a Chip Multiprocessor. International Journal of Parallel Programming, Vol. 35, pp. 233–262, 2007.

8. John Michalakes and Manish Vachharajani. GPU Acceleration of Numerical Weather Prediction. 22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008.