Решение сеточных уравнений на графических вычислительных устройствах. Метод пирамид.

А. В. Кочуров
Самарский государственный аэрокосмический университет, Самара
e-mail: ipris@inbox.ru

Д. Л. Головашкин
Институт систем обработки изображений РАН, Самара

Целью авторов было решение сеточных уравнений на GPU для сеток, превышающих размер доступной видеопамяти. Для этого был адаптирован известный метод пирамид. В работе исследованы два варианта декомпозиции сеточной области.

1. Введение

Разностное решение дифференциальных уравнений широко применяется для моделирования разнообразных явлений в физике, химии, экономике и других отраслях.

Повышение производительности центральных процессоров (CPU) достигается сегодня путем увеличения числа ядер при малом росте частоты. Однако количество ядер у процессоров x86 из-за недостатков архитектуры сравнительно невелико (достигает 16).

Архитектура графических процессоров (GPU) изначально разрабатывалась для параллельных вычислений. Для сравнения, один из наиболее производительных CPU Intel Core i7-975 обладает пиковой производительностью 55 Гфлопс при рассчетном тепловыделении (TDP) в 130 Ватт [1], в то время как GPU NVIDIA Tesla C2050 с 448 ядрами обладает пиковой производительностью в 1 Тфлопс при TDP 238 Ватт [2].

Одним из основных ограничений известных методов решения сеточных уравнений на GPU является требование, чтобы сеточная область целиком помещалась в графической памяти. При этом объемы видеопамяти сегодня куда скромнее, чем объем ОЗУ. Так, GeForce 8800 GTX с 768 Мб видеопамяти позволяет применять метод FDTD к с сеточным областям размером до $220 \times 220 \times 220$ узлов [3], а NVidia Tesla C2050 с 6 Гб памяти – до $550 \times 550 \times 550$ узлов.

На практике, например, при проектировании оптических элементов с линейными размерами в несколько микрон и нанометровыми неоднородностями, требуемый размер сеточной области составляет десятки тысяч узлов [4]. Всвязи с этим существует потребность в методах решения сеточных уравнений, не требующих размещения всей сеточной области в видеопамяти.

В настоящей работе предлагается подход к решению это задачи, основанный на применении метода пирамид, который ранее применялся для автоматического распараллеливания циклических фрагментов последовательных программ [5].

2. Метод пирамид в теории разностных схем

Автоматическое распараллеливание циклических фрагментов программ методом пирамид. По Лэмпорту [6], значительная часть времени выполнения большинства программ тратится на один или несколько одно- или многомерных циклов, поэтому параллельное исполнение итераций этих циклов позволяет значительно ускорить вычисления.

Векторы, координаты которых представляют собой параметры итераций цикла, образуют пространство итераций. Два вектора будем называть зависимыми, если между соответствующими им итерациями существует зависимость (информационная или иная). Задача распараллеливания цикла сводится к поиску такого покрытия пространства итераций, что вектора каждого множества независимы друг от друга, и соответствующие им итерации могут выполняться параллельно.

Применение метода пирамид подразумевает следующие действия:

- на пространстве итераций A вводится отношение частичного порядка как транзитивное замыкание $\tilde{\phi}$ введенного ранее отношения зависимости ϕ: $a_1 \phi a_2 \Leftrightarrow a_1$ зависит от a_2;
- из пространства итераций A выбираются множество результирующих векторов $R = \{a \in A | \ \nexists b \in A, b \phi a\}$ (т. е. от которых не зависят никакие другие);
- строится некоторое разбиение R на подмножества $R_i : R_1 \bigcup R_2 \bigcup \ldots \bigcup R_n = R, i \neq j \Leftrightarrow R_i \bigcap R_j = \emptyset$;
- каждое из множеств дополняется множеством итераций, от которых транзитивно зависят его итерации $A_i = R_i \bigcup \{a | a \in A, \exists r \in R, r \tilde{\phi} a\}$;
- каждой задаче назначается одно или несколько полученных частично-упорядоченных по отношению $\tilde{\phi}$ множеств A_i, задача последовательно исполняет итерации из этих множеств в соответствии с заданным порядком.

Существенной особенностью метода пирамид применительно к сеточным уравнениям является возможность выбора такого распределения результирующих итераций по задачам, которое обеспечивает локализацию данных, требуемых для выполнения каждой из задач, в пределах подобластей сеточной области, каждая из которых может быть помещена в видеопамять целиком.

Предлагаемая здесь модификация метода пирамид подразумевает отличный от традиционного подход: поскольку видеопамять каждого устройства ограничена, на одном устройстве невозможно одновременное исполнение нескольких задач. Поэтому задачи выполняются квазипараллельно, по очереди монопольно занимая устройство на время, необходимое для выполнения одного прохода.

Декомпозиция двумерной сеточной области. Иллюстрируем применения метода пирамид к вычислениям на GPU на примере двумерного однородного нестационарного уравнения теплопроводности $\frac{\partial}{\partial t} U = \alpha^2 \Delta U$, где $U(x, y, t), x \in [0, X], y \in [0, X], t \in [0, T]$ – распределение температур в пространстве и времени, α – коэффициент температуропроводности, с краевыми условиями первого рода и начальным условием $U(x, y, 0) = \phi(x, y)$.

Явная разностная схема для этой задачи имеет вид [7]:

$$U_{i,j}^{(k+1)} = \frac{\alpha^2 h_t}{h_x^2}\left(U_{i+1,j}^{(k)} + U_{i-1,j}^{(k)} + U_{i,j-1}^{(k)} + U_{i,j+1}^{(k)} - 4U_{i,j}^{(k)}\right), i \in \overline{0,I}, j \in \overline{0,I}, k \in \overline{0,K-1}$$

Традиционные алгоритмы решения этой задачи обладают существенным ограничением: они требуют наличия всей сеточной области в оперативной памяти. Для обхода этого ограничения можно пользоваться следующим подходом: сеточная область разбивается на подобласти, каждая из которых помещается в видеопамять, эта подобласть обрабатывается стандартным образом, после чего результаты вычислений перемещаются обратно. В этом случае число пересылаемых в видеопамять значений сеточной функции можно оценить как $K(I+2)^2\frac{R+2}{R}$, число пересылаемых из видеопамяти значений – как $K(I+2)^2$, а число вычислений по дифф. шаблону – как $K(I+2)^2$. Как видно, такое решение порождает высокую нагрузку на шину между ЦП и видеокартой и из-за этого может значительно увеличивать длительность работы программы.

Поэтому предлагается использовать следующую модификацию метода пирамид: для вычисления K слоев сеточного уравнения последовательно выполнять s проходов методом пирамид, вычисляя каждый раз n слоев, $ns = K$. Высота пирамид n выбирается из соображений минимизации общей длительности работы программы.

Далее будут рассмотрены два варианта декомпозиции сеточной области по задачам.

Одномерная декомопозиция. При одномерной декомпозиции сеточной области каждая из μ задач вычисляет значения сеточной функции на r строках сеточной области на n слоев вперед ($r\mu = I+1$). Результирующими итерациями для m-го прохода ($m \in \overline{1,s}$) будут все итерации, вычисляющие $m \cdot n$ слой по времени, и только они. Для работы программы требуется область видеопамяти для размещения $R = r + 2n$ строк сеточной области.

В таблице 1 приведена оценка сверху для числа различных операций, требуемых для выполнения одной задачи одного прохода (количество операций будет несколько меньше для задач, обрабатывающих области, прилегающие к краям сеточной области), а также оценка суммарного количества операций для всех проходов и задач.

Операция	Одна задача одного прохода	Всего
Пересылка зн-я ф-ии в видеопамять	$R(I+1)$	$\frac{R}{n(R-2n)}K(I+1)^2$
Вычисление зн-я по шаблону	$n(R-n-1)(I+1)$	$\frac{R-n-1}{R-2n}K(I+1)^2$
Пересылка зн-я из видеопамяти	$(R-2n)(I+1)$	$\frac{1}{n}K(I+1)^2$

Т а б л и ц а 1. Число операций при использовании метода пирамид с одномерной декомпозицией

Сравнивая общее число операций метода пирамид с числом операций тривиального алгоритма, использующего тот же объем памяти, можно говорить о том, что общий объем вычислений засчет применения метода пирамид возрастает в $1 + \frac{n-1}{R-2n}$ раз, однако объем пересылаемых в видеопамять данных уменьшается в $n\frac{R-2n}{R-2}$ раз, а объем получаемых из видеопамяти данных – в n раз.

Дадим оценку длительности работы программы в зависимости от высоты пирамиды n и определим её оптимальное значение. Допустим, что длительности пересылки

и длительность вычислений линейно зависят от количества обрабатываемых данных. Тогда общая длительность вычислений оценивается как

$$\tau_{st}(n) = \left(\tau_f \frac{R}{n(R-2n)} + \tau_b \frac{1}{n} + \tau_c \left(1 + \frac{n-1}{R-2n}\right)\right) K(I+1)^2$$

здесь τ_f, τ_b и τ_c – соответственно длительности пересылки одного значения сеточной функции на устройство, длительности пересылки значения с устройства, и длительность вычисления значения по шаблону. Эти велечины определяются особенностями конкретной вычислительной системы.

Для нахождения оптимальной высоты пирамиды n необходимо решить задачу оптимизации $\tau_{st}(n) \to min$.

Двумерная декомпозиция. При декомпозици сеточной области на квадратные блоки каждая из μ^2 задач вычисляет значения сеточной функции на блоке $b \times b$ узлов на n слоев вперед ($b\mu = I+1$). Результирующими итерациями для m-го прохода ($m \in \overline{1,s}$) будут все итерации, вычисляющие $m \cdot n$ слой по времени, и только они. Для работы программы требуется область памяти для размещения квадратной подобласти сеточной области со стороной $B = b + 2n$.

В таблице 2 приведена оценка числа различных операций, требуемых для выполнения одной задачи одного прохода (количество операций будет несколько меньше для задач, обрабатывающих области, прилегающие к краям сеточной области), а также оценка суммарного количества операций для всех проходов и задач.

Операция	Одна задача одного прохода	Всего
Пересылка зн-я в видеопамять	B^2	$\frac{1}{n}\left(\frac{B}{B-2n}\right)^2 K(I+1)^2$
Вычисление зн-я по шаблону	$n((B-2n)^2 + 2(n-1)(B - \frac{4n+1}{3}))$	$\left(1 + 2(n-1)\frac{B-\frac{4n+1}{3}}{(B-2n)^2}\right) K(I+1)^2$
Пересылка зн-я из видеопамяти	$(B-2n)^2$	$\frac{1}{n} K(I+1)^2$

Т а б л и ц а 2. Число операций при использовании метода пирамид с двумерной декомпозицией

Сравнивая общее число операций метода пирамид с числом операций тривиального алгоритма, использующего тот же объем памяти, можно говорить о том, что общий объем вычислений засчет применения метода пирамид возрастает в $1 + 2(n-1)\frac{B-\frac{4n+1}{3}}{(B-2n)^2}$ раз, однако объем пересылаемых в видеопамять данных уменьшается в $n\frac{B(1-2\frac{n}{B})^2}{B-2}$ раз, а объем получаемых из видеопамяти данных – в n раз.

По аналогии со случаем одномерной декомпозиции, оценка длительности работы программы в зависимости от высоты пирамиды n имеет вид

$$\tau_{sq}(n) = \left(\tau_f \frac{1}{n}\left(\frac{B}{B-2n}\right)^2 + \tau_b \frac{1}{n} + \tau_c \left(1 + 2(n-1)\frac{B - \frac{4n+1}{3}}{(B-2n)^2}\right)\right) K(I+1)^2$$

Для нахождения оптимальной высоты пирамиды n необходимо решить задачу оптимизации $\tau_{sq}(n) \to min$.

Вычислительный эксперимент. Были разработаны параллельные алгоритмы по технологии OpenCL для видеокарты GeForce GT330M, выполняющие решение поставленной задачи с применением обеих декомпозиций.

Исходные данные для теоретических оценок длительности работы (время пересылок и время вычислений одного значения) были рассчитаны на основе данных, полученых с помощью NVIDIA Compute Visual Profiler.

На рисунке 1 изображены теоретический (красный) и экспериментальный (синий) графики длительности выполнения программы на тестовом наборе данных при различной высоте пирамид, при $I = 16384$ и при ширине полосы $R = 1024$ для одномерной декомпозиции. Максимальное расхождение между теоретической оценкой длительности работы и экспериментальными данными составило 13%, среднее отклонение – 4%. Время на графике указано из рассчета на один узел.

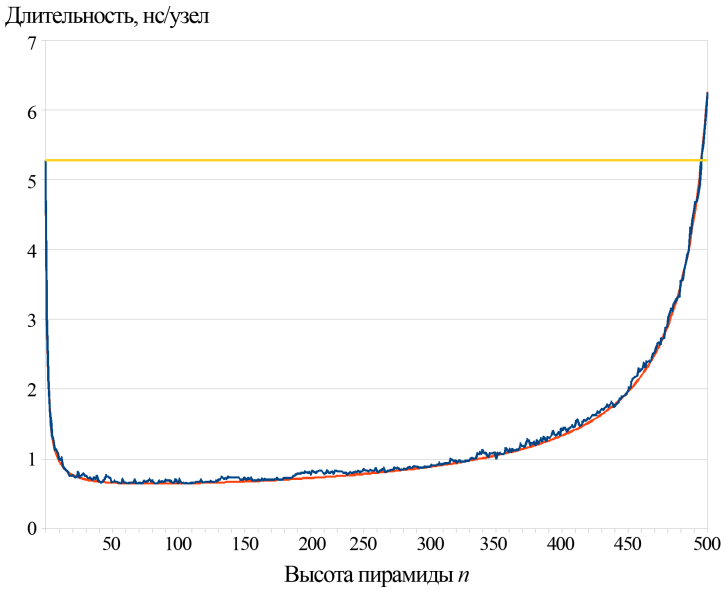

Рис. 1. Длительность работы программы в зависимости от высоты пирамиды при одномерной декомпозиции

На рисунке 2 изображены аналогичные графики для двумерной декомпозиции при размере блока $B = 4096$. Максимальное расхождение между теоретической оценкой и экспериментальными данными составило 17%, среднее отклонение – 6%.

Желтая прямая на обоих графиках обозначает время работы алгоритма с послойными пересылками.

Таким образом, одномерная декомпозиция обеспечивает большую производительность при одинаковом объеме используемой памяти засчет меньшего количества дублирующихся вычислений и пересылаемых данных.

3. Заключение

Разработанный метод позволяет решать сеточные уравнения на областях, размер которых превышает объем доступной видеопамяти, достигая при этом значительно большей производительности, чем известные методы.

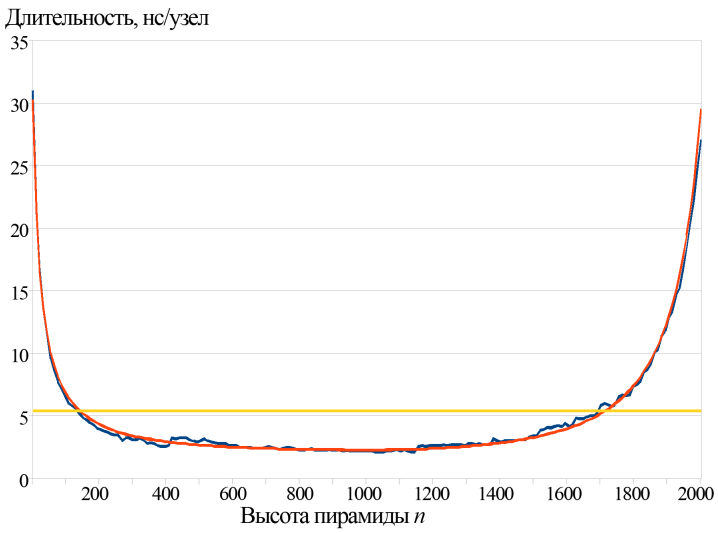

Рис. 2. Длительность работы программы в зависимости от высоты пирамиды при двумерной декомпозиции

Данный метод актуален для решения других дифференциальных уравнений, предполагается его применение для метода FDTD для задач электродинамики.

Список литературы

[1] «Intel microprocessor export compliance metrics» (`http://www.intel.com/support/processors/sb/cs-023143.htm`)

[2] «Tesla Workstation Solutions»(`http://www.nvidia.com/object/product_tesla_C2050_C2070_us.html`)

[3] S. Adams, J. Payne, R. Boppana. Finite Difference Time Domain (FDTD Simulations Using Graphics Processors) // Proceedings DoD High Performance Computing Modernization Program Users Group Conference, 2007. P. 334-338

[4] V. S. Pavelyev, S. V. Karpeev, P. N. Dyachenko, Y. V. Miklyaev. Fabrication of three-dimensional photonics crystals by interference lithography with low light absorption // Journal of Modern Optics. 2009. Vol. 56, № 9. P. 1133-1136.

[5] В. А. Вальковский. Параллельное выполнение циклов. Метод пирамид // Кибернетика. 1983. № 5. С. 51-55.

[6] Lamport L. The coordinate method for the parallel execution of DO-loops // Sagamore computer conference on parallel processing, 1973.

[7] А. Н. Тихонов, А. А. Самарский. Уравнения математической физики (5-е изд.). М.: Наука, 1977.