Решение эллиптического дифференциального уравнения в частных производных на графическом процессоре в технологии CUDA

Н. О. Матвеева

Рассматривается возможность использования графических процессоров для решения дифференциальных уравнений в частных производных. Предлагается алгоритм распараллеливания метода конечных разностей для решения дифференциальных уравнений эллиптического типа в частных производных с учетом архитектуры многоядерных графических процессоров nVidia и технологии CUDA. Проводится сравнение времени вычислений на центральном процессоре и видеокарте.

1. Введение

В настоящее время широко применяется математическое моделирование различных физических процессов. При математическом моделировании часто физические процессы описываются дифференциальными уравнениями в частных производных (ДУЧП). Одной из важных проблем, возникающих при решении дифференциальных уравнений, является большие вычислительные затраты, а следовательно длительное время вычислений. Но с другой стороны многим методам решения ДУЧП присущ внутренний параллелизм вычислений, что дает основания использовать параллельные системы для их решения.

Уже в течение нескольких лет для ускорения хорошо распараллеливаемых вычислений использовались графические процессоры (GPU), только с появлением нового поколения GPU с многоядерной архитектурой это направление стало давать ощутимые результаты.

Целью данной работы является разработка параллельной реализации метода конечных разностей для решения эллиптических дифференциальных уравнений в частных производных на графическом процессоре с использованием технологии CUDA и исследование эффективности распараллеливания путем сравнения времени решения ДУЧП на GPU и на центральном процессоре.

Рассматривалось двумерное уравнение следующего вида

$$\frac{\partial}{\partial x}\left(\sigma(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\sigma(x,y)\frac{\partial u}{\partial y}\right)=f(x,y),\quad (x,y)\in S, \tag{1}$$

с граничными условиями

$$u\big|_{\Gamma}=\varphi(x,y),\quad (x,y)\in\Gamma, \tag{2}$$

где Γ – некоторый прямоугольник.

Решение искалось в узлах равномерной сетки с постоянным шагом h. Использовался пятиточечный шаблон и итерационный метод Зейделя.

2. Описание метода конечных разностей для решения дифференциальных уравнений эллиптического типа

Для решения уравнения (1) с граничными условиями (2), строится равномерная сетка на плоскости *XOY*. Вместо функций $u(x,y)$ непрерывных аргументов $x,y\in S$ рассматриваются сеточные функции $u_h\left(x_i,y_j\right)$ узлов сетки. Если дан линейный дифференциальный оператор L, действующий на функцию u, то, заменяя входящие в Lu производные разностными отношениями, получим разностное выражение $L_h u_h$, являющееся линейной комбинацией значений

сеточной функции u_h на некотором множестве узлов сетки, называемом шаблоном. То есть шаблон задает множество узловых точек, входящих в разностное выражение.

Используя интегро-интерполяционный метод [1] и пятиточечный шаблон, запишем разностную аппроксимацию уравнения (1) для внутреннего узла

$$-\left[\frac{\sigma(x_i+h/2,y_j)}{h^2}+\frac{\sigma(x_i-h/2,y_j)}{h^2}+\frac{\sigma(x_i,y_j+h/2)}{h^2}+\frac{\sigma(x_i,y_j-h/2)}{h^2}\right]u_{i,j}+$$
$$+\frac{\sigma(x_i+h/2,y_j)}{h^2}u_{i+1,j}+\frac{\sigma(x_i-h/2,y_j)}{h^2}u_{i-1,j}+\frac{\sigma(x_i,y_j+h/2)}{h^2}u_{i,j+1}+$$
$$+\frac{\sigma(x_i,y_j-h/2)}{h^2}u_{i,j-1}=f(x_i,y_j),\quad (x_i,y_i)\in S_h$$

или

$$a_{i,j}u_{i,j}-a_{i+1,j}u_{i+1,j}-a_{i-1,j}u_{i-1,j}-a_{i,j+1}u_{i,j+1}-a_{i,j-1}u_{i,j-1}=-f_{i,j}, \qquad (3)$$

где

$$a_{i,j}=\frac{\sigma(x_i+h/2,y_j)}{h^2}+\frac{\sigma(x_i-h/2,y_j)}{h^2}+\frac{\sigma(x_i,y_j+h/2)}{h^2}+\frac{\sigma(x_i,y_j-h/2)}{h^2},$$

$$a_{i+1,j}=\frac{\sigma(x_i+h/2,y_j)}{h^2},\qquad a_{i-1,j}=\frac{\sigma(x_i-h/2,y_j)}{h^2},\qquad a_{i,j+1}=\frac{\sigma(x_i,y_j+h/2)}{h^2},$$
$$a_{i,j-1}=\frac{\sigma(x_i,y_j-h/2)}{h^2},\qquad f_{i,j}=f(x_i,y_j),\quad (x_i,y_i)\in S_h.$$

Записав данные уравнения для каждого узла, в котором неизвестна сеточная функция, учитывая граничные условия и перенеся все известные члены в правую часть, получим разреженную систему алгебраических разностных уравнений. Полученная система решается одним из методов решения систем алгебраических уравнений. При использовании итерационных методов решения систем разностных уравнений нет необходимости применять общие методы хранения разреженных матриц. Для систем вида (3) хранятся массивы коэффициентов конечно-разностной аппроксимации. В случае регулярного шаблона для двумерного уравнения можно хранить два массива коэффициентов A_x и A_y, связывающих узлы в шаблоне, соответственно, по осям *X* и *Y*.

В данном случае используется итерационный метод Зейделя для решения систем разностных уравнений.

3. Анализ средств программирования GPU

В качестве средства для программирования графического процессора была выбрана технология nVidia CUDA (Compute Unified Device Architecture) [2,3]. Первые возможности программирования графических процессоров (GPU – Graphical Processor Unit) появились в 2001 году в GPU второго поколения [4]. Программы, выполняемые на GPU, стали называться "шейдерами". Все это время основным средством программирования GPU оставались интерфейсы трехмерной графики DirectX и OpenGL, а также шейдерные языки HLSL, GLSL и Cg. Использование этих средств для программирования неграфических задач создает серьезные проблемы для прикладных программистов. Средства являются низкоуровневыми языками, которые явно учитывают характеристики и ограничения аппаратуры GPU. Они не виртуализируют аппаратные ресурсы, требуют изучения аппаратных характеристик GPU, ограничений на максимальный размер программы, получения информации об ошибках и компиляции и др. Такая программа, написанная для какого-то определенного типа GPU, требует переработки для обеспечения максимального использования вычислительного потенциала другого типа GPU.

С появлением графических процессоров с новой многоядерной архитектурой, появились и новые технологии их программирования, такие как nVidia CUDA [2,3] и AMD Close To Metal

(СТМ) [5]. Новые технологии в некоторой степени решают перечисленные проблемы. Данные технологии создавались для возможности максимально эффективного использования ресурсов GPU, они представляют новые модели программирования, адекватные параллелизму и поточности обработки данных. Технология CUDA по сравнению с другими технологиями имеет ряд преимуществ: является языком высокого уровня, имеет ряд особенностей предназначенных для решения задач общего назначения, обладает удобными средствами отладки.

CUDA предназначена для работы на новом поколении GPU NVIDIA от G80 и выше. В модели программировании CUDA GPU рассматривается как вычислительное устройство, способное поддерживать параллельное исполнение большого числа потоковых программ и являющееся сопроцессором CPU. На GPU можно перенести интенсивные вычисления, специфицируя их как функцию, исполняемую на GPU в виде большого числа параллельно работающих потоков.

4. Алгоритмы распараллеливания вычислений на графическом процессоре в технологии CUDA

При использовании итерационных методов решения систем разностных уравнений нет необходимости применять общие методы хранения разреженных матриц. Перед началом итерационного процесса решения системы алгебраических разностных уравнений необходимо вычислить два массива коэффициентов A_x и A_y, связывающие узлы в шаблоне, соответственно, по осям *X* и *Y*, значения в граничных точках, значения функции правой части в узлах сетки. Все элементы этих массивов вычисляются независимо друг от друга и на графическом процессоре вычисляются параллельно.

Распараллеливание итерационного процесса более сложная задача, так как используется метод Зейделя, в котором при вычислении каждого нового элемента вектора используются элементы вычисленные ранее. Основная формула, по которой строится итерационный процесс для отдельного узла сетки:

$$u_{i,j} = \frac{1}{(A_{x;i,j} + A_{x;i-1,j} + A_{y;i,j} + A_{y;i,j-1})}(A_{x;i,j}u_{i+1,j} + A_{x;i-1,j}u_{i-1,j} + A_{y;i,j}u_{i,j+1} + A_{y;i,j-1}u_{i,j-1} - f_{i,j})$$

Для параллельной реализации метода Зейделя при решении ДУЧП часто используется прием красно-черного упорядочения элементов [6], заключающийся в разделении узлов сетки на «красные» и «черные» как можно видеть на рисунке 1. Сначала ищется новое приближение к решению в узлах одного цвета, затем в узлах другого цвета.

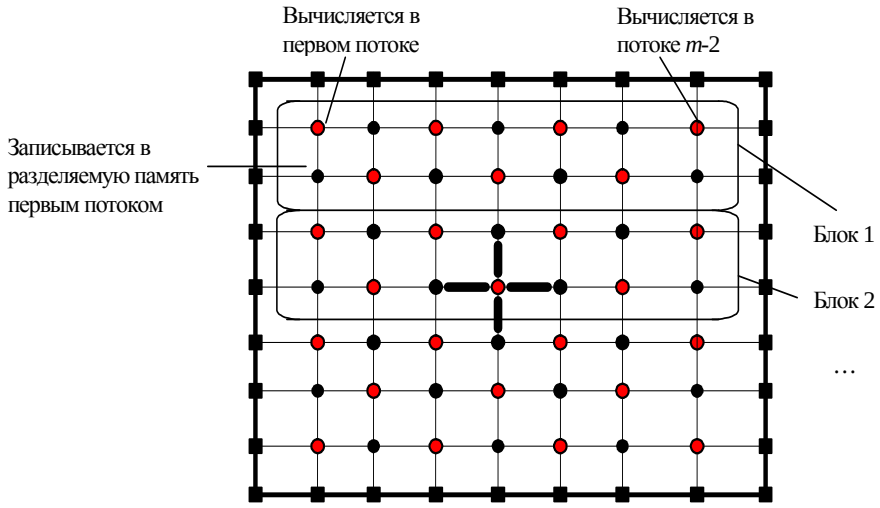

Рис. 1. Разбиение элементов по блокам и потокам

На графическом процессоре распараллеливание происходит следующим образом. Значения хранятся в виде матрицы размером $n \times m$, в одной матрице хранятся как значения внутренних, так и граничных узлов. Вычисления происходят в $n/2$ блоков независимо и параллельно (рисунок 1), в каждом блоке m-2 потоков. Каждый поток в блоке записывает по одному значению в разделяемую память, что позволяет значительно уменьшить вероятность двух одновременных обращений к одному и тому же элементу динамической памяти и ускоряет доступ потоков к требуемым данным.

5. Анализ результатов экспериментов

На графическом процессоре алгоритм реализовывался с помощью технологии CUDA, на центральном процессоре – в системе MATLAB, т.к. MATLAB эффективно использует ресурсы центрального процессора. Для тестового ДУЧП ($f(x,y) = 3(x^2 + y^2)$, $\sigma(x,y) = \frac{1}{xy}$, $\varphi(x,y) = x^3 y^3$) были проведены две серии экспериментов, в первой серии экспериментов исследовалась зависимость величины ускорения от количества узлов в сетки, во второй – от величины допустимой погрешности. Ускорение вычислялось по формуле $k = \frac{t_{CPU}}{t_{GPU}}$, где t_{CPU} - время решения на центральном процессоре и t_{GPU} - время решения на графическом процессоре. Как видно из рисунка 2, при числе узлов сетки 401 и допустимой погрешности 10^{-4}, ускорение равно округленно 47, т.е. GPU решает тестовое ДУЧП в 47 раз быстрее, чем центральный процессор. При изменении величины допустимой погрешности, которая вычислялась как относительная норма приращения, и при постоянном числе узлов сетки, равном 81, также достигается существенное ускорение, например при погрешности 10^{-4} ускорении округленно равно 16. Из рисунка 3 видно, что GPU затрачивает на решение ДУЧП на порядок меньше времени, чем центральный процессор.

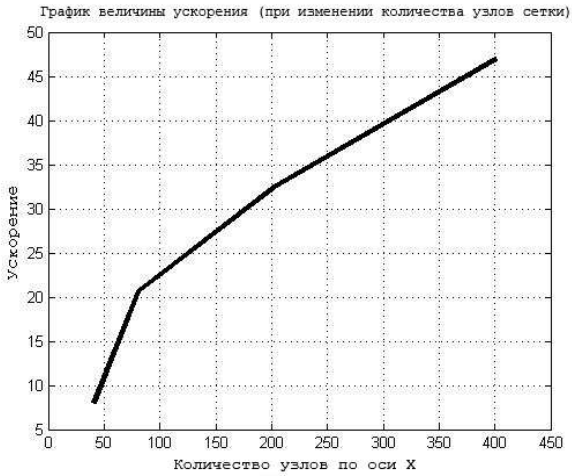

Рис. 2. График зависимости величины ускорения от числа узлов сетки

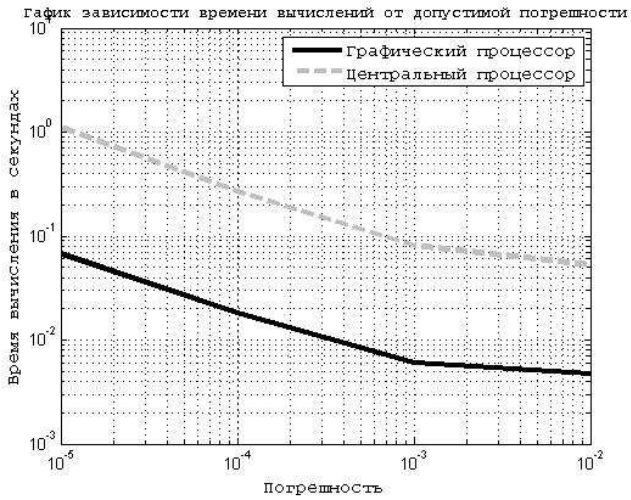

Рис. 3. Графики изменения времени решения от допустимой погрешности

На рисунках 2 и 3 отображены результаты экспериментов, которые доказали эффективность распараллеливания метода конечных разностей решения ДУЧП эллиптического типа на графическом процессоре в технологии CUDA.

6. Заключение

Разработан алгоритм распараллеливания решения методом конечных разностей эллиптического дифференциального уравнения в частных производных на графическом процессоре.

В результате проведенной работы была разработана программа, реализующая разработанный алгоритм.

В результате распараллеливания было достигнуто существенное уменьшение времени вычислений, например, при числе узлов сетки 401 ускорение приблизительно равно 47, что показывает эффективность использования GPU для решения описанного класса задач.

Разработанное программное обеспечение может стать основой для реализации других методов решения дифференциальных уравнений.

Литература

1. *Самарский А. А.* Теория разностных схем. – М.: Наука, 1989. – 616 с.
2. NVIDIA CUDA Compute Unified Device Architecture. Programming Guide // http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf. 2007
3. NVIDIA CUDA Homepage // http://www.nvidia.ru/object/cuda.html
4. Аляутдинов М. А., Троепольская Г. В. Использование современных многоядерных процессоров в нейрокомпьютерах для решения задач математической физики // Нейрокомпьютеры: разработка, применение. – 2007. – № 9. – С. 71 – 80.
5. AMD CTM Guide // http://ati.amd.com/companyinfo/researcher/documents/ATI_CTM_Guide.pdf
6. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991. – 367 с.