АЛГОРИТМЫ С «ДЛИННЫМИ» ВЕКТОРАМИ РЕШЕНИЯ СЕТОЧНЫХ УРАВНЕНИЙ ЯВНЫХ РАЗНОСТНЫХ СХЕМ

Воротникова Д.Г., Головашкин Д.Л.
Институт систем обработки изображений РАН,
Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ)

Аннотация

Предложены два варианта алгоритмов с «длинными» векторами для решения сеточных уравнений явных разностных схем, позволяющих задействовать одновременно максимальное количество ядер CUDA даже для сеточной области небольшой размерности. На примерах разностного решения уравнений теплопроводности и Максвелла продемонстрирована эффективная реализация предложенного подхода. Произведено сравнение предложенных авторами алгоритмов, реализованных при помощи библиотеки CUBLAS, со свободно распространяемыми пакетами B-CALM и OpenCurrent.

Ключевые слова: уравнение теплопроводности, уравнения Максвелла, векторные алгоритмы, разностные схемы, CUBLAS.

Введение

Ведущей современной тенденцией развития вычислительной техники следует признать наращивание производительности не за счёт увеличения тактовой частоты центрального процессора (проблема «кремниевого тупика» [1]), а посредством новых архитектурных решений (многопоточность, векторные сопроцессоры, многоядерность и др.). Наиболее популярным направлением в этой области авторы настоящей работы признают организацию вычислений общего назначения на графических процессорах (GPGPU [2]), относящихся к модели SIMD по классификации Флинна [3]. Реализация основных методов вычислительной математики в рамках упомянутой модели обсуждалась достаточно активно на протяжении последней четверти века [4, 5], хотя данная проблема и не находилась в центре внимания в силу ограниченной доступности соответствующего аппаратного обеспечения (суперкомпьютеры серии Cray [6]). В настоящее время в связи с широким распространением вычислений на графических процессорах (бытовых, таких как видеокарты серии NVIDIA GeForce [7], и профессиональных – NVIDIA Tesla [8]) создание специализированных алгоритмов и программных продуктов с учётом особенностей модели SIMD становится весьма актуальным.

Интересуясь теорией разностных схем, авторы отмечают традиционное предпочтение неявных схем явным, особенно заметное в отечественной литературе [9], [10] и объясняемое худшей устойчивостью последних в большинстве случаев. Действительно, необходимость налагать более густые сеточные области для сохранения устойчивости в случае решения параболических дифференциальных уравнений приводит к росту числа операций при вычислениях по явным схемам, несмотря на их простоту по сравнению с неявными. Однако с недавним появлением доступной аппаратной базы для организации векторных вычислений указанный недостаток нивелируется эффективностью графических процессоров в силу простоты векторизации вычислений по явным схемам и сложностью по неявным. Последнее утверждение относится к прямым методам решения сеточных уравнений неявных схем (прогонки, циклической редукции и др.) [5] и значительно ослабляется при рассмотрении итерационных методов [4]. К сожалению, применение последних подразумевает известный уровень квалификации исследователя (необходимо удачно выбрать сам метод, начальное приближение, задать точность), что ограничивает использование неявных схем в инженерной практике.

К настоящему времени известно множество коммерческих и свободно распространяемых пакетов, реализующих вычисления по явным схемам на графических процессорах. Авторы представленного исследования пользуются разработками OpenCurrent [11] и B-CALM [12] для численного решения уравнений теплопроводности и Максвелла. Изучение открытого кода данных пакетов, основанного на «коротковекторном» представлении (по классификации Ортеги [4]) привело к идее повысить эффективность вычислений переходом к «длинным» векторам, позволяющим задействовать ресурсы видеокарты более полно.

Возможность упомянутого перехода ранее продемонстрирована на примере двух «длинновекторных» алгоритмов метода Якоби для решения сеточных уравнений простейшей неявной схемы для уравнения Пуассона, представленных в работе [4]. Развитие этой методики организации вычислений в случае явных разностных схем (для уравнений теплопроводности и Максвелла) представлено далее.

1. Векторные алгоритмы для уравнения теплопроводности

В качестве первого примера для исследования методов, основанных на использовании «длинных» векторов, рассмотрим разностное решение двумерного однородного линейного нестационарного уравнения теплопроводности:

$$\frac{\partial U}{\partial t} = a\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right), \qquad (1)$$

где $U(x, y, t)$ – распределение температуры в пространстве и времени, $x \in [0,X], y \in [0,Y], t \in [0,T]$,

a – коэффициент температуропроводности. Пусть граничные условия соответствуют условиям Дирихле (температура на границах области равна нулю), а начальное условие имеет вид $U(x,y,0)=\varphi(x,y)$. Простейшая явная разностная схема [9] для этой задачи на сеточной области

$$\omega_h=\{(x_i,y_j,t_k): x_i=ih, y_j=jh(h_x=h_y=h),$$
$$t_k=k\tau, i=\overline{0:N+1}, j=\overline{0:N+1}, k=\overline{1:K}\}$$

имеет вид:

$$U_{i,j}^{k+1}=\alpha\left(U_{i,j-1}^k+U_{i+1,j}^k-4U_{i,j}^k+U_{i-1,j}^k+U_{i,j+1}^k\right)+U_{i,j}^k, \quad (2)$$

где $\alpha=a(\tau/h^2)$.

Далее нам удобнее будет представлять (2) в следующем итерационном виде, следуя рекомендациям А.А. Самарского [9]:

$$U^{k+1}=\alpha AU^k+U^k, \quad (3)$$

где $A=\begin{pmatrix} T_1 & B_1 & & & \\ B_1 & T_2 & B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & B_{N-2} & T_{N-1} & B_{N-1} \\ & & & B_{N-1} & T_N \end{pmatrix}$,

U^k – вектор, полученный из двумерного массива $U_{i,j}^k$ чередованием строк

$$U^k=\left(U_{11}^k,U_{12}^k\ldots U_{N1}^k\ldots U_{NN-1}^k,U_{NN}^k\right)^T.$$

В силу равенства коэффициентов при $U_{i,j+1}^k$, $U_{i,j-1}^k$, $U_{i-1,j}^k$, $U_{i-1,j}^k$ матрица A – симметричная, B_j – диагональная, хранящая коэффициенты при $U_{i,j-1}^k$ и $U_{i,j+1}^k$, а T_j – трёхдиагональная, в которой хранятся коэффициенты при $U_{i,j}^k$ (на главной диагонали), $U_{i-1,j}^k$ и $U_{i+1,j}^k$.

1.1. Алгоритм с «короткими» векторами

Наиболее очевиден вариант алгоритма с использованием «коротких» векторов, представляющих собой строки или столбцы сеточной области. Запишем алгоритмическую реализацию выражения (2) в форме следующего псевдокода для последующего анализа:

```
for j = 2 : N+1
  for i = 2 : N+1
  U_next(i,j) = α(U_last(i,j-1) + U_last(i+1,j) +
                + U_last(i-1,j) + U_last(i,j+1)) +
                + (1-4α)U_last(i,j);
  end
end
U_last = U_next.
```

В алгоритме U_{last} и U_{next} представляют собой двумерные массивы размера $(N+2)\times(N+2)$ элементов, содержащие значения сеточных функций. В большинстве современных векторных пакетов, например в OpenCurrent, используется именно такой способ представления данных. В такой форме записи алгоритма заметно, что он может быть векторизован по столбцам (или строкам) массива U_{last}, то есть операции сложения, умножения могут быть выполнены не поэлементно, а по отношению к столбцам указанных массивов с использованием нотации из [5] следующим образом.

```
for j = 1 : N
  U_next(1:N, j) = α(U_last(0:N-1, j) +
                   + U_last(2:N+1, j) + U_last(1:N, j+1) +
                   + U_last(1:N, j-1)) + (1-4α)U_last(1:N, j);
end
U_last = U_next.
```
(4)

В (4) запись $U_{last}(1{:}N,j)$ означает обращение к j-му столбцу массива U_{last} без первого и последнего элементов, содержащих граничные значения, $U_{last}(2{:}N+1,j)$ – это тот же столбец, сдвинутый на одну позицию вниз.

Главным недостатком данного алгоритма при реализации на видеокарте является загрузка не всех микропроцессоров при пересчёте сеточной функции для небольших значений N. К примеру, чтобы загрузить полностью видеокарту последнего поколения NVIDIA GeForce GTX TITAN Z [7], нам необходима матрица, строки или столбцы которой имеют размер не менее 5760 элементов.

1.2. Первый алгоритм с «длинными» векторами

Следуя (4), рассмотрим следующий способ хранения массива, содержащего значения сеточной функции. Пусть V – это одномерный массив длины $(N+2)^2$, содержащий значения сеточной функции, хранящиеся по строкам, показанный на рис. 1. В отличие от ранее введённого U^k в V входят граничные значения.

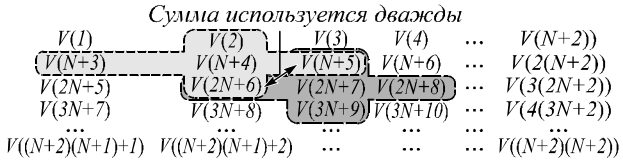

Рис. 1. Схема хранения в виде одномерного массива и попарные суммы

Разностной схеме для уравнения теплопроводности (2) при вычислении лапласиана соответствует дифференциальный шаблон «крест». На рис. 1 обозначены светло- и тёмно-серым выделением элементы вектора V, участвующие в расчёте новых значений $V(N+4)$ и $V(2N+7)$. Очевидно, что попарная сумма элементов $U_{last}(3,2)$ и $U_{last}(2,3)$, соответствующих $V(N+5)$ и $V(2N+6)$, показанная на рис. 1 двунаправленной стрелкой, подсчитывается дважды в алгоритме с короткими векторами, так как используется для нахождения двух элементов сеточных функций. Благодаря использованию «длинных» векторов вместо строк или столбцов матрицы, мы можем сократить общее число операций,

необходимое для расчёта значений сеточной функции, путём введения вспомогательного вектора T размерности $(N+2)(N+1)+1$, который будет хранить в себе результаты попарного сложения.

Таким образом, алгоритм подсчёта сеточной функции на каждом $k+1$-м временном шаге можно записать в следующем виде:

1. Заполнение вектора T попарными суммами:

$$T\left(2:(N+1)(N+2)\right)=V\left(2:(N+1)(N+2)\right)+$$
$$+V\left(N+3:(N+2)^2-1\right).$$

2. Подсчёт значений для следующего временного слоя:

$$V\left(N+4:(N+1)(N+2)-1\right)=$$
$$=\alpha\left(T\left(2:N(N+2)-1\right)+T\left(N+5:N(N+2)-1\right)\right)+$$
$$+V\left(N+4:(N+1)(N+2)-1\right).$$

3. Восстановление граничных значений.

Следующий временной слой пишется поверх предыдущего.

Основные векторные операции данного алгоритма – saxpy и векторные сложения. В результате выполнения данного алгоритма мы сэкономим одну операцию сложения при вычислении каждого значения сеточной функции по дифференциальному шаблону, однако нам придётся использовать дополнительную память для хранения вспомогательного вектора T.

Переход к работе с «длинными» векторами позволил перейти от цикла, повторяющегося N раз (количество столбцов матрицы), в котором выполняется 3 операции векторного сложения и одна операция saxpy с векторами длины N, к двум операциям сложения векторов длины $(N+1)(N+2)-2$ и одному выполнению saxpy для векторов длины $N(N+2)-1$, что (как будет показано далее) приводит к более полному использованию аппаратных возможностей видеокарты и, как следствие, ускорению вычислений.

Недостатком данного алгоритма является затирание граничных значений, которые заменяются на некие фиктивные результаты в процессе перехода на следующий слой по времени, с чем связана необходимость в последнем шаге алгоритма.

1.3. Второй алгоритм с «длинными» векторами

Второй вариант – векторизация с использованием gaxpy, основной операции в (3). Главная диагональ матрицы A в (3) соответствует центральному узлу дифференциального шаблона. Коэффициент при нём состоит из двух слагаемых: коэффициента при сеточной функции $U_{i,j}^k$ из конечной разности по пространству и единицы из конечной разности по времени. Первая и минус первая диагонали соответствуют коэффициентам при $U_{i+1,j}^k$ (первая) и $U_{i-1,j}^k$ (минус первая). Главная диагональ не содержит нулевых элементов в отличие от первой и минус первой, где блоки неплотно примыкают друг к другу (по одному нулю в каждой диагонали):

$$T_j=\begin{pmatrix}1-4\alpha & 1 & & & \\ 1 & 1-4\alpha & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & 1-4\alpha\end{pmatrix}. \qquad (5)$$

Две другие блочные диагонали (N–1 и –N+1) формируются коэффициентами при $U_{i,j+1}^k$ и $U_{i,j-1}^k$, расположенными на главных диагоналях указанных блоков. Таким образом, мы получаем матрицу с пятью ненулевыми диагоналями: главная длины $N\times N$, первая и минус первая диагонали длины $N\times N-1$ и N–1 и –N+1 диагонали длины $N\times(N-1)$.

В работе [13] авторами настоящей статьи предложены векторные алгоритмы умножения трёхдиагональной матрицы на столбец при помощи GPU. Основная операция, используемая там, – покомпонентное умножение векторов. В общем виде умножение трёхдиагональной матрицы, главная диагональ которой имеет размер $N\times N$, на вектор $z=Ax$, может быть записано следующим образом в нотации из [13].

$z = Diag_0 .* x$ % умножение главной диагонали на вектор x.

$z\left(1:N^2-1\right)=z\left(1:N^2-1\right)+Diag_1\left(1:N^2-1\right).*x\left(2:N^2\right);$

% работа с первой диагональю.

$z\left(2:N^2\right)=z\left(2:N^2\right)+Diag_{-1}\left(2:N^2\right).*x\left(1:N^2-1\right);$

% работа с минус первой диагональю.

$m=N-1;\quad t=(m-1)N^2-m(m-1)/2;$

$z\left(1:N^2-m\right)=z\left(1:N^2-m\right)+$
$+Diag_{N-1}\left(t+1:t+N^2-m\right).*x\left(m+1:N^2\right);$

% работа с N–1 диагональю.

$z\left(m+1:N^2\right)=z\left(m+1:N^2\right)+$
$+Diag_{-N+1}\left(t+1:t+N^2-m\right).*x\left(1:N^2-m\right);$

% работа с –N+1 диагональю.

Здесь операция « .* » обозначает покомпонентное умножение векторов.

В данном алгоритме мы также работаем с векторами наибольшей длины, а не столбцами (строками). В процессе работы алгоритма мы дважды выполняем операцию saxpy для векторов длины $N\times(N-1)$ и длины $N\times N-1$ и один раз производится покомпонентное умножение векторов длины $N\times N$. В данном варианте алгоритма с «длинными» векторами не происходит затирание, а, следовательно, и восстановление граничных значений.

2. Алгоритм FDTD-метода с «длинными» векторами

Рассмотрим двумерный вариант FDTD-метода, широко применяемого [15] в дифракционной нанофотонике, вычислительной электродинамике и в других

областях, касающихся исследования распространения электромагнитного излучения. В двумерном случае распространение H-волны соответствует системе уравнений Максвелла:

$$\mu_0 \frac{\partial Hy}{\partial t} = -\frac{\partial Ex}{\partial z}, \quad \mu_0 \frac{\partial Hz}{\partial t} = \frac{\partial Ex}{\partial y}, \qquad (6)$$
$$\varepsilon\varepsilon_0 \frac{\partial Ex}{\partial t} = \frac{\partial Hz}{\partial y} - \frac{\partial Hy}{\partial z},$$

где Ex, Hy, Hz – проекции векторов напряжённостей электромагнитного поля, μ_0, ε_0 – магнитная и электрическая константы, ε – диэлектрическая проницаемость. Определим область D ($0 \le t \le T$, $0 \le y \le L_y$, $0 \le z \le L_z$), на которую наложим сеточную область D_h. В этом случае проекцию Ex определим в узлах $\{(t_n, y_m, z_k)$: $t_n = nh_t$, $n = \overline{0, N}$, $N = T/h_t$, $y_m = mh_y$, $m = \overline{0, M}$, $M = L_y/h_y$, $z_k = kh_z$, $k = \overline{0, K}$, $K = L_z/h_z\}$, проекцию Hy – $\{(t_{n+0,5}, y_m, z_{k+0,5})$: $t_{n+0,5} = (n+0{,}5)h_t$, $n = \overline{0, N-1}$, $y_m = mh_y$, $m = \overline{1, M-1}$, $z_{k+0,5} = (k+0{,}5)h_z$, $k = \overline{0, K-1}\}$, проекцию Hz – $\{(t_{n+0,5}, y_{m+0,5}, z_k)$: $t_{n+0,5} = (n+0{,}5)h_t$, $n = \overline{0, N-1}$, $y_{m+0,5} = (m+0{,}5)h_y$, $m = \overline{0, M-1}$, $z_k = kh_z$, $k = \overline{0, K-1}\}$. Упрощая задачу, выделим на D подобласть D^c ($0 \le t \le T$, $L \le y \le R$, $B \le z \le U$), при работе с которой нет необходимости в заботе о значениях сеточных функций на краях D, задав на ней D_h^c. Будем считать, что $Ex_{m,k}^n$ определена в узлах $\{(t_n, y_m, z_k)$: $t_n = nh_t$, $n = \overline{0, N}$, $N = T/h_t$, $y_m = mh_y$, $m = \overline{L, R}$, $L = L_l/h_y$, $R = L_r/h_y$, $z_k = kh_z$, $k = \overline{B, U}$, $U = L_u/h_z$, $B = L_b/h_z\}$, $Hy_{m,\,k+0,5}^{n+0,5}$ – $\{(t_{n+0,5}, y_m, z_{k+0,5})$: $t_{n+0,5} = (n+0{,}5)h_t$, $n = \overline{0, N-1}$, $y_m = mh_y$, $m = \overline{L+1, R-1}$, $z_{k+0,5} = (k+0{,}5)h_z$, $k = \overline{B, U-1}\}$, $Hz_{m+0,5,\,k}^{n+0,5}$ – $\{(t_{n+0,5}, y_{m+0,5}, z_k)$: $t_{n+0,5} = (n+0{,}5)h_t$, $n = \overline{0, N-1}$, $y_{m+0,5} = (m+0{,}5)h_y$, $m = \overline{L, R-1}$, $z_k = kh_z$, $k = \overline{B+1, U-1}\}$. В предложенной области и подобласти индексы m, k обозначают узлы по пространству, n – по времени. Расстояния между узлами задаются пространственными (h_y и h_z) и временным (h_t) шагами сетки. Сеточное значение диэлектрической проницаемости ($\varepsilon_{j,k}$) характеризует изучаемый оптический элемент. Тогда явная разностная схема Yee для системы уравнений (6) выглядит как:

$$\mu_0 \frac{Hy_{m,k+1/2}^{n+1/2} - Hy_{m,k+1/2}^{n-1/2}}{h_t} = -\frac{Ex_{m,k+1}^n - Ex_{m,k}^n}{h_z},$$
$$\mu_0 \frac{Hz_{m+1/2,k}^{n+1/2} - Hz_{m+1/2,k}^{n-1/2}}{h_t} = -\frac{Ex_{m+1,k}^n - Ex_{m,k}^n}{h_z}, \qquad (7)$$
$$\varepsilon\varepsilon_0 \frac{Ex_{m,k}^{n+1} - Ex_{m,k}^n}{h_t} = \frac{Hz_{m+1/2,k}^{n+1/2} - Hz_{m-1/2,k}^{n+1/2}}{h_y} - \frac{Hz_{m,k+1/2}^{n+1/2} - Hz_{m,k-1/2}^{n+1/2}}{h_z}.$$

Рассмотрим данную явную двухслойную схему в нотации (3) и на её основе запишем первые два выражения (6) в итерационном виде, следуя рекомендациям А.А. Самарского [9]:

$$Hy^{k+1} = \alpha_1 AEx^k + Hy^k, \qquad (8.1)$$

где $A = \begin{pmatrix} -T_1 & T_1 & & & \\ & -T_2 & T_2 & & \\ & & \ddots & \ddots & \\ & & & -T_N & T_N \end{pmatrix}$.

$$Hz^{k+1} = \alpha_2 BEx^k + Hz^k, \qquad (8.2)$$

где $B = \begin{pmatrix} -T_1 & 0 & \dots & T_1 & & \\ & -T_2 & 0 & \dots & T_2 & \\ & & \ddots & \ddots & & \ddots \\ & & & -T_N & 0 & \dots & T_N \end{pmatrix}$,

$\alpha_1 = (h_t/\mu_0 h_z)$, $\alpha_2 = (h_t/\mu_0 h_y)$ и T_i – диагональная матрица, хранящая значения коэффициентов при Ex^k. Отбросим краевые значения, так как при добавлении PML данные граничные условия будут считаться с помощью других уравнений, отличных от тех, что используются для пересчёта в середине сетки.

Как и в случае с уравнением теплопроводности, мы приходим к работе с диагональными матрицами, диагонали которых состоят из коэффициентов при $Ex_{m,k+1}^n$ и $Ex_{m,k}^n$ для (8.1) и $Ex_{m+1,k}^n$ и $Ex_{m,k}^n$ для (8.2), следовательно, алгоритм сводится к описанному в пункте 1.3 алгоритму. Третье уравнение из (7) также считается при помощи «длинных» векторов.

3. Вычислительные эксперименты

В качестве аппаратного обеспечения использован ПК с операционной системой Debian 7.0 (wheezy) и установленными драйверами CUDA Toolkit 4 и компилятором gcc. Также была использована библиотека CUBLAS, являющаяся аппаратно-ориентированной реализацией программного интерфейса BLAS (Basic Linear Algebra Subprograms). Результаты, представленные далее, получены на видеокарте GeForce GTX 660Ti (табл. 1) и процессоре Intel Core2 Duo CPU E8500 (табл. 2).

Таблица 1. Основные характеристики GPU NVIDIA GeForce GTX 660Ti

Характеристика	Значение
Количество мультипроцессоров, шт.	1228
Размер видеопамяти, Мб	2048
Максимальное число потоков в блоке, шт.	512
Максимальная размерность блока потоков (x, y, z), шт.	512×512×64
Максимальная размерность сетки блоков, шт.	65535×65535×1
Тактовая частота ядра, МГц	915
Тактовая частота памяти, МГц	1502

Таблица 2. Основные характеристики CPU Intel Core2 Duo CPU E8500

Характеристика	Значение
Тактовая частота ядра, ГГц	3,16
Тактовая частота шины CPU, МГц	1333
Кеш L1, Кб	64×2
Кеш L2, Кб	6144
Тактовая частота ядра, ГГц	3,16

На рис. 2 представлено сравнение результатов, полученных с помощью первого алгоритма с «длинными» векторами, реализованного для двумерного уравнения теплопроводности, и с помощью коротковекторного для сеточных областей размером $N \times N$ ($N = 1..1000$). Реализация алгоритмов происходила с использованием таких функций стандартной библиотеки CUBLAS, работающей с GPU, как cublasSaxpy(), cublasSgemv(). Время работы алгоритма с «короткими» векторами растёт линейно с увеличением N, т.к. происходит линейное увеличение количества вызовов функций библиотеки CUBLAS, а следовательно, и количества арифметических операций. Алгоритмы, оперирующие «длинными» векторами, показывают незначительную разницу по времени работы программы как для сеточной области с $N = 100$, так и для $N = 1000$. Выигрыш алгоритма, работающего с «длинными» векторами, обусловлен, во-первых, меньшим количеством арифметических операций (для первого алгоритма из п. 3), во-вторых, особенностью библиотеки CUBLAS для векторных операций, заключающейся в том, что на каждую векторную операцию запускается отдельное ядро (CUDA Kernel), что привносит фиксированную задержку, зависящую не от длины вектора (до некоторого достаточно большого значения), а только от количества таких векторов.

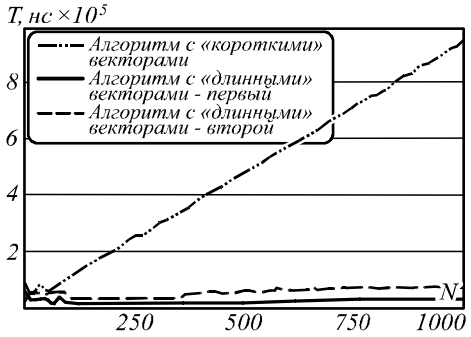

Рис. 2. Сравнение времени работы T коротковекторного и длинновекторных алгоритмов для уравнения теплопроводности

Второй алгоритм из п. 1.3 работает медленнее первого, т.к. первый, обладая всеми плюсами «длинных» векторов, имеет меньшее количество арифметических операций. На рис. 3 приведено сравнение лучшего алгоритма с «длинными» векторами для уравнения теплопроводности с реализацией на OpenCurrent, где видно, что первый алгоритм из п. 3, обозначенный пунктиром, работает быстрее – его ускорение составило 2.

Рис. 3. Сравнение времени работы T длинновекторных алгоритма и пакета OpenCurrent

На рис. 4 приведены результаты для алгоритма, работающего с «длинными» векторами для FDTD-метода, и реализация этой же задачи при помощи пакета B-CALM. Алгоритм для «длинных» векторов даёт выигрыш по времени примерно в 1,4 раза.

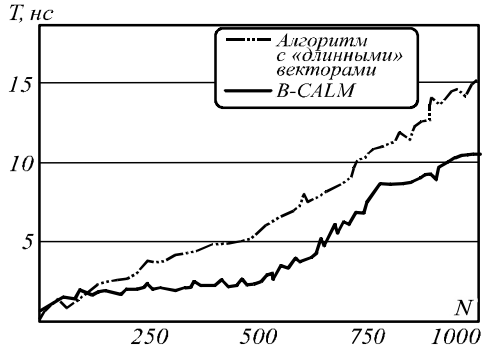

Рис. 4. Сравнение времени работы T длинновекторного алгоритма и реализации двумерного FDTD-метода при помощи пакета B-CALM

Заключение

В данной статье приведено описание двух алгоритмов, оперирующих с «длинными» векторами, взамен работы со строками или столбцами сеточной области. Данные алгоритмы позволяют более рационально использовать ядра CUDA, что, по мнению авторов статьи, особенно актуально в связи с тенденцией производителей видеокарт к увеличению количества микропроцессоров.

В результате экспериментальных исследований была показана эффективность алгоритмов с «длинными» векторами как для решения уравнения теплопроводности при помощи библиотеки CUBLAS, так и для двумерного варианта FDTD-метода, что доказывает актуальность предложенного подхода к решению исследуемой задачи.

В дальнейшем данный подход может быть также применён для решения задач оптики, нанофотоники, гидродинамики, аэродинамики и других, где уместно использование явных разностных схем.

Благодарности

Работа выполнена при поддержке Министерства образования и науки РФ и грантов РФФИ 14-07-00291А и 14-07-31178мол_а.

Литература

1. Глобальный прогноз развития кремниевых технологий [Электронный ресурс]. – http://www.russianelectronics.ru/leader-r/review/doc/64629/ (дата обращения 15.12.2014).
2. General-purpose computing on graphics processing units [Электронный ресурс]. – http://www.gpgpu.org/ (дата обращения 18.12.2014).
3. **Flynn, M.J.** Very High-Speed Computing System / M.J. Flynn // Proceedings IEEE – 1966. – Vol. 54(12). – P. 1901-1909.
4. **Ortega, J.** Introduction to Parallel and Vector Solution of Linear Systems / J. Ortega. – NY: Plenum Press, 1987. – 313 p.
5. **Golub**, **G.H.** Matrix Computations / G.H. Golub, Ch.F. Van Loan. – JHU Press, 1996. – 747 p.
6. **Murray, C.M**. The Supermen: The Story of Seymour Cray and the Technical Wizards Behind the Supercomputer / Ch.J. Murray. – Wiley, 1997. – 232 p.
7. NVIDIA GeForce [Электронный ресурс]. – http://www.nvidia.ru/object/geforce-family-ru.html (дата обращения 18.12.2014).
8. NVIDIA Tesla [Электронный ресурс]. – http://www.nvidia.ru/object/tesla-high-performance-computing-ru.html (дата обращения 16.12.2014).
9. **Самарский, А.А**. Введение в теорию разностных схем / А.А. Самарский – М.: Наука, 1971. – 552 с.
10. **Волков, К.Н**. Численное решение задач гидродинамики на графических процессорах общего назначения / К.Н. Волков, В.Н. Емельянов, А.Г. Карпенко, И.В. Курова, А.Е. Серов // Вычислительные методы и программирование – 2013. – Т. 14(1). – С. 82-90.
11. OpenCurrent is an open source C++ library for solving Partial Differential Equations (PDEs) over regular grids using the CUDA platform from NVIDIA [Электронный ресурс]. – https://code.google.com/p/opencurrent/ (дата обращения 18.12.2014).
12. **Wahl, P.** B-CALM: An open-source GPU-based 3D-FDTD with multi-pole dispersion for plasmonics / P. Wahl, D.-S. Ly-Gagnon, Ch. Debaes, D.A.B. Miller, H. Thienpont // Optical and Quantum Electronics. – 2012. – Vol. 44. – P. 285-290.
13. **Golovashkin, D.L**. Solving finite-difference equations for diffractive optics problems using graphics processing units / D.L. Golovashkin, D. Vorotnikova, A. Kochurov, S. Malysheva // Optical Engineering. – 2013. – Vol. 52(9). – Doi: 10.1117/1.OE.52.9.091719.
14. **Головашкин, Д.Л.** Решение сеточных уравнений на графических вычислительных устройствах. Метод пирамид / Д.Л. Головашкин, А.В. Кочуров // Вычислительные технологии. – 2012. – Т. 17, № 3. – С. 55-69.
15. Дифракционная нанофотоника / А.В. Гаврилов, Д.Л. Головашкин, Л.Л. Досколович, П.Н. Дьяченко, А.А. Ковалёв, В.В. Котляр, А.Г. Налимов, Д.В. Нестеренко, В.С. Павельев, Р.В. Скиданов, В.А. Сойфер, С.Н. Хонина, Я.О. Шуюпова. – Под ред. В.А. Сойфера. – М.: Физматлит, 2011. – 680 с.

References

1. The global forecast for the development of silicon technology [Electronical Resource]. – http://www.russianelectronics.ru/leader-r/review/doc/64629/ (request date 15.12.2014). – (In Russian).
2. General-purpose computing on graphics processing units [Electronical Resource]. – http://www.gpgpu.org/ (request date 18.12.2014).
3. **Flynn, M.J.** Very High-Speed Computing System / M.J. Flynn // Proceedings IEEE. – 1966. – Vol. 54(12). – P. 1901-1909.
4. **Ortega, J.** Introduction to Parallel and Vector Solution of Linear Systems / J. Ortega. – New York: Plenum Press, 1987. – 313 p.
5. **Golub**, **G.H.** Matrix Computations / G.H. Golub, Ch.F. Van Loan. – JHU Press, 1996. – 747 p.
6. **Murray, C.M**. The Supermen: The Story of Seymour Cray and the Technical Wizards Behind the Supercomputer / Ch.J. Murray. – Wiley, 1997. – 232 p.
7. NVIDIA GeForce [Electronical Resource]. – http://www.nvidia.ru/object/geforce-family-ru.html (request date 18.12.2014).
8. NVIDIA Tesla [Electronical Resource]. – http://www.nvidia.ru/object/tesla-high-performance-computing-ru.html (request date 16.12.2014).
9. **Samarski, A.A.** Introduction into the theory of difference schems / A.A. Samarski. – Moscow: “Nayka” Publisher, 1971. – 552 p. – (In Russian).
10. **Volkov, K.N.** Numerical solution of hydrodynamics problems by means of graphical processors / K.N. Volkov, V.N. Emelanov, A.G. Karpenko, I.V. Kyrova, A.E. Serov // Calculating methods and programming. – 2013. – Vol. 14(1). – P. 82-90. – (In Russian).
11. OpenCurrent is an open source C++ library for solving Partial Differential Equations (PDEs) over regular grids using the CUDA platform from NVIDIA [Electronical Resource]. – https://code.google.com/p/opencurrent/ (request date 18.12.2014).
12. **Wahl, P.** B-CALM: An open-source GPU-based 3D-FDTD with multi-pole dispersion for plasmonics / P. Wahl, D.-S. Ly-Gagnon, Ch. Debaes, D.A.B. Miller, H. Thienpont // Optical and Quantum Electronics. – 2012. – Vol. 44. – P. 285-290.
13. **Golovashkin, D.L**. Solving finite-difference equations for diffractive optics problems using graphics processing units / D.L. Golovashkin, D. Vorotnikova, A. Kochurov, S. Malysheva // Optical Engineering. – 2013. – Vol. 52(9). – Doi: 10.1117/1.OE.52.9.091719.
14. **Golovashkin, D.L.** Solution of difference equations in graphical computing devices. Method of pyramids / D.L. Golovashkin, A.V. Kochurov // Computing Technologies. – 2012. – Vol. 17(3). – P. 55-69. – (In Russian).
15. Diffractive nanophotonics / A.V. Gavrilov, D.L. Golovashkin, , L.L. Doskolovich, P.N. Dyachenko, A.A. Kovalev, V.V. Kotlyar, A.G. Nalimov, D.V. Nesterenko, V.S. Pavelyev, R.V. Skidanov, V.A. Soifer, S.N. Khonina, Ya.O. Shuyupova. – Ed. by V.A. Soifer. – Moscow, “Fizmatlit” Publisher, 2011. – 680 p. – (In Russian).

LONG VECTORS ALGORITHMS FOR SOLVING GRID EQUATIONS OF EXPLICIT DIFFERENCE SCHEMES

D.G. Vorotnikova, D.L. Golovashkin
Image Processing Systems Institute of the Russian Academy of Sciences,
Samara State Aerospace University

Abstract

We propose two variants of long vectors algorithms for solving grid equations of explicit difference schemes, allowing one to use the maximum number of CUDA cores even for a small dimension of the grid domain. The implementation of these algorithms is shown by the example

of the heat conduction equation and the solution of Maxwell's equations. The comparison between the proposed algorithms implemented by means of the CUBLAS library and free software packages B-CALM and OpenCurrent is done.

Keywords: heat equation, Maxwell's equations, difference scheme, PDE, CUBLAS.

Сведения об авторах

Воротникова Дарья Геннадьевна, 1989 года рождения. В 2012 году окончила Самарский государственный аэрокосмический университет по направлению «Прикладная математика и информатика». Работает стажёром-исследователем в Институте систем обработки изображений Российской академии наук и является аспирантом кафедры наноинженерии Самарского государственного аэрокосмического университета. Область научных интересов: векторные и параллельные вычисления, FDTD- и BPM-методы.

E-mail: *dayavorotnikova@gmail.com* .

Daria Gennadievna Vorotnikova, (b. 1989). Graduated (2012) from S.P. Korolyov Samara State Aerospace University (SSAU). Since 2012 she is the graduate student of SSAU and the trainee-researcher of the Image Processing Systems Institute of the RAS. Scientific interests: BPM- and FDTD-method, vector and parallel algorithms for matrix computation.

Головашкин Димитрий Львович, доктор физико-математических наук, доцент, ведущий научный сотрудник Института систем обработки изображений РАН. Область научных интересов: разностное решение уравнений Максвелла (FDTD-метод), дифракционная оптика, векторные и параллельные матричные вычисления.

E-mail: *dimitriy@smr.ru* .

Dimitry Lvovich Golovashkin, Doctor of Physical and Mathematical Sciences, Associate Professor, Leader Researcher of the Image Processing Systems Institute of the Russian Academy of Sciences. Scientific interests: FDTD-method, sub wave optics, vector and parallel algorithms of matrix computation.

Поступила в редакцию 20 января 2015