Длинновекторные алгоритмы, реализуемые при помощи библиотеки CUBLAS

Воротникова Д.Г., Головашкин Д.Л.

daryavorotnikova@gmail.com

Институт систем обработки изображений РАН

Россия, 443001, Самара, ул. Молодогвардейская, 151.

Аннотация

Данная статья предлагает векторные методы, направленные на более эффективную загрузку микропроцессоров GPU. Основная идея метода заключается в использовании длинных векторов аргументов взамен матричных строк или столбцов. Работоспособность алгоритма показана на сравнении предлагаемых подходов с реализацией при помощи библиотеки OpenCurrent и пакета B-CALM.

Введение

Ведущей современной тенденцией развития вычислительной техники следует признать наращивание производительности не за счет 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, с широким распространением вычислений на графических процессорах (бытовых, как видеокарты серии NVIDIA GeForce[7] и профессиональных — NVIDIA Tesla[8]), создание специализированных алгоритмов и программных продуктов с учетом особенностей модели SIMD становится весьма актуальным.

Интересуясь теорией разностных схем, авторы отмечают традиционное предпочтение неявных схем над явными, особенно заметное в отечественной литературе[9] 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 приближении, 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 настоящего исследования пользуются разработками OpenCurrent [10] и B-CALM [11] для численного решения уравнений теплопроводности и Максвелла. Изучение открытого кода данных пакетов, основанного на «коротко-векторном» представлении (по классификации Ортеги [4]) привело к идее повысить эффективность вычислений переходом к «длинным» векторам, позволяющем задействовать ресурсы видеокарты более полно.

Возможность упомянутого перехода ранее продемонстрирована на примере двух «длинно-векторных» 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. Длинновекторные 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:

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

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 для этой задачи на сеточной области $\omega_h=\{(x_i,y_j,t_k): x_i=ih, y_j=jh(h_x=h_y=h), t_k=k\,\tau; i,j=\overline{0:N+1}, k=\overline{1:K}\}$ имеет вид:

$$\begin{aligned} &U_{i,j}^{k+1}=U_{i,j}^k+a\frac{\tau}{h^2}\left(U_{i,j-1}^k-2U_{i,j}^k+U_{i+1,j}^k\right)+a\frac{\tau}{h^2}\left(U_{i-1,j}^k-2U_{i,j}^k+U_{i,j+1}^k\right); \\ &U_{i,j}^{k+1}=\alpha\left(U_{i,j-1}^k+U_{i+1,j}^k-4U_{i1,j}^k+U_{i-1,j}^k+U_{i,j+1}^k\right)+U_{i,j}^k, \ \text{где} \ \alpha=a\frac{\tau}{h^2} \end{aligned} \quad . (2)$$

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

$$U^{k+1}=\alpha AU^k+U^k \quad , \text{где} \quad A=\begin{pmatrix} T_1 & B_1 & & & \\ B_1 & T_2 & B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & B_{N-2} & T_{N-1} & B_{N-1} \\ & & & B_{N-1} & T_N \end{pmatrix} \quad , (3)$$

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

$U^k=(U^k_{11},U^k_{12}\dots U^k_{NN-1},U^k_{NN})^T$. В силу симметричности коэффициентов при $U^k_{i,j+1}$, $U^k_{i,j-1}$, $U^k_{i+1,j}$, $U^k_{i-1,j}$ матрица B_j - диагональная, хранящая коэффициенты при $U^k_{i,j-1}$ и $U^k_{i,j+1}$, а T_j - трехдиагональная, в которой хранятся коэффициенты при $U^k_{i,j}$, $U^k_{i-1,j}$ и $U^k_{i+1,j}$.

1. 2 Коротковекторный алгоритм

Наиболее очевиден вариант алгоритма с использованием коротких векторов. Запишем выражение (2) в форме следующего псевдокода для последующего анализа:

```
for j=1:N
 for i=1:N
  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 (матрицы значений сеточных функций) размера $(N+2)\times(N+2)$ элементов, содержащие значения сеточных функций и граничных значений. В большенстве современных векторных пакетов, например в OpenCurrent, используется именно такой способ представленя данных. В такой форме записи алгоритма заметно, что он может быть векторизован по столбцам (или строкам) массива U_{last} , то есть указанные операции сложения, умножения могут быть выполнены не поэлементно, а по отношению к столбцам указанных массивов. Данный алгоритм в будет выглядеть следующим образом в соответствии с [9]:

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

В (4) запись $U_{last}(1:N,j)$ означает j-ый столбец массива U^k без первого и последнего элементов, содержащих граничные значения, $U_{last}(0:N,j)$, $U_{last}(2:N,j)$ - это столбец, сдвинутый на одну позицию вверх или вниз.

Главным недостатком данного алгоритма при реализации на видеокарте является загрузка не всех микропроцессоров при пересчете сеточной функции для небольших значений N.

1. 3 Первый длинновекторный алгоритм

Следуя [4] рассмотрим следующий способ хранения массива, содержащего значения сеточной функции. Пусть U — это одномерный массив длины $(N+2)^2$, содержащий значения сеточной функции, хранящихся по строкам, показанный на рисунке 1.

Рисунок 1 — Схема хранения в виде одномерного массива и попарные суммы

По сути, разностной схеме уравнения теплопроводности (2) соответствует шаблон для подсчета разностной схемы «крест». На рис.1 обозначены элементы вектора U, участвующие в расчете значений $U^{k+1}(N+4)$ и $U^{k+1}(2N+7)$. Очевидно, что попарная сумма элементов $U(N+5)$ и $U(2N+6)$, показанная на рис.1 стрелкой, подсчитывается дважды в коротковекторном алгоритме, так как используются для нахождения и $U^{k+1}(N+4)$, и $U^{k+1}(2N+7)$. Таким образом, благодаря использованию длинных векторов вместо строк или столбцов матрицы, мы можем сократить общее число операций, необходимое для расчета значений сеточной функции, путем введения вспомогательного вектора T размерности $(N+2)(N+1)+1$, который будет хранить в себе результаты попарного сложения.

Следовательно, алгоритм подсчета сеточной функции на каждом $k+1$-ом временном шаге можно записать в следующем виде:

1. Заполнение вектора T попарными суммами:

$$T(2:(N+1)(N+2))=U^k(2:(N+1)(N+2))+U^k(N+3:(N+2)^2-1)\quad;$$

2. Подсчет значений для следующего временного слоя:

$$U^{k+1}(N+4:(N+1)(N+2)-1)=\alpha(T(2:N(N+2)-1)+T(N+5:N(N+2)-1))+$$
$$+U^k(N+4:(N+1)(N+2)-1)$$

3. Восстановление граничных значений.

Основные векторные операции данного алгоритма — gaxpy и векторные сложения. В результате выполнения данного алгоритма мы исключим одно векторное сложение, однако нам придется использовать дополнительную память для хранения вспомогательного вектора T. Недостатком данного алгоритма также является затирание граничных значений, которые заменяются на некие фиктивные результаты в процессе итераций. Т.е. после выполнения каждой итерации необходимо восстановление граничных значений.

1.4 Второй длинновекторный алгоритм

Второй вариант — векторизация с использованием матричного умножения, основной операцией в (3). Разворачиваем матрицу значений сеточных функций по строкам. Рассмотрим ее структуру подробнее. Тогда нулевая диагональ матрицы A в (3) соответствует центральному узлу дифференциального шаблона. Коэффициент при нем состоит из двух слагаемых: коэффициента при сеточной функции $U_{i,j}$ из конечной разности по пространству и единицы. Первая и минус первая диагонали соответствуют коэффициентам при $U_{i+1,j}$ - первая и $U_{i-1,j}$ - минус первая . Главная диагональ не

содержит нулевых элементов в отличае от остальных, где блоки не плотно примыкают друг к другу - в первой и минус первой по одному нулю в каждой:

$$T_j = \begin{pmatrix} c_{1,j} & a_{1,j} & & & \\ a_{1,j} & c_{2,j} & a_{2,j} & & \\ & \ddots & \ddots & \ddots & \\ & & & a_{N-1,j} & c_{N,j} \end{pmatrix} .$$

Две другие диагонали блоков формируются коэффициентами при $U_{i,j+1}$ и $U_{i,j-1}$. Таким образом мы получаем матрицу с пятью ненулевыми диагоналями: нулевая центральная длины $N*N$, первая и минус первая диагонали длины $N*N-1$ и $N+1$ и $N-1$ диагонали.

В работе [12] показаны векторные вычисления на GPU для схожей модели, где продемонстрирован подход распараллеливания алгоритма для умножения блочной трехдиагональной матрицы. Основная операция используемая в работе [12] — gaxpy, в нашем же варианте рассматривается более простое покомпонентное умножение. В общем виде умножение трехдиагональной матрицы $N*N$ на вектор $z = Ax$ может быть записана следующим образом в нотации из [12].

$z = A_{diag} .* x$ %умножение главной диагонали на вектор x

$z(1:N-1) += B_{diag}(1:N-1) .* x(2:N);$ % работа с первой диагональю

$z(2:N) += C_{diag}(2:N) .* x(1:N-1);$ % работа с минус первой диагональю

В данном алгоритме мы также работаем с векторами наибольшей длины, а не столбцами (строками), что приводит к более эффективной загрузке микропроцессоров, но в отличие от предыдущего длинновекторного алгоритма здесь нет явного уменьшения количества операций, но в нем не происходит затирание, а следовательно и восстановление, граничных значений.

2 Длинновекторный алгоритм для FDTD-метода

В качестве второго примера реализации длинновекторного алгоритма рассмотрим двумерный вариант FDTD-метода, широко применяемого в дифракционной нанофотонике[14], электродинамике и др. В двумерном случае распространение H-волны соответствует системе уравнений Максвелла:

$$\begin{aligned} \mu_0 \frac{\partial Hy}{\partial t} &= -\frac{\partial Ex}{\partial z}, \\ \mu_0 \frac{\partial Hz}{\partial t} &= \frac{\partial Ex}{\partial y}, \\ \epsilon \epsilon_0 \frac{\partial Ex}{\partial t} &= \frac{\partial Hz}{\partial y} - \frac{\partial Hy}{\partial z}, \end{aligned} \qquad (5)$$

где Ex , Hy , Hz - компоненты электромагнитного поля, μ_0 , ϵ_0 - магнитная и электрическая константы, ϵ - диэлектрическая проницаемость.

Рассмотрим распространение H-волны в двумерной области D . Пусть в узлах $\{(t_m, y_j, z_k): t_m = mh_t, m = \overline{0:M}, M = T/h_t,\ y_j = jh_y, j = \overline{0:J-1}, J = L_y/h_y,\ z_k = k h_z, k = \overline{0:K}, K = L_z/h_z\}$ сеточной структуры D_h^2 , наложенной на D определена сеточная проекция электрического поля на ось x - $Ex_{i,k}^m$. Сеточная проекция магнитного поля на ось z - $Hz_{j+1/2,k}^{m+1/2}$ определена в узлах $\{(t_{m+1/2}, y_{j+1/2}, z_k): t_{m+1/2} = (m+1/2)h_t, m = \overline{0:M-1},$

$y_{j+1/2}=(j+1/2)h_y, j=\overline{0:J-1}, J=L_y/h_y,\ z_k=k h_z, k=\overline{0:K-1}\}$ и проекция магнитного поля на y - $Hy_{j,k+1/2}^{m+1/2}$ в узлах $\{(t_{m+1/2}, y_j, z_{k+1/2}): t_{m+1/2}=(m+1/2)h_t, m=\overline{0:M-1}, y_j=(j)h_y,$ $z_{k+1/2}=(k+1/2)h_z, k=\overline{0:K-1}\}$. В предложенной области индексы j,k обозначают узлы по пространству, m - по времени. Расстояния между узлами задаются пространственными (h_y и h_z) и временным (h_t) шагами сетки. Сеточное значение диэлектрической проницаемости ($\epsilon_{j,k}$) характеризует изучаемый оптический элемент. Тогда явная разностная схема Yee для системы уравнений Максвелла, описывающая распространение TE-волны

$$\begin{aligned}
&\mu_0\frac{Hy_{j,k+1/2}^{m+1/2}-Hy_{j,k+1/2}^{m-1/2}}{h_t}=-\frac{Ex_{j,k+1}^{m}-Ex_{j,k}^{m}}{h_z},\\
&\mu_0\frac{Hz_{j+1/2,k}^{m+1/2}-Hz_{j+1/2,k}^{m-1/2}}{h_t}=\frac{Ex_{j+1,k}^{m}-Ex_{j,k}^{m}}{h_y},\\
&\epsilon\epsilon_{j,k}\frac{Ex_{j,k}^{m+1}-Ex_{j,k}^{m}}{h_t}=\frac{Hz_{j+1/2,k}^{m+1/2}-Hz_{j-1/2,k}^{m+1/2}}{h_y}-\frac{Hy_{j,k+1/2}^{m+1/2}-Hy_{j,k-1/2}^{m+1/2}}{h_z}.
\end{aligned} \qquad (6)$$

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

$$Hy^{k+1}=\alpha_1 AEx^k+Hy^k \text{ , где } A=\begin{pmatrix} -T_1 & T_1 & & & \\ & -T_2 & T_2 & & \\ & & \ddots & \ddots & \\ & & & -T_N & T_N \end{pmatrix}, \qquad (7.1)$$

$$Hz^{k+1}=\alpha_2 BEx^k+Hz^k \text{ , где } B=\begin{pmatrix} -T_1 & 0 & \dots & T_1 & & \\ & -T_2 & 0 & \dots & T_2 & \\ & & \ddots & \ddots & & \ddots \\ & & & -T_N & 0 & \dots & T_N \end{pmatrix}, \qquad (7.2)$$

где $\alpha_1=\frac{h_t}{\mu_0 h_z}, \alpha_2=\frac{h_t}{\mu_0 h_y}$ и T_i — диагональная матрица, хранящая значение 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 пересчета в середине сетки.

Как и в случае с уравнением теплопроводности мы приходим к работе с диагональными матрицами, диагонали которых состоят из коэффициентов при $Ex_{j,k+1}$ и $Ex_{j,k}$ для (9.1) и $Ex_{j+1,k}$ и $Ex_{j,k}$ для (9.2), следовательно алгоритм сводится к описанному в пункте 1.4 длинновекторному алгоритму. Третье уравнение из (6) также считается при помощи длинных векторов.

6 Вычислительные эксперименты

В качестве аппаратного обеспечения использован ПК с операционной системой Debian 7 и установленными драйверами CUDA Toolkit 5 и компилятором gcc. Также была использована библиотека CUBLAS, являющаяся аппаратно-ориентированной реализацией

программного интерфейса BLAS (Basic Linear Algebra Subprograms). Библиотека содержит уже оптимизированные реализации численных методов для выполнения на графическом устройстве. Результаты, представленные далее, получены на видеокарте GeForce GTX 660Ti.

Таблица 1. Основные характеристики GPU NVIDIA GeForce GTX 660Ti

Характеристика	Значение
Количество мультипроцессоров, шт.	28
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

Кроме того, использовался процессор Intel Core2 Duo CPU E8500.

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
Пропускная способность шины процессор-чипсет, ГБ/с	9.5
Ширина шины L2 кэша, бит	256

На рисунке 2 представлено сравнение первого варианта длинновекторного алгоритма, реализованного для двумерного уравнения теплопроводности, с коротковекторным для задач размером $N \times N$ $(N = 1..1000)$. Выигрыш алгоритма, работающего с длинными векторами обусловлен, во-первых, меньшим количеством арифметических операций, 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, не зависящую от длины вектора, а только от количества таких векторов. На рисунке 2 также приведено сравнение второго варианта алгоритма для уравнения теплопроводности с длинными векторами и коротковекторного алгоритма. В

данном случае нет явного уменьшения арифметических операций при использовании длинных векторов, а ускорение достигается за счет особенности запуска отдельных ядер библиотекой CUBLAS.

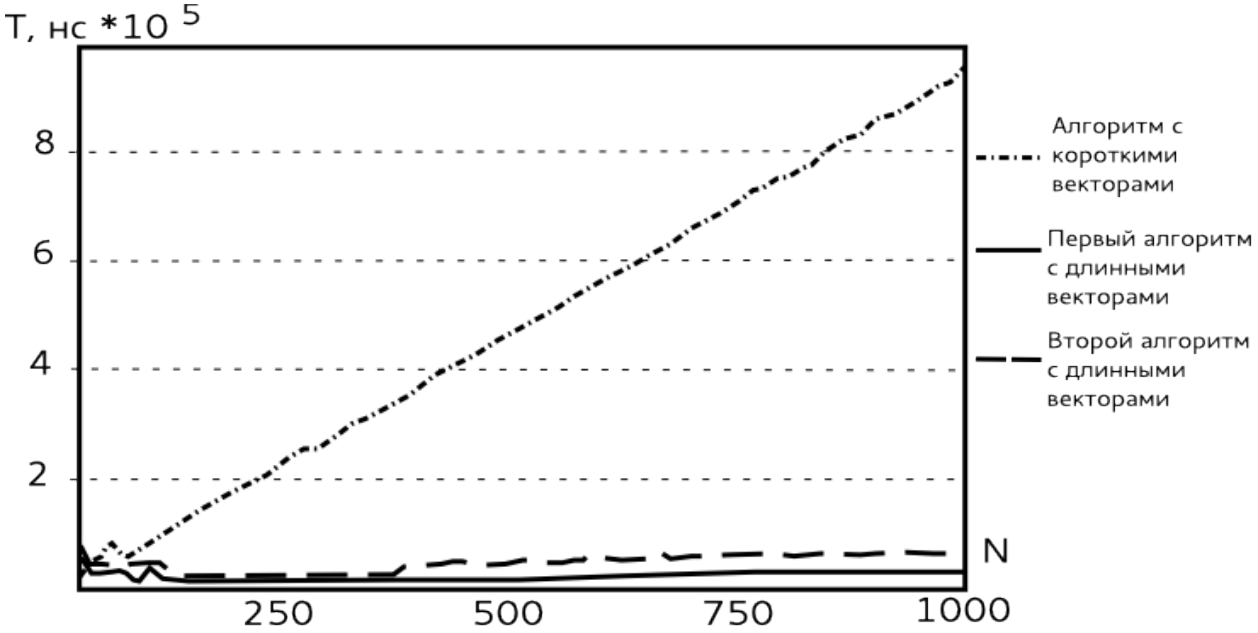

Рисунок 2 - Сравнение времени работы коротковекторного и длинновекторных алгоритмов для уравнения теплопроводности

Как видно из рис. 2 зависимость времени работы программы от размерности матрицы в случае с короткими векторами — линейная, что объясняется ростом количества арифметических операций с увеличением N . Время работы длинновекторных алгоритмов остается практически константным для векторов, которыми могут оперировать функции библиотеки CUBLAS, за счет полной загрузки микропроцессоров GPU.

На рис 3. приведено сравнение лучшего длинновекторного алгоритма для уравнения теплопроводности с реализацией на OpenCurrent, где видно, что первый алгоритм, обозначенный пунктиром, работает быстрее — его ускорение сотавило 2, но в силу особенностей самого алгоритма, он является менее универсальным методом, который мы не сможем использовать для реализации FDTD-метода.

На рисунке 4 приведен длинновекторный алгоритм для FDTD-метода и реализация этой же задачи при помощи пакета B-CALM. Длинновекторный алгоритм дает выигрыш во времени примерно в 1,4 раза.

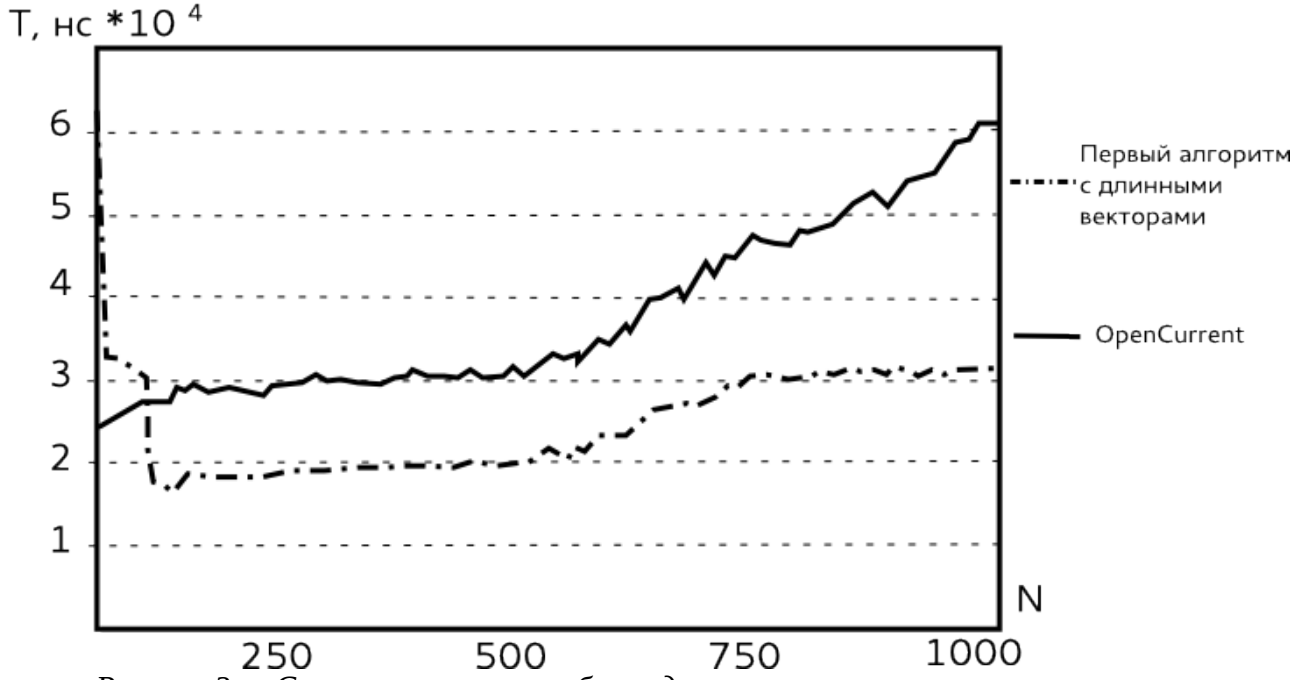

Рисунок 3 — Сравнение времени работы длинновекторного алгоритма и реализации уравнения теплопроводности при помощи OpenCurrent.

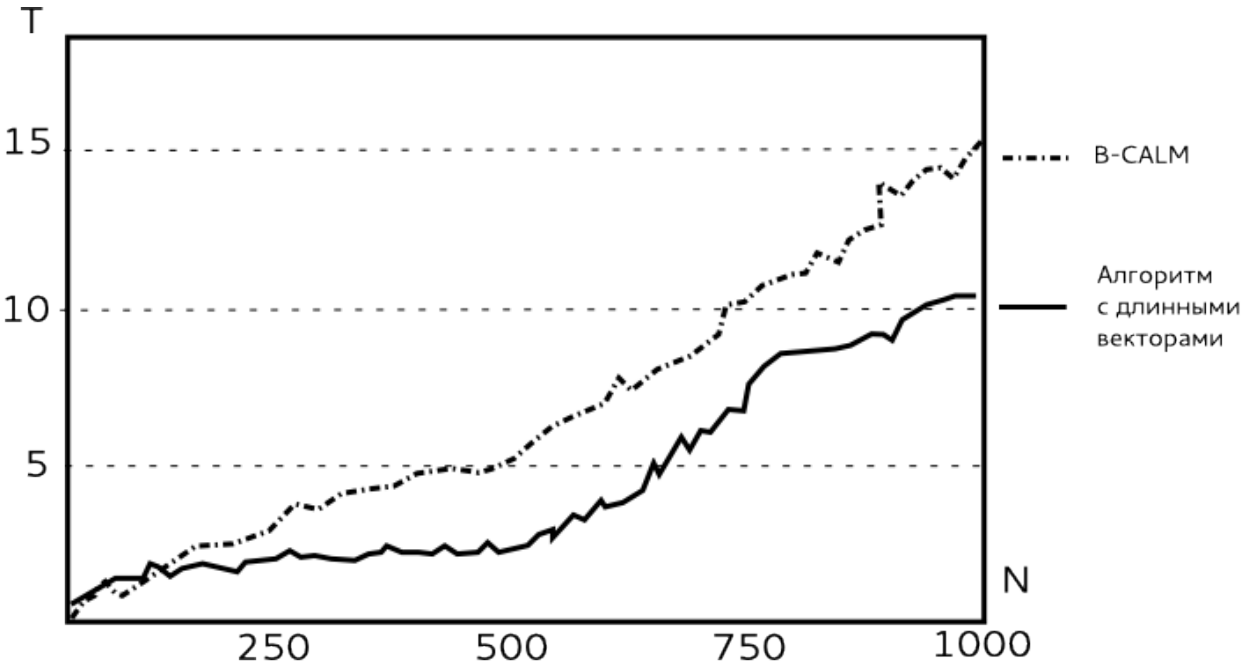

Рисунок 4 — Сравнение времени работы длинновекторного алгоритма и реализации двумерного FDTD-метода при помощи пакета B-CALM.

7 Заключение

В результате экспериментальных исследований была показана эффективность использования длинновекторных алгоритмов как для решения уравнения теплопроводности при помощи библиотеки CUBLAS, так и для простейшего варианта FDTD-метода, что

доказывает актуальность исследуемой задачи. Данный подход может быть применен для решения задач гидродинамики, аэродинамики, оптики и нанофотоники.

Благодарности

Работа выполнена при поддержке грантов РФФИ 14-01-31305мол_а «Анализ возбуждения и распространения мод лазерного излучения в оптическом волокне», 14-07-31178мол_а «Разработка методов решения обратных задач дифракционной нанофотоники на гетерогенных вычислительных системах с графическими процессорами».

Список источников:

1. Глобальный прогноз развития кремниевых технологий: http://www.russianelectronics.ru/leader-r/review/doc/64629/

2. General-purpose computing on graphics processing units: http://www.gpgpu.org/

3. Flynn MJ (1966) Very high speed computers. Proc IEEE 54:1901-1901

4. J. Ortega , Introduction to Parallel and Vector Solution of Linear Systems, Plenum Press, New York , 1987

5. Gene H. Golub, Charles F. Van Loan , Matrix Computations, JHU Press , 1996

6. Charles J. Murray. The Supermen: The Story of Seymour Cray and the Technical Wizards Behind the Supercomputer. — Wiley, 1997. — 232 p.

7. NVIDIA GeForce: http://www.nvidia.ru/object/geforce-family-ru.html

8. NVIDIA Tesla: http://www.nvidia.ru/object/tesla-high-performance-computing-ru.html

9. Самарский А. *А.* Введение в теорию разностных схем. — М.: Наука, 1971. — 552 с.

10. OpenCurrent is an open source C++ library for solving Partial Differential Equations (PDEs) over regular grids using the CUDA platform from NVIDIA: https://code.google.com/p/opencurrent/

11. Pierre Wahl, Dany-Sebastien Ly-Gagnon, Christof Debaes, David A. B. Miller, Hugo Thienpont , B-CALM: An open-source GPU-based 3D-FDTD with multi-pole dispersion for plasmonics, Optical and Quantum Electronics 44 (2012) 285-290.

12. Dimitry Golovashkin, Daria Vorotnokova, Alexander Kochurov, Svetlana Malysheva. Solving finite-difference equations for diffractive optics problems using graphics processing units, Opt. Eng. 52(9) (2013) doi: 10.1117/1.OE.52.9.091719.

13. A.V. Gavrilov, D.L. Golovashkin, L.L. Doskolovich, P.N. Dyachenko, S.N. Khonina,V.V. Kotlyar, A.A. Kovalev, A.G. Nalimov, D.V. Nesterenko, V.S. Pavelyev, Y.O. Shuyupova,R.V. Skidanov, V.A. Soifer "Diffracrive Nanophotonics", ed. by V.A. Soifer, CRC Press,Taylor&Francis Group, CISP, Boca Raton, 679p., 2014. I