РЕШЕНИЕ СЕТОЧНЫХ УРАВНЕНИЙ НЕЯВНЫХ РАЗНОСТНЫХ СХЕМ С ЦИКЛИЧЕСКИМИ КРАЕВЫМИ УСЛОВИЯМИ НА ДВУМЕРНЫХ СЕТОЧНЫХ ОБЛАСТЯХ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ГРАФИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

Головашкин Д.Л.[1,2], Логанова Л.В.[2]

[1] Институт систем обработки изображений РАН,

[2] Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет)

Аннотация

Разработаны алгоритмы метода циклической прогонки с применением технологии CUDA для реализации на одном и двух графических вычислительных устройствах. Проведённые вычислительные эксперименты продемонстрировали эффективность выбранного подхода.

Ключевые слова: CUDA, параллельные алгоритмы, трёхдиагональные СЛАУ.

Введение

Перспективным методом проведения исследований в компьютерной оптике является вычислительный эксперимент. При этом с развитием нанофотоники и нанооптики наиболее популярным инструментом для постановки таких экспериментов становятся методы решения разностных уравнений Максвелла.

Например, FDTD-метод (Finite-Difference Time-Domain method) моделирования распространения электромагнитного излучения в рамках строгой теории дифракции [1 – 3]. Характеризуясь универсальностью, метод известен высокой вычислительной сложностью. Так, при исследовании алмазных плёнок возникает необходимость в задании сеточной области протяжённостью до нескольких миллионов узлов, т.к. характерные неоднородности в поликристаллической структуре алмаза, являющиеся центрами когерентной эмиссии электронов, являются наноразмерными (около 1 н.м), а линейные размеры алмазной плёнки достигают десятка сантиметров.

Также среди методов решения уравнений Максвелла широкое распространение получили методы, основывающиеся на численном решении уравнений Гельмгольца. Они находят применение в волоконной и интегральной оптике при моделировании процессов в оптических волноводах и модовом анализе распространяющихся пучков. Одним из таких методов, позволяющих численно решить уравнение Гельмгольца, является метод распространяющегося пучка [4].

Исследование физических процессов на более длительных временных и более обширных пространственных областях стало возможным благодаря применению параллельных вычислений в математическом моделировании. Известная монография Wenhua Yu и др. [5] подытоживает многолетнюю практику составления параллельных алгоритмов FDTD-метода. На сайте NVidia размещены предложения реализации FDTD-метода с помощью GPU (например, FDTD solvers от компании Acceleware). Вычислительная сложность метода распространяющегося пучка подразумевает необходимость применения высокопроизводительных вычислений в расчётах (например, программный пакет OlympIos [6]). Однако нерассмотренными остались параллельные алгоритмы для разностных схем Zheng, Chen, Zhang [1 – 2].

Традиционно решение неявных разностных уравнений связано с методом прогонки, который по сравнению с методами циклической редукции и декомпозиции области характеризуется меньшим объёмом коммуникаций и арифметических операций. При этом ограничение на масштабируемость алгоритма может быть снято в случае многомерной области [7, 8]. Особенностью предлагаемой работы является постановка периодических граничных условий, необходимых, например, при моделировании тепловых процессов в периодических оптических элементах (дифракционных решётках и фотонных кристаллах) [9-11].

Известны эффективные параллельные реализации алгоритма циклической прогонки на кластерных вычислительных системах [12]. Однако данный тип систем характеризуется повышенными, хотя и оправданными, расходами на поддержку и установку и требует специально обученного персонала.

В последнее время возрос интерес к параллельным вычислениям на графических процессорных устройствах (GPU) NVIDIA и AMD в связи с их высокой производительностью, низким энергопотреблением и относительно низкой стоимостью. Современное графическое процессорное устройство (GPU) способно выполнять большое количество арифметических операций параллельно. К тому же программно-аппаратная архитектура NVIDIA CUDA позволяет использовать для программирования приложений на GPU стандартный язык программирования «C» с расширениями [13].

В данной работе представлены реализации алгоритмов с применением технологии CUDA. Вычисления по алгоритму выполняются на гетерогенной (гибридной) вычислительной системе, когда в вычислениях участвуют центральный процессор (ЦПУ) и GPU. До сих пор общепринятым подходом являлась организация вычислений на одном GPU (14). В настоящей работе авторы проводят вычисления на двух графических процессорных устройствах.

1. Решение сеточных уравнений методом циклической прогонки

Положим, что при решении классической неявной разностной схемы [15] двумерной задачи теплопроводности методом покоординатного расщепле-

ния используется следующая двумерная сеточная область $w_h = w_{h_x} x w_{h_y}$, где

$$w_{h_x} = \{x(i) = ih_x,\ i = 0,1,...,M-1,$$
$$t = k\tau, k = 0,1,...,K\},$$

а

$$w_{h_y} = \{y(j) = jh_y,\ j = 0,1,\ ...,N-1,$$
$$t = k\tau, k = 0,1,\ ...,K\}.$$

Переход от k-слоя к слою k+1 по времени осуществляем следующим образом:

$$\frac{(\overline{U}_{m,n} - U^k_{m,n})}{\tau} -$$
$$-\frac{\mu_1(\overline{U}^k_{m+1,n} - 2\overline{U}^k_{m,n} + \overline{U}^k_{m-1,n})}{h_x^2} = \frac{1}{2} f^k_{m,n},$$

$n = 1,2,...,N-1, m = 1,2,...,M-1$, с циклическими граничными условиями $U^0_{0,m} = U^0_{M,n}$;

$$\frac{(U^{k+1}_{m,n} - \overline{U}_{m,n})}{\tau} -$$
$$-\frac{\mu_2(U^{k+1}_{m,n+1} - 2U^{k+1}_{m,n} + U^{k+1}_{m,n-1})}{h_y^2} = \frac{1}{2} f^{k+1}_{m,n},$$

$n = 1,2,...,N-1, m = 1,2,...,M-1$, с циклическими граничными условиями $U^0_{m,o} = U^0_{m,N}$; $U^k_{m,n}$ – значение сеточной функции в узле (mh_x, nh_y) на k-м временном слое, $\overline{U}_{m,n}$ – изменение U на временном интервале τ за счёт теплопроводности только по направлению x.

На w_h задана сеточная функция $g_{ij} = g(ih_x, jh_y)$.

После составления разностной схемы (расщепления по пространственным переменным) необходимо найти решение совокупности систем линейных алгебраических уравнений (СЛАУ): каждой строке и столбцу сеточной области w_h соответствует одна система ленточного вида. Например, разностная краевая задача из [15]:

$$\begin{cases} -a_0 y_{M-1} + c_0 y_0 - b_0 y_1 = f_0, \\ -a_i y_{i-1} + c_i y_i - b_i y_{i+1} = f_i, 1 < i < M-1, \\ y_0 = y_M. \end{cases}$$

В целом, решение всех разностных уравнений схемы (на одном временном слое, если схема нестационарная) связано с решением M уравнений по N строкам w_h и N уравнений по M столбцам:

$$A_k y_k = b_k, \quad (1)$$

где $A_k \in R^{MXM}$, $y_k, b_k \in R^M$, $y_k = (g_{ko}, ..., g_{kM-1})$, $k = 0: N-1$, $a_{ij} \neq 0$, для $i \le j+1, j \le i+1$, а также $a_{1M} \le 0$ и $a_{M1} \le 0$.

$$A_l y_l = b_l, \quad (2)$$

где $A_l \in R^{NXN}$, $y_l, b_l \in R^N$, $y_l = (g_{ok}, ... g_{N-1k})T$, $l = 0: M-1$, $a_{ij} \neq 0$, для $i \le j+1, j \le i+1$, а также $a_{1N} \neq 0$ и $a_{N1} \neq 0$.

Для решения СЛАУ (1) обратимся к методу циклической прогонки [15]. Решение представим в виде линейной комбинации промежуточных сеточных функций (u_i, v_i):

$$y_i = u_i + y_0 v_i,\ 0 < i < M-1. \quad (3)$$

При этом u_i и v_i есть решение следующих систем трёхточечных уравнений:

$$\begin{cases} -a_i u_{i-1} + c_i u_i - b_i u_{i+1} = f_i, & 1 < i < M-1 \\ u_0 = 0, \ u_M = 0 \end{cases},$$

$$\begin{cases} -a_i v_{i-1} + c_i v_i - b_i v_{i+1} = 0, & 1 < i < M-1 \\ v_0 = 1, \ v_M = 1 \end{cases}.$$

Нахождение прогоночных коэффициентов α, β, γ для $2 < i < M-1$ выполним в соответствии с методом:

$$\alpha_2 = \frac{a_{23}}{a_{22}},\ \beta_2 = \frac{b_1}{a_{22}},\ \gamma = \frac{a_{21}}{a_{22}}, \quad (4)$$

$$\alpha_{i+1} = \frac{a_{i+1,i+2}}{a_{i+1,i+1} - a_{i+1,i}\alpha_i},\ \beta_{i+1} = \frac{(b_i + a_{i+1,i}\beta_i)}{a_{i+1,i+1} - a_{i+1,i}\alpha_i},$$
$$\gamma_{i+1} = \frac{a_{i+1,i}\gamma_i}{a_{i+1,i+1} - a_{i+1,i}\alpha_i},\ i = \overline{2, M-1}. \quad (5)$$

Найдём u_i, v_i, y_0 согласно формулам, после чего значения искомой сеточной функции (y_i) вычислим по формуле (3).

$$u_{M-1} = \beta_M,\ v_{M-1} = \alpha_M \gamma_M,\ v_M = 1, u_M = 0, \quad (6)$$

$$u_i = \alpha_{i+1} u_{i+1} + \beta_{i+1},$$
$$v_i = \alpha_{i+1} v_{i+1} + \gamma_{i+1}, i = M-2,...1, \quad (7)$$

$$y_0 = \frac{b_0 + a_{1M} u_{M-1} + a_{12} u_1}{a_{11} - a_{1M} v_{M-1} - a_{12} v_1}. \quad (8)$$

СЛАУ (2) решается аналогично.

2. Алгоритм циклической прогонки для одного GPU

Будем рассматривать GPU как устройство, обладающее собственной памятью, содержащее ряд потоковых мультипроцессоров, каждый из которых способен одновременно выполнять некоторое количество нитей. Каждый мультипроцессор работает независимо от остальных, и фактически GPU способен выполнять одни и те же операции над многими данными одновременно.

Возможность одновременного решения нескольких СЛАУ (1) или (2) обеспечивает параллельность предлагаемого алгоритма.

Т.к. решение всех разностных уравнений схемы связано с решением N СЛАУ (1) и M СЛАУ(2), то первоначально вычисления по алгоритму были организованы таким образом, что каждая нить вызываемого единственного ядра выполняла решение одной СЛАУ в соответствии с формулами метода циклической прогонки [15]. Данная реализация оказалась неэффективной ввиду необходимости пересылки и хранения больших объёмов данных

($10\times M\times N$, где M, N – размеры сеточной области) и невозможности полной загрузки нити в связи с ограничениями на память GPU. Поэтому была выполнена декомпозиция вычислений, т.е. алгоритм циклической прогонки разделили на 3 части, каждая из которых соответствовала отдельному ядру. Последний позволяет выделить следующие части: определение прогоночных коэффициентов, нахождение значений промежуточных сеточных функций и вычисление значений искомых сеточных функций. В результате такого подхода удалось снизить требования к памяти GPU, уменьшить объём пересылаемых и хранимых данных относительно объёма производимых вычислений на $3\times M\times N$. В этом случае алгоритм 1 можно представить следующим образом:

1. Задание характеристик ядра. Передача данных на GPU ($a, b, c, d \in RM$).
2. Вызов ядра. Вычисление прогоночных коэффициентов $\alpha_i, \beta_i, \gamma_i$ (4, 5).
3. Освобождение памяти GPU от a,b,c,d.
4. Вызов ядра. Нахождение векторов значений промежуточных сеточных функций u, v (6, 7).
5. Освобождение памяти GPU от векторов α, β, γ. Передача векторов значений промежуточных сеточных функций u, v на ЦПУ.
6. Вычисление y_0 на ЦПУ (8). Передача их на GPU.
7. Вызов ядра. Определение искомого вектора значений y и пересылка его на ЦПУ(3).

При этом наличие пунктов 6,7 отличает предложенный алгоритм от алгоритма обычной прогонки и обеспечивает возможность решения сеточных уравнений неявных разностных схем с циклическими граничными условиями.

Далее представлен фрагмент программы, содержащий ядро для определения прогоночных коэффициентов в соответствии с алгоритмом циклической прогонки (п. 2 из описания алгоритма).

```
//Ядро выполняется на N нитях параллельно
__global__ void pr ( float * a, float* b, float* c, float* d,
float* alfa, float* betta, float* gamma,int m, int n)

{
float zn=0.0f;

//Глобальный индекс нити

int idx=(blockIdx.x*blockDim.x) + threadIdx.x;

//Определение прогоночных коэффициентов α,β,γ

 alfa[idx+n]=b[idx+n]/c[idx+n];
  betta[idx+n]=d[idx+n]/c[idx+n];
  gamma[idx+n]=a[idx+n]/c[idx+n];

 for (int i=2; i<m-1; i++)
  {
    zn=c[idx+n*i]-a[idx+n*i]*alfa[idx+n*(i-1)];
    alfa[idx+n*i]=b[idx+n*i]/zn;
    betta[idx+n*i]=(d[idx+n*i]+a[idx+n*i]*betta[idx+n*(i-
1)])/zn;
    gamma[idx+n*i]=a[idx+n*i]*gamma[idx+n*(i-1)]/zn;
                }

}
```

Ядра, определяющие значения промежуточных (п. 4 алгоритма) и искомых сеточных функций (п. 7 алгоритма), реализованы в соответствии с формулами (8)–(9) аналогично. Реализации данного алгоритма были выполнены с использованием глобальной и разделяемой памяти GPU и позволили значительно повысить скорость выполнения вычислений без дополнительных изменений и в алгоритме, и в структуре программы. Оптимизация работы с глобальной памятью производилась путём объединения нескольких запросов в один (coalescing). Её эффективность далее будет подтверждена экспериментально.

Предварительная оценка прироста производительности при реализации алгоритма с использованием GPU позволяет говорить о целесообразности его разработки. Так, общее количество арифметических операций (не делая различия между ними) для применяемого метода составляет $\approx 2\times 14\times N\times M$ [14] для двумерной сеточной области. Благодаря широкому использованию укрупнённого (массивного) векторного параллелизма, характерного для технологии CUDA, предложенная реализация в случае, если бы ограничения на объём используемой памяти были сняты, позволила бы ускорить вычисления в $N-t_{load}$ раз при выполнении прогонок по строкам и $M-t_{load}$, где t_{load} – время пересылки данных на GPU. Учитывая организацию вычислений на GPU, естественно предположить, что время вычислений на GPU до некоторого момента будет больше времени вычислений на ЦПУ вследствие превышения времени пересылки данных над временем выполнения вычислений. Далее произойдёт прирост производительности.

3. Алгоритм циклической прогонки для двух GPU

Для использования нескольких GPU на одном вычислительном узле для выполнения вычислений необходимо одновременно запустить несколько потоков ЦПУ. Каждый из них выделит данные, обрабатываемые им, и выберет GPU для вычислений. Передача данных потокам ЦПУ реализуется средствами OpenMP [16]. На рис. 1 представлена схема вычислений по алгоритму.

В общем, описание алгоритма 2 можно представить следующим образом:

1. Средствами OpenMP запускается два потока. Из двух запущенных потоков поток 0 выбирает GPU 0, поток 1 выбирает GPU 1.
2. Каждый GPU производит вычисления по алгоритму для половины строк, т.е первый обрабатывает строки с 1 по $N/2$, второй – с $N/2+1$ по N. При этом каждый GPU выделяет

и освобождает память под выбранные данные, вызывает ядра определения прогоночных коэффициентов (4)–(5), нахождения значений промежуточных (6)–(8) и значений искомых сеточных функций (3).

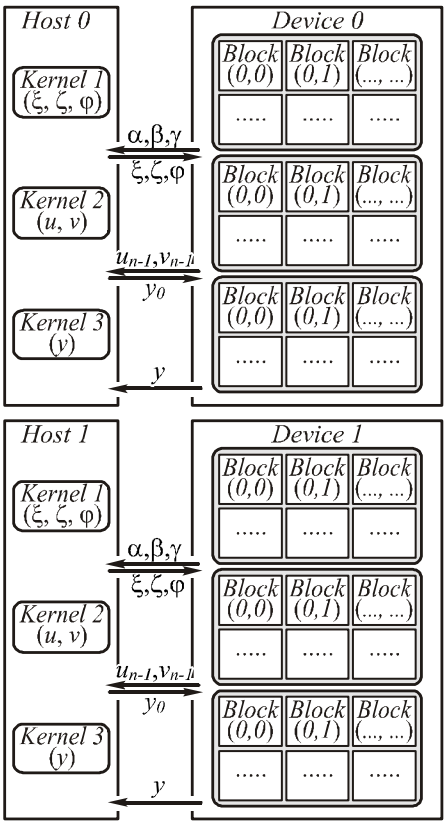

Рис. 1. Поток 0 и GPU 0 выполняют вычисления для строк с 1 по N/2, поток 1 и GPU 1 – для строк N/2+1 по N при циклических прогонках по строкам

Аналогично организуются циклические прогонки по столбцам. Обмен данными между GPU не требуется.

В этом случае объём вычислений на каждом GPU составит $\approx 14{\times}N{\times}M$. Таким образом, организация вычислений позволит рассматривать сеточные области с большими значениями N при циклических прогонках по строкам и больших значениях M при прогонках по столбцам. В данном случае предварительная оценка производительности предполагает прирост в 2 раза относительно алгоритма, рассмотренного выше.

4. Вычислительные эксперименты

Для подтверждения эффективности предложенных алгоритмов авторами были проведены вычислительные эксперименты, связанные с решением классической неявной разностной схемы двумерной задачи теплопроводности методом покоординатного расщепления. При этом размеры рассматриваемой сеточной области ($M{\times}N$) обусловлены ограничениями на память GPU. К тому же при выборе M учитывалась необходимость полной или близкой к полной загрузки каждой нити.

Исследования алгоритмов проводились на гибридной вычислительной системе с двумя видеокартами GeForce GTX 470, CPU Intel Core 2 Duo E8500 3,16 ГГц, операционной системой Microsoft Windows XP с установленным драйвером NVIDIA CUDA Version 2.3. Обмен данными между потоками ЦПУ реализуется средствами OpenMP, между памятью GPU и ЦПУ – осуществляется соответствующими функциями CUDA Runtime API [17] через PCI-Express ×16, скорость передачи которого 4 Гб/с. Предложенные алгоритмы реализованы средствами программного пакета Microsoft Visual Studio 2008.

Одним из возможных способов повышения скорости выполнения вычислительного процесса является оптимизация работы с глобальной памятью, которая позволяет хранить большие объёмы данных для обработки ядрами и сохраняет свои значения между вызовами ядер. При работе с глобальной памятью, обладающей высокой латентностью, для повышения скорости доступа к ней очень важной является возможность объединения нескольких запросов. Расположим последовательно элементы векторов, необходимых каждой нити в некоторый момент. Например, для вектора коэффициентов a, расположенных под главной диагональю исходной трёхдиагональной матрицы коэффициентов, порядок следования элементов будет следующим $a_{10}, a_{20}, \ldots, a_{N0}, \ldots; a_{11}, a_{21}, \ldots, a_{N1}, \ldots, a_{1M}, a_{2M}, \ldots, a_{NM}$, где первый индекс соответствует номеру строки (системы) при прогонках по строкам (номеру столбца при прогонках по столбцам), второй индекс – номеру уравнения в системе.

На рис. 2 приведены графики ускорений вычислительных процессов, порождённых алгоритмом 1, с оптимизацией доступа к памяти и без неё. Выбранный диапазон изменения значений N определяется ограничением на GPU для $M = 500$.

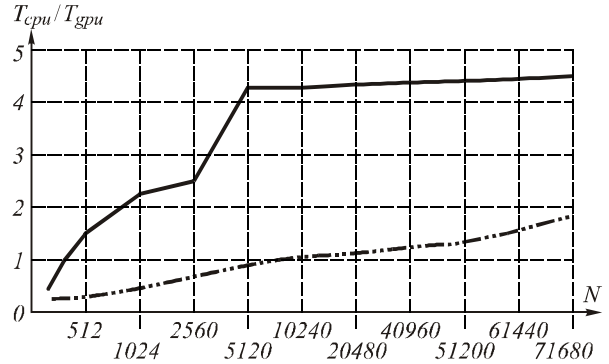

Рис. 2. Зависимость ускорения вычислительного процесса, выполненного на GPU, относительно выполнения его на ЦПУ от размеров сеточной области M=500 (непрерывная линия – после выполнения оптимизации доступа к памяти, штрихпунктирная – доступ к памяти не оптимизирован)

Из графиков видно, что имеет место общая тенденция роста ускорения с увеличением N, что объясняется параллельностью исполнения вычислительных операций. При этом ускорение меньше 1 свиде-

тельствует о том, что для соответствующих значений N время пересылки данных превышает время выполнения вычислительного процесса. Предложенная организация хранения данных заметно сказалась на скорости вычислений (в 4 раза).

Далее авторами исследовалась зависимость ускорения вычислительного процесса, выполненного на одном GPU, относительно выполнения его на ЦПУ от размеров сеточной области. Загрузка каждой нити определялась значением M, число нитей – значением N.

Результаты исследований представлены на рис. 3.

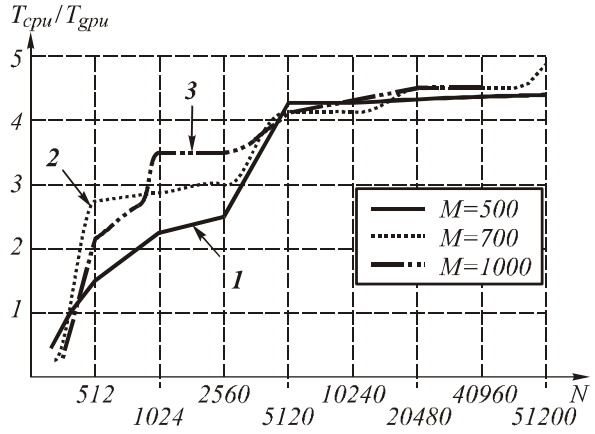

Рис. 3. Зависимость ускорения реализации алгоритма на GPU от размеров сеточной области

Результаты, представленные на графиках, свидетельствуют о том, что сначала рост скорости вычислений зависит и от M, и от N. Однако по достижении некоторого значения $M=7680$ изменения скорости вычислений в основном связаны с изменением N. Следовательно, можно говорить о зависимости скорости вычислений от объёма вычислений, производимых каждой нитью ядра и всеми нитями вместе. При этом зависимость ускорения от M можно проследить, скорее всего, до того момента, пока нить не окажется полностью загруженной.

Продолжая исследовать зависимость ускорения от размеров сеточной области, были рассмотрены случаи, когда значения N фиксировались, а M изменялись. На рис. 4 показаны графики этих зависимостей.

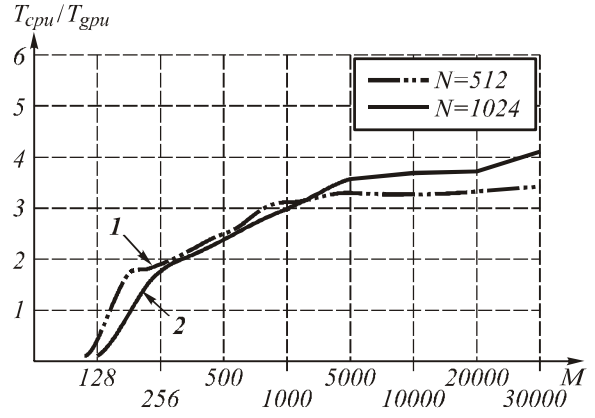

Рис. 4. Зависимость ускорения реализации алгоритма на GPU от размеров сеточной области

Из графиков видно, что до некоторого момента происходит рост ускорения, это связано с ростом загрузки нити. Как уже говорилось ранее, ускорение менее 1 объясняется превышением времени пересылки данных на GPU над временем выполнения вычислений. Сравнивая графики рис. 3 и 4, можно сделать вывод: ускорение выше для сеточных областей, размеры которых характеризуются большими значениями N по сравнению с M. Это объясняется тем, что, обеспечив полную или близкую к полной загрузку нити, на скорость вычислительного процесса можно влиять только изменяя N.

Исследования зависимостей ускорения вычислительного процесса, порождённого алгоритмом циклической прогонки, выполненного на двух GPU, от размеров сеточной области привели к результатам, аналогичным выполнению соответствующих вычислений на одном GPU. В этом случае рост ускорения также отмечался с ростом объёма производимых вычислений. А когда нить оказывалась полностью загруженной, рост ускорения происходил с увеличением N.

Графики, представленные на рис. 5, позволяют сравнить ускорения вычислительных процессов, порождённых алгоритмом циклической прогонки и выполненных на одном и двух GPU.

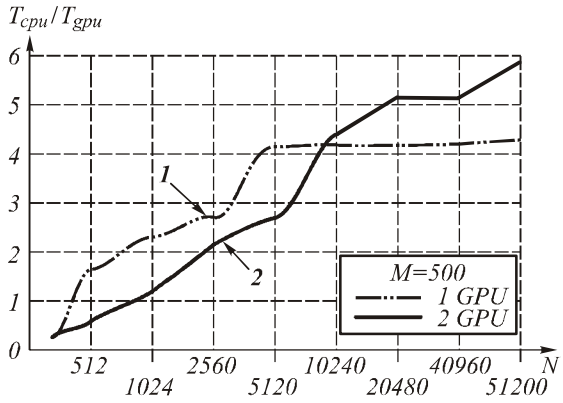

Рис. 5. Зависимость ускорения реализации алгоритма циклической прогонки от размеров сеточной области

Сначала ускорение вычислительного процесса, выполняемого на одном GPU, выше, чем выполняемого на двух GPU. Это объясняется тем, что время распределения данных превышает время их обработки в случае проведения расчётов на двух GPU. Начиная с $N=7680$, ускорение вычислительного процесса, выполненного на двух GPU, становится больше. При этом заметно увеличение разницы между ними с ростом N. Т.е. использование двух GPU становится целесообразным с $N=7680$. Для заданного диапазона разница значений ускорений составила 1,3 раза.

Предложенные реализации на одном и двух GPU алгоритма циклической прогонки позволили значительно уменьшить время выполнения вычислений несмотря на то, что прямые методы с узкой лентой эффективно не векторизуются [18]. В случае, когда размеры сеточной области превышают ограничения на память GPU, следует применять так называемые

гибридные алгоритмы, позволяющие провести декомпозицию одной системы или сократить число систем, решаемых одной нитью [19].

Заключение

В настоящей работе предложены реализации алгоритмов циклической прогонки для выполнения порождённого ими вычислительного процесса на гетерогенной вычислительной системе, состоящей из одного ЦПУ и двух графических вычислительных устройств. Проведены исследования зависимости ускорения выполнения вычислительного процесса от загрузки каждой нити и их количества. Полученные в результате проведённых вычислительных экспериментов ускорения позволяют говорить об эффективности алгоритмов. Максимальное ускорение равно 6. Интерес, по мнению авторов, представляют исследования, связанные с реализацией алгоритма с применением нескольких вычислительных узлов с несколькими GPU.

Благодарности

Исследования выполнены при поддержке грантов Президента РФ МД-6809.2012.9, НШ-4128.2012.9 и РФФИ № 10-07-00453-а, № 10-07-00533-а.

Литература

1. **Taflove, A.** Computational Electrodynamics: The Finite-Difference Time-Domain Method / A. Taflove, S. Hagness. 3-nd. ed. – Boston: Arthech House Publishers, 2005. – 852 p.
2. **Inan, U.S.** Numerical Electromagnetics: The FDTD Method / Umran S. Inan, Robert A. Marshall. – New York: Cambridge University Press, 2011. – 390 p.
3. **Rouf, H.K.** Implicit Finite Difference Time Domain Methods. Theory and Applications / Hasan Khaled Rouf. – LAP Lambert Academic, 2011. – 208 p.
4. **Гаврилов, А.В.** Модифицированный метод распространяющегося пучка и его применение к расчёту распространения в волноводах с изменяющимся профилем показателя преломления // Компьютерная оптика. – 2008. – Т. 32, № 1. – С. 15-22.
5. **Yu, W.** Parallel finite-difference time-domain method. / Wenhua Yu, Raj Mittra, Tao Su, Yongjun Liu, Xiaoling Yang. – Artech House electromagnetic analysis series, 2006. – 274 p.
6. OlympIOs: design, simulation and mask layout platform //URL: http://www.c2v.nl/products/software/olympios-software.shtml.
7. **Елизарова, Т.Г.** Применение многопроцессорных транспьютерных систем для решения задач математической физики / Т.Г. Елизарова, Б.Н. Четверушкин //Математическое моделирование. – 1992. – Т. 4, № 11. – С. 75-100.
8. **Головашкин, Д.Л.** Параллельные алгоритмы решения сеточных уравнений трёхдиагонального вида, основанного на методе встречных прогонок / Д.Л. Головашкин // Математическое моделирование. – 2005. – Т. 17, № 11. – С. 118-128.
9. **Tsukerman, I.** Computational Methods for Nanoscale Applications. Particles, Plasmons and Waves / Igor Tsukerman. – Springer Science, 2005. – 530 p.
10. **Шмаков, В.А.** Силовая оптика / В.А. Шмаков; отв.ред. В.И.Савин. – М.: Наука. 2004. – 418 с.
11. **Самарский, А.А.,** Вычислительная передача / А.А. Самарский, П.Н. Вабищевич. – М.: Едиториал УРСС,2003. – 784 с.
12. **Логанова, Л.В.** Параллельный алгоритм реализации метода встречных циклических прогонок для двумерных сеточных областей / Л.В. Логанова, Д.Л. Головашкин // Вычислительные технологии. – 2011. – Т. 16, № 4. – С. 64-71.
13. NVIDIA CUDA. Version 3.0. Reference Manual. February, 2010.
14. **Алексеев, В.А.** Векторизация метода распространяющегося пучка и его реализация по технологии CUDA / В.А. Алексеев, Д.Л. Головашкин // Компьютерная оптика. – 2010. – Т. 34, № 2. – С. 225-229.
15. **Самарский, А.А.** Методы решения сеточных уравнений / А.А. Самарский, Е.С. Николаев. – М.: Наука, 1978, – 561 с.
16. **Антонов, А.С.** Параллельное программирование с использованием технологии OpenMP: учеб. пособие / А.С. Антонов. – М.: Изд-во МГУ, 2009. – 77 с.
17. **Боресков, А.В.** Основы работы с технологией CUDA. / А.В. Боресков, А.А. Харламов. – М.: ДМК Пресс, 2010. – 232 с.
18. **Ортега, Дж.** Введение в параллельные и векторные методы решения линейных систем. / Дж. Ортега; пер. с англ. – М.: Мир, 1991. – 367 с.
19. **Andrew Davidson, Yao Zhang, John D. Owens** An Auto-tuned Method for Solving Large Tridiagonal Systems on the GPU Published in: IPDPS '11: Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium IPDPS '11 Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium Pages 956-965 IEEE Computer Society Washington, DC, USA ©2011.

References

1. **Taflove, A.** Computational Electrodynamics: The Finite-Difference Time-Domain Method / A. Taflove, S. Hagness. 3-nd. ed. – Boston: Arthech House Publishers, 2005. – 852 p.
2. **Inan, U.S.** Numerical Electromagnetics: The FDTD Method / Umran S. Inan, Robert A. Marshall. – New York: Cambridge University Press, 2011. – 390 p.
3. **Rouf, H.K.** Implicit Finite Difference Time Domain Methods. Theory and Applications / Hasan Khaled Rouf. – LAP Lambert Academic, 2011. – 208 p.
4. **Gavrilov, A.V.** The modified method of beam propagation and its application to distribution calculation in waveguides with a changing profile of an indicator of refraction / A.V. Gavrilov // Computer Optics. – 2008. – V. 32, N 1. – P. 15-22. – (In Russian).
5. **Yu, W.** Parallel finite-difference time-domain method. / Wenhua Yu, Raj Mittra, Tao Su, Yongjun Liu, Xiaoling Yang. – Artech House electromagnetic analysis series, 2006. – 274 p.
6. OlympIOs: design, simulation and mask layout platform //URL: http://www.c2v.nl/products/software/olympios-software.shtml.
7. **Elizarova, T.G.** The use of multi-processor transputer system for solving mathematical physics / T.G. Elizarova B.N. Chetverushkin // Mathematical modelling. – 1992. – V. 4, N 11. – P. 75-100. – (In Russian).
8. **Golovashkin, D.L.** Parallel algorithms of the grid equations decision of the three-diagonal kind based on a method of counter proraces / D.L. Golovashkin // Mathematical modelling. – 2005. – V. 17, N 11. – P. 118-128. – (In Russian).

9. **Tsukerman, I.** Computational Methods for Nanoscale Applications. Particles, Plasmons and Waves / Igor Tsukerman. – Springer Science, 2005. – 530 p.
10. **Shmakov, V.A.** Power Optics / VA. Shmakov; otv.red. V.I. Savin. - Moscow: "Nauka" Publisher, 2004. - 418 p. – (In Russian).
11. **Samarskiy, A.A.** Computer gear / A.A. Samarskiy, P.N. Vabishchevich. - Moscow: Editorial URSS, 2003. - 784 p. – (In Russian).
12. **Loganova, L.V.** A parallel algorithms in the cyclic counter-sweep method two-dimensional grid domains / L.V. Loganova, D.L. Golovashkin // Computational Technology. – 2011. – V. 16, N 4. – P. 64-71. – (In Russian).
13. NVIDIA CUDA. Version 3.0. Reference Manual. February, 2010.
14. **Alekseev, V.A.** Vector beam propagation method and its implementation technology CUDA / V.A. Alekseev, D.L. Golovashkin // Computer Optics. - 2010. – V. 34, № 2. – P. 225-229. – (In Russian).
15. **Samarskiy, A.A.** Methods of the decision of the grid equations / A.A. Samarskiy, E.S. Nilolayev. – Moscow: "Nauka" Publisher, 1978. – 561 p. – (In Russian).
16. **Antonov, A.S.** Parallel programming using technology OpenMP: Textbook / A.S. Antonov. – Moscow: MSU publishing house, 2009. – 77 p.– (In Russian).
17. **Boreskov, A.V.** Technology basics CUDA / A.V. Boreskov, A. Kharlamov. – Moscow: DMK Press Publisher, 2010. – 232 p. – (In Russian).
18. **Ortega, Dg.M.** Introduction in parallel and vector methods of linear systems decision / Dgaims M. Ortega, translate from English H.D. Ikramova, I.E. Kaporina; edited by H.D. Ikramova. – Moscow: "Mir" Publisher, 1991. – 364 p. – (In Russian).
19. **Andrew Davidson, Yao Zhang, John D. Owens** An Auto-tuned Method for Solving Large Tridiagonal Systems on the GPU Published in: IPDPS '11: Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium IPDPS '11 Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium Pages 956-965 IEEE Computer Society Washington, DC, USA ©2011.

SOLUTION OF DIFFERENCE EQUATIONS DIFFERENCE SCHEME WITH CYCLIC BOUNDARY CONDITIONS ON TWO-DIMENSIONAL GRID AREAS USING MULTIPLE GRAPHICS COMPUTING DEVICES

D.L. Golovashkin[1,2], L.V. Loganova[2]
[1] Image Processing Systems Institute of the RAS,
[2] S.P. Korolyov Samara State Aerospace University (National Research University)

Abstract

The algorithms using cyclic sweep using CUDA technology to implement in one and two graphical computing devices. Of computational experiments have demonstrated the effectiveness of the approach.

Key Words: CUDA, parallel algorithms, tridiagonal systems.

Сведения об авторах

Сведения об авторе ***Головашкин Димитрий Львович*** *– см. стр.533 этого номера.*

Логанова Лилия Владимировна, старший преподаватель кафедры технической кибернетики Самарского государственного аэрокосмического университета. Область научных интересов: векторные и параллельные алгоритмы решения СЛАУ.

E-mail: *lloganova@yandex.ru* .

Liliya Vladimirovna Loganova, S.P. Korolyov Samara State Aerospace University, senior lecturer at the Technical Cybernetics sub-department. Scientific interests: vector and parallel algorithms for matrix computation.

Поступила в редакцию 7 сентября 2012 г.