ВЕСТНИК ЮГОРСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2022 г. Выпуск 2 (65) С. 104–112

УДК 536.2
DOI: 10.18822/byusu202202104-112

РЕШЕНИЕ ЗАДАЧИ ТЕПЛОПРОВОДНОСТИ В ДВУХМЕРНОЙ ПОСТАНОВКЕ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Сеченов Павел Александрович
кандидат технических наук
доцент кафедры прикладных информационных технологий и программирования,
ФГБОУ ВО «Сибирский государственный индустриальный университет»
Новокузнецк, Россия
E-mail: pavesa89@mail.ru

Предмет: технология и алгоритмы параллельного программирования.

Цель: сравнение скорости выполнения последовательного алгоритма на центральном процессоре с параллельным алгоритмом на графическом процессоре при решении задачи двухмерной теплопроводности.

Методы: модифицированный автором метод Кранка – Николсона для решения задачи двухмерной теплопроводности на графическом процессоре.

Результаты исследования: 1. Решение двухмерной задачи теплопроводности по схеме Кранка – Николсона не является абсолютно параллельной, и максимально возможное ускорение не достигается. 2. При размерности матрицы 16 на 16 и одинарной точности время выполнения на графическом процессоре оказалось в 1,32 – 1,72 быстрее, чем на центральном процессоре. При размерности матрицы 32 на 32 время выполнения на графическом процессоре оказалось в 3,66 – 6,07 быстрее, чем на центральном процессоре. 3. При вычислениях с двойной точностью наблюдается наибольшее ускорение в 71,89 при 10 итерациях расчета, если итераций более 104, то ускорение в расчетах на графическом процессоре с двойной точностью приближается к расчетам с одинарной точностью.

Ключевые слова: параллельное программирование, графический процессор, метод Кранка – Николсона, уравнение теплопроводности, быстродействие.

SOLUTION TO THE PROBLEM OF THERMAL CONDUCTIVITY IN A TWO-DIMENSIONAL FORMULATION ON A GRAPHICS PROCESSING UNIT USING PARALLEL COMPUTING

Sechenov Pavel Aleksandrovich
Candidate of technical sciences
Associate Professor of the Department of Applied Information Technologies and Programming,
Siberian State Industrial University
Novokuznetsk, Russia
E-mail: pavesa89@mail.ru

Subject: technology and algorithms of parallel programming.

Objective: to compare the execution speed of a sequential algorithm on a central processor with a parallel algorithm on a graphics processor when solving a two-dimensional thermal conductivity problem.

Methods: the Crank – Nicholson method modified by the author for solving the problem of two-dimensional thermal conductivity on a graphics processor.

Research results: 1. The solution of the two-dimensional thermal conductivity problem according to the Crank – Nicholson scheme is not absolutely parallel and the maximum possible acceleration is not achieved 2. With a matrix dimension of 16 by 16 and single precision, the execution time on the GPU turned out to be 1.32 – 1.72 times faster than on the CPU. With a 32 by 32 matrix dimension, the execution time on the GPU turned out to be 3.66 – 6.07 times faster than on the CPU. 3. When calculating with double precision, the greatest acceleration is observed at 71.89 with 10 iterations of calculation, if there are more than 104 iterations, then the acceleration in calculations on a GPU with double precision approaches calculations with single precision.

Keywords: parallel programming; graphics processor; Crank – Nicholson method; thermal conductivity equation, performance.

Введение

Программно-аппаратная архитектура параллельных вычислений позволяет существенно повысить скорость вычислений. Технология CUDA применяется в здравоохранении для выявления полезной площади на рентгеновских снимках и сжатия изображения, при этом такой алгоритм работает в 34 раза быстрее, чем последовательный алгоритм, выполняемый на центральном процессоре [1, с. 1144]. Технология параллельного программирования эффективна при решении дифференциальных уравнений в частных производных [2, с. 41]. К сожалению, большинство научных статей про использование технологии параллельных вычислений пишутся без приведения программного кода. Исключением явилась статья [3, с. 9], в которой описывается решение задачи одномерной теплопроводности с использованием явной схемы. Статья [3, с. 6] подтолкнула к изучению технологии параллельного программирования и написанию статей [4, с. 9; 5, с. 24].

В статье [6, с. 346] решается аналогичная задача двухмерной теплопроводности, но с использованием явной схемы при количестве шагов по времени 100. Ускорение составило 48 раз. Явная схема с использованием параллельных вычислений для решения уравнения Навье – Стокса отображена в статье [7, с. 123]. Среднее ускорение составило 32 раза, а максимальное – 100. В статье [8, с. 431] авторы решают задачу теплопроводности явным методом и выделяют его недостаток: для обеспечения устойчивости метода требуется достаточно большое количество узлов. К недостаткам неявной схемы и схемы Кранка – Николсона относится высокая вычислительная сложность на каждой итерации, а преимуществом является устойчивость. Численные результаты решения задачи одномерной теплопроводности [5, с. 37] показали, что среди явного, неявного и метода Кранка – Николсона на графическом процессоре самым быстрым является явный, затем следуют метод Кранка – Николсона и неявный. Так как метод Кранка – Николсона обладает вторым порядком точности, то он был выбран для моделирования задачи двухмерной теплопроводности.

Существуют специальные программные продукты [9, с. 551], позволяющие преобразовать код, написанный на языке С, в код программы CUDA C. За счет таких программ увеличивается скорость выполнения по сравнению с последовательными вычислениями на центральном процессоре (ЦП), но они медленнее, чем если бы код был написан на языке CUDA C. Для написания параллельного кода требуется графический процессор (ГП) фирмы NVIDIA с поддержкой ядер CUDA и пакет программирования CUDA SDK, который позволит взаимодействовать разработчику с видеокартой. Для упрощения использования технологии CUDA в пакете программирования SDK встроено несколько библиотек, которые позволяют использовать возможности CUDA без необходимости писать код самостоятельно [10, с. 90; 11, с. 173–209].

При этом максимальное ускорение программы, выполняющейся параллельно, рассчитывается по формуле Амдала:

$$S = \frac{1}{(1-P) + \frac{P}{N}}, \tag{1}$$

где P – это часть времени выполнения программы, которая может быть распараллелена на N процессоров.

Практическое использование расчетов на ЦП и ГП показывает, что максимальную производительность можно получить при абсолютно параллельной задаче [4, с. 10], а при решении конечно-разносной схемы Кранка – Николсона требуется синхронизация ячеек (за счет этого тратится общее время выполнения) [5, с. 31].

В статье [5, с. 27] решалась задача одномерной теплопроводности на ГП с использованием технологии параллельного программирования CUDA. Наибольший выигрыш по производительности получился при большем количестве параллельно запущенных нитей с наименьшим количеством итераций.

Результаты и обсуждение

Для проведения вычислительного эксперимента использовались:

- операционная система Windows 10;
- среда программирования Visual Studio 2019 Community Edition;
- средства разработки на языке CUDA (CUDA Toolkit SDK версии 11.5);
- графический драйвер версии 511.79.

Расчеты производились на ЦП AMD Phenom 955 BE и графической карте начального уровня Gigabyte GeForce GT 1030 с частотой ядер 1252 MHz и количеством CUDA ядер 384.

При написании кода на языке CUDA используется как ЦП, так и ГП. Код на ГП пишется на языке CUDA, а на ЦП – на языках C или Python, в данном случае используется язык C.

Сравнение численных методов конечно-разностной аппроксимации показало, что схема Кранка – Николсона является более точной по сравнению с явной и неявной схемами [12, с. 1809], поэтому она использовалась при решении задачи двухмерной теплопроводности несмотря на то, что большая производительность и максимальная нагрузка получаются при реализации явных схем [13, с. 10].

Уравнение Фурье – Кирхгофа описывает нестационарный перенос тепла:

$$c\rho\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right) + Q(x,y,z,t,T), \tag{2}$$

где c – удельная теплоемкость, Дж/(кг·К); ρ – плотность, кг/м^3; $T(x, y, z, \tau)$ – поле температуры в трехмерном пространстве; t – время, с; λ – коэффициент теплопроводности, Вт/(м·К); $Q(x, y, z, \tau, T)$ – мощность внутренних источников тепловыделения, Вт.

Уравнение Фурье – Кирхгофа описывает множество вариантов процесса теплопроводности. Рассмотрим решение двухмерной задачи теплопроводности через изолированную пластину с постоянными коэффициентами. На нижней границе пластины, слева и справа поддерживается постоянная температура $T_Н = T_Л = T_П$, вверху пластины $T_В$ температура границы пластины равна температуре окружающей среды. Начальная температура пластины равна T_0.

Уравнение Фурье – Кирхгофа (2) в двухмерном случае без источников тепла внутри пластины будет выглядеть следующим образом:

$$c\rho\frac{\partial T}{\partial \tau} = \lambda\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right), 0 < x < L, 0 < y < L, \tag{3}$$

где L – длина и высота квадратной пластины, м.

Выразив коэффициент теплоотдачи через удельную теплоемкость, плотность и коэффициент температуропроводности $a = \frac{\lambda}{c\rho}$, получим уравнение (3) в виде

$$\frac{\partial T}{\partial \tau} = a\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right), 0 < x < L, 0 < y < L, \tag{4}$$

Начальные условия запишутся следующим образом:

$$\tau = 0 : T = T_0, 0 \le x \le L, 0 \le y \le L. \tag{5}$$

Граничные условия для уравнения (4)

$$x = 0 : T = T_{Л}, \tau > 0; \tag{6}$$

$$x = L : T = T_{П}, \tau > 0; \tag{7}$$

$$y = 0 : T = T_{В}, \tau > 0; \tag{8}$$

$$y = L : T = T_{Н}, \tau > 0. \tag{9}$$

Задача решается методом конечных разностей на равномерной квадратной сетке. Для этого пластина разбивается по длине и высоте на *N*-1 равных промежутков, а для аппроксимации уравнения теплопроводности применяется десятиточечный шаблон конечно-разностной схемы Кранка – Николсона (рис. 1).

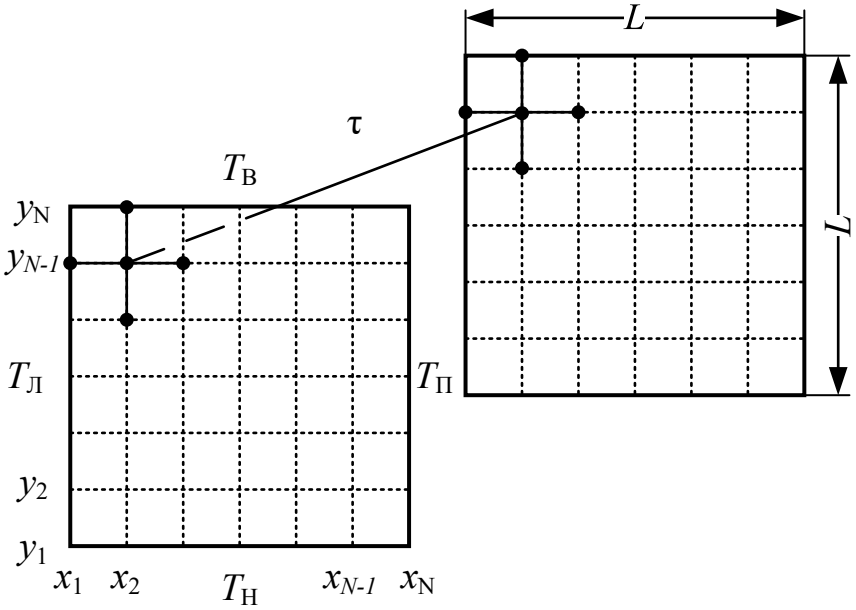

Рисунок 1 – Десятиточечный шаблон схемы Кранка – Николсона

При одностороннем нагреве снизу использовались граничные условия первого рода. Для изолированных сторон (левая, верхняя и нижняя) получим:

$$\frac{\partial T(x_{\min}, y, \tau)}{\partial \tau} = \frac{\partial T(x, y_{\max}, \tau)}{\partial \tau} = \frac{\partial T(x_{\max}, y, \tau)}{\partial \tau} = 0; \tag{10}$$

По всей нижней границе зададим постоянное значение температуры

$$\frac{\partial T(x, y_{\min}, \tau)}{\partial \tau} = 100. \tag{11}$$

Заменим дифференциальные операторы в уравнении (4):

$$\frac{\partial T}{\partial \tau} = \frac{T_i^{n+1} - T_i^n}{\tau}, \tag{12}$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1}^{n+1} - 2 \cdot T_i^{n+1} + T_{i+1}^{n+1}}{\Delta x^2}, \tag{13}$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{j-1}^{n+1} - 2 \cdot T_j^{n+1} + T_{j+1}^{n+1}}{\Delta y^2}. \tag{14}$$

Получим

$$\frac{T_i^{n+1} - T_i^n}{\tau} = a\left(\frac{T_{i-1}^{n+1} - 2 \cdot T_i^{n+1} + T_{i+1}^{n+1}}{\Delta x^2} + \frac{T_{j-1}^{n+1} - 2 \cdot T_j^{n+1} + T_{j+1}^{n+1}}{\Delta y^2} \right), \tag{15}$$

где T_i^{n+1} – значение температуры в *ij*-м узле разностной сетки в последующий момент времени, T_i^n – значение температуры в *ij*-м узле разностной сетки в *n*-й момент времени.

ЦП предназначен для выполнения одного потока инструкций, ГП – для выполнения множества потоков. Для ускорения на ГП используется быстрая общая память, размер которой ограничен. ЦП поддерживает один или два потока на ядро. ГП поддерживает до 1024 потоков на потоковый процессор.

На рисунке 2 показана структура потоков на ГП. Самая маленькая единица в архитектуре CUDA – это поток или нить. Нити объединяются в блоки. Блоки в свою очередь объединяются в сетку.

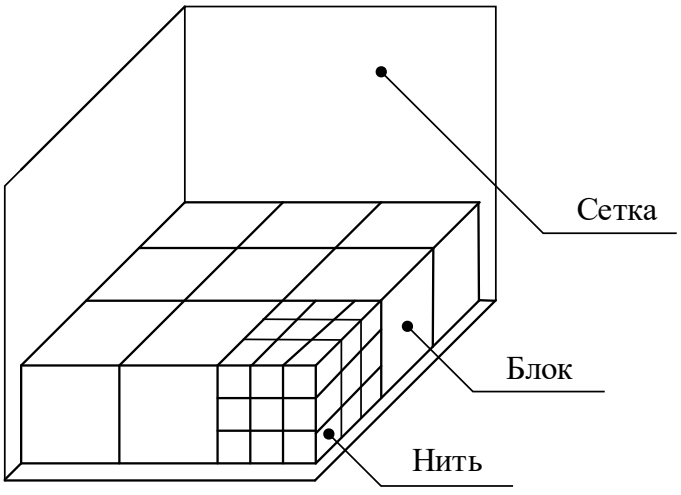

Рисунок 2 – Структура нитей, блоков и сетки

Все нити начинают работу одновременно, для обращения к номеру нити используется код, представленный в листинге 1.

Листинг 1

```
int idx = blockIdx.x * blockDim.x + threadIdx.x;
```

где blockIdx.x – индекс блока, blockDim.x – размерность блока, threadIdx.x – индекс нити в блоке.

Реализованная программа на ГП состоит из основного метода mai, который пишется на языке С, в нем происходит: 1) задание начальных параметров; 2) выделение памяти на ЦП и ГП; 3) копирование данных из ЦП в ГП; 4) выполнение расчетов на ГП; 5) копирование дан-

ных из ГП в ЦП; 6) вывод результатов и их обработка; 7) освобождение ресурсов на ГП и ЦП.

В статье [5, с. 10] решалась аналогичная задача в одномерной постановке, поэтому приведем отличный от одномерной постановки код.

На ГП используется функция пятиточечного шаблона, и на одной итерации для каждой нити она вызывается дважды для расчета на текущем и следующем шаге. Код приведен в листинге 2, а её графическая интерпретация показана на рисунке 1.

Листинг 2

```
__device__ double SH5(double L, double R, double T, double D, double M, double dx, double dy)
{
double res;
res = (L – 2 * M + R) / dx / dx + (D – 2 * M + T) / dy / dy;
return res;
}
```

Как видно из приведенного листинга, данная функция отличается от языка С только наличием специального слова из синтаксиса языка CUDA __device__, которое означает, что функция будет выполняться на ГП.

Основная функция (листинг 3) выполняется на ГП, принимает на вход одномерный массив данных и получает на выходе одномерный массив:

Листинг 3

```
__global__ void implicit(double* ziro, double* res)
{
//Задание начальных параметров расчета …
…
__shared__ double arr[SIZE]; //n
__shared__ double arrs[SIZE]; //n + 1
__shared__ double arrt[SIZE]; //t
__shared__ double _max; //Переменная, которая доступна всем потокам
_max = 1;

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

arr[idx] = ziro[idx];
arrs[idx] = ziro[idx];
arrt[idx] = ziro[idx];
__syncthreads(); //синхронизация потоков
for (int j = 0; j < 1000000; j++) { //количество итераций
_max = 1;
while (_max > eps) { //Пока заданная точность не достигнута
//В центре пластины
if ((idx >= _N) && (idx < SIZE – _N) && (idx % _N != 0) && (idx % _N != _N – 1)) {
//Находим значение на следующем шаге
arrs[idx] = arr[idx] + gamma * ((arr[idx] – arrt[idx]) / dt – a * (SH5(arr[idx – 1], arr[idx + 1],
arr[idx – _N], arr[idx + _N], arr[idx], _gdx, _gdy) + SH5(arrt[idx – 1], arrt[idx + 1], arrt[idx – _N],
arrt[idx + _N], arrt[idx], _gdx, _gdy)) / 2.0);
_max = fabs(arr[_N + 1] – arrs[_N + 1]);
if (_max < fabs(arr[idx] – arrs[idx])) {
_max = fabs(arr[idx] – arrs[idx]);
}
}
__syncthreads();
```

```
arr[idx] = arrs[idx];
}
arrt[idx] = arr[idx]; //Получаем значение через dt
__syncthreads();
}

res[idx] = arr[idx]; //Для вывода результат
__syncthreads();
}
```

Как видно из приведенного кода, __global__ обозначает, что функция будет выполняться на языке CUDA. Основной цикл for определяет количество шагов по времени. А условный цикл while определяет условие достижения заданной точности eps.

Так как функции, выполняемой на ГП, на вход подается одномерный массив, то переход от одномерного массива к двухмерному осуществляется с помощью количества ячеек по ширине и высоте пластины.

На четырех границах задаются условия с использованием функции, выполняющейся на ГП border(). Угловые точки считаются как средние значения между ближайшими существующими соседями по вертикали и горизонтали.

Ключевое слово __syncthreads() обозначает синхронизацию всех нитей, которая необходима для расчетов. Все нити дожидаются выполнения самой медленной из них. Как видно из приведенного кода, такая процедура повторяется на одной итерации несколько раз.

В центре пластины используется функция пятиточечного шаблона SH5(), которая приведена в листинге 2. Переход от одномерного массива к квадратному двухмерному представим графически (рис. 3). В данном случае для массива 4 на 4 остаются четыре центральные точки, для которых применяется пятиточечный шаблон. На рисунке 3 приведен шаблон для первой центральной точки с индексом 6.

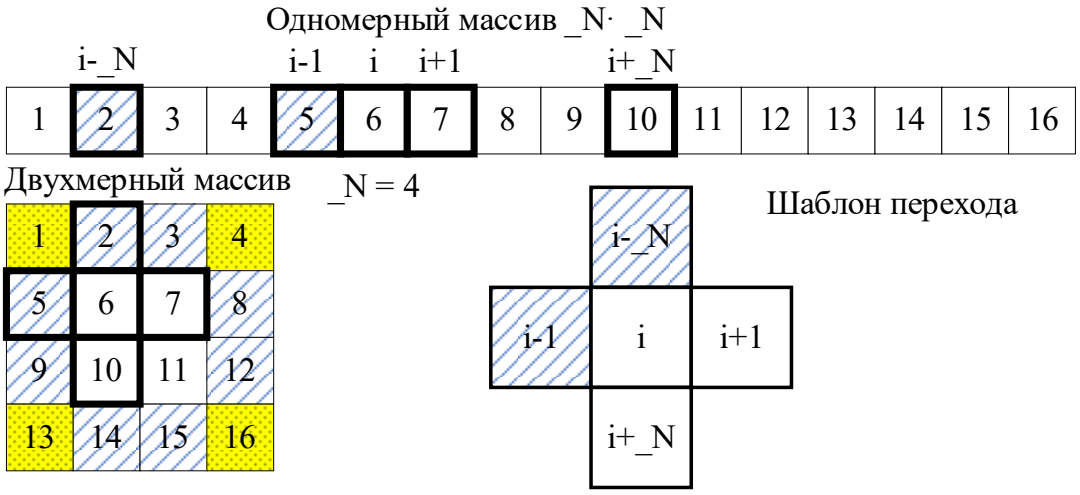

Рисунок 3 – Переход от одномерного массива к двухмерному

В таблице 1 приведены результаты сравнения скоростей вычислений на ЦП и ГП при размерности двухмерного массива 16 на 16 ячеек и количестве итераций от 10 до 10^6. Таблицы 2 и 3 показывают скорости выполнения решения задачи двухмерной теплопроводности с количеством ячеек 1024. В таблице 2 используются числа с одинарной точностью (float), а в таблице 3 используются числа с двойной точностью (double). Как видно из таблицы 1, скорость расчета на ГП быстрее, чем на ЦП, от 1,33 до 1,71 раза.

Таблица 1 – Сравнение времени выполнения алгоритма для массива 16 на 16 с использованием чисел одинарной точности

Количество итераций	**$\tau_{ЦП}$, мс**	**$\tau_{ГП}$, мс**	**$\tau_{ЦП} / \tau_{ГП}$**
10^1	993	651	1,53
10^2	7532	4401	1,71
10^3	10438	7865	1,33
10^4	10685	7917	1,35
10^5	12949	9813	1,32
10^6	37594	24479	1,54

Таблица 2 – Сравнение времени выполнения алгоритма для массива 32 на 32 с использованием чисел одинарной точности

Количество итераций	**$\tau_{ЦП}$, мс**	**$\tau_{ГП}$, мс**	**$\tau_{ЦП} / \tau_{ГП}$**
10^1	4293	920	4,67
10^2	29762	4902	6,07
10^3	45545	10107	4,51
10^4	46473	10994	4,23
10^5	56707	14001	4,05
10^6	159365	43570	3,66

Таблица 3 – Сравнение времени выполнения алгоритма для массива 32 на 32 с использованием чисел двойной точности

Количество итераций	**$\tau_{ЦП}$, мс**	**$\tau_{ГП}$, мс**	**$\tau_{ЦП} / \tau_{ГП}$**
10^1	4026	56	71,89
10^2	27446	730	37,60
10^3	44585	2809	15,87
10^4	43857	9303	4,71
10^5	56370	13970	4,04
10^6	149653	47122	3,18

Следует отметить, что расчеты с двойной точностью выполняются быстрее, за исключением расчета на ГП при количестве итераций 10^6. А максимальная производительность ГП по отношению к ЦП получается при меньшем количестве итераций.

Заключение и выводы

В статье рассмотрен алгоритм решения задачи двухмерной теплопроводности с использованием технологии параллельного программирования CUDA.

1. Литературный обзор показал, что быстродействие алгоритмов на графическом процессоре по сравнению с алгоритмами, выполняемыми на центральном процессоре, достигает от нескольких единиц до сотен раз.

2. Решение двухмерной задачи теплопроводности по схеме Кранка –Николсона не является абсолютно параллельной, и максимально возможное ускорение не достигается.

3. При размерности матрицы 16 на 16 и одинарной точности время выполнения на графическом процессоре оказалось в 1,32 – 1,72 быстрее, чем на центральном процессоре. При размерности матрицы 32 на 32 время выполнения на графическом процессоре оказалось в 3,66 – 6,07 быстрее, чем на центральном процессоре.

4. При вычислениях с двойной точностью наблюдается наибольшее ускорение в 71,89 при 10 итерациях расчета, если итераций более 104, то ускорение в расчетах на графическом процессоре с двойной точностью приближается к расчетам с одинарной точностью.

Литература

1. Unver, M. Compressing of magnetic resonance images with CUDA / M. Unver, A. Erguzen // International Journal of Trend in Scientific Research and Development. – 2018. – V. 3, № 1. – P. 1140–1145.

2. Вахлаева, К. П. Анализ эффективности применения технологии CUDA для численного решения дифференциальных уравнений в частных производных по схеме Кранка – Николсона / К. П. Вахлаева, Д. А. Сынкова, Д. С. Фаюстов. – Текст : непосредственный // Евразийский союз ученых. – 2015. – № 7–6 (16). – С. 38–41.

3. Афанасьева, Е. Ю. Использование технологии параллельного программирования CUDA для решения задачи теплопроводности / Е. Ю. Афанасьева. – Текст : непосредственный // Завершенные исследования. – 2015. – № 1. – С. 6–11.

4. Сеченов, П. А. Применение технологии параллельного программирования NVIDIA CUDA в задаче расплавления шарообразной частицы / П. А. Сеченов, А. А. Оленников. – Текст : непосредственный // Кибернетика и программирование. – 2018. – № 5. – С. 8–14.

5. Сеченов, П. А. Решение задачи одномерной теплопроводности на графических процессорах с использованием технологии CUDA / П. А. Сеченов, И. А. Рыбенко. – DOI: 10.15593/2499-9873/2021.4.02. – Текст : непосредственный // Прикладная математика и вопросы управления. – 2021. – № 4. – С. 23–41.

6. Деги, Д. В. Реализация явной разностной схемы для решения двумерного уравнения теплопроводности на графическом процессорном устройстве с использованием технологии CUDA / Д. В. Деги, А. В. Старченко, А. А. Трунов. – Текст : непосредственный // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи : труды Международной суперкомпьютерной конференции, Новороссийск, 20–25 сентября 2010 г. – Новороссийск, 2010. – С. 346–348.

7. Маркин, Е. Е. Гетерогенные параллельные вычисления на примере решения полной системы уравнений Навье-Стокса методом сеток / Е. Е. Маркин, П. П. Скачков. – Текст : непосредственный // Международный студенческий научный вестник. – 2017. – № 5. – С. 123.

8. Широканев, А. С. Разработка векторного алгоритма по технологии CUDA для трехмерного моделирования процесса лазерной коагуляции сетчатки / А. С. Широканев, Н. А. Андриянов, Н. Ю. Ильясова. – Текст : непосредственный // Компьютерная оптика. – 2021. – Т. 45, № 3. – С. 427–437.

9. Khan, A. H. RT-CUDA: A Software Tool for CUDA Code Restructuring / A. H. Khan, M. Al-Mouhamed, M. Al-Mulhem, A. F. Ahmed. – Doi.org/10.1007/s10766-016-0433-6 // International Journal of Parallel Programming. – 2017. V. 45. – P. 551-594.

10. Воротникова, Д. Г. Алгоритмы с «длинными» векторами решения сеточных уравнений явных разностных схем / Д. Г. Воротникова, Д. Л. Головашкин. – Текст : непосредственный // Компьютерная оптика. – 2015. – Т. 39, № 1. – С. 87–93.

11. Storti, D. CUDA for engineers : an introduction to high-performance parallel computing / D. Storti, M. Yurtoglu. – New York : Addison-Wesley, 2015. – 328 p.

12. Залевский, Д. В. Математическое моделирование процессов аутостабилизации температуры в клеточной ткани для одномерного случая на основе неявных разностных схем / Д. В. Залевский, А. А. Арзамасцев. – Текст : непосредственный // Вестник Тамбовского университета. Серия: Естественные и технические науки. – 2014. – Т. 19, № 6. – С. 1805–1812.

13. Прокофьев, В. А. Многослойная гидротермическая модель водоёма с реализацией вычислений на графическом акселераторе / В. А. Прокофьев. – Текст : непосредственный // Известия Всероссийского научно-исследовательского института гидротехники им. Б. Е. Веденеева. – 2017. – Т. 284. – С. 3–18.