DOI: 10.15593/2499-9873/2021.4.02
УДК 004.032.24

П.А. Сеченов, И.А. Рыбенко

Сибирский государственный индустриальный университет,
Новокузнецк, Россия

РЕШЕНИЕ ЗАДАЧИ ОДНОМЕРНОЙ ТЕПЛОПРОВОДНОСТИ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ CUDA

Разработана и программно реализована математическая модель для решения задачи одномерной теплопроводности. Целью моделирования является сравнение быстродействия алгоритмов на центральном и графическом процессорах.

Задача распараллеливания является актуальной, так как еще в 2015 г. количество потоковых процессоров в самой мощной видеокарте составляло 2816, а в 2021 г. появились видеокарты с 10 496 потоковыми процессорами. Приложения, работающие на графических процессорах NVIDIA, демонстрируют большую производительность в расчете на доллар вложенных средств и на ватт потребленной энергии по сравнению с реализациями, построенными на базе одних лишь центральных процессоров. Это подтверждается большим спросом видеокарт у майнеров, что привело на данный момент к подорожанию видеокарт в 1,5–2,5 раза.

Представлены требования к аппаратной и программной составляющим, необходимые для начала моделирования. Реализовано три метода конечно-разностной аппроксимации: явный, неявный и Кранка – Николсона на центральном и графическом процессорах. В качестве языков программирования выбраны языки C (центральный процессор) и CUDA C (графический процессор). Для хорошо распараллеленной задачи, когда каждый поток выполняется отдельно и ему не нужны данные из других потоков, ускорение вычислений на видеокарте увеличилось до 60 раз (при этом использовалась видеокарта начального уровня).

Язык CUDA C возник относительно недавно – в 2006 г. и имеет ряд особенностей при реализации параллельного алгоритма. Для выбранных схем – явной, неявной, Кранка – Николсона – на каждой итерации необходимо обращение к соседним потокам и синхронизация потоков. Синхронизация потоков происходит таким образом, что все потоки на каждой итерации дожидаются самого медленного из них, поэтому решение задач с использованием конечно-разностной аппроксимации будет выполняться медленнее.

Представлен фрагмент кода на графическом процессоре для реализации схемы Кранка – Николсона. При реализации схемы Кранка – Николсона требуется использование быстрой разделяемой памяти для обмена данными между потоками. Объем разделяемой памяти ограничен и сказывается на количестве ячеек в сетке.

Использование графических карт дало ощутимый прирост скорости выполнения даже на карте начального уровня с количеством потоковым процессоров 384. Приведен сравнительный анализ быстродействия вычислений при разных размерах сетки – от 1024 до 4000, а также при разном количестве объемов вычислений в одном потоке.

Ключевые слова: метод конечных разностей, параллельное программирование, уравнение теплопроводности, явный метод, неявный метод, метод Кранка – Николсона, графический процессор, закон Амдала, скорость выполнения.

P.A. Sechenov, I.A. Rybenlko

Siberian State Industrial University, Novokuznetsk, Russian Federation

SOLVING THE PROBLEM OF ONE-DIMENSIONAL THERMAL CONDUCTIVITY ON GRAPHICS PROCESSORS USING CUDA TECHNOLOGY

A mathematical model for solving the problem of one-dimensional thermal conductivity has been developed and implemented programmatically. The purpose of the simulation is to compare the performance of algorithms on the central and graphics processors.

The task of parallelization is relevant, since back in 2015 the number of stream processors in the most powerful video card was 2816, and in 2021 there were video cards with 10 496 stream processors. Applications running on NVIDIA GPUs demonstrate greater performance per dollar of invested funds and per watt of energy consumed compared to implementations built on the basis of central processors alone. This is confirmed by the high demand for video cards from miners, which has led to a 1.5–2.5 times increase in the price of video cards at the moment.

The requirements for the hardware and software components necessary for the start of modeling are presented. Three methods of finite difference approximation are implemented: explicit, implicit and Crank-Nicolson on the central and graphics processors. The programming languages chosen are C (CPU) and CUDA C (GPU). For a well-parallelized task, when each thread is executed separately and it does not need data from other threads, the acceleration of calculations on the video card increased up to 60 times (an entry-level video card was used).

The CUDA C language appeared relatively recently in 2006 and has a number of features when implementing a parallel algorithm. For the selected schemes: explicit, implicit, Crank–Nicolson, at each iteration, it is necessary to access neighboring threads and synchronize the threads. Synchronization of threads occurs in such a way that all threads wait for the slowest of them at each iteration, so solving problems using finite-difference approximation will be performed slower.

A fragment of code on a GPU for implementing the Crank–Nicolson scheme is presented. The implementation of the Crank-Nicolson scheme requires the use of fast shared memory for data exchange between threads. The amount of shared memory is limited and affects the number of cells in the grid.

The use of graphics cards gave a significant increase in execution speed even on an entry-level card with a number of 384 stream processors. The article presents a comparative analysis of the computing speed for different grid sizes from 1024 to 4000, as well as for different amounts of computing volumes in one thread.

Keywords: finite difference method, parallel programming, thermal conductivity equation, explicit method, implicit method, Crank – Nicolson method, GPU, Amdahl's law, execution speed.

Введение

Конечно-разностная аппроксимация является одним из самых простых и старейших методов решения уравнений в частных производных. Появление методов конечных разностей в численных приложениях началось в начале 1950-х гг. их развитие стимулировалось за счет появления компьютеров, которые позволили решать сложные проблемы науки и техники.

В настоящее время применение технологии параллельных вычислений CUDA (Compute Unified Device Architecture) [1, 2] позволяет существенно увеличить вычислительную производительность благодаря использованию ГП (графических процессоров, или видеокарт).

В статье рассмотрены реализации решений уравнений одномерной теплопроводности по следующим схемам: явной, неявной и Кранка – Николсона. Произведено сравнение быстродействия программ на центральном и графическом процессорах.

1. Теория

В ноябре 2006 г. NVIDIA выпустила первую в истории ГП GeForce 8800 GTX, построенную на архитектуре CUDA [3]. Для охвата большой аудитории разработчиков NVIDIA взяла стандартный язык C и дополнила его несколькими новыми ключевыми словами, позволяющими задействовать специальные средства, присущие архитектуре CUDA [4].

Важной характеристикой любого вычислительного устройства является его быстродействие, которое определяется количеством выполняемых операций в секунду. Существуют ограничения по размерам транзисторов (на данный момент это 7 нм), и поэтому рост быстродействия устройств идет за счет увеличения параллельно работающих ядер, т.е. через параллелизм.

Согласно закону Амдала, который гласит, что в случае, когда задача разделяется на несколько частей, суммарное время ее выполнения на параллельной системе не может быть меньше времени выполнения самого медленного фрагмента; максимальное ускорение, которое можно получить от распараллеливания программ на N процессоров (ядер), определяется по формуле [4]

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

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

При $N \to \infty$ выигрыш стремится к $\frac{1}{(1-P)}$.

Во многих приложениях, имеющих возможность параллельной обработки, удалось добиться повышения производительности на несколько порядков. Также «приложения, работающие на графических процессорах NVIDIA, демонстрируют большую производительность в расчете на доллар вложенных средств и на ватт потребленной энергии

по сравнению с реализациями, построенными на базе одних лишь центральных процессоров» [3].

Далее рассмотрим системные и программные требования для написания и запуска параллельных вычислений.

2. Данные и методы

Для написания кода на языке CUDA C требуются следующие элементы:

- графический процессор, поддерживающий архитектуру CUDA;
- драйвер устройства NVIDIA;
- комплект средств разработки CUDA (CUDA Toolkit SDK);
- компилятор языка C.

В качестве графического процессора использовалась видеокарта начального уровня 2017 г. выпуска GeForce 1030 с 384 ядрами CUDA. Стоит отметить, что такое же количество ядер было у видеокарты среднего сегмента GeForce 560 Ti 2011 г. выпуска, на которой были проведены первые эксперименты, в том числе с вычислением максимально возможного ускорения относительно вычислений на процессоре AMD Phenom 955 BE. Видеокарты 1030 нет в официальном списке поддерживаемых видеокарт, но тем не менее она работает и поддерживает технологию CUDA. На самой дешевой видеокарте GeForce 210 с использованием ОС Windows 8.0 не удалось скомпилировать проект.

Использовался последний на момент написания статьи драйвер устройства 471.41. В качестве средств разработки CUDA использовалась версия 10.2, поддерживаемая Windows 7. Начиная с 11-й версии CUDA SDK требуется операционная система Windows 10.

В качестве среды разработки была выбрана бесплатная среда разработки Visual Studio 2019 Community Edition с поддержкой языка C.

Архитектуру ГП можно кратко охарактеризовать как «макроархитектуру вычислительного кластера, реализованного в микромасштабе» [5]. В настоящее время ГП может содержать от нескольких сотен – 384 (GeForce 1030) до нескольких тысяч – 10 496 (GeForce RTX 3090) потоковых процессоров, а это значит, что при полной загруженности потоковых процессоров производительность отличается в 27 раз между видеокартами начального и верхнего уровней. Стоит также отметить, что в 2021 г. цены на видеокарты завышены в 1,5–2,5 раза по сравнению с 2017 и 2019 гг. из-за бума майнинга.

К особенностям языка CUDA C можно отнести то, что в нем используются два устройства: центральный процессор (host) и графический процессор (device). Программа на CUDA задействует как ЦП, так и ГП. Последовательный код выполняется на центральном процессоре, а для параллельных вычислений соответствующий код выполняется на ГП как набор одновременно выполняющихся нитей.

3. Модель

Методы конечных разностей решают дифференциальные уравнения в частных производных путем аппроксимации дифференциальных уравнений по области интегрирования системой алгебраических уравнений. Они являются средством получения численных решений уравнений в частных производных. Наиболее распространенными методами конечных разностей для решения уравнений в частных производных являются: явный метод, неявный метод, методы Кранка – Николсона [6]. Эти методы тесно связаны, но отличаются стабильностью, точностью и скоростью выполнения. Рассмотрим приведенный ниже рис. 1.

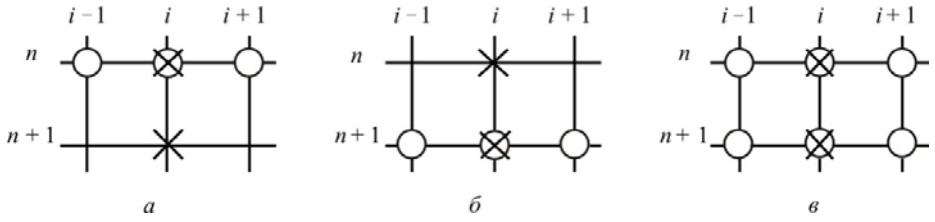

Рис. 1. Графическое сравнение методов конечных разностей:
а – явный метод, *б* – неявный метод, *в* – метод Кранка – Николсона,
(авторские результаты)

Нестационарный перенос тепла теплопроводностью описывается уравнением Фурье–Кирхгофа, записанным в декартовой системе координат:

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

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

Уравнение (2) описывает множество вариантов процесса теплопроводности. В качестве примера рассмотрим решение одномерной задачи теплопроводности через изолированный стержень с постоянными коэффициентами. На одной границе стержня поддерживается постоянная температура $T_л$, на другой границе температура равна температуре окружающей среды. Начальная температура равна T_0, источники тепла внутри пластины отсутствуют.

При таких начальных условиях температура будет изменяться вдоль длины стержня. Предположим, что теплофизические характеристики стержня не зависят от текущей температуры, т.е. являются постоянными величинами. Уравнение (2) в данном случае будет выглядеть следующим образом:

$$c\rho\frac{\partial T}{\partial \tau}=\lambda\frac{\partial^2 T}{\partial x^2}, 0<x<L, \tag{3}$$

где L – длина стержня, м.

Приняв $a=\frac{\lambda}{pc}$, получим уравнение (3) в виде

$$\frac{\partial T}{\partial \tau}=a\frac{\partial^2 T}{\partial x^2}, 0<x<L. \tag{4}$$

Начальные и граничные условия запишутся следующим образом:

$$\begin{aligned}&\tau=0: T=T_0, 0\le x\le L;\\ &x=0: T=T_л, \tau>0;\\ &x=L: T=T_п, \tau>0.\end{aligned} \tag{5}$$

Эту задачу решаем методом конечных разностей на равномерной сетке. Для этого стержень по длине разбиваем на N–1 равных промежутков (рис. 2).

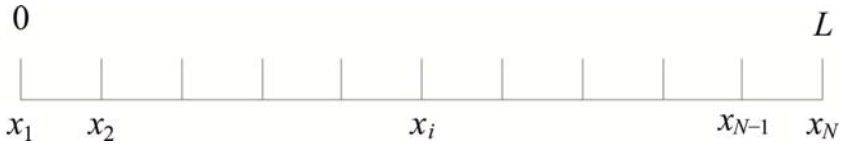

Рис. 2. Конечно-разностная сетка: $x_2, x_3, \ldots, x_{N-1}$ – координаты внутренних узлов, x_1, x_N – координаты граничных узлов *(авторские результаты)*

При одностороннем нагреве использовались граничные условия первого рода, когда температура остается постоянной во всем процессе теплообмена. Это может быть осуществлено при искусственном поддержании постоянной температуры. Слева будет изолированная сторона:

$$\frac{\partial T(0,\tau)}{\partial \tau} = 0, \tag{6}$$

с правой стороны будет постоянная поддерживаемая температура:

$$\frac{\partial T(L,\tau)}{\partial \tau} = 100. \tag{7}$$

Заменим дифференциальные операторы в уравнении (4) на их конечно-разностные аналоги. Для неявной схемы получаем

$$\begin{aligned} \frac{\partial T}{\partial t} &= \frac{T_i^{n+1} - T_i^n}{\tau}, \\ \frac{\partial^2 T}{\partial x^2} &= \frac{T_{i+1}^{n+1} - 2 \cdot T_i^{n+1} + T_{i-1}^{n+1}}{h^2}. \end{aligned} \tag{8}$$

В статье [7] предложена модель теплопроводности с использованием клеточных автоматов, которая требует значительных вычислительных ресурсов и может быть реализована на компьютерах, использующих технологию параллельных вычислений. Задача нагрева пластины для явной схемы рассматривалась в статье [8], поэтому код для расчета на центральном процессоре (ЦП) приводиться не будет.

Намного интереснее и необычнее выглядит код на графическом процессоре. Общая схема работы представлена на рис. 3.

Из статьи [8] был взят код реализации явной схемы на ЦП, который был модифицирован с учетом граничных условий 1-го рода. Реализация расчетов на ЦП и ГП также была взята из статьи [8], поэтому рассмотрим в общем виде функцию ядра, которая рассчитывается на графическом процессоре (рис. 4).

На рис. 5 показана иерархия потоков. Всё начинается с нити. Тридцать две нити (thread) объединяются в один пучок (warp) [9]. Блок может включать в себя 1024 нити и обладает разделяемой памятью внутри блока. Блоки объединяются в сетку.

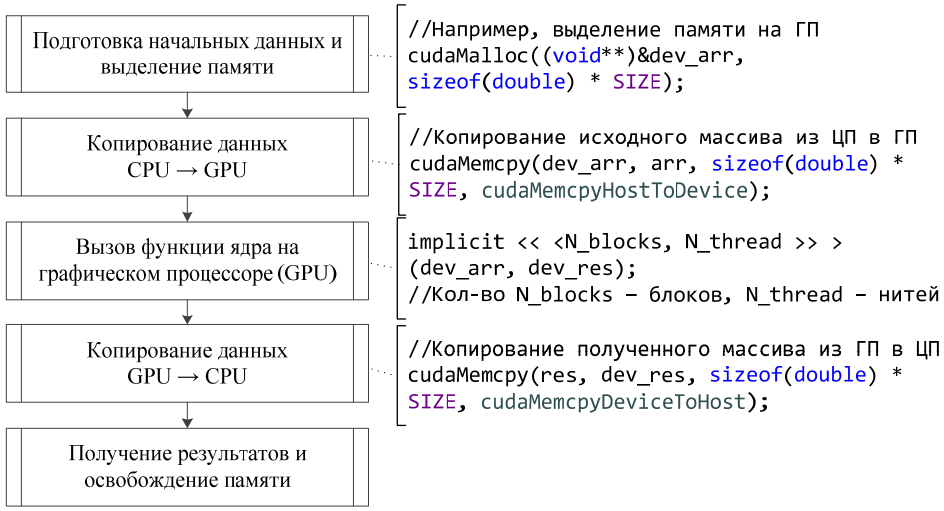

Рис. 3. Общая схема работы центрального и графического процессоров
(авторские результаты)

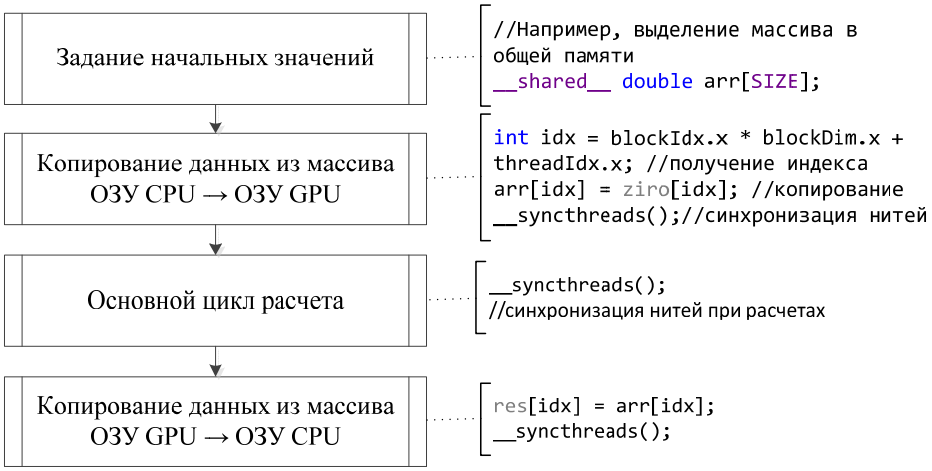

Рис. 4. Общая схема работы функции ядра на графическом процессоре
(авторские результаты)

Следует отметить, что функции, вычисляемые на графическом процессоре, начинаются с ключевого слова языка CUDA C __global__ [10]. Ключевое слово __shared__ [11] служит для объявления разделяемой памяти, куда могут обращаться все нити, принадлежащие одному блоку. А ключевое слово __syncthread() позволяет синхронизировать потоки внутри одного блока [4]. Иными словами, прежде чем перейти к но-

вой итерации, расчеты во всех потоках должны быть завершены. Синхронизация необходима в задачах, где данные из одной нити могут обращаться к данным другой нити. В явной, неявной и схеме Кранка – Николсона идет обращение к соседним ячейкам, в данном случае к соседним нитям, поэтому скорость выполнения будет зависеть от самой медленно выполняющейся нити внутри блока. В авторских статьях [12, 13] были проведены исследования, когда нити выполнялись абсолютно параллельно и скорость вычислений на центральном и графическом процессорах была в 60 раз быстрее на графическом ускорителе.

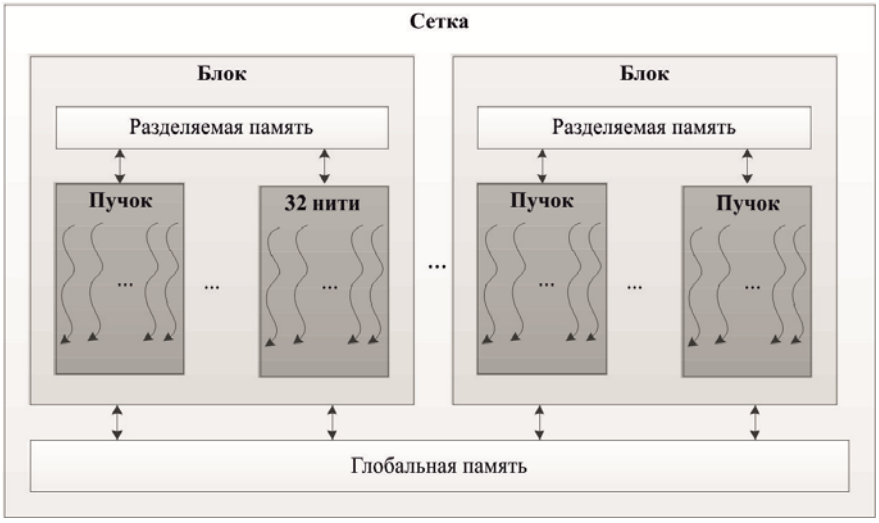

Рис. 5. Иерархия потоков и доступ к частям памяти на графическом процессоре *(авторские результаты)*

Из представленных схем самой точной является схема Кранка – Николсона, так как она берет данные как на текущем шаге, так и на предыдущем. Представим листинг ядра:

```
__global__ void implicit(double* ziro, double* res)
{
 double S = 0.025;      //Длина пластины
 double a = 0.0000064;          //коэффициент температуропроводности
 double dt = 0.0003; //шаг по времени
 double all_time = 20;//общее время
 double gamma = -0.0000005; //параметр итерационного процесса
 double eps = 0.000001; //точность вычисления на итерации
 double dx = S / SIZE;
```

```
double _f = a / dx / dx;
int NT = all_time / dt;
//Массив в памяти графической карты
__shared__ double arr[SIZE]; //n + 1
__shared__ double arrs[SIZE]; //n
__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];
__syncthreads();
arrs[idx] = ziro[idx];
__syncthreads();
arrt[idx] = ziro[idx];
__syncthreads(); //синхронизация потоков*/

//Время
for (int j = 0; j < NT; j++) {
//Один шаг
_max = 1;
while (_max > eps) {
arr[0] = 0;
arrs[0] = 0;
arrt[0] = 0;
arr[SIZE - 1] = 100;
arrs[SIZE - 1] = 100;
arrt[SIZE - 1] = 100;
//В центре пластины
if ((idx > 0) && (idx < SIZE - 1)) {
__syncthreads();
arrs[idx] = arr[idx] + gamma * ((arr[idx] - arrt[idx]) / dt - _f * ((arr[idx - 1]
- 2 * arr[idx] + arr[idx + 1]) + (arrt[idx - 1] - 2 * arrt[idx] + arrt[idx + 1])) /
2.0);
_max = fabsf(arr[1] - arrs[1]);
__syncthreads();
if (_max < fabsf(arr[idx] - arrs[idx])) {
_max = fabsf(arr[idx] - arrs[idx]);
}
__syncthreads();
arr[idx] = arrs[idx];
__syncthreads();
```

```
 }
 }
 //Обновление значений массива для следующей итерации
 arrt[idx] = arr[idx];
 __syncthreads();

 }

 res[idx] = arr[idx]; //Копирование ГП->ЦП
 __syncthreads();
}
```

К отличительным особенностям реализации схемы Кранка – Николсона относятся три массива разделяемой памяти, которые определяются ключевым словом языка CUDA C __shared__. Это массивы для запоминания значений температуры на текущем, следующем и шаге по времени массивах. Также можно выделить переменную max, необходимую для определения сходимости схемы Кранка – Николсона, которая тоже относится к разделяемой памяти.

4. Полученные результаты

В табл. 1 представлены результаты сравнения быстродействия алгоритмов на центральном и графическом процессорах.

При расчетах на ГП можно указывать такую величину, как количество нитей, используемых в блоках. При этом это количество должно быть кратно 16, согласно литературе [3, 4].

В табл. 1 и на рис. 6 показано время выполнения решения задачи одномерной теплопроводности на центральном процессоре (ЦП) и графическом процессоре (ГП), при этом для графического процессора были рассмотрены варианты с использованием общей памяти и три варианта с количеством выполняемых нитей 1024, 512 и 256 в блоке.

Как видно на рис. 6, для всех вычислений наблюдается прямая зависимость между количеством расчетов и временем выполнения как на центральном, так и на графическом процессоре, при этом большее ускорение получается при 512 нитях. Скорость расчетов на графическом процессоре была всего в 4 раза быстрее, чем на центральном процессоре.

Таблица 1

Сравнение времени выполнения по явной схеме на центральном и графическом процессорах при 1024 ячейках *(авторские результаты)*

Заданное время расчета, с	ЦП, мс	ГП (1024), мс	ГП (512), мс	ГП (256), мс	Количество измерений	ГП(512)/ ЦП
10	124	46	38	36	10 000	3,26
100	1134	415	311	243	100 000	3,65
500	5412	1587	1230	1210	500 000	4,40
1000	11105	3270	2562	2544	1 000 000	4,33
1500	16010	4747	3689	3757	1 500 000	4,34
2000	21235	6330	4978	4835	2 000 000	4,27
2500	26644	8049	6318	6188	2 500 000	4,22

Следующим этапом было увеличение количества расчетных ячеек, в табл. 1 показаны расчеты для 1024 ячеек. Количество ячеек увеличили в 2 раза и получили результаты, представленные в табл. 2.

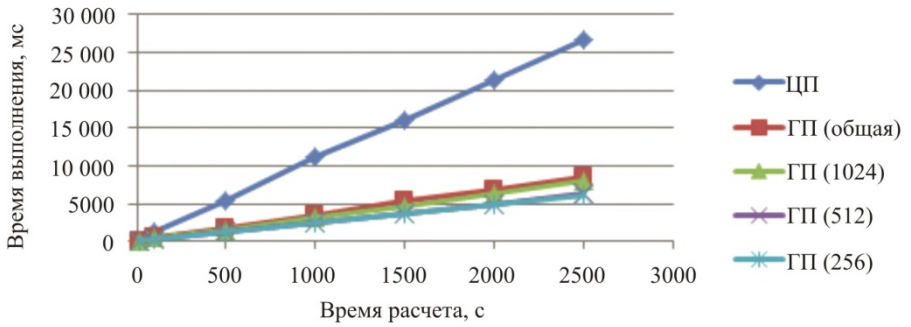

Рис. 6. Время выполнения на центральном и графическом процессорах *(авторские результаты)*

Таблица 2

Сравнение времени выполнения по явной схеме на центральном и графическом процессорах при 2048 ячейках *(авторские результаты)*

Заданное время расчета, с	ЦП, мс	ГП (512), мс	ГП (256), мс	Количество измерений	ГП(512)/ЦП
10	512	78	72	20 000	6,56
100	4363	673	495	200 000	6,48
500	21305	2666	2594	1 000 000	7,99
1000	42395	5437	4909	2 000 000	7,80
1500	63649	7995	7462	3 000 000	7,96

Как видно из табл. 2, быстрота расчетов увеличилась почти вдвое и максимальное значение составило 7,96. Увеличение количества ячеек позволило использовать потенциал графической карты и в следующей табл. 3 составило 4096.

Таблица 3

Сравнение времени выполнения по явной схеме на центральном и графическом процессорах при 4096 ячейках *(авторские результаты)*

Заданное время расчета, с	ЦП, мс	ГП (512), мс	ГП (256), мс	Количество измерений	ГП(512)/ЦП
10	4893	323	318	100 000	15,39
100	43 148	3343	3189	1 000 000	13,53
200	85 700	6424	6517	2 000 000	13,15
300	128 236	9782	9557	3 000 000	13,42

Как видно из табл. 3, быстрота расчетов увеличилась и составила в среднем 13,4. Следующая попытка при объеме данных 8192 ячейки не дала значительного прироста по скорости и составила в среднем 14,54 раза по сравнению с теми же расчетами на центральном процессоре.

Поскольку явная схема не является самой точной, были также рассмотрены неявная схема и схема Кранка – Николсона. Тривиальный пример с расчетом экспоненциальной функции представлен на рис. 7.

Отличительной особенностью реализации неявной схемы и схемы Кранка – Николсона является то, что требуется три одномерных массива для запоминания значений температуры на текущем, следующем шагах и массив для схождения схемы. При этом эти массивы должны располагаться в общей памяти, а также одно значение для запоминания текущего максимального значения. В графических процессорах существуют ограничения по объему такой памяти. Для ГП 1030 этот объем составляет 49 152 байта. Если использовать вещественный тип данных с одинарной точностью (float), одно значение которого занимает 4 байта, то можно использовать 12 288 значений. Если используется вещественный тип данных с двойной точностью (double), объем одного значения 8 байт, следовательно, получится в 2 раза меньше значений. Также требуется значение для поиска максимума, поэтому в расчетах не превышает 4096 – 1 значений для float и 2048 – 1 значений для double.

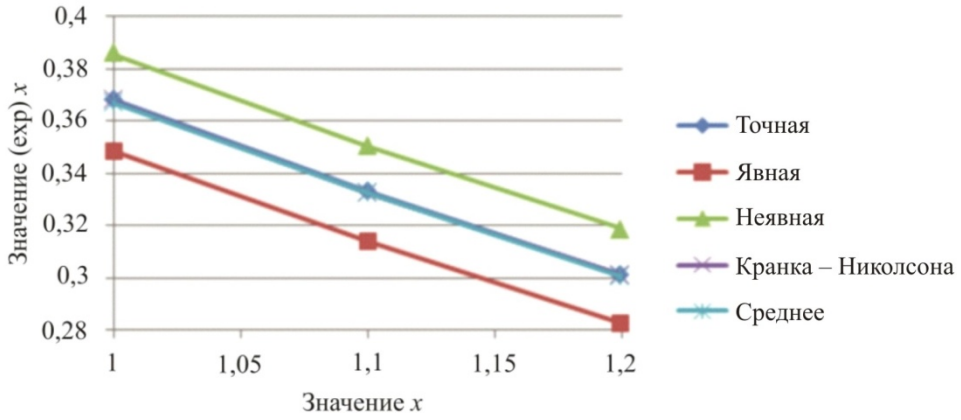

Рис. 7. Сравнение точности разных схем *(авторские результаты)*

Для неявной схемы при 4000 значениях получились результаты расчетов, представленные в табл. 4.

Таблица 4

Сравнение времени выполнения по неявной схеме на центральном и графическом процессорах при 4000 ячеек
(авторские результаты)

Время, с	ЦП	ГП (256)	Ускорение
10	66 702	33 287	2,003 85
20	72 844	55 908	1,302 93
30	76 921	78 626	0,978 32

Как видно из табл. 4, при увеличении времени расчета центральный процессор начинает опережать графический процессор. Это показательный пример того, что не во всех случаях графический процессор будет быстрее. Например, с ростом числа расчетных ячеек графический процессор работает быстрее. Это было видно по явной схеме, а также по схеме Кранка – Николсона.

Поскольку самой точной является схема Кранка – Николсона [6, 14], ей было уделено больше внимания. При 1024 ГП оказался быстрее от 3 до 4,2 раза. При 2000 – от 7,2 до 8,3, а самые лучшие результаты отражены в табл. 5.

Таблица 5

Сравнение времени выполнения по схеме Кранка – Николсона на центральном и графическом процессорах при 4000 ячеек *(авторские результаты)*

Время, с	ЦП, мс	ГП (400), мс	Ускорение
10	260 612	19 389	13,4412
20	263 631	25 723	10,2488
30	284 941	33 277	8,5627

Заключение

В статье рассмотрены алгоритмы реализации одномерной задачи теплопроводности на центральном и графическом процессорах. При обработке данных от 1000 до 4000 ячеек на графическом процессоре получается большая производительность от 0,97 до 13,5 раза. Наибольшая производительность показывается при 4000 ячеек, так как на графическом процессоре задача рассчитывается параллельно. При увеличении количества шагов по явной схеме замедления расчетов по сравнению с центральным процессором не наблюдается. При реализации схем неявной и Кранка – Николсона наблюдается замедление в ускорении вычислений.

В дальнейшем планируется применить данную методику и алгоритм параллельного программирования для двухмерной и трехмерной задачи нагрева угольного сырья [15].

Исследование не имело спонсорской поддержки. Авторы заявляют об отсутствии конфликта интересов.

Список литературы

1. Storti D., Yurtoglu M. CUDA for engineers: an introduction to high-performance parallel computing. – Addison-Wesley, New York, 2016. – 331 p.

2. Soyata T. GPU parallel program development using CUDA. – CRC Press: Taylor & Francis Group, 2018. – 477 p.

3. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров / пер. с англ. А.А. Слинкина, науч. ред. А.В. Боресков. – М.: ДМК Пресс, 2011. – 232 с.

4. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010. – 232 с.

5. Параллельные вычисления на GPU. Архитектура и программная модель CUDA: учеб. пособие / А.В. Боресков, А.А. Харламов, Н.Д. Марковский, Д.Н. Микушин, Е.В. Мортиков, А.А. Мыльцев, Н.А. Сахарных, В.А. Фролов. – М.: Изд-во Моск. ун-та, 2012. – 336 с.

6. Sunday F., Edogbanya O.H., Samuel C.Z. Crank Nicolson method for solving parabolic partial differential equations // International Journal of Applied Mathematics and Modeling IJA2M. – 2013. – Vol. 1, no 3. – P. 8–23.

7. Бобков С.П., Галиаскаров Э.Г. Моделирование процесса теплопроводности с использованием систем клеточных автоматов // Программные продукты и системы. – 2020. – № 4 (33). – С. 641–649.

8. Афанасьева Е.Ю. Использование технологии параллельного программирования CUDA для решения задачи теплопроводности // Завершенные исследования. – 2015. – № 1. – С. 6–11.

9. Cheng J., Grossman M., McKercher T. Professional CUDA C programming. – John Wiley & Sons, 2014. – 497 p.

10. Ватутин Э.И., Мартынов И.А., Титов В.С. Оценка реальной производительности современных видеокарт с поддержкой технологии CUDA в задаче умножения матриц // Известия Юго-Западного государственного университета. Сер.: Управление, вычислительная техника, информатика. Медицинское приборостроение. – 2014. – № 2. – С. 8–17.

11. Ружников В.О. Повышение производительности расчета динамики частиц на параллельных системах // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. – 2014. – № 1. – С. 147–156.

12. Сеченов П.А., Оленников А.А. Применение технологии параллельного программирования Nvidia CUDA в задаче расплавления шарообразной частицы // Кибернетика и программирование. – 2018. – № 5. – С. 8–14.

13. Сеченов П.А., Оленников А.А., Цымбал В.П. Применение технологии параллельного программирования CUDA в задаче расплавления шарообразной частицы // V Всерос. науч.-практ. конф. студентов, аспирантов и молодых ученых (ТИМ'2016), г. Екатеринбург, 12–13 мая 2016 г. / Урал. федер. ун-т им. первого Президента России Б.Н. Ельцина. – Екатеринбург, 2016. – С. 260–263.

14. Omowo B.J., Longe I.O. Crank-Nicolson and modified Crank-Nicolson scheme for one dimensional parabolic equation // International Journal of Applied Mathematics and Theoretical Physics. – 2020. – Vol. 6, № 3. – P. 35–40. DOI: 10.11648/j.ijamtp.20200603.11

15. Рыбенко И.А., Сеченов П.А., Калашников С.Н. Разработка детерминированной математической модели нестационарного теплового состояния смерзшегося в вагоне угольного сырья на установке для его разморозки // Наукоемкие технологии разработки и использования минеральных ресурсов. – 2021. – № 7. – С. 243–246.

References

1. Storti D., Yurtoglu M. CUDA for engineers: an introduction to high-performance parallel computing. Addison-Wesley, New York, 2016, 331 p.

2. Soyata T. GPU Parallel Program Development Using CUDA. CRC Press: Taylor & Francis Group, 2018, 477 p.

3. Sanders J., Kandrot E. CUDA by example: an introduction to general purpose GPU programming. Addison-Wesley, 2011, 232 p.

4. Boreskov A.V., Kharlamov A.A. Osnovy raboty s tekhnologiei CUDA [Basics of working with CUDA technology]. Moscow, DMK Press, 2010, 232 p.

5. Parallel'nye vychisleniia na GPU. Arkhitektura i programmnaia model' CUDA: uchebnoe posobie [Parallel computing on the GPU. Architecture and software model CUDA] / A.V. Boreskov, Kharlamov A.A., Markovskii N.D., Mikushin D.N., Mortikov E.V., Myl'tsev A.A., Sakharnykh N.A., Frolov V.A. Moscow: Izdatel'stvo Moskovskogo universiteta, 2012, 336 p.

6. Sunday F., Edogbanya O.H., Samuel C.Z. Crank Nicolson method for solving parabolic partial differential equations. *International Journal of Applied Mathematics and Modeling IJA2M*. 2013, Vol. 1, no 3, pp. 8–23.

7. Bobkov S.P., Galiaskarov E.G. Simulation of the heat conduction process using cellular automata systems. *Software & Systems*, 2020, vol. 33, no. 4, pp. 641–650 (in Russ.). DOI: 10.15827/0236-235X.132.641-650.

8. Afanas'eva E.Iu. Ispol'zovanie tekhnologii parallel'nogo programmirovaniia CUDA dlia resheniia zadachi teploprovodnosti [Using the CUDA Parallel Programming Technology to Solve the Heat Conduction Problem]. *Zavershennye issledovaniia*. 2015, no 1, pp. 6–11.

9. Cheng J., Grossman M., McKercher T. Professional CUDA C programming. John Wiley & Sons, 2014, 497 p.

10. Vatutin E.I., Martynov I.A., Titov V.S. Otsenka real'noi proizvoditel'nosti sovremennykh videokart s podderzhkoi tekhnologii CUDA v zadache umnozheniia matrits [Assessment of the real performance of modern video cards with support for CUDA technology in the problem of matrix multiplication] *Izvestiia Iugo-Zapadnogo gosudarstvennogo universiteta. Seriia: Upravlenie, vychislitel'naia tekhnika, informatika. Meditsinskoe priborostroenie*. 2014, no 2, pp. 8–17.

11. Ruzhnikov V.O. Povyshenie proizvoditel'nosti rascheta dinamiki chastits na parallel'nykh sistemakh [Improving the performance of calculating particle dynamics on parallel systems]. *Vestnik Sankt-Peterburgskogo universiteta. Prikladnaia matematika. Informatika. Protsessy upravleniia*. 2014, no 1, pp. 147–156.

12. Sechenov P.A., Olennikov A.A. Primenenie tekhnologii parallel'nogo programmirovaniia Nvidia CUDA v zadache rasplavleniia sharoobraznoi chastitsy

[Application of parallel programming technology Nvidia CUDA in the problem of melting a spherical particle]. *Kibernetika i programmirovanie*. 2018, no 5, pp. 8–14.

13. Sechenov P.A., Olennikov A.A., Tsymbal V.P. Primenenie tekhnologii parallel'nogo programmirovaniia CUDA v zadache rasplavleniia sharoobraznoi chastitsy [Application of parallel programming technology CUDA in the problem of melting a spherical particle]. *Proceedings of the V All-Russian meeting on control sciences*, *12–13 May 2016*, Ekaterinburg, Ural Federal University named after the first President of Russia B.N.Yeltsin, 2016, pp. 260-263.

14. Omowo B.J., Longe I.O. Crank-Nicolson and Modified Crank-Nicolson Scheme for One Dimensional Parabolic Equation. *International Journal of Applied Mathematics and Theoretical Physics*. 2020, Vol 6, no 3, pp. 35–40. DOI: 10.11648/j.ijamtp.20200603.11.

15. Rybenko I.A., Sechenov P.A., Kalashnikov S.N. Razrabotka determinirovannoi matematicheskoi modeli nestatsionarnogo teplovogo sostoianiia smerzshegosia v vagone ugol'nogo syr'ia na ustanovke dlia ego razmorozki [Development of a deterministic mathematical model of the non-stationary thermal state of coal raw materials frozen in a car at a plant for its defrosting]. *Naukoemkie tekhnologii razrabotki i ispol'zovaniia mineral'nykh resursov*. 2021, No 7, pp 243-246.

Сведения об авторах

Сеченов Павел Александрович (Новокузнецк, Россия) – кандидат технических наук, доцент кафедры «Прикладные информационные технологии и программирование», Сибирский государственный индустриальный университет (654007, Новокузнецк, ул. Кирова, 42, e-mail: pavesa89@mail.ru).

Рыбенко Инна Анатольевна (Новокузнецк, Россия) – доктор технических наук, доцент, заведующий кафедрой «Прикладные информационные технологии и программирование», Сибирский государственный индустриальный университет (654007, Новокузнецк, ул. Кирова, 42, e-mail: rybenkoi@mail.ru).

About the authors

Pavel A. Sechenov (Novokuznetsk, Russian Federation) – Ph.D. in Engineering, Associate Professor, Department of Applied Information Technologies and Programming, Siberian State Industrial University, Novokuznetsk (42, Kirova st., Novokuznetsk, 654007, e-mail: pavesa89@mail.ru).

Inna A. Rybenko (Novokuznetsk, Russian Federation) – Dr. Habil. in Engineering, Associate Professor, Head of Department of Applied Information Technologies and Programming, Siberian State Industrial University, Novokuznetsk (42, Kirova st., Novokuznetsk, 654007, e-mail: rybenkoi@mail.ru).

Статья получена:13.09.2021

Статья принята: 15.11.2021

Опубликовано: 26.01.2022

Библиографическое описание статьи согласно ГОСТ Р 7.0.100–2018:

Сенченко, П.А. Решение задачи одномерной теплопроводности на графических процессорах с использованием технологии CUDA / П. А. Сеченов, И. А. Рыбенко. – текст: непосредственный. – DOI: 10.15593/2499-9873/2021.4.02 // Прикладная математика и вопросы управления = Applied Mathematics and Control Sciences. – 2021. – № 4. – С. 23–41.

Цитирование статьи в изданиях РИНЦ:

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

Цитирование статьи в references и международных изданиях

Cite this article as:

Sechenov P.A., Rybenlko I.A. Solving the problem of one-dimensional thermal conductivity on graphics processors using CUDA technology. *Applied Mathematics and Control Sciences*, 2021, no. 4, pp. 23–41. DOI: 10.15593/2499-9873/2021.4.02 *(in Russian)*