МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Кафедра гидроаэродинамики, горения и теплообмена

Работа допущена к защите
Зав. кафедрой, проф., д.ф.-м.н.
_____ Смирнов Е.М.
"____"_____ 2016 г.

Оценка эффективности использования графических процессоров для ускорения расчетов по ряду алгоритмов вычислительной гидродинамики

Выпускная бакалаврская работа по направлению подготовки
03.03.01 – Прикладные математика и физика

Выполнил студент гр. 43603/1 Федотов А. В.

Руководитель, к.ф.-м.н., доцент Гарбарук А. В.

Санкт-Петербург
2016

РЕФЕРАТ

Работа содержит 27 страниц, 15 рисунков, 7 таблиц, 7 библиографических названий.

Ключевые слова: вычислительная гидродинамика, графический процессор, GPGPU, параллельные вычисления

Работа посвящена исследованию возможности ускорения NTS-кода, используемого в лаборатории Гидроаэроакустики Санкт-Петербургского Политехнического Университета Петра Великого, путем распределения вычислений между центральным и графическим процессорами. Выяснено, что во время расчетов наибольшее время тратиться на выполнение явных алгоритмов (порядка 66%). На основе анализа алгоритмов и схемы работы с графическим процессором составлена модельная задача, позволяющая оценить ускорение любого явного алгоритма. Произведены измерения на двух характерных конфигурациях оборудования: рабочей станции и вычислительном кластере. В зависимости от конфигурации оборудования возможное ускорение NTS кода составляет 9 – 15%.

Краткая аннотация

Данная работа посвящена исследованию возможности ускорения NTS-кода путем распределения вычислений между центральным и графическим процессорами. В зависимости от конфигурации оборудования возможное ускорение составляет 9 – 15%.

Содержание

1. Введение

Современный компьютер содержит в себе не только центральный процессор (CPU), но и дополнительные вычислительные ресурсы – периферийные устройства, увеличивающие вычислительные мощности машины. Одним из таких ресурсов является видеокарта, которая обладает собственным вычислительным процессором (Graphical Processor Unit - GPU) и памятью, которую часто называют видеопамятью.

С 2001 года активно развивается применение видеокарты для вычислений, не ориентированных на её профильную задачу - обработку трёхмерной графики. Техника использования GPU для таких задач, называется GPGPU (General-purpose computing for graphics processing units).

Раньше GPGPU, из-за сложной программной реализации, использовалась исключительно энтузиастами. Программирование видеокарты производилось с использованием языков программирования нижнего уровня, что значительно усложняло написание программ и делало их несовместимыми с другими устройствами. К тому же, недостатки мощности графических процессоров и низкая скорость обмена данными между оперативной памятью и видеопамятью не позволяли получить приемлемого ускорения для множества задач.

С развитием технологий производства видеокарты начали становиться все более производительными. Это сделало возможным использование GPU для общих вычислений, в том числе и для расчета задач вычислительной гидродинамики. В связи с этим стали появляться технологии, такие как CUDA и OpenCL, реализующие возможность программирования графических процессоров при помощи C-подобных языков программирования и обеспечивающие переносимость кода.

В связи с развитием GPGPU множество вычислительных кластеров и суперкомпьютеров (на 2015 год свыше 100 суперкомпьютеров из списка TOP500) оборудуются графическими процессорами. Некоторые вычислительные пакеты

уже поддерживают распараллеливание своих алгоритмов на графические процессоры. Среди таковых: ANSYS Fluent, OpenFOAM и aeroFluidX. Существует ряд программных библиотек (например: CUSP, cuSPARCE, cuSOLVER), реализующих решение систем линейных алгебраических уравнений, состоящих из плотных и разряженных матриц на графических процессорах.

Однако с реализацией программ на GPU связан ряд трудностей:

1. Графический процессор обладает гораздо большим количеством вычислительных ядер по сравнению с центральным (к примеру, процессор видеокарты NVIDIA Tesla K40 обладает 2880 ядрами, а процессор Intel Haswell – 14-ю). В связи с этим GPU обладает отличной от CPU схемой параллелизации (SIMD – Single Instructions Multiple Data у GPU и MIMD – Multiple Instructions Multiple Data у CPU)
2. Специфика параллелизации кода на графическом процессоре, а также сложная иерархия памяти (отличная от иерархии памяти CPU) делают производительность данного устройства сильно зависящей от оптимизации работы с памятью. Неоптимизированный доступ к данным может значительно повысить время работы алгоритма

Поскольку видеокарта является периферийным устройством, для того чтобы произвести на ней вычисления необходимо взять данные из оперативной памяти (ОЗУ), перенести их на видеокарту, отдать команду на обработку и, по завершении работы видеокарты, перенести данные обратно в оперативную память. Передача данных на видеокарту осуществляется посредством интерфейса PCI, скорость которого значительно ниже скорости интерфейса, обеспечивающего взаимодействие CPU и оперативной памяти. В связи с этим скорость обмена информацией с видеокартой становится узким местом работы с данным устройством. Подробную информацию о работе с видеокартой можно найти в [1].

В лаборатории Гидроаэроакустики Санкт-Петербургского Политехнического Университета Петра Великого используется расчетный код NTS. Данный

код может выполняться исключительно на центральном процессоре. Однако, в связи с доступностью вычислительных кластеров, оборудованных графическими процессорами, появился интерес в проверке возможности ускорения данного кода с использованием GPU.

Целью данной работы является исследование возможности ускорения NTS кода путем распределения вычислений между центральным и графическим процессорами.

2. Использование видеокарты для расчета задач вычислительной гидродинамики

В настоящее время проведён ряд исследований, направленных на измерение ускорения ряда алгоритмов для различных численных методов при использовании GPU вместо CPU.

В работе [2] описано ускорение алгоритма обращения матриц. Для тестирования использовались матрицы трёх различных типов: плотные, ленточные и разряженные. Данные типы матриц наиболее часто используются при расчете задач гидродинамики.

Тестирование производилось на трех различных конфигурациях оборудования:

1. CPU: Intel i7-3770K (4 ядра, 3.5 ГГц); GPU: Nvidia Quadro K2000M (384 ядра, 745 МГц, 2048 Мб видеопамяти)
2. CPU: Intel i5-3470 (4 ядра, 3.2 ГГц); GPU: Nvidia GTX 760 (1152 ядра, 980МГц, 2048 Мб видеопамяти)
3. CPU: Intel i5-3470 ; GPU: Nvidia GTX 760 SLI (используются две видеокарты GTX 760, объединенные по технологии NVIDIA SLI)

На рисунке 1 представлена зависимость ускорения алгоритма обращения матрицы относительно времени расчёта только на CPU от размера матрицы.

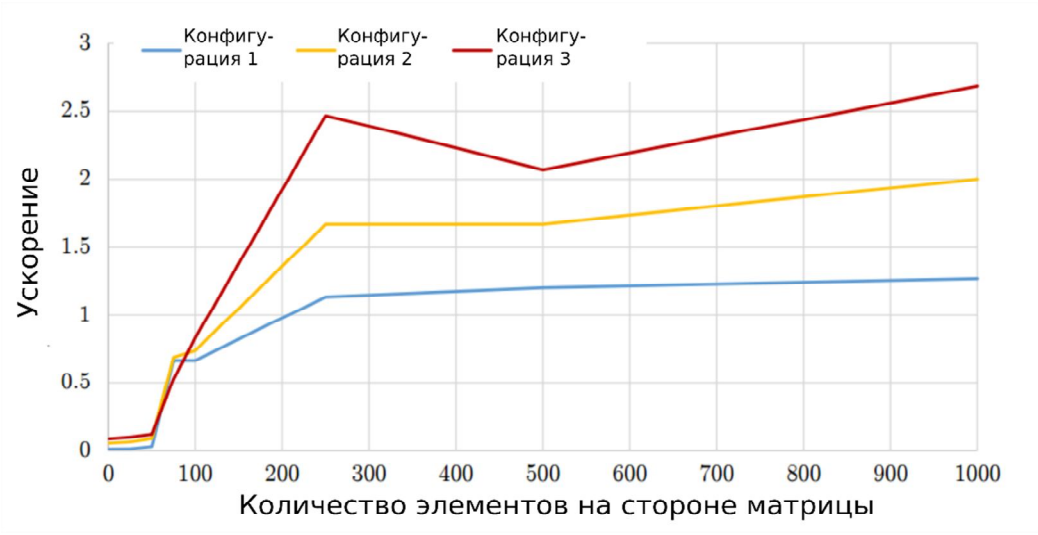


Рисунок 1 Зависимость ускорения алгоритма обращения матрицы посредством применения GPU от размерности матрицы

Видно, что при размере матрицы более 300x300 авторам удалось достичь ускорения алгоритма приблизительно в 1.1 – 2.1 раза.

В работе [3] рассматривается ускорение метода сопряженных градиентов с использованием различных предобуславливателей. Измерения производились на кластере, оборудованным четырьмя узлами, каждый из которых содержал два центральных процессора Intel Xeon X5675 (6 ядер, 3.06 ГГц) и четыре видеокарты NVIDIA TESLA M2070 (448 ядер, 1.15 ГГц, 3Гб видеопамяти).

В таблице 1 представлены ускорения, достигнутые авторами [3], при этом значение ускорения оценивалось как отношение времени исполнения кода на одном CPU ко времени исполнения на одном GPU. Результаты приведены для различных матриц, полученных из коллекции разряженных матриц Университета Флориды. Среди этих матриц есть две, полученные при решении задач вычислительной гидродинамики. Как видно из таблицы, использование видеокарты позволило ускорить алгоритм в 1.3 – 1.4 раза.

Таблица 1 Ускорение метода сопряженных градиентов с использованием предобуславливателя для различных матриц

№ матрицы	Количество элементов	Количество ненулевых элементов	Ускорение
1	4.9×10^{9}	1.8×10^{6}	1.3
2	1.5×10^{10}	3.1×10^{6}	1.4

В работе [4] произведено исследование ускорения алгоритмов следующих методов разложения матрицы: LU-разложение, метод Холецкого и QR-разложение. Измерения производились на центральном процессоре Intel Core2 Quad Q6850 (4 ядра, 3.0 ГГц) и двух видеокартах: NVIDIA GeForce 8800GTX (128 ядер, 575 МГц, 768 Мб видеопамяти) и NVIDIA GeForce GTX280 (240 ядер, 602 МГц, 1Гб видеопамяти). Результаты представлены в таблице 2:

Таблица 2 Ускорение методов разложения матриц

Видеокарта	8800GTX	GTX280
Алгоритм	Ускорение	
LU-разложение	2.5	4.1
Холецкий	2.7	4.4
QR-разложение	2.6	4.4

Видно, что для различных алгоритмов получаются приблизительно одинаковые ускорения на каждом графическом процессоре. Причем при использовании видеокарты NVIDIA GeForce 8800GTX удалось ускорить код более чем в 2.5 раза, а для NVIDIA GeForce GTX280 – более чем в 4.1 раза.

В статьях [5-6] исследовалось ускорение вычислительных кодов гидродинамики при применении графических процессоров.

В работе [5] рассматривается ускорение вычислительного кода MFBLO для расчета турбулентных течений. Данная программа написана на языке программирования Fortran. Расчеты проводились на кластере, состоящем из 4 узлов. Каждый узел оснащен двумя видеокартами Tesla C2050 (448 ядер, 3ГБ видеопамяти) и

двумя центральными процессорами Intel Xeon X5550 (2.66 ГГц, 4 ядра каждый). Для распределения задачи между узлами и ядрами CPU этих узлов использовалась технология «Message Passing Interface» (далее MPI), а для распределения между ядрами GPU – технология CUDA.

Измерение производилось для расчета турбулентного обтекания цилиндра на сетке O-типа с 3.2 миллионами узлов.

Для сравнения производительности использовалось три измерения:

1. Использовался только CPU
2. Совместное использование CPU и GPU
3. Совместное использование CPU и GPU с использованием асинхронной передачей данных на видеокарту. Информацию об этом методе ускорения работы с памятью можно прочитать в [1]

Авторам удалось ускорить код в 6.3 раза путем совместного использования графического и центрального процессоров (см. рис. 2). Проведя оптимизацию передачи данных на видеокарту, удалось добиться ускорения в 10.5 раз.

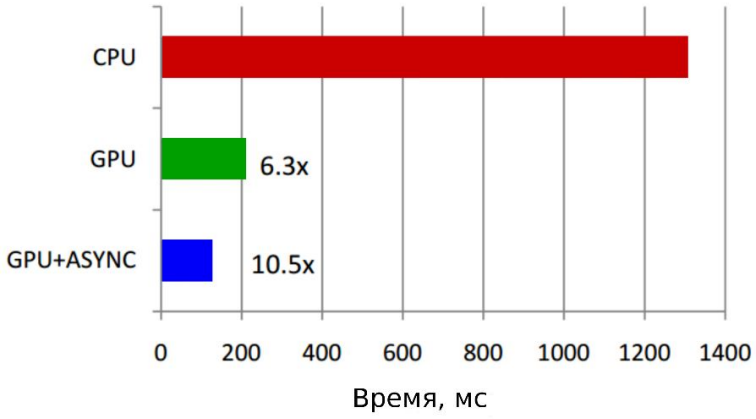


Рис. 2 Время вычисления задачи турбулентного обтекания цилиндра для трех случаев расчетов

В работе [6] представлено ускорение вычислительного кода, работающего с трехмерной многоблочной, структурированной сеткой. Измерения производились на суперкомпьютере TianHe-1A.

Каждый узел данного суперкомпьютера оборудован видеокартой NVIDA Tesla M2050 (448 ядер, 1150 МГц) и двумя центральными процессорами Intel Xeon X5670s (2.93 ГГц, 6 ядер).

В качестве тестового примера решалась задача обтекания крылового профиля. Результат получен для сеток одинакового размера с различным количеством блоков (см. рис. 3). В случае использования одного процессора и одной видеокарты получилось добиться ускорения в 1.2 – 1.4 раза.

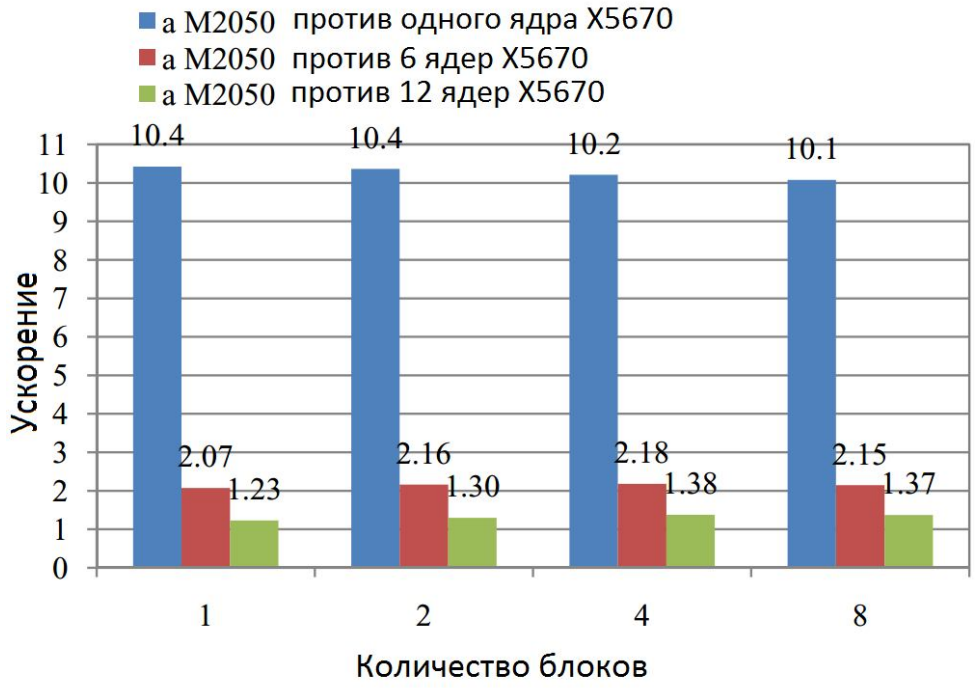


Рис. 3 Зависимость ускорения от количества блоков

Все коды, описанные в данных статьях, основаны на явных и полу-неявных схемах. Использование полностью неявных схем затруднено в силу специфики параллелизации графического процессора.

В работе [7] рассматривается ускорение расчёта нестационарного обтекания сферы невязкой, несжимаемой жидкостью на неструктурированной сетке с треугольными элементами. В основе решателя лежит неявный метод Галеркина с разрывными базисными функциями.

Расчеты проводились на машине, оборудованной центральным процессором AMD Opteron 6128 (2ГГц, 8 ядер) и видеокартой NVIDIA Tesla K20c (2496 ядер, 706 МГц, 5 Гб видеопамяти).

В таблице 3 представлена зависимость полученного ускорения и приведенного времени от количества элементов сетки. Приведенное время рассчитано по формуле:

$$T_{unit} = \frac{T_{run}}{N_{time} \times N_{elem}}$$

Здесь: T_{run} – время работы алгоритма, N_{time} – количество шагов по времени, N_{elem} – количество элементов сетки.

Таблица 3 результаты измерений в зависимости от размера сетки

	T_{unit}		
Кол-во элементов сетки	GPU	CPU	Ускорение
2426	41.63	74.60	1.79
16476	15.18	77.43	5.10
124706	11.18	80.99	7.12

Применение GPU ускорило решение данной задачи, причем с увеличением количества числа элементов сетки ускорение растет. Таким образом, использование графического процессора позволяет ускорить вычислительный код даже в случае использования неявных алгоритмов.

Как видно, использование графического процессора для решения задач вычислительной гидродинамики позволяет ускорить код. Так же следует заметить, что получаемое ускорение зависит от объема вычислений – чем больше задача, тем выше получаемое ускорение.

3. NTS-код

Прежде чем переходить к оценке возможного ускорения NTS-кода с помощью графических карт, необходимо рассмотреть принцип его работы. Данный код может выполняться как на персональных рабочих станциях, так и на вычислительных кластерах. В связи с этим в нем реализованы два уровня параллелизации (см. рис. 4):

1. параллелизация между вычислительными узлами кластера посредством технологии MPI (Message Passing Interface)
2. параллелизация между ядрами процессора в вычислительном узле посредством технологии OpenMP

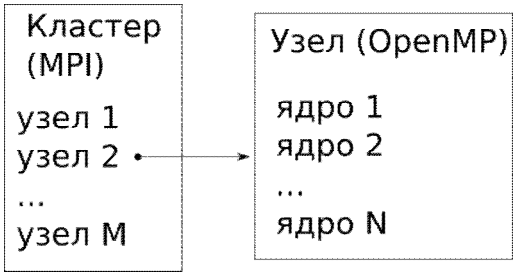


Рис. 4 Текущая схема параллелизации NTS кода

В таблице 4 представлено процентное соотношение времени выполнения наиболее затратных алгоритмов NTS-кода. Анализ производительности был выполнен при помощи программного продукта Intel VTune Amplifier XE 2016.

Таблица 4 Анализ производительности NTS-кода

Функция	Описание	Процент времени
jacob_gas	Вычисление матрицы Якоби	25.1%
vectorsolve3d	Решение матрицы левых частей	15.9%
block_lhs	Подготовка к решению левых частей	14%
jacob_scal	Вычисление матрицы Якоби	11.3%

Наиболее затратные процедуры, занимающие порядка 66% от полного времени расчета, содержат в себе явные алгоритмы. Время выполнения остальных процедур составляет менее 5% от общего времени выполнения программы.

4. Выбор модельной задачи

Для оценки возможного ускорения NTS-кода в настоящей работе используется модельная задача. Ускорение, получаемое при вычислении такой задачи, должно быть легко отображаемо в ускорение реальных алгоритмов.

Использование графического процессора в NTS-коде подразумевает добавление ещё одного уровня параллелизации в вычислительном узле между CPU и GPU (рис. 5).

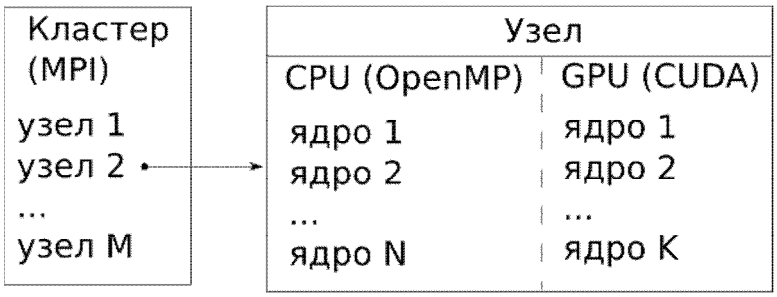


Рис. 5 Возможная позиция GPU в схеме параллелизации NTS кода

Таким образом, внедрение GPU подразумевает распределение задачи между центральным и графическим процессором. Производительность работы с центральным процессором в большей степени характеризуется его мощностью. А производительность видеокарты характеризуется не только мощностью графического процессора, но и скоростью передачи данных из оперативной памяти в видеопамять. Это связано с тем, что перед проведением вычислений на видеокарте требуется передать на нее данные, а затем получить их обратно, как показано на схеме:

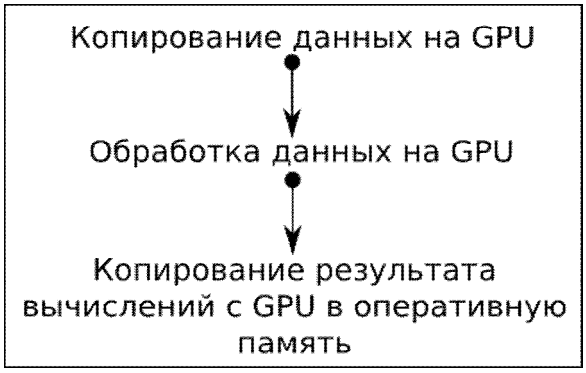


Рис. 6 Схема работы с графическим процессором

Однако, из-за того, что данные передаются на видеокарту с малой скоростью, время работы с этим устройством значительно увеличивается. Поэтому процесс передачи данных, по сути, является потерей времени.

Очевидно, что чем меньше операций будет приходится на единицу передаваемых данных, тем менее выгодным будет использование графического процессора. Таким образом, для модельной задачи определяющим будет отношение размера данных к количеству операций, которые алгоритм производит с этими данными, называемое $\frac{\text{передача}}{\text{расчет}}$ и измеряемое в $\frac{\text{байт}}{\text{операция}}$. Это позволит приблизительно смоделировать любой алгоритм NTS-кода и оценить для него ускорение.

В данной работе в качестве модельной задачи рассматривается перемножение заранее заданного количества элементов массива.

5. Результаты расчетов модельной задачи

Ускорение алгоритма зависит от оборудования, на котором он выполняется. В данной работе произведён ряд измерений на двух конфигурациях вычислительной техники:

1. Конфигурация 1: типичное оборудование среднего персонального компьютера, используемого в качестве рабочей или домашней станции:
   a. Процессор: «Intel i5-2500» 3.3 GHz (2 физических + 2 виртуальных ядра)
   b. Видеокарта: «NVIDIA GTX 550 Ti» 192 ядра по 1.8 GHz, 1GB памяти
2. Конфигурация 2: оборудование узла вычислительного кластера Политехнического университета «Tornado»:
   a. Процессор: два «Intel Haswell» 2.6 GHz (14 физических + 14 виртуальных ядер)
   b. Видеокарта: «NVIDIA Tesla K40» 2880 ядер, 745 MHz, 12GB памяти

Следует заметить, что в обоих центральных процессорах используется технология Intel Hyper-Treading.

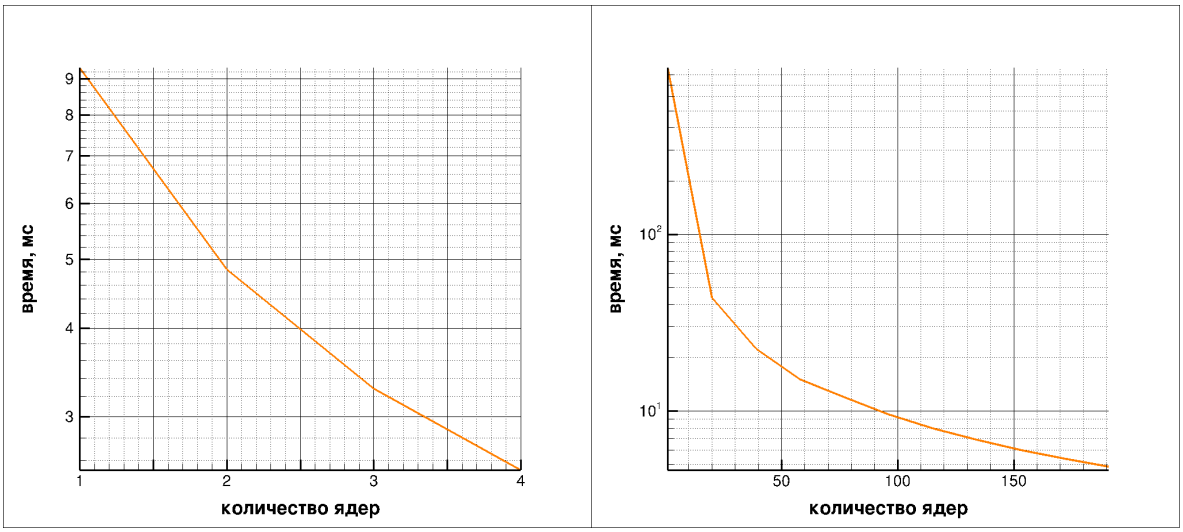
Для проведения измерений был реализован программный вычислительный код. Часть кода, выполняющаяся на центральном процессоре, написана на языке программирования C++ и использует технологию OpenMP для параллелизации между ядрами. В качестве технологии, используемой для управления графическим процессором, использовалась CUDA, поскольку в настоящее время она является наиболее удобной и эффективной. Соответственно часть кода, выполняющаяся на GPU, написана на C-подобном языке программирования, реализованном в рамках данной технологии.

Компиляция кода для центрального процессора производилась при помощи компилятора gcc 4.9 с флагом оптимизации – O2, а для компиляции GPU кода – nvcc 7.0.

5.1 Эффективность параллелизации

Прежде чем переходить к оценке ускорения, которое можно достигнуть при совместном использовании центрального и графического процессоров, необходимо исследовать эффективность параллелизации на центральном и графическом процессорах отдельно.

Для этого произведен ряд измерений времени расчета модельной задачи на обоих устройствах по отдельности. На рисунке 7 представлены зависимости времени вычисления модельной задачи от количества ядер для CPU и GPU первой конфигурации оборудования. Здесь и далее, для фильтрации различных случайных событий, каждое измерение производилось не менее 20 раз, после чего производилось осреднение результата.



а *б*

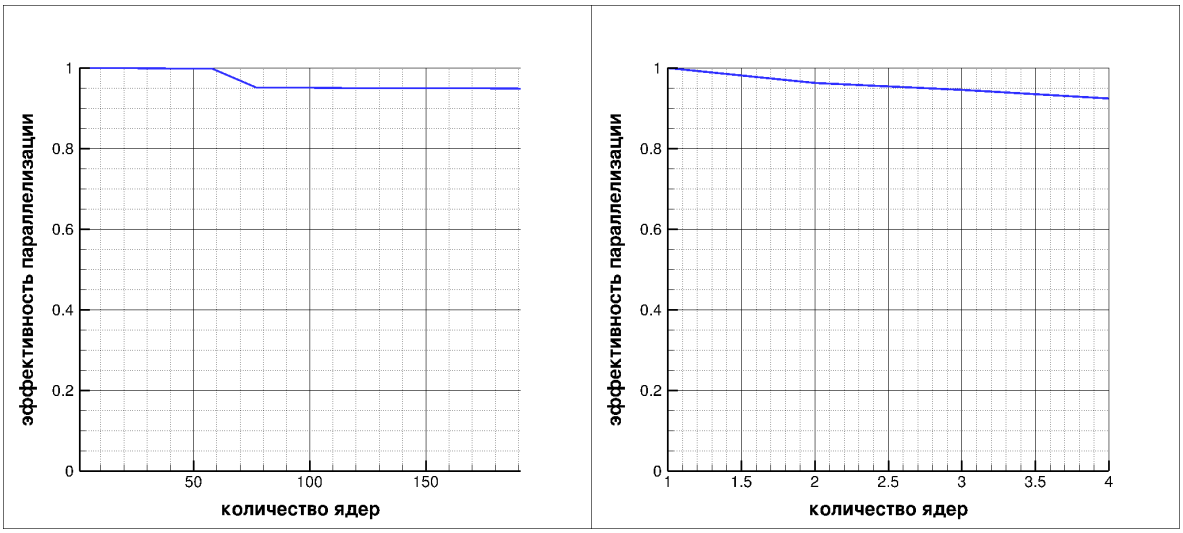
Рис. 7 Зависимость времени расчёта от количества ядер на: CPU (а) и GPU (б) для первой конфигурации оборудования

Из приведенных графиков видно, что время вычисления на обоих устройствах с увеличением количества используемых ядер уменьшается. Кроме того, в данной конфигурации оборудования графический процессор уступает по мощно-

сти центральному: при использовании всех ядер время работы на CPU составило 2 мс, а на GPU – 5 мс.

Эффективность параллелизации, представленная на рисунке 8, оценивается следующим образом:

$$\text{эффективность} = \frac{\text{время расчета на одном ядре}}{(\text{время расчета на нескольких ядрах}) \cdot (\text{количество ядер})}$$

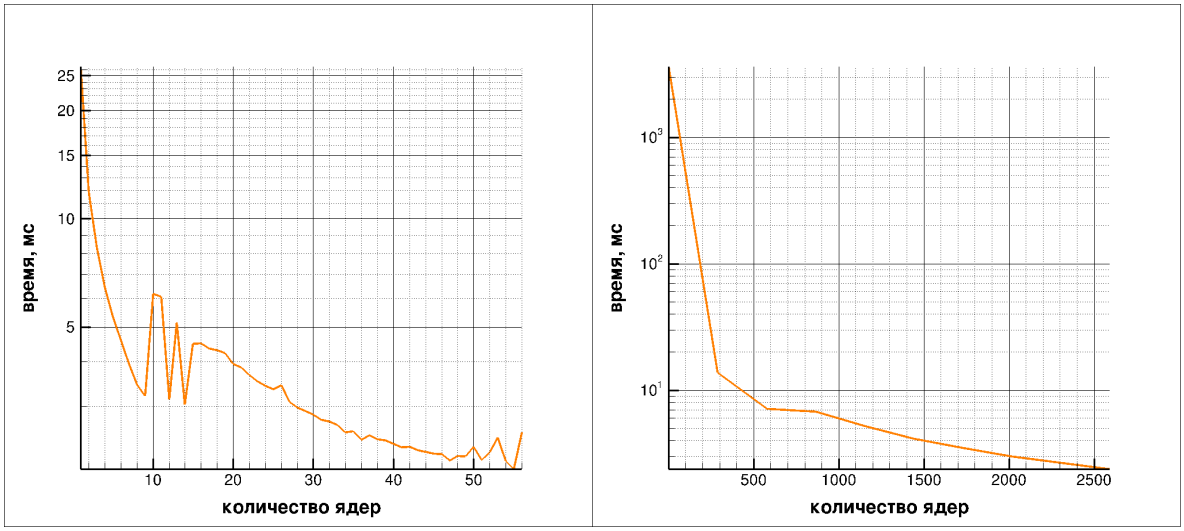


а *б*

Рис. 8 Эффективность параллелизации на: CPU (а) и GPU (б) для первой конфигурации оборудования

Видно, что эффективность параллелизации падает при увеличении количества ядер. Это связано с тем, что при параллелизации некоторое время отводится на распределение задачи между потоками. Однако для данной конфигурации оборудования это значение держится выше отметки 0.9.

Зависимость времени расчета задачи и эффективности параллелизации от количества используемых ядер для второй конфигурации оборудования представлена на рисунках 9 и 10 соответственно.



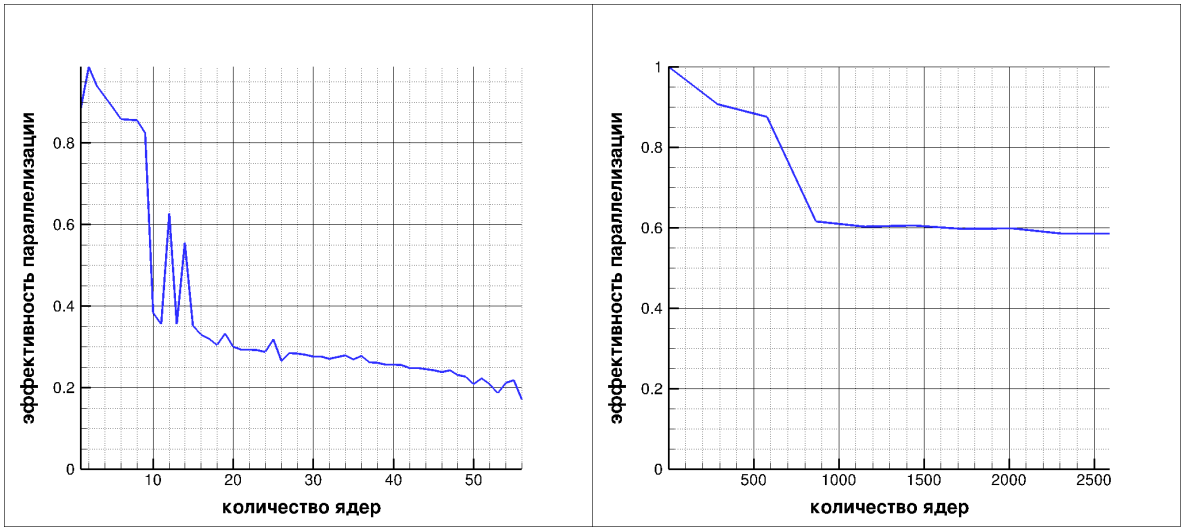
а *б*

Рис. 9 Зависимость времени расчёта от количества ядер на: CPU (а) и GPU(б) для второй конфигурации оборудования

Как и для первой конфигурации оборудования наблюдается замедляющееся падение значения времени расчета задачи при увеличении числа используемых ядер. Причем время расчета задачи на CPU и на GPU составляет приблизительно 2 мс при использовании всех ядер. Таким образом, в отличие от первой конфигурации оборудования, на второй графический процессор не уступает по производительности центральному.

Для данного оборудования наблюдается значительное превосходство эффективности параллелизации на графическом процессоре про сравнению с центральным.

Таким образом, во всех случаях время расчета задачи уменьшается с увеличением количества используемых ядер, поэтому при дальнейших расчетах для получения максимального ускорения будут использоваться все доступные ядра оборудования.



а *б*

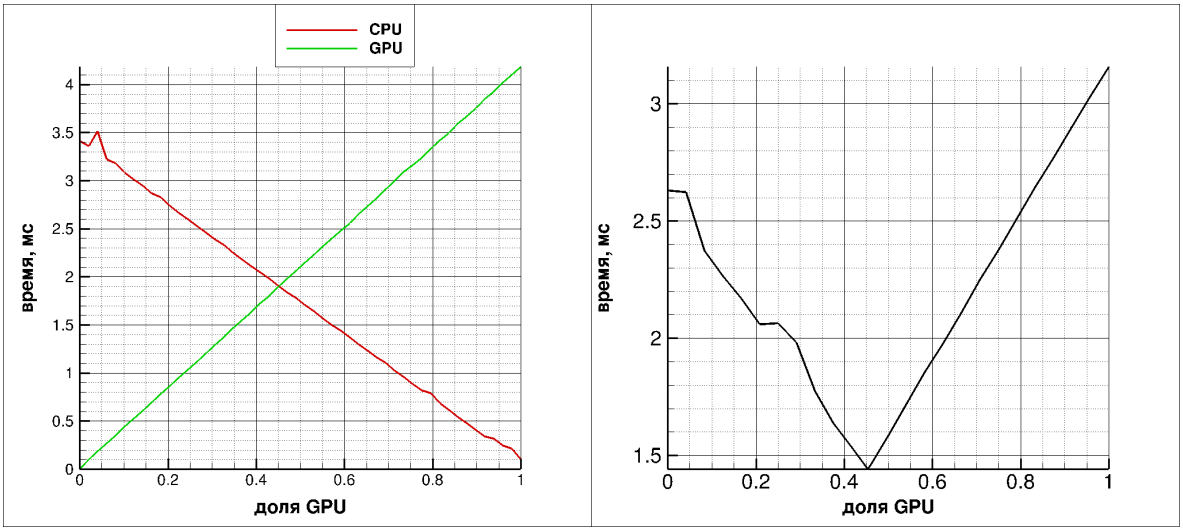
Рис. 10 Эффективность параллелизации на: CPU (а) и GPU (б) для второй конфигурации оборудования

5.2 Вычисление максимального ускорения

Для получения максимального ускорения так же требуется эффективно распределить данную задачу между центральным и графическим процессором. Такое распределение будет зависеть как от оборудования, так и от параметров задачи. Поскольку передача данных на видеокарту является потерей времени, для оценки максимального ускорения была выбрана задача с практически нулевым объемом обрабатываемых данных. Число производимых операций равнялось 3×10^6.

Была проведена серия расчетов при совместной работе центрального и графического процессоров. Так как эти устройства могут работать одновременно, максимальное ускорение достигается при их одновременном завершении работы.

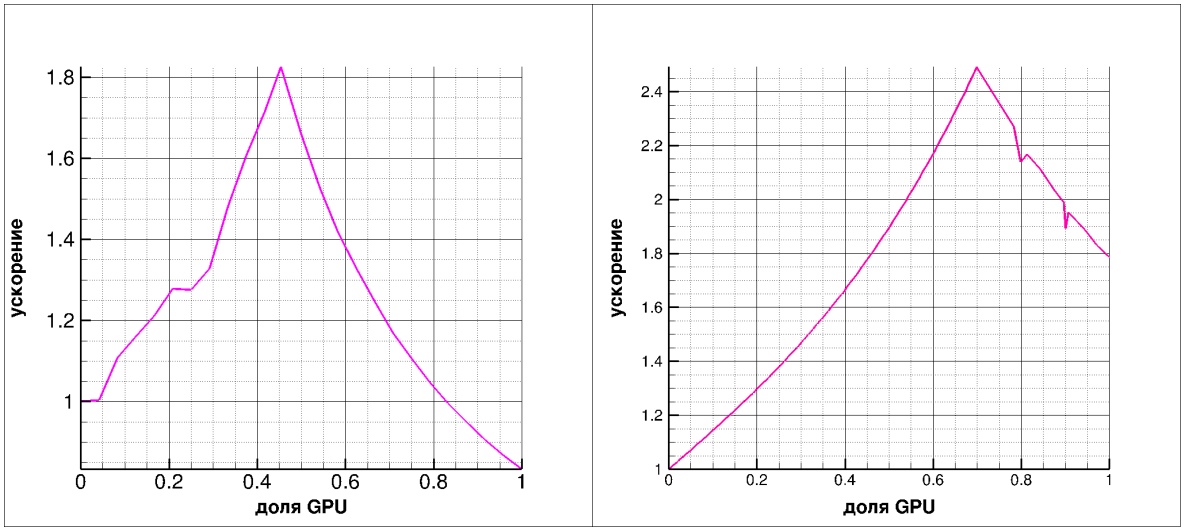
На рисунке 11 представлена зависимость времени вычисления на CPU и GPU отдельно и общее время расчета задачи от распределения операций между устройствами на первой конфигурации оборудования. Видно, что максимальное ускорение получается, если на видеокарту отправить приблизительно 45% задачи.



а *б*

Рис. 11 Зависимость времени раздельного вычисления на CPU и GPU (а) и совместного вычисления CPU и GPU (б) от распределения задачи между устройствами

Полученное распределение ускорения при совместном расчете на обоих устройствах относительно расчета только на центральном процессоре для обеих конфигураций оборудования представлено на рисунке 12. Видно, что максимальное ускорение составляет приблизительно 1.8 для первой конфигурации оборудования и 2.5 для второй. При этом оптимальная доля ресурсов, которую надо послать на видеокарту для получения такого ускорения, составила приблизительно 45% для первой конфигурации оборудования и 70% для второй.



а *б*

Рис. 12 Зависимость ускорения при совместном вычислении на CPU и GPU от распределения данных между устройствами для первой (а) и второй (б) конфигураций оборудования

5.3 Зависимость ускорения от количества передаваемых данных

Было произведено 50 измерений при фиксированном количестве операций и варьируемом размере для всех рассматриваемых конфигураций вычислительной техники (см. рис 13).

а *б*

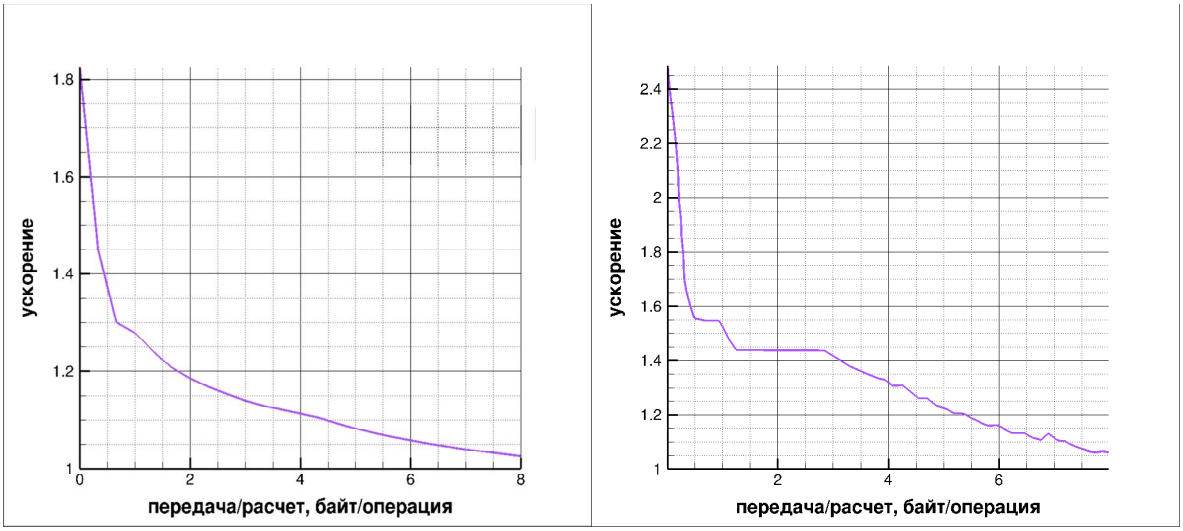
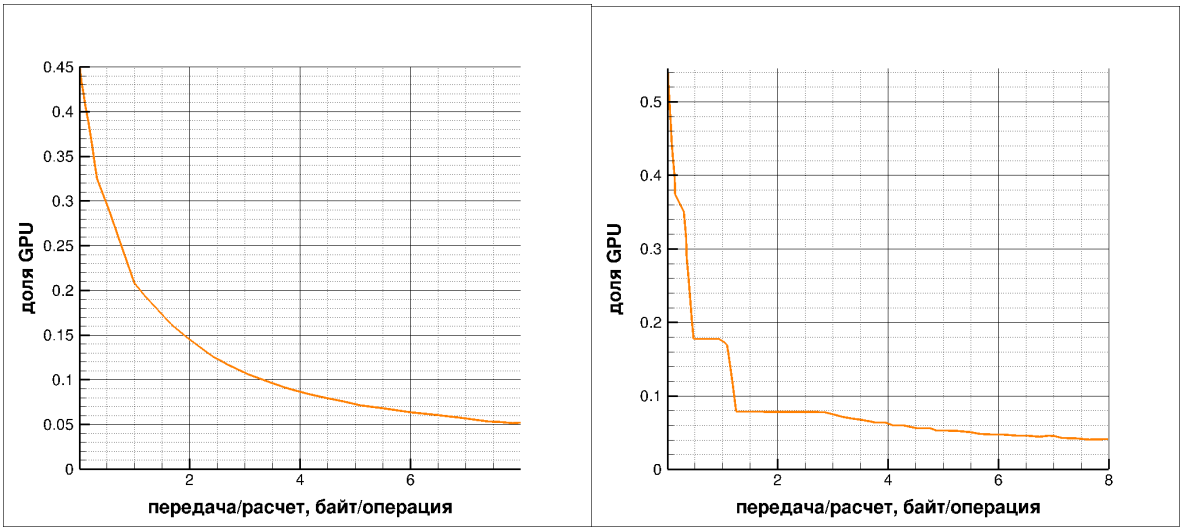


Рис 13 Зависимость максимального ускорения при совместном вычислении на CPU и GPU от соотношения передача/расчет для первой (а) и второй (б) конфигураций оборудования

Видно, что наиболее эффективно ускоряются те алгоритмы, для которых соотношение $\frac{\text{передача}}{\text{расчет}}$ минимально. Поэтому при увеличении данного соотношения необходимо уменьшать долю задачи, отправляемую на видеокарту. Распределение оптимальной доли ресурсов для данной модельной задачи изображено на рисунке 14.



а *б*

Рис. 14: Зависимость максимального ускорения при совместном вычислении на CPU и GPU от соотношения передача/расчет для первой (а) и второй (б) конфигураций оборудования

5.4 Влияние размера задачи на получаемое ускорение

Теперь необходимо проверить, что полученное распределение не зависит от размера задачи. Для исследования зависимости ускорения от размера задачи, произведены две серии расчетов. В первой серии количество операций составило 26×10^6, а во второй – 39×10^6. При этом размер обрабатываемых данных изменялся в диапазоне от 0 до 100 Мб в случае первой серии расчетов и от 0 до 150 Мб в случае второй. Измерения производились на первой конфигурации оборудования. Результат представлен на рисунке 15.

Рис. 15 Зависимость максимального ускорения при совместном вычислении на CPU и GPU от соотношения передача/расчет первого и второго наборов параметров измерения

Распределения ускорения практически совпали: максимальная разница составила порядка 3%. Это значит, что единственным критерием оценки ускорения данного алгоритма для задач с размером обрабатываемых данных не более 150 Мб, является соотношение $\frac{\text{передача}}{\text{расчет}}$.

5.5 Проверка точности оценки ускорения на примере алгоритма трехдиагональной прогонки

Убедившись в том, что соотношение $\frac{\text{передача}}{\text{расчет}}$ является достаточным параметром для оценки ускорения, становится возможным произвести сопоставление оценочного и реального ускорения любого алгоритма. Для этого был реализован вычислительный код, производящий решение системы линейных алгебраических уравнений с трехдиагональной матрицей.

Соотношение $\frac{\text{передача}}{\text{расчет}}$ для данного алгоритма равняется 6.7. В таком случае, оценка ускорения алгоритма, исходя из полученных результатов на графике 13, будет равняться 1.05 на первом оборудовании и 1.13 на втором.

Произведен ряд расчетов данной задачи. На первой конфигурации оборудования полученное ускорение равняется 1.05, а на второй конфигурации - 1.2. Этот результат говорит о достаточно высокой точности используемой методики оценки ускорения.

6 Оценка ускорения NTS-кода

Оценка ускорения NTS-кода произведена исходя из распределений ускорения, представленных на рисунке 13. Для этого были вычислены значения соотношения $\frac{\text{передача}}{\text{расчет}}$ для наиболее затратных алгоритмов данного кода (см. таблицу 5).

Таблица 5 Значение параметров алгоритмов NTS-кода

Функция	Описание	$\frac{\text{передача}}{\text{расчет}}$
jacob_gas	Вычисление матрицы Якоби	1.5
vectorsolve3d	Решение матрицы левых частей	5
block_lhs	Подготовка к решению левых частей	1.3
jacob_scal	Вычисление матрицы Якоби	22

Используя эти данные можно оценить ускорение данных алгоритмов из полученных распределений ускорений, представленных на графике 13. Результаты оценки представлены в таблице 6.

Таблица 6 ускорения алгоритмов NTS-кода для различных конфигураций оборудования

			Ускорение при совместной работе CPU и GPU	
Алгоритм	Процент времени	Передача/расчет	Первая конфигурация оборудования	Вторая конфигурация оборудования
jacob_gas	25.1%	1.5	1.23	1.45
vectorsolve3d	15.9%	5	1.08	1.22
block_lhs	14%	1.3	1.28	1.55
jacob_scal	11.3%	22	1	1

Учитывая долю расчетов, приходящуюся на рассматриваемые процедуры, можно оценить общее ускорение NTS-кода при применении графического процессора. Для первой конфигурации оборудования удастся ускорить NTS-код на 9%, а при использовании второй конфигурации – на 15%.

7. Заключение

Выяснено, что вычисление явных алгоритмов в NTS-коде составляет 66% от полного времени выполнения программы. В зависимости от конфигурации оборудования возможное ускорение составляет 9 – 15%.

Столь незначительное ускорение делает практически бессмысленным распараллеливание NTS кода с использованием GPU для тех систем, в которых расчетная мощность GPU несущественно превышает мощность CPU.

8. Список литературы

1. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / А. В. Боресков и др. Предисл.: В. А. Садовничий. 2-е издание. – М.: Издательство Московского Университета. 2015. – 336 с., илл. – (Серия «Суперкомпьютерное образование»)
2. Adam D. P. Gpu Accelerated Approach To Numerical Linear Algebra And Matrix Analysis With Cfd Applications. 2014.
3. Ruipeng L., Yousef S. GPU-accelerated preconditioned iterative linear solvers // The Journal of Supercomputing. 2013. Vol. 63(2). P. 443-466.
4. Vasily V., James W. D. Benchmarking GPUs to Tune Dense Linear Algebra // 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis. 2008. P. 1-11
5. Everett H. P., Roger L. D., John D. O. Unsteady Turbulent Simulations On A Cluster Of Graphics Processors // 40th Fluid Dynamics Conference and Exhibit. 2010.
6. Chuanfu X., Xiaogang D., Lilun Z., Yi J., Wei C., Jianbin F., Yonggang C., Yongxian W., Wei L. Parallelizing a High-Order CFD Software for 3D, Multi-block, Structural Grids on the TianHe-1A Supercomputer // 28th International Supercomputing Conference. 2013. P. 26-39.
7. Jialin L., Yidong X., Lixiang L., Hong L., Jack E., Frank M. OpenACC directive-based GPU acceleration of an implicit reconstructed discontinuous Galerkin method for compressible flows on 3D unstructured grids // 54th AIAA Aerospace Sciences Meeting. 2016.