УДК 532.529

ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧ ГИДРОДИНАМИКИ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ ОБЩЕГО НАЗНАЧЕНИЯ

К. Н. Волков[1], В. Н. Емельянов[1], А. Г. Карпенко[2], И. В. Курова[1], А. Е. Серов[1], П. Г. Смирнов[1]

Обсуждаются возможности использования графических процессоров общего назначения для численного решения задач гидродинамики. Для программной реализации параллельных вычислительных алгоритмов применяется технология CUDA. Приводится решение ряда модельных задач на графических процессорах и обсуждаются подходы к оптимизации программного кода, связанные с использованием различных типов памяти. Рассматриваются особенности реализации схемы расщепления (метод проекции), предназначенной для моделирования течений вязкой несжимаемой жидкости. Сравнивается ускорение счета на графических процессорах по отношению к расчетам на центральном процессоре при использовании сеток различной разрешающей способности и различных способах разбиения исходных данных на блоки.

Ключевые слова: графические процессоры общего назначения, параллельные алгоритмы, механика жидкости и газа, сеточные методы, технология CUDA.

1. Введение. Увеличение вычислительной мощности одноядерных процессоров за счет повышения их тактовой частоты и архитектурных усовершенствований представляется нерентабельным. Производители микропроцессоров переходят на разработку многоядерных (multicore) процессоров с новой архитектурой, обеспечивающей распараллеливание обработки данных [1, 2].

Среди многоядерных процессоров с параллельной архитектурой наиболее известными являются центральные процессоры (ЦПУ) и графические процессоры (ГПУ). Графические процессоры общего назначения обладают собственной динамической памятью и содержат множество мультипроцессоров, которые управляют высокоскоростной памятью, что делает их использование эффективным как для графических, так и для неграфических вычислений (например, платформы NVIDIA Tesla, Fermi и Kepler специально разрабатываются для вычислительных приложений).

Структурные отличия ЦПУ и ГПУ приводят к тому, что теоретическая производительность ГПУ, измеряемая количеством арифметических и логических операций в единицу времени, значительно превосходит теоретическую производительность ЦПУ (рис. 1). Значки • соответствуют вычислениям с двойной точностью на ЦПУ компании Intel, а значки ∘ и □ — вычислениям с одинарной и двойной точностью на ГПУ компании NVIDIA.

В имеющихся публикациях обсуждаются вопросы, связанные с производительностью вычислений на ГПУ и реализацией конкретных вычислительных задач [1–3]. Прирост производительности вычислений на ГПУ по сравнению с вычислениями на ЦПУ отличается в существенной степени, изменяясь от десятков до сотен раз [1–4]. Несмотря на то что потенциальные возможности ГПУ при решении широкого круга прикладных задач не подлежат сомнению, технологические вопросы реализации вычислительных задач на ГПУ общего назначения требуют дальнейшего развития.

Для поддержки неграфических приложений на ГПУ компании NVIDIA используется унифицированная архитектура компьютерных вычислений CUDA (Compute Unified Device Architecture), основанная на расширении языка Си и позволяющая получить доступ к набору инструкций ГПУ для управления его памятью [2, 3]. При этом ГПУ рассматривается как вычислительное устройство, способное поддерживать параллельное исполнение большого числа нитей или потоковых программ.

В настоящей статье обсуждается применение технологии CUDA для решения задач, связанных с моделированием течений жидкости и газа. Рассматриваются особенности реализации программного кода

[1] Балтийский государственный технический университет “Военмех” им. Д. Ф. Устинова, факультет энергетического машиностроения, 1-я Красноармейская ул., д. 1, 190005, Санкт-Петербург; К. Н. Волков, вед. науч. сотр., e-mail: dsci@mail.ru; В. Н. Емельянов, профессор, e-mail: vlademelyanov @ gmail.com; И. В. Курова, доцент, e-mail: yaiv@mail.ru; А. Е. Серов, аспирант, e-mail: ceroff@mail.ru; П. Г. Смирнов, аспирант, e-mail: petr.s.8314@mail.ru

[2] Санкт-Петербургский государственный университет, математико-механический факультет, Петродворец, Университетский просп., 28, 198504, Санкт-Петербург; аспирант, e-mail: aspera.2003.ru@mail.ru

на ГПУ и ряд вопросов, связанных с его оптимизацией. Приводятся детали реализации ряда частных подзадач, а также схемы расщепления по физическим процессам, предназначенной для моделирования течений вязкой несжимаемой жидкости. Сравнивается ускорение решения задачи на ГПУ по сравнению с расчетами на ЦПУ при использовании сеток различной разрешающей способности и различных способах разбиения исходных данных на блоки. Расчеты проводятся на центральном процессоре Intel Core 2 Duo с тактовой частотой 3 ГГц и графической плате NVIDIA GeForce GTX 480.

2. Устройство и использование графических процессоров. У современных ЦПУ имеется небольшое количество арифметико-логических устройств (АЛУ), контроллер управления, отвечающий за передачу следующей машинной инструкции АЛУ и за ряд других функций, контроллер доступа к внешней памяти и внешняя ОЗУ-память. Каждое АЛУ содержит набор регистров и свой сопроцессор для расчета сложных функций.

Структура и организация памяти ГПУ является более сложной, чем ЦПУ. В ГПУ находится большое количество АЛУ, несколько контроллеров управления и внешняя память. В отличие от ЦПУ, где имеется один тип памяти с несколькими уровнями кэширования в самом процессоре, ГПУ обладает более сложной организацией памяти [2, 3], связанной с их назначением (обработка графической информации). Часть памяти располагается непосредственно в каждом из потоковых мультипроцессоров (регистровая и разделяемая память), а часть памяти размещается в ОЗУ (локальная, глобальная, константная и текстурная памяти).

Для ГПУ требуется разработка алгоритмов, имеющих высокую степень параллелизма на уровне данных. При этом одна операция выполняется над всеми элементами массива данных (под элементом массива понимается структура данных или несколько чисел, хранящихся во внешней памяти). Из-за того, что пропускная способность канала передачи данных между АЛУ и внешней памятью ограничена (это является одним из наиболее слабых мест ГПУ), наиболее продуктивными оказываются алгоритмы, в которых минимизируется число обращений к внешней памяти, а отношение количества операций над данными к количеству обращений к внешней памяти является максимальным.

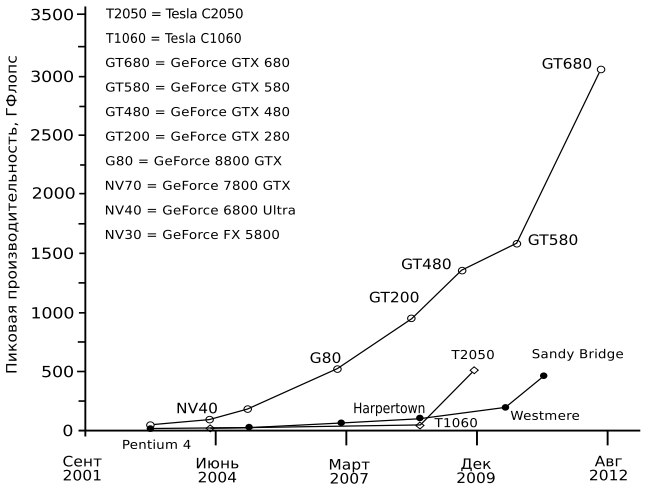

Рис. 1. Пиковая производительность ЦПУ и ГПУ

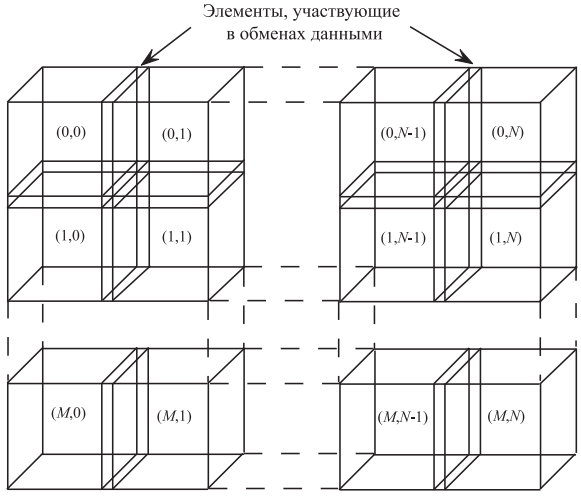

Рис. 2. Распределение данных на ГПУ

Программный код состоит из ЦПУ-кода и ядра (kernel), которое исполняется в несколько нитей (thread) с локальными переменными. Ядра содержат относительно небольшое количество операций, обрабатывающих элементы данных и обменивающихся данными при помощи нитей. Каждой нити, выполняющей ядро, дается уникальный идентификатор (целое число), доступный внутри ядра через встроенную переменную. На каждом шаге ядро берет по одному элементу данных из каждой входной нити, выполняет обработку данных и подает один или несколько элементов данных на выход.

3. Схема решения задачи. Для реализации параллельных вычислений на ГПУ массив входных данных (например, массив, содержащий значения искомой функции в узлах сетки) разбивается на ряд блоков (рис. 2). Блоки образуют сетку, которая имеет размер $M \times N$. Для обеспечения корректности вычислений имеется пересечение между соседними блоками, что позволяет осуществить обмен граничными данными между блоками.

Обмен данными между блоками обычно производится при помощи использования разделяемой памяти (использования разделяемой памяти удается избежать в том случае, когда одна нить обрабатывает

данные в одной ячейке сетки).

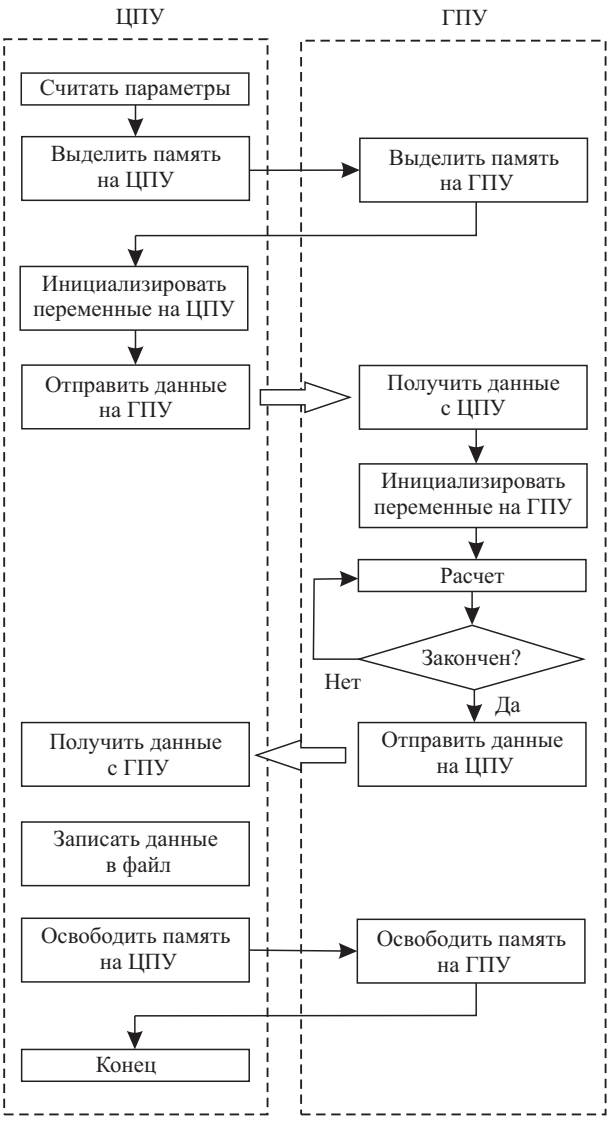

Рис. 3. Схема решения задачи при использовании ресурсов ГПУ

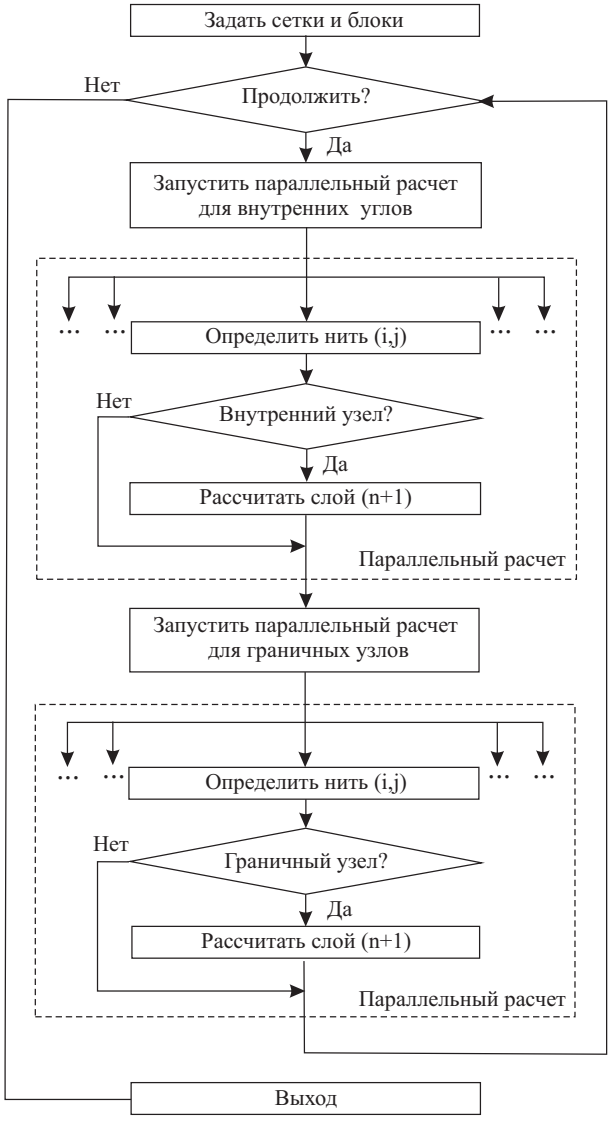

Рис. 4. Схема реализации итерационного процесса при использовании ресурсов ГПУ

Схему решения задачи на ГПУ поясняет рис. 3. Стрелки $\longrightarrow$ соответствуют командам, а стрелки $\Longrightarrow$ — командам и обменам данными между ЦПУ и ГПУ. Подготовка исходных данных (чтение с жесткого диска, формирование и инициализация массивов, сортировка) осуществляется на ЦПУ. Массивы данных копируются на ГПУ, где над массивами данных производится заданная последовательность действий. Каждая нить обрабатывает один узел сетки. Результирующие массивы копируются с ГПУ на ЦПУ, после чего производится обработка и визуализация данных, а также их сохранение на жесткий диск.

При использовании метода конечных разностей или метода конечных объемов решение дифференциальных уравнений в частных производных реализуется по схеме, показанной на рис. 4. Подготовка исходных данных (координаты узлов сетки, граничные условия, структуры данных для хранения топологии неструктурированной сетки) производится на ЦПУ. Осуществляется копирование данных в глобальную память ГПУ и инициализация соответствующих переменных. На ГПУ производится расчет невязких и вязких потоков, продвижение решения во времени и решение системы разностных уравнений тем или иным итерационным методом. Расчеты во внутренних и граничных узлах проводятся на ГПУ при помощи различных ядер (по разным правилам). Проверка сходимости численного решения производится на ЦПУ. Синхронизация между ЦПУ и ГПУ требуется при проверке условия сходимости, когда норма невязки копируется в память ЦПУ. При выполнении условия сходимости переменные переносятся из ГПУ в память ЦПУ.

Для проверки выполнения условия Куранта–Фридрихса–Леви (условие устойчивости) на каждом шаге по времени ищется максимальная скорость звука по всем ячейкам сетки. Для этого используется метод редукционного суммирования и его модификации [2].

4. Расчет потоков. При реализации разностных схем расчета потоков данные, требуемые на шаге эволюции, рассчитываются перед началом шага по времени и хранятся в текстурной памяти. Результаты, полученные на шаге по времени, хранятся в глобальной памяти. Вместо обменов данных, принадлежащих соседним блокам и соответствующих фиктивным ячейкам, проводится расчет параметров в фиктивных ячейках каждого блока.

Одна из проблем при реализации многих итерационных методов на ГПУ заключается в большом числе точек синхронизации вычислений, что приводит к увеличению количества операций чтения/записи для данных в медленной глобальной памяти устройства и сокращению числа вычислений в быстрой памяти. Повысить быстродействие итерационного метода на графических процессорах удается за счет группировки вычислений для уменьшения обращений к медленной памяти и выполнения асинхронных операций копирования памяти устройства.

При использовании неструктурированных сеток затрудняется объединение нескольких запросов к глобальной памяти в один (coalesced memory access) вследствие неупорядоченной нумерации узлов сетки, граней и ячеек (ядро, выполняющее расчет потоков, осуществляет перебор всех граней сетки, связанных с узлом).

В этой связи имеются существенные различия в реализации метода конечных объемов на неструктурированной сетке в случае, когда контрольный объем совпадает с ячейкой сетки [5] (cell-centered scheme), и в случае, когда контрольный объем строится около узла сетки (vertex-centered scheme) [6]. При использовании контрольных объемов, совпадающих с ячейками сетки, ядро выполняет перебор фиксированного числа граней и фиксированного числа соседних узлов. При использовании схемы, в которой контрольные объемы строятся вокруг узлов сетки, число граней, связанных с данным узлом, не является фиксированным, что приводит к дополнительным обращениям к глобальной памяти.

Для оптимизации работы с глобальной памятью используется схема перенумерации узлов (renumbering scheme), которая гарантирует, что грани, участвующие в параллельном вычислении потоков, располагаются по соседству в глобальной памяти. Такой подход увеличивает производительность вычислений приблизительно на 70% в трехмерном случае [6].

В случае расчета потоков с репликацией данных для исключения конфликтов по доступу к памяти на этапе вычисления потоков создается дополнительный массив для хранения потоков по граням контрольных объемов. Такой подход приводит к увеличению потребления оперативной памяти, поскольку для каждой грани записывается два набора из пяти значений (плотность, три компоненты скорости и давление). После вычисления потоков через грани для получения конечных значений в контрольных объемах производится суммирование потоков. Для этого требуется хранить в памяти дуальный граф связей контрольных объемов (вершинами графа являются контрольные объемы, а ребрами — связи контрольных объемов через их общие грани), который содержит $2n_c + n_e + 1$ целочисленных значений, где n_c — число ячеек сетки и n_e — число внутренних граней контрольных объемов. Достоинство похода заключается в том, что все задания по обработке внутренних граней контрольных объемов запускаются одновременно.

При потактовом вычислении потоков к дуальному графу сетки применяется алгоритм раскраски ребер (edge coloring scheme). Из множества ребер графа выделяется такое подмножество ребер, в котором каждый из узлов встречается не более одного раза. Обработка такого подмножества ребер нитями, запущенными на ГПУ, происходит параллельно и исключает возможность конфликтов нитей по модификации одних и тех же ячеек памяти. Процедура расчета потоков разбивается на такты. На каждом из таких тактов обрабатывается одно из выделенных подмножеств ребер сетки. Число тактов по вычислению потоков превосходит максимальное число граней контрольного объема (например, для сетки с ячейками в виде гексаэдра число тактов превышает шесть), а число одновременно обрабатываемых на каждом из тактов граней не превосходит $n_e/2$. Необходимость многократного запуска процедуры вычисления потоков для каждого из тактов и снижение числа одновременно обрабатываемых граней составляют отрицательные черты подхода. Однако при этом отсутствует необходимость в выделении дополнительных объемов оперативной памяти и лишнем суммировании вычисленных по граням потоков в ячейки.

Сравнение, проведенное в работе [4], показывает, что метод вычисления потоков с репликацией оказывается на 17% быстрее, чем метод потактового вычисления потоков.

5. Решение уравнения Лапласа. Рассмотрим решение уравнения Лапласа $\Delta u = 0$ в двумерной области, представляющей собой квадрат с единичной стороной $\Omega = (0,1) \times (0,1)$. На левой, правой и нижней границах полагается, что $u = 0$, а на верхней границе $u = 1$. Краевая задача Дирихле для

уравнения Лапласа описывает установившееся распределение скорости в квадратной каверне с подвижной верхней стенкой при малых числах Рейнольдса (конвективными слагаемыми пренебрегается по сравнению с вкладом вязких слагаемых).

Для дискретизации уравнения Лапласа используется метод конечных разностей на равномерной прямоугольной сетке (используется шаблон типа "крест"). Система разностных уравнений решается методом Гаусса–Зейделя или методом последовательной верхней релаксации с использованием красно-черной (red/black) параллелизации. Отображение вычислительной области на структуру памяти ГПУ поясняет рис. 5 (на этом рисунке красному цвету соответствует серый). Расчеты проводятся на сетке, содержащей `imax`×`jmax` узлов. При этом ряд узлов используется для постановки граничных условий и не обрабатывается кодом (соответствующие ячейки выделяются белым цветом). Код осуществляет обработку `imx`×`jmx` узлов.

Явная разностная схема для уравнения Лапласа на равномерной сетке с использованием географической нотации для узлов сетки (n = north, s = south, e = east, w = west) имеет вид $u_p^{n+1} = \dfrac{1}{a_p}\left(a_n u_n^n + a_s u_s^n + a_e u_e^n + a_w u_w^n\right)$, где $a_p = a_n + a_s + a_e + a_w$. Решение на итерации $n+1$ находится по формуле $u_p^{n+1} := \omega u_p^{n+1} + (1-\omega)u_p^n$, где ω — параметр релаксации. Узлы n, s, e и w, окружающие узел p, имеют одинаковый цвет.

Реализация кода, вызываемого на ЦПУ, заключается в указании размера блока и размера сетки, а также в вызове ядер, запускаемых на ГПУ для обработки красных узлов и черных узлов.

Код на ГПУ реализуется в виде следующей последовательности шагов: динамическое выделение памяти; постановка начальных и граничных условий; расчет коэффициентов a_n, a_s, a_e и a_w; выделение памяти на ГПУ и копирование переменных с ЦПУ на ГПУ; задание размеров блоков и сетки; вызов ядра для красных узлов и вызов ядра для черных узлов; копирование результатов с ГПУ на ЦПУ.

Реализация кода, запускаемого на ГПУ, заключается в обработке узлов одного цвета (ядро для красных узлов и ядро для черных узлов реализуются по одной и той же схеме). На ГПУ решение хранится в виде матрицы размером $m \times n$ (в одной матрице хранятся значения искомой функции как во внутренних, так и в граничных узлах сетки). Вычисления производятся параллельно по $n/2$ блоков, каждый из которых содержит $m-2$ нити. Каждая нить в блоке записывает по одному значению в разделяемую память, что позволяет уменьшить вероятность двух одновременных обращений к одному и тому же элементу динамической памяти и ускоряет доступ нитей к требуемым данным.

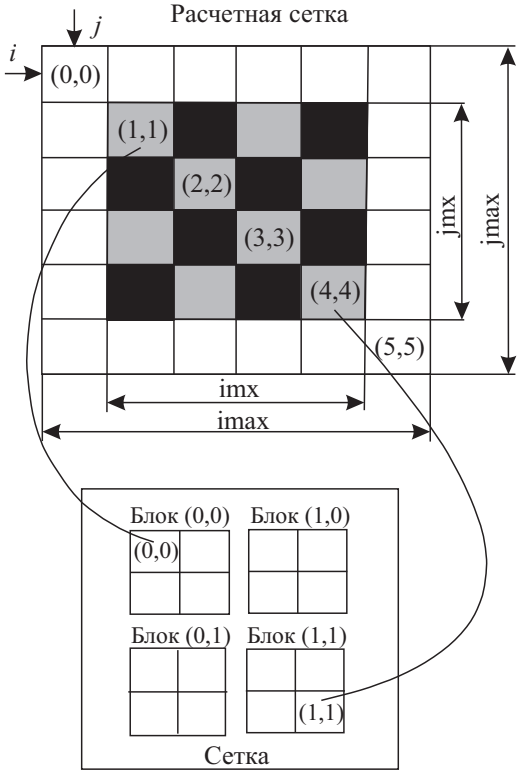

Рис. 5. Реализация метода Гаусса–Зейделя с использованием красно-черной параллелизации на ГПУ

Ускорение решения уравнений Лапласа на ГПУ по отношению к вычислениям на ЦПУ составляет около 20 (в расчетах используется только глобальная память).

6. Схема расщепления. Для решения уравнений Навье–Стокса, описывающих нестационарное течение вязкой несжимаемой жидкости, используется схема расщепления [7, 8] (метод проекции) и метод конечных разностей на равномерной сетке с шахматным расположением узлов [9]. Для дискретизации по времени применяется схема Адамса–Башфорта, а для дискретизации конвективных и диффузионных потоков — противопоточные и центрированные разности второго порядка. Уравнение Пуассона для давления решается методом Гаусса–Зейделя с использованием красно-черной параллелизации и методом бисопряженных градиентов со стабилизацией (BiConjugate Gradient Stabilized, BiCGStab).

Сетка размером $n_x \times n_y \times n_z$ на ГПУ представляется в виде набора из n_z матриц, каждая из которых имеет размер $n_x \times n_y$ (рис. 6). Ряд узлов используется для постановки граничных условий и не обрабатывается кодом. Такое отображение сетки с ЦПУ на ГПУ является удобным при использовании глобальной памяти.

Параллельный код на одном ГПУ, реализующий метод проекции, имеет два вложенных цикла, один из которых (внешний цикл) осуществляет продвижение решения во времени, а другой (внутренний цикл) —

интегрирование уравнения Пуассона для давления при помощи итерационного метода Якоби (или другого итерационного метода). При использовании схемы Эйлера решение на слое $n+1$ зависит только от значений искомых функций на слое n по времени. Для хранения скорости в узлах сетки на двух смежных слоях по времени используется шесть матриц (три матрицы на слой по времени). Обмен их значениями производится в конце каждого шага по времени. Для хранения давления на двух смежных слоях по времени используются две матрицы, а обмен их значениями осуществляется в конце каждой итерации.

Параллельный код на нескольких ГПУ имеет шесть ядер, обеспечивающих реализацию основных шагов метода проекции. Различные ядра требуются для того, чтобы обеспечить глобальную синхронизацию между блоками перед тем, как перейти к следующему слою по времени.

Для оптимизации вычислений используется разделяемая память. Блоки нитей копируют переменные из глобальной памяти в разделяемую память. Расчеты производятся нитями с использованием переменных, находящихся в разделяемой памяти. Перед выходом из ядра результаты расчета копируются из разделяемой памяти в глобальную. Для компенсации времени, затрачиваемого на обмен данными между глобальной и разделяемой памятью, увеличивается интенсивность вычислительной нагрузки, приходящейся на ядро. Один из способов состоит в том, чтобы увеличить размер подобласти, предназначающейся каждому блоку нитей.

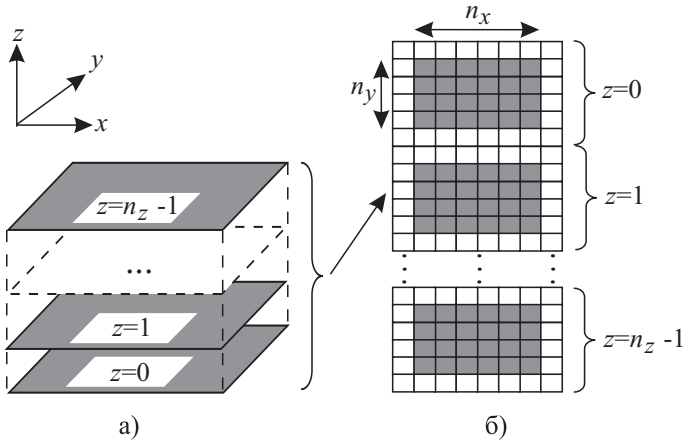

Рис. 6. Представление трехмерной сетки в вычислительной области (а) в виде набора матриц на ГПУ (б). Ячейки, выделенные белым цветом, являются фиктивными и используются для постановки граничных условий

Сравнение различных способов распределения данных, когда каждый блок нитей представляет собой 4×4-матрицу, показывает рис. 7. На фрагменте (а) блок отображается на подобласть, содержащую 4×4 вычислительных ячеек (108 ячеек, из которых 16 являются вычислительными, а 92 ячейки — фиктивными). Для обновления переменных в вычислительных ячейках подобласть и фиктивные ячейки копируются в разделяемую память. Для расчета переменных в 4×4 ячейках требуется скопировать в разделяемую память $6 \times 6 \times 3$ ячеек. При этом блок нитей обновляет менее 15% разделяемой памяти. Декомпозиция, приведенная на фрагменте (б) (144 ячейки, из которых 32 являются вычислительными, а 112 ячеек — фиктивными), позволяет нитям обновить переменные в ячейках, находящихся в различных вертикальных колонках. Каждая нить обновляет переменные в двух ячейках. Нити работают с $4 \times 4 \times 2$ ячейками, а общее число ячеек, вовлеченных в расчет, составляет $6 \times 6 \times 4$. При этом ячейки, в которых обновляются значения искомых функций, составляют 22% от их общего количества. Время, затрачиваемое на обмен данными между разделяемой и глобальной памятью, компенсируется увеличением вычислительной нагрузки, приходящейся на каждую нить.

Расчеты, проведенные в работе [10], показывают, что реализации ядер, предназначенных для расчета предварительного поля скорости и решения уравнения Пуассона для давления, оказываются наиболее критичными к использованию ресурсов разделяемой памяти. Использование разделяемой памяти позволяет получить почти двукратный выигрыш в производительности по сравнению с реализацией, использующей только глобальную память. Интенсивность вычислительной нагрузки, приходящейся на другие ядра, является сравнительно низкой.

Распараллеливание метода BiCGStab заключается в параллельной реализации матрично-векторных операций и операции скалярного произведения векторов, которые обсуждаются в работе [2]. Для их реализации используются функции библиотеки CUDBLAS. Ускорение вычислений составляет 12.

7. Многосеточный метод. Многосеточный метод используется для решения системы разностных уравнений, порожденной конечно-разностной или конечно-объемной дискретизацией уравнения Пуассона в методе проекции [9].

Отображение вычислительной сетки (массива данных) размером $n_x \times n_y \times n_z$ на блоки ГПУ поясняет рис. 8. Под n понимается число уровней сетки, используемых в многосеточном методе решения системы разностных уравнений. В качестве сглаживающего алгоритма используется метод Гаусса–Зейделя с красно-черной параллелизацией (применяется V-цикл).

Каждая нить производит расчет параметров в одной ячейке и обновляет один элемент массива дан-

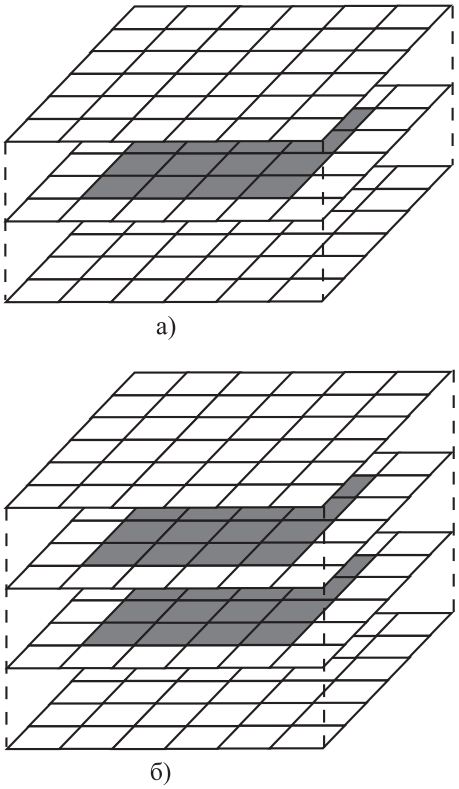

Рис. 7. Пиковая производительность ЦПУ и ГПУ. Различные подходы к использованию разделяемой памяти при использовании блока размером 4×4. Ячейки, закрашенные белым цветом, используются для постановки граничных условий

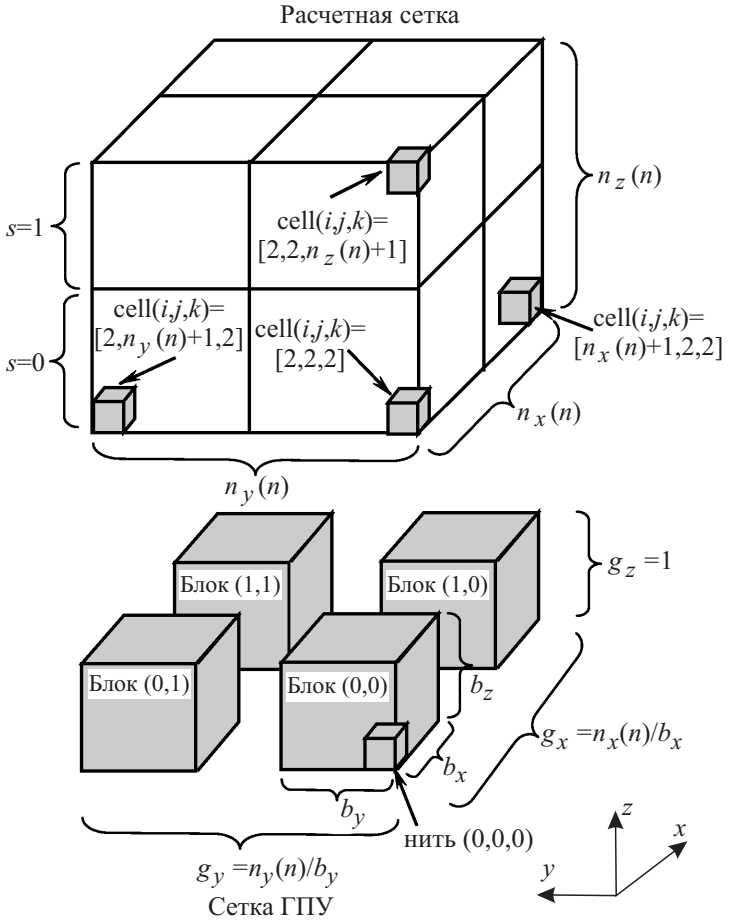

Рис. 8. Отображение сетки на блоки при реализации многосеточного метода

ных. Сетка блоков является двумерной структурой, в то время как блок нитей — трехмерной. Размерности массива в координатных направлениях y и z объединяются, а индекс каждой нити рассчитывается при помощи операции деления нацело.

Индексы внутренних ячеек изменяются от $(i, j, k) = (2, 2, 2)$ до $(i, j, k) = (n_x + 1, n_y + 1, n_z + 1)$. Граничные ячейки, не показанные на рисунке, располагаются вдоль плоскостей $(n_x + 2, n_y + 2, n_z + 2)$ и $(1, 1, 1)$. Сетка ГПУ имеет размер (g_x, g_y, g_z), а каждый блок — размер (b_x, b_y, b_z). Размеры сетки ГПУ в координатных направлениях g_x и g_y получаются путем деления размеров расчетной сетки на размер блока, поэтому $g_x = n_x/b_x$ и $g_y = n_y/b_y$. Нить имеет трехмерный индекс, равный индексу элемента массива, с которым она работает. В направлении оси z нить (i, j) имеет дело со слоями расчетной сетки, обрабатывая одну ячейку в каждом слое $s = n_z(n)/b_z - 1$. Нить обрабатывает множество ячеек в направлении k, находящихся в колонке с фиксированными индексами (i, j). Нить имеет индексы (t_x, t_y, t_z). Для расчета переменных во всех ячейках сетки внутри ядра используется цикл по всем слоям сетки (индексы i и j остаются фиксированными). Для постановки граничных условий используется массив, имеющий размер (g_x, g_y, g_z).

Размер блока задается исходя из размера расчетной сетки. Имеются конфликты между размером сетки и оптимальным размером блока. Наиболее грубой сеткой является сетка размером $4 \times 4 \times 4$, а оптимальный размер блока составляет $32 \times 1 \times 8$ (размер блока превышает размер сетки).

Код, выполняемый на ЦПУ, производит перебор всех сеточных уровней и осуществляет вызов ядра для фиксированного уровня сетки n.

Код, запускаемый на ГПУ, получает на вход номер обрабатываемого уровня сетки и выполняет необ-

ходимые вычисления.

Для увеличения эффективности многосеточных вычислений на ГПУ используется красно-черная параллелизация метода Гаусса–Зейделя. Для этого реализуются два ядра, осуществляющих обработку красных узлов и черных узлов. Код, выполняемый на ЦПУ, производит перебор всех сеточных уровней и вызывает ядра для обработки красных узлов и черных узлов. Код, выполняемый на ГПУ, реализует необходимые вычисления (ядра для серых узлов и черных узлов реализуются по схожей схеме).

Расчеты показывают, что наибольшие затраты компьютерного времени приходятся на работу сглаживающей процедуры, которая в зависимости от размера конечно-разностной сетки занимает от 72 до 78% от общего времени счета. При этом реализации процедур продолжения и ограничения требуют менее 8–10% и 12–20% от общего времени счета.

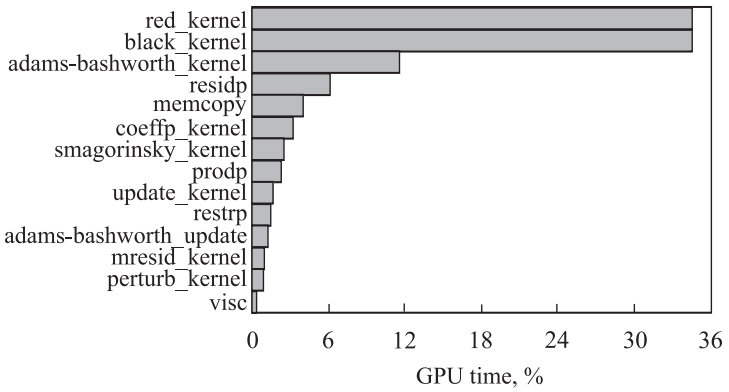

Рис. 9. Затраты времени на исполнение различных операций

Течение в каверне.
Время и ускорение вычислений

Сетка	$n_x = n_y = n_z = 4$		
	ЦПУ, с	ГПУ, с	Ускорение
16^3	0.34	0.39	0.86
32^3	3.08	1.24	2.50
64^3	31.14	6.49	4.81
128^3	291.05	50.92	5.72

Сетка	$n_x = 32$, $n_y = 1$, $n_z = 8$		
	ЦПУ, с	ГПУ, с	Ускорение
16^3	0.34	0.31	1.11
32^3	3.08	0.66	4.73
64^3	31.14	2.71	11.51
128^3	291.05	18.35	15.88

8. Течение в каверне. Рассмотрим моделирование крупных вихрей течения вязкой несжимаемой жидкости в каверне с подвижной верхней стенкой при $\mathrm{Re} = 10^3$, где Re — число Рейнольдса. Для дискретизации основных уравнений используется метод проекции [9], а в качестве модели подсеточной вязкости применяется модель Смагоринского [11]. Сетка наилучшей разрешающей способности содержит 128^3 узлов.

Затраты на исполнение различных участков кода показывает рис. 9. При этом время, затрачиваемое на реализацию метода Гаусса–Зейделя с красно-черной параллелизацией (функции `red_kernel` и `black_kernel`), составляет почти 2/3 от общего времени вычислений. Следующей по затратам процессорного времени является функция `adams_bashworth`, реализующая дискретизацию основных уравнений по времени. Функция `residp` осуществляет расчет невязки уравнения Пуассона для давления. Копирование и выделение памяти под переменные производится при помощи функции `memcopy`. Функции `coeffp_kernel` и `smagorinsky_kernel` производят расчет коэффициентов, связанных с дискретизацией уравнения Пуассона для давления, и расчет подсеточной вязкости на основе модели Смагоринского. Для перехода к следующему шагу интегрирования по времени используются функции `update_kernel` и `adams_bashworth_kernel`. Вклад других функций (`prodp`, `restrp`, `mresid_kernel`, `perturb_kernel`, `visc`), выполняющих вспомогательные операции, в общее время счета сравнительно невелик.

Ускорение счета при решении задачи на разных сетках с использованием различных способов разбиения области на блоки размером $n_x \times n_y \times n_z$ приводится в таблице (время расчета указывается для 100 шагов по времени).

На сетке, содержащей 128^3 узлов, изменение способа разбиения расчетной области на блоки позволяет получить выигрыш в ускорении решения задачи в 2.8 раза по сравнению с исходным разбиением.

9. Заключение. Рассмотрены подходы к решению задач механики жидкости и газа с использованием графических процессоров общего назначения. Для программной реализации кода используется технология CUDA для графических процессоров, производимых компанией NVIDIA.

Приведено решение ряда модельных задач, на основе которых сопоставлены ускорение счета на сетках различной разрешающей способности при использовании разных способов разбиения исходных данных на блоки. Для моделирования течений вязкой несжимаемой жидкости применяется схема расщепления по физическим процессам. Уравнение Пуассона для давления решается при помощи многосеточного метода и метода BiCGStab. Использование ГПУ позволяет ускорить расчеты в 2–50 раз (в зависимости от

постановки задачи и используемых вычислительных алгоритмов).

Разработанные средства численного моделирования включены в состав пакета программ Логос. Пакет Логос разработан в Институте теоретической и математической физики РФЯЦ–ВНИИЭФ (Саров, Россия) и предназначен для решения широкого круга задач газовой динамики и теплообмена.

СПИСОК ЛИТЕРАТУРЫ

1. *Owens J.D., Luebke D., Govindaraju N., Harris M., Krűger J., Lefohn A.E., Purcell T.J.* A survey of general-purpose computation on graphics hardware // Computer Graphics Forum. 2007. **26**, N 1. 80–113.
2. *Боресков А.В., Харламов А.А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2010.
3. *Сандерс Дж., Кэндрот Э.* Технология CUDA в примерах: введение в программирование графических процессоров. М.: ДМК Пресс, 2011.
4. *Горобец А.В., Суков С.А., Железняков А.О., Богданов П.Б., Четверушкин Б.Н.* Применение GPU в рамках гибридного двухуровневого распараллеливания MPI+OpenMP на гетерогенных вычислительных системах // Параллельные вычислительные технологии. Челябинск: Издательский центр ЮУрГУ, 2011. 452–460.
5. *Corrigan A., Camelli F., Lőhner R., Wallin J.* Running unstructured grid-based CFD solvers on modern graphics hardware // AIAA Paper. 2009. N 2009-4001.
6. *Kampolis I.C., Trompoukis X.S., Asouti V.G., Giannakoglou K.C.* CFD-based analysis and two-level aerodynamic optimization on graphics processing units // Computer Methods in Applied Mechanics and Engineering. 2010. **199**, N 9–12. 712–722.
7. *Chorin A.J.* Numerical solution of Navier–Stokes equations // Mathematics of Computation. 1968. **22**, N 104. 745–762.
8. *Белоцерковский О.М.* Численное моделирование в механике сплошных сред. М.: Физматлит, 1994.
9. *Волков К.Н.* Реализация схемы расщепления на разнесенной сетке для расчета нестационарных течений вязкой несжимаемой жидкости // Вычислительные методы и программирование. 2005. **6**, № 1. 269–282.
10. *Thibault J.C., Senocak I.* CUDA implementation of a Navier–Stokes solver on multi-GPU desktop platforms for incompressible flows // AIAA Paper. 2009. N 2009-758.
11. *Волков К.Н., Емельянов В.Н.* Моделирование крупных вихрей в расчетах турбулентных течений. М.: Физматлит, 2008.

Поступила в редакцию
20.06.2012