УДК 519.688, 532.529 MSC 65Y05, 76E09, 76F06, 76F25

Моделирование задач гравитационного перемешивания на GPU

П. А. Кучугов[*], Н. Д. Шувалов[†], А. М. Казённов[†]

[*] *Институт прикладной математики им. М.В. Келдыша РАН*
Миусская пл., д. 4, Москва, Россия, 125047

[†] *Московский физико-технический институт*
Институтский пер., д. 9, Долгопрудный, Московская область, Россия, 141700

Гравитационное перемешивание, индуцированное неустойчивостью Рэлея–Тейлора, возникает при контакте разноплотных веществ, когда вектор ускорения, действующего на систему в целом, направлен из более плотного вещества в менее плотное. В этом случае амплитуда малых возмущений контактной границы растёт с течением времени, вовлекая в перемешивание всё новые и новые области течения (Rayleigh, Proc. of the London Math. Soc., 14, 1883; Taylor G.I., Proc. of the R. Soc. of London, A201, 1950). Для численного расчёта задач подобного рода требуется применение методов, способных полноценно описать разрывный характер гидродинамических величин. Наиболее часто используемым методом для расчёта разрывных течений является метод Годунова (Godunov S.K., Mat. Sb. (N.S.), 47(89), 3, 1959), который базируется на решении задачи о распаде разрыва для нахождения потоков на гранях счётных ячеек. В то же время известно, что точное решение задачи Римана является достаточно дорогостоящим с точки зрения вычислительных ресурсов. Однако при использовании массивно-параллельной архитектуры, такой как GPU, можно добиться значительного ускорения за счёт большого количества вычислительных процессов, что позволяет проводить расчёты в разы быстрее.

В рамках выполненной работы было реализовано два варианта параллельного алгоритма для расчёта перемешивания. Была проведена оценка их эффективности и ускорения.

Ключевые слова: GPU, гидродинамические неустойчивости, турбулентное перемешивание, математическое моделирование, CUDA.

1. Введение

На протяжении нескольких последних десятилетий задача перемешивания, вызванного развитием неустойчивости Рэлея–Тейлора, интенсивно изучается во многих ведущих мировых лабораториях в связи с исследованием возможности термоядерного синтеза (ИТС). На сегодняшний день уровень теоретических работ, как правило, определяется возможностью проведения детальных численных расчётов в связи с дороговизной и технической сложностью экспериментальных исследований. В этом смысле представляемая работа затрагивает одну из важных тем в численном моделировании, а именно реализацию параллельных алгоритмов для проведения прецизионных расчётов.

Также следует отметить, что задачи ИТС не являются единственным приложением, в котором важное место занимает гравитационной перемешивание. Оно играет далеко не последнюю роль при исследовании астрофизических, геологических, океанических и других явлений.

2. Численная методика

В качестве основной физической модели используется система уравнений невязкой нетеплопроводной газовой динамики, записанная в дивергентной форме и замыкаемая уравнением состояния идеального газа. Для аппроксимации данной системы уравнений используется предложенный в [1] разностный метод. Кратко

Статья поступила в редакцию 30 декабря 2013 г.

Работа выполнена при финансовой поддержке российского фонда фундаментальных исследований, грант № 12-01-31245-мол_а.

опишем его суть. Все физические поля вычисляются в серединах счётных ячеек. Для нахождения физических потоков через грани используются значения полей справа и слева от грани, которые получаются путём интерполяции значений в центре ячейки. Данная процедура сопровождается введением дополнительных функций, лимитеров (ограничителей антидиффузионных потоков), на вид которых накладываются условия [2]. Вид лимитера может варьироваться, так же как и способ вычисления потоков. В данной работе используется «острый» лимитер, а потоки находятся из решения задачи о распаде разрыва на грани ячейки. Такой метод достаточно хорошо себя зарекомендовал и широко используется при проведении прямого численного моделирования разрывных течений.

3. Реализация с использованием GPU

При использовании численной методики, описанной в разделе выше, для разрешения мелкомасштабных деталей течения или достаточно малых амплитуд возмущения требуется задавать сопоставимый шаг по пространству. При использовании физически осмысленных размеров счётной области получаем, что требуется большое число ячеек, а также дополнительно значительное число шагов по времени, если интересоваться достаточно поздними временами процесса развития неустойчивости Рэлея–Тейлора. В связи с этим актуальной становится задача реализации эффективного параллельного алгоритма для проведения моделирования. Такие алгоритмы для кластерных систем с использованием MPI (Message Passing Interface) и OpenMP были разработаны, например, в работах [3, 4]. Однако более перспективной представляется возможность запуска подобных вычислений на GPU устройствах и реализация гибридных программных комплексов [5]. Как известно, процесс распараллеливания задачи сводится к её декомпозиции на независимые составляющие, которые не требуют последовательного исполнения. Существует два основных варианта декомпозиции: функциональная декомпозиция, когда разными процессам исполняются различные подзадачи основной задачи, и декомпозиция данных, когда разными процессами исполняется одна и та же задача, но над различными наборами данных. Результатом описанной в предыдущем разделе численной методики является явная разностная схема, что, во-первых, позволяет нам реализовать саму возможность написания параллельной программы, а во-вторых, определяет основной метод декомпозиции, который будет использоваться – это декомпозиция данных. В случае использования массивно-параллельной архитектуры, такой как GPU-устройства, где количество вычислительных потоков достигает десятков и сотен тысяч, можно предполагать, что одна нить, отвечающая одному процессу, будет обрабатывать одну счётную ячейку области моделирования. Если количество ячеек велико и объём необходимой для проведения расчёта глобальной памяти не может быть выделен на одном GPU-устройстве, тогда вычисления должны проводиться на нескольких GPU-устройствах. Этот вариант является более сложным и сопровождается необходимостью обмена данными между устройствами.

В нашем случае уравнения гидродинамики записаны в декартовой системе координат, а область моделирования представляет собой параллелепипед. При декомпозиции данных удобно пользоваться такой же геометрией, т.е. при формировании блоков нитей и сетки блоков следует задавать их трёхмерными. Далее, поскольку в CUDA нет возможности провести барьерную синхронизацию всех блоков внутри одного ядра, одно вычислительное ядро разбивается на несколько, между которыми вызывается команда cudaThreadSynchronize. В данной работе было реализовано два варианта программы. В первом варианте вычисление потоков и новых значений гидродинамических величин в ячейках происходит в одном ядре. Это позволяет сэкономить количество используемой глобально памяти устройства, так как не требует хранения массивов потоков, однако приводит к необходимости использования атомарных операций, что, как известно, снижает эффективность параллельных вычислений. Во втором варианте вычисление

потоков и новых гидродинамических величин в ячейках разбито на два ядра, запускаемых последовательно. Результаты измерений ускорения для того и другого случая приведены в следующем разделе.

4. Результаты

В качестве тестовой задачи была выбрана классическая постановка о развитии неустойчивости Рэлея–Тейлора с параметрами: $g = 1$, $\rho_H = 3$, $\rho_L = 1$, $p_0(0) = 10$ (значения приведены в безразмерных единицах). На всех стенках были заданы условия отражения, в качестве начальных возмущений задавались случайные возмущения плотности в одном слое на контактной границе. Размеру счётной области $1 \times 1 \times 2$ соответствовала сетка $50 \times 50 \times 100$. Любое изменение расчётной сетки сопровождалось пропорциональным изменением размера области моделирования так, чтобы пространственные шаги всё время оставались неизменными. Снятие временных показателей проводилось на вычислительном гибридном кластере института прикладной математики им. М.В. Келдыша К-100 на одном узле при использовании одного процессора (Intel Xeon X5670) и одной видеокарты (nVidia Fermi C2050). Для получения более выверенных значений измерения снимались $T = 10$ раз, а затем проводилось их усреднение. В работе было проведено вычисление ускорения для различных конфигураций блоков для двух вариантов сеток. Данные приведены в табл. 1.

Таблица 1

Определение оптимальной формы вычислительного блока. Для параллельной версии программы измерения проводились на 300-х временных шагах, для последовательной – на 3-х. Времена в таблице представлены в секундах. BS_i**,** $i = x,\ y,\ z$ **– размер блока,** $TpB = BS_x \cdot BS_y \cdot BS_z$**,** S **— ускорение**

BS_x	BS_y	BS_z	TpB	$128 \times 128 \times 256$			$50 \times 50 \times 100$		
				T_{GPU}	T_{CPU}	S	T_{GPU}	T_{CPU}	S
8	8	8	512	83.7	53.57	64.0	5.22	3.195	61.2
8	8	16	1024	167.0		32.1	11.37		28.1
4	4	8	128	89.8		59.65	5.39		59.28
4	8	8	256	101.0		53.0	5.91		54.06
4	4	4	64	92.7		57.79	5.67		56.3
2	2	2	8	325.0		16.48	21.49		14.87
512	1	1	512	197.0		27.19	23.88		13.38

На основе данных, приведённых в таблице 1, можно сделать вывод, что наибольшее ускорение наблюдается при использовании размерности блока $8 \times 8 \times 8$. В этом случае за счёт выбранной геометрии блока и количества процессов в блоке достигается максимальная утилизация ресурсов GPU-устройства [6]. Здесь следует пояснить ситуацию, которая реализуется при использовании конфигурации с максимально возможным числом потоков в блоке (1024). Основное вычислительное ядро использует 63 регистра на поток (--ptxas-options=-v) для хранения локальных переменных, что в случае 1024 потоков превышает максимальное количество регистров на блок 32768. Это приводит к смещению этих переменных в глобальную память, что, в свою очередь, увеличивает время, необходимое для их чтения/записи, но также должно увеличить эффективность использования ресурсов видеокарты (occupancy). В нашем случае видно, что данный эффект является отрицательным и приводит к деградации производительности. Здесь можно найти компромисс между количеством регистров, перемещаемых в глобальную память, и эффективностью использования ресурсов видеокарты, но в данной работе мы не проводили данные исследования. Также можно использовать разделяемую память для хранения некоторых локальных переменных.

Интересным представляется случай $512 \times 1 \times 1$: при таком же количестве процессов в блоке, как и для $8 \times 8 \times 8$, получаемое ускорение ниже. Это связано с тем, что при обработке конкретной счётной ячейки также требуются данные из соседних ячеек, соответственно отношение количества вычислений к количеству обращений к памяти будет наиболее оптимальным для конфигурации $8 \times 8 \times 8$, обеспечивая более высокую степень коллективности чтения/записи.

Для конфигурации с максимальным ускорением ($8 \times 8 \times 8$) дополнительно была проведена развёртка по размерностям сетки — табл. 2.

Таблица 2

Зависимость ускорения CUDA реализации от конфигурации расчётной сетки при размере блока $8 \times 8 \times 8$. Время приведено в секундах

N_x	55	65	75	85	90	95	100	105	110
N_y	55	65	75	85	90	95	100	105	110
N_z	110	130	150	170	180	190	200	210	220
T_{CPU}	4.33	6.98	10.92	16.28	18.14	22.51	26.03	31.06	3.27
T_{GPU}	5.99	11.1	16.3	22.1	28.3	31.4	37.7	43.7	48.9
S	72.29	62.88	66.99	73.66	64.1	71.69	69.04	71.07	68.04

Из приведённых данных видно, что наблюдаются незначительные вариации ускорения в зависимости от количества ячеек. Эти колебания достаточно хорошо коррелируют с количеством простаивающих процессов в последнем блоке, который формируется для обработки ячеек, число которых не может составить полный блок: чем меньше «холостых» процессов, тем выше значение ускорения. Все измерения, представленные выше, были выполнены для первого варианта программы, в котором используются атомарные операции. Для второго варианта приведём здесь лишь результат: полученные значения ускорения в среднем (по всем конфигурациям) на 17% выше, чем в первом случае, что соответствует ожиданиям.

5. Выводы

В данной работе найдена оптимальная конфигурация вычислительного блока CUDA для моделирования 3D перемешивания в декартовых координатах. Данная конфигурация обеспечивает наилучшую утилизацию ресурсов видеокарты за счёт достаточно большого числа процессов в блоке и его геометрии. Также проведено исследование зависимости ускорения от размерности расчётной сетки. Было показано, что наблюдаемые вариации значений ускорения достаточно хорошо коррелируют с количеством «холостых» процессов в блоках.

Проведено сравнение двух параллельных реализаций программы на GPU. Как и ожидалось, при отказе от атомарных операций происходит повышение эффективности счёта, сопровождаемое увеличением объёма используемой глобальной памяти.

Литература

1. Разностные схемы трехмерной газовой динамики для задачи о развитии неустойчивости Рихтмайера-Мешкова / В. Ф. Тишкин, В. В. Никишин, И. В. Попов, А. П. Фаворский // Математическое моделирование. — 1995. — Т. 7, № 5. — С. 15–25. [Tishkin V. F., Nikishin V. V., Popov I. V., Favorski A. P. Finite Difference Schemes of Three-Dimensional Gas Dynamics for the Study of Richtmyer–Meshkov Instability // Matematic Modeling. — 1995. — Vol. 7, No 5. — Pp. 15–25. — (in russian).]

2. *Вязников К. В., Тишкин В. Ф., Фаворский А. П.* Построение монотонных разностных схем повышенного порядка аппроксимации для систем уравнений гиперболического типа // Математическое моделирование. — 1989. — Т. 1, № 5. — С. 95–120. [Vyaznikov K. V., Tishkin V. F., Favorski A. P. Construction of Monotone High Resolution Difference Schemes for Hyperbolic Systems // Matematic Modeling. — 1989. — Vol. 1, No 5. — Pp. 95–120. — (in russian).]
3. *Ладонкина М. Е.* Численное моделирование турбулентного перемешивания с использованием высокопроизводительных систем: Кандидатская диссертация: Кандидатская диссертация / Институт математического моделирования РАН. — 2005. [Ladonkina M. E. Numerical Simulation of Turbulent Mixing using High-Performance Systems: Ph.D. Thesis // Institute of Mathematical Modelling. — 2005. — (in russian).]
4. *Чеванин В. С.* Численное моделирование развития гидродинамических неустойчивостей на многопроцессорных системах // Математическое моделирование. — 2012. — Т. 24, № 2. — С. 17–32. [Chevanin V. S. Numerical Simulation of Hydrodynamics Instabilities Evolution on Multiprocessor Systems // Matematic Modeling. — 2012. — Vol. 24, No 2. — Pp. 17–32. — (in russian).]
5. Molecular Dynamics Simulations of the Relaxation Processes in the Condensed Matter on GPUs / I. V. Morozov, A. M. Kazennov, R. G. Bystryi et al. // Computer Physics Communications. — 2011. — Vol. 182, No 9. — Pp. 1974–1978.
6. *Алексеенко А. Е., Казённов А. М.* Реализация клеточных автоматов «игра “Жизнь”» с применением технологий CUDA и OpenCL // Компьютерные исследования и моделирование. — 2010. — Т. 2, № 3. — С. 323–326. [Alekseenko A. E., Kazennov A. M. CUDA and OpenCL Implementations of Conway's Game of Life Cellular Automata // Computer Research and Modeling. — 2010. — Vol. 2, No 3. — Pp. 323–326. — (in russian).]

UDC 519.688, 532.529 MSC 65Y05, 76E09, 76F06, 76F25

Simulation of the Gravitational Mixing on GPU

P. A. Kuchugov*, N. D. Shuvalov†, A. M. Kazennov†

* *Keldysh Institute of Applied Mathematics RAS*
4, Miusskaya sq., Moscow, Russia, 125047
† *Moscow Institute of Physics and Technology*
9, Institutskiy per., Dolgoprudny, Moscow region, Russia, 141700

Gravitational mixing induced by the Rayleigh-Taylor instability, arises in the system of two fluids/gases with different densities when the acceleration acts from a more dense material to the less dense one. In this case, the amplitude of small perturbations of the contact boundary increases over time, involving new flow regions into the mixing (Rayleigh, Proc. Of the London Math. Soc., 14, 1883; Taylor GI, Proc. Of the R. Soc. of London, A201, 1950). The numerical calculation of such problems requires the use of methods that can fully describe the discontinuous nature of the flow variables. One of such methods is the Godunov's one (Godunov SK, Mat. Sb. (NS), 47 (89), 3, 1959), which is widely used and based on solving the Riemann problem for further calculation of the fluxes through the edges of cells. At the same time, we know that the exact solution of the Riemann problem is quite expensive in terms of computing resources. However, when using massively parallel architectures such as GPU, significant acceleration can be achieved due to the large number of computational processes which allows to perform calculations much faster.

As part of the performed work two versions of a parallel algorithm were implemented for the calculation of mixing. The estimation of efficiency and speed up was made.

Key words and phrases: GPU, hydrodynamic instabilities, turbulent mixing, mathematical modeling, CUDA.