Численная реализация алгорита SIMPLE на компьютерах с параллельной архитектурой

Д.В. Деги, А.В. Старченко
Томский государственный университет

Рассматривается численное решение уравнений Навье–Стокса с помощью алгоритма SIMPLE на следующих многопроцессорных ЭВМ: системы с общей памятью (используется технология OpenMP), системы с распределенной памятью (MPI), графических процессорах (CUDA).

Введение

К настоящему времени создано большое количество вычислительных алгоритмов, ориентированных на последовательную модель программирования. Однако далеко не всегда удается разработать эффективную параллельную реализацию для многих из них. Поэтому актуальной в настоящее время становится проблема создания новых параллельных алгоритмов решения задач науки и техники на суперкомпьютерах, в частности для имеющих большое теоретическое и прикладное значение задач гидродинамики.

Физическая постановка

Рассматривается стационарное, изотермическое, ламинарное течение несжимаемой вязкой жидкости в прямоугольной полости (каверне) с верхней стенкой, движущейся в своей плоскости со скоростью U [1]. Жидкость, целиком заполняющая каверну, вовлекается в движение силами вязкости. Действие других сил пренебрегается. Значения плотности ρ и коэффициента вязкости μ постоянны. Кроме того, требуется выполнение условия прилипания частиц жидкости к твердой стенке.

Математическая постановка

Рассмотренный физический процесс можно описать с помощью следующих уравнений:

– уравнения движения в проекции на ось Ox:

$$\frac{\partial v_x v_x}{\partial x}+\frac{\partial v_y v_x}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\mu(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial^2 v_x}{\partial y^2});$$

– уравнения движения в проекции на ось Oy:

$$\frac{\partial v_x v_y}{\partial x}+\frac{\partial v_y v_y}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\mu(\frac{\partial^2 v_y}{\partial x^2}+\frac{\partial^2 v_y}{\partial y^2});$$

– уравнения неразрывности:

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}=0.$$

Здесь $\bar{v}=(v_x,v_y)$ – скорость движения жидкости, p – давление.

Граничные условия будут следующими:

– из условия непротекания следует:

$$(v_x)_{x=0}=(v_x)_{x=L_x}=(v_y)_{y=0}=(v_y)_{y=L_y}=0;$$

– из условия прилипания следует:

$$(v_y)_{x=0}=(v_y)_{x=L_x}=(v_x)_{y=0}=0,\ (v_x)_{y=L_y}=U,$$

где L_x и L_y – соответственно длина и ширина каверны; U – скорость движения верхней крышки.

Численный метод решения

Приближенное решение ищется на равномерной, шахматной сетке с помощью метода конечных объемов. Конвективные члены аппроксимируются с помощью схемы «против потока», диффузионные и производные от давления – центрально-разностной схемой. В итоге получается, что разностная схема, аппроксимирующая уравнение неразрывности, имеет вид

$$[(v_x)_{i+1\mathrm{j}}-(v_x)_{ij}]h_y+[(v_y)_{i\,j+1}-(v_y)_{ij}]h_x=0,\ i=\overline{0,Nx}\ ;\ j=\overline{0,Ny};$$

разностная схема для проекции уравнения движения по оси Ox:

$$aP_{i+1\,\mathrm{j}}(v_x)_{i+1\,\mathrm{j}}=aE_{i+1\,j}\ (v_x)_{i+2\,j}+aW_{i+1\,j}(v_x)_{ij}+aN_{i+1\,j}(v_x)_{i+1\,j+1}+$$

$$+aS_{i+1\,j}(v_x)_{i+1\,j-1}+d_{i+1\,j}(p_{ij}-p_{i+1\,j}),\ i=\overline{0,Nx-1},j=\overline{0,Ny};$$

разностная схема для проекции уравнения движения по оси Oy:

$$bP_{i\,j+1}(v_y)_{i\,j+1}=bE_{i\,j+1}(v_y)_{i+1\,j+1}+bW_{i\,j+1}(v_y)_{i-1\,j+1}+bN_{i\,j+1}(v_y)_{i\,j+2}+$$

$$+bS_{i\,j+1}(v_y)_{ij}+l_{i\,j+1}(p_{ij}-p_{i\,j+1}),\ i=\overline{0,Nx},j=\overline{0,Ny-1}\ .$$

Здесь h_x и h_y – шаги сетки по горизонтальной и вертикальной оси соответственно.

Коэффициенты $aP_{ij}, aE_{ij}, aW_{ij}, aN_{ij}, aS_{ij}, bP_{ij}, bE_{ij}, bW_{ij}, bN_{ij}, bS_{ij}$ – известные сеточные функции, зависящие от поля скорости, полученного на предыдущей итерации; $l_{i\,j+1} = h_x / \rho$, $d_{i+1\,j} = h_y / \rho$; Nx и Ny – размер сетки по горизонтали и вертикали соответственно.

Погрешность аппроксимации разностной схемы для уравнения неразрывности имеет второй порядок относительно каждого координатного направления, а для уравнений движения – первый. В силу нелинейности уравнений Навье–Стокса становится невозможным прямое решение системы сеточных уравнений и рассматриваемый процесс приобретает итерационный характер. Для совместного решения системы из трех систем разностных уравнений был применен алгоритм *SIMPLE* [2]. Основные вычислительные трудности данного алгоритма – это решение уравнений для скоростей, а также для поправки давления на каждой итерации, которые выполнялись с помощью метода нижней и верхней релаксации соответственно. В качестве критерия завершения итерационного процесса выбрано выполнение условий: норма поправки давления меньше либо равно заданного ε (0.001), а также выполнения дискретного аналога уравнения неразрывности (с точностью ε). Для организации эффективного параллельного расчета неизвестных методом релаксации был применен способ «*красно-черного*» упорядочивания узлов сетки [3].

Последовательный алгоритм решения задачи

В табл. 1 показано время работы последовательной программы в зависимости от размера используемой сетки. Последовательная версия программы ориентирована на скорость вычислений, так, в ней произведены следующие преобразования:

- организована последовательная выборка элементов из оперативной памяти («кэш-механизмы») [4];
- проведена минимизация количества арифметических операций, в том числе и операций, требующих больше тактов работы процессора (например, деления).

Таблица 1. Время работы программы при использовании различных сеток (точность – ε =0.001)

Сетка	Итераций	Время, с
128×128	1004	21
256×256	3176	294
512×512	12160	5037
1024×1024	53713	107655

Из таблицы видно, что при уменьшении каждого размера ячейки сетки в 2 раза число итераций увеличивается, возрастает объем вычислений, и время работы программы увеличивается более чем в 10 раз. Так, на сетке размером 1024×1024 элементов время расчета составляет около 30 ч.

Параллельный алгоритм решения задачи в среде OpenMP

В качестве системы с общей памятью использовался один вычислительный узел кластера **СКИФ Cyberia** [5], имеющий следующие характеристики: 8 Гб оперативной памяти, два двухъядерных процессора Intel Xeon 5150 Woodcrest с тактовой частотой 2,66 ГГц. В системе OpenMP выполнялось распараллеливание циклов. Исходная последовательная программа состоит в основном из следующих друг за другом циклов, тем самым упрощается сам процесс распараллеливания. Для трудоёмкого в плане вычислений цикла (занимающего на каждой итерации более 20% процессорного времени выполнения всей итерации) создается отдельная параллельная область с помощью конструкции #pragma omp parallel for [6], в которой цикл распараллеливается. Для оставшихся менее трудоемких циклов (коррекция скорости, давления, проверка критерия завершения итерационного процесса) создается единая параллельная область, и в ней уже идет распараллеливание циклов. Итерации цикла распределялись в статическом режиме. В табл. 2 показано получающееся ускорение.

Таблица 2. Ускорение OpenMP программы

Количество вычислительных ядер	Ускорение при использовании различных сеток			
	128×128	256×256	512×512	1024×1024
2	1.6	1.7	1.8	1.5
4	2.9	3.2	3.4	2.2

Из табл. 2 видно, что при увеличении количества ядер ускорение увеличивается. Лучшее ускорение получается на сетке 512×512.

Параллельный алгоритм решения задачи в среде CUDA

Расчеты проводились на видеокарте GTX 260, состоящей из 192 потоковых ядер с тактовой частотой 1,24 ГГц и объемом памяти 2 Гб. Распараллеливание рассматривалось по принципу геометрической декомпозиции. Использовалась двумерная декомпозиция данных на сеточной области. При организации вычислений на видеокарте важной является работа с глобальной памятью [7], также как и для современных процессоров являются важными «кэш-механизмы». Оптимизация работы с глобальной памятью включает в себя следующее: использование общей памяти, объединение запросов на запись и считывание из глобальной памяти и т.д. При построении параллельного алгоритма применялись следующие действия по оптимизации работы с памятью:

- для уменьшения повторных запросов из мультипроцессоров в глобальную память данные сначала копировались в их общую память, а уже затем использовались в вычислениях (с учетом конфликтов банков);
- с целью минимизации обменов данных между ОЗУ и глобальной памятью GPU все вычисления происходили на видеокарте. После завершения каждой итерации из GPU в ОЗУ копировалось всего два значения для проверки критерия завершения итерационного процесса центральным процессором;
- использование блоков, размер которых кратен числу 16, для объединения запросов в глобальную память.

В табл. 3 показано время работы и ускорение CUDA-программы по сравнению с последовательной версией программы.

Таблица 3. Время работы и ускорение CUDA программы

Сетка	Время, с	Ускорение
128×128	1.7	12.4
256×256	13	22.6
512×512	136	37
1024×1024	2240	48

Как видно из табл. 3, с увеличением плотности узлов сетки ускорение возрастает. На сетке 1024×1024 ускорение уже достигает 48 и время вычислений сокращается до ~ 37 мин.

Параллельный алгоритм решения задачи в среде MPI

Вычисления проводились на кластере СКИФ Cyberia. Библиотека MPI представляет собой достаточно большой набор функций [4], с помощью которых выполняются различные действия, такие как обмен значений между процессами, создание декартовой топологии, объявления производных типов данных. Так, для рассматриваемой задачи применялись:

- функции, совмещающие пересылки и прием передаваемых данных (MPI_Sendrecv);
- для удобства передачи и приема набора чисел объявлялись специальные типы (MPI_Type_Vector), с помощью которых передача или прием осуществлялись как с одним значением;
- распараллеливание рассматривалось по принципу геометрической декомпозиции (использовалась двумерная декомпозиция данных), для этого вся совокупность загружаемых процессоров разбивалась с помощью функции создания декартовой топологии (MPI_Cart_create).

В процессе вычислений каждое вычислительное ядро хранило в доступной оперативной памяти лишь только обрабатываемые данные, таким образом, это позволило запускать на кластере задачи с большим объемом данных, не ограничиваясь объемом ОЗУ одного узла.

В табл. 4 показано ускорение MPI-программы.

Таблица 4. Ускорение MPI-программы

Кол-во вычислительных ядер	Ускорение при использовании различных сеток			
	128×128	256×256	512×512	1024×1024
2	1.8	1.9	1.9	1.7
4	3.3	4	2.6	2.1
8	4.8	7.1	8.2	4.3
16	6.6	11.8	16.6	11.2
32	7.5	18.4	29	32.2
64	9.1	24.5	46.6	74

При увеличении количества вычислительных ядер ускорение на всех сетках увеличивается. На сетке размером 1024×1024 элементов при использовании 64 вычислительных ядер ускорение равно 74 (наблюдается эффект сверхлинейного ускорения), тем самым время работы последовательной программы сокращается до ~ 24 мин.

Заключение

Таким образом, в данной работе разработаны эффективные параллельные реализации алгоритма вычислительной гидродинамики SIMPLE для многопроцессорных ЭВМ с общей и распределенной памятью, а также с гибридной архитектурой, объединяющей использование CPU и GPU. Для рассмотренных размеров сетки производительность параллельной программы на видеокарте GTX 260 выше, чем производительность параллельной MPI-программы, работающей на 32 ядрах (16 процессоров Intel Xeon 5150) linux-кластера СКИФ Cyberia. С другой стороны, наименьшее по объему кода изменение по сравнению с последовательной программой оказалась в OpenMP-программе (добавлено всего 12 строк). MPI-программа в 1,6 раза больше текста последовательной программы (с частичным изменением текста), параллельная CUDA-программа – в 2,4 раза с полным преобразованием кода последовательной программы.

Литература

1. **Лойцянский Л.Г.** Механика жидкости и газа: Учеб.для вузов. – 6-е изд., перераб. и доп. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 840 с.
2. **Патанкар С.** Численные методы решения задач теплообмена и динамики жидкости. – М.: Энергоатомиздат, 1984. – 124 с.
3. **Ортега Дж.** Введение в параллельные и векторные методы решения линейных систем/ Под ред. Х.Д. Икрамова. – М.: Мир, 1991. – 367с.
4. **Беликов Д.А.** Высокопроизводительные вычисления на кластерах: Учеб. пособие/ Д.А. Беликов, И.В. Говязов и др.; под ред. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2008. – 198 с.
5. http://skif.tsu.ru
6. **Левин М.П.** Параллельное программирование с использованием системы OpenMP. – М.: БИНОМ, 2008. – 118 с.
7. **Боресков А.В.** Основы работы с технологией CUDA,/ А.В. Боресков, А.А. Харламов. – М.: ДМК Пресс, 2010. – 232 с.