УДК 004.94

Моделирование движения вязкой несжимаемой жидкости на платформе NVIDIA CUDA[1]

А.Л. Архипов, Ф.Н. Ясинский
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
Иваново, Российская Федерация
E-mail: AlArMail999@mail.ru

Авторское резюме

Состояние вопроса: Задача моделирования движения вязкой несжимаемой жидкости имеет различные сферы практического приложения в энергетике (например, моделирование течения питательной воды в контуре тепловой электростанции). Подобные задачи являются ресурсоемкими в вычислительном плане. Для их решения применяются дорогостоящие высокопроизводительные кластерные многопроцессорные системы. В связи с этим актуальной является задача моделирования движения вязкой несжимаемой жидкости без использования дорогостоящих вычислительных кластеров.
Материалы и методы: Задача моделирования движения вязкой несжимаемой жидкости решается с использованием вычислений на графических процессорах (GPU). В качестве средства для проведения вычислений выбрана платформа NVIDIA CUDA, которая позволяет выполнять на графических процессорах параллельные вычисления общего назначения. Использована адаптированная вычислительная схема алгоритма решения задачи моделирования движения вязкой несжимаемой жидкости.
Результаты: Предлагается вариант решения задачи моделирования движения вязкой несжимаемой жидкости с использованием вычислений на графических процессорах (GPU), что позволит уменьшить стоимость вычислительной системы, сохранив высокую скорость вычислений и необходимую точность. Получено ускорение вычислений по сравнению с вариантом работы на центральном процессоре в 85 раз.
Выводы: Перенос алгоритма вычисления движения вязкой несжимаемой жидкости на платформу NVIDIA CUDA позволил получить ускорение вычислений по сравнению с вариантом программы, выполняющим вычисления только на центральном процессоре. Использование вычислений на графических процессорах позволяет решать задачу моделирования движения вязкой несжимаемой жидкости без использования дорогостоящих вычислительных кластеров. Стоимость необходимого для этого оборудования получается в 80–100 раз ниже.

Ключевые слова: численное моделирование, противоточная производная, параллельное программирование, NVIDIA CUDA.

Movement Simulation of Viscous Incompressible Fluid on the NVIDIA CUDA

A.L. Arkhipov, F.N. Yasinskiy
Ivanovo State Power Engineering University, Ivanovo, Russian Federation
E-mail: AlArMail999@mail.ru

Abstract

Background: The movement simulation of viscous incompressible fluid has a great number of different areas of practical application in engineering (for example, simulation of feed water flow in heat power plant loop). The similar tasks are resource-intensive in calculation. The expensive highly productive cluster multiprocessor systems are used to solve the task. So, the problem of the movement simulation of viscous incompressible fluid without expensive calculating clusters is urgent.
Materials and methods: The authors suggest solving the problem of the movement simulation of viscous incompressible fluid using the calculations on graphics processors (GPU). The NVIDIA CUDA platform is chosen as a means to carry out the calculations. It allows to make general parallel calculations on graphics processors. Besides, the authors use the adapted computing circuit of the problem solving algorithm of the movement simulation of viscous incompressible fluid.
Results: The article contains the way for simulating the movement of viscous incompressible fluid with using of calculations on graphics processors (GPU) that helps to reduce the computing system costs, save the high speed of calculations and their accuracy. The authors demonstrate the calculation acceleration in 85 times according to the operation in the central processor
Conclusions: The authors carry out thecalculations of the movement of viscous incompressible fluid on NVIDIA CUDA platform and it helps to accelerate the calculations in comparison with the program which can calculate only in central processor. The application of graphics processors allows to simulate the movement of viscous incompressible fluid without expensive calculating clusters. The costs of necessary equipment are less in 80-100 times.

Key words: numerical simulation, countercurrent derivative, parallel programming, NVIDIA CUDA.

[1]Работа выполнена при поддержке Министерства образования и науки РФ ГКN№13 G25.31.0077

Задача моделирования движения вязкой несжимаемой жидкости имеет различные сферы практического приложения в энергетике (например, моделирование течения питательной воды в контуре тепловой электростанции). Так как данная задача требует для своего решения большого объема вычислений, ее обычно решают с использованием многопроцессорных систем, таких как вычислительные кластеры [1]. Предлагается вариант решения этой задачи без использования дорогостоящих многопроцессорных систем. В качестве вычислительного устройства предлагается использовать графический процессор.

Использование графических процессоров (GPU) не только для визуализации трехмерных моделей, но и для выполнения вычислений практикуется уже сравнительно давно. С 2007 года, когда мировой лидер в разработке графических процессоров компания NVIDIA выпустила на рынок вычислительную платформу CUDA, это направление получило мощный толчок к развитию.

Скорость работы алгоритма моделирования движения вязкой несжимаемой жидкости можно повысить при помощи платформы NVIDIA CUDA [2]. В качестве основы для решения этой задачи была взята реализация алгоритма моделирования движения вязкой несжимаемой жидкости, приведенная в [3].

В сосуде, имеющем форму прямоугольного параллелепипеда, просверлены три отверстия: из отверстий в северной и южной стенках сосуда жидкость затекает, а из отверстия в восточной стенке – вытекает. Таким образом, расчетная область представляет собой прямоугольник, то есть является двухмерной (рис. 1). Цель – построить поле скорости жидкости в горизонтальном срезе сосуда, используя вычисления на графическом процессоре.

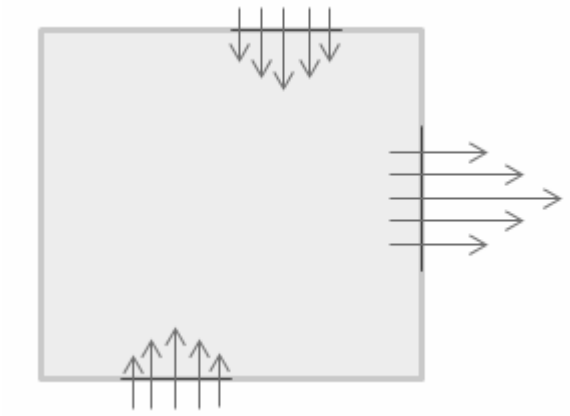


Рис. 1. Скорость по оси X (слева направо)

Расчеты ведутся с учетом того допущения, что моделируемая жидкость имеет слабую сжимаемость. Поэтому плотность жидкости ρ является не переменной, а константой. С учетом этого запишем дифференциальные уравнения:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right);$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right);$$

$$\frac{\partial p}{\partial t} = -c \cdot \rho \cdot divU;$$

$$divU = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y},$$

где u, v – скорость жидкости по осям x и y соответственно; p – давление; ρ – плотность; ν – вязкость; c – поправочный коэффициент.

Задача решается наложением на расчетную область сетки с размером ячейки $h \times h$. Вычислительная схема выглядит следующим образом:

$$p_{i,j}^{k+1} = p_{i,j}^{k} - \tau \cdot c \cdot \rho \cdot divU_{i,j}^{k},$$

$$divU_{i,j}^{k} = \frac{(u_{i+1,j+1} + u_{i+1,j-1}) - (u_{i-1,j-1} + u_{i-1,j+1})}{4h} +$$

$$+ \frac{(v_{i+1,j+1} + v_{i-1,j+1}) - (v_{i-1,j-1} + v_{i+1,j-1})}{4h};$$

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \tau \cdot du_{i,j}^{k},$$

$$du_{i,j} = -\left(u_1\frac{u_{i,j} - u_{i-1,j}}{h} + u_{-1}\frac{u_{i+1,j} - u_{i,j}}{h}\right) -$$

$$-\left(v_1\frac{u_{i,j} - u_{i,j-1}}{h} + v_{-1}\frac{u_{i,j+1} - u_{i,j}}{h}\right) -$$

$$-\frac{p_{i+1,j+1} + p_{i+1,j-1} - p_{i-1,j+1} - p_{i-1,j-1}}{4\rho h} +$$

$$+\nu\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2};$$

$$v_{i,j}^{k+1} = v_{i,j}^{k} + \tau \cdot dv_{i,j}^{k},$$

$$dv_{i,j} = -\left(u_1\frac{v_{i,j} - v_{i-1,j}}{h} + u_{-1}\frac{v_{i+1,j} - v_{i,j}}{h}\right) -$$

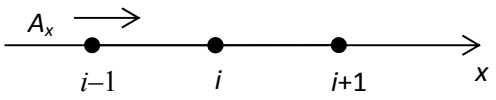
$$-\left(v_1\frac{v_{i,j} - v_{i,j-1}}{h} + v_{-1}\frac{v_{i,j+1} - v_{i,j}}{h}\right) -$$

$$-\frac{p_{i+1,j+1} + p_{i-1,j+1} - p_{i+1,j-1} - p_{i-1,j-1}}{4\rho h} +$$

$$+\nu\frac{v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j}}{h^2}.$$

Так как для вычисления состояния жидкости в текущей ячейке необходимы данные соседних ячеек, вычислить состояние жидкости в ячейках на границе расчетной области таким способом не представляется возможным. Поэтому для них состояние жидкости не вычисляется, а копируется из соседних ячеек в глубине расчетной области.

Для повышения вычислительной устойчивости используются противоточные производные. Для некоторой величины A противоточная производная имеет вид



$$\frac{\partial A}{\partial x} = \begin{cases} \dfrac{A_i - A_{i-1}}{h} & A_x \geq 0, \\ \dfrac{A_{i+1} - A_i}{h} & A_x < 0. \end{cases}$$

Противоточные производные удобно вычислять с использованием модуля величины:

$$u_1 = \frac{u_{i,j} + |u_{i,j}|}{2}, u_{-1} = \frac{u_{i,j} - |u_{i,j}|}{2},$$

$$v_1 = \frac{v_{i,j} + |v_{i,j}|}{2}, v_{-1} = \frac{v_{i,j} - |v_{i,j}|}{2},$$

Реализация. Сеточные алгоритмы позволяют использовать для своих вычислений массовый параллелизм. Поэтому их реализация с использованием вычислений на графическом процессоре (GPU) является, с нашей точки зрения, весьма эффективной.

На сегодняшний день наиболее развитой технологией выполнения вычислений общего назначения на GPU является платформа NVIDIA CUDA [2, 4].

Нами найдена такая реализация алгоритма, которая позволила получить ускорение вычислений от использования технологии NVIDIA CUDA: каждое ядро графического процессора обрабатывало одну ячейку расчетной области. Для вычисления состояния газа в ячейке необходимы следующие переменные: *u*, *v*, *p*. Только эти переменные передавались в функцию ядра (*kernel function*), которая и представляет собой код, выполняющийся на графическом процессоре. Плотность жидкости ρ являлась константой, которая задавалась непосредственно в коде программы. То же касается и остальных используемых констант.

Обнаружено, что цикл по времени нельзя организовать внутри функции ядра, так как каждое ядро графического процессора работает независимо от других и вычисления рассинхронизируются по времени. Поэтому был выбран следующий способ синхронизации вычислений: на графическом процессоре рассчитывался только один шаг по времени, после чего управление передавалось центральному процессору (CPU). После выполнения всего цикла по времени данные передавались на центральный процессор:

```
//Передача данных от CPU на GPU.
cudaMemcpy(du, u, numBytes, cudaMemcpyHostToDevice);

for (int t = 0; t <numberOfSteps; ++t) //nuberOfSteps – количествошаговповремени.
{
  //Выполнение функции ядра. du – массив размером numberOfCells, содержащий переменные u, v, p.
functionKernel<<<numBlocks, threadsPerBlock>>>(du);
  //СинхронизациявсехпотоковCUDA.
cudaDeviceSynchronize();
}
//Передача данных от GPU на CPU.
cudaMemcpy(u, du, numBytes, cudaMemcpyDeviceToHost);
```

Сама функция ядра выглядит следующим образом:

```
//u – массив размером numberOfCells, содержащий переменные u, v, p.
__global__ void functionKernel(floatP *u)
{
  //Каждое ядро GPU определяет координаты ячейки,
  //данные которой ему необходимо обработать.
intidx = blockIdx.x * blockDim.x + threadIdx.x;
intidy = blockIdx.y * blockDim.y + threadIdx.y;

//Проверка на принадлежность ячейки расчетной области.
  //Так как расчетная область может иметь размерности по осям X и Y не равные степени 2,
  //то некоторые ядра получат координаты ячеек, не попадающих в расчетную область.
if ((idx> 0) && (idx< DIM_POINTS_X - 1) && (idy> 0) && (idy< DIM_POINTS_Y - 1))
{
   //idt – номер ячейки в массиве u
intidt = idy * DIM_POINTS_X + idx;
//Так как для вычисления состояния жидкости в текущей ячейке необходимы данные
   //соседних ячеек, то на данном этапе необходимо получить эти данные.
   //Всего нужно 9 ячеек, включая текущую.
   //Для простоты реализации данные берутся непосредственно из глобальной памяти,
   //без кеширования в разделяемой памяти.
   //Впоследствии планируется использовать разделяемую память,
   //чтобы уменьшить задержки при работе с памятью,
   //и получить еще большее ускорение вычислений.
floatPu00 = u[idt];
floatPu10 = u[idt + 1];
floatPuM0 = u[idt - 1];
floatPu01 = u[idt + DIM_POINTS_X];
floatP u0M = u[idt - DIM_POINTS_X];
floatP u11 = u[(idy + 1) * DIM_POINTS_X + idx + 1];
floatPuMM = u[(idy - 1) * DIM_POINTS_X + idx - 1];
floatP uM1 = u[(idy + 1) * DIM_POINTS_X + idx - 1];
floatP u1M = u[(idy - 1) * DIM_POINTS_X + idx + 1];

   //Вычислениедавления.
   u00.z -= c * tau * ((u11.x + u1M.x) - (uMM.x + uM1.x) + (uM1.y + u11.y) - (uMM.y + u1M.y))/h/4;

//Обработка граничных условий для давления.
   //Копируем данные из соседних ячеек.
if (idy == 1)
u[idx].z = u00.z;
if (idy == DIM_POINTS_Y - 2)
u[idt + DIM_POINTS_X].z = u00.z;

if (idx == 1)
u[idt - 1].z = u00.z;
if (idx == DIM_POINTS_X - 2)
u[idt + 1].z = u00.z;

   //Обработка граничных условий для скорости.
   //Копируем данные из соседних ячеек.
if ((idy == DIM_POINTS_Y - 2) && (idx>= DIM_POINTS_X / 4 - 1) && (idx<= DIM_POINTS_X / 2))
u[(DIM_POINTS_Y - 1) * DIM_POINTS_X + idx].z = 2*u[(DIM_POINTS_Y - 2) * DIM_POINTS_X + idx].z - u[(DIM_POINTS_Y - 3) * DIM_POINTS_X + idx].z;
if ((idy == 1) && (idx>= DIM_POINTS_X / 2 - 1) && (idx<= 3 * DIM_POINTS_X / 4))
u[idx].z = 2*u[1 * DIM_POINTS_X + idx].z - u[2 * DIM_POINTS_X + idx].z;
if ((idx == DIM_POINTS_X - 2) && (idy>= DIM_POINTS_Y / 3 - 1) && (idy<= 2 * DIM_POINTS_Y / 3))
```

```
u[(idy + 1) * DIM_POINTS_X - 1].z = 2*u[(idy + 1) *
DIM_POINTS_X - 2].z - u[(idy + 1) * DIM_POINTS_X - 3].z;

//Вычисление новых значений скорости.
floatxx = u00.x;
floatyy = u00.y;
float x1 = (xx+fabsf(xx))/2;
floatxm = (xx-fabsf(xx))/2;
float y1 = (yy+fabsf(yy))/2;
floatym = (yy-fabsf(yy))/2;
   u00.x += tau*(-(x1 * (xx-uM0.x)/h + xm * (u10.x-xx)/h)-(y1 *
(xx-u0M.x)/h + ym * (u01.x-xx)/h)-(u11.z+u1M.z-uM1.z-
uMM.z)/h/ro/4 + nu*(u10.x+uM0.x+u01.x+u0M.x-4*xx)/h/h);
   u00.y += tau*(-(x1 * (yy-uM0.y)/h + xm * (u10.y-yy)/h)-(y1 *
(yy-u0M.y)/h + ym * (u01.y-yy)/h)-(u11.z+uM1.z-u1M.z-
uMM.z)/h/ro/4 + nu*(u10.y+uM0.y+u01.y+u0M.y-4*yy)/h/h);

//Запись вычисленных значений в массив u.
u[idt] = u00;
  }
}
```

Для уменьшения затрат памяти предлагается состояние газа (значение переменных *u*, *v*, *p*) в новый момент времени (момент времени *t*+1) записывать в тот же массив, откуда берутся значения в текущий момент времени (момент времени *t*). Такой подход вполне оправдан используемой схемой вычислений и не нарушает ее устойчивости.

Кроме того, исходный код программы был организован таким образом, что он мог быть скомпилирован для выполнения как на графическом процессоре (GPU), так и на центральном процессоре (CPU). Вычисления на центральном процессоре выполнялись в один поток в цикле по всем ячейкам сетки. Подобный подход позволил выполнить сравнение скорости работы этих двух вариантов программы и оценить полученное ускорение.

Выводы и результаты. Были проведены вычисления 100000 шагов по времени на сетке разной размерности. Шаг сетки составлял 0,1 м, шаг по времени – 0,0001 с. Время работы программы в секундах занесено в таблицу.

Размер сетки	GPU	CPU
768x768	45	4300
1024x1024	88	7100
2048x2048	316	26600

Описанная вычислительная схема в варианте для GPU обсчитывалась в 85 раз быстрее однопотокового варианта на CPU. Соответственно, многопроцессорный кластер должен иметь возможность обрабатывать одновременно 85 потоков для того, чтобы достичь такой же скорости вычислений. Такой компьютерный кластер стоит примерно в 80–100 раз больше обычного персонального компьютера с видеокартой NVIDIA GeForce. В результате расчетов получена картина поля скорости (рис. 2, 3):

Рис. 2. Скорость по оси X (слева направо)

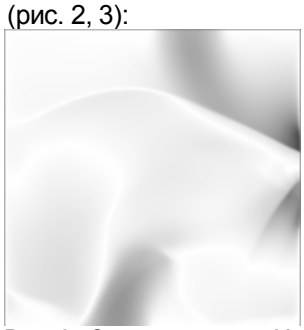


Рис. 3. Скорость по оси Y (сверху вниз)

Дальнейшее улучшение программы предполагает использование разделяемой памяти GPU (sharedmemory) с целью ускорить работу программы на этапе получения данных от соседних ячеек. Также предполагается включить в программы средства визуализации текущего состояния расчетной области.

Список литературы

1.**Гущин В.А., Матюшин П.В.** Математическое моделирование течений несжимаемой жидкости // Труды МФТИ. – 2009. – Т. 1. – № 4,. – С. 18–33.

2.**Википедия:** CUDA [Электронный ресурс]. Режим доступа: http://ru.wikipedia.org/wiki/CUDA

3.**Численные** методы и параллельные вычисления для задач механики жидкости, газа и плазмы / Э.Ф. Балаев, Н.В. Нуждин, В.В. Пекунов и др. – Иваново, 2003. – С. 83–87.

4.**NVIDIA** CUDA – неграфические вычисления на графических процессорах [Электронный ресурс]. – Режим доступа: http://www.ixbt.com/video3/cuda-1.shtml)

References

1.Gushchin V.A., Matyushin P.V. Matematicheskoe modelirovanie techeniy neszhimaemoy zhidkosti [Mathematical Simulation of Viscous Incompressible Fluid]. *TRUDY MFTI*, 2009, vol. 1, no. 4, pp. 18–33.

2.*CUDA*. Available at: http://en.wikipedia.org/wiki/CUDA

3. Balaev, E.F., Nuzhdin, N.V., Pekunov, V.V., Sidorov, S.G., Chernysheva, L.P., Yasinskiy, I.F., Yasinskiy, F.N. *Chislennye metody i parallel'nye vychisleniya dlya zadach mekhaniki zhidkosti, gaza i plazmy* [Numerical Analysis and Parallel Computing for Fluid and Plasma Mechanical Problems]. Ivanovo, 2003, pp. 83–87.

4.*NVIDIA CUDA – negraficheskie vychisleniya na graficheskikh protsessorakh* [NVIDIA CUDA – Nongraphical Calculations on graphics Processors]. Available at: http://www.ixbt.com/video3/cuda-1.shtml

Архипов Ален Леонидович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
программист ИВЦ, аспирант кафедры высокопроизводительных вычислительных систем,
e-mail: AlArMail999@mail.r