МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа № 7
по курсу «Программирование графических процессоров»

Message Passing Interface (MPI).

Выполнил: Ф.М. Шавандрин
Группа: 8О-408Б
Преподаватель: А.Ю. Морозов

Москва, 2022

Условие

Математическая постановка:

$$\frac{d^2u(x,y,z)}{dx^2} + \frac{d^2u(x,y,z)}{dy^2} + \frac{d^2u(x,y,z)}{dz^2} = 0,$$

$$u(x \le 0, y, z) = u_{left},$$
$$u(x \ge l_x, y, z) = u_{right},$$
$$u(x, y \le 0, z) = u_{front},$$
$$u(x, y \ge l_y, z) = u_{back},$$
$$u(x, y, z \le 0) = u_{down},$$
$$u(x, y, z \ge l_z) = u_{up}.$$

Входные данные: На первой строке заданы три числа: размер сетки процессов. Гарантируется, что при запуске программы количество процессов будет равно произведению этих трех чисел. На второй строке задается размер блока, который будет обрабатываться одним процессом: три числа. Далее задается путь к выходному файлу, в который необходимо записать конечный результат работы программы и точность ε. На последующих строках описывается задача: задаются размеры области l_x, l_y, l_z, граничные условия: $u_{down}, u_{up}, u_{\square} u_{\square}, u_{front}, u_{back}$ и начальное значениеu_0.

Выходные данные: В файл, определенный во входных данных, необходимо напечатать построчно значения $(u_{1,1,1}, u_{2,1,1}, \dots, u_{1,2,1}, u_{2,2,1}, \dots, u_{n_x-1,n_y,n_z}, u_{n_x,n_y,n_z})$ в ячейках сетки в формате с плавающей запятой с семью знаками мантиссы.

Цель работы: Знакомство с технологией MPI. Реализация метода Якоби. Решение задачи Дирихле для уравнения Лапласа в трехмерной области с граничными условиями первого рода.

Вариант 5. Обмен граничными слоями через send/receive, контроль сходимости allreduce.

Программное и аппаратное обеспечение

GPU:

- Название NVIDIA GeForce GT 545
- Compute capability: 2.1
- Графическая память: 3150381056
- Разделяемая память: 49152
- Константная память: 32768
- Количество регистров на блок: 32
- Максимальное количество нитей: (1024, 1024, 64)
- Максимальное количество блоков: (65535, 65535, 65535)
- Количество мультипроцессоров: 3

Сведения о системе:

- Процессор: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
- ОЗУ: 15 ГБ
- HDD 500 ГБ

Программное обеспечение:

- OS: Ubuntu 16.04.6 LTS
- Текстовый редактор: Vim
- Компилятор: nvcc

Метод решения

Сначала строим трехмерную сетку. Затем с каждой ячейкой сопоставляется значение функции *u* в точке соответствующей центру ячейки. Граничные условия реализуются через виртуальные ячейки, которые окружают рассматриваемую область. Каждый из процессов обрабатывает свой кусок сетки. Поиск решения сводится к итерационному процессу:

$$u_{i,j,k}^{(n+1)} = \frac{\left(u_{i+1,j,k}^{(n)}+u_{i-1,j,k}^{(n)}\right)h_x^{-2}+\left(u_{i,j+1,k}^{(n)}+u_{i,j-1,k}^{(n)}\right)h_y^{-2}+\left(u_{i,j,k+1}^{(n)}+u_{i,j,k-1}^{(n)}\right)h_z^{-2}}{2\left(h_x^{-2}+h_y^{-2}+h_z^{-2}\right)},$$

где

$$i = 1..n_x, j = 1..n_y, k = 1..n_z,$$
$$h_x = l_x n_x^{-1}, h_y = l_y n_y^{-1}, h_z = l_z n_z^{-1},$$
$$u_{0,j,k}^{(n)} = u_{left}, u_{n_x+1,j,k}^{(n)} = u_{right},$$
$$u_{i,0,k}^{(n)} = u_{front}, u_{i,n_y+1,k}^{(n)} = u_{back},$$
$$u_{i,j,0}^{(n)} = u_{down}, u_{i,j,n_z+1}^{(n)} = u_{up},$$
$$u_{i,j,k}^{(0)} = u^0.$$

Процесс останавливается, как только значение функции *u* на *(n+1)*-ой итерации станет меньше значение функции *u* на *n*-ой итерации на заданную ε.

Описание программы

Параллельно будем производить вычисления на сетке - у каждого из процессов будет свой кусок сетки. Одна итерация решения исходной задачи состоит из трех этапов:

1. Обмен граничными слоями между процессами. Будем передавать новую информацию о значениях на своих границах с помощью send/receive,
2. Обновление значений во всех ячейках по вышеописанной формуле.
3. Вычисление погрешности: сначала локально в рамках каждого процесса, а потом через обмены (с помощью allreduce) и во всей области, и сравнение её с ε.

Итоговый ответ отправляется главному процессу с помощью send/receive.

Результаты

Для тестирования программы будем замерять время её работы с разным количеством процессов. Размер сетки для каждого теста — 32 x 32. Будем учитывать только время выполнения программы (без учёта считывания и печати данных).

Число процессов	Время работы, мс
1	21.13
2	11.54
4	8.32
8	10.52
16	19.21

По результатам замера времени работы программы можно сделать вывод, что использование большего количества процессов, чем количество ядер процессора нецелесообразно. Оптимальнее всего в данном случае будет использовать 4 процесса.

Выводы

В данной лабораторной работе познакомился с технологией MPI, реализовал метод Якоби для решения задачи Дирихле (уравнения Лапласа в трехмерной области с граничными условиями первого рода). Технология MPI может быть полезна при решении задач, в которых необходимо параллельно произвести сложные вычисления, так как возможность создания нескольких параллельно работающих процессов при помощи данной технологии значительно ускорит время работы программы.