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:

# **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 , граничные условия: и начальное значение.

# **Выходные данные:** В файл, определенный во входных данных, необходимо напечатать построчно значения () 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
* Сompute 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:

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

# **Описание программ****ы**

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

# Обмен граничными слоями между процессами. Будем передавать новую информацию о значениях на своих границах с помощью send/receive,

# Обновление значений во всех ячейках по вышеописанной формуле.

# 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.