МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №1
по курсу «Параллельная обработка данных»

Message Passing Interface (MPI)

Выполнил: А. О. Тояков
Группа: М8О-407Б-18
Преподаватели: К. Г. Крашенинников,
А. Ю. Морозов

Москва, 2022

УСЛОВИЕ

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

Вариант № 7: Обмен граничными слоями через sendrecv, контроль сходимости allreduce;

ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ

Device: GeForce MX250

Размер глобальной памяти: 3150381056

Размер константной памяти : 65536

Размер разделяемой памяти: 49152

Регистров на блок: 32768

Максимум потоков на блок: 1024

Количество мультипроцессоров : 3

OS: Linux Ubuntu 18.04

Редактор: VSCode

Компилятор: nvcc версии 11.4 (g++ версии 7.5.0)

МЕТОД РЕШЕНИЯ

Сетка делится на области, за каждую из которых отвечает один процесс. Алгоритм состоит из трёх этапов. На первом процессы обмениваются данными в граничных ячейках сетки. На втором происходит вычисление значений во всех ячейках. На третьем этапе вычисляется погрешность и сравнение её с заданной точностью eps. Если достигли сходимости – выход из итерационного процесса и вывод результата в файл.

ОПИСАНИЕ ПРОГРАММЫ

#define _i(i, j, k) – переход из трёхмерной сетки в одномерную для элементов.

#define _ib(i, j, k) – переход из трёхмерной сетки в одномерную для блоков.

MPI_Bcast() – передача данных всем процессам.

MPI_Sendrecv – передача данных между процессами.

MPI_Allreduce – контроль сходимости всех процессов.

class Processor – класс для хранения всех функций и переменных для решения задачи.

int main() – создание экземпляра класса Processor, а также инициализация и завершение работы MPI.

ТЕСТЫ ПРОИЗВОДИТЕЛЬНОСТИ

блоков\процессов	1*1*1	2*2*1	2*2*2
4*4*4	0.29ms	3.2ms	8.7ms
10*10*10	16.8ms	86.5ms	148.9ms
20*20*20	459ms	2115.1ms	3394.5ms
30*30*30	3566.2ms	15230.8ms	27.7s

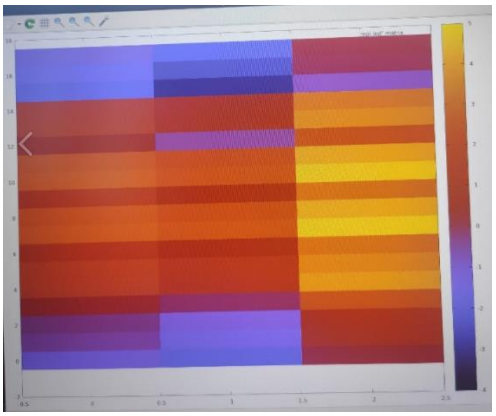
ВИЗУАЛИЗАЦИЯ

Тест:

test.txt

```
1 1 1
3 3 6
mpi.out
1e-10
1.0 1.0 2.0
-7.0 -10.0 5.0 10.0 -3.0 0.0
7.0
```

Срез:



ВЫВОДЫ

Выполнив лабораторную работу № 7, я познакомился с технологией MPI, которая является довольно удобным средством для выполнения параллельных вычислений на нескольких процессорах. Сложности возникают во время написания кода, т. к. он получается очень громоздким. Также необходимо следить, чтобы процессы корректно обменивались информацией, что в случае с трёхмерной сеткой довольно трудно выполнить.