МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

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

Лабораторная работа №3
по курсу «Параллельная обработка данных»

Технология MPI и технология OpenMP

Выполнил: Г.Н. Хренов
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

1. Цель работы: Совместное использование технологии MPI и технологии OpenMP. Реализация метода Якоби. Решение задачи Дирихле для уравнения Лапласа в трехмерной области с граничными условиями первого рода.
2. Вариант 2. Распараллеливание в общем виде с разделением работы между нитями вручную (“в стиле CUDA”).

Программное и аппаратное обеспечение

GPU name: NVIDIA GeForce RTX 2060
compute capability 7:5
totalGlobalMem: 6442450944
sharedMemPerBlock: 49152
totalConstMem: 65536
regsPerBlock: 65536
maxThreadsDim: 1024 1024 64
maxGridSize: 2147483647 65535 65535
multiProcessorCount: 30
CPU name: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx
MaxClockSpeed: 2300
NumberOfCourse: 4
RAM: 8
SSD: 256, HDD: 1024
OS: Windows10
Compiler: nvcc

Метод решения

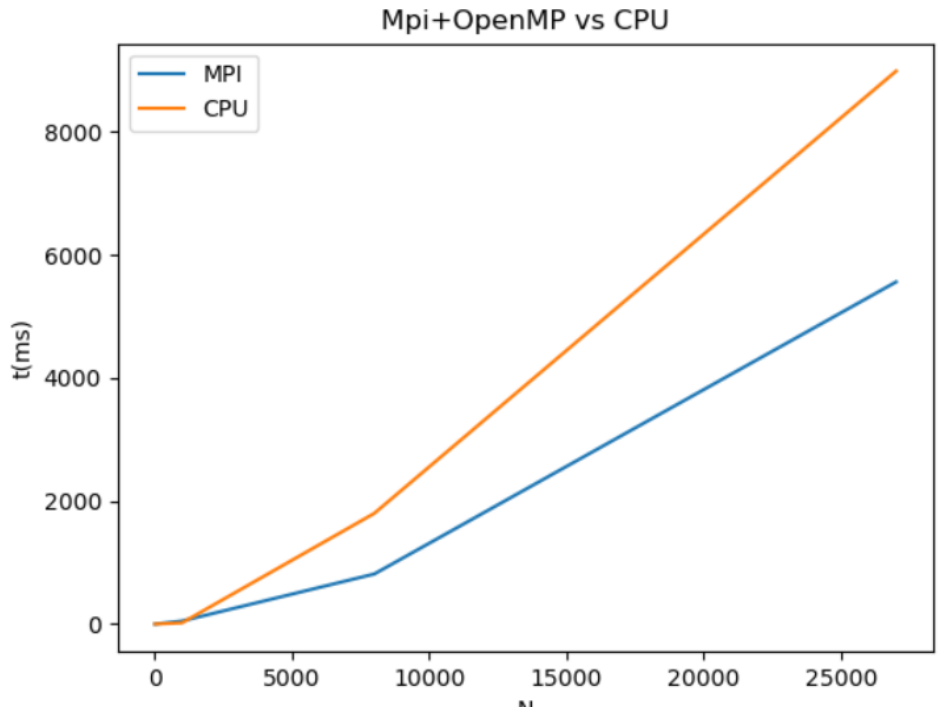
Задача решается аналогично предыдущей лабораторной. Для передачи данных между процессами вместо дополнительного буфера создаем производные типы данных, которые определяют расположения границ, которые нужно передать, на сетке процесса. Распараллеливание основного цикла делаем с помощью OpenMP.

Описание программы

Lab9.cpp:
#define _i(i, j, k) – переход из трехмерной сетки в линейную для элементов
#define _ib(i, j, k) - переход из трехмерной сетки в линейную для блоков
MPI_Type_create_subarray(3, sizes, subsizes_lr, starts_lfd, MPI_ORDER_C, MPI_DOUBLE, &left_bnd_send) – создание производного типа данных
MPI_Type_commit(&left_bnd_send) – регистрация производного типа данных
MPI_Bsend(data, 1, right_bnd_send, _ib(ib + 1, jb, kb), id, MPI_COMM_WORLD)- отправка сообщения, при этом указывается производный тип, аналогично с приемом
MPI_Type_free(&left_bnd_send) – освобождение производного типа, завершение работы с ним

Результаты(сервер)

блоков\процессов	1*1*1	2*2*1	2*2*2
4*4*4	0.85ms	3200.4ms	4105.61ms
10*10*10	50.3ms	18116.6ms	23992ms
20*20*20	814.7ms	51562ms	70187.3ms
30*30*30	5564.2ms	128801ms	168309ms



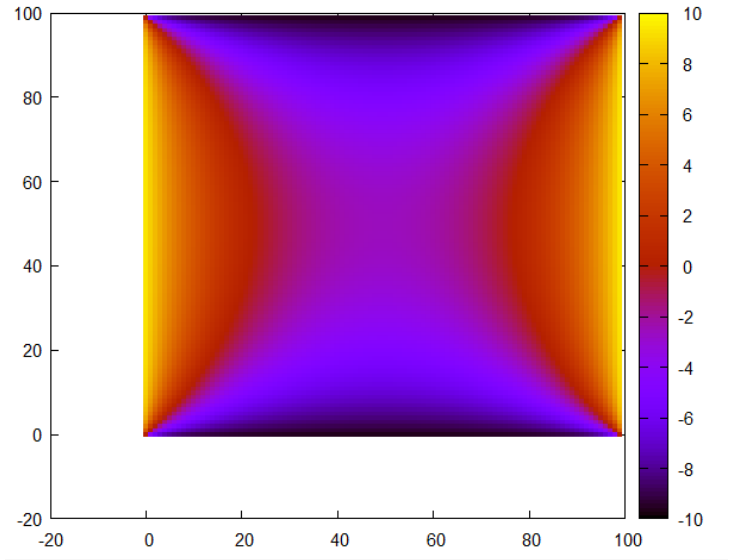
Визуализация

```
2 2 2
100 100 5
mpi.out
1e-10
1.0 1.0 1.0
-5 5 10.0 10.0 -10.0 -10.0
```

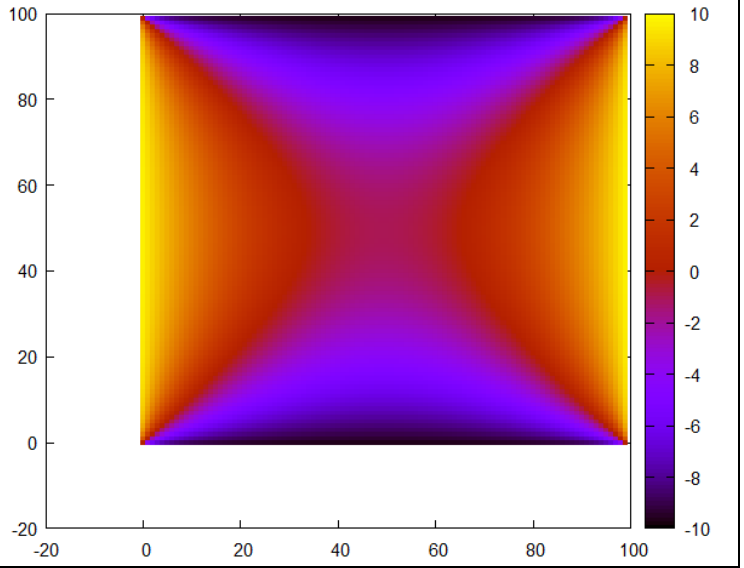
Внизу-холодно, вверху-жарко,
лево-право-жарко
зад-перед-холодно

Срезы по высоте(k)

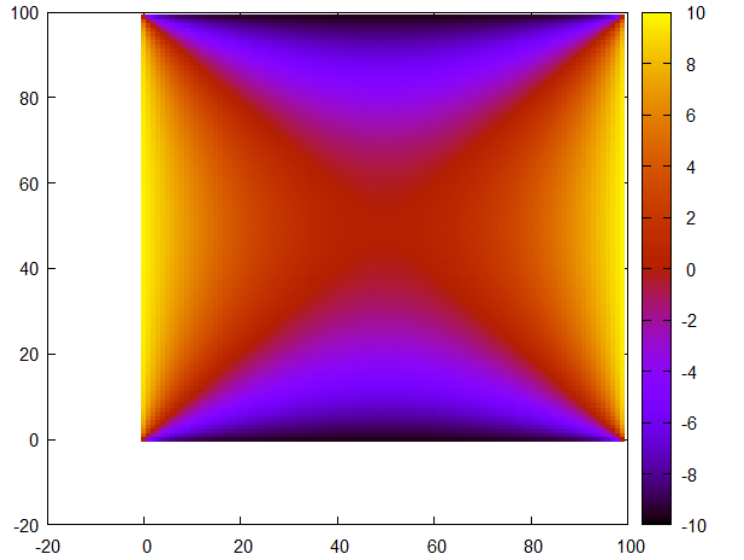
K=1


100
80
60
40
20
0
-20
-20
0
20
40
60
80
100
10
8
6
4
2
0
-2
-4
-6
-8
-10

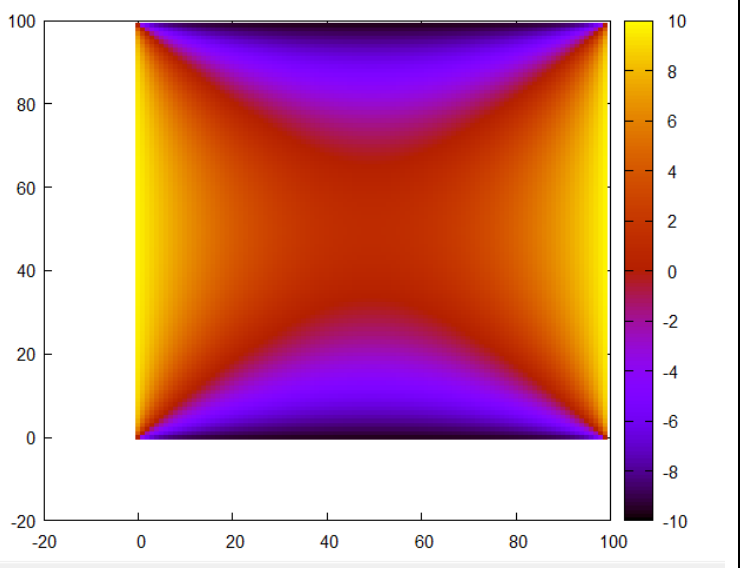
K=2


100
80
60
40
20
0
-20
-20
0
20
40
60
80
100
10
8
6
4
2
0
-2
-4
-6
-8
-10

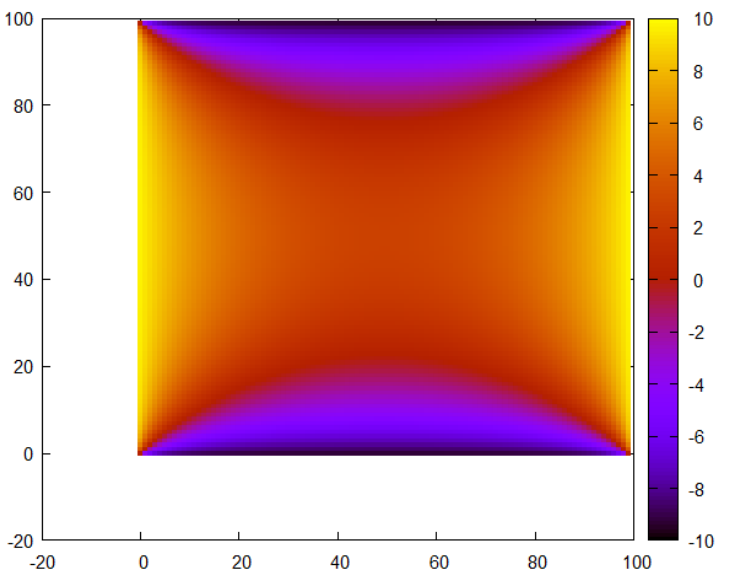
K = 3


100
80
60
40
20
0
-20
-20
0
20
40
60
80
100
10
8
6
4
2
0
-2
-4
-6
-8
-10

K = 4


100
80
60
40
20
0
-20
-20
0
20
40
60
80
100
10
8
6
4
2
0
-2
-4
-6
-8
-10

K = 5


100
80
60
40
20
0
-20
-20
0
20
40
60
80
100
10
8
6
4
2
0
-2
-4
-6
-8
-10

Выводы

Самое главное преимущество OpenMP – простота в использовании. Добавление пары строк кода достаточно, чтобы вычисления производились параллельно, при этом сохраняется функционал и нет необходимости писать отдельные ядра, как в случае с CUDA. Производные типы данных – удобный и мощный инструмент в MPI. Кроме записи в файл всеми процессами они позволяют не использовать дополнительную память процессам при отправке и получении сообщений.