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: Д.В. Коростелев
Группа: 8О-408Б
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 1:

обмен граничными слоями через send/receive, контроль сходимости allgather;

Программное и аппаратное обеспечение

- Графический процессор NVIDA GeForce GTX 1050

Графическая память	2 Gb
Разделяемая память на блок	48 Mb
Константная память	64 Mb
Количество регистров на блок	65536
Максимальное количество блоков на процессор	32
Максимальное количество потоков на блок	1024
Количество мультипроцессоров	5

- Процессор Intel Core i7-7700HQ 4x 2.808ГГц

Количество ядер	4
Количество потоков	8
Базовая тактовая частота	2.8 GHz
Максимальная тактовая частота	3.8 GHz
Кеш-память	6 Mb

- Оперативная память DDR4-SODIMM

Объем памяти	8 Gb
Частота	2400 MHz
Форм-фактор	SODIMM
Количество плашек	2

- SSD и HDD накопители

Объем SSD накопителя	128 Gb
Объем HDD накопителя	1 Tb

- Программное обеспечение

Операционная система	Windows 10 Pro
Средство разработки на CUDA (IDE)	Microsoft Visual Studio 2019
Компиляторы	MSVC 2019
Версия CUDA Toolkit	11.4.2
Дополнительный текстовый редактор	Notepad++

Метод решения

Требуется реализовать математическое решение уравнения:

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

Решение имеет следующий вид:

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

Решение будет получено, только если разница между значениями сетки на текущем и предыдущим шагом будет меньше эпсилон. На каждом шаге, согласно решению уравнения, будем считать новые значения u, находится наибольшую разницу между текущими и прошлыми ячейками и затем решать – продолжать ли вычисления или мы уже нашли решение

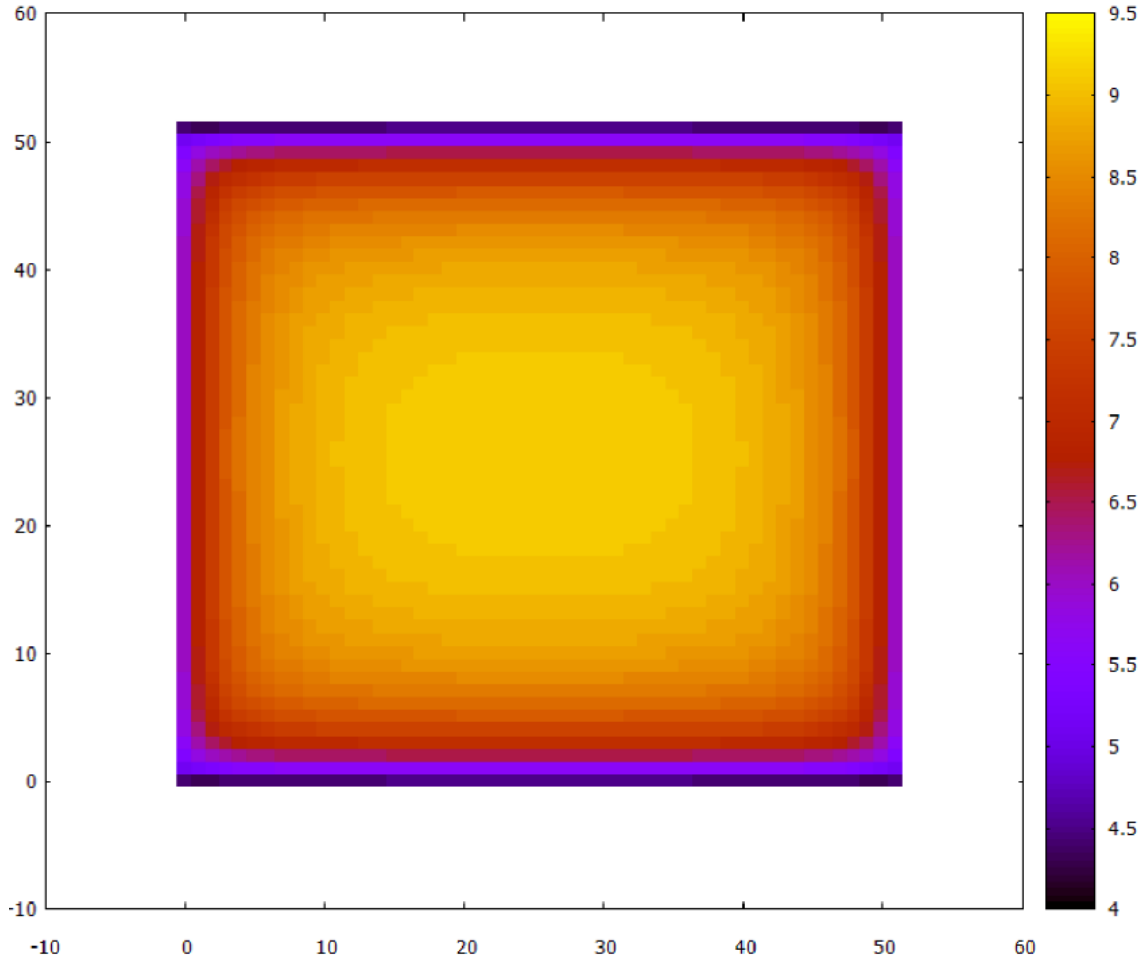
Описание программы

Осуществляем ввод всех данных в нулевом блоке, необходимых для решения задачи, затем при помощи Bcast отправляем данные с нулевого блока на все другие блоки. Аллоцируем память в каждом процессе под размерность блока внутри процесса для вычислений новых значений и значений, полученных на предыдущем шаге. Процесс вычисления итеративный – прекращается когда максимальная разница между ячейками текущего и прошлого блока будет меньше эпсилон, для этого в начале каждой итерации будем совершать обмен граничными значениями при помощи MPI_send/MPI_recv, синхронизировать процессы, затем вычислять новые значения, синхронизировать процессы и потом будут собирать максимальные разности со всех блоков при помощи MPI_Allgather.

Результаты

Сетка / Задача	1x1x1	2x2x2	3x3x3	4x4x4
32 x 32 x 32	4.756	2.059	3.555	3.781
42 x 42 x 42	17.22	8.451	12.34	11.01
54 x 54 x 54	64.12	31.56	23.56	25.25
64 x 64 x 64	230.1	54.23	40.34	25.36

Визуализация решения через gnuplot



Выводы

Технология MPI оказалась довольно удобной для организации межпроцессорного взаимодействия (механизм показался похожим на ZMQ) и погружение в библиотеку было очень быстрым и простым. Наиболее сложным оказалось реализовать межпроцессорное взаимодействие и передачу значений с 6 граней другим блокам (пришлось тянуть похожие друг на друга части кода).