МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

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

Лабораторная работа №1
по курсу «Параллельная обработка данных»

Message Passing Interface.

Выполнил: В. В. Бирюков
Группа: 8О-407Б
Преподаватель: 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: 8. Обмен граничными слоями через isend/irecv, контроль сходимости allreduce.

Программное и аппаратное обеспечение

Характеристики системы:

- Процессор: AMD Ryzen 7 3700U 2.3GHz
- Память: 20 ГБ
- SSD: 500 ГБ

Программное обеспечение:

- ОС: Windows 10
- IDE: Sublime Text 4
- Компилятор: mpic++: g++ 12.2.0

Метод решения

Необходимо решить эллиптическое уравнение в частных производных — уравнение Лапласа — с граничными условиями первого рода, методом простых итераций. Целевая функция задана в трехмерном области, имеющей вид параллелепипеда. Критерий остановки итераций — максимальная разница между текущим и следующим значением меньше некоторого значение.

Описание программы

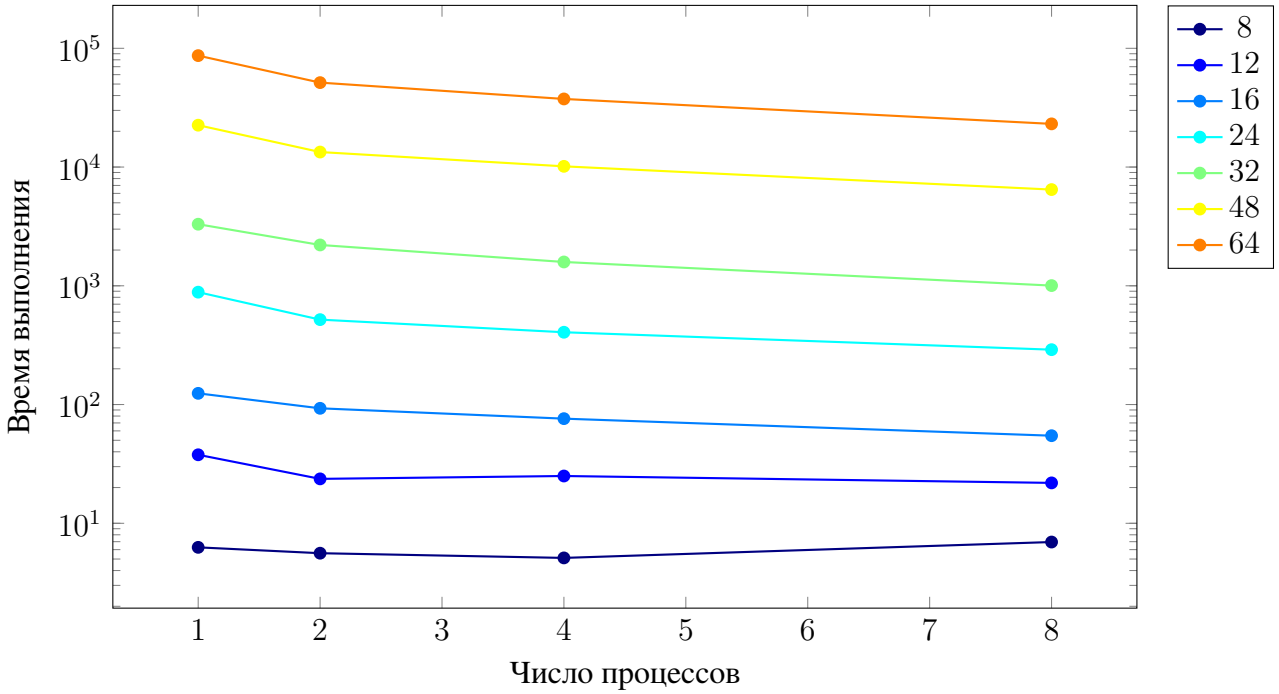
Дискретная сетка разбивается на блоки согласно входным данным. В каждом блоке итерации выполняются отдельным процессом. Каждая итерация в блоке происходит независимо, однако в конце нее процессы должны обменяться граничными слоями. Для передачи данных между процессами используются неблокирующие операции Isend/Irecv, поэтому каждый процесс записывает 6 граничных слоев в 6 различных буфер и осуществляет их отправку соседним процессам, затем получает граничные слои в другие 6 буферов и переписывает их в свой блок данных. Разница между итерациями вычисляется в каждом блоке в ходе обновления значений, затем находится глобальный максимум при помощи функции Allreduce. После окончания итераций данные со всех процессов агрегируются нулевым, передача осуществляется при помощи блокирующих операций Send/Recv.

Результаты

Сравнение времени работы Для замера времени работы программы зафиксируем граничные условия и значение погрешности (10^{-8}) и будем изменять разрешение глобальной сетки (сетка имеет вид N×N×N) и количество используемых процессов.

N	8	12	16	24	32	48	64
Число процессов	**Время выполнения, мс**						
1	6.2718	37.7551	124.263	883.817	3300.87	22517.9	86774.1
2	5.5988	23.6612	92.950	518.918	2207.44	13376.3	51410.7
4	5.1104	25.0131	76.076	406.282	1587.22	10138.9	37458.3
8	6.9534	21.8882	54.659	289.816	1004.07	6461.2	23100.3
16	11169	22666	37887	-	-	-	-
	Ускорение						
2	1.12	1.60	1.34	1.70	1.50	1.68	1.69
4	1.23	1.51	1.63	2.18	2.08	2.22	2.32
8	0.90	1.72	2.27	3.05	3.29	3.49	3.76
	Коэффициент распараллеливания						
2	0.56	0.80	0.67	0.85	0.75	0.84	0.84
4	0.30	0.38	0.41	0.54	0.52	0.56	0.58
8	0.11	0.22	0.28	0.38	0.41	0.44	0.47

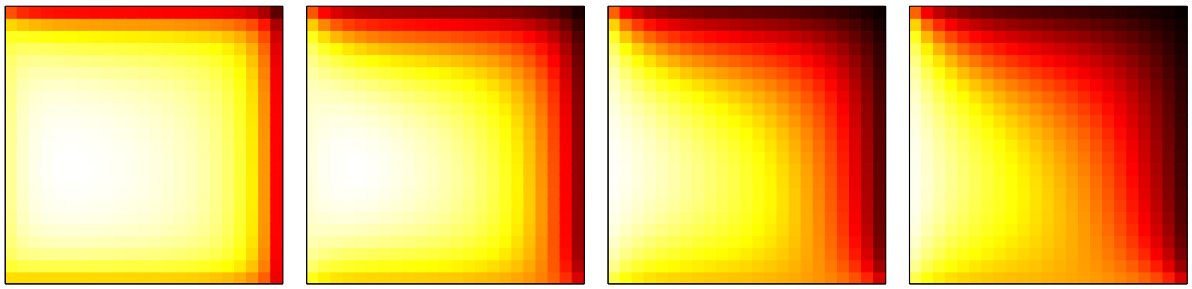
Измерения выполнялись на процессоре с 8 логическими ядрами, поэтому использование большего числа процессов приводит к абсолютно ужасным результатам.



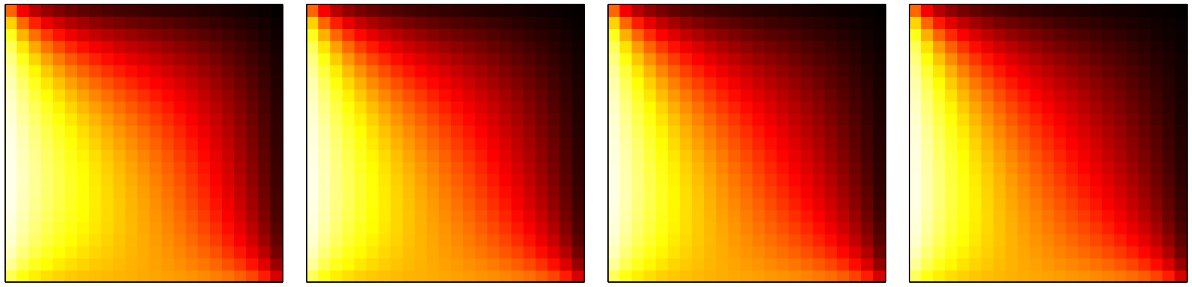
Результаты работы

Рассмотрим каждый второй срез по z для решения с глобальной сеткой $24 \times 24 \times 31$ для уравнения с граничными условиями $u_{down} = 7$, $u_{up} = 0$, $u_{left} = 5$, $u_{right} = 0$, $u_{front} = 3$, $u_{back} = 0$ и начальным значением $u = 5$.

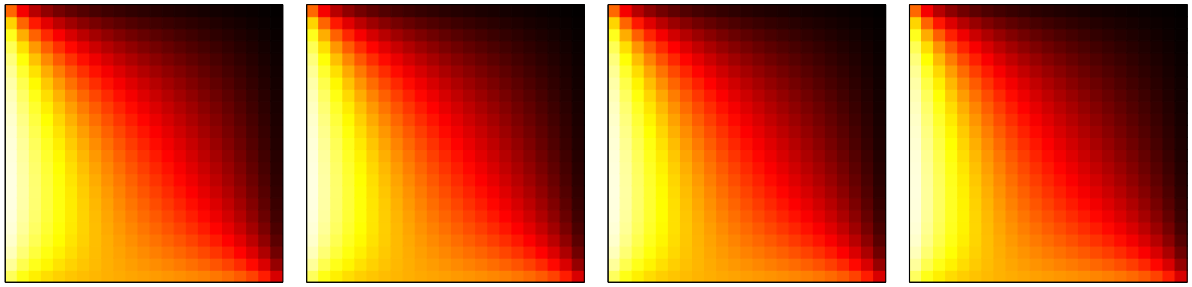
$z = 0$ $z = 2$ $z = 4$ $z = 6$



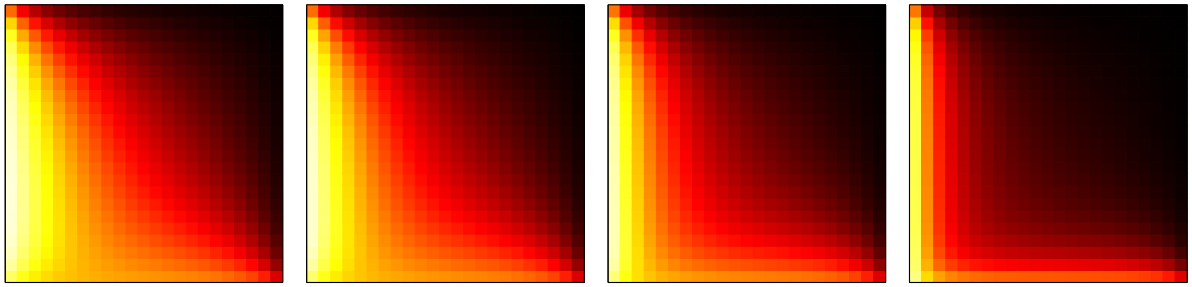
$z = 8$ $z = 10$ $z = 12$ $z = 14$



$z = 16$ $z = 18$ $z = 20$ $z = 22$



$z = 24$ $z = 26$ $z = 28$ $z = 30$



Выводы

В ходе выполнения лабораторной работы я познакомился с технологией MPI. Подход при распараллеливании задачи на заведомо небольшое количество процессов очень сильно отличается от распараллеливания на GPU, однако лучше масштабируется, позволяя проводить расчеты на нескольких вычислительных устройствах.

Что касается конкретно этой задачи и запусков в рамках одного процессора, то больше всего ускорения происходит при большом объеме данных и использовании максимального количества параллельных процессов.